NAS-DIP: Learning Deep Image Prior with Neural Architecture Search Supplementary Material

Anonymous ECCV submission

Paper ID 3006

Overview

In this supplementary document, we provide additional details and experimental results to complement the main manuscript. We will make the training and evaluation scripts publicly available for reproducible research. Our supplementary material consists of the following.

- We describe the objective for model training for each task. (Section 1)
- We describe the implementation details of the Neural Architecture Search training and model training. (Section 2)
- We describe the details of the exponential sliding window used when evaluating the performance of image denoising. (Section 3)
- We present the searched network architecture for each task and discuss the insights from the discovered architecture. (Section 4)
- We include the analysis of the performance gain over DIP [9]. (Section 5)
- We provide extensive visual comparisons with the state-of-the-art methods on all the tasks in the attached `index.html` file.

1 Objective for Model Training

In this section, we describe the objective functions used for each task.

1.1 Objective function for image super-resolution, inpainting, and denoising

Given an image x in the training set, we first generate a *degenerated* version x_0 by adding noise, downsampling, or dropping certain pixels from x depending on the task of interest. That is, for image denoising, $x_0 \in \mathbb{R}^{H \times W \times 3}$ denotes the noisy version of the clean image x, for single image super-resolution, $x_0 \in \mathbb{R}^{\frac{H}{r} \times \frac{W}{r} \times 3}$ denotes the low-resolution version of x where r represents the downsampling ratio, and for image inpainting, $x_0 \in \mathbb{R}^{H \times W \times 1}$ denotes the occluded version of x. We then sample a noise image $z \in \mathbb{R}^{H \times W \times C}$ and enforce the searched network f_θ to map the noise image z to the denoised, high-resolution, or inpainted version of x_0, i.e., map the noise image z to x.

To achieve this, we follow DIP [9] and optimize different objectives for each task. For single image super-resolution, we train the network using the objective function:

$$\mathcal{L}_{\text{super-resolution}}(\theta) = \|d(f_\theta(z)) - x_0\|_2^2, \tag{1}$$

where $d(\cdot)$ is a downsampling operator that downsamples the network's output $f_\theta(z) \in \mathbb{R}^{H\times W\times 3}$ (i.e., the high-resolution image) to the lower resolution version $d(f_\theta(z)) \in \mathbb{R}^{\frac{H}{r}\times\frac{W}{r}\times 3}$.

For image inpainting, we train the network using the objective function:

$$\mathcal{L}_{\text{inpainting}}(\theta) = \|(f_\theta(z) - x_0)\otimes m\|_2^2, \tag{2}$$

where $f_\theta(z) \in \mathbb{R}^{H\times W\times 1}$ is the inpainted image, m is the binary mask associated with the occluded image x_o, and $\otimes$ is the pixel-wise multiplication between the two operands.

For image denoising, we train the network using the objective function:

$$\mathcal{L}_{\text{denoising}}(\theta) = \|f_\theta(z) - x_0\|_2^2, \tag{3}$$

where $f_\theta(z) \in \mathbb{R}^{H\times W\times 3}$ is the denoised (clean) image.

1.2 Objective function for image dehazing

As mentioned in DoubleDIP [2], a hazy image I can be modeled by three components (1) the airlight map A, (2) the haze-free image J, and (3) the transmission map t:

$$I = t\otimes J + (1-t)\otimes A, \tag{4}$$

where $\otimes$ denotes pixel-wise multiplication operator.

To recover the airlight map A, the haze-free image J, and the transmission map t, we follow DoubleDIP [2] and leverage *three* DIP [9] models f_{θ_A}, f_{θ_J}, f_{θ_t}, each of which recovers one of the three components.

To achieve image dehazing using the DoubleDIP framework [2], we sample three noise images z_t, z_J, and z_A. We then enforce each of the DIP models to map the sampled noise image to the corresponding components, i.e., enforce f_{θ_A} to map z_A to A, f_{θ_J} to map z_J to J, and f_{θ_t} to map z_t to t. Specifically, we follow DoubleDIP [2] and optimize the objective:

$$\mathcal{L}_{\text{dehazing}}(\theta_A,\theta_J,\theta_t) = \Big\| I - \Big(f_{\theta_t}(z_t)\otimes f_{\theta_J}(z_J) + \big(1 - f_{\theta_t}(z_t)\big)\otimes f_{\theta_A}(z_A)\Big)\Big\|_2^2, \tag{5}$$

where $f_{\theta_A}(z_A)$ is the estimated airlight map, $f_{\theta_J}(z_J)$ is the estimated haze-free image, and $f_{\theta_t}(z_t)$ is the estimated transmission map.

