Smoother Network Tuning and Interpolation for Continuous-level Image Processing

Hyeongmin Lee[⋆1], Taeoh Kim[⋆1], Hanbin Son[1], Sangwook Baek[2], Minsu Cheon[3], and Sangyoun Lee[†1]

[1]Yonsei University, Seoul, Korea [2]Samsung Research, Seoul, Korea [3]MathWorks

{minimonia, kto, hbson, syleee}@yonsei.ac.kr, sw123.baek@samsung.com, mcheon@mathworks.com

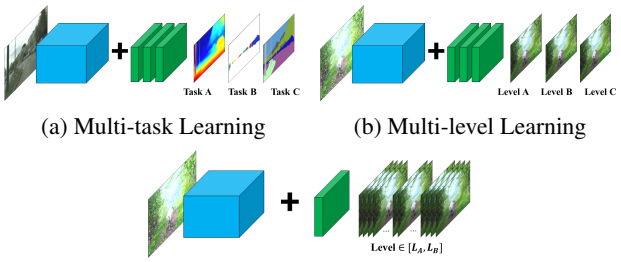
Abstract

In Convolutional Neural Network (CNN) based image processing, most studies propose networks that are optimized to single-level (or single-objective); thus, they underperform on other levels and must be retrained for delivery of optimal performance. Using multiple models to cover multiple levels involves very high computational costs. To solve these problems, recent approaches train networks on two different levels and propose their own interpolation methods to enable arbitrary intermediate levels. However, many of them fail to generalize or have certain side effects in practical usage. In this paper, we define these frameworks as *network tuning and interpolation* and propose a novel module for continuous-level learning, called Filter Transition Network (FTN). This module is a structurally smoother module than existing ones. Therefore, the frameworks with FTN generalize well across various tasks and networks and cause fewer undesirable side effects. For stable learning of FTN, we additionally propose a method to initialize non-linear neural network layers with identity mappings. Extensive results for various image processing tasks indicate that the performance of FTN is comparable in multiple continuous levels, and is significantly smoother and lighter than that of other frameworks.

Introduction

Image processing algorithms have various objectives that can include a combination of objective functions or a pair of target level-specific training datasets. For example, in restoration tasks such as denoising, there is an optimal level for each input whose noise level is unknown, and in image synthesis, balancing fidelity and naturalness (Blau and Michaeli 2018) depends on target applications. In style transfer, the user hopes to control various styles and stylization strengths continuously.

However, most image processing deep networks are trained and optimized for single-level. In this paper, the word ***level*** can be one of the following examples: a target noise level (standard deviation of Gaussian noise or quality factor of JPEG), a specific combination of objective functions to optimize, or a target style for style transfer. If we want to handle N multiple levels, we must train N different models or exploit the structure of multi-task learning (Lim et al. 2017) (Fig. 1 (b)), which is inefficient when N increases. In addition, in many image processing tasks, levels can be continuous. Therefore, designing a network for a continuous-level in an efficient way is a very practical issue. Fig. 1 describes the differences among multi-task learning, multi-level learning, and continuous-level learning. Compared to multi-task learning, multi-level learning solves single-task and multiple discrete-level problems. Continuous-Level Learning (CLL) is an extension of multi-level learning whose levels are continuous between two levels, which is a general issue in image processing tasks.



(a) Multi-task Learning (b) Multi-level Learning

(c) Continuous-level Learning

Figure 1: Comparison of multi-task learning, multi-level learning, and continuous-level learning. Every task or level shares a main network (blue) and introduces an additional branch (green) for task- or level-specific optimization

To deal with CLL problems, several frameworks have been proposed (He, Dong, and Qiao 2019; Shoshan, Mechrez, and Zelnik-Manor 2019; Wang et al. 2019a,b, 2018), and they have the following steps in common. In the training phase, they train their CNN network twice for each level. The first-training is similar to the general network training method. During the second-training, some parameters that were optimized in the first-training are fixed, and the other parameters are fine-tuned or some additional modules are trained for the second level. In the test phase, they make their networks available at any intermediate level with their own interpolation methods. We define these frameworks as *network tuning and interpolation*. These steps are derived from observations in (He, Dong, and Qiao 2019; Wang et al. 2019b). They show that the fine-tuned filters are similar to

[⋆]Equal Contribution

[†]Corresponding Author

those of the original filters, which makes the interpolation space between filters meaningful.

Although various *network tuning and interpolation* frameworks have been proposed, deeper analysis and stable generalization for practical usage are required. We define three aspects to better utilize CLL algorithms. The first is ***adaptation and interpolation performance***. After the second training, its performance might be lower than that of the one trained only for the second level, because it contains parameters for both levels. Therefore, CLL frameworks have to be flexible in order to adapt to new levels. In addition, even though the network works well on the two trained levels, it might not work for the other intermediate levels. Therefore, it is also important for the networks to maintain high performance and reasonable outputs for intermediate levels. We can measure them using the metrics for each task at arbitrary intermediate unseen continuous levels. The second one is ***smoothness*** for practical usage scenarios. During the experiment, we find that certain artifacts and unintended behaviors are caused by some CLL methods. Exhibiting stable performance across tasks and networks, not producing undesirable artifacts, and operating with interpretable control parameters are very important for real-world usage, and we define these aspects collectively as *smoothness*. The last one is ***efficiency***. Because one of the main objectives of CLL is to use a single network instead of using multiple networks trained for each level, requiring too large memory and computational resources is not practical for real-world applications.

