applied sciences

MDPI

Article

Latent Graph Attention for Spatial Context in Light-Weight Networks: Multi-Domain Applications in Visual Perception Tasks

Ayush Singh [1], Yash Bhambhu [1], Himanshu Buckchash [2,*], Deepak K. Gupta [1,2] and Dilip K. Prasad [2]

[1] Indian Institute of Technology, ISM Dhanbad, Jharkhand 826004, India; ayush.singh.222k@gmail.com (A.S.); yashbhambhu.18je0949@am.iitism.ac.in (Y.B.); guptadeepak2806@gmail.com (D.K.G.)

[2] Department of Computer Science, UiT The Arctic University of Norway, 9019 Tromsø, Norway; dilip.parsad@uit.no

* Correspondence: himanshu.buckchash@uit.no

Abstract: Global contexts in images are quite valuable in image-to-image translation problems. Conventional attention-based and graph-based models capture the global context to a large extent; however, these are computationally expensive. Moreover, existing approaches are limited to only learning the pairwise semantic relation between any two points in the image. In this paper, we present Latent Graph Attention (LGA), a computationally inexpensive (linear to the number of nodes) and stable modular framework for incorporating the global context in existing architectures. This framework particularly empowers small-scale architectures to achieve performance closer to that of large architectures, making the light-weight architectures more useful for edge devices with lower compute power and lower energy needs. LGA propagates information spatially using a network of locally connected graphs, thereby facilitating the construction of a semantically coherent relation between any two spatially distant points that also takes into account the influence of the intermediate pixels. Moreover, the depth of the graph network can be used to adapt the extent of contextual spread to the target dataset, thereby able to explicitly control the added computational cost. To enhance the learning mechanism of LGA, we also introduce a novel contrastive loss term that helps our LGA module to couple well with the original architecture at the expense of minimal additional computational load. We show that incorporating LGA improves performance in three challenging applications, namely transparent object segmentation, image restoration for dehazing and optical flow estimation.

Keywords: graph attention; light-weight network; convolutional neural network; transparent object segmentation; neural networks; optical flow estimation; attention; deep learning

check for updates

Citation: Singh, A.; Bhambhu, Y.; Buckchash, H.; Gupta, D.K.; Prasad, D.K. Latent Graph Attention for Spatial Context in Light-Weight Networks: Multi-Domain Applications in Visual Perception Tasks. *Appl. Sci.* **2024**, *14*, 10677. https://doi.org/10.3390/app142210677

Academic Editor: Pedro Couto

Received: 17 September 2024
Revised: 24 October 2024
Accepted: 25 October 2024
Published: 19 November 2024

1. Introduction

Recent advancements related to convolutional neural networks (CNNs) have led to significant advancements in image-to-image translation problems. Some popular examples include edge region-based segmentation [1] and dehazing with dark channel priors [2], among others. This success can be attributed to the ability of CNNs to extract deep and complex features from images without needing any hand-crafted features.

The underlying mechanism of CNNs is that they reduce the spatial information contained in images by gradually down-sampling to a set of feature maps. For this purpose, CNNs employ convolutional kernels that convolve pixels only locally, without taking the whole global context into consideration. This works well for conventional downstream tasks such as classification. However, the lack of global context limits their performance on challenging image-to-image translation problems such as the segmentation of transparent objects, dehazing, and optical flow estimation. In transparent objects, although the edges may bear a clear signature of the object, there are strong features of other background objects that adversely affect the segmentation process. To tackle this issue, it is desirable that the information from the edges be transferred to other parts of the object as well. In dehazing, shapes and edges of the object need to be restored from original images where hazing fuzzifies these features; information about the non-local context should help in this task.

Several previous works have been reported on the incorporation of the global spatial context into the learning process. Some examples include layer-wise average feature augmentation [3]; the use of mixtures of conditional random fields and CNNs [4]; and the construction of encoder–decoder-style, fully convolutional networks that use deconvolutions to create output, building upon U-Net [5]. Two popular approaches for introducing global context without compromising localization accuracy are the use of atrous or dilated convolutions [6] and transformer networks [7]. Although both approaches have been successful, they are accompanied by significant increases in the number of parameters and the associated FLOPs. Alternatively, criss-cross attention (referred to as CCNet) [8] represents a light-weight solution to model the global context in the learning process. It collects contextual information from vertical and horizontal criss-cross channels and applies an affinity operation in the latent space. Compared to other approaches, this method is computationally efficient, with a time complexity of $\mathcal{O}(N^{1.5})$, where N denotes the size of the feature map.

Figure 1 presents the segmentation results obtained using CCNet vs. LGA. Interestingly, the ability to capture the global context plays an adverse role in this case. CCNet relies on the semantic relation between any two distant points; however, it does not take into account the characteristics of the intermediate points between them. Clearly, as seen in Figure 1, parts of other objects being segmented falsely is our object of interest. To circumvent this issue, we propose an alternate scheme that spatially constructs *chained attention*, thereby taking into account the information of the entire route to compute the relation between any two distant pixels. This is achieved through the use of locally connected graph networks constructed over the latent feature maps, and these gradually propagate information to the distant neighbors of any point through a stacked set of layers used in the graph network. Our approach is adaptive and provides the flexibility of choosing the desired spatial extent to be captured. Furthermore, our approach is computationally faster and provides a speedup in $\sqrt{N}$ over CCNet.

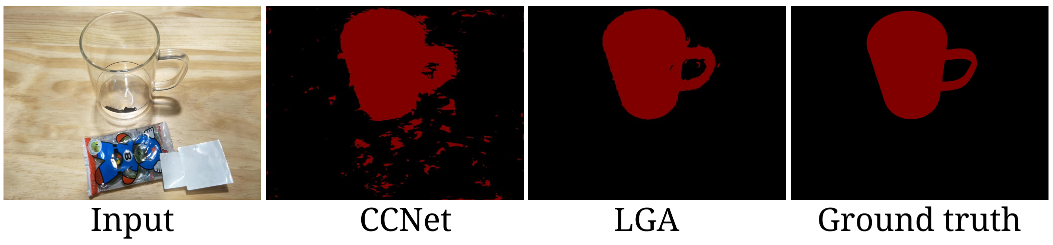


Figure 1. Segmentation results obtained for a transparent object using CCNet [8] and our method (LGA). Note that due to the lack of semantic coherence between far-away points, CCNet produces partially incorrect segmentation.

Contributions. The main contributions of this paper can be summarized as follows:

- We present Latent Graph Attention (LGA), a graph network-based module, to incorporate the global context in existing CNN architectures.
- LGA is computationally inexpensive, and it provides a speedup of $\sqrt{N}$ over previous methods. The time complexity of LGA scales linearly with the number of connected nodes in the graph.
- For stable and responsive learning, we introduce an LGA-specific contrastive loss term that amplifies the discrimination between the foreground and background and, accordingly, helps to learn the suitable weights for each edge of the graph.

