UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science

Label Propagation in Machine Learning Systems

Providing End-to-End Traceability with Explainable Artificial Intelligence

Marius Johan Ingebrigtsen

Master’s thesis in Computer Science, INF-3990– May 15, 2024

Supervisors

Main supervisor:	Elisavet Kozyri	University in Tromsø, Faculty of Science and Technology, Department of Computer Science
Co-supervisor:	Anders Tungeland Gjerdrum	University in Tromsø Faculty of Science and Technology, Department of Computer Science

This thesis document was typeset using the *UiT Thesis LaTeX Template*.

To ChatGPT (by ChatGPT).

“ I nod with begrudging appreciation for your
unwavering assistance and seemingly infinite wisdom.
Because who needs human intellect when you
have a language model that knows everything?
Right? ”

“ All models are wrong,
some are useful. ”
–George Box

“ Di satans rustmærr!
Gjorde æ mi rætt så røska æ dæ uinna akslingen
og ranga dæ med ræva førre over rækka
så du søkk så langt ned i det djupaste hælvete
at han Gammel Erik kunne brukt dæ til separator! ”
– An extract from a North-Norwegian boat-mechanic’s
angry rant directed at a malfunctioning engine.
Also, an appropriate analogy for working with machine learning models.

Abstract

Artificial Intelligence (AI) and the underlying Machine Learning (ML) technology is experiencing increased applications in various areas. The training of ML models requires significant amounts of data, and data might contain restrictions regarding their permitted usage. High-performant models are often called black-boxes because of their complex decision-making process. Thus, ML applications threaten compliance with data restrictions by the lack of explainability with this technology.

Data labels can enforce data restrictions in a system's computational pipeline by being propagated from input to procedure output. A Label Propagation Mechanism (LPM) can employ an influence-based policy to propagate labels of input data that contribute towards the computation of the output. However, the application of influence-based label propagation in ML faces challenges due to the complete cross-taint of information inside these models.

This thesis proposes an influence-based LPM that employs explanations from Explainable Artificial Intelligence (XAI) to propagate input labels to ML outputs. This thesis concerns the proof of concept regarding the application of XAI to propagate to the output of a black-box ML model only the labels of inputs that have a high influence to that output. We first bridge the gap between conventional label propagation and the problematic application in ML. We then detail how LPMs can use XAI explanations to inform their label propagation. Next, we design and execute experiments with different XAI methods, models, and data. We evaluate the results based on the propagated labels and the faithfulness of the explanations for the model output.

Acknowledgements

To the following I owe a sincere and honorable mention.

Øyvind Arne Moen Nohr.
For the friendship. For the devotion to everything that is nerdy. For the proof that dedication beats talent. For your impeccably wise attitude. My "better half"; a friend like you is a rare treasure.

Henrik Monsen.
For your calming aura. For the flawless hairstyle. For your dauntless will to argue. You've lifted my spirit the last months like Sam carried Frodo.

Moritz Jörg.
For your ceaseless quest for knowlegde. For educating us as a by-product. For your immensely optimistic approach. People should count themselves lucky in your presence.

Ole Alexander Berg and **Julia Mikhailova**.
For creating your home. For inviting me in the very next second. For providing me with a safe haven and countless samaritan deeds. You're the aspiration for true family.

Eliza Kozyri.
For your inclusive nature. For your meticulous mind. For accepting me into your research. For keeping me grounded and sane throughout the thesis. I truly cherish the experience working with you.

Anders Tungeland Gjerdrum.
For your impressive devotion to science. For the continuous enlightenment. For the constant patience. Your attention has been akin to the sun shining just for me.

The Cyber Security Research Group:
Dag Johansen, **Håvard Dagenborg**, **Robert Pettersen**, **Thomas Nilsen**, **Enrico Tedeschi**, **Tor-Arne Nordmo**, and **Aril Ovesen**.

For the vigilant and early integration of a novice bachelor student. For the encouragement towards the academic pursuit. The metaphor "*On the shoulders of giants*" best describes the respect I harbor, now and forwards.

UiT's Department of Computer Science.
For the loyal support towards the student mass. For letting me teach what I've been taught. I can't imagine a better place to live and learn.

My fellow students.
For every exercise, every assignment, every exam, every course, and other stressful obstacle we've endured and overcome together. My sincere gratitude and good luck to you all.

Mom and Dad:
For teaching me rigid and fair principles to live by. For the thirty years of joyous purpose of life. I've brought you many works of art from kindergarten before, and I hope this thesis can top them all. The many things I do in life is to earn the right to be as luck as I am with you.

Contents

List of Figures

List of Tables

Thesis Definitions

List of Abbreviations

ACM Association for Computing Machinery

ADAM Adaptive Momentum Estimation

AE Auto-Encoder

AHC Agglomerative Hierarchical Clustering

AI Artificial Intelligence

ALTAI The European Commission's Assessment List for Trustworthy Artificial Intelligence

API Application Programming Interface

CIA Confidentiality, Integrity, and Availability

CNN Convolutional Neural Network

CPU Central Processing Unit

CSG Cyber Security Group

CSV Comma-Separated Values

DAG Directed Acyclic Graph

DeepLIFT Deep Learning Important FeaTures

DL Deep Learning

DLIME Deterministic Local Interpretable Model-Agnostic Explanations

DNN Deep Neural Network

DNR Deep Residual Network

FaaS Function as a Service

FCL Fully Connected Layer

FCN Fully-Connected Network

FF Feed-Forward

FFN Feed-Forward Network

GAN Generative Adversarial Network

GB GigaBytes

GBP Guided Back-Propagation

GDPR General Data Protection Regulation

GHZ GigaHertz

GPU Graphics Processing Unit

HTML HyperText Markup Language

IFC Information Flow Control

IG Integrated Gradients

ILSVRC ImageNet Large Scale Visual Recognition Challenge

IMDb Internet Movie Database

IOT Internet of Things

JPEG Joint Photographic Experts Group

KNN K Nearest Neighbors

LBAC Lattice-Based Access Control

LIME Local Interpretable Model-agnostic Explanations

LLAMA Large Language Model Meta AI

LLM Large Language Models

LLM Large Language Model

LORA Low-Rank Adaptation

LPM Label Propagation Mechanism

LRP Layer-wise Relevance Propagation

MB MegaBytes

MHA Multi-Head Attention

ML Machine Learning

MLP Multi-Layer Perceptron

NLP Natural Language Processing

NN Neural Network

OOM Out Of Memory

OS Operating System

PCA Principal Component Analysis

RAM Random Access Memory

ResNet Residual Network

RGB Red Green Blue

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

SLIC Simple Linear Iterative Clustering

TEE Trusted Execution Environment

UiT University of Tromsø

VGG16 Visual Geometry Group 16 layers

WSL2 Windows Subsystem for Linux 2

XAI Explainable Artificial Intelligence

/1

Introduction

Artificial Intelligence (AI) is witnessing unprecedented adoption across various domains with the proliferation of increasingly sophisticated models. This sophistication goes hand-in-hand with creating more complex Machine Learning (ML) architectures, which often require vast and varied datasets. Consequently, this data is the source of compliance and explainability issues for AI systems because of the information's potential sensitivity. Privacy preservation and confidential leakage prevention for data are examples of sensitive data. This chapter's outline is as follows:

Section 1.1 identifies the high-level problems and challenges, and bootstraps them to real world applications.

Section 1.2 highlights the adjacent goals external to this thesis and clarifies our scientific perspective on the problem statement.

Section 1.3 states the questions regarding the problem statement that this thesis aims to answer.

Section 1.4 describes the high-level means applied to answer the research questions and the scientific value of this thesis.

Section 1.5 frames the thesis methodology in terms of computer science as a scientific discipline.

Section 1.6 provides a brief description and overview of the thesis' contents.

1.1 Problem Statement

Systems striving for compliance often impose restrictions regarding the use of data, and ML algorithms exacerbate the risk of violating these restrictions since the violations are not necessarily apparent. Two examples of such restrictions are the *Right to Erasure* from General Data Protection Regulation (GDPR) and *copyright infringement*, where breaches to these restrictions can have severe consequences. Systems compliant with the right to erasure should delete user identifying data and consequential by-products in ML models, and violations of GDPR can result in substantial fines [76, EU, 2016]. Generative models might violate legal regulations on intellectual property by recreating patented content, and such violations can lead to legal disputes [83, Zirpoli (US CRS), 2023].

Data labels can embody policy restrictions. Labels should propagate to outputs computed with labeled data to enforce the imposed restrictions. Labels are metadata containing information about the associated data and are agnostic to the propagation policy. This way, the system can enforce any restrictions on data derivatives via their propagated labels. Label propagation policies confront the impediment *given labeled data as input to a procedure, what should be the output's labels*? Propagating any label of influential data for the output of the computation is an intuitive approach, alternatively referred to as an *influence-based* propagation policy.

Label propagation faces challenges with state-of-the-art ML because the inner workings of advanced models become black-boxes due to their intricate architectures. Black-box models are considered black-boxes because constructing understandable explanations of the model's decision is a complicated matter. The black-box problem relates to a lack of explainability. This complication arises due to every input affecting everything inside the model and there being a complete cross-taint between inputs in the computation of the model outputs. Thus, models are black-boxes in the sense that investigating their creational or computational procedures yields no intuitive explanation for either how the model works or why the model makes one specific decision [46, Molnar, 2022, chapter 2.3]. The cross-tainting property is an obstacle to label propagation strategies based on discerning which input factors contribute to the output and which do not. Therefore, applying conventional influence-based label propagation policies on black-box models would mean every input label should propagate to every output. The consequence of such policies in ML models is poor scalability and unnecessary label propagation.

An influence-based policy propagates every contributing data's label to the model's output. However, this approach does not scale and is a conservative assumption that might not always be true. Consider the model $f(x) := x_1 + x_m$ that is treated as a black-box where the input is a sequence $x = \langle x_1, x_2, ..., x_m \rangle \mid m \in \mathbb{Z}^+$. Propagating every label to the output $f(x)$ scales poorly because the number of output labels grows linearly with m and should include the model's training set labels as well. Training sets contain a significant number of data samples n such that $n \gg m$, i.e., there are far more data points than model inputs [73, Theodoridis et al., chapter 2.5, page 55]. Similarly, propagating every label is conservative because the data points $x_i \; \forall \; i \in [2, m-1]$ do not contribute and, therefore, does not influence the output $f(x)$.

End-to-end traceability is the consecutive label propagation from input to output in a system and should enable tracing of the output provenance back to the input. Any system with consecutive label propagation cannot provide traceability if there is a ML component where the labeling policy propagates everything. The number of labels would not be helpful in reasoning over the output's provenance. To our knowledge, previous work has yet to address this problem.

1.1.1 Problem Significance

AI is looking to become as central and widespread in our everyday lives as having smartphones or access to electricity. The areas of applicability seem limitless, and there is rapid adoption and development in business, research, and open-source communities. Many new AI companies are founded purely on the newest developments in Large Language Models (LLM) (otherwise known as AI chat-bots), such as Mistral, OpenAI, and Anthropic. On the other hand, established companies, such as Amazon, Google, and Microsoft, are already utilizing and investing in AI technology. AI companies are gaining much popularity from a broad user base, and they usually offer integration of external actors with the AI company's models as one of the company's services. Ascertaining information flow becomes increasingly difficult within a system highly integrated with such services because the data flows outside the system's scope. End-users and companies might harbor reservations due to potential legislature if compliance is not pervasive across every integrated system. End-users might opt not to use the service due to trust issues, or companies might not use the service due to discrepancies between the companies' compliance requirements. Hence, such restraint hinders technology integration for many facets of society due to lack of information afforded through explainability.

State-of-the-art models train on data with various provenance, e.g., open web, blog posts, news articles, user chat histories, and various documentations [74, 34, Llama and Mistral]. Consequently, the access to significant amounts of data is becoming a contested resource in today's internet environment. The competition for this access has resulted in the commercialization of previously open sources for training data. This commercialization has caused technology disruption for end-users, e.g., via placing pay-walls in front of APIs crucial to the operation of community driven platforms [80, 31, X and Reddit API restriction]. Litigation against AI companies for misuse of training data further underscore the plausible conflict of interests that may arise from a lack of compliance and explainability [25, 12, Lawsuit against OpenAI and Microsoft]. Thus, AI's fingerprint on the world's digital arena continues to grow despite the notable effects from the already extensive presence of AI, and newer developments will require an improvement to explainability if the future brings even more complex black-box models.

1.2 Research Context

The writing of this thesis is in association with the Department of Computer Science's Cyber Security Group (CSG) at the University of Tromsø (UiT)[1]. The group's research pertains to fundamental systems design, focusing on proof of concept and applicability. CSG's previous work incorporates a principled approach considering the entire stack of system development. The group collaborates with cross-disciplinary experts from academia and industry in computer science, statistics, psychology, sports, medicine, legal science, and fishery. The research strategy addresses real-world applications with high impact through focusing on non-functional system requirements such as scalability, fault-tolerance, and the Confidentiality, Integrity, and Availability (CIA) security properties. Within computer science, the group works with technologies and areas such as multimedia, Internet of Things (IOT), cloud computing, Trusted Execution Environment (TEE), AI, blockchain, distributed systems, security, compliance, and privacy. Following are some brief descriptions of publications related to the group's interests.

Bagadus [65, Stensland et al., 2014] is a sensor integration system for real-time sports analysis. The system aspires to even out the financial advantages that larger soccer clubs exhibit over smaller ones. The system achieves this goal by stitching the video feed of four low-cost cameras running on commercial hardware instead of high-end panorama cameras with commercially licensed software.

[1] CSG homepage: `https://uit.no/research/csg`

Diggi [21, Gjerdrum et al., 2019] is a scalable and performant framework for native Function as a Service (FaaS) cloud execution. The system maintains security and requires minimal trust in the cloud's underlying infrastructure using a TEE with an acceptable performance trade-off.

Tedeschi et al. analyze market fees for transaction inclusion in proof-of-work cryptocurrencies. Their contributions enable users of the blockchain to optimize transaction time and inclusion fees through the use of ML trained on data extracted from Bitcoin [72, 71, 2019, 2022].

Dutkat [47, Nordmo et al., 2021] is a monitoring system combatting fishery crime. The system design is for a deployment scenario onboard fishing trawlers, which maintains the fault-tolerance of system in an untrusted environment. The system utilizes AI technology on the edge to provide surveillance evidence for legal requirements while preserving personal privacy.

Juliussen et al. explain the legal aspects related to transferring personal data outside countries that are compliant with GDPR [35, 2023]. They describe the requirements for such transfers and which privacy-enhancing technologies are available to remain compliant if such practices occur.

Kozyri et al. explore Information Flow Control (IFC) primitives to address security, compliance, and privacy issues in various areas, such as run-time environments, programming languages, or system inter-components [40, 41, 2019, 2022]. A previous thesis topic derived from this research is the enforcement of data restrictions in the programming language Rust by use of reclassification in IFC mechanisms [27, Hansen, 2022]. On-going projects under this scope envision to:

- Ensure integrity in end-to-end traceability at the sensor level for IOT devices deployed at the edge.

- Explore privacy violations occuring in federated ML by reconstruction of the sensitive data.

- Provide users with an informative summary of their public data found by crawling the web.

- Summarize variable flow in programming languages using IFC primitives.

This thesis falls under the group's category of IFC. Our greater goal is to support end-to-end data traceability in systems to empower end-users with control over their data. In this traceability context, the thesis address the problems with labeling in applications using ML, as described in chapter 1.1.

1.3 Research Questions

This thesis addresses the problem statement with an influence-based propagation policy where the label of contributing inputs with propagate to the output. Our approach addresses the label scalability issue by employing Explainable Artificial Intelligence (XAI) to answer questions regarding an input's influence towards the output. In the overlapping context of label propagation and XAI, we pose the following research questions:

I **How can we apply XAI toward the purpose of labeling model outputs based on labeled input data?**

II **How do the output labels change for image- and text datasets with different models and XAI methods?**

1.4 Thesis Contribution

This thesis proposes a Label Propagation Mechanism (LPM) for assigning labels of input data to outputs of ML models by leveraging explanations from XAI. The LPM uses an influence-based propagation policy where data must exhibit sufficient influence for its labels to propagate to the model's output. The XAI technique substantiates this propagation policy by creating explanations for the model output given one data point. The LPM utilizes this explanation for selecting which of the data point's labels to propagate.

This approach aims to support faithful labeling propagation for ML systems aspiring for compliance. That is, faithful in the sense of explanations from XAI are precise for the model's actual inner mechanisms given that specific data point and model. We test the faithfulness of the explanations by altering the important parts of the data point and observing the changes in the predictions. We discuss the label restrictiveness of various experiment configurations for different XAI. The thesis explores the proof of concept regarding traceability in the context of data as input to black-box models. The intention is to provide traceability despite the black-box problem rather than solving the black-box problem through conventional labeling schemes.

1.5 Thesis Methodology

The Association for Computing Machinery (ACM) defined computer science through creating a task force with this objective. The task force details the core findings in a summary of the full report: *Computing as a Discipline* [18, Denning et al., 1989]. This section defines this thesis in terms of the core points from the task force's definition of computer science. As such, we identify the thesis methodology in terms of the paradigms proposed in the definition. The definition covers *computer engineering* in addition to *computer science* because the core concerns of these domains relate to the same concepts. The term *Discipline of Computing* covers both of these domains where computer science and -engineering differ in terms of which concepts they emphasize. The task force provides three paradigms from various scientific fields as a context for the definition of discipline of computing. The following restates these paradigms and their characteristics:

Theory. From the field of mathematics, this paradigm concerns developing valid and coherent theories. The steps involve defining the research subject, propose a theory based on an hypothesis, prove or disprove the hypothesis, and evaluate possible conclusions based on the process. Scientists following the theory paradigm should employ iteration on these steps in the case of inconsistencies regarding the hypothesis and the proof.

Abstraction. From experimental scientific methods, this paradigm explores phenomena through experimentation. The steps involve proposing an hypothesis, modeling the phenomenon, designing and executing experiments, and analyzing the results from data collected from the experiments. This paradigm concerns modeling and experimentation more than creating abstractions that simplify the application of complex techniques and methods. Although, this paradigm's name is conventionally referred to as abstraction by the discipline of computing. The abstraction paradigm's steps are iterated when the analyses do not coincide with the experimentation of the model.

Engineering. From the field of engineering, this paradigm revolves around the creation of a system to solve a problem. The steps involve stating the system requirements and specifications, designing and implementing the system based on the requirements and specifications, and testing the system. This paradigm employs iteration of the steps when the testing demonstrates that the system's design or implementation does not satisfy the requirements or specifications.

From these paradigms, the thesis methodology identifies with *abstraction* and *engineering*. The only link to the theory paradigm is the application of valid and coherent techniques for creating models and explaining their predictions. The thesis uses a scientific procedure consisting of creating an hypothesis about label propagation from ML input to output. We then experiment with the ML model, data, and XAI methods to gather data for analysis. The abstraction hypothesis concept is comparable to the engineering paradigm's concept of system requirements and specifications. The methodology belongs to engineering through this comparable because labels enable the enforcement of data restrictions. On a higher level of application, label propagation enables enforcement of compliance in an end-to-end setting for a system's computational pipeline. In terms of the engineering's *system-test* step versus the abstraction's *experiment-design-and-execution* step, the methodology sides with abstraction more than engineering. The thesis experiments require a qualitative analysis of the results. System testing often use automatic evaluations based on the requirements and specifications, which requires a formal method of evaluation. Aside from analysis, there is no apparent formal method of evaluating the quality of labels propagated using XAI. Precisely defining the correctness criteria for propagated output labels is out-of-scope regarding the research questions of this thesis.

1.6 Thesis Outline

The contents of this thesis is summarized and structured as follows:

Chapter 2 details the related work and topics that the reader should familiarize with to readily understand this thesis.

Chapter 3 elaborates how our approach builds on related work to answer the research questions.

Chapter 4 describes the experiment methodology, presents results from the LPM, and evaluates the case studies.

Chapter 5 discusses topics for future work regarding the approach and results.

Chapter 6 provides the thesis summary and concluding remarks.

2 Background

The thesis problem statement from chapter 1.1 states that label propagation cannot enforce data restrictions in systems with Machine Learning (ML) components. ML is a sub-domain of Artificial Intelligence (AI) and recent advancements in the scientifics field has caused an boom in applicable areas. We address this problem with an influence-based Label Propagation Mechanism (LPM) that propagate labeled input data to a model's output depending on the input's influence for the model's output. The LPM applies explainability methods proposed in research publications to select which input labels to propagate. To our knowledge, no related work addresses this problem by using explanations for influence-based propagation of labels to model outputs. This chapter elaborates on the related work relevant for the thesis, and is structured as follows:

Section 2.1 describes programmatic label propagation and its application to ensure end-to-end traceability.

Section 2.2 provides an overview of Artificial Intelligence (AI) and Machine Learning (ML) and points out which subjects are problematic for label propagation.

Section 2.3 discusses Explainable Artificial Intelligence (XAI); the general objective and its common components.

Section 2.4 details specific XAI approaches and methods.

Section 2.5 describes lattices of labels and the semantics related to these labels.

Section 2.6 summarizes this chapter's contents.

2.1 End-to-End Traceability

End-to-end traceability can provide compliance and explainability for systems requiring these properties. Systems can enforce such property policies by tracing the information flow through the system's components, e.g., [50, Pasquier et al., 2017, CamFlow]. Compliance policies directly relate to the system's data restrictions, which risk violation with the use of black-box models. Alongside the development of faster and better AI, enforcing compliance in systems that include this technology is paramount. The European Commission's Assessment List for Trustworthy Artificial Intelligence (ALTAI) highlight the importance of data traceability for achieving transparent and trustworthy AI [77, ALTAI, requirement 4, page 14].

Traceability is the ability to trace an output's provenance to the original data that computed the output. Figure 2.1 demonstrates an example provenance trace of a system with an end-user's data spread throughout multiple branches, application procedures, and storage devices. Labels are used to implement end-to-end traceability by following the data through the system's computations. *Labels* are metadata describing the associated data. The exact semantics of the label is application specific, i.e., labels can mean anything and exist to fulfill any purpose. Thus, any LPM should be agnostic to the label's semantics. To maintain traceability, procedures executing consecutive operations on inputs should preserve the labels from the input to the output. Label propagation policies dictate which input labels should propagate to the output such that an output label comprises the original data labels. Inputs with equal outputs might not exert the same influence towards the procedure's computation. Hence, an influence-based propagation policy should exclude the label of inputs that does not sufficiently contribute to the output. Definition 1 formalizes influence-based label propagation policies for this thesis.

Definition 1. An influence-based label propagation policy states which labels should propagate to the output depending on the input data influencing the procedure's outcome.

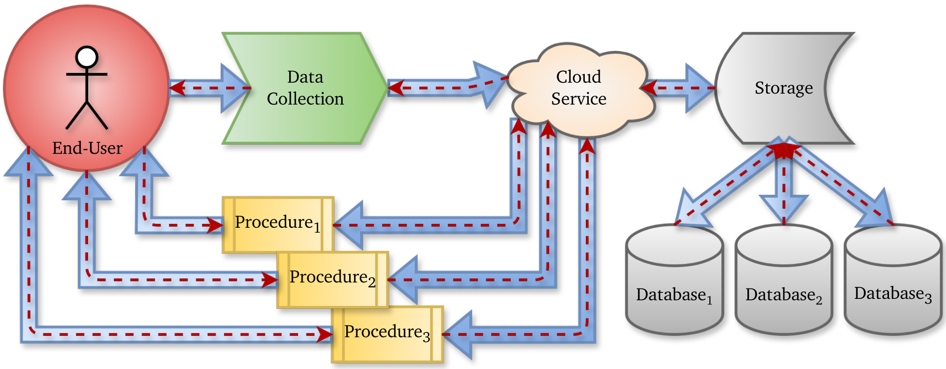


Figure 2.1: Demonstrates a multi-component system. The user's data is collected, then spread throughout the system's components, and finally returned to the user as an output from some application procedure. The blue arrows indicate the data flow direction, and the red dotted arrows indicate the reverse trace back to whence it came.

2.1.1 Label Propagation in Programs

The following discusses techniques and considerations related to label propagation in computer programs.

Explicit Propagation

Consider algorithm 1; a program where the input variables x and y directly compute the output w. We use l_x to denote the label of x and l_y to denote the label of y. In Information Flow Control (IFC), this direct assignment constitutes what is referred to as an *explicit* information flow from x and y to w [10, Bacon et al., 2014, section 4.a]. The program's output w always depends on x and y; therefore, the output label l_w amalgamates the two input labels l_x and l_y. Equation 2.1 expresses the program's label association rule where $\sqcup$ is the disjoint union between sets of labels.

Algorithm 1 A program computing the addition of two inputs.

```
  $w \leftarrow x + y$
  return $w$
```

$$l_w = l_x \sqcup l_y \tag{2.1}$$

Implicit Propagation

Conditional statements can generate complex patterns of information flow throughout a program. Input variables in the conditional statement contribute to the program's output, and the variable's labels should consequently propagate to the output. Algorithm 2 is a conditional program where the output w differs depending on the value of the input variable. The only explicit assignments to the output w are static values and the labels of static values are named the *bottom* label and denoted $\perp$. The bottom label's has the property $\perp \sqcup l' = l'$, i.e., they are always part of any other labels, and therefore, $\perp \in l_w$ is always true. However, the output w differs depending on the value of x, so propagating $l_w = \perp$ is not sufficient to express x's contribution to w. The explicit static value assignments occur within the context of the conditional statement $x < 0$ such that there is an *implicit* contribution from x to w [10, Bacon et al., 2014, section 4.a]. Therefore, the conditional statement variable's label l_x should propagate to the output's label l_w. Equation 2.2 expresses program 2's label propagation rule.

Algorithm 2 A program with the input variable in a conditional statement.

```
if $x < 0$ then
    $w \leftarrow 1$
else
    $w \leftarrow 2$
end if
return $w$
```

$$l_w = l_x \tag{2.2}$$

Output Label Variation

The previous examples have shown constant propagation rules, which is not always the case. Algorithm 3 demonstrates a program where the output labels varies depending on the input values. There are multiple explicit assignments to the output w from either y or z, and that specific assignment depends on the value of x. The program exhibits an implicit contribution across variables and to the output by extension such that the output receives y when $x < 0$ is true, and z otherwise. Thus, the output's label l_w should always include l_x and l_y or l_z depending on $x < 0$. Equation 2.3 expresses program 3's propagation rule for its output labels.

Algorithm 3 A program with three inputs x, y, and z where either y or z contributes to the output depending on x.

```
if $x < 0$ then
    $w \leftarrow y$
else
    $w \leftarrow z$
end if
return $w$
```

$$l_w = \begin{cases} l_x \sqcup l_y, & x < 0 \\ l_x \sqcup l_z, & \text{otherwise} \end{cases} \tag{2.3}$$

2.2 Artificial Intelligence and Machine Learning

The definition of Artificial Intelligence (AI) is difficult to bring under one statement where the scientific community can agree. AI is a field of study that has origins in Alan Turing's formulations on the topic of Computing Machinery and Intelligence [75, Turing, 1950]. One often cited AI definition is "*the science of making machines do things that would require intelligence if done by men*" [19, Minsky]. AI is an umbrella term extending to many sub-domains such as problem-solving, knowledge representation, automated decision-making, learning, natural language processing, and computer vision.

Machine Learning (ML) is an AI sub-domain concerned with the extraction of knowledge by *learning* from the data, i.e., learning in the sense of universal function approximation [29, Hornik et al., 1989]. ML algorithms can learn to model complex relationships between data and target objectives, often in a high-dimensional space. Deep Learning (DL) is a sub-domain of ML that employs the same technologies but on a larger scale, which results in highly accurate models that require large amounts of data to train. A Neural Network (NN) is a class of ML technology frequently used in practical applications and the class includes many variant networks such as the Feed-Forward Network (FFN), Recurrent Neural Network (RNN), Auto-Encoder (AE), Convolutional Neural Network (CNN), Generative Adversarial Network (GAN) and Deep Residual Network (DNR), to mention a few [8, Al-Aradi et al., 2018, section 4.1, figure 4.2]. Many NNs share the same core concepts, which are discussed in section 2.2.2 with focus on the Multi-Layer Perceptron (MLP) network, alternatively referred to as a Fully-Connected Network (FCN). In DL, a FFN is named Deep Neural Network (DNN) due to the depth and size of the network. Not every aspect of AI includes ML, but today's use of the term AI often incorporates some form of ML. This section defines only the concepts relevant for label propagation and is not a complete survey. Supervised, unsupervised, and reinforcement learning are the three main branches of ML, and this thesis is predominantly concerned with the class of supervised learning.

Supervised learning extracts the feature patterns with a priori knowledge about the data. This knowledge is usually an associated ground truth (or target) with each data point [23, Goodfellow et al., chapter 5.1.3]. Consider an object recognition task; the ground truth for an image of an airplane would be something the model can classify, such as a unique numeric identifier that signifies the airplane's class apart from the other object classes. In the context of ML, the terminologies *ground truth* and *label* are synonymous. However, due to the conflicting terminology between ML and traceability, this thesis will always refer to ML labels as ground truth and never as labels.

Unsupervised learning has the same goal as its counterpart, except without access to any ground truth. Instead, these learning techniques find an underlying structure by exploring similarities in the data [73, Theodoridis et al., chapter 1.3, page 7]. The challenge is often finding meaningful measures of similarity and constructing algorithms that generalize well across various data. The classic example of unsupervised learning is clustering algorithms, which discover groups in the data through automation or guided methods [73, Theodoridis et al., chapter 13]. **Semi-supervised learning** combines supervised and unsupervised learning that involves creating the ground truth from the data. The classical case of semi-supervised learning is the extraction of essential feature representations (used in denoising objectives) by learning to reconstruct the data from an under-complete latent space [23, Goodfellow et al., chapter 14.1-5, page 503].

Reinforcement learning defines an dynamic environment, an actor or agent, a set of interactions, and a feedback loop. Like supervised learning, reinforcement learning's ground truth is the feedback loop, which generates rewards when the actor interacts with the environment. An actor's goal is to learn which actions to perform given the situation considering the environment and previous actions such that the reward is maximized. The actor needs to optimize this reward without an instructor's guidance on which actions to take and through trial-and-error over multiple steps. The core difference with reinforcement learning is the consideration of the reward as an optimization-task over time rather than the immediate highest reward of the next step [67, Sutton et al., 2018, preview-version, section 1.1]. Consider an automated radio-controlled car learning to navigate a racing course without crashing into the course's barrier fence taking actions per-second. Here, the environment is the course, the actor is the car, the set of interactions are the speed, brakes, and car-steering, and the feedback is the collision detection between the car and fence. The feedback loop could include an overall lap time for circumventing the course, which the actor should learn to minimize while avoiding crashing.

2.2.1 Data Features

Data points are composed of feature vectors and the data point's features are the inputs to a ML model. A feature's value is a measurement of some object or event and is either a continuous or discrete value, alternatively referred to as numerical or categorical [23, Goodfellow et al., chapter 5.1.1]. The exact features composing the data depends on the problem, which often guides the feature generation process that works well with the target ML model. There are various feature preprocessing procedures that are applied before input into the model such as feature selection or -extraction methods such as sequential forward- and backward-selection, and Kernel Principal Component Analysis (PCA) [73, Theodoridis et al., chapter 5.7.2 and 6.7.1]. DL models have an inherent objective to learn meaningful representations from complex features, though occationally preprocess their input as well [23, Goodfellow et al., chapter 1, page 5]. The families of data generally divide into these types: *image*, *textual*, *tabular*, and *temporal* data. Although, this section only details the data types pertinent for the thesis.

