Tunable Convolutions with Parametric Multi-Loss Optimization

Supplementary Materials

Matteo Maggioni, Thomas Tanay, Francesca Babiloni, Steven McDonagh, Aleš Leonardis
Huawei Noah's Ark Lab
{matteo.maggioni, thomas.tanay, francesca.babiloni, steven.mcdonagh, ales.leonardis}@huawei.com

A. Implementation Aspects

In this section we will give detailed information on the architectures used in the experiments of the main paper.

A.1. Architectures

In our comparisons against controllable networks, we use two backbones: a ResNet [S6, S11] for image restoration experiments, and a UNet [S8, S14] for style transfer.

Image Restoration. An illustration of the architecture can be found in Fig. S1. All convolutions have 3×3 kernel size and 64 channels. The first convolution in each ShuffleBlock expands the number of channels from 64 to 256, and pixel shuffling [S15] upsamples the resolution by $\times 2$. A long skip connection is used connecting the output of the first layer to the output of the last residual block. Obviously the two ShuffleBlocks are only used in our $\times 4$ super-resolution experiments, and in the case of our denoising experiments the final predicted image is obtained through the last convolution directly after the skip connection.

Style Transfer. An illustration of the architecture can be found in Fig. S2. The first and last convolutions use 9×9 kernel size, whereas the size for all other convolutions is 3×3. Initial number of channels is 64. At every downsampling stage resolution is halved and number of channels is doubled. Downsampling is performed with a strided convolution with stride 2. Upsampling is done with $\times 2$ nearest interpolation. Instance normalization [S18] is used everywhere (aside from the very last layer), and the input of each convolution is padded with reflection mode. No skip connections are used between encoder and decoder stages.

A.2. Tunable Convolutions

During training, we sample a different –random– set of parameters for each instance in the batch. Consequently, we have in general different convolutional weights (*i.e.* kernels and biases) for each instance in the batch. When the batch size is equal to 1, or when the parameters of all instances in the batch are equal, we can apply a standard forward convolutional operation. Conversely, if batch size is larger than 1 and different parameters are used for different instances, we need a mechanism to apply a different convolution to each instance in the batch.

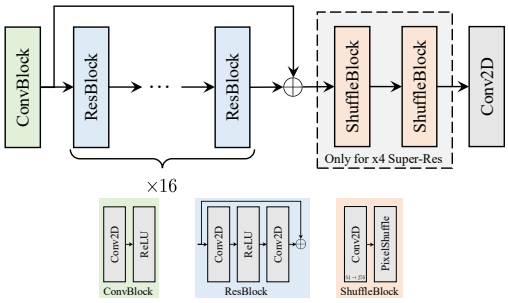

Figure S1. Illustration of the ResNet [S6, S11] architecture used for image restoration experiments.

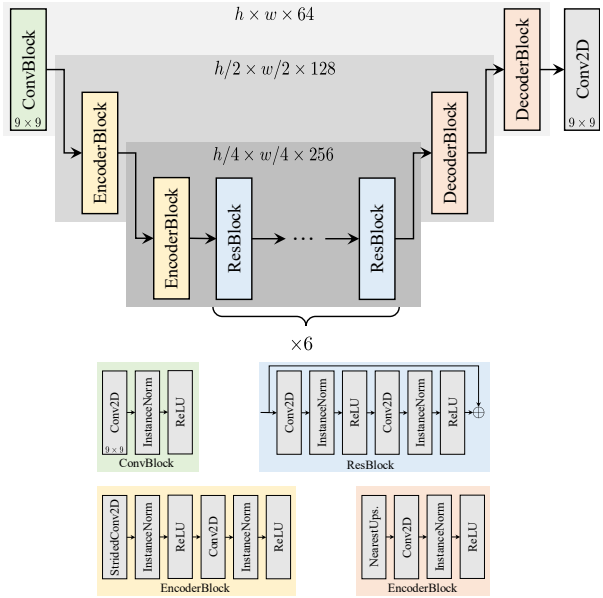

Figure S2. Illustration of the UNet [S8, S14] architecture used for style transfer experiments.

Algorithm S1 Forward pass of tunable convolutions.

Input: $\boldsymbol{X} \in \mathbb{R}^{n \times h \times w \times c}$
Input: $\boldsymbol{\Omega} \in \mathbb{R}^{n \times p}$ ▷ Parameters
Input: $\boldsymbol{K} \in \mathbb{R}^{p \times k \times k \times c \times d}$ ▷ Kernels
Input: $\boldsymbol{B} \in \mathbb{R}^{p \times d}$ ▷ Biases
Output: $\boldsymbol{Y} \in \mathbb{R}^{n \times h \times w \times d}$
  procedure TUNABLECONV2D($\boldsymbol{X}, \boldsymbol{\Omega}, \boldsymbol{K}, \boldsymbol{B}$)
    $\boldsymbol{A} \leftarrow \phi_t(\boldsymbol{\Omega}) \in \mathbb{R}^{n \times p}$
    $\boldsymbol{Y} \leftarrow \{\}$
    for $j \leftarrow 1, n$ **do**
      $\boldsymbol{\alpha} \leftarrow A_j \in \mathbb{R}^p$ ▷ Aggregation weights
      $\hat{\boldsymbol{k}} \leftarrow \sum_{i=1}^{p} \alpha_i \cdot \boldsymbol{K}_i \in \mathbb{R}^{k \times k \times c \times d}$
      $\hat{\boldsymbol{b}} \leftarrow \sum_{i=1}^{p} \alpha_i \cdot \boldsymbol{B}_i \in \mathbb{R}^{d}$
      $\boldsymbol{x}_j \leftarrow \boldsymbol{X}_j \in \mathbb{R}^{h \times w \times c}$
      $\boldsymbol{y}_j \leftarrow \boldsymbol{x}_j \circledast \hat{\boldsymbol{k}} + \hat{\boldsymbol{b}} \in \mathbb{R}^{h \times w \times d}$
      $\boldsymbol{Y} \leftarrow \boldsymbol{Y} + \{\boldsymbol{y}_j\}$
    end for
  end procedure

We implement our tunable convolution as a custom layer which includes an internal 5D kernel tensor $\boldsymbol{K} \in \mathbb{R}^{p \times k \times k \times c \times d}$ where p is the number of parameters, k is the kernel size, c is the number of input channels, and d is the number of output channels, and a 2D bias $\boldsymbol{B} \in \mathbb{R}^{p \times d}$. Then in the forward pass, given a batch of inputs $\boldsymbol{X} \in \mathbb{R}^{n \times h \times w \times c}$, being n the batch size, and a batch of parameters $\boldsymbol{\Omega} \in \mathbb{R}^{n \times p}$, we operate as outlined in Algorithm S1. Note that in the algorithm we have used a **for** loop to indicate that the convolution is instance-wise. However in our implementation the input to $\boldsymbol{X}$ is reshaped to $1 \times h \times w \times n \cdot c$, the tuned kernels $\hat{\boldsymbol{K}}$ (obtained by reducing over p) are reshaped to $k \times k \times n \cdot c \times d$, and then the final output is computed via a group convolution between $\boldsymbol{X}$ and $\hat{\boldsymbol{K}}$ with a number of groups equal to the batch size n.

B. Ablation

In this section we will perform ablation on various characteristics of the proposed tunable convolutions. Let us set the stage by building a multi-loss with $p = 2$ objectives, namely a reconstruction objective $\mathcal{L}_{\text{rec}}$ measuring distortion with an L_1 distance, and a perceptual objective $\mathcal{L}_{\text{per}}$ implemented as LPIPS loss [S24]. Our tunable model is trained using a combinations of these losses defined as

$$\widehat{\mathcal{L}}_{\text{rp}} = \omega_1 \cdot \mathcal{L}_{\text{rec}} + \omega_2 \cdot \mathcal{L}_{\text{per}}, \tag{S1}$$

where, as explained in Sec. 3.2 of the main paper, the parameter vector $\boldsymbol{\omega}$ is randomly sampled at training time to train all possible intermediate behaviors.