Most of the prior approaches have limitations in terms of the above three aspects. AdaFM (He, Dong, and Qiao 2019) introduces a tuning layer called the feature modification layer, which is a simple linear transition block (depth-wise convolution). However because AdaFM is originally proposed for image restoration tasks only, linearity reduces the flexibility of adaptation. Therefore, it is not appropriate for more complex tasks such as the perception-distortion (PD) trade-off in restoration or style transfer. Deep Network Interpolation (DNI) (Wang et al. 2019b, 2018) interpolates all parameters in two distinct networks trained for each level to increase flexibility. One is the version trained from the initial state, and the other is the version fine-tuned starting from the first one. However, fine-tuning the network without any constraint cannot consider the initial level, which might lead to a degraded performance at intermediate levels. In fact, from the experiments, DNI has limitations on the smoothness conditions. DNI also requires extra memory to save temporary network parameters and a third interpolated network for the inference. CFS-Net (Wang et al. 2019a) and Dynamic Net (Shoshan, Mechrez, and Zelnik-Manor 2019) propose frameworks that use additional tuning branches to interpolate the feature maps, not the model parameters. However, tuning branches require large memory and heavy computations up to twice the baseline networks. Training two branches independently can cause over-smoothing artifacts because each branch cannot consider each other. This side effect will be discussed later.

In this paper, we propose a novel smoother *network tuning and interpolation* method using a *Filter Transition Network (FTN)* that take CNN filters as input and learns the transitions between levels. Because FTN is a non-linear module, networks can better adapt to any new level than the linear one. Therefore, it can cover general image processing tasks from simple image denoising to complex stylization tasks. FTN *transforms* the filters of the main network via other learnable networks, and we can control the flexibility of transformation by restricting the learnable parameters for smooth and stable interpolation. For efficiency, from the motivations in (He, Dong, and Qiao 2019; Wang et al. 2019b), FTN directly changes filters to be data-agnostic. In other words, because FTN takes filters as input instead of feature maps, the computational complexity does not increase when the input images increase in size. In addition, randomly initialized FTN makes the training process unstable because it directly changes the model parameters. To solve this problem, we propose a method to initialize multiple nonlinear layers to be identity mappings.

In summary, the proposed framework has the following contributions:

- We propose a smoother network tuning and interpolation method for CLL using the FTN, which is structurally smoother than the other frameworks. In addition, for the stable learning of FTN, we propose a new initialization method that makes non-linear network layers be identity mapping.
- We define and point out the smoothness conditions for CLL which is important for practical applications such as generalization across tasks and networks, color artifacts, and interpretable control parameters.
- Our method is comparable in adaptation and interpolation performance on multiple imaging levels, significantly smoother in practice, and efficient in both memory and computational complexity.

Related Work

In this section, we summarize the CLL problems in image processing tasks which will be experimented in this paper.

Image Restoration. CNN-based image restoration has shown great performance improvements over handcrafted algorithms. After shallow networks, (Dong et al. 2015a,b), some works stacked deeper layers, exploiting the advantages of residual skip-connection (Kim, Kwon Lee, and Mu Lee 2016; Zhang et al. 2017). Following the evolution of image recognition networks, restoration networks have focused on the coarse-to-fine scheme (Lai et al. 2017), dense connections (Zhang et al. 2018b), attentions (Zhang et al. 2018a) and non-local networks (Liu et al. 2018). However, most networks are trained and optimized for a single level such as the Gaussian noise level in denoising, quality factor in JPEG compression artifact removal, and super-resolution scale in single-image super-resolution. If the levels of training and test do not match, then the optimal restoration performance cannot be achieved. To deal with this limitation, (Mildenhall et al. 2018; Zhang, Zuo, and Zhang 2018) proposed multiple noise-level training with a noise-level map, or noise estimation network (Guo et al. 2019) can be a solution. However,

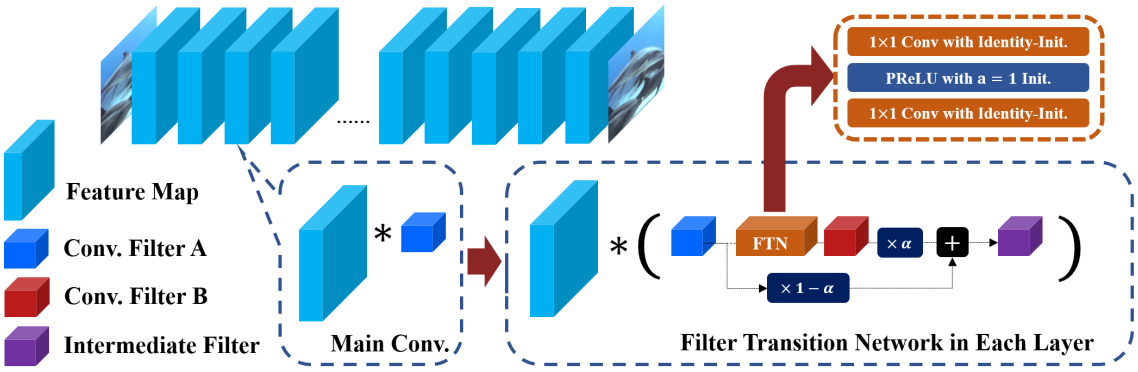


Figure 2: Network architecture of the proposed Filter Transition Network (FTN) when adapted in arbitrary main convolutional networks. Filter of main network (blue) is transformed via FTN for other levels (red). In inference phase, interpolated filter (purple) is used for intermediate levels

the user cannot control the test phase for better personalization (*e.g.* level of smoothing).

