– Supplementary Material –
Towards Unified Image Deblurring using a Mixture-of-Experts Decoder

Daniel Feijoo [1†], Paula Garrido-Mellado [1], Jaesung Rim [2], Álvaro García [1], Marcos V. Conde [1†]
[1] Cidaut AI, Valladolid, Spain
[2] POSTECH, Pohang, South Korea
† Corresponding Authors {danfei,marcos.conde@cidaut.es}
https://github.com/cidautai/DeMoE

1. Similarity Weights Analysis

Correlation Analysis During the preliminary step of this research, a study of weight similarities was developed. Given each of the datasets that form part of AIO-Blur, a baseline NAFNet [3] f_Θ has been trained on each of them. Then, pairs of training weights Θ_1, Θ_2 of the same architecture f_Θ in different datasets are compared using two similarity methods: the Pearson correlation and CKA [11]. In Figures 1, 2 and 3 the results of this study are shown.

To calculate the correlations, we classify the different blocks of the NAFNet architecture into four types of layers: pixel-wise 1×1 convolutions, 3×3 convolutions, layer normalization, and simplified channel attention (SCA) layers. Each of these layers is composed of a set of C filters that resemble the channel size of the features introduced to the layer. For each pair of weights Θ_1, Θ_2, we compute the correlation of their layer values per filter. Then, the mean of these filter correlations is calculated to define the final correlation value of the layer. The correlation for each filter is given by

$$r = \frac{\sum_i (\Theta_{1,i} - \bar{\Theta}_1)(\Theta_{2,i} - \bar{\Theta}_2)}{\sqrt{\sum_i (\Theta_{1,i} - \bar{\Theta}_1)^2 \sum (\Theta_{2,i} - \bar{\Theta}_2)^2}}, \tag{1}$$

where $\bar{\Theta}_j$ and $\Theta_{j,i}$ are the average filter values and an element value of a filter for Θ_j task-specific weights, respectively. Finally, we calculate the mean of the correlations of the filters in order to get the mean correlation for each block. The mean is calculated following

$$R = \frac{\sum_{i=1}^{C} r_i}{C}, \tag{2}$$

where C is the number of filters in the layer and r_i is the correlation of the ith filter.

Figure 1 gives us a general idea on how these weights are related and that in general, LOLBlur [26] has a lower correlation with the other datasets, which can be related not only to blur degradations, but also to low-light ones. We also observe that convolution blocks with a kernel size of 3 exhibit a strong correlation across all dataset pairs. This is likely because these blocks uniquely incorporate information from neighboring pixels in their computations, making them particularly well-suited for capturing the local structure involved in convolutional operations such as blurring. The results in Figure 1 suggest that the impact of these blocks on blur restoration is consistent across different cases, regardless of the specific type of blur degradation.

We confirmed that these results were consistent with our interpretation of the Pearson correlation values by computing the weight correlations between the models trained on each deblurring task and those trained on a low-light restoration task. The selected training dataset for the low-light task weights was LOLv2-real [24]. The results of this study are shown in Figure 2, where it can be seen that neither of the layers considered shows any correlation with the AIO-Blur dataset weights. This supports the idea that weights Θ_1, Θ_2 trained in different blur degradations share similarities.

Centered Kernel Alignment Following the same strategy to calculate correlations per filter, we computed the Centered Kernel Alignment similarity index by [11]. Since Pearson's coefficient does not capture non-linear relationships, we also performed a CKA analysis to ensure the results obtained in the previous study. The authors introduced two variants of the CKA by changing the kernel function: a linear kernel or an RBF (radial basis function) kernel. We chose to use the second one in order to capture non-linear relationships. The values range from 0 to 1, with higher values indicating greater similarity between the weights.

Let $X \in \mathbb{R}^{n_1 \times p}$ and $Y \in \mathbb{R}^{n_2 \times p}$ denote two matrices of activations of p neurons for n_1 and n_2 examples, respectively; $K_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$ and $L_{ij} = l(\mathbf{y_i}, \mathbf{y_j})$, with k and l two different kernels. The CKA is calculated following