Image features are the individual pixels or color channels, where each feature is the numerical intensity of the pixel [23, Goodfellow et al., chapter 5.1.1]. The number of features in an image is related to their resolution, e.g., the pixel resolution $299 \times 299 = 89401$ features and $299 \times 299 \times 3 = 268203$ features with the RGB color channels. Image formats typically use (or support) pixel encodings with one unsigned byte per color channel. Such encodings imply each feature will have a value between $[0, 255]$, and therefore, the format's pixel values are categorical. The mathematical definitions of ML require the values to be continuous. Therefore, images are often preprocessed from their format into normalized floating-point values between $[0, 1]$ before ML training or inference. Many algorithms require working with one specific image resolution, so they perform a resizing method on the input as a preprocessing step. The input preprocessing algorithms used to resize images often include nearest neighbor-, bilinear-, or bicubic interpolation [22, Gonzalez et al., 2018, chapter 2.4, page 77].

Textual data points are variable-length sequences of symbols (or characters), and preprocessing the text extracts the textual features. Text preprocessing often involves tokenization and vector embedding in that order. *Tokenization* creates tokens from the text by mapping subtexts to categorical integer identifiers. The tokenization process can be learned or based on rules, such as identifying the subtexts as words, characters, or characters of fixed length and mapping these to tokens. *Vector embedding* maps tokens into some high-dimensional vector space. This high-dimensional space exhibits the property that vectors of text with similar conceptual meanings are closer to each other where the similarity measure varies, e.g., Euclidean-, Manhattan-, cosine-, or Chebyshev distance [44, Mikolov et al., 2013]. Consider tokens from a dataset of cooking recipes and the vector embeddings corresponding to the words bread, flour, and apple. The words "bread" and "flour" should be closer together than "apple" because flour is an ingredient of bread. The dimensionality of the embedding space only requires enough n-volume in the n-dimensional hypercube of the space to express the nuanced differences between texts in the dataset.

2.2.2 Neural Networks

The NN model is fundamental in modern high-performance ML. The NN's design draws inspiration from the human brain, where layers of neurons connect to another layer's neurons. These networks are present in nearly every high-performing model architecture. On a high level, the NN is a functional mapping from some input to an output, learned through an iterative training process on data [29, Hornik et al., 1989]. This section details common properties and concepts related to NNs, but focuses on the FCN sub-category of NNs. The visual depiction of a FCN is a Directed Acyclic Graph (DAG) where each layer's neurons connect to every neuron in the next layer, as can be seen in figure 2.2.

Network Anatomy

NNs are created with layers and one layer in a FCN is often referred to as a Fully Connected Layer (FCL) where the number of neurons in each layer may vary. Each neuron in a FCL produces one numeric output activation that becomes input to each neuron in the subsequent layer. A neuron consists of one bias and a set of weights dependent on the number of neurons in the previous layer. The first layer's input is the data features, and the layer's weights depend on the number of features in the data [59, Shalev-Swartz, 2014, chapter 20].

The static number of weights in the first layer means they can only interact with statically structured data because there is one weight per input feature. This property implies that a NN cannot operate on free text with variable length characters or images of variable resolution and explains the need for preprocessing such as tokenization, vector embedding, and resizing. Figure 2.2 demonstrates a FCN design for a classification problem with two classes where the network's output is 0 or 1 depending on the which prediction is highest out of $\hat{y}_1$ and $\hat{y}_2$. The last layer's activations are the network's prediction, and

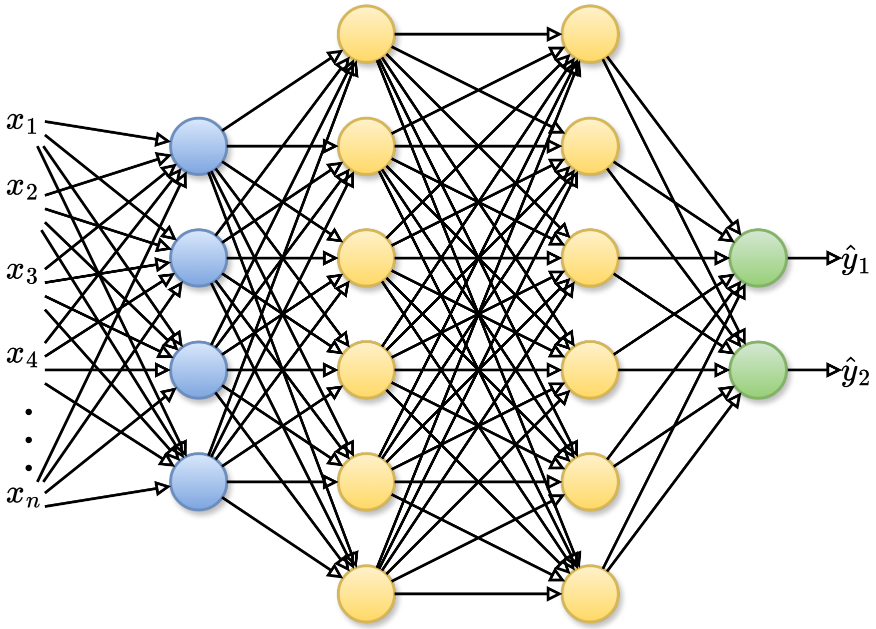


Figure 2.2: A FCN of four fully connected layers with 4, 6, 6, and 2 neurons per layer, respectively. The input to the blue layer is the actual data, the yellow neurons are the hidden layers, and the green layer is the output classifier.

usually the values are normalized to the range $[0, 1]$, so that the output is probabilistic, alternatively referred to as the confidence of the prediction.

Network Mathematical Expressions

Neuron outputs are weighted sums of the input features with their corresponding weights, which are then subjected to an activation function; hence, the term neuron activation. Equation 2.4 expresses the activation output for one neuron in a NN where k indicates which of the network's layers, n indicates which neuron in the kth layer, a is the activation function, L_{k-1} is the number of neurons in layer $k-1$, $w_{n,i}^{k}$ is the weight in the kth layer of the nth neuron for the ith layer input x_i^{k-1} from the previous layer, and b_n^k is the bias of the nth neuron [59, Shalev-Swartz, 2014, chapter 20].

$$\hat{y}_n^k = a\left(\sum_{i=1}^{L_{k-1}} w_{n,i}^{k} x_i^{k-1} + b_n^k\right) \tag{2.4}$$

The neuron's pre-activation (input to $a(\cdot)$) is a linear combination of weighted

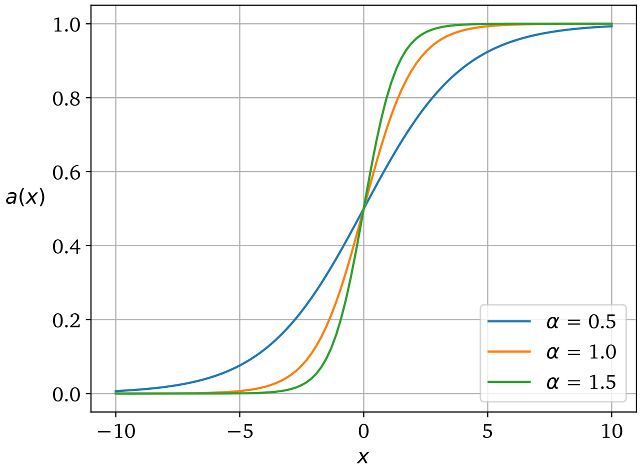


Figure 2.3: Plots the Sigmoid activation function's graph from equation 2.5 for different values of α.

input connections. The purpose of the activation function is to regularize the pre-activation and add non-linearity, which allows networks to learn more complex patterns [59, Shalev-Swartz, 2014, chapter 20]. The Sigmoid function is an activation function that exhibits these properties and sees regular application in practice. Equation 2.5 expresses the Sigmoid function, which compresses the input into the range $[0, 1]$ with non-linearity where α parameterizes the S-shaped curve of the activation. Figure 2.3 plots the graph of the Sigmoid activation function for different values of α.

$$a(x) = \frac{1}{1 + e^{-\alpha x}} \tag{2.5}$$

Network Training

NNs use optimization methods during training of which there are many, and one such optimization method is Stochastic Gradient Descent (SGD). SGD is an optimization technique that aims to minimize the network's output error over an iterative number of epochs. An epoch is synonymous with an iteration step. Error is synonymous with loss, and a loss function formalizes such errors. For supervised learning, loss functions compute the error based on the discrepancies between the network's output and the ground truth [59, Shalev-Swartz, 2014, chapter 14]. Equation 2.6 expresses the Cross-Entropy loss function where $\hat{y}$ is the vector of the neuron outputs from the last network's layer, y is the corresponding ground truth vector, $\hat{y}, y \in [0, 1]$, K denote the last layer in the network, and L_K denote the number of neurons in the last

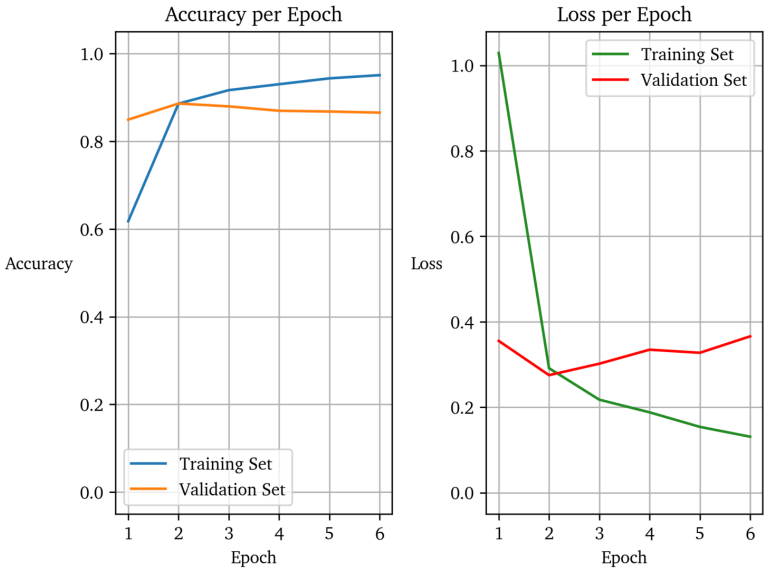


Figure 2.4: Demonstrates the correlation between the decreasing error and the increasing accuracy of an in-training NN per epoch for the training and validation data. This training session shows signs of overfitting where the validation metrics diverge from the trend of the training set.

layer. Figure 2.4 is the plot of the accuracy and loss of a NN during training over several epochs. The figure demonstrates the network's convergence toward an optimum from the increasing accuracy and decreasing loss per epoch.

$$\mathcal{L}(\hat{y}, y) = \sum_{n=1}^{L_K} \hat{y}_n^K \log y_n \tag{2.6}$$

SGD uses the chain derivative rule to compute an update delta $\delta w_{n,i}^k$ for each weight in the network. The weights are updated in reverse order, starting with the network's last layer and continuing backward. This update method's name is back-propagation. *Back-propagation* is backward because the chain rule expression for any weight is the derivative of the loss with respect to the network's weights. The loss computation involves the network's output and the data's ground truth; therefore, the update deltas trace back to the network's error. In that sense, each layer's updates base itself entirely on the subsequent layer's delta, which demands that the update algorithm be backward. The network minimizes the loss (and optimizes the weights) through iterative updates to the network's weights with the gradients in the negative direction, i.e., SGD is a search for a global minimum in the high-dimensional loss surface

[59, Shalev-Swartz, 2014, chapter 14 and 20.6].

Note that figure 2.4 is the *loss per epoch*, which is distinct from the *loss surface*. After enough epochs, the network should converge into weights that optimally predict outputs according to the data's ground truth. SGD is a prime example for why ML models are said to *learn* from data because of this *practice makes mastery* approach. Equation 2.7 expresses the chain derivative rule for the gradient update step for an arbitrary weight in the network. Equation 2.8 expresses the rule for each weight's next iteration update step where t is the epoch counter, and η is the learning rate that scales the size of each gradient step.

$$\delta w^k_{n,i} = \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial w^k_{n,i}} \tag{2.7}$$

$$(w^k_{n,i})_{t+1} = (w^k_{n,i})_t - \eta \delta w^k_{n,i} \tag{2.8}$$

Network Redundancy

NNs can exhibit the behavior of nuanced input-output contributions where not every input's features contribute towards the model's predictions. These non-contributions are highly relevant for information-based propagation policies because the policy should exclude labels from non-contributing features. However, propagation policies for NNs in modern DL-practices can not rely on this behavior due to accuracy optimization techniques, which DL employs to erase such behavior from NNs. Non-contributing features appear due to the network learning non-contributing weights, alternatively referred to as *redundant representations*. In other words, redundancy in the sense that highly correlated features can capture the exact feature-to-ground-truth correlation with a subset of the correlated features [23, Goodfellow et al., 2016, chapter 8.1.3, page 278].

Consider the data features and ground truth in table 2.1; the feature-to-ground-truth correlation $\{x_1, x_2, x_3\} \longrightarrow y$ is equivalently expressed with the subset $\{x_1, x_2\} \longrightarrow y$. This redundancy is due to $x_3 = x_1$ for every row in the table. Alternatively, networks can learn non-contributing relationships between neurons across layers. Consider the network design in figure 2.2 for the data in table 2.1. Assuming the top three neurons in the first layer learn to represent x_1, x_2, x_3, respectively, i.e., represent in the sense that the first neuron's output informs the second layer about x_1's relation to x_2 and x_3. The fourth neuron should then learn a redundant representation because there are three features.

Instances	x_1	x_2	x_3	$\longrightarrow$	y
x^1	0	0	0	$\longrightarrow$	0
x^2	0	1	0	$\longrightarrow$	1
x^3	1	0	1	$\longrightarrow$	1
x^4	1	1	1	$\longrightarrow$	0

Table 2.1: Demonstrates a dataset with high feature correlation (redundancy) between x_1 and x_3.

Neurons one and three should output the same values because $x_3 = x_1$. Hence, the second layer should learn that the incoming connections from either the first- or third neuron in the previous layer are redundant, expressed with the second layer's weights $w^2_{n,1} = 0$ or $w^2_{n,3} = 0$, and drop any contribution from that connection.

Neuron zero-activations are equivalent to broadcasting a *do-not-care* to the next layer, i.e., informing the next layer to disregard any contributions towards their output coming from zero-activations. During training with SGD, zero-activations cause their chain rule derivative contribution to be zero, and neurons might remain redundant despite updates. These traits are undesirable in NNs because redundant representations do not add value to the output. Dropout is a common technique to avoid redundant neurons by forcing each neuron to cooperate with an arbitrary set of neuron connections. Dropout randomly zeros out activations for some neurons in a network's layer during training [64, Srivastava et al., 2014], which is problematic for influence-based LPM policies since there are no traceable zero-activations in these NNs.

Consider the label propagation mechanisms explained in the programs from chapter 2.1.1, except in the context of NNs. The NN in figure 2.2 indicate that every input x from left to right bleed into every contributing computation of the network's output $\hat{y}$. The expression for neuron outputs in equation 2.4 state the same, i.e., every neuron's activation involves every activation from the previous layer. Under these circumstances, a LPM should evaluate the input's values to ascertain their influence over the network's output. Algorithm 4 defines a program computing the forward pass of a FCN. Defining a clear LPM policy for this program is difficult because of the absolute combination of weights and inputs.

Algorithm 4 A program computing the forward pass of a FCN.

```
$x^0 = \langle x_i \; \forall \; i \in \{1 ... m\}, x_i \in \mathbb{Q} \rangle$
$w = \langle w^k_{n,i} \; \forall \; k \in \{1 ... K\} \land n \in \{1 ... L_k\} \land i \in \{1 ... L_{k-1}\}, w^k_{n,i} \in [-1, 1] \rangle$
for $k \in \{1 ... K\}$ do
    for $n \in \{1 ... L_k\}$ do
        $\hat{y}^k_n \leftarrow 0$
        for $i \in \{1 ... n\}$ do
            $\hat{y}^k_n \leftarrow \hat{y}^k_n + w^k_{n,i} x^{k-1}_i$
        end for
        $\hat{y}^k_n \leftarrow a(\hat{y}^k_n + b^k_n)$
        $x^k_n \leftarrow \hat{y}^k_n$
    end for
end for
return $\hat{y}^K$
```

2.3 Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) is the study and development of methods and techniques for understanding and explaining (or interpreting) Artificial Intelligence (AI) [42, Linardatos et al., 2021, section 1]. There is an ML subdomain in XAI concerned with explaining black-box models' behavior and decision-making process. XAI is experiencing a boom in interest due to the advances made in Deep Learning (DL), which create accurate, though hard-to-explain, black-box models [26, Gunning et al., 2019, section 2]. Certain critical application areas are sensitive to failures or might have strict requirements on compliance and explainability. They remain reluctant to adopt the more powerful models due to their black-box property. In these areas, the prediction alone cannot satisfy the application's requirements without an accompanying explanation [46, Molnar, 2022, chapter 3.1]. Examples of these areas are the healthcare, manufacturing, criminology, and law. There have already been severe events to indicate how badly AI can go wrong without significant investments in robustness [9, AI lethal accident]. Thus, XAI receives much attention from research, business, and the open-source community to improve the AI model's lacking properties.

2.3.1 Explanations

Formally, XAI produce *explanations* where an explanation should mitigate the model's black-box characteristic. XAI varies between techniques that are model-agnostic or -specific. Another form of XAI are models with predictions that are inherently interpretable [42, Linardatos et al., 2020, section 1]. We refer to the explanations of model-specific and -agnostic XAI as *black-* and *white-box explanations*, respectively. Although, these explanations do not relate directly to black- and white-box models, i.e., a black-box explanation is not an explanation for a black-box model. Definitions 2 and 3 formalize the concepts of white- and black-box models versus explanations, respectively [46, Molnar, 2022, chapter 2.3 and 6]. Therefore, a black-box explanation might explain the predictions of a black-box model, though only because the explanation makes no assumptions regarding the model. There is nothing stopping model-agnostic XAI from explaining a white-box model.

Definition 2. Black-box models are models whose behavior is not readily understandable by inspecting the model's internal mechanisms. White-box models are the complementary opposite, i.e., models with behavior that is understandable through model inspection.

Definition 3. Black-box explanations are explanations produced by XAI methods, models, or techniques that explain an output without relying on the ML model's internal mechanisms. White-box explanations are the complementary opposite, i.e., explanations from XAI that does not apply to every model due to specific model requirements.

An explanation describes how the model behaves or why the model makes a specific prediction for a set of inputs. Explanations are divided into *global* and *local* explanations [46, Molnar, 2022, chapter 3.3]. Definition 4 formalizes the terminologies for explanations, both local and global. Related work puts emphasis on the user's understanding of the explanations; however, this is out of scope for this thesis since user-interaction is not the prime focus regarding label propagation.

Definition 4. An explanation is an understandable reason for the model's general behavior or singular prediction. A local explanation is specific to a subset of model predictions. A global explanation is general to the model's behavior.

Explanations manifest in various ways for different models and data types. The common denominator is that each explanation is in some form qualitatively understandable by humans. These manifestations can be categorized as follows [46, Molnar, 2022, chapter 3.2 and 3.6.1]:

I **Feature statistics** are numerical measures of importance for a set of features, e.g., contribution scores toward the output.

II **Feature visualization** highlights specific parts of the input or displays feature statistics in presentable formats, e.g., image superpixels, words in the text, or tabular column distribution plots.

III **Data points** found as an alternative to the original data point, e.g., data points with a different prediction or an input that is optimal for the model to make a certain prediction.

IV **Interpretable models** that are inherently understandable. These might be models taught to produce a local- or global explanation for the target model or models that output an explanation along with their regular output.

Global explanations reveal relations between data points concerning the model and these explanations could guide a LPM's policy. However, the contributing factors should relate to the explained data point and not some other data points. Propagating a data point's labels according to an explanation based on other data points does not follow the intuition of influence-based propagation from definition 1. Consider a probabilistic model taught to detect distributions and identify from which distribution a data point originates. A global explanation could justify that 95% of data points with a certain feature value belong to the predicted distribution. This explanation is sufficient to propagate that feature's label if the value is present, but raises the question: *Should the other* 95% *of the data features propagate their labels to the output as well*? The data points that created this statistical explanation might originate from the training set. Such an explanation could reintroduce the underlying cause for the problem statement from chapter 1.1, i.e., the number of labels to propagate does not scale. Thus, this thesis exclusively concerns local explanations for guiding the LPM.

2.3.2 Interpretability versus Explainability

The XAI scientific community often makes nuanced distinctions between ML models that are *interpretable* and *explainable*, alternatively referred to as *interpretability* and *explainability*. There is no scientific consensus on the criteria for when a model is either interpretable or explainable. One family of models considered interpretable is not always understandable, as per definition 2 for white-box models. This inconsistency occurs when an interpretable model scales to a size where the model loses its intuitive internal mechanisms. Decision trees are interpretable models that predict an output depending on the data's affiliated leaf node [26, Gunning et al., 2019, figure 1]. The tree's root-to-leaf-node path can translate into an intuitive series of feature-based if-then rules, which is understandable depending on the features. However, data with a complex ground truth relationship either render decision trees unviable due to low-performance relative to DL models or become so deep that they lose the resulting if-then intuition [46, Molnar, 2022, chapter 5.4]. For clarity, we employ the white-box model definition 2 whenever this thesis refers to an *interpretable model*.

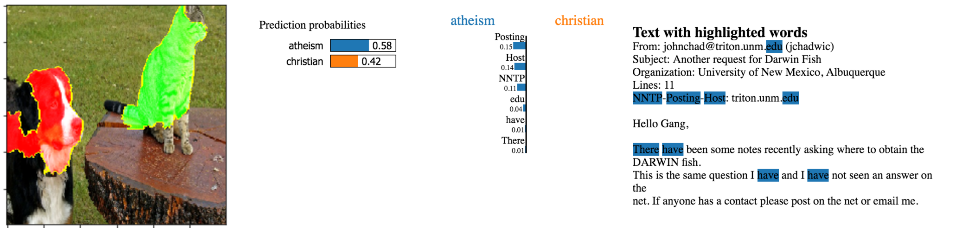
The XAI scientific community exhibits no consensus on the specific meaning and disambiguation between interpretability and explainability. Some sources state that explainability is a subset of interpretability [45, Miller et al., 2018, section 2.1.5] [46, Molnar, 2022, chapter 3]. Others define them separately: explainability as understanding why the model made one specific decision, and interpretability as understanding how the model works [42, Linardatos et al., 2021, section 2.1] [4, 32, 58]. Some mix the two definitions, and some blend the terms into the same concept. The underlying concepts that the different sources try to capture often relate to the explanation types that XAI can create, as discussed in chapter 2.3.1. For clarity, we do not attempt to disambiguate between explainability and interpretability and treat them synonymously with XAI. Instead, we refer to *local*- and *global* explanations rather than of explainability or interpretability.

2.4 Explainability Methods

The following sections details the specific XAI techniques and methods relevant for this thesis.

2.4.1 LIME

Ribeiro et al. [55, 2016] propose Local Interpretable Model-agnostic Explanations (LIME), a technique for creating local black-box explanations for black-box classification models. LIME trains a *linear model* to explain the predictions of a *target model* by sampling data points from the uniform distribution centered around the explainee data point. We refer to an *explainee data point* (or just *explainee*) when referring to a data point that is to receive an explanation for the model's prediction. LIME weights each sample's importance during training differently based on the similarity to the explainee such that samples further away affect the linear model less. The technique assumes a linear model can sufficiently capture the complexity of a target model's decision-boundaries around the prediction of the explainee. The method uses a linear model as the explainer model because its decisions are inherently explainable. LIME's explainer model is a modular design. That is, any model can implement the explainer model's role (instead of using a linear model), so long as the explainer model is *interpretable*. By interpretable, they refer to models whose decisions are explainable by the merits of their structure, e.g., logistic regression, decision



Text with highlighted words
From: johnchad@triton.unm.edu (jchadwic)
Subject: Another request for Darwin Fish
Organization: University of New Mexico, Albuquerque
Lines: 11
NNTP-Posting-Host: triton.unm.edu

Hello Gang,

There have been some notes recently asking where to obtain the
DARWIN fish.
This is the same question I have and I have not seen an answer on
the
net. If anyone has a contact please post on the net or email me.

(a) **(b)**

Figure 2.5: **(a)** LIME's explanation with positive (green) and negative (red) contributions to the image's classification as containing a cat.

(b) LIME's explanation with word-based contribution scores for text data. The explanation highlights the essential words for the model's classification of the text as atheist versus Christian.

trees, ensemble models, or falling rule lists.

LIME applies to image-, tabular-, and textual data types, and does not require access to any training data other than the explainee data point. The exception is tabular data because the sampling method requires the value distributions of individual columns. LIME produces explanations with understandable correlations between input features and output predictions. These explanations depend on the data feature's type but, in most cases, highlight the explainee's most significant features as discussed under item II in section 2.3.1. Additionally, LIME provides explanations with relative feature ranking- or contribution scores for some data types. Figure 2.5 [54, Ribeiro et al., 2016] depicts two examples of LIME's explanations for different data types with feature highlighting and ranking scores.

Later work by Zafar et al. [81, 2019] addresses the need for determinism related to LIME in application-specific areas. Many computer-aided diagnosis systems have strict requirements for output stability, and therefore, two XAI explanations of the same data point should not differ. Aside from providing a static seed for the number generator, LIME cannot guarantee this specification due to its stochastic sampling around the explainee data point. Therefore, they propose the alteration to LIME; Deterministic Local Interpretable Model-Agnostic Explanations (DLIME). The deterministic component of their work relates to the sampling method, which ensures that the sampled set of data points are identical for any two explanations of the same data point. They combine Agglomerative Hierarchical Clustering (AHC) with K Nearest Neighbors (KNN) to choose their exact samples from training data. Hence, their method introduces the requirement of access to training data representative of the explainee's distribution.

2.4.2 Anchor

Ribeiro et al. [57, 2018] propose Anchor, an algorithm for creating local black-box explanations for black-box classification models. Anchor is a continuation of LIME and the proposed explanation algorithm searches for *anchors* instead of training an interpretable model to explain the target model's prediction.

An anchor is a set of feature conditions for data points. A data point *satisfies* an anchor if its features are valid under the anchor's conditions. Any data point satisfying an anchor implies the model's prediction is the same as the explainee data point with high probability τ. Inversely, features not constrained by the anchor's conditions can change without significantly impacting the model's prediction. Thus, an anchor explains the model's prediction with the statement *the model's output is likely to change if we alter the features present in an anchor*.

The *high probability* factor τ is an input parameter for creating anchors, which acts as a requirement threshold for the confidence of sufficient anchors. That is, an anchor is sufficient only if any data point that satisfies the anchor meets the same prediction with τ probability. This means an anchor's conditional constraints are more relaxed with lower confidence thresholds but implies data points satisfying a low-confidence anchor yields an explanation that is wrong with probability $1 - \tau$. Lowering the confidence requirement is preferable when the model's prediction topography is too complex with the standard confidence level of $\tau = 0.95$ because the algorithm will find no anchor candidates. Therefore, a lower anchor confidence might relax the constraints such that at least some *less reliable* explanations are found.

The explanation algorithm finds anchors by stochastically sampling artificial data points around the explainee. An anchor is sufficient if its conditions hold for enough samples. The search objective includes a *coverage* criteria, meaning the final anchor choice comes from the set of sufficient anchors where the highest number of samples satisfy the anchor. The authors aspire to provide explanations that apply to more than one local data point because higher coverage offers improved human understanding of the model's behavior. Hence, coverage should help humans assess the outcome before the model's prediction. From the perspective of this thesis, an anchor explanation's coverage is a measure of *localized global explanations*. Exploring the anchor's coverage is out of scope for this thesis since the LPM emphasizes the explainee data point's influence on the output rather than the explanation's generalizability.

Anchors can explain models with image, textual, and tabular data input types and does not require training data, except tabular, for similar reasons as in LIME. The local sampling around the explainee varies depending on the data

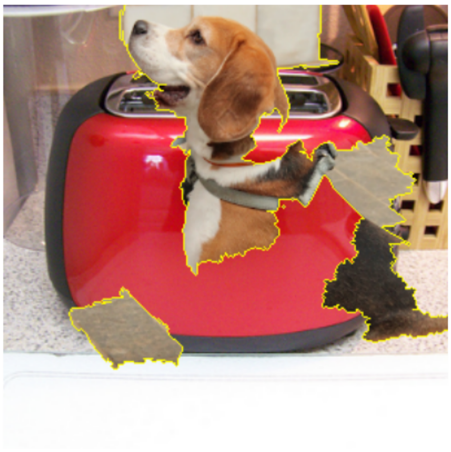


Figure 2.6: Demonstrates an example of an image anchor for a model's prediction of a dog. The anchor of the dog-image is put on top of a toaster to demonstrate that the model continues to classify the image as a dog.

type. **For images**, the algorithm initially applies a segmentation technique to create superpixels and then perturbs the individual pixels or color channels of those superpixels. The image's anchor explanation consists of the superpixels required for the model's prediction to match the explainee prediction with confidence higher than the parameterized threshold. Figure 2.6 [56, Ribeiro et al., 2018] demonstrates an example of anchor's explanation of an image. **For text**, the algorithm samples textual pertubations from the explainee on a per-word-basis. The explainee's words are replaced according to one out of three sampling strategies. The words are replaced with either:

1. *UNK*s (short for unknown).
2. Samples from a given corpus, e.g., English, that considers the word's position and the proportional probability between the word and replacement.
3. Suggestions from a given language model based on the model's probability distribution.

Similar to image anchors, the text's anchor explanation consists of words such that their presence in a data point suggests the model's output should correspond with the explainee output with probability τ. The implementation specific details listed here are partially based on Alibi's framework documentation [38, Klaise et al., 2021], which we discuss in chapter 2.4.4.

2.4.3 Integrated Gradients

Sundararajan et al. [66, 2017] propose Integrated Gradients (IG), a method producing local white-box explanations for DNN classification and regression models. The method takes inspiration from previous publications that use gradient inspection to attribute contributions to input features for the model's output. The previous work on the topic of gradient-based inspection include Deep Learning Important FeaTures (DeepLIFT) [60, Shrikumar et al., 2017], Deconvolutional Networks [82, Zeiler et al., 2010], Guided Back-Propagation (GBP) [63, Springenberg et al., 2014], Layer-wise Relevance Propagation (LRP) [11, Binder et al., 2016], Saliency Maps [61, Simonyan et al., 2014].

IG's main contribution is the identification and fulfillment of two axioms that gradient-based methods should satisfy where every preceding work fail at least once. These axioms are *sensitivity* and *implementation invariance*. A baseline x' is a data point from the input space where the model's prediction is impartial to any specific outcome. The baseline should have a $50/50\%$ model confidence for both classes in a two-class problem. For images and text data, the baseline might be the completely black image and the zero-vector from the embedding space, respectively, but it ultimately depends on the model. Thus, the gradient-based method satisfies sensitivity if every feature x_i is assigned a non-zero attribution where the input x and baseline x' have varying outputs $f(x) \neq f(x')$ and differing a feature value $x_i \neq x'_i$. A gradient-based method satisfies implementation invariance if the attributions are always identical for two functionally equivalent networks, i.e., functionally equivalent in the sense that the networks' input-to-output space is uniform $f(x) = f'(x) \; \forall \; x \in \mathbb{R}^n$.