- We experimentally demonstrate that our LGA module, when plugged into a small-scale architecture, helps to boost performance with only minimal additional computational load. This empowers the development of efficient and compact architectures for edge devices.
- We experimentally demonstrate the efficacy of LGA over a variety of hard image-to-image translation tasks, including the segmentation of transparent objects, image dehazing, and optical flow estimation.

2. Related Work

Context is a useful prior for understanding object–part relations with respect to each other and the background. We divide the literature review into three subsections based on the context aggregation technique, light-weight CNN architecture, and graph-based method for attention in CNNs.

Context aggregation. Spatial context provides valuable information for many image-to-image translation tasks [9]. Context aggregation is a recurrent theme that appears in different types of convolution strategies [6], such as pooling variations, attention mechanisms [8], channel operations, and architectural search [10]. Dilated convolution, multi-scale feature generation, and large-kernel approaches leverage contextual information based on modulation of the receptive field to extract information from non-local regions [6]. Jin et al. proposed a generalized model based on a physics-inspired network [11]. Architecture search approaches such as Auto-DeepLab, Global2Local, and DCNAS automate context modeling by searching multiple paths and channels at the cell or network level [12]. Unlike these methods, attention has served as a very versatile context aggregation mechanism, working at various scales, including local, non-local [9], global, and cross or self-attention [8] scales. Recent trends have shifted from variants of attention in CNNs to full attention in transformers [13]. However, this push towards the use of heavier networks for context aggregation has come at the cost of network size and resources [8]. These challenges demand the confluence of high-performing attention traits in small-size networks.

Light-weight architectures. Architectural buoyancy in CNNs is generally achieved through adaptations in convolutional operations such as grouping or reduction in memory access cost in small-scale architectures [14]. Zhang et al. used point-wise group convolutions and channel shuffling for cross-channel information exchange to greatly reduce computation cost while maintaining accuracy [15]. Huang et al. proposed light-weight attention in CNNs by aggregating the contextual information of pixels on orthogonal paths, as well as for the full image for every pixel [8]. Kong et al. proposed mutual knowledge distillation strategies for information flow between teacher–student networks for optical flow estimation [16]. Of these three method, an attention approach like the criss-cross method [8] is easily adaptable, highly modular and task-agnostic [8]. Criss-cross attention (or CCNet) is one of the best light-weight attention modules; however, it employs spatial position-agnostic correlation, which does not work well when sequential context propagation is required. Graph-based methods provide control over these pathways in devising chained attention mechanisms [17].

Graph-based networks. Despite developments in graph-based solutions like the generalization of CNNs as graphs [17], the use of graph attention networks for molecular substructures in drug discovery using feed-forward layer-based simple graph attention [18], and the use of sparse graph attention in scene graph generation [19], graph attention has not been explored from a chained attention perspective. Previously, Sun et al. presented an rgb-d-based dataset for transparent object segmentation [20]. Yu et al. presented polarization-based transparent object segmentation [21]. Banerjee et al. [22], proposed an end-to-end graph CNN method for transparent object segmentation by forming undirected edges between any two adjacent super-pixels. However, unlike directed graphs, undirected graphs are unable to associate two far-away nodes to the same structural context. Moreover, the adjacency matrix generation reported in [22] for an image resulting in N nodes bears a time complexity of N^2. This shows that achieving global context through the use of graphs is associated with high time complexity. To provide a more local context, Lu et al. [17] incorporated graphs in a fully connected network (FCN) for better segmentation. They considered FCN features located a distance of l away from each other as graph nodes. However, in the case of a context, their approach incurs higher space-time complexity.

We present latent graph attention (LGA), which overcomes the challenges outlined above. LGA employs outward directionality in an adjacency matrix through which it propagates information to its neighboring nodes. When LGA layers are stacked together multiple times, the information reaches outward from each node to the distant nodes, thereby providing a non-local spatial context.

3. The Proposed Approach

In this section, we present the concept of LGA and discuss its stability and its modular adoption across architectures.

3.1. Latent Graph Attention (LGA)

LGA is a computationally inexpensive graph-based attention network designed to improve the contextual information for any given image. It spatially constructs chained attention through the use of multiple layers of the graph, thereby taking into account the information of the entire route when computing the relation between two distant pixels in the latent space. LGA directly operates on the latent feature map and produces additional attention maps, which can directly be concatenated with the original feature map. Due to its simplistic nature, LGA can be incorporated with minimal effort into existing architectures. Figure 2 shows how LGA can be plugged into some popular yet challenging image-to-image translation tasks.

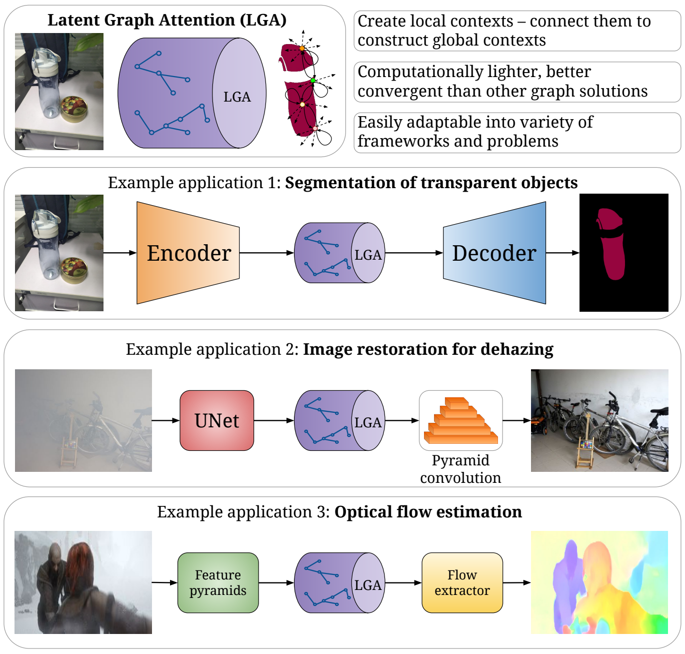

Latent Graph Attention (LGA)
LGA
Create local contexts – connect them to construct global contexts
Computationally lighter, better convergent than other graph solutions
Easily adaptable into variety of frameworks and problems
Example application 1: Segmentation of transparent objects
Encoder
LGA
Decoder
Example application 2: Image restoration for dehazing
UNet
LGA
Pyramid convolution
Example application 3: Optical flow estimation
Feature pyramids
LGA
Flow extractor

Figure 2. We present the novel concept of latent graph attention (LGA), which can be easily integrated in wide variety of applications and architectures. Three challenging and open problems are considered as example applications of LGA in this article.

Architecture of LGA. Figure 3 presents a schematic overview of the process of our LGA module. As input, it takes a feature map ($F^{\text{in}} \in \mathbb{R}^{H \times W \times C}$, where H, W and C denote the height, width and number of channels, respectively). It constructs a graph using this feature map (as shown in Figure 4). Every spatial feature on the feature map, with dimensions of $1 \times 1 \times C$, is considered a node of the graph, leading to $N = HW$ nodes in the entire graph, as shown in Figure 4.