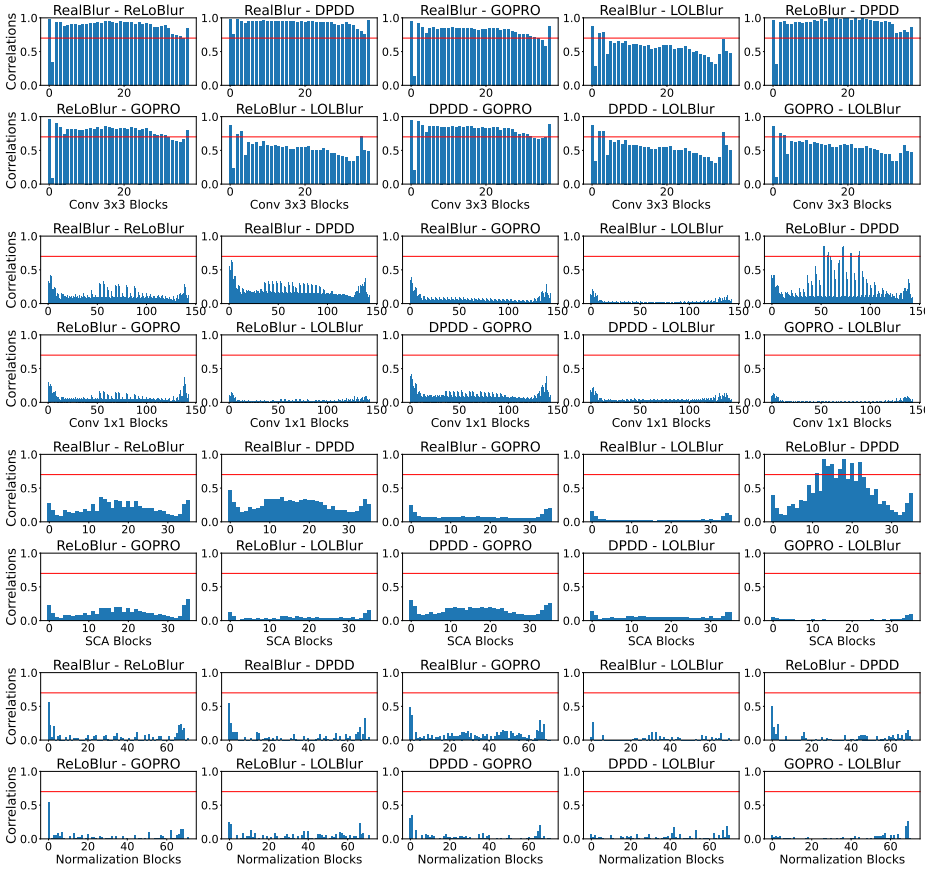

Figure 1. Correlations calculated for different types of neural layers in NAFNet between task-specific weights. The layer considered for each diagram is stated in x axis. Correlation values over 0.7 state a high-correlation [2].

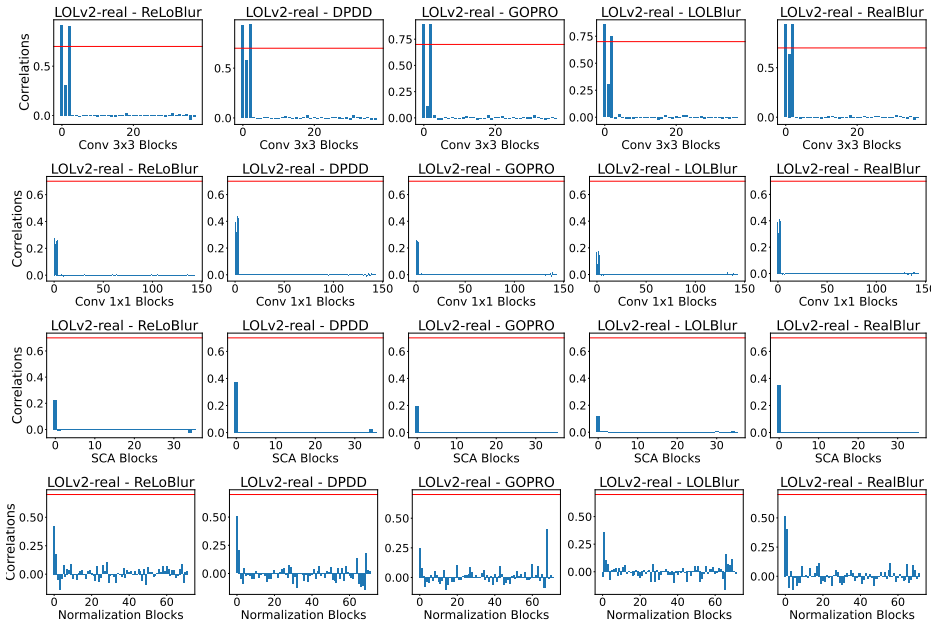

Figure 2. Correlations of the weights compared to a different task.

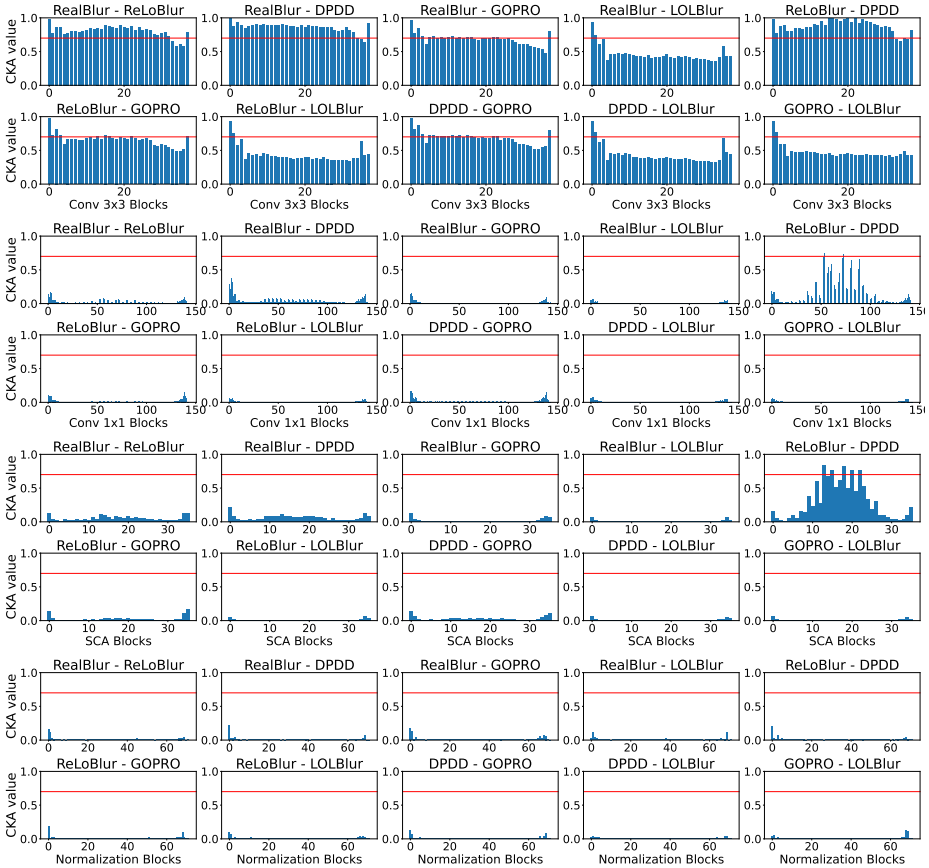

Figure 3. CKA similarity index calculated for different types of neural layers in NAFNet between task-specific weights. Values near 1 represent higher similarity values.

