Supplementary Materials

Supplementary for: 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

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

1. Introduction

This supplementary provides time and space complexity analysis of CCNet and the proposed LGA attention modules in Sec. 2. Extended analysis of the proposed LGA contrastive loss function in Sec. 3. The architectural details of different methods is provided in Sec. 4. An in-depth explanation of incorporation of LGA module in different architectures is provided in Sec. 5. Additional qualitative results are presented in Sec. 6.

2. Complexity analysis

2.1. Time complexity

This section provides the derivation of time complexity of LGA network in comparison to the best light weight criss-cross attention network (CCNet [1]). The time complexity of the attention networks can be defined as

$$\mathcal{O}\big(ConvTransformation\big) + \mathcal{O}\big(LGA/CCNet\ attention\big) \tag{S1}$$

$\mathcal{O}\big(ConvTransformation\big)$ is the time required to do channel transformation using convolution layer. For example, changing channel size to reduce overall complexity. For LGA, it can be written as

$$\begin{aligned}\mathcal{O}\big(ConvTransformation\big) = a_1 \times N \times C_{in} \times C_{out} \times k \times k \\ + b_1 \times N \times C_{out} \times C_{out} \times k \times k\end{aligned} \tag{S2}$$

Here, $k = 1$ is the kernel size, a_1 and b_1 are number of times such operations are applied. $N = n \times n$ is the feature shape. Equation (S2) can be simplified as

$$\begin{aligned}\mathcal{O}\big(ConvTransformation\big) = a_1 \times N \times C_{in} \times C_{out} \\ + b_1 \times N \times C_{out} \times C_{out}\end{aligned} \tag{S3}$$

Here, $C_{in} > C_{out}$, $a_1 = 1$ and $b_1 > a_1$. Using these conditions, we can approximate that $a_1 \times C_{in} \approx b_1 \times C_{out}$.

$$\therefore \mathcal{O}\big(ConvTransformation\big) = 2 \times b \times N \times C_{out} \times C_{out} \tag{S4}$$

$$\mathcal{O}\big(LGA\ attention\big) = c_1 \times N \times C_{out} \tag{S5}$$

$$\begin{aligned}\mathcal{O}(LGA) &= b_1 \times N \times C_{out} \times C_{out} \\ &+ c_1 \times N \times C_{out}\end{aligned} \tag{S6}$$

as b_1 and c_1 are small, we can rewrite Equation (S6) as

$$\begin{aligned}\mathcal{O}(LGA) &= N \times C_{out} \times C_{out} \\ &+ N \times C_{out}\end{aligned} \tag{S7}$$

Now, for CCNet we have

$$\begin{aligned}\mathcal{O}(ConvTransformation) &= a_2 \times N \times C_{in} \times C_{out} \times k \times k \\ &+ b_2 \times N \times C_{in} \times C_{in} \times k \times k\end{aligned} \tag{S8}$$

$$\mathcal{O}(CCNet\ attention) = c_2 \times N^{3/2} \times C_{out} \tag{S9}$$

$$\begin{aligned}\mathcal{O}(CCNet) &= N \times C_{in} \times C_{out} \\ &+ N \times C_{in} \times C_{in} \\ &+ N^{3/2} \times C_{out}\end{aligned} \tag{S10}$$

considering a_2, b_2, c_2, k as small. As $C_{in} > C_{out}$, we can rewrite Equation (S10) as

$$\begin{aligned}\mathcal{O}(CCNet) &= N \times C_{in} \times C_{in} \\ &+ N^{3/2} \times C_{out}\end{aligned} \tag{S11}$$

By comparing time complexities of CCNet and LGA in Equation (S7) and Equation (S11), we learn that CCNet has $C_{in} > C_{out}$ and $N^{3/2}$ as compared to N in LGA. Hence, it has higher FLOPs than LGA.

