Meta Networks for Neural Style Transfer

Falong Shen*
Peking University
shenfalong@pku.edu.cn

Shuicheng Yan
360 AI Institute
National University of Singapore
yanshuicheng@360.cn

Gang Zeng
Peking University
zeng@pku.edu.cn

Abstract

In this paper we propose a new method to get the specified network parameters through one time feed-forward propagation of the meta networks and explore the application to neural style transfer. Recent works on style transfer typically need to train image transformation networks for every new style, and the style is encoded in the network parameters by enormous iterations of stochastic gradient descent. To tackle these issues, we build a meta network which takes in the style image and produces a corresponding image transformations network directly. Compared with optimization-based methods for every style, our meta networks can handle an arbitrary new style within $19ms$ seconds on one modern GPU card. The fast image transformation network generated by our meta network is only 449KB, which is capable of real-time executing on a mobile device. We also investigate the manifold of the style transfer networks by operating the hidden features from meta networks. Experiments have well validated the effectiveness of our method. Code and trained models has been released `https://github.com/FalongShen/styletransfer`.

Introduction

Style transfer is a long-standing problem that seeks to migrate a style from a reference style image to another input picture (Efros and Freeman 2001; Efros and Leung 1999; Elad and Milanfar 2016; Frigo et al. 2016; Heeger and Bergen 1995; Kyprianidis et al. 2013). This task consists of two steps: (i) texture extraction from the style image and (ii) rendering the content image with the texture. For the first step, lots of methods have been proposed to represent the texture, most of which exploit the deep mid-layer features from the pre-trained convolutional neural network (CNN) (Gatys, Ecker, and Bethge 2015; Li et al. 2017a). Gatys et al. used the second-order statistics between feature activations across all channels in their pioneer work (Gatys, Ecker, and Bethge 2015), while Li et al. (Li et al. 2017a) found the channel-wise feature statistics (*e.g.*, mean and variance) are enough to give similar performance on representing the texture. For the second step, Gatys et al. (Gatys, Ecker, and Bethge 2015) used a gradient-descent-based method to find an optimal image, which minimizes the distance to both the content image and the style image. Despite the surprising success, their method needs thousands of iterations of gradient descent through large networks for a new input image. Recently, Johnson et al. (Johnson, Alahi, and Fei-Fei 2016) proposed an image transformation network, solving the gradient-descent-based optimization problem by one feed-forward propagation under the condition of a fixed style image. The effectiveness of this method indicates that the texture information of a style image can be encoded in one convolutional network. Several other works on image transformation networks have been proposed ever since the neural art transfer has emerged, but these pre-trained image transformation networks are limited to single (Ulyanov et al. 2016; Li and Wand 2016b) or several styles (Dumoulin, Shlens, and Kudlur 2017; Li et al. 2017b; Zhang and Dana 2017). Given a new style image, the image transformation network has to be re-trained end-to-end by an enormous number of iterations of stochastic gradient descent (SGD) to encode the new texture, which limits its scalability to large numbers of styles.

Let us re-think how the image transformation network replaces the gradient-descent-based optimization. It uses a CNN to learn a direct-mapping from the input image to the near-optimal solution image (Johnson, Alahi, and Fei-Fei 2016). To obtain such an image transformation network, we need to minimize the empirical loss on the training content images for a fixed style image, which can be solved by SGD (Johnson, Alahi, and Fei-Fei 2016).

Then we would ask a question, "*Is SGD the only method to get the solution network for a new style image?* "

The answer is *No*. As our target is to get a near-optimal network and the input is a style image, it is natural to build a direct mapping between the two domains. Instead of SGD, we propose to build a meta network which takes in the style image and produces the corresponding image transformation network.

The meta network is composed of a frozen VGG-16 network (Simonyan and Zisserman 2014) which extracts texture features from the given style image, and a series of fully connected layers to project the texture features to the transformation network space. It is optimized by the empirical risk minimization across both the training content images and the style images. In this way, for the first time, we provide a new approach to generate an image transformation network for neural style transfer. We name it the "meta network" because it is able to produce different networks for different style images. The model architecture is depicted in Figure 1.

*This work was done when Falong Shen was an Intern at 360 AI Institute.

