# Project 2: Image Style Transfer

Junhao Yu 2020012847
Yunyu Liang 2020012913
Jiawei He 2021012289

## Abstract

*Neural style transfer attempts to retain styles from style images in content images. Although current models have achieved state-of-art style transfer performance in most cases, they could perform rather badly when there are conspicuous distinctions between content images and style images since the difference could weaken the ability of existing models to couple the content features and style features. In this paper, we propose content-unbiased RBMAD based on neural flows to better adapt style features according to target content images. The neural-flow-based RBMAD model consists of reversible neural flows, inverted residual bottleneck, the mixture of style features and content features and the AdaIN module. The neural flow supports both forward and backward inference and operates in a projection-transfer-reversion scheme. The forward inference projects the content features into latent representations while the backward inference combines the style features and content features through RBMAD module and reconstructs the stylized images. Extensive experiments demonstrate the superiority of RBMAD when transferring style when achieving both content and style features preserving.*

## 1. Introduction

Neural style transfer(NST) is a long-standing topic in computer vision, which aims at transferring the artistic style form a style image to a content image. Since the first NST was proposed by Gatys *et al.* [4], numerous NSTs have been invented following his work. [8, 11, 12, 17, 21] improve style transfer either from loss function or image transform net. Based on the Instance Normalization in [21], Huang *et al.* [7] proposed AdaIN module, which subversively alters the way people utilizes style features. Inspired by Huang *et al.*, [14, 20, 23] have been introduced. To further address the content leak problem in these works, [1, 2, 6, 9] have been proposed to restore the content features more effectively while achieving better generalization. Although these methods have achieved impressive visual results in most applications, they can be limited when the style image and content image have great discrepancies, leading to different kinds of content or style distortion. Inspired by [18] and [3], we address a novel content-unbiased *Residual Bottleneck Mixed with AdaIN*(**RBMAD**) module and an unbiased neural style transfer framework based on neural flow to solve this problem.

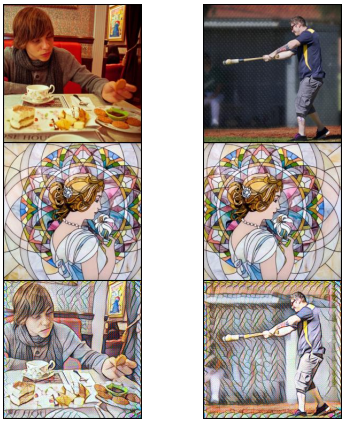

Figure 1. Visualization when there are large discrepencies between content images and style images. Existing models often fail to transfer style when the content is too complex(like the people in both images), leading to the distortion of either content or style.

**RBMAD** module consists of three parts: inverted residual bottleneck, mixture of style and content image features and AdaIN module. This is because some features of content image could greatly impair the ability of AdaIN to couple style and content features in some areas of stylized image, which further leads to the loss of content features or style features. To tackle this problem, **RBMAD** first uses inverted residual bottleneck to extract the most significant traits of style and content features, which will be later concatenated to form *Style and Content Mixture(SCM)*. By calculating the mean and standard of SCM, we obtain content-image-targeted style features. Finally, like AdaIN, combining these style features and normalized content features, we obtained the output content-image-fixed stylized image.

Our **RBMAD** is plugged into our neural-flow-based style transfer network, which consists of a chain of fully revertible transformation modules, including squeeze module, Activation-Normalization layer, $1\times1$ Convolution layer and Affine Coupling layer. The network first projects the

content features into latent representations via forward inference. Then via reversed feature inference, the network combines the style features and content features through **RBMAD** module and reconstructs the stylized images. Since all the modules in this network are unbiased, it makes unbiased content-preserving style transfer. Inspired by [8], we use pretrained VGG16 to extract style features from style images.

To display the superiority of our RBMAD and network, we test several models on the same dataset and compare the results both quantitatively and qualitatively. The experiment results prove that RBMAD indeed outperforms other models.

The contributions of this work are three-fold:

- We reveal the weakened ability of coupling style and content when the style image and content image have great discrepancies using model proposed by [4](which we implement as baseline model), ArtFlow [1] and StyleFlow [3].
- We propose an unbiased, content-image-targeted module RBMAD based on neural flows, which achieves unbiased content-preserving style transfer.
- Based on neural flows and RBMAD, we propose an unbiased, reversible and content-image-targeted style transfer network, which addressed the problem mentioned above and achieves competent style transfer results.