We found that it is possible to reduce the complexity of CCNet by adding a reducing Conv. layer. This makes the CCNet time complexity

$$\begin{aligned}\mathcal{O}(CCNet) &= N \times C_{out} \times C_{out} \\ &+ N^{3/2} \times C_{out}\end{aligned} \tag{S12}$$

Convolutional transformation term, $N \times C_{out} \times C_{out}$, can be reduced further by adding group convolutions. It modifies the time complexities as

$$\begin{aligned}\mathcal{O}(CCNet) &= \frac{N \times C_{out} \times C_{out}}{G} + N^{3/2} \times C_{out} \\ \mathcal{O}(LGA) &= \frac{N \times C_{out} \times C_{out}}{G} + N \times C_{out}\end{aligned} \tag{S13}$$

where G denotes the number of groups. If G is large, $\frac{N \times C_{out} \times C_{out}}{G}$ becomes small. Then the bottleneck remains the attention term which cannot be reduced.

$$\begin{aligned}\mathcal{O}(CCNet) &= N^{3/2} \times C_{out} \\ \mathcal{O}(LGA) &= N \times C_{out}\end{aligned} \tag{S14}$$

Hence, in general the time complexity of attention of CCNet is $\sqrt{N}$ times higher than LGA attention.

2.2. Space complexity

The space complexity of the attention based networks can be defined as Equation (S1). For LGA, we have

$$\begin{aligned}\mathcal{O}(ConvTransformation) &= a_1 \times N \times C_{in} \\ &+ b_1 \times N \times C_{out}\end{aligned} \tag{S15}$$

As $b_1 > a_1$ and $C_{in} > C_{out}$, $a_1 = 1$ implies that $a_1 \times C_{in} \approx b_1 \times C_{out}$. Using this, Equation (S15) can be rewritten as

$$\begin{aligned} \mathcal{O}\big(ConvTransformation\big) &= a_1 \times N \times C_{in} \\ Or,\ \ \mathcal{O}\big(ConvTransformation\big) &= \times N \times C_{in} \end{aligned} \tag{S16}$$

As the adjacency matrix is sparse, so the $\mathcal{O}\big(LGA\ attention\big) = N$, and LGA space complexity becomes

$$\mathcal{O}\big(LGA\big) = \times N \times C_{in} + N \tag{S17}$$

Now, for CCNet we have

$$\begin{aligned} \mathcal{O}\big(ConvTransformation\big) &= a_1 \times N \times C_{in} \\ &\quad + b_1 \times N \times C_{out} \\ &\approx a_1 \times N \times C_{in} \\ &= N \times C_{in} \end{aligned} \tag{S18}$$

Please note $N \times C_{in}$ cannot be reduced further because it is the input feature size which is dependent on the used encoder and not on the proposed LGA module. Also, if $a_1 \times N \times C_{in} \neq b1 \times N \times C_{out}$ and $b_1 \times N \times C_{out} < a_1 \times N \times C_{in}$ with $(a_1 = 1)$ then as $N \times C_{in}$ is from encoder, $N \times C_{in}$ is the bottleneck and it cannot be reduced. Hence, $N \times C_{in}$ is the lower bound on $\mathcal{O}\big(ConvTransformation\big)$, hence, we have

$$\begin{aligned} \mathcal{O}\big(CCNet\ attention\big) &= N^{3/2} \\ Or,\ \ \mathcal{O}\big(CCNet\big) &= N \times C_{in} + N^{3/2} \end{aligned} \tag{S19}$$

Comparing the space complexity of LGA with CCNet, we have

$$\begin{aligned} \mathcal{O}\big(LGA\big) &= N \times C_{in} + N \\ \mathcal{O}\big(CCNet\big) &= N \times C_{in} + N^{3/2} \end{aligned} \tag{S20}$$