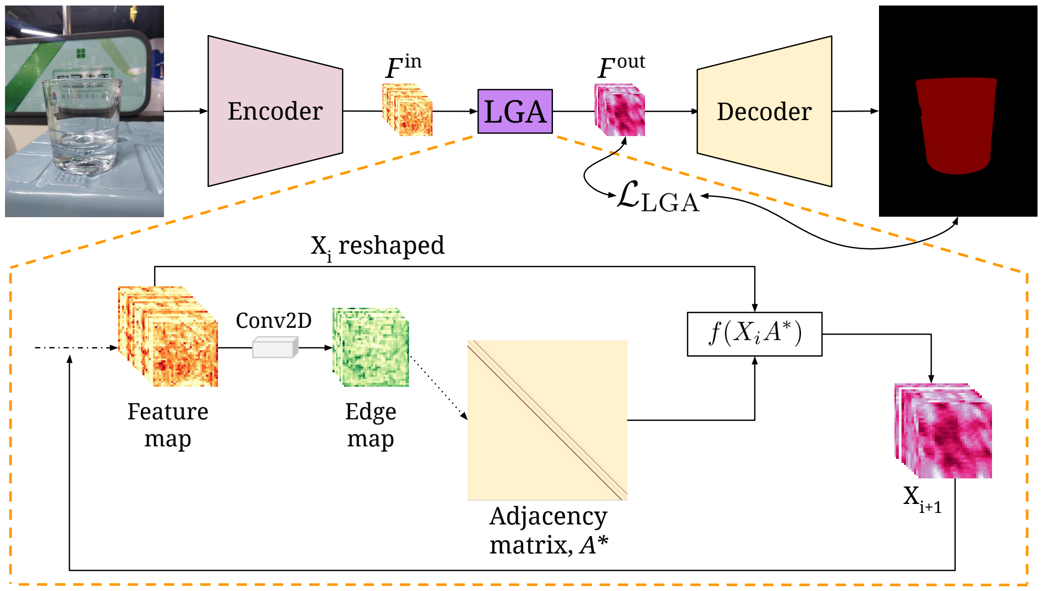
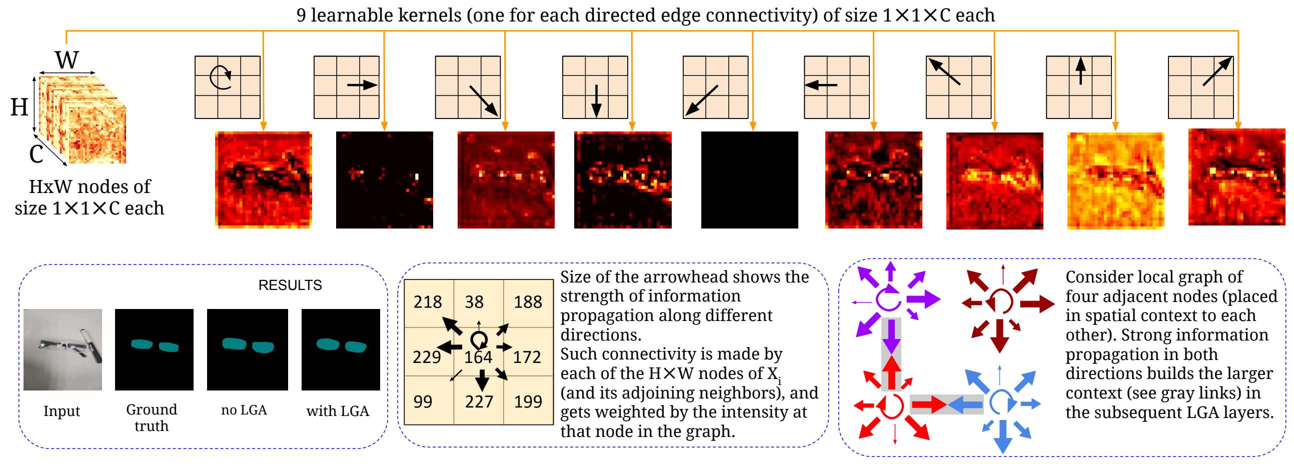


Figure 3. Operations performed by the LGA layer are shown in the expanded view. Starting with the encoder's feature maps (F^{in}), edge maps are created using 2D convolution. The dotted arrow between the *edge map* and *adjacency matrix* implies that this transfer happens only once, even if the LGA layer is repeated multiple times. Next, the normalized adjacency matrix is used to calculate the output (X_{i+1}) for the ith LGA layer. The LGA contrastive loss ($\mathcal{L}_{\text{LGA}}$) is computed between the output of the LGA module (F^{out}) and the ground truth. X_{i+1} becomes the input to the i + 1th LGA layer.



9 learnable kernels (one for each directed edge connectivity) of size 1×1×C each
W
H
C
HxW nodes of size 1×1×C each
RESULTS
Input
Ground truth
no LGA
with LGA
218
38
188
229
164
172
99
227
199

Size of the arrowhead shows the strength of information propagation along different directions.
Such connectivity is made by each of the H×W nodes of X_i (and its adjoining neighbors), and gets weighted by the intensity at that node in the graph.

Consider local graph of four adjacent nodes (placed in spatial context to each other). Strong information propagation in both directions builds the larger context (see gray links) in the subsequent LGA layers.

Figure 4. The top row shows the conversion of feature map X_i to graph nodes. Each cell of X_i in an $H \times W$ plane (at all depths) is considered as one node in the graph. The graph information is extracted by 9 kernels—1 for each direction. Edge maps (in top row) correspond to the input, gt, and predicted triplet (bottom left). The middle-bottom figure shows how the node intensities represent the edge weights. The bottom-right figure shows how connectivity strength information propagates through recursive LGA layers.

Learning the weights of the graph edges. Edges should be constructed in the graph depending on the directions in which a strong spatial relation is to be learned. We construct edges only between immediate spatial neighbors based on a 8-shaped connectivity pattern (see Figure 4). This leads to nine connections per node—one self-connection and eight with neighboring nodes. The weights of the edges are learnable, and each of these weights is generated using a one-layered CNN comprising a 1×1 convolution layer. Since we have nine edges, there are a total of nine such mini-networks that learn the edges.

Construction of adjacency matrix. The learned edges are used to construct an adjacency matrix (A). It is an $N \times N$ matrix, where $A_{ij} \in A$ denotes the weight of the edge connecting the ith node with the jth node in the graph. For the sake of convergence, A is normalized to form A^*. This normalization ensures that the feature vectors of all the nodes remain within a unit polysphere and are of comparable strengths with respect to each other. We define $A^* = AD^{-1}$, where $D_{ij} = \sum_j A_{ij} + \epsilon$. Here, ϵ is added to avoid zero division. The matrix (A) is normalized based on the weights of the outgoing edges, since it provides better results and stability during the training process.

Message passing. With this construction, LGA facilitates message passing between immediate neighbors in the latent space. To propagate information farther away, multiple such graphs are stacked to build a graph network that constitutes the LGA module. The information propagation at each LGA layer (Figure 3) can be stated as

$$X_{i+1} = f(X_i A^*). \tag{1}$$

where X_i and X_{i+1} denote the input and output of the ith layer, respectively (X_0 indicates F^{in}, and f is a learnable function used to change the dimensions of the feature vector of each node).