1.3 Objective function for matrix factorization

we follow CompMirror [1] and show how to leverage deep image prior for matrix factorization. Specifically, given an image $X \in \mathbb{R}^{h\times w}$, we aim to factorize X into

two components $L \in \mathbb{R}^{h \times q}$ and $T \in \mathbb{R}^{q \times w}$, i.e., $X = LT$. To achieve this, we follow CompMirror [1] and adopt two randomly initialized CNN models f_{θ_L} and f_{θ_T}, each of which aims at recovering one of the factorized components.

To achieve matrix factorization, we first sample two noise images $z_L \in \mathbb{R}^l$ and $z_T \in \mathbb{R}^t$. We then enforce each of the CNN models to map the associated noise image to the corresponding factorized image, i.e., enforce f_{θ_L} to map z_L to L and f_{θ_T} to map z_T to T. We follow CompMirror [1] and optimize the objective:

$$\mathcal{L}_{\text{matrix-factorization}}(\theta_L, \theta_T) = \|f_{\theta_L}(z_L) f_{\theta_T}(z_T) - X\|_2^2, \tag{6}$$

where $f_{\theta_L}(z_L) \in \mathbb{R}^{h \times q}$ and $f_{\theta_T}(z_T) \in \mathbb{R}^{q \times w}$ are the factorized images.

1.4 Objective for unpaired image-to-image translation

For the task of unpaired image-to-image translation, we assume that we are given two sets (domains) of images X and Y, two generators $G_{X \to Y}$ and $G_{Y \to X}$, and two discriminators D_X and D_Y. The two generators aim at translating images from one domain to the other. The two discriminators aim at aligning the distributions between images in the corresponding domain and those translated to that domain.

Following CycleGAN [14], the full training objective $\mathcal{L}$ for achieving the unpaired image-to-image translation task is composed of two loss functions. First, the adversarial loss $\mathcal{L}_{\text{adv}}$ aligns the distributions between the translated images and images in that corresponding domain. Second, the reconstruction loss $\mathcal{L}_{\text{rec}}$ enforces the consistency when translating an image from one domain to the other, followed by a reverse translation (i.e., cycle consistency for self reconstruction). Specifically, the full training objective $\mathcal{L}$ is defined as:

$$\begin{aligned} \mathcal{L} = & \mathcal{L}_{\text{adv}}(X, Y, G_{Y \to X}, D_X) \\ & + \mathcal{L}_{\text{adv}}(Y, X, G_{X \to Y}, D_Y) \\ & + \lambda_{\text{rec}} \cdot \mathcal{L}_{\text{rec}}(X, Y, G_{X \to Y}, G_{Y \to X}), \end{aligned} \tag{7}$$

where λ_{rec} is the hyperparameter used to control the relative importance of the reconstruction loss $\mathcal{L}_{\text{rec}}$.

The adversarial loss $\mathcal{L}_{\text{adv}}$ in the X domain is defined as:

$$\begin{aligned} \mathcal{L}_{\text{adv}}(X, Y, G_{Y \to X}, D_X) = & \mathbb{E}_{x \sim X}[\log(D_X(x))] \\ & + \mathbb{E}_{y \sim Y}[\log(1 - D_X(G_{Y \to X}(y)))]. \end{aligned} \tag{8}$$

Similarly, the adversarial loss in the Y domain is $\mathcal{L}_{\text{adv}}(Y, X, G_{X \to Y}, D_Y)$.

The reconstruction loss $\mathcal{L}_{\text{rec}}$ is defined as:

$$\begin{aligned} \mathcal{L}_{\text{rec}}(X, Y, G_{X \to Y}, G_{Y \to X}) = & \mathbb{E}_{x \sim X}[x - G_{Y \to X}(G_{X \to Y}(x))] \\ & + \mathbb{E}_{y \sim Y}[y - G_{X \to Y}(G_{Y \to X}(y))]. \end{aligned} \tag{9}$$

Following CycleGAN [14], we set $\lambda_{\text{rec}} = 10$ in the unpaired image-to-image translation experiments.

2 Details of NAS Training and Model Training

To search for the optimal network architecture f_θ^* for the task of interest (e.g., single image super-resolution), we leverage existing neural architecture search techniques (an RNN-based controller trained with reinforcement learning) [4,5,15] and adopt PSNR as our reward to search for an improved network structure on a held-out training set.

