BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

http://fit.vut.cz

KNN project

Image Style Transfer

Šimon Sedláček, Marek Sarvaš, Martin Osvald

1. Introduction

Image style transfer is simply the problem of applying the stylistic elements of one *style* image to a different, the so-called *content* image. Ideally, the resulting image would look as though it were painted/created by the same artist or technique of the style image, retaining its color scheme, stroke and detail granularity, etc. In this project, we explore and combine two novel approaches to style transfer, ArtFlow [1] and AdaAttN [8]. The code for this project was based on the official implementation of ArtFlow and the repository can be found on https://github.com/MarekSarvas/ArtFlow.

2. Neural Style Transfer

The first method that has used neural networks to achieve painting style transfer was presented in [3]. The authors of this paper used a pre-trained VGG convolutional neural network to separate the style and content factors from the input image. The content factor can be thought of as 'everything relevant for detecting and classifying objects present in the image'. On the other hand, the style factor could be regarded as a noise factor, that is spatially global to the whole image and has no effects on the classification/detection results. The exact principle of extracting the the content and style factors of an image using the VGG network can be found in [3], we will suffice with denoting this process as follows:

$$F_c = E_c(I) \quad (1)$$

$$F_s = E_s(I), \quad (2)$$

where F_c and F_s denote the extracted content and style feature maps, respectively, E_c and E_s denote the content and style extraction passes through the decoder, respectively and I is the input image.

Once separated, the content and style feature maps are then used as reference points for the style transfer itself, which in [3] was done by gradually (using gradient descent optimization) synthesising a completely new image from white noise, that would in the end match both the content and the style feature maps, obtained from the input images.

This method is, however, quite time consuming and therefore, other approaches to the style transfer process itself quickly followed. These newer approaches often use the same method for content/style separation. The new addition is, however, a more efficient style transfer module, that simply combines the style and content feature maps in a way that, once again, creates a new image with the style and content of the input images. These approaches include most notably the Adaptive Instance Normalization (AdaIN) approach [4] and the Whitening and Coloring Transform (WCT) [6]. However, there are some newer methods trying to expand on the basic idea of these style transfer modules such as AvatarNet or AdaAttN [10, 8], bringing some interesting results.

3. Artflow

Even though most neural style transfer approaches use a CNN trained for object detection to obtain the desired style and content feature maps, there are some novel approaches, which experiment with replacing this part

of the style transfer system. One of such approaches called ArtFlow was recently presented in [1].

This method replaces the original VGG-based encoder with a new neural network architecture, derived from the Glow [5] generative normalizing flow. The main idea behind using such an architecture is its capability for lossless decoding. This means, that once we obtain a latent vector z for an input image, we can pass this vector backwards through the network to reconstruct the original image perfectly.

This network architecture was used with hopes of mitigating some of the problems that come with the default 'inverse VGG' decoder used in other methods, mainly the problem of the so-called *content leak*. This flow-based network architecture should help retain more of the content information from the original image, therefore the stylized result should be less distorted, the content better distinguishable, etc.

3.1 ArtFlow network architecture

The basic building block of the network used in [1] is the so-called *flow step*. A flow step consists of three layers, that perform fully reversible transformations of their input. These layers include a reversible 1x1 convolution, and ActNorm layer and an affine coupling layer. Together, these three layers form one flow step.

These flow steps can be then grouped up, creating one *flow block*. Each flow block is also preceded with a special *squeeze* operation, which is simply a reversible alternative to the regular spacial pooling layers of typical CNNs. The inner workings and also problems linked with this layer are further discussed in section 5.

In contrast to the VGG-based approaches, the ArtFlow network used in the same manner for both style and content image encoding – both input images will go through the entire network, giving us their representations in the latent space at the output of the network. These latent z vectors then serve as input of the used style transfer module (e.g. AdaIN), resulting in a stylized z_{cs} latent vector that can then be passed through the network in reverse to obtain the final stylized image.

In the ArtFlow paper, best results were achieved when employing two blocks of eight flow modules each in the final model. The whole ArtFlow system is depicted in figure 1.

3.2 ArtFlow style transfer

As it was stated in the previous section, we first obtain the z-space representations for both the content and style images I_c and I_s:

$$z_c = A(I_c) \tag{3}$$
$$z_s = A(I_s), \tag{4}$$

with A denoting the ArtFlow encoder.

These vectors are then passed to the style transfer module, e.g. AdaIN. The style transfer module applies a rather simple transform to these vectors, resulting in a stylized feature map z_{cs}:

$$z_{cs} = AdaIN(z_c, z_s). \tag{5}$$

This stylized z-vector is passed backwards throught the encoder, resulting in the final stylized image:

$$I_{cs} = A^{-1}(z_{cs}). \tag{6}$$

The whole projection-transfer-reversion scheme is depicted in figure 1.