## 2. Related Work

Style Transfer is a long-researching project. Before the appearance of neural network, [10, 16, 19] proposed *image-based artistic rendering* (IB-AR) to achieve this task. Some ideas of these works have been greatly exploited by neural transfer network(NST). After the occurrence of *neural network*, great progress has been made in this field.

We divide the related work in NST into two categories: **Image-Transformation-Net Based Transfer** and **Neural-FLow Based Transfer**.

**Image-Transformation-Net Based Transfer**. Gatys *et al.* [4] proposed a general architecture and a Gram loss upon features to represent image styles, which is the foundation of this field. Johnson *et al.* [8] introduced perceptual losses for training feed-forward network, which accelerates the training process. The case of optimizing only for the loss function came to an end when Huang *et al.* [7] proposed AdaIN module, which greatly changes the way how to use features to achieve image transformation directly. Inspired by Huang *et al.*, numerous methods, such as [14, 20, 23], have been proposed. In this paper, first model utilizes the general architecture posed by Gatys *et al.* [4] with the bottleneck to extract significant features, and second model borrows ideas from AdaIN, further forming brand-new module RBMAD.

**Neural-Flow Based Transfer**. Neural flows refer to a subclass of deep generative models, which uses a series of reversible transformations to achieve high likelihood of observations. Dinh *et al.* [2], the pioneer of this field, introduced NICE, which takes advantage of affine coupling layers to model high-dimensional densities. Based on this work, a series of works, such as ArtFlow [1], Flow++ [6] and Glow [9], have been done to further explore the potential of neural flow with more flexible and powerful modules. In this paper, we borrow the idea from ArtFlow and Glow to form an unbiased reversible neural network. Specifically, ArtFlow shows us how to tackle the content leak problem, which is common in WCT variants [13, 15, 22], AdaIN variants [7, 23] and Avatar-Net [20]. Glow gives us inspiration to devise more efficient neural flows.

## 3. Approach

### 3.1. Baseline

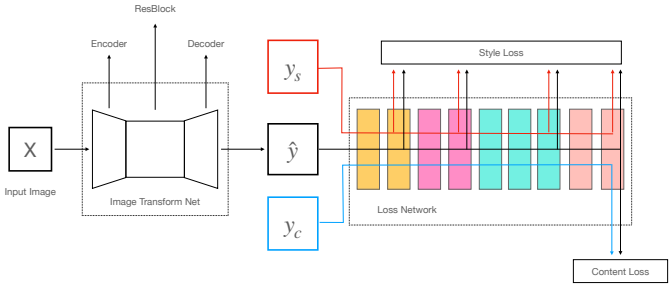


Figure 2. The framework of the baseline model. For image transform net, it consists of encoder, residual block and decoder. For loss network, we use pretrained VGG16 to extract content and style features.

Similar to [8], we use a deep residual convolutional neural network as the image transformation network. The network consists of several convolutional layers and five residual blocks. The structure of the residual blocks comes from [5], but we use instance normalization [21] instead. The implementation of this model is to reveal the weakened ability of coupling style features and content features when there is conspicuous difference between content image and style image and the superiority of RBMAD.

**Image Transform Net**. The architecture of Image Transform Net is shown in Figure 3. Encoder acts as a destylization module, which will strip original style features from the content images while convolving as shown in Figure 5. Decoder acts as a stylization module, which will combine style features with content features. The process of stylization is shown in Figure 6.