2.4.4 Alibi

Klaise et al. [37, 2021] proposes Alibi, a Python library unifying various state-of-the-art XAI techniques in one consistent Application Programming Interface (API). The library supports local-, global-, white- and black-box explanations by hosting various XAI methods for image, text and tabular data, and comes with documentation [39, Klaise et al., 2021] for its APIs and the theoretical background regarding the explainability techniques. The framework promotes production-ready deployment of ML explainability with a distributed backend that contemporary XAI libraries lack. The work aims at reducing the distance between XAI research and industry application, and puts emphasis on offering a broad collection of explainability options such that there are alternatives if one method fails to provide an informative explanation, or if consumers possess a varied basis for understanding explanations.

Table 2.2 lists the set of explainability methods mentioned in this thesis and their associated properties. Note that the table borrows from Alibi [37, table 2], though Alibi does not host every method listed in the table such as LIME and DLIME. The following is an elaboration on the content's meaning in the table columns. The **model type** column states whether the method procudes white- or black-box explanations as per definition 3, and an asterisk "*" indicates the model must be differentiable, meaning the output space is continuous rather than discrete. Probabilistic models $p(x) \in [0, 1]$ are differentiable and, therefore, explainable with asterisk-marked XAI techniques, but classification models $f(p(x)) \in \{1...C\}$ with C number of classes derived from an intermediary probability $p(x) \in [0, 1]$ are non-differentiable. The **explanation** column states which explanations (local, global) the method provides. The **task** column states if the explainability method supports explanations for model classification- (C) and regression (R) outputs. The **data type** column lists which data types the method applies to. The **data requirement** column states whether the technique requires a dataset to create explanations.

Method	Model Type	Explanation	Task	Data Type(s)	Data Req.
LIME	Black-box	Local	C	Image, Tabular, Textual	Tabular
DLIME	Black-box	Local	C	Image, Tabular, Textual	Yes
Anchor	Black-box	Local	C	Image, Tabular, Textual	Tabular
IG	White-box	Local	C, R	Image, Tabular, Textual	Optional

Table 2.2: An overview of XAI methods and their cross-comparable attributes.

2.5 Lattice Based Access Control

A Lattice-Based Access Control (LBAC) is an access control model that specifies which security levels a subject must possess to access an object [17, Denning et al., 1976]. The lattice model uses labels to enforce the access policy that states: "*Subjects can only access an object if the subject's security level is greater or equal to that of the object*". Hence, a subject s labeled l_s can access an object o if the object's label l_o is *at least as restrictive* as l_s. Equation 2.9 expresses this access control's policy statement for a lattice L and partial order operator

$\sqsubseteq$ as a restrictiveness comparison. $o \Rightarrow s$ denotes the permitted information flow from o to s, meaning s is allowed to access o. The restrictiveness predicate $l_o \sqsubseteq l_s$ is true when the restrictions that apply to the object also apply to the subject.

$$\langle L, \sqsubseteq \rangle = \begin{cases} o \Rightarrow s, & l_o \sqsubseteq l_s \\ o \nRightarrow s, & \text{otherwise} \end{cases}, \mid l_s, l_o \in L \tag{2.9}$$

Consider the object as a *confidential document* and the subject as a *public website*. In this case, the predicate $l_o \sqsubseteq l_s$ does not hold because the restrictions that apply for the confidential document o do not also apply for the public website s, and therefore, $o \nRightarrow s$. Consider instead the subject as a *secret website* where the lattice prohibits its content from flowing into confidential documents, i.e., $s \nRightarrow o$. In this case, the subject's label l_s is more restrictive than the object's label l_o. Thus, the restrictions for the confidential document also apply to the secret website, so the secret website can access the confidential document $o \Rightarrow s$. Figure 2.7b depicts this type of lattice with *public*, *confidential*, *secret*, and *top-secret* as the labels, each more restrictive than the last.

Consider an Operating System (OS) enforcing user access control with LBAC primitives where the system allows or denies users access to resources based on the user's permissions. The lattice in this scenario is the security levels for the OS's resources, the user's roles, and how they are allowed to interact. The lattice's subjects are the OS's users, and the objects are the resources such as files, memory regions, and network ports. Figure 2.7a depicts the lattice security labels for two OS users. Each label specifies which users are allowed to access resources with that label. Resources labeled with the lowest label indicate both the *owner*- and *guest* user can access that resource. Every lattice contains a *bottom* label such that information with this label can flow anywhere in the system without restriction, and lattices denote this label with $\perp$. The two labels one step up the lattice indicate just the *owner*- or *guest* user can access the resource, where the top label specifies that none can access the resource. This lattice demonstrates how the LBAC model can express complex access controls. The lattice in figure 2.7b is a more straightforward security level hierarchy than the lattice in figure 2.7a. This thesis applies lattice variations of the hierarchical type in figure 2.7b. Lattices provide label semantics that this thesis uses to answer research question two from chapter 1.3, namely, which observations can be made for evaluating the change in the propagated labels for different LPMs. This label semantic is the *label restrictiveness*, which this thesis elaborates in chapter 3.2.

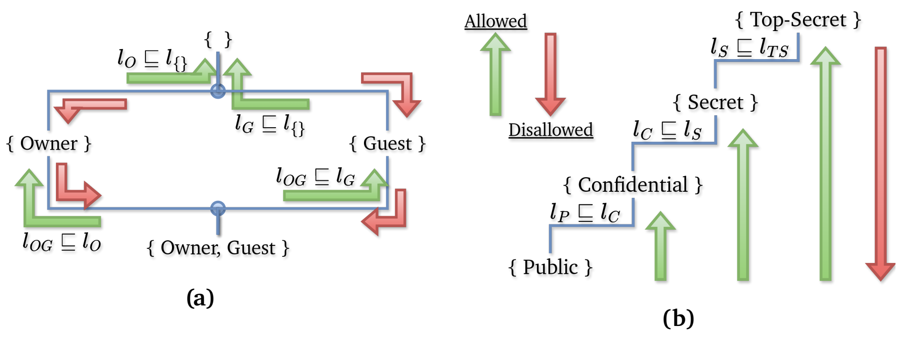


Figure 2.7: Two different lattices where the green and red arrows indicate the permitted and prohibited information flows from equation 2.9. The lattice labels are the curly brackets $\{\cdot\}$ and are denoted $l_{\{\cdot\}}$ where the subscript is the first letter of the label's source or destination. The restrictiveness predicates are only placed where the information flows in the lattice are allowed, e.g., {Owner, Guest} $\Rightarrow \{Owner\}$ because $l_{OG} \sqsubseteq l_O$ holds. The figures are more readable in down-up order, and the figure implies the flows between labels that are not shown, e.g., from the bottom to the top.

2.6 Summary

Influence-based label propagation in computer programs is a well established concept for enforcing end-to-end traceability through a system's components. There are various nuances in programs where the input variables explicitly and implicitly contribute to the program's output. Consequently, the propagated labels might vary depending on the variable's values.

AI is the study of making computers reason like humans, and the sub-domain of ML contain many models for learning to execute complex objectives with high accuracy. Features compose data points that are created through measurements, preprocessing, selection, and extraction before becoming a model's input. The complete set of data types partly consists of image and textual data, and the feature-creation process varies from type to type. The NN model consitutes much of modern applications for high-performant ML and is the quintessential black-box model. NNs consists of layers containing weights for the features that they represent, and the training process is an iterative search-based algorithm for weights that optimally make predictions with minimal error. These networks can learn non-contributing patterns in the data, which is a valueable factor to consider for conventional influence-based label propagation mechanisms. However, a LPM cannot rely on the presence of redundancies in the network because of the robust techniques often employed during training to ensure non-redundant representations. Hence, label propagation rules for the NN's output computation is not readily formulated compared to the primitives from label propagation rules in programs.

XAI methods produces explanations for a model's output, locally or globally. These explanations aim to alleviate the black-box property of ML models. Some techniques are specific or agnostic to which model they explain. There are various ways an explanation can manifest and this thesis exclusively concerns local explanations. Related work on XAI is booming in response to compliance issues with the increasing applications of high-performant black-box models. This thesis considers a subset of them such as LIME, Anchor and IG.

A LBAC model provides label semantics to evaluate the changes in the propagated labels for different LPMs. This semantic is the restrictiveness of labels. An example application is any system using ML with security concerns regarding access to data. Information flows from the ML model to unauthorized entities in these systems might occur. A faithful LPM should enable these systems to discover and prevent such flows by inspecting the propagated labels of the model's output.

3

Design & Implementation

Data restriction policies risk violation with the use of Machine Learning (ML), and data labels are means to implement restriction policies. A procedure computing an output propagate the input labels to the output in order to continuously enforce data restrictions. Influence-based label propagation policies state how and which labels should propagate to the output depending on the contribution of the input toward the output. These policies are difficult to enforce when applied to Neural Network (NN), and the use of NNs is widespread throughout Deep Learning (DL). DL is the sub-domain of ML offering less explainable and more powerful models than other types of ML available. The scientific field of Explainable Artificial Intelligence (XAI) provides methods and techniques for alleviating the black-box property of models. This thesis proposes a Label Propagation Mechanism (LPM) that propagates labels from input to output with an influence-based policy guided by XAI methods.

This chapter describes how LPMs employ explanations created by XAI to propagate labels. This is the answer to the first research question posed in chapter 1.3 regarding the application of explanations for label propagation from input data to model outputs. We propose to compare LPMs using *restrictiveness of labels* assigned to the outputs as a measure. This measure enables us to answer the second research question II regarding the observation of how the propagated output labels change with various models and XAI. This chapter is structured as follows:

Section 3.1 specifies how the input is labeled before the model prediction and explanation.

Section 2.5 details lattices of labels, the lattice semantics, and how this is measurable as label restrictiveness.

Section 3.3 elaborates on how the LPM applies each XAI explanation to propagate the input labels.

Section 3.4 summarizes the contents of this chapter.

3.1 Labeling Strategy

Our labeling strategy is feature-wise labeling. Each feature in the data point is assigned a label, though not necessarily a unique label. Labeling the input at this level makes sense because many XAI techniques create feature-wise explanations, as discussed in chapter 2.3.1. The LPM propagates a set of labels from the data features that are important according to the XAI explanation of the model prediction. Figure 3.1 demonstrates an example set of output labels propagated from the input based on an explanation considering the features and the model. The propagated label l_y is the union of the labels l_x for an input x that is influential $x_i \implies \hat{y}$ to the output $\hat{y}$. This example explanation states that the first, third, and fifth features are pertinent for the model output, and the second and fourth are not. That is, $\{x_1, x_3, x_5,\} \implies \hat{y}$ denotes the pertinent features and $\{x_2, x_4,\} \not\Rightarrow \hat{y}$ denotes the non-pertinent features. Hence, the output label does not contain l_{x_2} or l_{x_4}, instead marked with the bottom label $\perp$.

There are various levels of granularitiy to associate labels with data. Feature-wise labeling is one level where the granularity depends on the data, and the labeling is likely excessive for most applications. Data often measure one or more ontologies, so labeling data regarding these higher concepts is more sensible than individual measures. Therefore, we label the data ontologies via the more expressive form of feature-wise labeling.

There are tools for identifying and annotating data with labels. One of the traditional ML applications is detecting which features belong to which labels [53, YOLO, Redmon et al., 2016]. We employ manual labeling, which is a design choice. This is because the labeled ontologies in our experiments should demonstrate a practical and realistic application. This is *not to say* that labeling through ML is impractical, but that the restriction is motivated by the bias of the notion. Consider designing an experiment where the features are labeled with an ML model, XAI methods should explain this model predictions, and

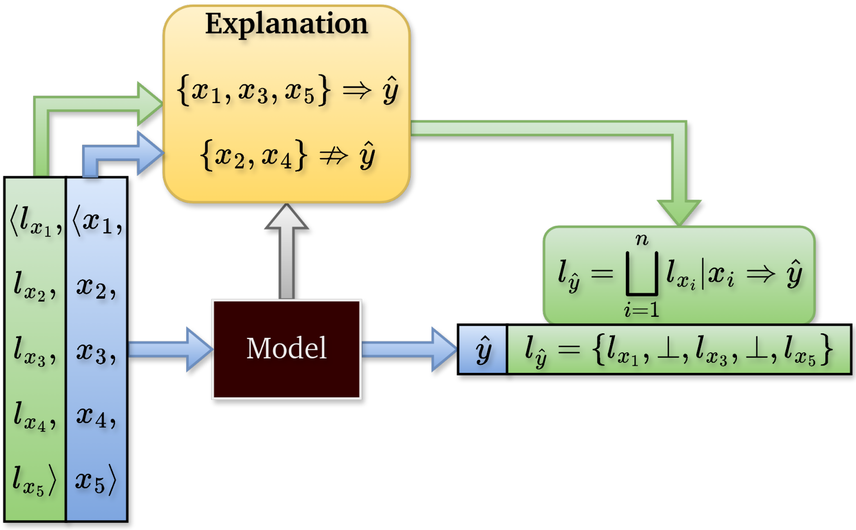

Explanation
$\{x_1, x_3, x_5\} \Rightarrow \hat{y}$
$\{x_2, x_4\} \not\Rightarrow \hat{y}$
$\langle l_{x_1}, l_{x_2}, l_{x_3}, l_{x_4}, l_{x_5}\rangle$
$\langle x_1, x_2, x_3, x_4, x_5\rangle$
Model
$l_{\hat{y}} = \bigsqcup_{i=1}^{n} l_{x_i} | x_i \Rightarrow \hat{y}$
$\hat{y}$
$l_{\hat{y}} = \{l_{x_1}, \perp, l_{x_3}, \perp, l_{x_5}\}$

Figure 3.1: Demonstrates an input vector of five features for a model with an adjoining label propagation to the output based on an XAI explanation.

an analysis should compare the restrictiveness of the propagated labels. The propagated labels of the model that assigned those labels might exhibit an inherent bias, which would make such a comparison with other models unfair and potentially lead to erroneous conclusions.

Image Labeling

The **image data** features are the pixels or color channels. Labeling each pixel is more sensible for real applications than labeling the color channels. We manually select and label superpixels for images, and color channels receive the same label as their composed pixel. The superpixels are the bounding boxes around the object we intend to highlight with the label semantic. The lattice of labels used with images is the one from figure 2.7b in chapter 2.5. Segmentation algorithms can automatically label images, such as Simple Linear Iterative Clustering (SLIC) [6, Achanta et al., 2010]. Segmented superpixels still require manual attention to which label should be associated with which segments, e.g., which superpixels compose faces in the image. Hence, there are few options outside manual labeling since the thesis does not use ML to classify the desired objects in images. We aim to present plausible examples for label propagation of images and, therefore, label superpixels of specific images that a model might find important for its prediction. Figure 3.2 is an example of pixel-wise labeling where a ML model might have learned to focus on the labeled superpixels. The colored bounding boxes corresponds to the annotated labels in the bottom of the figure. The entirety of the Snow Leopard's head

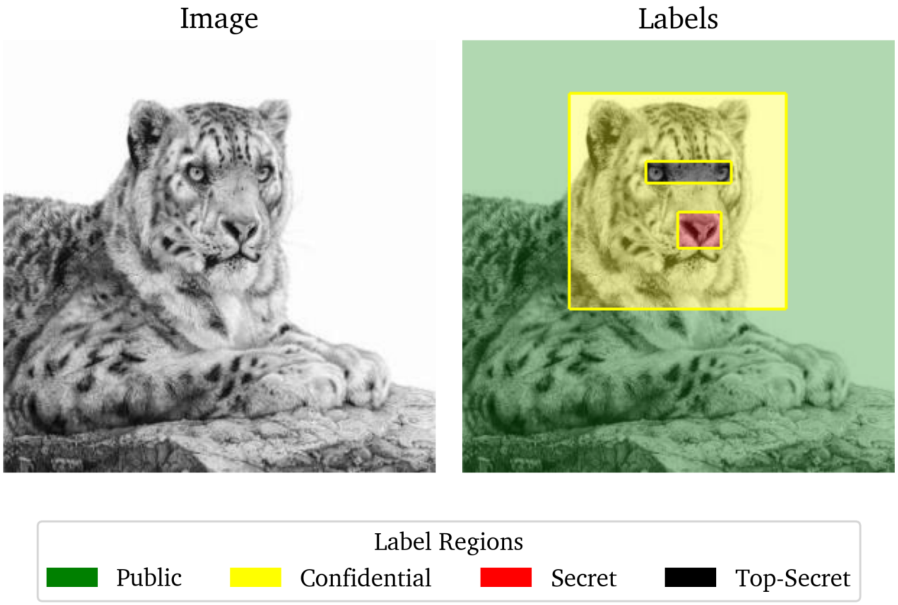


Figure 3.2: Demonstrates a labeled image with four labels, each more restrictive than the last. This image is a sample from ImageNet [7, Howard et al., 2018]. The ground truth for this data point is a *Snow Leopard*.

(yellow) is confidential, the nose (red) is secret, the eyes (black) are top-secret, and the remaining pixels (green) are labeled public.

Text Labeling

For **textual data**, we consider the text words as the individual features to label. We label sentences as the subject of interest in textual data with a LBAC smaller than figure 2.7b from chapter 2.5. The lattice is a reduced version of the *public*, *confidential*, *secret* and *top-secret*, and only consists of two labels *low* and *high*.

Each sentence is labeled based on whether its words occur in a labeled list of words, which we refer to as *keywords*. The words in the keyword list are curated based on the dataset semantics. A sentence receives the keyword list label if any word from the sentence occurs in the keywords. The words then inherit the sentence label, so each word is labeled. Hence, the sentence structure does not affect the labels of the words. Consider the two sentences "*They only asked me to tell them*" and "*They asked only me to tell them*". The structure of the sentences creates different sematical meanings while the word labels will be the same. We use this labeling because some XAI explanations for text report positional words as influential, as opposed to every occurrence of the word. Furthermore, some XAI methods operate on data with a per-word basis. We might observe varying restrictiveness measures for different explainability methods with this labeling strategy. This semi-automated labeling process is simple and efficient, and does not emphasize accuracy with respect to the data semantics. This labeling might be less accurate than using ML (such as Large Language Models (LLM)) to label the input sentences. However, we achieve labeled sentences that are *sensible* based on human evaluation. Sensibility should entail that the semantic meaning of the sentence and the semantics of the label match, as judged by the human.

Equation 3.1 expresses the labeling rule for each word w_s that occurs in a sentence s. The word is assigned a high label l_{w_s} if the word w_s occurs in the keywords $\mathcal{K}$. The word label is still high if not a keyword but there exists a neighbor word w_s' from the same sentence s, which is a keyword. Otherwise, every word label is low in sentences containing no keywords.

$$l_{w_s} = \begin{cases} l_{\text{high}}, & w_s \in \mathcal{K} \vee \exists w_s' : l_{w_s'} = l_{\text{high}} \\ l_{\text{low}}, & \text{otherwise} \end{cases} \tag{3.1}$$

Consider a textual data point $x =$ "*The first sentence. The second sentence. The third sentence.*" and a keyword list $\mathcal{K} = \{first, third\}$. The resulting word-wise labels are $l_x = \langle l_{\text{high}}, l_{\text{high}}, l_{\text{high}}, l_{\text{low}}, l_{\text{low}}, l_{\text{low}}, l_{\text{high}}, l_{\text{high}}, l_{\text{high}} \rangle$. There are nine words in x, three words per sentence, and there are nine corresponding labels l_x where the first label refers to the first word, and so on. Figure 3.3 illustrates the label assignment of this example. Notice that the label for "*The*" in the second sentence is low even though the same word elsewhere is high. Thus, we do not put emphasis on the meaning of the word except for the keywords. This allows LPMs to propagate labels with positional proximity to words with meaning that we actually emphasize. This text labeling strategy is motivated by the initial observations that some textual data explanations made strange emphasis on words such as *the*, *there* and *an*. Therefore, this sentence labeling is sensible regarding the dataset and the possible variation in label propagation for models that weight the position of words in texts. Figure 3.4 is an experiment example of a labeled movie review. The sentences are labeled as *Spoiler* versus *Non-spoiler* and the labels are shown by the sentence color. These spoiler sentences are labeled as such because they contain the term "*movie*". This uses the same *low-* and *high* lattice as before with label names that are more appropriate for the data.

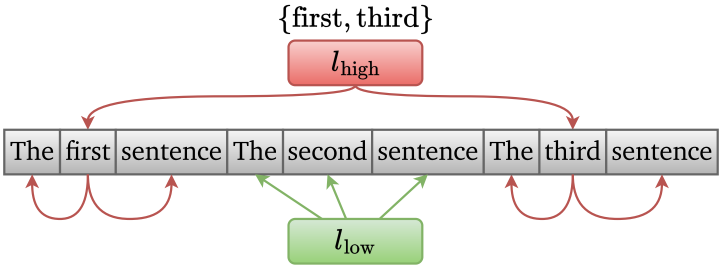
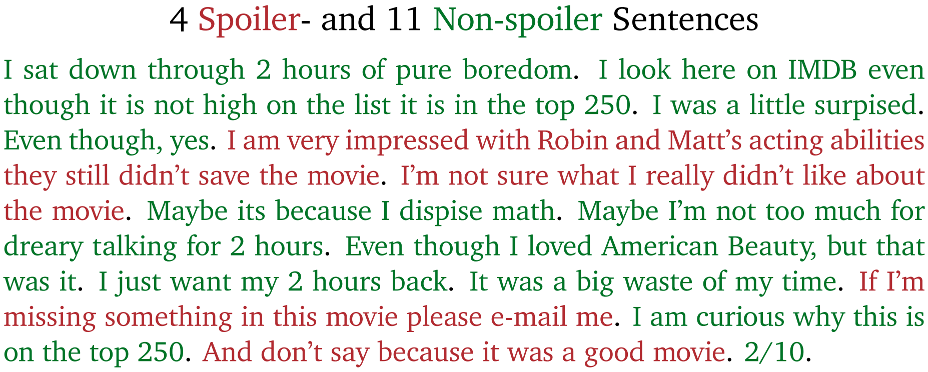


Figure 3.3: Depicts the word-wise labeling of a textual data point containing three sentences. The second sentence receives the low label because there are no words in the list labeled high, and vice versa for the first and third sentences.



4 Spoiler- and 11 Non-spoiler Sentences

I sat down through 2 hours of pure boredom. I look here on IMDB even though it is not high on the list it is in the top 250. I was a little surpised. Even though, yes. I am very impressed with Robin and Matt's acting abilities they still didn't save the movie. I'm not sure what I really didn't like about the movie. Maybe its because I dispise math. Maybe I'm not too much for dreary talking for 2 hours. Even though I loved American Beauty, but that was it. I just want my 2 hours back. It was a big waste of my time. If I'm missing something in this movie please e-mail me. I am curious why this is on the top 250. And don't say because it was a good movie. 2/10.

Figure 3.4: Demonstrates a movie review from the Internet Movie Database (IMDb) Review [43, Maas et al., 2011] with sentences labeled spoiler- (red) versus non-spoiler (green).

3.2 Comparing LPM Restrictiveness

Lattice-Based Access Control (LBAC) labels possess an inherent measure of restrictiveness, where one label might allow an operation that another prohibits. The LPM applies XAI explanations to guide the propagation policy. The LPM propagates the set of labels for features reported as influential by the explanation of the ML model prediction. We consider a LPM different from another if the underlying XAI method is not the same. Hence, a LPM can be more restrictive than another depending on the output labels, and by extension, different XAI methods might exhibit varying restrictiveness. Given some data point x, model f, and pair of LPMs ϵ^1 and ϵ^2. The LPM ϵ^2 is more restrictive than ϵ^1 for a particular x and f if the label restriction predicate in equation 3.2 holds. Meaning the propagated labels $\epsilon^2_f(x)$ is *more restrictive* than $\epsilon^1_f(x)$. This notation now uses the strict subset $\sqsubset$ instead of $\sqsubseteq$ due to the *more restrictive* requirement.

$$\epsilon^1_f(x) \sqsubset \epsilon^2_f(x) \tag{3.2}$$

Consider the lattice of labels from chapter 2.5 lattice in figure 2.7b, and the propagated labels $\epsilon^1_f(x) = \{$*public, confidential*$\}$ and $\epsilon^2_f(x) = \{$*secret, top-secret*$\}$. We denote the most restrictive label of a set with the ceiling notation $\lceil \epsilon_f(x) \rceil$. The most restrictive labels are then $\lceil \epsilon^1_f(x) \rceil = \{$*confidential*$\}$ and $\lceil \epsilon^2_f(x) \rceil = \{$*top-secret*$\}$. Equation 3.3 expresses the most restrictive label notation combined with equation 3.2. Substituting the propagated labels into this equation gives $\{$*confidential*$\} \sqsubset \{$*top-secret*$\}$. Thus, the information flow $\epsilon^1_f(x) \Rightarrow \epsilon^2_f(x)$ is allowed, which indicates the LPM on the right-hand side ϵ^2_f is more restrictive than ϵ^1_f.

$$\lceil \epsilon^1_f(x) \rceil \sqsubset \lceil \epsilon^2_f(x) \rceil \tag{3.3}$$

We can compare how restrictive the propagated labels are between LPMs applying different XAI methods. Label restrictiveness is essential in answering the second research question II stated in chapter 1.3 since it can measure change in propagated labels. Therefore, this thesis employs labels from lattices to provide an application-specific measure for the implementation of the LPM.

3.3 Explanation & Propagation

This thesis uses XAI for guiding the LPM and explanations concern only local explanations from definition 4. The motivations for how related XAI methods give explanations vary and might not conform to any shared interface. *Alibi* is a ML explainability library meant to provide such an interface and bridge the gap between research methods and industry application, as discussed in chapter 2.4.4. This thesis uses a feature-wise labeling strategy and Alibi's Application Programming Interface (API) to output feature statistical explanations, as detailed in chapter 2.3.1 under bullet-point I.

Feature statistical explanations are preferable due to the extra dimension of information in the numerical weighting of the feature importance. However, some explanations only specify which discrete features are important for the prediction. That is, the feature-wise explanations do not always contain a contribution weight. Therefore, the LPM puts minimal emphasis on evaluating feature contribution weights when applying explanations to propagate labels. Exploring parameters and contribution weights of the explainability methods is out of scope for this thesis, per the research questions posed in chapter 1.3. Moreover, comparing explainability method restrictiveness is potentially biased if some LPMs consider more than the feature presence in the explanation. For the XAI methods that provide them, exploring the contribution weights of explanations is left to future work.

The explanations that contain feature contribution weights tend to offer *negative* and *positive* contributions. A positive contribution for a prediction means the presence of that feature made the model more confident regarding the final decision of the output. Likewise, a negative contribution means the presence made the model less confident, though not necessarily enough to change the final output. While the weights of the explanations are not subject to consideration, we can still consider whether to propagate labels of negatively contributing features. One could argue that a negative influence does not contribute *towards* the output, but *against*, and that propagating negatively contributing feature labels is conservative because it is non-influential *towards* the output. This subject concerns the argument regarding how a LPM should consider the influence of variables. From definition 1, an influence-based propagation policy states the labels of *influential* input data should propagate to the output. Without any further disambiguation between types of influence, the question of how negative contributions should guide a LPM is a topic for future work. Hence, this thesis considers both negative and positive feature contributions as influential, so their labels should propagate to the output.

The following sections detail the method-specific steps a LPM makes for applying the explanations to label propagation. The sections address the first research question I from chapter 1.3.

3.3.1 LIME Propagation

An **image explanation** from Local Interpretable Model-agnostic Explanations (LIME) consists of the mask for pixels that contribute to the model output. The contribution mask contains the values $\{-1, 0, 1\}$ where pixels influence the model output with negative, neutral, and positive contributions, respectively. The method accepts a minimum weight parameter each pixel must exceed to include the pixel in the explanation. This parameter defaults to 0 such that any contributing feature is a part of the explanation. The LPM propagates each pixel label, except those explanation that are marked with a zero in the contribution mask.

LIME samples training data around the data subject to explanation to train the explainer model. For **textual data**, this sampling is a per-word procedure, so the explanation consists of words and their contribution weights for the model output. The explanation optionally encodes the position for which word occurrence in the text each weight applies. The LPM propagates the label of each positional word included in the explanation. This application of the explanation should propagate labels of words with close proximity to the words we emphasize, as per the labeling strategy detailed in section 3.1.

For textual data, the number of words to include is an input parameter such that the top-*n* words with the highest weights compose the explanation. From our experience, LIME attributes at least some contribution to each word. This sensitivity is possibly related to the text model we use. Consequently, LIME does not exclude features from the explanation unless specified by the top-*n* parameter. This lack of exclusion requires users to assess the quality of the explanation and filter out irrelevant words. The motivation of LIME is to provide users with an understandable explanation of the model output, so including as much information as possible is reasonable. However, an explanation containing every word will propagate every word label, which is undesirable due to the problem statement in chapter 1.1. Our solution is experimenting with various top-*n* parameters until the explanation for the prediction of the data point is *sensible*. This experimentation implies human evaluation of the explanations to judge their sensibility. Sensibility should entail that the subject of the text and the explanation of important words fit the model prediction, as judged by the human.

3.3.2 Anchor Propagation

An **image explanation** from Anchor is a copy of the explainee image where the 0 pixel intensities indicate non-contributing pixels. The method uses image segmentation prior to sampling perturbations of the explainee image, as discussed in chapter 2.4.2. The explanation consists of the superpixels necessary in the image for the model prediction to match with the prediction of the explainee image. The LPM propagates the explainee image labels with corresponding non-zero pixels in the explanation.

For **textual data**, the explanation consists of the words that must be present in the text to satisfy the anchor. Alibi's source code and documentation do not specify whether the explanation words are positional. That is, whether or not repeating words in an explanation (e.g., [*top, secret, top, document, top*]) indicates the first, second, and third positional occurrence of the word *top* in the text. This thesis conservatively assumes Anchor explanations for text data indicate that the word occurrence *anywhere* in the text is sufficient. The LPM propagates the label of any word occurring in the text where the word appears in the explanation. Hence, the LPM applies explanations from LIME and Anchor differently since any word in an Anchor explanation propagates to the output, but not necessarily with LIME explanations.

Anchor does not explain predictions with explicit positive versus negative contributions. An anchor is the necessary feature values required to exist in a data point for the model prediction to match the prediction of the explainee. In that sense, the explanation contains exclusively unweighted and positive contributions. The high probability parameter τ discussed in chapter 2.4.2 relates to the confidence of the explanation. Lowering the confidence requirements relaxes the search restrictions for anchors and implies the anchors are more numerous and less reliable. Thus, higher-confidence anchors are preferable, and this thesis uses the default confidence level of 95%.

