A Efficient Classification & Denoising: Additional results

The joint models in Section 3 are constructed using a separate denoiser and classifier. We describe the baseline models and several methods to construct the reduced versions.

Overview of the models used in the main paper. UNet-S: $\{d = 4, b = 8, c = 2, m = 1.5\}$, which is also called Reduced UNet. UNet-M: $\{d = 4, b = 16, c = 2, m = 2\}$. UNet: $\{d = 5, b = 64, c = 2, m = 2\}$, which is also called Baseline UNet. MB2.5-M: the classifier described in Section A.1 with an MBConv (expansion rate = 2.5) as second convolutional layer.

A.1 Efficient Classification

Experimental setup. Our baseline classifier (Conv-L) consists of two convolutional, one global max pooling, and a linear layer. Each convolutional layer also has a group normalization [44], max pooling, and ReLU activation function.

To construct the reduced version, we use two methods similar to previous works [33,37]. In the first method, we replace the second convolutional layer with an MBConv layer. Three expansion rates are used $\{1, 2.5, 4\}$: (i) rate 1 is the lowest possible value, (ii) rate 4 matches the number of FLOPs of the baseline, and (iii) rate 2.5 is in the middle of those two. The second reduction method is to lower the number of filters in the baseline, also called the model width. Using these techniques, models with three different FLOP sizes are constructed, {S, M, L}. We use the following naming scheme, Conv-x and MBe-x, where x represents the FLOP size and e is the expansion rate of the MBConv.

The models are trained using Cross Entropy loss. We report the accuracy averaged over all 11 noise levels.

Exp. 1: Conv vs MBConv comparison. According to [33], the MBConv layer should be more efficient than a normal convolutional layer. Therefore, when comparing the two operators in our network, we expect the version with an MBConv layer to need fewer FLOPs for the same accuracy. In Table 7, the MB models with expansion rates 2.5 (MB2.5-M) and 4 (MB4-L) classify better than the Conv-L model with fewer FLOPs. However, with an expansion rate of 1 (MB1-S), the accuracy drops 7% compared to Conv-L. Therefore, [33]'s theory also holds for the noisy classifier, but only for the higher expansion rates.

Exp. 2: MBConv width & expansion rate scaling. Since MBConv layers can be used to improve efficiency, we question how to further reduce the MB model's FLOP size. We compare two options: (i) reducing the expansion rate and (ii) scaling the width of the network. We take MB4-L as the starting model, as this is our best and largest model.

From the MB models with size S in Table 7, MB1-S performs the worst. It only has a reduced expansion rate from 4 to 1. MB4-S, which is obtained by scaling the width of MB-L, increases classification performance by only 0.4%. However, when slightly reducing MB4-L's width and expansion rate, we derive MB2.5-S, which reaches 58.4% accuracy, significantly outperforming both other S-sized MB models. So the combination of the two methods is most effective.

Table 7: Classification baseline and reduced models, designed for three different FLOP targets: {S, M, L}, to compare scaling methods: expansion rate and model width. Each section of rows is used by the experiments from Sec. A.1 defined in the *Exp.* column. MB models scale down more efficiently than normal Conv models.

Exp.	Model	Size	Exp. rate	FLOPs (K) ↓	Lat. (ms) ↓	Acc (%) ↑
1-3	Conv-L	L	-	447	0.336	63.2
	MB1-S	S	1	177	0.300	56.2
	MB2.5-M	M	2.5	350	0.384	64.1
	MB4-L	L	4	424	0.468	64.9
2-3	MB2.5-S	S	2.5	178	0.390	58.4
	MB4-S	S	4	188	0.403	56.6
3	Conv-S	S	-	163	0.281	55.4
	Conv-M	M	-	345	0.317	61.4

Exp. 3: Conv width scaling. In this experiment, we compare the width scaling of the Conv-L model. Table 7 shows that all S-sized MB models outperform Conv-S, MB2.5-S even by 3.0%. MB2.5-M also outperforms Conv-M, by 2.7%. Therefore, scaling is more efficient for the MB models than the Conv models when optimizing for FLOPs.