3.3 Training ArtFlow

It is now obvious that the aim when training the ArtFlow system is to find the parameters of the encoder so that the style transfer module (which has no learnable parameters) can do the best job possible.

In order to do that, some cost functions have to be defined for this purpose. There are two const functions used, one supervising the content and one supervising the style factor of the resulting images:

$$L_c = \| A(A^{-1}z_{cs}, z_{cs}) \|_2 \tag{7}$$
$$L_s = \sum_{i=1\ldots L} \| \mu(E_{si}(A^{-1}(z_{cs}))) - \mu(E_{si}(I_s)) \|_2 + \sum_{i=1\ldots L} \| \sigma(E_{si}(A^{-1}(z_{cs}))) - \sigma(E_{si}(I_s)) \|_2 . \tag{8}$$

It is important to note that the style loss function L_s makes use of the VGG encoder as a reference point for style separation, where E_{si} denotes a layer in the VGG net from which we extract the style features. In other words, the whole VGG encoder is a part of the loss function.

3.4 Style transfer bias

Last important thing to note about the ArtFlow approach is that it uses style transfer methods that are *unbiased*. This means that the style transfer module will perform style transfer without prioritizing content or style in the resulting image, retaining maximum original content information. This can be demonstrated by stylizing an input content image with a particular style and the stylizing this new image again with the style of the original input image. ArtFlow achieves perfect

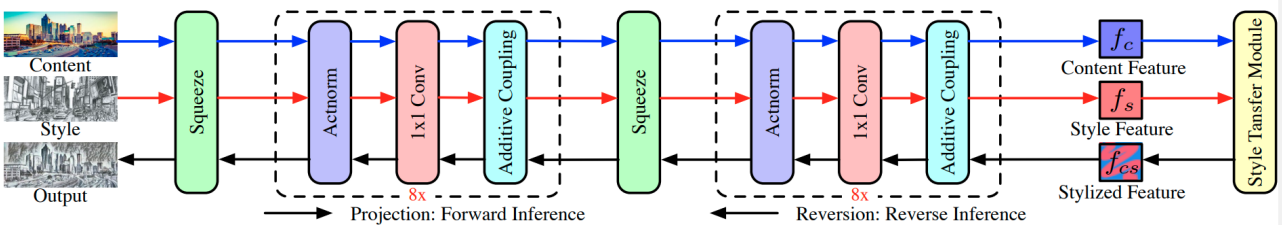

Figure 1. Depiction of the whole ArtFlow system with two Glow blocks of eight flow steps each. The diagram also shows the inference directions for both encoding and decoding. Diagram taken from [1].

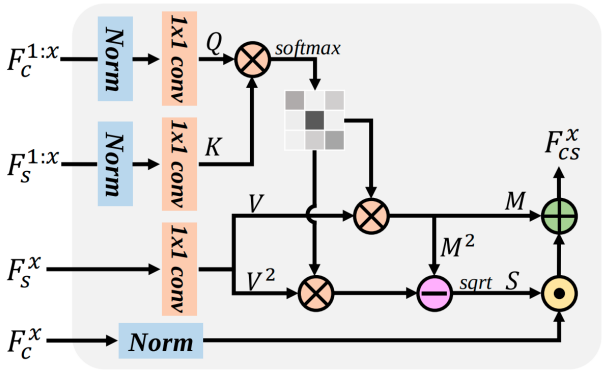

Figure 2. A diagram of the AdaAttN style transfer module taken from [8]. The *Norm* blocks are regular channel-wise mean-variance normalization layers.

Figure 3. A showcase of the grid-shaped artifacts present in the stylized images.

reconstruction of the original image this way. More on this topic can be found in section 7.3.

However, we found that some of the results may look 'too clean' and surgical and that some of the biased style transfer modules such as Avatar-Net or AdaAttN [10, 8] achieve subjectively better style transfer by more or less biasing towards style.

4. Our approach

The fact that biased style transfer modules can lead to better stylistic results prompted us to create our own system, which would combine the properties of the ArtFlow encoder and the capabilities of novel style transfer modules such as AdaAttN.

AdaAttN is essentially a generalization of AdaIN, taking the same base principle but adding learnable parameters and exploiting attention to better retain the aspects of the original style image. A diagram of the module can be seen in figure 2.

In combination with ArtFlow, we use the obtained z_c vector in place of the F_c^x and $F_c^{1:x}$ AdaAttN inputs and the z_s vector in place of F_s^x and $F_s^{1:x}$.

Ultimately, we would like the new system to better capture spacial style aspects than the AdaIN-based baseline, resulting in more lively, genuine and aesthetic stylization.

We adopt the same loss functions as regular ArtFlow, experimenting with different amounts of flow block and flow steps within.

5. Challenges of the proposed method

Even though the qualitative results of our proposed system are satisfactory and comparable to the original ArtFlow results, there are some significant drawbacks that hinder the usefulness of our method in the real world. We will discuss these drawbacks here, before presenting the results themselves, as it is important to understand some of the terms and aspects of the final models that were implemented in reaction to these problems.