The image transformation network is embedded as a hidden vector in the bottle-neck layer of the meta network. By interpolating the embedding hidden vectors of two networks induced from two real textures, we verify that the meta network generalizes the image textures rather than simply memorizing them.

The contributions of this paper are summarized as follows:

- We address the network generation task and provide a meta network to generate networks for a specific domain. Specifically, the meta network takes in the new style image and produces a corresponding image transformation network in one feed-forward propagation.
- Our method provides an explicit representation of image transformation networks for neural style transfer, which enables texture synthesis and texture generation naturally.
- The generated networks from the meta network have the similar performance compared with SGD-based methods, but with orders of magnitude faster speed ($19ms$ v.s. $4h$) for a new style.
- We provide a new perspective on the algorithms for neural style transfer, which indicates that convolutional neural networks can be applied to optimization problems.

Related Work

Hypernetworks and Meta Networks. A hypernetwork is a small network which is used to generate weights for a larger network. HyperNEAT (Stanley, D'Ambrosio, and Gauci 2009) takes in a set of virtual coordinates to produce the weights. Recently, Ha et al. (Ha, Dai, and Le 2016) proposed to use static hypernetworks to generate weights for a convolutional neural network and to use dynamic hypernetworks to generate weights for recurrent networks, where they took the hypernetwork as a relaxed form of weight sharing.

The works on meta networks adopt a two-level learning, a slow learning of a meta-level model performing across tasks and a rapid learning of a base-level model acting within each task (Mitchell, Thrun, and others 1993; Vilalta and Drissi 2002). Munkhdalai et al. (Munkhdalai and Yu 2017) proposed a kind of meta networks for one-shot classification via fast parameterization for the rapid generalization .

Style Transfer. Gatys et al. (Gatys, Ecker, and Bethge 2015) for the first time proposed the combination of content loss and style loss based on the pre-trained neural networks on ImageNet (Deng et al. 2009). They approached the optimal solution image with hundreds of gradient descent iterations and produced high quality results. Then (Johnson, Alahi, and Fei-Fei 2016) proposed to use image transformation networks to directly approach the near-optimal solution image instead of gradient descent. However, it needs to train an image transformation network for each new style, which is time consuming.

Recently, lots of improvements on (Gatys, Ecker, and Bethge 2015) have been made. Instead of using the gram matrix of feature maps to represent the style, Li et al. (Li et al. 2017a) demonstrated that several other loss functions can also work, especially the mean-variance representation, which is much more compact than the gram matrix representation while giving similar performance. There are also other representations of the style, such as histogram loss (Risser, Wilmot, and Barnes 2017), MRF loss (Li and Wand 2016a) and CORAL loss (Peng and Saenko 2017). Dumoulin et al. (Dumoulin, Shlens, and Kudlur 2017) proposed to use conditional instance normalization to accommodate each style. This method adjusts the weights of each channel of features and successfully represents many different styles. Unfortunately, it cannot be generalized to a new style image. Huang et al. (Huang and Belongie 2017) found matching the mean-variance statistics of features from VGG-16 between the style image and the input image is enough to transfer the style. Although this method is able to process arbitrary new style, it heavily relies on a VGG-16 network to encode the image and also decode the feature by a corresponding network, which makes it difficult to control the model size. Chen et al. (Chen and Schmidt 2016) introduced a style swap layer to handle an arbitrary style transfer. Similar as the work (Huang and Belongie 2017), they also proposed to adjust the feature in the content image towards the style image but in a patch-by-patch manner, which is much slower.

Proposed Method

To find the optimal point of a function, gradient descent is typically adopted. To find the optimal function for a specific task, the traditional method is to parameterize the function and optimize the loss function on the training data by SGD. In this paper, we propose meta networks to find the near-optimal network directly, which will be detailed in this section. We also discuss its application to neural style transfer.

Meta Networks

Denote $f(x)$ and $h(x)$ as fixed differentiable functions and $||\cdot||$ is a norm. Let us consider an optimization problem of this kind:

$$||f(x) - f(a)|| + \lambda||h(x) - h(b)||. \tag{1}$$

There are three situations depending on whether a and b are fixed or variable.

