Deep Network Interpolation for Continuous Imagery Effect Transition

Xintao Wang[1] Ke Yu[1] Chao Dong[2] Xiaoou Tang[1] Chen Change Loy[3]
[1]CUHK - SenseTime Joint Lab, The Chinese University of Hong Kong
[2]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[3]Nanyang Technological University, Singapore
{wx016, yk017, xtang}@ie.cuhk.edu.hk chao.dong@siat.ac.cn ccloy@ntu.edu.sg

Figure 1: Deep network interpolation is capable of generating continuous imagery effect transitions. (1*st row*) from MSE effect to GAN effect in super-resolution; (2*nd row*) from Van Gogh style to Ukiyo-e style; (3*rd row*) from day photo to night one; (4*th row*) from deep depth of field (DoF) to shallow one. More applications are provided in Sec. 4. (**Zoom in for best view**)

Abstract

Deep convolutional neural network has demonstrated its capability of learning a deterministic mapping for the desired imagery effect. However, the large variety of user flavors motivates the possibility of continuous transition among different output effects. Unlike existing methods that require a specific design to achieve one particular transition (e.g., *style transfer), we propose a simple yet universal approach to attain a smooth control of diverse imagery effects in many low-level vision tasks, including image restoration, image-to-image translation, and style transfer. Specifically, our method, namely Deep Network Interpolation (DNI), applies linear interpolation in the parameter space of two or more correlated networks. A smooth control of imagery effects can be achieved by tweaking the interpolation coefficients. In addition to DNI and its broad applications, we also investigate the mechanism of network interpolation from the perspective of learned filters.*

1. Introduction

Deep convolutional neural (CNN) network has achieved a great success in many low-level vision tasks, such as image restoration [3, 18, 20, 1, 44, 2, 28], image-to-image translation [15, 38, 26, 47] and image style transfer [8, 17, 6]. For each specific task, the deep network learns a deterministic mapping and outputs a fixed image for the same inputs. However, one determined output is unable to satisfy diverse user flavors and meet the needs in various scenarios, limiting the applicability for practical use.

In many real-world applications, it is desired to have a smooth control for continuous transition among different output effects. For instance, 1) in super-resolution, models trained with the mean-square-error (MSE) loss [35] tend to produce over-smooth images while those with the generative adversarial network (GAN) [20] generate vivid details but with some unpleasant noise (*e.g.*, Fig 1, 1*st* row). A balanced result between these two different effects would

be more visual-pleasing with reduced artifacts. 2) Many image restoration tasks deal with multiple degradation levels, such as different noise levels and blur kernels. Most existing methods can only handle limited degradation levels. It is costly to train lots of models for continuous degradation levels in practice. Thus, a model with the flexibility of adjusting the restoration strength would expand the application coverage. 3) In artistic manipulation like image-to-image translation and image style transfer, different users have different aesthetic flavors. Achieving a smooth control for diverse effects with a sliding bar are appealing in these applications.

Several approaches have been proposed to improve the CNN's flexibility for producing continuous transitions in different tasks. Take image style transfer as an example, adaptive scaling and shifting parameters are used in instance normalization layers [6, 12] for modeling different styles. Interpolating these normalization parameters for different styles produces the combination of various artistic styles. In order to further control the stroke size in the stylized results, a carefully-designed pyramid structure consisting of several stroke branches are proposed [16]. Though these methods are able to realize continuous transition, there are several drawbacks: 1) These careful designs are problem-specific solutions, lacking the generalizability to other tasks. 2) Modifications to existing networks are needed, thus complicate the training process. 3) There is still no effective and general way to solve the smooth control in tasks like balancing MSE and GAN effects in super-resolution.

In this paper, we address these drawbacks by introducing a more general, simple but effective approach, known as *Deep Network Interpolation* (DNI). Continuous imagery effect transition is achieved via linear interpolation in the *parameter space* of existing trained networks. Specifically, provided with a model for a particular effect $\mathcal{A}$, we fine-tune it to realize another relevant effect $\mathcal{B}$. DNI applies linear interpolation for all the corresponding parameters of these two deep networks. Various interpolated models can then be derived by a controllable interpolation coefficient. Performing feed-forward operations on these interpolated models using the same input allows us to outputs with a continuous transition between the different effects $\mathcal{A}$ and $\mathcal{B}$.

Despite its simplicity, the proposed DNI can be applied to many low-level vision tasks. Some examples are presented in Fig 1. Extensive applications showcased in Sec. 4 demonstrate that deep network interpolation is *generic* for many problems. DNI also enjoys the following merits. 1) The transition effect is *smooth* without abrupt changes during interpolation. The transition can be easily controlled by an interpolation coefficient. 2) The linear interpolation operation is *simple*. No network training is needed for each transition and the computation for DNI is negligible. 3) DNI is *compatible* with popular network structures, such as VGG [32], ResNet [10] and DenseNet [11].

Our main contribution in this work is the novel notion of interpolation in parameter space, and its application in low-level vision tasks. We demonstrate that interpolation in the parameter space could achieve much better results than mere pixel interpolation. We further contribute a systematic study that investigates the mechanism and effectiveness of parameter interpolation through carefully analyzing the filters learned.

2. Related Work

Image Restoration. CNN-based approaches have led to a series of breakthroughs for several image restoration tasks including super-resolution [3, 18, 25, 19, 34, 20], denoising [1, 44], de-blocking [41, 7] and deblurring [43, 33, 28]. While most of the previous works focus on addressing one type of distortion without the flexibility of adjusting the restoration strength, there are several pioneering works aiming to deal with various practical scenarios with controllable "hyper-parameters". Zhang *et al*. [45] adopt CNN denoisers to solve image restoration tasks by manually selecting the hyper-parameters in a model-based optimization framework. However, a bank of discriminative CNN denoisers are required and the hyper-parameter selection in optimization is not a trivial task [4]. SRMD [46] proposes an effective super-resolution network handling multiple degradations by taking an degradation map as extra inputs. However, the employed dimensionality stretching strategy is problem-specific, lacking the generalizability to other tasks.

Image Style Transfer. Gatys *et al*. [8] propose the neural style transfer algorithm for artistic stylization. A number of methods are developed to further improve its performance and speed [36, 17, 21]. In order to model various/arbitrary styles in one model, several techniques are developed, including conditional instance normalization [6], adaptive instance normalization [12, 9] and whitening and coloring transforms [23]. These carefully-designed approaches are also able to achieve user control. For example, interpolating the normalization parameters of different styles produces the combination of various artistic styles. The balance of content and style can be realized by adjusting the weights of their corresponding features during mixing. In order to control the stroke size in the stylized results, a specially designed pyramid structure consisting of several stroke branches are further proposed [16]. In these studies, different controllable factors require specific structures and strategies.

Image-to-image Translation. Image-to-image translation [15, 38, 26, 47, 13] aims at learning to translate an image from one domain to another. For instance, from landscape photos to Monet paintings, and from smartphone snaps to professional DSLR photographs. These methods

can only transfer an input image to a specific target manifold and they are unable to produce continuous translations. The controllable methods proposed in image restoration and image style transfer are problem-specific and cannot be directly applied to the different image-to-image translation task. On the contrary, the proposed DNI is capable of dealing with all these problems in a general way, regardless of the specific characteristics of each task.

Interpolation. Instead of performing parameter interpolation, one can also interpolate in the pixel space or feature space. However, it is well known that interpolating images pixel by pixel introduces ghosting artifacts since natural images lie on a non-linear manifold [42]. Upchurch *et al*. [37] propose a linear interpolation of pre-trained deep convolutional features to achieve image content changes. This method requires an optimization process when inverting the features back to the pixel space. Moreover, it is mainly designed for transferring facial attributes and not suitable for generating continuous transition effects for low-level vision tasks. More broadly, several CNN operations in the input and feature space have been proposed to increase the model's flexibility. Concatenating extra conditions to inputs [46] or to middle features [22] alters the network behavior in various scenes. Modulating features with an affine transformation [29, 5, 39] is able to effectively incorporate other information. Different from these works, we make an attempt to investigate the manipulation in the parameter space.

3. Methodology

3.1. Deep Network Interpolation

Many low-level vision tasks, *e.g*., image restoration, image style transfer, and image-to-image translation, aim at mapping a corrupted image or conditioned image x to the desired one y. Deep convolutional neural networks are applied to directly learn this mapping function G_θ parametrized by θ as $y = G_\theta(x)$.

