Test-Time Adaptation for Super-Resolution: You Only Need to Overfit on a Few More Images

1. Supplementary Material

1.1. Training details

Generator's architecture The architecture of the generator, based on [3], is shown in Figure 1. The generator is a feed-forward CNN, consisting of convolutional layers and several residual blocks; the low-resolution image I^{LR} is passed through the first convolutional layer with a ReLU activation function and a 64 channel output. This output is subsequently passed through 32 residual blocks. Each block has two convolutional layers with 3×3 filters and 256 channel feature maps. Each one is followed by a ReLU activation. By using a long skip connection, the output of the final residual block is concatenated with the features of the first convolutional layer and is then passed through two upsampling blocks, where each one doubles the size of the feature map. Finally, the result is filtered by the last convolutional layer to get the super-resolved image I^{SR}. This setup aims at upsampling with a scale factor of four; the number of upsampling blocks could be modified based on different scaling factors.

Fine-tuning In Fig. 1, the trainable convolutional layers are highlighted with the yellow box; other parameters are frozen. This has be done specifically to force the fine-tuning to make changes to the filters of the network' feature extractor rather than manipulating the upsampling layers of the network, thereby yielding a plausible solution. The fine-tuning is performed with the mini-batches of 4 images, corresponding to random crops of 32×32 pixels from our constructed dataset. We choose a relatively low learning rate of $1e-4$ for a gradual change in the network parameters.

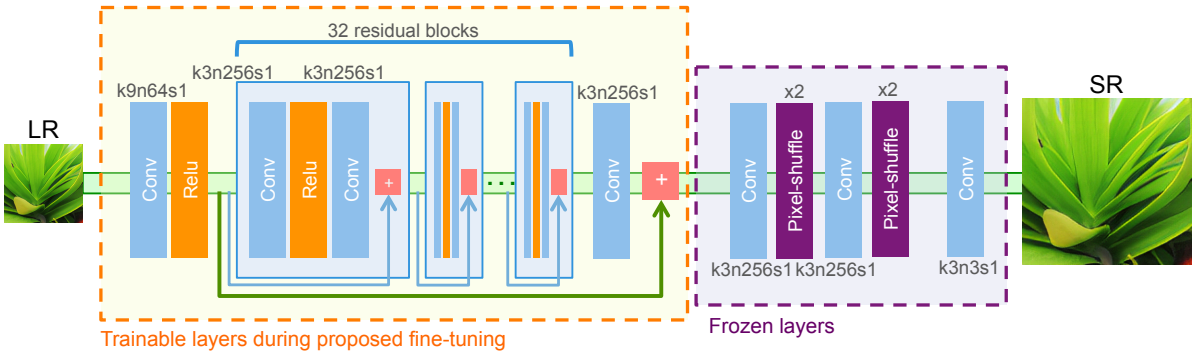

Figure 1. The network architecture of the generator. We highlight (yellow bounding box) the feature extractor layers which have been trained during fine-tuning stage, while keeping the other upsampling layers (purple bounding box) frozen.

Baseline with perceptual loss The generator used in this setting is the same as our PSNR-based approach.

The training is divided into two steps; first, the SR decoder was pre-trained with only the pixel-wise cost function for 20 epochs. Then, for the second step, we continue the training for 35 more epochs with a new loss function containing three loss terms: 1- Pixel-wise loss ($\mathcal{L}_1$), 2- an adversarial loss ($\mathcal{L}_{adv}$), and 3- the perceptual loss function [1] ($\mathcal{L}_{vgg}$) using a layer of the pretrained VGG-19 network [4]. The total loss can be formulated as follows:

$$\mathcal{L}_{total} = \alpha\mathcal{L}_1 + \beta\mathcal{L}_{vgg} + \gamma\mathcal{L}_{adv} \tag{1}$$

where α, β, γ and δ are the corresponding weights of the loss terms used to train our network, and as proposed by [5], were set to $1e-2$, $5e-3$ and 1, respectively. The Adam optimizer [2] was used during both steps. The learning rate was set to $1e-3$ and then has been decayed by a factor of 10 every 20 epochs. We also alternately optimized the discriminator with similar architecture and settings to those proposed by [5].