3.3.3 Integrated Gradients Propagation

Explanations from Integrated Gradients (IG) consist of feature-wise contribution weights, alternatively referred to as feature *attributions*. For **images**, IG attribute contributions to each color channel. The image labeling strategy concerns individual pixels, so the attributions of the color channel must consolidate into one-pixel attributions. Thus, an individual pixel attribution is the mean attribution of the pixel channels. The LPM propagates each label of the explainee image pixels, except where the corresponding pixel attribution is close to zero with ±0.001 as the closeness tolerance. The attributions are sums of model predictions for multiple images between the baseline and the explainee image, and the attributions close to zero does not contribute. Although, this sum often attribute some contribution to each pixel, which is a similar complication like LIME for text data. An LPM needs some threshold tolerance and we found ±0.001 yields explanations that does not propagate every label. The LPM is sensitive to this parameter and using a tolerance lower or higher than ±0.001 will propagate less and more labels, respectively.

3.4 Summary

A LBAC model provides valuable label semantics for answering the second research question. That is, which observations can be made for evaluating the change in the propagated labels for different LPMs. This semantic is the restrictiveness of labels. An example application is any system using ML with security concerns regarding access to data. Information flows from the ML model to unauthorized entities in these systems might occur. A faithful LPM should enable these systems to discover and prevent such flows by inspecting the propagated labels of the model outputs.

The LPM propagates a set of labels from the feature-wise labeled data to the ML model output. LPMs are different based on the XAI method that guides their label propagation policy. The explanation interfaces vary, and this thesis focuses on feature-wise influence explanations for both positive and negative contributions. Each application of the explanations from XAI method requires unique attention by the LPM to choose which labels to propagate. This chapter has described these attentions for the data types and explainability methods applied in this thesis: LIME, Anchor, and IG.

4

Experimentation

This chapter describes the step-wise design and execution of our experimentation. We detail the computer resources and the overall structure for each experiment, alternatively referred to as a *test case*. The number of experiments depends on the possible combinations of experiment parameters. We consider the experiment parameters from the second research question II in chapter 1.3. These parameters combine data, the Machine Learning (ML) model, and Explainable Artificial Intelligence (XAI) method. The experiments follow a similar structure we describe before listing each test case. Every experiment aims for self-containment, so every bit of information required to understand the specific test case is readily available when later referenced. The chapter's outline is as follows:

Section 4.1 details the experiment setup and justifies the relevance of commercial hardware regarding ML.

Section 4.2 describes the data and corresponding model that superposes the following test cases.

Section 4.3 details the test cases for each of the image experiments and evaluates the results.

Section 4.4 details the test cases for each of the text experiments and evaluates the results.

4.1 Computer Hardware Specifications

The experiments use a computer with the following specifications: A 13th Gen Intel(R) Core(TM) i7-13700 Central Processing Unit (CPU) with 16 cores and 24 logical processors. The CPU performance- and efficient-core *base* frequencies are 2.10 and 1.50 GigaHertz (GHZ), respectively. The performance- and efficient-core *max turbo* frequencies are 5.20 and 4.10 GHZ, respectively [33]. The memory size is 128 GigaBytes (GB) of Random Access Memory (RAM). The Graphics Processing Unit (GPU) is an NVIDIA GeForce RTX 3070 with 8 GB RAM and 64 GB Shared GPU usage [48]. The Operating System (OS) specification is Windows 11 Pro version 23H2 where we use Windows Subsystem for Linux 2 (WSL2) with Ubuntu 22.04.4 LTS as the Linux distribution.

Applicability of Commercial Computer Hardware

The scientific developments in ML for the past years have led to increased contributions from the open-source community. An anonymous employee at Google wrote an internal memo, which later leaked to the public. This memo concerns the company's ability to compete with the open-source community regarding high-performant Large Language Models (LLM) in the future [51]. Companies could previously dominate this market because of the computer hardware requirements for training Large Language Models (LLM). This hardware often consists of clusters of computers, which are not commercially or readily available to the open-source community. The developments that countered this imbalance can be summarized as follows: The leaking of Large Language Model Meta AI (LLAMA), so the open-source community can operate with a state-of-the-art LLM [74, Touvron et al., 2023]. *Weight quantization* of the model parameters such that the precision format of the weights is more memory efficient (and less accurate), such as from 32-bit into 16-bit floating-point weights [20, Dettmers et al., 2022]. The *Low-Rank Adaptation (LORA)* technique for fine-tuning a pretrained model without fine-tuning the complete set of weights [30, Hu et al. 2021]. Fine-tuning with smaller sets of *quality data* decreases the overall time required for fine-tuning instead of using larger quantities of data.

The experimentation in this thesis follows the notion of commercially viable computer hardware for state-of-the-art ML. That is, we aim to provide results that are reproducible by the open-source community on accessible computer hardware. The results should give the reader an overview of which explainability methods, datasets, and ML models apply to problems limited by this hardware. Therefore, the experiments are restricted to execute as much as possible on the GPU even though some experiments might fail due to resource limitations. These limitations should be considered part of the findings.

4.2 Methodology

This section describes the data and models that the XAI methods create an explanation for, which the LPM then applies to propagate labels. We select well-known datasets from previous applications in the ML field. This selection strategy is preferable because understanding test case results is easier if the data is familiar and frequently used in scientific publications. We prioritize image- and textual (not tabular) data because of the problem statement regarding propagating training data labels, which do not scale. This limitation is because the XAI methods able to explain tabular data often require training data as a parameter or at least a dataset representative of the feature distributions. The methods use these tabular features to construct a per-column distribution to sample data points around the explainee data point. Therefore, the experiments do not include tabular data due to this prioritization. Additionally, we exclude other data types (such as time series) since the Application Programming Interface (API) of explainability methods restrict themselves to images, text, and tabular data.

The experiments use Python as the programming language for executing the test cases. Python possesses a well-supported ML infrastructure, and Alibi is a Python package, so the programming language is a natural choice. We use the Keras API [13, Chollet et al., 2015] with the TensorFlow framework as background for every ML purpose [5, Abadi et al., 2016]. There are some applications of the Scikit-learn package [52, Pedregosa et al., 2011], though these are minimal and primarily for utility purposes. The models are either pretrained and publically available through the Keras application API or created and trained with Keras. The experiment only uses models with a classification objective (as opposed to regression) due to the nature of the selected data.

4.2.1 Datasets

The experiments consist of one dataset for image- and textual data. We employ the ImageNet and Internet Movie Database (IMDb) dataset for these data types, respectively. The following sections describe these datasets.

Image Data – ImageNet

ImageNet is the dataset for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [7, Howard et al., 2018]. The challenge's objective is object detection and -classification. The uncompressed dataset size is 159 GB and contains 1431167 images. The dataset is split into a training set of 1281167-, a validation set of 50000-, and a test set of 100000 images. The description of the competition mentions a different validation- and test set for evaluating the submissions to the challenge, and these sets are not the same as just mentioned. The dataset contains manually annotated ground truth with bounding boxes and identifiers for the objects in the images. The ground truth contains 1000 different object categories. Only the training set has associated ground truth; therefore, we only consider these images for our experiments.

Relevance of ImageNet ImageNet is an annual and popular challenge in the ML community. Numerous scientific publications created model architectures tailored solely to perform well with this dataset. The motivation is that if a model performs well with ImageNet, it might serve as a solid basis for other applications in computer vision. Thus, ImageNet is the quintessential image dataset for testing a model architecture's capabilities. The black-box models with a Neural Network (NN) constitute the winning category of the ILSVRC leaderboard.

ImageNet Management and Preprocessing The Keras preprocessing module manages the reading of images from storage into memory. We use this dataset for the sole purpose of object classification. The preprocessing API offers an *image-dataset-from-directory* utility procedure, which interprets the dataset's file structure into an image dataset with classification identifiers as ground truth. The images are in Joint Photographic Experts Group (JPEG) format, and the resolution varies, so every image is resized into $299 \times 299 \times 3$ with bilinear interpolation and Red Green Blue (RGB) color channels. We chose this resolution because it is compatible with the input shape of the model we initially experimented with. Hence, despite differing input shape requirements, we apply this image shape to every model in our experiments.

We instruct Keras to shuffle the data with a static seed of 42 for the reproducibility of the experiments. We normalize each pixel value of the images to values between $[0, 1]$. These experiments only concern *one* explainee image, so the image experiments do not provide extensive results regarding the variation of the ImageNet dataset. Figure 3.2 from chapter 3.1 shows the explainee image and the labeled pixels from the bounding boxes. The labels stems from the Lattice-Based Access Control (LBAC) model in figure 2.7b from chapter 2.5. We employ this image, labeled superpixels, and lattice for every image experiment. We exploit the many readily available ImageNet models to experiment with the XAI methods across several models.

Textual Data – IMDb Review

The Internet Movie Database (IMDb) Review dataset contains 50000 movie reviews collected from the IMDb website [43, 3, Maas et al., 2011]. From now on, we refer to this dataset as IMDb and the website as the IMDb website. The ground truth for the data is whether the sample is a positive or negative review. The dataset omits neutral reviews, i.e., reviews with ratings 5 and 6 out of 10 are not represented. There are 50% positive- and negative reviews, and no more than 30 reviews concerning each movie.

Relevance of IMDb The authors of the dataset published the dataset as a benchmark standard for future work in Natural Language Processing (NLP) tasks. As of writing this thesis, the dataset paper is cited by 5700 related works as reported by Google Scholar. Many of these works concern Deep Learning (DL) techniques, specifically models based on the Transformer architecture [78, Vaswani et al., 2017]. Hence, we consider this dataset as tried, tested, and highly applicable to the experiments.

Preprocessing Reviews IMDb comes in Comma-Separated Values (CSV) format and we use Pandas for handling the data [49, Reback et al., 2020] [79, McKinney et al., 2010]. We split the 50000 reviews into 70% training set, 10% validation set and 20% test set. These split fractions result from experimentation during development and have no inherent justification behind their choice. We preprocess each review by the following steps in strict order:

1. Removing HyperText Markup Language (HTML) code.
2. Removing internet hyperlinks.
3. Setting letters from *A* through *Z* to lowercase.
4. Replacing whitespace characters with one space character.
5. Removing apostrophe characters.
6. Replacing any punctuation characters with one space character.
7. Removing any non-printable characters (such as characters with byte values \0x93).
8. Replacing any continuous sequence of space characters with one space character.

Labeling Spoiler Review Sentences The reviews are labeled through word-wise label assignment, and words receive the labels of the sentence they occur in, as discussed in chapter 3.1. We use a subset of the lattice of labels in figure 2.7b from chapter 2.5. There are two labels in this subset lattice referred to as *spoiler*- and *non-spoiler* labels, replacing the *confidential* and *public* labels, respectively. Hence, the keyword list consists of movie terms such that any mention of a keyword applies a spoiler label to that sentence. The keywords are generated in cooperation with GitHub Copilot [1]. The following lists the keyword terms and categories applied for assigning *spoiler* labels to words in the reviews:

Subjects: *actor, actress, audience, character, critic, director, editor, fans, producer, reader, reviewer, spectator, viewer, watcher, writer*.

Production: *cut, editing, frame, screen, shot*.

Audio: *lyrics, rhythm, music, song, sound, soundtrack*.

Directing: *angle, atmosphere, credits, dialogue, line, location, quote, scene, script, setting*.

Miscellaneous: *film, movie, cinema, title*.

Sampling Reviews for Label Propagation We sample four reviews from the test set for labeling and subsequent label propagation. The reviews are chosen based on considerations regarding the model predictions. The considerations concern the *correctness*- and *confidence* of the classification regarding the reviews. We select one review from each of the four possible combinations with these considerations, which are listed as follows:

- Correct classification and high confidence.
- Correct classification and low confidence.
- Wrong classification and high confidence.
- Wrong classification and low confidence.

The four explainee reviews are the polar extremes of their categories. The selected sample from the category of wrong classifications with low confidence is the misclassified review with the *lowest* confidence, and vice versa for the other categories. The review selection applies one additional constraint: the review must not exceed 475 words. The reason for this limitation stems from hardware. Reviews that are too long cause the explainability methods to encounter an Out Of Memory (OOM) error in the GPU. The error occurs from attempts to allocate memory upwards of 617 GB during the forward pass of the model. Therefore, there are potential reviews with higher or lower confidence than the ones selected for the experiments. The experiments are limited to execution on GPU devices, so we do not attempt to execute any failed experiments on alternative devices such as the CPU. Investigating whether the errors stem from just the model or the explainability method is a subject of future work. The limit of 475 words is not the exact boundary for when OOM errors transpire. However, this limit is the first, with no error originating from explaining the predictions of the selected reviews.

4.2.2 Models

We experiment with a wide selection of pretrained models for ImageNet classifiers. The experiment uses a Transformer classification model trained on IMDb reviews for textual data. The following section describes the models.

Pretrained ImageNet Classifiers

We apply models available through the *Keras Applications* API for image classification [14, Keras Applications]. For ImageNet, these models are numerous, and the experiments do not cover everyone in the repertoire. The selection of models to experiment with is partially motivated by familiarity with the models or the research papers proposing them. The unfamiliar models are selected based on model-specific considerations we wish to cover. During the discovery phase regarding related work for this thesis, we did not prioritize exploring which models are the better candidates to explain with XAI. Thus, the unfamiliar models part of this experimentation are tested with a modicum of *blind application*. This means the resulting analysis might lack insight from a complete understanding of the unfamiliar models and how they relate to the ImageNet dataset. This section does not detail architectural specifics regarding the models beyond listing their common points of consideration.

Model Selection We weigh which models are suitable for experimentation with some high-level considerations. These considerations are meant to sufficiently answer the second research question regarding changes in propagated labels with different models. The goal is to provide a representative sample set of ML models for image classification. The experiment results should contain the following considerations:

- Models from different families of architectures.
- Models that originate from earlier- and recent developments in ML.
- Models with few- and many parameters.
- Models of varying ImageNet accuracy.

Selected Models Table 4.1 list the models considered for the image experiments [14, Keras Applications]. The first column states the model name, the size in MegaBytes (MB), and the number of weight parameters in *Millions* (M). The second column states the model's classification and prediction confidence of the selected explainee image (discussed in section 4.2.1) with correct classifications marked with ✓. The third column states whether we prepended a resizing layer to the model. The layer uses bilinear interpolation to resize from $299 \times 299 \times 3$ to the resolution listed in the column. *N/A* in the third column indicates the model does not need resizing because it is compatible with our standard $299 \times 299 \times 3$ ImageNet resolution, as discussed in section 4.2.1. The fourth column lists the top-one and -five ImageNet validation set accuracies for the model. Keras is ambiguous regarding whether the reported accuracies stem from the ILSVRC's submission validation set or the validation split of ImageNet's challenge dataset. Top k accuracy is the metric of how frequently the model's classifications contain the correct class in the first k predictions sorted from high to low regarding the class-wise confidence. Therefore, a top-five accuracy higher than a top-one is sensible. In the top-five accuracy, a model can make four mispredictions and still count the prediction as accurate. N/A in the fourth column indicates there is no entry for the model in the Keras documentation even though the API makes the model available. We attempted to compute these accuracies by using a validation split of the training set. However, the resource restrictions proved an obstacle in computing these accuracies.

Initial Evaluation The second column in table 4.1 relates to the model classification of the explainee image in figure 3.2 from chapter 3.1. The explainee image clearly does not represent the training data of some models. This is apparent from the misclassifications for some of the models, e.g., EfficientNet-V2-S misclassififies the *Snow Leopard* image as a *Matchstick* with a low confidence of 1.07%. This misinterpretation holds across models from different families, e.g., ResNet-RS-50 makes a similar mistake as EfficientNet-V2-S. The explainee image only matches with the ResNet V2 and Inception models. These discrepancies are not an intentional part of the experiment design, and this thesis does not investigate this until its completion. The low performance is unproblematic for creating explanations of the model predictions. However, we might lose the connection to practical and realistic labeling for some models in the context of label propagation, as discussed in chapter 3.1.

We use 299 × 299 × 3 as the ImageNet resolution, though some models differ in input shape due to their architecture. Hence, we prepend a resizing layer to the model to bridge the shape inconsistencies. There is a correlative indication between the models with a resizing layer and their misclassification of the explainee image. Each model of the Inception family correctly classifies the explainee as a Snow Leopard and does not resize the input image. However, this reasoning is not complete enough to conclude that this resizing is the cause of the misclassifications. As a counter-argument, each V2 variant of the ResNet family successfully classifies the image and resizes from 299 × 299 × 3 into 244×244×3 resolution. Moreover, the misclassifications seem unrelated to an increasing or decreasing target resize shape. Consider that EfficientNet-V2-S and ResNet-RS-50 have similar mispredictions and an increased- and decreased resizing from 299×299×3 into 384×384×3 and 244×244×3 input shapes, respectively. Therefore, we consider the resizing layer as a valid alteration to the pretrained models.

Consideration Summary The grouping in table 4.1 demonstrates the selection consideration towards both inter- and intra-family variation of model architectures. There are four different families as we consider them: EfficientNet, Inception, ResNet, and VGG16. From the citations in the table caption, there are models spanning from 2015 to 2021. The model sizes range between 88 and 528 MB and the number of parameters range between 21.6 and 138 million. The accuracy ranges between 71.30% and 83.90% for the top-1 accuracy and from 90.10% to 97.50% for the top-5.

Transformer IMDb Classifier

We use the Transformer architecture as part of the classifier model for the IMDb reviews [78, Vaswani et al., 2017]. Figure 4.1 depicts the layer-wise composition of the model. We use the Keras API for each of the specific layers. The text vectorization layer tokenizes words into sequences of integer tokens. The *max-token* parameter of the tokenization layer is set to 10000, and the *output-sequence-length* is 256. The embedding layer vectorizes the integer tokens from the text vectorization layer. The embedding *input-dimension* and *output-dimension* parameters are set to 10000 and 256, respectively. The transformer receives input from the flattened embedding output and performs the heaviest part of the forward pass. Finally, the output Fully Connected Layer (FCL) decides whether the review is positive or negative with two neurons based on the transformer output.

Model Name **Size & Parameters**	**Classification** **& Confidence**	**Resizing From** $299 \times 299 \times 3$	**Top** 1 **& 5** **Accuracy**
EfficientNet-V2-S 88 MB & 21.6 M	Matchstick 1.07%	$384 \times 384 \times 3$	83.90% (1) 96.70% (5)
EfficientNet-V2-M 220 MB & 54.4 M	Nematode 0.15%	$480 \times 480 \times 3$	85.30% (1) 97.40% (5)
EfficientNet-V2-L 479 MB & 119.0 M	Nematode 0.19%	$480 \times 480 \times 3$	85.70% (1) 97.50% (5)
Inception-ResNet-V2 215 MB & 55.9 M	Snow Leopard 89.12% ✓	N/A	80.30% (1) 95.30% (5)
Inception-V3 92 MB & 23.9 M	Snow Leopard 84.02% ✓	N/A	77.90% (1) 93.70% (5)
Xception 88 MB & 22.9 M	Snow Leopard 75.44% ✓	N/A	79.00% (1) 94.50% (5)
ResNet-50 98 MB & 25.6 M	Nematode 13.07%	$224 \times 224 \times 3$	74.90% (1) 92.10% (5)
ResNet-50-V2 98 MB & 25.6 M	Snow Leopard 99.98% ✓	$224 \times 224 \times 3$	76.00% (1) 93.00% (5)
ResNet-RS-50 136 MB & 123.0 M	Matchstick 0.68%	$224 \times 224 \times 3$	N/A
ResNet-101 171 MB & 44.7 M	Cleaver 6.75%	$224 \times 224 \times 3$	76.40% (1) 92.80% (5)
ResNet-101-V2 171 MB & 44.7 M	Snow Leopard 99.89% ✓	$244 \times 244 \times 3$	77.20% (1) 93.80% (5)
ResNet-RS-101 243 MB & 123.0 M	Matchstick 1.28%	$244 \times 244 \times 3$	N/A
ResNet-152 232 MB & 60.4 M	Cleaver 4.42%	$244 \times 244 \times 3$	76.60% (1) 93.10% (5)
ResNet-152-V2 232 MB & 60.4 M	Snow Leopard 99.97% ✓	$244 \times 244 \times 3$	78.00% (1) 94.20% (5)
ResNet-RS-152 331 MB & 123.0 M	Matchstick 10.05%	$244 \times 244 \times 3$	N/A
VGG16 528 MB & 138.4 M	Mosquito Net 5.87%	$244 \times 244 \times 3$	71.30% (1) 90.10% (5)

Table 4.1: Lists an overview of pretrained ImageNet classification models. The second column classification and confidence relates to the model prediction of the image in figure 3.2 from chapter 3.1.
EfficientNet [70, Tan et al., 2021].
Inception ResNet [68, Szegedy et al., 2016].
Inception V3 [69, Szegedy et al., 2015].
Xception [15, Chollet, 2017].
Residual Network (ResNet) [28, He et al., 2016].
Visual Geometry Group 16 layers (VGG16) [62, Simonyan et al., 2015].

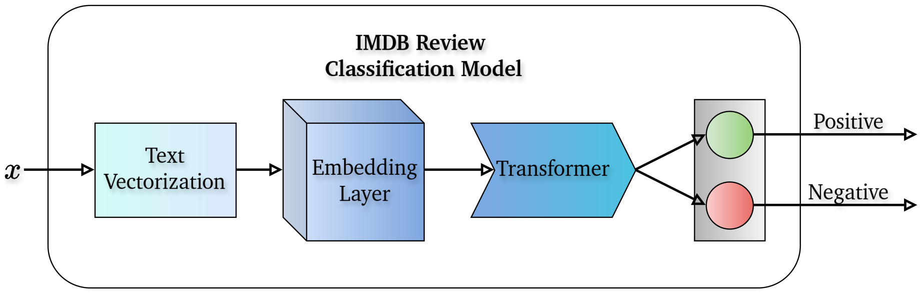


Figure 4.1: Layers in the classification model for the IMDb Review dataset.

Transformer Parameters This section details the high-level parameters for the transformer setup, though only briefly describes the transformer components and how they relate to the IMDb model. Figure 4.2 depicts the encoder-decoder architecture from the paper proposing the model, *Attention is All you Need*. We implement the transformer with some modifications but base the process on the paper. The transformer embeddings at the bottom of figure 4.2 are the embedding layer output in figure 4.1. The paper differs between the two because of the sequence-to-sequence translation objective in their paper, i.e., generating a Portuguese translation from an English input text. These input-output embeddings are identical for the IMDb model because the encoder output serves as a *context* for the decoder emphasize to classify the review instead of generating new ones based on a different input. The model uses the same positional encoding as the original work, except with the parameter $d_{\text{model}} = 256$. Transformers can consist of N encoders and M decoders, but the paper (and figure 4.2) uses the same number of encoders and decoders. Our model employs $N = 2$ for both encoders and decoders. For Multi-Head Attention (MHA), the model transformer employs the same number of heads $h = 8$. This number of heads implies the key-, value-, and query FCL dimensions are $d_k = d_v = d_q = d_{\text{model}}/h = 32$, respectively. There is a dropout layer applied to the following hidden outputs of the architecture (with a dropout chance of 10%):

- After the sum of the positional encoding and the embedding (before input to the encoder and decoder).
- After each the key-, value-, and query FCL for each MHA.
- After each Feed-Forward (FF) layer.

We use these same parameters in every MHA and FF layer for every encoder and decoder. The transformer implementation for this experiment is repurposed from a guide reproducing the transformer application in the original paper using Keras [24]. The transformer parameters result from experimentation to find the ones achieving the highest accuracy with the IMDb dataset. The presented parameters are those found during experimentation with the highest accuracy.

The transformer implementation contains one significant alteration, which must be mentioned. We discovered the transformer encoders and decoders from the Keras API are CPU bound during training. After reimplementing the transformer from the previously mentioned guide, we found the CPU bound component. This component is the decoder's first MHA layer, and the reason is the *causal masking*. This high usage of the CPU implies these operations are either incompatible- or not implemented on GPU. This implication might concern GPUs in general or just this exact hardware, though exploring this avenue is out-of-scope for this thesis. The model still manages to attain high accuracy (and the training executes on the GPU) when removing the causal masking step from the decoder's first MHA. We use a regular MHA instead of the masked MHA due to the CPU resource bottleneck. Thus, the model remains applicable to experimentation with label propagation but should not be considered a proper transformer.

Model Training The training of the IMDb model lasts for no more than 10 epochs. The training stops early if the model converges to a set of weights such that the training set accuracy is 95% or more. This low limit of epochs and early stop rule is to avoid overfitting to the training data. The training process employs Adaptive Momentum Estimation (ADAM) as the optimization technique [36, Kingma et al., 2017], and the loss function is Cross Entropy from equation 2.6, as discussed in chapter 2.2.2. Figure 2.4 from chapter 2.2.2 plots the accuracy and loss of the training- and validation set during training of the IMDb model. This figure shows the model spent six epochs before converging to the training accuracy threshold of 95%. There are indications of overfitting where the validation accuracy and -loss diverge from the training set. However, this might be an artifact of early training, where more epochs could show an accuracy increase and loss decrease.

Figure 4.3 demonstrates the performance of the model after training. The performance is in the form of confusion matrices for the training-, validation-, test-, and full IMDb dataset. The columns state the number of negative (*Neg*) and positive (*Pos*) predictions, while the rows relate to the ground truth. The diagonal indicates the number of matching predictions and ground truths. The bottom row of the matrices are the total predictions, and the right-most column contains the total number of data points from each class. The bottom right-most element in each matrix is the total number of data points for that dataset. The classification accuracies of each dataset split is stated in the matrix title. The

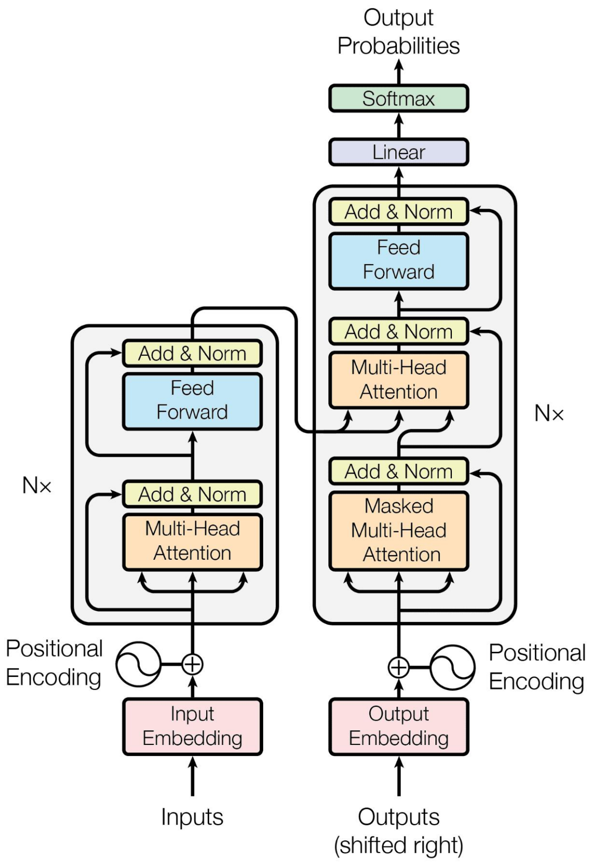
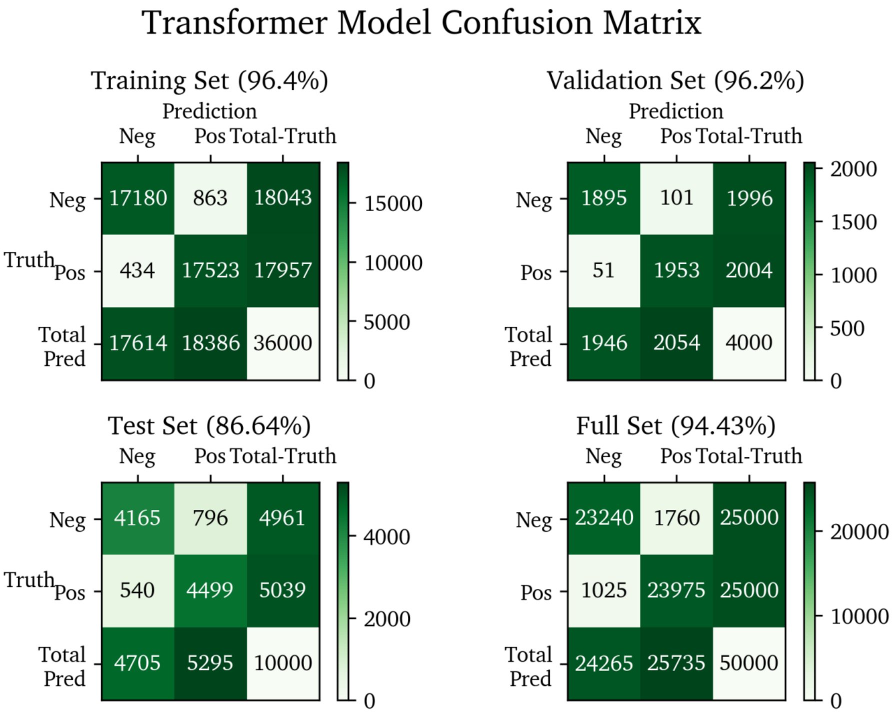


Figure 4.2: The Transformer encoder-decoder architecture [78, Vaswani et al., 2017]. The figure reads best in down-up order.

model compares in the lower range of other IMDb sentiment review classifiers but manages to beat some models with the test accuracy of 86.64% [16].

4.3 Image Experimentation

The image test cases in this section are a subset of the executed experiments. See appendix B for a full list of the results. We demonstrate the more interesting cases and highlight variations in the results. The cases compare the restrictiveness of the propagated labels when applicable. There are some explainability methods that fail and the actions of the LPM in these cases are also described.



Training Set (96.4%)

	Neg	Pos	Total-Truth
Neg	17180	863	18043
Pos	434	17523	17957
Total Pred	17614	18386	36000

Validation Set (96.2%)

	Neg	Pos	Total-Truth
Neg	1895	101	1996
Pos	51	1953	2004
Total Pred	1946	2054	4000

Test Set (86.64%)

	Neg	Pos	Total-Truth
Neg	4165	796	4961
Pos	540	4499	5039
Total Pred	4705	5295	10000

Full Set (94.43%)

	Neg	Pos	Total-Truth
Neg	23240	1760	25000
Pos	1025	23975	25000
Total Pred	24265	25735	50000

Figure 4.3: The model classification accuracy of the various splits of the IMDb reviews in the form of confusion matrices.

Table 4.2 shows an overview of the experiment ImageNet models and the outcome of the experiment with the explainability methods. The overview shows that the *InceptionV3* and *ResNet50V2* models are the only two that succeed in finding explanations for every XAI methods. There are more cases that fail to create explanations, or encounter an error, than those who succeed with each method. We compare these failing cases with the *Xception* and *EfficientNetV2S*. Xception is an interesting case to study due to the similar architecture of Inception, which succeeds in creating explanations. EfficientNetV2S is of a different architectural family than Xception, so this serves as a case for dissimilar architecture that also fail at some experiments.

Image Test Case Structure The test case sections briefly re-state the model specifics from table 4.1. Each test result consists of two figures visualizing the explanations created by the XAI methods: Anchor, IG and LIME. These explainability methods are used to explain the model prediction of the image in figure 4.4 with the corresponding lattice of labels from figure 2.7b in chapter 2.5. The LPM requires a way to assess the faithfulness of the explanations because they are central to the label propagation. Therefore, the first figure

Model	Anchor	IG	LIME
EfficientNetV2S	✗	OOM	✓
EfficientNetV2M	✗	OOM	✓
EfficientNetV2L	✗	OOM	✓
InceptionResNetV2	✗	OOM	✓
InceptionV3	✓	✓	✓
Xception	✗	OOM	✓
ResNet50	✓	OOM	✓
ResNet50V2	✓	✓	✓
ResNetRS50	✗	OOM	✓
ResNet101	✓	OOM	✓
ResNet101V2	✗	OOM	✓
ResNetRS101	✓	OOM	✓
ResNet152	✓	OOM	✓
ResNet152V2	✗	OOM	✓
ResNetRS152	✓	OOM	✓
VGG16	✓	✗	✓

Table 4.2: An overview of each ImageNet model result with the XAI methods for our hardware setup. ✓and ✗ indicates the model succeeded or failed in finding an explanation. The models that failed due to an Out Of Memory (OOM) error are marked with OOM.