Consider two networks G^A and G^B with the same structure, achieving different effects $\mathcal{A}$ and $\mathcal{B}$, respectively. The networks consist of common operations such as convolution, up/down-sampling and non-linear activation. The parameters in CNNs are mainly the weights of convolutional layers, called *filters*, filtering the input image or the precedent features. We assume that their parameters θ_A and θ_B have a "strong correlation" with each other, *i.e*., the filter orders and filter patterns in the same position of G^A and G^B are similar. This could be realized by some constraints like fine-tuning, as will be analyzed in Sec. 3.2. This assumption provides the possibility for meaningful interpolation.

Our aim is to achieve a continuous transition between the effects $\mathcal{A}$ and $\mathcal{B}$. We do so by the proposed Deep Network Interpolation (DNI). DNI interpolates all the corresponding parameters of these two models to derive a new interpolated model G^{interp}, whose parameters are:

$$\theta_{interp} = \alpha\,\theta_A + (1-\alpha)\,\theta_B, \tag{1}$$

where $\alpha \in [0, 1]$ is the interpolation coefficient. Indeed, it is a linear interpolation of the two parameter vectors θ_A and θ_B. The interpolation coefficient α controls a balance of the effect $\mathcal{A}$ and $\mathcal{B}$. By smoothly sliding α, we achieve continuous transition effects without abrupt changes.

Generally, DNI can be extended for N models, denoted by $G^1, G^2, ..., G^N$, whose parameters have a "close correlation" with each other. The DNI is then formulated as:

$$\theta_{interp} = \alpha_1\theta_1 + \alpha_2\theta_2 + ... + \alpha_N\theta_N, \tag{2}$$

where α_i satisfy $\alpha_i \geq 0$ and $\alpha_1 + \alpha_2 + \cdots + \alpha_N = 1$. In other words, it is a convex combination of the parameter vectors $\theta_1, \theta_2, ..., \theta_N$. By adjusting $(\alpha_1, \alpha_2, ..., \alpha_N)$, a rich and diverse effects with continuous transitions could be realized.

The interpolation is performed on all the layers with parameters in the networks, including convolutional layers and normalization layers. Convolutional layers have two parameters, namely *weights* (filters) and *biases*. The biases are added to the results after filtering operation with filters. Apart from the weights, DNI also operates on the biases, since the added biases influence the successive non-linear activation.

Batch normalization (BN) layers [14] have two kinds of parameters. 1) The statistics *running_mean* and *running_var* track the mean and variance of the whole dataset during training and are then used for normalization during evaluation. 2) the learned parameters γ and β are for a further affine transformation. During inference, all these four parameters actually could be absorbed into the precedent or successive convolutional layers. Thus, DNI also performs on normalization parameters. Instance normalization (IN) has a similar behavior as BN, except that IN uses instance statistics computed from input data in both training and evaluation. We take the same action as that for BN. In practice, the interpolation is performed on not only the weights but also the biases and further normalization layers. We believe a better interpolation scheme considering the property of different kinds of parameters is worthy of exploiting.

It is worth noticing that the choice of the network structure for DNI is flexible, as long as the structures of models to be interpolated are kept the same. Our experiments on different architectures show that DNI is compatible with popular network structures such as VGG [32], ResNet [10] and DenseNet [11]. We also note that the computation of DNI is negligible. The computation is only proportional to the number of parameters.

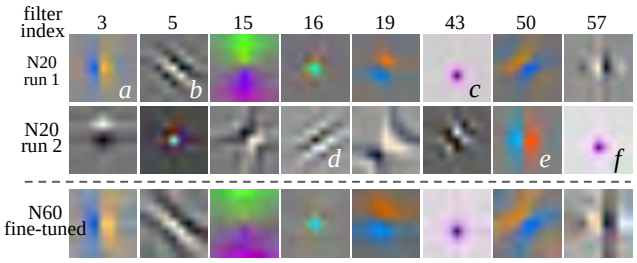


Figure 2: Filter correlations. The first two rows are the filters of different runs (both from scratch) for the denoising (N20) task. The filter orders and filter patterns in the same position are different. The *fine-tuned* model (N60) ($3rd$ row) has a "strong correlation" to the pre-trained one ($1st$ row).

3.2. Understanding Network Interpolation

We attempt to gain more understanding on network interpolation from some empirical studies. From our experiments, we observe that: 1) Fine-tuning facilitates high correlation between parameters of different networks, providing the possibility for meaningful interpolation. 2) Fine-tuned filters for a series of related tasks present continuous changes. 3) Our analyses show that interpolated filters could fit the actual learned filters well. Note that our analyses mainly focus on *filters* since most of the parameters in CNNs are in the form of filters.

We present our main observations with a representative denoising task and focus on increasing noise levels with N20, N30, N40, N50, and N60, where N20 denotes the Gaussian noise with zero mean and variance 20. In order to better visualize and analyze the filters, we adopt a three-layer network similar to SRCNN [3], where the first and last convolutional layers have 9×9 filter size. Following the notion of [3], the first and last layer layers can be viewed as a *feature extraction* and *reconstruction* layer, respectively.

Fine-tuning for inter-network correlation. Even for the same task like denoising with the N20 level, if we simply train two models from scratch, the filter orders among channels and filter patterns in the corresponding positions could be very different (Fig. 2). However, a core representation is shared between these two networks [24]. For instance, in Fig. 2, filer c is identical to filter f; filter a and filter e have a similar pattern but with different colors; filter b is a inverted and rotated counterpart of filter d.

Fine-tuning, however, can help to maintain the filters's order and pattern. To show this, we fine-tune a pre-trained network (N20) to a relevant task (N60). It is observed that the filter orders and filter patterns are maintained (Fig. 2). The "high correlation" between the parameters of these two networks provides the possibility for meaningful interpolation. We note that besides fine-tuning, other constraints such as joint training with regularization may also achieve such inter-network correlation.

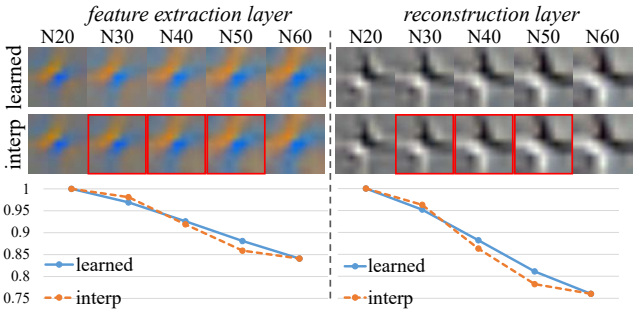


Figure 3: Filters with gradual changes. We show one representative filter for each layer. **$1st$ row**: The fine-tuned filters for different noise level show gradual changes. **$2nd$ row**: The interpolated filters (with red frames) from the N20 and N60 filters could visually fit those learned filters well. **$3rd$ row**: The correlation curves for learned and interpolated filters are also very close.

Learned filters for related tasks exhibit continuous changes. When we fine-tune several models for relevant tasks (N30, N40, N50, and N60) from a pre-trained one (N20), the correspondingly learned filters have intrinsic relations with a smooth transition. As shown in Fig. 3 ($1st$ row), the trained filters show gradual changes as the noise level increases. Apart from visualization, we also calculate a correlation index ρ_{ij} to measure the correlations of filters F_i and F_j:

$$\rho_{ij} = \frac{(F_i - \overline{F_i}) \cdot (F_j - \overline{F_j})}{\sqrt{\|F_i - \overline{F_i}\|_2} \sqrt{\|F_j - \overline{F_j}\|_2}}. \quad (3)$$

We choose this form (similar to the Pearson correlation) since it ignores the scale and shift influences and focus on the filter pattern itself. We calculate the correlation index for each filter with the first N20 filter and plot the curve (blue curve in Fig. 3). The results suggest a close relationship among the learned filters, exhibiting a gradual change as the noise level increases.

The interpolated filters fit the learned filters well. The continuous changes of learned filters suggest that *it it possible to obtain the intermediate filters by interpolating the two ends*. To further verify this observation, we perform linear interpolation between the filters from the N20 and N60 models. With optimal coefficients α, the interpolated filters could visually fit those learned filters ($2nd$ row with red frames, Fig. 3). We further calculate the correlation index for each interpolated filter with the first N20 filter. The correlation curves for learned and interpolated filters are also very close.

The optimal α is obtained through the final performance of the interpolated network. Specifically, we perform DNI with α from 0 to 1 with an interval of 0.05. The best α for each noise level is selected based on which that makes the interpolated network to produce the highest PSNR on the test dataset.

Discussion. It is noteworthy that similar observations could also be found in several other tasks, such as super-resolution with different kernels and JPEG artifacts removal with different compression levels. We provide the details in the Appendix A.