Conclusion. The MBConv layer can replace the convolutional layers. We find that compared to the Conv models, the MB models also scale down more efficiently by first reducing the expansion rate, possibly followed by a width reduction. Scaling effectively reduces the number of FLOPs.

MB2.5-M has the second-best accuracy with low FLOPs and a latency close to the baseline, Conv-L. Therefore, MB2.5-M is used as the reduced classifier. We also use MB2.5-M as the new baseline classifier as it outperforms the old baseline, Conv-L, in FLOPs and accuracy.

It is important to note that the reduction in FLOPs instantiated by using MBConvs, does not translate to a latency reduction in these experiments. This issue is discussed previously in [38]. Since the target of these experiments is FLOPs and the latency increase is manageable, we place minimal emphasis on the latency.

A.2 Efficient Denoising

Experimental setup. For denoising, the baseline and reduced version are constructed by performing a hyperparameter study on UNet similar to [25]. Figure 2 shows the UNet architecture along with its hyperparameters to tune. We explore the parameters one at a time, starting with the number of base features maps b, then the UNet depth d, the feature map multiplier m, and the number of convolutional blocks per layer c. In the original UNet: $\{b = 64,\ d = 5,\ m = 2,\ c = 2\}$. Altering these hyperparameters can greatly reduce the model size. Similar to the classification experiments, we also study the ability of the MBConv operator

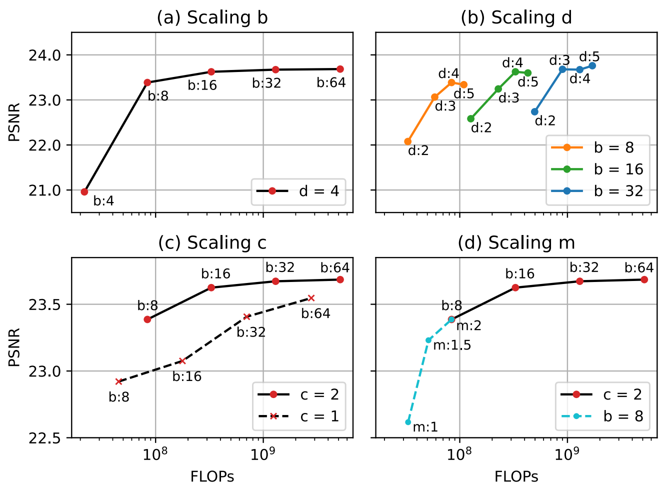

Fig. 9: UNet hyperparameter (Figure 2) scaling experiments. Shows how altering a specific hyper-parameter influences denoising performance and FLOPs. We only show PSNR results of σ=0.8, as the other results show the same trend. We find that b and m scale down efficiently, d and c do not.

to increase efficiency in the denoiser. The models are trained using Charbonnier loss [4].

Exp. 1: The base feature map width *b*. In this experiment, we aim to find the relevant range of b. Since b is multiplied at every level, the number of feature maps throughout all layers depends on it, which makes it a powerful hyper-parameter. We use $d = 4$ and the other hyper-parameters as in the original UNet, then we test $b \in \{4, 8, 16, 32, 64\}$. The trend in Figure 9.a shows that the performance and FLOPs increase with b. We observe that the trend is significantly disrupted by $b = 4$. Conversely, the performance difference between $b = 32$ and $b = 64$ is small, but the network size quadrupled. Therefore in further experiments, we focus on $b \in \{8, 16, 32\}$.

Exp. 2: The UNet depth *d*. Given the robustness of b, we are interested in how reducing d compares in terms of efficiency. Figure 9.b displays the performance of the architectures with the selected $b \in \{8, 16, 32\}$ testing $d \in \{2, 3, 4, 5\}$. We observe that reducing d causes a drop in denoising performance, whereas b retains performance better, also in Figure 9.a. Therefore b scales down more efficiently. The models with $d = 3$ or 4 denoise most efficient. Especially for the smaller models, $d = 4$ performs well.