1.2. Effect of different convolutional layers

In this section, we investigate the effectiveness of using different convolutional layers of the VGG network in our approach. Specifically, we show results using the conv2, conv4, and conv5 layers in Fig. 2. We base our selection on the visual/perceptual quality of the outputs. For example, we found that conv2 and conv5 produced suboptimal results compared to other layers. Ultimately, based on the visual/perceptual quality, we chose to use the conv3 layer.

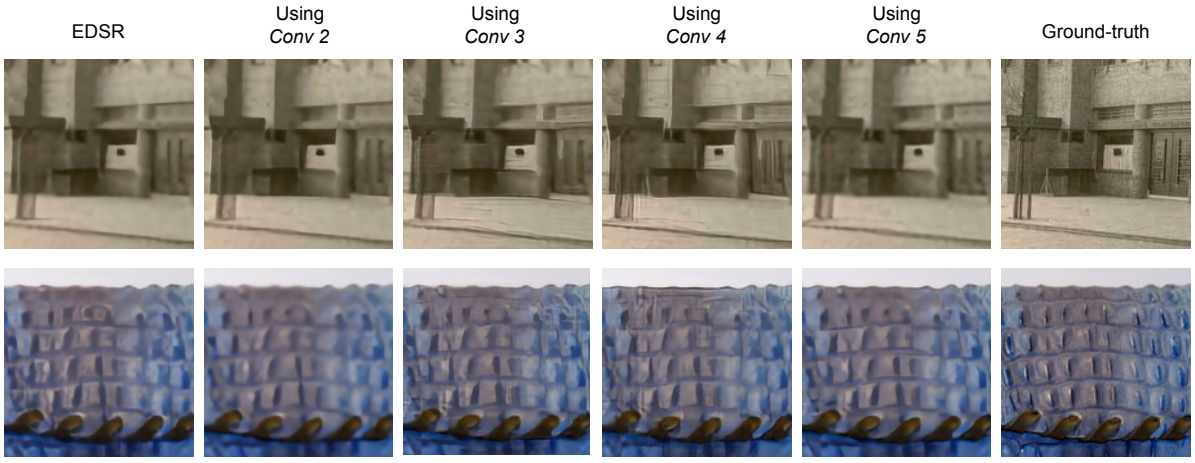

Figure 2. Differences in the perceptual quality obtained at different VGG network layers including conv2, conv3, conv4, and conv5, respectively.

1.3. Best values for K and M

In this section, we go more into detail about how we chose the number of images per filter K to construct our dataset used in the fine-tuning and the number of filters M to consider with respect to the test image. We presented results using $K = 2, M = 5$. We tuned these parameters based on the perceptual quality of the images generated by varying K and M over a range of values. We focused on the best perceptual quality, as some decreases in PSNR/SSIM values are expected. In Fig. 3, we show the results for the combinations generated by $K = 1, 2, 5, 9$ and $M = 1, 2, 5, 10$. We can observe that results obtained by very few images for fine-tuning (e.g. $K = 1$ and $M = 1$) contain artefacts, while increasing both K and M results in more realistic and appealing results ($2 \leq K, M \leq 5$). Finally, we note that increasing both K and M significantly (K, M ¿ 5) produces blurry images, toward same solution as the EDSR baseline.

1.4. User study

To further prove the effectiveness of the proposed fine-tuning approach and its benefits against the baseline model, we performed a subjective evaluation; in particular, 11 participants were asked to choose between three different reconstructions: 1- original model (pre-trained EDSR), 2- fine-tuned on random images, and 3- fine-tuned on images selected by our approach. Each user evaluated 15 random images of ImageNet. To avoid random selection of similar images, a choice as "Cannot decide" was also given. The results are shown in Table 1.4.