The Perception-Distortion Trade-off. In comparison with the general approach that attempts to reduce pixel-error with the ground truth, some works (Galteri et al. 2017; Ledig et al. 2017; Wang et al. 2018) attempted to produce more natural images using the generative power of GANs (Goodfellow et al. 2014; Mirza and Osindero 2014; Radford, Metz, and Chintala 2015). They used a combined loss of the fidelity and adversarial terms and then obtained better perceptual quality. However, when a more adversarial loss is used, worse fidelity with the ground truth occurs because of the perception-distortion (PD) trade-off (Blau and Michaeli 2018). In (Blau and Michaeli 2018), they proposed evaluating the restoration performance via a PD-plane considering the balance between fidelity and naturalness. However, the network must be retrained on another loss function to draw a continuous PD-function, which is very time-consuming.

Style Transfer. With regard to image style transfer, Gatys *et al.* (Gatys, Ecker, and Bethge 2015) proposed a combination of content loss and style loss, and optimized content images via pre-trained feature extraction networks. Johnson *et al.* (Johnson, Alahi, and Fei-Fei 2016) made it possible to operate in a feed-forward manner using an image transformation network. However, a network trained on a single objective cannot control the balance between content and style and cannot handle continuous styles when it is trained on a single style. Even though (Gatys et al. 2017) can control several factors in the training phase and arbitrary (Zero-shot) style transfer such as (Huang and Belongie 2017; Sheng et al. 2018) can handle infinite styles using adaptive instance normalization or style decorator, none of these can control continuous objectives (losses) during the test phase.

Proposed Approach

Filter Transition Networks

The general concept of our module is the same as that of the prior CLL frameworks, *network tuning and interpolation* which was described in the introduction. Our overall framework is detailed in Fig. 2. Our FTN module in an arbitrary convolutional layer can be described as

$$\mathbf{Y}_i = \mathbf{X}_i * (\mathbf{f}_i^{(1)} \times (1-\alpha) + FTN(\mathbf{f}_i^{(1)}) \times \alpha) \quad (1)$$

where $\mathbf{X}_i$ and $\mathbf{Y}_i$ are the input and output of the i-th convolution layer, $\mathbf{f}_i^{(1)}$ is the corresponding kernel, α is the control parameter between $[0, 1]$, and $*$ is the convolution operation. A remarkable difference from the existing frameworks is that FTN directly takes network kernels as inputs, instead of images or feature maps. It can be viewed as one variation of hyper-networks (Ha, Dai, and Le 2016) which takes and predicts network parameters. Our design goal is *complete adaptation with minimum filter change*. In other words, it is essential to keep our filters as similar as possible to the original filters while adapting another level well. We assume that this can increase *smoothness* and empirical results for this assumption are described in the *smoothness* section of the experimental results.

The FTN consists of two 1×1 convolutions with a G grouped convolution (Krizhevsky, Sutskever, and Hinton 2012; Xie et al. 2017), PReLU (He et al. 2015) activation functions, and skip-connection with weighted sum. First, we train the main convolutional filter for the initial level with $\alpha = 0$. Then, we freeze the main network and train the FTN only for the second level with $\alpha = 1$, which breaks skip-connection. Next, the FTN learns the task transition itself. To that end, the FTN approximates kernels of the second level, as in $FTN(\mathbf{f}_i^{(1)}) \approx \mathbf{f}_i^{(2)}$, where $\mathbf{f}_i^{(2)}$ is an optimal kernel for the second level. In the inference phase, we can interpolate between two kernels (levels) by choosing α in the 0-1 range, and Eq. (1). Consequently, the FTN implicitly learns continuous transitions between levels, and α represents the amount of filter transition towards the second level.

Group convolution can reduce the number of parameters in a network. If the number of groups is increased, the degrees of freedom to change the original filters decrease. We use this relationship to improve smoothness at the expense of adaptation and interpolation performance. 1×1 convolution is used because: 1) it is lightweight, and 2) padding

is not required. Because the input size of the FTN is quite small (usually $3 \times 3 \times C$), padding can be critical to each layer.

Initialization of FTN

During the second-training, since we set $\alpha = 1$, each convolution layer can be formulated as follows.

$$\mathbf{Y}_i = \mathbf{X}_i * (FTN(\mathbf{f}_i^{(1)})) \quad (2)$$

However, when we initialize FTN using general methods such as (Glorot and Bengio 2010; He et al. 2015), which will predict random filters from $\mathbf{f}_i^{(1)}$, the training cannot start from the first level ($FTN(\mathbf{f}_i^{(1)}) \neq \mathbf{f}_i^{(1)}$). These types of initialization make the training very unstable if special precautions are not taken. In our framework, every convolution and activation function is initialized as an identity function. Convolutions can easily become identities (He, Dong, and Qiao 2019). For activation functions, we use PReLU (He et al. 2015) with an initial negative slope $a = 1$, which will be learned through training.

Experiments

Experimental Settings

Baselines. To understand recent *network tuning and interpolation* frameworks for CLL, we evaluate FTNs against DNI (Wang et al. 2019b), AdaFM (He, Dong, and Qiao 2019), CFSNet (Wang et al. 2019a), and Dynamic-Net (Shoshan, Mechrez, and Zelnik-Manor 2019) on four general image processing tasks. We add a tuning layer of FTNs into every convolution, and the same for AdaFM (He, Dong, and Qiao 2019) except the last layer to prevent boundary artifacts. We add a ResBlock-wise (or DenseBlock-wise) tuning branch for CFSNet (Wang et al. 2019a). For a fair comparison, the main networks are identical and shared across frameworks, and every hyper-parameter is identical except for the tuning layers of each framework. More detailed configurations are described in the supplementary material.