Situation 1: $a = a_0$ *and* $b = b_0$. *Both are fixed.*

If $f(x)$ and $h(x)$ are convex functions, Eqn. (1) is a typical convex optimization problem with regard to x and we can apply gradient descent to obtain the optimal x directly.

$$\operatorname*{argmin}_x ||f(x) - f(a_0)|| + \lambda||h(x) - h(b_0)||. \tag{2}$$

Situation 2: $b = b_0$. *a is variable.*

For any given a, according to **Situation 1**, we have a corresponding x which satisfies the function. That is, there always exists a mapping function

$$\mathcal{N} : a| \rightarrow x. \tag{3}$$

We represent the mapping function as a deep neural network $\mathcal{N}(a; w)$ which is parameterized by w. Now consider the *empirical risk minimization* (ERM) problem to find the optimal mapping function:

$$\operatorname*{argmin}_w \sum_a ||f(x) - f(a)|| + \lambda||h(x) - h(b_0)|| \tag{4}$$

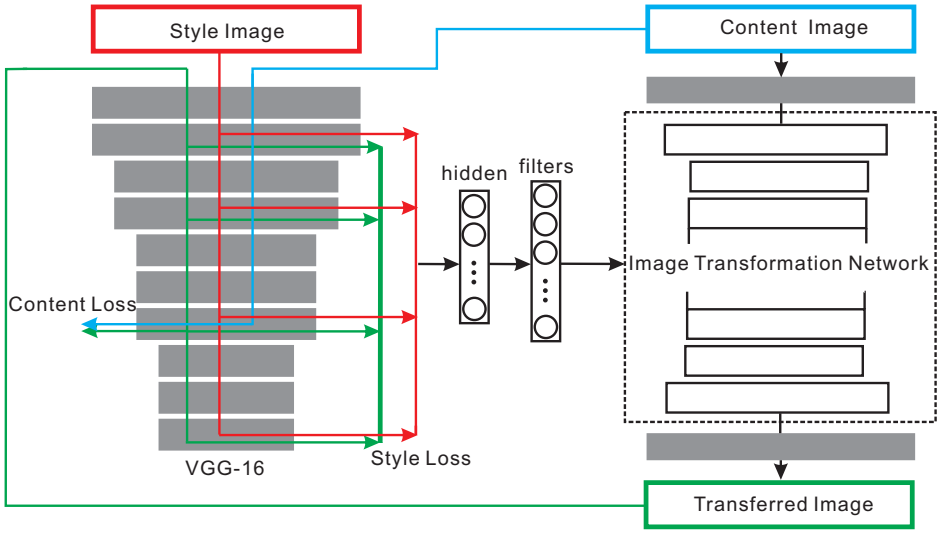

Figure 1: Model architecture. The style image is fed into the fixed VGG-16 to get the style feature, which goes through two fully connected layers to construct the filters for each `conv` layer in the corresponding image transformation network. We fixed the scale and bias in the instance `batchnorm` layer to 1 and 0. The dimension of hidden vector is 1792 without specification. The hidden features are connected with the filters of each `conv` layer of the network in a group manner to decrease the parameter size, which means a 128 dimensional hidden vector for each `conv` layer. Then we compute the style loss and the content loss for the transferred image with the style image and the content image respectively through the fixed VGG-16.

where $x = \mathcal{N}(a; w)$. However, it is not that trivial to find the optimal mapping function in this situation. The main difficulty lies in optimization and the design of $\mathcal{N}(\cdot; w)$. The function $f(\cdot)$ is also important for deciding whether a near-optimal mapping function can be approached by SGD.

Situation 3: *Both a and b are variable.*

For any given b, according to **Situation 2**, there exists a mapping function which is able to find the near-optimal x given a. Suppose this optimal mapping function is parameterized by w in $\mathcal{N}(\cdot; w)$ and can be approached by SGD. Under this condition, there exists a mapping function:

$$meta\mathcal{N} : b| \rightarrow \mathcal{N}(\cdot; w). \tag{5}$$

Similar to **Situation 2**, we again represent this mapping function as a deep neural network $meta\mathcal{N}(b; \theta)$ which is parameterized by θ. The meta network $meta\mathcal{N}(b; \theta)$ takes in b as input and produces a near-optimal network to Eqn. (2). Instead of iterations of SGD, the meta network needs only one feed forward propagation to find the near-optimal network.

