UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Physics and Technology

Improving Representation Learning for Deep Clustering and Few-shot Learning

Daniel Johansen Trosten

A dissertation for the degree of Philosophiae Doctor May 2023

Abstract

The amounts of data in the world have increased dramatically in recent years, and it is quickly becoming infeasible for humans to label all these data. It is therefore crucial that modern machine learning systems can operate with few or no labels. The introduction of deep learning and deep neural networks has led to impressive advancements in several areas of machine learning. These advancements are largely due to the unprecedented ability of deep neural networks to learn powerful representations from a wide range of complex input signals. This ability is especially important when labeled data is limited, as the absence of a strong supervisory signal forces models to rely more on intrinsic properties of the data and its representations.

This thesis focuses on two key concepts in deep learning with few or no labels. First, we aim to improve representation quality in deep clustering – both for single-view and multi-view data. Current models for deep clustering face challenges related to properly representing semantic similarities, which is crucial for the models to discover meaningful clusterings. This is especially challenging with multi-view data, since the information required for successful clustering might be scattered across many views. Second, we focus on few-shot learning, and how geometrical properties of representations influence few-shot classification performance. We find that a large number of recent methods for few-shot learning embed representations on the hypersphere. Hence, we seek to understand what makes the hypersphere a particularly suitable embedding space for few-shot learning.

Our work on single-view deep clustering addresses the susceptibility of deep clustering models to find trivial solutions with non-meaningful representations. To address this issue, we present a new auxiliary objective that – when compared to the popular autoencoder-based approach – better aligns with the main clustering objective, resulting in improved clustering performance. Similarly, our work on multi-view clustering focuses on how representations can be learned from multi-view data, in order to make the representations suitable for the clustering objective. Where recent methods for deep multi-view clustering have focused on aligning view-specific representations, we find that this alignment procedure might actually be detrimental to representation quality. We investigate the effects of representation alignment, and provide novel insights on when alignment is beneficial, and when it is not. Based on our findings, we present several new methods for deep multi-view clustering – both alignment and non-alignment-based – that out-perform current state-of-the-art methods.

Our first work on few-shot learning aims to tackle the hubness problem, which has been shown to have negative effects on few-shot classification performance. To this end, we present two new methods to embed representations on the hypersphere for few-shot learning. Further, we provide both theoretical and experimental evidence indicating that embedding representations as uniformly as possible on the hypersphere reduces hubness, and improves classification accuracy. Furthermore, based on our findings on hyperspherical embeddings for few-shot learning, we seek to improve the understanding of representation norms. In particular, we ask what type of information the norm carries, and why it is often beneficial to discard the norm in classification models. We answer this question by presenting a novel hypothesis on the relationship between representation norm and the number of a certain class of objects in the image. We then analyze our hypothesis both theoretically and experimentally, presenting promising results that corroborate the hypothesis.

Acknowledgements

First and foremost I would like to thank my main supervisor, Associate Professor Michael Kampffmeyer. Thank you for always having an open door, and for always answering my questions with enthusiasm and insight. Your guidance has been invaluable to me over the course of these last three years. Second, I would like to thank my co-supervisors, Professor Robert Jenssen and Sigurd Løkse. Your perfect balance between support and criticism has been crucial to the works included in this thesis.

To my fellow students and co-workers at the UiT Machine Learning Group, thank you all for the exciting and inspiring discussions, talks, and presentations. The last three years would not have been nearly as enjoyable without all of you.

Lastly, to my friends and family, to Eva, and to Elsa, thank you for your love and support, and for continuously reminding me that there is more to life than just work.

Daniel Johansen Trosten,
Tromsø, May 2023.

Contents

Figures

Abbreviations

Adam	Adaptive Moment Estimation
AE	Autoencoder
BN	Batch Normalization
CNN	Convolutional Neural Network
CoMVC	Contrastive Multi-view Clustering
CSD	Caucy-Schwarz Divergence
DDC	Deep Divergence-based Clustering
DEC	Deep Embedded Clustering
DNN	Deep Neural Network
EAMC	End-to-end Adversarial-attention Network For Multi-modal Clustering
EASE	Unsupervised Discriminant Subspace Learning
FSL	Few-shot Learning
GAP	Global Average Pooling
GPT	Generative Pre-trained Transformer
IB	Information Bottleneck
IDEC	Improved Deep Embedded Clustering
IID	Independent And Identically Distributed
KDE	Kernel Density Estimate
KLD	Kullback-Leibler Divergence
MLP	Multilayer Perceptron
MSE	Mean Squared Error
MVC	Multi-view Clustering
NCH	Norm-count Hypothesis

noHub	Uniform Hyperspherical Structure-preserving Embeddings
noHub-S	NoHub With Support Labels
NT-Xent	Normalized Temperature-scaled Cross Entropy
OFM	Objective Function Mismatch
PDF	Probability Density Function
PRI	Principle Of Relevant Information
RELAX	Representation Learning Explainability
ReRep	Parameterless Transductive Feature Re-representation
ResNet	Residual Network
RKHS	Reproducing Kernel Hilbert Space
SGD	Stochastic Gradient Descent
SiMVC	Simple Multi-view Clustering
SSL	Self-supervised Learning
TCPR	Task Centroid Projection Removing
UCO	Unsupervised Companion Objective
vMF	Von Mises–Fisher
XAI	Explainable Artificial Intelligence
ZN	Z-score Normalization

Notation

$\mathbf{x}$	Column vector $\mathbf{x}$.
$\|\|\mathbf{x}\|\|$	Euclidean norm of vector $\mathbf{x}$.
$\mathbf{X}$	Matrix $\mathbf{X}$.
$\mathbf{X}^\top$	Transpose of matrix $\mathbf{X}$.
$I(i,j,c)$	Pixel (i,j) in channel c of image I.
$I \star J$	Convolution between images I and J.
$f \circ g$	Composition of functions f and g .
$\mathbf{x} \odot \mathbf{y}$	Element-wise product between vectors $\mathbf{x}$ and $\mathbf{y}$.
$\mathbf{x} \oslash \mathbf{y}$	Element-wise division between vectors $\mathbf{x}$ and $\mathbf{y}$.
$\mathbb{1}_{\{C\}}$	Indicator function equal to 1 if C is true, and 0 otherwise.
$\mathbb{R}$	Set of real numbers.
$\mathbb{N}^0_{<a}$	Set of non-negative integers $< a$.
$\mathbb{N}_{\leq a}$	Set of positive integers $\leq a$.
$\mathbb{E}_{\mathbf{z}\sim p}(f(\mathbf{z}))$	Expectation of f applied to random vector $\mathbf{z}$ with density p.
$H(\mathbf{x})$	Entropy of random vector $\mathbf{x}$.
$I(\mathbf{x},\mathbf{y})$	Mutual information between random vectors $\mathbf{x}$ and $\mathbf{y}$.
$D(p_1\|\|p_2)$	Divergence between distributions p_1 and p_2.

1 Introduction

Deep learning has led to tremendous scientific and technological advancements in recent years [15]. However, the success of these systems has traditionally required large, fully labeled datasets and supervised learning procedures. With the increasing amounts of data produced and required to train large machine learning models, it is quickly becoming infeasible for humans to generate labels for all these data. Thus, to continue pushing the limits of what machine learning-based systems can achieve, the field has gradually shifted towards developing models capable of learning from raw data with limited labels. This has led to impressive developments in both computer vision [16–29] and natural language processing [30–38].

Clustering, which refers to the process of discovering groups in unlabeled data, lies at the extreme of the limited labels regime, requiring no forms of label information whatsoever. The recent success of deep learning has inspired the development of deep neural network (DNN)-based clustering models, resulting in the *deep clustering* subfield [39–42]. The flexibility and representational power of DNNs give these models great potential in developing classification systems without labels, and to uncover new and unknown groups in large datasets. Although single-view (unimodal) data has received the most attention in deep clustering [25, 26, 39–42], several models have also been developed to cluster multi-view (multi-modal) data [43–46]. Multi-view clustering (MVC) comes with a unique set of challenges related to properly integrating information from multiple data sources, in order to exploit potential synergies across data sources. To this end, the recent success of self-supervised learning (SSL) in learning general-purpose data representations [16–18, 21–23, 47–49], has inspired the adoption of SSL-based components for deep MVC [44, 45]. However, this direction of research is still in its infancy, with much untapped potential for future advancements.

Few-shot learning (FSL) is another branch of machine learning, which lies next to clustering at the extreme of the limited labels regime. Where most supervised classification models require large numbers of labeled examples to discriminate between a set of classes, FSL aims to develop models capable of discriminating between new classes, based on very few examples – often requiring as little as 1 or 5 labeled instances in each class [50–75].

The majority of methods for clustering and FSL – as well as methods in other branches of machine learning – rely on assumptions on geometrical properties of the data. Usually, these assumptions manifest themselves as choices of distance functions or inner products between observations – presuming that these functions represent true notions of dissimilarity or similarity in the data. However, the data we gather today is diverse, with many formats, sizes, and characteristics – meaning that most modern data do not meet the assumptions made by the models. According to the manifold hypothesis [76, 77], high-dimensional data tend to lie on a lower-dimensional manifold. This manifold is likely to be non-Euclidean, meaning that the popular Euclidean distance and inner product do not accurately reflect the structure of the data. It is therefore necessary to create representations of data that enable proper quantification of similarity and dissimilarity through mathematics, and in particular, through geometry. Thus, one can say that *a defining characteristic of good representations is that true notions of similarity and dissimilarity in the data, are quantified through well-understood geometrical operations, such as distances or inner products.* For example, when classifying images of different objects, having good image representations would mean that, in the representation space, images from the same class are located close to each other, while images from different classes are placed far away from each other.

The success of deep learning can largely be attributed to the ability of DNNs to learn good representations from raw data [15]. This has severely reduced the need for domain-specific, hand-crafted representations that can be time-consuming and expensive to obtain. The ability of DNNs to learn high-quality representations is especially important when label information is limited, such as in clustering or FSL. This is because these models must rely on intrinsic, geometrical properties in the data, in order to compensate for the lack of a strong supervisory signal. Although deep learning has contributed to significant advancements in clustering and FSL, there are key challenges that have to be addressed in order to harness the full potential of DNNs in these settings.

1.1 Challenges

Despite the recent advancements in deep clustering and FSL, both of these fields face important challenges related to representation quality. The purpose of this section is thus to identify and outline some of these challenges, focusing on the ones we believe have the biggest impact on their respective fields.

Representation quality in deep clustering

Deep clustering is a subfield of deep learning where DNNs are trained with unsupervised loss functions, to discover unknown groups in data. The clustering losses are minimized without any label information, which can lead to a "corrupted" representation space where the loss is minimized, but the representations are no longer representative of the input space [41]. This leads to clusters that do not reflect true semantic structure in the data. Low-quality representations and insufficient similarity preservation has inspired the development of auxiliary objectives for deep clustering, aiming to learn representations that better reflect the semantic similarities in the data. However, the difference between the clustering task and the auxiliary task makes deep clustering models susceptible to objective function mismatch (OFM), where optimizing the auxiliary objective has a negative impact on the clustering objective. Preserving semantic similarities from inputs to representations, without introducing OFM, thus remains a key challenge in deep clustering.

Self-supervised learning for deep multi-view clustering

Deep MVC is a generalization of deep clustering to data from multiple views or modalities. Similar to deep single-view clustering, it is also crucial in deep MVC to learn good representations that represent semantic information in the data. To this end, many forms of SSL have been adapted to improve representations for deep MVC [43–46, 78–88]. However, current works exhibit large variations in the motivation and development of SSL tasks for deep MVC, resulting in a lack of direction and consistent improvement in clustering performance.

One SSL task that has shown particularly promising results is aligning representations from different views [44, 45, 83]. Alignment encourages the model to learn equal representations for all views of a given instance. Crucially, alignment results in good representations only *when the information that carries the true cluster membership is present in all views.* When this is not the case, alignment can have a negative impact on clustering performance, since it discards the cluster membership information present in a subset of the views. In summary, we find that the effects of SSL – and in particular, representation alignment – on deep MVC are not sufficiently understood, inhibiting future advancements in the field.

Embedding representations for few-shot learning

The objective of FSL is to develop systems capable of classifying new query samples based on a small number of labeled support examples from each class. FSL classifiers are trained based on representations provided by a DNN, which is trained on a similar dataset containing other classes than those encountered in inference. Although earlier works on FSL have focused on inductive inference for the queries, transductive classifiers – which leverage both support and query information when classifying the queries – have become increasingly popular during the last few years [50, 57, 63, 64, 68, 72–74].

Recent work has found that FSL is susceptible to the hubness problem, where a few samples appear frequently among the nearest neighbors of other samples, in the representation space [55, 89] Reducing hubness through an additional embedding step can thus lead to improved classification performance [55]. However, means to alleviate the hubness problem are still severely under-explored in FSL. Current work [55] does not provide theoretical guarantees on the reduction of hubness, and makes strong assumptions on the mean of the representation distribution. Continued research on hubness in FSL therefore has potential to advance the understanding and performance of FSL classifiers.

Recently, the hypersphere has emerged as a particularly promising embedding space for FSL [55, 58, 71, 72, 75]. In fact, even simple L_2 normalization has demonstrated remarkable gains in performance over un-normalized representations [71]. Despite its popularity however, we still lack fundamental understanding on why hyperspherical embeddings are especially suitable for FSL, and why discarding the norm information often results in improved classification performance.

1.2 Objectives

The challenges above indicate that there is much untapped potential in improving and understanding representation learning. The primary objective of this thesis is thus to identify and address challenges related to representations in high-dimensional spaces, focusing on deep clustering and FSL. To this end, we have the following specific objectives for the work presented in this thesis.

1. Develop methodology that improves similarity preservation in deep clustering, with reduced levels of OFM compared to previous approaches.
2. Better understand the effects of SSL and representation alignment in deep MVC, and thus provide more consistent directions for future advancements in the field.
3. Address the hubness problem in transductive FSL by developing new embedding techniques.
4. Improve the understanding of representation norms and hyperspherical embeddings in FSL.

1.3 Solutions

Within deep single-view clustering, we develop a new auxiliary objective to improve similarity preservation from inputs to representations (Paper I). The proposed unsupervised companion objectives (UCOs) are clustering losses attached to intermediate outputs in the model's DNN, encouraging a consistent clustering structure throughout the network. We use tensor kernels [90] to quantify similarities between intermediate outputs. Using tensor kernels is beneficial over vectorization-based kernels, since the former take the tensor's inherent structural information into account. The UCOs are clustering-based, meaning that they have a reduced level of OFM compared to other auxiliary objectives, such as autoencoders.

In order to address the challenges in deep MVC, we first present 3 specific pitfalls related to the alignment of representation distributions (Paper II). In this work we also propose two new models: Simple multi-view clustering (SiMVC) is a simple baseline model for deep MVC, which performs remarkably well compared to other, more complex methods. Contrastive multi-view clustering (CoMVC) extends SiMVC with an adaptive alignment procedure, improving on the current state-of-the-art for deep MVC.

With Paper III we seek to further improve the understanding of SSL and contrastive alignment in deep MVC. In particular, we investigate how these components influence the clustering model when the number of views becomes large. In addition, we propose DeepMVC, which is a unified framework for consistent evaluation and development of new and existing methods in deep MVC. Crucially, our framework includes a large number of previous approaches as instances, allowing for fair and accurate comparisons between models. With the DeepMVC framework and its open source implementation, we make key discoveries about the effects of SSL and contrastive alignment in deep MVC. We find that while contrastive alignment works well for few views, it actually decreases performance when the number of views becomes large. This effect is demonstrated both experimentally, and theoretically in a simplified setting.

In order to address the hubness problem in FSL, we prove in Paper IV that hubness is eliminated by embedding representations uniformly on the hypersphere. We then present two new embedding methods for FSL that embed representations from the feature extractor, improving the performance of FSL classifiers. The proposed uniform hyperspherical structure-preserving embeddings (noHub) and noHub with support labels (noHub-S) embed representations on the hypersphere according to a tradeoff between uniformity and local similarity preservation. This reduces hubness in the embeddings, while retaining class structure. We demonstrate experimentally that embedding representations with noHub and noHub-S improve FSL performance for a broad range of classifiers, feature extractors, and datasets.

To continue improving our understanding of hyperspherical embeddings beyond the hubness problem, we turn our attention to representation norms in Paper V. In particular, we present the norm-count hypothesis (NCH), conjecturing that the norm of a representation is a monotonically increasing function of the number of certain objects in the input image. In the paper, we provide both theoretical and experimental evidence that corroborates our hypothesis in a controlled setting.

1.4 Brief summary of papers

This section summarizes the contributions in the included papers, and briefly explains how these papers tackle challenges related to representation learning in deep clustering and FSL. Figure 1 presents an overview of how the included papers are related to the above challenges and objectives.

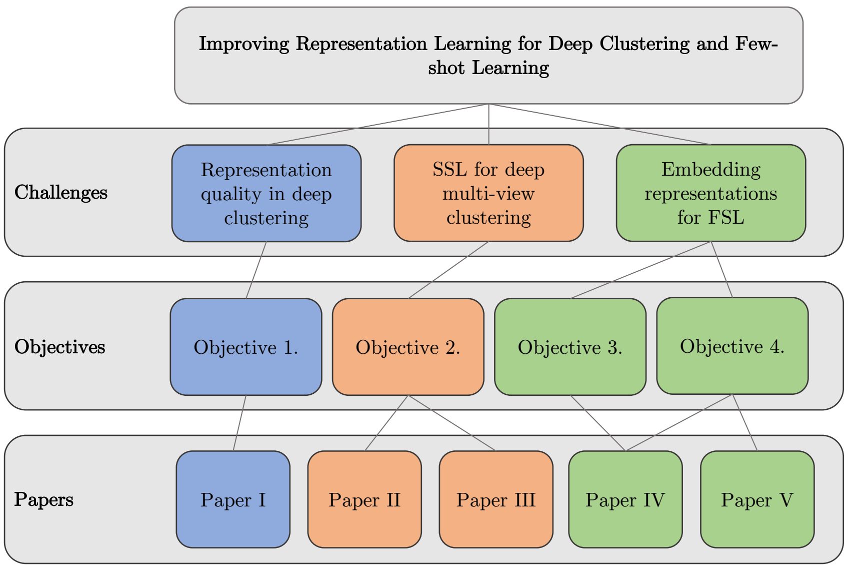

Improving Representation Learning for Deep Clustering and Few-shot Learning
Challenges
Representation quality in deep clustering
SSL for deep multi-view clustering
Embedding representations for FSL
Objectives
Objective 1.
Objective 2.
Objective 3.
Objective 4.
Papers
Paper I
Paper II
Paper III
Paper IV
Paper V

Figure 1: *Overview of challenges, objectives, and papers. Papers and objectives are colored according to the challenge they aim to address.*

Paper I

[I] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. "Leveraging Tensor Kernels to Reduce Objective Function Mismatch in Deep Clustering". *Pattern Recognition* (2023). Under Review.

This paper addresses the issue of similarity preservation in deep single-view clustering. We introduce the UCOs, which are novel auxiliary objectives designed to make the similarities in the representation space more representative of similarities in the input space. The UCOs use tensor kernels [90] to quantify similarities for both vectorial and tensorial intermediate representations. In the paper, we show that training a deep clustering model with the UCOs reduces OFM and improves performance, compared to analogous autoencoder-based models.

Paper II

[II] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. "Reconsidering Representation Alignment for Multi-view Clustering". In: *CVPR*. 2021.

This paper presents pitfalls of aligning distributions of view-specific representations in deep MVC. We find that aligning representation distributions can severely inhibit the model's ability to prioritize between views, forcing it to treat all views as equally informative. Distribution alignment can also lead to clusters being mis-aligned in the representation space, causing different clusters to overlap, thereby reducing cluster separability. Based on these insights, we develop two new models (SiMVC and CoMVC) for deep MVC. These models circumvent the pitfalls of distribution alignment, resulting in improved clustering performance on several multi-view datasets.

Paper III

[III] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. "On the Effects of Self-supervision and Contrastive Alignment in Deep Multi-view Clustering". In: *CVPR*. Highlight. 2023.

This paper generalizes results from Paper II to other forms of alignment, focusing on contrastive alignment. In a simplified setting, we prove that alignment enforces non-decreasing separability between clusters as the number of views increases. To further understand the effects of SSL and alignment, we develop the DeepMVC framework along with an open source implementation – enabling fair and accurate comparisons between methods and components. We propose several new instances of DeepMVC, advancing the current state-of-the-art in the field. Lastly, we provide key insights into SSL and contrastive alignment, showing experimentally that (i) SSL is beneficial in all evaluated models; and (ii) contrastive alignment has a negative impact on performance when the number of views becomes large. The latter is in-line with our theoretical analysis in the simplified setting.

Paper IV

[IV] Daniel J. Trosten, Rwiddhi Chakraborty, Sigurd Løkse, Kristoffer Wickstrøm, Robert Jenssen, and Michael Kampffmeyer. "Hubs and Hyperspheres: Reducing Hubness and Improving Transductive Few-shot Learning with Hyperspherical Embeddings". In: *CVPR*. 2023.

This paper addresses the hubness problem [89] in FSL. We begin by proving that the hyperspherical uniform distribution has a vanishing density gradient in all directions tangent to the hypersphere, at all points on the hypersphere. Consequentially, the hyperspherical uniform is hubness-free. Based on these findings, we present two new methods to embed representations on the hypersphere, which provably optimize a tradeoff between uniformity and local similarity preservation. The proposed noHub and noHub-S result in reduced hubness and improved overall performance for a wide range of classifiers with several feature extractors and datasets.

Paper V

[V] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. "Norm-Count Hypothesis: On the Relationship Between Norm and Object Count in Visual Representations" (2023). In submission.

This paper presents new insight on the information carried by norms of representations produced by convolutional neural networks (CNNs). We present the NCH – which hypothesizes that there is a monotonically increasing relationship between the norm of a representation, and the number of objects present in the corresponding image, for which the CNN is trained to detect. In the paper, we prove that the NCH is true under assumptions on the CNN and the input images. We then conduct several experiments, whose results corroborate the NCH for both supervised, self-supervised and few-shot learning.

1.5 Other works

The following is a list of other papers and works I have contributed to during the course of this project. Most of these works also fall under the main objective of this thesis, namely to improve or to better understand representation learning in deep clustering and FSL. Thereby providing me with further inspiration while working on the included papers.

[6] Daniel J. Trosten, Robert Jenssen, and Michael C. Kampffmeyer. “Reducing Objective Function Mismatch in Deep Clustering with the Unsupervised Companion Objective”. In: *NLDL*. 2021. DOI: 10.7557/18.5709.

[7] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. “Reconsidering Representation Alignment for Multi-view Clustering”. In: *National Conference on Image Processing and Machine Learning (NOBIM)*. Extended abstract and oral presentation. 2021.

[8] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. “Reconsidering Representation Alignment for Multi-view Clustering”. In: *Visual Intelligence Days*. Extended abstract. 2021.

[9] Daniel J. Trosten. “Deep Clustering”. Invited talk at University of Manitoba. 2021.

[10] Daniel J. Trosten, Kristoffer K. Wickstrøm, Shujian Yu, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. “Deep Clustering with the Cauchy-Schwarz Divergence”. In: *AAAI Workshop on Information Theory for Deep Learning (IT4DL)*. Extended abstract and oral presentation. 2022.

[11] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. “On the Role of Self-supervision in Deep Multi-view Clustering”. In: *Visual Intelligence Days*. Poster presentation. 2022.

[12] Daniel J. Trosten. “Questionable Practices in Methodological Deep Learning Research”. In: *NLDL*. 2023. DOI: 10.7557/18.6804.

[13] Kristoffer K. Wickstrøm, Daniel J. Trosten, Sigurd Løkse, Ahcène Boubekki, Karl Øyvind Mikalsen, Michael C. Kampffmeyer, and Robert Jenssen. “RELAX: Representation Learning Explainability”. *International Journal of Computer Vision* (2023). DOI: 10.1007/s11263-023-01773-2.

[14] Daniel J. Trosten, Rwiddhi Chakraborty, Sigurd Løkse, Kristoffer Wickstrøm, Robert Jenssen, and Michael Kampffmeyer. “Hubs and Hyperspheres: Reducing Hubness and Improving Transductive Few-shot Learning with Hyperspherical Embeddings”. In: *National Conference on Image Processing and Machine Learning (NOBIM)*. Extended abstract. 2023.

1.6 Open source and open science

Research on machine learning and artificial intelligence is currently in the midst of a reproducibility crisis [91, 92]. A key driving factor behind this crisis is closed source code, and the resulting lack of detailed information on new models and developments in the field. Fortunately, it has become increasingly popular for researchers and labs to make code for models and experiments openly available – gradually improving the reproducibility of machine learning and artificial intelligence research. To adhere to these standards, and to continue the push towards more reproducible and transparent research, we have made the code for all published papers publicly available online. The links to the respective repositories are given in Part II.

1.7 Thesis overview

The rest of this thesis is divided into three parts. Part I provides a complete overview of background material for the included papers. An illustration of how the chapters in this part are related to the included papers is shown in Figure 2. The next part, Part II, summarizes the main directions of research for this thesis, and briefly explains the main contributions of the included papers. Following this, I present some concluding remarks on our research, and the future of the field. Finally, Part III provides full versions with supplementary materials of the included papers.

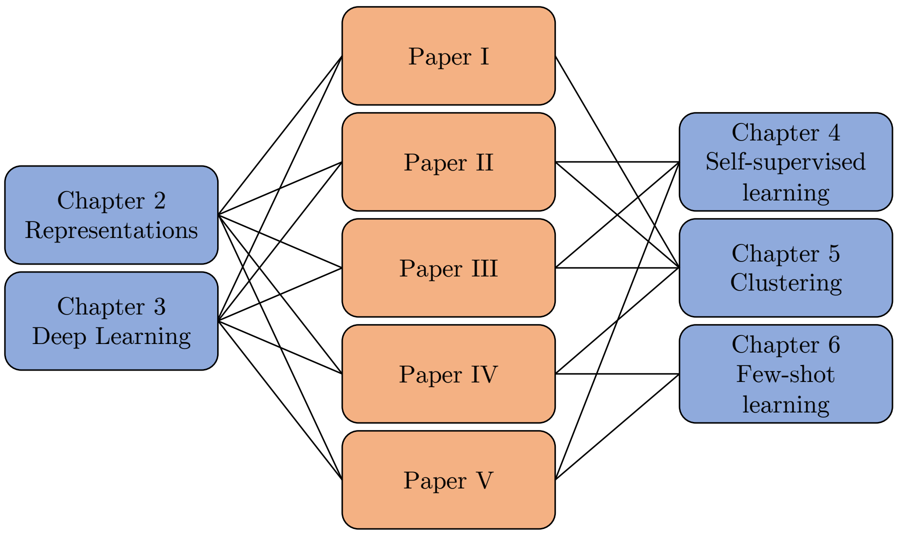


Figure 2: *Connections between chapters in Part I and the included papers. Chapters 2 and 3 are relevant for all papers, whereas Chapters 4, 5 and 6 provides background for different subsets of the included papers.*

Part I

Methodology and context

2
Representations

Most traditional machine learning systems make assumptions on the underlying data distribution, and on the space in which it resides. These assumptions are typically related to similarities in the data, and how these can be quantified by geometrical operations in the data space, such as inner products and distances. However, raw, real-world data might not meet these assumptions, causing traditional systems to fail in processing such data types. Because of this, it is often necessary to transform the data into representations, such that true notions of similarity are captured by geometrical operations in the representation space. Thus meeting the assumptions made by the models.

The first step to learning good representations is to quantify exactly what makes a representation good. Hence, in this chapter we will build upon the above intuition on representation quality, and formalize what we mean by a good representation. This will be done both in terms of pairwise similarities, and in terms of mutual information. Finally, we will discuss the manifold hypothesis [76, 77], and how it influences how we think about representation quality.

2.1 Representation quality

2.1.1 Quantification of pairwise similarities

Suppose there exists a "true" similarity function, $s_{\mathbf{x}}(\cdot, \cdot)$ that correctly encodes similarities in the input data. In a classification setting for instance, the value of $s_{\mathbf{x}}$ would be low between samples from different classes, and high between samples from the same class – implicitly encoding class memberships in the pairwise similarities. Although $s_{\mathbf{x}}$ is unknown and difficult to specify for most real-world data, we can use it to formalize notions of representation quality. For 3 different input samples $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ with representations $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3$, we say that for the representations to be good, they have to satisfy

$$s_{\mathbf{x}}(\mathbf{x}_1, \mathbf{x}_2) > s_{\mathbf{x}}(\mathbf{x}_1, \mathbf{x}_3) \Leftrightarrow s_{\mathbf{z}}(\mathbf{z}_1, \mathbf{z}_2) > s_{\mathbf{z}}(\mathbf{z}_1, \mathbf{z}_3) \tag{2.1}$$

where $s_{\mathbf{z}}$ is a known similarity function, such as the inner product or negative Euclidean distance. Alternatively, we can require that $s_{\mathbf{z}}$ is an approximation of $s_{\mathbf{x}}$

$$s_{\mathbf{x}}(\mathbf{x}_1, \mathbf{x}_2) \simeq s_{\mathbf{z}}(\mathbf{z}_1, \mathbf{z}_2). \tag{2.2}$$

Both of these formulations say that good representations transform the true, unknown similarity function, to a function that is known and computable. Interestingly, according to these conditions, it is not important what the representations actually contain, as long as the relationships between them meet the conditions.

The conditions in Equations (2.1) and (2.2) also illustrate that representation quality is context-dependent through the true similarity function $s_{\mathbf{x}}$. This means that a specific set of representations can have varying quality, based on how and where they are used.

If we assume that $s_{\mathbf{x}}$ encodes class memberships and that $s_{\mathbf{z}}$ is the negative Euclidean distance, representations that satisfy Equation (2.1) will be distributed as compact and well-separated clusters in the representation space. This makes it much easier for subsequent models to classify or cluster the data.

Within-class compactness and between-class separability naturally leads to a key property of good representations, namely *linear separability.* Informally, two classes are said to be linearly separable in the representation space, if there exists a hyperplane such that all representations from one class lie on one side of the hyperplane, and all representations from the other class lie on the other side of the hyperplane. This allows the search for discriminative models to be restricted to affine transformations in the representation space – significantly reducing the size of the search space.

2.1.2 Mutual information

The mutual information between two random variable is a measure on the degree of dependence between these two variables. For random vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, the mutual information is defined as the Kullback-Leibler divergence (KLD) between the joint distribution of $\mathbf{x}_1$ and $\mathbf{x}_2$, and the product of marginals [93, Ch. 1]

$$I(\mathbf{x}_1, \mathbf{x}_2) = D_{\text{KL}}(p_{(\mathbf{x}_1, \mathbf{x}_2)} || \, p_{\mathbf{x}_1} \cdot p_{\mathbf{x}_2}). \tag{2.3}$$

Intuitively, the mutual information measures the reduction of uncertainty in $\mathbf{x}_1$ after observing $\mathbf{x}_2$ [93, Ch. 1]. For instance, if $\mathbf{x}_1$ and $\mathbf{x}_2$ are independent, the joint distribution is equal to the product of marginals, causing the KLD, and thereby the mutual information, to be 0. If the degree of dependence between $\mathbf{x}_1$ and $\mathbf{x}_2$ starts to increase, so will the difference between the joint and the product of marginals, resulting in an increase in KLD and mutual information.

A straightforward way to think about representation quality in terms of mutual information, is to say that a representation $\mathbf{z}$ of an input $\mathbf{x}$ is good if it maximizes the mutual information between them

$$\max_{p(\mathbf{z}|\mathbf{x})} I(\mathbf{x}, \mathbf{z}). \tag{2.4}$$

This is known as the InfoMax principle [94], and has inspired the development of *e.g.* the Deep InfoMax model [95] for representation learning using deep neural networks (DNNs). However, it is often not beneficial for $\mathbf{z}$ to encode *all* information about $\mathbf{x}$, especially if $\mathbf{x}$ contains noise. Instead, we might say that $\mathbf{z}$ is a good representation if it maximizes the amount of information *relevant* for a certain task, while minimizing the irrelevant information contained in $\mathbf{x}$. This idea is formulated by the information bottleneck (IB) principle [96]

$$\min_{p(\mathbf{z}|\mathbf{x})} \left(I(\mathbf{x}, \mathbf{z}) - \beta I(\mathbf{z}, \mathbf{y}) \right) \tag{2.5}$$

where β is a tradeoff parameter, and $\mathbf{y}$ denotes the target random variable for the task. Unfortunately, adapting the IB principle to learning representations requires information about the target (label), $\mathbf{y}$, meaning that it is not directly applicable in unsupervised settings.

The dependence on label information in the IB principle has inspired an alternative information-theoretic principle for representations, namely the principle of relevant information (PRI) [93, Ch. 8]

$$\min_{p(\mathbf{z}|\mathbf{x})} \left(H(\mathbf{z}) + \lambda D(p_{\mathbf{z}} || p_{\mathbf{x}}) \right) \tag{2.6}$$

where H and D denote arbitrary entropy and divergence measures, respectively. The λ hyperparameter controls the tradeoff between minimizing entropy and divergence. This principle requires $\mathbf{z}$ to have minimal entropy, while simultaneously preserving a certain amount of information about $\mathbf{x}$.

InfoMax, IB, and PRI are all promising principles for learning high-quality representations. The optimal choice between the three is likely a matter of application and available data types. The difference between InfoMax, and IB and PRI is that the latter two both includes notions of *compression* – discarding information that is less relevant in the input. This can be beneficial if the input contains noise or other artifacts irrelevant to the application at hand.

2.1.3 Kernels

We will now shift our focus towards how similarities between representations can be computed, and how some of the above information-theoretic quantities can be estimated. The following provides relevant background for the main methodology in Paper I, as well as the deep divergence-based clustering (DDC) clustering module used in Papers II and III.

Mercer's theorem and the kernel trick

The inner product is a natural way to measure the similarity between two vectors. It is however, limited by its linearity, making it unable to capture non-linear structures in the data. This is where *kernels* play a key role in machine learning. A kernel is a particular type of similarity function that corresponds to an inner product in some reproducing kernel Hilbert space (RKHS) [97, 98]. According to

Mercer's theorem [97], for any symmetric function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that satisfies

$$\int_{\mathbb{R}^d} \int_{\mathbb{R}^d} g(\mathbf{x}_1) k(\mathbf{x}_1, \mathbf{x}_2) g(\mathbf{x}_2) \mathrm{d}\mathbf{x}_1 \mathrm{d}\mathbf{x}_2 \geq 0 \tag{2.7}$$

for any square-integrable function g, there exists a mapping $\phi : \mathbb{R}^d \to \mathbb{H}$, such that

$$k(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_2) \rangle_{\mathbb{H}} \tag{2.8}$$

where $\langle \cdot, \cdot \rangle_{\mathbb{H}}$ denotes the inner product in the Hilbert space $\mathbb{H}$.

Mercer's theorem implies that methods that only depend on the data through inner products can be performed in $\mathbb{H}$, instead of in $\mathbb{R}^d$, by simply replacing the ordinary inner product with a suitable kernel. Assuming that the mapping ϕ associated with the chosen kernel is non-linear, this allows linear methods to become non-linear, by simply replacing the inner product. This *kernel trick* has inspired the non-linearization of many popular machine learning methods [99–102].

Kernels in information theory

In addition to specifying inner products in some RKHS, kernels can also be used to estimate unknown probability density functions (PDFs) from data. If a kernel k can be written as

$$k(\mathbf{x}_1, \mathbf{x}_2) = \tilde{k}(\mathbf{x}_1 - \mathbf{x}_2) \tag{2.9}$$

for some function, $\tilde{k}$, satisfying

$$\tilde{k}(\mathbf{x}) \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^d \tag{2.10}$$

and

$$\int_{\mathbb{R}^d} \tilde{k}(\mathbf{x}) \mathrm{d}\mathbf{x} = 1 \tag{2.11}$$

the kernel density estimate (KDE) of the distribution p is given by

$$\hat{p}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \tilde{k}(\mathbf{x}_i - \mathbf{x}) \tag{2.12}$$

where $\mathbf{x}_1, \dots, \mathbf{x}_n$ is an independent and identically distributed (IID) random sample from p [103].

KDEs allows certain information-theoretic quantities to be estimated from data. The Caucy-Schwarz divergence (CSD) between densities for instance, takes a favorable form when computed from KDEs, only requiring the kernel to be avaluated at pairwise differences between samples [104]. This result has been used in DDC [42], and in Papers I, II and III to formulate a clustering loss on DNN-based representations.

Kernels have also been used to develop new formulations of entropies, mutual information, and divergences [105, 106]. Since these measure do not require knowledge of the underlying data distribution, they are especially suitable in real-world applications, where the true distribution often is unknown and intractable.

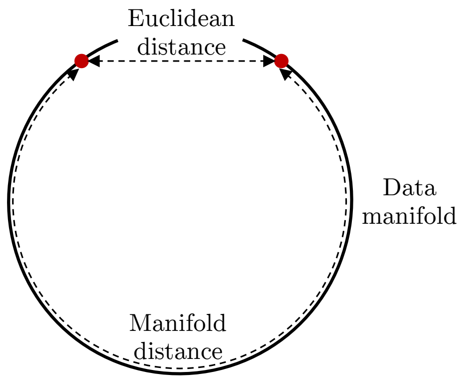


Figure 3: *Illustration of distances between two points on a circular arc manifold. The Euclidean distance is small, indicating that the points are similar, but the manifold distance (measured along the arc) is large, indicating that the points are dissimilar.*

Tensor kernels

The discussion on kernels has thus far assumed that kernels operate on vectors in $\mathbb{R}^d$. However, this is not always the case. In Paper I, we are interested in measuring similarities between intermediate representations in DNNs. These representations can be *tensors*, and not just vectors, meaning that the additional structure in the tensors' dimensions have to be taken into account in the similarity function. To this end, Paper I leverages the tensor kernel framework [90], which provides kernels specifically designed for tensors, to improve representation quality in deep clustering.

2.2 Manifold hypothesis

The manifold hypothesis states that real-world data tend to lie on a low-dimensional manifold in the high-dimensional ambient space [76, 77]. The reduction in dimensionality compared to the ambient space often originates from constraints imposed on the relationships between dimensions. Natural images for instance, can have a large number of pixels, but requirements on spatial coherence forces pixels within neighborhoods to be highly correlated, reducing the intrinsic dimensionality. Similarly, text, which is represented as a sequence of words, has constraints on how words should be ordered to make the resulting sequence meaningful.

Presuming that the data lies on a manifold, distances (or similarities) should be measured *along the manifold*, instead of in the ambient space. This will result in a distance function that is more true to the structure of the data, and incapable of "cheating" by measuring distance as a straight line between the points, passing through regions outside the manifold. Figure 3 shows an example where the manifold distance differs from the Euclidean distance. A method based on Euclidean distance would consider the two red points to be similar, perhaps belonging to the same class or cluster. A method based on the manifold distance however, would treat the red points as dissimilar, placing them in different classes or clusters.

Manifold-based distances have motivated the development of techniques for *manifold unwrapping* [76, 107–109]. The objective of these methods is to learn representations where the distance function on the manifold is transformed to the Euclidean distance in the representation space.

If we let $s_{\mathbf{x}}$ be a similarity function on the data manifold, the goal of manifold unwrapping essentially coincides with the conditions for representation quality (Equations (2.1) and (2.2)). As noted in Section 2.1.1, there might not exist a single $s_{\mathbf{x}}$ that is optimal for all tasks. Instead, we can envision many distance functions on the manifold, where some functions are better suited for certain tasks than others.

3 Deep learning

Deep learning – and in particular, deep neural networks (DNNs) – lie at the heart of many of today's most impactful advancements in machine learning and artificial intelligence [15, 110]. Their layered architectures with millions or billions of parameters, make DNNs extremely flexible and capable of producing high-quality representations from a wide range of complex input data.

In this chapter, we review some cornerstone DNN architectures relevant for this thesis. We also discuss how these architectures are optimized, and possible pitfalls and challenges related to the optimization procedure.

3.1 Multilayer perceptrons

3.1.1 The perceptron algorithm

The perceptron (McCulloch-Pitts neuron) [111, 112] is a simple algorithm for supervised binary classification. It is often regarded as the algorithm that later grew to the wide class of models that is DNNs [98]. The perceptron is inspired by the observation that the relations between nervous activity and neural events can be represented as propositional logic [111, 112]. Specifically, for a d-dimensional input observation $\mathbf{x}_i = [x_{i1}, \dots, x_{id}]^\top$, the perceptron computes the response

$$r_i = \sum_{j=1}^{d} w_j x_{ij} + b = \mathbf{w}^\top \mathbf{x} + b \tag{3.1}$$

and the output

$$y_i = \mathbb{1}_{\{r_i > 0\}} = \begin{cases} 1, & r_i > 0, \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

where $\mathbf{w} = [w_1, \dots, w_d]^\top$ is a vector of weights and b is a bias parameter. The perceptron thus computes a binary response that only triggers if the input signals are strong enough, w.r.t. the weights.

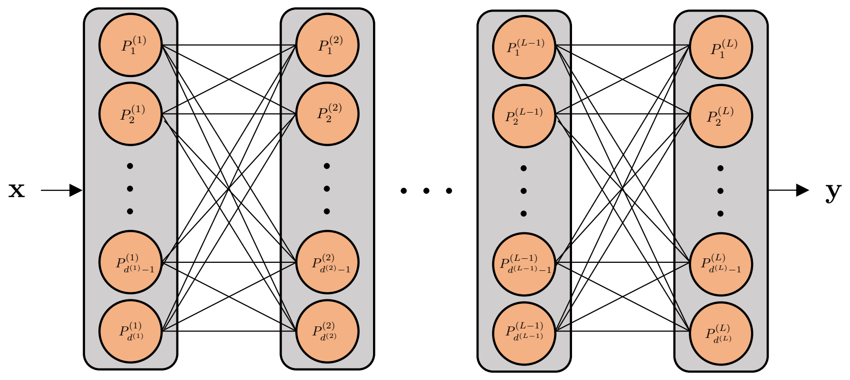


Figure 4: *Overview of a general multilayer perceptron with input* $\mathbf{x}$ *and output* $\mathbf{y} \in \mathbb{R}^{d^{(L)}}$. *In layer* l, *the* j*-th perceptron,* $P_j^{(l)}$, *performs the computation in Equation* (3.7).

Given a set of labeled training data $(\mathbf{x}_1, \tilde{y}_1), \ldots, (\mathbf{x}_n, \tilde{y}_n)$, where $\tilde{y}_i = 1$ if $\mathbf{x}_i$ belongs to the positive class, and $\tilde{y}_i = -1$ otherwise, the weights are determined by minimizing the *perceptron loss*

$$\mathcal{L}_{\text{Perceptron}} = \sum_{i=1}^{n} \mathbb{1}_{\{\text{sign}(r_i)=\tilde{y}_i\}} \tilde{y}_i r_i \tag{3.3}$$

where sign$(\cdot)$ is the sign function

$$\text{sign}(r_i) = \begin{cases} 1, & r_i \geq 0, \\ -1, & \text{otherwise} \end{cases}. \tag{3.4}$$

Minimizing $\mathcal{L}_{\text{Perceptron}}$ is done iteratively [98, 112], where at iteration t, the weights and bias are updated as

$$w_j(t+1) = w_j(t) + \eta(\tilde{y}_i - y_i)x_{ij}, \quad \forall\, j = 1, \ldots, d \tag{3.5}$$

$$b(t+1) = b(t) + \eta(\tilde{y}_i - y_i) \tag{3.6}$$

where η is the *learning rate*, which determines the step size taken in the minimization procedure. The optimization is terminated when all training samples are classified correctly, or when the loss converges.

3.1.2 Multilayer perceptrons

The perceptron is a linear model, and thus it makes strong assumptions on the geometry of the data. However, by stacking multiple perceptrons breadth-wise and depth-wise (see Figure 4), the multilayer perceptron (MLP) is capable of refining the data representation at each computational step. At an arbitrary layer l with $d^{(l)}$ perceptrons, the MLP computes the following output

$$y_{ij}^{(l)} = f^{(l)}(\mathbf{w}_j^{(l)\top} \mathbf{y}_i^{(l-1)} + b_j^{(l)}), \quad j = 1, \ldots, d^{(l)} \tag{3.7}$$

where $(\mathbf{w}_j^{(l)}, b_j^{(l)})$ are the weights and biases of perceptron j in layer l, and $\mathbf{y}_i^{(l-1)} = [y_{i1}^{(l-1)}, \dots, y_{id^{(l-1)}}^{(l-1)}]^\top$ is the output vector of the previous layer, with $\mathbf{y}_i^{(0)} = \mathbf{x}_i$.

The activation function $f^{(l)} : \mathbb{R} \to \mathbb{R}$ is now assumed to be a general, non-linear, piecewise-differentiable[1] function. We require $f^{(l)}$ to be non-linear since its argument is an affine transformation of the previous representation. Using a linear activation function would thus result in an affine model, severely limiting the model's representational power. Early work on MLPs focused on sigmoid-like activation functions, such as the hyperbolic tangent:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.8}$$

and the logistic sigmoid

$$\mathrm{sigmoid}(x) = \frac{1}{1 + e^{-x}}. \tag{3.9}$$

However, later work on neural network optimization (see Section 3.3) has found piecewise linear activation functions to work better for models with many layers. These include the rectified linear unit

$$\mathrm{ReLU}(x) = \max\{0, x\} \tag{3.10}$$

and the leaky rectified linear unit

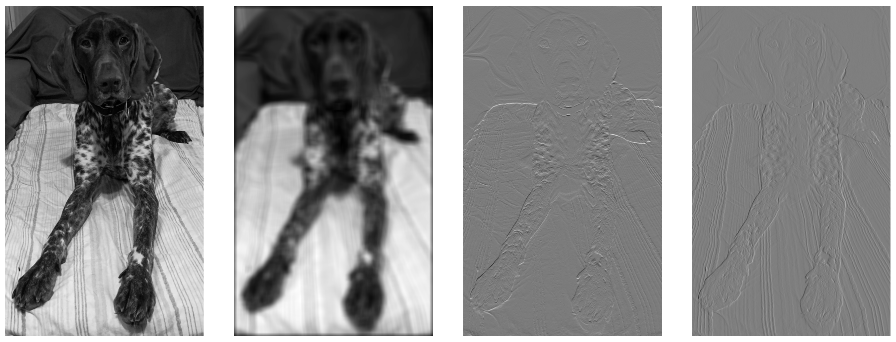
$$\mathrm{LeakyReLU}(x) = \begin{cases} \alpha x, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{3.11}$$

where $\alpha \in (0, 1)$ is a hyperparameter, typically set to 0.01.

From Equation (3.7), it is clear that an MLP with L layers produces $L - 1$ *intermediate* representations $\mathbf{y}_i^{(1)}, \dots, \mathbf{y}_i^{(L-1)}$ for the input observation $\mathbf{x}_i$. The key idea in MLP is then that each representation is somewhat better than the previous one, meaning that the model gradually refines the representation at each layer. Whether the model is actually capable of making the representations better depends on the weight vectors. However, due to the layer-wise stacking of perceptrons, it is no longer feasible to use the perceptron algorithm from Section 3.1.1 to determine the optimal weights. Instead, these models are trained using Backpropagation [113], which will be discussed in Section 3.3.

Several works have proven that MLP with arbitrary width [114, 115], arbitrary depth [116], and finite width and depth [117] are *universal approximators*. Informally, this implies that, for any continuous function, $g : [0, 1]^d \to \mathbb{R}$, there exists analytic activation functions, such that an MLP can approximate g arbitrarily well. This is a truly remarkable result on the representational power of MLPs, especially considering that they are constructed from a collection of linear units.

[1]The reason for requiring differentiability will be discussed in Section 3.3.



(a) *Original* (b) *Blurred* (c) *Horizontal edges detected* (d) *Vertical edges detected*

Figure 5: *Examples of three different convolution operators applied to a grayscale image.*

3.2 Convolutional neural networks

Images often have high extrinsic dimensionality, originating from high spatial resolution and multiple channels. However, due to geometric constraints on the space of images, the *intrinsic* dimensionality of the image space is likely to be much lower than the extrinsic dimensionality of the ambient space. Strong local correlation between pixels, combined with translation equivariance/invariance makes it natural to use functions that operate on local regions (patches), and treats each patch the same way, regardless of its position in the image. Combining these two properties, in addition to linearity, results in the *convolution operator* – the key building block in convolutional neural networks (CNNs) [118].

3.2.1 The convolution operator

The central idea in convolution is that a single filter – typically with size much smaller than the input – is applied to all patches of the input image. Convolution thus implements *parameter sharing*, meaning that the same parameters (filter) are applied to multiple parts of the input. The two main advantages of parameter sharing are:

1. The number of parameters is significantly reduced compared to an analogous operator with distinct parameters for each pixel in the input image. This makes the operator both more memory efficient, and faster to apply.

2. Regions in the image are processed in the same way, regardless of their global position. This is especially useful in classification models, where an image of a certain object should be classified in a certain way, regardless of the object's position in the image.

Convolution is defined as follows. First, let $\mathcal{I}_{C,H,W}$ denote the set of C-channel

Table 1: *Summation bounds and output sizes with different padding strategies for the convolution operator.*

	Padding		
	Valid	Same	Full
L_x	$\max\{0, x - W + 1\}$	$\max\{0, x - W - \lfloor\frac{w-1}{2}\rfloor + 1\}$	$\max\{0, x - W - w + 2\}$
U_x	$\min\{w - 1, x\}$	$\min\{w - 1, x + \lceil\frac{w-1}{2}\rceil\}$	$\min\{w - 1, x + w + 1\}$
L_y	$\max\{0, y - H\}$	$\max\{0, y - H - \lfloor\frac{h-1}{2}\rfloor + 1\}$	$\max\{0, y - H - h + 2\}$
U_y	$\min\{h - 1, y\}$	$\min\{h - 1, y + \lceil\frac{h-1}{2}\rceil\}$	$\min\{h - 1, y + h + 1\}$
H'	$H - h + 1$	H	$H + h - 1$
W'	$W - w + 1$	W	$W + w - 1$

images with height H and width W

$$\mathcal{I}_{C,H,W} = \{I : \mathbb{N}^0_{<C} \times \mathbb{N}^0_{<W} \times \mathbb{N}^0_{<H} \to \mathbb{R}\}. \tag{3.12}$$

Then, for an image $I \in \mathcal{I}_{C,H,W}$ and a filter $F \in \mathcal{I}_{C,h,w}$[2], the convolution between F and I is defined as

$$(F \star I)(x, y) = \sum_{c=1}^{C} \sum_{L_x \leq x' \leq U_x} \sum_{L_y \leq y' \leq U_y} F(c, x', y') I(c, x - x', y - y') \tag{3.13}$$

$$(x, y) \in \mathbb{N}^0_{<W'} \times \mathbb{N}^0_{<H'} \tag{3.14}$$

where the summation bounds (L_x, U_x) and (L_y, U_y), and the output dimensions H', W' are defined by the type of zero-padding used in the convolution (see Table 1).

Figure 5 shows examples of output images obtained by convolving the input image (Figure 5a) with different filters. The blurring filter is a 23×23 filter with all values equal to 23^{-2}. This introduces an averaging effect, where a pixel in the output image is equal to the average value of all pixels in a 23×23 neighborhood surrounding the corresponding pixel in the input image. The edge detectors used for Figures 5c and 5d are horizontal and vertical Sobel operators [119], defined as

$$F_{\text{Horizontal}} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \quad \text{and} \quad F_{\text{Vertical}} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}. \tag{3.15}$$

From Equation (3.13), it can be shown that the convolution operator has the following key properties [120].

- **Commutativity:** For an image $I \in \mathcal{I}_{C,H,W}$ and a filter $F \in \mathcal{I}_{C,h,w}$

$$F \star I = I \star F \tag{3.16}$$

[2]We assume $h \leq H$ and $w \leq W$ for convenience.

- **Linearity:** For images $I_1, I_2 \in \mathcal{I}_{C,H,W}$, and constants $c_1, c_2 \in \mathbb{R}$
$$F \star (c_1 I_1 + c_2 I_2) = c_1(F \star I_1) + c_2(F \star I_2). \tag{3.17}$$
This allows convolution to be implemented as matrix multiplication – an operation that has been made extremely efficient on modern computing hardware.

- **Translation equivariance:** For a translation operator
$$T_{(x',y')}(I)(c,x,y) = I(c, x - x', y - y') \tag{3.18}$$
we have
$$F \star (T_{(x',y')}(I)) = T_{(x,y)}(F \star I). \tag{3.19}$$
This property is closely related to the parameter sharing property discussed above.

In the following we will see how the convolution operator can be used to build a DNN architecture for image data.

3.2.2 Convolutional neural networks

Convolving an image with a filter results in a new, filtered image. This new image can be seen as a representation of the input image, where certain features are enhanced or discarded. If we now convolve the filtered image with a new filter, we can obtain another, more refined representation, further enhancing the relevant features, and discarding irrelevant features. This is the key idea behind CNNs: stacking multiple convolutions to gradually refine the representation to make it better for downstream processing.

In a CNN, each step (application of convolution) is referred to as a *layer*, and the ordered collection of layers is referred to as a CNN. In order to make it possible for the CNN to detect multiple types of features, each layer performs convolutions with a set of filters. If there are $C^{(l)}$ filters in layer l, the layer will produce $C^{(l)}$ single-channel images, which can be stacked channel-wise to produce a single $C^{(l)}$ channel image. Specifically, each layer computes its output $I^{(l)} \in \mathcal{I}_{C^{(l)},H^{(l)},W^{(l)}}$ as
$$I^{(l)}(c,x,y) = f^{(l)}((F_c^{(l)} \star I^{(l-1)})(x,y) + b_c^{(l)}), \quad c \in \mathbb{N}^0_{<C^{(l)}} \tag{3.20}$$
where $I^{(l)}$ is the output of the previous layer, and $f^{(l)} : \mathbb{R} \to \mathbb{R}$ is an activation function applied element-wise. The filters $\{F_1^{(l)}, \dots, F_{C^{(l)}}^{(l)}\}$ and biases $\{b_1^{(l)}, \dots, b_{C^{(l)}}^{(l)}\}$ are learnable parameters in layer l. We will discuss how these parameters are found in Section 3.3.

Activation functions

Similar to the MLP, CNNs are composed of linear operations. Therefore, we require the activation functions $f^{(l)}$ to be non-linear, so that the resulting CNN is a non-linear model. Frequently used activation functions include ReLU (Equation (3.10)), and LeakyReLU (Equation (3.11)).

Pooling: from translation *equivariance* to translation *invariance*

The convolution operator is translation equivariant, meaning that convolving a filter with a translated input image, is equivalent to convolving with the original image, and then translating the result. However, many downstream image processing tasks, such as classification, are inherently translation *invariant*: the model should provide the same output, regardless of the global location of the region of interest.

To make CNNs translation invariant, it is common to insert local pooling layers between convolutional layers in the network. A local pooling layer aggregates information from a local neighborhood into a single pixel, both reducing the size of the output image, and introducing invariance to local translation. The most common local pooling operation is max pooling

$$\mathrm{MaxPool}_{h,w}(I)(x,y) = \max_{\substack{wx \le x' \le w(x+1) \\ hy \le y' \le h(y+1)}} \{I(x',y')\}. \tag{3.21}$$

This operation produces a new image where an output pixel is the maximum value of the corresponding $h \times w$ neighborhood in the input image. Aggregating pixels from a local neighborhood in this way makes the network invariant to translations within the neighborhood. Including several max pooling operations after different layers in the network, makes the intermediate representations gradually more translation invariant.

In modern CNN design, such as in classification models, it is common to attach a global average pooling (GAP) operation to the last layer of the network. For a given channel, GAP computes the average across all spatial dimensions, collapsing all spatial information into a single value

$$\mathrm{GAP}(I)_k = \frac{1}{WH} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} I(k,x,y) \tag{3.22}$$

For an image with C channels, GAP produces the vectorial representation

$$\mathrm{GAP}(I) = \begin{bmatrix} \mathrm{GAP}(I)_1 \\ \vdots \\ \mathrm{GAP}(I)_C \end{bmatrix} \in \mathbb{R}^C. \tag{3.23}$$

Since GAP aggregates information from all spatial positions, the resulting representation is completely translation invariant. This makes the representation suited for downstream tasks that benefit from translation invariance – *e.g.* classification.

Overview of the CNN architecture

Figure 6 illustrates an example CNN used for classification. The network consists of L convolutional layers followed by a GAP step. The vector produced by the GAP step is processed by a perceptron layer, producing the predicted class for the input image.

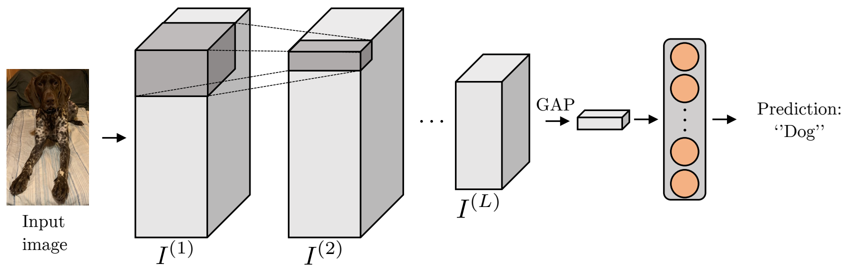


Figure 6: *CNN-based classification model consisting of L convolutional layers, global average pooling, and a perceptron classification layer. The size of $I^{(L)}$ is reduced compared to $I^{(1)}$ and $I^{(2)}$ due to intermediate pooling layers (not shown in the figure).*

3.2.3 Residual connections and ResNets

The power of CNNs, and DNNs in general, comes from their ability to stack multiple layers to produce semantically meaningful representations from raw data. The network depth is thus crucial for performance [121, 122]. However, He *et al.* [122] observe that, for standard CNNs as defined above, increasing the depth results in saturation followed by a rapid drop in classification performance and increase in training loss.

This behavior is counter-intuitive in two ways:

1. Deeper networks should provide better representations since they have higher representational power, compared to shallower networks. This should lead to improved performance.
2. If a shallow network is good for the task at hand, the deeper network should mimic the shallow network with some of its layers, and learn identity mappings for the remaining layers. The deeper model should thus have performance equal to, or better than, the shallow model.

He *et al.* [122] argue that the saturation and drop originates from difficulties in the optimization procedure: shallower models are "easier" to optimize compared to deeper models, and deep models are not able to learn the identity mapping required to mimic shallower models. To alleviate this problem, He *et al.* [122] propose to add explicit identity mappings, referred to as *residual connections*, bypassing pairs of convolutional layers. Recall that, for layers l and $l+1$, the standard network (without pooling) produces outputs

$$I^{(l)}(c,x,y) = f^{(l)}((F_c^{(l)} \star I^{(l-1)})(x,y) + b_c^{(l)}), \tag{3.24}$$

$$I^{(l+1)}(c,x,y) = f^{(l+1)}((F_c^{(l+1)} \star I^{(l)})(x,y) + b_c^{(l+1)}). \tag{3.25}$$

Adding a residual connection between these layers results in

$$\tilde{I}^{(l+1)} = f_{\text{residual}}^{(l+1)}(I^{(l+1)} + I^{(l-1)}) \tag{3.26}$$

where $f_{\text{residual}}^{(l+1)}$ is an activation function (typically ReLU). Subsequent layers of the network now receive the representation $\tilde{I}^{(l+1)}$ instead of $I^{(l+1)}$. The explicit residual connections make it easier for the network to learn identity mappings, for tasks where this is beneficial. A network with residual connections is called a residual network (ResNet). These networks have been shown to perform much better than standard CNNs, when the network depth becomes large [122].

3.3 Training deep neural networks

Up until this point we have not discussed how the parameters of DNNs are found based on the given training data. The process of determining a DNN's parameters is usually referred to as *training* or *optimizing* the network.

Throughout this section, we will assume that $g_{\boldsymbol{\theta}} : X \to Y \subseteq \mathbb{R}^d$ is a DNN with parameters $\boldsymbol{\theta}$, mapping from inputs in X to outputs $Y \subseteq \mathbb{R}^d$.

3.3.1 Loss functions

A loss function (or objective function), $\mathcal{L}$, is a real-valued function that measures the performance of a DNN, relative to a given training dataset. Lower values of the loss function indicate better performing models. Loss functions are said to be either *supervised* or *unsupervised* depending on whether labels are available when training the network. Examples of supervised loss functions include the cross-entropy loss for classification

$$\mathcal{L}_{\text{X-ent}}(\boldsymbol{\theta}, \mathbf{x}_1, \tilde{\mathbf{y}}_1, \dots, \mathbf{x}_n, \tilde{\mathbf{y}}_n) = \sum_{i=1}^{n} \sum_{j=1}^{c} \tilde{y}_{ij} \log y_{ij} \tag{3.27}$$

and the mean squared error for regression

$$\mathcal{L}_{\text{MSE}}(\boldsymbol{\theta}, \mathbf{x}_1, \tilde{\mathbf{y}}_1, \dots, \mathbf{x}_n, \tilde{\mathbf{y}}_n) = \frac{1}{n} \sum_{i=1}^{n} ||\tilde{\mathbf{y}}_i - \mathbf{y}_i||^2 \tag{3.28}$$

where $\mathbf{y}_i = g_{\boldsymbol{\theta}}(\mathbf{x}_i)$ is the network output for the input $\mathbf{x}_i$, and $\tilde{\mathbf{y}}_i$ is the corresponding ground truth.

Examples of unsupervised loss functions for representation learning, clustering and few-shot learning will be presented in detail in Chapters 4, 5 and 6, respectively.

3.3.2 Gradient descent

When training a DNN we are interested in finding the parameters, $\boldsymbol{\theta}^*$, which minimize the chosen loss function, $\mathcal{L}(\boldsymbol{\theta})$[3]

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \Theta} \mathcal{L}(\boldsymbol{\theta}) \tag{3.29}$$

[3]Note that we have omitted the dependence on the training data to simplify the notation.

where Θ denotes the space of possible parameters. However, finding the *global* minimum of $\mathcal{L}$ is not computationally feasible for complex models, such as DNNs. Instead, we are usually satisfied with a *local* minimum of $\mathcal{L}$, as such a minimum is much easier to find.

Gradient descent is a well known method to iteratively find local minima of a differentiable function. Given an initial parameter vector $\boldsymbol{\theta}(0)$, gradient descent is an iterative approach that, at step t, updates the current parameter vector as

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \mu(t)\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}(t)} \tag{3.30}$$

where $\mu(t)$ is the *learning rate* at step t. The learning rate is a hyperparameter that determines the size of the steps taken at each training iteration. Unfortunately, there is not a single optimal learning rate. Rather, it depends on the shape of the loss function w.r.t. the parameters – particularly whether the minima are "sharp" or "flat". The difficulties of setting the learning rate has lead to the development of methods that automatically tune the learning rate based on the geometry of the loss function. Examples of such methods will be given in Section 3.3.4.

It can be shown that, under mild constraints on the learning rate and loss function, the iterative procedure in Equation (3.30) converges to a local minimum of $\mathcal{L}(\boldsymbol{\theta})$ [123].

3.3.3 Backpropagation

Backpropagation [113] is a gradient descent-based algorithm used to train DNNs. DNNs have non-linear computations with a large number of parameters, making it computationally infeasible to compute gradients numerically. Therefore, in backpropagation the gradients are computed *analytically* – resulting in a much faster and more accurate optimization procedure. Despite their complexity and highly non-linear nature, the layer-wise architecture of DNNs means that they are essentially compositions of functions (layers). Gradients can therefore be computed using the multivariate chain rule of differentiation. Starting at the last layer of the network, gradients are propagated backward in order to update parameters in earlier layers – which is why the algorithm is called *backpropagation*.

Suppose that the DNN can be decomposed as

$$g_{\boldsymbol{\theta}} = g^{(L)}_{\boldsymbol{\theta}^{(L)}} \circ g^{(L-1)}_{\boldsymbol{\theta}^{(L-1)}} \circ \cdots \circ g^{(1)}_{\boldsymbol{\theta}^{(1)}} \tag{3.31}$$

where $\boldsymbol{\theta}^{(l)}$ are the parameters in layer l, and L is the number of layers. Further, let $\mathbf{y}^{(l)} = g^{(l)}_{\boldsymbol{\theta}^{(l)}}(\mathbf{y}^{(l-1)})$ be the output of layer l, given the output of the previous layer. Then, the gradient of the loss w.r.t. the parameters in layer l can be computed as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}^{(l)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}^{(L)}} \frac{\partial g^{(L)}_{\boldsymbol{\theta}^{(L)}}}{\partial \boldsymbol{\theta}^{(l)}} \tag{3.32}$$

$$= \frac{\partial \mathcal{L}}{\partial \mathbf{y}^{(L)}} \frac{\partial g^{(L)}_{\boldsymbol{\theta}^{(L)}}}{\partial \mathbf{y}^{(L-1)}} \cdots \frac{\partial g^{(l+1)}_{\boldsymbol{\theta}^{(l+1)}}}{\partial \mathbf{y}^{(l)}} \cdot \frac{\partial g^{(l)}_{\boldsymbol{\theta}^{(l)}}}{\partial \boldsymbol{\theta}^{(l)}} \tag{3.33}$$

```
input: Randomly initialized DNN, $g_{\boldsymbol{\theta}(0)}$, with $L$ layers
input: Training data, $\mathbf{x}_1, \dots, \mathbf{x}_n$
input: Number of iterations, $T$
input: Sequence of learning rates, $\mu(0), \dots, \mu(T-1)$
  for $t = 0, \dots, T-1$ do
      $\mathbf{y}_i \leftarrow g_{\boldsymbol{\theta}(t)}(\mathbf{x}_i), i = 1, \dots, n$                ▷ Compute outputs
      $\ell(t) \leftarrow \mathcal{L}(\boldsymbol{\theta}(t))$                                        ▷ Compute loss
      for $l = L, \dots, 1$ do                                                   ▷ Compute gradients
          $\mathbf{g}(t) \leftarrow \frac{\partial \ell(t)}{\partial \boldsymbol{\theta}^{(l)}}|_{\boldsymbol{\theta}^{(l)}=\boldsymbol{\theta}^{(l)}(t)}$
      for $l = 1, \dots, L$ do                                                   ▷ Update parameters
          $\boldsymbol{\theta}^{(l)}(t+1) \leftarrow \boldsymbol{\theta}^{(l)}(t) - \mu(t)\,\mathbf{g}(t)$
  output Trained DNN, $g_{\boldsymbol{\theta}(T)}$
```

Algorithm 1: *Backpropagation algorithm for training a DNN, $g_{\boldsymbol{\theta}}$.*

where the necessary Jacobian matrices

$$\frac{\partial g^{(L)}_{\boldsymbol{\theta}^{(L)}}}{\partial \mathbf{y}^{(L-1)}}, \dots, \frac{\partial g^{(l+1)}_{\boldsymbol{\theta}^{(l+1)}}}{\partial \mathbf{y}^{(l)}} \quad \text{and} \quad \frac{\partial g^{(l)}_{\boldsymbol{\theta}^{(l)}}}{\partial \boldsymbol{\theta}^{(l)}} \tag{3.34}$$

can be computed by differentiating Equation (3.7) if $g_{\boldsymbol{\theta}}$ is an MLP, and Equation (3.20) if $g_{\boldsymbol{\theta}}$ is a CNN. The resulting gradient can then be used in the iterative algorithm in Equation (3.30) to update the parameters of layer l. The complete backpropagation algorithm is illustrated in Algorithm 1.

3.3.4 Optimization techniques

Numerous methods have been proposed to improve the standard backpropagation algorithm [124–131]. The goal of these methods is to improve the optimization procedure, resulting in faster convergence, improved performance, and reduced hyperparameter sensitivity. In the following, we briefly summarize two of the most popular optimization techniques.

Stochastic gradient descent

Computing gradients and parameter updates based on the full dataset is not computationally feasible when the number of samples becomes large. Optimization is therefore typically performed with *mini batches*, which are random partitions of the dataset with size much smaller than the number of samples. In addition, introducing randomness to the optimization procedure has shown to be beneficial for convergence and performance [98]. Stochastic gradient descent (SGD) is the simplest backpropagation algorithm that uses randomly sampled mini batches. In SGD, the update rule is modified as

$$\boldsymbol{\theta}_{\mathcal{B}}(t+1) = \boldsymbol{\theta}(t) - \mu \frac{\partial \mathcal{L}_{\mathcal{B}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}(t)} \tag{3.35}$$

where subscript $\mathcal{B}$ means that the parameter update and loss are computed based on the mini batch $\mathcal{B}$.

Adaptive moment estimation

Adaptive moment estimation (Adam) extends the SGD algorithm to include an adaptive momentum term, improving convergence and performance. In Adam, the parameter update is formulated as

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \mu\ \hat{\mathbf{m}}(t) \oslash (\sqrt{\hat{\mathbf{v}}(t)} + \varepsilon) \tag{3.36}$$

where μ and ε are hyperparameters[4]. The vectors $\hat{\mathbf{m}}(t)$ and $\hat{\mathbf{v}}(t)$ are estimates of the element-wise un-centered first and second moments of the gradient

$$\hat{\mathbf{m}}(t) = \frac{\mathbf{m}(t)}{1-\beta_1^t}, \quad \mathbf{m}(t) = \beta_1 \mathbf{m}(t-1) + (1-\beta_1)\mathbf{g}(t) \tag{3.37}$$

$$\hat{\mathbf{v}}(t) = \frac{\mathbf{v}(t)}{1-\beta_2^t}, \quad \mathbf{v}(t) = \beta_2 \mathbf{v}(t-1) + (1-\beta_1)\mathbf{g}(t)^2 \tag{3.38}$$

where $\mathbf{g}(t)$ is the gradient $\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}(t)}$. The multiplication with $(1-\beta_{\{1,2\}})^{-1}$ is done to correct the bias in the parameter estimates, originating from zero-initialization.

3.3.5 Vanishing and exploding gradients

Despite its success in training DNNs, the backpropagation algorithm is not problem free. Early attempts at training DNNs often encountered problems with *vanishing or exploding gradients.*

Vanishing gradients

The gradient is said to vanish when the norm $||\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}^{(l)}}||$ becomes so small that the parameter updates done in Equation (3.30) are negligible. A key contributor to the vanishing gradient problem is saturating activation functions. For sigmoidal functions, like $\tanh(x)$ and $\text{sigmoid}(x)$, the derivative becomes almost 0 when $|x|$ becomes large, causing the gradients to vanish.

To solve the vanishing gradient problem arising from saturating activation functions, one can instead use *non-saturating* activation functions, like ReLU and LeakyReLU. Such activation functions allows DNN training with fewer vanishing gradients [132]. Figure 7 shows the activation functions $\text{sigmoid}(x)$ and $\text{LeakyReLU}(x)$, and their derivatives. The derivative of $\text{sigmoid}(x)$ is almost zero for $|x| > 5$, in contrast to $\frac{\mathrm{d}}{\mathrm{d}\,x}\,\text{LeakyReLU}(x)$, which is non-zero everywhere.

Finally, residual connections and ResNets [122] can also be used to further improve the gradient flow in the network.

[4]Adam is not particularly sensitive to hyperparameters, so the default values of $\mu = 10^{-3}$ and $\varepsilon = 10^{-8}$ work well in most cases.