השוו את הרשת לכמה מודלים מוכרים בתחום בכמה משימות. הרשת הראשית זהה כולל ההיפר פרמטרים. בשכבות של הtuning יש היפר פרמטרים שונים בין המודלים.

Denoising & DeJPEG. We use two baseline networks that were proposed in (He, Dong, and Qiao 2019) (AdaFM-Net) and (Wang et al. 2019a) (CFSNet-10). We use DIV2K (Agustsson and Timofte 2017) as the training set and test on the CBSD68 (Martin et al. 2001) dataset for denoising and LIVE1 (Moorthy and Bovik 2009) for deJPEG. We fine-tune the main network from the weaker noise (standard deviation 20 for denoising and quality factor 40 for deJPEG). The maximum PSNR is obtained via grid search of α.

Table 1: **Ablation study for structures of FTN.** Average PSNR (dB) on CBSD68 denoising test dataset. Unseen noise levels are denoted with *. The baseline network is **AdaFM-Net**. The best results are **bold-faced**.

Noise Level σ	20	30*	40*	50
FTN	**32.44**	**30.18**	**28.90**	**28.04**
FTN-deeper	**32.44**	30.06	28.81	28.03
FTN-spatial	**32.44**	30.16	28.88	**28.04**

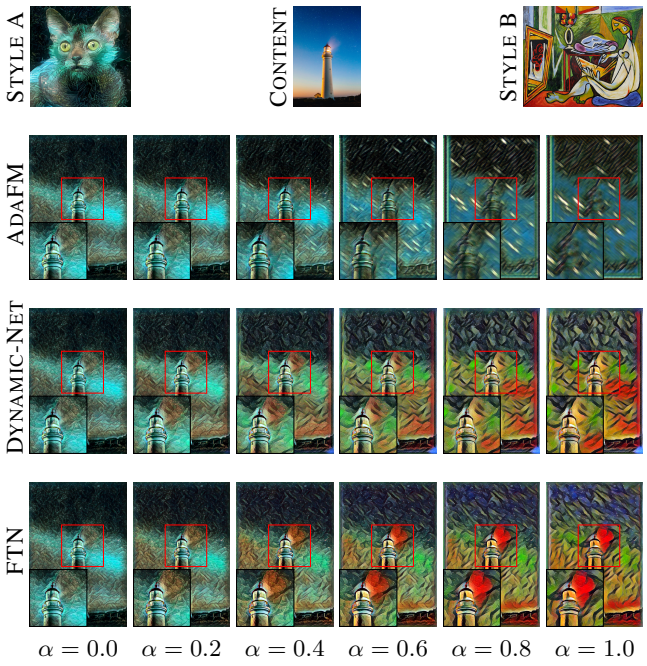


Figure 3: **Visual comparison of controllable style transfer results between the two styles.**

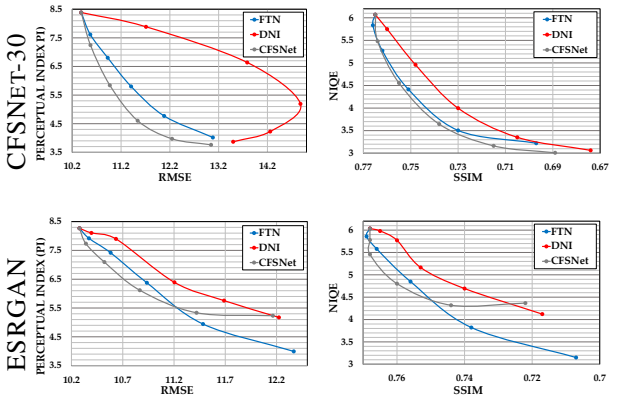
PD-Controllable Super-resolution. In image super-resolution, as reported in (Blau and Michaeli 2018), there is a trade-off between fidelity and naturalness. A comparison between algorithms should consider this trade-off by plotting perception (fidelity)-distortion (naturalness) (PD) curves. An algorithm that is closer to the origin in the PD-plane than the others implies it has better performance. Drawing this PD-curve is possible by changing the weights between loss terms. As in (Wang et al. 2018), we first-train the network using L_1 loss and second-train it using a combined loss of L_1, Perceptual (Johnson, Alahi, and Fei-Fei 2016), and GAN (Goodfellow et al. 2014) losses. We evaluate using two baseline networks that were proposed for (Wang et al. 2019a) (CFSNet-30) and (Wang et al. 2018) (ESRGAN). We use DIV2K as the training set and PIRM (Blau et al. 2018) as the test set. PSNR and SSIM (Wang et al. 2004) are used as distortion metrics, and NIQE (Mittal, Soundararajan, and Bovik 2012) and the Perceptual Index (Blau et al. 2018) are used as perception metrics.

דוק לטבעות

Style Transfer. In style transfer, we use Transform-Net which was proposed in (Johnson, Alahi, and Fei-Fei 2016) with instance normalization (Ulyanov, Vedaldi, and Lempitsky 2016). We follow the settings of Dynamic-Net (Shoshan, Mechrez, and Zelnik-Manor 2019). The COCO 2014 train dataset (Lin et al. 2014) is used for training. From the main network, Dynamic-Net inserts three tuning branches into pre-defined layers, while FTNs are inserted in every convolution layer. This means that FTNs have more opportunities to control in a layer-wise manner (Fig. 1 of the supplementary material of Dynamic-Net (Shoshan, Mechrez, and

Table 2: **Gaussian Denoising Results.** Average PSNR (dB) on CBSD68 test dataset. Unseen noise levels are denoted with *. The best results are **bold-faced** and the second best results are underlined.

