Style Transfer with Facial Preservation

Zachary Ferguson
New York University
zfergus@nyu.edu

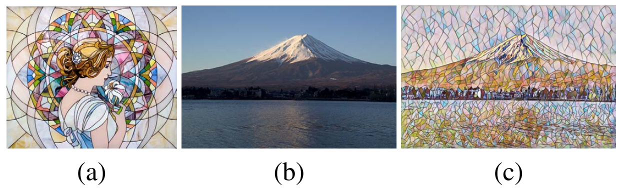

Figure 1: In style transfer, the style of image (a) is applied to the content of image (b) to produce a new stylized image (c).

1. Intro

Style transfer allows us to automatically transfer the style of one image onto the contents of another image, generating a stylized image. Figure 1 shows an example of style transfer. There are several techniques for performing style transfer including *non-photorealistic rendering* [10], optimization via *Neural Style Transfer* (NST) [8], and learning fast style transformations through deep learning [13].

In this paper, we follow up on the work of Gatys et al. [8] and Johnson et al. [13] by using and training convolutional neural networks (CNN) to transfer an arbitrary style from one image to another image. We propose the extended problem of stylizing an image while preserving facial features. This problem requires new extensions upon the original model proposed by Johnson et al. to account for facial differences in the stylized image.

2. Related Work

Gatys et al. [8] introduced the idea of Neural Style Transfer as an online optimization problem using the feature layers of a pretrained CNN to perform style transfer. Johnson et al. [13] then showed how to train a feed-forward network to approximate the optimization problem proposed by Gatys et al. By learning a function for style transfer, Johnson et al. were able to perform *fast style transfer* with runtimes capable of real-time video stylization.

Although Neural Style Transfer is a recent development, there has been a large amount of work in the area. Some more recent work have improved upon the work of Johnson et al. by allowing for multiple styles per trained model [3, 6, 14, 23] or an arbitrary style per model [21, 18, 15, 11, 9, 4]. Gatys et al. [7] also proposed a solution to preserving colors during style transfer. Jing et al. provides an in-depth survey and hierarchy of the work done in style transfer [12]. Our work, directly follows that of Johnson et al., but there may be room for improvement by using more recent methods.

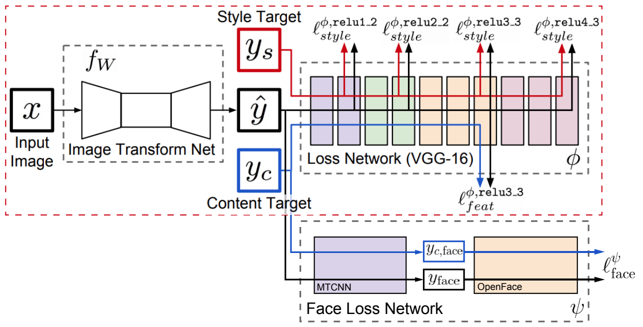
Style Target
$\ell_{style}^{\phi,\mathtt{relu1\_2}}$
$\ell_{style}^{\phi,\mathtt{relu2\_2}}$
$\ell_{style}^{\phi,\mathtt{relu3\_3}}$
$\ell_{style}^{\phi,\mathtt{relu4\_3}}$
y_s
f_W
x
Input Image
Image Transform Net
$\hat{y}$
y_c
Loss Network (VGG-16)
ϕ
Content Target
$\ell_{feat}^{\phi,\mathtt{relu3\_3}}$
$y_{c,\mathrm{face}}$
y_{face}
MTCNN
OpenFace
$\ell_{\mathrm{face}}^{\psi}$
Face Loss Network
ψ

Figure 2: Architecture used to train a style transfer network. The original architecture described by Johnson et al. [13] (circled in red) is extended by adding a face loss network. The face loss network is comprised of a face detection network (MTCNN) and a face recognition network (OpenFace). The loss network and face loss network compute the perceptual loss for training the image transformation network to transfer style from y_s onto content of $y_c = x$.

3. Method

Our style transfer methodology follows that of Johnson et al. [13], training a feed-forward convolutional neural network to learn style transfer using perceptual loss (described in Section 3.1). Figure 2 shows the original model architecture consisting of two networks, an image transformation network and a loss network. The loss network computes the perceptual loss from activation layers, and the image transformation network stylizes the input image. A single image transformation network is trained for each style, a limitation of the methodology of Johnson et al, using the computed loss of the loss network.

We extend upon this network by defining a new loss term, $\ell_{\text{face}}^{\psi}$, that penalizes changes in facial features. The original method of Johnson et al. took no special attention to any particular content of the image. We show that by defining specialized losses we can penalize changes in certain features of the original image.

3.1. Perceptual Loss

Perceptual loss, as opposed to per-pixel loss, uses the perceptual features of a pretrained "loss" network, ϕ, to compute the perceptual difference between two images [13]. In our case, the pretrained loss network is an image classifier trained on a image classification dataset. More specifically, we use the 16-layer VGG network [19] pretrained on ImageNet [5].

Put simply, the various activation layers of the loss network represent different perceptual knowledge of the image. Earlier activation layers quantify style elements of the image such as the color and texture. Later activation layers have a deeper understanding of the content of the image.