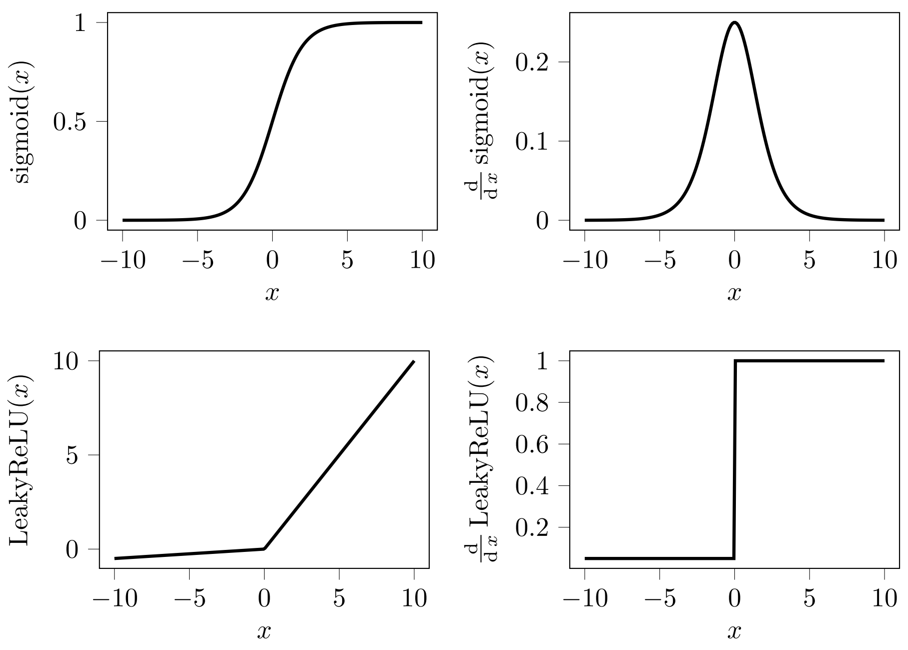


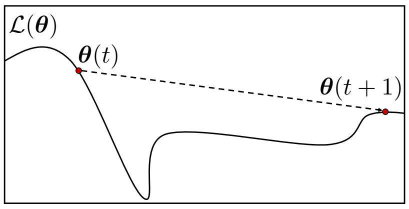
Figure 7: *Frequently used activation functions in DNNs.* $\frac{\mathrm{d}}{\mathrm{d}\,x}\,\mathrm{sigmoid}(x)$ *is close to* 0 *for* $|x| > 5$*, making it prone to the vanishing gradient problem. This is not the case for* $\mathrm{LeakyReLU}(x)$*, which has a non-zero derivative everywhere.*

Exploding gradients

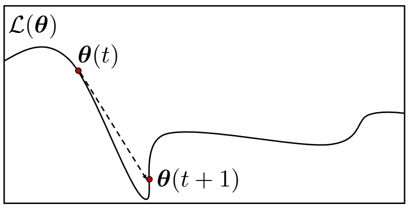
The loss landscapes of DNNs often include sharp "cliffs", originating from repeated multiplication of large weights [110]. At these points, the magnitude of the gradient is very large, which results in parameter updates that move the current parameter estimate far away, and possibly away from a desired local minimum – effectively undoing much of the optimization efforts done so far. The problem of exploding gradients can be alleviated with the gradient clipping heuristic. This essentially enforces a ceiling on the gradient norm, modifying the gradient in Equation (3.30) as

$$\mathbf{g}_{\text{clipped}} = \min\left\{1, \frac{T}{||\mathbf{g}||}\right\}\mathbf{g} \tag{3.39}$$

where T is the maximum allowed norm, and $\mathbf{g}$ is the gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}^{(l)}}$. By restricting the norm of the gradient, gradient clipping reduces the maximum step size, and thus avoids "overshooting" local minima close to cliffs in the loss landscape. See Figure 8 for an illustration of the exploding gradient problem, and the effect of gradient clipping.


$\mathcal{L}(\boldsymbol{\theta})$
$\boldsymbol{\theta}(t)$
$\boldsymbol{\theta}(t+1)$

(a) *Without gradient clipping.*


$\mathcal{L}(\boldsymbol{\theta})$
$\boldsymbol{\theta}(t)$
$\boldsymbol{\theta}(t+1)$

(b) *With gradient clipping.*

Figure 8: *Behavior of gradient descent near cliffs in the loss landscape, with and without gradient clipping. When the gradient is clipped, the updated parameters are much closer to the local minimum.*

3.3.6 Batch normalization

Batch normalization (BN) [133] is a technique to improve training of DNNs. According to Ioffe and Szegedy [133], the objective of BN is to reduce *internal covariate shift*. This is defined as a change in the distribution of outputs for a layer, originating from a change in network parameters. However, since it is computationally challenging to enforce the entire distribution to remain unchanged, BN instead fixes the mean and variance of the marginal distributions of the layer outputs. Let $\mathbf{y}_1^{(l)}, \dots, \mathbf{y}_n^{(l)}$ be the outputs of layer l for a batch consisting of n samples. Then, BN first computes the sample mean and variance as

$$\mathbf{m}^{(l)} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i^{(l)} \tag{3.40}$$

$$\mathbf{v}^{(l)} = \frac{1}{n-1}\,\mathrm{diag}\left(\sum_{i=1}^{n}(\mathbf{y}_i^{(l)} - \mathbf{m}^{(l)})(\mathbf{y}_i^{(l)} - \mathbf{m}^{(l)})^\top\right) \tag{3.41}$$

where diag($\mathbf{X}$) denotes the vector of diagonal elements in the square matrix $\mathbf{X}$. The layer output $\mathbf{y}_i^{(l)}$ is then normalized to zero mean and unit variance as

$$\mathbf{z}_i^{(l)} = (\mathbf{y}_i^{(l)} - \mathbf{m}^{(l)}) \oslash \mathbf{v}^{(l)} \tag{3.42}$$

which is used to compute the transformed output

$$\mathbf{y}_{i,\mathrm{new}}^{(l)} = \boldsymbol{\gamma}^{(l)} \odot \mathbf{z}_i^{(l)} + \boldsymbol{\beta}^{(l)} \tag{3.43}$$

where $\boldsymbol{\gamma}^{(l)}$ and $\boldsymbol{\beta}^{(l)}$ are trainable parameter vectors. These are included so that the model is able to learn output distributions with fixed mean and variance different from 0 and 1. The transformed outputs $\mathbf{y}_{1,\mathrm{new}}^{(l)}, \dots, \mathbf{y}_{n,\mathrm{new}}^{(l)}$ are then passed to the next layer in the model, instead of the original $\mathbf{y}_1^{(l)}, \dots, \mathbf{y}_n^{(l)}$.

Batch normalization in convolutional neural networks

The intermediate representations in CNNs are not vectors, but functions that can be regarded as three-dimensional arrays. The BN equations above thus have to be adapted to such arrays for BN to be applicable to CNNs. This is done by computing batch statistics for all spatial locations for all images in the batch, channel-wise.

Let $I_i^{(l)} \in \mathcal{I}_{C^{(l)},H^{(l)},W^{(l)}}$ be the output of layer l for sample i in the batch, and let

$$\mathbf{y}_{i,ab}^{(l)} = [I_i^{(l)}(1,a,b),\ldots,I_i^{(l)}(C^{(l)},a,b)]^\top \in \mathbb{R}^{C^{(l)}} \tag{3.44}$$

be the vector of channel values at position (a,b) in $I_i^{(l)}$. Equations (3.40) and (3.41) are then modified as

$$\mathbf{m}^{(l)} = \frac{1}{nW^{(l)}H^{(l)}} \sum_{i=1}^{n} \sum_{a=0}^{W^{(l)}-1} \sum_{b=0}^{H^{(l)}-1} \mathbf{y}_{i,ab}^{(l)} \tag{3.45}$$

$$\mathbf{v}^{(l)} = \frac{1}{n-1} \operatorname{diag}\left(\sum_{i=1}^{n} \sum_{a=0}^{W^{(l)}-1} \sum_{b=0}^{H^{(l)}-1} (\mathbf{y}_{i,ab}^{(l)} - \mathbf{m}^{(l)})(\mathbf{y}_{i,ab}^{(l)} - \mathbf{m}^{(l)})^\top \right) \tag{3.46}$$

resulting in the transformed image with elements

$$I_{i,\text{new}}^{(l)}(c,x,y) = \gamma_c^{(l)} \cdot \frac{I_i^{(l)}(c,x,y) - m_c^{(l)}}{v_c^{(l)}} + \beta_c^{(l)}. \tag{3.47}$$

Inference in deep neural networks with batch normalization

The transformation done by BN makes the layer output for one sample depend on the layer outputs of other samples, through the estimates of $\mathbf{m}^{(l)}$ and $\mathbf{v}^{(l)}$. Although this is acceptable during training, it is an unwanted property when the DNN is used for inference. In inference, we want a deterministic model, which produces outputs that are independent of other samples in the inference batch. Furthermore, in online inference where the batch size is 1, the normalization step would be undefined due to division by 0 in Equation (3.41).

To get a deterministic mapping independent of other samples, BN keeps running estimates of $\mathbf{m}^{(l)}$ and $\mathbf{v}^{(l)}$ during training. The estimates are an exponential moving average of the batch statistics. When the DNN is used for inference, the estimates are frozen, and used in place of $\mathbf{m}^{(l)}$ and $\mathbf{v}^{(l)}$ in Equation (3.42).

Why does batch normalization work?

In the original formulation of BN, Ioffe and Szegedy argue that BN reduces internal covariate shift, leading to faster convergence and improved accuracy [133]. However, later work by Santurkar *et al.* [134] found little correspondence between BN and internal covariate shift. This suggests that (i) BN does not reduce internal covariate shift; and (ii) there is another reason for BN's improvements to performance. Specifically, Santurkar *et al.* [134] discovered that the actual benefits of BN likely stems from a smoother loss landscape, resulting from the normalization. A smoother loss landscape makes gradients vary less in norm and direction, enabling higher learning rates and faster convergence.

4 Self-supervised learning

High-quality representations are crucial for machine learning systems, as they allow proper quantification of similarity and dissimilarity, preserving semantic information. Self-supervised learning (SSL) is a learning paradigm to learn representations of unlabeled data by synthesizing "labels" from the data itself, and training a "supervised" model with these labels. SSL has been widely successful in learning representations of complex data types, such as images [16–24, 47–49, 135–147] and text [30–35, 37, 38].

The key to unlocking the potential of SSL is to carefully design the synthesized labels, such that these accurately reflect the task we want to solve. This consideration is especially important if SSL is used as a component in a model, whose objective differs from learning general representations. In this thesis, we study SSL for single-view and multi-view deep clustering. For SSL to be successful in such models, the learned representations – and by extension, the generated labels – must reflect the true underlying group structure in the data. Furthermore, representations must be robust towards artifacts in the data that do not influence true cluster memberships. This chapter provides relevant background material on SSL, focusing on autoencoders and contrastive learning. Section 4.2 introduces autoencoders, which are frequently used in deep single-view clustering to preserve the similarity structure from inputs to representations. Section 4.3 presents contrastive learning and the SimCLR framework. In the context of deep multi-view clustering, contrastive learning can be used to align view-specific representations, making it relevant for our work in Papers II and III.

4.1 Pretext and downstream tasks

The task induced by the labels generated from data is called the *pretext task*. Predicting random rotations [147], predicting relative positions of patches [139], and discriminating between augmented images [22, 23, 138], are all examples of pretext tasks used to train SSL models on images.

A trained SSL model can be used to generate representations, which in turn can be used to solve a *downstream task*. The downstream task refers to the task we are actually interested in solving, but for which there exist little or no labeled data. This can be for instance classification, regression, retrieval, *etc*.

The key to designing good SSL models is to make the pretext task as similar as possible to the downstream task. This ensures that the representations obtained by training the model on the pretext task, successfully transfer to the downstream task. If the two tasks are not sufficiently similar, the pretext task can end up either preserving information irrelevant to the downstream task, discarding information relevant to downstream task, or both.

4.2 Autoencoders

Autoencoders (AEs) [148–151] are models that learn to encode input samples, requiring that the original inputs can be reproduced from the encoded representation. The AE's pretext task is thus *reconstruction of inputs based on learned representations*. AEs consist of two DNNs: an encoder network, $g_{\boldsymbol{\theta}}$, and a decoder network $h_{\boldsymbol{\phi}}$, where $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ denote the trainable parameters of g and h, respectively. The encoder network produces a representation

$$\mathbf{z} = g_{\boldsymbol{\theta}}(\mathbf{x}) \tag{4.1}$$

and the decoder network re-creates the input from the representation

$$\hat{\mathbf{x}} = h_{\boldsymbol{\phi}}(\mathbf{z}). \tag{4.2}$$

The reconstruction task can be trivially solved by setting both $g_{\boldsymbol{\theta}}$ and $h_{\boldsymbol{\phi}}$ to identity mappings, resulting in $\mathbf{x} = \mathbf{z} = \hat{\mathbf{x}}$. To avoid this, AEs often enforce a *bottleneck constraint*, requiring that the dimensionality of $\mathbf{z}$ is smaller than the dimensionality of $\mathbf{x}$. This forces $g_{\boldsymbol{\theta}}$ to learn a compressed version of the input, preserving relevant information and discarding noise.

To perform the pretext task, and reconstruct input samples as good as possible, AEs are trained using the mean squared error (MSE) loss

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^{n} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 \tag{4.3}$$

where $\mathbf{x}_1, \dots, \mathbf{x}_n$ are the training samples in the current batch, and $|| \cdot ||$ denotes the Euclidean norm.

4.2.1 Improvements to the standard autoencoder model.

Several modifications to the standard AE model have been proposed in order to enhance the quality of AE-based representations. Denoising AEs [152, 153] add a noise corruption step before the encoder. The pretext task is then changed to denoising, instead of reconstruction – leading to representations that are more robust to noise in the input. Variational AEs [28, 29, 154–157] are probabilistic

AE-based models that learn distributions of latent representations of the inputs. Lastly, masking-based AEs [158–160] are a particular, recent type of denoising AEs designed for images. Masking AEs remove a large proportion of patches in the input image, and then require that the input image is reconstructed from the remaining patches.

4.2.2 Objective function mismatch

Different types of tasks require different types of information to be carried over from inputs to representations. Reconstructing images, for instance, requires the representations to contain fine-grained information about color, texture, semantic content, *etc.* Classification, on the other hand, only requires information about whether an image contains a specific object or not. When a model is trained to perform one task and then evaluated on another, the difference in information required for these tasks can lead to objective function mismatch (OFM) [161]. OFM refers to a mismatch between objectives, where optimizing for one objective leads to reduced performance for another objective. This can also occur in models that simultaneously optimize two competing objectives, if the model learns to de-emphasize one objective in order to optimize the other objective.

AEs are particularly sensitive to OFM, since reconstruction often differs from common downstream tasks, such as classification, clustering, semantic segmentation, *etc.* In fact, OFM was first observed by Metz *et al.* [161] when training a variational AE, while simultaneously measuring few-shot classification accuracy using representations from the variational AE. In this case, after an initial increase, the accuracy started to decrease after a certain number of training iterations. OFM has also been observed in AE-based clustering models [162, 163], illustrating that mismatch between the reconstruction and clustering objectives can lead to reduced clustering performance.

4.3 Contrastive learning

Contrastive learning is a form of SSL where the pretext task is to discriminate between different instances in the training dataset. Most contrastive learning methods use random augmentations (transformations), in order to get multiple representations of the same input [17, 19, 21–23, 47, 48]. The objective in contrastive learning is then to “assign” representations originating from the same input to each other, while simultaneously repelling representations originating from different inputs.

This section focuses on the SimCLR framework, since it represents a base model for self-supervised contrastive learning, to which modifications and improvements can be made. The contrastive loss used in SimCLR is the basis for the generalized multi-view contrastive loss developed in Paper II, and further studied in Paper III.

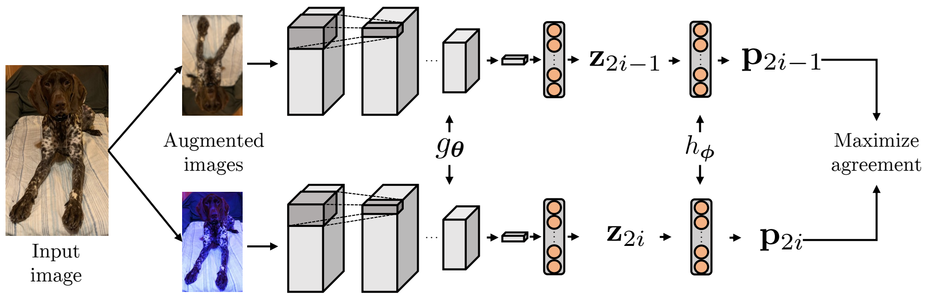


Figure 9: *Overview of the SimCLR architecture.*

4.3.1 SimCLR: a simple framework for contrastive learning of visual representations

SimCLR [23] is a simple contrastive learning model for images. The model consists of a feature extractor (typically a CNN), $g_{\boldsymbol{\theta}}$, and a projection head $h_{\boldsymbol{\phi}}$ (2-layer MLP), where $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are vectors of parameters. The feature extractor receives $\tilde{I}_{2i-1}$ and $\tilde{I}_{2i}$, which are augmented versions of the input image I_i, and produces representations $\mathbf{z}_{2i-1} = g_{\boldsymbol{\theta}}(\tilde{I}_{2i-1})$ and $\mathbf{z}_{2i} = g_{\boldsymbol{\theta}}(\tilde{I}_{2i})$. The projection head then receives the representations, and computes projections $\mathbf{p}_{2i-1} = h_{\boldsymbol{\phi}}(\mathbf{z}_{2i-1})$ and $\mathbf{p}_{2i} = h_{\boldsymbol{\phi}}(\mathbf{z}_{2i})$. Figure 9 shows an overview of the SimCLR architecture.

SimCLR uses the normalized temperature-scaled cross entropy (NT-Xent) loss to maximize the similarity between projections originating from the same image, and simultaneously repel projections from other images

$$\mathcal{L}_{\text{NT-Xent}} = \frac{1}{2n} \sum_{k=1}^{n} (\ell(2k-1, 2k) + \ell(2k, 2k-1)) \tag{4.4}$$

with

$$\ell(i,j) = -\log \frac{\exp(\text{sim}(\mathbf{p}_i, \mathbf{p}_j)/\tau)}{\sum_{k=1}^{2n} \mathbb{1}_{\{k \neq i\}} \exp(\text{sim}(\mathbf{p}_i, \mathbf{p}_k)/\tau)} \tag{4.5}$$

where sim denotes the cosine similarity

$$\text{sim}(\mathbf{p}_i, \mathbf{p}_j) = \frac{\mathbf{p}_i^\top \mathbf{p}_j}{||\mathbf{p}_i|| \cdot ||\mathbf{p}_j||}. \tag{4.6}$$

The introduction of the projection head, $h_{\boldsymbol{\phi}}$, means that the contrastive loss is applied to *projections* of the representations, instead of the representations themselves. Chen *et al.* [23] conjecture that this is beneficial since downstream performance might depend on some of the information discarded when learning augmentation invariance. Including a projection head enables this information to be retained in $\mathbf{z}$, and subsequently removed by $h_{\boldsymbol{\phi}}$ when computing the projections.

An interesting property of the NT-Xent loss is that is it based on an L_2 normalized similarity measure. The similarity between two projections is only based on angular information, and is invariant to the norm of the projections. We can therefore interpret the projection space as hyperspherical, instead of Euclidean.

4.3.2 Why does contrastive learning lead to good image representations?

Contrastive learning has been widely successful for learning image representations [164]. The key characteristic behind this success is that important semantic information is often augmentation invariant. Hence, learning representations that are approximately augmentation invariant, corresponds to retaining the important information, and discarding the rest. This results in representations that are informative for downstream tasks that depend on semantic information, such as classification and segmentation.

The SimCLR loss can also be interpreted in terms of the condition for good representations in Equation (2.1). Suppose that the similarity function in the input space is defined as

$$s_I(\tilde{I}_i, \tilde{I}_j) = \begin{cases} 1, & \tilde{I}_i \text{ and } \tilde{I}_j \text{ are augmentations of the same image} \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

and that the similarity function between representations is

$$s_{\mathbf{z}}(\mathbf{z}_1, \mathbf{z}_2) = \mathrm{sim}(h_{\boldsymbol{\phi}}(\mathbf{z}_i), h_{\boldsymbol{\phi}}(\mathbf{z}_j)). \tag{4.8}$$

Then, optimizing the loss in Equation (4.4) would result in

$$s_I(\tilde{I}_1, \tilde{I}_2) > s_I(\tilde{I}_1, \tilde{I}_3) \Leftrightarrow s_{\mathbf{z}}(\mathbf{z}_1, \mathbf{z}_2) > s_{\mathbf{z}}(\mathbf{z}_1, \mathbf{z}_3) \tag{4.9}$$

for arbitrary augmented images $\tilde{I}_1, \tilde{I}_2, \tilde{I}_3$. SimCLR therefore produces good representations w.r.t. these similarity functions.

One interesting observation about the projection similarity function used by SimCLR is that it is based on a cosine similarity, and thus implicitly maps projections on the hypersphere when computing similarities. In fact, many contrastive learning methods embed representations or projections on the hypersphere – either implicitly or explicitly. Wang and Isola [146] show that, when the representations or projections are on the hypersphere, the contrastive loss optimizes a tradeoff between *alignment* and *uniformity*: representations originating from the same image are aligned (made similar), and representations are distributed uniformly on the hypersphere. In addition, if classes are compact clusters on the hypersphere, these will be linearly separable from each other – possibly leading to improved downstream classification performance.

5 Clustering

Clustering is a long-standing problem in machine learning. The goal of clustering is to discover unknown group structure in the input data, without any form of label supervision. Since the introduction of cornerstone methods, such as k-means [165] and hierarchical clustering [166], clustering has been immensely useful in several applications [167–172].

The success of deep learning was initially limited to supervised applications, such as classification [15, 121, 122, 173]. Following this, translating the success of deep learning to unsupervised learning, and to clustering in particular, has become an active area of research. This has led to the development of the *deep clustering* subfield, where DNNs are adapted to clustering.

DNNs are particularly suitable for clustering, since they can provide good representations from a wide range of complex input signals, such as images, time series, graphs, *etc*. The unsupervised nature of clustering means that it can not rely on external information from labels, making it more dependent on geometrical properties of the data. Successful clustering therefore requires meaningful similarities to be encoded in the geometry of the representations – which is a key characteristic of good representations.

Part of the main objective of this thesis is to improve and better understand representation learning for deep clustering. To this end, this chapter presents key methods and challenges in deep clustering, focusing on those relevant for our work in Papers I, II and III. In Section 5.1 we consider deep clustering methods for data originating from a single view, or modality. This is the standard clustering setting which has received most attention in recent years. Section 5.2 presents methods for deep multi-view clustering, where it is assumed that the input data originates from multiple views or sources. This is a more challenging setting compared to single-view clustering, with unique considerations and challenges.

5.1 Deep single-view clustering

Models for deep single-view clustering typically consist of a DNN that computes representations from input data, and a clustering module that predicts cluster memberships based on the representations. To improve the quality of representations produced by the DNN, many models for deep clustering rely on AEs [39–41, 174–182]. However, we find that AEs might not be optimal in ensuring representation quality in deep clustering, due to OFM between the clustering and reconstruction objectives. This issue is discussed further in Section 5.1.4.

Before discussing representation quality and OFM, we review some recent methods for deep clustering. The first two methods are based on AEs, and illustrate how AEs are used to improve similarity preservation and representation quality in deep clustering. The last method, deep divergence-based clustering (DDC), is used in Paper I to analyze the effects of the proposed unsupervised companion objectives (UCOs). In addition, the DDC clustering module is adapted to deep multi-view clustering in Papers II and III.

5.1.1 Deep embedded clustering

Deep embedded clustering (DEC) [39] is one of the first successful applications of DNNs to clustering. In DEC the DNN used to compute representations is pre-trained as the encoder in a denoising AE [152, 153]. After training the AE, DEC discards the decoder, and initializes k prototypes, $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k$ in the representation space, using k-means [165].

Assuming the DNN produces representations $\mathbf{z}_1, \dots, \mathbf{z}_n$ for the inputs in a given batch, DEC minimizes the loss

$$\mathcal{L}_{\text{DEC}} = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{5.1}$$

w.r.t. the prototypes and network parameters. p_{ij} and q_{ij} specify discrete distributions on $\mathbb{N}_{\leq k}$, defined as

$$q_{ij} = \frac{(1 + ||\mathbf{z}_i - \boldsymbol{\mu}_j||^2/\alpha)^{-(\alpha+1)/2}}{\sum_{j'=1}^{k} (1 + ||\mathbf{z}_i - \boldsymbol{\mu}_{j'}||^2/\alpha)^{-(\alpha+1)/2}} \tag{5.2}$$

and

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'=1}^{k} q_{ij'}^2 / f_{j'}} \tag{5.3}$$

with $f_j = \sum_{i=1}^{n} q_{ij}$. α is a hyperparameter set to 1 for all experiments.

$\mathcal{L}_{\text{DEC}}$ minimizes the Kullback-Leibler divergence (KLD) between the distributions specified by q_{ij} and p_{ij}. q_{ij} can be seen as the soft assignment of sample i to cluster j. Since p_{ij} is computed from q_{ij}^2, minimizing $\mathcal{L}_{\text{DEC}}$ corresponds to learning from high-confidence cluster predictions. This means that samples close to a prototype will have a larger influence on the learning dynamics, compared to a sample which is further away from the prototypes.

5.1.2 Improved deep embedded clustering

Guo *et al.* [41] argue that fine-tuning with DEC's clustering loss can "corrupt" the representation space, such that it no longer accurately represents similarities from the input space. Improved deep embedded clustering (IDEC) addresses this issue by keeping the decoder and the reconstruction loss during fine-tuning. The purpose of the reconstruction loss is to regularize fine-tuning, preserving similarity structure and retaining the quality of representations. The fine-tuning loss is a weighted combination of clustering and reconstruction

$$\mathcal{L}_{\text{IDEC}} = \frac{1}{n} \sum_{i=1}^{n} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 + \gamma \mathcal{L}_{\text{DEC}} \tag{5.4}$$

$\hat{\mathbf{x}}_i$ is the reconstruction of $\mathbf{x}_i$, and γ is a hyperparameter set to 0.1 for all experiments.

Combining the reconstruction and clustering losses makes IDEC prone to OFM (Section 4.2.2), as demonstrated by Mrabah *et al.* [162, 163]. This issue is discussed further in Section 5.1.4.

5.1.3 Deep divergence-based clustering

Deep divergence-based clustering (DDC) [42] is a deep clustering method based on information theory and divergences between probability density functions. DDC was originally designed for image clustering, and is therefore composed of a CNN mapping input images $I_1, \dots, I_n$ to representations $\mathbf{z}_1, \dots, \mathbf{z}_n$, and a single-layer perceptron transforming representations to cluster membership vectors $\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_n$.

In DDC clusters are modelled as probability density functions (PDFs) in the representation space. Interpreting clusters as PDFs allows clustering to be performed using divergences, which are dissimilarity measures between densities. The key idea is to maximize the divergence between clusters (PDFs), which will encourage clusters to be compact and separable in the representation space. Since the cluster PDFs are unknown, they have to be estimated from data. This kan be done using a Gaussian kernel density estimate (KDE)

$$p_j(\mathbf{z}) = \frac{1}{\sum_{i=1}^{n} \alpha_{ij}} \sum_{i=1}^{n} \alpha_{ij} K_\sigma(||\mathbf{z} - \mathbf{z}_i||) \tag{5.5}$$

where K_σ is the Gaussian kernel with kernel width σ[1]

$$K_\sigma(x) = \frac{\exp\left(-\frac{x^2}{2\sigma^2}\right)}{\sqrt{2\pi}\sigma} \tag{5.6}$$

and α_{ij} is a cluster membership indicator

$$\alpha_{ij} = \begin{cases} 1, & \mathbf{z}_i \text{ is assigned to cluster } j \\ 0, & \text{otherwise} \end{cases}. \tag{5.7}$$

[1]σ is treated as a hyperparameter, which is set to 15% of the median of pairwise distances within a batch.

DDC uses the Caucy-Schwarz divergence (CSD) to quantify dissimilarity between clusters, since this particular divergence has a closed-form solution for KDEs. The CSD between PDFs, $p_1, \dots, p_k : \mathbb{R}^l \supseteq Z \to \mathbb{R}$ is defined as [104]

$$D_{\mathrm{CS}}(p_1, \dots, p_k) = -\log\left(\binom{k}{2}^{-1} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} d_{\mathrm{CS}}(p_i, p_j)\right) \tag{5.8}$$

where

$$d_{\mathrm{CS}}(p_i, p_j) = \frac{\int_Z p_i(\mathbf{z}) p_j(\mathbf{z}) \mathrm{d}\mathbf{z}}{\sqrt{\int_Z p_i(\mathbf{z})^2 \mathrm{d}\mathbf{z} \int_Z p_j(\mathbf{z})^2 \mathrm{d}\mathbf{z}}}. \tag{5.9}$$

Assuming $p_1, \dots p_k$ are Gaussian KDEs (Equation (5.5)), $d_{\mathrm{CS}}(p_i, p_j)$ can be written as [104]

$$d_{\mathrm{CS}}(p_i, p_j) = \frac{\sum\limits_{a=1}^{n} \sum\limits_{b=1}^{n} \alpha_{ai} \alpha_{bj} \kappa_{ab}}{\sqrt{\left(\sum\limits_{a=1}^{n} \sum\limits_{b=1}^{n} \alpha_{ai} \alpha_{bi} \kappa_{ab}\right)\left(\sum\limits_{a=1}^{n} \sum\limits_{b=1}^{n} \alpha_{aj} \alpha_{bj} \kappa_{ab}\right)}} \tag{5.10}$$

where

$$\kappa_{ab} = K_{\sqrt{2}\sigma}(||\mathbf{z}_a - \mathbf{z}_b||). \tag{5.11}$$

This is an important result, since it means that the integrals in Equation (5.9) are replaced by sums over kernel evaluations at the representations $\mathbf{z}_1, \dots, \mathbf{z}_n$. Maximizing $D_{\mathrm{CS}}(p_1, \dots, p_k)$ can now be done by minimizing the argument to the logarithm, resulting in the loss

$$\mathcal{L}_{\mathrm{DDC},1} = \binom{k}{2}^{-1} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{\sum\limits_{a=1}^{n} \sum\limits_{b=1}^{n} \alpha_{ai} \alpha_{bj} \kappa_{ab}}{\sqrt{\left(\sum\limits_{a=1}^{n} \sum\limits_{b=1}^{n} \alpha_{ai} \alpha_{bi} \kappa_{ab}\right)\left(\sum\limits_{a=1}^{n} \sum\limits_{b=1}^{n} \alpha_{aj} \alpha_{bj} \kappa_{ab}\right)}} \tag{5.12}$$

where the binary constraint on α (Equation (5.7)) is relaxed to allow $\alpha_{ai} \in [0, 1]$ with $\sum_{i=1}^{k} \alpha_{ai} = 1$ for all $a \in \mathbb{N}_{\leq n}$. This is done to make $\mathcal{L}_{\mathrm{DDC},1}$ differentiable.

In order to encourage confident cluster assignments, and to prevent the trivial solution of all samples being assigned to one cluster, DDC includes two additional loss terms

$$\mathcal{L}_{\mathrm{DDC},2} = \sum_{a=1}^{n} \sum_{b=1}^{n} \boldsymbol{\alpha}_a^\top \boldsymbol{\alpha}_b \tag{5.13}$$

and

$$\mathcal{L}_{\mathrm{DDC},3} = \binom{k}{2}^{-1} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{\sum\limits_{a=1}^{n} \sum\limits_{b=1}^{n} m_{ai} m_{bj} \kappa_{ab}}{\sqrt{\left(\sum\limits_{a=1}^{n} \sum\limits_{b=1}^{n} m_{ai} m_{bi} \kappa_{ab}\right)\left(\sum\limits_{a=1}^{n} \sum\limits_{b=1}^{n} m_{aj} m_{bj} \kappa_{ab}\right)}} \tag{5.14}$$

with

$$m_{ai} = \exp\left(||\boldsymbol{\alpha}_a - \mathbf{e}_i||^2\right) \tag{5.15}$$

where $\mathbf{e}_i$ is the i-th standard basis vector of $\mathbb{R}^k$. The loss minimized by DDC is the sum of the three terms

$$\mathcal{L}_{\text{DDC}} = \mathcal{L}_{\text{DDC},1} + \mathcal{L}_{\text{DDC},2} + \mathcal{L}_{\text{DDC},3}. \tag{5.16}$$

In summary, $\mathcal{L}_{\text{DDC}}$ encourages clusters in the representation space to be separated and compact, with confident cluster assignments. The latter is expressed geometrically as having the cluster assignment vectors $\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_n$ lie close to the corners of the probability simplex in $\mathbb{R}^k$.

5.1.4 Preserving similarity structure

The representational power of DNNs makes the optimization of deep clustering models susceptible to trivial solutions. Characteristic for these trivial solutions is that the clustering loss is minimized, but the representations are no longer representative of the true similarities in the input space. The representations thus no longer satisfy the conditions in Equations (2.1) and (2.2), impying that they are not good representations. This causes the model to output non-informative clusters, based on an incorrect similarity structure.

As argued by Guo *et al.* [41], deep clustering models often require additional objectives to preserve the similarity structure from inputs to representations. This has commonly been achieved by combining the clustering model with an AE, where the clustering is performed in the AE's code space [39–41]. However, the differences between the clustering and reconstruction objectives makes the model susceptible to OFM (Section 4.2.2). This happens because the information required for reconstruction – such as texture, color, background, *etc.*– differs from the information required for clustering.

Paper I addresses the similarity preservation issue in deep clustering by introducing the unsupervised companion objectives (UCOs). These are novel auxiliary objectives for deep clustering that encourage preservation of similarity structure through the DNN, with reduced levels of OFM, resulting in improved clustering performance.

5.2 Deep multi-view clustering

The objective in multi-view clustering (MVC) is to discover groups in data *originating from multiple views*, or multiple modalities. Examples of multi-view data include videos, consisting of a sequence of images and an audio signal, and captioned images, consisting of an image and a caption. Although additional views makes the clustering task more challenging, they also give additional information about the object they represent. This allows for better performance compared to independently applying a single-view clustering algorithm to each view.

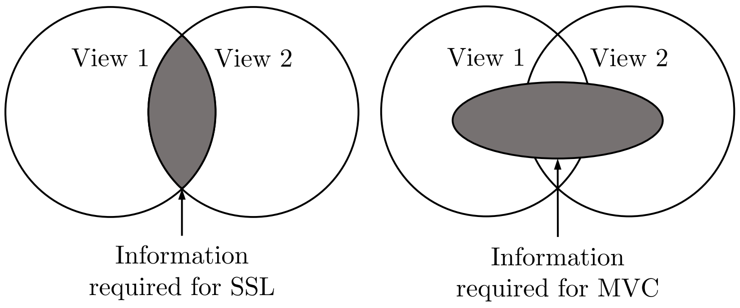


Figure 10: *Information required for instance discrimination-based SSL vs. information required for MVC. In SSL it is assumed that the relevant information lies in the intersection between views. However, in MVC, the relevant information can exist in either view alone, or in the intersection between them.*

Similar to in single-view clustering, the recent success of deep learning has inspired the development of DNN-based methods for MVC, resulting in the *deep multi-view clustering* subfield.

Leveraging information from multiple views is similar to what is done in instance discrimination-based SSL. These models discriminate between augmented versions (views) of the data, and effectively learn to be augmentation invariant (Section 4.3.2). Crucially, these SSL models assume that the relevant information for the downstream task is present in *all* views (augmentations). Learning augmentation invariant representations thus corresponds to discarding irrelevant information, while preserving relevant information. However, this assumption is not necessarily met by multi-view data. Naïvely removing information not present in all views can therefore negatively influence the performance of general multi-view clustering models. The difference in information required for SSL and MVC is illustrated in Figure 10.

Deep MVC models typically follow a design pattern with view-specific encoders, representation fusion, and a clustering module. Let $\mathbf{x}_i^{(v)}$ denote the observation of instance $i \in \mathbb{N}_{\leq n}$, through view $v \in \mathbb{N}_{\leq V}$. The view-specific encoder $f_{\boldsymbol{\theta}_e^{(v)}}$ then produces the view-specific representation

$$\mathbf{z}_i^{(v)} = f_{\boldsymbol{\theta}_e^{(v)}}(\mathbf{x}_i^{(v)}). \tag{5.17}$$

View-specific representations are then fused to a shared representation as

$$\mathbf{z}_i = \text{fusion}(\mathbf{z}_i^{(1)}, \dots, \mathbf{z}_i^{(V)}). \tag{5.18}$$

The clustering module then produces the cluster membership vectors $\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_n$ from either the view-specific representations or from the fused representations.

Much of the recent effort in deep MVC has been concentrated towards developing SSL-like tasks to learn better representations for the clustering module. Representation quality is particularly important in deep MVC since (i) there is no label supervision to guide the clustering module; and (ii) information required for the clustering task might be spread across different views. These considerations highlight both the importance and the challenge of developing SSL tasks to improve deep MVC models.

5.2.1 End-to-end adversarial-attention network for multi-modal clustering

When it was published in 2020, the end-to-end adversarial-attention network for multi-modal clustering (EAMC) [45] was the new state-of-the-art method for deep MVC. EAMC learns view-invariant representations by aligning the *distributions* of view-specific representations, through an adversarial learning procedure similar to generative adversarial networks [183]. This adversarial alignment serves as a pretext task for the view-specific encoders, potentially leading to better representations for the clustering module.

EAMC's distribution alignment procedure inspired the work in Paper II, where we identify pitfalls of distribution alignment in deep MVC. This model therefore serves as an example of the problems that can occur when SSL is applied to deep MVC, without properly understanding the impact on representation quality and cluster separability.

EAMC includes the following loss, aligning all subsequent views to the first view

$$\mathcal{L}_{\text{EAMC, Adv}} = \frac{1}{n} \sum_{v=2}^{V} \sum_{i=1}^{n} \left(\log h_{\boldsymbol{\theta}_d^{(v)}}(\mathbf{z}_i^{(1)}) + \log(1 - h_{\boldsymbol{\theta}_d^{(v)}}(\mathbf{z}_i^{(v)})) \right) \tag{5.19}$$

where $h_{\boldsymbol{\theta}_d^{(v)}}$ denotes a view-specific discriminator network. The task of $h_{\boldsymbol{\theta}_d^{(v)}}$ is to predict whether the input comes from view v or from view 1.

To further strengthen the alignment procedure, EAMC includes a kernel alignment loss

$$\mathcal{L}_{\text{EAMC, Ker}} = ||\mathbf{K}^{(\text{fused})} - \mathbf{K}^{(\text{avg})}||_F^2 \tag{5.20}$$

where $||\cdot||_F^2$ denotes the squared Frobenius norm, and $\mathbf{K}^{(\text{fused})}$ is a kernel matrix for the fused representations

$$\mathbf{K}_{ij}^{(\text{fused})} = \exp\left(-\frac{1}{2\sigma^2} ||\mathbf{z}_i - \mathbf{z}_j||^2 \right) \tag{5.21}$$

and $\mathbf{K}^{(\text{avg})}$ is the average of view-specific kernel matrices

$$\mathbf{K}^{(\text{avg})} = \sum_{v=1}^{V} w_v \mathbf{K}^{(v)} \tag{5.22}$$

where $w_1, \dots, w_V$ are the fusion weights, and $\mathbf{K}^{(v)}$ is the view-specific kernel matrix with elements

$$\mathbf{K}_{ij}^{(v)} = \exp\left(-\frac{1}{2\sigma^2} ||\mathbf{z}_i^{(v)} - \mathbf{z}_j^{(v)}||^2 \right). \tag{5.23}$$

The bandwidth parameter σ is set to 15% of median pairwise distances between the respective representations, following DDC [42].

Fusion in EAMC is implemented as a weighted average of view-specific representations

$$\mathbf{z}_i = \sum_{v=1}^{V} w_v \mathbf{z}_i^{(v)}. \tag{5.24}$$

The fusion weights are computed by an attention network $g_{\boldsymbol{\theta}_f}$, allowing the weights to be adapted by the learning process

$$(w_1, \dots, w_V) = g_{\boldsymbol{\theta}_f}(\mathbf{z}_1^{(1)}, \dots, \mathbf{z}_n^{(1)}, \dots, \mathbf{z}_1^{(V)}, \dots, \mathbf{z}_n^{(V)}) \tag{5.25}$$

Learning the fusion weights in this way enables adaptive view-prioritization, where the model can learn to focus more on informative views, and less on un-informative views.

Finally, EAMC uses the clustering module from DDC [42], and its loss, $\mathcal{L}_{\text{DDC}}$, to cluster the fused representations. This allows the view-specific encoders to be trained both to perform the pretext task of distribution alignment, and to produce compact and separable clusters in the space of fused representations.

Optimizing EAMC is done by alternating between the following steps.

1. Minimize the alignment and clustering losses over the clustering module, attention network and view-specific encoder parameters

$$\min_{\boldsymbol{\theta}_c, \boldsymbol{\theta}_f, \boldsymbol{\theta}_e^{(1)}, \dots, \boldsymbol{\theta}_e^{(V)}} (\mathcal{L}_{\text{DDC}} + \mathcal{L}_{\text{EAMC, Ker}} + \gamma \mathcal{L}_{\text{EAMC, Adv}}) \tag{5.26}$$

   Where $\boldsymbol{\theta}_c$ denotes the parameters of the clustering module, and γ is a hyperparameter.

2. Maximize the alignment loss over the discriminator parameters

$$\max_{\boldsymbol{\theta}_d^{(2)}, \dots, \boldsymbol{\theta}_d^{(V)}} \mathcal{L}_{\text{EAMC, Adv}} \tag{5.27}$$

5.2.2 Pitfalls of distribution alignment

The success of EAMC shows that aligning view-specific representations can serve as a useful pretext task to learn better representations in deep MVC. However, in Paper II we identify several drawbacks related to representation alignment. First, alignment only preserves the information that is shared across all views. If the information required to successfully cluster the data is contained in only a subset of the views, this information will be discarded by the alignment procedure. This leads to reduced clustering performance. Further, we find that alignment hinders view-prioritization. Successful alignment results in views which are inseparable in the space of view-specific representations, making it impossible to discriminate, and consequently, prioritize between them.

These pitfalls can be seen as consequences of OFM, where the auxiliary alignment objective has a negative impact on the clustering objective. The exact amount of mismatch depends on how information relevant to the clustering module is distributed across views. For instance, in the extreme case where all relevant information is contained in one view and the other views contain random noise, the mismatch will cause a severe reduction in cluster separability in the representation space.

In addition to identifying the above pitfalls, Paper II proposes new methods to bypass the pitfalls of distribution alignment – using either no alignment, or an adaptive alignment procedure. Paper III continues along these lines, generalizing the analysis to other forms of alignment, and making new discoveries on the influence of the number of views on alignment and clustering performance.

6
Few-shot learning

The objective of few-shot learning (FSL) is to classify samples from a set of novel classes with few labeled instances, based on a labeled dataset containing samples from other classes [184]. The number of labeled instances in novel classes is typically very small, *e.g.* 1 or 5. Similar to fully unsupervised learning, the limited availability of labels means that FSL is sensitive to representation quality. Successful classification requires that representations of the few labeled instances reflect the class from which they originate, without additional noise or distracting features.

In this thesis, we study FSL for image data – a subfield that has become increasingly popular in recent years [50–75]. In particular, our objective is to address the hubness problem in transductive FSL [55, 89], and to better understand the effects of representation norms and normalization.

This chapter first presents relevant background material on FSL. Since the focus of our work is on representations and embeddings, we then summarize recent methods to embed representations for FSL. Finally, Section 6.3 introduces the hubness problem, and explains why it can have a significant impact on FSL performance.

6.1 Few-shot learning preliminaries

Classes. The FSL setting includes two datasets. The first dataset consists of samples from the *base* classes, which are assumed to contain a large number of labeled observations. The second dataset consists of samples from the *novel* classes. These are the classes we are interested in classifying, where only few labeled samples exist.

Training. FSL models typically consist of a DNN-based feature extractor that computes representations for the input samples, and a classifier that classifies these representations. The DNN feature extractor is usually trained only on the base classes, in order to avoid overfitting on the few labeled examples from the novel classes. The classifier, on the other hand, is usually trained on both base and novel classes, or on the novel classes when they are presented as episodes (see below). Further specifics on how these components are trained vary from model to model.

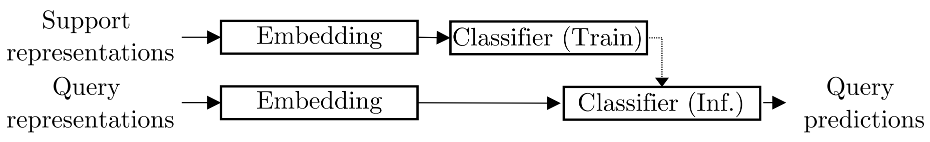
Evaluation. FSL methods are evaluated in *episodes*. In each episode, k classes are sampled from the novel classes. Then n_s *labeled support* samples, and n_q *unlabeled query* samples, are sampled from each class. The support and query samples now constitute a k-way n_s-shot task, where the objective is to predict class membership for the query samples, based on the labeled support samples. FSL models are typically evaluated with $n_s = 1$ and $n_s = 5$. The episodic evaluation is repeated for a large number of episodes, sampling a new set of novel classes at each episode. The resulting episode accuracies are used to compute mean and confidence intervals for the expected performance of the FSL model.

Transductive vs. inductive few-shot learning. FSL models are either *inductive* or *transductive*. In inductive FSL, the model should, based on the given support samples, produce a general classification rule, which is applied to the query samples. Conversely, in transductive FSL, the model is only required to produce predictions for the given query samples, and not for the whole population of possible query samples. Transductive FSL thus allows the model to use both support and query samples simultaneously to solve the classification task. By leveraging the additional information from the unlabeled query samples, transductive FSL models have largely outperformed inductive models in recent years [50, 57, 63, 64, 68, 72–74].

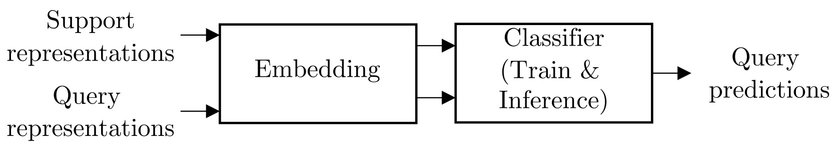
Transductive FSL models often train the DNN feature extractor using the supervised cross-entropy loss on the base dataset. This is in contrast to inductive models, which tend to use more complicated episodic, meta-learning approaches to train the feature extractor. Recent work by Laenen and Bertinetto [185] found that the episodic training of FSL models was unnecessary, and in some cases, led to inefficient use of training samples from the base dataset. These findings thus support training the feature extractor without episodic meta-learning, as is commonly done in recent transductive FSL models.

6.2 Embedding representations for few-shot learning

Recent works in FSL have found it beneficial to introduce an additional embedding step after the feature extractor [55, 58, 71, 72, 75]. Such embeddings have been particularly effective in transductive FSL, since these methods can embed support and query samples simultaneously, using information from both the labeled support samples, and the unlabeled query samples (see Figure 11).



(a) *Inductive*



(b) *Transductive*

Figure 11: *Embedding and classification for a single episode of inductive vs. transductive FSL. The transductive approach can exploit information from both support and query samples when embedding and classifying the samples. Representations are assumed to come from the DNN feature extractor, trained on the base dataset.*

6.2.1 L_2 and centered L_2 normalization

Among the simplest forms of embeddings are the L_2 and centered L_2 normalizations [71]. For a given support or query representation $\mathbf{z}_i$ with $i \in \mathbb{N}_{\leq k(n_s+n_q)}$, L_2 normalization produces a hyperspherical embedding

$$L_2(\mathbf{z}_i) = \frac{\mathbf{z}_i}{||\mathbf{z}_i||}. \tag{6.1}$$

Centered L_2 normalization first centers the representations, and then embeds them on the hypersphere

$$CL_2(\mathbf{z}_i) = L_2(\mathbf{z}_i - \mathbf{c}) \tag{6.2}$$

where

$$\mathbf{c} = \frac{1}{k(n_s + n_q)} \sum_{i=1}^{k(n_s+n_q)} \mathbf{z}_i \tag{6.3}$$

is the mean of the support and query samples.

Although these embeddings appear simple, they result in significant increases in FSL performance [71].

6.2.2 Z-score normalization

Z-score normalization (ZN) [55] is an embedding technique designed to alleviate the hubness problem in FSL (see Section 6.3). ZN transforms each representation to have zero mean and unit standard deviation *along the feature dimension.* For a support or query representation $\mathbf{z}_i = [z_{i1}, \dots, z_{il}]^\top \in \mathbb{R}^l$ as

$$ZN(\mathbf{z}_i) = \frac{\mathbf{z}_i - \mu_i \mathbf{1}}{\sigma_i} \tag{6.4}$$

where $\mathbf{1}$ is the l-dimensional vector with all elements equal to one, and

$$\mu_i = \frac{1}{l} \sum_{j=1}^{l} z_{ij}, \tag{6.5}$$

$$\sigma_i = \sqrt{\frac{1}{l-1} \sum_{j=1}^{l} (z_{ij} - \mu_i)^2}. \tag{6.6}$$

ZN can be interpreted geometrically as first projecting all representations to the hyperplane orthogonal to $\mathbf{1}$, and then normalizing these projections to have norm $\sqrt{l}$. This results in embeddings on a hypersphere in $\mathbb{R}^{l-1}$ with radius $\sqrt{l}$. Interestingly, if it is assumed that the mean of the original representations is proportional to $\mathbf{1}$, the mean of embeddings after ZN will be at the origin. The influence of ZN on hubness is discussed in Section 6.3.3.

6.2.3 Other embedding methods

In addition to the above methods, the following representation embeddings have been proposed to improve FSL performance.

Parameterless transductive feature re-representation (ReRep)

ReRep [53] aims to address the *sample bias*, where a small number of supports is unable to fully represent the given class, negatively influencing FSL performance. ReRep computes support and query embeddings in two stages. First, queries are re-represented (embedded) as a linear combination of other queries. The weights in the linear combination are computed by an attention mechanism, aiming to embed similar queries closer together. In the second step, each support sample is embedded as a linear combinations of itself and the query samples, moving the labeled support samples closer to similar queries. The authors claim that this reduces the sample bias by propagating information from queries to supports.

Unsupervised discriminant subspace learning (EASE)

EASE [72] linearly transforms the representations to a lower-dimensional space, and then projects the transformed representations to the hypersphere with L_2 normalization. The projection's weight matrix is determined by an optimization problem that maximizes similarity between similar representations, while preserving the maximal variance in the representations. The similarities between representations are based on low-rank representations [186], found by a singular-value decomposition of the matrix of representations. EASE thus learns a projection that maximizes inter-class distance while minimizing the intra-class distance, leading to improved class separability and classification performance.

Task centroid projection removing (TCPR)

The authors of TCPR [75] argue that the sample bias observed in FSL is related to distance between supports and the *task centroid*, defined as the mean of all class centroids in a given episode. Supports closer to the task centroid are more “ambiguous”, resulting in decision boundaries that are sensitive to small perturbations in the supports. To alleviate this problem, TCPR removes the projection along task centroid, by projecting representations onto the hyperplane orthogonal to the task centroid. The representations are L_2 normalized before and after projection, meaning that the resulting embeddings lie on the hypersphere within the hyperplane orthogonal to the task centroid. This causes all embeddings to be equidistant from the task centroid, eliminating possible ambiguity arising from closeness to the task centroid. In fact, TCPR is similar to ZN, since both methods project samples into a hypersphere embedded in the hyperplane orthogonal to the data mean (task centroid). However, while ZN assumes that this mean is proportional to $\mathbf{1}$, TCPR estimates this mean using representations of the base data.

6.2.4 Hyperspherical embeddings

Interestingly, almost all embedding techniques outlined above end up with embeddings on the hypersphere. Even the straightforward L_2 normalization – which just discards all information about the representation norm – results in significant improvements in FSL performance. The same trend can be observed in SSL for instance, where it is common to L_2 normalize representations or projections, prior to computing the loss [17, 19, 22–24, 47, 49]. These observations thus poses the question: *what information is carried in the norm, and why is it so often beneficial to suppress, or even completely discard this information?* Despite the recent successes of hyperspherical embeddings, answering this question, and understanding exactly what information is carried in representation norms, remains severely under-explored. To this end, Paper V presents a novel hypothesis on the relationship between representation norms, and the number of objects in the corresponding image. In fact, we show that there is a monotonically increasing relationship between the norm of a CNN-based representation, and the number of prototypical *object images* present in the representation’s input image.

6.3 The hubness problem

The benefits of hyperspherical representations in FSL can also be partially understood through the hubness problem [55]. The hubness problem refers to the tendency of a few points (hubs) to appear among the nearest neighbors of many other points [89]. This can have a negative impact on distance-based classification when hubs from one class appear among the nearest neighbors of many points from another class. In the following we explain the hubness problem in terms of distances to the data mean, and as a consequence of non-zero density gradients. The effect of hubness on FSL performance is then discussed in more detail in Sections 6.3.3 and 6.3.4.

6.3.1 Hubness and distances to the data mean

The intuition behind the hubness problem can be explained with distances between points and the mean of their associated PDF [55, 89, 187–190]. In short, a PDF $p : \mathbb{R}^l \to \mathbb{R}$ with mean $\boldsymbol{\mu} = \mathbb{E}_{\mathbf{z}\sim p}(\mathbf{z})$, exhibits hubness if there exists points $\mathbf{a}$ and $\mathbf{b}$ sampled from p, such that $||\mathbf{a} - \boldsymbol{\mu}|| \neq ||\mathbf{b} - \boldsymbol{\mu}||$. This essentially says that all distributions where all mass is not equidistant from the origin, will have some degree of hubness. Specifically, for a random point $\mathbf{z} \sim p$, we have

$$\begin{aligned}\mathbb{E}_{\mathbf{z}\sim p}(||\mathbf{a} - \mathbf{z}||^2) &= \mathbb{E}_{\mathbf{z}\sim p}(||(\mathbf{a} - \boldsymbol{\mu}) - (\mathbf{z} - \boldsymbol{\mu})||^2) && (6.7)\\ &= ||\mathbf{a} - \boldsymbol{\mu}||^2 + \mathbb{E}_{\mathbf{z}\sim p}(||\mathbf{z} - \boldsymbol{\mu}||^2) - 2(\mathbf{a} - \boldsymbol{\mu})^\top \underbrace{\mathbb{E}_{\mathbf{z}\sim p}(\mathbf{z} - \boldsymbol{\mu})}_{=0} && (6.8)\\ &= ||\mathbf{a} - \boldsymbol{\mu}||^2 + \mathbb{E}_{\mathbf{z}\sim p}(||\mathbf{z} - \boldsymbol{\mu}||^2). && (6.9)\end{aligned}$$

Similar computations for $\mathbb{E}_{\mathbf{z}\sim p}(||\mathbf{b} - \mathbf{z}||^2)$ and re-arranging terms gives

$$\mathbb{E}_{\mathbf{z}\sim p}(||\mathbf{a} - \mathbf{z}||^2) - \mathbb{E}_{\mathbf{z}\sim p}(||\mathbf{b} - \mathbf{z}||^2) = ||\mathbf{a} - \boldsymbol{\mu}||^2 - ||\mathbf{b} - \boldsymbol{\mu}||^2. \tag{6.10}$$

From this we observe that if $\mathbf{a}$ is closer to the data mean, $\boldsymbol{\mu}$, than $\mathbf{b}$, that is

$$||\mathbf{a} - \boldsymbol{\mu}||^2 < ||\mathbf{b} - \boldsymbol{\mu}||^2 \tag{6.11}$$

we get

$$\mathbb{E}_{\mathbf{z}\sim p}(||\mathbf{a} - \mathbf{z}||^2) < \mathbb{E}_{\mathbf{z}\sim p}(||\mathbf{b} - \mathbf{z}||^2). \tag{6.12}$$

This means that an arbitrary point, $\mathbf{z}$, from p is more likely to be close to $\mathbf{a}$ than $\mathbf{b}$, since $\mathbf{a}$ is closer to the data mean, $\boldsymbol{\mu}$. Points close to the data mean are therefore more likely to be hubs.

6.3.2 Spatial centrality and density gradients

The motivation above can be extended to include changes in probability density between regions in the representation space. High probability density results in local "means", as these are regions where we expect to find more points, compared to regions with lower density. This tendency is called *spatial centrality* (or just *centrality*) [191, 192]. Following the motivation in Section 6.3.1, points close to, or within high-density regions are more likely to be hubs. Spacial centrality is therefore closely related to the hubness problem. Density gradients provide a natural means to quantify spatial centrality. The gradient of a PDF indicates changes in density, meaning that the existence of a density gradient implies the existence of spacial centrality, and consequently, the existence of hubness.

Having established the connection between density gradients and the hubness problem, we can define hubness as follows. Suppose p is a PDF defined on a continuous smooth surface $S \subseteq \mathbb{R}^l$, and let $\Pi_{\mathbf{z}}(S)$ be the tangent plane of S at the point $\mathbf{z} \in S$. Then p is said to exhibit hubness if there exists a point $\mathbf{z} \in S$ and a direction $\mathbf{e} \in \Pi_{\mathbf{z}}(S)$ such that $\nabla_{\mathbf{e}}\, p \neq 0$, where $\nabla_{\mathbf{e}}$ denotes the directional derivative along $\mathbf{e}$ in the ambient space, $\mathbb{R}^l$.

Hence, any distribution with a non-zero density gradient has some degree of hubness, meaning that (i) hubness is a property of the data distribution, and not an artifact arising from *e.g.* a finite number of samples; and (ii) hubness exists in the majority of distributions, since most distributions have points with non-zero density gradients.

Interestingly, we can argue that if $S = \mathbb{R}^l$, there does not exist a valid PDF, p, without any hubness. Since $\Pi_{\mathbf{z}}(\mathbb{R}^l)$ is isomorphic to $\mathbb{R}^l$, the only PDF satisfying the zero-gradient requirement is the un-bounded, constant (uniform) PDF. This PDF does not exist, since its integral over $\mathbb{R}^l$ will always be equal to ∞ or 0, depending on the value of the constant.

Since no hubness-free PDF exists on $\mathbb{R}^l$, we have to look to other surfaces to find PDFs without hubness. To this end, we prove in Paper IV that the uniform distribution on the hypersphere has a vanishing density gradient everywhere, and thus no hubness.

6.3.3 Hubness in few-shot learning

FSL classifiers are often based on distances in the representation space. This, combined with the small number of labeled support samples, makes FSL methods particularly prone to the hubness problem. Specifically, if a support sample is a hub, many queries will be assigned to it even though they come from different classes, which is detrimental to classification performance.

Z-score normalization

Despite the FSL methods' susceptibility to hubness, there exists little research on mitigating the hubness problem in FSL. In their recent work, Fei *et al.* [55] attempt to address the hubness problem in FSL with Z-score normalization (Section 6.2.2) applied to the representations. Motivated by an "equidistant from the data mean" argument, similar to the one presented in Section 6.3.1, Fei *et al.* [55] argue that ZN embeds points on the hypersphere such that the data mean coincides with the origin. This causes the embeddings to lie at equal distance from the data mean, resulting in reduced hubness.

Although hyperspherical, zero-mean data reduces hubness, it is only guaranteed by ZN when the mean *before* the embedding is proportional to $\mathbf{1} = [1, \dots, 1]^\top$. This is a key limiting factor of ZN since the mean of an arbitrary distribution of representations can have a direction that differs from $\mathbf{1}$. Furthermore, we argue that hyperspherical, zero-mean representations is an insufficient condition for zero hubness. This can be seen by the following counterexample. Consider a mixture of two von Mises–Fisher (vMF) distributions with equal concentrations and equal prior probabilities, placed at opposite poles on the hypersphere. Since these distributions are equal, but located at opposite sides of the hypersphere, the overall mean will be 0. However, the mixture will have increasing density towards the poles at which the vMF distributions are located, meaning that it will also have a density gradient pointing towards one of these poles. Following the argument in Section 6.3.2, this distribution will have hubness, despite it being on the hypersphere with 0 mean.

Hyperspherical embeddings for few-shot learning

ZN is part of the recent trend in FSL where representations are embedded on a hypersphere before classification [50, 55, 68, 71, 72]. However, despite their popularity, it is not yet well understood why hyperspherical embeddings are beneficial for FSL performance. One possible reason for this is that the uniform distribution on the hypersphere satisfies the hubness-free condition, $\nabla_{\mathbf{e}}\, p = 0$ for all $\mathbf{e}$ tangent to the hypersphere at a given point (we prove this in Paper IV). Embeddings uniformly placed on the hypersphere are therefore hubness-free.

6.3.4 Tradeoff between hubness and class separability

Embedding representations uniformly on certain surfaces (*e.g.* hyperspheres) is a way to eliminate hubness in the embeddings. From a classification perspective however, the uniform distribution is not optimal. Classification rather benefits from compact and well separated, distinct classes in the embedding space, making it simple for a classification algorithm to discriminate between classes. This implies that there exists a tradeoff between minimizing hubness, and maximizing class separability in the data. One way to look at this tradeoff is to categorize hubs in the data as either *good hubs* or *bad hubs* [193]. Good hubs only appear among the nearest neighbors of points with the same class label as themselves, making it safe for classifiers to rely on such hubs when classifying the data. Bad hubs however, appear as neighbors for points from different classes, making it more difficult to classify the data correctly. The tradeoff between hubness and class separability can thus be reformulated as minimizing bad hubness, while retaining or increasing good hubness. We address this tradeoff in Paper IV, which proposes two new methods for embedding points on the hypersphere, reducing hubness while retaining class separability.

Part II

Summary of research

Paper I

Leveraging Tensor Kernels to Reduce Objective Function Mismatch in Deep Clustering

Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer.

Under review in Pattern Recognition.

- Code: `https://github.com/DanielTrosten/tk-uco`
- Open access conference version: `https://doi.org/10.7557/18.5709`
- Talk at NLDL 2021: `https://youtu.be/y-jACE0PzcI`

The representational power of DNNs, coupled with the lack of a proper supervision signal, makes deep clustering models prone to trivial solutions, where the representation similarities are no longer representative of the input similarities. Traditionally, this issue has been solved with AEs [41]. However, we find that the difference between the clustering and reconstruction objectives makes AE-based deep clustering models susceptible to OFM, negatively influencing the clustering performance.

The objective of this paper is thus to improve representations for single-view deep clustering, without introducing detrimental amounts of OFM between the clustering and auxiliary objectives. To this end, we propose the unsupervised companion objectives (UCOs), which are additional loss functions that encourage a consistent cluster structure throughout the layers of the model's DNN. Our approach is illustrated in Figure 12, which shows an example of how a deep clustering model can be augmented with the proposed UCOs.

The UCOs are designed to maximize the CSD between clusters at intermediate layers in the DNN. Since these representations might not be vectorial, the UCOs

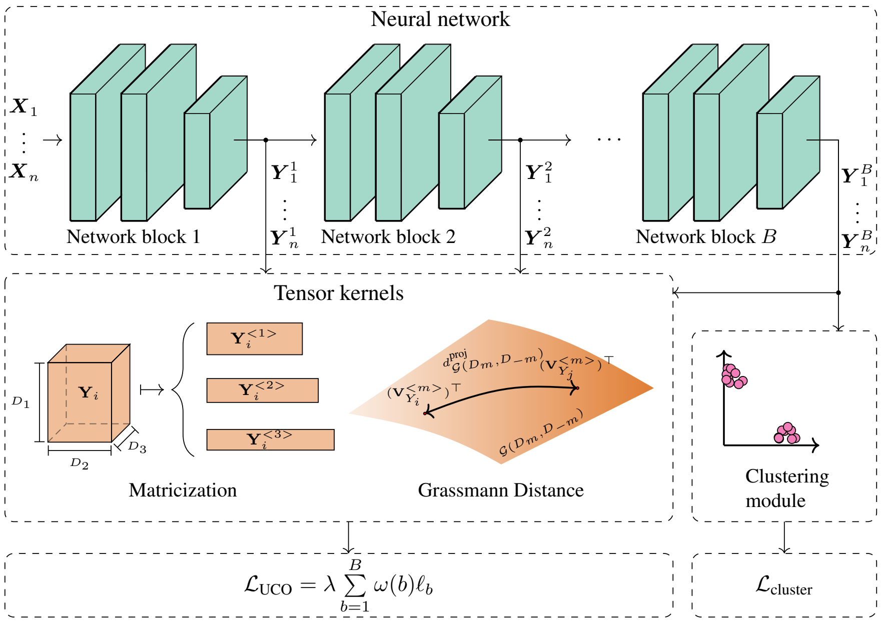


Figure 12: *Overview of a deep clustering model augmented with the UCOs. The UCOs use tensor kernels to encourage the same cluster structure at the output of all network blocks. Figure from Paper I.*

leverage tensor kernels [90] to quantify pairwise similarities between tensorial intermediate representations of arbitrary rank. The tensor kernels allow the similarity function to take structural, *e.g.* spatial or temporal, information into account. This ability is crucial in for instance CNNs, since the position of certain pixels relative to each other is often informative of the objects contained in the image.

Our experiments show that the UCOs lead to reduced OFM, and improves the overall clustering performance of DDC and DEC, also when compared to analogous AE-based models.

Contributions by the author

- The idea was conceived by me, and further developed along with all co-authors.
- I implemented the methods and conducted all experiments.
- I wrote the initial manuscript draft, which was then refined along with all co-authors.

Paper II

Reconsidering Representation Alignment for Multi-view Clustering

Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer.

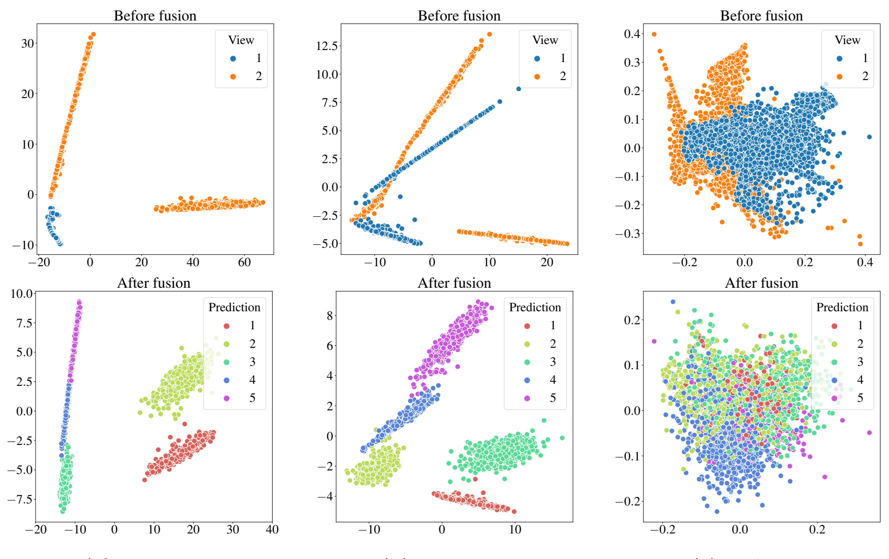
IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2021.

- Code: `https://github.com/DanielTrosten/mvc`
- Open access version: `https://openaccess.thecvf.com/content/CVPR2021/html/Trosten_Reconsidering_Representation_Alignment_for_Multi-View_Clustering_CVPR_2021_paper.html`
- Talk: `https://youtu.be/1WTjoCPyveM`

This paper studies the alignment of view-specific representation distributions, which is a form of SSL used in deep MVC [45]. Here, we identify 3 drawbacks of aligning representation distributions that can have a negative effect on the model's clustering performance.

1. *Aligning representations prevents view-prioritization.* If all views are equal in the representation space, it is not possible to emphasize their representations differently, according to the amount of relevant information contained in each view.

2. *One-to-one alignment of clusters is only attainable when encoders can separate all clusters in all views.* If two clusters are inseparable in the input space of a given view, alignment will cause the representations of these clusters to be inseparable for all views. This happens because the view specific encoders can not learn to separate inseparable clusters, but



(a) *SiMVC* (b) *CoMVC* (c) *EAMC*

Figure 13: *Performance of different methods for deep MVC on a two-view toy dataset. Figure adapted from Paper II.*

they can learn to merge two clusters that originally were separable. This drawback is formalized in a simplified setting in Proposition 1 in the paper. It is also elaborated and studied further in the next paper (Paper III).

3. *Aligning representation distributions can make it harder to discriminate between clusters.* Simply aligning distributions does not mean that the same clusters from different views are aligned in the representation space. Cluster i in view v can be aligned to cluster j in view u, making it more difficult to discriminate between clusters i and j in the representation space.

To address these drawbacks we propose two new methods for deep MVC. The first method, simple multi-view clustering (SiMVC), is a simple baseline model without any forms of alignment, or any other forms of SSL. We find that it is important to include such models in the evaluation, in order to have an accurate estimate of the model's performance before additional components are added. The second method, contrastive multi-view clustering (CoMVC), extends SiMVC with a contrastive learning component. CoMVC aligns angles of view-specific representations at the sample level, instead of at the distribution level. Furthermore, it includes an adaptive weighting mechanism that can disable the alignment procedure if the information content is lower in one of the views. This makes CoMVC much less sensitive to the above drawbacks of distribution alignment.

Figure 13 is extracted from the paper, and shows representations obtained from SiMVC, CoMVC, and EAMC for a two-view toy dataset. The dataset is designed such that different clusters overlap in the two views, requiring that the models use information from both views to cluster the data. SiMVC and CoMVC are able to prioritize, and use the information available in the two views. EAMC on the other hand, attempts to align the view-specific representation distributions, destroying the cluster strucure.

The other experimental results in the paper show that SiMVC performs remarkably well without any forms of SSL, outperforming several other, more complex methods. CoMVC performs even better than SiMVC, resulting in a new state-of-the-art method for deep MVC.

Contributions by the author

- The idea was conceived by me, and further developed along with all co-authors.
- I implemented the methods and conducted all experiments.
- I wrote the initial manuscript draft, which was then refined along with all co-authors.

Paper III

On the Effects of Self-supervision and Contrastive Alignment in Deep Multi-view Clustering

Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer.

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2023.

- **Selected as a highlight at CVPR 2023**
- Code: `https://github.com/DanielTrosten/DeepMVC`
- Preprint: `https://arxiv.org/abs/2303.09877`

In this paper we continue investigating the effects of SSL and contrastive alignment in deep MVC. We find that the field lacks direction and consistent results, especially when it comes to SSL-based representation learning. This motivates us to develop DeepMVC (Figure 14), which is a general framework for deep MVC models, including the majority of recent methods as instances. The DeepMVC framework, along with its open source implementation, allows us to systematically reason about and analyze different model components, focusing on SSL and representation learning. We extend the analysis from Paper II to arbitrary forms of alignment, proving in a simplified setting that alignment negatively influences cluster separability when the number of views increases. Next, we leverage these insights to develop 6 new instances of DeepMVC with varying SSL-based components.

We conduct extensive experiments with recent methods and our new instances, and find that contrastive alignment works well when the number of views is small, but degrades performance for a larger number of views. These results

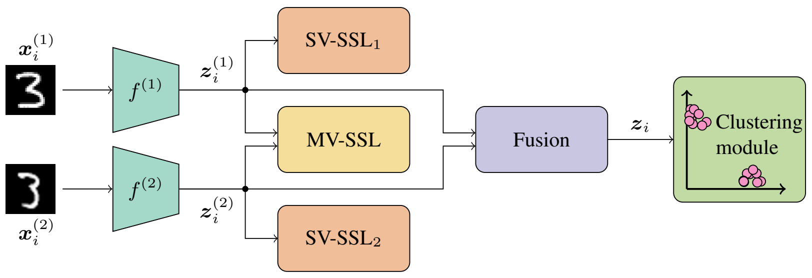


Figure 14: *Overview of the DeepMVC framework for a two-view dataset. The framework can be extended by adding more view-specific encoders and SSL components. Figure from Paper III.*

on contrastive alignment are in line with the findings in Paper II, and our theoretical analysis. Conversely, we find that methods based on maximizing mutual information between view-specific representations work well for many views, while not performing as good as contrastive alignment on datasets with few views.

Additional findings from our experiments include:

1. All methods benefit from some form of SSL. However, what type of SSL is beneficial varies across datasets.
2. The performance of methods relative to each other depends heavily on dataset characteristics, such as number of views and class imbalance.
3. Our new instances out-perform several recent methods, and one of the new instances achieves the best overall performance.

We conclude the paper by making recommendations for future work and advancements in the field of deep MVC.

Contributions by the author

- The idea was conceived by me, and further developed along with all co-authors.
- I formulated and proved the included propositions.
- I implemented the methods and conducted all experiments.
- I wrote the initial manuscript draft, which was then refined along with all co-authors.

Paper IV

Hubs and Hyperspheres: Reducing Hubness and Improving Transductive Few-shot Learning with Hyperspherical Embeddings

Daniel J. Trosten, Rwiddhi Chakraborty, Sigurd Løkse, Kristoffer Wickstrøm, Robert Jenssen, and Michael Kampffmeyer.

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2023.

- ▶ Code: `https://github.com/uitml/noHub`
- ▶ Preprint: `https://arxiv.org/abs/2303.09352`

This paper addresses the hubness problem in transductive FSL. As shown in Figure 15, we observe a correspondence between reduced hubness, and improved FSL performance.

We begin by proving that the hyperspherical uniform as zero directional derivative along all directions tangent to the hypersphere, at all points. This means that the hyperspherical uniform is essentially hubness-free. Motivated by these findings, we seek to embed representations for FSL uniformly on the hypersphere, as this will alleviate the hubness problem, possibly increasing classification performance. However, the uniform distribution is not suitable for classification. We therefore develop two new embedding methods – uniform hyperspherical structure-preserving embeddings (noHub) and noHub with support labels (noHub-S) – which provably optimize a tradeoff between local similarity preservation and uniformity. The noHub embedding is illustrated in Figure 16. The key idea behind noHub and noHub-S is to preserve class structure, while simultaneously distributing the representations as uniformly as possible, reducing hubness.