$$CKA(K, L) = \frac{HSIC(K, L)}{\sqrt{HSIC(K, K) * HSIC(L, L)}}, \quad (3)$$

where $HSIC(K, L)$ represents the Hilbert-Schmidt Independence Criterion empirical estimator [6]. This estimator is calculated by

$$HSIC(K, L) = \frac{1}{(n-1)^2} tr(KHLH), \quad (4)$$

where H is the centering matrix $H_n = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T$. We observed that the CKA results are similar to the ones obtained using correlation, as shown in Figure 3. This reinforces the statement on Θ_1 and Θ_2 weights similarities for different deblurring degradations.

Conclusions From this analysis, we can conclude that the operations needed for the deblurring task are similar for different types of blur, based on:

- There is a high correlation between the layer weights in the neighbor-based operations: the kernel size 3 convolutions. These layers have a similar behavior for any type of blur.
- The weighs of the layers that do not operate on neighbors show no significant correlation, thus, there is no similar behavior between these layers.
- CKA analysis shows great similarity indexes between the weights of the layers needed for the deblurring task, which means that they not only have a similar behavior, but they also learn similar representations.

2. More details of AIO-Blur

In this section, we provide more details of the datasets used to construct AIO-Blur.

GoPro [17] is a *synthetic* dataset for motion blur. Blurred images are generated by averaging consecutive frames from high-speed videos. Then, the center frame of each sequence is used as the ground-truth sharp image. The dataset consists of 2,103 training blur-sharp pairs and 1,111 test pairs.

RealBlur [19] is a *real-world* dataset for camera motion blur. RealBlur was collected using a dual-camera system, where one camera captures a sharp image with a short exposure time, and the other captures a blurred image with a long exposure time. Using the dual-camera system, RealBlur

provides real blurred images caused by camera motion and the corresponding ground-truth sharp images. We used the *RealBlur-J* subset, which provides JPEG RGB images. The dataset contains 3,758 training pairs and 980 testing pairs.

ReLoBlur [14] is a *real-world* dataset for local motion blur. It was introduced for the task of local motion deblurring, with an emphasis on blurred moving objects against static backgrounds. The dataset was collected by capturing moving objects in front of static backgrounds, using a dual-camera system. It consists of 2,010 training blur-sharp pairs and 395 test pairs.

LOLBlur [26] is a *synthetic* dataset for low-light motion blur. Blur typically occurs in low-light environments, such as dimly lit indoor scenes or nighttime, where captured images often suffer not only from motion blur but also from low-light degradation. To address this issue, the dataset was introduced for the joint task of low-light enhancement and deblurring. Blurred images are generated by averaging consecutive frames, and low-light degradation is simulated using EC-Zero-DCE (a variant of Zero-DCE [7]). LOLBlur consists of 10,200 low-light blurry training pairs and 1,800 testing pairs.

DPDD [1] is a *real-world* dataset for defocus blur. The dataset was collected on static scenes using a single camera mounted on a tripod. A blurred image was captured with a small aperture, and the ground-truth sharp image was captured immediately afterward using a large aperture. DPDD consists of 500 defocus-sharp pairs, split into training (70%), validation (15%) and testing (15%) sets. For AIO-Blur, only the training and testing sets are used.

In addition, we constructed another deblurring dataset, namely **AIO-Blur-OOD**, to evaluate the robustness of methods on out-of-distribution (OOD) data. The dataset is composed of the following open-source datasets:

RealDOF [13] is a *real-world* dataset for defocus blur. RealDOF was collected using a dual-camera setup, where one camera captures all-in-focus images with a small aperture, and the other captures defocused images with a large aperture. The dataset is test-only dataset consisting 50 scenes.

RSBlur [20] is a *real-world* dataset for motion blur. The dataset is composed of a total of 13,358 real blurred images of 697 scenes. This pairs are split into 8,878 training, 1,120 validation, and 3,360 test sets. The dataset was collected using a dual-camera system including both camera shake and object motion blur.

Real-LOLBlur [26] is a *real-world* dataset for low-light and motion blur. The dataset contains 872 real-world low-light blurry images without ground-truth sharp images. For Real-LOLBlur, we use non-reference metrics for evaluation.

For training with AIO-Blur, we equally scaled the size of the datasets to make sure that the network learned enough features for each dataset. The largest dataset is LOLBlur, with 10,200 pairs of images. As this is a very large compared with the other datasets, we worked on a reduction of this dataset to have a number of samples similar to the second largest dataset, RealBlur. The other datasets were upsampled to also have a similar number of samples like RealBlur. The final distribution of the different datasets in AIO-Blur can be seen in Table 1.

Dataset	Original Sample	Final Sample
RealBlur [19]	3758	3758
ReLoBlur [14]	2010	4018
LOLBlur [26]	10200	4200
GoPro [17]	2103	4206
DPDD [1]	350	3850
AIO-Blur	**18421**	**20032**

Table 1. Final distribution of images in the training set of AIO-Blur. The testing sets were not modified.

