Neural Style Transfer via Meta Networks

Falong Shen[1]* Shuicheng Yan[2,3] Gang Zeng[1]
[1]Peking University [2]360 AI Institute [3]National University of Singapore
{shenfalong,zeng}@pku.edu.cn, yanshuicheng@360.cn

Abstract

In this paper we propose a noval method to generate the specified network parameters through one feed-forward propagation in the meta networks for 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, which lacks the generalization ability to new style in the inference stage. To tackle these issues, we build a meta network which takes in the style image and generates a corresponding image transformation network directly. Compared with optimization-based methods for every style, our meta networks can handle an arbitrary new style within 19 milliseconds on one modern GPU card. The fast image transformation network generated by our meta network is only 449 KB, which is capable of real-time running 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 will be released.

1. Introduction

Style transfer is a long-standing problem that aims to migrate a style from a reference style image to another input picture [6, 7, 8, 12, 19]. 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) [9]. Gatys *et al.* used the second-order statistics between feature activations across all channels in their pioneer work [9], while Li *et al.* [23] 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.* [9] used a gradient-descent-based method to find an optimal

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

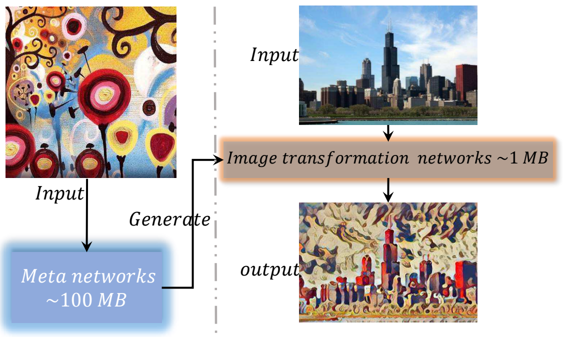


Figure 1: Flowchart of the meta network solution for neural style transfer. The new style image is uploaded to the meta networks, which generate an image transformation network by performing one forward propagation (cost about 20 ms). The architecture (layer numbers, filter kernel size, *etc.*) of the image transformation network is pre-defined while the weight values are generated by meta networks. Once the image transformation network is generated, it is able to real-time transfer any content image to the target style.

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.* [17] 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 [33, 21] or several styles [5, 22, 35]. 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 [17]. 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 [17].

Inspired by the work [17] which indicates the gradient-descent-based optimization is not the only method to find the local minima for a content image, we would further ask a question, "*Is SGD the only method to get the solution network for a new style image?* "

As the target output is 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 as shown in Figure 1.

The meta network is composed of a frozen VGG-16 network [31] 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 generate different networks for different style images. The model architecture is depicted in Figure 2.

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.

Recently many progresses have been made in video style transfer on mobile device [13, 1]. The meta network enjoys the feature of the separation of style image and content image, which means that the user can upload a style image to the cloud server and download the corresponding image transformation network generated by the meta network. The generated image transformation network from the meta network is capable of real-time running on a mobile device.

The contributions of this paper are summarized as follows:

- We address the network generation task and provide a meta network to generate networks for neural style transfer. 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 similar performance compared with SGD-based methods, but with orders of magnitude faster speed (19 ms vs. 4 h) to adapt to a new style.
- We provide a new perspective on algorithm design for neural style transfer, which indicates convolutional neural networks can be applied to optimization problems.

2. Related Work

Hypernetworks and Meta Networks. A hypernetwork is a small network which is used to generate weights for a larger network. HyperNEAT [32] takes in a set of virtual coordinates to produce the weights. Recently, Ha *et al.* [11] 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, where a slow learning of a meta-level model performing across tasks and a rapid learning of a base-level model acting within each task [25, 34]. Munkhdalai and Yu [26] proposed a kind of meta networks for one-shot classification via fast parameterizations for the rapid generalization . Similarly to our works, dynamic filter networks were proposed to generated filters dynamically conditioned on an input [28, 15].