Learning in LGA. The core of LGA relies on constructing a representative directed graph. Learning in this graph is guided by the direct loss ($\mathcal{L}_{\text{LGA}}$) on the output feature map (F^{out}). The main goal of LGA is to capture neighboring context via message passing such that nodes corresponding to the same information have a similar distribution. Each node in the graph belongs to a particular spatial location in the image. We assume that each node corresponds to a particular patch in the GT (X^{GT}). This patch is calculated either via receptive field estimation of each node or by dividing the X^{GT} in N equal patches, where every node corresponds to a unique patch. Then, we calculate the similarity between these patches. For example, for the segmentation task, we calculate whether the patches are of the same classes. For image restoration tasks such as image dehazing, we calculate the structural similarity index measure (SSIM) between the patches; if the SSIM is greater than a predefined threshold, the patches are considered similar and not otherwise.

The nodes corresponding to similar patches are likely to have similar distributions. $\mathcal{L}_{\text{LGA}}$ penalizes F^{out} if nodes belonging to similar patches have different distributions or nodes corresponding to non-similar patches have similar distributions. This loss is defined such that LGA learns to predict richer representations to avoid learning an identity mapping between F^{in} and F^{out}.

LGA contrastive loss. This novel loss term helps in the learning of our LGA module. Mathematically, LGA contrastive loss can be stated as

$$\mathcal{L}_{\text{LGA}} = E_{P_i,P_j}\left(\mathcal{C}_{ij}\log\left(\frac{V_{ij}^2}{U_{ij}} + 1\right) + \bar{\mathcal{C}}_{ij}\log\left(\frac{U_{ij}}{V_{ij}^2} + 1\right)\right), \tag{2}$$

$V_{ij} = D(F_i^{\text{out}}, F_j^{\text{out}})$ and $U_{ij} = D(F_i^{\text{in}}, F_j^{\text{in}})$, where F^{in} and F^{out} denote the input and output feature maps for the LGA module, respectively, and $D(\cdot)$ is a divergence function, such as KL-divergence or mean square error. The variables $\mathcal{C}$ and $\bar{\mathcal{C}}$ are Boolean, and their values depend on the similarity between neighboring nodes in the graph. For example, for nodes i and j, we look at the GT labels of patches containing them. Furthermore, we aggregate the labels to assign a single label to the node. For example, one aggregate measure could be to assign the majority class label to the node. Let $Agg(\cdot)$ denote the aggregate function and i and j denote two nodes from the graph; then, $\mathcal{C} = 1$ if $Agg(i) = Agg(j)$ and 0 otherwise.

Next, we present the interpretation and consequence of using this loss function. We note that U_{ij} is determined by the input to the LGA module and does not update as LGA learns. LGA essentially learns by tweaking the value of V_{ij}, which is determined by the output of LGA. Note that a larger diversity value D_{ij} indicates larger differences in the distributions of nodes i and j. With these points, we now assess how the loss function behaves and influences the learning of LGA in different situations. For the case when two nodes have a similar distribution, V_{ij} needs to be minimized, implying that the two nodes have similar output distributions as well. For cases where the input distributions of the two nodes are very different, V_{ij} is learned to be maximized, thereby setting the two output distributions apart. However, using only V_{ij} is not useful, since the encoder also learns to generate better input feature maps. Even if the LGA does not improve, the feature map produces a V_{ij} equivalent to U_{ij}, the overall loss decrease, and it looks as if LGA is learning. Hence, we also incorporate U_{ij} in the loss function. More details can be found in the Supplementary Materials.

3.2. Other Features of LGA

In Section 3.1, we discussed how the loss function of LGA is designed to not only help LGA learn but also to influence learning in the preceding or succeeding networks. In Section 4.1, we discussed how LGA can be easily incorporated in different architectures in a modular fashion. There are two other major benefits of LGA, namely efficiency and scalability, in comparison with globally connected graph and attention models.

Efficiency. Constructing a globally connected graph, i.e., a graph where any node can be connected to any other node, is a computationally demanding task. For our LGA, we need only $9 \times N$ different 1×1 convolutional layers, where N is the number of nodes. This solution is more space- and time-efficient than most other approaches. LGA needs only $\mathcal{O}(N)$ space for to store edge weights, whereas the spatial complexity is mostly of the order of $\mathcal{O}(N^2)$ in attention or globally connected graph models. CCNet attention [8] has a spatial complexity of $\mathcal{O}(N^{1.5})$, which is still higher than that of our model. With respect to time complexity, most attention or globally connected graph models have a complexity of the order of $\mathcal{O}(NC^2 + N^2C)$, where NC^2 corresponds to the application of convolution to the feature map for channel reduction and N^2C represent information propagation. The term NC^2 can be reduced by a significant amount by applying depth-wise or group convolution. Hence, the best optimized complexity for such models is $\mathcal{O}(N^2C)$. In the case of CCNet, the complexity is $\mathcal{O}(NC^2 + N^{1.5}C)$, and the optimized complexity is $\mathcal{O}(N^{1.5}C)$. On the other hand for our network, the optimized time complexity is $\mathcal{O}(NC)$ because our module's layer takes only $\mathcal{O}(NC)$ for information propagation. As there are four layers in our LGA, the complexity is $\mathcal{O}(4NC)$, which is $\sqrt{N}$ times smaller than that of the best light-weight attention model (CCNet). We also provide the derivation of the space-time complexity of LGA vs. CCNet in the Supplemental Materials.

Scalability. Here, we use a 1×1 convolutional layer, assuming that a local edge (i.e., edge between spatially close nodes) can be constructed using the information of the node itself. However, incorporating information of a larger context is possible simply by using convolutional layers of larger sizes, for example, kernels with dimensions of 2×2 or higher. Similarly, although we used only four LGA layers, this can be easily extended to more layers such that a larger scale of global context can be included. In comparison, it is difficult to incorporate scalability and correlate features that are spatially far apart in attention models.

Effective scene context capture. As our LGA module propagates information between two nodes by passing information via neighboring nodes, it gradually captures the context of surrounding nodes, i.e., non-local context, for information propagation. Our LGA module also automatically takes the distance between nodes into consideration, since more information from a node can reach its neighbor earlier and more effectively, thereby implying that that the nodes closer to each other are more likely to have a similar structure when LGA is included.

4. Experiments

In this section, we present the details of the training process, the management of training data, the results of all experiments, and additional ablation studies for the following three tasks: dehazing, segmentation, and optical flow estimation.