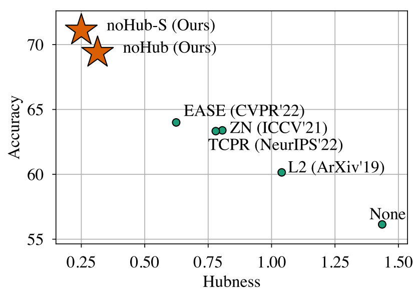


Figure 15: *Relationship between hubness and 1-shot classification accuracy on mini-ImageNet [69] using the SimpleShot classifier [71] with different embeddings. Figure from Paper IV.*

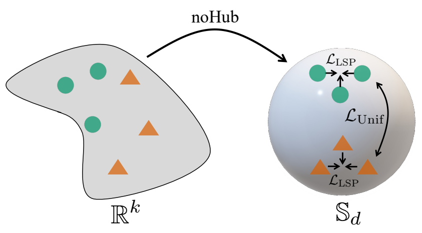

noHub
$\mathcal{L}_{\text{LSP}}$
$\mathcal{L}_{\text{Unif}}$
$\mathcal{L}_{\text{LSP}}$
$\mathbb{R}^k$
$\mathbb{S}_d$

Figure 16: *Illustration of noHub, which embeds representations on the hypersphere, optimizing a tradeoff between uniformity and local similarity preservation. Figure from Paper IV.*

Our experiments show that the embeddings increase performance, and reduce hubness for a wide range of FSL classifiers, with several feature extractors, on several datasets.

Contributions by the author

- The idea was conceived and developed in joint collaboration with all authors.
- I formulated and proved propositions 1, 2, and 5.
- Implementation and evaluation of baselines and proposed methods was done by me and RC.
- The initial manuscript draft was written by me and RC, and then refined along with all co-authors.

Paper V

Norm-count Hypothesis: On the Relationship Between Norm and Object Count in Visual Representations

Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer.

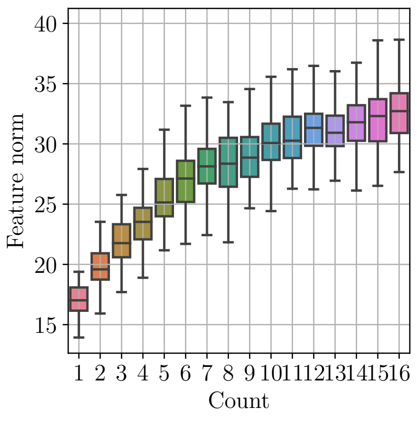
In submission.

▶ Code will be made publicly available upon publication of the paper.

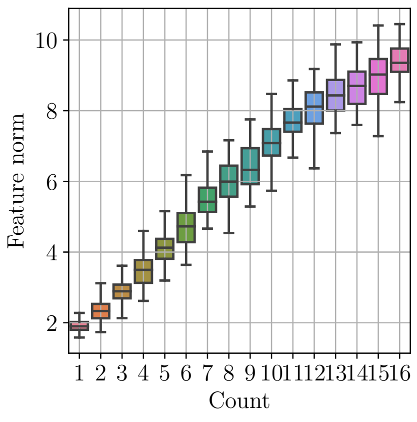
With this paper we aim to enhance the understanding of norms of representations produced by CNNs. The paper presents the norm-count hypothesis (NCH), in which we hypothesize that there is a monotonically increasing relationship between the norm of a representation, and the number of *object images* present in the input image. These object images are "prototypical" images that – when processed by a CNN – provide delta-like responses in the output feature maps. Aggregating these delta-responses using a global pooling operator then results in a representation, whose norm is proportional to the number of object images.

We prove that the NCH holds under certain assumptions on the model and input images – with the representation norm being upper-bounded by a monotonically increasing function of the object counts. For linear CNNs with GAP, the upper bound reduces to an equality, showing that there is an exact proportionality between norm and count in this case.

We conduct a controlled experimental evaluation where we have prior knowledge of object counts. The results of these experiments corroborate the NCH, showing an increasing relationship between norm and count (see Figure 17). We also investigate the effects of L_2 normalization on supervised, self-supervised, and few-shot classification performance. We find that L_2 normalized representations give better classification performance for datasets with varying counts – indicating



(a) *Supervised*



(b) *SimCLR*

Figure 17: *Boxplots illustrating the relationship between norm and count in CNN-based representations. The dataset consists of images from STL-10 [194], where a random number of images are placed at random positions in a 4×4 grid. The norms are computed from ResNet-50 [122] representations, trained using either a supervised cross-entropy loss, or with the SimCLR framework. Figure adapted from Paper V.*

that the norm acts as additional noise when classifying these datasets.

Contributions by the author

- The idea and hypothesis was conceived by me, and developed further with all co-authors.
- I formulated and proved all propositions, theorems, and corollaries.
- I implemented and conducted all experiments.
- I wrote the initial manuscript draft, which was then refined along with all co-authors.

7 Conclusion

In this thesis, we have addressed key challenges with representation learning for deep clustering and FSL. Along with our main objective of improving and understanding representation learning in these fields, we had the following specific objectives:

1. Improve similarity preservation in deep single-view clustering.
2. Understand and improve SSL for representation learning in deep MVC.
3. Address the hubness problem in transductive FSL.
4. Improve the understanding of representation norms in FSL.

In the following, we summarize our solutions to these objectives, and briefly discuss possible directions for future research. We also reflect on the future of learning with limited labels – a field in which deep clustering and FSL are key elements, now and in the future.

Deep clustering

In Paper I, we propose an alternative to the autoencoder-based approach to improve similarity preservation for deep clustering. The proposed UCOs improve similarity preservation and clustering performance, while simultaneously limiting the amount of OFM introduced by the auxiliary losses.

Papers II and III shed new light on SSL-based representation learning for deep MVC. In particular, we investigate the effects of aligning view-specific representations, and find that these alignment procedures sometimes have severe drawbacks. Naïvely aligning representations can result in reduced separability between clusters in the representation space, negatively influencing the clustering performance. We provide both theoretical and experimental evidence indicating that this effect becomes worse when the number of views increases. To enhance the understanding of different SSL-based components in deep MVC, we also propose the DeepMVC framework, along with an open-source implementation. The framework allows for fair and accurate comparisons between models and components, providing a deeper understanding of the effects of, and interactions between, different model components.

One interesting direction for future work is to use the DeepMVC framework to generalize the UCOs from Paper I to deep MVC. The UCOs can be used on each individual encoder, or they can be shared across encoders, depending on whether it is beneficial to enhance view-specific or shared cluster structure.

The DeepMVC framework, and other initiatives like it, are especially important in contemporary deep learning research. This is because – as I write in [12] – the deep learning community faces challenges related to questionable research practices in the evaluation of new and existing methodology. Although the questionable practices are visible in several areas of deep learning research, they are particularly important to address in settings with limited labels. This is because we no longer can rely on the ground truth information provided by the labels, to make sure that the model does what it is supposed to do.

Clustering will likely continue to be an important task in machine learning, as it has been for decades already. To continue pushing the state-of-the-art, it is vital to continue the efforts on learning high-quality representations within the clustering models. We have already seen massive advancements in generic representation learning frameworks for different modalities [17, 21, 35, 195], and there is much potential in adapting such frameworks to current and new clustering models. Multi-modality also plays an important role in future advancements, as multiple views or modalities provide additional information about observations. Exploiting the synergies across views without discarding view-specific information will thus be key to future advancements in the field.

Few-shot learning

With Paper IV, we tackle the hubness problem in transductive FSL. We prove that the uniform distribution on the hypersphere is hubness-free, motivating us to embed representations as uniformly as possible on the hypersphere. However, in order to also preserve class-structure in the embeddings, we develop two new methods that provably optimize a tradeoff between uniformity and local similarity preservation. Thus, our methods both reduce hubness and preserve class structure. With extensive experiments with several datasets, feature extractors and FSL classifiers, we show that the proposed embeddings result in better classification performance, and reduced hubness, compared to previous state-of-the-art embedding methods.

The NCH presented in Paper V provides further insight on representation norms, and hyperspherical embeddings through L_2 normalization. To the best of our knowledge, ours is the first work that rigorously relates norm and count in CNN-based representations.

The research on representation norms, hyperspherical embeddings, and hubness in FSL is still in its early stages. Hence, with our initial work in Papers IV and V we hope to inspire further interest in understanding these concepts in

FSL, and in other areas. In fact, there are already several works outside FSL generalizing deep learning to non-Euclidean embedding spaces. DNNs on graphs and other non-Euclidean data structures is a research subject that has seen growing interest in recent years [196–205]. We have also seen developments in non-Euclidean embedding spaces for classification and regression [206, 207], as well as hyperspherical regularization for DNNs [208]. Thus, we believe that our thoughts on hubness and distributions of representation in non-Euclidean spaces can be generalized to other subfields in machine learning and deep learning.

Future work on learning with limited labels

Learning with few labels has become increasingly important in recent years – a trend that will likely continue in the near future. This is because it is simply not possible for human annotators to keep up with the rapidly increasing amounts of new data that is gathered every day. The same goes for the creation of hand-crafted features for downstream applications, where it is becoming impossible for human experts to cope with the extreme amounts of data with ever-increasing diversity and complexity. Deep learning has become a key solution to these issues, and to leverage massive amounts of raw data. With the initial success of deep learning in supervised settings now being translated to applications with few or no labels, we are seeing impressive advancements in the development of intelligent systems – far beyond what we thought was possible only a few years ago [24, 27, 35, 209]. At the time of writing this thesis for instance, OpenAI's ChatGPT, powered by generative pre-trained transformer (GPT)-4 [35], has taken the world by storm, baffling the machine learning community, and society in general.

Although representation learning lies at the very core of these systems, we still have little understanding of how high-dimensional representations behave. Research along these directions is still in its early stages [146, 210, 211], meaning that there is much untapped potential in further understanding DNN-based representations, their quality, and how they can be learned from large amounts of unlabeled data.

Tangent to how representations can be improved, we also need to develop methods to explain what information a representation actually carries. Understanding exactly how a representation relates to its input is crucial if the representations are used in any application with non-zero risk. Fortunately, the field of explainable artificial intelligence (XAI) has grown alongside the development of better models. Not so fortunately however, methods for explaining model outputs have largely been focusing on explaining scalar predictions, and not vectorial representations. To address this severe gap in the field of XAI, our work on representation learning explainability (RELAX) [13] proposes a new method to explain neural-network representations, without relying on any predictions or label information. To the best of our knowledge, RELAX is the first method capable of explaining arbitrary representations. With the increasing popularity of unsupervised learning, SSL and FSL, we expect that methods for explaining representations will increase in number and performance – improving the transparency and trustworthiness of models for representation learning.

The recent leaps in performance of machine learning systems, and particularly in large language models, have massively accelerated initiatives to better regulate the development and use of machine learning models[1]. We are currently at a point where generative language and vision models could create enormous amounts of disinformation that is indistinguishable from human-generated images or text. Regulating the use of such models – and any other models capable of harmful behavior – is therefore a critical step towards responsible further development of intelligent systems.

In summary, we see that the potential of machine learning, deep learning, and learning with few labels is greater now than ever before. To harness this potential, and to make sure that it is used for good, we need to not only make the models perform better, but also to better understand, interpret, and regulate them.

[1]See for instance the EU Artificial Intelligence Act [212]

Part III

Included papers

Paper I

Leveraging Tensor Kernels to Reduce Objective Function Mismatch in Deep Clustering

Daniel J. Trosten[a,1,*], Sigurd Løkse[a,1], Robert Jenssen[a,b,c,1], Michael Kampffmeyer[a,c,1]

[a]*Department of Physics and Technology, UiT The Arctic University of Norway. Hansine Hansens veg 18, 9019 Tromsø, Norway.*
[b]*Department of Computer Science, University of Copenhagen, Universitetsparken 1, Copenhagen 2100, Denmark*
[c]*Norwegian Computing Center, Gaustadalleen 23a, 0373 Oslo, Norway*

Abstract

We investigate the impact of objective function mismatch in deep clustering. We find that the popular autoencoder-based approach to deep clustering can lead to both reduced clustering performance, and a significant amount of mismatch between the reconstruction and clustering objectives. To reduce the mismatch, while maintaining the structure-preserving property of an auxiliary objective, we propose a set of new auxiliary objectives for deep clustering, referred to as the Unsupervised Companion Objectives (UCOs). The UCOs rely on a kernel function to formulate a clustering objective on intermediate representations in the network. Generally, intermediate representations can include other dimensions, for instance spatial or temporal, in addition to the feature dimension. We therefore argue that the naïve approach of vectorizing and applying a vector kernel is suboptimal for such representations, as it ignores the information contained in the other dimensions. To address this drawback, we equip the UCOs with structure-exploiting tensor kernels, designed for tensors of arbitrary rank. The UCOs can thus be adapted to a broad class of network architectures.

*Corresponding author

Email addresses: `daniel.j.trosten@uit.no` (Daniel J. Trosten), `sigurd.lokse@uit.no` (Sigurd Løkse), `robert.jenssen@uit.no` (Robert Jenssen), `michael.c.kampffmeyer@uit.no` (Michael Kampffmeyer)

[1]DT, SL, RJ and MK are with the UiT Machine Learning Group, `machine-learning.uit.no`, and the Visual Intelligence Centre, `visual-intelligence.no`. RJ is with the Pioneer Centre for AI, `aicentre.dk`.

Our experiments show that the mismatch between the UCOs and the main clustering objective is lower, compared to a similar autoencoder-based model. Further, we illustrate that the UCOs improve the clustering performance of the model, in contrast to the autoencoder-based approach. The code for our experiments is available at https://github.com/danieltrosten/tk-uco.

Keywords: tensor kernels, unsupervised companion objectives, objective function mismatch, deep clustering

1. Introduction

Clustering is a long-standing problem in machine learning research. The recent success of deep learning [1] has given rise to *deep clustering* – a subfield of deep learning where deep neural networks are trained with unsupervised loss functions designed for clustering.

Many of the loss functions created for deep clustering attempt to discover and strengthen clusters in the representations produced by the neural network. However, the minimization of these objectives can often result in trivial solutions, where the network maps all inputs to a constant representation, forcing all points to be assigned to the same cluster. This behavior is an extreme case of a more general problem in deep clustering, where minimizing the clustering loss results in an embedding which does not respect the local similarity structure of the input space [2].

To address the similarity preservation problem, several recent deep clustering models employ deep neural networks that have been pre-trained as autoencoders [3, 2, 4]. In the majority of these models, clustering is performed in the autoencoder's latent space. The model is fine-tuned by minimizing either the clustering loss alone, or both the clustering loss, and the autoencoder's reconstruction loss. Guo *et al*. [2] argue that fine-tuning with both the reconstruction loss and clustering loss leads to improved preservation of local similarities in autoencoder-based deep clustering models. They retain the reconstruction loss during fine-tuning, and illustrate that it leads to improved clustering performance for the well known Deep Embedded Clustering (DEC) model [3].

However, clustering and reconstruction are fundamentally different tasks, and thus they require the representations produced by the network to encode different types of information. When analyzing images for instance, successful reconstruction requires the representation to encode the position, color, and fine-grained details of objects in the images. Successful clustering on the other hand, often only depends on higher level information from the image, such as the presence of certain objects or features.

Based on these observations, we hypothesize that deep clustering models trained to simultaneously cluster and reconstruct, are prone to suffer from *Objective Function Mismatch* (OFM) [5, 6]. OFM occurs when the optimization of an auxiliary objective (*e.g.* reconstruction) has a negative impact on the optimization of the main objective (*e.g.* clustering).

In this work we study OFM in deep clustering, and show that models trained to reconstruct and cluster simultaneously, can indeed exhibit a significant amount of OFM. Further, we develop a set of new auxiliary objectives for deep clustering. Our *Unsupervised Companion Objectives* (UCOs) are specifically designed for clustering, in order to reduce the OFM between them and the main clustering objective. In addition, UCOs help preserve the local similarity structure by encouraging a consistent cluster structure throughout the network.

The UCOs rely on a kernel function to measure the similarity between intermediate representations in the network. In general, these intermediate representations may have several dimensions, such as spatial or temporal, in addition to the feature dimension – meaning that they are naturally represented as tensors, rather than vectors. Here, we argue that naïve kernels based on vectorization of tensors do not take this structural (*e.g.* spatial or temporal) information into account. To address these drawbacks, we utilize tensor kernels [7] to measure similarities in the proposed UCOs. The tensor kernels are *structure-exploiting*, meaning that they are able to exploit both structural information and feature information.

We use convolutional neural networks (CNNs) and image data, as well as multi-layer perceptrons (MLPs) and vector data in this work. However we emphasize that the tensor kernels make the UCOs compatible with a wide range of network architectures and data types, as long as they produce tensorial intermediate representations. Furthermore, the UCOs only depend on the intermediate representations through the kernel function – allowing them to be adapted to all architectures for which a kernel can be specified for the intermediate representations.

In our experiments, we demonstrate that adding the UCOs to different deep clustering models both improves the overall clustering performance, and leads to lower OFM when compared to an analogous autoencoder-based model.

A preliminary version of this paper was published in [8]. Here, we extend several aspects of our previous work:

1. We argue that the vectorization-based kernel used in [8] does not take the structural information of intermediate representations into account. To address this issue, we propose an improved version of the UCOs that incorporate structural information by using tensor kernels [7]. This allows us to formulate a structure-exploiting kernel on the intermediate representations of a wide range of deep neural networks, without resolving to suboptimal heuristics, such as vectorization.

2. We conduct an extensive evaluation of the proposed methodology, with several additional experiments. In contrast to [8], which only included image datasets, we include two extra vector-based datasets – illustrating that the UCOs improve clustering performance on these datasets as well. In addition, we analyze the performance of the UCOs across different kernels and weighting strategies, and provide a thorough experimental analysis of the OFM observed for the different models. Finally, we attach the UCOs to additional base models, illustrating that the UCOs work well with these models as well.

3. We provide new insight into the observed increase in OFM during training with the UCOs.

4. We include an extensive overview of related work, placing our work in the current context of deep clustering.

The rest of the paper is structures as follows: Section 2 gives an overview of related work on deep clustering with and without autoencoders, as well as previous work on OFM in different areas of deep learning. In Section 3 we present the proposed UCOs, and in Section 4 we motivate and introduce the tensor kernels, and show how they are adapted to the UCOs. Section 5 gives a mathematical definition of OFM, which allows us to measure OFM when training deep clustering models. Section 6 provides the details and results of our experimental evaluation, before some concluding remarks are given in Section 7.

2. Related work

2.1. Objective function mismatch (OFM) in autoencoder-based models

Metz *et al.* [5] demonstrate the presence of OFM in the context of unsupervised representation learning. They measure few-shot classification accuracy on representations obtained with a variational autoencoder, trained on the CIFAR-10 dataset. After an initial increase, they observe that the few-shot classification accuracy decreases in later stages of training the variational autoencoder. A more extensive study by Stuhr and Brauer [9] finds that OFM is present in several autoencoder-based models used for downstream image classification.

Objective function mismatch has also previously been observed in deep clustering models. Mrabah *et al.* [6] shows that the autoencoder-based Improved Deep Embedded Clustering (IDEC) [10] suffers from *Feature Drift* [6], which is defined as the cosine of the angle between the gradient of the clustering loss, and the gradient of the auxiliary loss. Feature Drift is thus closely related to OFM.

2.2. Deep clustering with autoencoders

Autoencoders are widely used building blocks in deep clustering models. A straightforward approach to integrating autoencoders in a deep clustering model is to use the autoencoder to pre-train the network, and then fine-tune it using only the clustering loss [3, 11, 12]. Alternatively, there are methods that retain the reconstruction loss during fine-tuning – either by minimizing it jointly with the clustering loss [2, 13, 14, 4], or by alternating between minimizing the reconstruction loss and clustering loss [15]. Autoencoders have also been used to encode self-expressiveness in subspace-based methods [16, 17]. Lastly, there are probabilistic methods to deep clustering that employ variational autoencoders in various manners [18, 19].

Deep Embedded Clustering (DEC) [3]. DEC is one of the first methods for clustering with deep neural networks, and a cornerstone method in deep clustering. It uses a multilayer perceptron (MLP), pre-trained as a stacked denoising autoencoder, and a clustering module based on soft-assignments to a set of centroids. DEC's loss function is constructed to force the distribution of soft cluster assignments closer to a target distribution, by means of the Kullback-Leibler (KL) divergence. The target distribution is constructed from the soft assignments, and designed to strengthen predictions and put more emphasis on high-confidence assignments. In Section 6.5, we investigate the performance of two variants of DEC, with and without the proposed UCOs.

2.3. Deep clustering without autoencoders

Several methods that do not use autoencoders in their architectures have also been developed in recent years. These include methods based on hierarchical clustering [20], subspace clustering [21, 22, 23], generative adversarial networks [24], and adversarial learning [25]. The recent success of self-supervised representation learning for images has also given rise to numerous methods for image clustering [26, 27]. However, these methods depend heavily on image augmentations and are thus not directly applicable to other data types. Finally, there are approaches to deep clustering that do not rely on any auxiliary model components or losses [28, 29]. In these models, the networks are trained end-to-end from random initializations by minimizing their respective clustering losses.

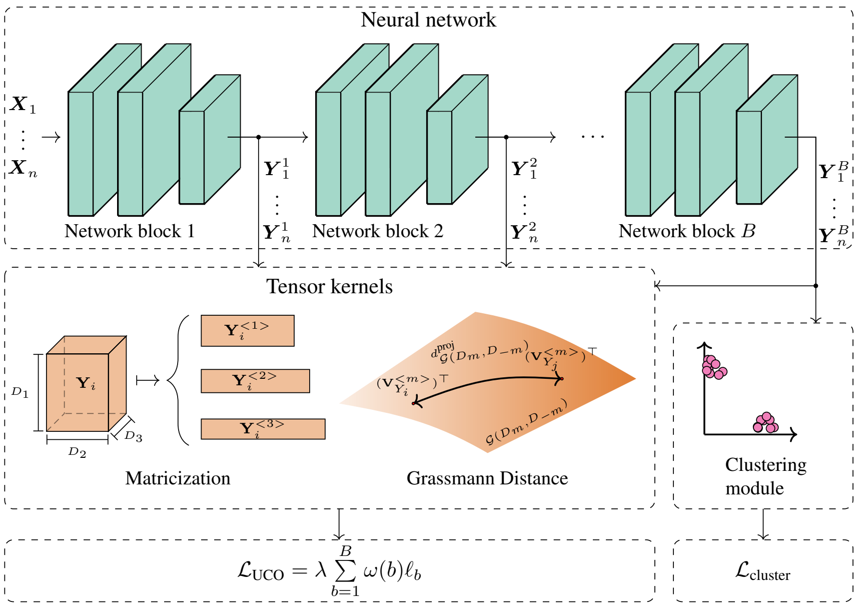


Figure 1: Deep clustering model augmented with the proposed unsupervised companion objectives (UCOs). This model uses tensor kernels to estimate the CS-divergence between clusters at the block outputs. The estimate is then used in the computation of the UCOs.

3. Unsupervised Companion Objectives

In deep clustering, the input observation $\mathbf{X}_i$, $i = 1, \ldots, n$ is transformed by an encoder network f, producing a hidden representation $\boldsymbol{z}_i$. The cluster membership vector is then determined by a clustering module, g, as $\boldsymbol{\alpha}_i = g(\boldsymbol{z}_i)$.

Since f is a neural network, we assume that it can be decomposed into consecutive computational *blocks*: $f = f^B \circ f^{B-1} \circ \cdots \circ f^1$, where f^b denotes block b. A block can be, for instance, a single layer, or a collection of adjacent layers. We let $\mathbf{Y}_i^b = f^b(\mathbf{Y}_i^{(b-1)})$ be the output of block b for observation i, where $\mathbf{Y}_i^0 = \mathbf{X}_i$. When we augment a deep clustering model with the UCOs, we attach companion objectives to the outputs of the blocks in the network, and minimize them alongside the main clustering objective. Figure 1 illustrates a deep clustering model with the UCOs attached to the neural network.

When minimized, the UCOs should enforce clusters to be separable and compact at the output of their respective blocks. Inspired by Deep Divergence-based Clustering (DDC) [28], we do this by maximizing an estimate of the Cauchy-Schwarz (CS) divergence between clusters [30]. For a set of k clusters characterized by k probability density functions $p_1, \dots, p_k$, the CS divergence between them at block b is given by:

$$D_{cs}^b = -\log\left(\binom{k}{2}^{-1} \sum_{\substack{i=1\\j>i}}^{k} \frac{\int p_i(\boldsymbol{Y}^b) p_j(\boldsymbol{Y}^b) \mathrm{d}\boldsymbol{Y}^b}{\sqrt{\int p_i(\boldsymbol{Y}^b)^2 \mathrm{d}\boldsymbol{Y}^b \int p_j(\boldsymbol{Y}^b)^2 \mathrm{d}\boldsymbol{Y}^b}}\right) \tag{1}$$

where $\boldsymbol{Y}^b$ is the output of block b. Following common practice in deep clustering, we assume that the number of clusters, k, is known [2, 3, 28][2].

D_{cs}^b can be maximized by minimizing the argument of the logarithm. Using a Gaussian kernel density estimate gives the following UCO for block b in the network:

$$\ell_b = \sum_{\substack{i=1\\j>i}}^{k} \frac{\binom{k}{2}^{-1} \sum\limits_{l,m=1}^{n} \alpha_{li}\alpha_{mj} k_\sigma^{\text{UCO}}(\boldsymbol{Y}_l^b, \boldsymbol{Y}_m^b)}{\sqrt{\sum\limits_{l,m=1}^{n} \alpha_{li}\alpha_{mi} k_\sigma^{\text{UCO}}(\boldsymbol{Y}_l^b, \boldsymbol{Y}_m^b) \sum\limits_{l,m=1}^{n} \alpha_{lj}\alpha_{mj} k_\sigma^{\text{UCO}}(\boldsymbol{Y}_l^b, \boldsymbol{Y}_m^b)}} \tag{2}$$

where α_{li} denotes element i of the cluster membership vector $\boldsymbol{\alpha}_l$, and $k_\sigma^{\text{UCO}}(\boldsymbol{Y}_l^b, \boldsymbol{Y}_m^b)$ is a Gaussian kernel evaluated at $(\boldsymbol{Y}_l^b, \boldsymbol{Y}_m^b)$. σ denotes the kernel width. Collecting the UCOs for all blocks in the network gives the loss:

$$\mathcal{L}_{\text{UCO}} = \sum_{b=1}^{B} w_b \ell_b = \lambda \sum_{b=1}^{B} \omega(b) \ell_b \tag{3}$$

where w_b is the weight of the UCO attached to block b, and B is the number of blocks. To avoid having to specify the weight for each block individually, we compute w_b as $w_b = \lambda \cdot \omega(b)$. Here λ is a hyperparameter that specifies the overall weight for the UCOs, and $\omega : \{1, \dots, B\} \to [0, 1]$ is a function that computes the relative weight of the UCO attached to block b.

[2]The case when k is unknown, as in [27], is an interesting direction in deep clustering. However, it is beyond the scope of this paper.

The final loss used to train the model is then:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cluster}} + \mathcal{L}_{\text{UCO}}. \tag{4}$$

Connection to Deep Divergence-based Clustering [28]. The UCOs are related to DDC [28] as they both minimize the CS divergence between clusters. In DDC however, the CS divergence is estimated with hidden representations that are computed within the clustering module. These hidden representations, $\boldsymbol{h}_i$ are obtained by transforming the encoder outputs, $\mathbf{Y}_i^B$ with a fully-connected layer. A second fully-connected layer is then applied to the hidden representations, producing the cluster membership vectors $\boldsymbol{\alpha}_i$.

DDC's clustering loss consists of three terms:

$$\mathcal{L}_{\text{cluster}}^{\text{DDC}} = \mathcal{L}_{\text{DDC,1}} + \mathcal{L}_{\text{DDC,2}} + \mathcal{L}_{\text{DDC,3}}. \tag{5}$$

The first term is similar to the UCOs, and minimizes the CS divergence between clusters in the space of hidden representations:

$$\mathcal{L}_{\text{DDC,1}} = \sum_{\substack{i=1\\j>i}}^{k} \frac{\binom{k}{2}^{-1} \sum\limits_{l,m=1}^{n} \alpha_{li}\alpha_{mj} k_\sigma(\boldsymbol{h}_l, \boldsymbol{h}_m)}{\sqrt{\sum\limits_{l,m=1}^{n} \alpha_{li}\alpha_{mi} k_\sigma(\boldsymbol{h}_l, \boldsymbol{h}_m) \sum\limits_{l,m=1}^{n} \alpha_{lj}\alpha_{mj} k_\sigma(\boldsymbol{h}_l, \boldsymbol{h}_m)}} \tag{6}$$

where k is the number of clusters, α_{ij} is the soft cluster assignment for observation i to cluster j, $\boldsymbol{h}_i$ is the hidden representation for the i-th observation, and k_σ is a Gaussian kernel with kernel width σ. As with the UCOs, minimizing this loss term results in clusters that are separable and compact in the hidden space.

The second term encourages the cluster membership vectors ($\boldsymbol{\alpha}_i$) for different observations to be orthogonal:

$$\mathcal{L}_{\text{DDC,2}} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_j. \tag{7}$$

The last term is also based on the CS divergence and pushes the cluster membership vectors closer to the corners of the standard simplex in $\mathbb{R}^k$:

$$\mathcal{L}_{\text{DDC},3} = \sum_{\substack{i=1 \\ j>i}}^{k} \frac{\binom{k}{2}^{-1} \sum\limits_{l,m=1}^{n} \eta_{li}\eta_{mj} k_\sigma(\boldsymbol{h}_l, \boldsymbol{h}_m)}{\sqrt{\sum\limits_{l,m=1}^{n} \eta_{li}\eta_{mi} k_\sigma(\boldsymbol{h}_l, \boldsymbol{h}_m) \sum\limits_{l,m=1}^{n} \eta_{lj}\eta_{mj} k_\sigma(\boldsymbol{h}_l, \boldsymbol{h}_m)}} \tag{8}$$

where $\eta_{ij} = \exp(-||\boldsymbol{\alpha}_i - \boldsymbol{e}_j||^2)$, and $\boldsymbol{e}_j$ is a k-dimensional one-hot vector pointing to the j-th corner of the simplex.

4. Tensor Kernels

The UCOs in Eq. (2) depend on a kernel function (k_σ^{UCO}), which computes the similarity between block-outputs for different inputs. However, these outputs are not necessarily vectors. In CNNs for instance, the intermediate representations have two spatial dimensions, in addition to the feature dimension. The Gaussian kernel with Euclidean distance between vectors is thus not directly applicable to the block-outputs (without vectorizing them first). To address this issue, we consider the feature maps produced by a network block as rank-3 tensors. Building on Grassmannian learning [31] and tensor kernels [7], then allows us to specify a tensor kernel for the proposed UCOs.

4.1. The naïve kernel

For vectors we have the Gaussian kernel:

$$k_\sigma(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{y}||^2}{2\sigma^2}\right) = \prod_{i=1}^{D_1} \exp\left(-\frac{(x_i - y_i)^2}{2\sigma^2}\right) \tag{9}$$

where x_i (y_i) refers to element i in the vector $\boldsymbol{x}$ ($\boldsymbol{y}$). Generalizing this kernel to tensors $\mathbf{X}$ and $\mathbf{Y}$ of rank r gives the naïve kernel [7]:

$$k_\sigma^{\text{naïve}}(\mathbf{X}, \mathbf{Y}) = \prod_{i_1=1}^{D_1} \cdots \prod_{i_r=1}^{D_r} \exp\left(-\frac{(X_{i_1 \cdots i_r} - Y_{i_1 \cdots i_r})^2}{2\sigma^2}\right) \tag{10}$$

where $X_{i_1 \cdots i_r}$ ($Y_{i_1 \cdots i_r}$) denotes the element of $\mathbf{X}$ ($\mathbf{Y}$) with indices $i_1, \ldots, i_r$, and $D_1, \ldots, D_r$ denote the number of elements along each dimension.

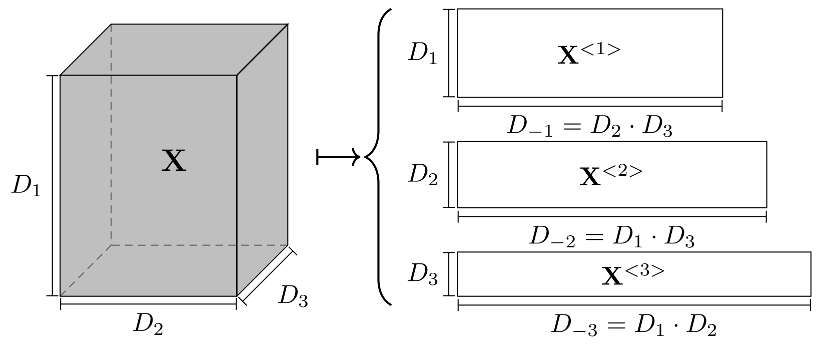


Figure 2: Matricization of a rank-3 tensor $\mathbf{X}$.

This kernel is equivalent to first vectorizing the representations, and then applying a Gaussian kernel using the Euclidean distance between the vectorized representations. However, the intermediate representations often lie on a low-dimensional manifolds where the Euclidean distance fails to be a good measure of dissimilarity. A potential reason for this is that the naïve kernel ignores the *structure* of the input tensors [7] – such as the spatial structure of the intermediate representations in CNNs. This means that the kernel is invariant to a fixed permutation rule P:

$$k_\sigma^{\text{naïve}}(\mathbf{X}, \mathbf{Y}) = k_\sigma^{\text{naïve}}(P(\mathbf{X}), P(\mathbf{Y})). \tag{11}$$

This effect can be especially destructive for images, for instance, since images can be transformed beyond recognition by permuting the spatial indices.

4.2. Structure-exploiting tensor kernels

The shortcomings of the naïve kernel outlined above calls for a kernel that takes the tensors' structure into account. The kernels are therefore formulated using the *matricizations* of input tensors. The matricization of a tensor $\mathbf{X}$ along dimension m is obtained by vectorizing along all dimensions of $\mathbf{X}$, except dimension m. This results in a matrix with shape (D_m, D_{-m}) where $D_{-m} = \frac{1}{D_m} \prod_{l=1}^{r} D_l$ (see Figure 2).

The matricizations along the tensor dimensions can then be used to form the

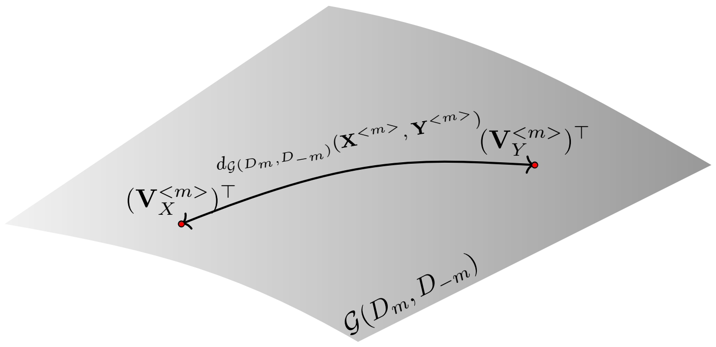

$d_{\mathcal{G}(D_m,D_{-m})}(\mathbf{X}^{<m>},\mathbf{Y}^{<m>})$
$(\mathbf{V}_Y^{<m>})^\top$
$(\mathbf{V}_X^{<m>})^\top$
$\mathcal{G}(D_m,D_{-m})$

Figure 3: The distance $d_{\mathcal{G}(D_m,D_{-m})}(\mathbf{X}^{<m>},\mathbf{Y}^{<m>})$ between two matricizations $\mathbf{X}^{<m>}$ and $\mathbf{Y}^{<m>}$ on the Grassmann manifold $\mathcal{G}(D_m,D_{-m})$. Since the Grassmann manifold consists of linear subspaces, the matricizations are represented by the respective orthonormal representations $(\mathbf{V}_X^{<m>})^\top$ and $(\mathbf{V}_Y^{<m>})^\top$.

tensor kernel:

$$k_\sigma^{\text{tensor}}(\mathbf{X},\mathbf{Y}) = \prod_{m=1}^{r} k_\sigma^m(\mathbf{X}^{<m>},\mathbf{Y}^{<m>}) \tag{12}$$

$$= \prod_{m=1}^{r} \exp\left(-\frac{d_{\mathcal{G}(D_m,D_{-m})}(\mathbf{X}^{<m>},\mathbf{Y}^{<m>})^2}{2\sigma^2}\right) \tag{13}$$

where $\mathbf{X}^{<m>}$ ($\mathbf{Y}^{<m>}$) is the matricization of $\mathbf{X}$ ($\mathbf{Y}$) along dimension m. In order to specify a suitable distance function between the matricizations, we consider the span of their rows as points on the Grassmann manifold, $\mathcal{G}(D_m,D_{-m})$, which consists of all D_m dimensional linear subspaces of $\mathbb{R}^{D_{-m}}$. This is a central concept in *Grassmannian learning* [32] – a field which has shown promising results, but is still in its early stages. Specifically, we exploit the one-to-one correspondence between linear subspaces and their orthogonal projection operators to define our distance function on the Grassmannian manifold [7]. Figure 3 illustrates the distance function on the Grassmann manifold. The orthogonal projection operator is given by

$$\mathbf{\Pi}_X^{<m>} = \mathbf{V}_X^{<m>}(\mathbf{V}_X^{<m>})^\top \tag{14}$$

where $\mathbf{V}_X^{<m>}$ is obtained through the singular value decomposition (SVD):

$$\mathbf{X}^{<m>} = \mathbf{U}_X^{<m>}\mathbf{\Sigma}_X^{<m>}(\mathbf{V}_X^{<m>})^\top. \tag{15}$$

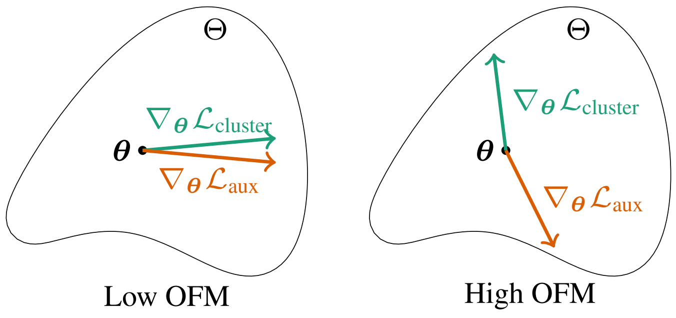

Θ
$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{cluster}}$
$\boldsymbol{\theta}$
$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{aux}}$
Low OFM
Θ
$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{cluster}}$
$\boldsymbol{\theta}$
$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{aux}}$
High OFM

Figure 4: Illustration of OFM for different loss gradients. The OFM is determined by the angles between the gradients of the clustering loss and the auxiliary loss wrt. the encoder parameters $\boldsymbol{\theta} \in \Theta$.

Considering the Frobenius norm between projection operators then gives the distance function:

$$
\begin{aligned}
&d^{\text{proj}}_{\mathcal{G}(D_m, D_{-m})}(\mathbf{X}^{<m>}, \mathbf{Y}^{<m>}) \\
&\quad = ||\mathbf{\Pi}_X^{<m>} - \mathbf{\Pi}_Y^{<m>}||_F \qquad (16) \\
&\quad = \sqrt{2(D_m - \operatorname{Tr}((\mathbf{V}_Y^{<m>})^\top \mathbf{V}_X^{<m>} (\mathbf{V}_X^{<m>})^\top \mathbf{V}_Y^{<m>}))}. \qquad (17)
\end{aligned}
$$

where $||\cdot||_F$ denotes the Frobenius norm. Inserting $d^{\text{proj}}_{\mathcal{G}(D_m, D_{-m})}(\mathbf{X}^{<m>}, \mathbf{Y}^{<m>})$ into (13) allows us to compute a structure-exploiting tensor kernel $k_\sigma^{\text{tensor}}(\mathbf{X}, \mathbf{Y})$, which can then be included in the proposed UCOs by setting $k_\sigma^{\text{UCO}} = k_\sigma^{\text{tensor}}$ in (2).

We note that the projection distance gives rise to a positive semi-definite kernel. This allows the UCOs to be interpreted as a cosine distance between the mean of intermediate representations from different clusters, in the Reproducing Kernel Hilbert Space induced by the tensor kernel in Eq. (13). However, a thorough analysis of this property is left to future work.

5. Objective Function Mismatch

Intuitively, the objective function mismatch (OFM) should be high if the clustering loss and auxiliary loss disagree on how the encoder network should be optimized. Correspondingly, it should be low if the losses agree on the optimization. Thus, it is natural to consider the gradient of the losses wrt. the encoder's parameters, as these guide the direction of optimization in the parameter space. Let $\boldsymbol{\theta}$ denote the network parameters at a particular step in the optimization procedure. The OFM between the clustering loss $\mathcal{L}_{\text{cluster}}$ and an auxiliary loss (*e.g.* reconstruction or UCOs) is then be measured by:

$$\text{OFM}(\mathcal{L}_{\text{cluster}}, \mathcal{L}_{\text{aux}})(\boldsymbol{\theta}) = \frac{1 - \cos(\nabla_{\theta}\mathcal{L}_{\text{cluster}}, \nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{aux}})}{2} \tag{18}$$

$$= \frac{1}{2} - \frac{(\nabla_{\theta}\mathcal{L}_{\text{cluster}})^{\top}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{aux}}}{2||\nabla_{\theta}\mathcal{L}_{\text{cluster}}|| \cdot ||\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{aux}}||}. \tag{19}$$

Here, the cosine measures the agreement between directions of the gradients, and the affine transformation transforms the measure to lie in $[0, 1]$, and such that higher values indicate larger mismatch. The measure of OFM is illustrated graphically in Figure 4. We note that this measure is related to the Feature Drift [6], but includes an affine transformation which makes the measure more intuitive, and directly indicative of a *mismatch* between the losses.

6. Experiments

In this section we report the effects of augmenting two different deep clustering models with the proposed UCOs – both in terms of clustering performance, and the OFM between the main clustering objective and the auxiliary objective. Further, we investigate the effect of varying the weighting hyperparameters, ω and λ, in the UCOs.

6.1. Setup

Models. Our experiments are based on DDC [28], as it is an excellent general-purpose deep clustering method, which has demonstrated state-of-the-art clustering performance on images [28], time series [33], and several types of multi-view data [34]. In Section 6.5 we also show some results with DEC-based models.

We augment DDC with UCOs based on tensor kernels, and refer to the resulting model as DDC-UCO$_T$. To evaluate the effects of the structure-exploiting

Table 1: Details of the benchmark datasets.

Name	Type	Dimension	Samples	Classes
MNIST [35]	Images	$28 \times 28 \times 1$	60000	10
Fashion-MNIST [36]	Images	$28 \times 28 \times 1$	60000	10
COIL-100 [37]	Images	$128 \times 128 \times 3$	7200	100
USPS	Images	$16 \times 16 \times 1$	9298	10
Reuters [38]	Vectors	2000	216000	4
10×73k[39]	Vectors	720	73233	8

tensor kernels, we also include a model with UCOs based on vectorization and naïve kernels, referred to as DDC-UCO$_N$.

Finally, we train DDC with a decoder and a reconstruction loss (we refer to this model as DDC-AE). This allows us to accurately compare DDC-UCO$_{N/T}$ to an analogous autoencoder-based model.

We train the models with randomly initialized parameters, using their respective loss functions. The batch size is set to 120, and we use the Adam optimizer with default parameters. Following [28], we let the σ hyperparameter be 15% of the median distance between observations in a batch. This rule of thumb is used for both the UCOs and the clustering module.

Due to the difficulty of hyperparameter validation in unsupervised learning, we compute the UCO weights with $\omega(b) = 1$ (constant) and $\lambda = 0.01$ for all datasets. This choice is further studied in Section 6.4.

Datasets. We test the models on 6 different benchmark datasets:

1. MNIST [35]: Grayscale images of handwritten digits.

2. Fashion-MNIST [36]: Grayscale images of clothing items.

3. COIL-100 [37]: RGB Images of 100 common objects depicted from 72 different angles.

4. USPS: Grayscale images of handwritten digits.

5. Reuters [38]: TF-IDF features from news stories[3].

6. 10$\times$73k[39]: Feature vectors for RNA-transcripts from different cell types.

More details on the datasets can be found in Table 1

Evaluation. We train each model 20 times per dataset, and report the performance of the run which resulted in the lowest value of the total loss function (Eq. (4)). This is the same evaluation procedure as in DDC [28].

To measure the clustering performance of the different algorithms, we use the unsupervised clustering accuracy (ACC) and the normalized mutual information (NMI). Both ACC and NMI are bounded in $[0, 1]$, and higher values indicate better clusterings, with respect to the ground-truth labels.

6.2. Clustering results

The clustering results in Table 2 show that adding the UCOs to DDC improves the overall performance of the model. Furthermore, DDC-UCO$_T$ consistently outperforms DDC-UCO$_N$, indicating that the structure-exploiting tensor kernels are suitable for quantifying the similarities between the intermediate representations of the network.

Interestingly, we also observe that adding a decoder and reconstruction loss to DDC only improves ACC and NMI on COIL-100, as well as the NMI on USPS. On MNIST and F-MNIST, the performance of DDC-AE is significantly worse than the original DDC, indicating that the addition of an autoencoder can be harmful to the performance of DDC.

Figure 5 shows intermediate MNIST representations after the first block, projected to 2 dimensions with t-SNE [40]. From these plots we make the following observations:

- The clusters of 4 , 5 and 7 are all more compact for the representations produced by DDC-UCO$_T$, compared to the others.

[3]As in [28], we select 54000 samples from each class to balance the dataset.

Table 2: Clustering results for DDC-based models on the benchmark datasets. Standard deviations obtained by bootstrapping are shown in parentheses. The best result for each base model is highlighted in **bold**, and results that are within one standard deviation of the best are underlined.

	MNIST		F-MNIST	
	ACC	NMI	ACC	NMI
DDC	88.2 (2.1)	87.7 (2.3)	48.6 (8.1)	44.3 (6.0)
DDC-AE	87.9 (1.3)	86.2 (0.7)	60.3 (4.7)	55.1 (3.8)
DDC-UCO$_N$	85.5 (4.7)	82.4 (3.3)	**65.5** (5.0)	59.1 (3.6)
DDC-UCO$_T$	**96.0** (6.4)	**91.7** (5.2)	65.3 (4.1)	**59.2** (3.4)
	COIL-100		**USPS**	
	ACC	NMI	ACC	NMI
DDC	58.0 (1.1)	82.6 (0.2)	67.9 (2.8)	70.1 (1.6)
DDC-AE	61.2 (1.3)	**84.4** (0.2)	72.1 (5.4)	71.0 (4.9)
DDC-UCO$_N$	58.5 (1.4)	80.8 (0.3)	75.5 (4.8)	76.6 (3.3)
DDC-UCO$_T$	**62.9** (1.1)	83.8 (0.5)	**75.6** (3.1)	**77.4** (2.5)
	Reuters		**10×73k**	
	ACC	NMI	ACC	NMI
DDC	50.9 (3.6)	24.2 (5.8)	62.3 (1.0)	55.2 (2.3)
DDC-AE	37.2 (5.3)	12.1 (6.1)	78.6 (3.7)	74.4 (3.1)
DDC-UCO$_N$	55.5 (4.8)	30.4 (4.2)	73.1 (3.2)	70.2 (1.9)
DDC-UCO$_T$	**61.2** (5.3)	**35.6** (4.8)	**81.1** (1.4)	**76.3** (0.9)

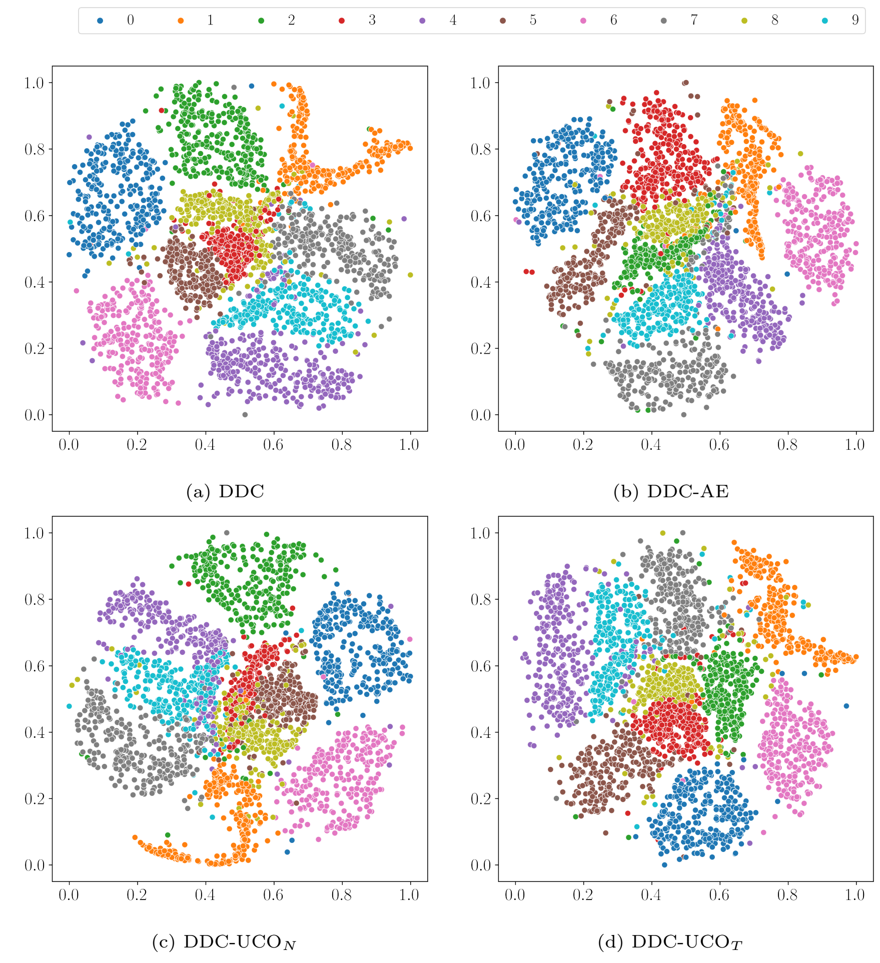
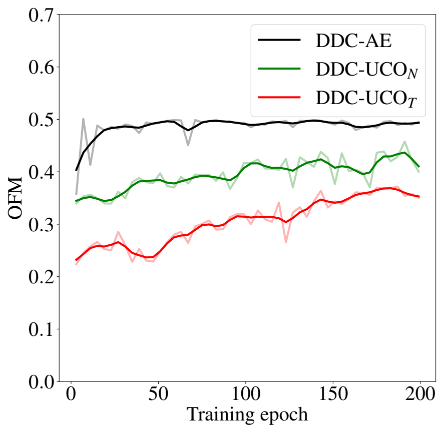
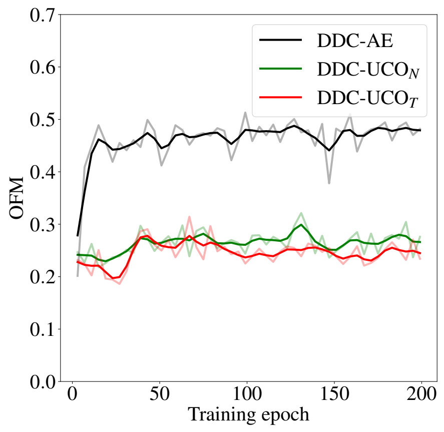


Figure 5: *t*-SNE [40] plot of intermediate MNIST representations from the first network block $(\mathbf{Y}_1^1, \ldots, \mathbf{Y}_n^1)$.



(a) MNIST



(b) COIL-100

Figure 6: Objective function mismatch during training of DDC-AE, DDC-UCO$_N$, and DDC-UCO$_T$. The OFM is is significantly lower for DDC-UCO$_N$ and DDC-UCO$_T$, compared to DDC-AE on both datasets.

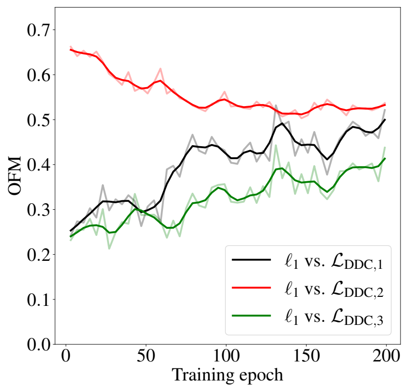
- Similar looking digits, such as 4 and 9, as well as 3, 5, and 8, are better separated in the DDC-UCO$_T$ representations, compared to the other models.

These observations illustrate that the UCOs indeed encourage compact and separable clusters in the space of intermediate representations. This results in better separability for the challenging cases where different digits look similar to each other.

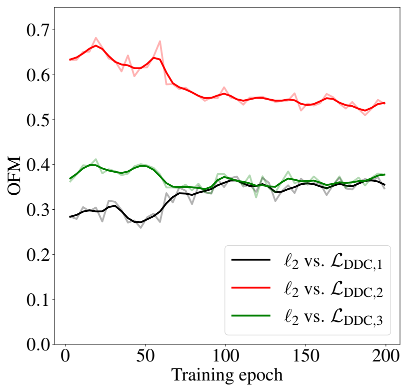
6.3. Objective function mismatch

Figure 6 shows the OFM during training of DDC-AE, DDC-UCO$_N$, and DDC-UCO$_T$ on MNIST and COIL-100. These plots show that the OFM is significantly lower for both DDC-UCO$_N$ and DDC-UCO$_T$ during training, when compared to DDC-AE. The UCOs thus “agree” more with the clustering objective, resulting in both a lower OFM, and improved clustering performance (as can be seen in Table 2).

When comparing DDC-UCO$_N$ and DDC-UCO$_T$, we either see that the OFM



(a) UCO at block 1.



(b) UCO at block 2.

Figure 7: OFM between the UCOs and the three terms in DDC's clustering loss, during training on MNIST.

is similar for the two models (COIL-100), or that the OFM for DDC-UCO$_T$ is lower than for DDC-UCO$_N$ (MNIST).

Lastly, we observe a slightly increasing trend in the OFM for DDC-UCO$_N$ and DDC-UCO$_T$ on MNIST. This increase in OFM during training is caused by a mismatch between the UCOs and the CS divergence-based loss terms in DDC. This can be observed in Figure 7, which shows that the OFM between the UCOs (ℓ_1 and ℓ_2), and DDC's CS divergence terms ($\mathcal{L}_{\text{DDC},1}$ and $\mathcal{L}_{\text{DDC},3}$) increases during training. Both the UCOs and DDC's losses encourage separable and compact clusters for the intermediate and hidden representations.

However, we hypothesize that the gradients contributing to separability and compactness differ for different representations, due to the non-linearity of the network blocks. In the early stages of training, the model focuses on simpler lower-level groupings in the data. The UCOs and DDC's loss terms thus tend to agree more on what constitutes separable and compact representations, resulting in a relatively low OFM. However, as the training progresses, and the network begins to refine the clusters to make them as separable and compact as possible, the non-linearities cause the gradients to become increasingly different, resulting in an increased OFM. We further hypothesize that this effect is stronger for the

Table 3: Clustering results (ACC) for DDC-UCO$_{N/T}$ with different UCO weighting methods.

		DDC-UCO$_N$			DDC-UCO$_T$		
Dataset	$\omega(\cdot)\downarrow\quad\lambda\rightarrow$	0.001	0.010	0.100	0.001	0.010	0.100
MNIST	Constant	85.2	85.5	87.4	86.6	**96.0**	87.2
	Exponential	**88.2**	87.1	87.1	87.3	85.7	87.7
	Linear	76.0	81.3	82.3	87.8	87.6	88.6
COIL-100	Constant	60.5	58.5	63.1	59.6	62.9	61.7
	Exponential	59.0	60.4	60.8	**65.9**	63.0	62.7
	Linear	61.9	**63.7**	60.5	60.1	61.4	58.5
F-MNIST	Constant	61.8	**65.5**	49.9	59.5	**65.3**	51.0
	Exponential	52.8	51.7	60.0	59.1	59.1	60.1
	Linear	55.4	59.1	46.1	59.8	60.3	50.0
USPS	Constant	65.4	75.5	66.8	69.6	**75.6**	72.9
	Exponential	69.4	58.3	66.3	72.6	66.8	73.3
	Linear	66.5	69.3	**76.1**	74.4	73.5	69.6
Reuters	Constant	45.8	54.4	66.3	58.4	55.1	62.8
	Exponential	60.2	66.8	**75.6**	50.0	57.2	62.7
	Linear	48.1	63.1	50.6	60.3	**71.7**	58.3
10×73k	Constant	82.4	73.1	87.9	78.4	81.1	**86.9**
	Exponential	81.2	84.6	**89.2**	78.8	70.4	86.2
	Linear	82.6	78.8	78.0	81.7	82.1	85.1

separability between clusters, as it is a more global property, and is thus affected more by the non-linear blocks, compared to compactness.

6.4. UCO weighting strategy

To validate our chosen UCO weighting strategy (choice of ω and λ), we train DDC-UCO$_{N/T}$ with $\lambda \in \{0.001, 0.01, 0.1\}$, and the following three ω functions: (i) $\omega(b) = 1$ (Constant); (ii) $\omega(b) = 10^{B-b}$ (Exponential); and (iii) $\omega(b) = b/B$ (Linear).

The resulting clustering accuracies for the different configurations are listed in Table 3. These results show that DDC-UCO$_{N/T}$ are not sensitive to the choice of ω and λ. This is an important property as labeled validation data is generally not available in unsupervised learning.

We note that, for some configurations on F-MNIST and Reuters, the performance degrades significantly when $\lambda = 0.1$ or $\lambda 0.001$, indicating that the influence of the UCOs is too strong or too weak in these cases.

6.5. Experiments with other base models

In order to evaluate the effect of the UCOs on other deep clustering models, we implement two variants of the well-known DEC model [3]. The first is the original DEC model with an MLP encoder. The second model, DEC-Conv, uses the same clustering module as DEC, but has a CNN encoder instead. We also train these variants with a decoder and reconstruction loss during fine-tuning (as is done in Improved DEC [2] and Deep Convolutional Embedded Clustering (DCEC) [10], respectively). Finally, we create two new models by augmenting the two variants with our UCOs. Since CNNs are designed for images and MLPs are designed for vectors, we evaluate DEC-Conv on the image datasets (MNIST, F-MNIST, COIL-100, USPS), and DEC on the vector datasets (Reuters, 10$\times$73k).

The results in Table 4 show that the DEC-based models also benefit from the UCOs. This indicates that the UCOs work well with different clustering modules – not only DDC. However, the improvement in clustering performance is not as large for DEC and DEC-Conv, as for DDC. This discrepancy is likely due to the difference between DEC's clustering objective, and the UCOs. In DDC, the clustering objective is partially based on the CS divergence between clusters. This is the same divergence measure that we use in the UCOs. DEC's clustering objective however, is based on a Kullback-Leibler divergence between densities of cluster assignments. It is therefore more dissimilar to the UCOs, compared to the clustering objective in DDC.

Table 4: Clustering results with the models based on DEC-Conv (image datasets) and DEC (vector datasets). Standard deviations obtained by bootstrapping are shown in parentheses. The best result for each base model is highlighted in **bold**, and results that are within one standard deviation of the best are underlined.

	MNIST		F-MNIST	
	ACC	NMI	ACC	NMI
DEC-Conv	78.3 (5.3)	73.6 (2.8)	54.7 (1.4)	60.1 (0.9)
DCEC	77.8 (1.3)	72.6 (1.2)	54.3 (2.7)	60.0 (0.8)
DEC-Conv-UCO$_N$	80.0 (1.1)	77.0 (0.5)	57.0 (2.9)	60.8 (1.0)
DEC-Conv-UCO$_T$	**81.2** (0.8)	**79.1** (0.6)	**59.8** (3.5)	**62.0** (1.3)
	COIL-100		USPS	
	ACC	NMI	ACC	NMI
DEC-Conv	**50.3** (3.7)	**74.5** (2.2)	73.6 (0.3)	73.1 (0.2)
DCEC	47.6 (2.7)	71.7 (2.0)	75.3 (0.5)	75.5 (0.6)
DEC-Conv-UCO$_N$	42.9 (2.5)	70.0 (1.8)	76.9 (0.7)	78.1 (1.1)
DEC-Conv-UCO$_T$	38.9 (0.7)	68.0 (1.0)	**78.0** (0.6)	**80.7** (0.9)
	Reuters		10×73k	
	ACC	NMI	ACC	NMI
DEC	68.6 (1.1)	**41.7** (1.7)	74.9 (0.7)	74.6 (0.8)
IDEC	66.1 (1.1)	35.9 (2.5)	73.0 (0.6)	73.2 (0.7)
DEC-UCO$_N$	64.1 (2.9)	35.4 (3.5)	71.4 (0.4)	69.5 (0.6)
DEC-UCO$_T$	**69.3** (0.8)	40.0 (1.1)	**75.8** (0.7)	**76.2** (1.1)

7. Conclusion

This paper sheds new light on the problem of objective function mismatch in deep clustering. The results of our experiments show that coupling a deep clustering model with an autoencoder can cause a significant amount of mismatch between the clustering objective and the reconstruction objective – possibly leading to reduced clustering performance.

To address the issue of OFM in deep clustering, we introduced the unsupervised companion objectives (UCOs). These are auxiliary objectives that – when compared to a decoder and reconstruction loss – lead to a lower OFM with the main clustering objective. The UCOs employ structure-exploiting tensor kernels, addressing the drawbacks of the naïve vectorization-based kernel for tensorial intermediate representations, such as those produced by convolutional neural networks. Our experiments with DDC and DEC show that adding the tensor kernel-based UCOs improves the clustering performance of the models, outperforming the analogous autoencoder-based approach.

Acknowledgements

This work was financially supported by the Research Council of Norway (RCN), through its Centre for Research-based Innovation funding scheme (Visual Intelligence, grant no. 309439), and Consortium Partners. It was further funded by RCN FRIPRO grant no. 315029, RCN IKTPLUSS grant no. 303514, and the UiT Thematic Initiative “Data-Driven Health Technology”.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.

[2] X. Guo, L. Gao, X. Liu, J. Yin, Improved Deep Embedded Clustering with Local Structure Preservation, in: IJCAI, 2017.

[3] J. Xie, R. Girshick, A. Farhadi, Unsupervised Deep Embedding for Clustering Analysis, in: ICML, 2016.

[4] H. Lu, C. Chen, H. Wei, Z. Ma, K. Jiang, Y. Wang, Improved deep convolutional embedded clustering with re-selectable sample training, Pattern Recognition 127 (2022) 108611.

[5] L. Metz, N. Maheswaranathan, B. Cheung, J. Sohl-Dickstein, Meta-Learning Update Rules for Unsupervised Representation Learning, in: ICLR, 2019.

[6] N. Mrabah, M. Bouguessa, R. Ksantini, Adversarial Deep Embedded Clustering: On a better trade-off between Feature Randomness and Feature Drift, IEEE Transactions on Knowledge and Data Engineering (2020) 1–1.

[7] M. Signoretto, Kernels and Tensors for Structured Data Modelling, Ph.D. thesis, Katholieke Universiteit Leuven – Faculty of Engineering, 2011.

[8] D. J. Trosten, R. Jenssen, M. C. Kampffmeyer, Reducing Objective Function Mismatch in Deep Clustering with the Unsupervised Companion Objective, in: NLDL, 2021.

[9] B. Stuhr, J. Brauer, Don't miss the Mismatch: Investigating the Objective Function Mismatch for Unsupervised Representation Learning, arXiv:2009.02383 [cs] (2020).

[10] X. Guo, X. Liu, E. Zhu, J. Yin, Deep Clustering with Convolutional Autoencoders, in: ICONIP, 2017.

[11] Y. Ren, N. Wang, M. Li, Z. Xu, Deep density-based image clustering, Knowledge-Based Systems 197 (2020) 105841.

[12] S. Affeldt, L. Labiod, M. Nadif, Spectral clustering via ensemble deep autoencoder learning (SC-EDAE), Pattern Recognition 108 (2020) 107522.

[13] A. Boubekki, M. Kampffmeyer, U. Brefeld, R. Jenssen, Joint optimization of an autoencoder for clustering and embedding, Machine Learning 110 (2021) 1901–1937.

[14] J. Cai, S. Wang, C. Xu, W. Guo, Unsupervised deep clustering via contractive feature representation and focal loss, Pattern Recognition 123 (2022) 108386.

[15] B. Yang, X. Fu, N. D. Sidiropoulos, M. Hong, Towards K-Means-Friendly Spaces: Simultaneous Deep Learning and Clustering, in: ICML, 2017.

[16] M. Abavisani, D. N. Metaxas, A. Naghizadeh, V. M. Patel, Deep Subspace Clustering with Data Augmentation, in: NeurIPS, 2020.

[17] S. Baek, G. Yoon, J. Song, S. M. Yoon, Deep self-representative subspace clustering network, Pattern Recognition 118 (2021) 108041.

[18] X. Li, Z. Chen, L. K. M. Poon, N. L. Zhang, Learning Latent Superstructures in Variational Autoencoders for Deep Multidimensional Clustering, in: ICLR, 2019.

[19] L. Yang, N.-M. Cheung, J. Li, J. Fang, Deep Clustering by Gaussian Mixture Variational Autoencoders With Graph Embedding, in: ICCV, 2019.

[20] J. Yang, D. Parikh, D. Batra, Joint Unsupervised Learning of Deep Representations and Image Clusters, in: CVPR, 2016.

[21] X. Peng, S. Xiao, J. Feng, W.-Y. Yau, Z. Yi, Deep Subspace Clustering with Sparsity Prior, in: IJCAI, 2016.

[22] X. Peng, J. Feng, J. T. Zhou, Y. Lei, S. Yan, Deep Subspace Clustering, IEEE Transactions on Neural Networks and Learning Systems (2020) 1–13.

[23] S. Zhang, C. You, R. Vidal, C.-G. Li, Learning a Self-Expressive Network for Subspace Clustering, in: CVPR, 2021.

[24] J. T. Springenberg, Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks., in: ICLR, 2016.

[25] X. Yang, C. Deng, K. Wei, J. Yan, W. Liu, Adversarial Learning for Robust Deep Clustering, in: NeurIPS, 2020.

[26] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, L. Van Gool, SCAN: Learning to Classify Images Without Labels, in: ECCV, 2020.

[27] M. Ronen, S. E. Finder, O. Freifeld, DeepDPM: Deep Clustering With an Unknown Number of Clusters, in: CVPR, 2022.

[28] M. Kampffmeyer, S. Løkse, F. M. Bianchi, L. Livi, A.-B. Salberg, R. Jenssen, Deep Divergence-based Approach to Clustering, Neural Networks 113 (2019) 91–101.

[29] J. Wang, L. Wang, J. Jiang, Preserving similarity order for unsupervised clustering, Pattern Recognition 128 (2022) 108670.

[30] R. Jenssen, J. C. Principe, D. Erdogmus, T. Eltoft, The Cauchy–Schwarz divergence and Parzen windowing: Connections to graph theory and Mercer kernels, Journal of the Franklin Institute 343 (2006) 614–629.

[31] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, M. Harandi, Kernel Methods on Riemannian Manifolds with Gaussian RBF Kernels, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (2015) 2464–2477.

[32] J. Zhang, G. Zhu, R. W. Heath Jr., K. Huang, Grassmannian Learning: Embedding Geometry Awareness in Shallow and Deep Learning, arXiv:1808.02229 [cs, eess, math, stat] (2018).

[33] D. J. Trosten, A. S. Strauman, M. Kampffmeyer, R. Jenssen, Recurrent Deep Divergence-Based Clustering for Simultaneous Feature Learning and Clustering of Variable Length Time Series, in: ICASSP, 2019.

[34] D. J. Trosten, S. Løkse, R. Jenssen, M. Kampffmeyer, Reconsidering Representation Alignment for Multi-view Clustering, in: CVPR, 2021.

[35] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (1998) 2278–2324.

[36] H. Xiao, K. Rasul, R. Vollgraf, Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms, arXiv:1708.07747 [cs, stat] (2017).

[37] S. A. Nene, S. K. Nayar, H. Murase, Columbia Object Image Library (COIL-100), Techincal Report CUCS-006-96, 1996.

[38] D. D. Lewis, Y. Yang, T. G. Rose, F. Li, RCV1: A New Benchmark Collection for Text Categorization Research, Journal of Machine Learning Research (2004) 37.

[39] G. X. Y. Zheng, J. M. Terry, P. Belgrader, P. Ryvkin, Z. W. Bent, R. Wilson, S. B. Ziraldo, T. D. Wheeler, G. P. McDermott, J. Zhu, M. T. Gregory, J. Shuga, L. Montesclaros, J. G. Underwood, D. A. Masquelier, S. Y. Nishimura, M. Schnall-Levin, P. W. Wyatt, C. M. Hindson, R. Bharadwaj, A. Wong, K. D. Ness, L. W. Beppu, H. J. Deeg, C. McFarland, K. R. Loeb, W. J. Valente, N. G. Ericson, E. A. Stevens, J. P. Radich, T. S. Mikkelsen, B. J. Hindson, J. H. Bielas, Massively parallel digital transcriptional profiling of single cells, Nature Communications 8 (2017) 14049.

[40] L. van der Maaten, G. Hinton, Visualizing Data using t-SNE, Journal of Machine Learning Research 9 (2008) 2579–2605.

Daniel J. Trosten received the degree of M.Sc. in machine learning and statistics in 2019 at UiT – The Arctic University of Norway. He is currently working at the UiT Machine Learning group, where he is pursuing a Ph.D. in deep learning with limited labels.

Sigurd Løkse received the degree of M.Sc. in Electrical engineering in 2014 and PhD in 2020 from UiT the Arctic University of Norway, where he is currently a postdoctoral researcher with the Machine Learning group. Research interests include information theoretic learning and learning with limited labels within deep learning.

Robert Jenssen directs the Visual Intelligence Centre (`visual-intelligence.no`). He is a Professor in the Machine Learning Group (`machine-learning.uit.no`) at UiT The Arctic University of Norway. He is an Adjunct Professor at the Pioneer Centre for AI, University of Copenhagen, and at the Norwegian Computing Center.

Michael Kampffmeyer is an Associate Professor and Head of the Machine Learning Group at UiT The Arctic University of Norway. Further, he is a Senior Researcher at the Norwegian Computing Center. Research interests include the development of deep learning algorithms that learn from limited labeled data and their interpretability.

Paper II

This CVPR 2021 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Reconsidering Representation Alignment for Multi-view Clustering

Daniel J. Trosten Sigurd Løkse Robert Jenssen Michael Kampffmeyer
Department of Physics and Technology, UiT The Arctic University of Norway[*]

Abstract

Aligning distributions of view representations is a core component of today's state of the art models for deep multi-view clustering. However, we identify several drawbacks with naïvely aligning representation distributions. We demonstrate that these drawbacks both lead to less separable clusters in the representation space, and inhibit the model's ability to prioritize views. Based on these observations, we develop a simple baseline model for deep multi-view clustering. Our baseline model avoids representation alignment altogether, while performing similar to, or better than, the current state of the art. We also expand our baseline model by adding a contrastive learning component. This introduces a selective alignment procedure that preserves the model's ability to prioritize views. Our experiments show that the contrastive learning component enhances the baseline model, improving on the current state of the art by a large margin on several datasets[1].

1. Introduction

Several kinds of real world data are gathered from different points of view, or by using a collection of different sensors. Videos, for instance, contain both visual and audible components, while captioned images include both the raw image data and a descriptive text. In both of these examples, the low-level content of the views are vastly different, but they can still carry the same high-level cluster structure. The objective of multi-view clustering is to discover this common clustering structure, by learning from all available views simultaneously.

Learning from multiple sources at once is not a trivial task [6]. However, the introduction of deep learning [33], has led to the development of several promising deep multi-view clustering models [1, 36, 48, 61, 64]. These models efficiently learn from multiple views by transforming each view with a view-specific encoder network. The resulting representations are fused to obtain a common representation for all views, which can then be clustered by a subsequent clustering module.

The current state of the art methods for deep multi-view clustering use adversarial training to align the representation distributions from different views [36, 64].

Aligning distributions leads to view invariant representations, which can be beneficial for the subsequent fusion of views, and the clustering module [64]. View invariant representations preserve the information present in all views, while discarding information that only exists in a subset of views. If the view-specific information is irrelevant to the clustering objective, it will be advantageous for the clustering module that the encoders learn to remove it. Moreover, aligning representation distributions introduces an auxiliary task, which regularizes the encoders, and helps preserve the local geometric structure of the input space. This has been shown to improve single-view deep clustering models [21].

Despite these advantages however, we identify three important drawbacks of distribution alignment for multi-view clustering:

Aligning representations prevents view-prioritization in the representation space. Views are not necessarily equally important to the clustering objective. The model should therefore be able to adaptively prioritize views, based on the information contained in the view representations. However, aligning representation distributions makes it harder for the model to prioritize views in the representation space, by making these distributions as similar as possible.

One-to-one alignment of clusters is only attainable when encoders can separate all clusters in all views. When the clustering structure is only partially present in the individual views, alignment causes clusters to merge together in the representation space. This makes the clustering task more difficult for the subsequent clustering module.

Aligning representation distributions can make it harder to discriminate between clusters. Since adversarial alignment only considers the representation distributions, a given cluster from one view might be aligned with a different cluster from another view. This misalignment of label distributions has been shown to have a negative impact on discriminative models in the representation space [62].