Style Transfer. Gatys *et al.* [9] for the first time proposed the combination of content loss and style loss based on the pre-trained neural networks on ImageNet [4]. They approached the optimal solution image with hundreds of gradient descent iterations and produced high quality results. Then [17] 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, considerable improvements on [9] have been made. Instead of using the gram matrix of feature maps to represent the style, Li *et al.* [23] demonstrated that several other loss functions can also work, especially the mean-variance representation that 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 [30], MRF loss [20] and CORAL loss [29]. Chen *et al.* [2] built a explicit representation for each style. Dumoulin *et al.* [5] 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 and Belongie [14] 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

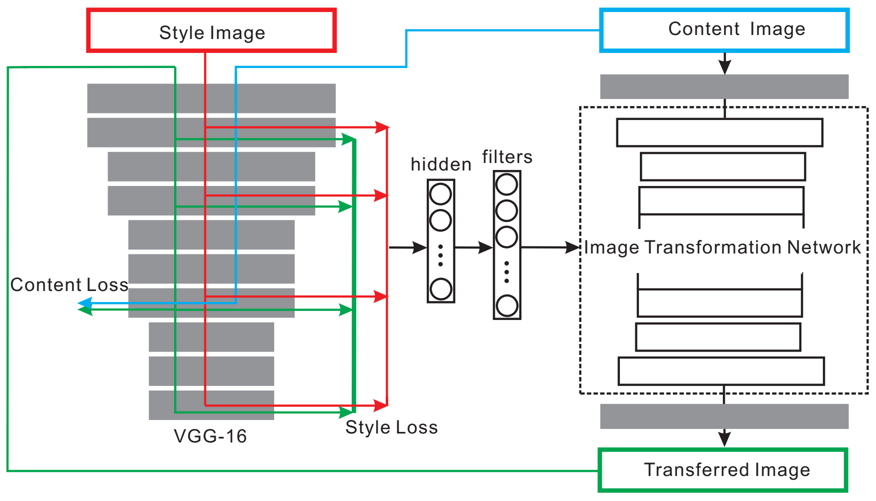


Figure 2: Model architecture. The style image is fed into the fixed VGG-16 to get the style feature, which goes through two fully connected layers to construct the filters for each `conv` layer in the corresponding image transformation network. We fixed the scale and bias in the instance `batchnorm` layer to 1 and 0. The dimension of `hidden` vector is 1792 without specification. The dimension of `filters` vector is in the range from 1×10^5 to 2×10^6, depending on the size of image transformation networks. 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.

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 and Schmidt [3] introduced a style swap layer to handle an arbitrary style transfer. Similar to [14] 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. Another concurrent works by [10] propose learning to predict the conditional instance normalization parameters from a style image. Similar to [14] their method also exploit the the mean-variance statistics of features and their image transformation network has less freedom to adapt to different styles comparing to our method.

3. Meta Networks for Neural Style Transfer

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 empirical loss function on the training data by SGD.

In this paper, we propose meta networks to find the near-optimal network for neural style transfer directly, which will be detailed in this section.

3.1. Definition of Neural Style Transfer

Neural style transfer was first proposed by Gatys *et al.* [9] 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 [31]. For a pair of given images (I_s, I_c), the target of neural style transfer 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:

$$\min_{I_x} \mathbf{PLoss}(I_x | I_c, I_s), \tag{1}$$

where the perceptron loss function **PLoss** is defined as the Euclidean distances between the output features from two perceptrons

$$\lambda_c ||\mathbf{CP}(I) - \mathbf{CP}(I_c)||_2^2 + \lambda_s ||\mathbf{SP}(I) - \mathbf{SP}(I_s)||_2^2. \tag{2}$$

In the formulation of Gatys *et al.* [9], the content perceptron **CP** is defined as the features from VGG-16 `relu3_3` and the style perceptron **SP** is defined as the gram matrix of features from VGG-16 `relu1_2`, `relu2_2`, `relu3_3` and `relu4_3`. λ_s and λ_c are scalars which are used to balance the importance of the content image and the style image.

For a fixed content image I_c and a fixed style image I_s, it is straightforward to apply the gradient descent method to Equation 1 as the loss function is founded on the convolutional neural network which is differentiable. Started from a random image I_0, the gradient information from back-propagation is used to synthesize an image to minimize the loss function. According to the definition of the loss function of neural style transfer, this method often produces high quality results for the given style images and content images because gradient descent method is good at finding local minima. However, it needs hundreds of optimization iterations to get a converged result for each sample, which can bring a large computation burden.