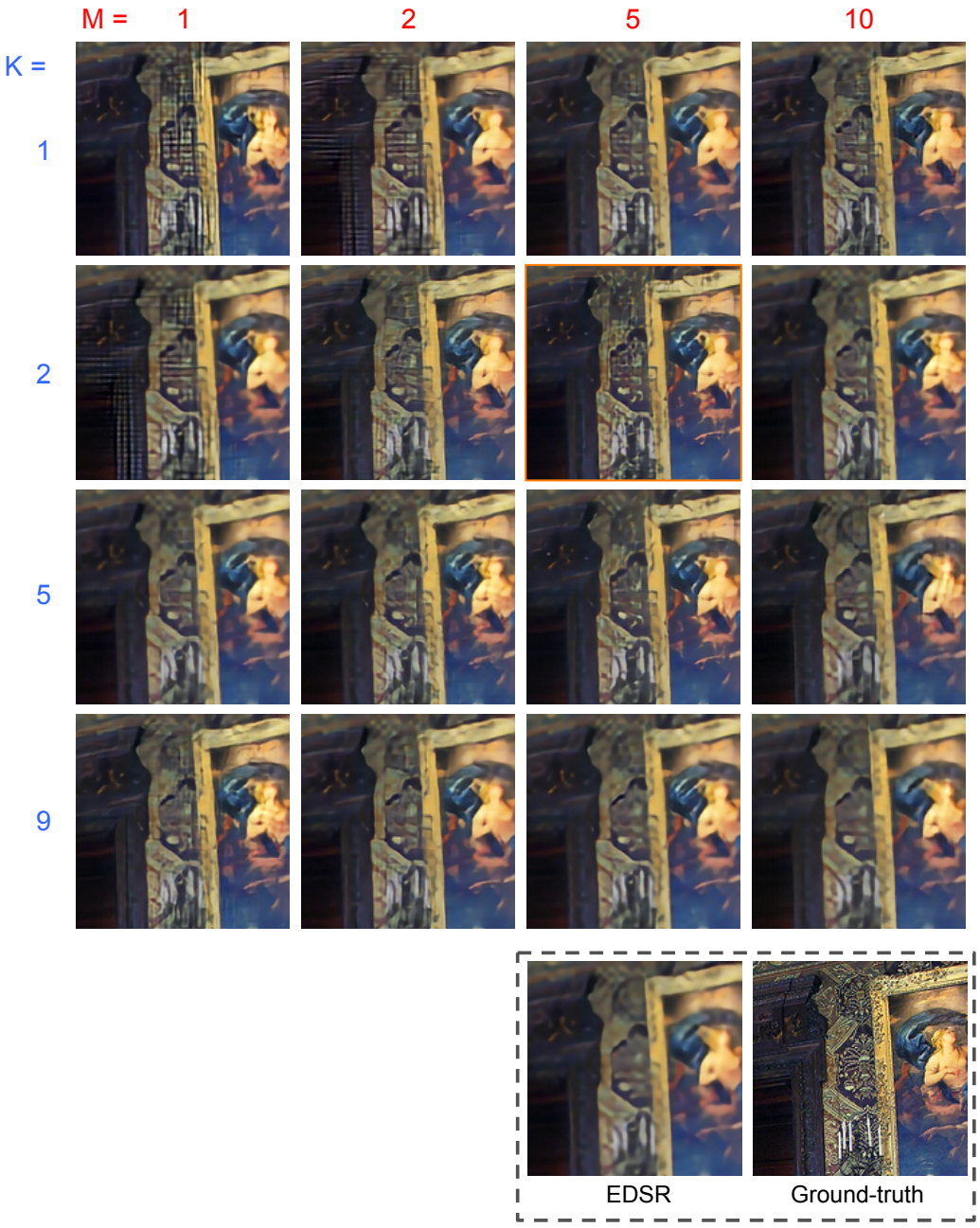

Figure 3. Differences in the perceptual quality obtained with different combinations of K and M.

	Baseline (EDSR)	Fine-tuned (rand. images)	Fine-tuned **(ours)**	Cannot decide
% of votes	9.09	2.42	**71.52**	16.97

Table 1. Results of the user study to further prove the effectiveness of the proposed fine-tuning approach; comparing: 1- original model (pre-trained EDSR), 2- fine-tuned on random images, and 3- fine-tuned on images selected by our approach.

We will expand this study (in terms of number of participant) and will add it in the final manuscript.

1.5. Activation dataset

You can download the activation dataset here:

- Link to Activation dataset - Conv1
- Link to Activation dataset - Conv2
- Link to Activation dataset - Conv3
- Link to Activation dataset - Conv4
- Link to Activation dataset - Conv5

Each of the above links contains the top 9 activated images for the filters from a different convolutional layer of the pre-trained VGG-19 [4] network. We provide both HR version and a LR version which is downsampled using the bicubic downscaling kernel implemented in Matlab.

References

[1] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[2] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

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

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[5] 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 *ECCV Workshops*, 2018.