To optimize θ in the meta network, we consider the following ERM problem:

$$\underset{\theta}{\operatorname{argmin}} \sum_{b} \sum_{a} ||f(x) - f(a)|| + \lambda ||h(x) - h(b)||, \tag{6}$$

where $x = \mathcal{N}(a; w)$, and $w = meta\mathcal{N}(b; \theta)$. For every given b, there exists an optimal w, thus in the training stage it needs iterations of SGD to update the meta network parameter θ in order to produce an appropriate w for each given b.

Neural Style Transfer

Neural style transfer was first proposed by Gatys et al. (Gatys, Ecker, and Bethge 2015) to render a content image in the style of another image based on features extracted from a pre-trained deep neural network like VGG-16 (Simonyan and Zisserman 2014). There are four situations depending on whether the content image and the style image are fixed or variable.

Fixed Content Image and Fixed Style Image For a pair of given images (I_s, I_c), the target is to find an optimal image I which minimizes the perceptual loss function to combine the style of I_s and the content of I_c:

$$\min_{I} \Big(\lambda_c ||\mathbf{CP}(I; w_f) - \mathbf{CP}(I_c; w_f)||_2^2 + \lambda_s ||\mathbf{SP}(I; w_f) - \mathbf{SP}(I_s; w_f)||_2^2 \Big) \tag{7}$$

where $\mathbf{SP}(\cdot; w_f)$ and $\mathbf{CP}(\cdot; w_f)$ are perceptual functions based on pre-trained deep neural networks parameterized by the fixed weights w_f, which represent the style perceptron and the content perceptron individually. λ_s and λ_c are scalars. According to **Situation 1**, it is straightforward to apply gradient descent to the whole networks and use the gradient information from back-propagation to synthesize an image to minimize the loss function. This method produces high quality results for any given style image, but needs hundreds of optimization iterations to get a converged result for each sample, which brings a large computation burden.

Variable Content Image and Fixed Style Image Instead of directly using the gradient information with regard to the input image to synthesize a new image, Johnson et al. (Johnson, Alahi, and Fei-Fei 2016) applied feed-forward image transformation to generate the target image. The image transformation networks are optimized across a large natural image dataset using the gradient of the parameters by back

propagation:

$$\min_{w} \sum_{I_c} \Big(\lambda_c ||\mathbf{CP}(I_w; w_f) - \mathbf{CP}(I_c; w_f)||_2^2 + \lambda_s ||\mathbf{SP}(I_w; w_f) - \mathbf{SP}(I_s; w_f)||_2^2 \Big), \tag{8}$$

where $I_w = \mathcal{N}(I_c; w)$ and $\mathcal{N}$ is the image transformation network which is parameterized by w. The style of I_s is encoded in w. For a new content image, it only needs a forward propagation through the transformation network to generate the transferred image.

Fixed Content Image and Variable Style Image As aforementioned in **Situation 2**, for some specified condition, there is a direct-mapping parameterized by CNN to approach the near-optimal solution for the loss. We consider the symmetry question. For a fixed content image, we try to find the transferred image for every style image and get the direct-mapping by

$$\min_{w} \sum_{I_s} \Big(\lambda_c ||\mathbf{CP}(I_w; w_f) - \mathbf{CP}(I_c; w_f)||_2^2 + \lambda_s ||\mathbf{SP}(I_w; w_f) - \mathbf{SP}(I_s; w_f)||_2^2 \Big), \tag{9}$$

where $I_w = \mathcal{N}(I_c; w)$. However, it has been found that it is not able to find the right mappings. The image transformation network only gives the style image as the transferred image, indicating that it is not that trivial to get a direct mapping to approach the gradient-descent solution.

Variable Content Image and Variable Style Image To find a transformation network $\mathcal{N}(\cdot; w)$ which is parameterized by w for a given style image I_s, it needs tens of thousands of iterations of SGD to get a satisfied network. As shown in **Situation 3**, we propose a natural way to get the transformation network $\mathcal{N}(\cdot; w)$ directly by a meta network:

$$Meta\mathcal{N} : I_s \rightarrow \mathcal{N}(\cdot; w). \tag{10}$$

The meta network is parameterized by θ and is optimized across a dataset of content images and a dataset of style images by

$$\min_{\theta} \sum_{I_c, I_s} \Big(\lambda_c ||(\mathbf{CP}(I_{w_\theta}; w_f) - \mathbf{CP}(I_c; w_f))||_2^2 + \lambda_s ||(\mathbf{SP}(I_{w_\theta}; w_f) - \mathbf{SP}(I_s; w_f))||_2^2 \Big), \tag{11}$$

where $I_{w_\theta} = \mathcal{N}(I_c; w_\theta)$ and $w_\theta = Meta\mathcal{N}(I_s; \theta)$. The style image I_s is used as the supervised target in the loss fucntion as well as input features to the meta network, which means the meta network takes a style image as input and generates a network which is able to transfer the content image towards the style image. Algorithm 1 details the training strategy.