3.1.1 Content Loss

Using the loss network, we can compute the loss in content of the stylized image as being the difference in activation layers. For a stylized image y and the original content image y_c the content loss is defined as

$$\ell_{\text{content}}^{\phi,j}(y, y_c) = \frac{1}{C_j H_j W_j} \|\phi_j(y) - \phi_j(y_c)\|_2^2 \tag{1}$$

where $\phi_j(x)$ is the j^{th} activation layer (of shape $C_j \times H_j \times W_j$) of the network ϕ for input x.

3.1.2 Style Loss

Using the loss network, we can compute the loss in style of the stylized image as being the difference in activation layers. Borrowing the style loss from Gatys et al. [8], we first need to define the *Gram matrix* $G_j^{\phi}(x)$ as

$$G_j^{\phi}(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'} \tag{2}$$

For a stylized image y and the style image y_s the style loss is then defined as

$$\ell_{\text{style}}^{\phi,j}(y, y_s) = \|G_j^{\phi}(y) - G_j^{\phi}(y_s)\|_F^2 \tag{3}$$

For multiple activation layers, J, rather than a single layer, j, we take the sum of the individual styles losses, $\ell_{\text{style}}^{\phi,J}(y, y_s) = \sum_{j \in J} \ell_{\text{style}}^{\phi,j}(y, y_s)$.

3.1.3 Total Variation Regularization

Additionally, Johnson et al. [13] suggests using total variation regularization, $\ell_{\text{TV}}(y)$ to encourage spacial smoothness.

3.2. Style Transfer

Using perceptual loss, we can defined a loss function, $\ell(y, y_c, y_s)$, for our image transformation network to train under.

$$\begin{aligned} \ell(y, y_c, y_s) = & \lambda_c \ell_{\text{content}}^{\phi,j}(y, y_c) \\ & + \lambda_s \ell_{\text{style}}^{\phi,J}(y, y_s) \\ & + \lambda_{\text{TV}} \ell_{\text{TV}}(y) \end{aligned} \tag{4}$$

λ_c, λ_s, and λ_{TV} are scalar weights used to weight the importance of each component as well as normalize each loss relative to each other.

3.3. Facial Preservation

The base perceptual loss only preserves content on a global level, according to the features of the higher activation layers of the loss network. We propose adding an additional loss term to account for the difference in facial features between the original content image and the stylized image. By including this loss term we can specifically penalize any "drastic" changes to the facial features. Figure 2 show the new face loss network used in combination with the original network of Johnson et al.

3.3.1 Face Loss

To quantify the difference in facial features between the original, $y_c = x$, and stylized image, y, we use face recognition. We first find the faces in y_c using Multi-task Cascaded Convolutional Networks (MTCNN) by Zhang et al. [24][1]. MTCCN produces a bounding box for the faces. Using these bounding boxes, patches are extracted and resized to be 96×96 ($y_{c,\text{face}}$ for the patch found in y_c and y_{face} for the corresponding patch in y). We then use these resized patches and OpenFace [1] to compute a face descriptor embeded on a 128-dimension hyper-sphere.[2] The distance between two face descriptors is a measure of similarity (a unique property of OpenFace). Finally, we define the face loss as

$$\ell_{\text{face}}^{\psi}(y, y_c) = \sum_{\text{face} \in y_c} \|\psi(y_{\text{face}}) - \psi(y_{c,\text{face}})\|_2^2 \tag{5}$$

[1]A pretrained PyTorch version off MTCCN by Dan Antoshchenko (https://github.com/TropComplique/mtcnn-pytorch) is used with minor modifications.

[2]A pretrained PyTorch version of OpenFace by GitHub user TwoBranchDracaena (https://github.com/TwoBranchDracaena/OpenFace-PyTorch) is used.