$N \times C_{in}$ term is same for LGA and CCNet. If N is large then $N \times C_{in} + N \approx N$ and the space complexities become

$$\begin{aligned} \mathcal{O}\big(LGA\big) &= N \\ \mathcal{O}\big(CCNet\big) &= N^{3/2} \end{aligned} \tag{S21}$$

3. LGA contrastive loss analysis

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

$$\mathcal{L}_{\text{LGA}} = E_{P_i,P_j}\left(\mathcal{C}_{ij}\log\left(\frac{V_{ij}^2}{U_{ij}}+1\right) + \bar{\mathcal{C}}_{ij}\log\left(\frac{U_{ij}}{V_{ij}^2}+1\right)\right) \tag{S22}$$

$V_{ij} = D(F_i^{\text{out}}, F_j^{\text{out}})$ and $U_{ij} = D(F_i^{\text{in}}, F_j^{\text{in}})$, F^{in} and F^{out} denote the input and output feature maps for LGA module, and $D(\cdot)$ is the divergence function such as KL-divergence or mean square error. The two variables $\mathcal{C}$ and $\bar{\mathcal{C}}$ are boolean, and their values depend on the similarity between the neighboring nodes in the graph. For example, for nodes i and j, we look at the GT labels of patches containing them. Further, 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 know that having only V_{ij} will not be sufficient to enforce learning in LGA. To fulfill this, U_{ij} is incorporated in the proposed contrastive loss.

We note that U_{ij} is determined by the input to LGA module and does not update as LGA learns. LGA essentially learns through tweaking the value of V_{ij}, which is determined by the output of LGA. We further note that larger diverse value D_{ij} indicates larger difference in the distributions of the 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 learnt to be maximized thereby setting the two output distributions apart.

Case: similar distribution. Let us first consider the case of two nodes with similar distribution, such that only the first term (i.e. term with $\mathcal{C}_{ij}$) contributes to $\mathcal{L}_{\text{LGA}}$. In this condition, we want divergence between nodes to be lower. If U_{ij} is large and V_{ij} is small, then the net value of $\mathcal{L}_{\text{LGA}}$ is low. This indicates that LGA is able to identify that the nodes have similar distribution, even if their input features do not indicate so. On the other hand if U_{ij} is small and V_{ij} is large, then $\mathcal{L}_{\text{LGA}}$ is large. This drives LGA layers to update themselves for reducing V_{ij}. This situation actually indicates that the network preceding LGA is able to identify that the nodes have similar distributions, but LGA should learn to propagate this property ahead. If both U_{ij} and V_{ij} are high, the second power of V_{ij} in the numerator helps LGA focus on learning to reduce V_{ij}, albeit at a slower pace than the previous situation (i.e. small U_{ij}). This is because in this situation both the LGA and the network preceding it have the scope of improvement, and the slow pace of update of LGA is an indirect nudge to the preceding or succeeding networks to improve their performance. In the ideal situation, when both U_{ij} and V_{ij} are low, $\mathcal{L}_{\text{LGA}}$ remains low and LGA is effective in propagating the correctly learnt attributes of the preceding network.

Case: different distribution. Now, let us consider the second case when the nodes i and j do not have the same distribution. $\mathcal{L}_{\text{LGA}}$ then corresponds to only the second term (with $\bar{\mathcal{C}}_{ij}$). In this condition, we want divergence between nodes to be high. If U_{ij} is large but V_{ij} is small, then the value of $\mathcal{L}_{\text{LGA}}$ is large. Then LGA updates itself to increase the value of V_{ij}. This also indicates that the preceding network is effective in separating the F^{in} of the two nodes and the LGA needs to learn to propagate this attribute of the preceding network efficiently. On the other hand if U_{ij} is small but V_{ij} is large, the loss function has a small value due to the second power of V_{ij} in the denominator, which indicates that LGA is able to learn that the nodes have different distributions despite the poor performance of the preceding network. In the situation that both U_{ij} and V_{ij} are low, $\mathcal{L}_{\text{LGA}}$ is high due to the second power of V_{ij} in the denominator. Then, LGA learns to increase the value of V_{ij} in a relatively slow and linear manner. This is so because this situation indicates that there is a scope of improvement for both LGA and the network preceding it. This slow update of LGA can provide a nudge to the preceding or succeeding networks to improve their learning as well.