The End-to-end Adversarial-attention network for Multi-

[*]UiT Machine Learning Group, machine-learning.uit.no
[1]The source code for the experiments performed in this paper is available at https://github.com/DanielTrosten/mvc

modal Clustering (EAMC) [64] represents the current state of the art for deep multi-view clustering. EAMC aligns the view representations by optimizing an adversarial objective on the encoder networks. The resulting representations are fused with a weighted average, with weights produced by passing the representations through an attention network. Following our reasoning above, we hypothesize that the alignment done by the adversarial module may defeat the purpose of the attention mechanism. Thus inhibiting view prioritization, and leading to less separable clusters after fusion. Our hypothesis is supported by the empirical results of EAMC [64], where the fusion weights are close to uniform for all datasets. Equal fusion weights cause all views to contribute equally to the fused representation, regardless of their contents. Moreover, the fusion weights produced by the attention network depend on all the samples within the current batch. Out-of-sample inference is therefore impossible with EAMC, without making additional modifications to the attention mechanism.

In this work, we seek to alleviate the problems that can arise when aligning distributions of representations in deep multi-view clustering. To this end, we make the following key contributions:

- We highlight pitfalls of aligning representation distributions in deep multi-view clustering, and show that these pitfalls limit previous state of the art models.

- We present Simple Multi-View Clustering (SiMVC) – a new and simple baseline model for deep multi-view clustering, without any form of alignment. Despite its simplicity compared to existing methods, our experiments show that this baseline model performs similar to – and in some cases, even better than – current state of the art methods. SiMVC combines representations of views using a learned linear combination – a simple but effective mechanism for view-prioritization. We empirically demonstrate that this mechanism allows the model to suppress uninformative views and emphasize views that are important for the clustering objective.

- In order to leverage the advantages of alignment – *i.e.* preservation of local geometric structure, and view invariance – while simultaneously avoiding the pitfalls, we attach a selective contrastive alignment module to SiMVC. The contrastive module aligns angles between representations at the sample level, circumventing the problem of misaligned label distributions. Furthermore, in the case that one-to-one alignment is not possible, we make the model capable of ignoring the contrastive objective, preserving the model's ability to prioritize views. We refer to this model as Contrastive Multi-View Clustering (CoMVC).

2. Pitfalls of distribution alignment in multi-view clustering

Here, we consider an idealized version of the multi-view clustering problem. This allows us to investigate and formalize our observations on alignment of representation distributions in multi-view clustering. By assuming that, for each view, all samples within a cluster are located at the same point in the input space, we develop the following proposition[2]:

Proposition 1. *Suppose our dataset consists of V views and k ground truth clusters, and we wish to cluster the data according to this ground truth clustering. Furthermore, we make the following assumptions:*

1. *For each view, all observations that belong to the same ground truth cluster, are located at the same point in the input space.*
2. *For a given view v, $v \in \{1, \dots, V\}$, the number of unique points (*i.e.* distinct/separable clusters) in the input space is k_v.*
3. *The views are mapped to representations using view-specific encoders, and subsequently fused according to a linear combination with unique weights.*

Then, the maximum number of unique clusters after fusion is

$$\kappa_{aligned}^{fused} = \min \left\{ k, \left(\min_{v=1,\dots,V} \{k_v\} \right)^V \right\} \tag{1}$$

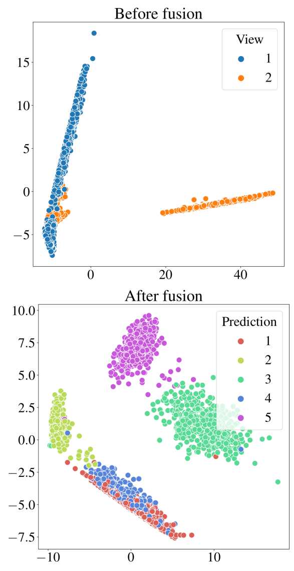
if the distributions of representations from different views are perfectly aligned, and

$$\kappa_{not\ aligned}^{fused} = \min \left\{ k, \prod_{v=1}^{V} k_v \right\} \tag{2}$$

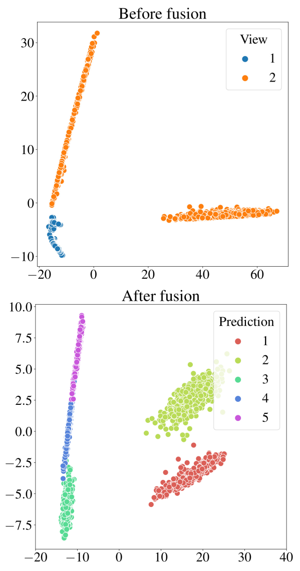
if no alignment is performed.

Implications of Proposition 1. $\kappa_{\cdot}^{\text{fused}}$ in Proposition 1 controls how well the clustering module is able to cluster the fused representations. If $\kappa_{\cdot}^{\text{fused}} < k$, it means that some of the clusters are located at the same point after fusion, making it impossible for the clustering module to discriminate between these clusters. In the extreme case that one of the views groups all the clusters together (*i.e.* $k_v = 1$), it follows that $\kappa_{\text{aligned}}^{\text{fused}} = 1$. This happens because all other views are aligned to the uninformative view (for which $k_v = 1$), collapsing the cluster structure in the representation space. Alignment thus prevents the suppression of this view, and makes it harder to discriminate between clusters in the representation space.

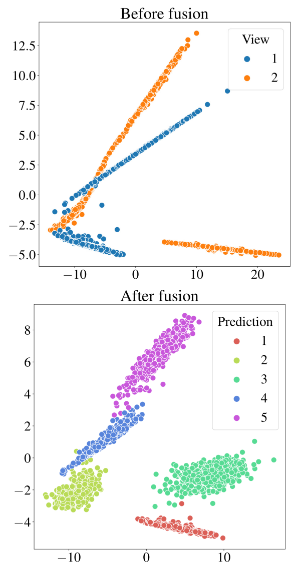
[2]We provide a proof sketch for Proposition 1 in the supplementary.



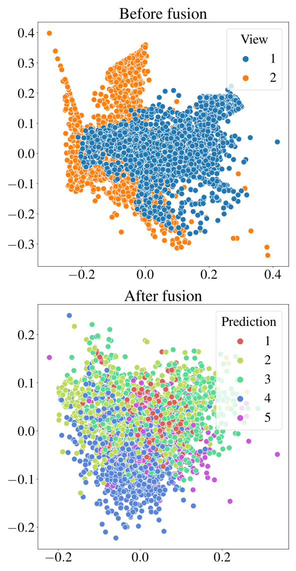
(a) SiMVC +Adv. ACC $= 0.80$



(b) SiMVC. ACC $= 0.99$.



(c) CoMVC. ACC $= 0.99$.



(d) EAMC. ACC $= 0.368$.

Figure 1: Representations for SiMVC with and without adversarial alignment, CoMVC, and EAMC on our toy dataset.

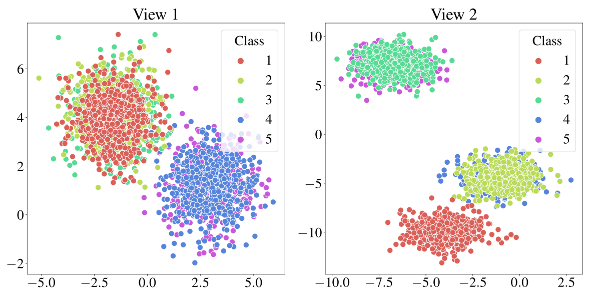


Figure 2: Toy dataset. View 1: Classes (1-3) and (4,5) overlap. View 2: Class 1 is isolated, and classes (2,4) and (3,5) overlap.

However, if we are able to discriminate between all clusters in all views, we have $k_v = k$ for all views, resulting in $\kappa^{\text{fused}}_{\text{aligned}} = \kappa^{\text{fused}}_{\text{not aligned}} = k$. In this case it is possible for both alignment-based models and non-alignment-based models to perfectly cluster the data, provided that the clustering module is sufficiently capable. Alignment-based models can thus benefit from the advantages of alignment, while still being able to separate clusters after fusion.

Experiments on toy data. Proposition 1 makes the simplification that all samples within a cluster are located at the same point, for each view. In order to demonstrate the potential negative impact of aligning representation distributions in a less idealistic setting, and to further motivate the problem, we create a simple two-view dataset. The dataset is shown in Figure 2, and contains five elliptical clusters in two two-dimensional views[3].

We fit SiMVC and SiMVC with adversarial alignment (SiMVC +Adv.) to this dataset, in order to demonstrate the effects of aligning distributions, in a controlled setting. Additionally, we fit our CoMVC and the current state of the art, EAMC, to evaluate more advanced alignment procedures. Note that, for all of these models, the fusion is implemented as a weighted average of view representations, as in Proposition 1. The remaining details on SiMVC and CoMVC are provided in the next section.

Figures 1a and 1b show that attempting to align distributions with adversarial alignment prevents SiMVC from separating between clusters 1 and 4. By adding the adversarial alignment to SiMVC, the number of visible clusters after fusion is reduced from 5 to 4. This is in line with Proposition 1, since we have $\kappa^{\text{fused}}_{\text{aligned}} = 4$ and $\kappa^{\text{fused}}_{\text{not aligned}} = 5$ for this dataset. Figure 1c shows that CoMVC, which relies on the cosine similarity, aligns the angles between the majority of observations. This alignment does not cause classes to overlap in the fused representation. EAMC attempts to align the distributions of view representations (Figure 1d), resulting in a fused representation where the classes are hard to separate. Interestingly, the resulting fused representation exhibits a single group of points, which is significantly worse than the upper bound $\kappa^{\text{fused}}_{\text{aligned}} = 4$ in the analogous idealistic setting. We hypothesize that this is due to EAMC's fusion weights, which we observed to be almost equal for this experiment – thus breaking assumption 3 in Proposition 1.

[3]We repeat this experiment for a 3-cluster dataset in the supplementary.

3. Methods

3.1. Simple Multi-View Clustering (SiMVC)

Suppose our dataset consists of n objects observed from V views. Let $x_i^{(v)}$ be the observation of object i from view v. The objective of our models is then to assign the set of views for each object, $\{x_i^{(1)}, \dots, x_i^{(V)}\}$, to one of k clusters.

To achieve this, we first transform each $x_i^{(v)}$ to its representation $z_i^{(v)}$ according to

$$z_i^{(v)} = f^{(v)}(x_i^{(v)}) \tag{3}$$

where $f^{(v)}$ denotes the encoder network for view v. We then compute the fused representation as a weighted average

$$z_i = \sum_{v=1}^{V} w_v z_i^{(v)} \tag{4}$$

where $w_1, \dots, w_v$ are the *fusion weights*, satisfying $w_v > 0$ for $v = 1, \dots, V$ and $\sum_{v=1}^{V} w_v = 1$. We enforce these constraints by keeping a set of unnormalized weights, from which we obtain $w_1, \dots, w_V$ using the softmax function. We let the unnormalized weights be trainable parameters – a design choice which has the following advantages: (i) During training, the model has a simple and interpretable way to prioritize views according to its clustering objective. By not relying on an auxiliary attention network, we also make the model more efficient – both in terms of memory consumption and training time[4]. (ii) In inference, the weights act as any other model parameters, meaning that out-of sample inference can be done with arbitrary batch sizes, without any modifications to the trained model. Fixed fusion weights also result in deterministic predictions, which are independent of any other samples within the batch.

To obtain the final cluster assignments, we pass the fused representation through a fully connected layer, producing the hidden representation h_i. This is processed by another fully connected layer with a softmax activation, to obtain the k-dimensional vector of soft cluster assignments, α_i.

Loss function. We adopt the Deep Divergence-based Clustering (DDC) [30] loss, which has shown state of the art performance in single-view image clustering [30]. This is also the clustering loss used by EAMC [64] – the current state of the art method for multi-view clustering.

The clustering loss consists of three terms, enforcing cluster separability and compactness, orthogonal cluster assignments, and closeness of cluster assignments to simplex corners, respectively. The first loss term is derived from the multiple-density generalization of the Cauchy-Schwarz divergence [28], and requires clusters to be separable and compact in the space of hidden representations:

$$\mathcal{L}_1 = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{\binom{k}{2}^{-1} \sum_{a=1}^{n} \sum_{b=1}^{n} \alpha_{ai} \kappa_{ab} \alpha_{bj}}{\sqrt{\sum_{a=1}^{n} \sum_{b=1}^{n} \alpha_{ai} \kappa_{ab} \alpha_{bi} \sum_{a=1}^{n} \sum_{b=1}^{n} \alpha_{aj} \kappa_{ab} \alpha_{bj}}} \tag{5}$$

where k denotes the number of clusters, $\kappa_{ij} = \exp(-||h_i - h_j||^2/(2\sigma^2))$, and σ is a hyperparameter.

The second term encourages the cluster assignment vectors for different objects to be orthogonal:

$$\mathcal{L}_2 = \binom{n}{2}^{-1} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \alpha_i^T \alpha_j. \tag{6}$$

Finally, the third term pushes the cluster assignment vectors close to the standard simplex in $\mathbb{R}^k$:

$$\mathcal{L}_3 = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{\binom{k}{2}^{-1} \sum_{a=1}^{n} \sum_{b=1}^{n} m_{ai} \kappa_{ab} m_{bj}}{\sqrt{\sum_{a=1}^{n} \sum_{b=1}^{n} m_{ai} \kappa_{ab} m_{bi} \sum_{a=1}^{n} \sum_{b=1}^{n} m_{aj} \kappa_{ab} m_{bj}}} \tag{7}$$

where $m_{ij} = \exp(-||\alpha_i - e_j||^2)$, and e_j is corner j of the standard simplex in $\mathbb{R}^k$.

The final clustering loss which we minimize during training of SiMVC is the sum of these three terms:

$$\mathcal{L}_{\text{cluster}} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3. \tag{8}$$

3.2. Contrastive Multi-View Clustering (CoMVC)

Contrastive learning offers a way to align representations from different views at the sample level, forcing the label distributions to be aligned as well. Our hypothesis is therefore that a *selective* contrastive alignment will allow the model to learn common representations that are well suited for clustering – while simultaneously avoiding the previously discussed pitfalls of distribution alignment. Self-supervised contrastive models have shown great potential for a large variety of downstream computer vision tasks [5, 12, 13, 20, 22, 40, 49]. These models learn image representations by requiring that representations from *positive* pairs are mapped close together, while representations from *negative* pairs are mapped sufficiently far apart. In multi-view learning, each object has a set of observations from different views associated with it. This admits a natural definition of pairs: Let views of the *same* object be positive pairs, and views of *different* objects be negative pairs.

[4]Average training times for SiMVC, CoMVC, and EAMC are given in the supplementary.

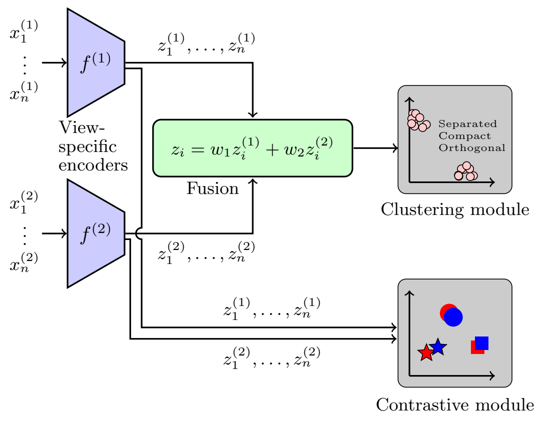


Figure 3: Overview of our proposed models for a two-view dataset. In both SiMVC and CoMVC, the views are first encoded by the view-specific encoder networks $f^{(1)}$ and $f^{(2)}$. The resulting representations are fused with a weighted mean, and then clustered by the clustering module. CoMVC includes an additional contrastive module.

Following [12], we compute the similarity of two representations $z_i^{(v)}$ and $z_j^{(u)}$ as the cosine similarity:

$$s_{ij}^{(vu)} = \frac{z_i^{(v)T} z_j^{(u)}}{||z_i^{(v)}|| \cdot ||z_j^{(u)}||}. \tag{9}$$

Note that in [12], they show that the addition of a projection head between the representations and the similarity, results in better representations – in terms of linear classification accuracy on the learned representations. We found that this was not the case for our model, so we chose to compute the similarity on the representations directly. Experiments comparing versions of our model with and without the projection head can be found in the supplementary.

In order to define a contrastive loss for an arbitrary number of views, we introduce the following generalized version of the NT-Xent loss [12]:

$$\mathcal{L}_{\text{contrastive}} = \frac{1}{nV(V-1)} \sum_{i=1}^{n} \sum_{v=1}^{V} \sum_{u=1}^{V} \mathbb{1}_{\{u \neq v\}} \ell_i^{(uv)} \tag{10}$$

where $\mathbb{1}_{\{u \neq v\}}$ evaluates to 1 when $u \neq v$ and 0 otherwise, and

$$\ell_i^{(uv)} = -\log \frac{e^{s_{ii}^{(uv)}/\tau}}{\sum_{s' \in \text{Neg}(z_i^{(v)}, z_i^{(u)})} e^{s'/\tau}}. \tag{11}$$

Here, τ is a hyperparameter[5], and $\text{Neg}(z_i^{(v)}, z_i^{(u)})$ denotes the set of similarities for negative pairs corresponding to the positive pair $(z_i^{(v)}, z_i^{(u)})$.

[5]We set $\tau = 0.1$ for all experiments, following [12].

A straightforward way to construct $\text{Neg}(z_i^{(v)}, z_i^{(u)})$ would be to include the similarity between all views of object i, and all views of all the other objects within the current batch. However, minimizing Eq. (11) will result in negative samples having a low similarity score. This is indeed the objective of ordinary contrastive learning, but it might be counteractive to the clustering objective, where we want objects from the same cluster to be grouped together in the representation space, and thus be similar to each other. To prevent the contrastive loss from breaking this group structure, we construct $\text{Neg}(z_i^{(v)}, z_i^{(u)})$ in the following manner: First, we define the set

$$\mathcal{N}_i = \{s_{ij}^{(uv)} : j = 1,\ldots,n,\ j \neq i, u, v = 1,\ldots,V, \\ \arg\max \alpha_i \neq \arg\max \alpha_j\}, \tag{12}$$

which consists of all similarities between all views of object i, and all views of all other objects *that were assigned to a different cluster than object i*. We then construct $\text{Neg}(z_i^{(v)}, z_i^{(u)})$ by sampling a fixed number of similarities from $\mathcal{N}_i$. This procedure ensures that we only repel representations of objects that were assigned to different clusters by the clustering module.

CoMVC is the result of adding this contrastive learning framework to SiMVC. Figure 3 shows a schematic overview of the model for a dataset containing two views.

The loss we use to train CoMVC is

$$\mathcal{L} = \mathcal{L}_{\text{cluster}} + \delta \cdot \min\{w_1,\ldots,w_V\} \mathcal{L}_{\text{contrastive}} \tag{13}$$

where $\mathcal{L}_{\text{cluster}}$ is the clustering loss defined in Eq. (8), and δ is a hyperparameter which influences the strength of the contrastive loss. $w_1, \ldots w_V$ are the fusion weights from SiMVC[6].

Minimizing the contrastive loss results in representations that have high cosine similarities. The contrastive alignment is therefore (i) *approximate*, since only the angles between representations, and not the representations themselves, are considered; and (ii) *at the sample level*, preventing misaligned label distributions. Furthermore, multiplying the contrastive loss with the smallest fusion weight automatically adjusts the strength of the contrastive loss, according to the weight of the least informative view. The alignment is therefore *selective*: If the model learns to discard a view by setting its fusion weight to 0, it will simultaneously disable the alignment procedure. By adapting the alignment weight and not relying on adversarial training, CoMVC can benefit from the advantages of aligning representations, while circumventing both the drawbacks of adversarial alignment, and possible difficulties with min-max optimization [4, 19].

[6]Note that we do not propagate gradients through the min operation, in order to avoid the trivial solution of setting the smallest fusion weight to 0.

4. Related work

In this section we will give a brief summary of the existing work on multi-view clustering, as well as related work discussing modality alignment in multi-modal learning. Existing methods for multi-view clustering can be divided into two categories: Traditional (non-deep learning based) methods and deep learning based methods.

Traditional methods. Two-stage methods first learn a common representation from all the views, before clustering them using a single-view clustering algorithm [7, 11]. However, recent work shows that letting the learned representation adapt to the clustering algorithm leads to better clusterings [56]. In order to avoid this drawback of two-stage approaches, non-negative matrix factorization [8, 15, 24, 57, 63] has been used to compute the cluster assignment matrix directly from the data matrices. Similarly, subspace methods assume that observations can be represented by one or more self-representation matrices [5, 10, 37, 41, 55, 58, 59, 61] and use the self-representation matrices to identify linear subspaces of the vector space spanned by all the observations, that represent distinct clusters. Alternative popular approaches include methods based on graphs [44, 48, 50, 51, 60, 65] and kernels [16, 18, 35, 39], which both assume that the data can be represented with one or more kernel (or affinity) matrices such that respective clusterings can be found based on these matrices.

Deep learning based methods. Deep learning based two-stage methods [2, 43, 52] work similarly to the two-stage methods described above, but instead use deep neural networks to learn the common representation. However, the two-stage methods are regularly outperformed by deep end-to-end methods that adapt their representation learning networks to the subsequent clustering module. Deep graph-based methods [14, 25, 26, 34] for instance, use affinity matrices together with graph neural networks to directly cluster the data. Similarly, deep subspace methods [1, 3] make the same subspace assumption as above, but compute the self-representation matrix from an intermediate representation in their deep neural network. Lastly, adversarial methods [36, 64] use generators and discriminators to align distributions of hidden representations from different views. These adversarial methods have outperformed the previous approaches to multi-view clustering, yielding state of the art clustering performance on several multi-view datasets.

Distribution alignment. Outside the field of multi-view clustering, the problem of naïvely aligning distributions has recently found increasing attention [62, 53], and led to more efficient fusion techniques [23, 9, 46]. However, this effort has largely been restricted to supervised multi-modal learning frameworks and domain adaptation approaches.

5. Experiments

5.1. Setup

Implementation. Our models are implemented in the PyTorch [45] framework. We train the models for 100 epochs on mini-batches of size 100, using the Adam optimization technique [31] with default parameters. We observe that 100 epochs is sufficient for the training to converge. Training is repeated 20 times, and we report the results from the run resulting in the lowest value of $\mathcal{L}_1$ in the clustering loss. The σ hyperparameter was set to 15% of the median pairwise distance between hidden representations within a mini-batch, following [30]. For the contrastive model, we set the number of negative pairs per positive pair to 25 for all experiments. We set $\delta = 0.1$ for the two-view datasets, and $\delta = 20$ for the three-view datasets. We observe that the three-view datasets benefit from stronger contrastive alignment. Our implementation and a complete overview of the architecture details can be found in the supplementary.

Datasets. We evaluate our models using six well-known multi-view datasets [36, 64], containing both raw image data, and vector data. These are: (i) *PASCAL VOC 2007 (VOC)* [17]. We use the version provided by [27], which contains GIST features and word frequency counts for manually tagged natural images. (ii) *Columbia Consumer Video (CCV)* [29], which consists of SIFT, STIP and MFCC features from internet videos. (iii) *Edge-MNIST (E-MNIST)* [38], which is a version of the ordinary MNIST dataset where the views contain the original digit, and an edge-detected version, respectively. (iv) *Edge-FashionMNIST (E-FMNIST)* [54], which consists of grayscale images of clothing items. We synthesize a second view by running the same edge-detector as the one used to create E-MNIST. (v) *COIL-20* [42], which contains grayscale images of 20 items, depicted from different angles. We create a three-view dataset by randomly grouping the images for an item into groups of three. (vi) *SentencesNYU v2 (RGB-D)* [32], which consists of images of indoor scenes along with descriptions of each image. Following [64], we use image features from a ResNet-50 without the classification head, pre-trained on the ImageNet dataset, as the first view. Embeddings of the image descriptions using a pre-trained doc2vec model on the Wikipedia dataset constitute the second view[7].

Note that, for the datasets with multiple labels, we select the objects with exactly one label. See Table 1 for more information on the evaluation datasets.

Baseline models. We compare our models to an extensive set of baseline methods, which represent the current state of the art for multi-view clustering: (i) Spectral Clustering (SC) [47] on each view, and the concatenation of all views SC(con); (ii) Robust Multi-view K-means Clustering (RMKMC) [8]; (iii) tensor-based Representation Learn-

[7]We provide the details of these pre-trained models in the supplementary.

Dataset	Objs.	Cats.	Views	Dims.
VOC	5649	20	2	512, 399
CCV	6773	20	3	5000, 5000, 4000
E-MNIST	60000	10	2	28×28
E-FMNIST	60000	10	2	28×28
COIL-20	480	20	3	128×128
RGB-D	1449	13	2	2048, 300

Table 1: Summary of the datasets used for evaluation. *Objs.* and *Cats.* denote the number of objects and categories present in the dataset, respectively. *Views* and *Dims.* denote the number of views, and the dimensionality of each view, respectively. Note that for E-MNIST, E-FMNIST, and COIL-20, the input dimensionality is the same for all views.

ing Multi-view clustering tRLMvc [15]; (iv) Consistent and Specific Multi-view Subspace Clustering (CSMSC) [41]; (v) Weighted Multi-view Spectral Clustering (WMSC) [65]; (vi) Multi-view Consensus Graph Clustering (MCGC) [60]; (vii) Deep Canonical Correlation Analysis (DCCA) [2]; (viii) Deep Multimodal Subspace Clustering (DMSC) [1]; (ix) Deep Adversarial Multi-view Clustering (DAMC) [36]; and (x) End-to-end Adversarial attention network for Multi-modal Clustering (EAMC) [64].

Evaluation protocol. To ensure a fair comparison, we report the baseline results over multiple runs, following [64][8]. To assess the models' clustering performance, we use the unsupervised clustering accuracy (ACC) and normalized mutual information (NMI). For both these metrics, higher values correspond to better clusterings.

5.2. Results

Quantitive results on VOC, CCV and E-MNIST are shown in Table 2. The results illustrate that not aligning representations can have a significant improvement (relative gain in ACC larger than 29% on E-MNIST) compared to adversarial alignment, while selective alignment always improves performance. Note that entries for E-MNIST in Table 2 are missing as the number of samples makes the traditional approaches computationally infeasible.

Table 3 compares SiMVC and CoMVC to the previous state of the art, EAMC on E-FMNIST, COIL-20 and RGB-D. Again, we observe that naïvely aligning feature representations tends to worsen performance. This highlights the importance of being considerate when aligning representations in multi-view clustering.

Ablation study. We perform an ablation study in order to evaluate the effects of the different components in the contrastive loss[9]. Specifically, we train CoMVC with and without the proposed negative pair sampling and the adaptive weight factor ($\min\{w_1, \dots, w_V\}$), on E-MNIST and VOC. When we remove the negative sampling, we construct $\text{Neg}(z_i^{(v)}, z_i^{(u)})$ by including the similarities between all views of object i, and all views of all the other objects within the current batch.

[8]The details of the evaluation protocol are given in the supplementary.

[9]We include an ablation study with the DDC loss in the supplementary.

Dataset	VOC		CCV		E-MNIST	
Metric	ACC	NMI	ACC	NMI	ACC	NMI
SC(1)	38.4	39.2	10.2	0.5		
SC(2)	40.2	41.1	18.8	17.3		
SC(3)			11.3	0.8		
SC(con)	37.2	38.7	9.3	7.4		
RMKMC	45.8	46.9	17.6	16.5		
tRLMvc	53.4	54.7	21.2	22.6		
CSMSC	48.8	49.6	19.4	18.6		
WMSC	47.1	46.2	20.5	19.6		
MCGC	52.7	54.6	22.4	21.6		
DCCA	39.7	42.5	17.3	18.2	47.6	44.3
DMSC	54.1	53.8	18.3	19.4	65.3	61.4
DAMC	56.0	55.2	24.3	23.1	64.6	59.4
EAMC	**60.7**	**61.5**	**26.1**	**26.6**	66.8	62.8
SiMVC	55.1 (-5.6)	**61.5** (+0.0)	14.4 (-11.7)	11.2 (-15.4)	**86.2** (+19.4)	**82.6** (+19.8)
CoMVC	**61.9** (+1.2)	**67.5** (+6.0)	**29.5** (+3.4)	**28.7** (+2.1)	**95.5** (+28.7)	**90.7** (+27.9)

Table 2: Clustering metrics [%] on VOC, CCV, and E-MNIST. The best and second best are highlighted in bold. The differences between our models and the best baseline model are shown in parentheses. Green differences indicate improvements. Baseline results are taken from [64].

Dataset	E-FMNIST		COIL-20		RGB-D	
Metric	ACC	NMI	ACC	NMI	ACC	NMI
EAMC	55.2	**62.5**	69.0	75.3	32.3	20.7
SiMVC	**56.8** (+1.6)	50.7 (-11.8)	**77.5** (+8.5)	**91.8** (+16.5)	**39.6** (+7.3)	**35.6** (+14.9)
CoMVC	**59.5** (+4.3)	52.3 (-10.2)	**89.4** (+20.4)	**95.7** (+20.4)	**41.3** (+9.0)	**40.5** (+19.8)

Table 3: Clustering metrics [%] on E-FMNIST, COIL-20 and RGB-D. Same formatting as in Table 2.

	Neg. samp.	Ad. weight	ACC [%]	NMI [%]
E-MNIST	–	–	87.4	86.8
	–	✓	94.7	89.5
	✓	–	87.5	86.6
	✓	✓	**95.5**	**90.7**
VOC	–	–	54.7	61.3
	–	✓	55.3	60.7
	✓	–	58.5	67.4
	✓	✓	**61.9**	**67.5**

Table 4: Ablation study results for CoMVC on E-MNIST and VOC.

	EAMC			SiMVC			CoMVC		
View	1	2	3	1	2	3	1	2	3
VOC	48	52		47	53		64	36	
CCV	26	38	36	32	35	33	1	75	24
E-MNIST	48	52		95	05		67	33	
E-FMNIST	53	47		78	22		99	1	
COIL-20	32	32	36	33	35	32	34	32	34
RGB-D	53	47		59	41		59	41	

Table 5: Fusion weights [%] for EAMC, SiMVC, and CoMVC. For EAMC, we split the entire dataset into batches of size 100 and report the average weight over these batches.

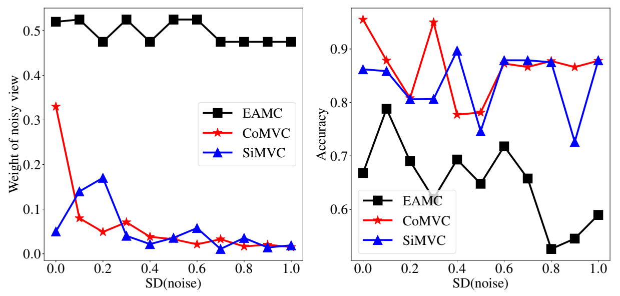


Figure 4: Fusion weights and clustering accuracy (ACC) on E-MNIST, with increasing levels of Gaussian noise added to the second view.

Results of the ablation study (Table 4) show that dropping the adaptive weighting and the negative sampling strategy both have a negative impact on CoMVC's performance. This justifies their inclusion in the final contrastive loss.

View prioritization. Table 5 shows the weight parameters that are obtained for EAMC, SiMVC and CoMVC for all datasets. EAMC always produces close to uniform weight distributions, while SiMVC and CoMVC are able to suppress uninformative views. Note, for datasets, such as COIL-20, where views are assumed equally important[10], we do also observe close to uniform weight distributions for SiMVC and CoMVC.

To further assess our models' capabilities to prioritize views, we corrupt the edge-view (view 2) in E-MNIST with additive Gaussian noise, and record the models' performance as the standard deviation of the noise increases. We also repeat the experiment for the EAMC model, as it represents the current state of the art. Figure 4 shows the resulting fusion weights for the noisy view and the clustering accuracies, for different noise levels. For SiMVC and CoMVC, we observe that the weight of the noisy view decreases as the noise increases. The mechanism for prioritizing views thus works as expected. SiMVC and CoMVC can therefore produce accurate clusterings, regardless of the noise level. Conversely, we observe that the attention mechanism in EAMC is unable to produce fusion weights that suppress the noisy view. This results in a significant drop in clustering accuracy, as the noise increases.

[10]Since views in COIL-20 refer to objects depicted from random angles.

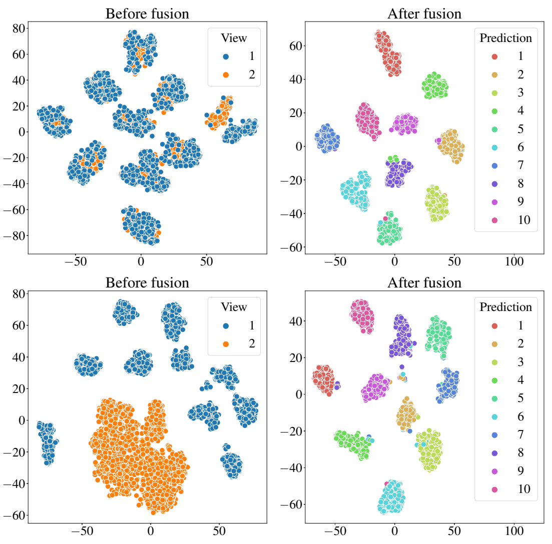


Figure 5: Learned representations before and after fusion for regular (top) and noisy ($\sigma = 1$) E-MNIST (bottom). Projected to 2-D using T-SNE.

Selective alignment in CoMVC. Figure 5 demonstrates the selective alignment in CoMVC, for the noise-free and noisy variants of the E-MNIST dataset. In the noise-free case, CoMVC aligns the representations, resulting in clusters that are well separated. When the second view has been corrupted by noise however, it is discarded by the view prioritization mechanism, by setting its fusion weight to 0. This simultaneously disables the alignment procedure, preventing the fused representation from being corrupted by the noisy view, thus preserving the cluster structure.

6. Conclusion

Our work highlights the importance of considering representation alignment when performing multi-view clustering. Comparing the results of our SiMVC to previous results illustrates that naïvely aligning distributions using adversarial learning can prevent the model from learning good clusterings, while CoMVC illustrates the benefit of selective alignment, leveraging the best of both worlds.

Acknowledgements. This work was financially supported by the Research Council of Norway (RCN), through its Centre for Research-based Innovation funding scheme (Visual Intelligence, grant no. 309439), and Consortium Partners. The work was further funded by RCN FRIPRO grant no. 315029, RCN IKTPLUSS grant no. 303514, and the UiT Thematic Initiative "Data-Driven Health Technology"

References

[1] Mahdi Abavisani and Vishal M. Patel. Deep Multimodal Subspace Clustering Networks. *IEEE Journal of Selected Topics in Signal Processing*, 12(6), 2018.

[2] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep Canonical Correlation Analysis. In *International Conference on Machine Learning*, 2013.

[3] Aluizio F. R. Araújo, Victor O. Antonino, and Karina L. Ponce-Guevara. Self-organizing subspace clustering for high-dimensional and multi-view data. *Neural Networks*, 2020.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *International Conference on Machine Learning*, 2017.

[5] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning Representations by Maximizing Mutual Information Across Views. In *Neural Information Processing Systems*, 2019.

[6] Tadas Baltrusaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal Machine Learning: A Survey and Taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2), 2019.

[7] M. B. Blaschko and C. H. Lampert. Correlational spectral clustering. In *Computer Vision and Pattern Recognition*, 2008.

[8] Xiao Cai, Feiping Nie, and Heng Huang. Multi-View K-Means Clustering on Big Data. In *International Joint Conference on Artificial Intelligence*, 2013.

[9] Cătălina Cangea, Petar Veličković, and Pietro Liò. Xflow: Cross-modal deep neural networks for audiovisual classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[10] Xiaochun Cao, Changqing Zhang, Huazhu Fu, Si Liu, and Hua Zhang. Diversity-induced Multi-view Subspace Clustering. In *Computer Vision and Pattern Recognition*, 2015.

[11] Kamalika Chaudhuri, Sham Kakade, K. Livescu, and Karthik Sridharan. Multi-View Clustering via Canonical Correlation Analysis. In *International Conference on Machine Learning*, 2009.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning*, 2020.

[13] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big Self-Supervised Models are Strong Semi-Supervised Learners. *arXiv:2006.10029 [cs, stat]*, 2020.

[14] Jiafeng Cheng, Qianqian Wang, Quanxue Gao, Deyan Xie, and Zhiqiang Tao. Multi-View Attribute Graph Convolution Networks for Clustering. In *International Joint Conference on Artificial Intelligence*, 2020.

[15] Miaomiao Cheng, Liping Jing, and Michael K. Ng. Tensor-Based Low-Dimensional Representation Learning for Multi-View Clustering. *IEEE Transactions on Image Processing*, 28(5), 2019.

[16] Liang Du, Peng Zhou, Lei Shi, Hanmo Wang, Mingyu Fan, Wenjian Wang, and Yi-Dong Shen. Robust multiple kernel k-means using l21-norm. In *AAAI Conference on Artificial Intelligence*, 2015.

[17] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. In *International Journal of Computer Vision*, 2010.

[18] Mehmet Gönen and Adam A. Margolin. Localized Data Fusion for Kernel k-Means Clustering with Application to Cancer Biology. In *Neural Information Processing Systems*, 2014.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Neural Information Processing Systems*, 2014.

[20] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised Learning. *arXiv:2006.07733 [cs, stat]*, 2020.

[21] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved Deep Embedded Clustering with Local Structure Preservation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, Melbourne, Australia, 2017.

[22] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *Computer Vision and Pattern Recognition*, 2020.

[23] Ming Hou, Jiajia Tang, Jianhai Zhang, Wanzeng Kong, and Qibin Zhao. Deep multimodal multilinear fusion with high-order polynomial pooling. In *Advances in Neural Information Processing Systems*, pages 12136–12145, 2019.

[24] Aiping Huang, Tiesong Zhao, and Chia-Wen Lin. Multi-View Data Fusion Oriented Clustering via Nuclear Norm Minimization. *IEEE Transactions on Image Processing*, 29, 2020.

[25] Shudong Huang. Auto-weighted multi-view clustering via deep matrix decomposition. *Pattern Recognition*, 2020.

[26] Shuning Huang, Kaoru Ota, Mianxiong Dong, and Fanzhang Li. MultiSpectralNet: Spectral Clustering Using Deep Neural Network for Multi-View Data. *IEEE Transactions on Computational Social Systems*, 6(4), 2019.

[27] Sung Ju Hwang and Kristen Grauman. Accounting for the relative importance of objects in image retrieval. In *British Machine Vision Conference*, 2010.

[28] Robert Jenssen, Jose C. Principe, Deniz Erdogmus, and Torbjørn Eltoft. The Cauchy–Schwarz divergence and Parzen windowing: Connections to graph theory and Mercer kernels. *Journal of the Franklin Institute*, 343(6), 2006.

[29] Yu-Gang Jiang, Guangnan Ye, Shih-Fu Chang, Daniel Ellis, and Alexander C. Loui. Consumer video understanding: A benchmark database and an evaluation of human and machine performance. In *International Conference on Multimedia Retrieval*, 2011.

[30] Michael Kampffmeyer, Sigurd Løkse, Filippo M. Bianchi, Lorenzo Livi, Arnt-Børre Salberg, and Robert Jenssen. Deep

divergence-based approach to clustering. *Neural Networks*, 113, 2019.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

[32] Chen Kong, Dahua Lin, Mohit Bansal, Raquel Urtasun, and Sanja Fidler. What Are You Talking About? Text-to-Image Coreference. In *Computer Vision and Pattern Recognition*, Columbus, OH, USA, 2014.

[33] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. *Nature*, 521(7553), 2015.

[34] Jianqiang Li, Guoxu Zhou, Yuning Qiu, Yanjiao Wang, Yu Zhang, and Shengli Xie. Deep graph regularized non-negative matrix factorization for multi-view clustering. *Neurocomputing*, 390, 2020.

[35] Miaomiao Li, Xinwang Liu, Lei Wang, Yong Dou, Jianping Yin, and En Zhu. Multiple kernel clustering with local kernel alignment maximization. In *International Joint Conference on Artificial Intelligence*, 2016.

[36] Zhaoyang Li, Qianqian Wang, Zhiqiang Tao, Quanxue Gao, and Zhaohua Yang. Deep Adversarial Multi-view Clustering Network. In *International Joint Conference on Artificial Intelligence*, 2019.

[37] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust Recovery of Subspace Structures by Low-Rank Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1), 2013.

[38] Ming-Yu Liu and Oncel Tuzel. Coupled Generative Adversarial Networks. In *Neural Information Processing Systems*, 2016.

[39] Xinwang Liu. Multiple Kernel k-Means Clustering with Matrix-Induced Regularization. In *AAAI Conference on Artificial Intelligence*, 2016.

[40] Sindy Löwe, Peter O'Connor, and Bastiaan Veeling. Putting An End to End-to-End: Gradient-Isolated Learning of Representations. In *Neural Information Processing Systems*, 2019.

[41] Shirui Luo, Changqing Zhang, Wei Zhang, and Xiaochun Cao. Consistent and Specific Multi-View Subspace Clustering. In *AAAI Conference on Artificial Intelligence*, 2018.

[42] Sameer A. Nene, Shree K. Nayar, and Hiroshi Murase. Columbia Object Image Library (COIL-20). Techincal Report CUCS-006-96, 1996.

[43] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Ng. Multimodal Deep Learning. In *International Conference on Machine Learning*, 2011.

[44] Feiping Nie, Jing Li, and Xuelong Li. Self-weighted Multiview Clustering with Multiple Graphs. In *International Joint Conference on Artificial Intelligence*, 2017.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems*, 2019.

[46] Juan-Manuel Pérez-Rúa, Valentin Vielzeuf, Stéphane Pateux, Moez Baccouche, and Frédéric Jurie. Mfas: Multimodal fusion architecture search. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pages 6966–6975, 2019.

[47] Jianbo Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 2000.

[48] Zhiqiang Tao, Hongfu Liu, Sheng Li, Zhengming Ding, and Yun Fu. Marginalized Multiview Ensemble Clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 31(2), 2020.

[49] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv:1807.03748 [cs, stat]*, 2019.

[50] Beilei Wang, Yun Xiao, Zhihui Li, Xuanhong Wang, Xiaojiang Chen, and Dingyi Fang. Robust Self-Weighted Multi-View Projection Clustering. In *AAAI Conference on Artificial Intelligence*, 2020.

[51] R. Wang, F. Nie, Z. Wang, H. Hu, and X. Li. Parameter-Free Weighted Multi-View Projected Clustering with Structured Graph Learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(10), 2020.

[52] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On Deep Multi-View Representation Learning. In *International Conference on Machine Learning*, 2015.

[53] Yifan Wu, Ezra Winston, Divyansh Kaushik, and Zachary Lipton. Domain adaptation with asymmetrically-relaxed distribution alignment. In *International Conference on Machine Learning*, pages 6872–6881, 2019.

[54] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747 [cs, stat]*, 2017.

[55] Deyan Xie, Xiangdong Zhang, Quanxue Gao, Jiale Han, Song Xiao, and Xinbo Gao. Multiview Clustering by Joint Latent Representation and Similarity Learning. *IEEE Transactions on Cybernetics*, 2019.

[56] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised Deep Embedding for Clustering Analysis. In *International Conference on Machine Learning*, 2016.

[57] Chang Xu, D. Tao, and Chao Xu. Multi-view Self-Paced Learning for Clustering. In *International Joint Conference on Artificial Intelligence*, 2015.

[58] Zhiyong Yang, Qianqian Xu, Weigang Zhang, Xiaochun Cao, and Qingming Huang. Split Multiplicative Multi-View Subspace Clustering. *IEEE Transactions on Image Processing*, 28(10), 2019.

[59] Ming Yin, Junbin Gao, Shengli Xie, and Yi Guo. Multiview Subspace Clustering via Tensorial t-Product Representation. *IEEE Transactions on Neural Networks and Learning Systems*, 30(3), 2019.

[60] Kun Zhan, Feiping Nie, Jing Wang, and Yi Yang. Multiview Consensus Graph Clustering. *IEEE Transactions on Image Processing*, 28(3), 2019.

[61] Changqing Zhang, Huazhu Fu, Qinghua Hu, Xiaochun Cao, Yuan Xie, Dacheng Tao, and Dong Xu. Generalized Latent Multi-View Subspace Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1), 2020.

[62] Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J. Gordon. On Learning Invariant Representation for Domain Adaptation. *arXiv:1901.09453 [cs, stat]*, 2019.

[63] Handong Zhao, Zhengming Ding, and Yun Fu. Multi-View Clustering via Deep Matrix Factorization. In *AAAI Conference on Artificial Intelligence*, 2017.

[64] Runwu Zhou and Yi-Dong Shen. End-to-End Adversarial-Attention Network for Multi-Modal Clustering. In *Computer Vision and Pattern Recognition*, 2020.

[65] Linlin Zong, Xianchao Zhang, Xinyue Liu, and Hong Yu. Weighted Multi-View Spectral Clustering Based on Spectral Perturbation. In *AAAI Conference on Artificial Intelligence*, 2018.

Supplementary material:
Reconsidering Representation Alignment for Multi-view Clustering

Daniel J. Trosten Sigurd Løkse Robert Jenssen Michael Kampffmeyer
Department of Physics and Technology, UiT The Arctic University of Norway[*]

1. Pitfalls of distribution alignment in multi-view clustering

1.1. Proof sketch for Proposition 1

Proof sketch. Suppose that, for view v, the k_v clusters in the input space, are mapped to c_v unique points in the representation space. This is possible under assumptions 1 and 2. The number of unique clusters after fusion is then upper bounded by the number of unique linear combinations on the form:

$$\sum_{v=1}^{V} w_v c_\star^{(v)} \tag{1}$$

where $c_\star^{(v)}$ is one of the c_v points obtained by mapping the k_v unique points (clusters) for view v, to the representation space. Note that the encoders might not be injective, meaning that we can have $c_v < k_v$. Under assumption 3, the maximum number of unique such linear combinations is equal to $c_1 \cdot c_2 \cdots c_V = \prod_{v=1}^{V} c_v$. Since we only have k clusters in the entire dataset, the number of unique clusters after fusion will also be upper bounded by k. This gives:

$$\kappa_\cdot^{\text{fused}} = \min\left\{ k, \prod_{v=1}^{V} c_v \right\}. \tag{2}$$

Perfectly aligned representations. Clusters that are separated in the input space can be mapped to the same centroid in the representation space, but not vice versa. I.e. it is not possible for the encoding network to separate two clusters that lie at the same point in the input space. The perfect alignment constraint therefore forces the number of unique points to be equal to the smallest k_v for each cluster. That is:

$$c_v = \min_{w=1,\dots,V} \{k_w\}, \quad v = 1, \dots, V. \tag{3}$$

We then get

$$\kappa_{\text{aligned}}^{\text{fused}} = \min\left\{ k, \prod_{v=1}^{V} \min_{w=1,\dots,V} \{k_w\} \right\} \tag{4}$$

$$= \min\left\{ k, \left(\min_{v=1,\dots,V} \{k_v\} \right)^V \right\} \tag{5}$$

Unaligned representations. Here the encoder for view v has the ability to map the k_v separable clusters to k_v unique representations, which do not coincide with the representations from any other views. We therefore get $c_v = k_v$, and

$$\kappa_{\text{not aligned}}^{\text{fused}} = \min\left\{ k, \prod_{v=1}^{V} c_v \right\} \tag{6}$$

$$= \min\left\{ k, \prod_{v=1}^{V} k_v \right\}. \tag{7}$$

□

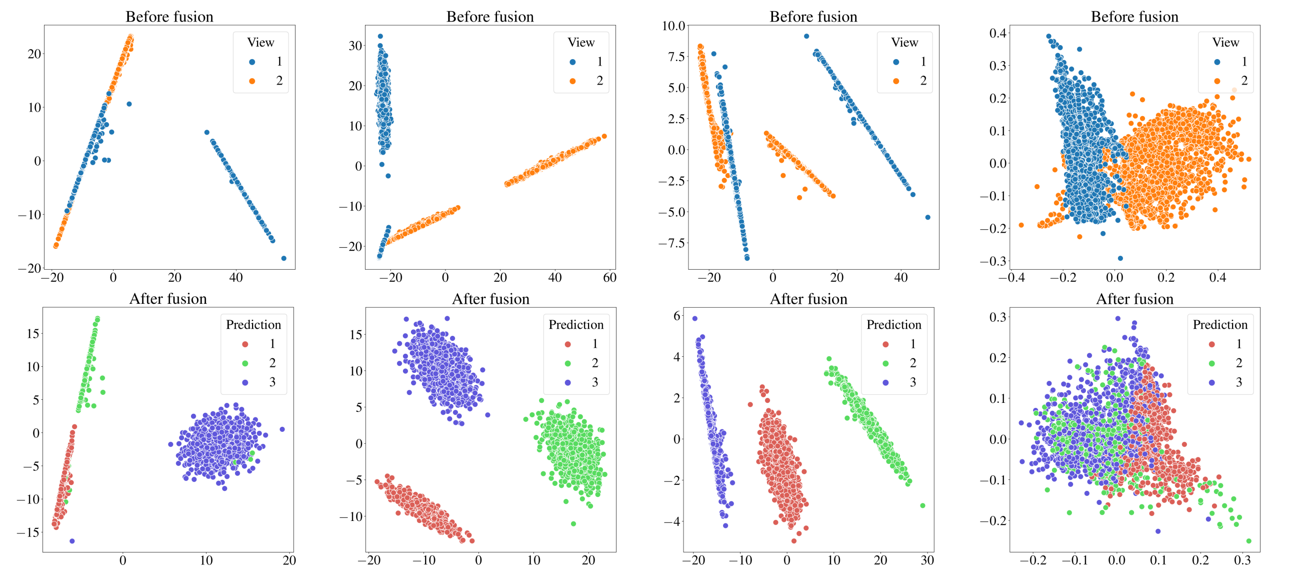
1.2. Experiments with toy data

Figure 1 shows the results of our toy experiment with 3 clusters instead of 5. The dataset is shown in Figure 2. Similarly to the experiment with 5 clusters, we observe that SiMVC + Adv. partially aligns the distributions. Due to the reduced number of clusters however, it is still possible to separate the clusters after fusion. This outcome is consistent with Proposition 1, from which we get $\kappa_{\text{aligned}}^{\text{fused}} = \min\{3, 2^2\} = 3$.

For CoMVC, we see that the angles between representations have been aligned, which also results in separable clusters.

EAMC attempts to align the distributions, which results in all clusters being mixed together after fusion – similar to what we observed for the experiment with 5 clusters. For this experiment, we also observe that EAMC produces approximately equal fusion weights. This violates assumption 3 in Proposition 1, and can further reduce the cluster separability in the space of fused representations.

[*]UiT Machine Learning Group, machine-learning.uit.no



(a) SiMVC +Adv. ACC $= 0.99$ (b) SiMVC. ACC $= 1.0$. (c) CoMVC. ACC = 1.0. (d) EAMC. ACC $= 0.44$.

Figure 1: Representations for SiMVC with and without adversarial alignment, CoMVC, and EAMC on a version of our toy dataset with 3 clusters.

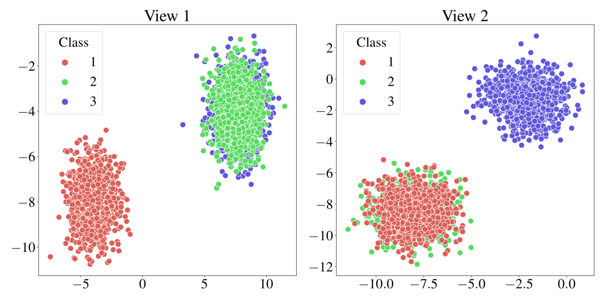


Figure 2: Toy dataset with 3 clusters. View 1: Class 1 is isolated, and classes (2,3) overlap. View 2: Class (1,2) overlap, and class 3 is isolated.

Dataset	Projection head	Negative sampling	Adaptive weight	ACC [%]	NMI [%]
E-MNIST	–	–	–	87.4	86.8
	–	✓	–	87.5	86.6
	–	–	✓	94.7	89.5
	–	✓	✓	**95.5**	**90.7**
	✓	–	–	87.5	86.9
	✓	✓	–	88.2	86.3
	✓	–	✓	87.4	87.2
	✓	✓	✓	77.1	77.5
VOC	–	–	–	54.7	61.3
	–	✓	–	58.5	67.4
	–	–	✓	55.3	60.7
	–	✓	✓	61.9	**67.5**
	✓	–	–	53.4	58.2
	✓	✓	–	57.0	63.2
	✓	–	✓	**62.4**	65.3
	✓	✓	✓	55.2	59.6

Table 1: CoMVC ablation study with and without a projection head.

2. Methods

2.1. CoMVC with projection head

Table 1 shows the results of an extension of the ablation study for CoMVC, where we also include a projection head between the view representations and the cosine similarity. Following [1], we let the projection head be two fully connected layers, separated by a ReLU-nonlinearity. Batch normalization is applied after both layers. The results show that some configurations benefit marginally from the addition of a projection head. However, adding the projection head does not improve the overall performance of CoMVC. We therefore chose to not include it in the final model.

2.2. Ablation study: clustering loss

Here, we perform an ablation study on the E-MNIST dataset, in order to show the effects of the individual terms in the DDC [2] clustering loss. Note that, since not all config-

Model	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	ACC [%]	NMI [%]
SiMVC	✓	–	–	19.2	19.6
	–	✓	–	38.1	31.4
	–	–	✓	75.2	73.9
	✓	✓	–	78.2	78.6
	✓	–	✓	76.6	77.5
	–	✓	✓	77.4	76.9
	✓	✓	✓	**86.2**	**82.6**
CoMVC	✓	–	–	19.3	20.6
	–	✓	–	36.5	25.2
	–	–	✓	72.8	71.7
	✓	✓	–	71.3	73.2
	✓	–	✓	78.0	78.2
	–	✓	✓	74.8	73.5
	✓	✓	✓	**95.5**	**90.7**

Table 2: Results of an ablation study where we systematically drop terms from the clustering loss. The checkmarks indicate which terms that are included in each configuration.

urations include the $\mathcal{L}_1$ term, we select models based on the sum of the included terms instead. The resulting accuracies for SiMVC and CoMVC when we systematically drop terms from the clustering loss, are listed in Table 2. These results are in line with previous ablation studies conducted on the DDC clustering loss [2, 5]: The models perform best when all terms are included – dropping terms from the clustering loss reduces the performance of both SiMVC and CoMVC.

3. Experiments

3.1. Source code

The source code for our experiments is publicly available at `https://github.com/DanielTrosten/mvc`.

3.2. Details of pre-trained models

For RGB-D, we use the following pre-trained models to extract features for the respective views:

- View 1: ResNet-50 pre-trained on the ImageNet dataset. We use the version available in PyTorch[1], and remove the last (classification) layer.
- View 2: Doc2Vec pre-trained on the Wikipedia dataset. We use the pre-trained model available at `https://github.com/jhlau/doc2vec`.

Note that the same types of architectures are used in [5] to extract features for RGB-D. However, the authors do not supply the model and training details required to exactly reproduce their features.

[1]Documentation for the model can be found at `https://pytorch.org/docs/stable/torchvision/models.html`

Layer type	Neurons	Activation	Batch-norm
`FC`	512	ReLU	×
`FC`	512	ReLU	×
`FC`	256	ReLU	×

(a) Fully connected encoder. `FC`: fully connected layer.

Layer type	Filter size	Filters	Activation	Batch-norm
`Conv`	5×5	32	ReLU	×
`Conv`	5×5	32	ReLU	✓
`MaxPool`	2×2	–	–	×
`Conv`	3×3	32	ReLU	×
`Conv`	3×3	32	ReLU	✓
`MaxPool`	2×2	–	–	×

(b) Convolutional neural network encoder. `Conv`: convolutional layer. `MaxPool`: max-pooling layer. Note that Batch normalization is applied before the activation function.

Layer type	Neurons	Activation	Batch-norm
FC	100	ReLU	✓
FC	k	softmax	×

(c) Clustering module. k denotes the number of clusters.

Table 3: Network architectures.

3.3. Model architectures and hyperparameters

SiMVC and CoMVC trained on VOC, CCV, and RGB-D use fully connected encoders (Table 3a) for all views. On E-MNIST, E-FMNIST and COIL our models use convolutional neural network encoders for all views (Table 3b). The clustering module (Table 3c) is the same for all experiments with SiMVC and CoMVC.

Table 4 lists the other hyperparameters that are not part of the model architectures. We use gradient clipping, and clip gradients with norms greater than ”Max gradient norm”. For some datasets, we found that decaying the learning rate helped the models converge. On these datasets, we reduce the learning rate once, at epoch ”Decay step”, with a factor of ”Decay factor”.

3.4. Evaluation protocol

For VOC, CCV, and E-MNIST, we use the baseline results obtained by [5]. For all baseline models, except EAMC, they run the model 10 times and report the average ACC and NMI. For EAMC, they train the model 20 times, and report the results from the run which resulted in the lowest value of the loss function. We follow the same evaluation procedure for EAMC, when we evaluate it on E-FMNIST, COIL-20,

Dataset	Model	Batch size	Epochs	τ	δ	Negative samples	Max gradient norm	Initial learning rate	Decay step	Decay factor
VOC	SiMVC	100	100	0.1	0.1	–	5	0.001	50	0.1
	CoMVC	100	100	0.1	0.1	25	5	0.001	–	–
CCV	SiMVC	100	100	0.1	20	–	5	0.001	–	–
	CoMVC	100	100	0.1	20	25	5	0.001	50	0.1
E-MNIST	SiMVC	100	100	0.1	0.1	–	5	0.001	–	–
	CoMVC	100	100	0.1	0.1	25	5	0.001	–	–
E-FMNIST	SiMVC	100	100	0.1	0.1	–	5	0.001	–	–
	CoMVC	100	100	0.1	0.1	25	5	0.001	–	–
COIL-20	SiMVC	100	100	0.1	20	–	5	0.001	–	–
	CoMVC	100	100	0.1	20	25	5	0.001	–	–
RGB-D	SiMVC	100	100	0.1	0.1	–	5	0.001	–	–
	CoMVC	100	100	0.1	0.1	25	5	0.001	50	0.5

Table 4: Hyperparameters used to train SiMVC and CoMVC.

	SwMPC [4]	RSwMPC [3]	SiMVC	CoMVC
ACC	0.1679	0.2778	0.2717	**0.2892**
NMI	0.0899	0.1810	0.1767	**0.2052**

Table 5: Comparison with [4, 3] on NUS-WIDE-Animal [3].

	EAMC	SiMVC	CoMVC
sec/epoch	33.48	**12.18**	14.28

Table 6: Time spent per training epoch for EAMC, SiMVC and CoMVC on E-FMNIST.

and RGB-D.

3.5. Experiments on NUS-WIDE-Animal

Table 5 shows the performance of our models on NUS-WIDE-Animal [3] (a subset of NUS-WIDE containing the animal classes only) compared to two additional models [4, 3] (as reported in [3]). SiMVC performs comparable, while CoMVC outperforms the competitors.

3.6. Training times

Table 6 shows the average time spent per training epoch for EAMC, SiMVC, and CoMVC. Both SiMVC and CoMVC are more than twice as fast to train per epoch, when compared to EAMC. We believe that this is due to the extra components (attention network, discriminator) included in EAMC. SiMVC is a bit faster to train than CoMVC, due to the extra computations introduced by the contrastive loss.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning*, 2020.

[2] Michael Kampffmeyer, Sigurd Løkse, Filippo M. Bianchi, Lorenzo Livi, Arnt-Børre Salberg, and Robert Jenssen. Deep divergence-based approach to clustering. *Neural Networks*, 113, 2019.

[3] Beilei Wang, Yun Xiao, Zhihui Li, Xuanhong Wang, Xiaojiang Chen, and Dingyi Fang. Robust Self-Weighted Multi-View Projection Clustering. In *AAAI Conference on Artificial Intelligence*, 2020.

[4] R. Wang, F. Nie, Z. Wang, H. Hu, and X. Li. Parameter-Free Weighted Multi-View Projected Clustering with Structured Graph Learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(10), 2020.

[5] Runwu Zhou and Yi-Dong Shen. End-to-End Adversarial-Attention Network for Multi-Modal Clustering. In *Computer Vision and Pattern Recognition*, 2020.

Paper III

On the Effects of Self-supervision and Contrastive Alignment in Deep Multi-view Clustering

Daniel J. Trosten[*], Sigurd Løkse[*], Robert Jenssen[*†‡§], Michael C. Kampffmeyer[*†]
Department of Physics and Technology, UiT The Arctic University of Norway
`firstname[.middle initial].lastname@uit.no`

Abstract

Self-supervised learning is a central component in recent approaches to deep multi-view clustering (MVC). However, we find large variations in the development of self-supervision-based methods for deep MVC, potentially slowing the progress of the field. To address this, we present DeepMVC, a unified framework for deep MVC that includes many recent methods as instances. We leverage our framework to make key observations about the effect of self-supervision, and in particular, drawbacks of aligning representations with contrastive learning. Further, we prove that contrastive alignment can negatively influence cluster separability, and that this effect becomes worse when the number of views increases. Motivated by our findings, we develop several new DeepMVC instances with new forms of self-supervision. We conduct extensive experiments and find that (i) in line with our theoretical findings, contrastive alignments decreases performance on datasets with many views; (ii) all methods benefit from some form of self-supervision; and (iii) our new instances outperform previous methods on several datasets. Based on our results, we suggest several promising directions for future research. To enhance the openness of the field, we provide an open-source implementation of DeepMVC, including recent models and our new instances. Our implementation includes a consistent evaluation protocol, facilitating fair and accurate evaluation of methods and components[1].

1. Introduction

Multi-view clustering (MVC) generalizes the clustering task to data where the instances to be clustered are observed through multiple views, or by multiple modalities. In recent years, deep learning architectures have seen widespread adoption in MVC, resulting in the *deep MVC* subfield. Methods developed within this subfield have shown state-of-the-art clustering performance on several multi-view datasets [14, 19–21, 29, 33], largely outperforming traditional, non-deep-learning-based methods [33].

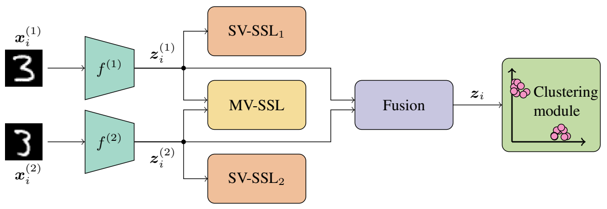


Figure 1. Overview of the DeepMVC framework for a two-view dataset. Different colors denote different components. The framework is generalizable to an arbitrary number of views by adding more view specific encoders (f) and SV-SSL blocks.

Despite these promising developments, we identify significant drawbacks with the current state of the field. Self-supervised learning (SSL) is a crucial component in many recent methods for deep MVC [14, 19–21, 29, 33]. However, the large number of methods, all with unique components and arguments about how they work, makes it challenging to identify clear directions and trends in the development of new components and methods. Methodological research in deep MVC thus lacks foundation and consistent directions for future advancements. This effect is amplified by large variations in implementation and evaluation of new methods. Architectures, data preprocessing and data splits, hyperparameter search strategies, evaluation metrics, and model selection strategies all vary greatly across publications, making it difficult to properly compare methods from different papers. To address these challenges, we present a unified framework for deep MVC, coupled with a rigorous and consistent evaluation protocol, and an open-source implementation. Our main contributions are summarized as follows:

(1) DeepMVC framework. Despite the variations in the development of new methods, we recognize that the majority of recent methods for deep MVC can be decomposed

[*]UiT Machine Learning group (`machine-learning.uit.no`) and Visual Intelligence Centre (`visual-intelligence.no`).
[†]Norwegian Computing Center (`nr.no`).
[‡]Department of Computer Science, University of Copenhagen.
[§]Pioneer Centre for AI (`aicentre.dk`).
[1]Code: `https://github.com/DanielTrosten/DeepMVC`

into the following fixed set of components: (i) view-specific encoders; (ii) single-view SSL; (iii) multi-view SSL; (iv) fusion; and (v) clustering module. The DeepMVC framework (Figure 1) is obtained by organizing these components into a unified deep MVC model. Methods from previous work can thus be regarded as *instances* of DeepMVC.

(2) Theoretical insight on alignment and number of views. Contrastive alignment of view-specific representations is an MV-SSL component that has demonstrated state-of-the-art performance in deep MVC [19]. We study a simplified case of deep MVC, and find that contrastive alignment can only decrease the number of separable clusters in the representation space. Furthermore, we show that this potential negative effect of contrastive alignment becomes worse when the number of views in the dataset increases.

(3) New instances of DeepMVC. Inspired by initial findings from the DeepMVC framework, and our theoretical findings on contrastive alignment, we develop 6 new instances of DeepMVC, which outperform current state-of-the-art methods on several multi-view datasets. The new instances include both novel and well-known types of self-supervision, fusion and clustering modules.

(4) Open-source implementation of DeepMVC and evaluation protocol. We provide an open-source implementation of DeepMVC, including several recent methods and our new instances. The implementation includes a shared evaluation protocol for all methods, and all datasets used in the experimental evaluation. By making the datasets and our implementation openly available, we aim to facilitate simpler development of new methods, as well as rigorous and accurate comparisons between methods and components.

(5) Evaluation of methods and components. We use the implementation of DeepMVC to evaluate and compare several recent state-of-the-art methods and SSL components – both against each other, and against our new instances. In our experiments, we both provide a consistent evaluation of methods in deep MVC, and systematically analyze several SSL-based components – revealing how they behave under different experimental settings.

The main findings from our work are:

- We show that aligning view-specific representations can have a negative impact on cluster separability, especially when the number of views becomes large. In our experiments, we find that contrastive alignment of view-specific representations works well for datasets with few views, but *significantly degrades performance when the number of views increases*. Conversely, we find that maximization of mutual information performs well with many views, while not being as strong with fewer views.
- All methods included in our experiments benefit from at least one form of SSL. In addition to contrastive alignment for few views and mutual information maximization for many views, we find that autoencoder-style reconstruction improves overall performance of methods.
- Properties of the datasets, such as class (im)balance and the number of views, heavily impact the performance of current MVC approaches. There is thus not a single "state-of-the-art" – it instead depends on the datasets considered.
- Results reported by the original authors differ significantly from the performance of our re-implementation for some baseline methods, illustrating the necessity of a unified framework with a consistent evaluation protocol.

2. DeepMVC framework

In this section we present the DeepMVC framework, its components and their purpose, and how they fit together. This allows us to, in the next section, summarize recent work on deep MVC, and illustrate that the majority of recent methods can be regarded as instances of DeepMVC.

Suppose we have a multi-view dataset consisting of n instances and V views, and let $\boldsymbol{x}_i^{(v)}$ be the observation of instance i through view v. The task of the DeepMVC framework is then to cluster the instances into k clusters, and produce cluster membership indicators $\alpha_{ic} \in [0, 1]$, $c = 1, \dots, k$. The framework is illustrated in Figure 1. It consists of the following components.

View-specific encoders. The framework is equipped with V deep neural network encoders $f^{(1)}, \dots, f^{(V)}$, one for each view. Their task is to produce the view-specific representations $\boldsymbol{z}_i^{(v)} = f^{(v)}(\boldsymbol{x}_i^{(v)})$ from the input data.

Single-view self-supervised learning (SV-SSL). The SV-SSL component consists of a set of pretext tasks (auxiliary objectives) that are designed to aid the optimization of the view-specific encoders. Specifically, the tasks should be designed to help the encoders learn representations that simplify the clustering task. Each pretext task is specific to its designated view, and is isolated from all other views.

Multi-view self-supervised learning (MV-SSL). MV-SSL is similar to SV-SSL – they are both self-supervised modules whose goals are to help the encoders learn representations that are suitable for clustering. However, MV-SSL leverages all views simultaneously in the pretext tasks, allowing the model to exploit information from all views simultaneously to learn better features.

Fusion. This component combines view-specific representations into a shared representation for all views. Fusion is typically done using a (weighted) average [11, 19], or by concatenation [5, 26, 29]. More complex fusion modules using *e.g.* attention mechanisms [33], are also possible.

Clustering module (CM). The CM is responsible for determining cluster memberships based on view-specific or fused representations. The CM can consist of a traditional clustering method, such as k-means [13] or Spectral Clustering [16]. Such CMs are applied to the fused representations after other components have been trained, resulting in a two-stage method that first learns fused representations, and then applies a clustering algorithm to these representations.

Alternatively, the CM can be integrated into the model [19, 33], allowing it to be trained alongside other components, resulting in fused representations that are better suited for clustering.

Loss functions and training. The loss functions for the models are specified by the SV-SSL, MV-SSL, and CM components. To train the model, the terms arising from the different components can be minimized simultaneously or they can be minimized in an alternating fashion. It is also possible with pre-training/fine-tuning setups where the model is pre-trained with one subset of the losses and fine-tuned with another subset of the losses.

We note that DeepMVC is a conceptual framework, and that a model is not necessarily completely described by a list of its DeepMVC components. Consequently, it is possible for two models with similar DeepMVC components to have slightly different implementations. This illustrates the importance of our open-source implementation of DeepMVC, which allows the implementation of a model to be completely transparent.

3. Previous methods as instances of DeepMVC

Table 1 shows selected recent methods for deep MVC (the full table can be found in the supplementary), categorized by its DeepMVC components, allowing for systematic comparisons between models[2].

View-specific encoders. As can be seen in Table 1, all models use view-specific encoders to encode views into view-specific embeddings. Multi-layer perceptrons (MLPs) are usually used for vector data, while convolutional neural networks (CNNs) are used for image data.

SV-SSL and MV-SSL. Alongside the encoder network, many methods use decoders to reconstruct the original views from either the view-specific representations or the fused representation. The reconstruction task is the most common self-supervised pretext task, both for SV-SSL and for MV-SSL. In SV-SSL, the views are reconstructed from their respective view-specific representations, without any influence from the other views [1, 17, 18, 22, 28, 31, 32, 35]. In MV-SSL, it is common to either do (i) cross view reconstruction, where all views are reconstructed from all view-specific representations [34]; or (ii) fused view reconstruction, where all views are reconstructed from the fused representation [11, 20, 30, 34].

Aligning distributions of view-specific representations is another MV-SSL pretext task that has been shown to produce representations suitable for clustering [33]. However, [19] demonstrate that the alignment of representation distributions can be detrimental to the clustering performance – especially in the presence of noisy or non-informative views. To avoid these drawbacks, they propose Simple MVC (SiMVC) and Contrastive MVC (CoMVC). In the former, the alignment is dropped altogether, whereas the latter includes a contrastive learning module that aligns the view-specific representations at the instance level, rather than at the distribution level.

Clustering modules. Many deep MVC methods use subspace-based clustering modules [1, 17, 21, 34]. These methods assume that representations, either view-specific or fused, can be decomposed into linear combinations of each other. Once determined, the self-representation matrix containing the coefficients for these linear combinations is used to compute an affinity matrix, which in turn is used as input to spectral clustering. This requires the full $n \times n$ self-representation matrix available in memory, which is computationally prohibitive for datasets with a large number of instances.

Other clustering modules have also been adapted to deep MVC. The clustering module from Deep Embedded Clustering (DEC) [25], for instance, is used in several models [2, 11, 20, 26, 28]. Recently, the Deep Divergence-Based Clustering (DDC) [7] clustering module has been used in several state-of-the-art deep MVC models [19, 33]. In addition, some methods treat either the encoder output or the fused representation as cluster membership vectors [14, 31].

Lastly, some methods adopt a two-stage approach, where they first use the SSL components to learn representations, and then apply a traditional clustering method, such as k-means [4, 5, 29, 32, 35], a Gaussian mixture model [30], or spectral clustering [22], on the trained representations.

4. Contrastive alignment in deep MVC

As can be seen in Table 1, SSL components are crucial in recent state-of-the-art methods for deep MVC. Recent works have focused on aligning view-specific representations [19, 33], and in particular, contrastive alignment [19]. We study a simplified setting where, for each view, all observations in a cluster are located at the same point. This allows us to prove that aligning view-specific representations has a negative impact on the cluster separability after fusion. This is the same starting point as in [19], but we extend the analysis to investigate contrastive alignment when the number of views increases.

Proposition 1 (Adapted from [19]). *Suppose the dataset consists of n instances, V views, and k ground-truth clusters, and that view-specific representations are computed with view-specific encoders as $\boldsymbol{z}_i^{(v)} = f^{(v)}(\boldsymbol{x}_i^{(v)})$. Furthermore, assume that:*

- *For all $v \in \{1, \dots V\}$ and $j \in \{1, \dots, k\}$,*
$$\forall i \in \mathcal{C}_j, \boldsymbol{x}_i^{(v)} = \boldsymbol{c}^{(v)} \in \{\boldsymbol{c}_1^{(v)}, \dots, \boldsymbol{c}_{k_v}^{(v)}\} \qquad (1)$$

[2]Note, here we limit our discussion to MVC approaches without missing data. While most of the theoretical and empirical results also generalize to the emerging incomplete MVC setting [12, 23, 27], we consider it out of scope of this work.