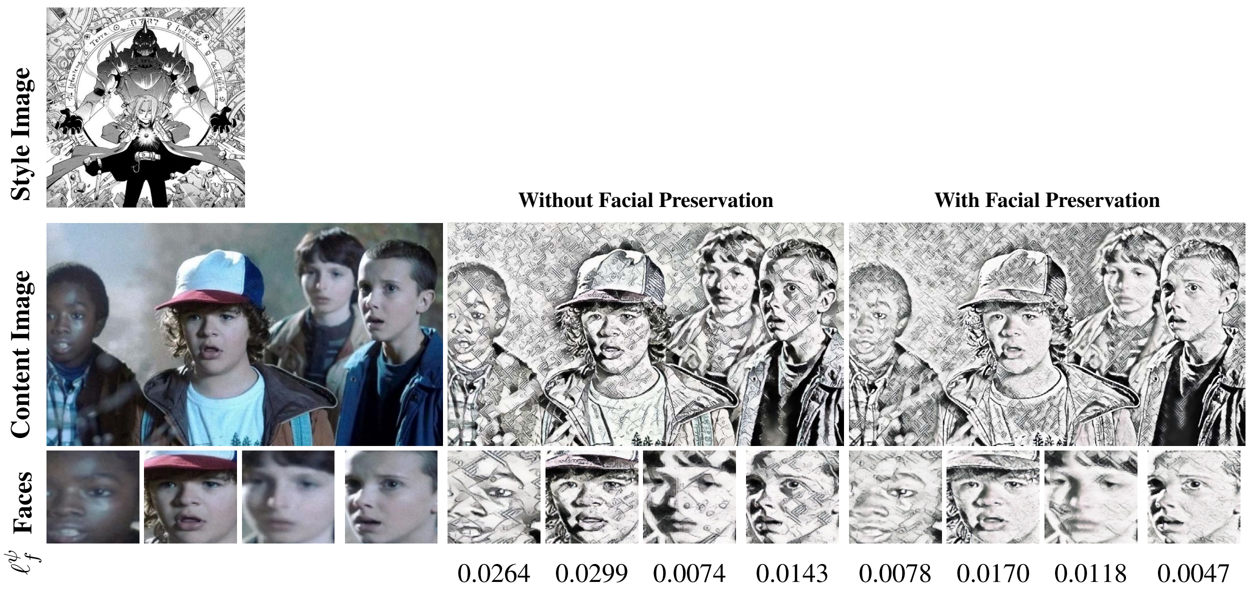

Figure 3: We train two models with the manga style image, one without and one with facial preservation. Shown here are the results of stylizing the content image using the two models. The Faces row shows the face patches found using MTCNN. The last row shows the face loss between the stylized faces and the original faces. In the majority of cases facial preservation decreases the face loss, but the third face's loss increased in this instance.

where ψ is our face recognition network and $\psi(x)$ is the face descriptor.

Using the face loss we define a new total loss, ℓ', to optimize.

$$\ell'(y, y_c, y_s) = \ell(y, y_c, y_s) + \lambda_f \ell^{\psi}_{\text{face}}(y, y_c) \tag{6}$$

Figure 3 shows the affects of using this additional face loss as well as the faces patches found by MTCNN and the face loss with and without facial preservation.

4. Experiments

We implement the network described by Johnson et al. [13] in PyTorch. The loss network uses PyTorch's VGG-16 network pretrained on ImageNet image classification. Additionally, we utilize PyTorch's implementation of instance normalization to replace the batch normalization in the original image transformation model. Ulyanov et al. showed that instance normalization improves the quality of style transfer [20]. The original model used fractional striding to up-sample the convolutional layers. To avoid checkerboarding artifacts, the results are first up-sampled using nearest-neighbor interpolation then convolved. Oden et al. showed that this reduces artifacts over transposed convolution by avoiding overlap [17].

4.1. Results of Style Transfer

We trained an image transformation model for six different style images (five shown in Figure 4 and one show in Figure 5). We train by completing two epochs of training using the 2017 COCO Dataset, an image dataset consisting of 118 thousand images [16]. Some results of stylized images are shown in Figure 4.

4.1.1 Style Weight

Careful choice of weights (λ_s, λ_c, and λ_{TV}) is important to get good results out of style transfer. For all models we fixed the weights to be $\lambda_s = 10^{10}$, $\lambda_c = 10^5$, and $\lambda_{\text{TV}} = 10^{-6}$.[3] We experimented with different style weights (results shown in Figure 5) to improved stylization, but the original weight of $\lambda_s = 10^{10}$ provided the best results.

4.1.2 Resolution Dependence

One issue with the style transfer of Johnson et al. is that the style features the model learns are resolution dependent. That is, when stylizing an image the resolution of the input affects the scale of style elements. Figure 6 shows this with the tiling of the mosaic style. We hypothesis the issue stems from the training. During training all content images are resized to a fixed size (256×256 in all of our examples), so the network learns to stylize that fixed resolution. Further experiments are need to prove this hypothesis, however. We

[3]The weights are borrowed from the open source implementation of fast style transfer (https://github.com/pytorch/examples/tree/master/fast_neural_style).