We denote the parameters of the RNN controller as ϕ and the parameters of the candidate network architecture as θ. The training process is composed of two alternating phases: 1) train the model weights θ of the candidate network architecture with weights ϕ of the RNN controller fixed and 2) train ϕ of the RNN controller with θ of the candidate network architecture fixed. We summarize the training algorithm in Algorithm 1.

In the first phase, we train the parameters θ of the candidate network architecture with parameters ϕ of the RNN controller fixed. First, the RNN controller samples a candidate network architecture f_θ with random initialization. We then train the sampled candidate model with Equation 1 (for single image super-resolution), Equation 2 (for image inpainting), or Equation 3 (for image denoising).

In the second phase, we train the parameters ϕ of the RNN controller with the parameters θ of the candidate network architecture fixed. We first compute the PSNR between the restored prediction and the ground truth as the reward and apply reinforcement learning to update the RNN controller.

The RNN controller is updated using the Adam optimizer via REINFORCE, while the sampled candidate model is updated using the SGD optimizer. We set the number of epochs to $1,000$. For single image super-resolution and image denoising, we set the number of iterations for each image to $6,000$. For image inpainting, the number of iterations for each image is set to $12,000$.

3 Exponential Sliding Window

For image denoising, we follow DIP [9] and average the restored predictions $\{x_t\}_{t=1}^{100}$ obtained in the last 100 iterations with an exponential sliding window (moving average) to obtain the final result x^*:

$$x^* = \sum_{t=1}^{100} x_t \cdot \gamma^{101-t} \cdot (1-\gamma)^{t-1}, \tag{10}$$

where x_t is the restored prediction obtained at the t^{th} iteration, and γ is the weight. We follow DIP [9] and set $\gamma = 0.99$.

4 Searched Architectures

We present and discuss the searched architecture for each task in this section.

Algorithm 1: Pseudo code of NAS-DIP.

```
for epoch i ∈ 1 ... num_of_epochs do
    for each noise image j ∈ 1 ... num_of_images do
        Randomly initialize the parameters θ of the sampled model.
        for iter k ∈ 1 ... num_of_iterations do
            Fix the parameters φ of the RNN controller.
            Optimize the sampled candidate model with Equation (1), (2), or (3),
             depending on the task of interest.
            if iter % 1000 == 0 then
                Fix the parameters θ of the sampled candidate model.
                Compute the PSNR between the restored prediction and the
                 ground truth as the reward.
                Free the parameters φ of the RNN controller.
                Use reinforcement learning with the computed reward to update
                 the RNN controller.
                Sample a new candidate model with random initialization.
```

4.1 Searched upsampling cell

- **Image super-resolution**: bicubic upsampling → depth-wise convolution [3, 6, 7] with a kernel size of 5×5 → LeakyReLU with slope 0.2.
- **Image denoising**: nearest neighbor upsampling → add every N consecutive channels [10, 11] with a kernel size of 5×5 → LeakyReLU with slope 0.2.
- **Image inpainting**: bicubic upsampling → transposed convolution with a kernel size of 7×7 → LeakyReLU with slope 0.2.

There are several observations from the searched upsampling cells. First, the optimal upsampling cell for each task is different. Second, all three upsampling cells choose to use a LeakyReLU with a slope of 0.2 as the activation. This suggests that avoiding "dying ReLU" is important for image restoration problems. Third, only the task of denoising adopts the nearest neighbor upsampling. We believe this is because bicubic upsampling may produce overly smoothed feature maps.

4.2 Searched cross-scale residual connections

Figure 1 presents the searched cross-scale residual connections for each task. From left to right are the discovered cross-level feature connections for single image super-resolution, image denoising, and image inpainting. There are a couple of interesting cross-level connections discovered by the neural architecture search process.

Image super-resolution. For the single image super-resolution task, we observe that there is an additional connection from a feature map in the encoder to a *lower resolution layer* in the decoder. This could potentially be an effective