The analysis above is by no means complete, but it gives a preliminary explanation behind the DNI from the filter perspective. As the network goes deeper, the non-linearity increases and the network behaviors become more complicated. However, we still observe a similar phenomenon for deeper networks. Since filter visualization is difficult in deep networks, which typically designed with a stack of convolutional layers of 3×3 kernels, we adopt the correlation index (Eq. 3) to analyze the filter correlations among models for a series of noise levels. We employ DnCNN [44] with 17 layers and analyze the 5th (the front) and 12th (the back) convolutional layers. In Fig. 4, the correlation curves show the median of correlation indexes w.r.t. the first N20 model and the correlation distribution are also plotted. Besides the high correlation among these models, we can also observe gradual changes as the noise level increases. Furthermore, the front and the back convolution layers present a similar transition trend, even their distributions highly coincide.

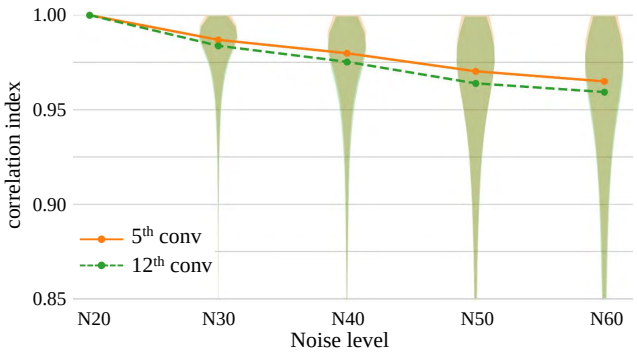


Figure 4: Filter correlation for deeper denoising networks. We show a front ($5th$) and a back ($12th$) layer. The curves present the median of correlation indexes and the correlation distribution are also plotted. Note that the two distributions highly coincide. Zoom in for subtle differences.

4. Applications

We experimentally show that the proposed DNI can be applied to extensive low-level vision tasks, *e.g.*, image restoration (Sec. 4.1), image-to-image translation (Sec. 4.2) and image style transfer (Sec. 4.3). Another example of smooth transitions on face attributes are presented in Sec. 4.4, indicating its potential for semantic changes. Due to space limitations, more examples and analyses are provided in the Appendix C and the project page[1].

[1] https://xinntao.github.io/projects/DNI

4.1. Image Restoration

Balance MSE and GAN effects in super-resolution. The aim of super-resolution is to estimate a high-resolution image from its low-resolution counterpart. A super-resolution model trained with MSE loss [35] tends to produce over-smooth images. We fine-tune it with GAN loss and perceptual loss [20], obtaining results with vivid details but always together with unpleasant artifacts (*e.g.*, the eaves and water waves in Fig. 5). We use dense blocks [11, 40] in the network architecture and the MATLAB bicubic kernel with a scaling factor of 4 is adopted as the down-sampling kernel.

As presented in Fig. 5, DNI is able to smoothly alter the outputs from the MSE effect to the GAN effect. With appropriate interpolation coefficient, it produces visually-pleasing results with largely reduced artifacts while maintaining the textures. We also compare it with pixel interpolation, *i.e.*, interpolating the output images pixel by pixel. However, the pixel interpolation is incapable of separating the artifacts and details. Taking the water waves for example (Fig. 5), the water wave texture and artifacts simultaneously appear and enhance during the transition. Instead, DNI first enhances the vivid water waves without artifacts and then finer textures and undesirable noise appears successively. The effective separation helps to remove the displeasing artifacts while keeping the favorable textures, superior to the pixel interpolation. This property could also be observed in the transition of animal fur in Fig. 5, where the main structure of fur first appears followed by finer structures and subtle textures.

One can also obtain several models for different mixture effects, by tuning the weights of MSE loss and GAN loss during training. However, this approach requires tuning on the loss weights and training many networks for various balances, and thus it is too costly to achieve continuous control.

Adjust denoising strength. The goal of denoising is to recover a clean image from a noisy observation. In order to satisfy various user demands, most popular image editing softwares (*e.g.*, Photoshop) have controllable options for each tool. For example, the noise reduction tool comes with sliding bars for controlling the denoising strength and the percentage of preserving or sharpening details.

We show an example to illustrate the importance of adjustable denoising strength. We are provided with a denoising model specialized in addressing a specific Gaussian noise level N40. We use DnCNN [44] as our implementation. As shown in Fig. 6, however, the determined outputs (with yellow frames) are not satisfactory due to the different imagery contents. In particular, the denoising strength for the grass is too strong, producing over-smooth results, while in the smooth sky region, it requires a larger strength to remove the undesired artifacts.

Existing deep-learning based approaches fail to meet this

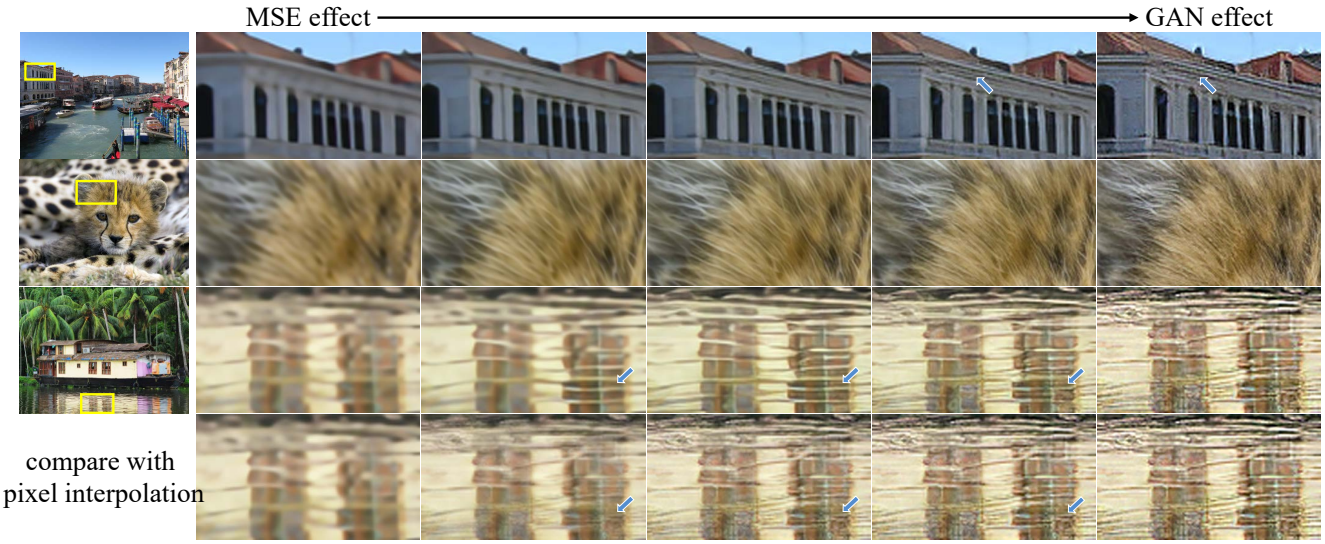


Figure 5: Balancing the MSE and GAN effects with DNI in super-resolution. The MSE effect is over-smooth while the GAN effect is always accompanied with unpleasant artifacts (*e.g.*, the eaves and water waves). DNI allows smooth transition from one effect to the other and produces visually-pleasing results with largely reduced artifacts while maintaining the textures. In contrast, the pixel interpolation strategy fails to separate the artifacts and textures. (**Zoom in for best view**)

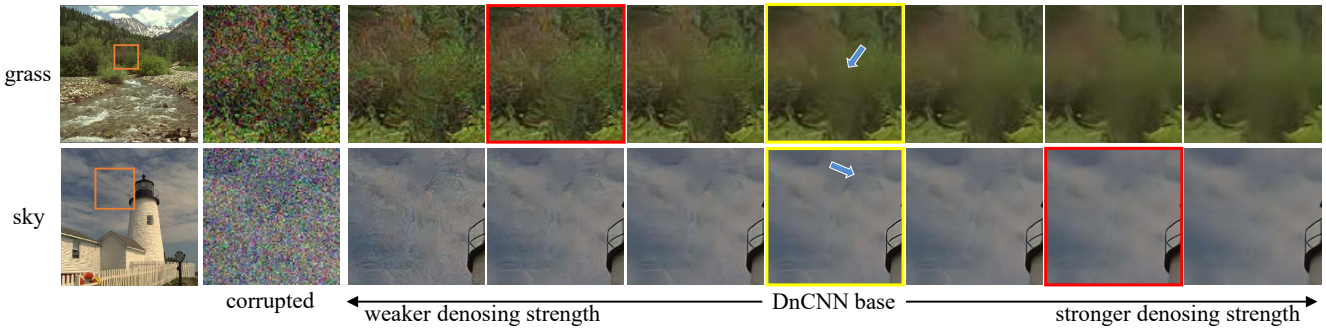


Figure 6: Adjustable denoising strength with DNI. One model without adjustment (with yellow frames) is unable to balance the noise removal and detail preservation. For the grass, a weaker denoising strength could preserve more textures while for the sky, the stronger denoising strength could obtain an artifact-free result, improving the visual quality (with red frames). (**Zoom in for best view**)