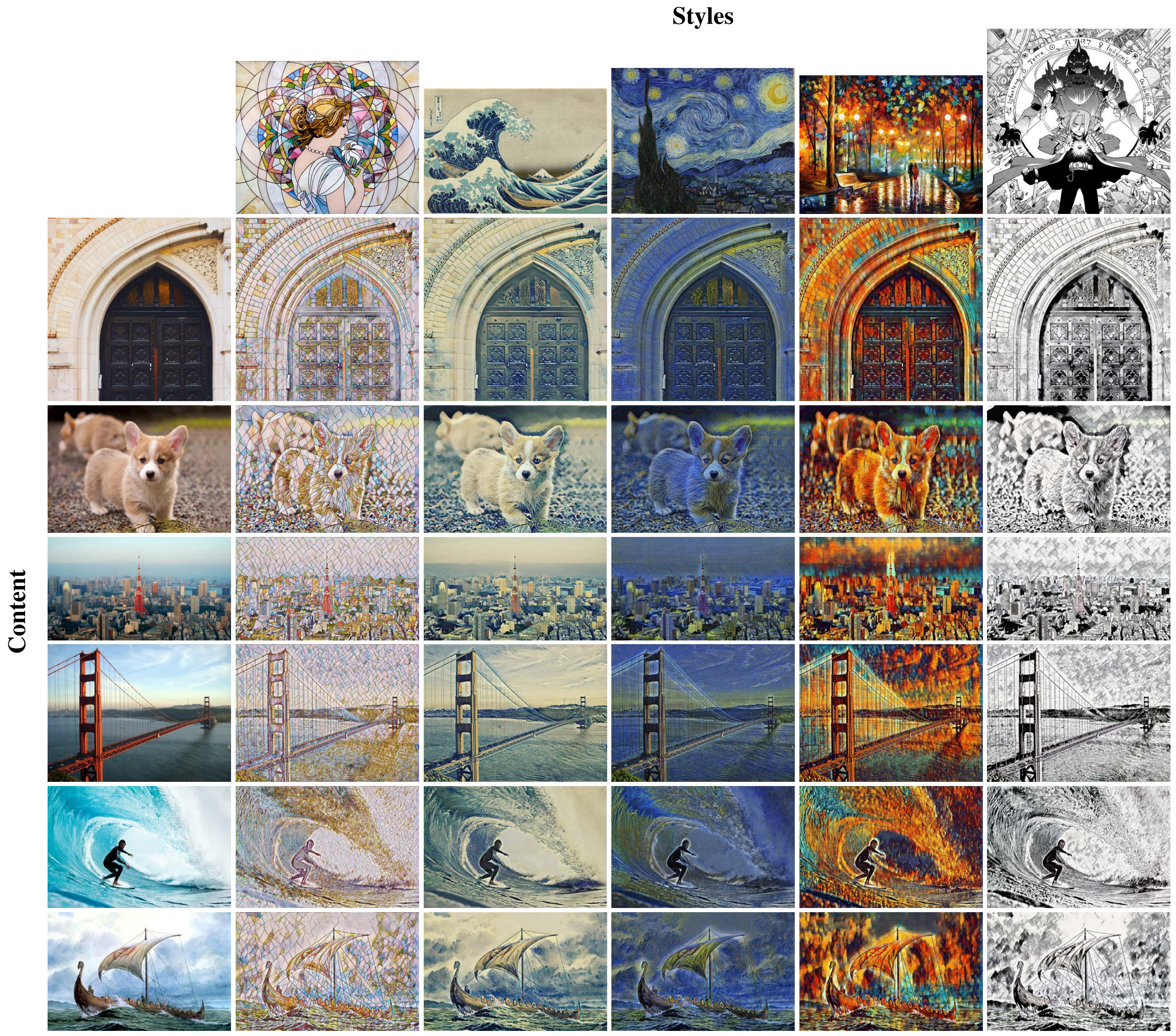

Figure 4: Style transfer results for 5 different style images. Styles images are (from left to right) *Mosaic*, *The Great Wave off Kanagawa* by Hokusai, *The Starry Night* by Vincent van Gogh, *Rain's Rustle* by Leonid Afremov, and artwork from *Fullmetal Alchemist* by Hiromu Arakawa.

suggest training more models on different content resolution (for example 128×128 or 512×512).

4.2. Stylizing Styles

As a experiment, we stylized each style image with the style of the others style images. From this we got mixed results. Some styles (e.g. mosaic style) transferred well, overriding the style of the content image. Other results, however, are subpar (e.g. *A Sunday Afternoon on the Island of La Grande Jatte*), resulting in patches of style placed on top of the original image. We hypothesis these issues come from transferring to an image that already has a lot of color and texture. In these cases, the style transfer has a hard time overriding the original style of the image. Interestingly, we can also exam the results of stylizing the same style image. These results for the most part are good with some of them resulting is slightly less contrast.

4.3. Results of Facial Preservation

We trained an image transformation model with style preservation for two different style images (one shown in Figure 9 and the other show in Figure 8). We continued to use the 2017 COCO Dataset and found many faces in the training images.

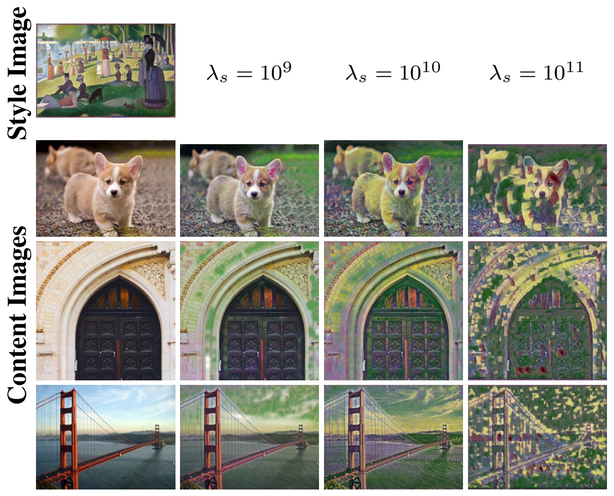

Figure 5: Varying the style weight λ_s by one order of magnitude in either direction can greatly affect the results of style transfer. Here the style of *A Sunday Afternoon on the Island of La Grande Jatte* by Georges Seurat is applied to three different content images with three different values of style weight. A value of $\lambda_s = 10^{10}$ produces the best results. With lighter weight, $\lambda_s = 10^9$, the content of the original image takes precedent and the stylization is weak. With heavier weight, $\lambda_s = 10^{11}$, the style takes precedent and original content is partially ignored.