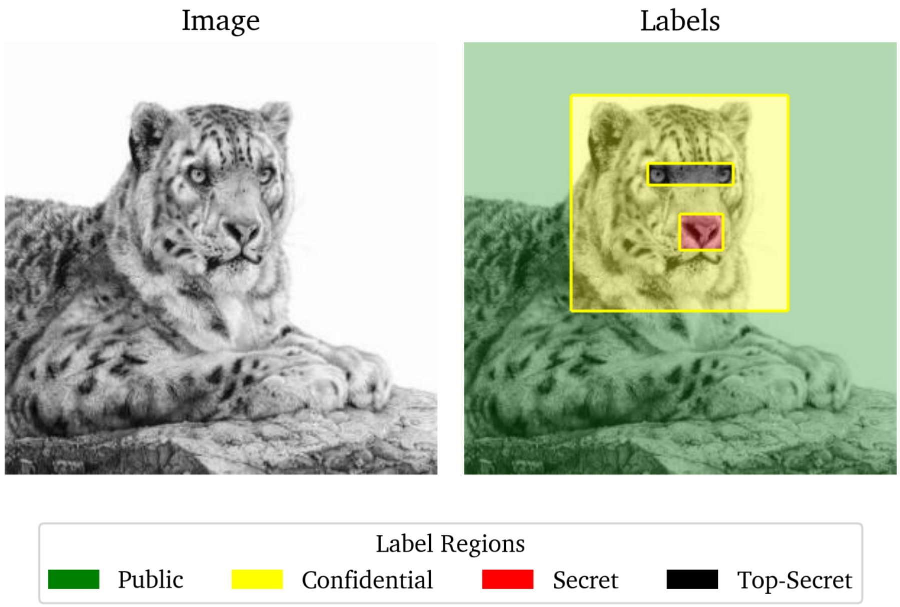


Figure 4.4: The ImageNet sample for experimentation with explainability methods and various models.

contains the results from classifying the pertinent features in the explanations, and the second from classifying the non-pertinent features. We refer to the classification of the important- and non-important features in the explanation as a *classification of the explanation* and *classification of the inverse explanation*, respectively. The classification of the explanation classifies only the important pixels of the image, with zero pixels otherwise. The classification of the inverse explanation zeroes every pixel important according to the explanation and classifies the remaining image. The titles in the figures state the XAI methods, the classification, and the confidence of predicting the (non-)pertinent features of the explanations. Additionally, each test result contains a table with an overview of the propagated labels for each XAI method regarding the visualized explanations. The cases evaluate each XAI explanation and comment whether the method succeeds or fails to create a faithful explanation for the prediction. We do not provide an elaboration on the methods that encounter errors beyond stating when the errors occur. These errors mostly relates to a common OOM error in the GPU

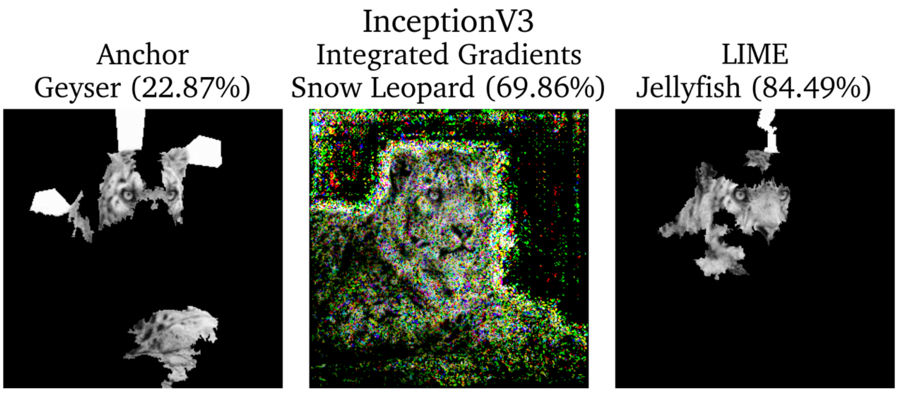
4.3.1 Inception-V3

Inception-V3 [69, Szegedy et al., 2015] consists of 23.9 million weights with a size of 92 MB. The model performs no resizing of the input image because the input shape of the architecture coincides with our ImageNet resolution of $299 \times 299 \times 3$. The model *correctly* classifies the explainee image in figure 4.4 as a *Snow Leopard* with 84.02% confidence. The top-1 and top-5 accuracies are 77.90% and 93.70%, respectively. Figure 4.5a visualizes the explanations from the explainability methods and each explanation classification. Figure 4.5b demonstrates the classification of the inverse explanations.

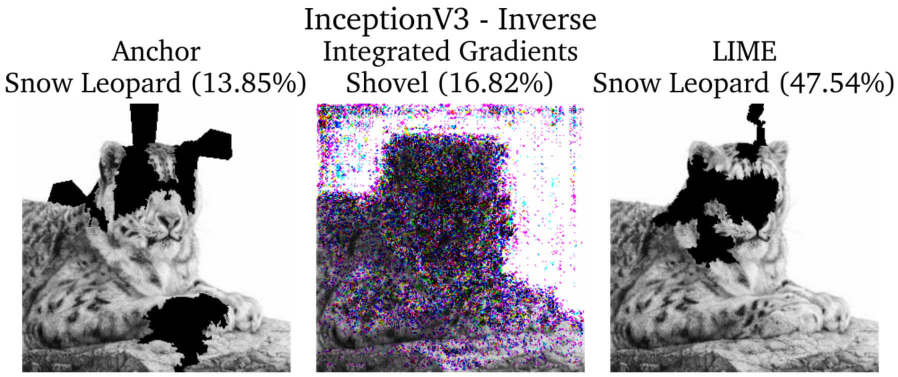
Anchor (left figure) succeeds at finding a set of feature conditions with enough coverage, as discussed in chapter 2.4.2. The model classifies the anchor as a *Geyser* with 22.87% confidence. The model classifies the inverse explanation as a *Snow Leopard* with 13.85% confidence. The classification of the explanation changes from correct *Snow Leopard* with 84.02% confidence to wrong *Geyser* with 22.87% confidence. From the inverse classification, removing the important features from the image causes a drop from 84.02% to 13.85% confidence, which is a good indication of the faithfulness of the explanation.

IG (middle figure) successfully explains the prediction of the image. The model classifies the feature attributions in the explanation as a *Snow Leopard* with 69.86% confidence. The model classifies the inverse explanation as a *Shovel* with 16.82% confidence. The classification of the explanation gets the same outcome as the full image, with some drop in confidence from 84.02% to 69.86%. The classification of the inverse explanation demonstrates the importance of the attributed features. From the full image classification to the inverse classification, the model confidence drops from 84.02% to 16.82%. Hence, this explanation faithfully captures the features necessary for the model prediction of the full image.

LIME (right figure) successfully creates an explanation for the explainee image. The model classifies the explanation as a *Jellyfish* with 84.49% confidence. The model classifies the inverse explanation as a *Snow Leopard* with 47.54% confidence. The classification of the explanation changes from *Snow Leopard* to *Jellyfish* with confidences 84.02% and 84.49%, respectively. The classification of the inverse explanation maintains the full image classification of *Snow Leopard* with drop from 84.02% and 47.54% confidence. This is similar to the classifications of Anchor. The explanation itself is not enough for the model to make the same prediction, but the absence of the pixels lower the confidence while maintaining the class.



(a) The explanations from Anchor, IG, and LIME for the Inception-V3 model prediction of the explainee image in figure 4.4.



(b) The classification of the inverse explanations from Anchor, IG, and LIME for the Inception-V3 model.

Inception-V3					
Explainability Method	**Public**	**Confidential**	**Secret**	**Top-Secret**	**Most Restrictive Label**
Anchor	✓	✓	✗	✓	Top-Secret
Integrated Gradients	✓	✓	✓	✓	Top-Secret
LIME	✓	✓	✓	✓	Top-Secret

Table 4.3: The propagated labels per explainability method for the model. ✓indicates the label propagated to the model output, ✗ indicates the label did not propagate, and N/A indicates an error with XAI method. The right-most column contains the most restrictive propagated label.

Label Propagation Table 4.3 lists the resulting label propagation for the model prediction with each of the explainability methods. The LPM using Anchor propagates every label, except the *Secret* label, and thus, the most restrictive label is *Top-Secret*. The LPM using IG propagates every label because the explanation contains at least one pixel from each labeled bounding box from figure 4.4. The LPM using LIME propagates every label to the output since the important pixels in the explanation are in each labeled bounding box.

This test case shows an all-over strict label propagation in terms of label restrictiveness, as discussed in chapter 3.2. The most restrictive label of each propagated set of labels is the highest in the lattice, i.e., *Top-Secret*. Without considering the lattice and simply which labels the LPMs propagated, Anchor is the only method showing some form of relaxation.

4.3.2 ResNet-50-V2

ResNet-50-V2 [28, He et al., 2016] consists of 25.6 million weights with a size of 98 MB. The model resizes the input image from 299×299×3 into 224×224×3 resolution. The model *correctly* classifies the explainee image in figure 4.4 as a *Snow Leopard* with 99.98% confidence. The top-1 and top-5 accuracies are 76.00% and 93.00%, respectively. Figure 4.6a visualizes the explanations from the explainability methods and each explanation classification. Figure 4.6b demonstrates the classification of the inverse explanations.

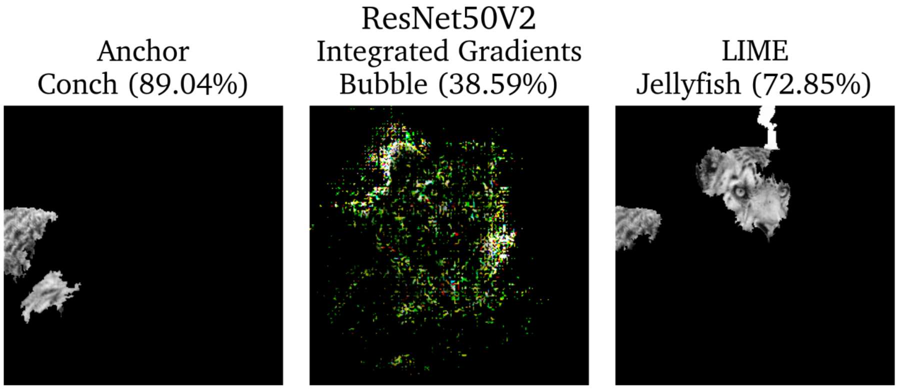
Anchor (left figure) succeeds at finding a set of feature conditions with enough coverage, as discussed in chapter 2.4.2. The model classifies the anchor as a *Conch* with 89.04% confidence. The model classifies the inverse explanation as a *Snow Leopard* with 99.99% confidence. Like Inception-V3, the classification of the explanation changes from correct to wrong. From visual inspection, the explanation indicates the model prediction focuses at the fur-patterns on the back. However, the inverse classification increases the model confidence of *Snow Leopard* from 99.98% to 99.99%. Anchor does not seem to have found a faithful explanation for the model prediction of the explainee image.

IG (middle figure) succeeds at explaining the prediction of the image. The model classifies the feature attributions in the explanation as a *Bubble* with 38.59% confidence. The model classifies the inverse explanation as a *Snow Leopard* with 96.65% confidence. The classification of the attributed features changes from *Snow Leopard* to *Bubble* with confidences 84.02% and 38.59%, respectively. The classification of the inverse explanation remains the same as the full image with similar confidence. The feature attributions in this explanation is more sparse than IG with InceptionV3 in figure 4.5a, and lacks the same faithfulness.

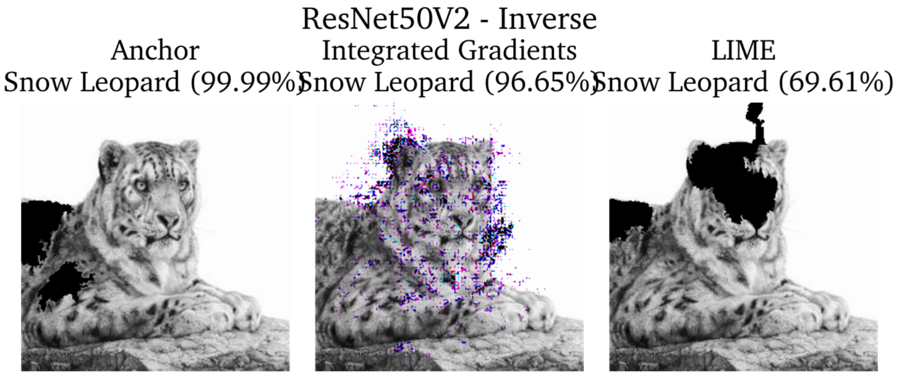
LIME (right figure) successfully creates an explanation for the explainee image. The model classifies the explanation as a *Jellyfish* with 72.85% confidence. The model classifies the inverse explanation as a *Snow Leopard* with 69.61% confidence. The classification of the explanation changes from *Snow Leopard* to *Jellyfish* with confidences 84.02% and 72.85%, respectively. The classification of the inverse explanation maintains the full image classification of *Snow Leopard* with drop from 84.02% and 69.61% confidence. The reactions of the Inception-V3 and ResNet-50-V2 models regarding the explanations from LIME appear similar, even though the explanations differ in which features they emphasize.

Label Propagation Table 4.4 lists the resulting label propagation for the model prediction with each of the explainability methods. The LPM using Anchor propagates only the *Public* label. As with the Inception-V3, the LPM using IG propagates every label model, even though the explanation is more sparse regarding the features in the explanation. The LPM using LIME propagates every label to the output, as seen in the previous experiment.

The label restrictiveness for the Inception-V3 and ResNet-50-V2 models are identical regarding the XAI methods IG and LIME. The restrictiveness of the propagated labels for this model using Anchor is the polar-opposite of Inception-V3, i.e., *Public* and *Top-Secret*. Therefore, the LPM using Anchor again demonstrates the most relaxed label propagation compared between models and methods.



(a) The explanations from Anchor, IG, and LIME for the ResNet-50-V2 model prediction of the explainee image in figure 4.4.



(b) The classification of the inverse explanations from Anchor, IG, and LIME for the ResNet-50-V2 model.

ResNet-50-V2					
Explainability Method	**Public**	**Confidential**	**Secret**	**Top-Secret**	**Most Restrictive Label**
Anchor	✓	✗	✗	✗	Public
Integrated Gradients	✓	✓	✓	✓	Top-Secret
LIME	✓	✓	✓	✓	Top-Secret

Table 4.4: The propagated labels per explainability method for the model. ✓indicates the label propagated to the model output, ✗ indicates the label did not propagate, and N/A indicates an error with XAI method. The right-most column contains the most restrictive propagated label.

4.3.3 Xception

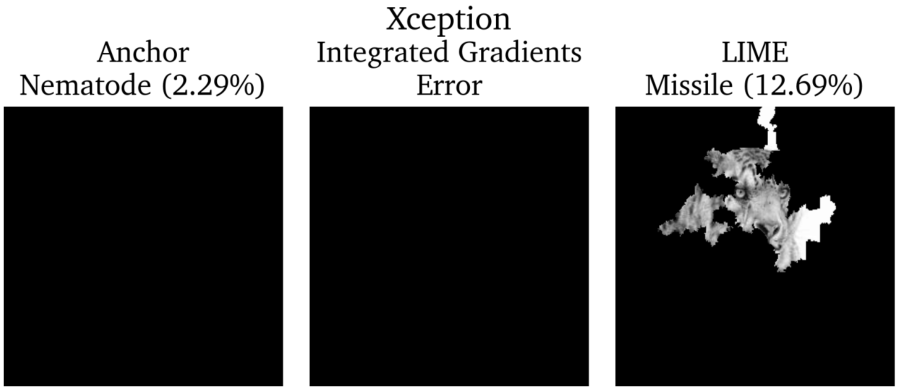
Xception [15, Chollet, 2017] consists of 22.9 million weights with a size of 88 MB. The model performs no resizing of the input image because the input shape of the architecture coincides with our ImageNet resolution of 299 × 299 × 3. The model *correctly* classifies the explainee image in figure 4.4 as a *Snow Leopard* with 75.44% confidence. The top-1 and top-5 accuracies are 79.00% and 94.50%, respectively. Figure 4.7a visualizes the explanations from the explainability methods and each explanation classification. Figure 4.7b demonstrates the classification of the inverse explanations.

Anchor (left figure) fails at finding a set of feature conditions with enough coverage, as discussed in chapter 2.4.2. The resulting explanation contains only black pixels. We distinguish between *failure* to explain and *run-time errors* during explanation, though both leave the LPM without an explanation to leverage for label propagation. The model classifies the black explanation pixels as *Nematode* with 2.29% confidence. Thus, the inverse explanation is equivalent to the full image, which the model classifies as a *Snow Leopard* with 75.44% confidence.

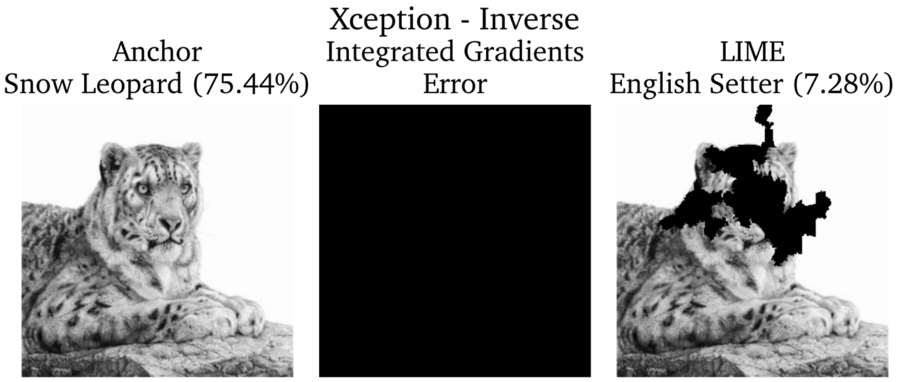
IG (middle figure) encounters an OOM error during execution, and therefore, is blacked out without a classification of the explanation.

LIME (right figure) successfully creates an explanation for the explainee image. The model classifies the explanation as a *Missile* with 12.69% confidence. The model classifies the inverse explanation as an *English Setter* with 7.28% confidence. This coincides with the previous observations with the explanations from LIME, except for the inverse classification. In this case, the classification of the inverse LIME explanation overturns the prediction class, as seen with Anchor in the previous experiments. The inverse classification changes from *Snow Leopard* to *English Setter* with 75.44% and 7.28% confidence, respectively. The overall reasoning is still the same, i.e., the explanation alone is not enough, but the features present in the explanation impact the output.

Label Propagation Table 4.6 lists the resulting label propagation for the model prediction with each of the explainability methods. The LPM using Anchor propagates the *Top-Secret* label because of the lack of explanation regarding the model prediction. The motivation behind this choice is that without a successful explanation the LPM must decide without information regarding the decision of the model. This is a conservative choice that is to ensure no restriction policy is violated by a more relaxed propagation policy, and by no means the correct choice for every application. The propagated label for IG is the same as Anchor for the same reasons. The LPM using LIME



(a) The explanations from Anchor, IG, and LIME for the Xception model prediction of the explainee image in figure 4.4.



(b) The classification of the inverse explanations from Anchor, IG, and LIME for the Xception model.

Xception					
Explainability Method	**Public**	**Confidential**	**Secret**	**Top-Secret**	**Most Restrictive Label**
Anchor	✗	✗	✗	✗	Top-Secret
Integrated Gradients	N/A	N/A	N/A	N/A	Top-Secret
LIME	✓	✓	✓	✓	Top-Secret

Table 4.5: The propagated labels per explainability method for the model. ✓indicates the label propagated to the model output, ✗ indicates the label did not propagate, and N/A indicates an error with XAI method. The right-most column contains the most restrictive propagated label.

propagates the every label since there are important features in the explanation from each of the labeled bounding boxes in figure 4.4. We skip evaluation of the label restrictiveness, as this test case does not contain enough successful label propagations for comparison.

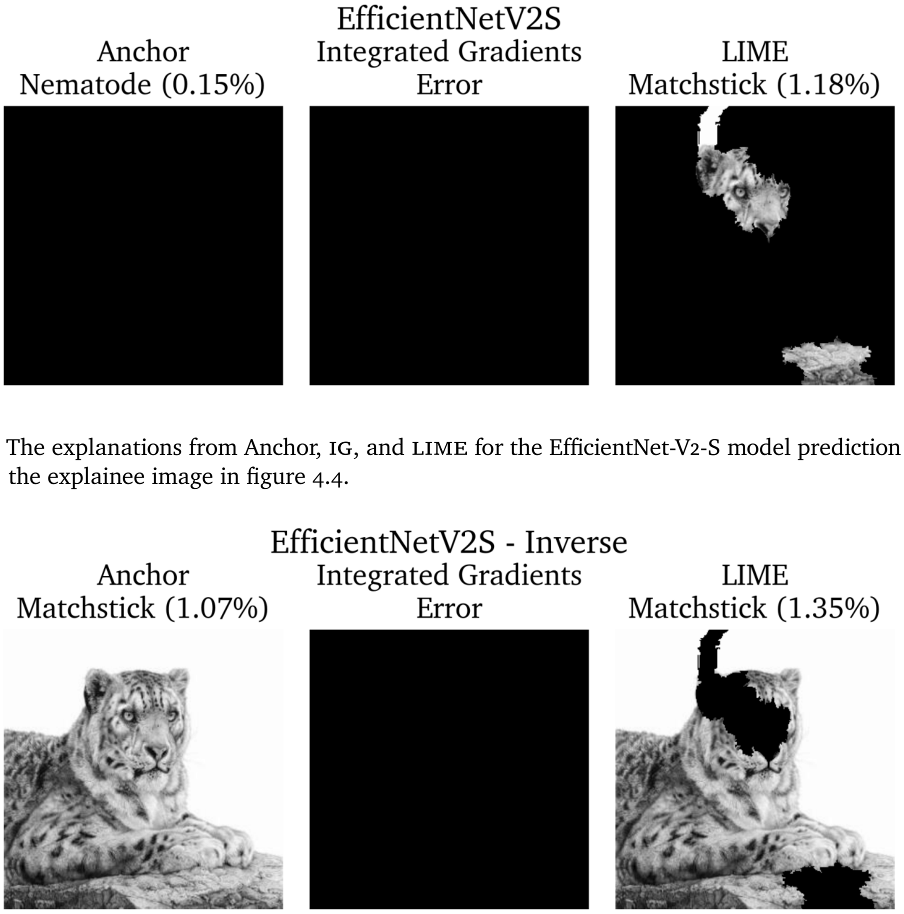
4.3.4 EfficientNet-V2-S

EfficientNet-V2-S [70, Tan et al., 2021] consists of 21.6 million weights with a size of 88 MB. The model resizes the input image from 299 × 299 × 3 into 384 × 384 × 3 resolution. The model *misclassifies* the explainee image in figure 4.4 as a *Matchstick* with 1.07% confidence. The top-1 and top-5 accuracies are 83.90% and 96.70%, respectively. Figure 4.8a visualizes the explanations from the explainability methods and each explanation classification. Figure 4.8b demonstrates the classification of the inverse explanations.

Anchor (left figure) fails to create an explanation, as with Xception. The model classifies the black explanation pixels as *Nematode* with 0.15% confidence. The classification of the inverse explanation is equivalent to the classification of the full image.

IG (middle figure) encounters an OOM error during execution.

LIME (right figure) successfully creates an explanation for the explainee image. The model classifies the explanation as a *Matchstick* with 1.18% confidence. This *Matchstick* classification is consistent with the classification of the full image with a small increase in confidence from 1.07% to 1.18%. This increase is an indication that the explanation might be faithful to the decision-making of the model. However, the classification of the inverse explanation is also *Matchstick* with a higher confidence of 1.35%, so the explanation seems to contradict itself. The pixels in the explanation is a reasonable emphasis for what could be a *Snow Leopard*. The classification is still wrong, so we do not



(a) The explanations from Anchor, IG, and LIME for the EfficientNet-V2-S model prediction of the explainee image in figure 4.4.

(b) The classification of the inverse explanations from Anchor, IG, and LIME for the EfficientNet-V2-S model.

comment further on the explanation from visual evaluation.

Label Propagation Table 4.6 lists the resulting label propagation for the model prediction with each of the explainability methods. The LPM using Anchor propagates the *Top-Secret* label due to the lack of an explanation, and similarly for IG. The LPM using LIME propagates the every label because of the important pixels in the explanation. For this test case, the label restrictiveness evaluation is omitted.

EfficientNet-V2-S					
Explainability Method	**Public**	**Confidential**	**Secret**	**Top-Secret**	**Most Restrictive Label**
Anchor	✗	✗	✗	✗	Top-Secret
Integrated Gradients	N/A	N/A	N/A	N/A	Top-Secret
LIME	✓	✓	✓	✓	Top-Secret

Table 4.6: The propagated labels per explainability method for the EfficientNet-V2-S model. ✓indicates the label propagated to the model output, ✗ indicates the label did not propagate, and N/A indicates an error with XAI method. The right-most column contains the most restrictive propagated label.

4.3.5 Image Experiment Evaluation

The LPMs employing Anchor are the least restrictive for the image experiments. The Anchor explanations contain features that highly impact the model prediction when removed, though Anchor is not the only method with this property. LIME manages to consistently and reliably create explanations for the image. Removing the features in LIME explanations tend to impact the classification, though not in the same relative consistency as Anchor. See appendix B for the overall impact from explanations of LIME. IG offers far more precise explanations than any of the other two. This precision demonstrates how model-specific XAI methods can provide more informative explanations than black-box explanations. However, only 3 out of 16 models could be explained without an OOM error on our experiment computer hardware, and the resulting label propagation is strict. The two cases IG manages to create an explanation for are shown above. The third model is VGG16 and IG produces a black pixel explanation, as seen previously with Anchor.

Anchor tends to be slower to create explanations than LIME and IG spending around 1 minute on average to explain the image, regardless of which model. This relative slowness is likely because of the search complexity for the algorithm. LIME trains a linear model based on local image samples. IG integrates over gradients for images between the baseline image and the explainee image. Both LIME and IG spend between 15 to 30 seconds creating their explanations.

Anchor holds a significant track-record of not finding a set of feature constraints to serve as an explanation, which is visible in table 4.2. This relates more to the explainee image and the model size than it does with the experiment hardware. One possible reason might be that the high-dimensional prediction surface around the image is very rough such that the sampled set of perturbations does not provide enough coverage for the Anchor's confidence threshold parameter. This complication is due to a combination of the data point sampled for explanation and the number of parameters in the model. The more parameters a model consists of the more non-linear a decision-boundary potentially becomes. An interesting point of future exploration is whether relaxing the confidence threshold of Anchor produces explanations for the models who failed in this experiment.

The high-level take-aways are that the label propagation differs depending on the parameters considered by the LPM. The XAI methods emphasize important features in various fashion. The complexity of the model influence the XAI's ability to create meaningful explanations. Exploring the faithfulness of the explanations is crucial for applications of label propagation because of how heavily the LPM depend on the explanations.

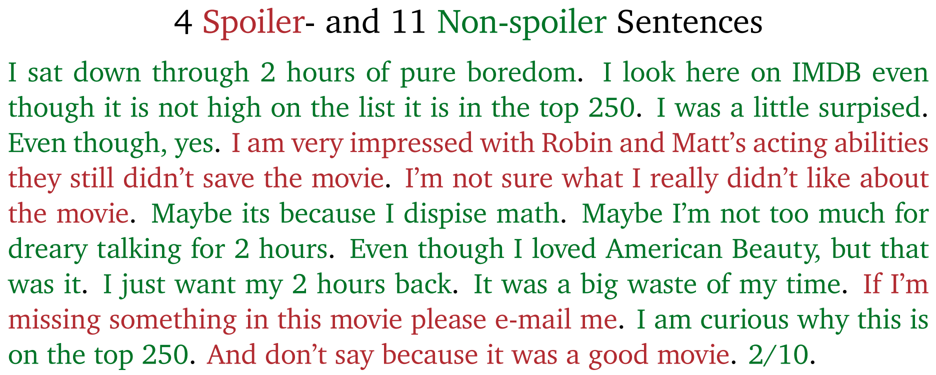
4.4 Text Experimentation

This section concerns four IMDb reviews. The reviews are labeled per-word depending on which sentence they occur in. The sentences are labeled with spoiler- and non-spoiler labels from a lattice similar to the one in figure 2.7b. These experiments use LIME and Anchor (not IG) to explain the model prediction of the reviews as positive versus negative. The classifier model is the transformer model described in section 4.2.2. We present the results for each of the reviews with both LIME and Anchor guiding the LPM to propagate the labels.

Text Test Case Structure The reviews are correctly or wrongly classified with high or low confidence. The cases present the reviews as figures of text where the labels are shown by the colors red (for spoiler) and green (for non-spoiler). Each case describes the model prediction of the review and then details the explanations from the explainability methods.

The explanations from LIME consists of positional words and contribution weights for the classification. LIME includes every word in the explanation unless somehow filtered, as discussed in chapter 3.3.1. These experiments select only the top-n words with the highest contribution weights. We chose $n = \frac{1}{4}w_r$ where w_r is the number of words in the review. Hence, each explanation from LIME contains no more than a quarter of the words present in the review. The explanations from Anchor consists of words anywhere in the input that sufficiently ensures a model prediction similar to that of the full review. Anchor does not exhibit weights for the words.

We experiment with replacing the words in the reviews with *UNK* words



4 Spoiler- and 11 Non-spoiler Sentences

I sat down through 2 hours of pure boredom. I look here on IMDB even though it is not high on the list it is in the top 250. I was a little surpised. Even though, yes. I am very impressed with Robin and Matt's acting abilities they still didn't save the movie. I'm not sure what I really didn't like about the movie. Maybe its because I dispise math. Maybe I'm not too much for dreary talking for 2 hours. Even though I loved American Beauty, but that was it. I just want my 2 hours back. It was a big waste of my time. If I'm missing something in this movie please e-mail me. I am curious why this is on the top 250. And don't say because it was a good movie. 2/10.

Figure 4.9: A *negative* review classified as *negative* with 100.00% Confidence.

and classifying the *new reviews* to observe the change in model prediction. We replace with *UNK* instead of removing because the XAI methods use this replacement procedure during sampling of perturbations to create the explanations. Finally, we discuss the propagated labels for the reviews and the explainability methods.

4.4.1 Correct Classification & High Confidence

Figure 4.9 shows the review with correct classification and highest confidence. The sample is a *negative* review that is classified as *negative* with 100% confidence. The review consists of 138 words.