Subsample of LOLBlur To subsample this dataset, we calculate the mean-squared error (MSE) of all the images in the train split of the dataset. When checking the histogram of MSE values of all the images we found a large shift into small MSE values, which in most cases can be related to images that are easier to restore. To have a diverse set of images in this dataset, we draw the same histogram considering only four bins. Both of these histograms can be seen in Figure 4. Based on the low population of the last bin in the four-bin histogram, we did not considered images in this subset. Of the remaining 3 bins, we randomly picked 1,400 image pairs from each. The whole process led to the final 4200 pairs of training images of LOLBlur dataset in AIO-Blur dataset.

3. Implementation Details

Our implementation is based on PyTorch. We train DeMoE using the training set of AIO-Blur. We randomly cropped 384×384 patches and applied vertical and horizontal flip augmentations. The batch size is set to 32 and we used 4 H100 GPUs for training. The optimizer used is AdamW [16], setting $\beta_1 = 0.9$ and $\beta_2 = 0.9$, with an initial learning rate $1e^{-3}$ and updated to a minimum value of $1e^{-7}$ by the cosine annealing strategy [15]. The training is divided into two steps: pretraining the baseline and finetuning the experts and decoder layers. Each of the steps has been trained for 400 epochs, for a total time of ≈ 6 days.

4. Experts Specialization

To show the specialization of each of the experts in the DeMoE network, in Figure 5 the correlations between the different experts are presented. Apart from the LayerNorm

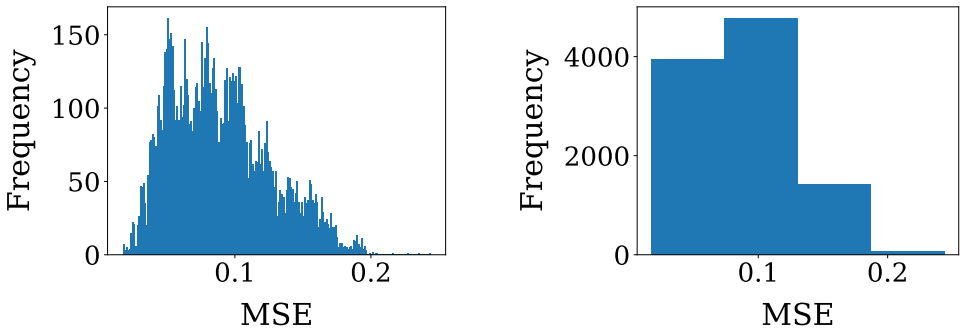

Figure 4. (Left) Distribution of MSE in LOLBlur. (Right) MSE-based classification for the downscaling of the dataset. Note that due its low population the last bin is not relevant in the whole distribution, thus the images that belong to this set are not considered in the final training set.

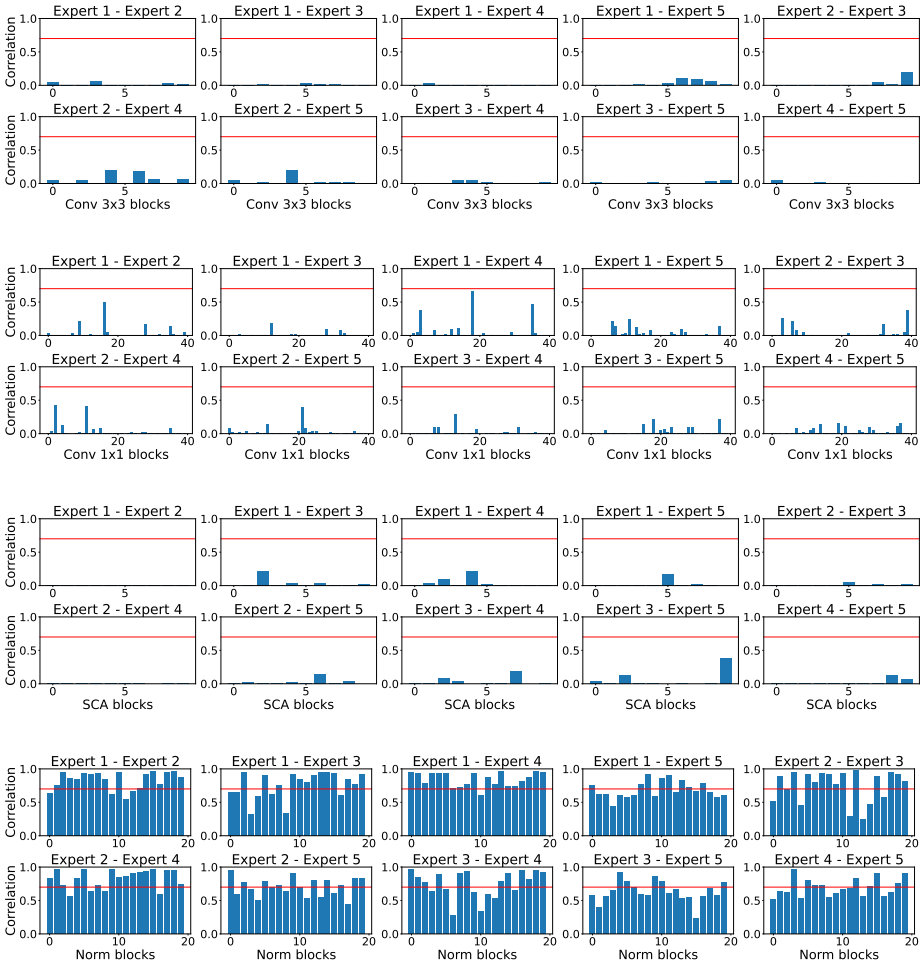

Figure 5. Correlations of the different experts in MoEBlock for the DeMoE network proposed. The lack of correlation in every layer, apart from normalization, suggest that the experts are specialized in the different deblurring tasks.