Experiment

Implementation Details

The meta networks for neural style transfer are trained on the content images from MS-COCO (Lin et al. 2014) *trainval* set and the style images from the *test* set of the WikiArt dataset (Nicol. 2016). There are about $120k$ images in MS-COCO *trainval* set and about $80k$ images in the *test* set of WikiArt. During training, each content image or style image is resized to keep the smallest dimension in the range $[256, 480]$, and randomly cropped regions of size 256×256. We use Adam (Kingma and Ba 2014) with fixed learning rate 0.001 for $600k$ iterations without weight decay. The batch size of content images is 8 and the meta network is trained for 20 iterations before changing the style image. The transferred images are regularized with total variations loss with a strength of 10. Our image transformation network shares the same structure with (Johnson, Alahi, and Fei-Fei 2016), except that we remove the `Tanh` layer at last and the instance `BN` layer after the first `Conv` layer.

We compute the content loss at the `relu3_3` layer and the style loss at layers `relu1_2`, `relu2_2`,`relu3_3` and `relu4_3` of VGG-16. The weight of content loss is 1 while the weight of style loss is 250. The content loss is the Euclidean distance between two feature maps of the content image and the transferred image. We compute the mean and stand deviations of two feature maps of the style image and the transferred image as style features.

Algorithm 1 Minibatch stochastic gradient descent training of meta networks for neural style transfer. We use $k = 20$ and $m = 8$ in our experiments.

for number of training iterations **do**
- Sample a style image I_s.

for k steps **do**
- Feed-forward propagation of the meta network to get the transformation network

$$w \leftarrow meta\mathcal{N}(I_s; \theta).$$

- Sample minibatch of m input images $\{I_c^{(1)}, ..., I_c^{(m)}\}$.
- Feed-forward propagation of the transformation network to get transferred images.
- Computing the content loss and style loss and update θ

$$\nabla_\theta \sum_{I_c} \Big(\lambda_c ||(\mathbf{CP}(I) - \mathbf{CP}(I_c))||_2^2 + \lambda_s ||(\mathbf{SP}(I) - \mathbf{SP}(I_s))||_2^2 \Big)$$

end for
end for

(a) Input (b) Style (c) Gatys *et al.* (d) John *et al.* (e) Ours (e) Ours-Fast

Figure 2: Qualitative Comparisons of different methods. Both the style images and the content images are unseen in the *train* set for *our* model.

Comparison with Other Methods

We compare our method with other style transfer methods to evaluate its effectiveness and efficiency. Gatys et al. (Gatys, Ecker, and Bethge 2015) proposed to find the optimal image by gradient descent and Johnson et al. (Johnson, Alahi, and Fei-Fei 2016) proposed to find the near-ptimal image transformation network by SGD. Compared with these two previous works, our meta network produces the image transformation network for each style by one forward-propagation. As the gradient-descent-based optimization and direct-mapping-based optimization share the same loss function, we make the comparison in terms of the converged loss, transferred image quality and running speed.

We show example style transfer results of different methods in Figure 2. Visually, there is no significantly difference between the three methods. Note that both the style image and the content image are unseen during the training stage of our meta networks while the model in (Johnson, Alahi, and Fei-Fei 2016) needs to be specifically trained end-to-end for every style. The image transformation network of our model and the model in (Johnson, Alahi, and Fei-Fei 2016) share the same architecture, our model can be easily generalized to any new style by only one forward propagation of the meta-network. According to our experiments, it costs about $19ms$ to produce an image transformation network which is used in (Johnson, Alahi, and Fei-Fei 2016) for a single Titan X GPU card.