Anchor For this review, the Anchor explanation consists of the word "**waste**". The label of the word comes from the sentence "It was a big waste of my time.", which is a *non-spoiler* sentence. The following paragraph contains the review with the explanation word annotated in gray:

i sat down through 2 hours of pure boredom i look here on imdb even though it is not high on the list it is in the top 250 i was a little surpised even though yes i am very impressed with robin and matts acting abilities they still didnt save the movie im not sure what i really didnt like about the movie maybe its because i dispise math maybe im not too much for dreary talking for 2 hours even though i loved american beauty but that was it i just want my 2 hours back it was a big waste of my time if im missing something in this movie please e mail me i am curious why this is on the top 250 and dont say because it was a good movie 2 10

The following states the model classifications when replacing "**waste**" with *UNK* and only classifying the explanation:

Prediction of the **review without the Anchor words**:
Negative review predicted as *negative* with confidence 99.95%.
Prediction of **only the Anchor words**:
Negative review predicted as *negative* with confidence 99.44%.

The review is *negative* and classified as *negative* with 100% confidence. The replacement of the important word causes a 0.05% drop in confidence without affecting the model classification. The classification of only "**waste**" intuitively indicates the model associates the word "**waste**" negatively. The low impact of the replacement indicates the prediction is based on other words than the sole Anchor word. Hence, the most restrictive propagated output label for this review is *non-spoiler*.

LIME Table 4.7 lists the explanation of LIME with the top-34 words from the explanation. The explanation contains 34 words because the review contains 138 words and 34 is a quarter of the review. The words are sorted by highest absolute contribution weights. The third column contains the positional words in the review that are important for the model prediction. The fourth column states the labels of the words as they correspond to the review in figure 4.9. The following paragraph contains the review with the explanation words annotated in gray:

i sat down through 2 hours of pure boredom i look here on imdb even though it is not high on the list it is in the top 250 i was a little surpised even though yes i am very impressed with robin and matts acting abilities they still didnt save the movie im not sure what i really didnt like about the movie maybe its because i dispise math maybe im not too much for dreary talking for 2 hours even though i loved american beauty but that was it i just want my 2 hours back it was a big waste of my time if im missing something in this movie please e mail me i am curious why this is on the top 250 and dont say because it was a good movie 2 10

The following states the model classifications when replacing the words with *UNK*s and classifying the explanation:

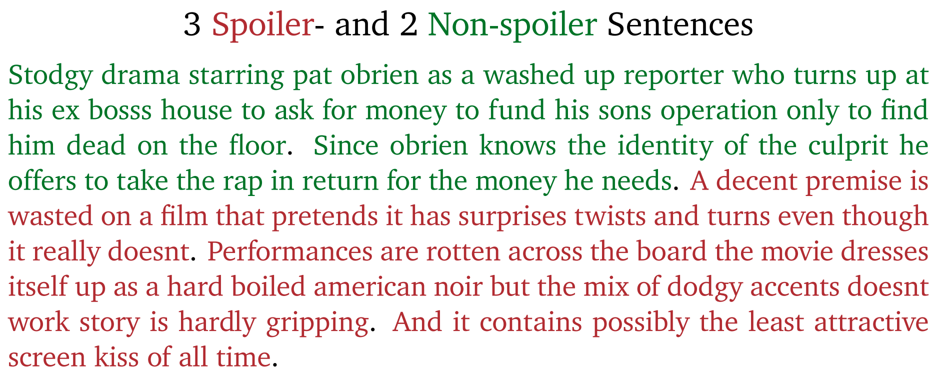
Prediction of the review **without the LIME words**:
Negative review predicted as *positive* with confidence 62.20%.
Prediction of **only the LIME words**:
Negative review predicted as *negative* with confidence 99.91%.

The review is *negative* and classified as *negative* with 100% confidence. The replacement of each important words causes a shift in the model classification from *negative* to *positive* with 100.00% and 62.20% confidence, respectively.

The impact of the words is likely due to the number of words in the explanation, as opposed to Anchor above. The classification of only the words maintains the *negative* classification with a close confidence to the original output. The input text for the explanation words is orderer with the most to least influential words as they appear in table 4.7. This input includes the words once despite the multiple positional occurrences listed in column three. The shift in classification when replacing the words, and the high confidence of just the explanation words, indicate the explanation is faithful to the decision of the model. The positional occurrences appears to include every occurence in the text. Consider rows 18, 27 and 31 with words "*it*", "*2*" and "*a*", these occur multiple times in the review and each one is a part of the annotated review above. The explanation contains 23 words that are labeled *non-spoiler* and 11 *spoiler* words: *save, maybe, 250, though, mail, much, dont, movie, still, down*, and *curious*. Thus, the most restrictive propagated output label for this review is *spoiler*.

Correct Classification & High Confidence: LIME				
Highest Contribution	**Words < 34**	**Weight**	**Positions**	**Label**
1	boredom	0.083958	[8]	Non-Spoiler
2	waste	0.068526	[102]	Non-Spoiler
3	save	0.038567	[50]	Spoiler
4	sat	0.037744	[1]	Non-Spoiler
5	loved	−0.033068	[84]	Non-Spoiler
6	impressed	−0.027901	[40]	Non-Spoiler
7	even	0.025654	[14, 34, 81]	Non-Spoiler
8	dreary	0.023308	[76]	Non-Spoiler
9	hours	0.017068	[5, 80, 96]	Non-Spoiler
10	maybe	0.016447	[64, 70]	Spoiler
11	through	0.015972	[3]	Non-Spoiler
12	beauty	−0.015897	[86]	Non-Spoiler
13	pure	0.01581	[7]	Non-Spoiler
14	250	−0.01568	[28, 126]	Spoiler
15	why	0.015337	[120]	Non-Spoiler
16	though	−0.014721	[15, 35, 82]	Spoiler
17	not	0.011861	[18, 54, 72]	Non-Spoiler
18	it	−0.011433	[16, 23, 90, 98, 131]	Non-Spoiler
19	mail	0.011193	[115]	Spoiler
20	big	0.009922	[101]	Non-Spoiler
21	much	0.009154	[74]	Spoiler
22	good	−0.009107	[134]	Non-Spoiler
23	missing	0.009085	[108]	Non-Spoiler
24	my	−0.008547	[94, 104]	Non-Spoiler
25	was	0.008001	[30, 89, 99, 132]	Non-Spoiler
26	dont	0.007473	[128]	Spoiler
27	2	−0.007444	[4, 79, 95, 136]	Non-Spoiler
28	movie	0.007304	[52, 63, 112, 135]	Spoiler
29	sure	−0.007151	[55]	Non-Spoiler
30	still	−0.006612	[48]	Spoiler
31	a	−0.006158	[31, 100, 133]	Non-Spoiler
32	down	−0.00607	[2]	Spoiler
33	and	−0.00587	[43, 127]	Non-Spoiler
34	curious	−0.005409	[119]	Spoiler

Table 4.7: LIME's explanation for the review in figure 4.9 with top-34 words as filter.



3 Spoiler- and 2 Non-spoiler Sentences

Stodgy drama starring pat obrien as a washed up reporter who turns up at his ex bosss house to ask for money to fund his sons operation only to find him dead on the floor. Since obrien knows the identity of the culprit he offers to take the rap in return for the money he needs. A decent premise is wasted on a film that pretends it has surprises twists and turns even though it really doesnt. Performances are rotten across the board the movie dresses itself up as a hard boiled american noir but the mix of dodgy accents doesnt work story is hardly gripping. And it contains possibly the least attractive screen kiss of all time.

Figure 4.10: A *negative* review classified as *negative* with 50.04% confidence.

4.4.2 Correct Classification & Low Confidence

Figure 4.10 shows the review with correct classification and lowest confidence. The sample is a *negative* review that is classified as *negative* with 50.04% confidence. The review consists of 120 words.

Anchor For this review, the Anchor explanation consists of the words in table 4.8. The first column lists the words required for the classification to match with the original review. The labels of the explanation words are listed in the second column. The following paragraph contains the review with the explanation words annotated in gray:

stodgy drama starring pat obrien as a washed up reporter who turns up at his ex bosss house to ask for money to fund his sons operation only to find him dead on the floor since obrien knows the identity of the culprit he offers to take the rap in return for the money he needs a decent premise is wasted on a film that pretends it has surprises twists and turns even though it really doesnt performances are rotten across the board the movie dresses itself up as a hard boiled american noir but the mix of dodgy accents doesnt work and the story is hardly gripping and it contains possibly the least attractive screen kiss of all time

The following states the model classifications when replacing words with *UNK* and only classifying the explanation:

Prediction of the **review without the Anchor words**:
Negative review predicted as *positive* with confidence 98.18%.
Prediction of **only the Anchor words**:
Negative review predicted as *negative* with confidence 98.81%.

Correct Classification & Low Confidence: Anchor	
Word	**Label**
wasted	Spoiler
pretends	Spoiler
obrien	Non-Spoiler
fund	Non-Spoiler
hardly	Spoiler
rotten	Spoiler
obrien	Non-Spoiler
operation	Non-Spoiler
washed	Non-Spoiler
contains	Spoiler
across	Spoiler
reporter	Non-Spoiler

Table 4.8: Anchor's explanation for the review in figure 4.10.

The review is *negative* and classified *negative* with 50.04% confidence. The replacement of the important words overturns the classification from a correct to wrong classification with high confidence of 98.18%. Like the previous example, the classification of only the Anchor words indicates the model associates the words negatively. The impact of the replacement indicates the explanation is faithful to the model prediction because the correct classification is maintained and the confidence increases from 50.04% to 98.81%. The explanation contains 6 words that are labeled non-spoiler and 6 spoiler words: The most restrictive propagated output label for this review is *spoiler* from the labels in table 4.8.

LIME The explanation tables of LIME are found in the appendix for the following sections. While the explanations are verbose and offer rich information, they quickly become repetitive and we omit placing them here when referencing them. Table C.2 lists the explanation of LIME with the top-30 words from the explanation. The following paragraph contains the review with the explanation words annotated in gray:

stodgy drama starring pat obrien as a washed up reporter who turns up at his ex bosss house to ask for money to fund his sons operation only to find him dead on the floor since obrien knows the identity of the culprit he offers to take the rap in return for the money he needs a decent premise is wasted on a film that pretends it has surprises twists and turns even though it really doesnt performances are rotten across the board the movie dresses itself up as a hard boiled american noir but the mix of dodgy

3 Spoiler- and 3 Non-spoiler Sentences

A sequel to angels with dirty faces in name only the angels wash their faces suffers somewhat from the usual shenanigans of the dead end kids. As a matter of fact with the presence of the dead end kids and ann sheridan this should have been treated as an actual sequel to angels with dirty faces at least for continuitys sake. Speaking of ann sheridan she is the one true shining light of this movie. To paraphrase a cliché ann sheridan could read from a phone book for two hours and i would buy the dvd another virtue of this movie is the chemistry between ann sheridan and ronald reagan. Unfortunately this aspect of the film is kept too far in the background. For a better example of the sheridan reagan duo i would recommend juke girl or kings row.

Figure 4.11: A *positive* review classified as *negative* with 99.94% confidence.

accents doesnt work and the story is hardly gripping and it contains possibly the least attractive screen kiss of all time

The following states the model classifications when replacing the words with *UNK*s and classifying the explanation:

Prediction of the review **without the LIME words**:
Negative review predicted as *positive* with confidence 84.30%.
Prediction of **only the LIME words**:
Negative review predicted as *negative* with confidence 63.22%.

The review is *negative* and classified *negative* with 50.04% confidence. The replacement shifts the model classification from *negative* to *positive* with 50.04% and 84.30% confidence, respectively. The classification of only the words maintains the *negative* classification with an increased confidence. These results indicate the explanation is faithful, as with the previous example of LIME. The explanation contains 12 words that are labeled non-spoiler and 18 spoiler words. The most restrictive propagated output label for this review is *spoiler*.

4.4.3 Wrong Classification & High Confidence

Figure 4.11 shows the review with wrong classification and highest confidence. The sample is a *positive* review that is classified as *negative* with 99.94% confidence. The review consists of 140 words.

Anchor For this review, the Anchor explanation consists of the words "**unfortunately**" and "**suffers**". The labels of the words are *spoiler* and *non-spoiler*, respectively. The following paragraph contains the review with the explanation words annotated in gray:

a sequel to angels with dirty faces in name only the angels wash their faces suffers somewhat from the usual shenanigans of the dead end kids as a matter of fact with the presence of the dead end kids and ann sheridan this should have been treated as an actual sequel to angels with dirty faces at least for continuitys sake speaking of ann sheridan she is the one true shining light of this movie to paraphrase a cliché ann sheridan could read from a phone book for two hours and i would buy the dvd another virtue of this movie is the chemistry between ann sheridan and ronald reagan unfortunately this aspect of the film is kept too far in the background for a better example of the sheridan reagan duo i would recommend juke girl or kings row

The following states the model classifications when replacing words with *UNK* and only classifying the explanation:

Prediction of the **review without the Anchor words**:
Positive review predicted as *negative* with confidence 99.82%.
Prediction of **only the Anchor words**:
Positive review predicted as *negative* with confidence 99.45%.

The review is *positive* and classified *negative* with 99.94% confidence. As previous Anchor explanations show, the replacement classification does not affect the original outcome or confidence much due to the few words in the explanation. The classification of only the Anchor words still inspire faithfulness in the explanation because it shines light on the model's heavy negative association with certain words. Between "**unfortunately**" and "**suffers**", the former's *spoiler* label is the most restrictive propagated output label for this review.

LIME Table C.3 lists the explanation of LIME with the top-35 words from the explanation. The following paragraph contains the review with the explanation words annotated in gray:

a sequel to angels with dirty faces in name only the angels wash their faces suffers somewhat from the usual shenanigans of the dead end kids as a matter of fact with the presence of the dead end kids and ann sheridan this should have been treated as an actual sequel to angels with dirty faces at least for continuitys sake speaking of ann sheridan she is the one true shining light of this movie to paraphrase a cliché

1 Spoiler- and 1 Non-spoiler Sentences

What did producer director stanley kramer see in adam kennedys novel and kennedys very puzzling screenplay were there a few pieces left out on purpose and what about gene hackman richard widmark edward albert eli wallach and mickey rooney what did they see in this very muddled story and why did candice bergen who gave a horrible performance accept such a thankless role the domino principle wants to be on the same footing as the parallax view or the manchurian candidate and misses the mark by a very wide margin. A major misfire by stanley kramer.

Figure 4.12: A *negative* review classified as *positive* with 50.07% confidence.

ann sheridan could read from a phone book for two hours and i would buy the dvd another virtue of this movie is the chemistry between ann sheridan and ronald reagan unfortunately this aspect of the film is kept too far in the background for a better example of the sheridan reagan duo i would recommend juke girl or kings row

The following states the model classifications when replacing the words with *UNK*s and classifying the explanation:

Prediction of the review **without the LIME words**:
Positive review predicted as *positive* with confidence 83.56%.
Prediction of **only LIME words**:
Positive review predicted as *negative* with confidence 99.78%.

The review is *positive* and classified *negative* with 99.94% confidence. The replacement classification shifts the outcome from *negative* to *positive* with 83.56% confidence. The classification of only the words remains *negative* classification with high confidence. As with previous examples of LIME, the explanation is faithful regarding which words are influential for the model's original *negative* classification. The explanation contains 19 words that are labeled non-spoiler and 16 spoiler words. The most restrictive propagated output label for this review is *spoiler*.

4.4.4 Wrong Classification & Low Confidence

Figure 4.12 shows the review with wrong classification and lowest confidence. The sample is a *negative* review classified as *positive* with 50.07% confidence. The review consists of 96 words.

Anchor For this review, the Anchor explanation consists of the words in table 4.9 where every word is labeled *spoiler*. The following paragraph contains the review with the explanation words annotated in gray:

what did producer director stanley kramer see in adam kennedys novel and kennedys very puzzling screenplay were there a few pieces left out on purpose and what about gene hackman richard widmark edward albert eli wallach and mickey rooney what did they see in this very muddled story and why did candice bergen who gave a horrible performance accept such a thankless role the domino principle wants to be on the same footing as the parallax view or the manchurian candidate and misses the mark by a very wide margin a major misfire by stanley kramer

The following states the model classifications when replacing words with *UNK* and only classifying the explanation:

Prediction of the **review without the Anchor words**:
Negative review predicted as *negative* with confidence 99.48%.
Prediction of **only the Anchor words**:
Negative review predicted as *positive* with confidence 77.84%.

The review is *negative* and classified *positive* with 50.07% confidence. Replacing the Anchor words with UNK overturns the classification from *positive* to *negative* with high confidence. The explanation seems faithful judging from the drastic change in classification by removing of the features contributing towards the prediction of the full review. This is underscored by the classification of the explanation where the model associates these words on the side of *positive* instead of *negative* classification. Every word in the explanation is labeled *spoiler*, so the propagated output label is *spoiler*. By observing the labeled sentences in figure 4.12, this is not too strange since there are two sentences where one of the sentences consist of nearly the entire review.

Notice the three words in last part of the review: "*a*", "*major*", "*stanley*", and "*kramer*". This sentence is labeled *non-spoiler*, but the LPM considers the lattice from which the labels originate during label propagation. The three words also occur outside of this sentence, which is labeled *spoiler* and is more restrictive than the last sentence label. Hence, in the case of such *re-occurences* for explanation words, the LPM should consider the lattice to decide which label to propagate.

LIME Table C.4 lists the explanation of LIME with the top-24 words from the explanation. The following paragraph contains the review with the explanation words annotated in gray:

Wrong Classification & Low Confidence: Anchor	
Word	**Label**
candidate	Spoiler
kramer	Spoiler
same	Spoiler
pieces	Spoiler
principle	Spoiler
stanley	Spoiler
mark	Spoiler

Table 4.9: Anchor's explanation for the review in figure 4.12.

what did producer director stanley kramer see in adam kennedys novel and kennedys very puzzling screenplay were there a few pieces left out on purpose and what about gene hackman richard widmark edward albert eli wallach and mickey rooney what did they see in this very muddled story and why did candice bergen who gave a horrible performance accept such a thankless role the domino principle wants to be on the same footing as the parallax view or the manchurian candidate and misses the mark by a very wide margin a major misfire by stanley kramer

The following states the model classifications when replacing the words with *UNK*s and classifying the explanation:

Prediction of the review **without the LIME words**:
Negative review predicted as *positive* with confidence 62.54%.
Prediction of **only the LIME words**:
Negative review predicted as *negative* with confidence 93.63%.

The review is *negative* and classified *positive* with 50.07% confidence. Unlike Anchor with this sample, the replacement classification is not enough to change the model classification. This is the first example where LIME does not overturn the classification by removing the words from the review. Moreover, classifying only the explanation upholds the classification of the review. The explanations in previous examples are faithful because the explanation classification matches with the classification of the review. In this case, the explanation seems unfaithful even though some of the explanation words match with the Anchor explanation for this review. As with Anchor, each explanation word belongs to the *spoiler* label, so the propagated output label is *spoiler*.

4.4.5 Text Experiment Evaluation

The explanations appear faithful in most cases. The explanations containing few words exhibit little impact when replacing them, as opposed to the explanations containing more words. Every explanation with insignificant impact to the classification of the review still salvages some credability when classifying the explanation words. The only case where the explanation does not appear faithful is the LIME explanation with the review category *wrong classification and low confidence*. This might suggest that the prediction surface around the decision boundary is highly complex.

The Anchor creates explanations with both few and many words. LIME consistently produces explanations consisting of a quarter of the review's number of words. The explanation words in the review are by no means clustered. That is, the positional spread of the explanation words in the reviews tend to cover many sentences. Therefore, the propagated labels are almost exclusively *spoiler*, which is the highest label in the lattice. The only case where the propagated label happends to be *non-spoiler* is when the explanation contains few words.

The high-level take-aways from these experiments are that the label propagation differs depending on the reviews and the labeling strategy. The labeling strategy in these experiments are different from the image experiments. In the image experiments, the labeling and the explainee data point is constant. These experiments show that different labeling for different samples result in a difference in the propagated labels. From review to review, the XAI methods might find explanations that vary in how many words constitute an explanation. That is, given one model, one XAI method, and two different reviews, the resulting explanations might contain different words and number of words. Consequently, the propagated labels depend on the data point subject to explanation.

5 Discussion

This section discusses some topics for future work.

5.1 Revisit – IMDB Preprocessing

This section elaborates on the step-wise preprocessing of the Internet Movie Database (IMDb) Review dataset, as discussed in chapter 4.2.1. We considered adding the token word "*UNK*" at random or specific places in each review, where *UNK* is a short-hand for *unknown*. Related work uses *UNK* as a unique token to avoid conflicts with texts that contain the actual word *unknown*. The reason for this consideration is based on the interaction between explainability methods and the trained model.

Local Interpretable Model-agnostic Explanations (LIME) and Anchor sample perturbations of the explainee review on a per-word basis, as discussed in chapters 2.4.1 and 2.4.2. A perturbation consists of the review where *UNK* tokens replace some words. The methods then classify the perturbation samples as part of creating the explanation. A model might respond unpredictably to an input never observed during training. Thus, randomly or uniformly adding UNKs to the training set of our model is sensible because the model should learn these words are redundant, as discussed in chapter 2.2.2. In other words, we would like a 50% model confidence for both a positive- and negative classification with just the *UNK* token as input. In this case, the *UNK* token is a *baseline* input, as proposed by the XAI method Integrated Gradients (IG) discussed in chapter 2.4.3.

The IMDb model (as discussed in chapter 4.2.2) classifies a single *UNK* as a positive review with 75% confidence. The confusion matrices in figure 4.3 indicate that the number of false-positive predictions is more numerous than the false-negative predictions. This implies the model is more lenient towards one class than another. This consideration is not applied to the experiments and warrants further exploration regarding a new research question:
How significant is the interaction between the perturbations of an explainability method and a model's experience with these inputs?

5.2 Revisit – Integrated Gradients

In chapter 3.3.3, we discussed the LPM's application of the explanation from IG regarding image data. This thesis does not explain textual data with IG, leaving this to future work. This limitation is due to difficulties formatting our text classification model to work with Alibi's implementation of IG. The following elaborates why and details the conceptual strategy regarding propagating text labels based on IG explanations with models using tokenization- and embedding input layers.

IG is a model-specific explainability method as discussed in chapter 2.3.1, which means the explanations are white-box explanations from definition 3. This thesis employs Tensorflow models [5, Abadi et al., 2016] when invoking explainability methods. Alibi's implementation of IG strictly requires TensorFlow models and numerical inputs. The numerical limitation complicates explaining models for text input more than with images because pixels are numerical.

Text is usually preprocessed from a sequence of symbols to tokens and then into embedding vectors, as discussed in chapter 2.2.1. The text model applied in this thesis employs tokenization- and embedding steps. The model tokenization layer maps the input words into a token sequence containing unique integer tokens for each unique word. The model then embeds this sequence into an embedding space, which creates one floating-point embedding vector per token in the sequence.

A LPM can input the text to the model and intersect the output of the embedding layer. Next, the LPM can instruct IG to explain these intermediary *hidden features* instead of the text input. This explanation can translate back from the embedding vector attributions to the word tokens and likewise from the tokens to the words. IG uses the concept of a baseline, and a baseline is a data point for which the model's confidence is completely undecided. A common baseline for text is the zero embedding vector, so the LPM can use a distance measurement (e.g., Euclidean) for the attributions of the embedding vector to obtain the contribution weight of one word. Therefore, based on the embedding-to-token-to-word translation, each word will have one embedding vector with contribution weights. The LPM should propagate each word label where the corresponding contribution vector is distant enough from the embedding baseline. This strategy is similar to the zero closeness approach as with IG explanations for images from chapter 3.3.3. This label propagation strategy warrants further exploration in future work.

6

Conclusion

To conclude this thesis, we revisit the problem statement and research questions, detail the high-level findings of the experiments, point out areas for future work, and provide concluding remarks.

6.1 Problem Statement & Research Questions

Data labels enforce compliance with data restrictions via label propagation from the input to the output of a procedure. End-to-end traceability is the consecutive label propagation in a system's computational pipeline. Systems containing Machine Learning (ML) components increase the risk of violating data restrictions, which can lead to significant legal consequences. ML models combine the information of input data completely and hinder the application of influence-based propagation policies. Propagating every label to the model output is a potentially conservative assumption since not every input always contributes to the output. To our knowledge, previous work has yet to address this problem.

Explainable Artificial Intelligence (XAI) is the study of methods and techniques to explain ML models and alleviate their black-box properties. We propose an LPM which leverages the XAI method's explanations to act as propagation policies for the black-box model outputs. We identify the following research questions related to this approach and the problem statement:

I **How can we apply XAI toward the purpose of labeling model outputs based on labeled input data?**

II **How do the output labels change for image- and text datasets with different models and XAI methods?**

We answer the first research question as follows:
Each explainability method requires special attention to the constitution of the explanation to apply them for label propagation. We work with methods that create explanations containing the features important to the model prediction. A LPM must consider the XAI method parameters to produce an explanation that is successful and faithful to the inner workings of the model. Hence, the LPM can propagate labels of input features depending on the features present in the XAI explanation.

We answer the second research question as follows:
The methodology of this thesis follows the *Abstraction* principle from the three paradigms of computer science. We design and conduct experiments and analyze from the results whether the approach is a valid proof of concept. The experiments consist of one image from the ImageNet dataset and four samples from the Internet Movie Database (IMDb) Sentiment Review dataset. We select various models for ImageNet classification and train a transformer classification model for text. The experimentation combines the XAI methods with the data and models such that the results show variation in each category. Finally, we use label restrictiveness as a measure to evaluate the change in the propagated labels.

6.2 Experiment Findings

The results show a strong limitation in which test case combinations are applicable with the specified hardware. The experiments aim to provide results reproducible on commercial hardware, and many explainability methods encounter difficulties in creating explanations. These difficulties relate to both image and text data but mostly depend on the size of the model. The XAI methods that do not successfully create explanations either fail due to Out Of Memory (OOM) errors or completing the procedure without a proper explanation.

There are multiple factors affecting the propagated output labels of a LPM. Larger models imply a more complex decision-making process and some XAI exhibit difficulties in faithfully explaining the model prediction. The data point subject to explanation can have a model prediction near a complex decision boundary between classes. The prediction surface near these boundaries can influence XAI methods with an erratic behavior such that faithful explanations are hard to find. Various labeling strategies for data directly affect the propagated labels. If the labels change and the data does not, an XAI method might put the same emphasis on the data such that the propagated labels change according to the new labels. XAI methods employ fundamentally different strategies for explaining models. Naturally, the methods cause the propagated labels to differ since the explanation is core to the LPM's decision. The methods considered in this thesis can be summarized from our experiences as follows:

- Anchor is a model-agnostic XAI method that produces explanations with nuanced attention to the strictly necessary input features but tends to find explanations with few features relative to the others.

- Integrated Gradients (IG) is a model-specific XAI method that produces explanations with highly precise emphasis on the input features but often fails due to our hardware limitations.

- Local Interpretable Model-agnostic Explanations (LIME) is a model-specific XAI method that reliably produces explanations but tends to be more misleading than the others.

Each method exhibits high faithfulness with their explanations. The propagated labels in our results are very restrictive according to the Lattice-Based Access Control (LBAC) and labeling strategy we have used. The few cases with relaxed label propagation are the ones where the explanations contain few features.

6.3 Future Work

The results from this thesis are inconclusive regarding the enforcement of data restrictions in an end-to-end system containing ML models. Further exploration should be made before this approach is applicable to this problem. An interesting direction to investigate is whether the parameters for specific XAI methods create explanations where they failed in our experiments. Some methods offer both positive and negative contributing features as part of the explanation. The argument regarding whether or not the labels of negatively contributing features should propagate to the output remains unsettled in this thesis.

There are many circumstances surrounding the experiment design that are highly improvable with a more applicable context than the datasets and labeling we have used. Consider a context involving video surveillance data with an object detection model annotating the frames and bounding boxes of people in the video. In this case, a model with a classification objective different from the detection model could be explained with XAI where the propagated labels depend on the bounding boxes labeled "*person*". An approach like this or similar seems less arbitrary than the data and labeling strategies employed in this thesis.

Bibliography

[1] Github copilot. Accessed 13:50 12th of May, 2024. URL: https://github.com/features/copilot/.

[2] Grammarly. Acccessed 10:38 15th of May, 2024. URL: https://www.grammarly.com/.

[3] Imdb website. Accessed 10:51 12th of May, 2024. URL: https://www.imdb.com/.

[4] ISO/IEC TR 29119-11:2020. Software and systems engineering, software testing, part 11: Guidelines on the testing of ai-based systems, 11 2020. Citation retrieved 9th of January 2024 from https://en.wikipedia.org/wiki/Explainable_artificial_intelligence#cite_ref-ISO29119_35-0. URL: https://www.iso.org/standard/79016.html.

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016. URL: https://arxiv.org/abs/1603.04467, arXiv:1603.04467.

[6] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels. *Technical report, EPFL*, 6 2010. URL: https://www.researchgate.net/publication/44234783_SLIC_superpixels.

[7] Wendy Kan Addison Howard, Eunbyung Park. Imagenet object localization challenge, 2018. URL: https://kaggle.com/competitions/imagenet-

object-localization-challenge.

[8] Ali Al-Aradi, Adolfo Correia, Danilo Naiff, Gabriel Jardim, and Yuri Saporito. Solving nonlinear and high-dimensional partial differential equations via deep learning. 11 2018. URL: https://www.researchgate.net/publication/329115789_Solving_Nonlinear_and_High-Dimensional_Partial_Differential_Equations_via_Deep_Learning.

[9] Emily Atkinson. Man crushed to death by robot in south korea, 11 2023. BCC article on robot accident. Accessed on 18th of March 2024. URL: https://bbc.com/news/world-asia-67354709.

[10] Jean Bacon, David Eyers, Thomas F. J.-M. Pasquier, Jatinder Singh, Ioannis Papagiannis, and Peter Pietzuch. Information flow control for secure cloud computing. *IEEE Transactions on Network and Service Management*, 11(1):76–89, 2014. URL: https://ieeexplore.ieee.org/abstract/document/6701293, doi:10.1109/TNSM.2013.122313.130423.

[11] Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for neural networks with local renormalization layers. In Alessandro E.P. Villa, Paolo Masulli, and Antonio Javier Pons Rivero, editors, *Artificial Neural Networks and Machine Learning – ICANN 2016*, pages 63–71, Cham, 2016. Springer International Publishing. URL: https://link.springer.com/chapter/10.1007/978-3-319-44781-0_8#citeas.

[12] Blake Brittain. Openai hit with new lawsuits from news outlets over ai training, 2 2024. Reuters' article on lawsuit against OpenAI and Microsoft. Accessed on 14th of March 2024. URL: https://www.reuters.com/legal/litigation/openai-hit-with-new-lawsuits-news-outlets-over-ai-training-2024-02-28/.

[13] François Chollet et al. Keras. https://keras.io, 2015.

[14] François Chollet et al. Keras applications. https://keras.io, 2015. Accessed 20:25 7th of May, 2024. URL: https://keras.io/api/applications/.

[15] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2017. URL: https://arxiv.org/abs/1610.02357, arXiv:1610.02357.

[16] Papers With Code. Sentiment analysis on imdb. Accessed 20:43 12th of May, 2024. URL: https://paperswithcode.com/sota/sentiment-analysis-on-

imdb.

[17] Dorothy E. Denning. A lattice model of secure information flow. *Commun. ACM*, 19(5):236–243, 5 1976. doi:10.1145/360051.360056.