In order to better demonstrate the limits of baseline

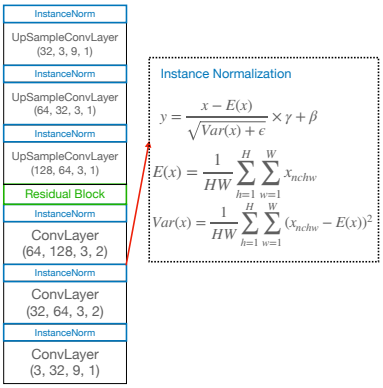


Figure 3. The CNN-based Image Transform Net of Baseline Model

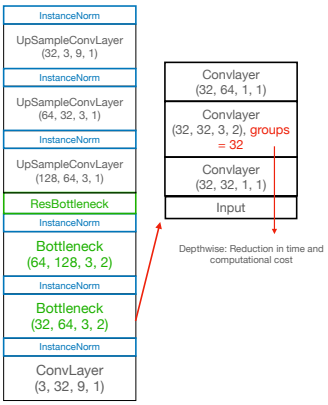


Figure 4. The ResBottleneck-based Image Transform Net of Baseline Model

model, we further replace the CNN-based Image Transform Net with ResBottleneck-based Image Transform Net [18], which makes great progress in efficiency and effectiveness. But in Experiment4, we will show that its loss of content features and style features is still considerable compared with neural-flow-based RBMAD. Also, exactly when utilizing ResBottleneck to achieve better performance did we realize the superiority of ResBottleneck, which further lead us to devise RBMAD module. The architecture of ResBottleneck-based Image Transform Net is shown in Figure 4.

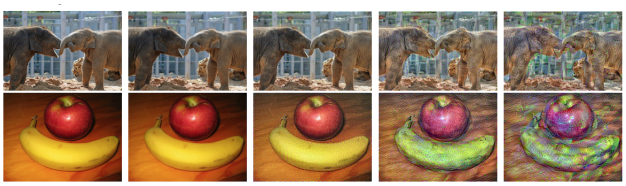

Figure 5. The loss of original style of content image while convolving [8](From left to right, the CNN network goes deeper, just like Encoder). The deepest layer preserves most content features and little style features.

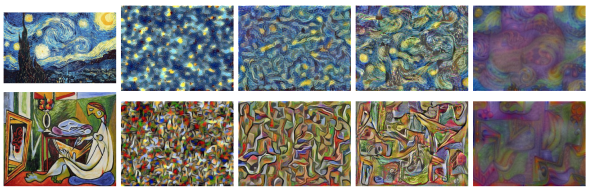

Figure 6. The construction of style image [8](From right to left, the CNN network goes shallower, just like Decoder). The shallowest layer preserves most style features and little content features.

**Loss Network**. As shown in Figure 5 and Figure 6, we can see that the shallower layers preserve more style information(texture, edge, color and etc.) while the deeper layers preserve more content information(the elephant, apple and banana without little original style features such as color). Hence, we can treat the shallower layers as style features and deeper layers as content features and compare the stylized image with content image(using deeper layers) and style image(using shallower layers) respectively. According to [4] and [8], we use the loss functions shown as below:

$$\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2}\sum_{i,j}\left(F_{ij}^{l} - P_{ij}^{l}\right)^2. \tag{1}$$

where $\vec{p}$ and $\vec{x}$ are the content image and the stylized image and $P^l$ and $F^l$ are their respective feature representation in layer $l$. Then, we can calculate the Gram matrix $G^l \in \mathcal{R}^{N_l \times N_l}$ where $G_{ij}^l = \sum_k F_{ik}^l F_{kj}^l$. Then we can similarly define style loss function:

$$E_l = \frac{1}{4N_l^2 M_l^2}\sum_{i,j}(G_{ij}^l - A_{ij}^l)^2,\ \mathcal{L}_{style}(\vec{a}, \vec{p}) = \sum_{l=0}^{L} w_l E_l \tag{2}$$

where $\vec{a}$ and $\vec{x}$ are the style image and the stylized image and $A^l$ and $G^l$ are their respective feature representation in layer $l$. $w_l$ are weighting factors.

Therefore, the total loss function we minimize in this baseline model is:

$$\mathcal{L}_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}) + \beta\mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) \tag{3}$$

where $\alpha/\beta = 1 \times 10^{-5}$ in our experiment. We also slightly altered this loss function according to different models but their basic ideas are all the same. Hence, we do not list all loss functions we use for simplicity.

### 3.2. Neural-Flow

We use Artflow and Styleflow as the flow network.

Artflow use a reversible neural flow model *Glow* to extract the features of the image and restore image from stylized features. The network has several blocks and each blocks contains an activation normalization, a reversible 1x1 convolution and an additive coupling.