3.2. Fast Neural Style Transfer via Image Transformation Networks

If the style image I_s is fixed, then for a given content image I_c, there is always a solution image I_x^* which minimizes the loss function in Equation 1. The optimal image I_x^* can be approached by gradient descent. That is, there always exists a mapping function

$$\mathcal{N} : I_c| \rightarrow I_x^*. \tag{3}$$

A simple solution to get the mapping function $\mathcal{N}$ in Equation 3 is to construct both the domain $\{I_c\}$ and the domain $\{I_x^*\}$ and train an end-to-end neural network by SGD using a pre-defined loss function. Instead of explicitly finding out the solution image I_x^* for every content image I_c, Johnson *et al.* [17] approached the image transformation networks by optimized the *empirical risk minimization* (ERM) problem across a large natural image dataset,

$$\min_w \sum_{I_c \in \mathcal{D}_c} \mathbf{PLoss}(I_x|I_c, I_s), \tag{4}$$

where $I_x = \mathcal{N}(I_c; w)$ and $\mathcal{N}$ is the image transformation network which is parameterized by w. I_c is the content image from a large natural image dataset $\mathcal{D}_c$ and I_s is a fixed given style image. The content image I_c is used as one of the supervised target in the **Ploss** function as well as input features to the transformation network $\mathcal{N}$. The image transformation network $\mathcal{N}$ is optimized using the gradient of the parameters by back propagation. Once the training stage is finished, we get an image transformation network $\mathcal{N}$ which encodes the style of I_s in parameter w. For a new content image I_c, it only needs a forward propagation through the transformation network $\mathcal{N}$ to generate the transferred image I_x^*.

3.3. Meta Networks for Neural Style Transfer

To find an image transformation network $\mathcal{N}(\cdot; w)$ which is parameterized by w for a given style image I_s, it needs tens of thousands of SGD iterations to get a satisfied network. That is to say, there is always an image transformation network $\mathcal{N}$ for every style image I_s,

$$I_s \xrightarrow{SGD} \mathcal{N}(\cdot; w_*). \tag{5}$$

The architecture (layer numbers, filter kernel size, *etc.*) of the network $\mathcal{N}$ is pre-defined and is only parameterized by w_* (weight value). To achieve the target of supervised training, we can construct the style image domain $\{I_s\}$ and the network parameter domain $\{w_*\}$. Once the two domains are built explicitly, it is straightforward to train an end-to-end neural networks by minimizing a pre-defined loss function. However, there are two defects in this explicit method. Firstly, it is time consuming to get an optimal w_* for every style image I_s. Second, even though the domain $\{I_s\}$ and the domain $\{w_*\}$ have been built, it is difficult to measure the distance between two networks which is necessary in defining the loss function.

To avoid the above defects, we propose an implicit way to get the transformation network $\mathcal{N}(\cdot; w_*)$ by a meta network:

$$Meta\mathcal{N} : I_s \rightarrow \mathcal{N}(\cdot; w). \tag{6}$$

Here the meta network is parameterized by θ. In the training stage, the parameter θ is optimized by minimizing the empirical loss across a dataset of content images $\mathcal{D}_c$ and a dataset of style images $\mathcal{D}_s$ by

$$\min_\theta \sum_{I_c \in \mathcal{D}_c} \sum_{I_s \in \mathcal{D}_s} \mathbf{PLoss}(I_x|I_c, I_s), \tag{7}$$

where $I_x = \mathcal{N}(I_c; w_\theta)$ and $w_\theta = Meta\mathcal{N}(I_s; \theta)$. In the training stage, the style image I_s is used as one of the su-