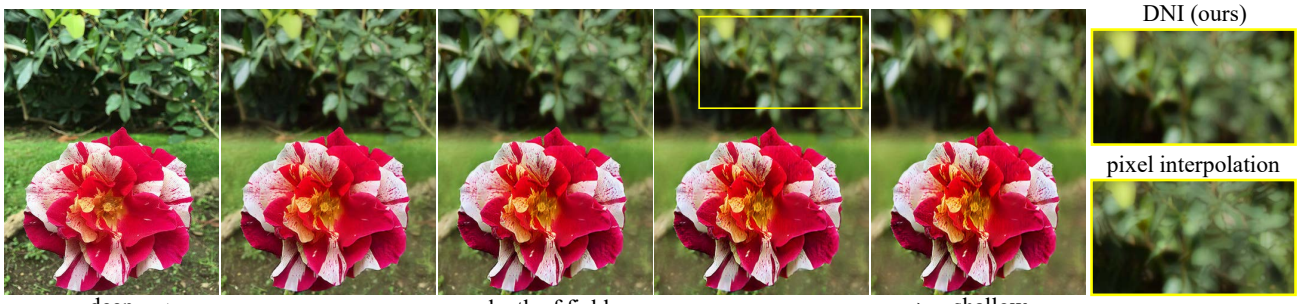
user requirements since they are trained to generate deterministic results without the flexibility to control the denoising strength. On the contrary, our proposed DNI is able to achieve adjustable denoising strength by simply tweaking the interpolation coefficient α of different denoising models for N20, N40 and N60. For the grass, a weaker denoising strength could preserve more details while in the sky region, the stronger denoising strength could obtain an artifact-free result (red frames in Fig. 6). This example demonstrates the flexibility of DNI to customize restoration results based on the task at hand and the specific user preference.

4.2. Image-to-image Translation

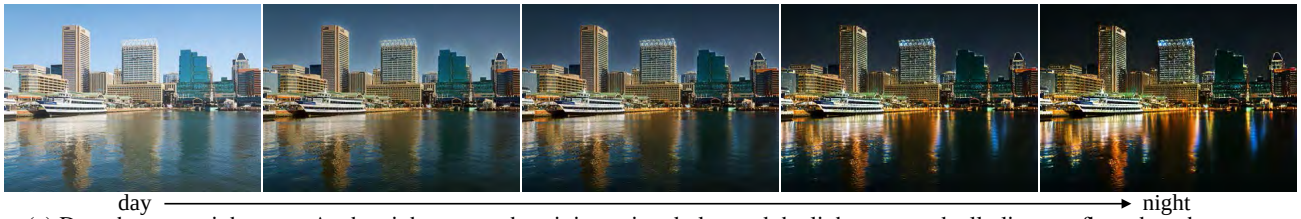
Image-to-image translation aims at learning to translate an image from one domain to another. Most existing approaches [15, 38, 26, 47] can only transfer one input image to several discrete outputs, lacking continuous translation for diverse user flavors. For example, one model may be able to mimic the style of Van Gogh or Cézanne, but translating a landscape photo into a mixed style of these two painters is still challenging.

The desired continuous transition between two painters' styles can be easily realized by DNI. The popular CycleGAN [47] is used as our implementation. We first train a network capturing characteristics of Van Gogh, and then fine-tune it to produce paintings of Cézanne's style. DNI is capable of generating various mixtures of these two styles given a landscape photo, by adjusting the interpolation coefficient. Fig. 7a presents a smooth transition from Van Gogh's style to Cézanne's style both in the palette and brush strokes. We note that DNI can be further employed to mix styles of more than two painters using Eq. 2. Results are

(a) Photos to paintings. DNI produces a smooth transition from Van Gogh's style to Cézanne's style both in the palette and brush strokes.



(b) Smooth transition on depth of field with DNI. However, pixel interpolation generates ghosting artifacts. (**Zoom in for best view**)



(c) Day photos to night ones. As the night approaches, it is getting darker and the lights are gradually lit up, reflected on the water.

Figure 7: Several applications for image-to-image translation. (**Zoom in for best view**)

provided in the Appendix C.

In addition to the translation between painting styles of a whole image, DNI can also achieve smooth and natural translation for a particular image region. Fig. 7b shows an example of photo enhancement to generate photos with shallower depth of field (DoF). We train one model to generate flower photos with shallow DoF and then fine-tune it with identity mapping. DNI is then able to produce continuous transitions of DoF by interpolating these two models. We also compare DNI with pixel interpolation, where the results look unnatural due to the ghosting artifacts, *e.g.*, translucent details appearing at the edge of blurry leaves.

DNI can be further applied to achieve continuous imagery translations in other dimensions such as light changes, *i.e.*, transforming the day photos to night ones. Only trained with day and night photos, DNI is capable of generating a series of images, simulating the coming of nights. In Fig. 7c, as the night approaches, it is getting darker and the lights are gradually lit up, reflected on the water.

4.3. Style Transfer

There are several controllable factors when transferring the styles of one or many pieces of art to an input image, *e.g.*, style mixture, stroke adjustment and the balance of content and style. Some existing approaches design specific structures to achieve a continuous control of these factors [16]. On the contrary, DNI offers a general way to attain the same goals without specific solutions. As shown in Fig. 8, DNI is capable of generating smooth transitions between different styles, from large to small strokes, together with balancing the content and style. Furthermore, DNI can be applied among multiple models to achieve a continuous control of various factors simultaneously. For instance, the stroke and style can be adjusted at the same time based on user flavors, as shown in Fig. 8.

Another branch of existing methods achieves a combination of various artistic styles by interpolating the parameters of instance normalization (IN) [6, 12]. These approaches can be viewed as a special case of DNI, where only IN parameters are fine-tuned and interpolated. To clarify the difference between DNI and IN interpolation, we conduct experiments with 3 settings: 1) fine-tune IN; 2) fine-tune convolutional layers and 3) fine-tune both IN and convolutional layers. Specifically, as shown in Fig. 9, we try a challenging task to fine-tune the model from generating images with mosaic styles to the one with fire styles, where the two styles look very different in both color and texture. It is observed that fine-tuning only IN is effective in color transformation, however, it is unable to transfer the fire texture effectively compared with the other two settings. This observation suggests that convolutional layers also play an important role in style modeling, especially for the textures,

Figure 8: In image style transfer, without specific structures and strategies, DNI is capable of generating smooth transitions between different styles, from large strokes to small strokes, together with balancing the content and style. (**Zoom in for best view**)

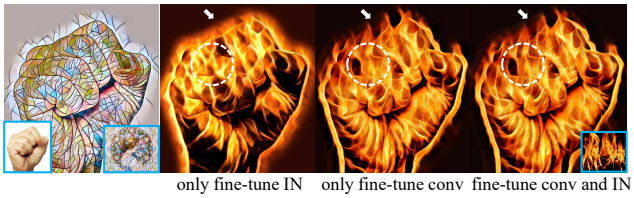


Figure 9: Fine-tuning only IN is effective in color transformation, however, it is unable to transfer the fire texture effectively. (**Zoom in for best view**)

since IN parameters may not be effective for capturing spatial information. However, we do not claim that DNI is consistently better than IN interpolation, since IN is also effective in most cases. A more thorough study is left for future work.

4.4. Semantic Transition

Apart from low-level vision tasks, we show that DNI can also be applied for smooth transitions on face attributes, suggesting its potential for semantic adjustment. We first train a DCGAN model [30] using the CelebA [27] dataset with one attribute (*e.g.*, young or male). After that, we fine-tune it to generate faces with another attribute (*e.g.*, old or female). DNI is then able to produce a series of faces with smoothly transformed attributes by interpolating these models (Fig. 10). Although neither of the interpolated models observes any data with middle attributes, the faces in middle states has an intermediate attribute and looks natural.

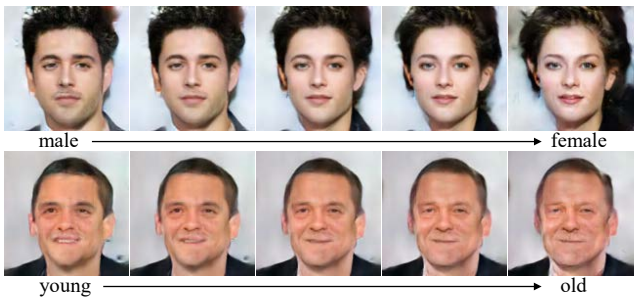


Figure 10: Smooth transitions on face attributes with DNI.