**Additive coupling layer**. Dinh *et al.* [2] invented NICE with a reversible transformation named affine coupling layer. In order to better fit our task, We make a small modification to the additive coupling in [1], which is a special case of affine coupling layer:

$$x_a, x_b = \text{split}(x), y = \text{concat}(x_a, \text{NN}(x_a) + x_b) \quad (4)$$

$$x_b, x_a = \text{split}(x), y = \text{concat}(x_a, \text{NN}(x_a) + x_b) \quad (5)$$

where the split() function splits a tensor into two halves along the channel dimension. NN() is a neural network with same dimension input and output. The concat() function is the reverse function of split() function, which concatenates two tensors along the channel dimension.

We use the two slightly different modules alternatively in one block of *Glow*.

**Invertible** $1 \times 1$ **Convolution**. Again, we borrow ideas from Glow [9] to utilize a learnable invertible $1 \times 1$ convolution layer for more flexible channel permutation, which copes with the problem that only half of the feature maps are processed by additive coupling layer. The permutation function is $y_{i,j} = W x_{i,j}$, where $W \in \mathcal{R}^{c \times c}$ and $c$ is the channel dimension of tensor $x$ and $y$.

**Actnorm**. We also borrow ideas of activation normalization layer(Actnorm) to replace instance normalization. The function of Actnorm is $y_{i,j} = w \odot x_{i,j} + b$, where $w$ and $b$ are just like instance normalization, which are learnable scale and bias parameters of Actnorm.

After extracting the content image feature $f_s$ and the style image feature $f_s$, we need a feature transfer module to generate a stylized feature $f_{cs}$ such that it contains the content feature in $f_c$ and the style feature in $f_s$.

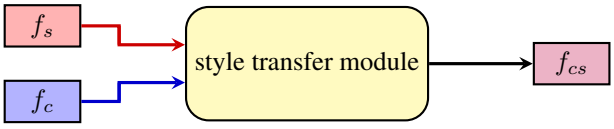


Figure 7. feature transfer module

Here we use AdaIN from [7] and our RBMAD as the feature transfer module.

### 3.3. RBMAD

In this work, we present a novel neural-flow-based content-unbiased module *RBMAD*, which is shown in Figure 8. Different from primitive AdaIN module, which uses the mean and standard deviation of style image only, *RBMAD* first utilizes ResBottleneck to select the most significant traits of style features and content features. Then, RBMAD will use Invertible $1 \times 1$ convolution to rescale the style feature tensor and content feature tensor. The next step is based on this intuition: Even for the same style image, different content images should require different style means and standards to realize AdaIN transformation, since some features of content image could greatly impair the ability of AdaIN to couple style and content features in some areas of stylized image, which further leads to the loss of content features or style features. Hence, we concatenate the style features and content features to form Style and Content Mixture(SCM) so that the mean and standard deviation of SCM have both content features and style features. Then combining such mean and standard deviation with normalized content features to achieve AdaIN transformation, which gives us the output unbiased content-image-targeted stylized image.

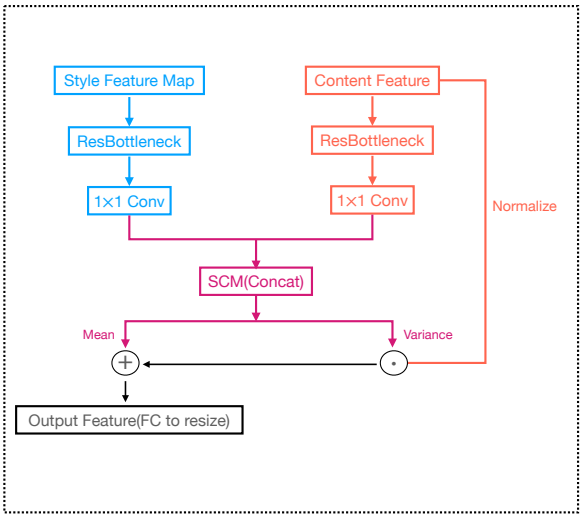


Figure 8. the structure of RBMAD

**ResBottleneck** is adopted from [18], it consists of three convolutional layers and three instance normalization layers. The first convolutional layer restores the features to high dimension. The second one uses depthwise convolution to filters the data and selects the important, reducing computational cost as well. The last one compresses the data, making its shape the same as input for residual operation. This bottleneck structure is used to capture important features in the input feature map.