Case: ideal distribution. Lastly, in the ideal situation where both U_{ij} and V_{ij} are large, the loss function is low owing to the second order of V_{ij} in the denominator. This indicates that LGA has learned to propagate the good performance of the preceding network in separating the two nodes in an effective manner.

4. Architectural details

Table S1, S2, S3, S4 present the architectural details of LGA module when applied to the three sets of problems on SqueezeNet [2], ShuffleNet [3], BPPNet [4] and ARFlow [5]. The tables describe operations at each step, associated kernels, inputs and outputs.

Table S1. Architecture details of incorporation of LGA into SqueezeNet [2] for segmentation. Starting from the input F^{in}, till the LGA output after concatenation.

	Size of tensor	Input size	Conv. parameter (input & output channel)	Output size	Groups
F^{in}	32x32x512				
(1x1) 2d conv		32x32x512	512,128	32x32x128	8
9 (1x1) 2d conv		32x32x128	128,1		1
Adj. matrix	1024x1024				
(1x1) 1d conv		1024x128	128,128	1024x128	8
(1x1) 1d conv		1024x128	128,128	1024x128	8
(1x1) 1d conv		1024x128	128,128	1024x128	8
(1x1) 1d conv		1024x128	128,128	1024x128	8
Concat		32x32x512 & 32x32x128		32x32x640	
Output	32x32x640				

Table S2. Architecture details of incorporation of LGA into ShuffleNet [3] for segmentation. Starting from the input F^{in}, till the LGA output after concatenation.

	Size of tensor	Input size	Conv. parameter (input & output channel)	Output size	Groups
F^{in}	32x32x232				
(1x1) 2d conv		32x32x232	232,136	32x32x136	8
9 (1x1) 2d conv		32x32x128	128,1		1
Adj. matrix	1024x1024				
(1x1) 1d conv		1024x136	136,136	1024x136	8
(1x1) 1d conv		1024x136	136,136	1024x136	8
(1x1) 1d conv		1024x136	136,136	1024x136	8
(1x1) 1d conv		1024x136	136,136	1024x136	8
Concat		32x32x232 & 32x32x136		32x32x368	
Output	32x32x368				

Table S3. Architecture details of incorporation of LGA into BPPNet [4] for dehazing. Starting from the input F^{in}, till the LGA output after concatenation.

	Size of tensor	Input size	Conv. parameter (input & output channel)	Output size	Groups
F^{in}	512x512x16				
resize		512x512x16		32x32x16	
9 (1x1) 2d conv		32x32x16	16,1		1
Adj. matrix	1024x1024				
(1x1) 1d conv		16x16	16,16	16x16	1
(1x1) 1d conv		16x16	16,16	16x16	1
(1x1) 1d conv		16x16	16,16	16x16	1
(1x1) 1d conv		16x16	16,16	16x16	1
resize		32x32x16		512x512x16	
Concat		512x512x16 & 512x512x16		512x512x32	
Output	512x512x32				

Table S4. Architecture details of incorporation of LGA into ARFlow [5] for optical flow estimation. Starting from the input F^{in}, till the LGA output after concatenation.