Table 1 lists the advantages and defects of these methods. The gradient descent method of optimization in (Gatys, Ecker, and Bethge 2015) is the most flexible but cannot process images in real time. The method of (Huang and Belongie 2017) is restricted by the VGG-16 network. Our method enjoys the flexibility while can process images in real time for both encoding the style and transferring images. Figure 3 plots the training curve for image transformation networks with different `conv` filter numbers. In our experiments, the converged losses on the training styles and testing styles are both around 4.0×10^5 for $dim32$, which equal approximately 200 steps of gradient descent in (Gatys, Ecker, and Bethge 2015) (our own implementation).

Table 1: Comparison of different models. Our transfer network shares a similar structure with but is much faster to encode the new style. The model in can also be adapted to the new style efficiently at the price of relying on a VGG-16 network in the transferring stage. Both the style image and content image are resized to 256×256. Time is measured on a Pascal Titan X GPU.

Method	*Time(encode)*	*Time(transfer)*	*Size(transfer)*	*#Style*
(Gatys, Ecker, and Bethge 2015)	N/A	$9.52s$	N/A	∞
(Johnson, Alahi, and Fei-Fei 2016)	$4h$	$15ms$	7MB	1
(Chen and Schmidt 2016)	$407ms$	$171ms$	10MB	∞
(Huang and Belongie 2017)	$27ms$	$18ms$	25MB	∞
Ours	$19ms$	$15ms$	7MB	∞
Ours-Fast	$11ms$	$8ms$	449KB	∞

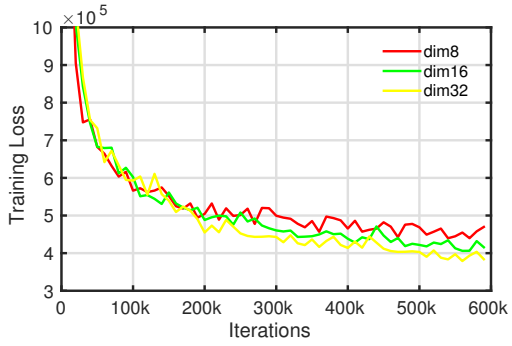

Figure 3: Training curves for image transformation networks of different model size. `dim32` denotes the filter numbers is 32 in the first `conv` layer.

Difference with SGD solution. Comparing to (Johnson, Alahi, and Fei-Fei 2016), our meta network also takes in all the style features and generates almost the whole image transformation networks. Similar to SGD solution, our meta network has both the complete supervised information (all style features) and the total freedom (almost the whole image transformation networks). Therefore our image transformation networks can adopt any other useful structure, which is the same with SGD. To show the effectiveness of the meta network, we have experimented with much smaller image transformation networks (449KB), the network architecture is characterized in Table 2. This small and fast image transformation networks also give satisfying results as shown in Figure 4 (best viewed in color).

Difference with AdaBN solution. Recently, Huang et al. (Huang and Belongie 2017) also proposed an approach which can process an arbitrary style. They transfer images by adjusting the statistics (mean and std deviations) of the feature map in `AdaIN` layer and achieve surprisingly good results. However, their model needs to encode the image by fixed VGG-16 and decode the feature, which also requires a VGG-16-like architecture. This makes their image transformation network relatively large (∼25MB). The difference between this work and our method lies in the freedom of image transformation network for different style image. Huang et al. (Huang and Belongie 2017) aligns the mean and variance of the content feature maps to those of style feature maps, which means the style image only injects a 1024-dim vector to influence one layer in image transformation networks. The influence is quite limited because not all style features in supervised stage is used and only one layer is changed for different styles.

The key to success of the work in (Huang and Belongie 2017) is the high-level encoded feature from VGG-19. It should be noticed that they rely on VGG for both style image and image transformation networks. According to our experiments, their method fails when the pre-trained VGG-19 in image transformation networks is replaced. In the work by (Huang and Belongie 2017), both the encoder (fixed to VGG) and decoder (trained) of image transformation networks is fixed except the `AdaIN` layer for any given style image. Although the image transformation networks is large (∼25MB), it has little freedom for a new style image.