Algorithm 1 Minibatch SGD training of meta networks for neural style transfer. We use $k = 20$, $m = 8$, $\lambda_c = 1$ and $\lambda_s = 250$ 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 $\mathcal{N}(\cdot; w)$ to get transferred images $\{I^{[1]}, ..., I^{[m]}\}$.
        • Compute the content loss and style loss and update $\theta$
              $\nabla_\theta \sum_{i=1}^{m} \left( \lambda_c ||(\mathbf{CP}(I^{[i]}) - \mathbf{CP}(I_c^{[i]}))||_2^2 + \lambda_s ||(\mathbf{SP}(I^{[i]}) - \mathbf{SP}(I_s))||_2^2 \right)$.
    end for
end for
```

pervised target in the **Ploss** function as well as input image to the meta network $Meta\mathcal{N}$. The content image I_c is also used as the supervised target in the **Ploss** function as well as input initial image to the transformation network $\mathcal{N}$. In the inference stage, the meta network takes in a new style image I_s as input and generates a transformation network $\mathcal{N}$ which is able to transfer any given content image I_c towards the style image I_s.

Both the style image I_s and the content image I_c are taken as inputs in Equation 7. According to our experiments, the training of meta networks will collapse if the pair of (I_s, I_c) changes too frequently. As is shown in [17], for every given style image I_s, there exists an optimal w_*. In the training stage it needs iterations of SGD steps to update the meta network parameter θ in order to produce an appropriate w for the given style image I_s. Algorithm 1 details the training strategy.

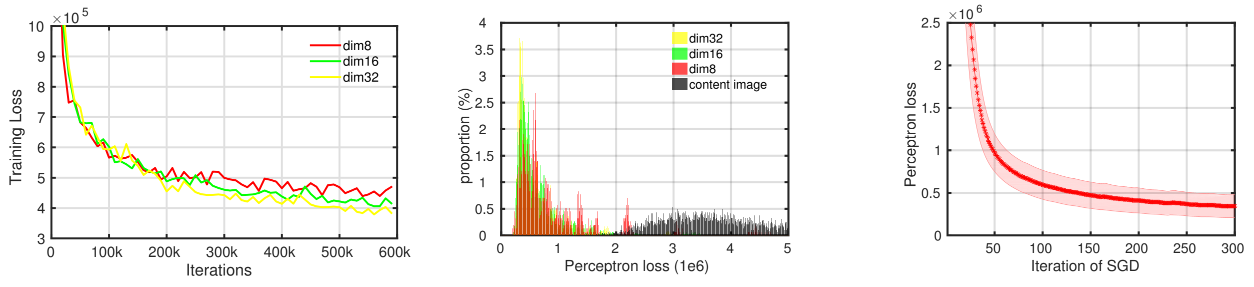
4. Experiment

4.1. Implementation Details

The meta networks for neural style transfer are trained on the content images from MS-COCO [24] *trainval* set and the style images from the *test* set of the WikiArt dataset [27]. There are about $120k$ images in MS-COCO *trainval* set and about $80k$ images in the *train* 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 [18] 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 [17], except that we remove the `Tanh` layer at last and the instance `BN` layer after the first `Conv` layer.

Gatys *et al.* [9] firstly proposed to use the gram matrix of features to represent the style. Recently Li *et al.* [23] found the mean-variance representation of the style is much more compact while having similar performance. Therefore in this paper, we adopt the mean-variance representation of the style. 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 while the style loss is the Euclidean distance between the mean and stand deviations of two feature maps of the style image and the transferred image.

Our code is based on Caffe [16]. Training a meta network takes about 3 days on four modern GPUs. The results are shown in Figure 3. We tested three kinds of image transformation networks. The $dim32$ version is similar to the architecture proposed in [17] while $dim16$ version has half the feature channel numbers and $dim8$ version has a quarter. Figure 3(a) plots the training curves for image transformation networks with different feature channel numbers. As it is shown in Figure 3(b), the trained meta networks perform well on the *test* set of $40k$ style images and reach a much lower perceptron loss comparing to the baseline of the initial content images. In our experiments, the converged losses on the training styles and testing styles are both around 4.0×10^5, which equal approximately 200 steps of gradient descent in [9] (our own implementation) as shown in Figure 3(c). While the converged loss of meta networks is still relatively large, the transferred images from even the