In this experiment, we objectively evaluate the different methods by measuring their ability to maximize PSNR when $\mathcal{L}_{\text{rec}}$ dominates, and minimizing LPIPS when $\mathcal{L}_{\text{per}}$ dominates. In Table S1, we report performance for different combinations of the ω_i parameters averaged over three noise levels $\sigma \in [5, 15, 30]$. In addition to the baseline methods DNI [S20], DyNet [S16], and CFSNet [S19], we also provide comparison against *Fixed* networks trained individually for each parameter combination to provide upper-bound performances (*i.e.* five separate training runs). From the table we can clearly see that the proposed method almost always achieves better PSNR accuracy and LPIPS quality among the controllable methods, and it is even comparable if not better than the *Fixed* models which, we recall, are specialized networks individually trained for each combination. Among the baseline methods we note that DNI and DyNet are unable to generalize well to the intermediate parameter combinations.

	Kodak [S9]					CBSD68 [S13]					
ω_1	0.00	0.25	0.50	0.75	1.00	0.00	0.25	0.50	0.75	1.00	$\mathcal{L}_{\text{rec}}$
ω_2	1.00	0.75	0.50	0.25	0.00	1.00	0.75	0.50	0.25	0.00	$\mathcal{L}_{\text{per}}$
PSNR	34.10	34.59	35.12	35.26	**35.61**	33.16	34.04	34.19	34.29	**34.60**	Fixed
	34.10	23.40	20.65	22.55	**35.61**	33.16	22.63	19.74	21.65	**34.60**	DNI [S20]
	34.14	31.93	31.75	33.23	35.33	33.31	31.40	31.16	32.43	34.42	DyNet [S16]
	34.90	35.11	35.26	35.34	35.33	33.97	34.18	34.34	34.42	34.42	CFSNet [S19]
	34.95	**35.26**	**35.40**	**35.45**	35.40	**34.04**	**34.31**	**34.44**	**34.48**	34.43	Ours
LPIPS	**0.084**	**0.087**	**0.095**	0.103	0.106	**0.094**	**0.095**	**0.105**	0.113	0.118	Fixed
	0.084	0.129	0.129	0.129	0.106	**0.094**	0.144	0.144	0.144	0.118	DNI [S20]
	0.094	0.107	0.098	**0.099**	0.102	0.104	0.116	0.108	**0.111**	0.115	DyNet [S16]
	0.095	0.097	0.099	**0.099**	0.102	0.104	0.106	0.108	0.112	0.115	CFSNet [S19]
	0.089	0.096	0.098	**0.099**	**0.101**	0.098	0.106	0.107	**0.111**	**0.114**	Ours

Table S1. Tuning denoising perceptual quality. Our method can tune its behaviour to optimize perceptual quality $(\omega_1, \omega_2) = (0.00, 1.00)$ or data fidelity $(1.00, 0.00)$ with no retraining, while outperforming other tunable methods in the most of the evaluated cases, and being even comparable to a Fixed model trained specifically for each parameter combination.

B.1. Number of Tunable Convolutions

We recall that our tunable convolutions act as a drop-in replacement for traditional convolutions in any existing neural network. In the main paper, the underlying strategy in all our experiments was to replace all convolutions of the baseline networks with our tunable variants. However natural questions arise related to how network performance is affected when a smaller subset of the original convolutions is replaced.

For this ablation, we follow the same denoising setup explained previously using the loss (S1) to control fidelity and perceptual objectives, and we train different versions of the same model by varying the number of layers replaced with our tunable convolutions. As illustrated in Fig. S1, we have 18 layers in total, *i.e.* 16 residual blocks plus 1 first and 1 last convolution. In what follows, we express the number of tunable layers as a percentage relative to the total number of layers. We instantiate five variants replacing: 5% of layers (*i.e.* replacing only layer 18), 30% (layer 13 to 18),

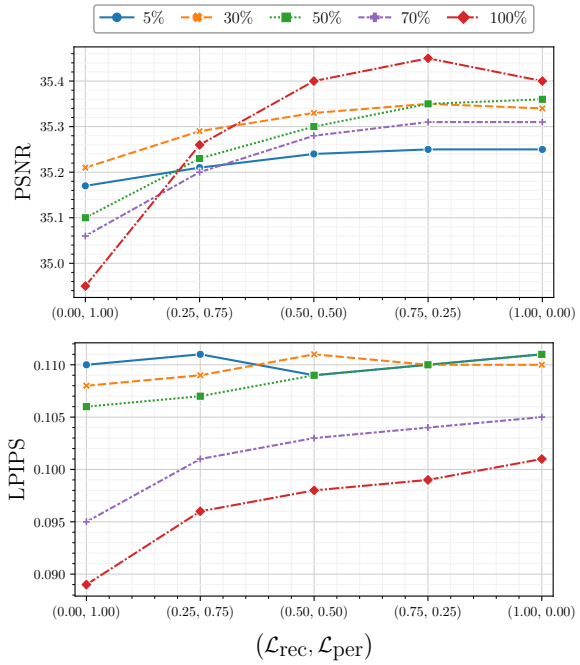

Figure S3. Tunable denoising performance for a model trained for reconstruction fidelity $\mathcal{L}_{\text{rec}}$ and perceptual quality $\mathcal{L}_{\text{per}}$ objectives on Kodak [S9] dataset with varying percentage of tunable convolutional layers.

50% (9 to 18), 70% (5 to 18), and the 100% (1 to 18). Note that in all cases we place all the tunable layers at the end of the model to maximize their influence on the predicted output image. The 100% configuration is the same one used in the main paper.

In Fig. S3 we illustrate the performance of these different tunable networks. It is evident that, as the replacement percentage decreases, the performance tends to become uniform over the parameter range. In fact in these cases the model is not able to adapt to the different configurations of the parameters, and thus converges to an optimal average solution. The 100% configuration is clearly optimal as it shows the largest degree of adaptation, and furthermore has the best PSNR and LPIPS when the corresponding tuning parameters are dominant. Interestingly the best PSNR score is obtained with tuning $(0.75, 0.25)$, thus highlighting that some contribution of perceptual objective $\mathcal{L}_{\text{per}}$ also benefits the reconstruction fidelity.

B.2. Number of Objectives

In the main paper we focused on experiments based on $p = 2$ objectives, as this is the working setup of our main comparison methods [S16, S19]. An advantage of the proposed tunable convolutions is the ability to deal with even more objectives, which are all explicitly optimized during training. In the main paper, we have subjectively demonstrated this ability in Fig. 7 through style transfer [S8]; here we also present objective evaluation in this scenario to further substantiate our claim.

	Kodak [S9]			CBSD68 [S13]			
ω_1	0.00	0.00	**1.00**	0.00	0.00	**1.00**	$\mathcal{L}_{\text{rec}}$
ω_2	0.00	**1.00**	0.00	0.00	**1.00**	0.00	$\mathcal{L}_{\text{per}}$
ω_3	**1.00**	0.00	0.00	**1.00**	0.00	0.00	$\mathcal{L}_{\text{noise}}$
PSNR	26.73	33.66	**35.31**	26.78	32.77	**34.37**	Ours
LPIPS	0.099	**0.083**	0.098	0.108	**0.093**	0.107	Ours
PSNR_η	**51.91**	33.66	35.31	**50.53**	32.77	34.37	Ours

Table S2. Tuning a single model for $p = 3$ objectives. We test different parameter $(\omega_1, \omega_2, \omega_3)$ configurations to optimize fidelity $(1.00, 0.00, 0.00)$, perceptual quality $(0.00, 1.00, 0.00)$ or noise preservation $(0.00, 0.00, 1.00)$. As expected, each configuration maximizes the corresponding metric.