layers, the remaining layers of the five experts do not show significant correlation. Two ideas can be extracted from this analysis:

1. The normalization layers are very similar, so the features

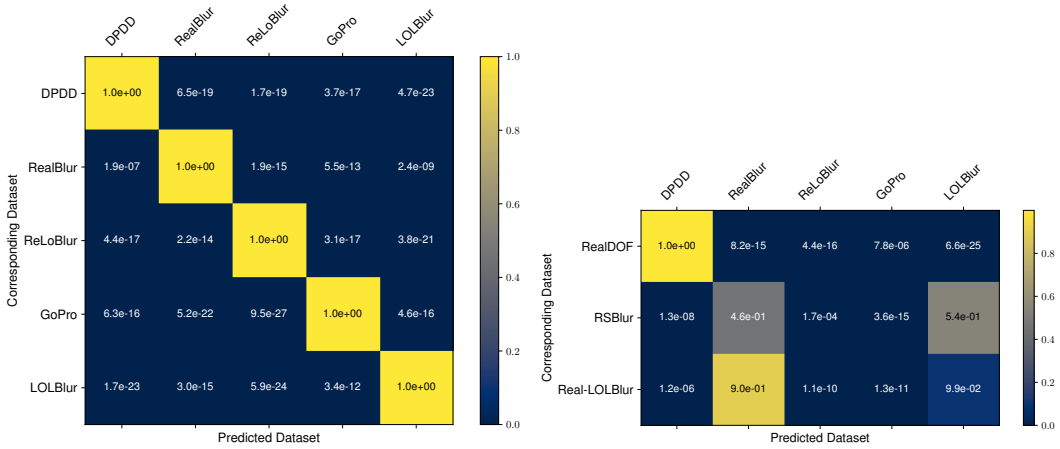

Figure 6. Average output tensor of the router for the AIO-Blur (Left) and AIO-Blur-OOD (Right) Datasets.

Table 2. (Left) Architectural ablations performed in DeMoE. (Right) Ablations performed in the pretrained baseline of DeMoE and number of blocks of DeMoE.

Ablation	PSNR↑	SSIM↑	LPIPS↓
Addition residual to MoEBlock output	29.10	0.876	0.209
Attention connection to MoEBlock output	28.83	0.870	0.218
Larger channel embedding (32 → 64)	**29.23**	**0.880**	**0.203**
DeMoE$_{k=1}$	<u>29.19</u>	<u>0.877</u>	<u>0.208</u>

Ablation	Params (M)↓	MACs (G)↓	Runtime (ms)↓	PSNR↑	SSIM↑	LPIPS↓
Baseline w/o classification loss	10.08	11.02	8	28.84	0.868	0.225
Baseline w/ original architecture	22.07	20.37	14.8	<u>29.19</u>	<u>0.878</u>	0.207
Baseline w/ classification loss	**10.08**	**11.02**	**8**	29.12	0.873	0.215
More middle NAFBlocks	38.67	15.11	15.6	29.18	0.878	<u>0.206</u>
More MoEBlocks	32.54	13.18	17.4	**29.26**	**0.879**	**0.203**

that are introduced in each of the experts are in a similar space.

2. Other layers do not share correlations because of the specialization of the different experts to each task. Thus, the MoEBlocks work as expected.

5. Classification error of the router

In Section 4.2, it has been pointed out how the router classification of the AIO-Blur-OOD is the one that produces the low results of DeMoE when manual expert selection is not used. To ilustrate this bad performance of the router, in Figure 6 we represent the average output tensor of the router for the different datasets on AIO-Blur and AIO-Blur-OOD. It can be seen that the expert usage for the AIO-Blur datasets is the expected one, while in the OOD case the router fails to classify the RSBlur dataset and Real-LOLBlur one.

6. Further ablation study

Architecture Ablation In Table 2 (Left) we present the results of using different fusions of the generated features of each expert. The addition residual can be formulated as

$$\hat{\mathbf{h}} = \mathbf{h} + \sum_{i=0}^{N} \mathbf{w}_i \cdot \mathrm{e}_i(\mathbf{h}) \quad , \tag{5}$$

where $\mathbf{h}$ and $\hat{\mathbf{h}}$ are the input and output features, respectively. $\mathbf{w}_i$ is the corresponding weight of the expert e_i. Following the same formulation, the attention connection presented in Table 2 can be stated as

$$\hat{\mathbf{h}} = \mathbf{h} \cdot \sum_{i=0}^{N} \mathbf{w}_i \cdot \mathrm{e}_i(\mathbf{h}) \quad . \tag{6}$$

We also trained a model with a larger embedding depth, increasing the channel count from 32 to 64. While this modification yielded slightly higher performance, the improvement was insufficient to justify the substantial increase in computational cost (79.71 M parameters, 42.77 G MACs, 22.4 ms runtime). Consequently, this model was not considered for further qualitative or out-of-distribution (OOD) analysis. Furthermore, we experimented with adding more NAFBlocks at the deepest encoder level and doubling the number of MoEBlocks in each decoder level. As shown in Table 2 (Right), these architectural expansions did not yield a more favorable trade-off. Collectively, these ablation studies

confirm that the proposed DeMoE architecture achieves the optimal balance between performance and efficiency.