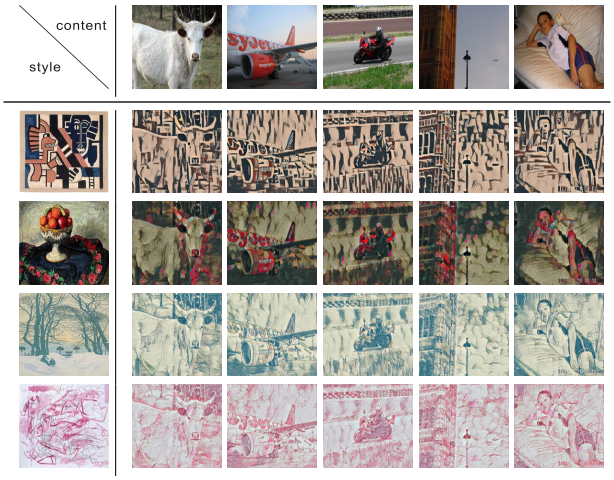

Figure 4: Examples of style transfer by the fast version of our meta networks. The size of image transformation network is 449KB, which is able to real-time execute on a mobile device. Best viewed in color.

layer	activation size
input	$3 \times 256 \times 256$
Reflection Padding (40×40)	$3 \times 336 \times 336$
$8 \times 9 \times 9$ conv, stride 1	$8 \times 336 \times 336$
$16 \times 3 \times 3$ conv, stride 2	$16 \times 168 \times 168$
$32 \times 3 \times 3$ conv, stride 2	$32 \times 84 \times 84$
Residual block, 32 filters	$32 \times 80 \times 80$
Residual block, 32 filters	$32 \times 76 \times 76$
Residual block, 32 filters	$32 \times 72 \times 72$
Residual block, 32 filters	$32 \times 68 \times 68$
Residual block, 32 filters	$32 \times 64 \times 64$
$16 \times 3 \times 3$ deconv, stride 2	$16 \times 128 \times 128$
$8 \times 3 \times 3$ deconv, stride 2	$8 \times 256 \times 256$
$3 \times 9 \times 9$ conv, stride 1	$3 \times 256 \times 256$

Table 2: The fast version of image transformation networks. Every `conv` layer except for the first and the last is followed by a instance `batchnorm` layer and a `relu` layer sequentially, which are omitted in the table for clarity. All the filters of the `conv` layers in pink region are generated by meta networks in the inference stage. The filters of the `conv` layers in gray region are jointly trained with meta networks and are fixed in the inference stage. There is no parameter in other layers. The model size is only 449KB, which is capable of executing on a mobile device.

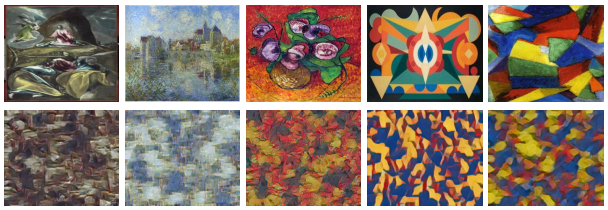

Figure 5: Texture Visualization. The first row displays style images and the second row displays the corresponding textures encoded in the image transformation networks.

Figure 6: Interpolations of different styles. The fist two rows show the combination of two styles in the hidden state. In the third row, by feeding the content image to the meta network we can get an identity transformation network, which enables the control of the strength of the other style.

Figure 7: Randomly generated styles at varying hidden state $h \in \{14, 224\}$.

Additional experiments

In this subsection, we further explore the fascinating features of the network manifold from the meta network.

Texture Visualization. After one forward-propagation, the meta network encodes the style image into the image transformation network. To better understand what kind of target style is learned, we visualize the texture by feeding Gaussian white noise as the content image to the image transformation network to get the uniform texture. Figure 5 displays the image-texture pair.

Style Interpolation. Because of an explicit representation of every network, we could compute the linear interpolation of the hidden states and get a sensible network, which proves the space continuity in the network manifold. Figure 6 displays interpolations of hidden states between two real styles. These style images are not part of the training style images. The first and last columns contain images transferred by real style images while the images in between are the results of linear interpolation in hidden states. In the third row, we treat the content image as the style image and get an `identity transformation network`. By interpolating the hidden state of the identity transformation network and style image transformation network, we can control the strength of the style.

Texture Generation. Our meta network could also produce sensible texture given random hidden states. Figure 7 shows some representative stylish images drawn uniformly from the hidden state. We observe varied light exposure, color and pattern. The meta network works well, possessing a large diversity of styles. We compare the effects of the varying dimensions of hidden states. Apparently, the large dimension of hidden states makes better style images.