4.1. Training Process

We converted all the input images and ground truth for all segmentations to dimensions of 512 × 512. The optimizer used in our training process was Adam [23]. For segmentation, we used various encoders, such as squeezenet, etc., adding our own decoder network. We added the LGA layer after the encoder and used the encoder output as the LGA input. For segmentation experiments, an initial learning rate of 10^{-4} was used, and after 30 epochs, we reduced it to 10^{-5}. We trained the models for 40 epochs. For dehazing, we used the approach reported in [24] as the base method. We reduced its size and FLOPs by replacing the standard convolutional layer with a group convolution layer. It is represented as BPPNet-reduced in this work. LGA was added after the stacked UNet layer proposed in [24]. The learning rate and decay mechanism were chosen to be similar to those used to train a generator, as described in [24]. The dataset used for segmentation was Trans10Kv2 [25]. This dataset is a modified version of Trans10k [25] with more fine-tuned classes i.e., 12. For dehazing, we used the I-Haze dataset [7]. For optical flow estimation, we used the ARFlow [26] architecture as the base method. We added the LGA layer after the feature extraction layer. We only applied LGA to a single feature map with a spatial size of 28 × 24. Additional details can be found in the Supplemental Materials .

4.2. Training Data

In this section, we describe the datasets used in this paper, along with their division into training, validation, and test sets. For the segmentation task, we used the Trans10K-v2 dataset, which is based on the original Trans10K dataset. It consists of 5000 images for training, 1000 images for validation, and 4428 images for testing. For the dehazing task, we utilized the I-Haze dataset, which contains 25 indoor hazy images (size 2833 × 4657 pixels) for training. It includes 5 hazy images for validation, each paired with its corresponding ground-truth image. We used the validation data as the test set in our experiments. For the optical flow task, we used the MPI Sintel benchmark. The MPI (Max Planck Institute) Sintel dataset is designed for optical flow evaluation and contains 1064 synthesized stereo images with ground-truth disparity data. Sintel is derived from the open-source 3D animated short film Sintel. The dataset includes 23 different scenes. The stereo images are RGB, while the disparity maps are grayscale. Both types of images have a resolution of 1024 × 436 pixels and 8 bits per channel. The test data are hosted on a server, and the ground truth is not publicly available. To obtain final metrics, we had to submit our results to the server for evaluation.

4.3. Segmentation

We compared LGA-incorporated squeeze and shuffle with previous challenging methods. Both of the base models can operate in real time and are computationally efficient. However, there performance is not at par with that of bulkier and/or computationally more intensive models (see Table 1 and Figure 5). The models including LGA outperformed all the efficient and real-time architectures in terms of performance. Shuffle with LGA also has the smallest number of parameters. Furthermore, both our models are computationally less expensive, but their performance is at par with that of lager models. In terms of comparison with larger models that comprise millions of parameters and need billions of FLOPs, our models are less demanding and perform better than PSPNet in terms of mIoU, although they are marginally inferior in terms of SSIM. DeepLab performed the best; however, this model is relatively very large. Nonetheless, it is evident that for real-time or compact application scenarios, LGA provides an efficient and performance-effective solution.

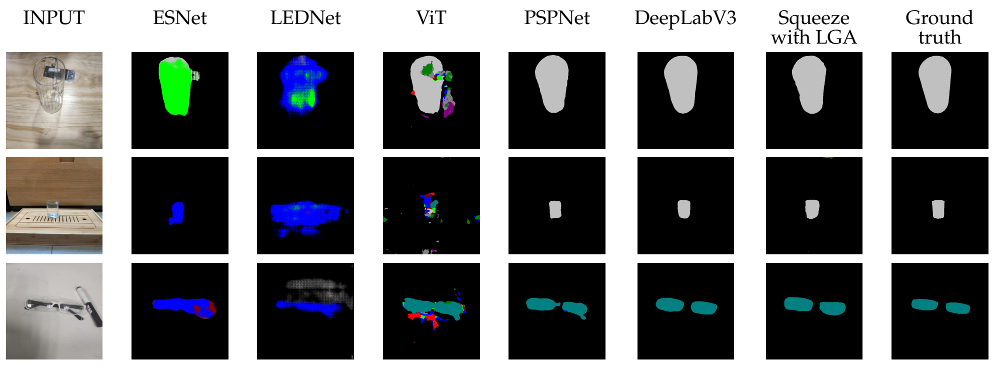


Figure 5. Examples of segmentation results of previous methods. Eleven classes of objects are color-coded.

Table 1. Segmentation results: quantitative comparison of various models.

MODEL	**Performance (↑)**		**Efficiency (↓)**	
	mIoU (%)	**Accuracy (%)**	**Parameters ($\times10^6$)**	**FLOPS ($\times10^9$)**
Efficient/real-time small architectures				
FPENet [27]	10.1	70.3	0.5	0.8
ESPNet-v2 [28]	12.3	73.0	3.5	0.8
ENet [29]	23.4	78.2	0.4	2.1
DABNet [30]	15.3	77.4	0.8	5.2
LEDNet [5]	30.3	72.9	1.1	19.6
ICNet [31]	23.4	78.2	7.8	10.6
MobileNet-v2 [32]	17.6	77.6	3.3	29.3
ESNet [33]	43.6	45.5	1.7	27.3
LGA incorporation into small architectures				
Shuffle [15] with LGA	44.5	78.7	0.4	3.5
Squeeze [34] with LGA	44.6	79.6	1.1	13.5
Large architectures				
ViT [13]	29.6	67.8	171.6	176.7
PSPNet (Res34) [10]	43.2	82.8	21.5	19.3
PSPNet (Res50) [10]	43.2	83.2	24.4	24.0
DeepLab [6]	59.1	89.6	39.6	328.0

4.4. Dehazing

Most dehazing models are quite large in size and need a large number of FLOPs. We compared several dehazing methods with the BPPNet's reduced version with and without LGA (refer to the ablation study in Table 7). The quantitative results presented in Table 2 show that the model with LGA performed the best in terms of SSIM. It also performed better than all the other methods in terms of PSNR. The significant reduction in computational cost of models with LGA was already established in Table 7. Therefore, it is evident that LGA also supports computationally light-weight and high-performance models for dehazing. The qualitative results of dehazing shown in Figure 6 also illustrate the superiority of image restoration achieved using the LGA-incorporated reduced BPPNet model.

Table 2. Comparison of performance of different methods with our adaptation (BPPNet + LGA) in the dehazing task on the I-HAZE [7] dataset.

MODEL	SSIM (↑)	PSNR (↑)
Input (hazy image)	0.7302	13.80
He et al. [35]	0.7516	14.43
Zhu et al. [36]	0.6065	12.24
Ren et al. [37]	0.7545	15.22
Berman et al. [38]	0.6537	14.12
Li et al. [39]	0.7323	13.98
BPPNet-reduced	0.8482	18.89
BPPNet-reduced with LGA	**0.8663**	**20.17**

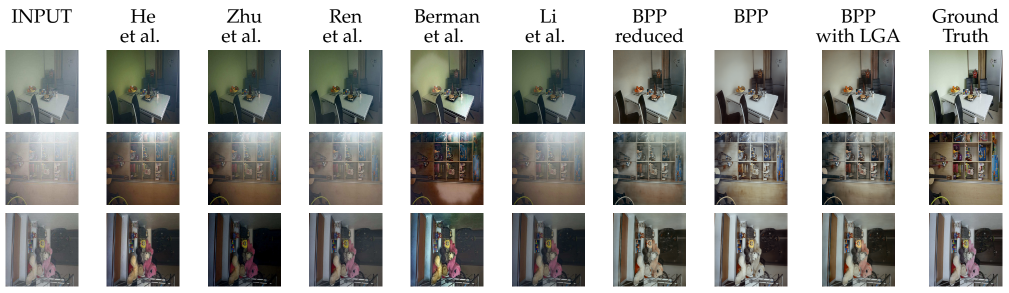


Figure 6. Example input and ground-truth samples from the I-Haze dataset and the respective results of dehazing obtained using various methods. BPP results are presented with and without the inclusion of our LGA module [35–39].