Following the denoising setup discussed in Sec. 4.1.1 of the main paper, let us train a tunable network using an increased number ($p = 3$) of objectives. In order to evaluate objective performance, we use three (competing) objectives that explicitly optimize different metrics: namely a reconstruction objective $\mathcal{L}_{\text{rec}}$ optimizing PSNR, a perceptual objective $\mathcal{L}_{\text{per}}$ optimizing LPIPS, and a noise preservation objective $\mathcal{L}_{\text{noise}}$ optimizing PSNR_η. As in the main paper, we report performance averaged over three levels of noise $\sigma \in [5, 15, 30]$. Numerical results reported in Table S2 clearly show that the three different metrics are maximized when the corresponding parameter is dominant. In other words, best PSNR is obtained when ω_1, *i.e.* the parameter controlling the reconstruction objective, is equal to 1. Similar reasoning applies to the other metrics and corresponding parameters. Further, we notice that the performance obtained by simultaneously training for the three objectives is similar, and in some cases even better, than the performance obtained by training two objectives, *i.e.* $(\mathcal{L}_{\text{rec}}, \mathcal{L}_{\text{per}})$ in Table S1 and $(\mathcal{L}_{\text{rec}}, \mathcal{L}_{\text{noise}})$ in Table 1 of the main paper, thus showing that our method can retain accuracy even with an increased number of tunable objectives.

In Fig. S4, we show results from the Kodak dataset [S9] obtained by tuning our model to generate outputs with different characteristics. Noise in the input image is Gaussian with standard deviation $\sigma = 30$. As one can see, minimizing denoising strength retains a large amount of residual noise, maximizing fidelity generates smooth and noise-free images, and maximizing perceptual quality generates sharper details.

B.3. Number of Kernels and Biases

As a natural extension for our tunable convolutions, we explore the potential of increasing the number of kernels (and biases) by modifying the function to predict aggre-

	Kodak [S9]					CBSD68 [S13]					
ω_1	0.00	0.25	0.50	0.75	1.00	0.00	0.25	0.50	0.75	1.00	$\mathcal{L}_{\text{rec}}$
ω_2	1.00	0.75	0.50	0.25	0.00	1.00	0.75	0.50	0.25	0.00	$\mathcal{L}_{\text{per}}$
PSNR	34.95	35.26	35.40	35.45	35.40	34.04	34.31	34.44	34.48	34.43	$q=2$
	35.25	35.33	35.40	35.44	35.46	34.30	34.38	34.44	34.47	34.47	$q=4$
	35.32	35.40	35.46	35.50	35.52	34.35	34.42	34.47	34.50	34.50	$q=8$
LPIPS	0.089	0.096	0.098	0.099	0.101	0.098	0.106	0.107	0.111	0.114	$q=2$
	0.095	0.099	0.104	0.109	0.112	0.104	0.109	0.113	0.120	0.124	$q=4$
	0.093	0.098	0.105	0.105	0.106	0.101	0.106	0.109	0.116	0.119	$q=8$

Table S3. Tuning denoising with $p = 2$ objectives, namely reconstruction fidelity and perceptual quality. We denote with q the number of kernels and biases inside each tunable convolution. The q kernels (and biases) are aggregated by predicting weights from the $p = 2$ parameters using a fully-connected layer. We highlight the baseline case where $q = p = 2$.

gation weights defined in Eq. (6) of the main paper from $\phi_t : \mathbb{R}^p \rightarrow \mathbb{R}^p$ to $\phi_t : \mathbb{R}^p \rightarrow \mathbb{R}^q$, for some $q > p$. In other words, instead of having exactly one kernel for every external parameter $\boldsymbol{\omega}$, we can use ϕ_t as a small expansion network to map the p input parameters to a larger number q of tunable kernels. Under this notation, it is obvious that the baseline case of tunable convolutions described in the main paper can be simply obtained as a special case of this more general formulation by setting $q = p$.

We report in Table S3 PSNR and LPIPS results of our usual tunable network trained to optimize denoising reconstruction and perceptual quality. Performance is as always averaged over noise levels $\sigma \in [5, 15, 30]$. As one can see, increasing q will generally provide an increase in performance, and in particular the PSNR improvements are significant in cases where $\mathcal{L}_{\text{per}}$ dominates, and PSNR seems to saturate at $q = 8$. Differently, LPIPS is similar to the baseline case $q = 2$ where $\mathcal{L}_{\text{per}}$ dominates, however, with larger q, the LPIPS decreases more as the influence of $\mathcal{L}_{\text{rec}}$ increases. This might also be a consequence of the fact that PSNR increases as well in these cases. In conclusion, we note that increasing the number q of kernels indeed augment the representation power of the model, as evidenced by the improved PSNR and the increased ability of the model to adapt to changes in the parameters, *i.e.* the difference in LPIPS between parameters $(0.00, 1.00)$ and $(1.00, 0.00)$ is larger than in the baseline case, hence showing a better ability to disentangle the two different objectives. However the baseline case offers a good compromise between overall performance, dynamic behavior over the parameters, and it is also has the smallest memory requirements.

C. Additional Experiments

In this section we will provide additional details and visual results, supplementing the experimental section of the main paper.

	Denoising		Super-Resolution		Style Transfer	
	GFLOPS	Params	GFLOPS	Params	GFLOPS	Params
Baseline	19.38	1.18 M	31.89	1.48 M	13.24	8.78 M
DyNet [S16]	19.38	1.18 M	51.21	1.55 M	13.24	8.78 M
CFSNet [S19]	19.38	1.43 M	43.97	2.02 M	13.24	9.15 M
CResMD [S5]	19.38	1.18 M	31.89	1.48 M	N/A	
Ours	19.38	2.37 M	31.89	2.96 M	13.24	17.55 M

Table S4. Number of trainable network parameters and GFLOPS complexity of all networks used in our experiments. GFLOPS is computed with respect to an input resolution of 128×128 and $p = 2$ input tuning parameters. The proposed method always maintains the same complexity as the baseline (non-tunable) model.

C.1. Computational Complexity

Tunable Networks. Let us recall that we use a ResNet backbone for our image restoration experiments, and a UNet backbone for style transfer. Detailed settings of the baseline architectures can be found in Sec. A.1. Here we explain how these backbones have been configured to implement DyNet [S16], CFSNet [S19], CResMD [S5], as well as our proposed method. As mentioned in Sec. B.1, for the proposed method we replace all convolutions in the model with our tunable variants. For DyNet, CFSNet, and CResMD, we follow the settings proposed by the original authors, and apply feature modulation at every layer except the first and last one. Since DyNet and CFSNet needs a second –tuning– branch to enable feature modulation, in order to maintain the same complexity as all other compared methods, we halve the size of the convolutions, *i.e.* in our ResNet we use 8 residual blocks instead of 16, whereas in our UNet we use only 1 convolution at every encoder and decoder stages, and 3 residual blocks in the latent stage.

A summary of the total number of network parameters (*i.e.* trainable weights, not to be confused with the tuning parameter $\boldsymbol{\omega}$) and computational complexity measured as floating point operations per second (FLOPS) for all configuration is summarized in Table S4. From the table we notice that, after reducing the number of convolutions by half, the complexity of DyNet and CFSNet is equal to the baseline case for denoising and style transfer. However in the case of super-resolution we still notice an increase in complexity due to the modulation of the features in the upsampling stage which requires additional tuning blocks that cannot be removed. Conversely the proposed method is able to increase the capacity of the underlying networks (by means of the increased number of parameters) while maintaining virtually the same complexity as the baseline. Note that we do not provide numbers for CResMD in the case of style transfer because the authors did not provide implementation of their residual modulation strategy in case of blocks that changes the feature resolution, so we cannot directly apply their strategy to a UNet.