5. Conclusion

In this paper, we propose a novel notion of interpolation in the parameter space, *i.e.*, applying linear interpolation among the corresponding parameters of multiple correlated networks. The imagery effects change smoothly while adjusting the interpolation coefficients. With extensive experiments on super-resolution, denoising, image-to-image translation and style transfer, we demonstrate that the proposed method is applicable for a wide range of low-level vision tasks despite its simplicity. Compared with existing methods that achieve continuous transition by task-specific designs, our method is easy to generalize with negligible computational overhead. Future work will investigate the effects of network interpolation on high-level tasks.

References

[1] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *CVPR*, 2012. 1, 2

[2] C. Dong, Y. Deng, C. C. Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. In *ICCV*, 2015. 1

[3] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014. 1, 2, 4, 11

[4] W. Dong, L. Zhang, G. Shi, and X. Li. Nonlocally centralized sparse representation for image restoration. *TIP*, 22(4):1620–1630, 2013. 2

[5] V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. d. Vries, A. Courville, and Y. Bengio. Feature-wise transformations. *Distill*, 2018. https://distill.pub/2018/feature-wise-transformations. 3

[6] V. Dumoulin, J. Shlens, M. Kudlur, A. Behboodi, F. Lemic, A. Wolisz, M. Molinaro, C. Hirche, M. Hayashi, E. Bagan, et al. A learned representation for artistic style. In *ICLR*, 2016. 1, 2, 7, 14

[7] L. Galteri, L. Seidenari, M. Bertini, and A. Del Bimbo. Deep generative adversarial compression artifact removal. In *ICCV*, 2017. 2

[8] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 1, 2

[9] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. In *BMVC*, 2017. 2

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3

[11] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 2, 3, 5, 13

[12] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2, 7, 14

[13] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018. 2

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICMR*, 2015. 3

[15] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 1, 2, 6

[16] Y. Jing, Y. Liu, Y. Yang, Z. Feng, Y. Yu, D. Tao, and M. Song. Stroke controllable fast style transfer with adaptive receptive fields. In *ECCV*, 2018. 2, 7

[17] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 1, 2

[18] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 1, 2

[19] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, 2017. 2

[20] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 1, 2, 5, 13

[21] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *CVPR*, 2016. 2

[22] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. In *CVPR*, 2017. 3

[23] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. In *NIPS*, 2017. 2

[24] Y. Li, J. Yosinski, J. Clune, H. Lipson, and J. E. Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *ICLR*, 2016. 4

[25] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *CVPRW*, 2017. 2

[26] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *NIPS*, 2017. 1, 2, 6

[27] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 8

[28] S. Nah, T. H. Kim, and K. M. Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*, 2017. 1, 2

[29] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. *arXiv preprint arXiv:1709.07871*, 2017. 3

[30] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 8

[31] H. Sheikh. Live image quality assessment database release 2. *http://live. ece. utexas. edu/research/quality*, 2005. 14

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 3

[33] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *CVPR*, 2015. 2

[34] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *CVPR*, 2017. 2

[35] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, B. Lim, S. Son, H. Kim, S. Nah, K. M. Lee, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *CVPRW*, 2017. 1, 5, 13

[36] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICMR*, 2016. 2

[37] P. Upchurch, J. R. Gardner, G. Pleiss, R. Pless, N. Snavely, K. Bala, and K. Q. Weinberger. Deep feature interpolation for image content changes. In *CVPR*, 2017. 3

[38] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018. 1, 2, 6

[39] X. Wang, K. Yu, C. Dong, and C. C. Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform. In *CVPR*, 2018. 3

[40] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy. ESRGAN: Enhanced super-resolution generative adversarial networks. In *ECCVW*, 2018. 5

[41] Z. Wang, D. Liu, S. Chang, Q. Ling, Y. Yang, and T. S. Huang. D3: Deep dual-domain based fast restoration of jpeg-compressed images. In *CVPR*, 2016. 2

[42] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *IJCV*, 70(1):77–90, 2006. 3

[43] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *NIPS*, 2014. 2

[44] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *TIP*, 26(7):3142–3155, 2017. 1, 2, 5, 14

[45] K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning deep cnn denoiser prior for image restoration. In *CVPR*, 2017. 2

[46] K. Zhang, W. Zuo, and L. Zhang. Learning a single convolutional super-resolution network for multiple degradations. In *CVPR*, 2018. 2, 3

[47] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 1, 2, 6, 13

Appendix

We first provide more details about filter analyses on other tasks in Sec. A, such as super-resolution with different kernels and JPEG artifacts removal with different compression qualities. We then provide the implementation details of deep network interpolation for different applications in Sec. B. Additional applications and analyses are presented in Sec. C.

A. Filter Analyses

In the main paper, we provide a preliminary explanation behind the deep network interpolation (DNI) from the filter perspective, through analyses of the denoising task. Specifically, the observations can be summarized as: 1) Fine-tuning facilitates high correlation between parameters of different networks, providing the possibility for meaningful interpolation. 2) Fine-tuned filters for a series of related tasks present continuous changes. 3) Our analyses show that interpolated filters could fit the actual learned filters well.

The similar observations could also be found in several other tasks, such as super-resolution with different kernels and JPEG artifacts removal with different compression levels. In particular, we also adopt a three-layer network similar to SRCNN [3], where the first and last convolutional layers have 9×9 filter size (*i.e.*, the same architecture as that in the main paper). Following the notion of [3], the first and last layer can be viewed as a *feature extraction* and *reconstruction* layer, respectively.

For super-resolution, we focus on a series of blurring kernels with different kernel widths, followed by a downsampling operation. The kernel widths in our experiments are K3, K5, K7, K9, and K11, where K3 denotes a Gaussian blur kernel with size 3. We use the *OpenCV GaussianBlur* function and the Gaussian kernel standard deviation can be derived from the kernel width.

For JPEG compression artifacts removal, we employ increasing compression qualities with Q10, Q20, Q30, Q40, and Q50, (the larger the number, the better image quality after compression).

We obtain all the models from one pre-trained model by fine-tuning. The fine-tuning maps are depicted in Fig. 11. Each dot in the figure represents a model. Lines with arrows denote fine-tuning. For instance, the super-resolution (K3) model is fine-tuned from the denoising (N20) model.

Fig. 12 visualizes several filter examples of the feature extraction and reconstruction layers for denoising, super-resolution and DeJPEG tasks. We can found that: 1) Under the constraint of fine-tuning, the learned filters for related tasks exhibit continuous changes. This phenomenon is observed in all the tasks, including denoising, super-resolution and DeJPEG. 2) Except the similarity, we further see that

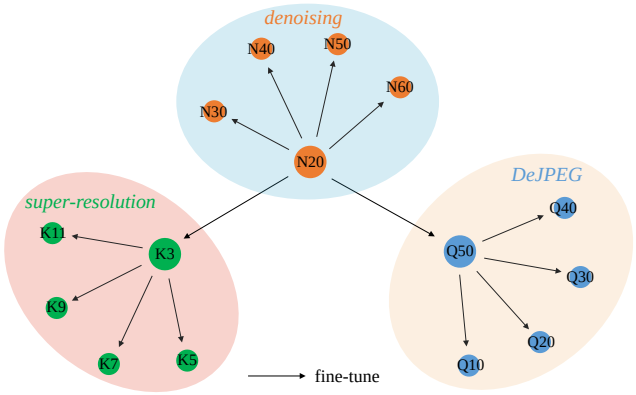


Figure 11: Each dot represents a model. Lines with arrows denote fine-tuning. For instance, the super-resolution (K3) model is fine-tuned from the denoising (N20) model.

filters for different image distortions (*e.g.*, denoising and super-resolution) capture the special characteristics of their own tasks, representing different filter patterns, especially in the reconstruction layer. Thus, if two tasks are far way from each other, the weak correlation of filters could result in unsatisfying, even meaningless interpolated results. The definition of task distances and the application boundaries of DNI are still open questions. Our analyses focus on related tasks of different degradations under the same distortion. It is noteworthy that DNI is capable of dealing with lot of tasks for continuous imagery effect transition and its broad applications indicate that related tasks with close "distances" are common and practical in real-world scenarios.

We also calculate the filter correlation indexes and plot their correlation distribution. The curves of the *feature extraction* and *reconstruction* layers for denoising, super-resolution and DeJPEG tasks are shown in Fig. 13. We again observe continuous changes of learned filters for different distortion levels under the same distortion.