Pretrained baseline ablations Table 2 (Right) also presents ablations concerning the pretraining of the NAFNet baseline. Our proposed architecture has two NAFBlocks per each encoder layer and three for the middle block and decoder layers. This is different from the original design of NAFNet [3] for image deblurring, which typically employs one NAFBlock per encoder-decoder step, except for the final encoder step that uses 28 blocks. We evaluated the pretrained NAFNet using this original architecture (with three NAFBlocks per decoder layer) and include the results in Table 2. However, due to the substantial increase in operations, parameters, and runtime, we selected our more efficient baseline architecture.

In addition, we studied the impact of the classification loss on the pretrained baseline. The quantitative results can also be seen in Table 2, where it is shown that the inclusion of the classification loss notably increases the performance of the network. In Figure 7, we present some qualitative results on the use of the original baseline and classification loss. Given all the images in the test sets of the AIO-Blur dataset, we apply CLIP [18] and t-SNE [23] in cascade to the encoder features of different weights: random initialized weights, AIO-Blur weights without classification loss, AIO-Blur weights with classification loss, and original NAFNet architecture with classification loss. As expected, trained weights produce distinct clusters while random weights do not. Contrary to our expectation, the point clouds for all trained models appear remarkably similar. The key insight is that the similarity between the cluster plots of the original and our proposed architecture suggests that a heavily parameterized encoder stage is unnecessary. Instead, parameters are more effectively allocated to the task-specific restoration stage in the decoder. This finding reinforces the advantage of our proposed architecture over the original design.

TLC Ablation Table 3 presents the results of applying the test-time local converter (TLC [4]) to NAFNet blocks during inference. This method adapts the behavior of specific network layers at test time and is designed to improve performance on images larger than those used in training. While TLC leads to notable metric improvements in some restoration tasks, our study on its application to DeMoE considered three scenarios: (1) no TLC, (2) TLC applied to all layers, and (3) TLC applied partially only to experts where it yielded significant gains. The results in Table 3 indicate that TLC adversely affects the LOLBlur and ReLoBlur experts, leading to performance degradation in low-light and local-motion deblurring tasks. Since TLC did not provide a general improvement and even hampered performance on the low-light task, we excluded it from the final version of DeMoE.

7. More results in OOD

Quantitative results In addition to the quantitative results discussed in OOD in the main article, we present extended results in this dataset in Tables 4 and 5.

Qualitative results We show the performance of DeMoe compared to the other general deblur methods using the OOD datasets in Figure 8. A state-of-the-art method is also included in each of the qualitative samples.

8. Broader impacts of DeMoE

As a preliminary exploration of task-related restoration, DeMoE has the following impacts:

- Applications in many fields: Compared to existing methods, DeMoE offers higher robustness to different scenarios where blurry artifacts can be generated. It can be widely applied to any computer vision task with images that can potentially suffer blur degradation, such as autonomous driving or commercial photography.
- Negative social impacts: To the best of the authors knowledge, there are no negative social impacts.

9. LLM disclosure

During the preparation of this manuscript, the authors utilized large language models (LLMs) exclusively for proofreading and grammatical refinement. All scientific content, analysis, and intellectual contributions remain entirely human-authored.

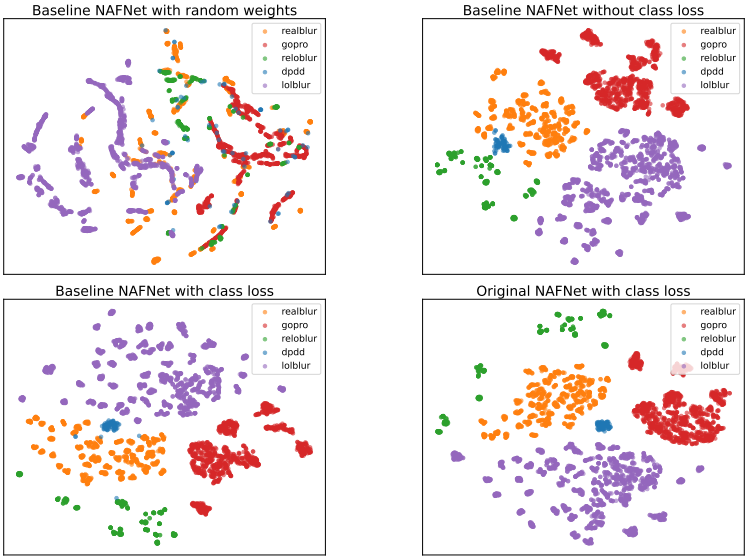

Figure 7. t-SNE representations of different baseline models.

Table 3. Comparison of the results of applying TLC during inference for NAFNet and DeMoE. It can be seen that some of the datasets metrics are improved when using TLC while others suffer a decrease. The average result suggests that it is better to not use TLC.