4.5. Optical Flow Estimation

We incorporated LGA in ARFlow [26], which is an unsupervised optical flow estimation method. The results of two-frame flow estimation on the MPI Sintel benchmarkare reported in Table 3 [40] for final and clean sets using the the standard end-point-error (EPE) metric. ARFlow with LGA achieves superior performance in comparison to the other unsupervised methods on all six metrics. It surpasses ARFlow, with an EPE-all of 0.387. Qualitative results are shown in Figure 7. As highlighted by the red bounding boxes, the LGA-based method produced fewer errors than ARFlow at object edges, which shows that LGA achieves a better understanding of moving and static backgrounds by identifying non-local contexts.

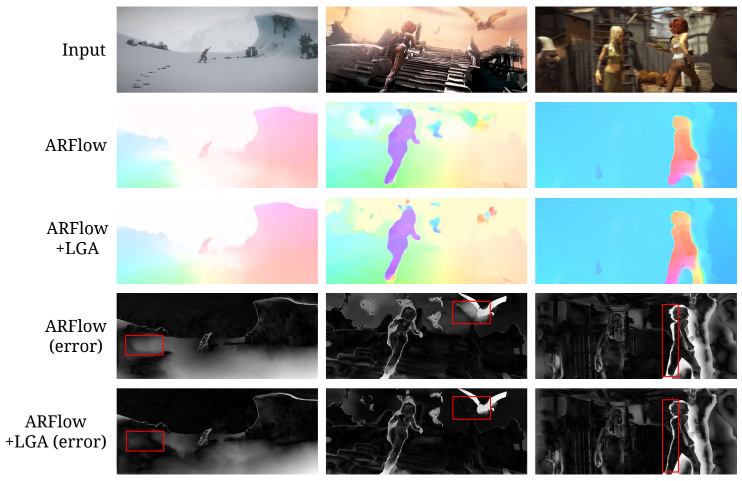


Figure 7. Results of unsupervised optical flow prediction on the MPI Sintel final-pass benchmark [40]. The first row shows the input samples, and the next two rows show the estimated flow for ARFlow [26] and LGA with ARFlow. The last two rows visualize the incurred EPE-all errors for each of the methods on the final pass. Notable differences are highlighted by red boxes.

Table 3. Optical flow estimation comparison on the MPI Sintel benchmark [40] for final and clean sets. The proposed LGA method achieved superior end-point error (EPE) scores in the two-frame unsupervised flow estimation task.

MODEL	Final (↓)			Clean (↓)		
	EPE All	**EPE Matched**	**EPE Unmatched**	**EPE All**	**EPE Matched**	**EPE Unmatched**
UFlow [41]	6.498	3.078	34.398	5.205	2.036	31.058
FastFlowNet [42]	6.080	2.942	31.692	4.886	1.789	30.182
MDFlow-fast [16]	5.994	2.770	32.283	4.733	1.673	29.718
UnsupSimFlow [43]	6.916	3.017	38.702	5.926	2.159	36.655
ARFlow [26]	5.889	2.734	31.602	4.782	1.908	28.261
LGA	**5.502**	**2.604**	**29.142**	**4.109**	**1.597**	**24.626**

4.6. Ablation Study

For the study presented in Sections 4.6.1–4.6.3, we considered the transparent object segmentation problem, and for that presented in Section 4.6.4, we considered the problem of dehazing.

4.6.1. Effect of Incorporating LGA

We considered the following encoder architectures: squeeze [34] and shuffle [15]. We evaluated the improvement in performance (in terms of mIoU %) of both models when the LGA module was incorporated. For the squeeze architecture, the mIoU values of the original model before and after incorporating the LGA module were 41.5% and 44.6%, respectively, respectively indicating an improvement of more than 3%. The shuffle architecture significantly benefited from including the LGA module, with an improved mIoU of 45.5% against the 36.9% of the original architecture. We can conclude from the Table 4 that the inclusion of LGA improved the performance of these networks. We also note that the problem of transparent object segmentation is quite challenging, with the best mIoU value of ∼41% and ∼45% for the squeeze architecture in the original form and after incorporating LGA, respectively.

Table 4. Ablation study 1: Improvement in mIoU (%) for different architectures after incorporating LGA.

MODEL	Squeeze [34]	Shuffle [15]
Original model	41.5	36.9
With LGA	44.6	44.5

4.6.2. Original SqueezeNet with LGA or CCNet

We compared the performance of the SqueezeNet architecture with CCNet and LGA is incorporated. The results are presented in Table 5. As compared to the original version, LGA, indeed, adds extra computation and storage demands but significantly fewer than CCNet. This was also discussed in Section 3.2. Moreover, it is noteworthy that the performance of SqueezeNet with LGA is better than with CCNet in terms of both mIoU and accuracy.

Table 5. Ablation study 2: Comparison of LGA (with simple and group convolutions) vs. CCNet with a SqueezeNet encoder. The original mIoU and accuracy without any LGA or CCNet module were 41.5 and 76.2, respectively.

MODEL	**Performance (↑)**		**Extra resources (↓)**	
	mIoU (%)	**Accuracy (%)**	**Parameters ($\times 10^3$)**	**FLOPS ($\times 10^6$)**
CCNet [8]	42.8	79.6	2686	5652
LGA	45.8	81.8	132	140
LGA small	44.6	79.6	17	22

4.6.3. Effects of Divergence Loss and Group Convolution

We performed a qualitative evaluation of different models with the LGA module by training them with and without the proposed divergence loss. The results presented in Table 6 show that all the models experienced a consistent improvement when the divergence loss was introduced. In addition, we note that the configuration for the best performance (LGA using simple convolution and divergence loss) requires only marginally more parameters and computations over the original SqueezeNet without LGA. Using group-transpose convolution strategy instead of simple convolution in message propagation further reduces storage and computation needs, with only a minor drop in performance. The uses of group convolution reduces the computational cost term (NC^2), as mentioned in Section 3.2, with a similar result in the case of Shuffle-Net.

Table 6. Ablation study 3: Comparison of mIOU value (%) of LGA using simple convolution (SC) or group convolution (GC) and with or without divergence loss. We also include the number of parameters and FLOPs in the first three rows to illustrate that the computational needs of incorporating LGA are marginal and even further reduced when employing group-transposed convolution.