C.2. Training

Tunable Networks. In all our experiments we sample the tuning parameters $\boldsymbol{\omega}$ uniformly in $[0, 1]$, and we sample a different set of parameters for each instance and for each batch during optimization. In the case of DyNet [S16] and CFSNet [S19], we follow the training setup proposed by the original authors, and we first train the main branch of the networks using the first objective for 250K iterations, and then we fix the trained weights of the main branch while we train the second tuning branch for the remaining 250K iterations using the second objective. In the case of CResMD [S5], we simply train the network using a single fixed objective and we randomize the degradation settings (noise standard deviation σ and blur standard deviation ρ) using the same settings as proposed by the original authors. In all cases we use Adam [S17] optimizer and the learning rate is fixed and equal to $1e^{-4}$ throughout the training. In all cases, we optimize for 500K iterations using batch size 16 and input patch size 64×64.

Traditional Networks. For our experiments using state-of-the-art baselines, we take SwinIR [S10] and NAFNet [S4] implementation provided by the authors, and replace every learnable operation with our tunable counterparts, including standard, depth-wise, point-wise, strided convolutions, linear (dense, fully-connected MLPs) layers as well as spatial, channel, and window attention.

Traditional Network – SwinIR Following [S10], we train our tunable SwinIR model defined as proposed by the authors for the case of "color image denoising". This model is trained for 1.6M iterations using Adam [S17], batch size 8, patch size 128×128, and learning rate $2e^{-4}$ which is halved four times at iterations [800K, 1.2M, 1.4M, 1.5M]. We optimize a reconstruction and noise loss based on Charbonnier distance [S3]. For training we use a combination of DIV2K [S2] and Flickr2K [S11] datasets, as opposed to the original SwinIR, which also uses additional BSD500 and WED datasets [S10]. We train three different models, one for each tested Gaussian noise level, that is $\sigma \in [15, 25, 50]$. In our super-resolution experiment, we use a tunable SwinIR as defined by the authors for the case of "classical image SR" [S10] trained on 48×48 patch size from DIV2K. In this case we train for 500K iterations, batch size 32, learning rate $2e^{-4}$ which is halved four times at iterations [250K, 400K, 450K, 475K]. The tunable network is optimized with a reconstruction L_1 objective for data fidelity, and the GAN objective [S21] defined in Sec. 4.1.3 of the main paper.

Traditional Network – NAFNet. The second state-of-the-art architecture considered in the main paper is NAFNet [S4]. In our experiments we use NAFNet version with "width 32" and "36 # of blocks" originally proposed by the authors. We train on (medium) sRGB SIDD [S1] using 320 training images and $1,280$ validation crops. In this experiment, we optimize using AdamW [S12] for 200K iterations, with $\beta_1 = \beta_2 = 0.9$, batch size 32, patch size 256×256, and initial learning rate $1e^{-3}$ which is then decreased with cosine annealing to the final $1e^{-7}$ value. Our tunable loss includes a reconstruction objective defined as PSNR, and a noise preservation objective defined as PSNR_η. Note that in Table 2b of the main paper, in order to include all metrics reported in the original paper [S4], we also provide SSIM_η, which is the SSIM [S22] between the predicted image and the target image y_η used in Eq. (9). Note that for SSIM, in line with the authors of NAFNet, we use a 3D SSIM implementation to compute the metric directly on the 3D RGB image domain, rather than averaging 2D SSIM of each image channel.

Style Transfer. The images used for style transfer are *Mosaic* [S25], *Edtaonisl* [S26], and *Kandinsky* [S27]. If needed, we reduce the resolution of the style images such that their maximum dimension is 800px. The weights $(\lambda_{\text{gram}}, \lambda_{\text{vgg}}, \lambda_{\text{tv}})$ for the style transfer loss defined in Sec. 4.2 of the main paper are $(3e^{-3}, 3e^{2}, 1e^{-4})$ for *Mosaic*, $(1e^{-3}, 4e^{2}, 1e^{-6})$ for *Edtaonisl*, and $(1e^{-3}, 6e^{2}, 1e^{-6})$ for *Kandinsky*.

C.3. Additional Visual Results

In the remainder of these supplementary materials, we provide the following additional figures. More information about the experiments can be found in the corresponding sections, indicated in parentheses.

- Fig. S4. Denoising: new visualizations with $p = 3$ objectives for our method (Sec. B.2);
- Fig. S5. Denoising: additional comparisons against CFSNet [S19] (Sec. 4.1.1 of the main paper);
- Fig. S6. Denoising and deblurring. New comparison against CResMD [S5] (Sec. 4.1.2 of the main paper);
- Fig. S7. Denoising and deblurring: additional visualizations for our method (Sec. 4.1.2 of the main paper);
- Fig. S8. Super-resolution: additional comparisons against CFSNet [S19] (Sec. 4.1.3 of the main paper);
- Fig. S9. Denoising: new visualizations for tunable NAFNet [S4] applied on real raw images from SIDD [S1] (Sec. 4.1.1 of the main paper);
- Fig. S10. Super-resolution: new visualizations for tunable SwinIR [S10] (Sec. 4.1.3 of the main paper);
- Fig. S11. Style transfer: new visualizations with $p = 3$ objectives for our method (Sec. 4.2 of the main paper);
- Fig. S12 & Fig. S13. Style transfer: additional comparisons against DyNet [S16] (Sec. 4.2 of the main paper).