[18] P.J. Denning, D.E. Comer, D. Gries, M.C. Mulder, A. Tucker, A.J. Turner, and P.R. Young. Computing as a discipline. *Computer*, 22(2):63–70, 1989. URL: https://ieeexplore.ieee.org/document/19833, doi:10.1109/2.19833.

[19] Michael Aaron Dennis. Marvin minsky - encyclopedia britannica, 4 2024. Accessed 17:27 29th of April, 2024. URL: https://www.britannica.com/biography/Marvin-Lee-Minsky.

[20] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale, 2022. URL: https://arxiv.org/abs/2208.07339, arXiv:2208.07339.

[21] Anders Tungeland Gjerdrum, Håvard Dagenborg Johansen, Lars Brenna, and Dag Johansen. Diggi: A secure framework for hosting native cloud functions with minimal trust. In *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 18–27, 2019. URL: https://ieeexplore.ieee.org/abstract/document/9014360, doi:10.1109/TPS-ISA48467.2019.00012.

[22] R.C. Gonzalez and R.E. Woods. *Digital Image Processing, 4th Ed.* Pearson, 2018. URL: https://dl.icdst.org/pdfs/files4/01c56e081202b62bd7d3b4f8545775fb.pdf.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[24] Google. Neural machine translation with a transformer and keras, 3 2024. Accessed 19:05 12th of May, 2024. URL: https://www.tensorflow.org/text/tutorials/transformer.

[25] Michael Grynbaum and Ryan Mac. The times sues openai and microsoft over a.i. use of copyrighted work, 12 2023. New York Times' article on lawsuit against OpenAI and Microsoft. Accessed on 14th of March 2024. URL: https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html.

[26] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, and G-Z. Yang. Xai-explainable artificial intelligence. *Science Robotics*, 4(37), 12 2019. This

is the author's version of the work. It is posted here by permission of the AAAS for personal use, not for redistribution. The definitive version was published in Science Robotics on 4 (37) 18 December 2019, DOI: 10.1126/scirobotics.aay7120. URL: https://openaccess.city.ac.uk/id/eprint/23405/, doi:10.1126/scirobotics.aay7120.

[27] Steinar Brenna Hansen. Correctness criteria for function-based reclassifiers: A language based approach. Master's thesis, UiT Norges arktiske universitet, 6 2022. URL: https://hdl.handle.net/10037/26005.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2016. URL: https://arxiv.org/abs/1512.03385.

[29] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208, doi:https://doi.org/10.1016/0893-6080(89)90020-8.

[30] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL: https://arxiv.org/abs/2106.09685, arXiv:2106.09685.

[31] Steve Huffman. Addressing the community about changes to our api, 6 2023. Reddit post from CEO. Accessed on 14th of March 2024. URL: https://www.reddit.com/r/reddit/comments/145bram/addressing_the_community_about_changes_to_our_api/.

[32] IBM. What is explainable ai? Retrieved 9th of January 2024. URL: https://www.ibm.com/topics/explainable-ai#:~:text=Explainable%20AI%20is%20used%20to,putting%20AI%20models%20into%20production.

[33] Intel. 13th gen intel(r) core(tm) i7-13700, 5 2024. Accessed 20:30 8th of May, 2024. URL: https://www.intel.com/content/www/us/en/products/sku/230490/intel-core-i713700-processor-30m-cache-up-to-5-20-ghz/specifications.html.

[34] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang,

Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL: https://arxiv.org/abs/2310.06825, arXiv:2310.06825.

[35] Bjørn Aslak Juliussen, Elisavet Kozyri, Dag Johansen, and Jon Petter Rui. The third country problem under the gdpr: enhancing protection of data transfers with technology. *International Data Privacy Law*, 13(3):225–243, 7 2023. arXiv:https://academic.oup.com/idpl/article-pdf/13/3/225/51556073/ipad013.pdf, doi:10.1093/idpl/ipad013.

[36] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL: https://arxiv.org/abs/1412.6980, arXiv:1412.6980.

[37] Janis Klaise, Arnaud Van Looveren, Giovanni Vacanti, and Alexandru Coca. Alibi explain: Algorithms for explaining machine learning models. *J. Mach. Learn. Res.*, 22(1), 1 2021. URL: https://dl.acm.org/doi/10.5555/3546258.3546439.

[38] Janis Klaise, Arnaud Van Looveren, Giovanni Vacanti, and Alexandru Coca. Alibi's anchor documentation, 1 2021. Accessed 20:48 28th of April, 2024. URL: https://docs.seldon.io/projects/alibi/en/stable/methods/Anchors.html.

[39] Janis Klaise, Arnaud Van Looveren, Giovanni Vacanti, and Alexandru Coca. Alibi's documentation, 1 2021. Accessed 20:48 28th of April, 2024. URL: https://docs.seldon.io/projects/alibi/en/stable/.

[40] Elisavet Kozyri, Owen Arden, Andrew C. Myers, and Fred B. Schneider. *JRIF: Reactive Information Flow Control for Java*, pages 70–88. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-030-19052-1_7.

[41] Elisavet Kozyri, Stephen Chong, and Andrew C. Myers. Expressing information flow properties. *Foundations and Trends® in Privacy and Security*, 3(1):1–102, 2022. URL: http://dx.doi.org/10.1561/3300000008, doi:10.1561/3300000008.

[42] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1), 2021. URL: https://www.mdpi.com/1099-4300/23/1/18, doi:10.3390/e23010018.

[43] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis.

In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, 6 2011. Association for Computational Linguistics. URL: http://www.aclweb.org/anthology/P11-1015.

[44] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL: https://arxiv.org/abs/1301.3781, arXiv:1301.3781.

[45] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences, 2018. URL: https://arxiv.org/abs/1706.07269, arXiv:1706.07269.

[46] Christoph Molnar. *Interpretable Machine Learning*. bookdown.org, 2 edition, 2022. URL: https://christophm.github.io/interpretable-ml-book.

[47] Tor-Arne S. Nordmo, Aril B. Ovesen, Håvard D. Johansen, Michael A. Riegler, Pål Halvorsen, and Dag Johansen. Dutkat: A multimedia system for catching illegal catchers in a privacy-preserving manner. In *Proceedings of the 2021 ACM Workshop on Intelligent Cross-Data Analysis and Retrieval*, ICDAR '21, pages 57–61, New York, NY, USA, 8 2021. Association for Computing Machinery. doi:10.1145/3463944.3469102.

[48] NVIDIA. Nvidia geforce rtx 3070, 5 2024. Accessed 20:30 8th of May, 2024. URL: https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3070-3070ti/#specs.

[49] The pandas development team. pandas-dev/pandas: Pandas, 2 2020. doi:10.5281/zenodo.3509134.

[50] Thomas Pasquier, Xueyuan Han, Mark Goldstein, Thomas Moyer, David Eyers, Margo Seltzer, and Jean Bacon. Practical whole-system provenance capture. In *Proceedings of the 2017 Symposium on Cloud Computing*, SoCC '17, pages 405–418, New York, NY, USA, 2017. Association for Computing Machinery. URL: https://dl.acm.org/doi/abs/10.1145/3127479.3129249, doi:10.1145/3127479.3129249.

[51] Dylan Patel and Afzal Ahmad. We have no moat, and neither does openai, 4 2023. Accessed 11:40 13th of May, 2024. URL: https://www.semianalysis.com/p/google-we-have-no-moat-and-neither.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Pas-

sos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL: https://www.jmlr.org/papers/v12/pedregosa11a.html.

[53] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2016. URL: https://arxiv.org/abs/1506.02640.

[54] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Lime's github repository, 8 2016. Accessed 20:48 28th of April, 2024. URL: https://github.com/marcotcr/lime/blob/master/doc/images/.

[55] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA, 8 2016. Association for Computing Machinery. doi:10.1145/2939672.2939778.

[56] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchor's github repository, 4 2018. Accessed 20:48 28th of April, 2024. URL: https://github.com/marcotcr/anchor/blob/master/notebooks/Images_todo.ipynb.

[57] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 4 2018. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11491, doi:10.1609/aaai.v32i1.11491.

[58] Amazon Web Services. Model explainability with aws artificial intelligence and machine learning solutions, 9 2021. URL: https://docs.aws.amazon.com/whitepapers/latest/model-explainability-aws-ai-ml/interpretability-versus-explainability.html.

[59] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. URL: https://www.cambridge.org/core/books/understanding-machine-learning/3059695661405D25673058E43C8BE2A6.

[60] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings*

of Machine Learning Research, pages 3145–3153. PMLR, 8 2017. URL: `https://proceedings.mlr.press/v70/shrikumar17a.html`.

[61] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014. URL: `https://arxiv.org/abs/1312.6034`, `arXiv:1312.6034`.

[62] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. URL: `https://arxiv.org/abs/1409.1556`, `arXiv:1409.1556`.

[63] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. URL: `https://arxiv.org/abs/1412.6806`, `arXiv:1412.6806`.

[64] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. URL: `https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf`.

[65] Håkon Kvale Stensland, Vamsidhar Reddy Gaddam, Marius Tennøe, Espen Helgedagsrud, Mikkel Næss, Henrik Kjus Alstad, Asgeir Mortensen, Ragnar Langseth, Sigurd Ljødal, Østein Landsverk, Carsten Griwodz, Pål Halvorsen, Magnus Stenhaug, and Dag Johansen. Bagadus: An integrated real-time system for soccer analytics. *ACM Trans. Multimedia Comput. Commun. Appl.*, 10(1s), 1 2014. `doi:10.1145/2541011`.

[66] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks, 2017. URL: `https://arxiv.org/abs/1703.01365`, `arXiv:1703.01365`.

[67] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, Second Edition: An Introduction*. The MIT Press, 11 2018. URL: `https://mitpress.mit.edu/9780262039246/reinforcement-learning/`.

[68] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016. URL: `https://arxiv.org/abs/1602.07261`, `arXiv:1602.07261`.

[69] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens,

and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015. URL: https://arxiv.org/abs/1512.00567, arXiv:1512.00567.

[70] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training, 2021. URL: https://arxiv.org/abs/2104.00298, arXiv:2104.00298.

[71] Enrico Tedeschi, Tor-Arne S. Nordmo, Dag Johansen, and Håvard D. Johansen. On optimizing transaction fees in bitcoin using ai: Investigation on miners inclusion pattern. *ACM Trans. Internet Technol.*, 22(3), 6 2022. doi:10.1145/3528669.

[72] Enrico Tedeschi, Tor-Arne S. Nordmo, Dag Johansen, and Håvard D. Johansen. Predicting transaction latency with deep learning in proof-of-work blockchains. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4223–4231, 2019. URL: https://ieeexplore.ieee.org/abstract/document/9006228, doi:10.1109/BigData47090.2019.9006228.

[73] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Fourth Edition*. Academic Press, Inc., USA, 4th edition, 2008.

[74] 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, 2023. URL: https://arxiv.org/abs/2302.13971, arXiv:2302.13971.

[75] A. M. TURING. I.—computing machinery and intelligence. *Mind*, LIX(236):433–460, 10 1950. arXiv:https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf, doi:10.1093/mind/LIX.236.433.

[76] European Union. Gdpr fines and penalties, 04 2016. Accessed on 1st of April 2024. URL: https://gdpr-info.eu/issues/fines-penalties.

[77] European Union. Assessment list for trustworthy artificial intelligence (altai) for self-assessment, 7 2020. European Commision's final assessment presentation. URL: https://digital-strategy.ec.europa.eu/en/library/assessment-list-trustworthy-artificial-intelligence-altai-self-assessment.

[78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[79] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi:10.25080/Majora-92bf1922-00a.

[80] XDevelopers. Discontinued free api access, 2 2023. X tweet announcement from developers. Accessed on 14th of March 2024. URL: https://twitter.com/XDevelopers/status/1621026986784337922.

[81] Muhammad Rehman Zafar and Naimul Mefraz Khan. Dlime: A deterministic local interpretable model-agnostic explanations approach for computer-aided diagnosis systems, 2019. URL: https://arxiv.org/abs/1906.10263, arXiv:1906.10263.

[82] Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2528–2535, 2010. URL: https://ieeexplore.ieee.org/abstract/document/5539957, doi:10.1109/CVPR.2010.5539957.

[83] Christopher T. Zirpoli. Generative artificial intelligence and copyright law (v6), 9 2023. US Congressional Research Service report. Accessed on 1st of April 2024. URL: https://crsreports.congress.gov/product/details?prodcode=LSB10922.

A

Declaration of the Use of Artificial Intelligence

This chapter states the usage of Artificial Intelligence (AI) in conjunction with the thesis. The writing of this thesis includes no application of *generative* Artificial Intelligence (AI) from scratch. Grammarly spell-checked the thesis and provided suggestions to rephrase sentences [2]. This tool applies AI (even generative) to suggest improvements to written text. As previously stated, we do not use this AI to create content from nothing but only to improve text already written. This process involved copying from the thesis into the website and copying them back after corrections. The process of applying corrections consists of manually considering each suggestion in these steps:

1. Re-reading the sentence subject to a suggestion.

2. Re-reading the surrounding paragraph for context.

3. Evaluating whether the change would deliver the same message as intended.

4. Evaluating the reason for the suggestion (e.g., "*Rewrite for clarity*" or "*Rewrite in active voice*").

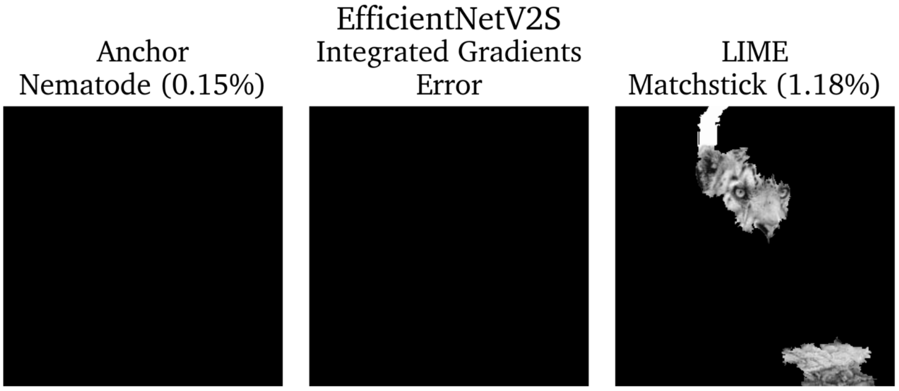
5. Accepting or dismissing the suggestion.

The implementation of the experiments in this thesis extensively utilized Github Copilot [1], a tool that provides intelligent code suggestions and completions. We use the AI primarily for auto-completion of repetitive pieces of code in addition to generating some experiments from scratch. Copilot requires constant manual attention and proofing of the code suggestions, as experienced through usage. This tool was turned off halfway through the experimentation due to a lack of suggestions relevant to the experiments.

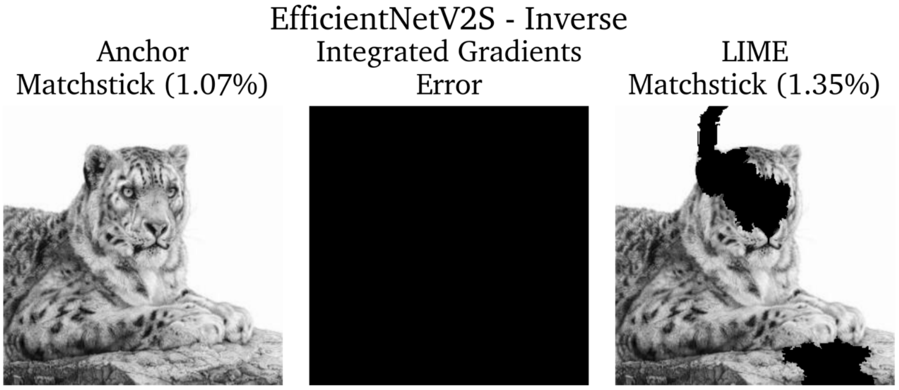
/B

Image Experiments Appendix

B.1 EfficientNet-V2-S

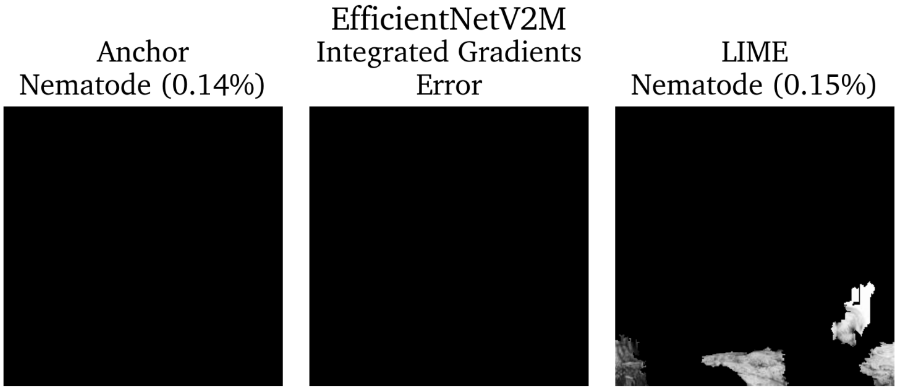


(a) The explanations from Anchor, IG, and LIME for the EfficientNet-V2-S model prediction of the explainee image in figure 4.4.

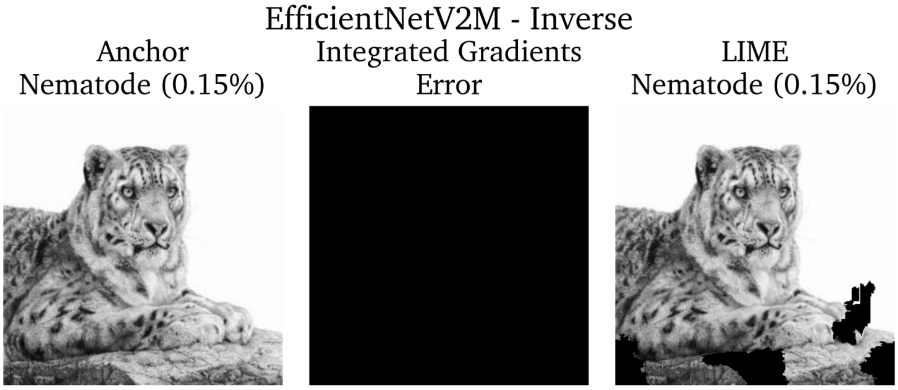


(b) The classification of the inverse explanations from Anchor, IG, and LIME for the EfficientNet-V2-S model.

B.2 EfficientNet-V2-M

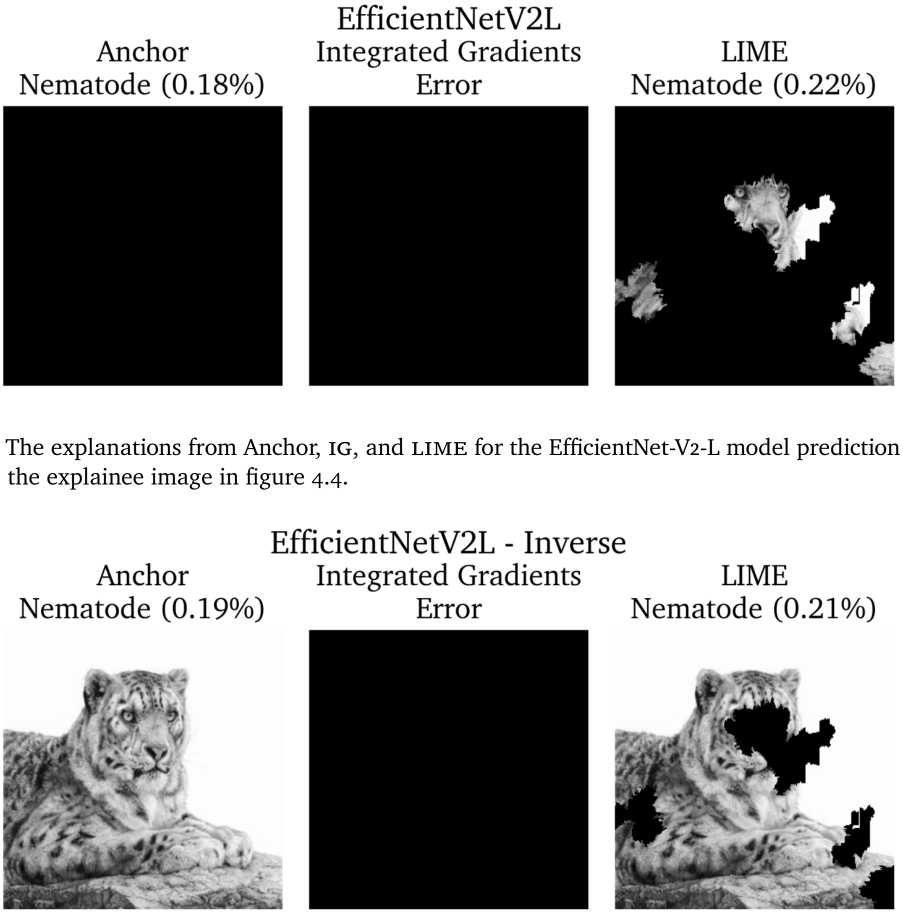


(a) The explanations from Anchor, IG, and LIME for the EfficientNet-V2-M model prediction of the explainee image in figure 4.4.



(b) The classification of the inverse explanations from Anchor, IG, and LIME for the EfficientNet-V2-M model.

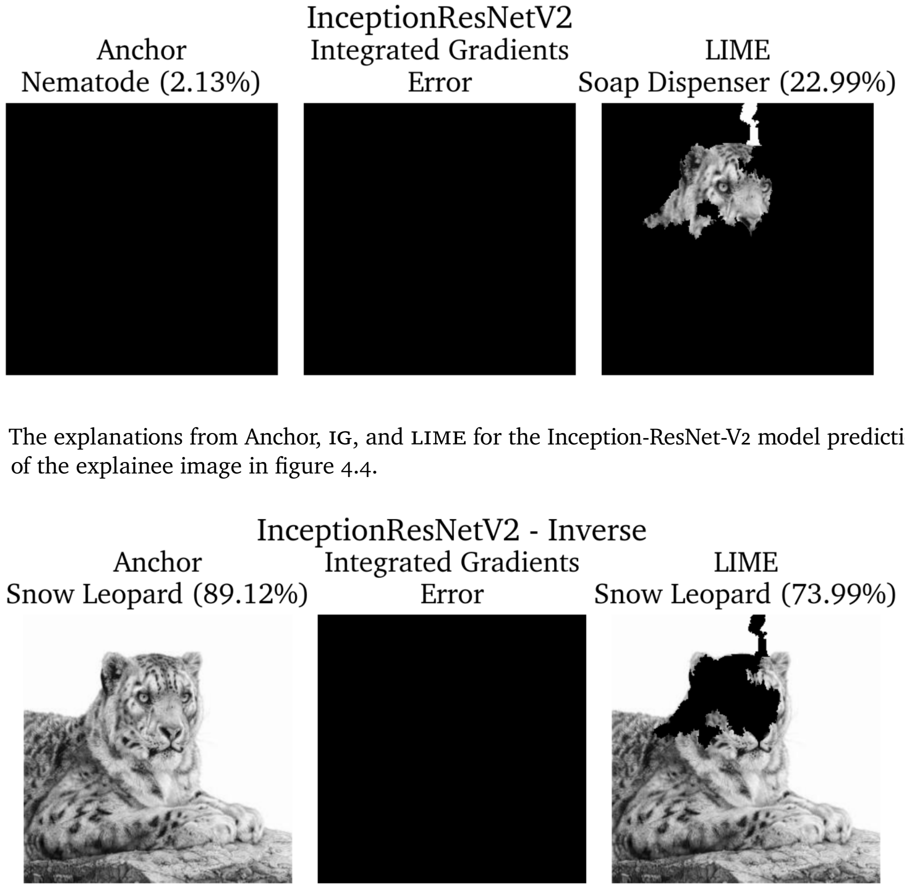
B.3 EfficientNet-V2-L



(a) The explanations from Anchor, IG, and LIME for the EfficientNet-V2-L model prediction of the explainee image in figure 4.4.

(b) The classification of the inverse explanations from Anchor, IG, and LIME for the EfficientNet-V2-L model.

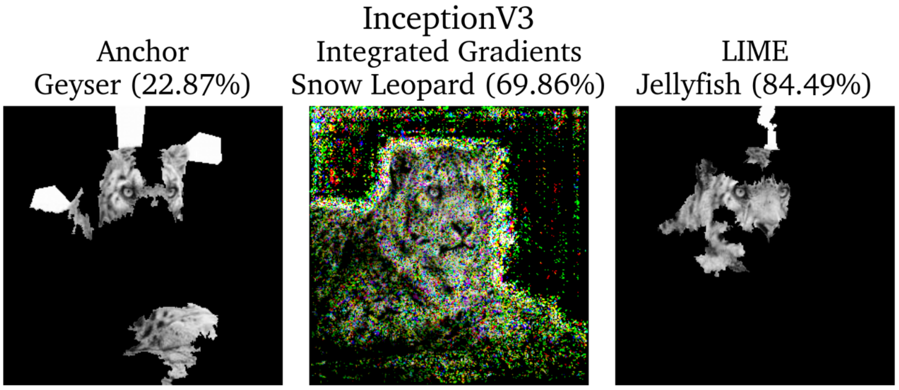
B.4 Inception-ResNet-V2



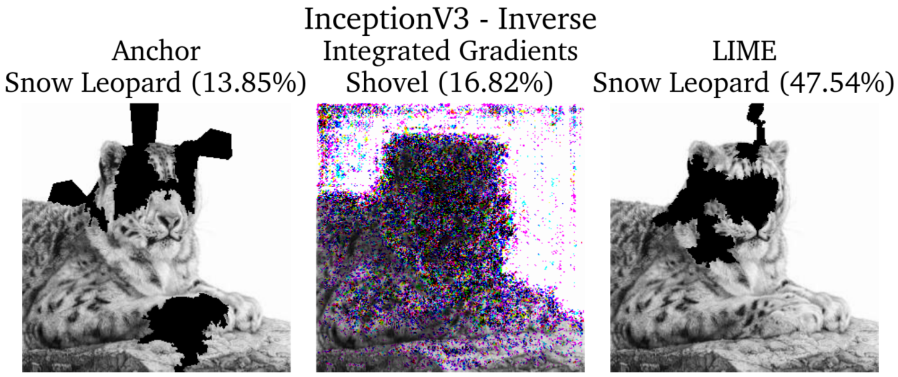
(a) The explanations from Anchor, IG, and LIME for the Inception-ResNet-V2 model prediction of the explainee image in figure 4.4.

(b) The classification of the inverse explanations from Anchor, IG, and LIME for the Inception-ResNet-V2 model.

B.5 Inception-V3

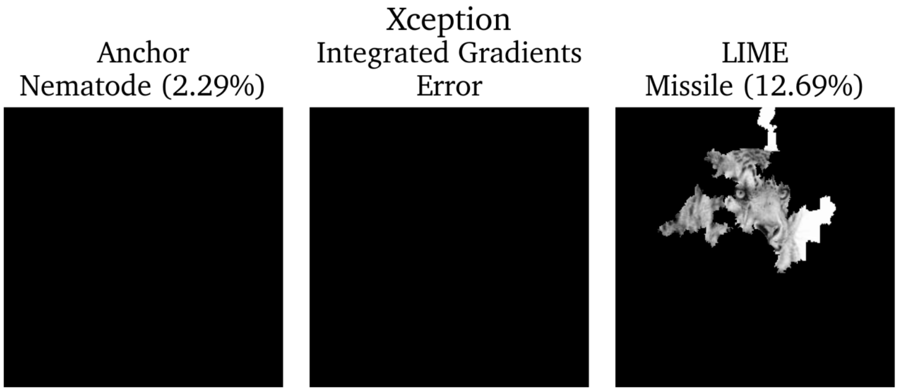


(a) The explanations from Anchor, IG, and LIME for the Inception-V3 model prediction of the explainee image in figure 4.4.

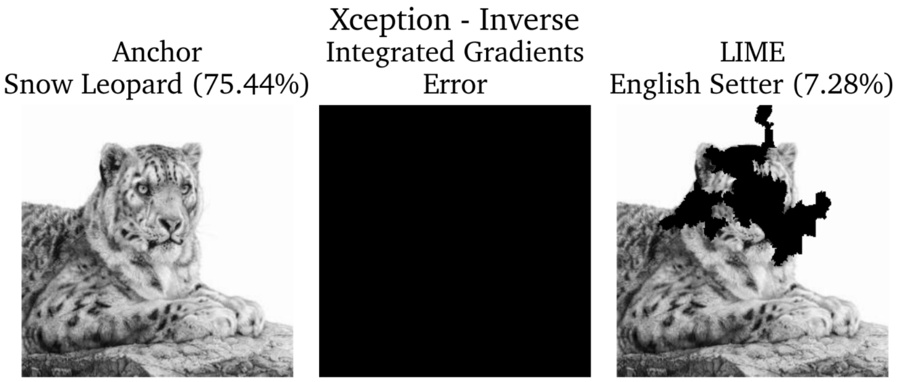


(b) The classification of the inverse explanations from Anchor, IG, and LIME for the Inception-V3 model.

B.6 Xception

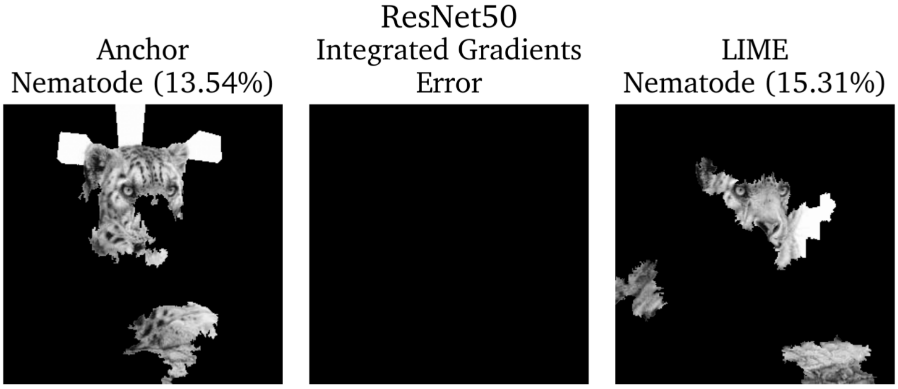


(a) The explanations from Anchor, IG, and LIME for the Xception model prediction of the explainee image in figure 4.4.

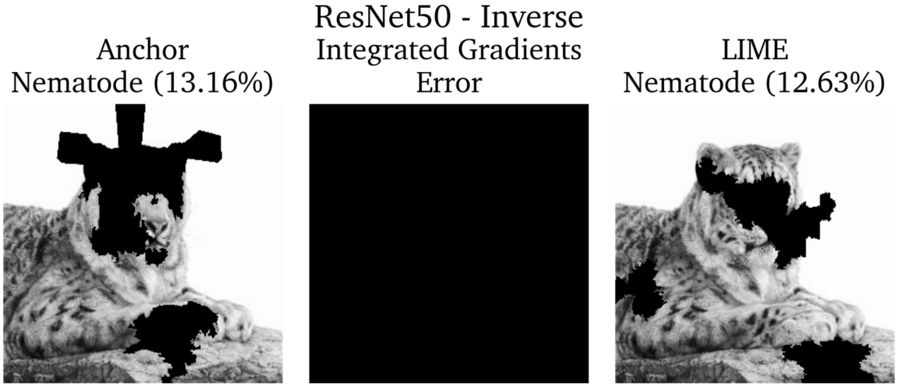


(b) The classification of the inverse explanations from Anchor, IG, and LIME for the Xception model.