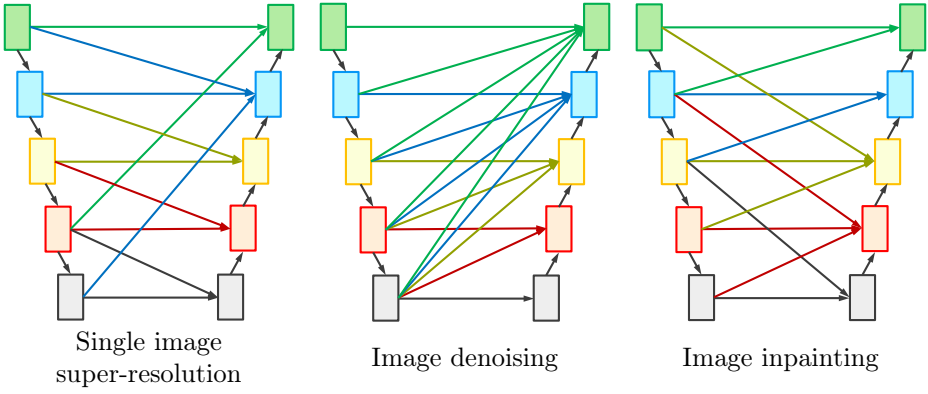

Fig. 1: **The discovered pattern of cross-scale residual connections.** Figures from left to right present the discovered pattern of cross-scale residual connections for single image super-resolution, image denoising, and image inpainting, respectively.

strategy for super-resolving feature maps by reusing higher-level features from the encoder (not just from the features at the same level as in the standard U-Net architecture).

Image denoising. For the image denoising task, we observe that all the lower level feature maps in the encoder are reused in all the other (same or higher) levels in the decoder. This empirical discovery shares a similar high-level spirit with several designs in object detection and semantic segmentation architectures. In particular, the fusion of lower level features with higher-level ones resembles the feature pyramid network [8] and pyramid pooling module [13].

Image inpainting. For the image inpainting task, the searched architecture also shows an interesting connection pattern that combines features from *both* higher level and lower level feature maps from the encoder (in addition to the same-level connection in U-Net).

Table 1: **Comparison to DIP [9].** We report the number of parameters and the average PSNR results on the Set14 dataset [12] with 8× scaling factor.

Method	# params	Avg. PSNR
DIP [9]	1.3M	24.15
DIP-large [9]	1.78M	24.31
Ours	1.8M	24.59

5 Analysis of the Performance Gain over Deep Image Prior

As our searched architecture contains more number of parameters than that in the DIP [9] model, one natural question is: does the performance improvement shown by our architecture solely depend on the increased number of parameters?

To address this question, we increase the number of parameters in DIP [9] by increasing the number of channels in the decoder of DIP from $\{128, 128, 128, 128, 128\}$ to $\{256, 256, 256, 256, 256\}$. We denote this model as DIP-large. With this modification, the number of parameters of the DIP-large model is comparable to our model.

Table 1 reports the experimental results on the Set14 dataset [12] with $8\times$ scaling factor. As we increase the model capacity of the DIP model, we do observe that the performance improves from 24.15 to 24.31 (PSNR). However, the improvement is not as significant as our approach. The result suggests that the performance gain lies in the *design* of the network structure, i.e., the upsampling cell and the cross-scale residual connections, not solely the number of network parameters.

References

1. Aittala, M., Sharma, P., Murmann, L., Yedidia, A., Wornell, G., Freeman, B., Durand, F.: Computational mirrors: Blind inverse light transport by deep matrix factorization. In: NeurIPS (2019)
2. Gandelsman, Y., Shocher, A., Irani, M.: "double-dip": Unsupervised image decomposition via coupled deep-image-priors. In: CVPR (2019)
3. Gao, H., Wang, Z., Ji, S.: Channelnets: Compact and efficient convolutional neural networks via channel-wise convolutions. In: NeurIPS (2018)
4. Ghiasi, G., Lin, T.Y., Le, Q.V.: Nas-fpn: Learning scalable feature pyramid architecture for object detection. In: CVPR (2019)
5. Gong, X., Chang, S., Jiang, Y., Wang, Z.: Autogan: Neural architecture search for generative adversarial networks. In: ICCV (2019)
6. Guo, J., Li, Y., Lin, W., Chen, Y., Li, J.: Network decoupling: From regular to depthwise separable convolutions. In: BMVC (2018)
7. Guo, Y., Li, Y., Wang, L., Rosing, T.: Depthwise convolution is all you need for learning multiple visual domains. In: AAAI (2019)
8. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
9. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Deep image prior. In: CVPR (2018)
10. Wojna, Z., Ferrari, V., Guadarrama, S., Silberman, N., Chen, L.C., Fathi, A., Uijlings, J.: The devil is in the decoder: Classification, regression and gans. IJCV (2019)
11. Wojna, Z., Uijlings, J.R.R., Guadarrama, S., Silberman, N., Chen, L.C., Fathi, A., Ferrari, V.: The devil is in the decoder. In: BMVC (2017)
12. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: International conference on curves and surfaces (2010)
13. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR (2017)
14. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV (2017)
15. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: ICLR (2017)