Model	Pub.	Enc.	SV-SSL	MV-SSL	Fusion	CM
DCCAE [22]	ICML'15	MLP	Reconstruction	CCA	1st view	SC
DMSC [1]	J. STSP'18	CNN	Reconstruction	–	Affinity fusion	SR, SC
MvSCN [5]	IJCAI'19	MLP	Sp. Emb.	MSE Al.	Concat.	k-means
DAMC [11]	IJCAI'19	MLP	–	Reconstruction	Average	DEC
SGLR-MVC [30]	AAAI'20	MLP	Variational Reconstruction	Variational Reconstruction	Weighted sum	GMM
EAMC [33]	CVPR'20	MLP	–	Distribution Al., Kernel Al.	Attention	DDC
SiMVC [19]	CVPR'21	MLP/CNN	–	–	Weighted sum	DDC
CoMVC [19]	CVPR'21	MLP/CNN	–	Contrastive Al.	Weighted sum	DDC
Multi-VAE [29]	ICCV'21	CNN	–	Variational Reconstruction	Concat.	Gumbel, k-means
DMIM [14]	IJCAI'21	MLP	Min. superflous information	Max. shared information	?	Encoder output

Model	Category	Enc.	SV-SSL	MV-SSL	Fusion	CM
AE–KM	Simple	MLP/CNN	Reconstruction	–	Concat.	k-means
AE–DDC	Simple	MLP/CNN	Reconstruction	–	Weighted sum	DDC
AECoKM	Contrastive Al.	MLP/CNN	Reconstruction	Contrastive Al.	Concat.	k-means
AECoDDC	Contrastive Al.	MLP/CNN	Reconstruction	Contrastive Al.	Weighted sum	DDC
InfoDDC	Mutual info.	MLP/CNN	–	Max. mutual info.	Weighted sum	DDC
MV-IIC	Mutual info.	MLP/CNN	–	IIC Overclustering	–	IIC, k-means

Table 1. Overview of selected methods from previous work (top) and proposed new instances (bottom), and their DeepMVC components. The complete table of previous methods is included in the supplementary. **Abbreviations:** "–" = Not included, "?" = Not specified, Al. = Alignment, Concat. = Concatenate, CCA = Canonical correlation analysis, DDC = Deep divergence-based clustering, DEC = Deep embedded clustering, SC = Spectral clustering, Sp. Emb. = Spectral Embedding, SR = Self-representation

where $\mathcal{C}_j$ is the set of indices for instances in cluster j, and $k_v \in \{1, \dots, k\}$ is the number of separable clusters in view v.

- *Representations are fused as $\boldsymbol{z}_i = \sum_{v=1}^{V} w_v \boldsymbol{z}_i^{(v)}$ where $w_1, \dots, w_V$ are all unique.*
- *For all $j \in \{1, \dots, k\}$,*
$$\forall i \in \mathcal{C}_j, \boldsymbol{z}_i = \boldsymbol{z}^\star \in \{\boldsymbol{z}_1^\star, \dots, \boldsymbol{z}_\kappa^\star\} \tag{2}$$

Then if $\boldsymbol{z}_i^{(1)} = \dots = \boldsymbol{z}_i^{(V)}$ (perfectly aligned view-specific representations),

$$\kappa = \min\{k, (\min_{v=1,\dots,V} \{k_v\})^V\} \tag{3}$$

Proof. See [19]. □

According to Proposition 1, when the view-specific representations are perfectly aligned, the number of separable clusters after fusion, κ, depends on the number of separable clusters in the *least informative view* – the view with the lowest k_v. The following propositions show what happens to $\min\{k_v\}$ when the number of views increases[3].

Proposition 2. *Suppose $k_v, v \in \mathbb{N}$ are random variables taking values in $\{1, \dots, k\}$. Then, for any $V \geq 1$,*

$$\mathbb{P}\{\min_{v=1,\dots,V+1} \{k_v\} \leq \min_{v=1,\dots,V} \{k_v\} \;\Big|\; k_1, \dots, k_V\} = 1 \tag{4}$$

[3]The proofs of Propositions 2 and 3 are given in the supplementary

Proposition 3. *Suppose $k_v, v \in \mathbb{N}$ are iid. random variables taking values in $\{1, \dots, k\}$. Then, for any $V \geq 1$,*

$$\mathbb{E}(\min_{v=1,\dots,V+1} \{k_v\}) \leq \mathbb{E}(\min_{v=1,\dots,V} \{k_v\}) \tag{5}$$

Assuming the view-specific representations are perfectly aligned, Propositions 2 and 3 show that: (i) Given a number of views, adding another view will, with probability 1, not increase $\min\{k_v\}$. (ii) Among two datasets with the same distribution for the k_v, the dataset with the *smallest number of views* will have the highest expected value of $\min\{k_v\}$.

In summary, we have shown that contrastive alignment-based models perform worse when the number of views in a dataset increases. These findings are supported by the experimental results in Figure 2 and Table 2 which show that, when the number of views increases, the contrastive alignment-based model is outperformed by the model without any alignment.

Alignment as a pretext task. In contrast to our theoretical findings in the simplified case, Figure 2 and Table 2 show that contrastive alignment can sometimes be beneficial for the performance, particularly when the number of views is small. This is because alignment might be a good pretext task that helps the encoders learn informative representations, by learning to represent the information that is shared across views. However, we emphasize that this is only true when the number of views is small (≤ 4 in Figure 2), meaning that alignment should be used with caution when the number of views increases beyond this point.

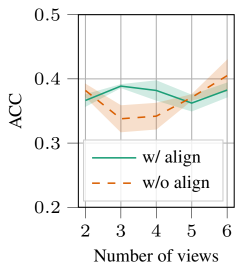


Figure 2. Clustering accuracy for an increasing number of views on Caltech7.

Dataset	w/o align	w/ align
Edge-MNIST (2 views)	0.89	**0.97**
Caltech7 (6 views)	**0.41**	0.38
Patched-MNIST (12 views)	**0.84**	0.73

Table 2. Clustering accuracies on datasets with varying number of views.

5. New instances of DeepMVC

With our new instances of DeepMVC, we aim to further analyze and address the many-views-issue with contrastive alignment highlighted above, as well as to investigate the effect of other SSL components. In addition to alignment of view-specific representations [2, 19, 33], we identify reconstruction [1, 11, 22] and mutual information maximization [6, 21] to be promising directions for the new instances. Maximizing mutual information is particularly interesting, as it enables the view-specific encoders to represent the information which is shared across views, without explicitly forcing the view-specific representations to be aligned. Furthermore, we recognize that simple baselines with few or no SSL components – exemplified by SiMVC [19] – might perform similarly to more complicated methods, while being significantly easier to implement and faster to train. It is therefore crucial to include such methods in an experimental evaluation, in order to properly determine whether additional SSL-based components are beneficial for the models' performance. Finally, our overview of recent work shows that both traditional clustering modules (*e.g.* k-means) and deep learning-based clustering modules (*e.g.* DDC) are commonly used in deep MVC.

In total, we develop 6 new DeepMVC instances in 3 categories. The new instances are summarized in Table 1. Evaluating these instances and several methods from recent work, allows us to accurately evaluate methods and components, and investigate how they behave for datasets with varying characteristics.

Simple baselines: AE–KM has view-specific autoencoders (AEs) with a mean-squared-error (MSE) loss

$$\mathcal{L}^{\text{SV}}_{\text{Reconstruction}} = \frac{1}{nV} \sum_{i=1}^{n} \sum_{v=1}^{V} ||\boldsymbol{x}_i^{(v)} - \hat{\boldsymbol{x}}_i^{(v)}||^2 \tag{6}$$

as its SV-SSL task. The views are fused by concatenation and the concatenated representations are clustered using k-means after the view-specific autoencoders have been trained. **AE–DDC** uses view-specific autoencoders with an MSE loss (Eq. (6)) as its SV-SSL task. The views are fused using a weighted sum and the fused representations are clustered using the DDC clustering module [7].

Contrastive alignment-based: AECoKM extends AE–KM with a contrastive loss on the view-specific representations. We use the multi-view generalization of the NT-Xent (contrastive) loss by Trosten *et al.* [19], without the "other clusters" negative sampling

$$\mathcal{L}^{\text{MV}}_{\text{Contrastive}} = \frac{1}{nV(V-1)} \sum_{i=1}^{n} \sum_{v=1}^{V} \sum_{u=1}^{V} \mathbb{1}_{\{u \neq v\}} \, \ell_i^{(uv)}, \tag{7}$$

$$\ell_i^{(uv)} = -\log \frac{\exp(s_{ii}^{(uv)})}{\sum_{s' \in \text{Neg}(\boldsymbol{z}_i^{(u)}, \boldsymbol{z}_i^{(v)})} \exp(s')} \tag{8}$$

and $s_{ij}^{(uv)} = \frac{1}{\tau} \frac{\boldsymbol{z}_i^u \cdot \boldsymbol{z}_j^{(v)}}{||\boldsymbol{z}_i^u|| \cdot ||\boldsymbol{z}_j^{(v)}||}$ denotes the cosine similarity between $\boldsymbol{z}_i^u$ and $\boldsymbol{z}_j^v$. The set $\text{Neg}(\boldsymbol{z}_i^{(u)}, \boldsymbol{z}_i^{(v)})$ is the set of similarities of negative pairs for the positive pair $(\boldsymbol{z}_i^{(u)}, \boldsymbol{z}_i^{(v)})$, which consists of $s_{ij}^{(uv)}$, $s_{ij}^{(uu)}$, and $s_{ij}^{(vv)}$, for all $j \neq i$. τ is a hyperparameter, which we set to 0.1 for all experiments. **AECoDDC** extends AE–DDC using the same generalized NT-Xent contrastive loss on the view-specific representations.

Mutual information-based: InfoDDC maximizes the mutual information (MI) between the view-specific representations, using the MI loss from Invariant Information Clustering (IIC) [6][4]. The MI maximization is regularized by also maximizing the entropy of view-specific representations. The view-specific representations are fused using a weighted sum, and the fused representations are clustered using DDC [7]. **MV-IIC** is a multi-view generalization of IIC [6], where cluster assignments are computed for each of the view-specific representations. The MI between pairs of these view-specific cluster assignments is then maximized using the information maximization loss from IIC. In order to get a final shared cluster assignment for all views, the view-specific cluster assignments are concatenated and clustered using k-means. As in IIC, this model includes 5 over-clustering heads as its MV-SSL task. In both InfoDDC and MV-IIC, we generalize the loss from IIC to an arbitrary number of views:

$$\mathcal{L}^{\text{MV}}_{\text{MI}} = \frac{2}{V(V-1)} \sum_{u=1}^{V-1} \sum_{v=u+1}^{V} - \Big(\underbrace{I(\boldsymbol{Z}^{(u)}, \boldsymbol{Z}^{(v)})}_{\text{mutual information}} + (\lambda - 1) \underbrace{(H(\boldsymbol{Z}^{(u)}) + H(\boldsymbol{Z}^{(v)}))}_{\text{entropy regularization}} \Big) \tag{9}$$

[4]The supplementary includes a brief overview of the connection between InfoDDC and contrastive alignment.

where the summands are computed as

$$I(\boldsymbol{Z}^{(u)}, \boldsymbol{Z}^{(v)}) + (\lambda - 1)(H(\boldsymbol{Z}^{(u)}) + H(\boldsymbol{Z}^{(v)})) \\ = -\sum_{a=1}^{D}\sum_{b=1}^{D} \boldsymbol{P}_{ab}^{(uv)} \log \frac{\boldsymbol{P}_{ab}^{(uv)}}{(\boldsymbol{P}_a^{(u)}\boldsymbol{P}_b^{(v)})^\lambda}, \tag{10}$$

where D denotes the dimensionality of the view-specific representations. λ is a hyperparameter that controls the strength of the entropy regularization. We set $\lambda = 10$ for InfoDDC, and $\lambda = 1.5$ for MV-IIC. The joint distribution $\boldsymbol{P}^{(uv)}$ is estimated by first computing $\tilde{\boldsymbol{P}}^{(uv)} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{z}_i^{(u)}(\boldsymbol{z}_i^{(v)})^\top$, and then symmetrizing it $\boldsymbol{P}^{(uv)} = \frac{1}{2}(\tilde{\boldsymbol{P}}^{(uv)} + (\tilde{\boldsymbol{P}}^{(uv)})^\top)$. We assume that each view-specific representation is normalized such that its elements sum to one, and are all non-negative. The marginals $\boldsymbol{P}^{(u)}$ and $\boldsymbol{P}^{(v)}$ are obtained by summing over the rows and columns of $\boldsymbol{P}^{(uv)}$, respectively.

6. Experiments

In this section we provide a rigorous evaluation of methods and their DeepMVC components. Inspired by the initial findings in Section 4 and our overview of recent methods in Section 3, we focus mainly on the SSL and CM components in our evaluation. We found these components to be most influential on the methods' performance. For completeness, we include experiments with different fusion and CM components in the supplementary.

6.1. Setup

Baselines. In addition to the new instances presented in Section 5, we include 6 baseline models from previous work in our experiments. The following baseline models were selected to include a diverse set of framework components in the evaluation: (i) Deep Multimodal Subspace Clustering (DMSC) [1]; (ii) Multi-view Spectral Clustering Network (MvSCN) [5]; (iii) End-to-end Adversarial-attention Multimodal Clustering (EAMC) [33]; (iv) Simple Multi-View Clustering (SiMVC) [19]; (v) Contrastive Multi-View Clustering (CoMVC) [19]; (vi) Multi-view Variational Autoencoder (Multi-VAE) [29].

As can be seen in Table 1, this collection of models includes both reconstruction-based and alignment-based SSL, as well as traditional (k-means and spectral) and deep learning-based CMs. They also include several fusion strategies and encoder networks. Section 6.3 includes an ablation study that examines the influence of SSL components in these models.

Datasets. We evaluate the baselines and new instances on 8 widely used benchmark datasets for deep MVC. We prioritize datasets that were also used in the original publications for the selected baselines. Not only does this result in a diverse collection of datasets common in deep MVC – it also allows us to compare the performance of our implementations to what was reported by the original authors. The results of this comparison are given in the supplementary.

The following datasets are used for evaluation: (i) **NoisyMNIST / NoisyFashion**: A version of MNIST [9] / FashionMNIST [24] where the first view contains the original image, and the second view contains an image sampled from the same class as the first image, with added Gaussian noise ($\sigma = 0.2$). (ii) **EdgeMNIST / EdgeFashion**: Another version of MNIST / FashionMNIST where the first view contains the original image, and the second view contains an edge-detected version of the same image. (iii) **COIL-20**: The original COIL-20 [15] dataset, where we randomly group the images of each object into groups of size 3, resulting in a 3-view dataset. (iv) **Caltech7 / Caltech20**: A subset of the Caltech101 [3] dataset including 7 / 20 classes. We use the 6 different features extracted by Li *et al.* [10], resulting in a 6-view dataset[5]. (v) **PatchedMNIST**: A subset of MNIST containing the first three digits, where views are extracted as 7×7 non-overlapping patches of the original image. The corner patches are dropped as they often contain little information about the digit, resulting in a dataset with 12 views. Each patch is resized to 28×28.

All views are individually normalized so that the values lie in $[0, 1]$. Following recent work on deep MVC, we train and evaluate on the full datasets [14, 19, 29, 33]. More dataset details are provided in the supplementary.

Hyperparameters. The baselines use the hyperparameters reported by the original authors, because (i) it is not feasible for us to tune hyperparameters individually for each model on each dataset; and (ii) it is difficult to tune hyperparameters in a realistic clustering setting due to the lack of labeled validation data. For each method, the same hyperparameter configuration is used for all datasets.

New instances use the same hyperparameters as for the baselines wherever possible[6]. Otherwise, we set hyperparameters such that loss terms have the same order of magnitude, and such that the training converges. We refrain from any hyperparameter tuning that includes the dataset labels to keep the evaluation fair and unsupervised. We include a hyperparameter sweep in the supplementary, in order to assess the new instances' sensitivity to changes in their hyperparameter. However, we emphasize that the results of this sweep were *not* used to select hyperparameters for the new instances. All models use the same encoder architectures and are trained for 100 epochs with the Adam optimizer [8].

Evaluation protocol. We train each model from 5 different initializations. Then we select the run that resulted in the lowest value of the loss and report the performance metrics from that run, following [7, 19]. This evaluation protocol is both fully unsupervised, and is not as impacted by poorly performing runs, as for instance the mean performance of all

[5]The list of classes and feature types is included in the supplementary.

[6]Hyperparameters for all models are listed in the supplementary.

(a) **All datasets**

Model	BL	$\bar{Z}$
MvSCN	✘	−2.23
AECoKM	✘	−0.32
EAMC	✘	−0.22
DMSC	✘	−0.11
AE–KM	✓	0.16
InfoDDC	✘	0.20
AE–DDC	✓	0.26
SiMVC	✓	0.27
MV-IIC	✘	0.27
CoMVC	✘	0.29
Multi-VAE	✘	0.43
AECoDDC	✘	0.65

(b) **Random pairings**

Model	CA	$\bar{Z}$
MvSCN	✘	−2.49
DMSC	✘	−0.54
InfoDDC	✘	−0.41
EAMC	✘	−0.17
MV-IIC	✘	0.05
AE–KM	✘	0.11
Multi-VAE	✘	0.32
SiMVC	✘	0.35
AE–DDC	✘	0.56
AECoKM	✓	0.59
CoMVC	✓	0.63
AECoDDC	✓	0.82

(c) **Many views**

Model	MI	CA	$\bar{Z}$
MvSCN	✘	✘	−1.78
AECoKM	✘	✓	−0.83
EAMC	✘	✘	−0.75
AE–DDC	✘	✘	−0.36
CoMVC	✘	✓	−0.33
SiMVC	✘	✘	−0.12
AE–KM	✘	✘	0.23
AECoDDC	✘	✓	0.28
Multi-VAE	✘	✘	0.38
DMSC	✘	✘	0.45
MV-IIC	✓	✘	0.98
InfoDDC	✓	✘	1.15

(d) **Balanced vs. imbalanced**

Model	DDC	$\bar{Z}_{\text{bal}}$	$\bar{Z}_{\text{imb}}$
MvSCN	✘	−2.41	−1.78
DMSC	✘	−0.39	0.45
InfoDDC	✓	−0.13	1.18
EAMC	✓	0.00	−0.75
MV-IIC	✘	0.01	1.06
AE–KM	✘	0.03	0.56
AECoKM	✘	0.08	−1.54
CoMVC	✓	0.30	0.25
SiMVC	✓	0.31	0.16
AE–DDC	✓	0.33	0.06
Multi-VAE	✘	0.42	0.47
AECoDDC	✓	0.92	−0.13

Table 3. Aggregated evaluation results for the dataset groups. Models are sorted from lowest to highest by average Z-score for each group. Higher Z-scores indicate better clusterings. Our new instances are underlined. **Abbreviations:** BL = Simple baseline, CA = Contrastive alignment, DDC = Deep divergence-based clustering MI = Mutual information, $\bar{Z}$ = Average Z-score for group.

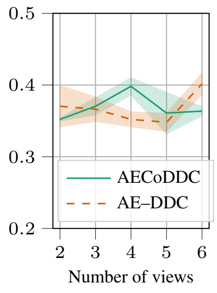


Figure 3. Accuracies on Caltech7 with increasing number of views.

runs. The uncertainty of the performance metric under this model selection protocol is estimated using bootstrapping[7]. We measure clustering performance with the accuracy (ACC) and normalized mutual information (NMI). Both metrics are bounded in $[0, 1]$, and higher values correspond to better performing models, with respect to the ground truth labels.

6.2. Evaluation results

To emphasize the findings from our experiments, we compute the average Z-score for each model, for 4 groups of datasets[8]. Z-scores are calculated by subtracting the mean and dividing by the standard deviation of results, per dataset and per metric. Table 3 shows Z-scores for the groups: (i) **All datasets**. (ii) **Random pairings:** Datasets generated by randomly pairing within-class instances to synthesize multiple views (NoisyMNIST, NoisyFashion, COIL-20). (iii) **Many views:** Datasets with many views (Caltech7, Caltech20, PatchedMNIST). (iv) **Balanced vs. imbalanced:** Datasets with balanced classes (NoisyMNIST, NoisyFashion, EdgeMNIST, EdgeFashion, COIL-20, PatchedMNIST) vs. datasets with imbalanced classes (Caltech7, Caltech20). Our main experimental findings are:

Dataset properties significantly impact the performance of methods. We observe that the ranking of methods varies significantly based on dataset properties, such as the number of views (Table 3c) and class (im)balance (Table 3d). Hence, there is not a single "state-of-the-art" for all datasets.

Our new instances outperform previous methods. In Table 3a we see that the simple baselines perform remarkably well, when compared to the other, more complex methods. This highlights the importance of including simple baselines like these in the evaluation. Table 3a shows that AECoDDC overall outperforms the other methods, and on datasets with many views (Table 3c) we find that InfoDDC and MV-IIC outperform the others by a large margin.

[7]Details on uncertainty computations are included in the supplementary.
[8]Results for all methods/datasets are included in the supplementary.

Maximization of mutual information outperforms contrastive alignment on datasets with many views. Contrastive alignment-based methods show good overall performance, but they struggle when the number of views becomes large (Table 3c). This holds for both baseline methods (as observed in Section 4), and the new instances. As in Section 4, we hypothesize that this is due to issues with representation alignment, where the presence of less informative views is more likely when the number of views becomes large. Contrastive alignment attempts to align view-specific representations to this less informative view, resulting in clusters that are harder to separate in the representation space. This is further verified in Figures 2 and 3, illustrating a decrease in performance on Caltech7 for contrastive alignment-based models with 5 or 6 views. Models based on maximization of mutual information do not have the same problem. We hypothesize that this is because maximizing mutual information still allows the view-specific representations to be different, avoiding the above issues with alignment. The MI-based models also include regularization terms that maximize the entropy of view-specific representations, preventing the representations from collapsing to a single value.

Contrastive alignment works particularly well on datasets consisting of random pairings (Table 3b). In these datasets, the class label is the only thing the views have in common. Contrastive alignment, *i.e.* learning a shared representation for all pairs within a class, thus asymptotically amounts to learning a unique representation for each class, making it easier for the CM to separate between classes.

The DDC CM performs better than the other CMs on balanced datasets. With the DDC CM, the models are end-to-end trainable – jointly optimizing all components in the model. The view-specific representations can thus be adapted to suit the CM, potentially improving the clustering result. DDC also has an inherent bias towards balanced clusters [7], which helps produce better clusterings when the ground truth classes are balanced.

Model	NoisyMNIST w/o SV-SSL	NoisyMNIST w/ SV-SSL	Caltech7 w/o SV-SSL	Caltech7 w/ SV-SSL
DMSC	0.54	0.66 (+0.12)	0.35	0.50 (+0.15)
AE–DDC	1.00	1.00 (0.00)	0.41	0.40 (0.00)
AE–KM	0.67	0.74 (+0.07)	0.39	0.44 (+0.05)
AECoDDC	1.00	1.00 (0.00)	0.38	0.36 (−0.02)
AECoKM	0.56	1.00 (+0.44)	0.22	0.20 (−0.02)
Model	w/o MV-SSL	w/ MV-SSL	w/o MV-SSL	w/ MV-SSL
EAMC	1.00	0.83 (−0.17)	0.36	0.44 (+0.08)
Multi-VAE	0.52	0.98 (+0.46)	0.31	0.47 (+0.15)
CoMVC	1.00	1.00 (0.00)	0.41	0.38 (−0.02)
AECoDDC	1.00	1.00 (0.00)	0.40	0.36 (−0.04)
AECoKM	0.74	1.00 (+0.26)	0.44	0.20 (−0.24)
InfoDDC	1.00	0.90 (−0.10)	0.41	0.51 (+0.10)
MV-IIC	0.52	0.52 (0.00)	0.53	0.53 (0.00)

Table 4. Accuracies from ablation studies with SSL components.

Reproducibility of original results. During our experiments we encountered issues with reproducibility with several of the methods from previous work. In the supplementary we include a comparison between our results and those reported by the original authors of the methods from previous work. We find that most methods use different network architectures and evaluation protocols in the original publications, making it difficult to accurately compare performance between methods and their implementations. This illustrates the difficulty of reproducing and comparing results in deep MVC, highlighting the need for a unified framework with a consistent evaluation protocol and an open-source implementation.

6.3. Effect of SSL components

Table 4 shows the results of ablation studies with the SV-SSL and MV-SSL components. These results show that having at least one form of SSL is beneficial for the performance of all models, with the exception being AE–DDC/AECoDDC, which on Caltech7 performs best without any self-supervision. We suspect that this particular result is due to the issues with many views and class imbalance discussed in Section 6.2. Further, we observe that having both forms of SSL is not always necessary. For instance is there no difference with and without SV-SSL for AECoDDC and AECoKM, both of which include contrastive alignment-based MV-SSL. Lastly, we note that contrastive alignment-based MV-SSL decreases performance on Caltech7 for most models. This is consistent with our theoretical findings in Section 4, as well as the results in Section 6.2 and in Figures 2 and 3 – illustrating that contrastive alignment is not suitable for datasets with a large number of views.

7. Conclusion

We investigate the role of self-supervised learning (SSL) in deep MVC. Due to its recent success, we focus particularly on contrastive alignment, and prove that it can be detrimental to the clustering performance, especially when the number of views becomes large. To properly evaluate models and components, we develop DeepMVC – a new unified framework for deep MVC, including the majority of recent methods as instances. By leveraging the new insight from our framework and theoretical findings, we develop 6 new DeepMVC instances with several promising forms of SSL, which perform remarkably well compared to previous methods. We conduct a thorough experimental evaluation of our new instances, previous methods, and their DeepMVC components – and find that SSL is a crucial component in state-of-the-art methods for deep MVC. In line with our theoretical analysis, we observe that contrastive alignment worsens performance when the number of views becomes large. Further, we find that performance of methods depends heavily on dataset characteristics, such as number of views, and class imbalance. Developing methods that are robust towards changes in these properties can thus result in methods that perform well over a wide range of multi-view clustering problems. To this end, we make the following recommendations for future work in deep MVC:

Improving contrastive alignment or maximization of mutual information to handle both few and many views. Addressing pitfalls of alignment to improve contrastive alignment-based methods on many views, is a promising direction for future research. Similarly, we believe that improving the methods based on maximization of mutual information on few views, will result in better models.

Developing end-to-end trainable clustering modules that are not biased towards balanced clusters. The performance of the DDC clustering module illustrates the potential of end-to-end trainable clustering modules, which are capable of adapting the representations to produce better clusterings. Mitigating the bias towards balanced clusters thus has the potential to produce models that perform well, both on balanced and imbalanced datasets.

Proper evaluation and open-source implementations. Finally, we emphasize the importance of evaluating new methods on a representative collection of datasets, *e.g.* many views and few views, paired, imbalanced, *etc*. Also, in the reproducibility study (see supplementary), we find that original results can be difficult to reproduce. We therefore encourage others to use the open-source implementation of DeepMVC, as open code and datasets, and consistent evaluation protocols, are crucial to properly evaluate models and facilitate further development of new methods and components.

Acknowledgements

This work was financially supported by the Research Council of Norway (RCN), through its Centre for Research-based Innovation funding scheme (Visual Intelligence, grant no. 309439), and Consortium Partners. It was further funded by RCN FRIPRO grant no. 315029, RCN IKTPLUSS grant no. 303514, and the UiT Thematic Initiative "Data-Driven Health Technology".

References

[1] Mahdi Abavisani and Vishal M. Patel. Deep Multimodal Subspace Clustering Networks. *IEEE Journal of Selected Topics in Signal Processing*, 12(6):1601–1614, 2018. 3, 4, 5, 6

[2] Guowang Du, Lihua Zhou, Yudi Yang, Kevin Lü, and Lizhen Wang. Deep Multiple Auto-Encoder-Based Multi-view Clustering. *Data Science and Engineering*, 6(3):323–338, 2021. 3, 5

[3] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007. 6

[4] Shuning Huang, Kaoru Ota, Mianxiong Dong, and Fanzhang Li. MultiSpectralNet: Spectral Clustering Using Deep Neural Network for Multi-View Data. *IEEE Transactions on Computational Social Systems*, 6(4):749–760, 2019. 3

[5] Zhenyu Huang, Joey Tianyi Zhou, Xi Peng, Changqing Zhang, Hongyuan Zhu, and Jiancheng Lv. Multi-view Spectral Clustering Network. In *IJCAI*, 2019. 2, 3, 4, 6

[6] Xu Ji, Andrea Vedaldi, and Joao Henriques. Invariant Information Clustering for Unsupervised Image Classification and Segmentation. In *ICCV*, 2019. 5

[7] Michael Kampffmeyer, Sigurd Løkse, Filippo M. Bianchi, Lorenzo Livi, Arnt-Børre Salberg, and Robert Jenssen. Deep Divergence-based Approach to Clustering. *Neural Networks*, 113:91–101, 2019. 3, 5, 6, 7

[8] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 6

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 6

[10] Yeqing Li, Feiping Nie, Heng Huang, and Junzhou Huang. Large-Scale Multi-View Spectral Clustering via Bipartite Graph. In *AAAI*, 2015. 6

[11] Z. Li, Q. Wang, Z. Tao, Q. Gao, and Z. Yang. Deep Adversarial Multi-view Clustering Network. In *IJCAI*, 2019. 2, 3, 4, 5

[12] Yijie Lin, Yuanbiao Gou, Zitao Liu, Boyun Li, Jiancheng Lv, and Xi Peng. COMPLETER: Incomplete Multi-View Clustering via Contrastive Prediction. In *CVPR*, 2021. 3

[13] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, 1967. 2

[14] Yiqiao Mao, Xiaoqiang Yan, Qiang Guo, and Yangdong Ye. Deep Mutual Information Maximin for Cross-Modal Clustering. In *IJCAI*, 2021. 1, 3, 4, 6

[15] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-20). Technical report, 1996. 6

[16] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. 2

[17] Xiukun Sun, Miaomiao Cheng, Chen Min, and Liping Jing. Self-Supervised Deep Multi-View Subspace Clustering. In *ACML*, 2019. 3

[18] Xiaoliang Tang, Xuan Tang, Wanli Wang, Li Fang, and Xian Wei. Deep Multi-view Sparse Subspace Clustering. In *ICNCC*, 2018. 3

[19] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. Reconsidering Representation Alignment for Multi-view Clustering. In *CVPR*, 2021. 1, 2, 3, 4, 5, 6

[20] Qianqian Wang, Zhiqiang Tao, Wei Xia, Quanxue Gao, Xiaochun Cao, and Licheng Jiao. Adversarial Multiview Clustering Networks With Adaptive Fusion. *IEEE transactions on neural networks and learning systems*, PP, 2022. 3

[21] Shiye Wang, Changsheng Li, Yanming Li, Ye Yuan, and Guoren Wang. Self-Supervised Information Bottleneck for Deep Multi-View Subspace Clustering. *arXiv:2204.12496 [cs]*, 2022. 1, 3, 5

[22] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On Deep Multi-View Representation Learning. In *ICML*, 2015. 3, 4, 5

[23] Jie Wen, Zheng Zhang, Guo-Sen Xie, Lunke Fei, Bob Zhang, and Yong Xu. CDIMC-net: Cognitive Deep Incomplete Multi-view Clustering Network. In *IJCAI*, 2020. 3

[24] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747 [cs, stat]*, 2017. 6

[25] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised Deep Embedding for Clustering Analysis. In *ICML*, 2016. 3

[26] Bowen Xin, Shan Zeng, and Xiuying Wang. Self-Supervised Deep Correlational Multi-View Clustering. In *IJCNN*, 2021. 2, 3

[27] Cai Xu, Ziyu Guan, Wei Zhao, Hongchang Wu, Yunfei Niu, and Beilei Ling. Adversarial Incomplete Multiview Clustering. In *IJCAI*, 2019. 3

[28] Jie Xu, Yazhou Ren, Guofeng Li, Lili Pan, Ce Zhu, and Zenglin Xu. Deep embedded multi-view clustering with collaborative training. *Information Sciences*, 573:279–290, 2021. 3

[29] Jie Xu, Yazhou Ren, Huayi Tang, Xiorong Pu, Xiaofeng Zhu, Ming Zeng, and Lifang He. Multi-VAE: Learning Disentangled View-Common and View-Peculiar Visual Representations for Multi-View Clustering. In *ICCV*, 2021. 1, 2, 3, 4, 6

[30] Ming Yin, Weitian Huang, and Junbin Gao. Shared

Generative Latent Representation Learning for Multi-View Clustering. In *AAAI*, 2020. 3, 4

[31] Xianchao Zhang, Jie Mu, Linlin Zong, and Xiaochun Yang. End-To-End Deep Multimodal Clustering. In *ICME*, 2020. 3

[32] Xianchao Zhang, Xiaorui Tang, Linlin Zong, Xinyue Liu, and Jie Mu. Deep Multimodal Clustering with Cross Reconstruction. In *PAKDD*, 2020. 3

[33] Runwu Zhou and Yi-Dong Shen. End-to-End Adversarial-Attention Network for Multi-Modal Clustering. In *CVPR*, 2020. 1, 2, 3, 4, 5, 6

[34] Pengfei Zhu, Binyuan Hui, Changqing Zhang, Dawei Du, Longyin Wen, and Qinghua Hu. Multi-view Deep Subspace Clustering Networks. *arXiv:1908.01978 [cs]*, 2019. 3

[35] Linlin Zong, Faqiang Miao, Xianchao Zhang, and Bo Xu. Multimodal Clustering via Deep Commonness and Uniqueness Mining. In *ICIKM*, 2020. 3

Supplementary Material – On the Effects of Self-supervision and Contrastive Alignment in Deep Multi-view Clustering

Daniel J. Trosten[*], Sigurd Løkse[*], Robert Jenssen[*†‡§], Michael C. Kampffmeyer[*†]
Department of Physics and Technology, UiT The Arctic University of Norway
firstname[.middle initial].lastname@uit.no

1. Introduction

Here, we provide the proofs for Propositions 2 and 3; additional details on the proposed new instances of DeepMVC; the datasets used for evaluation; the hyperparameters used by baselines and new instances; and the computation of metrics and uncertainties used in our evaluation protocol. We also include the full list of recent methods and their DeepMVC components, the complete table of results from the experimental evaluation. In addition, we include additional experiments and analyses of reproducibility, hyperparameters, and the Fusion and CM components. Finally, we reflect on possible negative societal impacts of our work.

Our implementation of the DeepMVC framework, as well as the datasets and the evaluation protocol used in our experiments, is available at https://github.com/DanielTrosten/DeepMVC. See README.md in the repository for more details about the implementation, and how to reproduce our results.

2. Previous methods as instances of DeepMVC

The full list of recent methods and their DeepMVC components is given in Table 1. We observe that all but one model includes at least one form of SSL, but the type of SSL, and also fusion and CM, vary significantly for the different models. This illustrates the importance of the SSL components in deep MVC, as well as the need for a unified framework with a consistent evaluation protocol, in order to properly compare and evaluate methods.

3. Contrastive alignment in deep MVC

Proof of propositions

Proposition 2. *Suppose $k_v, v \in \mathbb{N}$ are random variables taking values in $\{1, \dots, k\}$. Then, for any $V \geq 1$,*

$$\mathbb{P}\{\min_{v=1,\dots,V+1} \{k_v\} \leq \min_{v=1,\dots,V} \{k_v\} \;\Big|\; k_1, \dots, k_V\} = 1 \tag{1}$$

Proof. Let $M_V = \min_{v=1,\dots,V} \{k_v\}$, then we need to prove that

$$\mathbb{P}(M_{V+1} \leq M_V \mid k_1, \dots, k_V) = 1. \tag{2}$$

Due to the properties of the minimum operator, we have

$$\begin{cases} M_{V+1} = M_V, & \text{if } k_{V+1} \geq M_V \\ M_{V+1} < M_V, & \text{otherwise} \end{cases}. \tag{3}$$

Hence, $M_{V+1} \leq M_V$ regardless of the value of k_{V+1}, which gives

$$\mathbb{P}(M_{V+1} \leq M_V \mid k_1, \dots, k_V) = 1. \tag{4}$$

□

Proposition 3. *Suppose $k_v, v \in \mathbb{N}$ are iid. random variables taking values in $\{1, \dots, k\}$. Then, for any $V \geq 1$,*

$$\mathbb{E}(\min_{v=1,\dots,V+1} \{k_v\}) \leq \mathbb{E}(\min_{v=1,\dots,V} \{k_v\}) \tag{5}$$

Proof. Let $M_V = \min_{v=1,\dots,V} \{k_v\}$, then

$$\begin{aligned} F_{M_V}(x) &:= \mathbb{P}(M_V \leq x) = 1 - \mathbb{P}(M_V > x) && (6) \\ &= 1 - \mathbb{P}(k_1 > x \cap \dots \cap k_V > x) && (7) \\ &= 1 - (1 - F_{k_v}(x))^V && (8) \end{aligned}$$

where $F_{k_v}(x) = \mathbb{P}(k_v \leq x)$.

Since M_V is a non-negative random variable, we have

$$\mathbb{E}(M_V) = \sum_{x=0}^{\infty} (1 - F_{M_V}(x)) = \sum_{x=0}^{\infty} (1 - F_{k_v}(x))^V. \tag{9}$$

[*]UiT Machine Learning group (machine-learning.uit.no) and Visual Intelligence Centre (visual-intelligence.no).
[†]Norwegian Computing Center (nr.no).
[‡]Department of Computer Science, University of Copenhagen.
[§]Pioneer Centre for AI (aicentre.dk).

Model	Pub.	Enc.	SV-SSL	MV-SSL	Fusion	CM
DCCAE [17]	ICML'15	MLP	Reconstruction	CCA	1st view	SC
DMSC [1]	J. STSP'18	CNN	Reconstruction	–	Affinity fusion	SR, SC
DMVSSC [13]	ICNCC'18	CNN	Reconstruction	–	–	Sparse SR, SC
MvSN [4]	T. CSS'19	MLP	Sp. Emb.	–	Weighted sum	k-means
MvSCN [5]	IJCAI'19	MLP	Sp. Emb.	MSE Al.	Concat.	k-means
MvDSCN [26]	arXiv'19	CNN	–	Reconstruction	Shared network	SR, SC
DAMC [9]	IJCAI'19	MLP	–	Reconstruction	Average	DEC
S2DMVSC [12]	ACML'19	MLP	Reconstruction	–	MLP	SR, SC
DCMR [24]	PAKDD'20	MLP	Variational Reconstruction	Variational Reconstruction	MLP	k-means
DMMC [23]	ICME'20	MLP	Reconstruction	–	MLP	Fusion output
DCUMC [27]	ICIKM'20	MLP	Reconstruction	Commonness uniqueness	MLP	k-means
SGLR-MVC [22]	AAAI'20	MLP	–	Variational Reconstruction	Weighted sum	GMM
EAMC [25]	CVPR'20	MLP	–	Distribution Al., Kernel Al.	Attention	DDC
MVC-MAE [2]	DSE'21	MLP	Reconstruction, Ngh. preserv.	Contrastive Al.	–	DEC
SDC-MVC [19]	IJCNN'21	MLP	–	CCA	Concat.	DEC
DEMVC [20]	Inf. Sci.'21	CNN	Reconstruction	–	–	DEC
SiMVC [14]	CVPR'21	MLP/ CNN	–	–	Weighted sum	DDC
CoMVC [14]	CVPR'21	MLP/ CNN	–	Contrastive Al.	Weighted sum	DDC
Multi-VAE [21]	ICCV'21	CNN	–	Variational Reconstruction	Concat.	Gumbel, k-means
DMIM [10]	IJCAI'21	MLP	Min. superflous information	Max. shared information	?	Encoder output
AMvC [15]	TNNLS'22	MLP	–	Reconstruction	Weighted sum	DEC
SIB-MSC [16]	arXiv'22	CNN	–	Reconstruction, Inf. Bottleneck	Affinity fusion	SR, SC

Abbreviations: "–" = Not included, "?" = Not specified, Al. = Alignment, Concat. = Concatenate, CCA = Canonical correlation analysis, DDC = Deep divergence-based clustering, DEC = Deep embedded clustering, Inf. Bottleneck = Information bottleneck, Ngh. preserv. = Neighborhood preservation, SC = Spectral clustering, Sp. Emb. = Spectral Embedding, SR = Self-representation, Sparse SR = Sparse self-representation,

Table 1. Full overview of methods from previous work and their DeepMVC components.

Hence

$$\mathbb{E}(M_V) - \mathbb{E}(M_{V+1})$$

$$= \sum_{x=0}^{\infty} (1 - F_{k_v}(x))^V - \sum_{x=0}^{\infty} (1 - F_{k_v}(x))^{V+1} \quad (10)$$

$$= \sum_{x=0}^{\infty} (1 - F_{k_v}(x))^V (1 - (1 - F_{k_v}(x))) \quad (11)$$

$$= \sum_{x=0}^{\infty} \underbrace{(1 - F_{k_v}(x))^V}_{\geq 0} \underbrace{F_{k_v}(x)}_{\geq 0} \geq 0 \quad (12)$$

which is a sum of non-negative terms, since $F_{k_v}(x) \in [0, 1]$ is a probability. This gives

$$\mathbb{E}(M_{V+1}) \leq \mathbb{E}(M_V) \quad (13)$$

$\square$

4. New instances of DeepMVC

In this section we provide additional details on loss functions, particularly the weighted sum fusion, and the DDC [7] clustering module. The loss functions used to train the new instances are on the form

$$\mathcal{L}^{\text{Total}} = w^{\text{SV}} \mathcal{L}^{\text{SV}} + w^{\text{MV}} \mathcal{L}^{\text{MV}} + w^{\text{CM}} \mathcal{L}^{\text{CM}} \quad (14)$$

where $\mathcal{L}^{\text{SV}}$, $\mathcal{L}^{\text{MV}}$, and $\mathcal{L}^{\text{CM}}$ denote the losses from the SV-SSL, MV-SSL, and CM components, respectively. Note that the losses $\mathcal{L}^{\text{SV}}$ and $\mathcal{L}^{\text{MV}}$ correspond to the losses in Section 5 of the main paper. $(w^{\text{SV}}, w^{\text{MV}}, w^{\text{CM}})$ are optional weights for the respective losses, which are all set to 1 unless specified otherwise.

Connection between InfoDDC and contrastive self-supervised learning For two views $u \neq v \in 1, \dots, V$, contrastive SSL can be regarded as variational maximization of the mutual information

$$I(\boldsymbol{z}^{(v)}, \boldsymbol{z}^{(u)}) \tag{15}$$

where $\boldsymbol{z}^{(v)}$ and $\boldsymbol{z}^{(u)}$ have *multi-variate, continuous* distributions in $\mathbb{R}^d$.

In InfoDDC, we instead maximize mutual information between pairs of *uni-variate, discrete* random variables

$$I(c^{(v)}, c^{(u)}) \tag{16}$$

where we assume that the distributions of $c^{(v)}$ and $c^{(u)}$ are given by the view-specific representations

$$\mathbb{P}(c^{(w)} = i) = z^{(w)}_{[i]}, \quad i = 1, \dots, d, \quad w \in \{u, v\} \tag{17}$$

where $z^{(w)}_{[i]}$ denotes component i of the view-specific representation $\boldsymbol{z}^{(w)} = f^{(w)}(\boldsymbol{x}^{(w)})$. Hence, although InfoDDC might appear similar to CA-based methods, the maximization of mutual information is done for different pairs of random variables.

Weighted sum fusion. As [14], we implement the weighted sum fusion as

$$\boldsymbol{z}_i = \sum_{v=1}^{V} w^{(v)} \boldsymbol{z}_i^{(v)}, \tag{18}$$

where the weights $w^{(1)}, \dots, w^{(V)}$ are non-negative and sum to 1. These constraints are implemented by keeping a vector of trainable, un-normalized weights, from which $w^{(1)}, \dots, w^{(V)}$ can be computed by applying the softmax function.

DDC clustering module. The DDC [7] clustering module consists of two fully-connected layers. The first layer calculates the hidden representation $\boldsymbol{h}_i \in \mathbb{R}^{D_{DDC}}$ from the fused representation $\boldsymbol{z}_i$. The dimensionality of the hidden representation, D_{DDC} is a hyperparameter set to 100 for all models. The second layer computes the cluster membership vector $\boldsymbol{\alpha}_i \in \mathbb{R}^k$ from the hidden representation.

DDC's loss function consists of three terms

$$\mathcal{L}^{\text{CM}}_{\text{DDC}} = \mathcal{L}_{\text{DDC, 1}} + \mathcal{L}_{\text{DDC, 2}} + \mathcal{L}_{\text{DDC, 3}}. \tag{19}$$

The three terms encourage (i) separable and compact clusters in the hidden space; (ii) orthogonal cluster membership vectors; and (iii) cluster membership vectors close to simplex corners, respectively.

The first term maximizes the pairwise Cauchy-Schwarz divergence [6] between clusters (represented as probability densities) in the space of hidden representations

$$\mathcal{L}_{\text{DDC, 1}} = \tag{20}$$

$$\binom{k}{2}^{-1} \sum_{a=1}^{k-1} \sum_{b=a}^{k} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_{ia} \kappa_{ij} \alpha_{jb}}{\sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_{ia} \kappa_{ij} \alpha_{ja} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_{ib} \kappa_{ij} \alpha_{jb}}} \tag{21}$$

where $\kappa_{ij} = \exp\left(-\frac{||\boldsymbol{h}_i - \boldsymbol{h}_j||^2}{2\sigma^2}\right)$ and σ is a hyperparameter. Following [7], we set σ to 15% of the median pairwise difference between the hidden representations.

The second term minimizes the pairwise inner product between cluster membership vectors

$$\mathcal{L}_{\text{DDC, 2}} = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j^\top. \tag{22}$$

The third term encourages cluster membership vectors to be close to the corners of the probability simplex in $\mathbb{R}^k$

$$\mathcal{L}_{\text{DDC, 3}} = \tag{23}$$

$$\binom{k}{2}^{-1} \sum_{a=1}^{k-1} \sum_{b=a}^{k} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} m_{ia} \kappa_{ij} m_{jb}}{\sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} m_{ia} \kappa_{ij} m_{ja} \sum_{i=1}^{n} \sum_{j=1}^{n} m_{ib} \kappa_{ij} m_{jb}}} \tag{24}$$

where $m_{ia} = \exp(-||\boldsymbol{\alpha}_i - \boldsymbol{e}_a||^2)$, and $\boldsymbol{e}_a$ is the a-th simplex corner.

5. Experiments

5.1. Datasets

Dataset details are listed in Table 2. The code repository includes pre-processed Caltech7 and Caltech20 datasets. The other datasets can be generated by following the instructions in `README.md` (these could not be included in the archive due to limitations on space).

Caltech details. We use the same features and subsets of the Caltech101 [3] dataset as [5].

- **Features:** Gabor, Wavelet Moments, CENsus TRansform hISTogram (CENTRIST), Histogram of Oriented Gradients (HOG), GIST, and Local Binary Patterns (LBP).
- **Caltech7 classes:** Face, Motorbikes, Dolla-Bill, Garfield, Snoopy, Stop-Sign, Windsor-Chair.

Dataset	n	v	k	n_{small}	n_{big}	Dim.	Licence
NoisyMNIST [8]	70000	2	10	6313	7877	$(28 \times 28)^2$	CC BY-SA 3.0
NoisyFashion [18]	70000	2	10	7000	7000	$(28 \times 28)^2$	MIT
EdgeMNIST [8]	70000	2	10	6313	7877	$(28 \times 28)^2$	CC BY-SA 3.0
EdgeFashion [18]	70000	2	10	7000	7000	$(28 \times 28)^2$	MIT
COIL-20 [11]	480	3	20	24	24	$(64 \times 64)^3$	None
Caltech7 [3]	1474	6	7	34	798	$48, 40, 254, 1984, 512, 928$	CC BY 4.0
Caltech20 [3]	2386	6	20	33	798	$48, 40, 254, 1984, 512, 928$	CC BY 4.0
PatchedMNIST [8]	21770	12	3	6903	7877	$(28 \times 28)^{12}$	CC BY-SA 3.0

Table 2. Dataset details. n = number of instances, v = number of views, k = number of classes/clusters, n_{small} = number of instances in smallest class, n_{big} = number of instances in largest class, Dim. = view dimensions.

- **Caltech20 classes:** Face, Leopards, Motorbikes, Binocular, Brain, Camera,Car-Side, Dolla-Bill, Ferry, Garfield, Hedgehog, Pagoda, Rhino, Snoopy, Stapler, Stop-Sign, Water-Lilly, WindsorChair, Wrench, Yin-yang.

5.2. Hyperparameters

Network architectures. The encoder and decoder architectures are listed in Table 3. MLP encoders/decoders are used for Caltech7 and Caltech20 as these contain vector data. The other datasets contain images, so CNN encoders and decoders are used for them.
Other hyperparameters. Table 4 lists other hyperparameters used for the baselines and new instances.

5.3. Computational resources

We run our experiments on a Kubernetes cluster, where jobs are allocated to nodes with Intel(R) Xeon(R) E5-2623 v4 or Intel(R) Xeon(R) Silver 4210 CPUs (2 cores allocated per job); and Nvidia GeForce GTX 1080 Ti or Nvidia GeForce RTX 2080 Ti GPUs. Each job has 16 GB RAM available.

With this setup, 5 training runs on NoisyMNIST, NoisyFashion, EdgeMNIST, and EdgeFashion take approximately 24 hours. Training times for the other datasets are approximately between 1 and 3 hours.

The Dockerfile used to build our docker image can be found in the code repository.

5.4. Evaluation protocol

Metrics. We measure performance using the accuracy

$$\text{ACC} = \max_{m \in \mathcal{M}} \frac{\sum_{i=1}^{n} \delta(m(\hat{y}_i) - y_i)}{n} \tag{25}$$

where $\delta(\cdot)$ is the Kronecker-delta, $\hat{y}_i$ is the predicted cluster of instance i, and y_i is the ground truth label of instance i. The maximum runs over $\mathcal{M}$, which is the set of all bijective mappings from $\{1, \dots, k\}$ to itself.

We also compute the normalized mutual information

$$\text{NMI} = \frac{MI(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\frac{1}{2}(H(\hat{\boldsymbol{y}}) + H(\boldsymbol{y}))} \tag{26}$$

where $\hat{\boldsymbol{y}} = [\hat{y}_1, \dots, \hat{y}_n]$, $\boldsymbol{y} = [y_1, \dots, y_n]$, $MI(\cdot, \cdot)$ and $H(\cdot)$ denotes the mutual information and entropy, respectively.
Uncertainty estimation. The uncertainty of our performance statistic can be estimated using bootstrapping. Suppose the R training runs result in the R tuples

$$(L_1, M_1), \dots, (L_R, M_R) \tag{27}$$

where L_i is the final loss of run i, and M_i is resulting performance metric for run i. We then sample B bootstrap samples uniformly from the original results

$$(L_j^b, M_j^b) \sim \text{Uniform}\{(L_1, M_1), \dots, (L_R, M_R)\}, \quad j = 1, \dots, R, \quad b = 1, \dots B. \tag{28}$$

The performance statistic for bootstrap sample b is then given by

$$M_\star^b = M_{j_\star^b}^b, \quad j_\star^b = \arg \min_{j=1,\dots,R} \{L_j^b\}. \tag{29}$$

We then estimate the uncertainty of the performance statistic by computing the standard deviation of the bootstrap statistics $M_\star^1, \dots M_\star^B$

$$\hat{\sigma}_{M_\star} = \sqrt{\frac{\sum_{b=1}^{B} (M_\star^b - \bar{M}_\star)^2}{B - 1}}, \quad \text{where} \quad \bar{M}_\star = \frac{\sum_{b=1}^{B} M_b^\star}{B}. \tag{30}$$

5.5. Results

Evaluation results. The complete evaluation results are given in Table 5.
Ablation study – Fusion and Clustering module. Table 6 shows the results of ablation studies with the fusion and clustering module (CM) components. Since these components can not be completely removed, we instead replace more

CNN encoder	CNN decoder	MLP encoder	MLP decoder
`Conv`$(64 \times 3 \times 3)$	`UpSample`(2×2)	`Dense`(1024)	`Dense`(256)
`ReLU`	`TransposeConv`$(64 \times 3 \times 3)$	`BatchNorm`	`BatchNorm`
`Conv`$(64 \times 3 \times 3)$	`ReLU`	`ReLU`	`ReLU`
`BatchNorm`	`TransposeConv`$(64 \times 3 \times 3)$	`Dense`(1024)	`Dense`(1024)
`ReLU`	`BatchNorm`	`BatchNorm`	`BatchNorm`
`MaxPool`(2×2)	`ReLU`	`ReLU`	`ReLU`
`Conv`$(64 \times 3 \times 3)$	`UpSample`(2×2)	`Dense`(1024)	`Dense`(1024)
`ReLU`	`TransposeConv`$(64 \times 3 \times 3)$	`BatchNorm`	`BatchNorm`
`Conv`$(64 \times 3 \times 3)$	`ReLU`	`ReLU`	`ReLU`
`BatchNorm`	`TransposeConv`$(1 \times 3 \times 3)$	`Dense`(1024)	`Dense`(1024)
`ReLU`	`Sigmoid`	`BatchNorm`	`BatchNorm`
`MaxPool`(2×2)		`ReLU`	`ReLU`
		`Dense`(256)	`Dense(input dim)`
			`Sigmoid`

Table 3. Network architectures.

Model	Batch size	Learning rate	w^{SV}	w^{MV}	w^{CM}	Pre-train	Gradient clip
DMSC	100	10^{-3}	1.0	–	–	✓	10
MvSCN	512	10^{-4}	0.999	0.001	–	✘	10
EAMC	100	†	–	1.0	1.0	✘	10
SiMVC	100	10^{-3}	–	–	1.0	✘	10
CoMVC	100	10^{-3}	–	0.1	1.0	✘	10
Multi-VAE	64	$5 \cdot 10^{-4}$	–	1.0	–	✓	10
AE–DDC	100	10^{-3}	1.0	–	1.0	✓	10
AECoDDC	100	10^{-3}	1.0	0.1	1.0	✓	10
AE–KM	100	10^{-3}	1.0	–	–	✘	10
AECoKM	100	10^{-3}	1.0	0.1	–	✘	10
InfoDDC	256	10^{-3}	–	0.1	1.0	✘	10
MV-IIC	256	10^{-3}	–	0.01	1.0	✘	10

Table 4. Hyperparameters used to train the models. † = EAMC [25] has different learning rates for the different components, namely 10^{-5} for the encoders and clustering module, and 10^{-4} for the attention module and discriminator.

complicated components, with the simplest possible component. Thus, we replace weighted sum with concatenate for the fusion component, and DDC with k-means for the CM component.

For the fusion component, we see that the weighted sum tends to improve over the concatenation. For the CM, we observe that the performance is better with DDC than with k-means on NoisyMNIST, but the improvement more varied on Caltech7. This is consistent with what we observed in the evaluation results in the main paper.

Reproducibility of original results. Table 7 compares the results of our re-implementation of the baselines, to the results reported by the original authors. The comparison shows large differences in performance for several methods, and the differences are particularly large for MvSCN and Multi-VAE. For MvSCN, we do not use the same autoencoder preprocessing of the data. We also had difficulties getting the Cholesky decomposition to converge during training. For MultiVAE, we note that NoisyMNIST and NoisyFashion are generated without noise in the original paper, possibly resulting in datasets that are simpler to cluster. We were however not able to determine the reason for the difference in performance on COIL-20.

Additionally, all methods use different network architectures and evaluation protocols in the original publications, making it difficult to accurately compare performance between methods and their implementations. This illustrates the difficulty of reproducing and comparing results in deep MVC, highlighting the need for a unified framework with a consistent evaluation protocol and an open-source implementation.

Sensitivity to hyperparameters Table 8 shows the results of hyperparameter sweeps for the following hyperparameters:

- Weight of reconstruction loss (w^{SV}).
- Weight of contrastive loss (w^{MV}).

	NoisyMNIST		NoisyFashion		EdgeMNIST		EdgeFashion	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
DMSC	0.66 (0.02)	0.67 (0.01)	0.49 (0.05)	0.48 (0.03)	0.51 (0.02)	0.47 (0.02)	0.52 (0.01)	0.47 (0.00)
MvSCN	0.15 (0.00)	0.02 (0.00)	0.14 (0.00)	0.01 (0.00)	0.14 (0.00)	0.01 (0.01)	0.12 (0.00)	0.03 (0.00)
EAMC	0.83 (0.04)	0.90 (0.02)	0.61 (0.02)	0.71 (0.02)	0.76 (0.05)	0.79 (0.03)	0.51 (0.03)	0.47 (0.01)
SiMVC	**1.00** (0.02)	**1.00** (0.02)	0.52 (0.02)	0.51 (0.02)	0.89 (0.06)	0.90 (0.04)	0.61 (0.01)	0.56 (0.02)
CoMVC	**1.00** (0.00)	**1.00** (0.00)	0.67 (0.03)	0.68 (0.03)	**0.97** (0.08)	**0.94** (0.07)	0.56 (0.03)	0.52 (0.01)
Multi-VAE	0.98 (0.05)	0.96 (0.02)	0.62 (0.02)	0.60 (0.01)	0.85 (0.01)	0.76 (0.01)	0.58 (0.01)	**0.64** (0.00)
AE–KM	0.74 (0.03)	0.71 (0.00)	0.58 (0.02)	0.59 (0.01)	0.60 (0.00)	0.57 (0.00)	0.54 (0.00)	0.58 (0.00)
AE–DDC	**1.00** (0.04)	**1.00** (0.03)	0.69 (0.06)	0.65 (0.05)	0.88 (0.11)	0.88 (0.09)	0.60 (0.01)	0.58 (0.01)
AECoKM	**1.00** (0.00)	0.99 (0.00)	0.63 (0.07)	0.73 (0.03)	0.38 (0.03)	0.31 (0.02)	0.39 (0.04)	0.34 (0.02)
AECoDDC	**1.00** (0.00)	0.99 (0.00)	**0.80** (0.02)	**0.77** (0.01)	0.89 (0.10)	0.90 (0.09)	**0.67** (0.09)	0.62 (0.06)
InfoDDC	0.90 (0.05)	0.92 (0.04)	0.54 (0.03)	0.52 (0.04)	0.62 (0.04)	0.52 (0.06)	0.43 (0.01)	0.43 (0.03)
MV-IIC	0.52 (0.04)	0.79 (0.02)	0.52 (0.07)	0.74 (0.02)	0.31 (0.04)	0.21 (0.05)	0.52 (0.04)	0.59 (0.04)

	COIL-20		Caltech7		Caltech20		PatchedMNIST	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
DMSC	$-^\dagger$ (–)	$-^\dagger$ (–)	0.50 (0.03)	0.50 (0.02)	0.35 (0.01)	0.55 (0.00)	$-^\dagger$ (–)	$-^\dagger$ (–)
MvSCN	0.21 (0.00)	0.23 (0.01)	0.29 (0.02)	0.02 (0.00)	0.13 (0.01)	0.09 (0.01)	$-^\dagger$ (–)	$-^\dagger$ (–)
EAMC	0.39 (0.15)	0.52 (0.22)	0.44 (0.02)	0.23 (0.03)	0.22 (0.04)	0.23 (0.02)	$-^\ddagger$ (–)	$-^\ddagger$ (–)
SiMVC	**0.90** (0.04)	**0.96** (0.02)	0.41 (0.02)	0.51 (0.09)	0.34 (0.02)	0.52 (0.01)	0.84 (0.04)	0.64 (0.11)
CoMVC	0.87 (0.03)	**0.96** (0.02)	0.38 (0.01)	0.55 (0.02)	0.34 (0.01)	0.59 (0.02)	0.73 (0.12)	0.57 (0.19)
Multi-VAE	0.74 (0.02)	0.84 (0.01)	0.47 (0.02)	0.47 (0.01)	0.40 (0.01)	0.57 (0.01)	0.94 (0.00)	0.77 (0.00)
AE–KM	0.88 (0.04)	0.92 (0.01)	0.44 (0.03)	0.52 (0.01)	0.45 (0.02)	0.57 (0.01)	0.87 (0.00)	0.68 (0.01)
AE–DDC	0.80 (0.04)	0.93 (0.02)	0.40 (0.01)	0.54 (0.07)	0.34 (0.01)	0.44 (0.03)	0.77 (0.10)	0.59 (0.17)
AECoKM	0.84 (0.04)	0.94 (0.02)	0.20 (0.01)	0.05 (0.00)	0.22 (0.02)	0.27 (0.02)	0.96 (0.00)	0.85 (0.00)
AECoDDC	0.87 (0.01)	**0.96** (0.00)	0.36 (0.01)	0.43 (0.03)	0.31 (0.02)	0.51 (0.02)	**0.99** (0.00)	**0.97** (0.00)
InfoDDC	0.25 (0.04)	0.54 (0.03)	0.51 (0.01)	0.60 (0.04)	**0.58** (0.07)	**0.63** (0.03)	**0.99** (0.00)	0.96 (0.00)
MV-IIC	0.83 (0.05)	0.94 (0.02)	**0.53** (0.00)	**0.63** (0.04)	0.49 (0.01)	0.61 (0.01)	0.97 (0.00)	0.90 (0.01)

Table 5. Clustering results. Standard deviations (obtained by bootstrapping) are shown in parentheses. † = training ran out of memory, ‡ = training resulted in NaN loss.

- Temperature in contrastive loss (τ).

- Weight of entropy regularization (λ).

We emphasize that these results were *not* used to tune hyperparameters for the new instances. Rather, they are included to investigate how robust these methods are towards changes in the hyperparameter configuration. The results show that the new instances are mostly insensitive to changes in their hyperparameters. We however observe two cases where the hyperparameter configurations can have significant impact on the model performance. First, AECoDDC shows a drop in performance when the weight of the contrastive loss is set to high on Caltech7 (Table 8b). This is consistent with our observations regarding contrastive alignment on datasets with many views. Second, InfoDDC and MV-IIC performs worse when the entropy regularization weight is set too low, indicating that sufficient regularization is required for these models to perform well.

6. Potential negative societal impacts

As is the case with most methodological research, our work can be applied to downstream applications with negative societal impact – for instance by reflecting biases in the dataset the model was trained on. We note that in unsupervised learning, it is particularly important to check what a model has learned, due to the lack of label supervision. This is crucial if the models are used to make high-stakes decisions.

References

[1] Mahdi Abavisani and Vishal M. Patel. Deep Multimodal Subspace Clustering Networks. *IEEE Journal of Selected Topics in Signal Processing*, 12(6):1601–1614, 2018. 2

[2] Guowang Du, Lihua Zhou, Yudi Yang, Kevin Lü, and Lizhen Wang. Deep Multiple Auto-Encoder-Based Multi-view Clustering. *Data Science and Engineering*, 6(3):323–338, 2021. 2

[3] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories.

(a) Fusion

Model	NoisyMNIST		Caltech7	
	Concat.	Weighted	Concat.	Weighted
SiMVC	1.00	1.00 (0.00)	0.36	0.41 (+0.04)
CoMVC	1.00	1.00 (0.00)	0.42	0.38 (−0.04)
AE–DDC	1.00	1.00 (0.00)	0.36	0.40 (+0.04)
AECoDDC	1.00	1.00 (0.00)	0.39	0.36 (−0.03)
InfoDDC	0.93	0.90 (−0.03)	0.36	0.51 (+0.15)

(b) CM

Model	NoisyMNIST		Caltech7	
	k-means	DDC	k-means	DDC
SiMVC	0.67	1.00 (+0.33)	0.39	0.41 (+0.01)
CoMVC	0.56	1.00 (+0.44)	0.22	0.38 (+0.16)
AE–DDC	0.74	1.00 (+0.26)	0.44	0.40 (−0.04)
AECoDDC	1.00	1.00 (0.00)	0.20	0.36 (+0.16)
InfoDDC	0.14	0.90 (+0.76)	0.59	0.51 (−0.08)

Table 6. Accuracies from ablation studies with the Fusion and CM components.

Model	Dataset	Orig.	Ours
MvSCN	N-MNIST	0.99	0.15 (−0.84)
	Caltech20	0.59	0.13 (−0.46)
EAMC	E-MNIST	0.67	0.76 (+0.09)
SiMVC	E-MNIST	0.86	0.89 (+0.03)
	E-Fashion	0.57	0.61 (+0.04)
	COIL-20	0.78	0.90 (+0.12)
CoMVC	E-MNIST	0.96	0.97 (+0.01)
	E-Fashion	0.60	0.56 (−0.04)
	COIL-20	0.89	0.87 (−0.02)
Multi-VAE	N-MNIST†	1.00	0.98 (−0.02)
	N-Fashion†	0.91	0.62 (−0.29)
	COIL-20	0.98	0.74 (−0.24)

Table 7. Accuracies from our experiment vs. accuracies reported by the original authors. † = method is originally evaluated on a slightly different dataset.

Computer Vision and Image Understanding, 106(1):59–70, 2007. 3, 4

[4] Shuning Huang, Kaoru Ota, Mianxiong Dong, and Fanzhang Li. MultiSpectralNet: Spectral Clustering Using Deep Neural Network for Multi-View Data. *IEEE Transactions on Computational Social Systems*, 6(4):749–760, 2019. 2

[5] Zhenyu Huang, Joey Tianyi Zhou, Xi Peng, Changqing Zhang, Hongyuan Zhu, and Jiancheng Lv. Multi-view Spectral Clustering Network. In *IJCAI*, 2019. 2, 3

[6] Robert Jenssen, Jose C. Principe, Deniz Erdogmus, and Torbjørn Eltoft. The Cauchy–Schwarz divergence and Parzen windowing: Connections to graph theory and Mercer kernels. *Journal of the Franklin Institute*, 343(6):614–629, 2006. 3

[7] Michael Kampffmeyer, Sigurd Løkse, Filippo M. Bianchi, Lorenzo Livi, Arnt-Børre Salberg, and Robert Jenssen. Deep Divergence-based Approach to Clustering. *Neural Networks*, 113:91–101, 2019. 2, 3

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4

[9] Z. Li, Q. Wang, Z. Tao, Q. Gao, and Z. Yang. Deep Adversarial Multi-view Clustering Network. In *IJCAI*, 2019. 2

[10] Yiqiao Mao, Xiaoqiang Yan, Qiang Guo, and Yangdong Ye. Deep Mutual Information Maximin for Cross-Modal Clustering. In *IJCAI*, 2021. 2

[11] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-20). Technical report, 1996. 4

[12] Xiukun Sun, Miaomiao Cheng, Chen Min, and Liping Jing. Self-Supervised Deep Multi-View Subspace Clustering. In *ACML*, 2019. 2

[13] Xiaoliang Tang, Xuan Tang, Wanli Wang, Li Fang, and Xian Wei. Deep Multi-view Sparse Subspace Clustering. In *ICNCC*, 2018. 2

[14] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. Reconsidering Representation Alignment for Multi-view Clustering. In *CVPR*, 2021. 2, 3

[15] Qianqian Wang, Zhiqiang Tao, Wei Xia, Quanxue Gao, Xiaochun Cao, and Licheng Jiao. Adversarial Multiview Clustering Networks With Adaptive Fusion. *IEEE transactions on neural networks and learning systems*, PP, 2022. 2

[16] Shiye Wang, Changsheng Li, Yanming Li, Ye Yuan, and Guoren Wang. Self-Supervised Information Bottleneck for Deep Multi-View Subspace Clustering. *arXiv:2204.12496 [cs]*, 2022. 2

[17] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On Deep Multi-View Representation Learning. In *ICML*, 2015. 2

[18] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747 [cs, stat]*, 2017. 4

[19] Bowen Xin, Shan Zeng, and Xiuying Wang. Self-Supervised Deep Correlational Multi-View Clustering. In *IJCNN*, 2021. 2

[20] Jie Xu, Yazhou Ren, Guofeng Li, Lili Pan, Ce Zhu, and Zenglin Xu. Deep embedded multi-view clustering with collaborative training. *Information Sciences*, 573:279–290, 2021. 2

[21] Jie Xu, Yazhou Ren, Huayi Tang, Xiaorong Pu, Xiaofeng Zhu, Ming Zeng, and Lifang He. Multi-VAE: Learning Disentangled View-Common and View-Peculiar Visual Representations for Multi-View Clustering. In *ICCV*, 2021. 2

[22] Ming Yin, Weitian Huang, and Junbin Gao. Shared Generative Latent Representation Learning for Multi-View Clustering. In *AAAI*, 2020. 2

[23] Xianchao Zhang, Jie Mu, Linlin Zong, and Xiaochun Yang. End-To-End Deep Multimodal Clustering. In *ICME*, 2020. 2

[24] Xianchao Zhang, Xiaorui Tang, Linlin Zong, Xinyue Liu, and Jie Mu. Deep Multimodal Clustering with Cross Reconstruction. In *PAKDD*, 2020. 2

[25] Runwu Zhou and Yi-Dong Shen. End-to-End Adversarial-Attention Network for Multi-Modal Clustering. In *CVPR*, 2020. 2, 5

[26] Pengfei Zhu, Binyuan Hui, Changqing Zhang, Dawei Du, Longyin Wen, and Qinghua Hu. Multi-view Deep Subspace Clustering Networks. *arXiv:1908.01978 [cs]*, 2019. 2

(a) Weight of reconstruction loss (w^{SV}).

	NoisyMNIST				Caltech7			
Rec. weight	0.01	0.1	1.0	10.0	0.01	0.1	1.0	10.0
AE–DDC	1.00 (0.03)	0.94 (0.03)	1.00 (0.03)	0.94 (0.01)	0.41 (0.01)	0.41 (0.03)	0.44 (0.02)	0.45 (0.02)
AECoDDC	0.99 (0.00)	0.99 (0.00)	0.99 (0.00)	0.99 (0.00)	0.40 (0.05)	0.33 (0.02)	0.34 (0.03)	0.49 (0.04)
AECoKM	0.74 (0.02)	0.70 (0.04)	0.74 (0.03)	0.93 (0.02)	0.07 (0.01)	0.05 (0.00)	0.04 (0.01)	0.04 (0.02)

(b) Weight of contrastive loss (w^{MV}).

	NoisyMNIST				Caltech7			
Con. weight	0.01	0.1	1.0	10.0	0.01	0.1	1.0	10.0
AECoDDC	1.00 (0.00)	0.99 (0.00)	0.99 (0.00)	0.99 (0.00)	0.46 (0.02)	0.40 (0.05)	0.19 (0.03)	0.09 (0.01)
AECoKM	0.89 (0.01)	0.73 (0.03)	0.77 (0.01)	0.67 (0.02)	0.04 (0.02)	0.04 (0.01)	0.06 (0.01)	0.05 (0.00)

(c) Temperature in the contrastive loss (τ).

	NoisyMNIST				Caltech7			
τ	0.01	0.07	0.1	1.0	0.01	0.07	0.1	1.0
AECoDDC	0.99 (0.00)	1.00 (0.00)	0.99 (0.00)	1.00 (0.00)	0.31 (0.03)	0.39 (0.01)	0.35 (0.02)	0.48 (0.01)
AECoKM	0.99 (0.00)	0.91 (0.02)	0.74 (0.02)	0.78 (0.05)	0.34 (0.01)	0.05 (0.01)	0.06 (0.01)	0.46 (0.01)

(d) Weight of the entropy regularization (λ).

	NoisyMNIST				Caltech7			
λ	0.5	1.5	5.0	10.0	0.5	1.5	5.0	10.0
MV-IIC	0.03 (0.01)	0.81 (0.01)	0.82 (0.00)	0.82 (0.00)	0.04 (0.01)	0.64 (0.04)	0.60 (0.01)	0.52 (0.01)
InfoDDC	0.21 (0.02)	0.37 (0.02)	0.84 (0.04)	0.94 (0.07)	0.60 (0.06)	0.60 (0.02)	0.57 (0.01)	0.51 (0.01)

Table 8. Results (NMI) of hyperparameter sweeps for the new instances.

[27] Linlin Zong, Faqiang Miao, Xianchao Zhang, and Bo Xu. Multimodal Clustering via Deep Commonness and Uniqueness Mining. In *ICIKM*, 2020. 2

Paper IV

Hubs and Hyperspheres: Reducing Hubness and Improving Transductive Few-shot Learning with Hyperspherical Embeddings

Daniel J. Trosten[*,†], Rwiddhi Chakraborty[*,†], Sigurd Løkse[†], Kristoffer Knutsen Wickstrøm[†], Robert Jenssen[†‡§¶], Michael C. Kampffmeyer[†‡]
Department of Physics and Technology, UiT The Arctic University of Norway
`firstname[.middle initial].lastname@uit.no`

Abstract

Distance-based classification is frequently used in transductive few-shot learning (FSL). However, due to the high-dimensionality of image representations, FSL classifiers are prone to suffer from the hubness problem, where a few points (hubs) occur frequently in multiple nearest neighbour lists of other points. Hubness negatively impacts distance-based classification when hubs from one class appear often among the nearest neighbors of points from another class, degrading the classifier's performance. To address the hubness problem in FSL, we first prove that hubness can be eliminated by distributing representations uniformly on the hypersphere. We then propose two new approaches to embed representations on the hypersphere, which we prove optimize a tradeoff between uniformity and local similarity preservation – reducing hubness while retaining class structure. Our experiments show that the proposed methods reduce hubness, and significantly improves transductive FSL accuracy for a wide range of classifiers[1].