**Invertible** $1 \times 1$ **Convolution**. The exact function and detailed analysis of this module has been presented in Neural-Flow. In RBMAD module, Invertible $1 \times 1$ Convolution is used to halve the channels of the output, then we can concatenate them and get SCM.

**AdaIN** is used to get the output or RBMAD. We use SCM as the style instead of the style feature when calling AdaIN, because (based on intuition) for the same style image, different source image should use different parameter. The function of AdaIN is stated below:

$$\text{AdaIN}(x, y) = \sigma(y) \cdot \frac{x - \mu(x)}{\sigma(x)} + \mu(y) \quad (6)$$

**Property of RBMAD: Content Unbiased**. Let's as-

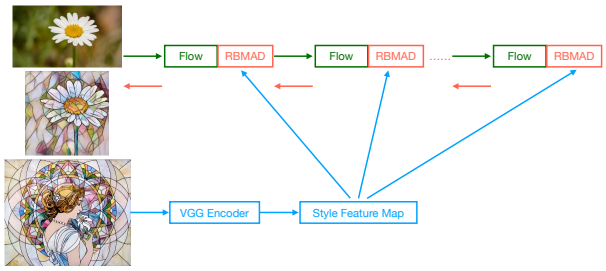


Figure 9. Neural-Flow-Based RBMAD Architecture

sume that the input unnormalized content feature is $f_x$ while the normalized content feature is $f_c$(According to [7], normalizing content feature $f_x$ is destylization, which gives us pure and without-styled content feature $f_c$). According to the AdaIN function presented above, we can get the expression of $f_{out}$:

$$f_{out} = \frac{f_x - \mu(f_x)}{\sigma(f_x)} \cdot \sigma(SCM) + \mu(SCM) \tag{7}$$

We can reform the equation 7 and get the following equation:

$$\frac{f_{out} - \mu(SCM)}{\sigma(SCM)} = \frac{f_x - \mu(f_x)}{\sigma(f_x)} \tag{8}$$

According to the definition of content features given by [7], we have the following function:

$$\hat{f}_c = \frac{f_{out} - \mu(f_{out})}{\sigma(f_{out})} \tag{9}$$

We further take the fact that $f_{out}$ is obtained by normalized content feature multiplying $\sigma(SCM)$ then adding $\mu(SCM)$ into account. Therefore, $\mu(f_{out}) = \mu(SCM)$ and $\sigma(f_{out}) = \sigma(SCM)$ should hold. Finally, combined with equation 8, we can get the following equation, which proves that RBMAD is content unbiased.

$$\begin{aligned} \hat{f}_c &= \frac{f_{out} - \mu(f_{out})}{\sigma(f_{out})} \\ &= \frac{f_{out} - \mu(SCM)}{\sigma(SCM)} \\ &= \frac{f_x - \mu(f_x)}{\sigma(f_x)} \\ &= f_c \end{aligned} \tag{10}$$

### 3.4. Neural-Flow-Based RBMAD

The architecture of Neural-Flow-Based RBMAD is shown in Figure 9.

This framework first utilizes VGG16 to extract style features and then put different layers of style features into different layers of RBMAD. The green arrows mean forward inference while the orange arrows mean backward inference. Green blocks(i.e. Flow module) receives both forward and backward inference while orange blocks only appear in backward inference.

## 4. Experiments

### 4.1. Overview

For content images, we use COCO's test2017 and val2017 datasets as our training dataset. For style images, we choose several images from the wikiart dataset. We run models on the same style and content images, then compare their effectiveness both quantitatively(loss curves and loss table1) and qualitatively(Figure 10 and 17).