	Size of tensor	Input size	Conv. parameter (input & output channel)	Output size	Groups
F^{in}	24x28x96				
resize		24x28x96		32x32x96	
9 (1x1) 2d conv		32x32x96	96,1		1
Adj. matrix	1024x1024				
(1x1) 1d conv		96x96	96,96	96x96	8
(1x1) 1d conv		96x96	96,96	96x96	8
(1x1) 1d conv		96x96	96,96	96x96	8
(1x1) 1d conv		96x96	96,96	96x96	8
resize		32x32x96		24x28x96	
Concat		24x28x96 & 24x28x96		24x28x192	
(1x1) 2d conv		24x28x192	192,96	24x28x96	8
Output	24x28x96				

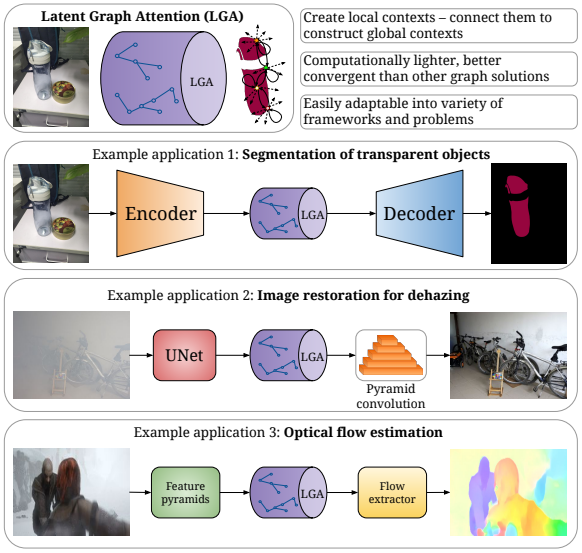

Figure S1. We present a novel concept of Latent Graph Attention (LGA), which can be easily integrated in wide variety of applications and architectures. Three challenging and open problems are considered as example applications of LGA in this article.

5. Incorporation of LGA in different architectures

Our LGA module takes a feature map F^{in} of size $H \times W \times C$ as input, where H, W, C are height, width and number of channels respectively. It then returns an output feature F^{out} of the same size i.e. $H \times W \times C$ in which each feature of size $1 \times 1 \times C$ also contains the information of other spatially closed features. Later, F^{in} and F^{out} are concatenated as the final output of LGA, F^{cat}.

Example schematics of incorporating LGA into existing networks is shown in Figure S1. We pass an image X to the model, the feature map generated by initial blocks of the model (or the network preceding the LGA) is passed as the input feature F^{in} to LGA and the concatenated output of LGA, i.e. F^{cat}, is passed into the later blocks of the model (or the network succeeding LGA). For example, for a segmentation model consisting of an encoder-decoder architecture, the feature map generated by encoder can be considered as F^{in} and the F^{cat} is passed to the decoder block. Incorporation of LGA module is not limited to encoder-decoder kind of networks only. The LGA module can be inserted between any two blocks in an architecture as suitable. In this paper, we have presented three example applications, *viz.* segmentation of transparent objects, image dehazing and optical flow estimation. In the following sections, we present the details of how we incorporated LGA into the architectures solving these three vision problems.

5.1. LGA adaptation for semantic segmentation

Our network for semantic segmentation is shown in Figure S1's top panel. Our network consists of three blocks, namely an encoder, an LGA module and an up-sample block. We removed some of the last layers of the encoder and did adequate padding (where needed) so that, given the input image of shape $H \times W \times 3$, the output is a feature map of size $H' \times W' \times C$. This feature map is then passed into LGA and the output is concatenated with the input of LGA. Then this concatenated feature map is passed to up-sample block. This block upscales the input feature map to generate a mask of shape $H \times W \times C_1$, where C_1 is the total number of classes. We have used group transposed convolution in the up-sample block for efficient up-sampling. For segmentation, we adapted LGA into SqueezeNet [2] and ShuffleNet [3].