Conclusion

We introduce the meta network, a novel method to generate the near-optimal network instead of stochastic gradient descent. Especially for neural style transfer, the meta network takes in any given style image and produces a corresponding image transformation network. Our approach provides an efficient solution to real time neural style transfer of any given style. We also explore the network manifold by operating on the hidden state in the meta network. From the experimental results that validate the faster speed of our method with similar performance, we can see that meta networks and direct-mapping for optimization have a successful application to neural style transfer.

References

[Chen and Schmidt 2016] Chen, T. Q., and Schmidt, M. 2016. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*.

[Deng et al. 2009] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 248–255. IEEE.

[Dumoulin, Shlens, and Kudlur 2017] Dumoulin, V.; Shlens, J.; and Kudlur, M. 2017. A learned representation for artistic style.

[Efros and Freeman 2001] Efros, A. A., and Freeman, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 341–346. ACM.

[Efros and Leung 1999] Efros, A. A., and Leung, T. K. 1999. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, 1033–1038. IEEE.

[Elad and Milanfar 2016] Elad, M., and Milanfar, P. 2016. Style-transfer via texture-synthesis. *arXiv preprint arXiv:1609.03057*.

[Frigo et al. 2016] Frigo, O.; Sabater, N.; Delon, J.; and Hellier, P. 2016. Split and match: example-based adaptive patch sampling for unsupervised style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 553–561.

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

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

[Heeger and Bergen 1995] Heeger, D. J., and Bergen, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 229–238. ACM.

[Huang and Belongie 2017] Huang, X., and Belongie, S. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. *arXiv preprint arXiv:1703.06868*.

[Johnson, Alahi, and Fei-Fei 2016] 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.

[Kingma and Ba 2014] Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kyprianidis et al. 2013] Kyprianidis, J. E.; Collomosse, J.; Wang, T.; and Isenberg, T. 2013. State of the" art": A taxonomy of artistic stylization techniques for images and video. *IEEE transactions on visualization and computer graphics* 19(5):866–885.

[Li and Wand 2016a] Li, C., and Wand, M. 2016a. Combining markov random fields and convolutional neural networks for image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2479–2486.

[Li and Wand 2016b] Li, C., and Wand, M. 2016b. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, 702–716. Springer.

[Li et al. 2017a] Li, Y.; Wang, N.; Liu, J.; and Hou, X. 2017a. Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036*.

[Li et al. 2017b] Li, Y.; Fang, C.; Yang, J.; Wang, Z.; Lu, X.; and Yang, M.-H. 2017b. Diversified texture synthesis with feed-forward networks. *arXiv preprint arXiv:1703.01664*.

[Lin et al. 2014] 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.

[Mitchell, Thrun, and others 1993] Mitchell, T. M.; Thrun, S. B.; et al. 1993. Explanation-based neural network learning for robot control. *Advances in neural information processing systems* 287–287.

[Munkhdalai and Yu 2017] Munkhdalai, T., and Yu, H. 2017. Meta networks. *arXiv preprint arXiv:1703.00837*.

[Nicol. 2016] Nicol., K. 2016. Painter by numbers. wikiart. *https://www.kaggle.com/c/painter-by-numbers*.

[Peng and Saenko 2017] Peng, X., and Saenko, K. 2017. Synthetic to real adaptation with deep generative correlation alignment networks. *arXiv preprint arXiv:1701.05524*.

[Risser, Wilmot, and Barnes 2017] Risser, E.; Wilmot, P.; and Barnes, C. 2017. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*.

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

[Stanley, D'Ambrosio, and Gauci 2009] Stanley, K. O.; D'Ambrosio, D. B.; and Gauci, J. 2009. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life* 15(2):185–212.

[Ulyanov et al. 2016] Ulyanov, D.; Lebedev, V.; Vedaldi, A.; and Lempitsky, V. 2016. Texture networks: Feed-forward synthesis of textures and stylized images. In *Int. Conf. on Machine Learning (ICML)*.

[Vilalta and Drissi 2002] Vilalta, R., and Drissi, Y. 2002. A perspective view and survey of meta-learning. *Artificial Intelligence Review* 18(2):77–95.

[Zhang and Dana 2017] Zhang, H., and Dana, K. 2017. Multi-style generative network for real-time transfer. *arXiv preprint arXiv:1703.06953*.