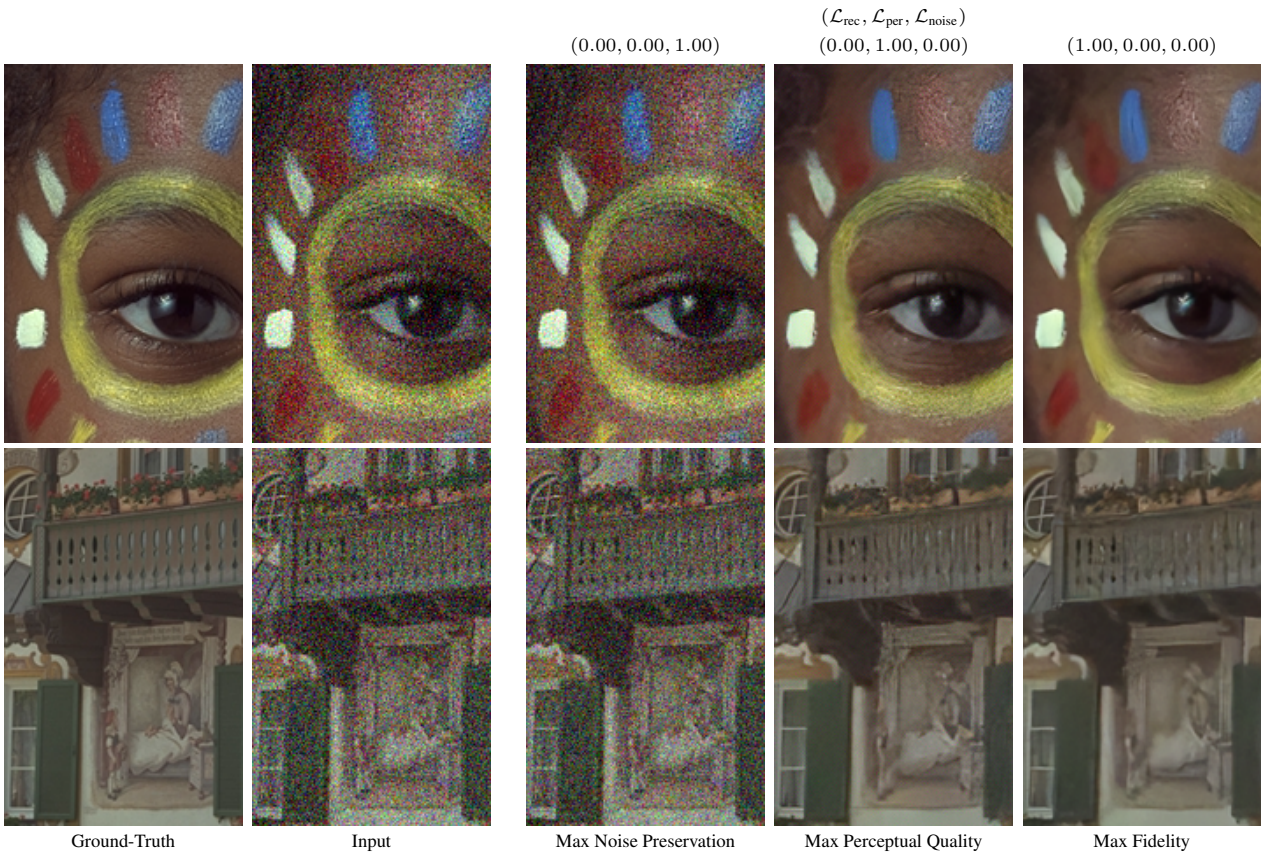

Figure S4. Image 15 (top) and 24 (bottom) from the Kodak dataset [S9] corrupted by Gaussian noise with standard deviation $\sigma = 30$. A single model is tuned at inference time to generates images with different characteristics by interacting with the three parameters corresponding to reconstruction fidelity $\mathcal{L}_{\text{rec}}$, perceptual quality $\mathcal{L}_{\text{per}}$, and noise preservation $\mathcal{L}_{\text{noise}}$.

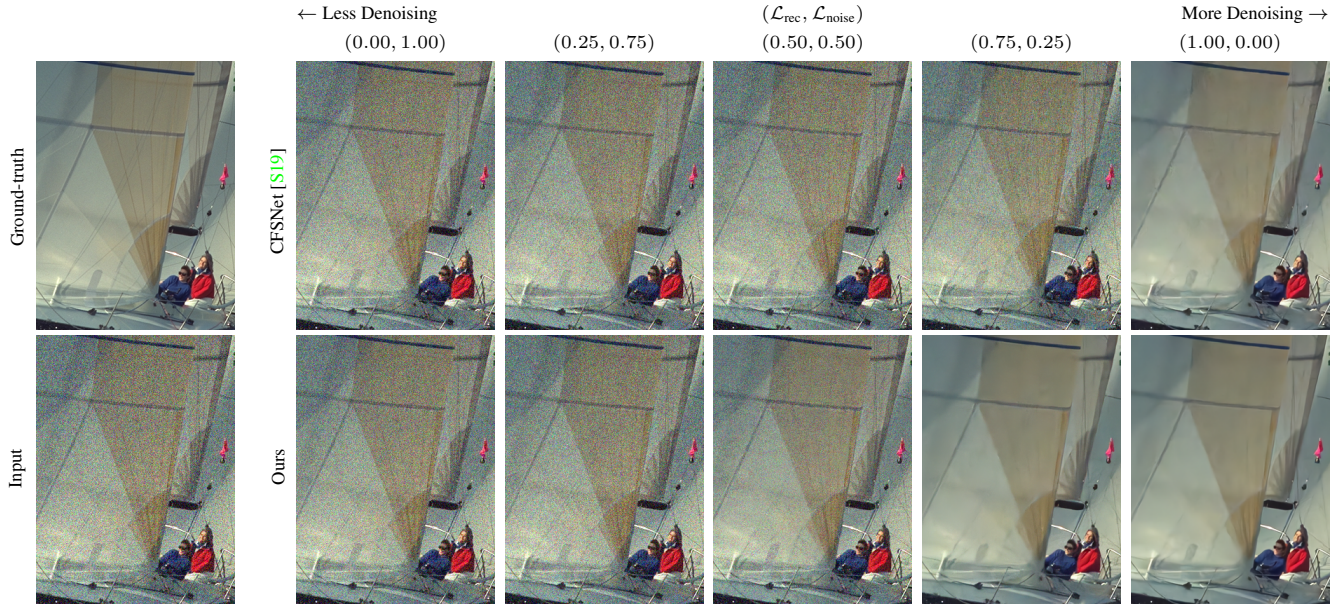

Figure S5. Tuning denoising strength on image 10 from the Kodak dataset [S9] corrupted by Gaussian noise with standard deviation $\sigma = 30$. Our method generates images with a more consistent increase in denoising strength as well as better details and fewer artifacts.