5.2. LGA incorporation in BPPNet [4] for image restoration task

The network for image restoration task is given in Figure S1's middle panel. We adapted BPPNet [4] by incorporating LGA with single UNet instead recurrent UNet by original BPPNet. We have incorporated our LGA module in between a UNet and pyramid convolutional network. We did not used 2d conv layer to change the input channel size because the size was already low (16). We added an additional 2D convolution layer of size 1×1 after LGA to make the channel size of the concatenated output feature equal to the channel size of LGA's input feature map. We included this layer because the pyramid block does maximum number of operations and it is proportional to the input size. Hence, maintaining the same channel size gives better computational efficiency. All the convolutional layers of UNet and pyramid convolutional block were replaced by group convolutions in order to further improve the overall computational efficiency.

5.3. LGA incorporation in ARFlow [5] for optical flow estimation

The network for optical flow estimation task is given in Figure S1's lower panel. We used ARFlow [5] as our base model for unsupervised optical flow estimation. ARFlow extracts the feature pyramid of the frames and obtains a flow map via the correlation between feature maps and uses the convolution layer for refining it. We believe that if we can extract much better features, which have information of its neighbors, we will be able to improve the flow map. For this reason, we added our LGA module after output feature extractor module. In our experiment, we only applied LGA at a single feature map of spatial size (24, 28). This is because the smaller maps don't have much spatial information because of their smaller size, which becomes a challenge for LGA to propagate information and enhance the map. For higher maps, because the information of smaller scale features are used for high scale flow generation via using small scale flow map as prior, applying LGA at latter stages does not make much difference and it also increases the overall time complexity. When applying LGA feature of size (24,28) we did not apply 2d Conv layer for channel resizing because the input size was optimal (96). We also, applied a 2d Conv layer after concatenating the input and output map, so as to make the input channel size and final output channel size equal.

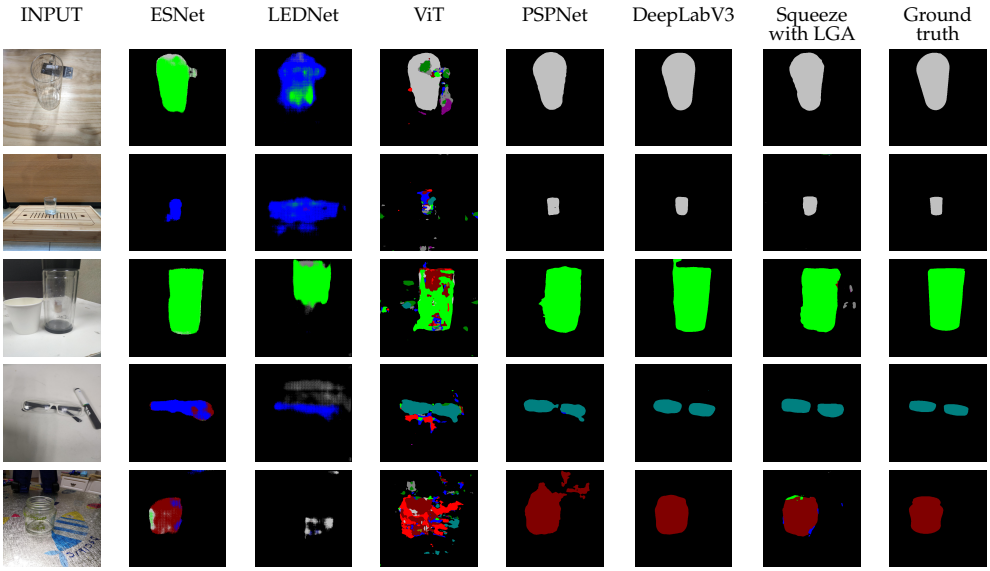

Figure S2. Examples from segmentation results of different methods on Trans10Kv2 dataset [6] are presented. Eleven classes of objects are color coded.