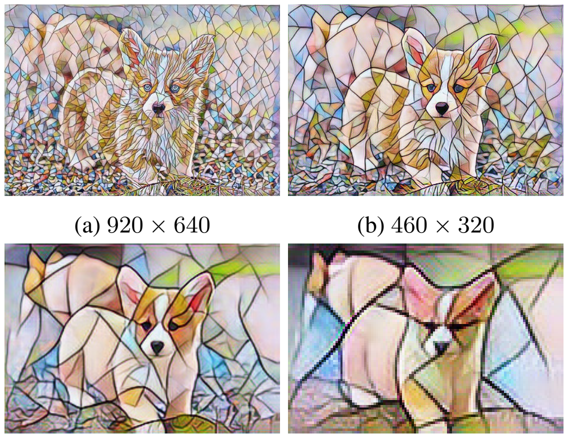

Figure 6: The network is trained on images of size 256 × 256, but the resolution can vary at evaluation. The learned style features depend on the size at training time, however. When the input is high resolution the style features are fine, as seen in (a), and when the input is small the transferred features are coarse, covering large areas of the stylized image, as in (d).

4.3.1 Face Weight

We experiment with various values for λ_f to fine results of facial preservation. The results for four different weights can be seen in Figure 10. In the end, we landed on a weight of $\lambda_f = 10^7$, three orders of magnitude less than the style weight. This weight worked well for the manga style (see in Figure 9), but may have been to high for other styles. Fine tuning this hyperparameter is the most challenging aspect of getting good facial preservation.

4.3.2 Quantitative Tests

To verify the results of facial preservation we stylized a face dataset (VGGFace2 [2]) and computed the difference in face loss without and with facial preservation. The results of these test are shown in Figure 11. Overall, facial preservation works well in terms of facial recognition. These quantitative results also confirm qualitative difference between the mosaic and manga style.

5. Discussion

While our face preserving model works well it is not without limitation. Resolving these limitations and expanding upon target content preservation is our biggest goals going forward.

5.1. Limitations

5.1.1 One Style Per Model

A limitation with the method of Johnson et al. is the limitation to one style per model. Others have show its is possible to train for multiple or an arbitrary style. Incorporating their work with ours is interesting future work.

5.1.2 Training Performance

Training fast style transfer models is fast, around four hours with a single Nvidia GeForce GTX 1080 Ti. However, the addition of facial preservation brings training time up to 10 or 12 hours. The additional time is due to several factors. First we have to process each image in a batch separately because faces and face locations are not shared between images. Additionally, not all training images have faces in them, but we still spend time using MTCCN to try and find faces. It may be beneficial to include meta-information to determine if face detection should be run.

5.2. Testing Method

We showed that we can test the performance of our style transfer using the face loss to quantify the quality of results. This does not guarantee that we will get appealing style transfer, just that the faces will still be recognizable.