	CFSNet-10 (10 Blocks)				AdaFM-Net (16 Blocks)			
Noise Level	20	40*	60*	80	20	40*	60*	80
From Scratch	32.42	28.98	27.13	25.90	32.44	28.90	26.92	25.60
DNI	**32.42**	**28.87**	**27.01**	**25.96**	**32.44**	28.20	26.98	25.97
AdaFM	**32.42**	28.48	26.75	25.84	**32.44**	28.17	26.77	25.96
CFSNet	**32.42**	28.65	26.95	25.93	**32.44**	28.41	26.87	26.00
FTN-gc16	**32.42**	28.77	**27.01**	25.89	**32.44**	**28.78**	**27.05**	25.98
FTN-gc4	**32.42**	28.65	26.90	25.90	**32.44**	28.64	26.95	26.00
FTN	**32.42**	28.45	26.86	25.93	**32.44**	28.48	26.89	**26.03**



Naturalness Image Quality Evaluator (NIQE) no-reference image quality score ככל שיותר נמוך יותר טוב, מטריקה כלשהי

Figure 4: **Results of PD-controllable image super-resolution (×4).** Combined, various adaptation results, and results images are described in supplementary material.

Zelnik-Manor 2019)).

Ablation Study

We perform an ablation study on AdaFM-Net for image denoising to compare different structures of FTNs in Table 1. We define two additional versions of FTN: FTN-*deeper* and FTN-*spatial*. FTN-*deeper* is a three-layer version of the FTN whose intermediate results are worse than others because excessive modification of the filters affects the interpolation results. FTN-*spatial* is a depth-wise convolution version whose performance is inferior to other channel-wise convolutions.

Adaptation and Interpolation Performance

Adaptation performance refers to the performance on the tuned second level compared to a network trained only for the level. The interpolation performance refers to the performance on the unseen intermediate interpolated levels. The degree of performance depends on the evaluation metrics of each task.

First, Fig. 3 depicts the result of the style transfer task, which requires a large transition of the model parameters as the style changes. According to Fig. 3, AdaFM has difficulty

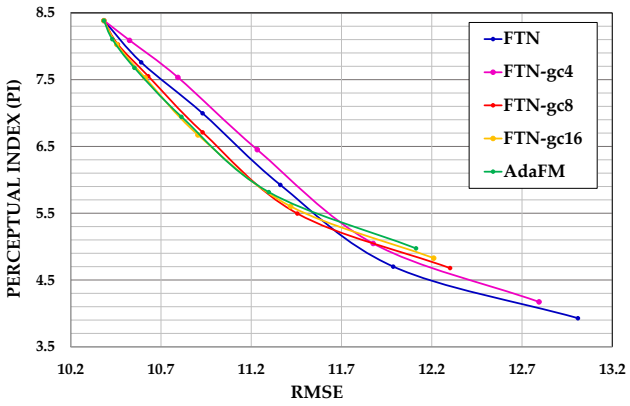


Figure 5: **Ablation study for group convolutions in PD-control**.

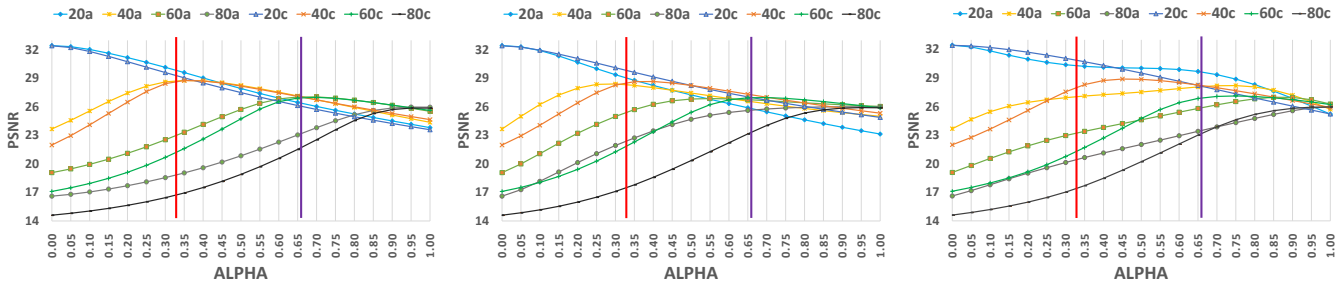
adapting from style A to the style B. These results show that linear adaptation has limitations in reaching the hard second level. In Dynamic-Net, it cannot deliver the second style smoothly because it only changes three pre-defined layers while FTN changes all convolutional filters. More results for stylization are described in the supplementary material.

Subsequently, Table 2 shows the adaptation/interpolation performances on the denoising task (Results images and results on deJPEG task are reported in the supplementary material.). The table shows that there is not notable difference between performances over the compared methods, including AdaFM which uses linear adaptation. This is because denoising tasks require their model parameters to be changed less as the level changes compared to other tasks (*e.g.* style transfer). FTN-gc4, 16 indicate the group convolution version of FTN with 4 and 16 groups, respectively. They show better interpolation performances at the expense of adaptation performance. Specifically, in AdaFM-Net, FTN-gc4 and FTN-gc16 outperforms the other frameworks. In CFSNet-10, DNI outperforms other frameworks but the margin is not large. In CFSNet-10, the network is shallower (10 ResBlocks) than AdaFM-Net (16 ResBlocks), which means that the parameter space can be easily linear. This can increase the filter similarity of simple fine-tuning (DNI). Compared to AdaFM and CFSNet, the performance of FTNs is superior.