Method	RealBlur	ReLoBlur	DPDD	LOLBlur	GoPro	Average
NAFNet	**29.18** / 0.883 / 0.203	**34.56** / **0.926** / 0.228	25.60 / 0.795 / 0.266	26.69 / 0.871 / 0.188	29.56 / 0.890 / 0.191	29.12 / 0.873 / 0.215
NAFNet w/ TLC	29.15 / 0.884 / 0.199	34.33 / 0.925 / **0.227**	**25.77** / **0.806** / 0.258	25.80 / 0.873 / 0.185	29.95 / 0.897 / 0.184	29.00 / 0.877 / 0.211
DeMoE$_{k=1}$ w/ TLC	29.00 / **0.885** / **0.195**	34.23 / 0.923 / 0.232	25.70 / 0.806 / **0.250**	26.13 / 0.880 / 0.172	**30.42** / **0.906** / **0.170**	29.10 / 0.880 / 0.204
DeMoE$_{k=1}$ w/ partial TLC	29.00 / 0.885 / 0.195	34.26 / 0.924 / 0.232	25.70 / 0.806 / 0.250	26.32 / **0.881** / **0.172**	30.42 / 0.906 / 0.170	29.14 / **0.880** / **0.204**
DeMoE$_{k=1}$	28.96 / 0.884 / 0.198	34.52 / 0.925 / 0.232	25.56 / 0.797 / 0.258	**26.84** / 0.878 / 0.175	30.06 / 0.900 / **0.176**	**29.19** / 0.877 / 0.208

Table 4. Quantitative evaluations on various datasets. Results with † were extracted from [13](RealDOF) and [5](RSBlur). Results with * were trained in the AIO-Blur dataset.

Method	PSNR↑	SSIM↑	LPIPS↓
Restormer* [25]	22.98	0.684	0.471
FFTFormer* [10]	23.36	0.697	0.440
SFHFormer* [8]	24.20	0.736	0.428
NAFNet* [3]	24.64	**0.762**	0.382
JNB† [21]	22.36	0.635	0.601
EBDB† [9]	22.38	0.638	0.594
DMENet† [12]	22.41	0.639	0.597
DPDNet† [1]	22.67	0.666	0.420
IFAN† [13]	**24.71**	0.748	**0.306**
DeMoE$_{k=1}$	24.59	0.758	0.377

RealDOF results

Method	PSNR↑	SSIM↑
Restormer* [25]	14.04	0.556
SFHFormer* [8]	13.54	0.553
NAFNet* [3]	15.00	0.576
FFTFormer* [10]	15.60	0.589
FFTFormer† [10]	29.70	0.787
BANet+† [22]	30.24	0.809
MLWNet-B† [5]	**30.91**	**0.818**
DeMoE$_{k=1}$	19.23	0.640
DeMoE$_{k=realblur}$	27.53	0.763

RSBlur results

Table 5. Robustness study of OOD night blurry images using Real-LOLBlur dataset [26]. Methods with * were trained in AIO-Blur dataset, but methods with † were extracted from [26].

	RUAS → MIMO†	MIMO → Zero-DCE†	FFTFormer*	NAFNet*	Restormer*	SFHFormer*	LEDNet†	DarkIR	**DeMoE**$_{k=1}$	**DeMoE**$_{k=lolblur}$
MUSIQ↑	34.39	28.36	30.18	32.02	27.71	29.58	39.11	**48.36**	31.34	38.91
NRQM↑	3.322	3.697	5.780	5.515	5.169	5.320	5.643	4.983	5.47	**6.113**
NIQE↓	6.812	6.892	6.469	6.865	7.381	6.806	**4.764**	4.998	7.56	5.82