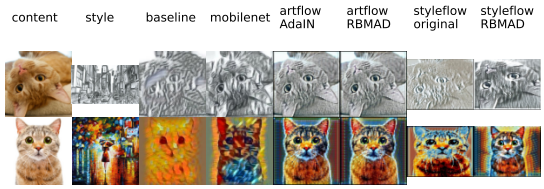


Figure 10. Qualitative Comparison(the clearer original image is in the folder)

### 4.2. Baseline

Settings: 1 epoch, batch size=4, lr=1e-3

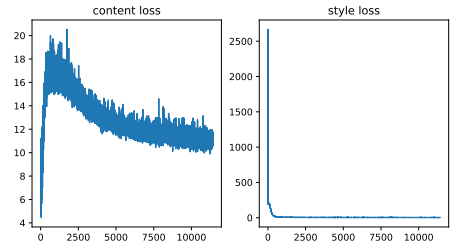


Figure 11. baseline training loss

We can see that content loss is large and many details in the content image are lost from comparison of Figure 10. Since the deep features extracted by Vgg is only used in the loss function but not in the image transformation network(which is not capable of the transfer procedure), the poor result is reasonable.

### 4.3. Mobilenet

Settings: 1 epoch, batch size=4, lr=1e-3

The content loss(Figure 12) is significantly smaller, but the style is not good enough. Although this model is almost the best among all the image transformation network, the texture of the output image is still not clear.

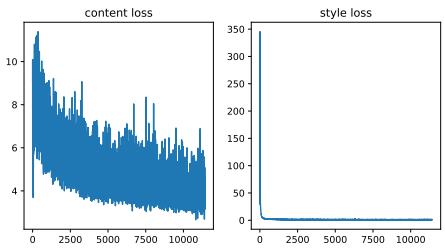


Figure 12. mobilenet training loss

### 4.4. Artflow+AdaIN

We use the official pytorch implementation of [1] as the code framework.

Settings: 2 blocks per flow, 8 flows, batch size=1, lr=1e-4, inverse linear lr decay(5e-5)

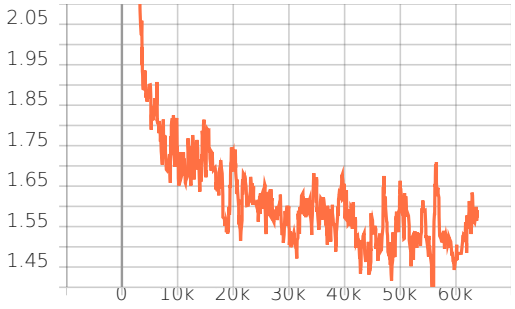

Figure 13. artflow+AdaIN training loss

Some details of the content image are kept from comparison of Figure 10.

### 4.5. Artflow+RBMAD

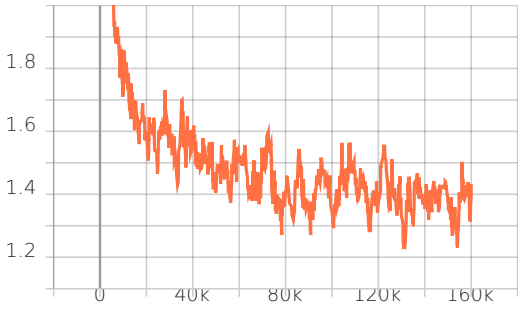

Figure 14. artflow+RBMAD training loss

The details of the content image are well kept, but the style is not significant. Compared with AdaIN, RBMAD makes the loss smaller.

When using Artflow, it's hard to say which style transfer module(AdaIN/RBMAD) is better because this flow network tends to preserve all the content but lacks details.

### 4.6. Styleflow(original)

We use the official pytorch implementation of [3] as the code framework.

Settings: 2 blocks per flow, 15 flows, lr=5e-5, halves every 40k iterations.

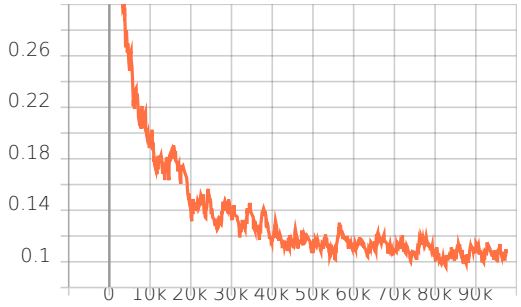

Figure 15. styleflow training loss

Some details of the content image vanish from comparison of Figure 10.

### 4.7. Styleflow+RBMAD

Since the size of the style feature extracted by Styleflow is very small, we only apply ResBottleneck on the content feature.