1. Introduction

While supervised deep learning has made a significant impact in areas where large amounts of labeled data are available [6, 11], few-shot learning (FSL) has emerged as a promising alternative when labeled data is limited [3, 12, 14, 16, 21, 26, 28, 31, 33, 39, 40]. FSL aims to design classifiers that can discriminate between novel classes based on a few labeled instances, significantly reducing the cost of the labeling procedure.

In transductive FSL, one assumes access to the entire query set during evaluation. This allows transductive FSL classifiers to learn representations from a larger number of samples, resulting in better performing classifiers. However, many of these methods base their predictions on distances to prototypes for the novel classes [3, 16, 21, 28, 39, 40]. This makes these methods susceptible to the hubness problem [10, 22, 24, 25], where certain exemplar points (hubs) appear among the nearest neighbours of many other points. If a support sample is a hub, many query samples will be assigned to it regardless of their true label, resulting in low accuracy. If more training data is available, this effect can be reduced by increasing the number of labeled samples in the classification rule – but this is impossible in FSL.

Several approaches have recently been proposed to embed samples in a space where the FSL classifier's performance is improved [4, 5, 7, 17, 33, 35, 39]. However, only one of these directly addresses the hubness problem. Fei *et al.* [7] show that embedding representations on a hypersphere with zero mean reduces hubness. They advocate the use of Z-score normalization (ZN) along the feature axis of each representation, and show empirically that ZN can reduce hubness in FSL. However, ZN does not guarantee a data mean of zero, meaning that hubness can still occur after ZN.

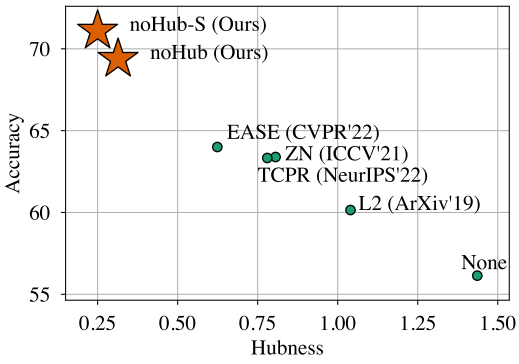


Figure 1. Few-shot accuracy increases when hubness decreases. The figure shows the 1-shot accuracy when classifying different embeddings with SimpleShot [33] on mini-ImageNet [29].

[*]Equal contributions.
[†]UiT Machine Learning group (`machine-learning.uit.no`) and Visual Intelligence Centre (`visual-intelligence.no`).
[‡]Norwegian Computing Center.
[§]Department of Computer Science, University of Copenhagen.
[¶]Pioneer Centre for AI (`aicentre.dk`).
[1]Code available at `https://github.com/uitml/noHub`.

In this paper we propose a principled approach to embed representations in FSL, which both reduces hubness and improves classification performance. First, we prove that hubness can be eliminated by embedding representations uniformly on the hypersphere. However, distributing representations uniformly on the hypersphere without any additional constraints will likely break the class structure which is present in the representation space – hurting the performance of the downstream classifier. Thus, in order to both reduce hubness and preserve the class structure in the representation space, we propose two new embedding methods for FSL. Our methods, Uniform Hyperspherical Structure-preserving Embeddings (noHub) and noHub with Support labels (noHub-S), leverage a decomposition of the Kullback-Leibler divergence between representation and embedding similarities, to optimize a tradeoff between Local Similarity Preservation (LSP) and uniformity on the hypersphere. The latter method, noHub-S, also leverages label information from the support samples to further increase the class separability in the embedding space.

Figure 1 illustrates the correspondence between hubness and accuracy in FSL. Our methods have both the *least hubness* and *highest accuracy* among several recent embedding techniques for FSL.

Our contributions are summarized as follows.

- We prove that the uniform distribution on the hypersphere has zero hubness and that embedding points uniformly on the hypersphere thus alleviates the hubness problem in distance-based classification for transductive FSL.
- We propose noHub and noHub-S to embed representations on the hypersphere, and prove that these methods optimize a tradeoff between LSP and uniformity. The resulting embeddings are therefore approximately uniform, while simultaneously preserving the class structure in the embedding space.
- Extensive experimental results demonstrate that noHub and noHub-S outperform current state-of-the-art embedding approaches, boosting the performance of a wide range of transductive FSL classifiers, for multiple datasets and feature extractors.

2. Related Work

The hubness problem. The hubness problem refers to the emergence of *hubs* in collections of points in high-dimensional vector spaces [22]. Hubs are points that appear among the nearest neighbors of many other points, and are therefore likely to have a significant influence on *e.g.* nearest neighbor-based classification. Radovanovic *et al.* [22] showed that points closer to the expected data mean are more likely be among the nearest neighbors of other points, indicating that these points are more likely to be hubs. Hubness can also be seen as a result of large density gradients [9], as points in high-density areas are more likely to be hubs. The hubness problem is thus an intrinsic property of data distributions in high-dimensional vector spaces, and not an artifact occurring in particular datasets. It is therefore important to take the hubness into account when designing classification systems in high-dimensional vector spaces.

Hubness in FSL. Many recent methods in FSL rely on distance-based classification in high-dimensional representation spaces [1, 3, 19, 33, 36, 38, 40], making them vulnerable to the hubness problem. Fei *et al.* [7] show that hyperspherical representations with zero mean reduce hubness. Motivated by this insight, they suggest that representations should have zero mean and unit standard deviation (ZN) *along the feature dimension*. This effectively projects samples onto the hyperplane orthogonal to the vector with all elements $= 1$, and pushes them to the hypersphere with radius $\sqrt{d}$, where d is the dimensionality of the representation space. Although ZN is empirically shown to reduce hubness, it does not guarantee that the data mean is zero. The normalized representations can therefore still suffer from hubness, potentially decreasing FSL performance.

Embeddings in FSL. FSL classifiers often operate on embeddings of representations instead of the representations themselves, to improve the classifier's ability to generalize to novel classes [5, 33, 35, 39]. Earlier works use the L2 normalization and Centered L2 normalization to embed representations on the hypersphere [33]. Among more recent embedding techniques, ReRep [5] performs a two-step fusing operation on both the support and query features with an attention mechanism. EASE [39] combines both support and query samples into a single sample set, and jointly learns a similarity and dissimilarity matrix, encouraging similar features to be embedded closer, and dissimilar features to be embedded far away. TCPR [35] computes the top-k neighbours of each test sample from the base data, computes the centroid, and removes the feature components in the direction of the centroid. Although these methods generally lead to a reduction in hubness and an increase in performance (see Figure 1), they are not explicitly designed to address the hubness problem resulting in suboptimal hubness reduction and performance. In contrast, our proposed noHub and noHub-S directly leverage our theoretic insights to target the root of the hubness problem.

Hyperspherical uniformity. Benefits of uniform hyperspherical representations have previously been studied for contrastive self-supervised learning (SSL) [32]. Our work differs from [32] on several key points. First, we study a non-parametric embedding of support and query samples for FSL, which is a fundamentally different task from contrastive SSL. Second, the contrastive loss studied in [32] is a combination of different cross-entropies, making it different from our KL-loss. Finally, we introduce a tradeoff-parameter between uniformity and LSP, and connect our theoretical results to hubness and Laplacian Eigenmaps.

3. Hyperspherical Uniform Eliminates Hubness

We will now show that hubness can be eliminated completely by embedding representations *uniformly* on the hypersphere[2].

Definition 1 (Uniform PDF on the hypersphere.)**.** *The uniform probability density function (PDF) on the unit hypersphere* $\mathbb{S}_d = \{\boldsymbol{x} \in \mathbb{R}^d \mid ||\boldsymbol{x}|| = 1\} \subset \mathbb{R}^d$ *is*

$$u_{\mathbb{S}_d}(\boldsymbol{x}) = A_d^{-1}\delta(||\boldsymbol{x}|| - 1) \tag{1}$$

where $A_d = \frac{2\pi^{d/2}}{\Gamma(d/2)}$ *is the surface area of* $\mathbb{S}_d$*, and* $\delta(\cdot)$ *is the Dirac delta distribution.*

We then have the following propositions[3] for random vectors with this PDF.

Proposition 1. *Suppose* $\boldsymbol{X}$ *has PDF* $u_{\mathbb{S}_d}(\boldsymbol{x})$*. Then*

$$\mathbb{E}(\boldsymbol{X}) = 0 \tag{2}$$

Proposition 2. *Let* $\Pi_{\boldsymbol{p}}$ *be the tangent plane of* $\mathbb{S}_d$ *at an arbitrary point* $\boldsymbol{p} \in \mathbb{S}_d$*. Then, for any direction* $\boldsymbol{\theta}^*$ *in* $\Pi_{\boldsymbol{p}}$ *the directional derivative of* $u_{\mathbb{S}_d}$ *along* $\boldsymbol{\theta}^*$ *is*

$$\nabla_{\boldsymbol{\theta}^*} u_{\mathbb{S}_d} = 0 \tag{3}$$

These two propositions show that the hyperspherical uniform has (i) zero mean; and (ii) zero density gradient along all directions tangent to the hypersphere's surface, at all points on the hypersphere. The hyperspherical uniform thus provably eliminates hubness, both in the sense of having a zero data mean, and having zero density gradient everywhere. We note that the latter property is un-attainable in Euclidean space, as it is impossible to define a uniform distribution over the whole space. It is therefore necessary to embed points on a non-Euclidean sub-manifold in order to eliminate hubness.

4. Method

In the preceding section, we proved that uniform embeddings on the hypersphere eliminate hubness. However, naïvely placing points uniformly on the hypersphere does not incorporate the inherent class structure in the data, leading to poor FSL performance. Thus, there exists a tradeoff between uniformity on the hypersphere and the preservation of local similarities. To address this tradeoff, we introduce

[2]Our results assume hyperspheres with unit radius, but can easily be extended to hyperspheres with arbitrary radii.

[3]The proofs for all propositions are included in the supplementary.

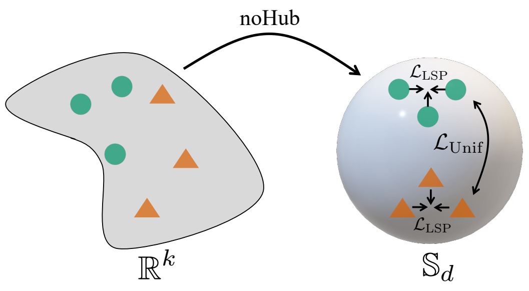

noHub
$\mathcal{L}_{\text{LSP}}$
$\mathcal{L}_{\text{Unif}}$
$\mathcal{L}_{\text{LSP}}$
$\mathbb{R}^k$
$\mathbb{S}_d$

Figure 2. Illustration of the noHub embedding. Given representations $\in \mathbb{R}^k$, $\mathcal{L}_{\text{LSP}}$ preserves local similarities. $\mathcal{L}_{\text{Unif}}$ simultaneously encourages uniformity in the embedding space $\mathbb{S}_d$. This feature embedding framework helps reduce hubness while improving classification performance.

two novel embedding approaches for FSL, namely noHub and noHub-S. noHub (Sec. 4.1) incorporates a novel loss function for embeddings on the hypersphere, while noHub-S (Sec. 4.2), guides noHub with additional label information, which should act as a supervisory signal for a class-aware embedding that leads to improved classification performance. Figure 2 provides an overview of the proposed noHub method. We also note that, since our approach generates embeddings, they are compatible with most transductive FSL classifier.

Few-shot Preliminaries. Assume we have a large labeled *base* dataset $\mathcal{X}_{\text{Base}} = \{(\boldsymbol{x}_i, y_i) \mid y_i \in \mathcal{C}_{\text{Base}};\ i = 1, \dots, n_{\text{Base}}\}$, where $\boldsymbol{x}_i$ and y_i denotes the raw features and labels, respectively. Let $\mathcal{C}_{\text{Base}}$ denote the set of classes for the base dataset. In the few–shot scenario, we assume that we are given another labeled dataset $\mathcal{X}_{\text{Novel}} = \{(\boldsymbol{x}_i, y_i) \mid y_i \in \mathcal{C}_{\text{Novel}};\ i = 1, \dots, n_{\text{Novel}}\}$ from *novel*, previously unseen classes $\mathcal{C}_{\text{Novel}}$, satisfying $\mathcal{C}_{\text{Base}} \cap \mathcal{C}_{\text{Novel}} = \emptyset$. In addition, we have a test set $\mathcal{T}$, $\mathcal{T} \cap \mathcal{X}_{\text{Novel}} = \emptyset$, also from $\mathcal{C}_{\text{Novel}}$.

In a K–way N_S–shot FSL problem, we create randomly sampled *tasks* (or episodes), with data from K randomly chosen novel classes. Each task consists of a *support* set $\mathcal{S} \subset \mathcal{X}_{\text{Novel}}$ and a *query* set $\mathcal{Q} \subset \mathcal{T}$. The support set contains $|\mathcal{S}| = N_S \cdot K$ random examples (N_S random examples from each of the K classes). The query set contains $|\mathcal{Q}| = N_Q \cdot K$ random examples, sampled from the same K classes. The goal of FSL is then to predict the class of samples $\boldsymbol{x} \in \mathcal{Q}$ by exploiting the labeled support set $\mathcal{S}$, using a model trained on the base classes $\mathcal{C}_{\text{Base}}$. We assume a fixed feature extractor, trained on the base classes, which maps the raw input data to the representations $\boldsymbol{x}_i$.

4.1. noHub: Uniform Hyperspherical Structure-preserving Embeddings

We design an embedding method that encourages uniformity on the hypersphere, and simultaneously preserves local similarity structure. Given the support and query representations $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^k$, $n = K(N_S + N_Q)$, we wish to find suitable embeddings $\boldsymbol{z}_1, \dots, \boldsymbol{z}_n \in \mathbb{S}_d$, where local similarities are preserved. For both representations and embeddings, we quantify similarities using a softmax over pairwise cosine similarities

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}, \quad p_{i|j} = \frac{\exp(\kappa_i \frac{\boldsymbol{x}_i^\top \boldsymbol{x}_j}{||\boldsymbol{x}_i|| \cdot ||\boldsymbol{x}_j||})}{\sum\limits_{l,m} \exp(\kappa_i \frac{\boldsymbol{x}_l^\top \boldsymbol{x}_m}{||\boldsymbol{x}_l|| \cdot ||\boldsymbol{x}_m||})} \tag{4}$$

and

$$q_{ij} = \frac{\exp(\kappa \boldsymbol{z}_i^\top \boldsymbol{z}_j)}{\sum\limits_{l,m} \exp(\kappa \boldsymbol{z}_l^\top \boldsymbol{z}_m)}, \tag{5}$$

where κ_i is chosen such that the effective number of neighbours of $\boldsymbol{x}_i$ equals a pre-defined perplexity[4]. As in [27, 30], local similarity preservation can now be achieved by minimizing the Kullback-Leibler (KL) divergence between the p_{ij} and the q_{ij}

$$KL(P||Q) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{6}$$

However, instead of directly minimizing $KL(P||Q)$, we find that the minimization problem is equivalent to minimizing the sum of two loss functions[5]

$$\underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \; KL(P||Q) = \underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \; \mathcal{L}_{\text{LSP}} + \mathcal{L}_{\text{Unif}} \tag{7}$$

where

$$\mathcal{L}_{\text{LSP}} = -\kappa \sum_{i,j} p_{ij} \boldsymbol{z}_i^\top \boldsymbol{z}_j, \tag{8}$$

$$\mathcal{L}_{\text{Unif}} = \log \sum_{l,m} \exp(\kappa \boldsymbol{z}_l^\top \boldsymbol{z}_m). \tag{9}$$

In Sec. 5 we provide a thorough theoretical analysis of these losses, and how they relate to LSP and uniformity on the hypersphere. Essentially, $\mathcal{L}_{\text{LSP}}$ is responsible for the local similarity preservation by ensuring that the embedding similarities ($\boldsymbol{z}_i^\top \boldsymbol{z}_j$) are high whenever the representation similarities (p_{ij}) are high. $\mathcal{L}_{\text{Unif}}$ on the other hand, can be interpreted as a negative entropy on $\mathbb{S}_d$, and is thus minimized when the embeddings are uniformly distributed on $\mathbb{S}_d$. This is discussed in more detail in Sec. 5.

Based on the decomposition of the KL divergence, and the subsequent interpretation of the two terms, we formulate the loss in noHub as the following tradeoff between LSP and uniformity

$$\mathcal{L}_{\text{noHub}} = \alpha \mathcal{L}_{\text{LSP}} + (1 - \alpha) \mathcal{L}_{\text{Unif}} \tag{10}$$

[4]Details on the computation of the κ_i are provided in the supplementary.
[5]Intermediate steps are provided in the supplementary.

```
Input: Features ∈ $\mathbb{R}^k$, $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$; perplexity, $P$;
        number of iterations, $T$ ; learning rate, $\eta$.
Output: Embeddings ∈ $\mathbb{S}_d$, $\{\boldsymbol{z}_1, \dots, \boldsymbol{z}_n\}$
Compute $p_{ij}$ from Eq (4)
Initialize solution $\boldsymbol{Z}^0 = \{\boldsymbol{z}_1, \dots, \boldsymbol{z}_n\}$ with PCA
for $i \leftarrow 1$ to $T$ do
    Compute $q_{ij}$ from Eq. (5)
    Compute gradients $\frac{d\mathcal{L}_{\text{noHub}}}{d\boldsymbol{Z}}$, using loss from Eq. (10)
    Update $\boldsymbol{Z}^t$ using the ADAM optimizer with learning
      rate $\eta$ [15]
    Re-normalize elements of $\boldsymbol{Z}^t$ using $L_2$ normalization
end
return $\boldsymbol{Z}^T$
```

Algorithm 1: noHub algorithm for embeddings on the hypersphere

where α is a weight parameter quantifying the tradeoff. $\mathcal{L}_{\text{noHub}}$ can then be optimized directly with gradient descent. The entire procedure is outlined in Algorithm 1.

4.2. noHub-S: noHub with Support labels

In order to strengthen the class structure in the embedding space, we modify $\mathcal{L}_{\text{LSP}}$ and $\mathcal{L}_{\text{Unif}}$ by exploiting the additional information provided by the support labels. For $\mathcal{L}_{\text{LSP}}$, we change the similarity function in p_{ij} such that

$$p_{i|j} = \frac{\exp(\kappa_i s_{\boldsymbol{x}}(\boldsymbol{x}_i, \boldsymbol{x}_j))}{\sum\limits_{l,m} \exp(\kappa_i s_{\boldsymbol{x}}(\boldsymbol{x}_l, \boldsymbol{x}_m))} \tag{11}$$

where

$$s_{\boldsymbol{x}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} 1 & \text{if } \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{S}, \text{ and } y_i = y_j \\ -1 & \text{if } \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{S}, \text{ and } y_i \neq y_j \\ \boldsymbol{x}_i^\top \boldsymbol{x}_j & \text{otherwise} \end{cases}. \tag{12}$$

With this, we encourage embeddings for support samples in the *same class* to be maximally similar, and support samples in *different classes* to be maximally dissimilar. Similarly, for $\mathcal{L}_{\text{Unif}}$

$$\mathcal{L}_{\text{Unif}} = \log \sum_{l,m} \exp(\kappa s_{\boldsymbol{z}}(\boldsymbol{z}_i, \boldsymbol{z}_j)) \tag{13}$$

where

$$s_{\boldsymbol{z}}(\boldsymbol{z}_i, \boldsymbol{z}_j) = \begin{cases} -\infty, & \text{if } \boldsymbol{z}_i, \boldsymbol{z}_j \in \mathcal{S}, \text{ and } y_i = y_j \\ \varepsilon \, \boldsymbol{z}_i^\top \boldsymbol{z}_j, & \text{if } \boldsymbol{z}_i, \boldsymbol{z}_j \in \mathcal{S}, \text{ and } y_i \neq y_j \\ \boldsymbol{z}_i^\top, \boldsymbol{z}_j & \text{otherwise} \end{cases} \tag{14}$$

where ε is a hyperparameter. This puts more emphasis on between-class uniformity by weighting the similarity higher for embeddings belonging to different classes ($\varepsilon > 1$), and ignoring the similarity between embeddings belonging to the same class[6]. The final loss function is the same as Eq. (10), but with the additional label-informed similarities in Eqs. (11)–(14).

5. Theoretical Results

In this section we provide a theoretical analysis of $\mathcal{L}_{\text{LSP}}$ and $\mathcal{L}_{\text{Unif}}$. Based on our analysis, we interpret these losses with regards to the Laplacian Eigenmaps algorithm and Rényi entropy, respectively.

Proposition 3. *Let $W_{ij} = \frac{1}{2}\kappa p_{ij}$, where $\sum_{i,j} p_{ij} = 1$, and let $\boldsymbol{z}_1, \dots, \boldsymbol{z}_n \in \mathbb{S}_d$. Then we have*

$$\mathcal{L}_{\text{LSP}} = \sum_{i,j} \|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2 W_{ij} - \kappa. \tag{15}$$

Proposition 4 (Minimizing $\mathcal{L}_{\text{Unif}}$ maximizes entropy)**.** *Let $H_2(\cdot)$ be the 2-order Rényi entropy, estimated with a kernel density estimator using a Gaussian kernel. Then*

$$\underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \; \mathcal{L}_{\text{Unif}} = \underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\max} \; H_2(\boldsymbol{z}_1, \dots, \boldsymbol{z}_n). \tag{16}$$

Definition 2 (Normalized counting measure)**.** *The normalized counting measure associated with a set B on A is*

$$\nu_B(A) = \frac{|B \cap A|}{|B|} \tag{17}$$

Definition 3 (Normalized surface area measure on $\mathbb{S}_d$)**.** *The normalized surface area measure on the hypersphere $\mathbb{S}_d \subset \mathbb{R}^d$, of a subset $S' \subset \mathbb{S}_d$ is*

$$\sigma_d(S') = \frac{\int_{S'} \mathrm{d}S}{\int_{\mathbb{S}_d} \mathrm{d}S} = A_d^{-1} \int_{S'} \mathrm{d}S \tag{18}$$

where A_d is defined as in Eq. (1), and $\int \mathrm{d}S$ denotes the surface integral on $\mathbb{S}_d$.

Definition 4 (Weak* convergence of measures [32])**.** *A sequence of Borel measures $\{\mu_n\}_{n=1}^{\infty}$ in $\mathbb{R}^d$ converges weak* to a Borel measure μ, if for all continuous functions $f: \mathbb{R}^d \to \mathbb{R}$,*

$$\lim_{n\to\infty} \int f(x)\mathrm{d}\mu_n(x) = \int f(x)\mathrm{d}\mu(x) \tag{19}$$

Proposition 5 (Minimizer of $\mathcal{L}_{\text{Unif}}$)**.** *For each $n > 0$, the n point minimizer of $\mathcal{L}_{\text{Unif}}$ is*

$$\boldsymbol{z}_1^\star, \dots, \boldsymbol{z}_n^\star = \underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \; \mathcal{L}_{\text{Unif}}. \tag{20}$$

Then $\nu_{\{\boldsymbol{z}_1^\star,\dots,\boldsymbol{z}_n^\star\}}$ converge weak to σ_d as $n \to \infty$.*

[6]Although any constant value would achieve the same result, we set the similarity to $-\infty$ in this case to remove the contribution to the final loss.

Interpretation of Proposition 3–5. Proposition 3 states an alternative formulation of $\mathcal{L}_{\text{LSP}}$, under the hyperspherical assumption. We recognize this formulation as the loss function in Laplacian Eigenmaps [2], which is known to produce *local similarity-preserving* embeddings from graph data. When unconstrained, this loss has a trivial solution where the embeddings for all representations are equal. This is avoided in our case since $\mathcal{L}_{\text{noHub}}$ (Eq. (10)) can be interpreted as the Lagrangian of minimizing $\mathcal{L}_{\text{LSP}}$ subject to a specified level of *entropy*, by Proposition 4.

Finally, Proposition 5 states that the normalized counting measure associated with the set of points that minimize $\mathcal{L}_{\text{Unif}}$, converges to the normalized surface area measure on the sphere. Since $u_{\mathbb{S}_d}$ is the density function associated with this measure, the points that minimize $\mathcal{L}_{\text{Unif}}$ will tend to be uniform on the sphere. Consequently, minimizing $\mathcal{L}_{\text{LSP}}$ also minimizes hubness, by Propositions 1 and 2.

6. Experiments

6.1. Setup

Implementation details. Our implementation is in PyTorch [20]. We optimize noHub and noHub-S for $T = 150$ iterations, using the Adam optimizer [15] with learning rate $\eta = 0.1$. The other hyperparameters were chosen based on validation performance on the respective datasets[7]. We analyze the effect of α in Sec. 6.2. Analyses of the κ and ε hyperparameters are provided in the supplementary.

Initialization. Since noHub and noHub-S reduce the embedding dimensionality ($d = 400$), we initialize embeddings with Principal Component Analysis (PCA) [13], instead of a naïve, random initialization. The PCA initialization is computationally efficient, and approximately preserves global structure. It also resulted in faster convergence and better performance, compared to random initialization.

Base feature extractors. We use the standard networks ResNet-18 [11] and Wide-Res28-10 [37] as the base feature extractors with pretrained weights from [28] and [18], respectively.

Datasets. Following common practice, we evaluate FSL performance on the *mini-ImageNet (mini)* [29], *tiered-ImageNet (tiered)* [23], and *CUB-200 (CUB)* [34] datasets.

Classifiers. We evaluate the baseline embeddings and our proposed methods using both established and recent FSL classifiers: *SimpleShot* [33], *LaplacianShot* [40], *$\alpha-$TIM* [28], *Oblique Manifold (OM)* [21], *iLPC* [16], and *SIAMESE* [39].

Baseline Embeddings. We compare our proposed method with a wide range of techniques for embedding the base features: *None* (No embedding of base features), *L2* [33], *Centered L2* [33], *ZN* [7], *ReRep* [5], *EASE* [39], and *TCPR* [35].

[7]Hyperparameter configurations for all experiments are included in the supplementary.

		mini		tiered		CUB	
Embedding	Feature Extractor	1-shot↑	5-shot↑	1-shot↑	5-shot↑	1-shot↑	5-shot↑
None	ResNet-18	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*
L2 (ARXIV'19 [33])	ResNet-18	73.77 (0.24)	83.14 (0.14)	80.46 (0.26)	87.04 (0.16)	83.1 (0.23)	89.48 (0.12)
CL2 (ARXIV'19 [33])	ResNet-18	75.56 (0.26)	84.04 (0.15)	82.1 (0.26)	87.9 (0.16)	84.35 (0.24)	90.14 (0.12)
ZN (ICCV'21 [7])	ResNet-18	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*
ReRep (ICML'21 [5])	ResNet-18	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*
EASE (CVPR'22 [39])	ResNet-18	76.05 (0.27)	84.61 (0.15)	82.57 (0.27)	88.33 (0.16)	85.24 (0.24)	90.42 (0.12)
TCPR (NEURIPS'22 [35])	ResNet-18	75.99 (0.26)	84.39 (0.15)	82.65 (0.26)	88.26 (0.16)	85.34 (0.23)	90.5 (0.11)
noHub (OURS)	ResNet-18	76.65 (0.28)	84.05 (0.16)	82.94 (0.27)	87.87 (0.17)	**85.88 (0.24)**	90.34 (0.12)
noHub-S (OURS)	ResNet-18	**76.68 (0.28)**	**84.67 (0.15)**	**83.09 (0.27)**	**88.43 (0.16)**	85.81 (0.24)	**90.52 (0.12)**
None	WideRes28-10	45.69 (0.31)	58.82 (0.31)	75.29 (0.28)	82.56 (0.22)	61.36 (0.55)	82.22 (0.37)
L2 (ARXIV'19 [33])	WideRes28-10	80.2 (0.23)	87.11 (0.13)	80.89 (0.26)	87.34 (0.15)	91.98 (0.18)	94.15 (0.1)
CL2 (ARXIV'19 [33])	WideRes28-10	75.23 (0.27)	83.99 (0.16)	79.59 (0.27)	86.71 (0.16)	92.17 (0.18)	94.48 (0.09)
ZN (ICCV'21 [7])	WideRes28-10	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*	20.0 (0.0)*
ReRep (ICML'21 [5])	WideRes28-10	36.69 (0.28)	36.41 (0.3)	67.41 (0.29)	76.49 (0.24)	57.62 (0.56)	60.36 (0.6)
EASE (CVPR'22 [39])	WideRes28-10	81.19 (0.25)	87.82 (0.13)	82.04 (0.26)	88.06 (0.16)	91.99 (0.19)	94.36 (0.09)
TCPR (NEURIPS'22 [35])	WideRes28-10	81.27 (0.24)	87.8 (0.13)	81.89 (0.26)	87.95 (0.16)	91.91 (0.18)	94.25 (0.1)
noHub (OURS)	WideRes28-10	81.97 (0.25)	87.78 (0.14)	82.8 (0.27)	87.99 (0.17)	92.53 (0.18)	94.56 (0.09)
noHub-S (OURS)	WideRes28-10	**82.0 (0.26)**	**88.03 (0.13)**	**82.85 (0.27)**	**88.31 (0.16)**	**92.63 (0.18)**	**94.69 (0.09)**

Table 1. Accuracies (Confidence interval) with the SIAMESE [39] classifier for different embedding approaches. Best and second best performance are denoted in **bold** and underlined, respectively. *The SIAMESE classifier is sensitive to the norm of the embedding, thus leading to detrimental performance for some of the embedding approaches.

Evaluation protocol. We follow the standard evaluation protocol in FSL and calculate the accuracy for 1-shot and 5-shot classification with 15 images per class in the query set. We evaluate on 10000 episodes, as is standard practice in FSL. Additionally, we evaluate the hubness of the representations after embedding using two common hubness metrics, namely the skewness (Sk) of the k-occurrence distribution [22] and the hub occurrence (HO) [8], which measures the percentage of hubs in the nearest neighbour lists of all points.

6.2. Results

Comparison to the state-of-the-art. To illustrate the effectiveness of noHub and noHub-S as an embedding approach for FSL, we consider the current state-of-the-art FSL method, which leverages the EASE embedding and obtains query predictions with SIAMESE [39]. We replace EASE with our proposed embedding approaches noHub and noHub-S, as well as other baseline embeddings, and evaluate performance on all datasets in the 1 and 5-shot setting. As shown in Table 1, noHub and noHub-S outperform all baseline approaches in both settings across all datasets, illustrating noHub's and noHub-S' ability to provide useful FSL embeddings, and updating the state-of-the-art in transductive FSL.

Aggregated FSL performance. To further evaluate the general applicability of noHub and noHub-S as embedding approaches, we perform extensive experiments for all classifiers and all baseline embeddings on all datasets. Tables 2a and 2b provide the results averaged over classifiers[8]. To clearly present the results, we aggregate the accuracy and a ranking *score* for each embedding method across all classifiers. The ranking score is calculated by performing a paired Student's t-test between all pairwise embedding methods for each classifier. We then average the ranking scores across all classifiers. A high ranking score then indicates that a method often significantly outperforms the competing embedding methods. We set the significance level to 5%. noHub and noHub-S consistently outperform previous embedding approaches – sometimes by a large margin. Overall, we further observe that noHub-S outperforms noHub in most settings and is particular beneficial in the 1-shot setting, which is more challenging, given that fewer samples are likely to generate noisy embeddings.

Hubness metrics. To further validate noHub's and noHub-S' ability to reduce hubness, we follow the same procedure of aggregating results for the hubness metrics and average over classifiers. Compared to the current state-of-the-art embedding approaches, Table 3 illustrates that noHub and noHub-S consistently result in embeddings with lower hubness.

Visualization of similarity matrices. As discussed in Sec. 4, completely eliminating hubness by distributing points uniformly on the hypersphere is not sufficient to obtain good FSL performance. Instead, representations need to also capture the inherent class structure of the data. To further evaluate the embedding approaches, we therefore compute the pairwise inner products for the embeddings of a random 5-shot episode on tiered-ImageNet with ResNet-18 features in Figure 3. It can be observed that the block structure is considerably more distinct for noHub and noHub-S, with

[8]The detailed results for all classifiers are provided in the supplementary.

		mini		tiered		CUB	
	Embedding	Acc$\uparrow$	Score$\uparrow$	Acc$\uparrow$	Score$\uparrow$	Acc$\uparrow$	Score$\uparrow$
ResNet18	None	55.74	0.17	62.61	0.0	63.78	0.17
	L2 (ArXiv'19 [33])	68.22	2.33	75.94	2.17	78.09	2.33
	CL2 (ArXiv'19 [33])	69.56	2.83	76.97	3.0	78.26	2.83
	ZN (ICCV'21 [7])	60.0	2.33	66.21	2.5	67.43	2.67
	ReRep (ICML'21 [5])	60.76	4.0	67.07	3.67	69.6	4.17
	EASE (CVPR'22 [39])	69.63	3.67	77.05	4.0	78.84	3.67
	TCPR (NeurIPS'22 [35])	69.97	4.0	77.18	3.33	78.83	4.0
	noHub (Ours)	72.58	6.83	79.77	6.83	81.91	6.83
	noHub-S (Ours)	**73.64**	**7.67**	**80.6**	**7.67**	**83.1**	**7.67**
WideRes28-10	None	63.59	1.0	71.29	0.83	79.23	1.17
	L2 (ArXiv'19 [33])	74.3	3.0	76.19	2.67	88.61	3.5
	CL2 (ArXiv'19 [33])	71.32	1.33	75.17	2.0	88.52	3.33
	ZN (ICCV'21 [7])	64.27	2.5	65.64	2.5	76.0	1.5
	ReRep (ICML'21 [5])	65.51	3.0	71.83	3.17	83.1	3.5
	EASE (CVPR'22 [39])	74.95	4.33	76.59	3.67	88.51	3.5
	TCPR (NeurIPS'22 [35])	75.64	4.83	76.51	4.0	88.22	2.5
	noHub (Ours)	78.22	7.0	79.76	7.0	90.25	5.67
	noHub-S (Ours)	**79.24**	**7.67**	**80.46**	**7.67**	**90.82**	**7.67**

(a) 1-shot

		mini		tiered		CUB	
		Acc$\uparrow$	Score$\uparrow$	Acc$\uparrow$	Score$\uparrow$	Acc$\uparrow$	Score$\uparrow$
ResNet18	None	69.83	0.83	74.38	0.67	76.01	1.17
	L2 (ArXiv'19 [33])	81.58	2.33	86.05	1.83	88.43	2.83
	CL2 (ArXiv'19 [33])	81.95	2.67	86.43	3.0	88.49	2.5
	ZN (ICCV'21 [7])	71.49	4.0	75.32	3.83	76.92	3.5
	ReRep (ICML'21 [5])	70.25	2.5	74.52	1.83	76.43	2.5
	EASE (CVPR'22 [39])	81.84	3.5	86.4	3.17	88.57	3.5
	TCPR (NeurIPS'22 [35])	82.1	4.0	86.54	3.83	88.79	4.33
	noHub (Ours)	82.58	5.5	86.9	4.5	**89.13**	**6.0**
	noHub-S (Ours)	**82.61**	**6.5**	**87.13**	**6.67**	88.93	5.33
WideRes28-10	None	78.77	1.5	84.1	1.67	89.49	1.67
	L2 (ArXiv'19 [33])	85.65	4.0	86.29	3.83	93.47	3.67
	CL2 (ArXiv'19 [33])	83.14	1.33	85.47	1.5	93.49	4.0
	ZN (ICCV'21 [7])	74.61	4.33	75.34	5.0	81.02	3.17
	ReRep (ICML'21 [5])	73.86	1.83	81.51	1.67	87.2	2.0
	EASE (CVPR'22 [39])	85.51	3.5	86.29	3.33	93.34	3.5
	TCPR (NeurIPS'22 [35])	86.03	**6.0**	86.37	4.0	93.3	3.0
	noHub (Ours)	**86.44**	5.67	**87.07**	5.5	93.65	4.17
	noHub-S (Ours)	85.95	5.5	87.05	**5.83**	**93.76**	**5.0**

(b) 5-shot

Table 2. Aggregated FSL performance for all embedding approaches on the mini-ImageNet, tiered-ImageNet, and CUB-200 datasets. Results are averaged over FSL classifiers. Best and second best performance are denoted in **bold** and underlined, respectively.

		mini		tiered		CUB	
		Sk$\downarrow$	HO$\downarrow$	Sk$\downarrow$	HO$\downarrow$	Sk$\downarrow$	HO$\downarrow$
ResNet18	None	1.349	0.407	1.211	0.408	0.887	0.341
	L2 (ArXiv'19 [33])	0.937	0.301	0.812	0.265	0.691	0.236
	CL2 (ArXiv'19 [33])	0.667	0.233	0.679	0.249	0.549	0.201
	ZN (ICCV'21 [7])	0.68	0.231	0.698	0.264	0.564	0.216
	ReRep (ICML'21 [5])	3.655	0.548	3.604	0.549	3.565	0.513
	EASE (CVPR'22 [39])	0.521	0.16	0.479	0.158	0.466	0.153
	TCPR (NeurIPS'22 [35])	0.651	0.228	0.65	0.25	0.532	0.204
	noHub (Ours)	0.315	**0.095**	0.303	**0.102**	0.32	**0.112**
	noHub-S (Ours)	**0.276**	0.13	**0.283**	0.127	**0.296**	0.162
WideRes28-10	None	1.6	0.459	1.81	0.494	1.073	0.369
	L2 (ArXiv'19 [33])	0.781	0.296	0.737	0.275	0.475	0.228
	CL2 (ArXiv'19 [33])	0.981	0.288	0.817	0.307	0.52	0.267
	ZN (ICCV'21 [7])	0.73	0.287	0.769	0.302	0.517	0.263
	ReRep (ICML'21 [5])	3.56	0.704	3.55	0.777	3.026	0.47
	EASE (CVPR'22 [39])	0.47	0.177	0.477	0.175	0.437	0.213
	TCPR (NeurIPS'22 [35])	0.589	0.236	0.685	0.264	0.477	0.231
	noHub (Ours)	0.29	**0.111**	0.301	**0.111**	0.188	**0.108**
	noHub-S (Ours)	**0.258**	0.148	**0.274**	0.135	**0.162**	0.13

(a) 1-shot

		mini		tiered		CUB	
		Sk$\downarrow$	HO$\downarrow$	Sk$\downarrow$	HO$\downarrow$	Sk$\downarrow$	HO$\downarrow$
ResNet18	None	1.436	0.422	1.339	0.432	0.987	0.364
	L2 (ArXiv'19 [33])	1.04	0.318	0.914	0.287	0.812	0.263
	CL2 (ArXiv'19 [33])	0.786	0.264	0.821	0.28	0.698	0.236
	ZN (ICCV'21 [7])	0.806	0.264	0.839	0.296	0.716	0.25
	ReRep (ICML'21 [5])	1.631	0.863	1.721	0.872	1.432	0.869
	EASE (CVPR'22 [39])	0.624	0.186	0.598	0.183	0.607	0.186
	TCPR (NeurIPS'22 [35])	0.78	0.259	0.796	0.283	0.687	0.235
	noHub (Ours)	0.286	0.096	0.289	0.104	**0.329**	0.12
	noHub-S (Ours)	**0.25**	**0.074**	**0.213**	**0.078**	0.433	**0.097**
WideRes28-10	None	1.709	0.473	1.937	0.51	1.16	0.395
	L2 (ArXiv'19 [33])	0.887	0.322	0.86	0.305	0.632	0.266
	CL2 (ArXiv'19 [33])	1.12	0.318	0.956	0.337	0.701	0.31
	ZN (ICCV'21 [7])	0.858	0.32	0.912	0.335	0.699	0.305
	ReRep (ICML'21 [5])	1.597	0.819	1.617	0.846	1.299	0.549
	EASE (CVPR'22 [39])	0.579	0.199	0.585	0.193	0.572	0.241
	TCPR (NeurIPS'22 [35])	0.717	0.27	0.815	0.294	0.634	0.264
	noHub (Ours)	**0.294**	0.115	**0.298**	**0.115**	**0.195**	**0.1**
	noHub-S (Ours)	0.494	**0.103**	0.407	0.12	0.421	0.127

(b) 5-shot

Table 3. Aggregated hubness metrics for all embedding approaches on the Mini-ImageNet, Tiered-ImageNet and CUB-200 dataset. Results are averaged over FSL classifiers. Best and second best performance are denoted in **bold** and underlined, respectively.

noHub-S slightly improving upon noHub. These results indicate that (i) samples are more uniform, indicating the reduced hubness; and (ii) classes are better separated, due to the local similarity preservation.

Tradeoff between uniformity and similarity preservation. We analyze the effect of α on the tradeoff between LSP and Uniformity in the loss function in Eq. (10), on tiered-ImageNet with ResNet-18 features in the 5-shot setting and with the SIAMESE [39] classifier. The results are visualized in Figure 4. We notice a sharp increase in performance when we have a high emphasis on uniformity. This demonstrates the impact of hubness on accuracy in FSL performance. As we keep increasing the emphasis on LSP, however, after a certain point we notice a sharp drop off in performance. This is due to the fact that the classifier does not take into account the uniformity constraint on the features, resulting in a large number of misclassifications. In general, we observe that noHub-S is slightly more robust compared to noHub.

Increasing number of classes. We analyze the behavior of noHub and noHub-S for an increasing number of classes (ways) on the tiered-ImageNet dataset with SIAMESE [39]

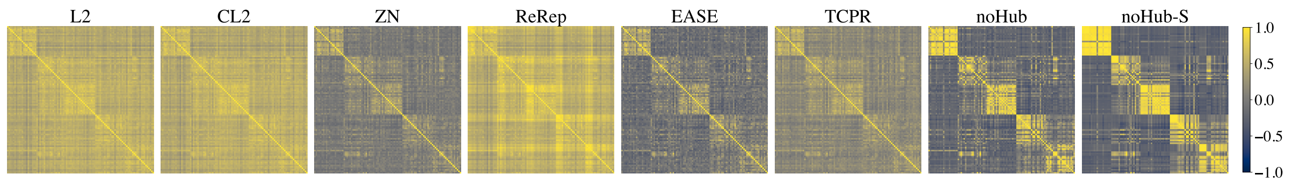


Figure 3. Inner product matrices between features for a random episode for all embedding approaches.

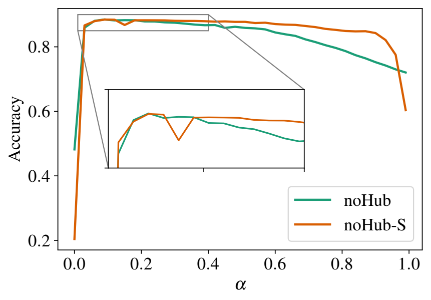


Figure 4. Accuracies for different values of the weighting parameter, α, which quantifies the tradeoff between $\mathcal{L}_{\text{LSP}}$ and $\mathcal{L}_{\text{Unif}}$.

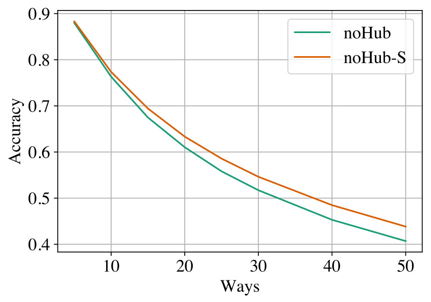


Figure 5. Accuracies for an increasing number of classes (ways) for noHub and noHub-S.

as classifier. While classification accuracy generally decreases with an increasing number of classes, which is expected, we observe from Figure 5 that noHub-S has a slower decay and is able to leverage the label guidance to obtain better performance for a larger number of classes.

Effect of label information in $\mathcal{L}_{\text{LSP}}$ and $\mathcal{L}_{\text{Unif}}$. To validate the effectiveness of using label guidance in noHub-S, we study the result of including label information in $\mathcal{L}_{\text{LSP}}$ and $\mathcal{L}_{\text{Unif}}$ (Eqs. (11)–(14)). We note that the default setting of noHub is that none of the two losses include label information. Ablation experiments are performed on tiered-ImageNet with the ResNet-18 feature extractor and the SimpleShot and SIAMESE classifier [39]. In Table 4, we generally see improvements of noHub-S when *both* the loss terms are label-informed, indicating the usefulness of label guidance.

We further observe that incorporating label information in $\mathcal{L}_{\text{Unif}}$ tends to have a larger contribution than doing the same for $\mathcal{L}_{\text{LSP}}$. This aligns with our observations in Figure 4,

	Label-informed		SimpleShot [33]		SIAMESE [39]	
	$\mathcal{L}_{\text{LSP}}$	$\mathcal{L}_{\text{Unif}}$	1-shot$\uparrow$	5-shot$\uparrow$	1-shot$\uparrow$	5-shot$\uparrow$
noHub	–	–	76.72 (0.23)	**86.31 (0.16)**	82.94 (0.27)	87.87 (0.17)
noHub-S	✓	–	78.25 (0.24)	85.46 (0.16)	82.56 (0.28)	88.07 (0.17)
noHub-S	–	✓	78.33 (0.23)	86.15 (0.15)	82.81 (0.27)	**88.43 (0.16)**
noHub-S	✓	✓	**78.35 (0.23)**	86.22 (0.15)	**83.09 (0.27)**	**88.43 (0.16)**

Table 4. Ablation study with the label-informed losses in noHub-S. Check marks (✓) indicate that the loss uses information from the support labels.

where a small α yielded the best performance.

7. Conclusion

In this paper we have addressed the hubness problem in FSL. We have shown that hubness is eliminated by embedding representations uniformly on the hypersphere. The hyperspherical uniform distribution has zero mean and zero density gradient at all points along all directions tangent to the hypersphere – both of which are identified as causes of hubness in previous work [9, 22]. Based on our theoretical findings about hubness and hyperspheres, we proposed two new methods to embed representations on the hypersphere for FSL. The proposed noHub and noHub-S leverage a decomposition of the KL divergence between similarity distributions, and optimize a tradeoff between LSP and uniformity on the hypersphere – thus reducing hubness while maintaining the class structure in the representation space. We have provided theoretical analyses and interpretations of the LSP and uniformity losses, proving that they optimize LSP and uniformity, respectively. We comprehensively evaluate the proposed methods on several datasets, features extractors, and classifiers, and compare to a number of recent state-of-the-art baselines. Our results illustrate the effectiveness of our proposed methods and show that we achieve state-of-the-art performance in transductive FSL.

Acknowledgements

This work was financially supported by the Research Council of Norway (RCN), through its Centre for Research-based Innovation funding scheme (Visual Intelligence, grant no. 309439), and Consortium Partners. It was further funded by RCN FRIPRO grant no. 315029, RCN IKTPLUSS grant no. 303514, and the UiT Thematic Initiative "Data-Driven Health Technology".

References

[1] Kelsey R. Allen, Evan Shelhamer, Hanul Shin, and Joshua B. Tenenbaum. Infinite mixture prototypes for few-shot learning. In *ICML*, 2019. 2

[2] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 2003. 5

[3] Malik Boudiaf, Ziko Imtiaz Masud, Jérôme Rony, José Dolz, Pablo Piantanida, and Ismail Ben Ayed. Transductive Information Maximization For Few-Shot Learning. In *NeurIPS*, 2020. 1, 2

[4] Philip Chikontwe, Soopil Kim, and Sang Hyun Park. CAD: Co-Adapting Discriminative Features for Improved Few-Shot Classification. In *CVPR*, 2022. 1

[5] Wentao Cui and Yuhong Guo. Parameterless Transductive Feature Re-representation for Few-Shot Learning. In *ICML*, 2021. 1, 2, 5, 6, 7

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1

[7] Nanyi Fei, Yizhao Gao, Zhiwu Lu, and Tao Xiang. Z-Score Normalization, Hubness, and Few-Shot Learning. In *ICCV*, 2021. 1, 2, 5, 6, 7

[8] Arthur Flexer and Dominik Schnitzer. Choosing ℓ^p norms in high-dimensional spaces based on hub analysis. *Neurocomputing*, 2015. 6

[9] Kazuo Hara, Ikumi Suzuki, Kei Kobayashi, Kenji Fukumizu, and Milos Radovanovic. Flattening the Density Gradient for Eliminating Spatial Centrality to Reduce Hubness. In *AAAI*, 2016. 2, 8

[10] Kazuo Hara, Ikumi Suzuki, Masashi Shimbo, Kei Kobayashi, Kenji Fukumizu, and Milos Radovanovic. Localized Centering: Reducing Hubness in Large-Sample Data. In *AAAI*, 2015. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 5

[12] Shell Xu Hu, Da Li, Jan Stuhmer, Minyoung Kim, and Timothy M Hospedales. Pushing the Limits of Simple Pipelines for Few-Shot Learning: External Data and Fine-Tuning Make a Difference. In *CVPR*, 2022. 1

[13] Ian T Jolliffe. *Principal component analysis for special types of data*. Springer, 2002. 5

[14] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *CVPR*, 2019. 1

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 4, 5

[16] Michalis Lazarou, Tania Stathaki, and Yannis Avrithis. Iterative Label Cleaning for Transductive and Semi-Supervised Few-Shot Learning. In *ICCV*, 2021. 1, 5

[17] Duong H Le, Khoi D Nguyen, and Khoi Nguyen. POODLE: Improving Few-shot Learning via Penalizing Out-of-Distribution Samples. In *NeurIPS*, 2021. 1

[18] Puneet Mangla, Mayank Singh, Abhishek Sinha, Nupur Kumari, Vineeth N Balasubramanian, and Balaji Krishnamurthy. Charting the Right Manifold: Manifold Mixup for Few-shot Learning. In *WACV*, 2020. 5

[19] Van Nhan Nguyen, Sigurd Løkse, Kristoffer Wickstrøm, Michael Kampffmeyer, Davide Roverso, and Robert Jenssen. Sen: A novel feature normalization dissimilarity measure for prototypical few-shot learning networks. In *ECCV*, 2020. 2

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 5

[21] Guodong Qi, Huimin Yu, Zhaohui Lu, and Shuzhao Li. Transductive Few-Shot Classification on the Oblique Manifold. In *ICCV*, 2021. 1, 5

[22] Miloš Radovanovic, Alexandros Nanopoulos, and Mirjana Ivanovic. Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data. *JMLR*, 2010. 1, 2, 6, 8

[23] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for Semi-supervised Few-shot Classification. In *ICLR*, 2018. 5

[24] Yutaro Shigeto, Ikumi Suzuki, Kazuo Hara, Masashi Shimbo, and Yuji Matsumoto. Ridge Regression, Hubness, and Zero-Shot Learning. In *ECML-PKDD*, 2015. 1

[25] Ikumi Suzuki, Kazuo Hara, Masashi Shimbo, Marco Saerens, and Kenji Fukumizu. Centering Similarity Measures to Reduce Hubs. In *EMNLP*, 2013. 1

[26] Ran Tao, Han Zhang, Yutong Zheng, and Marios Savvides. Powering Finetuning in Few-Shot Learning: Domain-Agnostic Bias Reduction with Selected Sampling. In *AAAI*, 2022. 1

[27] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *JMLR*, 2008. 4

[28] Olivier Veilleux, Malik Boudiaf, Pablo Piantanida, and Ismail Ben Ayed. Realistic evaluation of transductive few-shot learning. In *NeurIPS*, 2021. 1, 5

[29] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. In *NeurIPS*, 2016. 1,

5

[30] Mian Wang and Dong Wang. VMF-SNE: embedding for spherical data. *arxiv:1507.08379 [cs]*, 2015. 4

[31] Ruohan Wang, Massimiliano Pontil, and Carlo Ciliberto. The Role of Global Labels in Few-Shot Classification and How to Infer Them. In *NeurIPS*, 2021. 1

[32] Tongzhou Wang and Phillip Isola. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *ICML*, 2020. 2, 5

[33] Yan Wang, Wei-Lun Chao, Kilian Q. Weinberger, and Laurens van der Maaten. SimpleShot: Revisiting Nearest-Neighbor Classification for Few-Shot Learning. *arXiv:1911.04623 [cs]*, 2019. 1, 2, 5, 6, 7, 8

[34] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. Technical Report CNS-TR-201, Caltech, 2010. 5

[35] Jing Xu, Xu Luo, Xinglin Pan, Wenjie Pei, Yanan Li, and Zenglin Xu. Alleviating the Sample Selection Bias in Few-shot Learning by Removing Projection to the Centroid. In *NeurIPS*, 2022. 1, 2, 5, 6, 7

[36] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *CVPR*, 2020. 2

[37] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 5

[38] Manli Zhang, Jianhong Zhang, Zhiwu Lu, Tao Xiang, Mingyu Ding, and Songfang Huang. IEPT: Instance-level and episode-level pretext tasks for few-shot learning. In *ICLR*, 2021. 2

[39] Hao Zhu and Piotr Koniusz. EASE: Unsupervised Discriminant Subspace Learning for Transductive Few-Shot Learning. In *CVPR*, 2022. 1, 2, 5, 6, 7, 8

[40] Imtiaz Ziko, Jose Dolz, Eric Granger, and Ismail Ben Ayed. Laplacian Regularized Few-Shot Learning. In *ICML*, 2020. 1, 2, 5

Supplementary material –
Hubs and Hyperspheres: Reducing Hubness and Improving Transductive Few-shot Learning with Hyperspherical Embeddings

Daniel J. Trosten[*,†], Rwiddhi Chakraborty[*,†], Sigurd Løkse[†], Kristoffer Knutsen Wickstrøm[†],
Robert Jenssen[†,‡,§,¶], Michael C. Kampffmeyer[†,‡]
Department of Physics and Technology, UiT The Arctic University of Norway
`firstname[.middle initial].lastname@uit.no`

1. Introduction

Here we provide proofs for our theoretical results on the hyperspherical uniform and hubness; the decomposition of the KL divergence; and the minima of our methods' loss functions. We also give additional details on the implementation and hyperparameters for noHub and noHub-S– and include the complete tables of 1-shot and 5-shot results for all classifiers, datasets and feature extractors. Finally, we briefly reflect on potential negative societal impacts of our work.

2. Hyperspherical Uniform Eliminates Hubness

Definition 1 (Uniform PDF on the hypersphere.)**.** *The uniform probability density function (PDF) on the unit hypersphere $\mathbb{S}_d = \{\boldsymbol{x} \in \mathbb{R}^d \mid ||\boldsymbol{x}|| = 1\} \subset \mathbb{R}^d$ is*

$$u_{\mathbb{S}_d}(\boldsymbol{x}) = A_d^{-1}\delta(||\boldsymbol{x}|| - 1) \tag{1}$$

where $A_d = \frac{2\pi^{d/2}}{\Gamma(d/2)}$ is the surface area of $\mathbb{S}_d$, and $\delta(\cdot)$ is the Dirac delta distribution.

Lemma 1 (Trisection of hypersphere)**.** *The trisection of the hypersphere along coordinate i is given by the three-tuple of disjoint sets $(\mathbb{S}_d^{i,+}, \mathbb{S}_d^{i,-}, \mathbb{S}_d^{i,0})$ where*

$$\mathbb{S}_d^{i,+} = \{\boldsymbol{x} = [x^1, \dots, x^d]^\top \in \mathbb{S}_d \mid x^i > 0\} \tag{2}$$
$$\mathbb{S}_d^{i,-} = \{\boldsymbol{x} = [x^1, \dots, x^d]^\top \in \mathbb{S}_d \mid x^i < 0\} \tag{3}$$
$$\mathbb{S}_d^{i,0} = \{\boldsymbol{x} = [x^1, \dots, x^d]^\top \in \mathbb{S}_d \mid x^i = 0\} \tag{4}$$

and

$$\mathbb{S}_d^{i,+} \cup \mathbb{S}_d^{i,-} \cup \mathbb{S}_d^{i,0} = \mathbb{S}_d \tag{5}$$

Then we have

$$\mathbb{S}_d^{i,+} = -\mathbb{S}_d^{i,-} = \{-\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{S}_d^{i,-}\} \tag{6}$$

Proof. Let $\boldsymbol{x} \in \mathbb{S}_d^{i,+}$, then

$$||(-x)|| = ||x|| = 1, \tag{7}$$

and

$$-(x^i) < 0. \tag{8}$$

Hence $\boldsymbol{x} \in -\mathbb{S}_d^{i,-}$, and $\mathbb{S}_d^{i,+} \subseteq -\mathbb{S}_d^{i,-}$.

Similarly, let $-\boldsymbol{x} \in -\mathbb{S}_d^{i,-}$, then

$$|| - (-x)|| = ||x|| = 1, \tag{9}$$

and

$$-(-x^i) = x^i > 0. \tag{10}$$

Hence $\boldsymbol{x} \in \mathbb{S}_d^{i,+}$, and $-\mathbb{S}_d^{i,-} \subseteq \mathbb{S}_d^{i,+}$.

It then follows that $\mathbb{S}_d^{i,+} = -\mathbb{S}_d^{i,-}$. □

Proposition 1. *Suppose $\boldsymbol{X}$ has PDF $u_{\mathbb{S}_d}(\boldsymbol{x})$. Then*

$$\mathbb{E}(\boldsymbol{X}) = 0 \tag{11}$$

Proof. The expectation $\mathbb{E}(\boldsymbol{X})$ is given by

$$\mathbb{E}(\boldsymbol{X}) = \int_{\mathbb{R}^d} \boldsymbol{x} u_{\mathbb{S}_d}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \tag{12}$$

Since $u_{\mathbb{S}_d}$ is non-zero only on the hypersphere $\mathbb{S}_d$, the integral can be rewritten as a surface integral over $\mathbb{S}_d$

$$\mathbb{E}(\boldsymbol{X}) = \int_{\mathbb{S}_d} \boldsymbol{x} A_d^{-1} \mathrm{d}S. \tag{13}$$

[*]Equal contributions.
[†]UiT Machine Learning group (machine-learning.uit.no) and Visual Intelligence Centre (visual-intelligence.no).
[‡]Norwegian Computing Center.
[§]Department of Computer Science, University of Copenhagen.
[¶]Pioneer Centre for AI (aicentre.dk).

Decomposing the integral over the trisection of $\mathbb{S}_d$ along coordinate i gives

$$\int_{\mathbb{S}_d} \boldsymbol{x} A_d^{-1} \mathrm{d}S = \tag{14}$$

$$A_d^{-1}\left(\int_{\mathbb{S}_d^{i,+}} \boldsymbol{x}\mathrm{d}S + \int_{\mathbb{S}_d^{i,-}} \boldsymbol{x}\mathrm{d}S + \int_{\mathbb{S}_d^{i,0}} \boldsymbol{x}\mathrm{d}S\right). \tag{15}$$

By Lemma 1 we have

$$\mathbb{S}_d^{i,+} = -\mathbb{S}_d^{i,-} \Rightarrow \int_{\mathbb{S}_d^{i,+}} \boldsymbol{x}\mathrm{d}S = -\int_{\mathbb{S}_d^{i,-}} \boldsymbol{x}\mathrm{d}S. \tag{16}$$

Furthermore, since the set $\mathbb{S}_d^{i,0}$ has zero width along coordinate i, $\int_{\mathbb{S}_d^{i,0}} \boldsymbol{x}\mathrm{d}S = 0$. Hence

$$\mathbb{E}(\boldsymbol{X}) = \tag{17}$$

$$A_d^{-1}\left(\int_{\mathbb{S}_d^{i,+}} \boldsymbol{x}\mathrm{d}S - \int_{\mathbb{S}_d^{i,+}} \boldsymbol{x}\mathrm{d}S + \int_{\mathbb{S}_d^{i,0}} \boldsymbol{x}\mathrm{d}S\right) = 0 \tag{18}$$

□

Proposition 2. *Let $\Pi_{\boldsymbol{p}}$ be the tangent plane of $\mathbb{S}_d$ at an arbitrary point $\boldsymbol{p} \in \mathbb{S}_d$. Then, for any direction $\boldsymbol{\theta}^*$ in $\Pi_{\boldsymbol{p}}$ the directional derivative of $u_{\mathbb{S}_d}$ along $\boldsymbol{\theta}^*$ is*

$$\nabla_{\boldsymbol{\theta}^*} u_{\mathbb{S}_d} = 0 \tag{19}$$

Proof. $u_{\mathbb{S}_d}(\boldsymbol{x})$ can be written in polar coordinates as

$$u_{\mathbb{S}_d}(\boldsymbol{x}(r,\boldsymbol{\theta})) = u_{\mathbb{S}_d}^{\text{Polar}}(r,\boldsymbol{\theta}) = A_d^{-1}\delta(r-1) \tag{20}$$

The gradient of $u_{\mathbb{S}_d}^{\text{Polar}}(r,\boldsymbol{\theta})$ is then

$$\nabla_{(r,\boldsymbol{\theta})} u_{\mathbb{S}_d}^{\text{Polar}}(r,\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial}{\partial r} u_{\mathbb{S}_d}^{\text{Polar}}(r,\boldsymbol{\theta}) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{21}$$

For an arbitrary point $\boldsymbol{p} \in \mathbb{S}_d$, an arbitrary unit vector (direction), $\boldsymbol{\theta}^*$, in the tangent plane $\Pi_{\boldsymbol{p}}$ is given by

$$\boldsymbol{\theta}^* = \begin{bmatrix} 0 \\ \theta_1^* \\ \vdots \\ \theta_{d-1}^* \end{bmatrix} \tag{22}$$

The directional derivative of $u_{\mathbb{S}_d}(\boldsymbol{x})$ along $\boldsymbol{\theta}^*$ is then

$$\nabla_{\boldsymbol{\theta}^*} u_{\mathbb{S}_d}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial}{\partial r} u_{\mathbb{S}_d}^{\text{Polar}}(r,\boldsymbol{\theta}) \\ 0 \\ \vdots \\ 0 \end{bmatrix}^\top \cdot \begin{bmatrix} 0 \\ \theta_1^* \\ \vdots \\ \theta_{d-1}^* \end{bmatrix} = 0 \tag{23}$$

□

3. Method

Computing κ_i. Following [5], we compute κ_i using a binary search such that

$$|\log_2(P) - H(P_i)| \leq 0.1 \cdot \log_2(P) \tag{24}$$

where P is a hyperparameter, and $H(P_i)$ is the Shannon entropy of similarities for representation i

$$H(P_i) = \sum_{j=1}^{n} p_{i|j} \log_2(p_{i|j}). \tag{25}$$

Decomposition of $KL(P||Q)$. Recall that

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}, \quad p_{i|j} = \frac{\exp(\kappa_i \boldsymbol{x}_i^\top \boldsymbol{x}_j)}{\sum\limits_{l,m} \exp(\kappa_i \boldsymbol{x}_l^\top \boldsymbol{x}_m)} \tag{26}$$

and

$$q_{ij} = \frac{\exp(\kappa \boldsymbol{z}_i^\top \boldsymbol{z}_j)}{\sum\limits_{l,m} \exp(\kappa \boldsymbol{z}_l^\top \boldsymbol{z}_m)}. \tag{27}$$

Since p_{ij} is constant w.r.t. q_{ij}, we have

$$\underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min}\ KL(P||Q) = \underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{28}$$

$$= \underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \underbrace{\sum_{i,j} p_{ij} \log p_{ij}}_{\text{constant}} - \sum_{i,j} p_{ij} \log q_{ij} \tag{29}$$

$$= \underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \underbrace{-\sum_{i,j} p_{ij} \log q_{ij}}_{=:\tilde{\mathcal{L}}} \tag{30}$$

Minimizing $KL(P||Q)$ over $\boldsymbol{z}_1, \dots, \boldsymbol{z}_n \in \mathbb{S}_d$ is therefore equivalent to minimizing $\tilde{\mathcal{L}}$.

Decomposing $\tilde{\mathcal{L}}$ gives

$$\tilde{\mathcal{L}} = -\sum_{i,j} p_{ij} \kappa \boldsymbol{z}_i^\top \boldsymbol{z}_j + \tag{31}$$

$$\sum_{i,j}\left(p_{ij} \log \sum_{l,m} \exp(\kappa \boldsymbol{z}_l^\top \boldsymbol{z}_m)\right) \tag{32}$$

$$= -\sum_{i,j} p_{ij} \kappa \boldsymbol{z}_i^\top \boldsymbol{z}_j \tag{33}$$

$$+ \left(\log \sum_{l,m} \exp(\kappa \boldsymbol{z}_l^\top \boldsymbol{z}_m)\right) \cdot \underbrace{\left(\sum_{i,j} p_{ij}\right)}_{=1} \tag{34}$$

$$= \underbrace{-\sum_{i,j} p_{ij} \kappa \boldsymbol{z}_i^\top \boldsymbol{z}_j}_{=:\ \mathcal{L}_{\text{LSP}}} + \underbrace{\log \sum_{l,m} \exp(\kappa \boldsymbol{z}_l^\top \boldsymbol{z}_m)}_{=:\ \mathcal{L}_{\text{Unif}}} \tag{35}$$

Thus, we have shown that

$$\underset{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\in\mathbb{S}_d}{\arg\min}\; KL(P||Q) = \underset{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\in\mathbb{S}_d}{\arg\min}\; \mathcal{L}_{\text{LSP}} + \mathcal{L}_{\text{Unif}}. \tag{36}$$

4. Theoretical Results

Proposition 3. *Let $W_{ij} = \frac{1}{2}\kappa p_{ij}$, where $\sum_{i,j} p_{ij} = 1$, and let $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n \in \mathbb{S}_d$. Then we have*

$$\mathcal{L}_{\text{LSP}} = \sum_{i,j} \|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2 W_{ij} - \kappa. \tag{37}$$

Proof. We have

$$\mathcal{L}_{\text{LSP}} = -\kappa \sum_{i,j} p_{ij}\boldsymbol{z}_i^\top \boldsymbol{z}_j \tag{38}$$

$$= -2\sum_{i,j}\frac{1}{2}\kappa p_{ij}\boldsymbol{z}_i^\top \boldsymbol{z}_j + \sum_{i,j} 2\frac{1}{2}\kappa p_{ij} - \kappa \tag{39}$$

$$(\sum_{i,j} p_{ij} = 1)$$

$$= -2\sum_{i,j}\boldsymbol{z}_i^\top \boldsymbol{z}_j W_{ij} + \sum_{i,j}(\|\boldsymbol{z}_i\| + \|\boldsymbol{z}_j\|)W_{ij} - \kappa \tag{40}$$

$$(\|\boldsymbol{z}_i\| = \|\boldsymbol{z}_j\| = 1)$$

$$= \sum_{i,j}(\|\boldsymbol{z}_i\| - 2\boldsymbol{z}_i^\top \boldsymbol{z}_j + \|\boldsymbol{z}_j\|)W_{ij} - \kappa \tag{41}$$

$$= \sum_{i,j} |\boldsymbol{z}_i - \boldsymbol{z}_j\|^2 W_{ij} - \kappa. \tag{42}$$

□

Proposition 4 (Minimizing $\mathcal{L}_{\text{Unif}}$ maximizes entropy)**.** *Let $H_2(\cdot)$ be the 2-order Rényi entropy, estimated with a kernel density estimator using a Gaussian kernel. Then*

$$\underset{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\in\mathbb{S}_d}{\arg\min}\; \mathcal{L}_{\text{Unif}} = \underset{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\in\mathbb{S}_d}{\arg\max}\; H_2(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n). \tag{43}$$

Proof. Using a Gaussian kernel, the 2-order Rényi entropy can be estimated as [4, Eq. (2.13)]

$$H_2(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n) = -\log\left(\frac{1}{n^2}\sum_{l,m}\exp(-\frac{1}{2}\kappa\|\boldsymbol{z}_l - \boldsymbol{z}_m\|^2)\right) \tag{44}$$

Thus, we have

$$\underset{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\in\mathbb{S}_d}{\arg\max}\; H_2(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n) \tag{45}$$

$$= \underset{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\in\mathbb{S}_d}{\arg\max}\; -\log\left(\frac{1}{n^2}\sum_{l,m}\exp(-\frac{1}{2}\kappa\|\boldsymbol{z}_l - \boldsymbol{z}_m\|^2)\right) \tag{46}$$

$$= \underset{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\in\mathbb{S}_d}{\arg\min}\; \log\left(\sum_{l,m}\exp(-\frac{1}{2}\kappa\|\boldsymbol{z}_l - \boldsymbol{z}_m\|^2)\right) \tag{47}$$

$$= \underset{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\in\mathbb{S}_d}{\arg\min}\; \log\left(\sum_{l,m}\exp(-\frac{1}{2}\kappa(\|\boldsymbol{z}_l\|^2\right. \tag{48}$$

$$\left. - 2\boldsymbol{z}_l^\top \boldsymbol{z}_m + \|\boldsymbol{z}_m\|^2)\right) \tag{49}$$

$$= \underset{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\in\mathbb{S}_d}{\arg\min}\; \log\left(\sum_{l,m}\exp(-\kappa(1 - \boldsymbol{z}_l^\top \boldsymbol{z}_m))\right) \tag{50}$$

$$(\|\boldsymbol{z}_l\| = \|\boldsymbol{z}_m\| = 1)$$

$$= \underset{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\in\mathbb{S}_d}{\arg\min}\; \log\left(\exp(-\kappa)\sum_{l,m}\exp(\kappa\boldsymbol{z}_l^\top \boldsymbol{z}_m)\right) \tag{51}$$

$$= \underset{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\in\mathbb{S}_d}{\arg\min}\; \log\sum_{l,m}\exp(\kappa\boldsymbol{z}_l^\top \boldsymbol{z}_m) \tag{52}$$

$$= \underset{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\in\mathbb{S}_d}{\arg\min}\; \mathcal{L}_{\text{Unif}}. \tag{53}$$

□

Definition 2 (Normalized counting measure)**.** *The normalized counting measure associated with a set B on A is*

$$\nu_B(A) = \frac{|B \cap A|}{|B|} \tag{54}$$

Definition 3 (Normalized surface area measure on $\mathbb{S}_d$)**.** *The normalized surface area measure on the hypersphere $\mathbb{S}_d \subset \mathbb{R}^d$, of a subset $S' \subset \mathbb{S}_d$ is*

$$\sigma_d(S') = \frac{\int_{S'} \mathrm{d}S}{\int_{\mathbb{S}_d} \mathrm{d}S} = A_d^{-1}\int_{S'} \mathrm{d}S \tag{55}$$

where A_d is defined as in Eq. (1)*, and $\int \mathrm{d}S$ denotes the surface integral on $\mathbb{S}_d$.*

Definition 4 (Weak* convergence of measures [8])**.** *A sequence of Borel measures $\{\mu_n\}_{n=1}^\infty$ in $\mathbb{R}^d$ converges weak* to a Borel measure μ, if for all continuous functions $f : \mathbb{R}^d \to \mathbb{R}$,*

$$\lim_{n\to\infty}\int f(x)\mathrm{d}\mu_n(x) = \int f(x)\mathrm{d}\mu(x) \tag{56}$$

Proposition 5 (Minimizer of $\mathcal{L}_{\text{Unif}}$). *For each $n > 0$, the n point minimizer of $\mathcal{L}_{\text{Unif}}$ is*

$$\boldsymbol{z}_1^\star, \dots, \boldsymbol{z}_n^\star = \underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \ \mathcal{L}_{\text{Unif}}. \tag{57}$$

Then $\nu_{\{\boldsymbol{z}_1^\star,\dots,\boldsymbol{z}_n^\star\}}$ converge weak to σ_d as $n \to \infty$.*

Proof. We have

$$\underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \ \mathcal{L}_{\text{Unif}} \tag{58}$$

$$= \underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \ \log \sum_{l,m} \exp(\kappa \boldsymbol{z}_l^\top \boldsymbol{z}_m) \tag{59}$$

$$= \underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \ \sum_{l,m} \exp(\kappa \boldsymbol{z}_l^\top \boldsymbol{z}_m) \tag{60}$$

(monotonicity of logarithm)

$$= \underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \ \sum_{1 \le l < m \le n} \exp(\kappa \boldsymbol{z}_l^\top \boldsymbol{z}_m) \tag{61}$$

(symmetry of inner product)

$$= \underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \ \sum_{1 \le l < m \le n} \underbrace{\exp(-\kappa ||\boldsymbol{z}_l - \boldsymbol{z}_m||_2^2)}_{=: \, G(\boldsymbol{z}_l, \boldsymbol{z}_m)} \tag{62}$$

(multiplication by positive constant)

$$= \underset{\boldsymbol{z}_1,\dots,\boldsymbol{z}_n \in \mathbb{S}_d}{\arg\min} \ \sum_{1 \le l < m \le n} G(\boldsymbol{z}_l, \boldsymbol{z}_m) \tag{63}$$

The result then follows directly from [8, Proposition 2]. □

5. Experiments

5.1. Implementation details

This section covers the additional implementation details not provided in the main paper. These include the initialization of the embeddings in Algorithm 1, hyperparameters, additional transformations wherever required, the architectures used, and a note on accessing the code, datasets, and dataset splits.