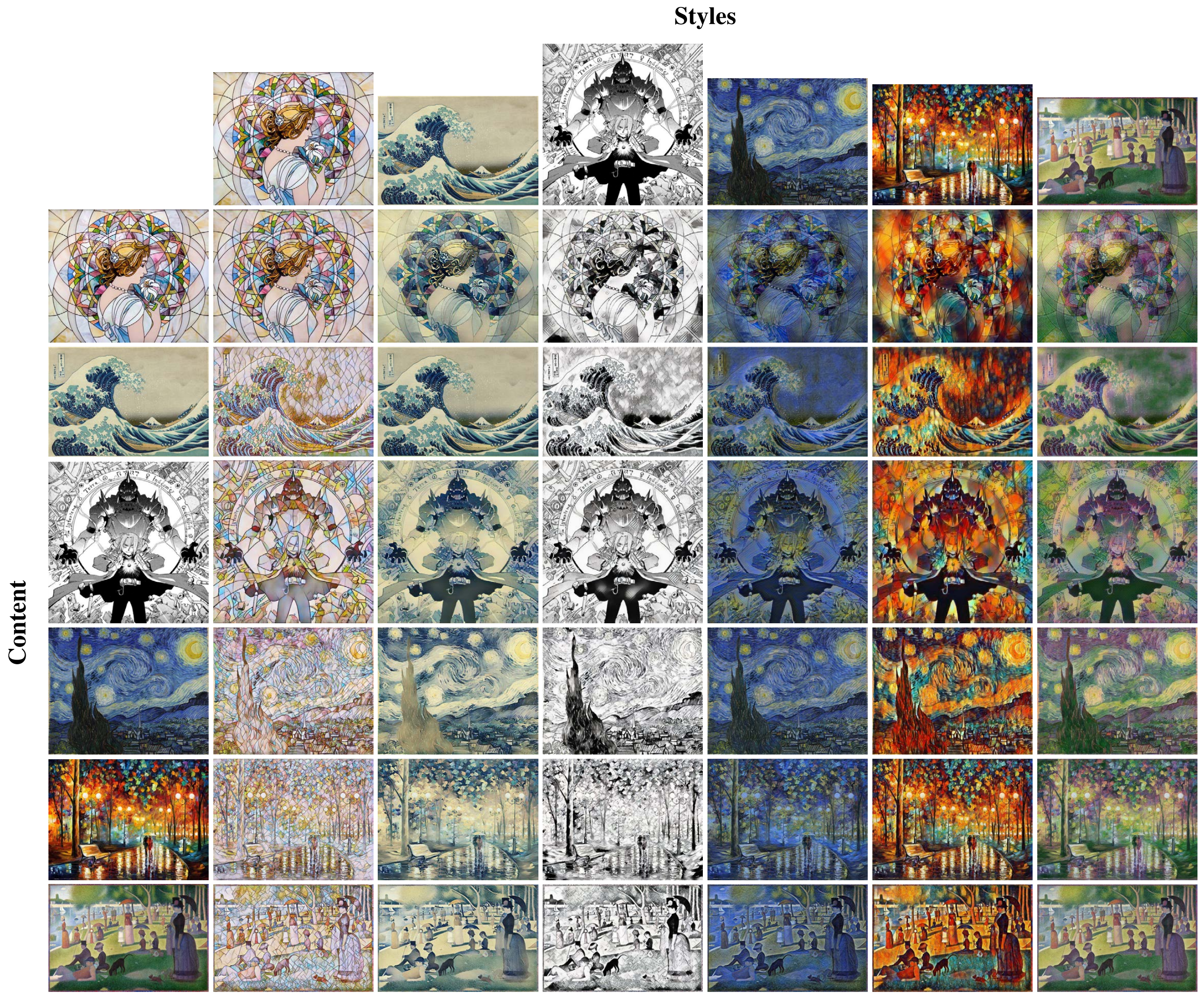

Figure 7: Style images used as style and content. The diagonal shows the style image stylized by the same style image. The residual blocks in the network help the network more easily learn the identity, so the diagonal images are only slightly transformed.

It may be worth to craft a measure of stylized quality or to perform a user study over a large dataset of stylized images.

5.3. Future Work

5.3.1 Learning Better Hyperparameters

One of the biggest challenges in training our network is the tuning of hyperparameters to weight loss terms. Their is no guarantee the weights of one style will work well with another. In the future we would like to incorporate hyperparameter optimization to learn weights for loss terms.

5.4. Training on a Face Dataset

We used the VGGFace2 test face dataset [2] to test the performance of our models, but we may get improved results by training on a general dataset hallways and then fine tuning on a training face dataset like VGGFace2's training dataset.

5.4.1 General Target Content Preservation

We have show that it is possible to curate losses that preserve target features of the content. While we limit ourselves to faces, we expect that this same methodology can be used to target any particular content. In the future, we would like to explore general user specified preservation. This maybe possible using visual attention in the loss network. One direction to improve our work would be to use a Face Attention Network by Wang et al. [22]. This may produce better results for smaller or more obscured faces.

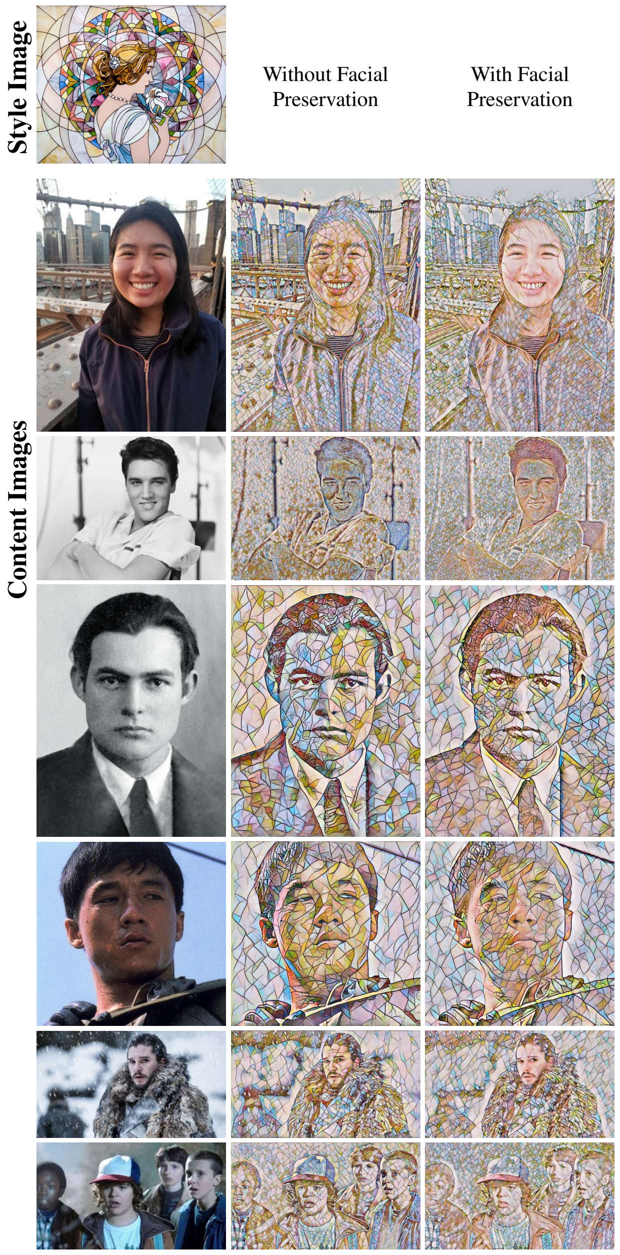

Figure 8: Results of stylizing faces without and with facial preservation with the mosaic style. We get mixed results with some faces appearing cleaner (row 1, 2, and 5), others showing little difference (row 3 and 6), and some worsening the face (row 4). Interestingly with the black and white picture of Elvis (row 2), the colorization temperature are opposite.