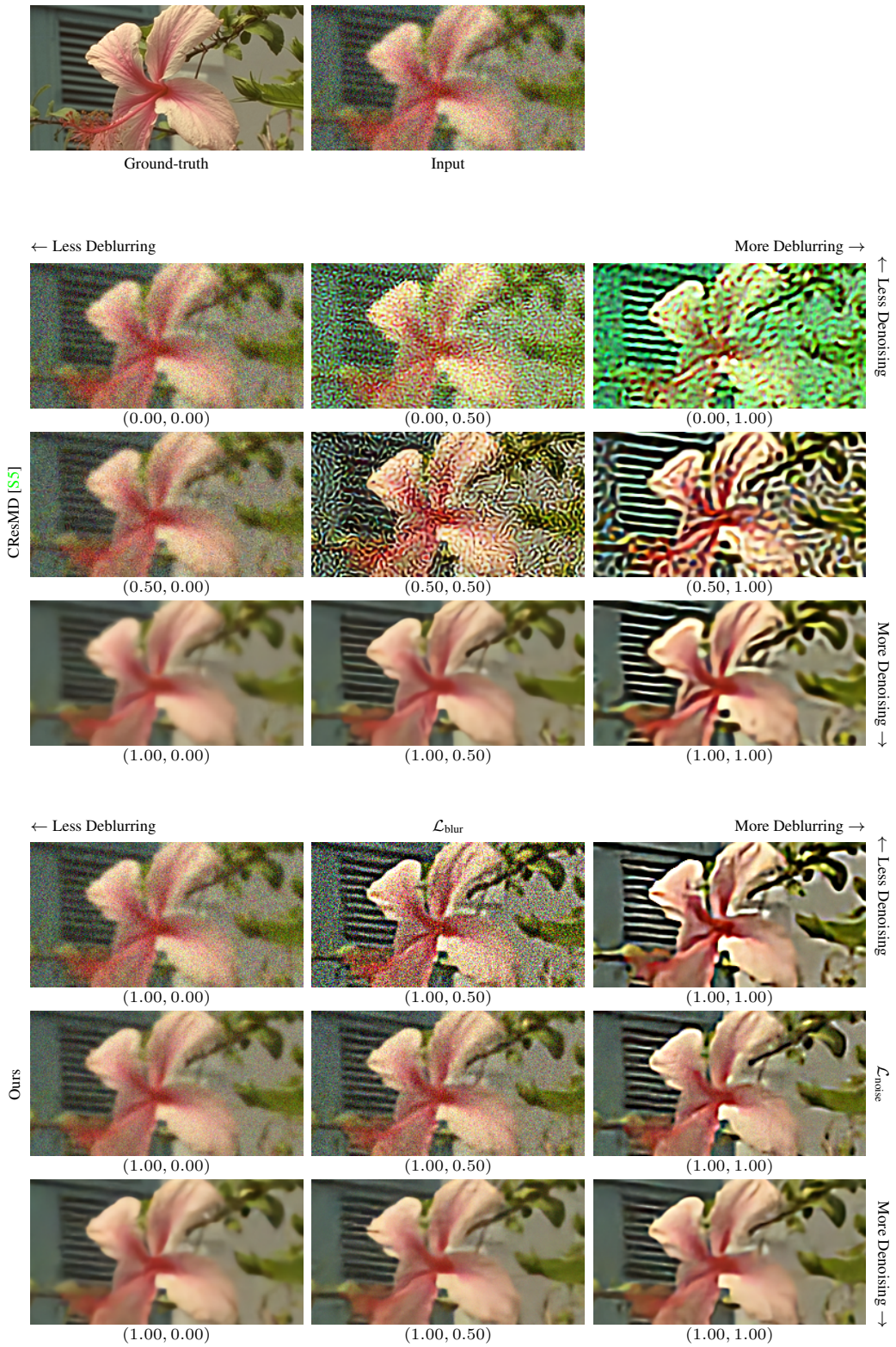

Figure S6. Image 07 from the Kodak dataset [S9] corrupted by Gaussian noise $\sigma = 30$ and Gaussian blur with size $\rho = 2$. Our method optimizes denoising and deblurring objective for all combinations of parameters (ω_1, ω_2). CResMD is trained on a single objective to maximize fidelity conditioned on the true parameters, *i.e.* $(1.00, 0.50)$, and thus generates artifacts for out-of-distribution inputs.

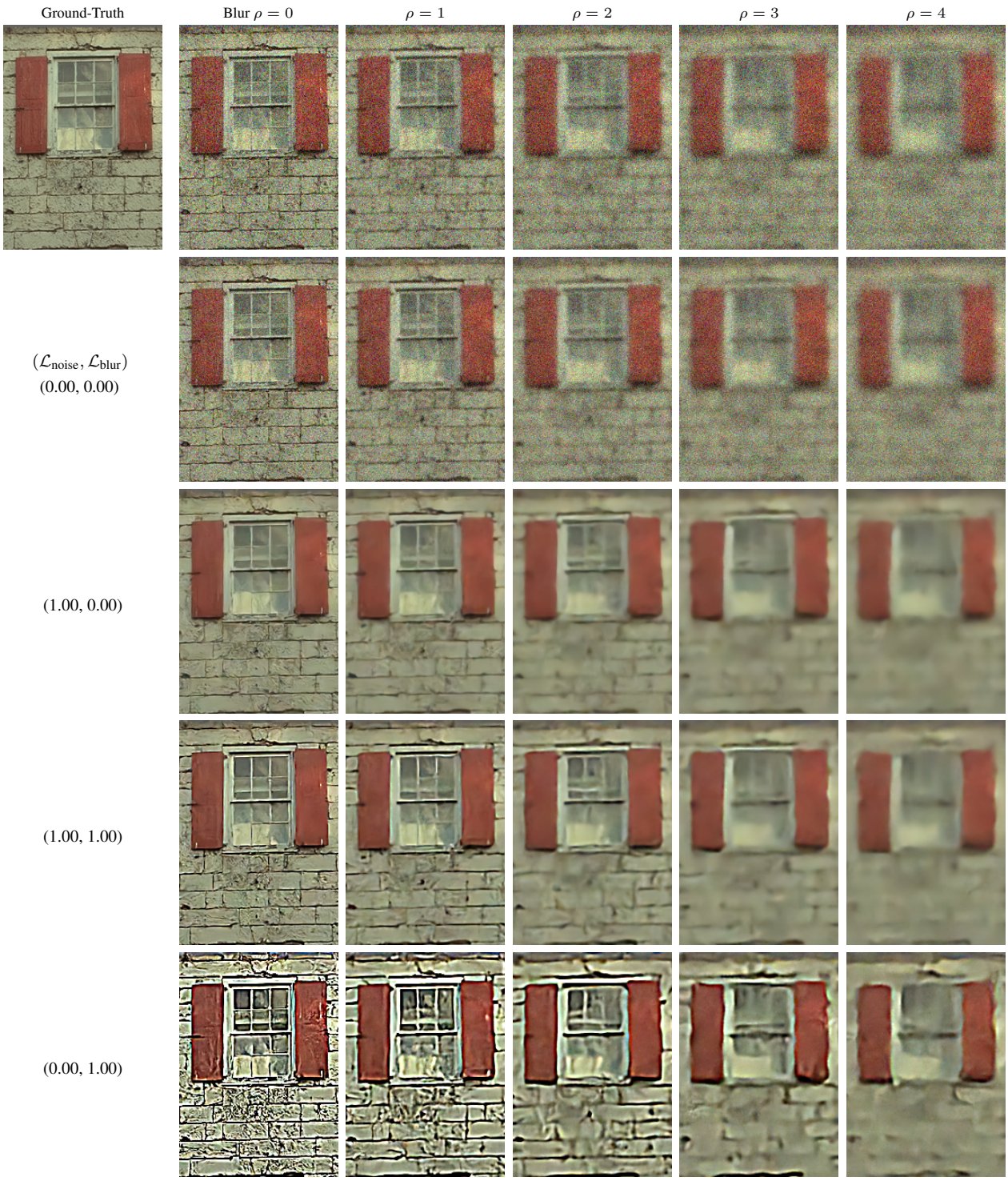

Figure S7. Image 01 from the Kodak dataset [S9] corrupted by Gaussian noise with standard deviation $\sigma = 30$ and Gaussian blur with size 21×21 and various levels of standard deviation ρ. A single model is tuned at inference time to generates images with different characteristics by interacting with two parameters corresponding to noise $\mathcal{L}_{\text{noise}}$ and deblurring $\mathcal{L}_{\text{blur}}$ strength.