Initialization and normalization. Instead of a random initialization of our embeddings Z_0, we follow a PCA based initialization, as in [5]. The weights are computed using the cached features from the base classes, the support and query features are then transformed using these weights. This procedure is also fast as we do not need to compute the PCA weights on every episode. To ensure that the resulting features lie on the hypersphere after each gradient update in noHub and noHub-S, we re-normalize the embeddings using L2 normalization.

Hyperparameters. noHub and noHub-S have the following hyperparameters.

- P – perplexity for computing the κ_i.

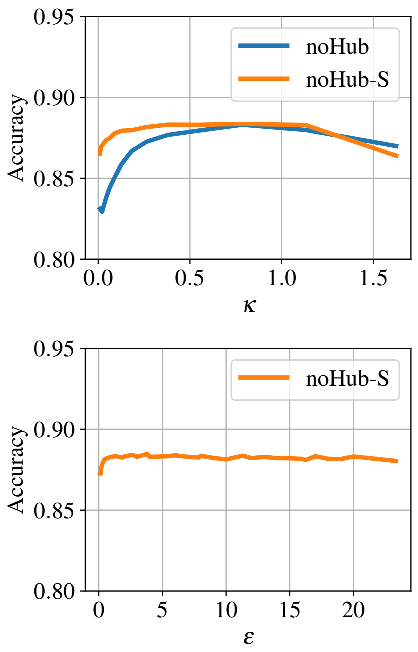


Figure 1. Accuracy for different values for κ and ε. Neither noHub nor noHub-S are particularly sensitive the the choice of these parameters.

- T – number of iterations.
- α – tradeoff parameter in the loss ($\mathcal{L}_{\text{noHub}} = \alpha \mathcal{L}_{\text{LSP}} + (1 - \alpha)\mathcal{L}_{\text{Unif}}$).
- η – learning rate for the Adam optimizer.
- κ – concentration parameter for the embeddings.
- ε – exaggeration of similarities between supports from different classes.
- d – dimensionality of embeddings.

All hyperparameter values used in in noHub and noHub-S are given in Table 1

Code. The code for our experiments is available at: `https://github.com/uitml/noHub`

Data splits. Details to access the datasets used with the requisite splits (both are consistent with [6]) are available in the code repository.

Base feature extractors.

- **Resnet-18**: As in [1, 6], we use the weights from [6]. The model is trained using a cross-entropy loss on the base classes.
- **WideRes28-10**: Following [3, 9], we use the weights from [3]. The model is pre-trained using a combination of cross-entropy and rotation prediction [2], and then fine-tuned with Manifold Mixup [7].

Arch.	Param.	Method	mini 1-shot	mini 5-shot	tiered 1-shot	tiered 5-shot	CUB 1-shot	CUB 5-shot
ResNet18	P	noHub	45	45	45	45	45	45
		noHub-S	45	45	40	45	45	45
	T	noHub	50	50	50	50	50	50
		noHub-S	150	150	150	150	150	150
	α	noHub	0.2	0.2	0.2	0.2	0.2	0.2
		noHub-S	0.3	0.2	0.2	0.2	0.3	0.2
	η	noHub	0.1	0.1	0.1	0.1	0.1	0.1
		noHub-S	0.1	0.1	0.1	0.1	0.1	0.1
	κ	noHub	0.5	0.5	0.5	0.5	0.5	0.5
		noHub-S	0.5	0.5	0.5	0.5	0.5	0.5
	ε	noHub	–	–	–	–	–	–
		noHub-S	8	8	5	8	8	8
	d	noHub	400	400	400	400	400	400
		noHub-S	400	400	400	400	400	400
WideRes28-10	P	noHub	45	45	45	45	45	45
		noHub-S	45	45	40	35	45	30
	T	noHub	50	50	50	50	50	50
		noHub-S	150	150	150	150	150	150
	α	noHub	0.2	0.2	0.2	0.2	0.2	0.2
		noHub-S	0.3	0.2	0.2	0.1	0.3	0.1
	η	noHub	0.1	0.1	0.1	0.1	0.1	0.1
		noHub-S	0.1	0.1	0.1	0.1	0.1	0.1
	κ	noHub	0.5	0.5	0.5	0.5	0.5	0.5
		noHub-S	0.5	0.5	0.5	0.2	0.5	0.2
	ε	noHub	–	–	–	–	–	–
		noHub-S	8	8	5	12	8	8
	d	noHub	400	400	400	400	400	400
		noHub-S	400	400	400	400	400	400

Table 1. Hyperparameter values used in our experiments.

5.2. Results

FSL performance. The complete lists of accuracies and hubness metrics for all embeddings, classifiers, and feature extractors, are given in Tables 2, 3, 4, and 5. The exhaustive results in these tables form the basis of Table 1, Table 2 and Table 3 in the main text. The two proposed approaches consistently outperform prior embeddings across several classifiers, feature extractors and datasets.

Effect of the κ and ε hyperparameters. The plots in Figure 1 show accuracy on *tiered* 5-shot with SIAMESE for increasing κ and ε. Neither method is particularly sensitive to the choice of κ and ε, and noHub-S is less sensitive to variations in κ, than noHub. Choosing $\kappa \in [0.5, 1]$ and $\varepsilon \in [3, 20]$ will result in high classification accuracy

6. Potential Negative Societal Impacts

As is the case with most methodological research in machine learning, the methods developed in this work could be used in downstream applications with potential negative societal impacts. Real world machine learning-based systems that interact with humans, or the environment in general, should therefore be properly tested and equipped with adequate safety measures.

Since our work relies on a large number of labeled examples from the base classes, un-discovered biases from the base dataset could be transferred to the trained models. Furthermore, the small number of examples in the inference stage could make the query predictions biased towards the included support examples, and not accurately reflect the diversity of the novel classes.

				mini			tiered			CUB	
Arch.	Clf.	Emb.	Acc	Skew	Hub. Occ.	Acc	Skew	Hub. Occ.	Acc	Skew	Hub. Occ.
ResNet18	ILPC	None	64.07 (0.28)	1.411 (0.01)	0.408 (0.001)	75.5 (0.28)	1.213 (0.009)	0.41 (0.001)	76.06 (0.27)	0.886 (0.006)	0.34 (0.001)
		L2	69.28 (0.27)	0.966 (0.007)	0.298 (0.001)	77.84 (0.28)	0.811 (0.007)	0.267 (0.001)	79.91 (0.26)	0.688 (0.006)	0.236 (0.001)
		CL2	71.48 (0.27)	0.661 (0.005)	0.229 (0.001)	79.8 (0.27)	0.679 (0.006)	0.249 (0.001)	80.97 (0.26)	0.553 (0.005)	0.203 (0.001)
		ZN	71.48 (0.27)	0.677 (0.006)	0.227 (0.001)	79.95 (0.27)	0.694 (0.006)	0.263 (0.001)	81.49 (0.25)	0.57 (0.005)	0.217 (0.001)
		ReRep	65.49 (0.28)	3.688 (0.007)	0.559 (0.001)	76.75 (0.28)	3.61 (0.01)	0.55 (0.001)	77.73 (0.26)	3.563 (0.007)	0.512 (0.001)
		EASE	71.79 (0.28)	0.515 (0.005)	0.157 (0.001)	80.2 (0.27)	0.48 (0.005)	0.158 (0.001)	81.88 (0.25)	0.463 (0.004)	0.153 (0.001)
		TCPR	71.77 (0.28)	0.647 (0.005)	0.223 (0.001)	80.01 (0.28)	0.652 (0.006)	0.249 (0.001)	81.75 (0.25)	0.534 (0.004)	0.203 (0.001)
		noHub	73.18 (0.28)	0.308 (0.005)	**0.094 (0.001)**	80.76 (0.28)	0.296 (0.004)	**0.101 (0.001)**	82.74 (0.26)	0.32 (0.004)	**0.112 (0.001)**
		noHub-S	**74.02 (0.28)**	**0.276 (0.004)**	0.13 (0.001)	**81.34 (0.27)**	**0.281 (0.004)**	0.127 (0.001)	**83.92 (0.25)**	**0.296 (0.003)**	0.163 (0.001)
	LaplacianShot	None	68.92 (0.23)	1.341 (0.009)	0.408 (0.001)	76.43 (0.25)	1.214 (0.009)	0.41 (0.001)	79.17 (0.23)	0.887 (0.006)	0.34 (0.001)
		L2	69.3 (0.23)	0.945 (0.007)	0.302 (0.001)	77.2 (0.25)	0.808 (0.007)	0.265 (0.001)	79.65 (0.23)	0.682 (0.006)	0.236 (0.001)
		CL2	70.68 (0.23)	0.661 (0.005)	0.231 (0.001)	77.98 (0.24)	0.689 (0.006)	0.248 (0.001)	79.99 (0.22)	0.547 (0.005)	0.201 (0.001)
		ZN	70.51 (0.23)	0.688 (0.006)	0.233 (0.001)	77.51 (0.24)	0.697 (0.006)	0.264 (0.001)	79.86 (0.22)	0.564 (0.005)	0.217 (0.001)
		ReRep	72.75 (0.24)	3.653 (0.007)	0.548 (0.001)	78.95 (0.25)	3.605 (0.011)	0.549 (0.001)	82.38 (0.22)	3.565 (0.007)	0.512 (0.001)
		EASE	72.19 (0.23)	0.526 (0.005)	0.161 (0.001)	79.34 (0.24)	0.481 (0.005)	0.158 (0.001)	81.5 (0.22)	0.459 (0.004)	0.152 (0.001)
		TCPR	71.79 (0.23)	0.654 (0.005)	0.228 (0.001)	78.41 (0.24)	0.651 (0.005)	0.249 (0.001)	80.86 (0.22)	0.537 (0.004)	0.203 (0.001)
		noHub	73.63 (0.25)	0.305 (0.005)	**0.094 (0.001)**	**80.84 (0.25)**	0.3 (0.005)	**0.101 (0.001)**	83.23 (0.22)	0.318 (0.004)	**0.112 (0.001)**
		noHub-S	**73.79 (0.25)**	**0.276 (0.004)**	0.13 (0.001)	80.83 (0.25)	**0.275 (0.004)**	0.125 (0.001)	**83.47 (0.22)**	**0.299 (0.003)**	0.164 (0.001)
	ObliqueManifold	None	68.89 (0.23)	1.412 (0.01)	0.407 (0.001)	77.07 (0.25)	1.21 (0.009)	0.409 (0.001)	79.4 (0.22)	0.887 (0.006)	0.341 (0.001)
		L2	68.92 (0.23)	0.964 (0.007)	0.299 (0.001)	77.17 (0.25)	0.806 (0.007)	0.266 (0.001)	79.32 (0.22)	0.691 (0.006)	0.237 (0.001)
		CL2	70.86 (0.24)	0.66 (0.005)	0.228 (0.001)	78.92 (0.25)	0.68 (0.006)	0.249 (0.001)	80.29 (0.23)	0.547 (0.005)	0.202 (0.001)
		ZN	71.25 (0.24)	0.679 (0.006)	0.227 (0.001)	79.54 (0.25)	0.697 (0.006)	0.263 (0.001)	81.38 (0.23)	0.562 (0.005)	0.216 (0.001)
		ReRep	73.3 (0.25)	3.682 (0.007)	0.559 (0.001)	80.26 (0.26)	3.608 (0.01)	0.551 (0.001)	**83.84 (0.23)**	3.559 (0.008)	0.513 (0.001)
		EASE	68.4 (0.24)	0.516 (0.005)	0.156 (0.001)	77.33 (0.25)	0.477 (0.004)	0.158 (0.001)	79.03 (0.24)	0.461 (0.004)	0.152 (0.001)
		TCPR	70.74 (0.24)	0.646 (0.005)	0.223 (0.001)	78.92 (0.25)	0.649 (0.005)	0.249 (0.001)	80.18 (0.23)	0.537 (0.004)	0.204 (0.001)
		noHub	72.55 (0.26)	0.309 (0.005)	**0.095 (0.001)**	79.97 (0.26)	0.302 (0.005)	**0.102 (0.001)**	82.21 (0.24)	0.319 (0.004)	**0.112 (0.001)**
		noHub-S	**74.24 (0.26)**	**0.274 (0.004)**	0.13 (0.001)	**80.84 (0.26)**	**0.282 (0.004)**	0.127 (0.001)	83.67 (0.23)	**0.294 (0.003)**	0.162 (0.001)
	SIAMESE	None	20.0 (0.0)	1.345 (0.009)	0.407 (0.001)	20.0 (0.0)	1.222 (0.009)	0.41 (0.001)	20.0 (0.0)	0.885 (0.006)	0.339 (0.001)
		L2	73.77 (0.24)	0.949 (0.007)	0.301 (0.001)	80.46 (0.26)	0.811 (0.007)	0.265 (0.001)	83.1 (0.23)	0.691 (0.006)	0.237 (0.001)
		CL2	75.56 (0.26)	0.666 (0.005)	0.232 (0.001)	82.1 (0.26)	0.68 (0.006)	0.248 (0.001)	84.35 (0.24)	0.549 (0.005)	0.201 (0.001)
		ZN	20.0 (0.0)	0.686 (0.006)	0.232 (0.001)	20.0 (0.0)	0.69 (0.006)	0.262 (0.001)	20.0 (0.0)	0.565 (0.005)	0.217 (0.001)
		ReRep	20.0 (0.0)	3.653 (0.007)	0.549 (0.001)	20.0 (0.0)	3.616 (0.01)	0.549 (0.001)	20.0 (0.0)	3.559 (0.007)	0.512 (0.001)
		EASE	76.05 (0.27)	0.529 (0.005)	0.162 (0.001)	82.57 (0.27)	0.485 (0.005)	0.159 (0.001)	85.24 (0.24)	0.464 (0.004)	0.153 (0.001)
		TCPR	75.99 (0.26)	0.655 (0.005)	0.227 (0.001)	82.65 (0.26)	0.651 (0.005)	0.249 (0.001)	85.34 (0.23)	0.535 (0.004)	0.203 (0.001)
		noHub	76.65 (0.28)	0.308 (0.005)	**0.095 (0.001)**	82.94 (0.27)	0.303 (0.004)	**0.101 (0.001)**	**85.88 (0.24)**	0.322 (0.004)	**0.112 (0.001)**
		noHub-S	**76.68 (0.28)**	**0.275 (0.004)**	0.13 (0.001)	**83.09 (0.27)**	**0.281 (0.004)**	0.128 (0.001)	85.81 (0.24)	**0.295 (0.003)**	0.161 (0.001)
	SimpleShot	None	56.14 (0.2)	1.349 (0.009)	0.407 (0.001)	63.34 (0.23)	1.211 (0.009)	0.408 (0.001)	64.02 (0.21)	0.887 (0.006)	0.341 (0.001)
		L2	60.15 (0.2)	0.937 (0.007)	0.301 (0.001)	68.02 (0.23)	0.812 (0.007)	0.265 (0.001)	69.05 (0.21)	0.691 (0.006)	0.236 (0.001)
		CL2	63.1 (0.2)	0.667 (0.005)	0.233 (0.001)	69.76 (0.22)	0.679 (0.006)	0.249 (0.001)	70.16 (0.2)	0.549 (0.005)	0.201 (0.001)
		ZN	63.39 (0.2)	0.68 (0.005)	0.231 (0.001)	70.04 (0.22)	0.698 (0.006)	0.264 (0.001)	71.03 (0.2)	0.564 (0.005)	0.216 (0.001)
		ReRep	66.66 (0.22)	3.655 (0.007)	0.548 (0.001)	73.23 (0.23)	3.604 (0.01)	0.549 (0.001)	76.8 (0.21)	3.565 (0.007)	0.513 (0.001)
		EASE	64.0 (0.2)	0.521 (0.005)	0.16 (0.001)	71.0 (0.21)	0.479 (0.005)	0.158 (0.001)	72.38 (0.2)	0.466 (0.004)	0.153 (0.001)
		TCPR	63.33 (0.2)	0.651 (0.005)	0.228 (0.001)	69.82 (0.22)	0.65 (0.005)	0.25 (0.001)	70.75 (0.2)	0.532 (0.004)	0.204 (0.001)
		noHub	69.38 (0.22)	0.315 (0.005)	**0.095 (0.001)**	76.72 (0.23)	0.303 (0.004)	**0.102 (0.001)**	78.21 (0.21)	0.32 (0.004)	**0.112 (0.001)**
		noHub-S	**71.1 (0.22)**	**0.276 (0.004)**	0.13 (0.001)	**78.35 (0.23)**	**0.283 (0.004)**	0.127 (0.001)	**80.31 (0.21)**	**0.296 (0.003)**	0.162 (0.001)
	α-TIM	None	56.39 (0.2)	1.342 (0.009)	0.406 (0.001)	63.32 (0.23)	1.216 (0.009)	0.411 (0.001)	64.02 (0.22)	0.886 (0.006)	0.341 (0.001)
		L2	67.91 (0.23)	0.942 (0.007)	0.301 (0.001)	74.94 (0.24)	0.814 (0.007)	0.266 (0.001)	77.49 (0.23)	0.694 (0.006)	0.236 (0.001)
		CL2	65.68 (0.21)	0.665 (0.005)	0.232 (0.001)	73.23 (0.23)	0.681 (0.006)	0.248 (0.001)	73.79 (0.21)	0.552 (0.005)	0.202 (0.001)
		ZN	63.36 (0.2)	0.682 (0.005)	0.232 (0.001)	70.19 (0.22)	0.693 (0.006)	0.263 (0.001)	70.85 (0.21)	0.566 (0.005)	0.215 (0.001)
		ReRep	66.37 (0.22)	3.656 (0.007)	0.55 (0.001)	73.24 (0.24)	3.605 (0.011)	0.55 (0.001)	76.86 (0.21)	3.555 (0.007)	0.514 (0.001)
		EASE	65.32 (0.2)	0.526 (0.005)	0.163 (0.001)	71.88 (0.22)	0.477 (0.005)	0.158 (0.001)	73.03 (0.21)	0.459 (0.004)	0.151 (0.001)
		TCPR	66.19 (0.21)	0.65 (0.005)	0.227 (0.001)	73.24 (0.23)	0.649 (0.005)	0.25 (0.001)	74.07 (0.21)	0.532 (0.004)	0.203 (0.001)
		noHub	70.08 (0.23)	0.312 (0.005)	**0.094 (0.001)**	77.39 (0.24)	0.304 (0.004)	**0.101 (0.001)**	79.19 (0.22)	0.319 (0.004)	**0.112 (0.001)**
		noHub-S	**72.04 (0.23)**	**0.273 (0.004)**	0.13 (0.001)	**79.13 (0.24)**	**0.282 (0.004)**	0.126 (0.001)	**81.42 (0.22)**	**0.296 (0.003)**	0.161 (0.001)

Table 2. Resnet-18: 1-shot.

			mini			tiered			CUB		
Arch.	Clf.	Emb.	Acc	Skew	Hub. Occ.	Acc	Skew	Hub. Occ.	Acc	Skew	Hub. Occ.
WideRes28-10	ILPC	None	71.27 (0.28)	1.595 (0.01)	0.46 (0.001)	75.01 (0.28)	1.807 (0.01)	0.494 (0.001)	89.75 (0.19)	1.072 (0.009)	0.367 (0.001)
		L2	76.41 (0.26)	0.773 (0.006)	0.295 (0.001)	78.25 (0.27)	0.731 (0.006)	0.274 (0.001)	90.27 (0.2)	0.473 (0.004)	0.228 (0.001)
		CL2	74.13 (0.27)	0.993 (0.009)	0.29 (0.001)	78.2 (0.27)	0.815 (0.006)	0.306 (0.001)	90.34 (0.2)	0.524 (0.004)	0.267 (0.001)
		ZN	77.76 (0.26)	0.728 (0.005)	0.287 (0.001)	79.42 (0.27)	0.776 (0.006)	0.302 (0.001)	90.21 (0.2)	0.516 (0.004)	0.263 (0.001)
		ReRep	62.51 (0.34)	3.56 (0.002)	0.704 (0.001)	60.66 (0.37)	3.55 (0.002)	0.776 (0.001)	87.44 (0.25)	3.033 (0.008)	0.472 (0.001)
		EASE	78.01 (0.26)	0.47 (0.004)	0.176 (0.001)	79.64 (0.27)	0.479 (0.004)	0.175 (0.001)	90.76 (0.19)	0.437 (0.003)	0.212 (0.001)
		TCPR	78.37 (0.26)	0.584 (0.005)	0.237 (0.001)	79.55 (0.28)	0.683 (0.006)	0.265 (0.001)	90.77 (0.19)	0.476 (0.004)	0.23 (0.001)
		noHub	78.84 (0.27)	0.293 (0.004)	**0.112 (0.001)**	80.75 (0.28)	0.3 (0.004)	**0.112 (0.001)**	90.91 (0.2)	0.189 (0.004)	**0.109 (0.001)**
		noHub-S	**79.77 (0.26)**	**0.262 (0.004)**	0.148 (0.001)	**81.24 (0.27)**	**0.278 (0.004)**	0.135 (0.001)	**91.28 (0.19)**	**0.16 (0.004)**	0.13 (0.001)
	LaplacianShot	None	72.56 (0.23)	1.599 (0.01)	0.459 (0.001)	75.58 (0.25)	1.795 (0.01)	0.495 (0.001)	88.71 (0.19)	1.071 (0.009)	0.369 (0.001)
		L2	75.18 (0.23)	0.777 (0.006)	0.296 (0.001)	77.03 (0.24)	0.732 (0.006)	0.274 (0.001)	89.73 (0.17)	0.474 (0.004)	0.229 (0.001)
		CL2	71.29 (0.24)	0.987 (0.009)	0.29 (0.001)	75.42 (0.25)	0.819 (0.006)	0.309 (0.001)	89.61 (0.18)	0.52 (0.004)	0.268 (0.001)
		ZN	75.18 (0.22)	0.724 (0.005)	0.286 (0.001)	77.0 (0.24)	0.768 (0.006)	0.301 (0.001)	89.22 (0.18)	0.517 (0.004)	0.263 (0.001)
		ReRep	75.25 (0.22)	3.562 (0.002)	0.704 (0.001)	77.12 (0.24)	3.548 (0.002)	0.776 (0.001)	88.98 (0.18)	3.024 (0.008)	0.47 (0.001)
		EASE	77.29 (0.22)	0.473 (0.004)	0.177 (0.001)	78.97 (0.24)	0.475 (0.004)	0.175 (0.001)	90.06 (0.17)	0.435 (0.003)	0.213 (0.001)
		TCPR	76.77 (0.22)	0.593 (0.005)	0.236 (0.001)	77.49 (0.24)	0.686 (0.006)	0.264 (0.001)	89.42 (0.17)	0.475 (0.004)	0.231 (0.001)
		noHub	**79.13 (0.23)**	0.29 (0.004)	**0.111 (0.001)**	80.5 (0.25)	0.302 (0.004)	**0.112 (0.001)**	**90.73 (0.18)**	0.19 (0.004)	**0.109 (0.001)**
		noHub-S	79.13 (0.23)	**0.259 (0.004)**	0.147 (0.001)	**80.59 (0.24)**	**0.277 (0.004)**	0.135 (0.001)	90.61 (0.17)	**0.164 (0.004)**	0.13 (0.001)
	ObliqueManifold	None	76.02 (0.22)	1.599 (0.01)	0.46 (0.001)	77.75 (0.25)	1.801 (0.01)	0.494 (0.001)	90.82 (0.18)	1.07 (0.009)	0.368 (0.001)
		L2	76.11 (0.22)	0.779 (0.006)	0.295 (0.001)	77.74 (0.25)	0.731 (0.006)	0.274 (0.001)	90.89 (0.18)	0.475 (0.004)	0.228 (0.001)
		CL2	74.43 (0.24)	0.985 (0.009)	0.289 (0.001)	77.98 (0.25)	0.816 (0.007)	0.307 (0.001)	90.6 (0.18)	0.523 (0.004)	0.267 (0.001)
		ZN	77.69 (0.23)	0.724 (0.005)	0.286 (0.001)	79.32 (0.24)	0.767 (0.006)	0.301 (0.001)	90.73 (0.18)	0.519 (0.004)	0.263 (0.001)
		ReRep	78.08 (0.23)	3.56 (0.002)	0.703 (0.001)	79.46 (0.25)	3.549 (0.002)	0.777 (0.001)	91.16 (0.18)	3.032 (0.008)	0.471 (0.001)
		EASE	74.77 (0.23)	0.472 (0.004)	0.178 (0.001)	77.07 (0.25)	0.473 (0.004)	0.174 (0.001)	89.2 (0.18)	0.439 (0.003)	0.212 (0.001)
		TCPR	77.39 (0.23)	0.587 (0.005)	0.236 (0.001)	78.75 (0.24)	0.687 (0.006)	0.265 (0.001)	89.93 (0.19)	0.474 (0.004)	0.23 (0.001)
		noHub	78.44 (0.24)	0.292 (0.004)	**0.112 (0.001)**	79.99 (0.26)	0.302 (0.004)	**0.113 (0.001)**	90.59 (0.19)	0.185 (0.004)	**0.108 (0.001)**
		noHub-S	**79.89 (0.24)**	**0.259 (0.004)**	0.148 (0.001)	**80.67 (0.26)**	**0.279 (0.004)**	0.137 (0.001)	**91.37 (0.18)**	**0.162 (0.004)**	0.13 (0.001)
	SIAMESE	None	45.69 (0.31)	1.594 (0.009)	0.459 (0.001)	75.29 (0.28)	1.801 (0.01)	0.495 (0.001)	61.36 (0.55)	1.074 (0.009)	0.37 (0.001)
		L2	80.2 (0.23)	0.776 (0.006)	0.296 (0.001)	80.89 (0.26)	0.735 (0.006)	0.275 (0.001)	91.98 (0.18)	0.476 (0.004)	0.23 (0.001)
		CL2	75.23 (0.27)	0.988 (0.009)	0.289 (0.001)	79.59 (0.27)	0.82 (0.006)	0.307 (0.001)	92.17 (0.18)	0.518 (0.004)	0.266 (0.001)
		ZN	20.0 (0.0)	0.726 (0.005)	0.286 (0.001)	20.0 (0.0)	0.775 (0.006)	0.302 (0.001)	20.0 (0.0)	0.517 (0.004)	0.264 (0.001)
		ReRep	36.69 (0.28)	3.561 (0.002)	0.705 (0.001)	67.41 (0.29)	3.55 (0.002)	0.776 (0.001)	57.62 (0.56)	3.027 (0.008)	0.472 (0.001)
		EASE	81.19 (0.25)	0.474 (0.004)	0.178 (0.001)	82.04 (0.26)	0.476 (0.004)	0.176 (0.001)	91.99 (0.19)	0.436 (0.003)	0.213 (0.001)
		TCPR	81.27 (0.24)	0.582 (0.005)	0.236 (0.001)	81.89 (0.26)	0.681 (0.006)	0.264 (0.001)	91.91 (0.18)	0.477 (0.004)	0.232 (0.001)
		noHub	81.97 (0.25)	0.291 (0.004)	**0.111 (0.001)**	82.8 (0.27)	0.298 (0.004)	**0.112 (0.001)**	92.53 (0.18)	0.189 (0.004)	**0.109 (0.001)**
		noHub-S	**82.0 (0.26)**	**0.258 (0.004)**	0.148 (0.001)	**82.85 (0.27)**	**0.278 (0.004)**	0.137 (0.001)	**92.63 (0.18)**	**0.159 (0.004)**	0.13 (0.001)
	SimpleShot	None	55.66 (0.21)	1.6 (0.01)	0.459 (0.001)	54.71 (0.22)	1.81 (0.01)	0.494 (0.001)	70.92 (0.23)	1.073 (0.009)	0.369 (0.001)
		L2	65.78 (0.2)	0.781 (0.006)	0.296 (0.001)	68.75 (0.22)	0.737 (0.006)	0.275 (0.001)	82.85 (0.19)	0.475 (0.004)	0.228 (0.001)
		CL2	64.33 (0.2)	0.981 (0.009)	0.288 (0.001)	67.66 (0.22)	0.817 (0.006)	0.307 (0.001)	82.8 (0.19)	0.52 (0.004)	0.267 (0.001)
		ZN	67.31 (0.2)	0.73 (0.005)	0.287 (0.001)	69.14 (0.22)	0.769 (0.006)	0.302 (0.001)	82.79 (0.19)	0.517 (0.004)	0.263 (0.001)
		ReRep	67.38 (0.2)	3.56 (0.002)	0.704 (0.001)	70.17 (0.22)	3.55 (0.002)	0.777 (0.001)	84.86 (0.19)	3.026 (0.008)	0.47 (0.001)
		EASE	68.62 (0.2)	0.47 (0.004)	0.177 (0.001)	70.26 (0.21)	0.477 (0.004)	0.175 (0.001)	84.14 (0.18)	0.437 (0.003)	0.213 (0.001)
		TCPR	68.45 (0.2)	0.589 (0.005)	0.236 (0.001)	68.68 (0.22)	0.685 (0.006)	0.264 (0.001)	82.28 (0.19)	0.477 (0.004)	0.231 (0.001)
		noHub	75.06 (0.21)	0.29 (0.004)	**0.111 (0.001)**	76.7 (0.23)	0.301 (0.004)	**0.111 (0.001)**	88.06 (0.18)	0.188 (0.004)	**0.108 (0.001)**
		noHub-S	**76.86 (0.21)**	**0.258 (0.004)**	0.148 (0.001)	**78.4 (0.23)**	**0.274 (0.004)**	0.135 (0.001)	**89.25 (0.18)**	**0.162 (0.004)**	0.13 (0.001)
	α-TIM	None	60.31 (0.2)	1.603 (0.01)	0.458 (0.001)	69.42 (0.25)	1.811 (0.01)	0.494 (0.001)	73.83 (0.21)	1.072 (0.009)	0.369 (0.001)
		L2	72.11 (0.22)	0.778 (0.006)	0.295 (0.001)	74.45 (0.23)	0.73 (0.006)	0.275 (0.001)	85.96 (0.19)	0.476 (0.004)	0.229 (0.001)
		CL2	68.5 (0.21)	0.988 (0.009)	0.29 (0.001)	72.17 (0.23)	0.811 (0.006)	0.306 (0.001)	85.6 (0.18)	0.522 (0.004)	0.267 (0.001)
		ZN	67.69 (0.2)	0.73 (0.005)	0.287 (0.001)	68.94 (0.22)	0.769 (0.006)	0.302 (0.001)	83.03 (0.19)	0.518 (0.004)	0.263 (0.001)
		ReRep	73.15 (0.23)	3.56 (0.002)	0.704 (0.001)	76.19 (0.25)	3.551 (0.002)	0.778 (0.001)	88.55 (0.18)	3.027 (0.008)	0.472 (0.001)
		EASE	69.83 (0.2)	0.468 (0.004)	0.176 (0.001)	71.54 (0.22)	0.481 (0.004)	0.175 (0.001)	84.9 (0.19)	0.436 (0.003)	0.213 (0.001)
		TCPR	71.6 (0.21)	0.586 (0.005)	0.237 (0.001)	72.71 (0.22)	0.689 (0.006)	0.264 (0.001)	84.99 (0.19)	0.479 (0.004)	0.231 (0.001)
		noHub	75.87 (0.22)	0.29 (0.004)	**0.111 (0.001)**	77.83 (0.23)	0.302 (0.004)	**0.112 (0.001)**	88.7 (0.17)	0.189 (0.004)	**0.108 (0.001)**
		noHub-S	**77.76 (0.22)**	**0.259 (0.004)**	0.147 (0.001)	**79.04 (0.24)**	**0.276 (0.004)**	0.136 (0.001)	**89.77 (0.17)**	**0.163 (0.003)**	0.13 (0.001)

Table 3. WideRes28-10: 1-shot.

				mini			tiered			CUB	
Arch.	Clf.	Emb.	Acc	Skew	Hub. Occ.	Acc	Skew	Hub. Occ.	Acc	Skew	Hub. Occ.
ResNet18	ILPC	None	76.46 (0.18)	1.503 (0.01)	0.421 (0.001)	84.46 (0.18)	1.334 (0.008)	0.433 (0.001)	85.86 (0.14)	0.981 (0.005)	0.364 (0.001)
		L2	80.9 (0.16)	1.051 (0.007)	0.314 (0.001)	86.23 (0.17)	0.912 (0.006)	0.289 (0.001)	88.03 (0.13)	0.808 (0.005)	0.264 (0.001)
		CL2	81.64 (0.16)	0.778 (0.005)	0.262 (0.001)	86.88 (0.17)	0.823 (0.006)	0.281 (0.001)	88.44 (0.13)	0.695 (0.005)	0.235 (0.001)
		ZN	81.61 (0.16)	0.793 (0.005)	0.258 (0.001)	86.9 (0.17)	0.841 (0.006)	0.297 (0.001)	88.44 (0.12)	0.717 (0.004)	0.25 (0.001)
		ReRep	74.83 (0.19)	1.623 (0.003)	0.871 (0.001)	83.96 (0.19)	1.722 (0.004)	0.873 (0.001)	84.54 (0.15)	1.432 (0.003)	0.869 (0.001)
		EASE	81.75 (0.16)	0.618 (0.005)	0.182 (0.001)	86.84 (0.17)	0.593 (0.004)	0.181 (0.001)	88.85 (0.12)	0.606 (0.004)	0.186 (0.001)
		TCPR	81.76 (0.16)	0.766 (0.005)	0.254 (0.001)	86.78 (0.17)	0.801 (0.005)	0.284 (0.001)	88.69 (0.13)	0.683 (0.004)	0.237 (0.001)
		noHub	82.09 (0.16)	**0.295 (0.004)**	0.097 (0.001)	86.81 (0.17)	**0.289 (0.004)**	0.102 (0.001)	88.85 (0.13)	**0.333 (0.004)**	0.12 (0.001)
		noHub-S	**82.33 (0.16)**	0.488 (0.006)	**0.086 (0.001)**	**87.05 (0.17)**	0.475 (0.006)	**0.091 (0.001)**	**89.12 (0.13)**	0.438 (0.006)	**0.097 (0.001)**
	LaplacianShot	None	81.97 (0.15)	1.442 (0.009)	0.422 (0.001)	86.17 (0.16)	1.336 (0.008)	0.432 (0.001)	88.58 (0.12)	0.985 (0.005)	0.365 (0.001)
		L2	81.89 (0.14)	1.035 (0.007)	0.319 (0.001)	86.19 (0.16)	0.913 (0.006)	0.289 (0.001)	88.52 (0.11)	0.811 (0.005)	0.264 (0.001)
		CL2	81.93 (0.14)	0.786 (0.005)	0.265 (0.001)	86.16 (0.16)	0.82 (0.006)	0.282 (0.001)	88.46 (0.12)	0.7 (0.005)	0.235 (0.001)
		ZN	82.57 (0.14)	0.803 (0.005)	0.263 (0.001)	86.67 (0.16)	0.838 (0.006)	0.296 (0.001)	88.88 (0.11)	0.714 (0.004)	0.25 (0.001)
		ReRep	82.32 (0.14)	1.633 (0.003)	0.863 (0.001)	86.09 (0.16)	1.721 (0.004)	0.873 (0.001)	88.74 (0.12)	1.431 (0.002)	0.869 (0.001)
		EASE	82.57 (0.14)	0.627 (0.005)	0.186 (0.001)	86.82 (0.15)	0.596 (0.004)	0.182 (0.001)	88.94 (0.11)	0.608 (0.004)	0.185 (0.001)
		TCPR	82.24 (0.14)	0.781 (0.005)	0.259 (0.001)	86.27 (0.16)	0.797 (0.005)	0.284 (0.001)	88.63 (0.11)	0.687 (0.004)	0.236 (0.001)
		noHub	82.55 (0.15)	0.285 (0.004)	0.096 (0.001)	86.75 (0.16)	0.29 (0.004)	0.103 (0.001)	**89.08 (0.11)**	**0.329 (0.004)**	0.12 (0.001)
		noHub-S	**82.81 (0.14)**	**0.25 (0.005)**	**0.073 (0.001)**	**87.12 (0.16)**	**0.214 (0.005)**	**0.077 (0.001)**	88.99 (0.11)	0.438 (0.006)	**0.096 (0.001)**
	ObliqueManifold	None	83.53 (0.15)	1.497 (0.01)	0.421 (0.001)	87.85 (0.15)	1.334 (0.009)	0.433 (0.001)	90.28 (0.11)	0.987 (0.005)	0.364 (0.001)
		L2	83.66 (0.15)	1.051 (0.007)	0.314 (0.001)	87.83 (0.15)	0.922 (0.006)	0.289 (0.001)	90.21 (0.11)	0.81 (0.005)	0.263 (0.001)
		CL2	83.62 (0.15)	0.775 (0.005)	0.261 (0.001)	88.1 (0.15)	0.823 (0.006)	0.281 (0.001)	90.09 (0.11)	0.701 (0.005)	0.236 (0.001)
		ZN	**83.86 (0.15)**	0.795 (0.005)	0.258 (0.001)	**88.47 (0.15)**	0.835 (0.006)	0.296 (0.001)	**90.47 (0.11)**	0.716 (0.004)	0.251 (0.001)
		ReRep	82.44 (0.15)	1.62 (0.003)	0.871 (0.001)	86.85 (0.16)	1.725 (0.004)	0.872 (0.001)	89.83 (0.11)	1.431 (0.003)	0.869 (0.001)
		EASE	82.83 (0.15)	0.628 (0.005)	0.185 (0.001)	87.63 (0.16)	0.597 (0.005)	0.182 (0.001)	89.74 (0.12)	0.609 (0.004)	0.186 (0.001)
		TCPR	83.51 (0.15)	0.766 (0.005)	0.255 (0.001)	88.09 (0.15)	0.795 (0.005)	0.283 (0.001)	90.28 (0.11)	0.687 (0.004)	0.236 (0.001)
		noHub	83.28 (0.15)	**0.287 (0.004)**	0.096 (0.001)	87.58 (0.16)	**0.288 (0.004)**	0.102 (0.001)	89.89 (0.12)	**0.334 (0.004)**	0.121 (0.001)
		noHub-S	83.25 (0.16)	0.487 (0.006)	**0.086 (0.001)**	87.82 (0.16)	0.469 (0.006)	**0.091 (0.001)**	89.38 (0.17)	nan (nan)	**0.097 (0.001)**
	SIAMESE	None	20.0 (0.0)	1.441 (0.009)	0.421 (0.001)	20.0 (0.0)	1.339 (0.009)	0.433 (0.001)	20.0 (0.0)	0.984 (0.005)	0.364 (0.001)
		L2	83.14 (0.14)	1.035 (0.007)	0.319 (0.001)	87.04 (0.16)	0.912 (0.006)	0.288 (0.001)	89.48 (0.12)	0.808 (0.005)	0.264 (0.001)
		CL2	84.04 (0.15)	0.788 (0.005)	0.264 (0.001)	87.9 (0.16)	0.816 (0.006)	0.28 (0.001)	90.14 (0.12)	0.698 (0.005)	0.235 (0.001)
		ZN	20.0 (0.0)	0.8 (0.005)	0.263 (0.001)	20.0 (0.0)	0.84 (0.006)	0.296 (0.001)	20.0 (0.0)	0.713 (0.004)	0.251 (0.001)
		ReRep	20.0 (0.0)	1.633 (0.003)	0.863 (0.001)	20.0 (0.0)	1.724 (0.004)	0.872 (0.001)	20.0 (0.0)	1.428 (0.002)	0.869 (0.001)
		EASE	84.61 (0.15)	0.63 (0.005)	0.187 (0.001)	88.33 (0.16)	0.594 (0.004)	0.182 (0.001)	90.42 (0.12)	0.607 (0.004)	0.185 (0.001)
		TCPR	84.39 (0.15)	0.772 (0.005)	0.259 (0.001)	88.26 (0.16)	0.791 (0.005)	0.283 (0.001)	90.5 (0.11)	0.686 (0.004)	0.235 (0.001)
		noHub	84.05 (0.16)	0.292 (0.004)	0.096 (0.001)	87.87 (0.17)	**0.291 (0.004)**	0.103 (0.001)	90.34 (0.12)	**0.334 (0.004)**	0.12 (0.001)
		noHub-S	**84.67 (0.15)**	**0.247 (0.005)**	**0.074 (0.001)**	**88.43 (0.16)**	0.473 (0.006)	**0.092 (0.001)**	**90.52 (0.12)**	0.443 (0.006)	**0.097 (0.001)**
	SimpleShot	None	78.5 (0.14)	1.436 (0.009)	0.422 (0.001)	83.95 (0.16)	1.339 (0.008)	0.432 (0.001)	85.65 (0.12)	0.987 (0.005)	0.364 (0.001)
		L2	79.89 (0.14)	1.04 (0.007)	0.318 (0.001)	84.5 (0.16)	0.914 (0.006)	0.287 (0.001)	86.46 (0.12)	0.812 (0.005)	0.263 (0.001)
		CL2	80.0 (0.14)	0.786 (0.005)	0.264 (0.001)	84.66 (0.16)	0.821 (0.006)	0.28 (0.001)	86.3 (0.12)	0.698 (0.005)	0.236 (0.001)
		ZN	80.57 (0.14)	0.806 (0.005)	0.264 (0.001)	84.97 (0.16)	0.839 (0.006)	0.296 (0.001)	86.76 (0.12)	0.716 (0.005)	0.25 (0.001)
		ReRep	80.86 (0.14)	1.631 (0.003)	0.863 (0.001)	85.05 (0.16)	1.721 (0.004)	0.872 (0.001)	87.83 (0.12)	1.432 (0.002)	0.869 (0.001)
		EASE	80.13 (0.14)	0.624 (0.005)	0.186 (0.001)	84.74 (0.16)	0.598 (0.004)	0.183 (0.001)	86.76 (0.12)	0.607 (0.004)	0.186 (0.001)
		TCPR	80.15 (0.14)	0.78 (0.005)	0.259 (0.001)	84.86 (0.15)	0.796 (0.005)	0.283 (0.001)	86.8 (0.12)	0.687 (0.004)	0.235 (0.001)
		noHub	**82.13 (0.14)**	0.286 (0.004)	0.096 (0.001)	**86.31 (0.16)**	0.289 (0.004)	0.104 (0.001)	**88.46 (0.11)**	**0.329 (0.004)**	0.12 (0.001)
		noHub-S	81.22 (0.14)	**0.25 (0.005)**	**0.074 (0.001)**	86.22 (0.15)	**0.213 (0.005)**	**0.078 (0.001)**	87.6 (0.12)	0.433 (0.006)	**0.097 (0.001)**
	α-TIM	None	78.51 (0.15)	1.45 (0.009)	0.42 (0.001)	83.86 (0.16)	1.341 (0.009)	0.433 (0.001)	85.7 (0.12)	0.981 (0.005)	0.363 (0.001)
		L2	80.02 (0.16)	1.036 (0.007)	0.318 (0.001)	84.49 (0.18)	0.92 (0.006)	0.288 (0.001)	87.88 (0.13)	0.812 (0.005)	0.264 (0.001)
		CL2	80.46 (0.16)	0.784 (0.005)	0.264 (0.001)	84.86 (0.17)	0.82 (0.006)	0.281 (0.001)	87.53 (0.13)	0.701 (0.005)	0.235 (0.001)
		ZN	80.32 (0.14)	0.802 (0.005)	0.263 (0.001)	84.93 (0.16)	0.834 (0.006)	0.295 (0.001)	86.95 (0.12)	0.715 (0.004)	0.25 (0.001)
		ReRep	81.05 (0.14)	1.63 (0.003)	0.863 (0.001)	85.18 (0.16)	1.718 (0.004)	0.872 (0.001)	87.63 (0.12)	1.43 (0.002)	0.87 (0.001)
		EASE	79.13 (0.15)	0.632 (0.005)	0.188 (0.001)	84.04 (0.17)	0.596 (0.004)	0.181 (0.001)	86.7 (0.13)	0.607 (0.004)	0.186 (0.001)
		TCPR	80.52 (0.16)	0.776 (0.005)	0.259 (0.001)	85.01 (0.17)	0.796 (0.005)	0.283 (0.001)	87.81 (0.13)	0.681 (0.004)	0.234 (0.001)
		noHub	**81.39 (0.15)**	0.29 (0.004)	0.096 (0.001)	86.09 (0.16)	0.292 (0.004)	0.103 (0.001)	**88.16 (0.12)**	**0.336 (0.004)**	0.121 (0.001)
		noHub-S	81.37 (0.15)	**0.253 (0.005)**	**0.074 (0.001)**	**86.14 (0.16)**	**0.219 (0.005)**	**0.078 (0.001)**	87.97 (0.12)	0.437 (0.006)	**0.096 (0.001)**

Table 4. Resnet-18: 5-shot.

			mini			tiered			CUB		
			Acc	Skew	Hub. Occ.	Acc	Skew	Hub. Occ.	Acc	Skew	Hub. Occ.
Arch.	Clf.	Emb.									
WideRes28-10	ILPC	None	81.93 (0.16)	1.717 (0.01)	0.473 (0.001)	84.34 (0.17)	1.927 (0.011)	0.509 (0.001)	93.18 (0.11)	1.164 (0.008)	0.396 (0.001)
		L2	85.74 (0.14)	0.888 (0.005)	0.322 (0.001)	86.26 (0.17)	0.859 (0.005)	0.306 (0.001)	93.77 (0.1)	0.636 (0.004)	0.266 (0.001)
		CL2	83.33 (0.16)	1.12 (0.009)	0.318 (0.001)	85.99 (0.17)	0.957 (0.006)	0.338 (0.001)	93.79 (0.1)	0.703 (0.004)	0.309 (0.001)
		ZN	85.96 (0.14)	0.858 (0.005)	0.32 (0.001)	86.77 (0.16)	0.909 (0.006)	0.335 (0.001)	93.73 (0.1)	0.696 (0.004)	0.305 (0.001)
		ReRep	72.11 (0.27)	1.601 (0.003)	0.819 (0.001)	71.68 (0.3)	1.616 (0.004)	0.845 (0.001)	91.52 (0.13)	1.301 (0.005)	0.548 (0.002)
		EASE	85.89 (0.14)	0.577 (0.004)	0.198 (0.001)	86.83 (0.17)	0.583 (0.004)	0.193 (0.001)	**93.87 (0.1)**	0.576 (0.004)	0.242 (0.001)
		TCPR	86.29 (0.14)	0.715 (0.004)	0.27 (0.001)	86.96 (0.17)	0.819 (0.005)	0.295 (0.001)	93.82 (0.1)	0.634 (0.004)	0.265 (0.001)
		noHub	86.07 (0.15)	**0.295 (0.004)**	0.115 (0.001)	86.75 (0.17)	**0.299 (0.004)**	**0.115 (0.001)**	93.72 (0.1)	**0.2 (0.004)**	**0.101 (0.001)**
		noHub-S	**86.41 (0.14)**	0.499 (0.006)	**0.104 (0.001)**	**87.31 (0.17)**	0.406 (0.005)	0.121 (0.001)	93.79 (0.1)	0.416 (0.005)	0.126 (0.001)
	LaplacianShot	None	85.23 (0.13)	1.711 (0.01)	0.474 (0.001)	86.14 (0.15)	1.921 (0.011)	0.509 (0.001)	92.61 (0.1)	1.164 (0.008)	0.395 (0.001)
		L2	85.9 (0.13)	0.892 (0.006)	0.321 (0.001)	86.47 (0.15)	0.867 (0.006)	0.304 (0.001)	93.17 (0.09)	0.635 (0.004)	0.267 (0.001)
		CL2	82.08 (0.15)	1.112 (0.009)	0.318 (0.001)	84.62 (0.16)	0.954 (0.006)	0.34 (0.001)	93.01 (0.1)	0.702 (0.004)	0.309 (0.001)
		ZN	85.97 (0.13)	0.86 (0.005)	0.319 (0.001)	86.67 (0.15)	0.912 (0.006)	0.335 (0.001)	93.3 (0.1)	0.698 (0.004)	0.305 (0.001)
		ReRep	84.34 (0.14)	1.599 (0.003)	0.819 (0.001)	85.61 (0.16)	1.615 (0.004)	0.845 (0.001)	92.2 (0.1)	1.304 (0.005)	0.549 (0.002)
		EASE	86.24 (0.13)	0.573 (0.004)	0.198 (0.001)	86.74 (0.15)	0.582 (0.004)	0.194 (0.001)	93.31 (0.09)	0.578 (0.004)	0.243 (0.001)
		TCPR	86.16 (0.13)	0.712 (0.004)	0.269 (0.001)	85.72 (0.16)	0.813 (0.005)	0.293 (0.001)	92.99 (0.1)	0.638 (0.004)	0.264 (0.001)
		noHub	**86.25 (0.13)**	**0.292 (0.004)**	0.115 (0.001)	**86.78 (0.16)**	**0.299 (0.004)**	**0.115 (0.001)**	**93.38 (0.09)**	**0.197 (0.004)**	**0.1 (0.001)**
		noHub-S	85.79 (0.13)	0.494 (0.006)	**0.103 (0.001)**	86.44 (0.16)	0.397 (0.005)	0.12 (0.001)	93.36 (0.1)	0.42 (0.005)	0.126 (0.001)
	ObliqueManifold	None	87.46 (0.13)	1.712 (0.01)	0.472 (0.001)	88.16 (0.15)	1.913 (0.01)	0.509 (0.001)	94.75 (0.09)	1.161 (0.008)	0.395 (0.001)
		L2	87.61 (0.13)	0.889 (0.005)	0.321 (0.001)	88.14 (0.15)	0.862 (0.006)	0.306 (0.001)	**94.8 (0.09)**	0.642 (0.004)	0.268 (0.001)
		CL2	86.03 (0.14)	1.112 (0.009)	0.317 (0.001)	87.64 (0.16)	0.949 (0.006)	0.338 (0.001)	94.67 (0.09)	0.703 (0.004)	0.31 (0.001)
		ZN	87.88 (0.13)	0.852 (0.005)	0.32 (0.001)	**88.43 (0.15)**	0.908 (0.006)	0.335 (0.001)	94.77 (0.08)	0.697 (0.004)	0.306 (0.001)
		ReRep	87.62 (0.12)	1.599 (0.003)	0.819 (0.001)	88.15 (0.15)	1.616 (0.004)	0.845 (0.001)	94.48 (0.09)	1.302 (0.005)	0.547 (0.002)
		EASE	86.75 (0.13)	0.573 (0.004)	0.198 (0.001)	87.78 (0.15)	0.583 (0.004)	0.193 (0.001)	94.16 (0.09)	0.57 (0.004)	0.24 (0.001)
		TCPR	**87.94 (0.12)**	0.718 (0.004)	0.271 (0.001)	88.15 (0.15)	0.816 (0.005)	0.294 (0.001)	94.47 (0.09)	0.635 (0.004)	0.265 (0.001)
		noHub	87.23 (0.13)	**0.297 (0.004)**	0.115 (0.001)	87.95 (0.16)	**0.296 (0.004)**	**0.114 (0.001)**	94.13 (0.09)	**0.197 (0.004)**	**0.1 (0.001)**
		noHub-S	87.13 (0.14)	0.495 (0.006)	**0.103 (0.001)**	87.84 (0.16)	0.399 (0.005)	0.12 (0.001)	94.06 (0.09)	0.421 (0.005)	0.126 (0.001)
	SIAMESE	None	58.82 (0.31)	1.722 (0.01)	0.473 (0.001)	82.56 (0.22)	1.93 (0.01)	0.511 (0.001)	82.22 (0.37)	1.154 (0.008)	0.396 (0.001)
		L2	87.11 (0.13)	0.894 (0.006)	0.321 (0.001)	87.34 (0.15)	0.861 (0.005)	0.305 (0.001)	94.15 (0.1)	0.638 (0.004)	0.266 (0.001)
		CL2	83.99 (0.16)	1.107 (0.009)	0.318 (0.001)	86.71 (0.16)	0.953 (0.006)	0.339 (0.001)	94.48 (0.09)	0.704 (0.004)	0.31 (0.001)
		ZN	20.0 (0.0)	0.856 (0.005)	0.319 (0.001)	20.0 (0.0)	0.913 (0.006)	0.334 (0.001)	20.0 (0.0)	0.702 (0.004)	0.305 (0.001)
		ReRep	36.41 (0.3)	1.597 (0.003)	0.818 (0.001)	76.49 (0.24)	1.613 (0.004)	0.846 (0.001)	60.36 (0.6)	1.299 (0.005)	0.547 (0.002)
		EASE	87.82 (0.13)	0.579 (0.004)	0.199 (0.001)	88.06 (0.16)	0.586 (0.004)	0.192 (0.001)	94.36 (0.09)	0.571 (0.004)	0.241 (0.001)
		TCPR	87.8 (0.13)	0.717 (0.004)	0.27 (0.001)	87.95 (0.16)	0.822 (0.005)	0.295 (0.001)	94.25 (0.1)	0.637 (0.004)	0.266 (0.001)
		noHub	87.78 (0.14)	**0.29 (0.004)**	0.114 (0.001)	87.99 (0.17)	**0.297 (0.004)**	**0.115 (0.001)**	94.56 (0.09)	**0.196 (0.004)**	**0.1 (0.001)**
		noHub-S	**88.03 (0.13)**	0.492 (0.006)	**0.103 (0.001)**	**88.31 (0.16)**	0.398 (0.005)	0.12 (0.001)	**94.69 (0.09)**	0.416 (0.005)	0.127 (0.001)
	SimpleShot	None	78.56 (0.14)	1.709 (0.01)	0.473 (0.001)	80.32 (0.16)	1.937 (0.01)	0.51 (0.001)	89.27 (0.11)	1.16 (0.008)	0.395 (0.001)
		L2	83.81 (0.13)	0.887 (0.005)	0.322 (0.001)	84.82 (0.15)	0.86 (0.006)	0.305 (0.001)	92.06 (0.1)	0.632 (0.004)	0.266 (0.001)
		CL2	81.05 (0.14)	1.12 (0.009)	0.318 (0.001)	83.82 (0.16)	0.956 (0.006)	0.337 (0.001)	92.19 (0.1)	0.701 (0.004)	0.31 (0.001)
		ZN	83.92 (0.13)	0.858 (0.005)	0.32 (0.001)	85.1 (0.15)	0.912 (0.006)	0.335 (0.001)	92.17 (0.1)	0.699 (0.004)	0.305 (0.001)
		ReRep	79.26 (0.16)	1.597 (0.003)	0.819 (0.001)	82.7 (0.16)	1.617 (0.004)	0.846 (0.001)	91.48 (0.11)	1.299 (0.005)	0.549 (0.002)
		EASE	83.65 (0.13)	0.579 (0.004)	0.199 (0.001)	84.47 (0.15)	0.585 (0.004)	0.193 (0.001)	92.01 (0.1)	0.572 (0.004)	0.241 (0.001)
		TCPR	83.77 (0.13)	0.717 (0.004)	0.27 (0.001)	84.81 (0.15)	0.815 (0.005)	0.294 (0.001)	91.84 (0.1)	0.634 (0.004)	0.264 (0.001)
		noHub	**85.73 (0.13)**	**0.294 (0.004)**	0.115 (0.001)	**86.58 (0.15)**	**0.298 (0.004)**	**0.115 (0.001)**	93.21 (0.09)	**0.195 (0.004)**	**0.1 (0.001)**
		noHub-S	84.39 (0.13)	0.494 (0.006)	**0.103 (0.001)**	86.38 (0.15)	0.407 (0.005)	0.12 (0.001)	**93.39 (0.09)**	0.421 (0.005)	0.127 (0.001)
	α-TIM	None	80.61 (0.15)	1.711 (0.01)	0.473 (0.001)	83.05 (0.18)	1.928 (0.01)	0.51 (0.001)	84.89 (0.29)	1.153 (0.008)	0.396 (0.001)
		L2	83.71 (0.16)	0.892 (0.005)	0.323 (0.001)	84.69 (0.18)	0.863 (0.005)	0.304 (0.001)	92.88 (0.1)	0.633 (0.004)	0.266 (0.001)
		CL2	82.35 (0.16)	1.111 (0.009)	0.318 (0.001)	84.06 (0.18)	0.949 (0.006)	0.339 (0.001)	92.81 (0.1)	0.7 (0.004)	0.31 (0.001)
		ZN	83.93 (0.13)	0.857 (0.005)	0.321 (0.001)	85.07 (0.15)	0.912 (0.006)	0.336 (0.001)	92.15 (0.1)	0.698 (0.004)	0.306 (0.001)
		ReRep	83.4 (0.14)	1.596 (0.003)	0.82 (0.001)	84.4 (0.16)	1.615 (0.004)	0.845 (0.001)	93.19 (0.09)	1.302 (0.005)	0.547 (0.002)
		EASE	82.72 (0.14)	0.576 (0.004)	0.2 (0.001)	83.86 (0.16)	0.583 (0.004)	0.193 (0.001)	92.31 (0.1)	0.572 (0.004)	0.242 (0.001)
		TCPR	84.21 (0.15)	0.718 (0.004)	0.27 (0.001)	84.63 (0.18)	0.814 (0.005)	0.293 (0.001)	92.44 (0.1)	0.635 (0.004)	0.265 (0.001)
		noHub	**85.56 (0.13)**	**0.293 (0.004)**	0.115 (0.001)	**86.37 (0.16)**	**0.3 (0.004)**	**0.115 (0.001)**	92.89 (0.1)	**0.193 (0.004)**	**0.099 (0.001)**
		noHub-S	83.96 (0.15)	0.496 (0.006)	**0.102 (0.001)**	86.01 (0.16)	0.395 (0.005)	0.12 (0.001)	**93.24 (0.1)**	0.422 (0.005)	0.126 (0.001)

Table 5. WideRes28-10: 5-shot.

References

[1] Malik Boudiaf, Ziko Imtiaz Masud, Jérôme Rony, José Dolz, Pablo Piantanida, and Ismail Ben Ayed. Transductive Information Maximization For Few-Shot Learning. In *NeurIPS*, 2020. 4

[2] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised Representation Learning by Predicting Image Rotations. In *ICLR*, 2018. 4

[3] Puneet Mangla, Mayank Singh, Abhishek Sinha, Nupur Kumari, Vineeth N Balasubramanian, and Balaji Krishnamurthy. Charting the Right Manifold: Manifold Mixup for Few-shot Learning. In *WACV*, 2020. 4

[4] Jose C Principe. *Information theoretic learning: Renyi's entropy and kernel perspectives*. Springer Science & Business Media, 2010. 3

[5] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *JMLR*, 2008. 2, 4

[6] Olivier Veilleux, Malik Boudiaf, Pablo Piantanida, and Ismail Ben Ayed. Realistic evaluation of transductive few-shot learning. In *NeurIPS*, 2021. 4

[7] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold Mixup: Better Representations by Interpolating Hidden States. In *ICML*, 2019. 4

[8] Tongzhou Wang and Phillip Isola. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *ICML*, 2020. 3, 4

[9] Hao Zhu and Piotr Koniusz. EASE: Unsupervised Discriminant Subspace Learning for Transductive Few-Shot Learning. In *CVPR*, 2022. 4

Paper V

Norm-count Hypothesis: On the Relationship Between Norm and Object Count in Visual Representations

Daniel J. Trosten *daniel.j.trosten@uit.no*

Sigurd Løkse

Robert Jenssen

Michael Kampffmeyer

Department of Physics and Technology, UiT The Arctic University of Norway.
UiT Machine Learning Group

Abstract

We present a novel hypothesis on norms of representations produced by convolutional neural networks (CNNs). In particular, we propose the norm-count hypothesis (NCH), which states that there is a monotonically increasing relationship between the number of certain objects in the image, and the norm of the corresponding representation. We formalize and prove our hypothesis in a controlled setting, showing that the NCH is true for linear CNNs followed by global average pooling, when they are applied to a certain class of images. We present experimental evidence that corroborates our hypothesis for CNN-based representations. Our experiments are conducted on several image datasets, in both supervised, self-supervised, and few-shot learning – providing new insight on the relationship between object counts and representation norms.

1 Introduction

The ability to learn high-quality representations from a wide range of complex data types lies at the heart of the success of deep learning. Recently, several works have studied how deep learning-based representations can be embedded in non-Euclidean spaces, to further improve representation quality (Bronstein et al., 2017). In particular, embedding representations on the hypersphere using L_2 normalization has proven to be particularly promising direction for several downstream applications, such classification and regression (Mettes et al., 2019; Scott et al., 2021; Tan et al., 2022), self-supervised learning (SSL) (Chen et al., 2020; Caron et al., 2021), and few-shot learning (FSL) (Wang et al., 2019; Fei et al., 2021; Trosten et al., 2023).

However, despite the widespread use of L_2 normalization in several aspects of deep learning, little work exists on understanding exactly what type of information the norm contains, and why discarding this information improves representation quality. Hence, we still lack critical understanding of the role of norms and normalization in deep learning. In this work, we aim to improve the understanding of norms of convolutional neural network (CNN)-based representations, and thus to better understand the benefits of L_2 normalization. Our work is built on a novel hypothesis for image representations obtained with a CNN:

Informal Definition 1 (Norm-count hypothesis)**.** There is a monotonically increasing relationship between the norm of a representation produced by a CNN, and the number of objects in the image for which the CNN is trained to recognize.

The norm-count hypothesis (NCH) proposes a theory on the relationship between norm, and the number of detections produced by the CNN. Moreover, an implicit consequence of the NCH is that angles encode information about the types of objects detected by the CNN in the given image. In this work, we assess the validity of the NCH for CNNs used in supervised, self-supervised and few-shot learning.

The main contributions of our work are:

1. We propose the NCH – stating that there is a monotonically increasing relationship between the norm of a representation, and the number of objects in the given image.
2. We prove that the NCH is true in a controlled setting, assuming that input images are composed of several *object images*, for which the feature extractor provides a delta-like response in a single channel.
3. We conduct an extensive experimental evaluation with images from MNIST, STL-10, and Pascal VOC, in supervised, self-supervised and few-shot learning. Our results show monotonically increasing relationships between norm and count, for several models and datasets – corroborating the NCH. We also find that discarding norm with L_2 normalization improves classification performance in the majority of experimental configurations.

The rest of the paper is structured as follows: Section 2 gives an overview of work related to ours. In Section 3, we theoretically analyze the NCH, and prove that it holds under certain assumptions. Section 4 includes the results of our experiments. We finish the paper with Section 5, presenting some concluding remarks and directions for future work.

2 Related work

In this section, we summarize other work related to this paper. We emphasize that our work is complementary to these, as none of the works below provide an accurate and rigorous understanding of the information contained in norms of CNN-based representations.

2.1 Embedding representations on the hypersphere

Embedding representations on the hypersphere instead of in Euclidean space has shown to be beneficial for both supervised classification and regression (Mettes et al., 2019; Scott et al., 2021; Tan et al., 2022). Mettes et al. (2019) develop classification and regression losses on the hypersphere, illustrating that L_2 normalized representations and prototypes are beneficial for both classification and regression. The more recent work by Tan et al. (2022) shows that a supervised classification model can be regularized with a self-supervised contrastive loss on the unit hypersphere.

L_2 normalization is also common in models for self-supervised learning of image representations (Chen et al., 2020; He et al., 2020; Grill et al., 2020; Caron et al., 2020; 2021; Goyal et al., 2022; Li et al., 2023). The benefit of L_2 normalization appears to stem from similarity measures and contrastive losses being more well-behaved after discarding the norm – resulting in compact and well-separated classes (Wang & Isola, 2020). However, little work exists on this topic.

Recent methods for transductive FSL have also found L_2 normalized representations to be beneficial for classification performance (Wang et al., 2019; Veilleux et al., 2021; Zhu & Koniusz, 2022; Xu et al., 2022; Trosten et al., 2023). Trosten et al. (2023) argue that L_2 normalization helps reduce the hubness problem (Radovanovic et al., 2010; Fei et al., 2021) in FSL, and show that embedding points uniformly on the hypersphere completely eliminates hubness. To the best of our knowledge, the work by Trosten et al. (2023) is one of the first works that attempt to understand why L_2 normalization is beneficial in FSL. Nevertheless, it is limited to the hubness problem, and does not make any advances in understanding the information contained in the representation norm.

2.2 Hyperspherical regularization

Hyperspherical embeddings have also shown to be beneficial to regularize training of deep neural networks (DNNs) (Salimans & Kingma, 2016; Liu et al., 2017; 2018; 2021). These methods constrain the weight vectors in DNNs to lie on the unit hypersphere. Liu et al. (2017) show that hyperspherical weights improve the conditioning of the optimization problem, helping the optimizer converge faster to potentially better solutions. However, our work is orthogonal to this, since we aim to understand norms of *representations*, and not norms of weights.

3 Norm-count hypothesis

The purpose of this section is to formalize the NCH, and to analyze it in a rigorous theoretical setting. To do so, we assume certain properties of the feature extractor (*e.g.* CNN). We then argue that these models admit a certain class of images, referred to as *object images*. These images can be seen as "prototypes" of the objects the feature extractor is trained to detect. In Section 4 we demonstrate that, for a supervised model, object images coincide with the classes the model is trained to recognize.

Having established properties of the feature extractor and corresponding object images, we prove that the norm of the representation produced by the feature extractor is proportional to the number of object images present in the given image. Thereby corroborating the NCH. Note that all proofs are given in Appendix A.

We start by providing exact definitions of images, image translation, detectors, and global pooling operators.

Definition 1 (Images)**.** The set of images with C channels and size $W \times H$ is defined as

$$\mathcal{I}_{C,W,H} = \{I : \mathbb{N}^0_{<C} \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{R} \mid I(c,x,y) = 0 \text{ if } (x,y) \notin \mathbb{N}^0_{<W} \times \mathbb{N}^0_{<H}\} \tag{1}$$

where $\mathbb{N}^0_{<a} = \{0, 1, \dots, a-1\}$.

Definition 2 (Translation operator)**.** A translation operator $\mathrm{Tr}_{x',y'} : \mathcal{I}_{C,W,H} \to \mathcal{I}_{C,W,H}$ shifts the given image by x', y' pixels

$$\mathrm{Tr}_{x',y'}(I)(c,x,y) = I(c, x-x', y-y') \tag{2}$$

Definition 3 (Translation equivariance)**.** A mapping $f : \mathcal{I}_{C,W,H} \to \mathcal{I}_{C',W',H'}$ is translation equivariant iff for a translation operator $\mathrm{Tr}_{x',y'}$, we have

$$f \circ \mathrm{Tr}_{x',y'} = \mathrm{Tr}_{x',y'} \circ f \tag{3}$$

where $\circ$ denotes the composition of functions.

Definition 4 (Strict and relaxed detectors)**.** A detector is a *translation equivariant* mapping $f : \mathcal{I}_{C,W,H} \to \mathcal{I}_{C',W',H'}$, which also satisfies at least one of the conditions

$$f(I_1 + I_2) = f(I_1) + f(I_2) \tag{4}$$
$$|f(I_1 + I_2)| \preccurlyeq |f(I_1) + f(I_2)| \tag{5}$$

where addition and absolute value are defined element-wise, and $I_1 \preccurlyeq I_2$ implies that $I_1(k,x,y) \le I_2(k,x,y)$ for all k, x, y.
If f satisfies condition (4) (and thereby also condition (5)), then f is said to be a **strict detector**. If f only satisfies condition (5), it is called a **relaxed detector**.

The following propositions show properties of detectors that are relevant for CNNs.

Proposition 1 (Composition of detectors)**.** *For detectors f and g, the following holds:*

1. *If f and g are strict detectors, then $g \circ f$ is a strict detector.*
2. *If f is a strict detector and g is a relaxed detector, then $g \circ f$ is a relaxed detector.*

Proposition 2 (Convolutions are strict detectors.). *Let* $\mathrm{Conv}_K : \mathcal{I}_{C,W,H} \to \mathcal{I}_{1,W+w-1,H+h-1}$ *be the convolution operator convolving the given image,* $I \in \mathcal{I}_{C,W,H}$*, with a filter,* $K \in \mathcal{I}_{C,h,w}$

$$\mathrm{Conv}_K(I)(0,x,y) = \sum_{c=0}^{C-1} \sum_{x'=-\infty}^{\infty} \sum_{y'=-\infty}^{\infty} K(c,x',y') I(c, x-x', y-y'). \tag{6}$$

Then Conv_K *is a strict detector with* $(C', H', W') = (1, W+w-1, H+h-1)$*.*

Proposition 3 (LeakyReLU is a relaxed detector.). *Let* $\mathrm{LeakyReLU}_\alpha : \mathcal{I}_{C,W,H} \to \mathcal{I}_{C,W,H}$ *be defined element-wise as*

$$\mathrm{LeakyReLU}_\alpha(I(k,x,y)) = \begin{cases} I(k,x,y), & \textit{if } I(k,x,y) > 0 \\ \alpha \cdot I(k,x,y), & \textit{otherwise} \end{cases} \tag{7}$$

for all k, x, y*, and* $\alpha \in [0, 1)$*. Then* $\mathrm{LeakyReLU}_\alpha$ *is a relaxed detector with* $(C', W', H') = (C, W, H)$*. This also holds for the standard* $\mathrm{ReLU}(x) = \max\{0, x\}$ *activation, since* $\mathrm{ReLU} = \mathrm{LeakyReLU}_0$*.*

From Propositions 1 and 2, we see that linear CNNs – *i.e.* networks consisting only of compositions of convolutions – are strict detectors. Furthermore, combining Propositions 1, 2 and 3 shows that a CNN consisting of convolutions and LeakyReLU (or ReLU) activations are compositions of relaxed detectors. These propositions thus cover the most important building blocks of CNNs, along with the most common activation functions.

CNNs for classification and representation learning are often followed by a global pooling operator that aggregates information over the spatial dimensions. The following definition considers a general pooling operation, which we use in our theoretical analysis. We then prove that global average pooling (GAP) – one of the most frequently used pooling operations – is a special case of the general pooling operation.

Definition 5 (Global pooling operator). Let $I \in \mathcal{I}_{C,W,H}$ be an image. The mapping $\mathrm{Pool} : \mathcal{I}_{C,W,H} \to \mathbb{R}^C$ is called a global pooling operator if there exists non-negative real numbers $\gamma_0, \dots, \gamma_{C-1}$ independent of I, such that

$$|\,\mathrm{Pool}(I)_k| \leq \gamma_k \left| \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} I(k,x,y) \right|, \quad k \in \mathbb{N}^0_{<C}. \tag{8}$$

Proposition 4 (GAP is a global pooling operator). *For an image* $I \in \mathcal{I}_{C,W,H}$*, let GAP be defined as*

$$\mathrm{GAP}(I)_k = \frac{1}{WH} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} I(k,x,y), \quad k \in \mathbb{N}^0_{<C}. \tag{9}$$

Then GAP *is a global pooling operator with* $\gamma_k = \frac{1}{WH}$*,* $\forall k \in \mathbb{N}^0_{<C}$*.*

Our objective is now to understand norms of representations computed by a detector followed by a global pooling operator. Hence, we define object images as "prototypical" images that give a delta-like response when processed by the given detector.

Definition 6 (Object images)**.** An image $O_j \in \mathcal{I}_{C,h,w}$ is said to be an object image of type j w.r.t. the detector f, iff

$$f(O_j) = \delta_{j,0,0} \tag{10}$$

where $\delta_{j,x',y'}$ is the Kronecker delta function

$$\delta_{j,x',y'}(k,x,y) = \begin{cases} 1, & (k,x,y) = (j,x',y') \\ 0, & \text{otherwise} \end{cases}. \tag{11}$$

The set of all object images of type j is denoted $\Omega_j = \{O \mid f(O) = \delta_{j,0,0}\}$.

In a supervised classification setting, the object image types will tend to coincide with the classes the model is trained to detect. This is because supervised models are trained to output a one-hot prediction vector, resembling the delta-response assumed in Definition 6, after global pooling.

Object images can also be interpreted as "parts of a whole", where it is assumed that image motifs consist of a collection of object images. An image of a car, for instance, will be composed of object images with type "wheel", "car body", *etc.*

We will now define multiple objects images (MO-images) as images composed of one or more object images. This definition gives rise to a natural notion of object "count" in the image, which is necessary to formalize the NCH.

Definition 7 (Multiple objects image)**.** An MO-image, $I \in \mathcal{I}_{C,W,H}$, constructed from object images in $\Omega_0, \dots, \Omega_{C'-1}$ is defined as

$$I = \sum_{j=0}^{C'-1} \sum_{(O,x',y') \in \mathcal{P}_j} T_{(x',y')}(O). \tag{12}$$

where C' is the number of object types. $\mathcal{P}_j$ is a set of 3-tuples, where the first element is an object image from Ω_j, and the second third elements are the positions of that object image in I.

We now have the following theorem stating that the NCH is true for detectors and global pooling operators applied to MO-images.

Theorem 1 (Norm-count hypothesis – simplified setting)**.** *Let $f : \mathcal{I}_{C,W,H} \to \mathcal{I}_{C',W',H'}$ be a relaxed detector with object images $\Omega_0, \dots, \Omega_{C'-1}$, and let $I \in \mathcal{I}_{C,W,H}$ be a MO-image constructed from the same object images. Then, if $\mathbf{z} = [z_0, \dots, z_{C'-1}]^\top \in \mathbb{R}^{C'}$ is the output of a global pooling operator applied to the feature maps $f(I)$, we have*

$$|z_k| = |\operatorname{Pool}(I)_k| \leq \gamma_k |\mathcal{P}_k|, \quad k \in \mathbb{N}^0_{<C'} \tag{13}$$

for non-negative numbers $\gamma_0, \dots, \gamma_{C-1}$ independent of I.