We then show that the interpolated filters fit the learned filters well for different layers in the networks, and also for different tasks including denoising, super-resolution and DeJPEG. We perform linear interpolation between the filters from the two ends of degradation levels (*e.g.*, the N20 and N60 models for denoising; the K3 and K11 models for super-resolution). With optimal interpolation coefficients α, the interpolated filters could visually fit those learned filters for all the three tasks, as shown in Fig. 14. The observations could be held for both the feature extraction and reconstruction layers.

The optimal α is obtained through the final performance of the interpolated network. Specifically, we perform DNI with α from 0 to 1 with an interval of 0.05. The best α for each degradation level is selected based on the highest PSNR on the test dataset.

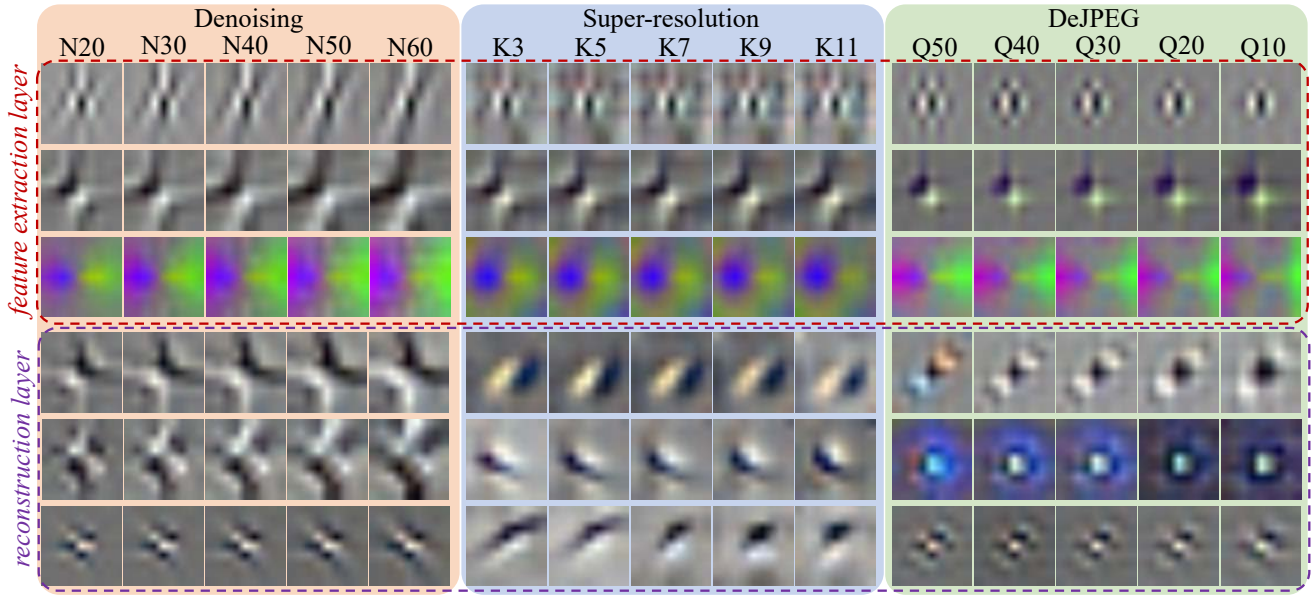


Figure 12: Filter visualization examples of the *feature extraction* and *reconstruction* layers for denoising, super-resolution and DeJPEG tasks. 1) The learned filters for different distortion levels under the same degradation exhibit continuous changes. 2) Filters for different image distortions (*e.g.*, denoising and super-resolution) capture the special characteristics of their own tasks, representing different filter patterns, especially in the *reconstruction* layer.

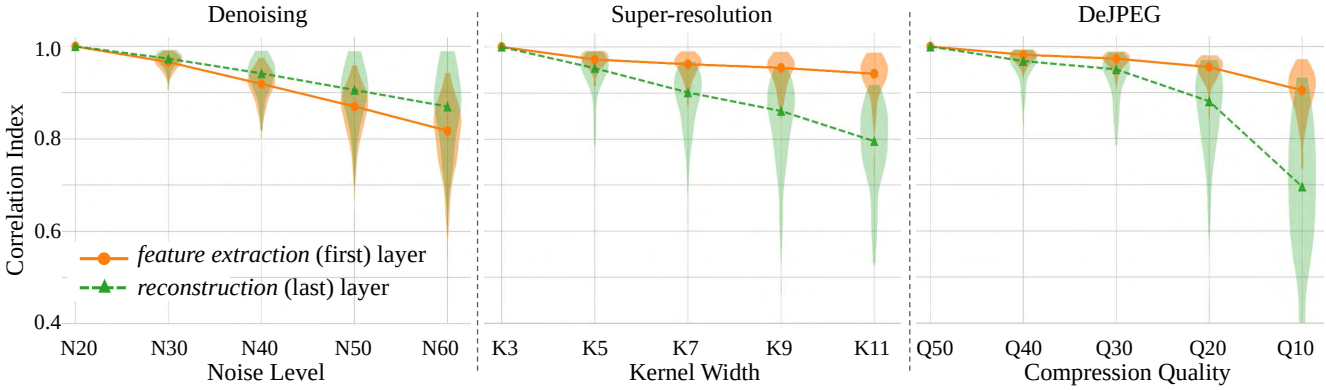
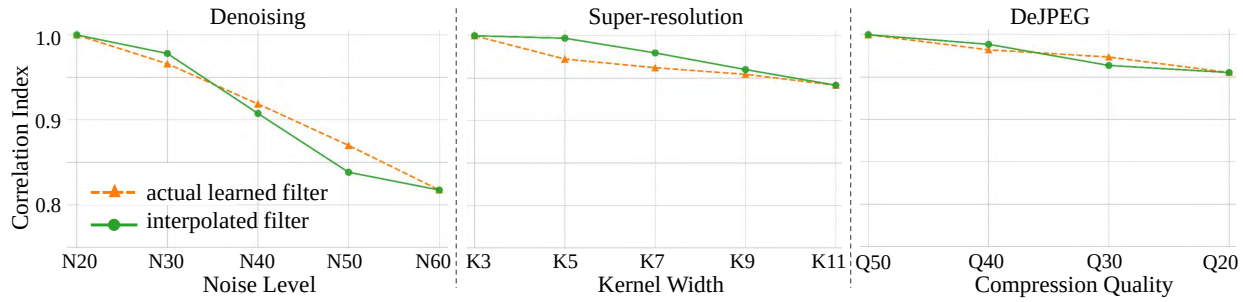


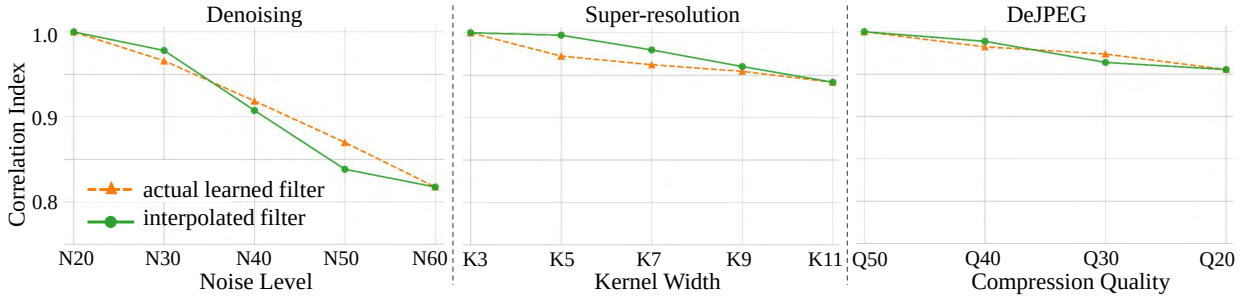
Figure 13: Filter correlation index curves of the *feature extraction* and *reconstruction* layers for denoising, super-resolution and DeJPEG tasks. The curves present the median of correlation indexes and the correlation distributions are also plotted. These curves show that the learned filters for different distortion levels under the same distortion exhibit continuous changes.

As the network goes deeper, the non-linearity increases and the network behaviors become more complicated. We provide a preliminary analysis of deeper denoising network in the main paper, where the observations are consistent with our conclusion. For other tasks, such as image style transfer or image translation, the investigation from the filter perspective becomes more sophisticated, since these tasks cannot be defined continuously like the continuous degradation levels in image restoration. The exploration of inherent filter correlations and the in-depth reason why DNI works are worth investigated in the future work.

The non-linearity between interpolation coefficients and effects. In our analyses, in order to realize "linear effects" of outputs, (*i.e.* obtaining denoising models dealing with N30, N40 and N50), the different interpolation coefficients do not present a linear relationship. The practical optimal α, obtained through the final performance of the interpolated network, for each noise level is shown in Fig. 15 (orange curve). One reason is that the effects of filter interpolation and the output effects are inherently not linear. We can also obtain the optimal α by optimizing the filter correlation index with models trained for each noise levels. A similar



(a) The *feature extraction* layer (first layer).