B.7 ResNet-50

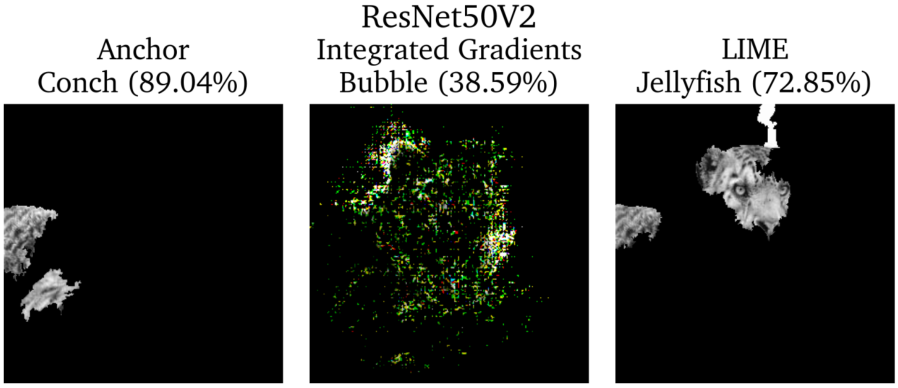


(a) The explanations from Anchor, IG, and LIME for the ResNet-50 model prediction of the explainee image in figure 4.4.

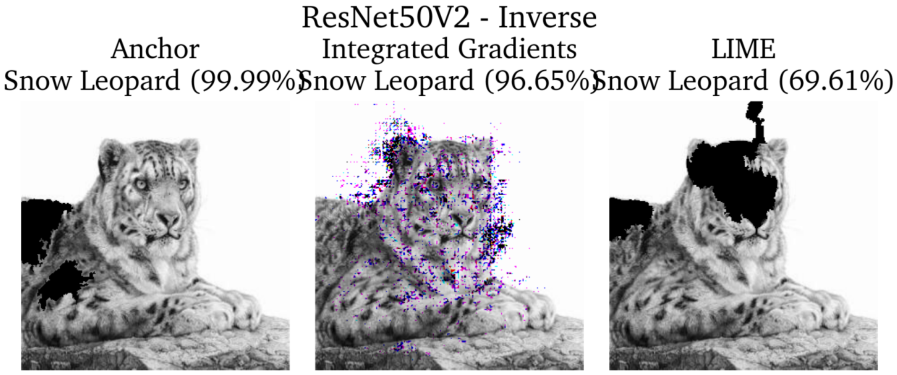


(b) The classification of the inverse explanations from Anchor, IG, and LIME for the ResNet-50 model.

B.8 ResNet-50-V2

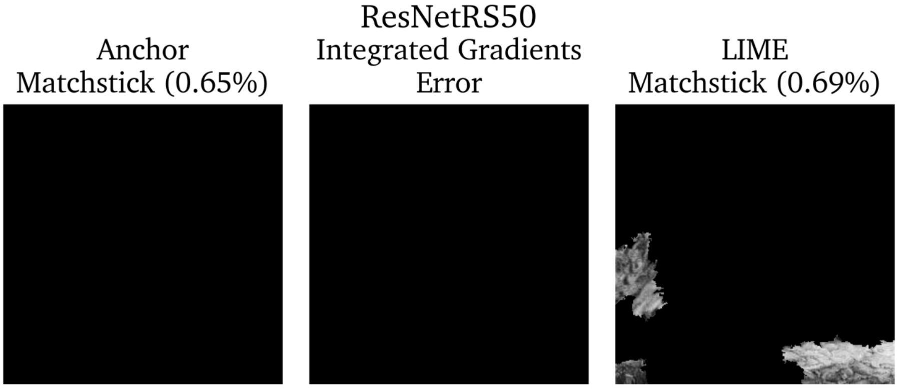


(a) The explanations from Anchor, IG, and LIME for the ResNet-50-V2 model prediction of the explainee image in figure 4.4.

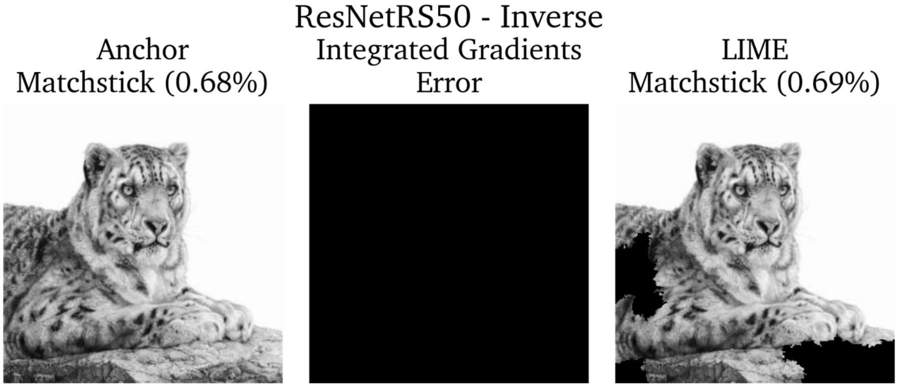


(b) The classification of the inverse explanations from Anchor, IG, and LIME for the ResNet-50-V2 model.

B.9 ResNet-RS-50

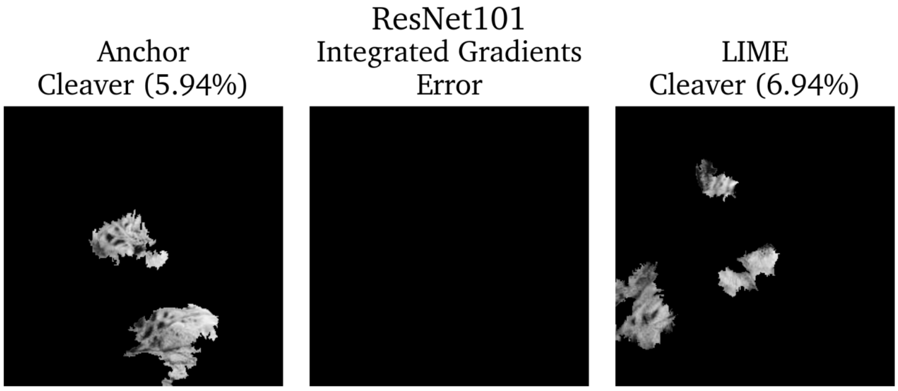


(a) The explanations from Anchor, IG, and LIME for the ResNet-RS-50 model prediction of the explainee image in figure 4.4.

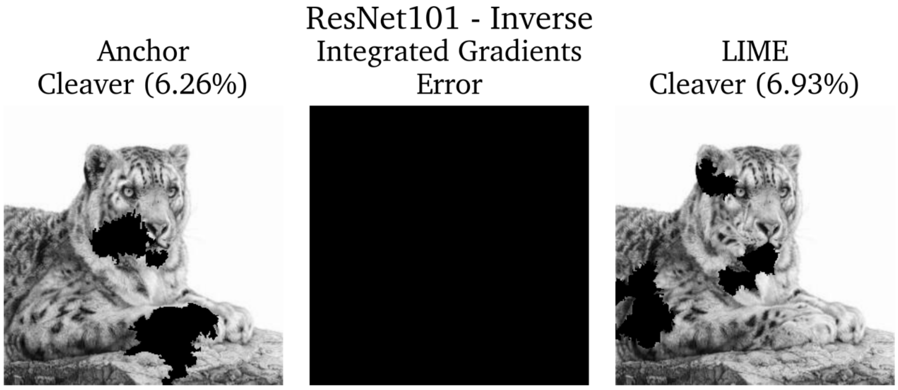


(b) The classification of the inverse explanations from Anchor, IG, and LIME for the ResNet-RS-50 model.

B.10 ResNet-101

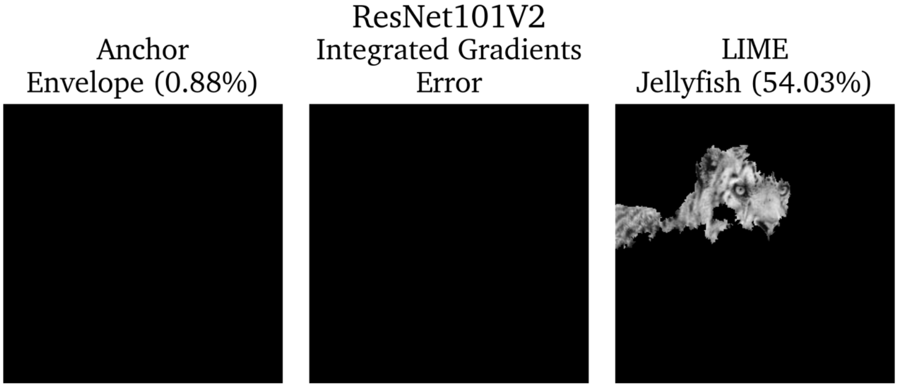


(a) The explanations from Anchor, IG, and LIME for the ResNet-101 model prediction of the explainee image in figure 4.4.

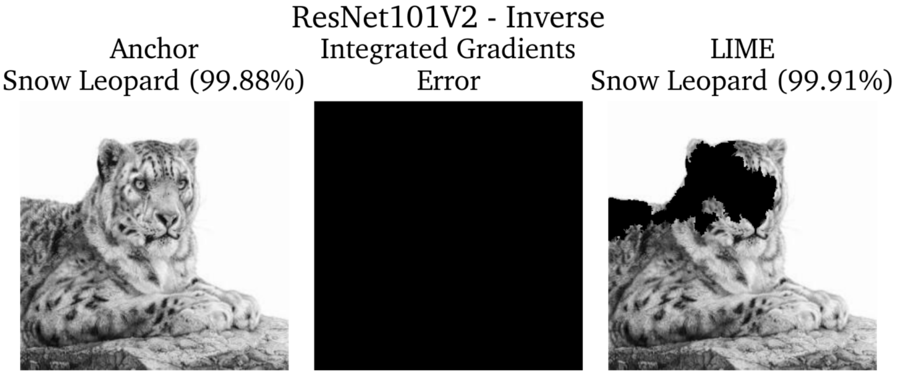


(b) The classification of the inverse explanations from Anchor, IG, and LIME for the ResNet-101 model.

B.11 ResNet-101-V2

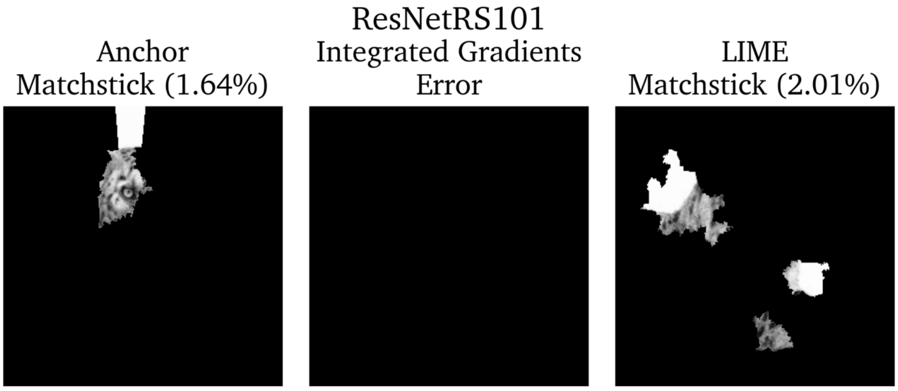


(a) The explanations from Anchor, IG, and LIME for the ResNet-101-V2 model prediction of the explainee image in figure 4.4.

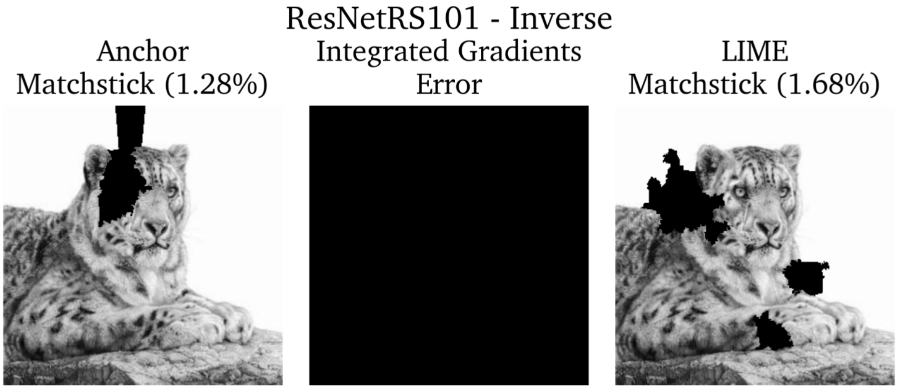


(b) The classification of the inverse explanations from Anchor, IG, and LIME for the ResNet-101-V2 model.

B.12 ResNet-RS-101

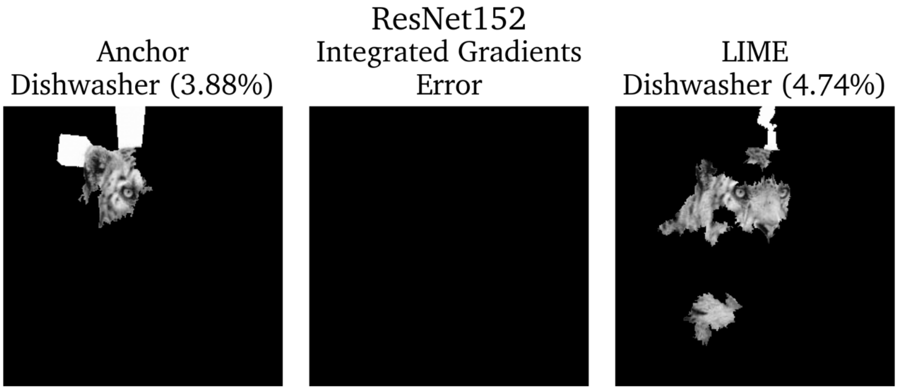


(a) The explanations from Anchor, IG, and LIME for the ResNet-RS-101 model prediction of the explainee image in figure 4.4.

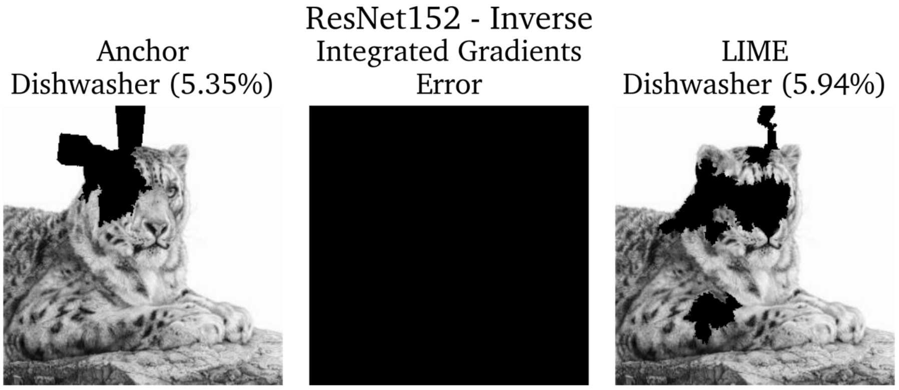


(b) The classification of the inverse explanations from Anchor, IG, and LIME for the ResNet-RS-101 model.

B.13 ResNet-152

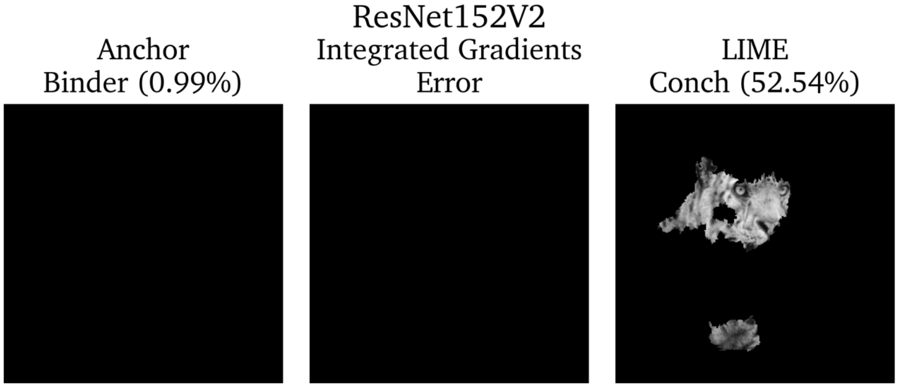


(a) The explanations from Anchor, IG, and LIME for the ResNet-152 model prediction of the explainee image in figure 4.4.

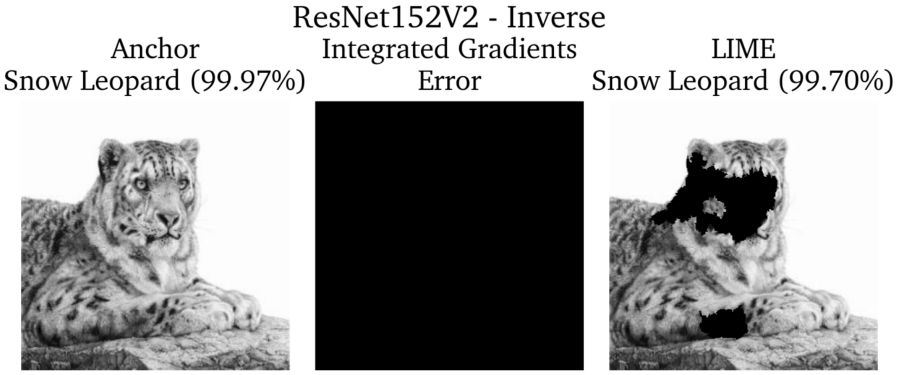


(b) The classification of the inverse explanations from Anchor, IG, and LIME for the ResNet-152 model.

B.14 ResNet-V2-152

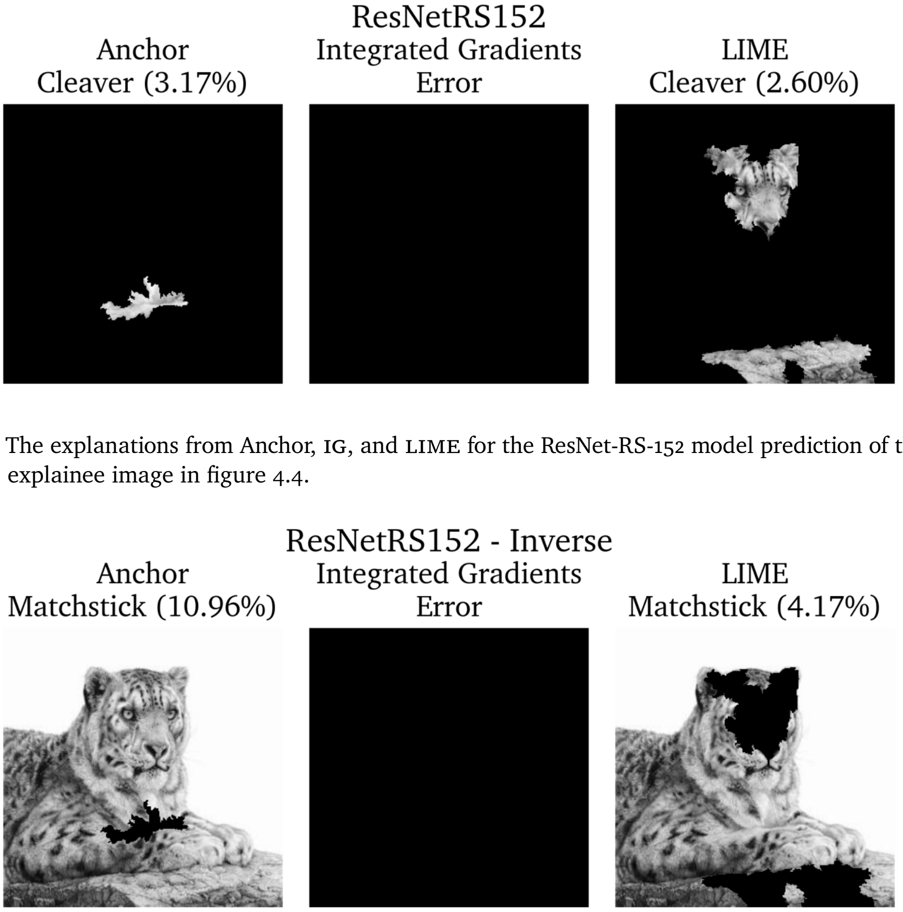


(a) The explanations from Anchor, IG, and LIME for the ResNet-V2-152 model prediction of the explainee image in figure 4.4.



(b) The classification of the inverse explanations from Anchor, IG, and LIME for the ResNet-V2-152 model.

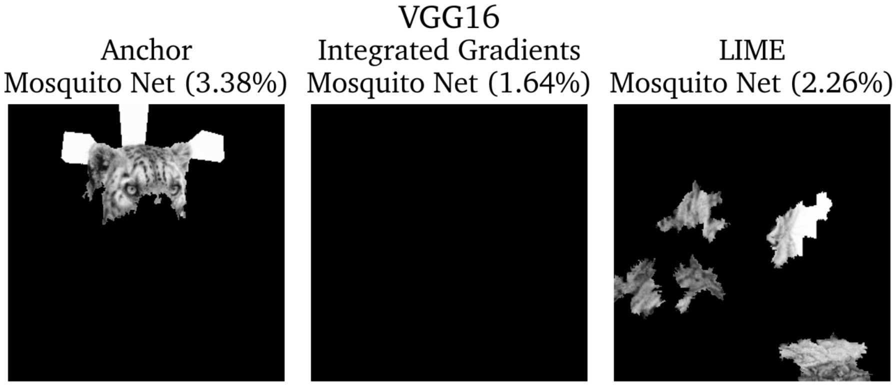
B.15 ResNet-RS-152



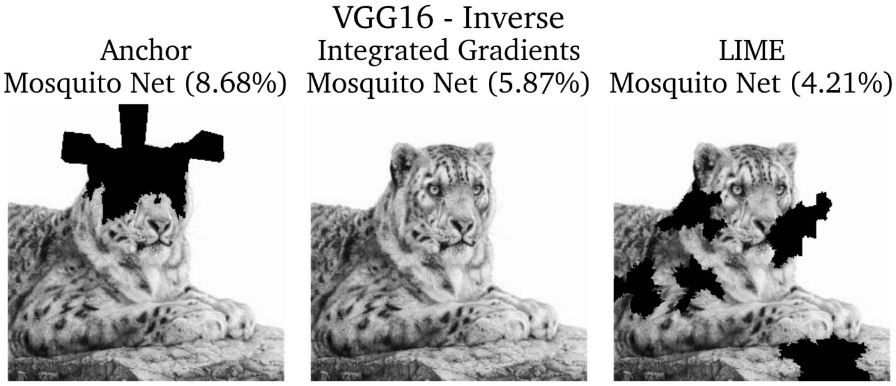
(a) The explanations from Anchor, IG, and LIME for the ResNet-RS-152 model prediction of the explainee image in figure 4.4.

(b) The classification of the inverse explanations from Anchor, IG, and LIME for the ResNet-RS-152 model.

B.16 VGG16



(a) The explanations from Anchor, IG, and LIME for the VGG16 model prediction of the explainee image in figure 4.4.



(b) The classification of the inverse explanations from Anchor, IG, and LIME for the VGG16 model.

/C

Text Experiment Appendix

C.1 Correct Classification & High Confidence – LIME

Correct Classification & High Confidence: LIME				
Highest Contribution	Words < 34	Weight	Positions	Label
1	boredom	0.083958	[8]	Non-Spoiler
2	waste	0.068526	[102]	Non-Spoiler
3	save	0.038567	[50]	Spoiler
4	sat	0.037744	[1]	Non-Spoiler
5	loved	−0.033068	[84]	Non-Spoiler
6	impressed	−0.027901	[40]	Non-Spoiler
7	even	0.025654	[14, 34, 81]	Non-Spoiler
8	dreary	0.023308	[76]	Non-Spoiler
9	hours	0.017068	[5, 80, 96]	Non-Spoiler
10	maybe	0.016447	[64, 70]	Spoiler
11	through	0.015972	[3]	Non-Spoiler
12	beauty	−0.015897	[86]	Non-Spoiler
13	pure	0.01581	[7]	Non-Spoiler
14	250	−0.01568	[28, 126]	Spoiler
15	why	0.015337	[120]	Non-Spoiler
16	though	−0.014721	[15, 35, 82]	Spoiler
17	not	0.011861	[18, 54, 72]	Non-Spoiler
18	it	−0.011433	[16, 23, 90, 98, 131]	Non-Spoiler
19	mail	0.011193	[115]	Spoiler
20	big	0.009922	[101]	Non-Spoiler
21	much	0.009154	[74]	Spoiler
22	good	−0.009107	[134]	Non-Spoiler
23	missing	0.009085	[108]	Non-Spoiler
24	my	−0.008547	[94, 104]	Non-Spoiler
25	was	0.008001	[30, 89, 99, 132]	Non-Spoiler
26	dont	0.007473	[128]	Spoiler
27	2	−0.007444	[4, 79, 95, 136]	Non-Spoiler
28	movie	0.007304	[52, 63, 112, 135]	Spoiler
29	sure	−0.007151	[55]	Non-Spoiler
30	still	−0.006612	[48]	Spoiler
31	a	−0.006158	[31, 100, 133]	Non-Spoiler
32	down	−0.00607	[2]	Spoiler
33	and	−0.00587	[43, 127]	Non-Spoiler
34	curious	−0.005409	[119]	Spoiler

Table C.1: LIME's explanation for the review in figure 4.9 with top-34 words as filter.

C.2 Correct Classification & Low Confidence – LIME

Correct Classification & Low Confidence: LIME				
Highest Contribution	Words < 30	Weight	Positions	Label
1	boiled	−0.458414	[91]	Spoiler
2	obrien	0.221142	[4, 36]	Non-Spoiler
3	identity	−0.119855	[39]	Non-Spoiler
4	money	0.101197	[21, 53]	Non-Spoiler
5	fund	0.100563	[23]	Non-Spoiler
6	wasted	0.095916	[60]	Spoiler
7	operation	0.08888	[26]	Non-Spoiler
8	pretends	0.080576	[65]	Spoiler
9	noir	−0.067264	[93]	Spoiler
10	hardly	0.056785	[106]	Spoiler
11	kiss	−0.05273	[116]	Spoiler
12	gripping	−0.05224	[107]	Spoiler
13	doesnt	0.050284	[76, 100]	Spoiler
14	pat	−0.04769	[3]	Non-Spoiler
15	starring	0.045817	[2]	Non-Spoiler
16	surprises	−0.045371	[68]	Spoiler
17	drama	−0.044856	[1]	Non-Spoiler
18	least	0.044371	[113]	Spoiler
19	premise	0.043567	[58]	Spoiler
20	washed	0.042329	[7]	Non-Spoiler
21	rap	−0.04149	[48]	Non-Spoiler
22	up	−0.03945	[8, 12, 87]	Spoiler
23	return	−0.03854	[50]	Non-Spoiler
24	accents	0.03704	[99]	Spoiler
25	contains	−0.036883	[110]	Spoiler
26	reporter	0.036409	[9]	Non-Spoiler
27	even	0.031734	[72]	Spoiler
28	performances	−0.031715	[77]	Spoiler
29	it	−0.031684	[66, 74, 109]	Spoiler
30	across	0.029029	[80]	Spoiler

Table C.2: LIME's explanation for the review in figure 4.10 with top-30 words as filter.

C.3 Wrong Classification & High Confidence – LIME

Correct Classification & Low Confidence: LIME				
Highest Contribution	**Words < 35**	**Weight**	**Positions**	**Label**
1	reagan	0.08581	[109, 130]	Spoiler
2	suffers	0.071932	[15]	Non-Spoiler
3	wash	−0.054339	[12]	Non-Spoiler
4	faces	0.054161	[6, 14, 55]	Non-Spoiler
5	unfortunately	0.051615	[110]	Spoiler
6	sequel	0.049254	[1, 50]	Non-Spoiler
7	ann	0.044727	[40, 63, 79, 105]	Spoiler
8	dirty	−0.038141	[5, 54]	Non-Spoiler
9	cliché	0.02365	[78]	Spoiler
10	angels	−0.019757	[3, 11, 52]	Non-Spoiler
11	shining	−0.019468	[70]	Spoiler
12	would	0.018333	[92, 133]	Spoiler
13	end	−0.018102	[24, 37]	Non-Spoiler
14	buy	−0.016803	[93]	Spoiler
15	kings	0.01662	[138]	Non-Spoiler
16	sake	0.016476	[60]	Non-Spoiler
17	usual	−0.015445	[19]	Non-Spoiler
18	far	0.01471	[119]	Spoiler
19	another	0.014434	[96]	Spoiler
20	example	0.014405	[126]	Non-Spoiler
21	only	0.01437	[9]	Non-Spoiler
22	true	−0.0125	[69]	Spoiler
23	least	0.012424	[57]	Non-Spoiler
24	shenanigans	0.011946	[20]	Non-Spoiler
25	row	−0.011409	[139]	Non-Spoiler
26	this	0.011373	[42, 73, 99, 111]	Spoiler
27	an	0.010031	[48]	Spoiler
28	with	0.009815	[4, 31, 53]	Non-Spoiler
29	have	0.009629	[44]	Non-Spoiler
30	better	−0.008986	[125]	Non-Spoiler
31	their	−0.008697	[13]	Non-Spoiler
32	paraphrase	−0.007841	[76]	Spoiler
33	dvd	−0.007516	[95]	Spoiler
34	of	−0.007351	[21, 29, 34, 62, 72, 98, 113, 127]	Spoiler
35	from	−0.007218	[17, 83]	Spoiler

Table C.3: LIME's explanation for the review in figure 4.11.

C.4 Wrong Classification & Low Confidence – LIME

Correct Classification & Low Confidence: LIME				
Highest Contribution	**Words** < 24	**Weight**	**Positions**	**Label**
1	horrible	−0.157005	[56]	Spoiler
2	kramer	0.140233	[5, 95]	Spoiler
3	stanley	0.119845	[4, 94]	Spoiler
4	candidate	0.086456	[80]	Spoiler
5	principle	0.071538	[65]	Spoiler
6	misses	−0.066101	[82]	Spoiler
7	muddled	−0.057274	[46]	Spoiler
8	puzzling	−0.052353	[14]	Spoiler
9	and	0.050627	[11, 25, 36, 48, 81]	Spoiler
10	pieces	0.043782	[20]	Spoiler
11	rooney	0.038	[38]	Spoiler
12	why	−0.03262	[49]	Spoiler
13	hackman	−0.031381	[29]	Spoiler
14	a	0.030758	[18, 55, 60, 86, 90]	Spoiler
15	producer	0.029517	[2]	Spoiler
16	edward	−0.028455	[32]	Spoiler
17	very	0.025499	[13, 45, 87]	Spoiler
18	richard	−0.025049	[30]	Spoiler
19	wide	−0.02246	[88]	Spoiler
20	gave	−0.021328	[54]	Spoiler
21	bergen	−0.020924	[52]	Spoiler
22	wants	0.020421	[66]	Spoiler
23	left	−0.020109	[21]	Spoiler
24	such	−0.018358	[59]	Spoiler

Table C.4: LIME's explanation for the review in figure 4.12.