However, compared to the denoising task, the DNI shows a different pattern in the PD-control (Fig. 4). Although DNI performs well for both end levels, it shows significantly unstable and low performance for the intermediate levels. This

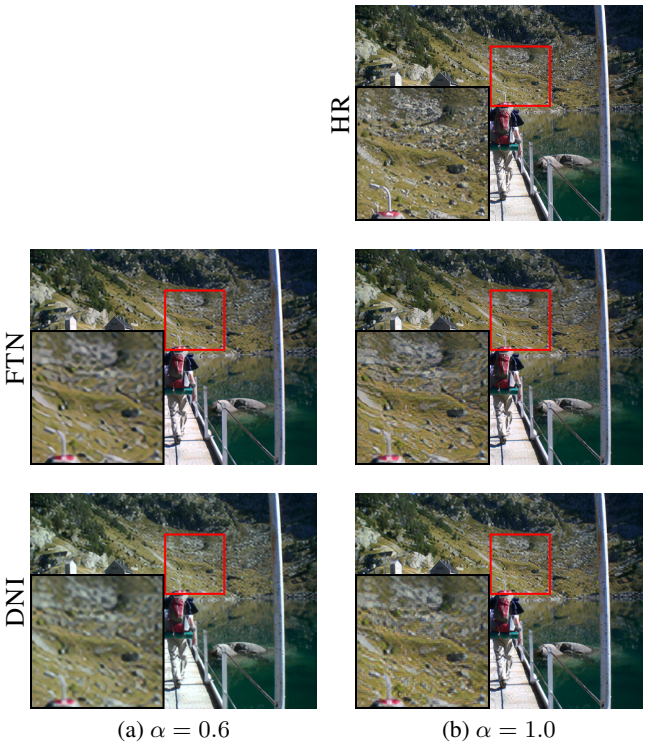
Table 3: **Filter analysis for regularization.** Distance and Similarity between the two levels. We measure Mean Average Error (MAE) for linear filter interpolation and filter-wise normalized cosine similarity. The task is PD-controllable super-resolution and baseline network is **CFSNet-30**

	FTN (G=16)	FTN	Fine-tuning
MAE	**0.0082**	0.0118	0.0139
Cos Sim.	**0.9443**	0.8937	0.8666



(a) FTN-gc16 (b) CFSNet (c) DNI

Figure 6: **Smoothness analysis for denoising.** ($\sigma = 20$ **to** $\sigma = 80$) We plot $\sigma = 40$ (red) and $\sigma = 60$ (purple) lines as linearly optimal interpolation points. For each curve, number indicates input quality factor, a denotes AdaFM-Net network and c denotes CFSNet-10 network. For example, *40a* indicates $\sigma = 40$ results on AdaFM-Net. Our FTN-gc16 results show that the choice of α is closest to the lines



(a) $\alpha = 0.6$ (b) $\alpha = 1.0$

Figure 7: **Perception-distortion controllable super-resolution results.**

is because the fine-tuning process of the DNI simply updates the parameters, without considering the initial state. Therefore, the relation between the parameters for the two levels becomes weaker compared to simple denoising tasks, and the interpolated parameters start not to behave as intended. A detailed analysis will be described in the following section.

Smoothness

In this section, we empirically analyze the definition of *smoothness* (in the introduction) and prove our filter similarity assumption that keeping the filters similar to the first level ($\mathbf{f}_i^{(1)}$) improves smoothness performance (in the proposed approach).

Filter Similarity. FTNs are designed to keep the original filters when they are tuned non-linearly to the second level. To verify this, we measure filter similarity in terms of absolute distance and cosine similarity for the super-resolution task. Table 3 shows the filter distance with the filters of the main network when they are fine-tuned (DNI) or passed through FTNs. The results show that filter-conditioned tuning is effective in preventing significant filter change, and using group convolution further restricts filter changes.

Group Convolution. Group convolution in the FTNs restricts filter changes and this restriction guarantees smoothness. Fig. 5 verifies this from the results of FTNs by changing the number of groups and AdaFM (linear version). The curves prove that the large filter similarity exhibits better interpolation performance at the expense of the second-level adaptation performance.

Color Distortion. In Fig. 4, DNI indicates unstable intermediate results in some metrics. From the intermediate images in Fig. 7 for DNI, a slight color difference can cause significant pixel-error (RMSE), but a similar value in the SSIM metric. In contrast, FTN has no such color distortions. The full results are described in the supplementary material.

Interpretability. For practical use, it will be essential for the users to know which value of α corresponds to which level. For example, in the denoising task, suppose that we train a network to work between the levels $\sigma = 20$ and $\sigma = 80$. When we set $\alpha = 0.5$, it is reasonable that the network will perform best for the level $\sigma = 50$, which is the middle point of the interval. In other words, α must be linear along with the level. Fig. 6 shows the result of the denoising task over various noise level σ of the test set and the parameter α. According to the figure, the maximum performance of our