(b) The *reconstruction* layer (last layer).

Figure 14: The correlation curves for actual learned filters and interpolated filters are very close for both the *feature extraction* and *reconstruction* layers, indicating that the interpolated filters could fit learned filters well. The similar observations could be found on denoising, super-resolution and DeJPEG tasks.

non-linear trend as the practical curve could be observed (Fig. 15, green curve). We suspect that the extra gap between the green and orange curves may be from the non-linearity in networks.

The non-linear sampling of α for "linear" transition effects could be observed in many DNI applications. However, it does not influence its extensive applications. A solution is to control the sampling density of α, simply resulting in "linear" transition effects.

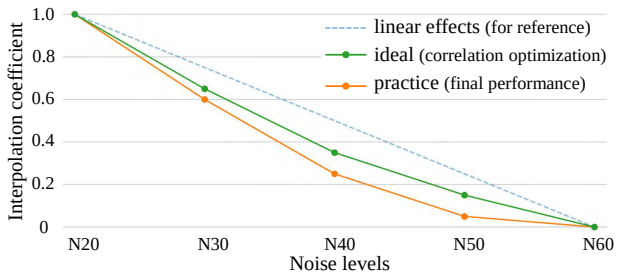


Figure 15: To achieve "linear" transition effects, the coefficients for each level are non-linear both from ideal analyses (optimizing the filter correlation) and practical performance (obtaining from the final PSNR.)

B. Implementation Details

In this section, we provide the implementation details of DNI and the fine-tuning strategy for several applications in the main paper.

Balance MSE and GAN effects in super-resolution. The MATLAB bicubic kernel with a scaling factor of 4 is adopted as the down-sampling kernel. We first train a super-resolution model with MSE loss [35], which tends to produce over-smooth images. We then fine-tune it with GAN loss and perceptual loss [20], obtaining results with vivid details yet accompanied with unpleasant artifacts. DNI is applied in these two models. We use dense blocks [11] as the network architecture.

Image-to-image Translation. In image-to-image translation, we use the popular CycleGAN [47] to explore the broad applications of DNI[2]. In the original CycleGAN, there are two networks G_A and G_B to learn a mapping and its inverse mapping, respectively. For instance, in order to translate landscape photos to paintings, G_A learns the mapping from paintings to photos while G_B learns an inverse mapping from photos to painting.

We use slightly different settings for different applications. For *mixing various painting styles*, we first train a Cy-

[2]We use the official released codes: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix.

cleGAN model for translating photos to paintings with Van Gogh's style, *i.e.*, $G_A^{VanGogh}$ is used for turning Van Gogh paintings into photos, and $G_B^{VanGogh}$ for translating photos into paintings with Van Gogh's style. We then fine-tune these two networks $G_A^{VanGogh}$ and $G_B^{VanGogh}$ (together with the discriminators) to models with another painter style, such as Monet, and obtain G_A^{Monet} and G_B^{Monet}.

During inference, our aim is to translate a landscape photo to paintings with various styles, even the mixtures of several famous painters. Thus, we only keep the $G_B^{VanGogh}$ and G_B^{Monet} and perform DNI on these two networks.

For *day-to-night application*, we first train a CycleGAN model as usual, *i.e.*, G_A is used for translating a day photo to a night one while G_B translates the night photo to the day one. Note that the CycleGAN model is only able to translate between two states and cannot produce a series of images with smooth transitions from day to night. We then fine-tune the whole pre-trained model with identity mapping. Specifically, we remove the GAN loss, adopt a $10\times$ identity loss and keep the cyclic loss in the CycleGAN framework. Thus, the fine-tuned network $G_A^{identity}$ always outputs identical results, *i.e.*, it receives a day photo and outputs the same day photo.

DNI is then applied in G_A and $G_A^{identity}$. With an appropriate coefficient, the interpolated network is able to produce images with arbitrary effects between day and night. The same operations are also employed in the *deep-to-shallow depth of filed* application.

Style Transfer. We use the PyTorch example codes for style transfer[3]. We first train a model for one certain style and then fine-tune it for another style. DNI is performed on these two models. For the stroke factor, we fine-tune the pre-trained model for a style image with different size. In order to balance the content and style, we fine-tune the pre-trained model with smaller style loss, resulting in almost identity mapping.

We note that DNI is generic for several controllable factors in style transfer, such as styles, strokes, and the balance of content and style. The proposed DNI could be also applied to more advanced models [6, 12] that address multiple or arbitrary style transfer, resulting in more diverse outputs.

C. More Applications and Analyses

C.1. Extend Restoration Models to Unseen Distortion Levels

In Sec. A, we reveal the inherent correlation of learned filters for a series of related tasks. Here, we take advantage of this observation to extend restoration models to deal with unseen distortion levels. We take the denoising task for example and it could also be applied to other restoration tasks, *e.g.*, super-resolution with different down-sampling kernels.

We employ an extreme case, where only the data with noise level N20 and N60 are available during training and we expect the trained models are able to handle arbitrary noise levels in the middle during testing. In particular, we test unseen N30, N40, and N50 noise levels. The upper bound for each level is the model trained with its corresponding data. The baseline is the model trained with both the N20 and N60 data. We adopt DnCNN [44] as the denoising model[4] and use two variants, with and without BN.

We first train a model using N20 data and then fine-tune it with N60 data. DNI is then performed with interpolation coefficients $\alpha \in [0, 1]$ with an interval of 0.1. The best result for each noise level is selected among these interpolated models (the chosen α for each level is shown in Tab. 1). We note that the selection is simple with a few trials due to the smooth transition. For automatic denoising, we could further train a shallow network to regress a proper α according to the noisy input.

We evaluate DNI using the LIVE1 [31] dataset. Quantitative results with the PSNR metric and qualitative results are shown in Tab. 1 and Fig. 16 respectively. The baseline model is incapable of removing unseen noise (*e.g.*, N30 and N40 in Fig. 16), leading to drastic drops in performance. However, our method could deal with those unseen scenes, even approaching to the upper bound. (Note that the upper bound observes the corresponding data.)

Though our DNI with BN can outperform its corresponding baseline, there is still a little drop compared with that without BN. The BN effects for DNI are still an open question and a better interpolation scheme needs to be explored for normalization layers. We also note that this application demonstrates that it is worth exploiting the underlying relations of learned filters to further extend the ability and practicality of existing models.

Table 1: The average denoising results of PSNR (dB) on the LIVE1 test dataset. Unseen noise levels are denoted with *. Note that the upper bound have seen the corresponding data.

	Noise level	N20	N30*	N40*	N50*	N60
w/o BN	Upper bound	32.38	30.39	29.01	27.98	27.18
	Baseline	32.36	23.86	21.90	27.34	27.14
	DNI (ours)	**32.38**	**29.84**	**28.28**	**27.67**	**27.18**
	α	1	0.7	0.4	0.1	0
w/ BN	Upper bound	32.49	30.48	29.09	27.96	27.25
	Baseline	32.42	24.42	26.58	27.44	27.21
	DNI (ours)	**32.49**	**29.46**	**28.08**	**27.66**	**27.25**
	α	1	0.6	0.3	0.1	0

[3] https://github.com/pytorch/examples/tree/master/fast_neural_style.

[4] We use the official released codes:https://github.com/cszn/DnCNN (PyTorch version).