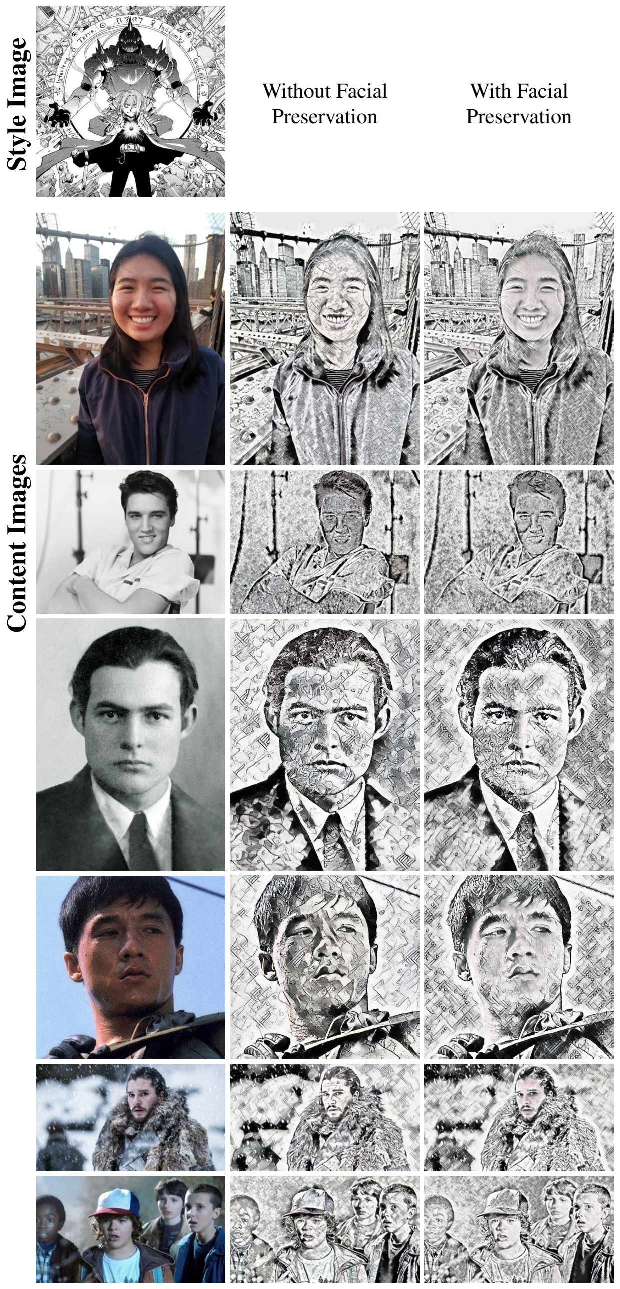

Figure 9: Results of stylizing faces without and with facial preservation with the manga style. With this style we get better results then that of Figure 8, both qualitatively and quantitatively as seen in Figure 11. Faces, in general, appear clearer, devoid of any style artifacts. The face loss, however, prevents some loss in color specifically around the face. For example, John Snow's face (row 5) has some color with facial preservation. In experimentation we found that gray-scale versions of a face has non-zero face loss.

References

[1] B. Amos, B. Ludwiczuk, and M. Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science, 2016.