MODEL	**Variation**	**mIoU (%)**	**Parameters ($\times 10^6$)**	**FLOPS ($\times 10^9$)**
Squeeze	No LGA	41.5	1.018	13.140
	LGA with SC and L_{div}	<u>**45.8**</u>	1.150	13.280
	LGA with GC and L_{div}	**44.6**	1.035	13.162
	LGA with SC, no L_{div}	43.6		
	LGA with GC, no L_{div}	42.3		
Shuffle	No LGA	36.9	0.395	3.42
	LGA with SC and L_{div}	<u>**44.6**</u>	0.506	3.69
	LGA with GC and L_{div}	**44.5**	0.412	3.5
	LGA with SC, no L_{div}	43.0		
	LGA with GC, no L_{div}	41.6		

4.6.4. Ablation on BPPNet for Dehazing Problem

We compared the performance and computational load of the original BPPNet [24], the reduced BPPNet (as explained in Section 4.1) without LGA, and reduced BPPNet with LGA. The results are reported in Table 7. The reduced BPPNet without LGA comprised fewer than 10% of the parameters of the original and approximately 20% the number of FLOPs. This resulted in poorer performance in terms of both the structural similarity index (SSIM) and peak-signal-to-noise-ratio (PSNR). However, the inclusion of LGA improved both the SSIM and the PSNR, with almost no extra computational load.

Table 7. Ablation study 4: Ablation on BPPNet [24] for dehazing of the I-HAZE [7] dataset.

	Original	Reduced	Reduced with LGA
SSIM	0.8994	0.8482	0.8663
PSNR	22.56	18.89	20.17
Parameters ($\times 10^6$)	8.851	0.685	0.687
FLOPS ($\times 10^9$)	348.49	87.44	87.78

4.6.5. Ablation for Number of LGA Layers

As shown in Table 8, with ablation, the mIoU increases continuously until a certain number of layers as the region of spatial context increases, then falls marginally due to an over-smoothing effect. We set a fixed kernel size 9, according to which we can infer the number of edge connections from a single node. This is not a hyperparameter in LGA but a design choice based on the neighborhood. A smaller number of kernels inhibits the propagation of information to the nearest neighbor. Therefore, LGA needs multiple layers to reach a neighbor excluded from connectivity. A larger number of kernels implies redundancy.

Table 8. Ablation for no. of LGA layers on the Trans10Kv2 (squeeze) dataset.

No. of LGA Layers	0	1	2	4	8
mIoU (%)	36.9	43.1	43.9	44.5	41.1

4.6.6. Performance and Efficiency Comparison of LGA vs. CCNet

Table 9 presents a comparison of CCNet [8] with the proposed LGA module on the Trans10Kv2 dataset [25]. The performance of LGA is better than that of CCNet in terms of both mIoU and accuracy. Even after using group convolutions in LGA, the performance is better than that of CCNet, with significant reductions in space and time complexity. Among all three models, LGA with group convolution needs orders of magnitude fewer extra parameters and less computation. For both parameters and computation, convolution channel resizing is the costliest operation, costing much more than the attention module. By using a graph-based approach, LGA saves a significant amount of resources.

Table 9. Ablation study for performance and efficiency comparison of LGA vs. CCNet [8] with a SqueezeNet encoder on the Trans10Kv2 dataset [25]. Here, LGA indicates simple convolutions, and LGA small indicates group convolutions. The original mIoU and accuracy of SqueezeNet [34] without any LGA or CCNet module were 41.5 and 76.2, respectively.

MODEL	Performance (↑)		Extra Parameters (10^3) (↓)			Extra FLOPS (10^6) (↓)			
							Attention Module		
	mIoU	Accuracy	Conv. Channel Resizing	Attention Module	Total	Conv. Channel Resizing	Information Propagation	Other Conv. Operations	Total
CCNet [8]	42.8	79.6	2359	327	2686	4832	150	670	5652
LGA	45.8	81.8	66	67	132	67	5	68	140
LGA small	44.6	79.6	8	9	17	8	5	9	22

5. Conclusions

This paper presents the novel concept of latent graph attention (LGA). Attention in CNNs is either local, non-local, or global. However, the concept of LGA progressively includes the global context in a chained fashion without significant computational complexity by propagating information across networks of locally connected graphs. Chained attention makes LGA particularly suitable for the challenging problem of transparent object segmentation. Furthermore, a novel divergence loss function is incorporated to help the LGA module perform learning in a collaborative manner with the parent architecture and enhance the performance of the original architecture with a minor additional computational load. The computational efficiency of LGA and its contribution to performance improvement and versatility were demonstrated through multiple studies. The LGA module was employed in different architectures to solve three challenging image-to-image translation problems, namely transparent object segmentation, image dehazing, and unsupervised optical flow estimation. It was shown to be an effective mechanism to learn global context and contribute this information to the parent architecture in which it is included as a module.

We hope that LGA will find use in many challenging applications and across a wide variety of architectures where it is not practical or straightforward to incorporate attention or global context mechanisms. In particular, we expect that LGA will be a powerful tool to upgrade the performance of edge devices with very limited memory and computational resources. Possible future research directions include the fusion of criss-cross attention with LGA; the reduction of multi-layered LGA into single-layered LGA; the application of LGA to object detection, video understanding, or 3D computer vision tasks such as point cloud processing and 3D scene reconstruction; the exploration of the combination of LGA with generative adversarial networks (GANs) to enhance their performance while maintaining computational efficiency; the exploration of LGA in geometric deep learning models for clustering; and the integration of LGA into multi-modal scenarios, where both images and textual data are involved, such as image captioning or visual question answering.

The source code, models, and LGA libraries will be released after notification of paper acceptance notification. Furthermore, we hope that this novel concept will incite more activity in devising other inexpensive global context mechanisms for such problems.

Supplementary Materials: The following supporting information can be downloaded at: www.mdpi.com/xxx/s1, Figure S1: LGA applications; Figure S2: More segmentation results; Figure S3: More dehazing results; Figure S4: More optical flow results; Table S1: Architecture of LGA in SqueezeNet; Table S2: Architecture of LGA in ShuffleNet; Table S3: Architecture of LGA in BPPNet; Table S4: Architecture of LGA in ARFlow.

Author Contributions: Experiment: A.S. and Y.B. Writing: A.S., Y.B., H.B., D.K.G. and D.K.P. Review: A.S., Y.B., H.B., D.K.G. and D.K.P. Funding: D.K.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a Research Council of Norway Project (nanoAI, Project ID: 325741), H2020 Project (OrganVision, Project ID: 964800), HORIZON-ERC-POC Project (Spermotile, Project ID: 101123485), and VirtualStain (UiT, Cristin Project ID: 2061348).

Data Availability Statement: All datasets used in this work are publicly available. Appropriate references for each dataset are provided in the manuscript.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Wani, M.; Batchelor, B. Edge-region-based segmentation of range images. *IEEE Trans. Pattern Anal. Mach. Intell.* **1994**, *16*, 314–319. [CrossRef]
2. He, K.; Sun, J.; Tang, X. Single Image Haze Removal Using Dark Channel Prior. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 1956–1963. [CrossRef]
3. Liu, W.; Rabinovich, A.; Berg, A. ParseNet: Looking Wider to See Better. In Proceedings of the International Conference on Learning Representations Workshops, San Juan, Puerto Rico, 2–4 May 2016.
4. Wang, S.; Lokhande, V.; Singh, M.; Kording, K.; Yarkony, J. End-to-end Training of CNN-CRF via Differentiable Dual-Decomposition. *arXiv* **2019**, arXiv:1912.02937.
5. Wang, Y.; Zhou, Q.; Liu, J.; Xiong, J.; Gao, G.; Wu, X.; Latecki, L.J. Lednet: A lightweight encoder-decoder network for real-time semantic segmentation. In Proceedings of the IEEE International Conference on Image Processing, Taipei, Taiwan, 22–25 September 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1860–1864.
6. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 801–818.