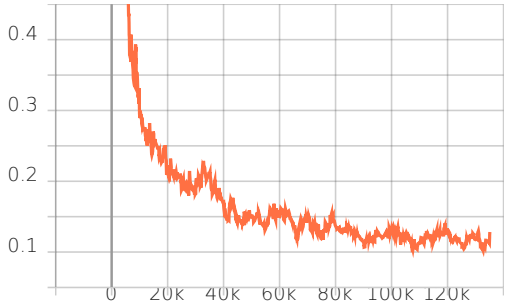

Figure 16. styleflow+RBMAD training loss

Figure 17. styleflow+RBMAD result

Notice that there's a lot of details in the output image, the texture of the content image is preserved nearly perfectly from Figure 10.

### 4.8. Loss

The loss function used by different models are different, we choose the common parts and make appropriate scaling so that they can be compared.

| Method | Loss ↓ |
|---|---|
| Baseline | 11.1 |
| Mobilenet | 4.69 |
| Artflow+AdaIN | 1.52 |
| Artflow+RBMAD | 1.38 |
| Styleflow | **1.12** |
| Styleflow+RBMAD | 1.29 |

Table 1. Loss of different models

When using Artflow as the flow network, RBMAD decreases the loss. Although it doesn't make progress when it comes to Styleflow, it did make the output image better.

## 5. Conclusion

In this paper, we propose a new style transfer module **RBMAD**, by combining RMBAD with Styleflow, we can preserve the content feature(especially the details) as well as merging the style. This module successfully addresses the style-content-uncoupled problem when there are large discrepencies between content images and style images. Based on neural flows and RBMAD, the model we propose solved the problem mentioned above and achieves competent style transfer results. In the future, we can explore more architectures of RBMAD(for instance, more important-trait-extraction method other than ResBottleneck) and apply it to more kinds of style-transfer models rather than only neural flows. Also, we will try to find out more competent and effective feature-extraction module other than VGG16 to achieve better results.

## References


[1] Jie An, Siyu Huang, Yibing Song, Dejing Dou, Wei Liu, and Jiebo Luo. Artflow: Unbiased image style transfer via reversible neural flows, 2021. 1, 2, 4, 6

[2] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation, 2014. 1, 2, 4

[3] Weichen Fan, Jinghuan Chen, Jiabin Ma, Jun Hou, and Shuai Yi. Styleflow for content-fixed image to image translation, 2022. 1, 2, 6

[4] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015. 1, 2, 3

[5] Wilber M. Gross, S. Training and investigating residual nets., 2016. 2

[6] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design, 2019. 1, 2

[7] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017. 1, 2, 4, 5

[8] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016. 1, 2, 3

[9] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions, 2018. 1, 2, 4

[10] Jan Eric Kyprianidis, John Collomosse, Tinghuai Wang, and Tobias 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, 2012. 2

[11] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis, 2016. 1

[12] Shaohua Li, Xinxing Xu, Liqiang Nie, and Tat-Seng Chua. Laplacian-steered neural style transfer. In *Proceedings of the 25th ACM international conference on Multimedia*. ACM, oct 2017. 1

[13] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast arbitrary style transfer, 2018. 2

[14] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms, 2017. 1, 2

[15] Ming Lu, Hao Zhao, Anbang Yao, Yurong Chen, Feng Xu, and Li Zhang. A closed-form solution to universal style transfer, 2019. 2

[16] John Collomosse Paul Rosin. Image and video-based artistic stylisation, 2012. 2

[17] Eric Risser, Pierre Wilmot, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses, 2017. 1

[18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018. 1, 3, 4

[19] Amir Semmo, Tobias Isenberg, and Jürgen Döllner. Neural style transfer: A paradigm shift for image-based artistic rendering? In *Proceedings of the symposium on non-photorealistic animation and rendering*, pages 1–13, 2017. 2

[20] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration, 2018. 1, 2

[21] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis, 2017. 1, 2

[22] Huan Wang, Yijun Li, Yuehai Wang, Haoji Hu, and Ming-Hsuan Yang. Collaborative distillation for ultra-resolution universal style transfer, 2020. 2

[23] Zhizhong Wang, Lei Zhao, Haibo Chen, Lihong Qiu, Qihang Mo, Sihuan Lin, Wei Xing, and Dongming Lu. Diversified arbitrary style transfer via deep feature perturbation, 2019. 1, 2