(a) Perceptron loss in the training stage (b) Statistics on $40k$ testing styles (c) Gradient descent optimization for 40 styles.

Figure 3: The meta network is trained over $80k$ style images in the *train* set and the training loss converges to about 4.0×10^5 in figure (a). dim32 denotes the filter numbers is 32 in the first `conv` layer of the image transformation network. The same to $dim16$ and $dim8$. After the training of the meta networks, we evaluate the performance on $40k$ style images in the *test* set. We compute the average perceptron loss of 100 content images for every style image in the *test* set. The changing of the perceptron loss distribution across $40k$ style images in the *test* set is depicted in figure (b). The perceptron losses of the initial content images are concentrated on 3.5×10^6 and the meta network moves them to around 4.0×10^5. In figure (c) we compare the gradient descent method and compute the perceptron loss over 40 styles. The perceptron loss of the images from meta networks equals about 200 gradient descent steps.

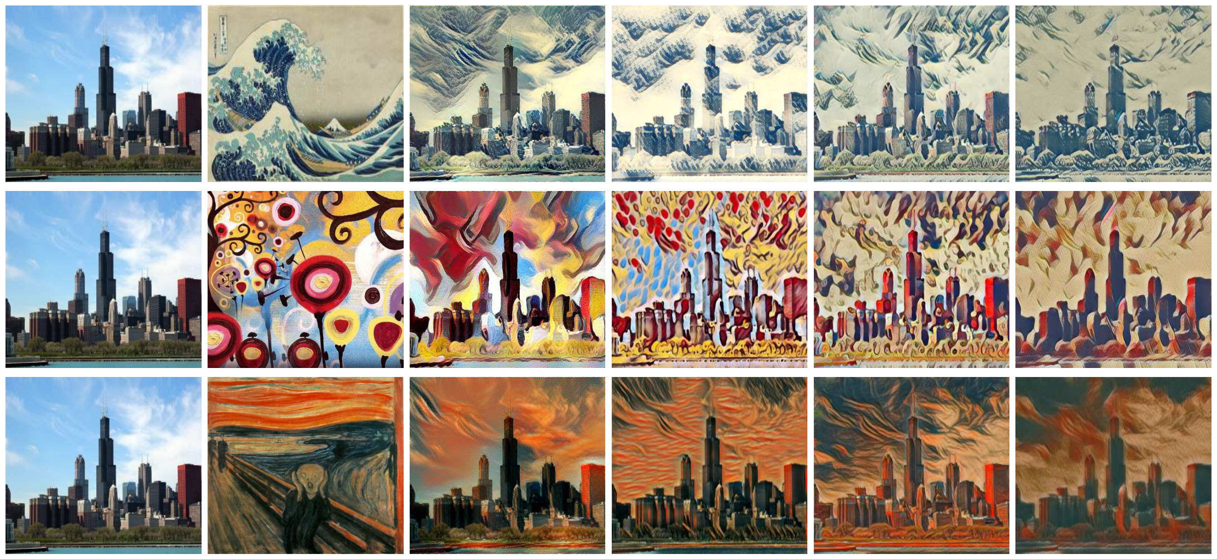


Figure 4: Qualitative Comparisons of different methods. Both the style images and the content images are unseen in the *train* set for *our* model.

Table 1: Comparison of different models. Our image transformation networks share a similar structure with [17] but our meta networks are much faster to encode the new style. The model in Huang and Belongie [14] can also be adapted to the new style efficiently at the price of relying on a VGG-16 network in the transferring stage. As comparison, our meta network only needs a 449 KB model in the transferring stage, which is capable of executing on a mobile device. Both the style image and content image are resized to 256×256. Time is measured on a Pascal Titan X GPU.

Method	*Encode*	*Transfer*	*Model*[†]
Gatys *et al.* [9]	N/A	9.52 s	N/A
Johnson *et al.* [17]	4 h	15 ms	7 MB
Chen and Schmidt [3]	0.4 s	0.17 s	10 MB
Huang and Belongie [14]	27 ms	18 ms	25 MB
Ours	19 ms	15 ms	7 MB
Ours-Fast	11 ms	8 ms	449 KB

$dim8$ generated model still seem visually good as displayed in Figure 5. Both the style images and content images in the test stage are not observed in the $train$ set. We do not use any cherry-picking for style images or content images.

4.2. Comparison with Other Methods

We compare our method with other style transfer methods to evaluate its effectiveness and efficiency. Gatys *et al.* [9] proposed to find the optimal image by gradient descent and Johnson *et al.* [17] proposed to find the near-optimal 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 compare the model size of image transformation networks as it is often used to real-time transfer content images.

We show example style transfer results of different methods in Figure 4. Visually, there is no significantly difference between the three methods. Note that both the style images and the content images are unseen during the training stage of our meta networks while the model in [17] needs to be specifically trained end-to-end for every style. The image transformation network of our model and the model in [17] 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 19 ms to produce an image transformation network which is used in [17] for a single Titan X GPU card. Table 1 lists the advantages and defects of these methods. The gradient descent method of optimization in [9] is flexible but cannot process images in real time. The method of [14] 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.