Corollary 1.1 (L_p norm of $\mathbf{z}$)**.** *For $p > 0$, the L_p norm of $\mathbf{z}$ from Theorem 1 is*

$$||\mathbf{z}||_p \leq \left(\sum_{k=0}^{C'-1} (\gamma_k |\mathcal{P}_k|)^p \right)^{\frac{1}{p}} \tag{14}$$

Corollary 1.1 shows that the L_p norm of representations is upper bounded by a monotonically increasing function of the count, corroborating the NCH.

Corollary 1.2 (Strict detector and GAP)**.** *If f is a strict detector, and* GAP *is used in place of* Pool*, Theorem 1 simplifies to*

$$z_k = \text{GAP}(I)_k = \frac{|\mathcal{P}_k|}{W'H'} \tag{15}$$

Corollary 1.2 shows that for strict detectors followed by GAP, there is an exact proportionality between each component of $\mathbf{z}$, and the number of objects of the corresponding type present in the image. Since linear CNNs are strict detectors, this corollary proves that each dimension, in a representation produced by linear CNNs, is proportional to the number of object images in the given MO-image.

Furthermore, Theorem 1 states that the norm of $\mathbf{z}$ is directly related to the absolute (total) count of objects in the image, regardless of the type of the object images. This is expected, since the perfect detector produces a set of delta-responses, and the global pooling operator aggregates these over the spatial dimensions.

In contrast to the norm, the angle of $\mathbf{z}$ depends on the count of one object type relative to the count of another object type. This is demonstrated by the following result.

Result 1 (Semantic information in angles)**.** *Suppose $I_1, I_2 \in \mathcal{I}_{C,W,H}$ are MO-images processed by a perfect detectors f followed by GAP. Furthermore, assume that I_1 only consists of objects of type j, and that I_2 only consists of objects of type k. This gives*

$$\mathbf{z}_1 = \text{GAP}(f(I_1)) = \frac{|\mathcal{P}_j^{(1)}|}{W'H'}\mathbf{e}_j \quad \textit{and} \quad \mathbf{z}_2 = \text{GAP}(f(I_2)) = \frac{|\mathcal{P}_k^{(2)}|}{W'H'}\mathbf{e}_k \tag{16}$$

where $\mathbf{e}_j$ ($\mathbf{e}_k$) denotes the vector where element j (k) is 1, and all other elements are 0.
Then, if $(||\mathbf{z}||, \boldsymbol{\theta}(\mathbf{z}))$ denotes the transformation of $\mathbf{z}$ to hyperspherical coordinates, we can consider the following two cases:

1. Different class, same count*: $j \neq k$ and $|\mathcal{P}_j^{(1)}| = |\mathcal{P}_k^{(2)}|$, which gives*

$$||\mathbf{z}_1|| = ||\mathbf{z}_2|| \quad \textit{and} \quad \boldsymbol{\theta}(\mathbf{z}_1) \neq \boldsymbol{\theta}(\mathbf{z}_2) \tag{17}$$

2. Same class, different count*: $j = k$ and $|\mathcal{P}_j^{(1)}| \neq |\mathcal{P}_k^{(2)}|$, which gives*

$$||\mathbf{z}_1|| \neq ||\mathbf{z}_2|| \quad \textit{and} \quad \boldsymbol{\theta}(\mathbf{z}_1) = \boldsymbol{\theta}(\mathbf{z}_2) \tag{18}$$

In both cases in Result 1, the angles $\boldsymbol{\theta}(\mathbf{z}_1)$ and $\boldsymbol{\theta}(\mathbf{z}_2)$ are most informative of the image classes (object types). When the images belong to different classes (case 1), the discriminative power lies in the angles and not in the norms. Conversely, when I_1 and I_2 belong to the same class (case 2), the within class distance is 0 for the angles, but non-zero for the norms. The angle thus encodes information about which classes (object types) that were detected in the image – *i.e.* the semantic information.

4 Experiments

The purpose of these experiments is to experimentally investigate the NCH in a controlled setting. We design the experiments to have fine-grained control of the "count" in each image. This allows us to properly examine the relationship between norm and count – both quantitatively and qualitatively. Our experiments are conducted with both supervised, self-supervised and few-shot learning models.

Although our theoretical results hold for arbitrary L_p norms, we focus on L_2 norms in the experimental evaluation. This is because the L_2 norm is the one most frequently encountered in other works (see Section 2), and has known benefits related to optimization (Liu et al., 2021), as well as alignment, uniformity, and class separability (Wang & Isola, 2020).

4.1 Setup

Models and architectures. Our evaluation is performed with models using the following two CNN architectures:

- `Simple-6`: A simple 6-layer CNN followed by GAP. The model has ReLU activations, and max pooling after every second convolutional layer. No batch-normalization or other forms of normalization is applied anywhere in the architecture. The models based on this architecture are trained by us, using either a supervised cross-entropy loss, or the self-supervised loss from SimCLR (Chen et al., 2020).
- `ResNet-50`: The standard 50-layer residual network architecture by He et al. (2016). We use the supervised model available in Torchvision[1], which is trained on the ImageNet dataset. For the self-supervised version of this architecture, we use the model available in Lightning Bolts[2], also trained on ImageNet.

Datasets. In order to mimic the properties of MO-images in evaluation, we start with datasets consisting of natural images (MNIST (Lecun et al., 1998) and STL-10 (Coates et al., 2011)). Then, to generate a single evaluation image, we sample a random number of images, and place them at random positions in a 4×4 grid. This gives us an image that resembles an MO-image, where we know the true count – *i.e.* the number of object images.

These grid-based datasets are generated either with all images in the grid from the same class (single label), or with random class affiliations (mixed) for the small images.

For MNIST, we fill the empty grid positions with 0-values, as indicated by Definition 7. For STL-10, we generate datasets with 3 different approaches to filling the empty grid slots:

- *Zeros*: empty grid positions are filled with 0-values.
- *Random*: empty grid positions are filled with random Gaussian noise with the same mean and standard deviation as the object images.
- *Blur*: the background for the whole grid is a random blurred image, and the object images are placed on top of this image.

We experiment with different fill modes to ensure that the results are not skewed by changes in global image statistics, such as mean and variance.

In addition to the grid datasets, we use the Pascal VOC 2007–2013 object detection datasets (Everingham et al., 2010) to evaluate the relationship between norm and count in real images. The object count in an image is computed as the total number of ground-truth bounding boxes in the image. We then select images with $2 \leq \text{count} \leq 9$.

We note that the VOC dataset can contain objects that are not labeled, meaning that they do not have a bounding box, and are thus not included in the count. This makes the VOC dataset more challenging, but also more realistic, compared to the grid datasets.

Quantitative evaluation of monotonic increase. We use a weighted quadratic regression model to quantitatively assess whether there is an increasing relationship between feature norm and count

$$||\mathbf{z}_i|| = \beta_0 + \beta_1 c(\mathbf{x}_i) + \beta_2 c(\mathbf{x}_i)^2 + \epsilon_i \tag{19}$$

where the residual, ϵ_i, is assumed to be Gaussian with zero mean and standard deviation $\sigma_{c(\mathbf{x}_i)}$. We allow the standard deviation to be count-dependent to account for heteroskedasticity in the data. The parameter

[1]`https://pytorch.org/vision/stable/models/resnet.html`
[2]`https://lightning-bolts.readthedocs.io/en/0.3.4/self_supervised_models.html`.

estimates $(\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2)$ are computed using weighted least squares. Based on these estimates, we can test for monotonic increase by checking whether the derivative

$$\frac{\mathrm{d}\,||\mathbf{z}||}{\mathrm{d}\,c(\mathbf{x})} = \hat{\beta}_1 + 2\,\hat{\beta}_2\,c(\mathbf{x}) \tag{20}$$

is positive.

We report the value of the slope $s_q = \hat{\beta}_1 + 2\,\hat{\beta}_2\,c_q$, and the p-value resulting from testing

$$H_0 : \hat{\beta}_1 + 2\,\hat{\beta}_2\,c_q \leq 0 \quad \text{vs.} \quad H_1 : \hat{\beta}_1 + 2\,\hat{\beta}_2\,c_q > 0. \tag{21}$$

The p-value is denoted by p_q.

The slopes and p-values are computed at equally spaced points, $c_{0.25}$, $c_{0.5}$, and $c_{0.75}$, where

$$c_q = c_{\min} + q \cdot (c_{\max} - c_{\min}) \tag{22}$$

and $c_{\min}$ and $c_{\max}$ are the minimum and maximum counts in the dataset, respectively.

Few-shot learning experiments. In our few-shot experiments, we use the `ResNet-50` models described above, and evaluate on STL-10 and VOC, where we know the ground-truth counts. We report both 1 and 5-shot classification accuracy, using the Simpleshot classifier (Wang et al., 2019). The evaluation includes 10000 episodes, sampling 5 random classes and 15 queries from each class, in each episode.

Implementation. The experiments are implemented in Python with the PyTorch framework (Paszke et al., 2019). Our implementation will be made publicly available upon publication of the paper.

4.2 Results: supervised and self-supervised learning

Relationship between norm and count. Table 1 lists the results of the regression analyses for norm vs. count for the different configurations. The slopes and p-values show that the majority of configurations result in a statistically significant, monotonically increasing relationship between norm and count. The same trend can be observed in the box-plots in Figure 1.

For the supervised model on STL-10 and VOC, we observe that the increasing trend is not as clear as for the other models. This is likely because the supervised `ResNet-50` is trained on the ImageNet dataset. This dataset contains natural images with a varying number of objects. Since the model is trained to output the same prediction, regardless of the number of objects, the model learns to be approximately *count invariant*, resulting in a weaker relationship between norm and count.

Finally, we observe no increasing trend between norm and count for the SimCLR model on VOC with mixed labels. This is the most challenging configuration, since the images depict complex scenes with multiple objects of different type, and the model is trained without supervision. We hypothesize that the non-increasing relationship between norm and count is a result of SimCLR detecting objects that do not necessarily coincide with the labels. The object images for this model do therefore not correspond to the ground-truth objects, resulting in a norm that is no longer representative of the count.

L_2 normalization and classification performance. Table 2 shows the change in classification accuracy after L_2 normalizing the representations. The accuracies are computed both based on a prototypical classifier (classifying samples according to the closest class mean), and a linear classifier trained with stochastic gradient descent (SGD). These results show that L_2 normalization – *i.e.* discarding the norm – is beneficial for classification performance, resulting in increased accuracy. The increased accuracy after normalization illustrates that the norm contains little relevant class information, and acts as additional noise in the classifier. These findings are in line with the theory in Result 1, stating that angles carry most of the semantic information.

Table 1: Estimated slopes (s_q) and p-values (p_q) for the quadratic regression model for norm vs. count. A positive slope with a low p-value indicates a statistically significant, monotonically increasing relationship between norm and count.

Dataset	Model	Fill	Labels	$s_{0.25}$	$s_{0.5}$	$s_{0.75}$	$p_{0.25}$	$p_{0.5}$	$p_{0.75}$
MNIST	SimCLR (`Simple-6`)	Zeros	Mixed	0.147	0.206	0.264	0.000	0.000	0.000
			Single	0.131	0.190	0.249	0.000	0.000	0.000
	Supervised (`Simple-6`)	Zeros	Mixed	-0.075	0.142	0.359	1.000	0.000	0.000
			Single	0.027	0.152	0.277	0.000	0.000	0.000
STL-10	SimCLR (`ResNet-50`)	Blur	Mixed	0.288	0.203	0.117	0.000	0.000	0.000
			Single	0.295	0.202	0.109	0.000	0.000	0.000
		Random	Mixed	0.253	0.194	0.136	0.000	0.000	0.000
			Single	0.260	0.196	0.132	0.000	0.000	0.000
		Zeros	Mixed	0.231	0.207	0.183	0.000	0.000	0.000
			Single	0.237	0.210	0.183	0.000	0.000	0.000
	Supervised (`ResNet-50`)	Blur	Mixed	0.196	0.132	0.069	0.000	0.000	0.000
			Single	0.143	0.073	0.003	0.000	0.000	0.404
		Random	Mixed	0.292	0.152	0.012	0.000	0.000	0.052
			Single	0.250	0.105	-0.039	0.000	0.000	1.000
		Zeros	Mixed	0.305	0.184	0.062	0.000	0.000	0.000
			Single	0.283	0.144	0.004	0.000	0.000	0.317
VOC	SimCLR (`ResNet-50`)	–	Mixed	-0.017	-0.017	-0.016	1.000	1.000	0.996
			Single	0.001	0.016	0.030	0.391	0.051	0.019
	Supervised (`ResNet-50`)	–	Mixed	0.081	0.059	0.037	0.000	0.000	0.000
			Single	0.065	0.047	0.030	0.000	0.000	0.024

Table 2: Differences in classification accuracy [%] after L_2 normalizing the representations, for the prototypical and SGD classifiers. Positive values indicate an that L_2 normalization improves classification accuracy.

Dataset	Model	Fill	Δ (Proto)	Δ (SGD)
MNIST	SimCLR (`Simple-6`)	Zeros	2.92	0.0
	Supervised (`Simple-6`)	Zeros	3.78	-1.01
STL-10	SimCLR (`ResNet-50`)	Blur	13.4	0.2
		Random	10.8	-0.7
		Zeros	11.4	0.4
	Supervised (`ResNet-50`)	Blur	1.9	0.6
		Random	1.2	0.2
		Zeros	5.8	1.1
VOC	SimCLR (`ResNet-50`)	–	2.55	1.4
	Supervised (`ResNet-50`)	–	-0.49	1.44

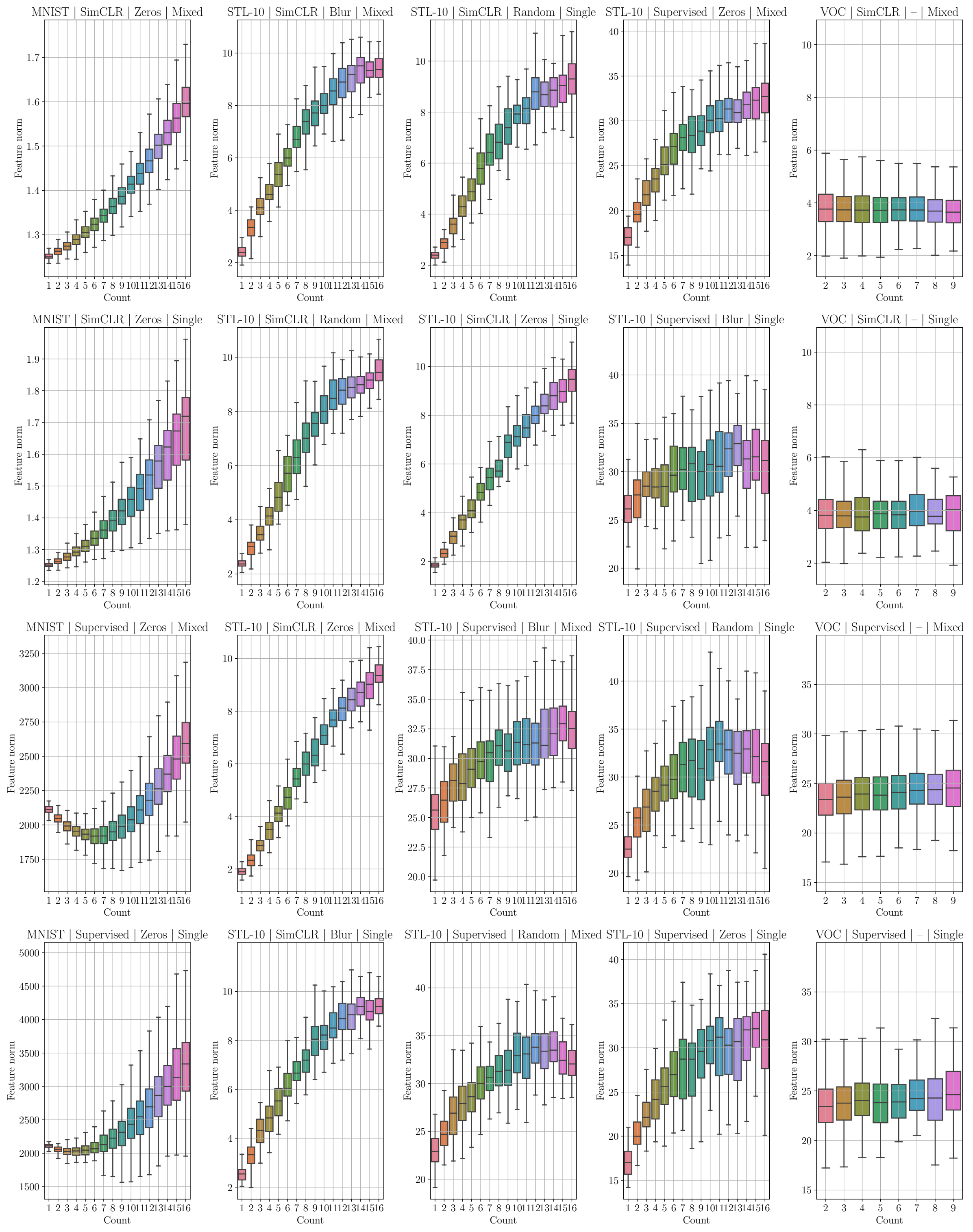


Figure 1: Boxplots illustrating the relationship between norm and count for the experimental configurations.

Table 3: 1 and 5-shot few-shot learning results on STL-10 and VOC, with the Simpleshot classifier (Wang et al., 2019). The results are averaged across 10000 episodes, with 95 % confidence intervals shown in parentheses. The correlation between norm and count, $\rho_{\text{norm,count}}$, could only be computed for evaluations without L_2 normalization, since the norm has zero variance after L_2 normalization.

				1-shot		5-shot	
Dataset	Model	Fill	Norm	Accuracy	$\rho_{\text{norm,count}}$	Accuracy	$\rho_{\text{norm,count}}$
STL-10	SimCLR (`ResNet-50`)	Blur	None	0.431 (0.006)	0.95 (0.001)	0.56 (0.006)	0.952 (0.001)
			L_2	0.522 (0.007)	– (–)	0.655 (0.006)	– (–)
		Random	None	0.432 (0.005)	0.95 (0.001)	0.558 (0.006)	0.952 (0.001)
			L_2	0.514 (0.007)	– (–)	0.654 (0.006)	– (–)
		Zeros	None	0.437 (0.006)	0.95 (0.001)	0.56 (0.006)	0.952 (0.001)
			L_2	0.518 (0.006)	– (–)	0.654 (0.006)	– (–)
	Supervised (`ResNet-50`)	Blur	None	0.774 (0.006)	0.323 (0.008)	0.882 (0.005)	0.155 (0.012)
			L_2	0.801 (0.006)	– (–)	0.925 (0.003)	– (–)
		Random	None	0.754 (0.006)	0.491 (0.007)	0.894 (0.003)	0.504 (0.007)
			L_2	0.772 (0.007)	– (–)	0.919 (0.003)	– (–)
		Zeros	None	0.717 (0.006)	0.674 (0.005)	0.872 (0.003)	0.676 (0.005)
			L_2	0.786 (0.007)	– (–)	0.914 (0.003)	– (–)
VOC	SimCLR (`ResNet-50`)	–	None	0.511 (0.007)	-0.109 (0.008)	0.721 (0.006)	-0.116 (0.006)
			L_2	0.651 (0.006)	– (–)	0.765 (0.006)	– (–)
	Supervised (`ResNet-50`)	–	None	0.688 (0.007)	0.044 (0.007)	0.826 (0.005)	0.086 (0.006)
			L_2	0.742 (0.006)	– (–)	0.827 (0.006)	– (–)

4.3 Results: few-shot learning

Table 3 shows 1 and 5-shot classification results on the STL-10 and VOC datasets. We observe that L_2 normalization results in a statistically significant improvement in performance for all configurations – illustrating that discarding the norm information is beneficial for few-shot classification performance.

We also compute the average correlation between norm and count ($\rho_{\text{norm,count}}$) across episodes. The correlation values show that also for few samples, there is a clear correlation between norm and count. This holds for all configurations, except for VOC with the SimCLR model. Similar to in Section 4.2, we hypothesize that this is because SimCLR detects another set of object images, whose counts are not well-represented by the ground-truth object count.

5 Conclusion and future work

We have presented the NCH – providing novel insight in the norms of representations produced by CNNs. Under certain assumptions on the model and input images, we proved the NCH, showing that each component in a given representation is upper-bounded by the number of object images present in the input image. Moreover, from our theoretical analysis, it follows that representation norms carry information related to count, whereas angles represent semantic information. This is a key result that helps explain why L_2 normalization is beneficial for downstream classification tasks.

In addition, we conduct a controlled experimental evaluation, showing that the NCH holds for supervised and self-supervised models. Our experiments also show that discarding the representation norm – which is strongly correlated with the count – using L_2 normalization, improves classification performance, both for standard and few-shot classifiers.

We believe that understanding representation norms through object counts is a promising direction of research. The popularity of hyperspherical embeddings in FSL, along with our results on the benefits of L_2 normalization, indicate that there is extra potential to improve FSL models if we better understand the effects of representation norms. Furthermore, although we focus on CNNs in this work, it is entirely possible that our results will generalize to other architectures. This can be verified by either showing that certain architectures meet the detector conditions, or by relaxing these conditions and extending the theoretical analysis.

References

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. doi: 10.1109/MSP.2017.2693418.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *NeurIPS*, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. In *ICCV*, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, 2020.

Adam Coates, Honglak Lee, and Andrew Y Ng. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *AISTATS*, 2011.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 88(2):303–338, 2010. doi: 10.1007/s11263-009-0275-4.

Nanyi Fei, Yizhao Gao, Zhiwu Lu, and Tao Xiang. Z-Score Normalization, Hubness, and Few-Shot Learning. In *ICCV*, 2021. doi: 10.1109/ICCV48922.2021.00021.

Priya Goyal, Quentin Duval, Isaac Seessel, Mathilde Caron, Ishan Misra, Levent Sagun, Armand Joulin, and Piotr Bojanowski. Vision Models Are More Robust And Fair When Pretrained On Uncurated Images Without Supervision. `http://arxiv.org/abs/2202.08360`, 2022.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised Learning. In *NeurIPS*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. doi: 10.1109/CVPR.2016.90.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*, 2020. doi: 10.1109/CVPR42600.2020.00975.

Bernd Jähne. *Digital image processing*. Springer, Berlin Heidelberg, 5., rev. and extended edition, 2002. ISBN 978-3-540-67754-3.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Alexander C. Li, Ellis Brown, Alexei A. Efros, and Deepak Pathak. Internet Explorer: Targeted Representation Learning on the Open Web. `http://arxiv.org/abs/2302.14051`, 2023.

Weiyang Liu, Yan-Ming Zhang, Xingguo Li, Zhiding Yu, Bo Dai, Tuo Zhao, and Le Song. Deep Hyperspherical Learning. In *NeurIPS*, 2017.

Weiyang Liu, Rongmei Lin, Zhen Liu, Lixin Liu, Zhiding Yu, Bo Dai, and Le Song. Learning towards Minimum Hyperspherical Energy. In *NeurIPS*, 2018.

Weiyang Liu, Rongmei Lin, Zhen Liu, Li Xiong, Bernhard Scholkopf, and Adrian Weller. Learning with Hyperspherical Uniformity. In *AISTATS*, 2021.

Pascal Mettes, Elise van der Pol, and Cees G. M. Snoek. Hyperspherical Prototype Networks. In *NeurIPS*, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Milosˇ Radovanovic, Alexandros Nanopoulos, and Mirjana Ivanovic. Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data. *JMLR*, pp. 45, 2010.

Tim Salimans and Durk P Kingma. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *NeurIPS*, 2016.

Tyler R. Scott, Andrew C. Gallagher, and Michael C. Mozer. Von Mises-Fisher Loss: An Exploration of Embedding Geometries for Supervised Learning. In *ICCV*, 2021.

Cheng Tan, Zhangyang Gao, Lirong Wu, Siyuan Li, and Stan Z. Li. Hyperspherical Consistency Regularization. In *CVPR*, 2022.

Daniel J. Trosten, Rwiddhi Chakraborty, Sigurd Løkse, Kristoffer Wickstrøm, Robert Jenssen, and Michael Kampffmeyer. Hubs and Hyperspheres: Reducing Hubness and Improving Transductive Few-shot Learning with Hyperspherical Embeddings. In *CVPR*, 2023.

Olivier Veilleux, Malik Boudiaf, Pablo Piantanida, and Ismail Ben Ayed. Realistic evaluation of transductive few-shot learning. In *NeurIPS*, 2021.

Tongzhou Wang and Phillip Isola. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *ICML*, 2020.

Yan Wang, Wei-Lun Chao, Kilian Q. Weinberger, and Laurens van der Maaten. SimpleShot: Revisiting Nearest-Neighbor Classification for Few-Shot Learning. *arXiv:1911.04623 [cs]*, 2019.

Jing Xu, Xu Luo, Xinglin Pan, Wenjie Pei, Yanan Li, and Zenglin Xu. Alleviating the Sample Selection Bias in Few-shot Learning by Removing Projection to the Centroid. In *NeurIPS*, 2022.

Hao Zhu and Piotr Koniusz. EASE: Unsupervised Discriminant Subspace Learning for Transductive Few-Shot Learning. In *CVPR*, 2022.

A Proofs

A.1 Proposition 1

Proof.

1. Invoking condition (4) of strict detectors gives

$$\begin{aligned}(g\circ f)(I_1+I_2) &= g(f(I_1+I_2) = g(f(I_1)+f(I_2)) &(23)\\ &= g(f(I_1))+g(f(I_2))) = (g\circ f)(I_1)+(g\circ f)(I_2) &(24)\end{aligned}$$

2. By conditions (4) and (5), we have

$$\begin{aligned}|(g\circ f)(I_1+I_2)| &= |g(f(I_1+I_2) = g(f(I_1)+f(I_2))| &(25)\\ &\leq |g(f(I_1))+g(f(I_2)))| = |(g\circ f)(I_1)+(g\circ f)(I_2)| &(26)\end{aligned}$$

□

A.2 Proposition 2

Proof. The proposition follows directly from convolutions being linear and translation equivariant. See *e.g.* (Jähne, 2002, Ch. 4). □

A.3 Proposition 3

Proof. Let $a_1 = I_1(k,x,y)$ and $a_2 = I_2(k,x,y)$. Since addition is commutative, we can assume $a_1 \geq a_2$ without loss of generality.

Observe that, if a_1 and a_2 is positive (negative), then $a_1 + a_2$ will be positive (negative). This means that $\text{LeakyReLU}_\alpha(a_1+a_2) = \text{LeakyReLU}_\alpha(a_1) + \text{LeakyReLU}_\alpha(a_2)$ in this case.

On the other hand, if $a_2 < 0 < a_1$, we have $|\text{LeakyReLU}_\alpha(a_1) + \text{LeakyReLU}_\alpha(a_2)| = |a_1 + \alpha a_2|$, and

$$|\text{LeakyReLU}_\alpha(a_1+a_2)| = \begin{cases}|a_1+a_2|, & |a_1|>|a_2|\\ \alpha|a_1+a_2|, & \text{otherwise}\end{cases}. \tag{27}$$

Since $\alpha \in [0,1)$, we have

$$|\text{LeakyReLU}_\alpha(a_1+a_2)| \leq |a_1+a_2| \leq |a_1+\alpha a_2| = |\text{LeakyReLU}_\alpha(a_1) + \text{LeakyReLU}_\alpha(a_2)| \tag{28}$$

□

A.4 Proposition 4

Proof. Setting $\frac{1}{WH} = \gamma_k$ gives

$$GAP(I)_k = \gamma_k \sum_{x=0}^{W-1}\sum_{y=0}^{H-1} I(k,x,y), \quad k\in\mathbb{N}^0_{<C} \tag{29}$$

from which it follows that

$$|GAP(I)_k| \leq \gamma_k \left|\sum_{x=0}^{W-1}\sum_{y=0}^{H-1} I(k,x,y)\right|, \quad k\in\mathbb{N}^0_{<C} \tag{30}$$

□

A.5 Theorem 1

Proof. Since f is a relaxed detector, we have

$$|f(I)| = \left| f\left(\sum_{j=0}^{C'-1} \sum_{(O,x',y')\in\mathcal{P}_j} T_{(x',y')}(O) \right) \right| \preccurlyeq \left| \sum_{j=0}^{C'-1} \sum_{(O,x',y')\in\mathcal{P}_j} f(T_{(x',y')}(O)) \right| \tag{31}$$

Then, since f is translation equivariant, and provides delta detections

$$|f(I)| \preccurlyeq \left| \sum_{j=0}^{C'-1} \sum_{(O,x',y')\in\mathcal{P}_j} f(T_{(x',y')}(O)) \right| = \left| \sum_{j=0}^{C'-1} \sum_{(i,x',y')\in\mathcal{P}_j} \delta_{(j,x',y')} \right|. \tag{32}$$

Applying a global pooling operator to $f(I)$ then gives

$$\begin{aligned}
|z_k| = |\,\mathrm{Pool}(f(I))_k| &\leq \gamma_k \left| \sum_{x=0}^{W'-1} \sum_{y=0}^{H'-1} f(I)(k,x,y) \right| && (33)\\
&\leq \gamma_k \left| \sum_{x=0}^{W'-1} \sum_{y=0}^{H'-1} \left(\sum_{j=0}^{C'-1} \sum_{(i,x',y')\in\mathcal{P}_j} \delta_{(j,x',y')}(k,x,y) \right) \right| && (34)\\
&\leq \gamma_k \left| \sum_{x=0}^{W'-1} \sum_{y=0}^{H'-1} \sum_{(i,x',y')\in\mathcal{P}_k} \delta_{(k,x',y')}(k,x,y) \right| && (35)\\
&\leq \gamma_k \left| \sum_{(i,x',y')\in\mathcal{P}_k} \delta_{(k,x',y')}(k,x',y') \right| && (36)\\
&\leq \gamma_k |\mathcal{P}_k| && (37)
\end{aligned}$$

□

A.6 Corollary 1.1

Proof. The L_p norm of $\mathbf{z}$ is defined as

$$||\mathbf{z}||_p = \left(\sum_{k=0}^{C'-1} |z_k|^p \right)^{\frac{1}{p}} \tag{38}$$

for $p > 0$. Since each $|z_k|$ is positive and upper bounded by $\gamma_k |\mathcal{P}_k|$ (by Theorem 1), we have

$$\sum_{k=0}^{C'-1} \gamma_k |\mathcal{P}_k|^p \geq \sum_{k=0}^{C'-1} |z_k|^p \tag{39}$$

which gives

$$\left(\sum_{k=0}^{C'-1} \gamma_k |\mathcal{P}_k|^p \right)^{\frac{1}{p}} \geq \left(\sum_{k=0}^{C'-1} |z_k|^p \right)^{\frac{1}{p}} = ||\mathbf{z}||_p \tag{40}$$

□

A.7 Corollary 1.2

Proof. This proof follows the same steps as the proof of Theorem 1, but without absolute values, and with equality instead of inequality.

Since f is a strict detector, we have

$$f(I) = f\left(\sum_{j=0}^{C'-1} \sum_{(O,x',y')\in\mathcal{P}_j} T_{(x',y')}(O)\right) = \sum_{j=0}^{C'-1} \sum_{(O,x',y')\in\mathcal{P}_j} f(T_{(x',y')}(O)) \tag{41}$$

Then, since f is translation equivariant, and provides delta detections

$$f(I) = \sum_{j=0}^{C'-1} \sum_{(O,x',y')\in\mathcal{P}_j} f(T_{(x',y')}(O)) = \sum_{j=0}^{C'-1} \sum_{(i,x',y')\in\mathcal{P}_j} \delta_{(j,x',y')}. \tag{42}$$

Applying a global pooling operator to $f(I)$ then gives

$$\begin{aligned}
z_k = \mathrm{GAP}(f(I))_k &= \frac{1}{W'H'} \sum_{x=0}^{W'-1} \sum_{y=0}^{H'-1} f(I)(k,x,y) && (43)\\
&= \frac{1}{W'H'} \sum_{x=0}^{W'-1} \sum_{y=0}^{H'-1} \left(\sum_{j=0}^{C'-1} \sum_{(i,x',y')\in\mathcal{P}_j} \delta_{(j,x',y')}(k,x,y)\right) && (44)\\
&= \frac{1}{W'H'} \sum_{x=0}^{W'-1} \sum_{y=0}^{H'-1} \sum_{(i,x',y')\in\mathcal{P}_k} \delta_{(k,x',y')}(k,x,y) && (45)\\
&= \frac{1}{W'H'} \sum_{(i,x',y')\in\mathcal{P}_k} \delta_{(k,x',y')}(k,x',y') && (46)\\
&= \frac{1}{W'H'} |\mathcal{P}_k| && (47)
\end{aligned}$$

□

Bibliography

[I] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. "Leveraging Tensor Kernels to Reduce Objective Function Mismatch in Deep Clustering". *Pattern Recognition* (2023). Under Review.

[II] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. "Reconsidering Representation Alignment for Multi-view Clustering". In: *CVPR*. 2021.

[III] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. "On the Effects of Self-supervision and Contrastive Alignment in Deep Multi-view Clustering". In: *CVPR*. Highlight. 2023.

[IV] Daniel J. Trosten, Rwiddhi Chakraborty, Sigurd Løkse, Kristoffer Wickstrøm, Robert Jenssen, and Michael Kampffmeyer. "Hubs and Hyperspheres: Reducing Hubness and Improving Transductive Few-shot Learning with Hyperspherical Embeddings". In: *CVPR*. 2023.

[V] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. "Norm-Count Hypothesis: On the Relationship Between Norm and Object Count in Visual Representations" (2023). In submission.

[6] Daniel J. Trosten, Robert Jenssen, and Michael C. Kampffmeyer. "Reducing Objective Function Mismatch in Deep Clustering with the Unsupervised Companion Objective". In: *NLDL*. 2021. DOI: `10.7557/18.5709`.

[7] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. "Reconsidering Representation Alignment for Multi-view Clustering". In: *National Conference on Image Processing and Machine Learning (NOBIM)*. Extended abstract and oral presentation. 2021.

[8] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. "Reconsidering Representation Alignment for Multi-view Clustering". In: *Visual Intelligence Days*. Extended abstract. 2021.

[9] Daniel J. Trosten. "Deep Clustering". Invited talk at University of Manitoba. 2021.

[10] Daniel J. Trosten, Kristoffer K. Wickstrøm, Shujian Yu, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. "Deep Clustering with the Cauchy-Schwarz Divergence". In: *AAAI Workshop on Information Theory for Deep Learning (IT4DL)*. Extended abstract and oral presentation. 2022.

[11] Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. "On the Role of Self-supervision in Deep Multi-view Clustering". In: *Visual Intelligence Days*. Poster presentation. 2022.

[12] Daniel J. Trosten. "Questionable Practices in Methodological Deep Learning Research". In: *NLDL*. 2023. DOI: 10.7557/18.6804.

[13] Kristoffer K. Wickstrøm, Daniel J. Trosten, Sigurd Løkse, Ahcène Boubekki, Karl Øyvind Mikalsen, Michael C. Kampffmeyer, and Robert Jenssen. "RELAX: Representation Learning Explainability". *International Journal of Computer Vision* (2023). DOI: 10.1007/s11263-023-01773-2.

[14] Daniel J. Trosten, Rwiddhi Chakraborty, Sigurd Løkse, Kristoffer Wickstrøm, Robert Jenssen, and Michael Kampffmeyer. "Hubs and Hyperspheres: Reducing Hubness and Improving Transductive Few-shot Learning with Hyperspherical Embeddings". In: *National Conference on Image Processing and Machine Learning (NOBIM)*. Extended abstract. 2023.

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning". *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[16] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. "Deep Clustering for Unsupervised Learning of Visual Features". In: *ECCV*. 2018. DOI: 10.1007/978-3-030-01264-9_9.

[17] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. "Emerging Properties in Self-Supervised Vision Transformers". In: *ICCV*. 2021. arXiv: 2104.14294.

[18] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, and Piotr Bojanowski. "Self-Supervised Pretraining of Visual Features in the Wild". *arXiv:2103.01988 [cs]* (2021). arXiv: 2103.01988 [cs].

[19] Alexander C. Li, Ellis Brown, Alexei A. Efros, and Deepak Pathak. *Internet Explorer: Targeted Representation Learning on the Open Web*. http://arxiv.org/abs/2302.14051. 2023. arXiv: 2302.14051 [cs].

[20] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stephane Deny. "Barlow Twins: Self-Supervised Learning via Redundancy Reduction". In: *ICML*. 2021.

[21] Adrien Bardes, Jean Ponce, and Yann LeCun. "VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning". In: *ICLR*. 2022.

[22] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum Contrast for Unsupervised Visual Representation Learning". In: *CVPR*. 2020. DOI: 10.1109/CVPR42600.2020.00975.

[23] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A Simple Framework for Contrastive Learning of Visual Representations". In: *ICML*. 2020.

[24] Priya Goyal, Quentin Duval, Isaac Seessel, Mathilde Caron, Ishan Misra, Levent Sagun, Armand Joulin, and Piotr Bojanowski. *Vision Models Are More Robust And Fair When Pretrained On Uncurated Images Without Supervision*. http://arxiv.org/abs/2202.08360. 2022. arXiv: 2202.08360 [cs].

[25] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. "SCAN: Learning to Classify Images Without Labels". In: *ECCV*. 2020. DOI: `10.1007/978-3-030-58607-2_16`.

[26] Xu Ji, Andrea Vedaldi, and Joao Henriques. "Invariant Information Clustering for Unsupervised Image Classification and Segmentation". In: *ICCV*. 2019. DOI: `10.1109/ICCV.2019.00996`.

[27] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. "High-Resolution Image Synthesis with Latent Diffusion Models". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022. DOI: `10.1109/CVPR52688.2022.01042`.

[28] Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. "BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling". In: *NeurIPS*. 2019. arXiv: `1902.02102`.

[29] Rewon Child. "Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images". In: *ICLR*. 2021.

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *NAACL-HLT*. 2019. DOI: `10.18653/v1/N19-1423`.

[31] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators". In: *ICLR*. 2019.

[32] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving Language Understanding by Generative Pre-Training" (2018).

[33] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language Models Are Unsupervised Multitask Learners" (2019).

[34] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models Are Few-Shot Learners". In: *NeurIPS*. 2020.

[35] OpenAI. *GPT-4 Technical Report*. `http://arxiv.org/abs/2303.08774`. 2023. arXiv: `2303.08774 [cs]`.

[36] "What's the next Word in Large Language Models?" *Nature Machine Intelligence* 5.4 (2023). Editorial, pp. 331–332. DOI: `10.1038/s42256-023-00655-z`.

[37] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. *LLaMA: Open and Efficient Foundation Language Models*. `http://arxiv.org/abs/2302.13971`. 2023. DOI: `10.48550/arXiv.2302.13971`. arXiv: `2302.13971 [cs]`.

[38] Rui-Jie Zhu, Qihang Zhao, and Jason K. Eshraghian. *SpikeGPT: Generative Pre-trained Language Model with Spiking Neural Networks*. http://arxiv.org/abs/2302.13939. 2023. arXiv: 2302.13939 [cs].

[39] Junyuan Xie, Ross Girshick, and Ali Farhadi. “Unsupervised Deep Embedding for Clustering Analysis”. In: *ICML*. 2016.

[40] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. “Towards K-Means-Friendly Spaces: Simultaneous Deep Learning and Clustering”. In: *ICML*. 2017.

[41] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. “Improved Deep Embedded Clustering with Local Structure Preservation”. In: *IJCAI*. 2017. DOI: 10.24963/ijcai.2017/243.

[42] Michael Kampffmeyer, Sigurd Løkse, Filippo M. Bianchi, Lorenzo Livi, Arnt-Børre Salberg, and Robert Jenssen. “Deep Divergence-based Approach to Clustering”. *Neural Networks* 113 (2019), pp. 91–101. DOI: 10.1016/j.neunet.2019.01.015.

[43] Zhenyu Huang, Joey Tianyi Zhou, Xi Peng, Changqing Zhang, Hongyuan Zhu, and Jiancheng Lv. “Multi-View Spectral Clustering Network”. In: *IJCAI*. 2019. DOI: 10.24963/ijcai.2019/356.

[44] Z. Li, Q. Wang, Z. Tao, Q. Gao, and Z. Yang. “Deep Adversarial Multi-view Clustering Network”. In: *IJCAI*. 2019. DOI: 10.24963/ijcai.2019/409.

[45] Runwu Zhou and Yi-Dong Shen. “End-to-End Adversarial-Attention Network for Multi-Modal Clustering”. In: *CVPR*. 2020. DOI: 10.1109/CVPR42600.2020.01463.

[46] Jie Xu, Yazhou Ren, Huayi Tang, Xiaorong Pu, Xiaofeng Zhu, Ming Zeng, and Lifang He. “Multi-VAE: Learning Disentangled View-Common and View-Peculiar Visual Representations for Multi-View Clustering”. In: *ICCV*. 2021. DOI: 10.1109/ICCV48922.2021.00910.

[47] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. “Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning”. In: *NeurIPS*. 2020.

[48] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. *Improved Baselines with Momentum Contrastive Learning*. http://arxiv.org/abs/2003.04297. 2020. DOI: 10.48550/arXiv.2003.04297. arXiv: 2003.04297 [cs].

[49] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. “Unsupervised Learning of Visual Features by Contrasting Cluster Assignments”. In: *NeurIPS*. 2020.

[50] Malik Boudiaf, Ziko Imtiaz Masud, Jérôme Rony, José Dolz, Pablo Piantanida, and Ismail Ben Ayed. “Transductive Information Maximization For Few-Shot Learning”. In: *NeurIPS*. 2020.

[51] Chaofan Chen, Xiaoshan Yang, Changsheng Xu, Xuhui Huang, and Zhe Ma. “ECKPN: Explicit Class Knowledge Propagation Network for Transductive Few-Shot Learning”. In: *CVPR*. 2021.

[52] Philip Chikontwe, Soopil Kim, and Sang Hyun Park. “CAD: Co-Adapting Discriminative Features for Improved Few-Shot Classification”. In: *CVPR*. 2022.

[53] Wentao Cui and Yuhong Guo. “Parameterless Transductive Feature Re-representation for Few-Shot Learning”. In: *ICML*. 2021.

[54] Guneet S. Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. “A Baseline for Few-Shot Image Classification”. In: *ICLR*. 2020.

[55] Nanyi Fei, Yizhao Gao, Zhiwu Lu, and Tao Xiang. “Z-Score Normalization, Hubness, and Few-Shot Learning”. In: *ICCV*. 2021. DOI: `10.1109/ICCV48922.2021.00021`.

[56] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Perez Perez, and Matthieu Cord. “Boosting Few-Shot Visual Learning With Self-Supervision”. In: *ICCV*. 2019. DOI: `10.1109/ICCV.2019.00815`.

[57] Michalis Lazarou, Tania Stathaki, and Yannis Avrithis. “Iterative Label Cleaning for Transductive and Semi-Supervised Few-Shot Learning”. In: *ICCV*. 2021.

[58] Duong H Le, Khoi D Nguyen, and Khoi Nguyen. “POODLE: Improving Few-shot Learning via Penalizing Out-of-Distribution Samples”. In: *NeurIPS*. 2021.

[59] Jie Ling, Lei Liao, Meng Yang, and Jia Shuai. “Semi-Supervised Few-Shot Learning via Multi-Factor Clustering”. In: *CVPR*. 2022.

[60] Yang Liu, Weifeng Zhang, Chao Xiang, Tu Zheng, Deng Cai, and Xiaofei He. “Learning To Affiliate: Mutual Centralized Learning for Few-Shot Classification”. In: *CVPR*. 2022.

[61] Jinlu Liu, Liang Song, and Yongqiang Qin. “Prototype Rectification for Few-Shot Learning”. In: *ECCV*. 2020. DOI: `10.1007/978-3-030-58452-8_43`.

[62] Puneet Mangla, Mayank Singh, Abhishek Sinha, Nupur Kumari, Vineeth N Balasubramanian, and Balaji Krishnamurthy. “Charting the Right Manifold: Manifold Mixup for Few-shot Learning”. In: *WACV*. 2020. DOI: `10.1109/WACV45572.2020.9093338`.

[63] Yuqing Ma, Shihao Bai, Wei Liu, Shuo Wang, Yue Yu, Xiao Bai, Xianglong Liu, and Meng Wang. “Transductive Relation-Propagation With Decoupling Training for Few-Shot Learning”. *IEEE Transactions on Neural Networks and Learning Systems* (2021), pp. 1–13. DOI: `10.1109/TNNLS.2021.3082928`.

[64] Guodong Qi, Huimin Yu, Zhaohui Lu, and Shuzhao Li. “Transductive Few-Shot Classification on the Oblique Manifold”. In: *ICCV*. 2021.

[65] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. “Meta-Learning for Semi-supervised Few-shot Classification”. In: *ICLR*. 2018.

[66] Xi SHEN, Yang Xiao, Shell Xu Hu, Othman Sbai, and Mathieu Aubry. “Re-Ranking for Image Retrieval and Transductive Few-Shot Classification”. In: *NeurIPS*. 2021.

[67] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B. Tenenbaum, and Phillip Isola. “Rethinking Few-Shot Image Classification: A Good Embedding Is All You Need?” In: *ECCV*. 2020. DOI: `10.1007/978-3-030-58568-6_16`.

[68] Olivier Veilleux, Malik Boudiaf, Pablo Piantanida, and Ismail Ben Ayed. “Realistic Evaluation of Transductive Few-Shot Learning”. In: *NeurIPS*. 2021.

[69] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. “Matching Networks for One Shot Learning”. In: *NeurIPS*. 2016.

[70] Yaqing Wang, Quanming Yao, James Kwok, and Lionel M. Ni. *Generalizing from a Few Examples: A Survey on Few-Shot Learning*. `http://arxiv.org/abs/1904.05046`. 2020. arXiv: `1904.05046`.

[71] Yan Wang, Wei-Lun Chao, Kilian Q. Weinberger, and Laurens van der Maaten. “SimpleShot: Revisiting Nearest-Neighbor Classification for Few-Shot Learning”. *arXiv:1911.04623 [cs]* (2019). arXiv: `1911.04623 [cs]`.

[72] Hao Zhu and Piotr Koniusz. “EASE: Unsupervised Discriminant Subspace Learning for Transductive Few-Shot Learning”. In: *CVPR*. 2022.

[73] Imtiaz Ziko, Jose Dolz, Eric Granger, and Ismail Ben Ayed. “Laplacian Regularized Few-Shot Learning”. In: *ICML*. 2020.

[74] Imtiaz Masud Ziko, Malik Boudiaf, Jose Dolz, Eric Granger, and Ismail Ben Ayed. *Transductive Few-Shot Learning: Clustering Is All You Need?* `http://arxiv.org/abs/2106.09516`. 2021. arXiv: `2106.09516 [cs]`.

[75] Jing Xu, Xu Luo, Xinglin Pan, Wenjie Pei, Yanan Li, and Zenglin Xu. “Alleviating the Sample Selection Bias in Few-shot Learning by Removing Projection to the Centroid”. In: *NeurIPS*. 2022. arXiv: `2210.16834 [cs]`.

[76] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation Learning: A Review and New Perspectives”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828. DOI: `10.1109/TPAMI.2013.50`.

[77] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. “Testing the Manifold Hypothesis”. *Journal of the American Mathematical Society* 29.4 (2016), pp. 983–1049. DOI: `10.1090/jams/852`.

[78] Guowang Du, Lihua Zhou, Yudi Yang, Kevin Lü, and Lizhen Wang. “Deep Multiple Auto-Encoder-Based Multi-view Clustering”. *Data Science and Engineering* 6.3 (2021), pp. 323–338. DOI: `10.1007/s41019-021-00159-z`.

[79] Yiqiao Mao, Xiaoqiang Yan, Qiang Guo, and Yangdong Ye. “Deep Mutual Information Maximin for Cross-Modal Clustering”. In: *IJCAI*. 2021.

[80] Qianqian Wang, Zhiqiang Tao, Wei Xia, Quanxue Gao, Xiaochun Cao, and Licheng Jiao. “Adversarial Multiview Clustering Networks With Adaptive Fusion”. *IEEE Transactions on Neural Networks and Learning Systems* (2022). DOI: `10.1109/TNNLS.2022.3145048`.

[81] Shiye Wang, Changsheng Li, Yanming Li, Ye Yuan, and Guoren Wang. "Self-Supervised Information Bottleneck for Deep Multi-View Subspace Clustering". *IEEE Transactions on Image Processing* 32 (2023), pp. 1555–1567. DOI: 10.1109/TIP.2023.3246802.

[82] Bowen Xin, Shan Zeng, and Xiuying Wang. "Self-Supervised Deep Correlational Multi-View Clustering". In: *IJCNN*. 2021. DOI: 10.1109/IJCNN52387.2021.9534345.

[83] Jie Xu, Chao Li, Liang Peng, Yazhou Ren, Xiaoshuang Shi, Heng Tao Shen, and Xiaofeng Zhu. "Adaptive Feature Projection With Distribution Alignment for Deep Incomplete Multi-View Clustering". *IEEE Transactions on Image Processing* 32 (2023), pp. 1354–1366. DOI: 10.1109/TIP.2023.3243521.

[84] Jie Xu, Yazhou Ren, Guofeng Li, Lili Pan, Ce Zhu, and Zenglin Xu. "Deep Embedded Multi-View Clustering with Collaborative Training". *Information Sciences* 573 (2021), pp. 279–290. DOI: 10.1016/j.ins.2020.12.073.

[85] Xianchao Zhang, Xiaorui Tang, Linlin Zong, Xinyue Liu, and Jie Mu. "Deep Multimodal Clustering with Cross Reconstruction". In: *PAKDD*. 2020. DOI: 10.1007/978-3-030-47426-3_24.

[86] Xianchao Zhang, Jie Mu, Linlin Zong, and Xiaochun Yang. "End-To-End Deep Multimodal Clustering". In: *ICME*. 2020. DOI: 10.1109/icme46284.2020.9102921.

[87] Linlin Zong, Faqiang Miao, Xianchao Zhang, and Bo Xu. "Multimodal Clustering via Deep Commonness and Uniqueness Mining". In: *ICIKM*. 2020. DOI: 10.1145/3340531.3412103.

[88] Linlin Zong, Faqiang Miao, Xianchao Zhang, Wenxin Liang, and Bo Xu. "Self-Supervised Deep Multiview Spectral Clustering". *IEEE Transactions on Neural Networks and Learning Systems* (2022), pp. 1–10. DOI: 10.1109/TNNLS.2022.3195780.

[89] Milosˇ Radovanovic, Alexandros Nanopoulos, and Mirjana Ivanovic. "Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data". *JMLR* (2010), p. 45.

[90] Marco Signoretto. "Kernels and Tensors for Structured Data Modelling". PhD thesis. Katholieke Universiteit Leuven – Faculty of Engineering, 2011.

[91] Matthew Hutson. "Artificial Intelligence Faces Reproducibility Crisis". *Science* 359.6377 (2018), pp. 725–726. DOI: 10.1126/science.359.6377.725.

[92] Edward Raff. "A Step Toward Quantifying Independently Reproducible Machine Learning Research". In: *NeurIPS*. 2019.

[93] Jose C. Principe. *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*. New York, NY: Springer, 2010. ISBN: 978-1-4419-1569-6 978-1-4419-1570-2. DOI: 10.1007/978-1-4419-1570-2.

[94] R. Linsker. "Self-Organization in a Perceptual Network". *Computer* 21.3 (1988), pp. 105–117. DOI: 10.1109/2.36.

[95] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. "Learning Deep Representations by Mutual Information Estimation and Maximization". In: *ICLR*. 2019.

[96] Naftali Tishby, Fernando C Pereira, and William Bialek. "The Information Bottleneck Method". In: *Annual Allerton Conference on Communication, Control and Computing*. 1999.

[97] James Mercer. "XVI. Functions of Positive and Negative Type, and Their Connection the Theory of Integral Equations". *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 209.441-458 (1909), pp. 415–446. DOI: 10.1098/rsta.1909.0016.

[98] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. 4th ed. Burlington, MA London: Academic Press, 2009. ISBN: 978-1-59749-272-0.

[99] Bernhard Schölkopf and Alexander Johannes Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Reprint. Cambridge, Mass.: MIT Press, 2002. ISBN: 978-0-262-53657-8 978-0-262-19475-4.

[100] Robert Jenssen and Torbjørn Eltoft. "A New Information Theoretic Analysis of Sum-of-Squared-Error Kernel Clustering". *Neurocomputing* 72.1 (2008), pp. 23–31. DOI: 10.1016/j.neucom.2008.03.017.

[101] R. Jenssen. "Kernel Entropy Component Analysis". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.5 (2010), pp. 847–860. DOI: 10.1109/TPAMI.2009.100.

[102] Vidar V. Vikjord and Robert Jenssen. "Information Theoretic Clustering Using a K-Nearest Neighbors Approach". *Pattern Recognition* 47.9 (2014), pp. 3070–3081. DOI: 10.1016/j.patcog.2014.03.018.

[103] Emanuel Parzen. "On Estimation of a Probability Density Function and Mode". *The Annals of Mathematical Statistics* 33.3 (1962), pp. 1065–1076. DOI: 10.1214/aoms/1177704472.

[104] Robert Jenssen, Jose C. Principe, Deniz Erdogmus, and Torbjørn Eltoft. "The Cauchy–Schwarz Divergence and Parzen Windowing: Connections to Graph Theory and Mercer Kernels". *Journal of the Franklin Institute* 343.6 (2006), pp. 614–629. DOI: 10.1016/j.jfranklin.2006.03.018.

[105] Luis Gonzalo Sanchez Giraldo, Murali Rao, and Jose C. Principe. "Measures of Entropy From Data Using Infinitely Divisible Kernels". *IEEE Transactions on Information Theory* 61.1 (2015), pp. 535–548. DOI: 10.1109/TIT.2014.2370058.

[106] Shujian Yu, Luis Sanchez Giraldo, Robert Jenssen, and Jose Principe. "Multivariate Extension of Matrix-Based Rényi's α-Order Entropy Functional". *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (2019), pp. 1–1. DOI: 10.1109/TPAMI.2019.2932976.

[107] Mikhail Belkin and Partha Niyogi. "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation". *Neural Computation* 15 (2002), pp. 1373–1396. DOI: 10.1162/089976603321780317.

[108] Laurens van der Maaten and Geoffrey Hinton. “Visualizing Data Using T-SNE”. *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[109] Leland McInnes, John Healy, and James Melville. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”. *arXiv:1802.03426 [cs, stat]* (2018). arXiv: `1802.03426 [cs, stat]`.

[110] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Online. MIT Press, 2016.

[111] Warren S. McCulloch and Walter Pitts. “A Logical Calculus of the Ideas Immanent in Nervous Activity”. *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133. DOI: `10.1007/BF02478259`.

[112] F. Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.” *Psychological Review* 65.6 (1958), pp. 386–408. DOI: `10.1037/h0042519`.

[113] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning Representations by Back-Propagating Errors”. *Nature* 323.6088 (1986), pp. 533–536. DOI: `10.1038/323533a0`.

[114] G. Cybenko. “Approximation by Superpositions of a Sigmoidal Function”. *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314. DOI: `10.1007/BF02551274`.

[115] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer Feedforward Networks Are Universal Approximators”. *Neural Networks* 2.5 (1989), pp. 359–366. DOI: `10.1016/0893-6080(89)90020-8`.

[116] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. “The Expressive Power of Neural Networks: A View from the Width”. In: *NeurIPS*. 2017.

[117] Vitaly Maiorov and Allan Pinkus. “Lower Bounds for Approximation by MLP Neural Networks”. *Neurocomputing* 25.1 (1999), pp. 81–91. DOI: `10.1016/S0925-2312(98)00111-8`.

[118] Yann Lecun. “Generalization and network design strategies”. *Connectionism in perspective* (1989).

[119] Michael Thompson, Richard O. Duda, and Peter E. Hart. “Pattern Classification and Scene Analysis”. *Leonardo* 7.4 (1974), p. 370. DOI: `10.2307/1573081`. JSTOR: `1573081`.

[120] Bernd Jähne. *Digital image processing*. 5., rev. and extended. Berlin Heidelberg: Springer, 2002. ISBN: 978-3-540-67754-3.

[121] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *NeurIPS*. 2012.

[122] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *CVPR*. 2016. DOI: `10.1109/CVPR.2016.90`.

[123] Haskell B. Curry. “The Method of Steepest Descent for Non-Linear Minimization Problems”. *Quarterly of Applied Mathematics* 2.3 (1944), pp. 258–261. DOI: `10.1090/qam/10667`.

[124] Yuri Nesterov. “A Method for Unconstrained Convex Minimization Problem with the Rate of Convergence $o(1/K^2)$”. *Doklady AN USSR* 269 (1983), pp. 543–547.

[125] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. *Journal of Machine Learning Research* 12 (2011), pp. 2121–2159.

[126] Matthew D. Zeiler. “ADADELTA: An Adaptive Learning Rate Method”. *arXiv:1212.5701 [cs]* (2012). arXiv: `1212.5701 [cs]`.

[127] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *ICLR*. 2015.

[128] Timothy Dozat. “Incorporating Nesterov Momentum into Adam”. In: *ICLR, Workshop Track*. 2016.

[129] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *ICLR*. 2019.

[130] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. “On the Variance of the Adaptive Learning Rate and Beyond”. In: *ICLR*. 2020.

[131] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. *Symbolic Discovery of Optimization Algorithms*. `http://arxiv.org/abs/2302.06675`. 2023. arXiv: `2302.06675 [cs]`.

[132] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *AISTATS*. 2011.

[133] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *ICML*. 2015.

[134] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. “How Does Batch Normalization Help Optimization?” In: *NeurIPS*. 2018.

[135] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. “Self-Labelling via Simultaneous Clustering and Representation Learning”. In: *ICLR*. 2020. arXiv: `1911.05371`.

[136] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. “Unsupervised Pre-Training of Image Features on Non-Curated Data”. In: *ICCV*. 2019. arXiv: `1905.01278`.

[137] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. “Big Self-Supervised Models Are Strong Semi-Supervised Learners”. In: *NeurIPS*. 2020.

[138] Xinlei Chen and Kaiming He. “Exploring Simple Siamese Representation Learning”. *arXiv:2011.10566 [cs]* (2020). arXiv: `2011.10566 [cs]`.

[139] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. “Unsupervised Visual Representation Learning by Context Prediction”. In: *ICCV*. 2015.

[140] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. “Discriminative Unsupervised Feature Learning with Convolutional Neural Networks”. *NeurIPS* (2014), p. 9.

[141] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, and S M Ali Eslami. "Data-Efficient Image Recognition with Contrastive Predictive Coding". In: *ICML*. 2020.

[142] Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. "Self-Supervised Label Augmentation via Input Transformations". In: *ICML*. 2020.

[143] Ishan Misra and Laurens van der Maaten. "Self-Supervised Learning of Pretext-Invariant Representations". In: *CVPR*. 2020.

[144] Yuhui Quan, Mingqin Chen, Tongyao Pang, and Hui Ji. "Self2Self With Dropout: Learning Self-Supervised Denoising From Single Image". In: *CVPR*. 2020. DOI: `10.1109/CVPR42600.2020.00196`.

[145] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. "On Mutual Information Maximization for Representation Learning". In: *ICLR*. 2020.

[146] Tongzhou Wang and Phillip Isola. "Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere". In: *ICML*. 2020. arXiv: `2005.10242`.

[147] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised Representation Learning by Predicting Image Rotations". In: *ICLR*. 2018.

[148] Geoffrey E Hinton and Richard Zemel. "Autoencoders, Minimum Description Length and Helmholtz Free Energy". In: *NeurIPS*. 1993.

[149] H. Bourlard and Y. Kamp. "Auto-Association by Multilayer Perceptrons and Singular Value Decomposition". *Biological Cybernetics* 59.4 (1988), pp. 291–294. DOI: `10.1007/BF00332918`.

[150] M. A. Kramer. "Autoassociative Neural Networks". *Computers & Chemical Engineering* 16.4 (1992), pp. 313–328. DOI: `10.1016/0098-1354(92)80051-A`.

[151] Mark A. Kramer. "Nonlinear Principal Component Analysis Using Autoassociative Neural Networks". *AIChE Journal* 37.2 (1991), pp. 233–243. DOI: `10.1002/aic.690370209`.

[152] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. "Extracting and Composing Robust Features with Denoising Autoencoders". In: *ICML*. 2008. DOI: `10.1145/1390156.1390294`.

[153] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". *JMLR* 11.Dec (2010), pp. 3371–3408.

[154] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *ICLR*. 2014.

[155] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. "Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *ICLR*. 2017.

[156] Arash Vahdat and Jan Kautz. "NVAE: A Deep Hierarchical Variational Autoencoder". In: *NeurIPS*. 2020.

[157] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. “Neural Discrete Representation Learning”. In: *NeurIPS*. 2017.

[158] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. “Context Encoders: Feature Learning by Inpainting”. In: *CVPR*. 2016. DOI: 10.1109/CVPR.2016.278.

[159] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollar, and Ross Girshick. “Masked Autoencoders Are Scalable Vision Learners”. In: *CVPR*. 2022. DOI: 10.1109/CVPR52688.2022.01553.

[160] Yabo Chen, Yuchen Liu, Dongsheng Jiang, Xiaopeng Zhang, Wenrui Dai, Hongkai Xiong, and Qi Tian. “SdAE: Self-distillated Masked Autoencoder”. In: *ECCV*. 2022. DOI: 10.1007/978-3-031-20056-4_7.

[161] Luke Metz, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein. “Meta-Learning Update Rules for Unsupervised Representation Learning”. In: *ICLR*. 2019.

[162] N. Mrabah, M. Bouguessa, and R. Ksantini. “Adversarial Deep Embedded Clustering: On a Better Trade-off between Feature Randomness and Feature Drift”. *IEEE Transactions on Knowledge and Data Engineering* (2020), pp. 1–1. DOI: 10.1109/TKDE.2020.2997772.

[163] Nairouz Mrabah, Naimul Mefraz Khan, Riadh Ksantini, and Zied Lachiri. “Deep Clustering with a Dynamic Autoencoder: From Reconstruction towards Centroids Construction”. *Neural Networks* 130 (2020), pp. 206–228. DOI: 10.1016/j.neunet.2020.07.005.

[164] Adnan Khan, Sarah AlBarri, and Muhammad Arslan Manzoor. “Contrastive Self-Supervised Learning: A Survey on Different Architectures”. In: *ICAI*. 2022. DOI: 10.1109/ICAI55435.2022.9773725.

[165] J. MacQueen. “Some Methods for Classification and Analysis of Multivariate Observations”. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. 1967.

[166] Joe H. Ward. “Hierarchical Grouping to Optimize an Objective Function”. *Journal of the American Statistical Association* 58.301 (1963), pp. 236–244. DOI: 10.1080/01621459.1963.10500845.

[167] A. K. Jain, M. N. Murty, and P. J. Flynn. “Data Clustering: A Review”. *ACM Computing Surveys* 31.3 (1999), pp. 264–323. DOI: 10.1145/331499.331504.

[168] Anil K. Jain. “Data Clustering: 50 Years beyond K-means”. *Pattern Recognition Letters* 31.8 (2010), pp. 651–666. DOI: 10.1016/j.patrec.2009.09.011.

[169] A. H. S. Solberg, T. Taxt, and A. K. Jain. “A Markov Random Field Model for Classification of Multisource Satellite Imagery”. *IEEE Transactions on Geoscience and Remote Sensing* 34.1 (1996), pp. 100–113. DOI: 10.1109/36.481897.

[170] T. Taxt and A. Lundervold. “Multispectral Analysis of the Brain Using Magnetic Resonance Imaging”. *IEEE Transactions on Medical Imaging* 13.3 (Sept./1994), pp. 470–481. DOI: 10.1109/42.310878.

[171] Hong Hai Nguyen and Paul Cohen. “Gibbs Random Fields, Fuzzy Clustering, and the Unsupervised Segmentation of Textured Images”. *CVGIP: Graph. Models Image Process.* 55.1 (1993), pp. 1–19. DOI: `10.1006/cgip.1993.1001`.

[172] A. Rosenfeld, Han Huang, and V. Schneider. “An Application of Cluster Detection to Text and Picture Processing”. *IEEE Transactions on Information Theory* 15.6 (1969), pp. 672–681. DOI: `10.1109/TIT.1969.1054378`.

[173] Alex Graves, Abdel-Rahman Mohamed, and Geoffrey Hinton. “Speech Recognition with Deep Recurrent Neural Networks”. In: *International Conference on Acoustics, Speech and Signal Processing.* 2013. DOI: `10.1109/ICASSP.2013.6638947`.

[174] Yazhou Ren, Ni Wang, Mingxia Li, and Zenglin Xu. “Deep Density-Based Image Clustering”. *Knowledge-Based Systems* 197 (2020), p. 105841. DOI: `10.1016/j.knosys.2020.105841`.

[175] Séverine Affeldt, Lazhar Labiod, and Mohamed Nadif. “Spectral Clustering via Ensemble Deep Autoencoder Learning (SC-EDAE)”. *Pattern Recognition* 108 (2020), p. 107522. DOI: `10.1016/j.patcog.2020.107522`.

[176] Ahcène Boubekki, Michael Kampffmeyer, Ulf Brefeld, and Robert Jenssen. “Joint Optimization of an Autoencoder for Clustering and Embedding”. *Machine Learning* 110.7 (2021), pp. 1901–1937. DOI: `10/gnfpzk`.

[177] Jinyu Cai, Shiping Wang, Chaoyang Xu, and Wenzhong Guo. “Unsupervised Deep Clustering via Contractive Feature Representation and Focal Loss”. *Pattern Recognition* 123 (2022), p. 108386. DOI: `10.1016/j.patcog.2021.108386`.

[178] Hu Lu, Chao Chen, Hui Wei, Zhongchen Ma, Ke Jiang, and Yingquan Wang. “Improved Deep Convolutional Embedded Clustering with Re-Selectable Sample Training”. *Pattern Recognition* 127 (2022), p. 108611. DOI: `10.1016/j.patcog.2022.108611`.

[179] Mahdi Abavisani, Dimitris N Metaxas, Alireza Naghizadeh, and Vishal M Patel. “Deep Subspace Clustering with Data Augmentation”. In: *NeurIPS.* 2020.

[180] Sangwon Baek, Gangjoon Yoon, Jinjoo Song, and Sang Min Yoon. “Deep Self-Representative Subspace Clustering Network”. *Pattern Recognition* 118 (2021), p. 108041. DOI: `10.1016/j.patcog.2021.108041`.

[181] Xiaopeng Li, Zhourong Chen, Leonard K. M. Poon, and Nevin L. Zhang. “Learning Latent Superstructures in Variational Autoencoders for Deep Multidimensional Clustering”. In: *ICLR.* 2019.

[182] Linxiao Yang, Ngai-Man Cheung, Jiaying Li, and Jun Fang. “Deep Clustering by Gaussian Mixture Variational Autoencoders With Graph Embedding”. In: *ICCV.* 2019. DOI: `10.1109/ICCV.2019.00654`.

[183] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative Adversarial Nets”. In: *NeurIPS.* 2014.

[184] Qiang Yang, Yu Zhang, Wenyuan Dai, and Sinno Jialin Pan. "Few-Shot Learning". In: *Transfer Learning*. First. Cambridge University Press, 2020. ISBN: 978-1-139-06177-3 978-1-107-01690-3. DOI: `10.1017/978113906177 3`.

[185] Steinar Laenen and Luca Bertinetto. "On Episodes, Prototypical Networks, and Few-Shot Learning". In: *NeurIPS*. 2021.

[186] Guangcan Liu, Zhouchen Lin, and Yong Yu. "Robust Subspace Segmentation by Low-Rank Representation". In: *ICML*. 2010.

[187] "The Hubness Phenomenon in High-Dimensional Spaces". *Research in Data Science* 17 (2019). DOI: `10.1007/978-3-030-11566-1`.

[188] Dominik Schnitzer, Arthur Flexer, Markus Schedl, and Gerhard Widmer. "Local and Global Scaling Reduce Hubs in Space". *Journal of Machine Learning Research* 13 (2012), p. 32.

[189] Kazuo Hara, Ikumi Suzuki, Masashi Shimbo, Kei Kobayashi, Kenji Fukumizu, and Milos Radovanovic. "Localized Centering: Reducing Hubness in Large-Sample Data". In: *AAAI*. 2015. DOI: `10.1609/aaai.v29i1.9629`.

[190] Ikumi Suzuki, Kazuo Hara, Masashi Shimbo, Marco Saerens, and Kenji Fukumizu. "Centering Similarity Measures to Reduce Hubs". In: *EMNLP*. 2013.

[191] Kazuo Hara, Ikumi Suzuki, Kei Kobayashi, Kenji Fukumizu, and Milos Radovanovic. "Flattening the Density Gradient for Eliminating Spatial Centrality to Reduce Hubness". In: *AAAI*. 2016. DOI: `10.1609/aaai.v30 i1.10240`.

[192] Thomas Low, Christian Borgelt, Sebastian Stober, and Andreas Nürnberger. "The Hubness Phenomenon: Fact or Artifact?" In: *Towards Advanced Data Analysis by Combining Soft Computing and Statistics*. Vol. 285. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 267–278. ISBN: 978-3-642-30277-0 978-3-642-30278-7. DOI: `10.1007/978-3-64 2-30278-7_21`.

[193] Miloš Radovanović. "Hubs in Nearest-Neighbor Graphs: Origins, Applications and Challenges". In: *WIMS*. 2018. DOI: `10.1145/3227609.3227691`.

[194] Adam Coates, Honglak Lee, and Andrew Y Ng. "An Analysis of Single-Layer Networks in Unsupervised Feature Learning". In: *AISTATS*. 2011.

[195] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. *A Cookbook of Self-Supervised Learning*. `http://arxiv.org/abs/2304.12210`. 2023. arXiv: `2304.12210 [cs]`.

[196] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. "Learning Convolutional Neural Networks for Graphs". In: *ICML*. 2016.

[197] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *NeurIPS*. 2016.

[198] Thomas N Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *ICLR*. 2017.

[199] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *NeurIPS*. 2017.

[200] Stefanos Zafeiriou, Michael Bronstein, Taco Cohen, Oriol Vinyals, Le Song, Jure Leskovec, Pietro Liò, Joan Bruna, and Marco Gori. "Guest Editorial: Non-Euclidean Machine Learning". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.2 (2022), pp. 723–726. DOI: 10.1109/TPAMI.2021.3129857.

[201] Stefan Sommer and Alex Bronstein. "Horizontal Flows and Manifold Stochastics in Geometric Deep Learning". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.2 (2022), pp. 811–822. DOI: 10.1109/TPAMI.2020.2994507.

[202] Monami Banerjee, Rudrasis Chakraborty, Jose Bouza, and Baba C. Vemuri. "VolterraNet: A Higher Order Convolutional Network With Group Equivariance for Homogeneous Manifolds". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.2 (2022), pp. 823–833. DOI: 10.1109/TPAMI.2020.3035130.

[203] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. "Geometric Deep Learning: Going beyond Euclidean Data". *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42. DOI: 10.1109/MSP.2017.2693418.

[204] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M. Bronstein. "Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs". In: *CVPR*. 2017. DOI: 10.1109/CVPR.2017.576.

[205] Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. "Geodesic Convolutional Neural Networks on Riemannian Manifolds". In: *ICCVW*. 2015. DOI: 10.1109/ICCVW.2015.112.

[206] Pascal Mettes, Elise van der Pol, and Cees G. M. Snoek. "Hyperspherical Prototype Networks". In: *NeurIPS*. 2019.

[207] Tyler R. Scott, Andrew C. Gallagher, and Michael C. Mozer. "Von Mises-Fisher Loss: An Exploration of Embedding Geometries for Supervised Learning". In: *ICCV*. 2021.

[208] Weiyang Liu, Rongmei Lin, Zhen Liu, Li Xiong, Bernhard Scholkopf, and Adrian Weller. "Learning with Hyperspherical Uniformity". In: *AISTATS*. 2021.

[209] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. *Zero-Shot Text-to-Image Generation*. http://arxiv.org/abs/2102.12092. 2021. DOI: 10.48550/arXiv.2102.12092. arXiv: 2102.12092 [cs].

[210] Kawin Ethayarajh. "How Contextual Are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings". In: *EMNLP*. 2019. DOI: 10.18653/v1/D19-1006.

[211] Ravid Shwartz-Ziv and Yann LeCun. *To Compress or Not to Compress-Self-Supervised Learning and Information Theory: A Review.* `http://arxiv.org/abs/2304.09355`. 2023. arXiv: `2304.09355 [cs, math]`.

[212] The AI Act. *Proposal for a Regulation of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative Acts.* `https://artificialintelligenceact.eu/the-act/`. Accessed 05.05.2023. 2021.