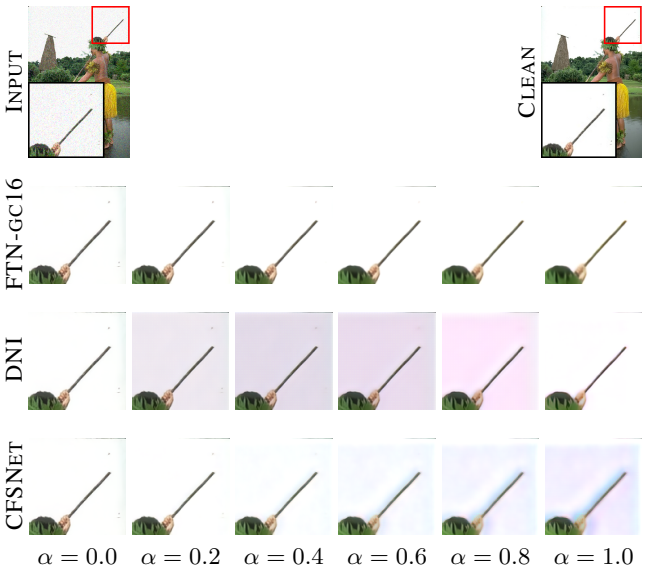


Figure 8: **Denoising results on weak noise level ($\sigma = 20$). When user controls α to a large value, over-smoothing artifacts arise**

FTN for $\sigma = 40$ and $\sigma = 60$ best matches the vertical lines of $\alpha = 0.33$ and $\alpha = 0.66$, compared to the other methods.

Over-smoothing Artifacts. In real-world applications, because the user may not know the degradation level, the user hopes to control the *strength* of the denoising. We describe our visual denoising result in an extreme case as presented in Fig. 8. In Fig. 8, the input noise level is 20, which means that the optimal results come from $\alpha = 0$ in all frameworks. $\alpha = 1$, which is optimal for noise level 80, can over-smoothen the image. When α increases, the DNI and CFSNet results reveal some color artifacts in the background, while the FTN-gc16 results are much cleaner, which is significant for the real-world feedback-based systems. Because CFSNet exploits a dual network structure, each network cannot consider the other level in the test phase. In contrast, in FTNs, two filters for both sides of the FTNs are highly correlated.

Efficiency

Complexity If any tuning layer with convolutions on a feature map is added, additional computations (MACs) are $H \times W \times K_H \times K_W \times C_{in} \times C_{out}$ where H, W, K_H, K_W, C_{in} and C_{out} are the height and width of the feature map (e.g., image size), height and width of the filter, and the number of input channels and output channels, respectively. Dominant computations arise from H and W. In our network, which is a data-independent module, only $K_H \times K_W \times C_{in} \times (C_{out}/Groups) \times N$ is needed for a single tuning layer, where N is the depth of the FTNs. As shown in Table 4, FTNs have extremely reduced computational complexity and a similar or much lower number of parameters than other frameworks in various tasks and networks.

Table 4: Overall computations, relative computations from baseline (in percentage), and number of parameters of the frameworks.

Task	Denoising		×2 Super-Resolution	
Network	AdaFM-Net		CFSNet-30	
	GFLOPs	Params(M)	GFLOPs	Params(M)
Baseline	25.11	1.41	155.96	2.37
+ CFSNet	46.96 (87.02%)	3.06	311.36 (99.64%)	4.93
+ AdaFM	26.01 (3.58%)	**1.46**	162.50 (4.20%)	**2.47**
+ **FTN**	**25.36 (0.10%)**	1.83	**156.34 (0.02%)**	3.01
+ **FTN(G=16)**	**25.13 (0.01%)**	**1.44**	**156.00 (0.00%)**	**2.41**

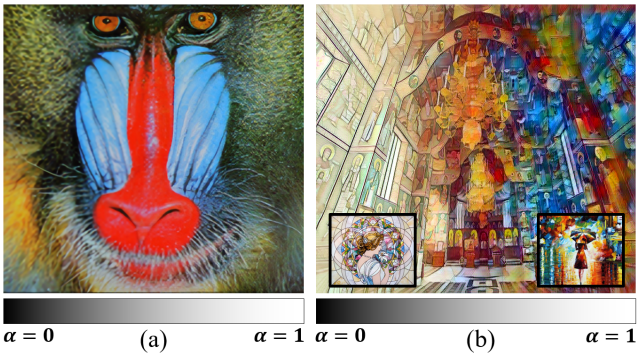


Figure 9: **Pixel-adaptive control results.** Zoom in for a better view

Pixel-adaptive Extension. Considering real-world imaging applications, the user wants to control not only the global level but also locally (pixel-wise). In this case, every pixel has its own imaging levels from $\alpha = 0$ to $\alpha = 1$. Naive pixel-adaptive control requires filters for every level, which can cause large memory issues. For efficient inference of pixel-adaptive continuous control, we propose a simple modification from the pixel-adaptive convolution. This is described as follows.

$$\begin{aligned} \mathbf{Y} &= \mathbf{X} *_{i,j} (\mathbf{f} \times (1 - \alpha_{i,j}) + FTN(\mathbf{f}) \times \alpha_{i,j}) \\ &= (\mathbf{1} - \mathbf{A}) \odot (\mathbf{X} * \mathbf{f}) + \mathbf{A} \odot (\mathbf{X} * FTN(\mathbf{f})) \end{aligned} \tag{3}$$

where $\mathbf{f}$ is the global filter, $*_{i,j}$ is the pixel-adaptive convolution, $\alpha_{i,j}$ is the per-pixel level, and $\mathbf{A}$ is the global level map that describes the pixel-wise levels. $\odot$ denotes element-wise multiplication. This modification makes implementation much simpler because only two global convolutions and multiplications are needed for pixel-adaptive control. Examples are shown in Fig. 9. We test two examples: *PD-control* and *style control*. In Fig. 9 (a), from the leftmost pixels to the rightmost pixels, the PSNR decreases and the texture becomes sharper (higher perceptual quality) continuously. In Fig. 9 (b), the pixels are smoothly stylized from one style to the other. More results with high-resolution sources can be found in the supplementary material.