4.2.1 Difference with SGD solution

Comparing to [17], 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 (the weight values of almost the

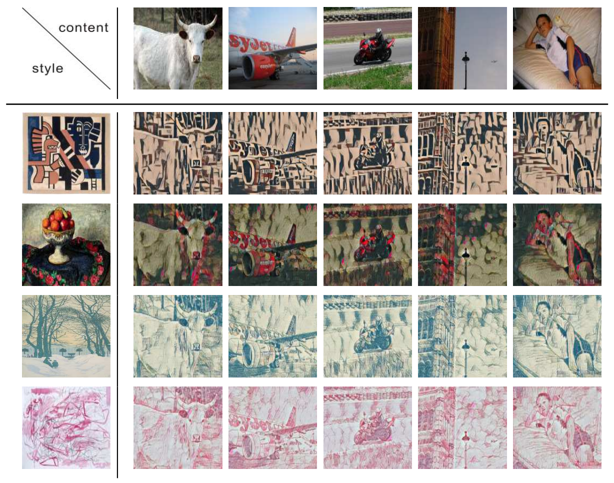


Figure 5: Examples of style transfer by the fast version of our meta networks. Both the style images and content images are *unseen* in the training stage. The size of image transformation network is 449 KB, which is able to real-time execute on a mobile device. Best viewed in color.

Layer	Activation size
Input style image	$3 \times 256 \times 256$
VGG-16 perceptron	1920×1
FC, 1792×1920	1792×1
Group FC, group = 14	$\sim 1.1 \times 10^5$
Input content image	$3 \times 256 \times 256$
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 and its corresponding meta network. 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. The meta network is in the purple region, which takes in a style image and gets the 1920-dim style features from VGG-16 perceptron. The style features go through two fully connected layers to generate the parameters for the filters of the `conv` layers in the pink region of the image transformation network. The filters of the `conv` layers in the gray region are jointly trained with meta networks and are fixed in the inference stage. There is no parameters in other layers of the image transformation network.

whole image transformation networks). Therefore our image transformation networks can adopt any other useful structure, which is the same with SGD. To show the adaptive ability of the meta network for different styles, we have experimented with much smaller image transformation networks (449 K-B) and the network architecture is characterized in Table 2. This small and fast image transformation network also gives satisfying results as shown in Figure 5.

4.2.2 Difference with AdaIN solution

Recently Huang and Belongie [14] 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 and decoding the features also requires a VGG-like architecture. This makes their image transformation network relatively large (~25 MB). The difference between this work and our method lies in the freedom of image transformation network for different style images. Huang and Belongie [14] align the mean and variance of the content feature maps to those of style feature map, 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 training stage is used and only one layer is changed for different styles. The high-level encoded features from VGG are important in [14] and they rely on VGG for both style image and image transformation networks. In the work by [14], both the encoder (fixed to VGG) and decoder (by training) of the image transformation network is fixed except the `AdaIN` layer to adapt to a new style image. Although the image transformation network is large (~25 MB), it has little freedom for a new style image, while our method generates the weight values of the whole image transformation network to adapt to a new style image.

4.3. 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

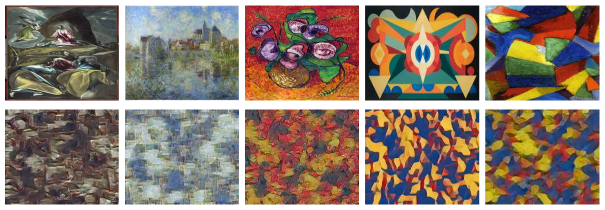


Figure 6: Texture Visualization. The first row displays style images and the second row displays the corresponding textures encoded in the image transformation networks.

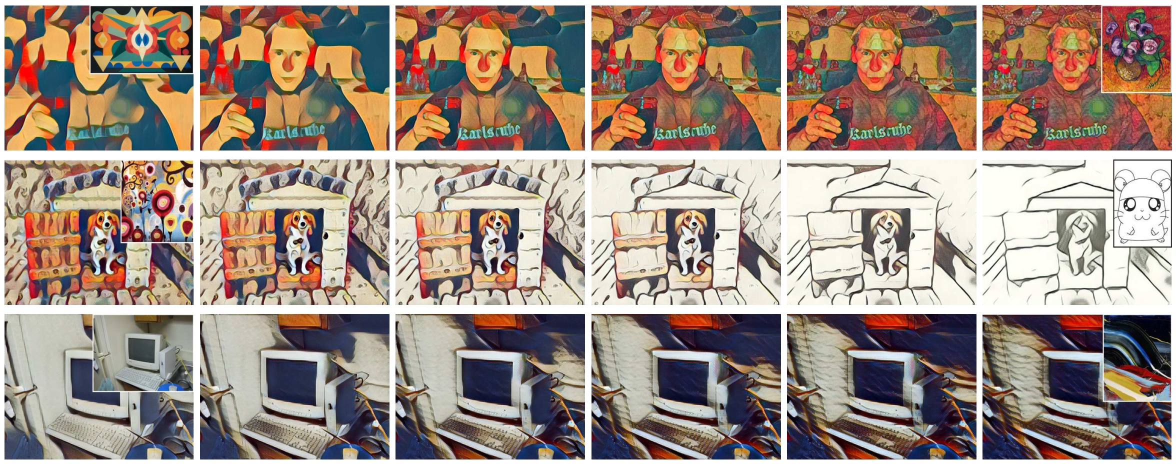


Figure 7: Interpolations of different styles. The fist two rows show the combination of two styles in the hidden state. In the third row, by feeding the content image to the meta network we can get an identity transformation network, which enables the control of the strength of the other style.