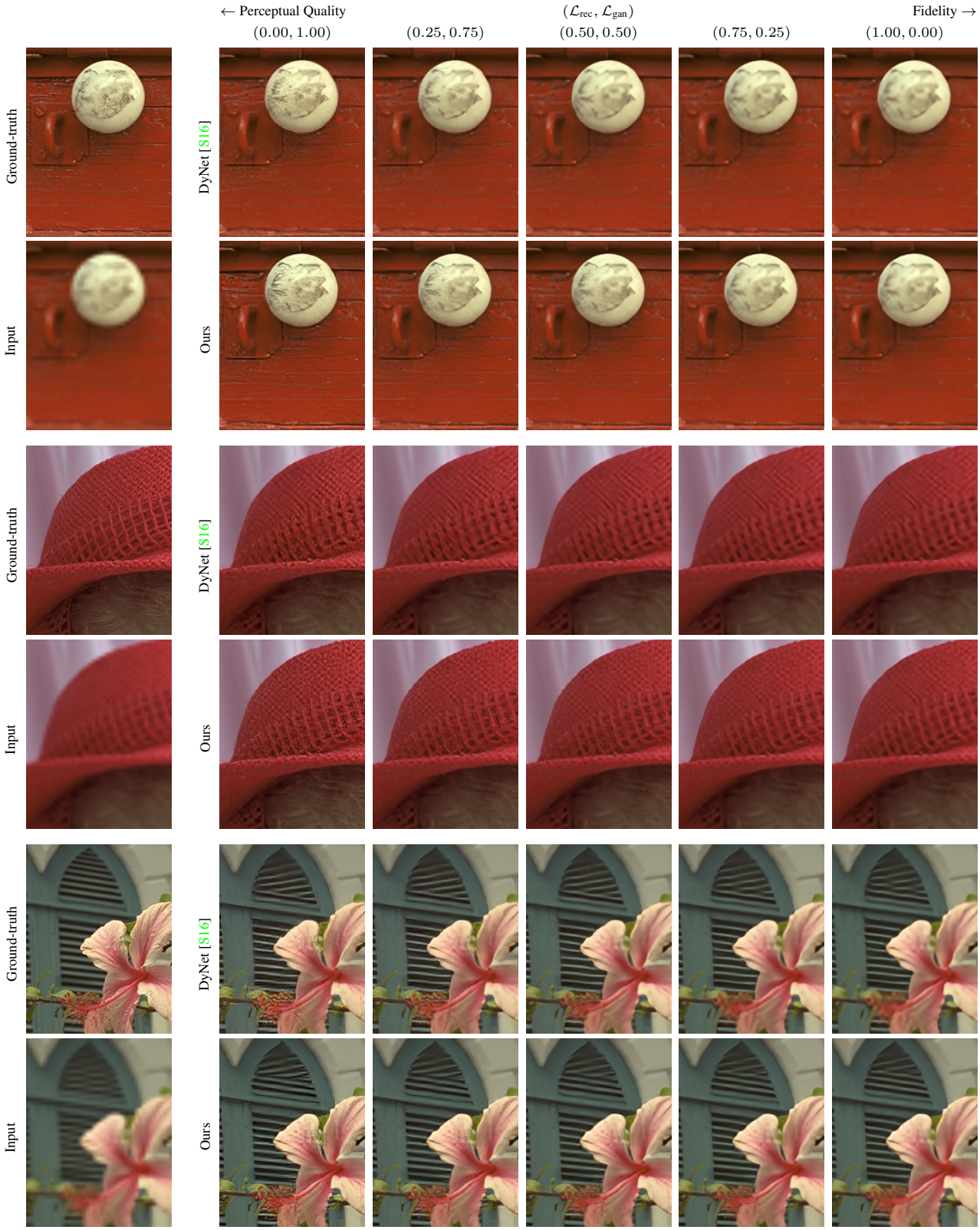

Figure S8. Tuning $\times 4$ super-resolution fidelity and perceptual quality. From top to bottom: image 02, 04, and 07 from the Kodak dataset [S9]. Our method generates more detailed and perceptually pleasing results and while the same time also minimizing artifacts.

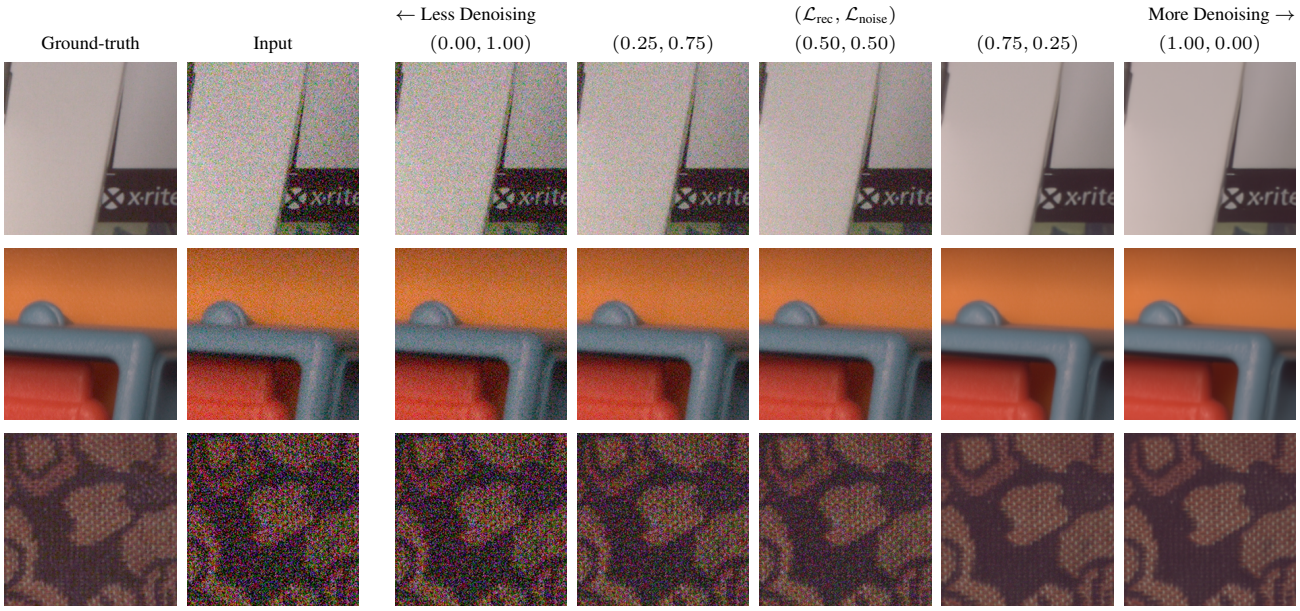

Figure S9. Tuning denoising strength using our tunable NAFNet [S4]. From top to bottom: image 0350, 0900, 1150 from the SIDD validation dataset [S1].

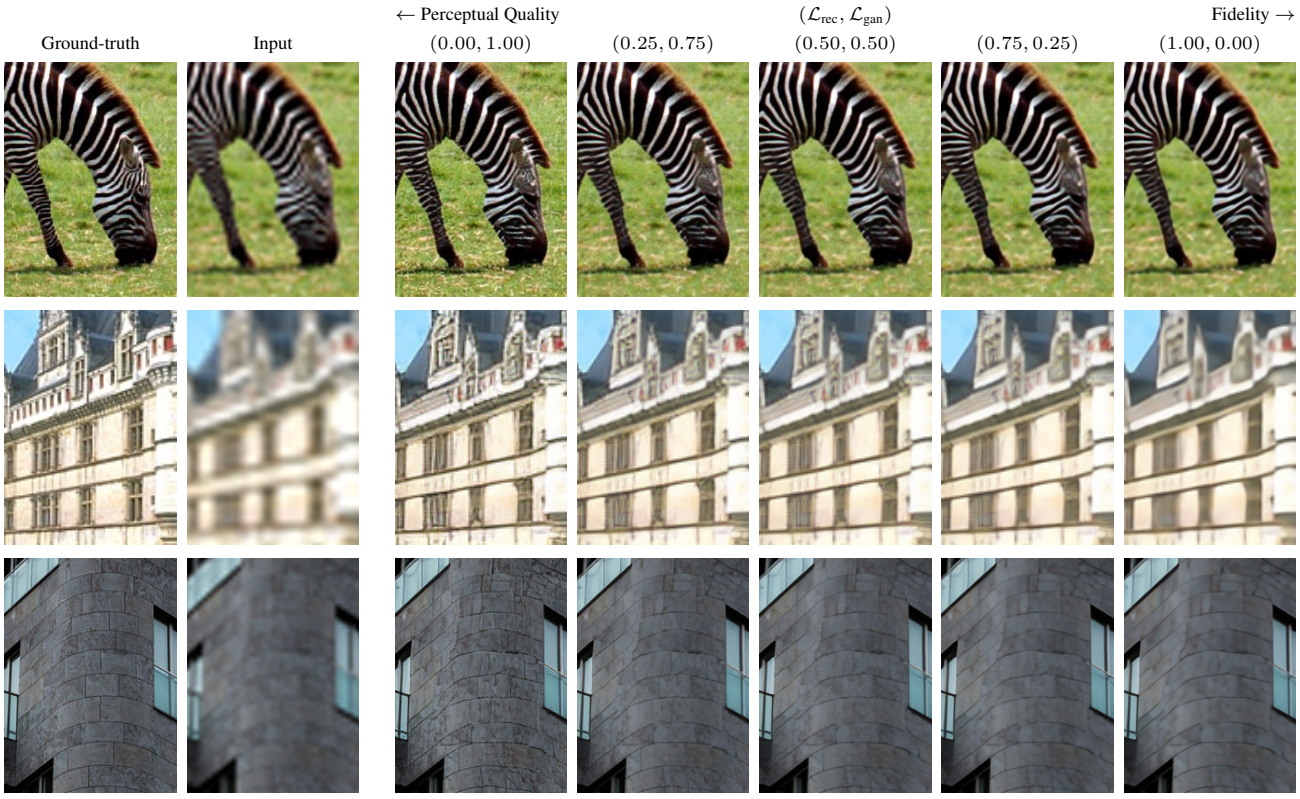

Figure S10. Tuning $\times 4$ super-resolution fidelity and perceptual quality using our tunable SwinIR [S10]. From top to bottom: image zebra from Set14 [S23], 102061 from BSD100 [S13], and img001 from Urban100 [S7].