Conclusion

In this paper, we propose a module called FTNs for effectively and smoothly solving continuous-level learning problems in image processing. FTNs show very smooth results

in practical applications because of their large filter similarity, producing reasonable performance on continuous levels. FTNs have fewer undesirable artifacts, more interpretability, and are extremely lightweight compared to the existing *network tuning and interpolation* frameworks. We hope that our analysis of continuous-level learning and experiments in various scenarios can help in the development of real-world imaging applications.

References

Agustsson, E.; and Timofte, R. 2017. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Blau, Y.; Mechrez, R.; Timofte, R.; Michaeli, T.; and Zelnik-Manor, L. 2018. The 2018 PIRM challenge on perceptual image super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 0–0.

Blau, Y.; and Michaeli, T. 2018. The perception-distortion tradeoff. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6228–6237.

Dong, C.; Deng, Y.; Change Loy, C.; and Tang, X. 2015a. Compression artifacts reduction by a deep convolutional network. In *Proceedings of the IEEE International Conference on Computer Vision*, 576–584.

Dong, C.; Loy, C. C.; He, K.; and Tang, X. 2015b. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence* 38(2): 295–307.

Galteri, L.; Seidenari, L.; Bertini, M.; and Del Bimbo, A. 2017. Deep generative adversarial compression artifact removal. In *Proceedings of the IEEE International Conference on Computer Vision*, 4826–4835.

Gatys, L. A.; Ecker, A. S.; and Bethge, M. 2015. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576* .

Gatys, L. A.; Ecker, A. S.; Bethge, M.; Hertzmann, A.; and Shechtman, E. 2017. Controlling perceptual factors in neural style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3985–3993.

Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Guo, S.; Yan, Z.; Zhang, K.; Zuo, W.; and Zhang, L. 2019. Toward convolutional blind denoising of real photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1712–1722.

Ha, D.; Dai, A.; and Le, Q. V. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106* .

He, J.; Dong, C.; and Qiao, Y. 2019. Modulating Image Restoration with Continual Levels via Adaptive Feature Modification Layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 11056–11064.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

Huang, X.; and Belongie, S. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, 1501–1510.

Johnson, J.; Alahi, A.; and Fei-Fei, L. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, 694–711. Springer.

Kim, J.; Kwon Lee, J.; and Mu Lee, K. 2016. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1646–1654.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Lai, W.-S.; Huang, J.-B.; Ahuja, N.; and Yang, M.-H. 2017. Deep laplacian pyramid networks for fast and accurate super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 624–632.

Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4681–4690.

Lim, B.; Son, S.; Kim, H.; Nah, S.; and Mu Lee, K. 2017. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 136–144.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer.

Liu, D.; Wen, B.; Fan, Y.; Loy, C. C.; and Huang, T. S. 2018. Non-local recurrent network for image restoration. In *Advances in Neural Information Processing Systems*, 1673–1682.

Martin, D.; Fowlkes, C.; Tal, D.; Malik, J.; et al. 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. Iccv Vancouver:.

Mildenhall, B.; Barron, J. T.; Chen, J.; Sharlet, D.; Ng, R.; and Carroll, R. 2018. Burst denoising with kernel prediction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2502–2510.

Mirza, M.; and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* .

Mittal, A.; Soundararajan, R.; and Bovik, A. C. 2012. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters* 20(3): 209–212.

Moorthy, A. K.; and Bovik, A. C. 2009. Visual importance pooling for image quality assessment. *IEEE journal of selected topics in signal processing* 3(2): 193–201.

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* .

Sheng, L.; Lin, Z.; Shao, J.; and Wang, X. 2018. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8242–8250.

Shoshan, A.; Mechrez, R.; and Zelnik-Manor, L. 2019. Dynamic-Net: Tuning the Objective Without Re-training for Synthesis Tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, 3215–3223.

Ulyanov, D.; Vedaldi, A.; and Lempitsky, V. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022* .

Wang, W.; Guo, R.; Tian, Y.; and Yang, W. 2019a. CFSNet: Toward a Controllable Feature Space for Image Restoration. In *Proceedings of the IEEE International Conference on Computer Vision*, 4140–4149.

Wang, X.; Yu, K.; Dong, C.; Tang, X.; and Loy, C. C. 2019b. Deep network interpolation for continuous imagery effect transition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1692–1701.

Wang, X.; Yu, K.; Wu, S.; Gu, J.; Liu, Y.; Dong, C.; Qiao, Y.; and Change Loy, C. 2018. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 0–0.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; Simoncelli, E. P.; et al. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13(4): 600–612.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1492–1500.

Zhang, K.; Zuo, W.; Chen, Y.; Meng, D.; and Zhang, L. 2017. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing* 26(7): 3142–3155.

Zhang, K.; Zuo, W.; and Zhang, L. 2018. FFDNet: Toward a fast and flexible solution for CNN-based image denoising. *IEEE Transactions on Image Processing* 27(9): 4608–4622.

Zhang, Y.; Li, K.; Li, K.; Wang, L.; Zhong, B.; and Fu, Y. 2018a. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 286–301.

Zhang, Y.; Tian, Y.; Kong, Y.; Zhong, B.; and Fu, Y. 2018b. Residual dense network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2472–2481.