7. Ancuti, C.; Ancuti, C.O.; Timofte, R.; De Vleeschouwer, C. I-HAZE: A Dehazing Benchmark with Real Hazy and Haze-Free Indoor Images. In *Proceedings of the Advanced Concepts for Intelligent Vision Systems*; Blanc-Talon, J., Helbert, D., Philips, W., Popescu, D., Scheunders, P., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 620–631.
8. Huang, Z.; Wang, X.; Wei, Y.; Huang, L.; Shi, H.; Liu, W.; Huang, T.S. CCNet: Criss-Cross Attention for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *45*, 6896–6908. [CrossRef] [PubMed]
9. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-local neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.
10. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid scene parsing network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2881–2890.
11. Jin, L.; Xie, J.; Pan, B.; Luo, G. Generalized Phase Retrieval Model Based on Physics-Inspired Network for Holographic Metasurface. *Prog. Electromagn. Res.* **2023**, *178*, 103–110.
12. Zhang, X.; Xu, H.; Mo, H.; Tan, J.; Yang, C.; Wang, L.; Ren, W. Dcnas: Densely connected neural architecture search for semantic image segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 13956–13967.
13. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
14. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 116–131.
15. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv* **2017**, arXiv:1707.01083.
16. Kong, L.; Yang, J. MDFlow: Unsupervised Optical Flow Learning by Reliable Mutual Knowledge Distillation. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *33*, 677–688. [CrossRef]
17. Lu, Y.; Chen, Y.; Zhao, D.; Chen, J. Graph-FCN for image semantic segmentation. In *Proceedings of the International Symposium on Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 97–105.
18. Ye, X.b.; Guan, Q.; Luo, W.; Fang, L.; Lai, Z.R.; Wang, J. Molecular substructure graph attention network for molecular property identification in drug discovery. *Pattern Recognit.* **2022**, *128*, 108659. [CrossRef]
19. Zhou, H.; Yang, Y.; Luo, T.; Zhang, J.; Li, S. A unified deep sparse graph attention network for scene graph generation. *Pattern Recognit.* **2022**, *123*, 108367. [CrossRef]
20. Sun, T.; Zhang, G.; Yang, W.; Xue, J.H.; Wang, G. Trosd: A new rgb-d dataset for transparent and reflective object segmentation in practice. *IEEE Trans. Circuits Syst. Video Technol.* **2023**, *33*, 5721–5733. [CrossRef]
21. Yu, R.; Ren, W.; Zhao, M.; Wang, J.; Wu, D.; Xie, Y. Transparent objects segmentation based on polarization imaging and deep learning. *Opt. Commun.* **2024**, *555*, 130246. [CrossRef]
22. Banerjee, S.; Hati, A.; Chaudhuri, S.; Velmurugan, R. Image co-segmentation using graph convolution neural network. In Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing, Hyderabad, India, 18–22 December 2018; pp. 1–9.
23. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
24. Singh, A.; Bhave, A.; Prasad, D.K. Single image dehazing for a variety of haze scenarios using back projected pyramid network. In Proceedings of the European Conference on Computer Vision Workshop, Glasgow, UK, 23–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 166–181.
25. Xie, E.; Wang, W.; Wang, W.; Sun, P.; Xu, H.; Liang, D.; Luo, P. Segmenting transparent object in the wild with transformer. *arXiv* **2021**, arXiv:2101.08461.
26. Liu, L.; Zhang, J.; He, R.; Liu, Y.; Wang, Y.; Tai, Y.; Luo, D.; Wang, C.; Li, J.; Huang, F. Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 6489–6498.
27. Liu, M.; Yin, H. Feature pyramid encoding network for real-time semantic segmentation. *arXiv* **2019**, arXiv:1909.08599.
28. Mehta, S.; Rastegari, M.; Shapiro, L.; Hajishirzi, H. Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 9190–9200.
29. Paszke, A.; Chaurasia, A.; Kim, S.; Culurciello, E. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv* **2016**, arXiv:1606.02147.
30. Li, G.; Yun, I.; Kim, J.; Kim, J. Dabnet: Depth-wise asymmetric bottleneck for real-time semantic segmentation. In Proceedings of the British Machine Vision Conference, Cardiff, UK, 9–12 September 2019.
31. Zhao, H.; Qi, X.; Shen, X.; Shi, J.; Jia, J. Icnet for real-time semantic segmentation on high-resolution images. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 405–420.
32. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.
33. Wang, Y.; Zhou, Q.; Xiong, J.; Wu, X.; Jin, X. Esnet: An efficient symmetric network for real-time semantic segmentation. In Proceedings of the Chinese Conference on Pattern Recognition and Computer Vision, Long Beach, CA, USA, 15–20 June 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 41–52.

34. Iandola, F.N. SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters. 2016. Available online: https://github.com/forresti/SqueezeNet (accessed on 20 October 2024).
35. He, K.; Sun, J.; Tang, X. Single Image Haze Removal Using Dark Channel Prior. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *33*, 2341–2353. [CrossRef] [PubMed]
36. Zhu, Q.; Mai, J.; Shao, L. A fast single image haze removal algorithm using color attenuation prior. *IEEE Trans. Image Process.* **2015**, *24*, 3522–3533. [PubMed]
37. Ren, W.; Liu, S.; Zhang, H.; Pan, J.; Cao, X.; Yang, M.H. Single image dehazing via multi-scale convolutional neural networks. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 154–169.
38. Berman, D.; Treibitz, T.; Avidan, S. Non-local image dehazing. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 1674–1682.
39. Li, B.; Peng, X.; Wang, Z.; Xu, J.; Feng, D. AOD-Net: All-In-One Dehazing Network. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.
40. Butler, D.J.; Wulff, J.; Stanley, G.B.; Black, M.J. A naturalistic open source movie for optical flow evaluation. In Proceedings of the European Conference on Computer Vision, Florence, Italy, 7–13 October 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 611–625.
41. Jonschkowski, R.; Stone, A.; Barron, J.T.; Gordon, A.; Konolige, K.; Angelova, A. What matters in unsupervised optical flow. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 557–572.
42. Kong, L.; Shen, C.; Yang, J. Fastflownet: A lightweight network for fast optical flow estimation. In Proceedings of the IEEE International Conference on Robotics and Automation, Xi'an, China, 30 May–5 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 10310–10316.
43. Im, W.; Kim, T.K.; Yoon, S.E. Unsupervised learning of optical flow with deep feature similarity. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 172–188.