Figure S11. Tuning style transfer for different combination of parameters controlling *Mosaic*, *Edtaonisl*, and *Kandinsky* influence. From top to bottom: image 02, 09, 15, 22, and 23 from the Kodak [S9] dataset.

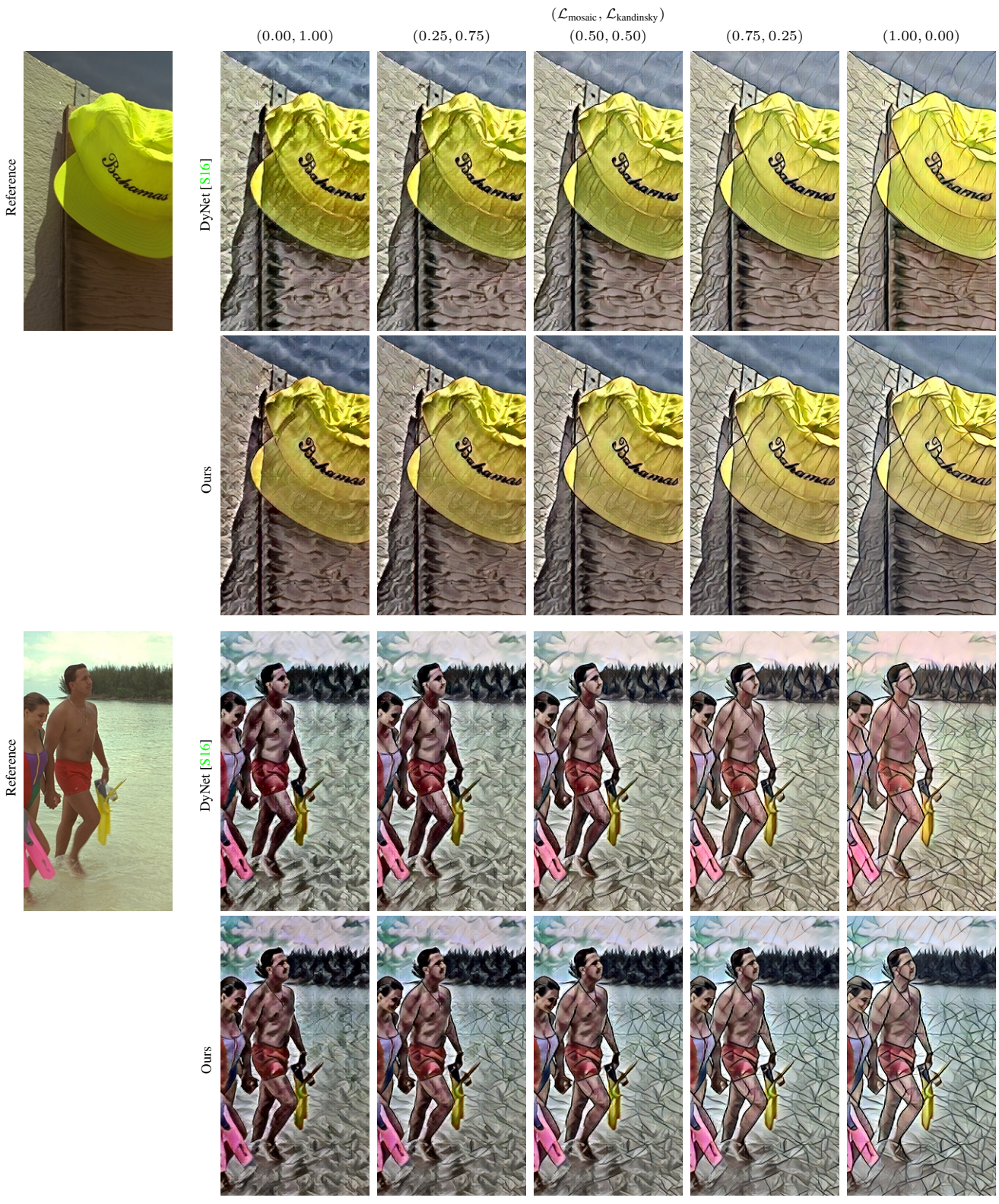

Figure S12. Tuning style transfer for different combination of parameters controlling *Mosaic* and *Kandinsky* influence. From top to bottom: image 03 and 12 from the Kodak [S9] dataset.

Figure S13. Tuning style transfer for different combination of parameters controlling *Mosaic* and *Edtaonisl* influence. From top to bottom: image 13 and 19 from the Kodak [S9] dataset.

References

[S1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. A high-quality denoising dataset for smartphone cameras. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1692–1700, 2018. 5, 10

[S2] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 126–135, 2017. 5

[S3] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *International Conference on Image Processing (ICIP)*, volume 2, pages 168–172, 1994. 5

[S4] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *European Conference on Computer Vision (ECCV)*, pages 17–33, 2022. 5, 10

[S5] Jingwen He, Chao Dong, and Yu Qiao. Interactive multi-dimension modulation with dynamic controllable residual learning for image restoration. In *European Conference on Computer Vision (ECCV)*, pages 53–68, 2020. 4, 5, 7

[S6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 1

[S7] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5197–5206, 2015. 10

[S8] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, pages 694–711, 2016. 1, 3

[S9] Kodak Image Dataset. `http://r0k.us/graphics/kodak/`, 1999. 2, 3, 4, 6, 7, 8, 9, 11, 12, 13

[S10] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. SwinIR: Image restoration using swin transformer. In *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1833–1844, 2021. 5, 10

[S11] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1132–1140, 2017. 1, 5

[S12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. 5

[S13] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE International Conference on Computer Vision*, volume 2, pages 416–423, 2001. 2, 3, 4, 10

[S14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015. 1

[S15] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016. 1

[S16] Alon Shoshan, Roey Mechrez, and Lihi Zelnik-Manor. Dynamic-Net: Tuning the objective without re-training for synthesis tasks. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3214–3222, 2019. 2, 3, 4, 5, 9, 12, 13

[S17] Karen Simonyan and Andrew Zisserman. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 5

[S18] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:2004.10694*, 2016. 1

[S19] Wei Wang, Ruiming Guo, Yapeng Tian, and Wenming Yang. CFSNet: Toward a controllable feature space for image restoration. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4139–4148, 2019. 2, 3, 4, 5, 6

[S20] Xintao Wang, Ke Yu, Chao Dong, Xiaoou Tang, and Chen Change Loy. Deep network interpolation for continuous imagery effect transition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1692–1701, 2019. 2

[S21] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. ESRGAN: Enhanced super-resolution generative adversarial networks. In *European Conference on Computer Vision Workshops (ECCV)*, pages 63–79, 2018. 5

[S22] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 5

[S23] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, pages 711–730, 2012. 10

[S24] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018. 2

[S25] Mosaic Style Image. `https://github.com/Malikanhar/Neural-Style-Transfer/tree/master/style%20image`. [Accessed March 2023]. 5

[S26] Edtaonisl Style Image. `https://github.com/skq024/Real-time-Coherent-Style-Transfer-For-Videos/tree/master/styles`. [Accessed March 2023]. 5

[S27] Kandinsky Style Image. `https://github.com/zehuac/Neural-Style-Transfer-based-on-CNN/tree/master/images`. [Accessed March 2023]. 5