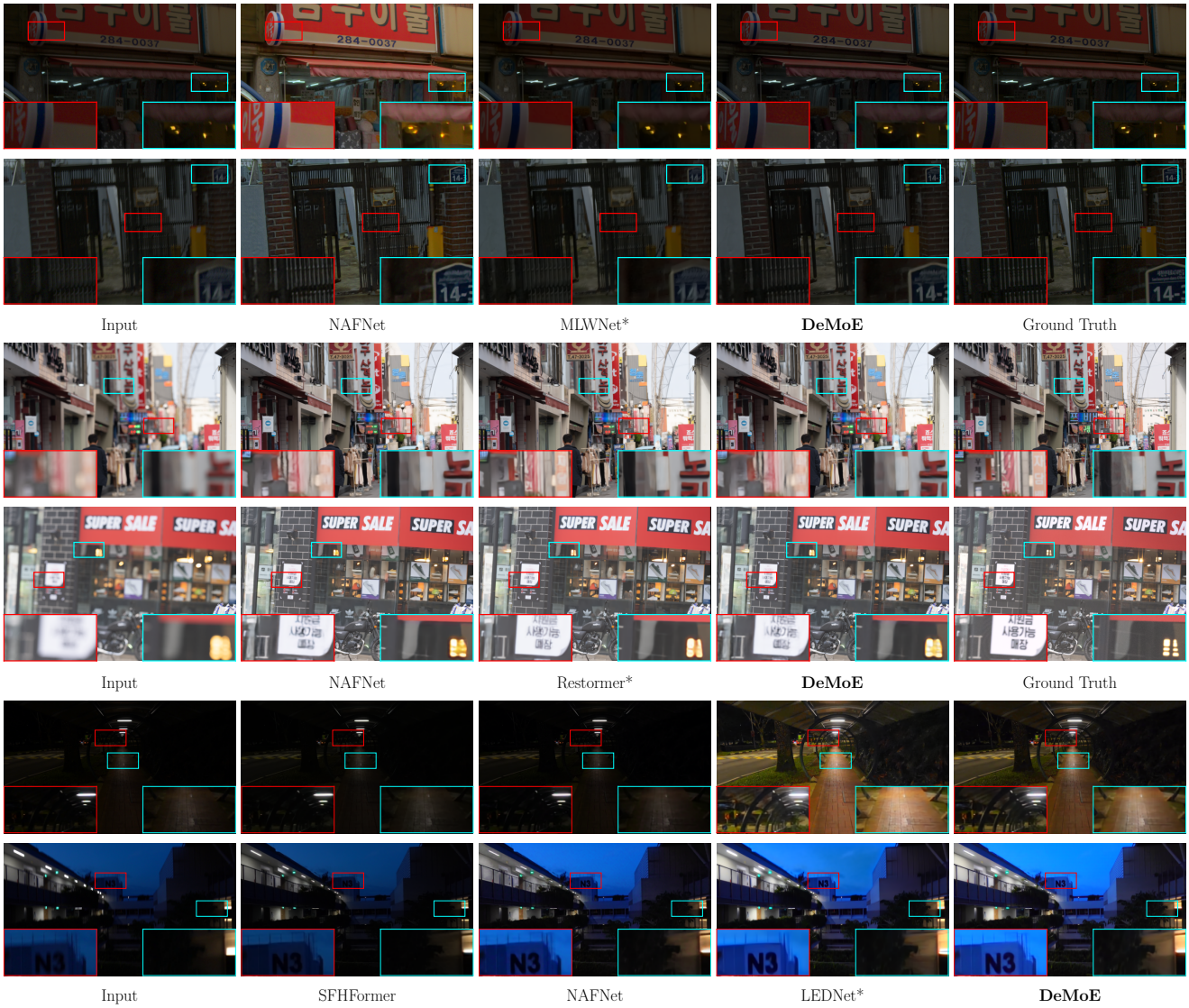

Figure 8. Qualitative comparison of the general deblur methods in OOD datasets. Methods with * are task-specific ones. The first two rows are images from RSBlur [20], the next two rows are from RealDOF [13], and the final rows are from Real-LOLBlur [26]. Zoom in for better view.

References

[1] Abdullah Abuolaim and Michael S. Brown. Defocus deblurring using dual-pixel data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 4, 8

[2] A. G. Asuero, A. Sayago, and A. G. González. The correlation coefficient: An overview, 2006. 2

[3] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. *arXiv preprint arXiv:2204.04676*, 2022. 1, 7, 8

[4] Xiaojie Chu, Liangyu Chen, Chengpeng Chen, and Xin Lu. Improving image restoration by revisiting global information aggregation. In *European Conference on Computer Vision*, pages 53–71. Springer, 2022. 7

[5] Xin Gao, Tianheng Qiu, Xinyu Zhang, Hanlin Bai, Kang Liu, Xuan Huang, Hu Wei, Guoying Zhang, and Huaping Liu. Efficient multi-scale network with learnable discrete wavelet transform for blind motion deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2733–2742, 2024. 8

[6] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005. 3

[7] Chunle Guo Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runmin Cong. Zero-reference deep curve estimation for low-light image enhancement. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1780–1789, 2020. 4

[8] Xingyu Jiang, Xiuhui Zhang, Ning Gao, and Yue Deng. When fast fourier transform meets transformer for image restoration. In *European Conference on Computer Vision*, pages 381–402. Springer, 2024. 8

[9] Ali Karaali and Changick Jung. Edge-based defocus blur estimation with adaptive scale selection. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 2617–2621, 2017. 8

[10] Lingshun Kong, Jiangxin Dong, Jianjun Ge, Mingqiang Li, and Jinshan Pan. Efficient frequency domain-based transformers for high-quality image deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5886–5895, 2023. 8

[11] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR, 2019. 1

[12] Junyong Lee, Sungkil Lee, Sunghyun Cho, and Seungyong Lee. Deep defocus map estimation using domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12222–12230, 2019. 8

[13] Junyong Lee, Hyeongseok Son, Jaesung Rim, Sunghyun Cho, and Seungyong Lee. Iterative filter adaptive network for single image defocus deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2034–2043, 2021. 4, 8, 9

[14] Haoying Li, Ziran Zhang, Tingting Jiang, Peng Luo, Huajun Feng, and Zhihai Xu. Real-world deep local motion deblurring. In *proceedings of the AAAI conference on artificial intelligence*, pages 1314–1322, 2023. 4

[15] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 4

[16] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 4

[17] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3, 4

[18] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 7

[19] Jaesung Rim, Haeyun Lee, Jucheol Won, and Sunghyun Cho. Real-world blur dataset for learning and benchmarking deblurring algorithms. In *Computer vision–ECCV 2020: 16th European conference, glasgow, UK, August 23–28, 2020, proceedings, part XXV 16*, pages 184–201. Springer, 2020. 3, 4

[20] Jaesung Rim, Geonung Kim, Jungeon Kim, Junyong Lee, Seungyong Lee, and Sunghyun Cho. Realistic blur synthesis for learning image deblurring. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 4, 9

[21] Jianping Shi, Li Xu, and Jiaya Jia. Just noticeable blur detection via sparse representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 657–665, 2015. 8

[22] Fu-Jen Tsai, Yan-Tsung Peng, Chung-Chi Tsai, Yen-Yu Lin, and Chia-Wen Lin. Banet: a blur-aware attention network for dynamic scene deblurring. *IEEE Transactions on Image Processing*, 31:6789–6799, 2022. 8

[23] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 7

[24] Wenhan Yang, Wenjing Wang, Haofeng Huang, Shiqi Wang, and Jiaying Liu. Sparse gradient regularized deep retinex network for robust low-light image enhancement. *IEEE Transactions on Image Processing*, 30:2072–2086, 2021. 1

[25] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 8

[26] Shangchen Zhou, Chongyi Li, and Chen Change Loy. LEDNet: Joint low-light enhancement and deblurring in the dark. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 1, 4, 8, 9