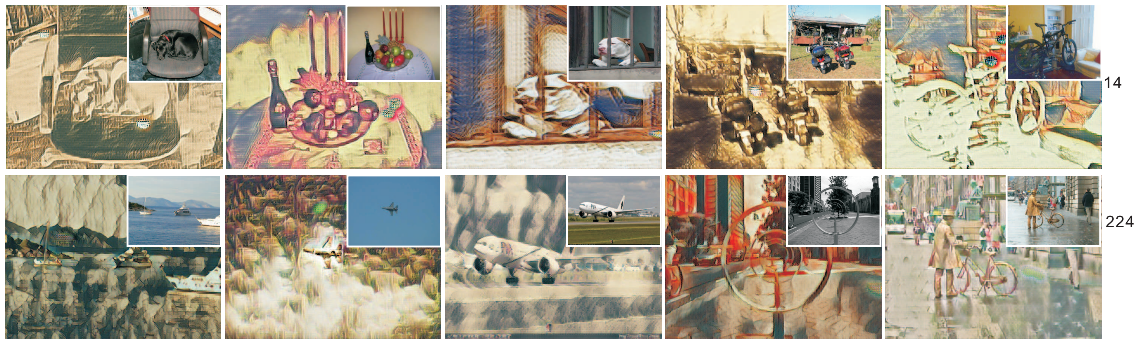


Figure 8: Randomly generated styles at varying hidden state $h \in \{14, 224\}$.

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 6 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 7 displays interpolations of hidden states between two real styles. These style images do not come from 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 8 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.

5. Conclusion

In this paper we introduce the meta networks, a novel method to generate the near-optimal network instead of stochastic gradient descent for neural style transfer. The meta network takes in a new style image and produces the 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.

ACKNOWLEDGEMENT This work is supported by the National Key Research and Development Program of China (2017YFB1002601) and National Natural Science Foundation of China (61375022,61403005,61632003).

References

[1] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua. Coherent online video style transfer. *arXiv preprint arXiv:1703.09211*, 2017.

[2] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stylebank: An explicit representation for neural image style transfer. *arXiv preprint arXiv:1703.09210*, 2017.

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

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

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

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

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

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

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

[10] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *arXiv preprint arXiv:1705.06830*, 2017.

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

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

[13] H. Huang, H. Wang, W. Luo, L. Ma, W. Jiang, X. Zhu, Z. Li, and W. Liu. Real-time neural style transfer for videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 783–791, 2017.

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

[15] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, pages 667–675, 2016.

[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[17] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

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

[19] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. 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, 2013.

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

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

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

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

[24] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

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

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

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

[28] H. Noh, P. Hongsuck Seo, and B. Han. Image question answering using convolutional neural network with dynamic parameter prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 30–38, 2016.

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

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

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

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

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

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

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