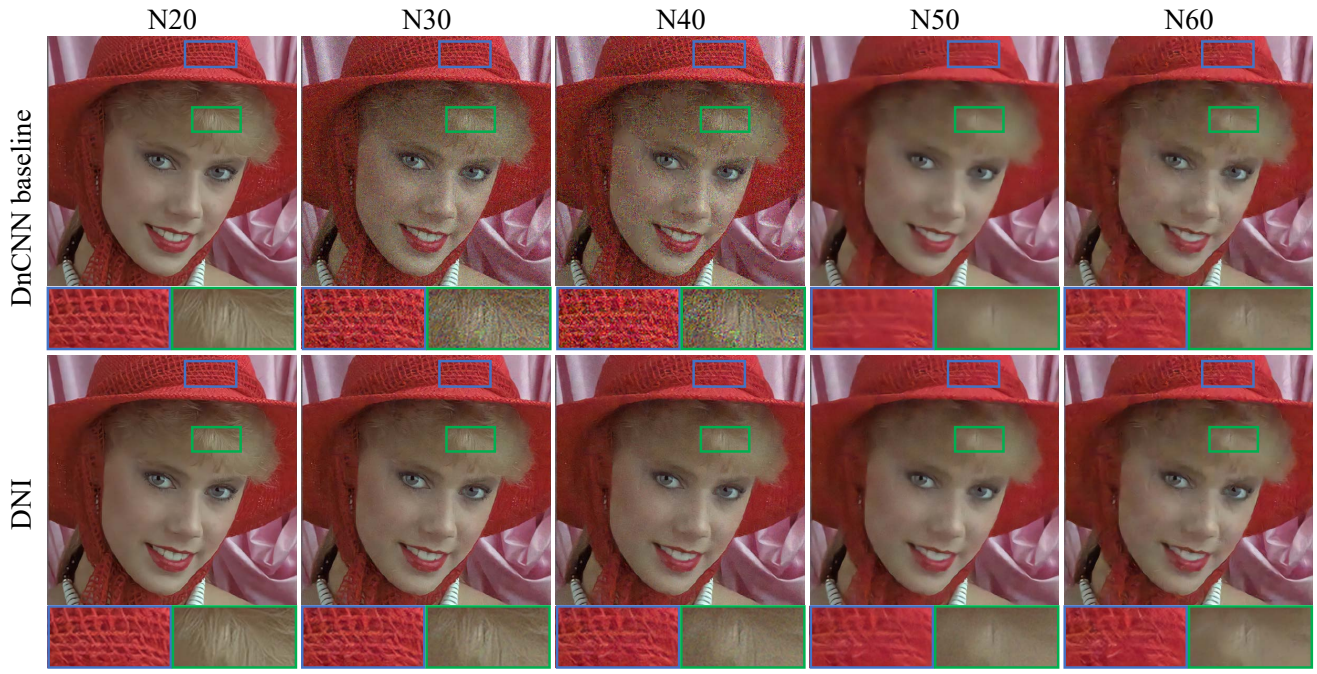


Figure 16: Extending denoising models to unseen noise levels. The baseline model trained with N20 and N60 data is incapable of removing unseen noise (*e.g.*, N30 and N40), while our method could deal with these unseen scenes. (**Zoom in for best view**)

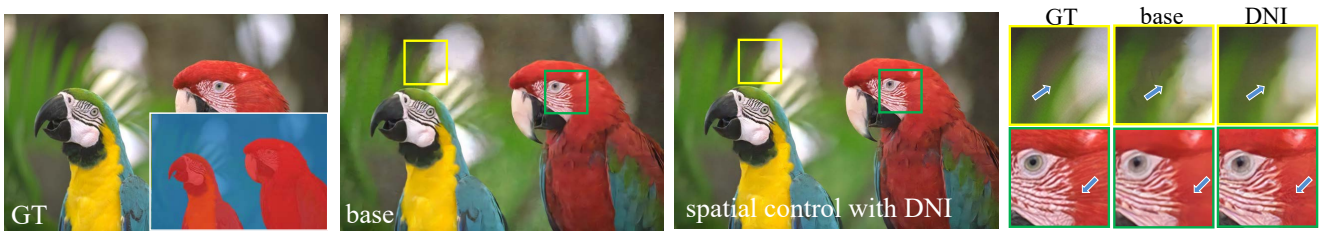


Figure 17: Spatial control for adjustable denoising. With a mask, different denoising strengths are applied separately for the foreground and the background. (**Zoom in for best view**)

C.2. Spatial Control for Adjustable Denoising

In the main paper, we emphasize the importance of adjustable denoising strength and show the ability of DNI to satisfy the needs. Here, we further present an application of spatial control for adjustable denoising. For the DSLR photos with shallow depth-of-filed, the background is usually blurred while the foreground contains rich details. We can easily separate them with a mask and adopt different denoising strengths respectively, obtaining better visual quality. From Fig. 17, we can see that with adjustable denoising realized by DNI, the blurry area is more smooth without artifacts, while there are rich details in texture regions.

Apart from the denoising task, the adjustments with DNI can also be applied to other image restoration tasks, *e.g.*, super-resolution with different down-sampling kernels and JPEG artifacts removal with different compression qualities.

C.3. Multi-ends DNI

A general form of DNI is also capable of interpolating more than two networks. Fig. 18 shows two examples of translating landscape photos to paintings with various styles – Van Gogh, Cézanne, Monet and Ukiyo-e. By adjusting the interpolated coefficients, richer and more diverse effects with continuous transitions could be realized.

Another example of image style transfer in Fig. 19 presents the ability of DNI to transfer among different styles – Mosaic style, Candy style, Mondrian style and Udnie style. It generates diverse and new styles, meeting users' various aesthetic flavors.

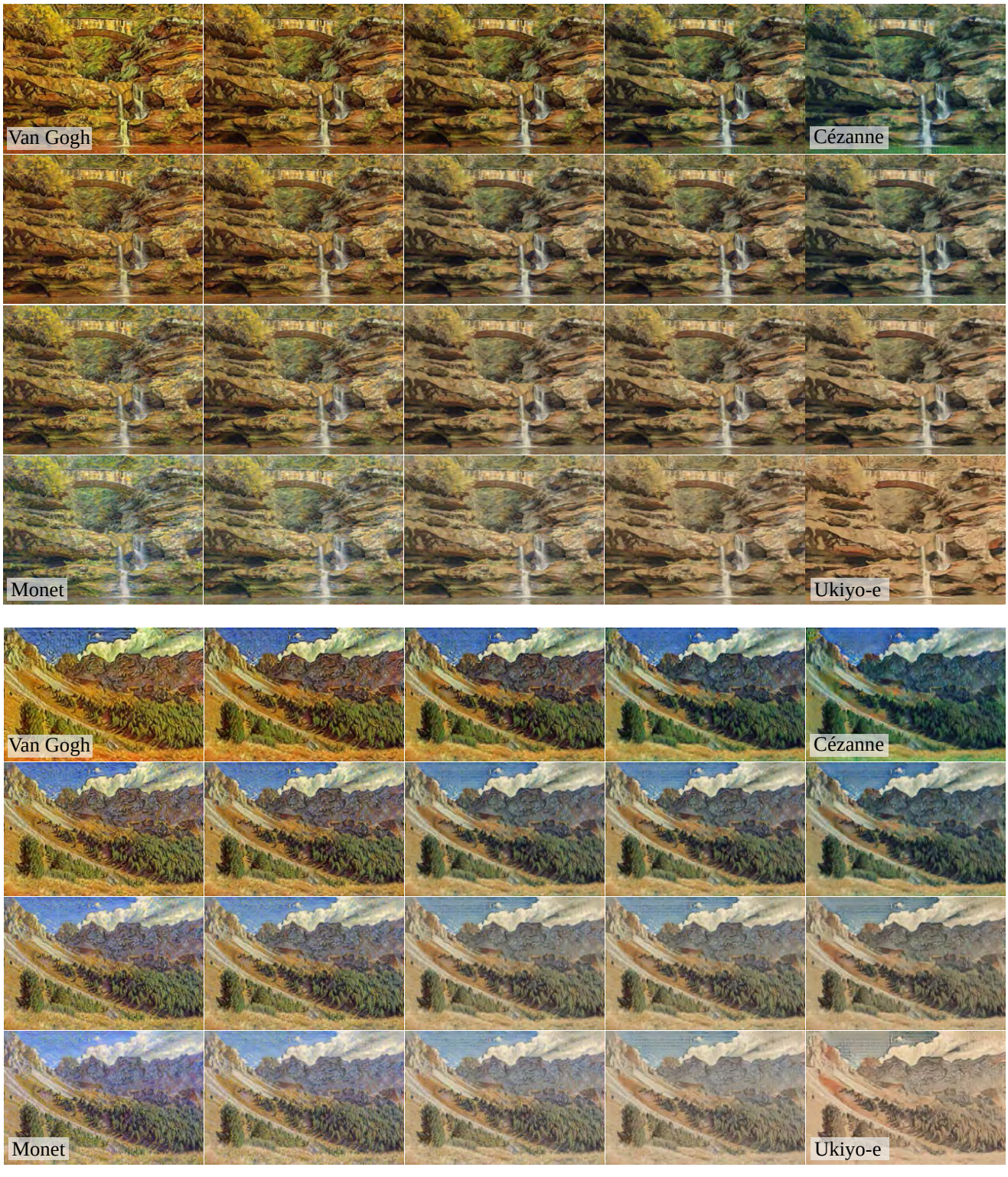


Figure 18: (Two examples) Translating landscape photos to paintings with various styles – Van Gogh, Cézanne, Monet and Ukiyo-e. By adjusting the interpolated coefficients, richer and more diverse effects with continuous transitions could be realized.

Figure 19: Image style transfer among different styles – Mosaic style, Candy style, Mondrian style and Udnie style. It generates diverse and new styles, meeting users' various aesthetic flavors.