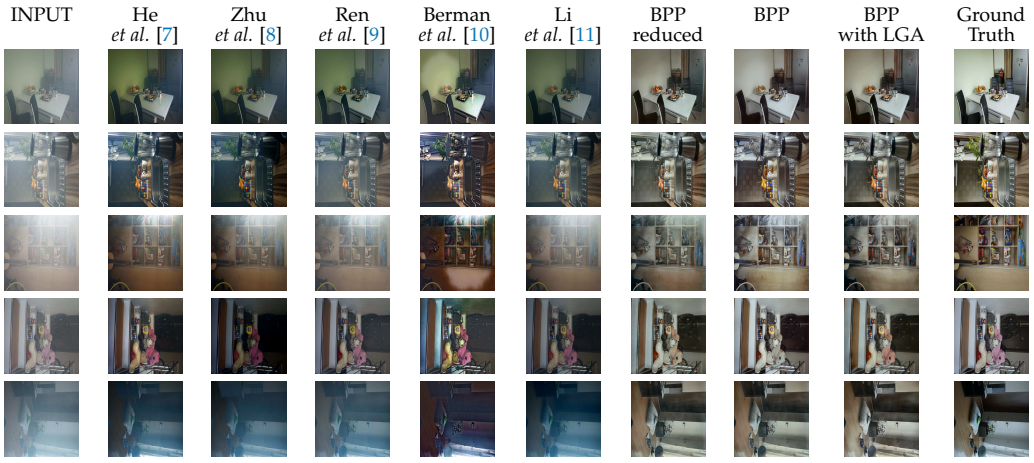

Figure S3. Example input and ground-truth samples from the I-Haze dataset and the respective results of dehazing obtained using various methods. BPP results are presented with as well as withou the inclusion of our LGA module.



Figure S4. Results of unsupervised optical flow prediction on the MPI Sintel final pass benchmark [12]. First row shows the input samples, then next two rows show estimated flow for ARFlow [5] and LGA with ARFlow. The last two rows visualize the incurred EPE-all errors for the methods on the final pass. Notable differences are highlighted in red box.

6. Additional results

This supplementary provides some more qualitative results on the application of LGA for segmentation, dehazing and optical flow estimation. The results are presented in Figure S2, S3, S4 respectively. In Figure S2, we can see that LGA results are comparable to its heavier counterparts DeepLabV3 [13] and PSPNet [14], much better than ViT [15] and LEDNet [16]. In Figure S3, we see that for third sample, LGA removes haze better than BPP model. For fourth sample, LGA preserves better colors than BPP model. In Figure S4, LGA gives less errors than the baseline ARFlow [5] as highlighted by the red boxes.

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).

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. Huang, Z.; Wang, X.; Wei, Y.; Huang, L.; Shi, H.; Liu, W.; Huang, T.S. CCNet: Criss-Cross Attention for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2020**, pp. 1–1. https://doi.org/10.1109/TPAMI.2020.3007032.
2. Iandola, F.N. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters. https://github.com/forresti/SqueezeNet, 2016.
3. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices.(2017). *arXiv preprint arXiv:1707.01083* **2017**.

4. 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. Springer, 2020, pp. 166–181.
5. 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, 2020, pp. 6489–6498.
6. Xie, E.; Wang, W.; Wang, W.; Sun, P.; Xu, H.; Liang, D.; Luo, P. Segmenting transparent object in the wild with transformer. *arXiv preprint arXiv:2101.08461* **2021**.
7. He, K.; Sun, J.; Tang, X. Single image haze removal using dark channel prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2010**, *33*, 2341–2353.
8. Zhu, Q.; Mai, J.; Shao, L. A fast single image haze removal algorithm using color attenuation prior. *IEEE Transactions on Image Processing* **2015**, *24*, 3522–3533.
9. 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, 2016, pp. 154–169.
10. Berman, D.; Avidan, S.; et al. Non-local image dehazing. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1674–1682.
11. 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, 2017.
12. 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. Springer, 2012, pp. 611–625.
13. 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, 2018, pp. 801–818.
14. 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, 2017, pp. 2881–2890.
15. 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 preprint arXiv:2010.11929* **2020**.
16. 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. IEEE, 2019, pp. 1860–1864.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.