Exp. 3: The number of conv blocks per layer *c*. Does reducing c further increase efficiency? To test this, we take the best-performing settings, $d = 4$ and $b \in \{8, 16, 32, 64\}$, and compare $c = 1$ and $c = 2$. Figure 9.c shows that the model with $c = 2$ outperforms $c = 1$. Therefore reducing c does not benefit the model's efficiency.

Exp. 4: The feature map multiplier m. We test if our smallest model could be further reduced in size by lowering m. We take $d = 4$ and $b = 8$, and compare $m \in \{1, 1.5, 2\}$. Figure 9.d shows that the reduction to $m = 1.5$ retains performance. For $m = 1$, the performance drops. Reducing m to 1.5 could therefore be used to scale down the model when further reducing b significantly decreases performance.

Conclusion. To construct the reduced and baseline denoiser, we use the smallest and largest values from the found hyperparameter ranges. Resulting in baseline (UNet): $\{b = 32, d = 4, m = 2, c = 2\}$ and reduced (UNet-S): $\{b = 8, d = 4, m = 1.5, c = 2\}$. Table 8 compares the two models for a selection of the noise levels. Although the reduced model has significantly fewer FLOPs and lower latency, the denoising performance is relatively similar to the baseline denoiser.

The UNet hyper-parameter experiments are replicated using MBConvs, which lead to similar findings. Moreover, the Conv UNet slightly outperforms the MB model. Therefore, the Conv model is used.

Table 8: Compares Baseline and Reduced UNet denoisers. The reduced model has significantly lower FLOPs and latency yet similar denoising performance.

Model	FLOPs (M) ↓	Lat. (ms) ↓	Metric	Noise level (σ)			
				0.2	0.4	0.8	1
UNet	1301.8	7.10	PSNR ↑	33.9	29.5	23.8	22.3
			SSIM ↑	0.99	0.98	0.95	0.92
UNet-S	51.2	2.38	PSNR ↑	33.2	28.7	23.3	22.0
			SSIM ↑	0.99	0.98	0.94	0.92

B Search space

In Section 4.1 different variations of the TF-NAS search space are used [19]. Table 9 displays the candidate operations and for which search space size they are used. The search space with 4 operators is constructed using the MBConvs without SE-layer, as this is most common in recent NAS methods [38,42]. For the 6-operator search space, we add the possibility of using an SE layer on the operators where the kernel size is three and the expansion rate is three or six. We use the two smallest operators as they can be used for smaller target latencies too. The search space with 8 operators simply uses all combinations.

Table 9: Overview of the candidate blocks for the different search space sizes {4, 6, 8}. MBConv operators are used with different kernel sizes k, expansion rate e, and in- or excluding the squeeze- and excitation-layer.

Name	Kernel	Expansion rate	SE-layer	4	6	8
MB-k3-e3	3	3	-	✓	✓	✓
MB-k3-e6	3	6	-	✓	✓	✓
MB-k5-e3	5	3	-	✓	✓	✓
MB-k5-e6	5	6	-	✓	✓	✓
MB-k3-e3-se	3	3	✓	-	✓	✓
MB-k3-e6-se	3	6	✓	-	✓	✓
MB-k5-e3-se	5	3	✓	-	-	✓
MB-k5-e6-se	5	6	✓	-	-	✓

C Learned vs. Removed β: Additional results

In Experiment 1 of Section 4.1, we test the influence of removing β from the search approach. The models with Removed β significantly outperform the models with Learned β in accuracy. Besides, the found models are more similar for Removed than Fixed, Removed β differs only 0.04ms and 0.2% accuracy, while Learned β differs 0.57ms and 1.4% accuracy. This indicates that the search for Removed is more stable.

Table 10: Compares four searched models with target latency 6 ms. Trained on clean images. Two models are searched without β and the other two using learned β. Removed outperforms learned β.

Type	Search id	LAT (ms) ↓	Acc (%) ↑
Removed β	1	5.85	**86.2**
	2	5.81	**86.4**
Learned β	1	5.04	84.9
	2	4.47	83.5