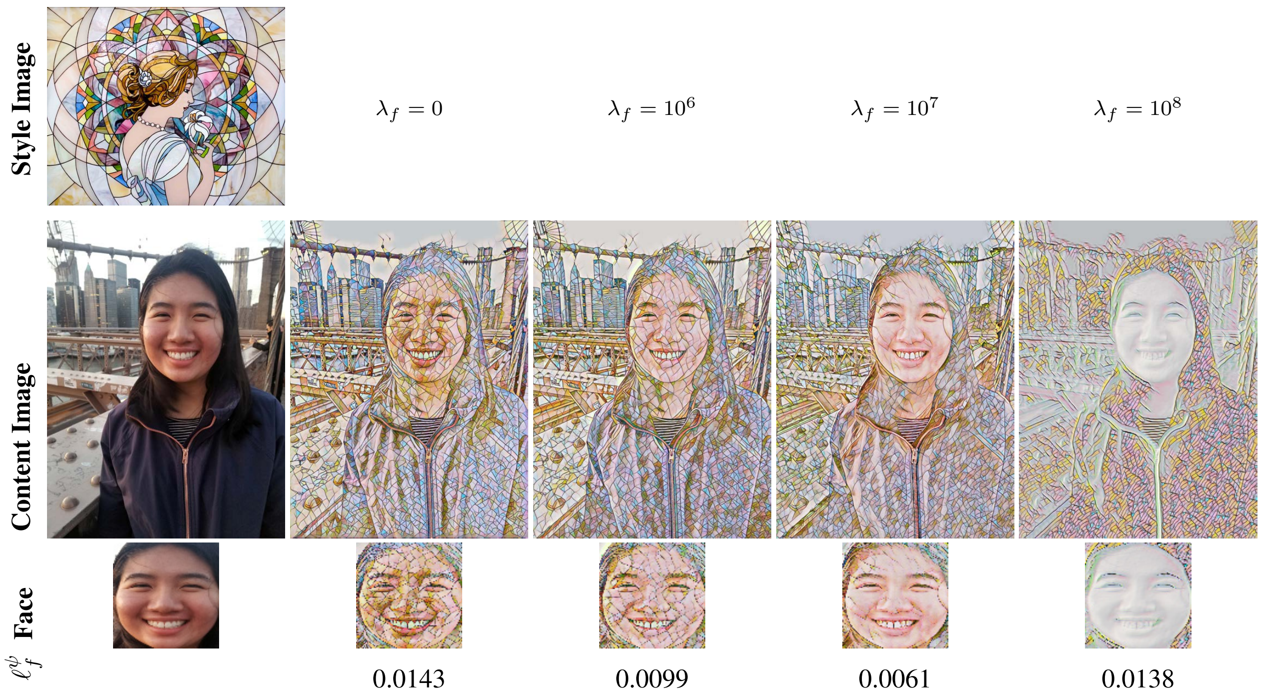

Figure 10: The weighting of the face loss in combination with the other loss weights can significantly impact the results of the facial preservation and style transfer. Without facial preservation ($\lambda_f = 0$) the tiling across the faces cause discolorations in the mosaic tiling. With facial preservation of $\lambda_f = 10^6$, the mosaic tiling is still present but the tiles are more uniform in color. With a $\lambda_f = 10^7$ the tiling starts to disappear and the face loss decreases even further. However, if the face loss weight increases further, $\lambda_f = 10^8$ the facial preservation significantly interferes with the stylization. Fine tuning this hyper-parameter is a challenge, but somewhere between 10^6 and 10^7 produces the best results without sacrificing style.

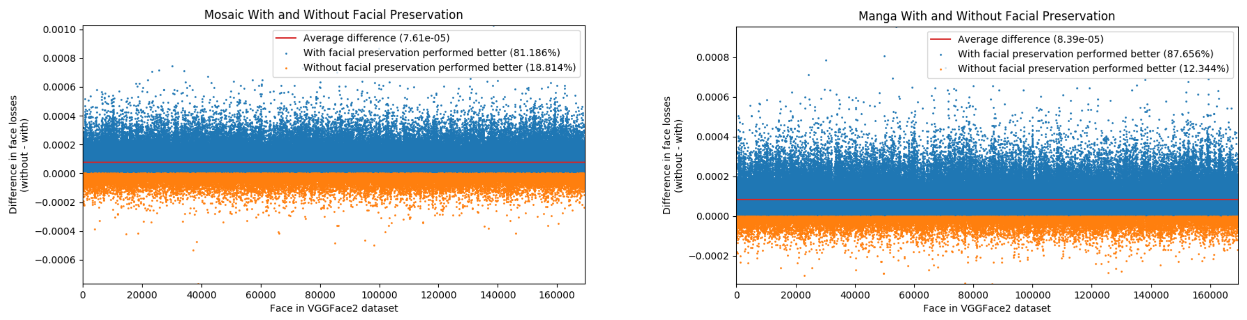

Figure 11: Face loss over the VGGFace2 test dataset [2]. Each image in the dataset is stylized with and without facial preservation (using our trained models), and the face loss of each image is computed. Plotted here are the differences between the face losses. Although the average face loss is less when using facial preservation there are instances where the face loss is less without facial preservation ($\sim 18.8\%$ for the mosaic style and $\sim 12.3\%$ for the manga style).

[2] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *International Conference on Automatic Face and Gesture Recognition*, 2018.

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

[4] T. Q. Chen and M. Schmidt. Fast patch-based style transfer of arbitrary style. *CoRR*, abs/1612.04337, 2016.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[6] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. *CoRR*, abs/1610.07629, 2016.

[7] L. A. Gatys, M. Bethge, A. Hertzmann, and E. Shechtman. Preserving color in neural artistic style transfer. *CoRR*, abs/1606.05897, 2016.

[8] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[9] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *CoRR*, abs/1705.06830, 2017.

[10] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 327–340, New York, NY, USA, 2001. ACM.

[11] X. Huang and S. J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *CoRR*, abs/1703.06868, 2017.

[12] Y. Jing, Y. Yang, Z. Feng, J. Ye, and M. Song. Neural style transfer: A review. *CoRR*, abs/1705.04058, 2017.

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

[14] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M. Yang. Diversified texture synthesis with feed-forward networks. *CoRR*, abs/1703.01664, 2017.

[15] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M. Yang. Universal style transfer via feature transforms. *CoRR*, abs/1705.08086, 2017.

[16] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[17] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[18] F. Shen, S. Yan, and G. Zeng. Meta networks for neural style transfer. *CoRR*, abs/1709.04111, 2017.

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[20] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[21] H. Wang, X. Liang, H. Zhang, D. Yeung, and E. P. Xing. Zm-net: Real-time zero-shot image manipulation network. *CoRR*, abs/1703.07255, 2017.

[22] J. Wang, Y. Yuan, and G. Yu. Face attention network: An effective face detector for the occluded faces. *CoRR*, abs/1711.07246, 2017.

[23] H. Zhang and K. J. Dana. Multi-style generative network for real-time transfer. *CoRR*, abs/1703.06953, 2017.

[24] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, Oct 2016.