5.1 Grid artifacts

The first and probably the biggest limitation of the ArtFlow-based methods are the inevitable grid-shaped artifacts that will appear in the reconstructed images. This phenomenon was not discussed in [1] the original paper at all and posed an unpleasant surprise for us when we got our first results. An example of the grid artifacts on a zoomed-in image can be seen in figure **??**

This phenomenon was, however, mentioned in [9] and is caused by the usage of the previously described squeeze layers at the beginning of each flow block. This squeeze layer performs a spatial splitting operation, which essentially splits the original image into multiple channels simply by dissecting neighbouring pixels from each other in a checkerboard-like pattern. Each squeeze layer will therefore quadruple the cur-

rent number channels and will half the current width and height of the image. However, it appears that after applying style transfer, some some of the local correlations between pixels are corrupted, resulting in this grid-like artifact pattern in the final reconstructed image.

The same paper also suggests that a way to mitigate this phenomenon is to add a few *transition* layers to the model. Each transition layer is simply the original flow step, but with the affine coupling layer removed. Adding these transition layers to the model right after the squeeze operation should allow the model to learn a linear interpolation between the neighbouring pixels, allowing the model to fight the artifacts.

Unfortunately, we had no success with removing the grid patterns from the images even after adding the transition layers. It is, however, possible, that by designing a custom loss function, which would penalize the grid artifacts in the final image, we would be able to push the final model into the right direction.

What does help is using fewer squeeze operations in the model. The models with only one flow block instead of two have the grid effect noticeably reduced, this is, however, at the cost of slight color saturation loss and slight general style transfer capability deterioration. Also, the artifacts are much more visible in images with many colors. When going for a drawing-type monochrome stylization, the artifacts tend to recede.

The grid pattern is most noticeable on the edges of each image, resulting not only in strong and visible grid lines, but also in some unexpected color stripes along the edges of the image. This is depicted in figure 4. To work around this, we tried to wrap the input content image with 64 pixels of padding on each side, effectively trying to push these edge artifacts beyond the edges of the image before cropping it back to the original size. This yielded some satisfying results, significantly reducing the corruption of the image edges after style transfer.

5.2 Memory consumption

Because the AdaAttN style transfer module uses the attention mechanism, it is at one point necessary to generate an attention map between the key and the query. This is done by applying the outer product to these two tensors. However, if these two input query and key tensors are too large, the outer product matrix will be quite large.

This problem gets even more pronounced when working with higher resolution images, say 512 pixels in each dimension. The query and key tensors can then have shapes that are simply too large for us to compute their outer product (in some instances, this results in tensors of more than 40 GB of CUDA memory in size).

This issue is even more pronounced when using an ArtFlow architecture with only one block – the output vector z will have most of its original information retained in the original width and height dimensions. As we increase the number of flow blocks of the network, more and more of the original image dimensionality will be moved to the channel dimensions, resulting in a more manageable width and height values.

There is, however, one way to tackle this problem, which is simply to limit the width and height values of the query and key tensors, for example by randomly selecting a number of indices in the original latent vector to create a new query/key tensor. We observed, that it is useful to keep the size of this newly created tensor to more than 64 values per dimension, otherwise the attention map simply will not retain enough detail and information to perform well.

Even though that limiting the attention map size does solve the problem partially, most style transfer instances will still require gigabytes of CUDA memory to generate the attention maps, making this method hardly usable in most real life scenarios, at least in its current state and without a GPU.

6. Experimental setup

For training, we used the MS-COCO [7] dataset as a source of content images, totalling at 120 thousand images. For styles, we used the Wikiart[1], totalling at 40 thousand style images.

All models were trained for a total of 40000 iterations with a batch size of 4. The initial learning rate was set to $1 \cdot 10^{-4}$ with a decay factor of $5 \cdot 10^{-5}$.

We make use of the computational resources provided by Metacentrum VO[2], specifically the Nvidia A40 48 GB GPUs on the new Galdor cluster, each model taking seven to eight hours to train on average.

7. Results, architecture comparison

We train the system in multiple configurations, namely 1x16, 2x8, 1x24, 3x4 flow steps with all of these configurations also having its variant with and without one or two transition flow layers in each block. In this section, we will discuss the results of the best performing trained models, as some of the models were completely unusable – basically any architecture with three or more flow blocks. The images that such systems would produce failed to retain enough content information to be aesthetically pleasing and would

[1] https://www.wikiart.org/
[2] https://metavo.metacentrum.cz/

Figure 4. Comparison of style transfer on content image (left) and content image padded with 64 pixels with value 0 on each side (right). Padding solves quite noticeable artifacts that appear mostly on the sides (in this case, right and bottom sides of the left image). In addition style transfer on padded images has at least subjectively more pleasant colors, less artifacts and keeps a bit more content information and detail. The style used was the Edvard Munch's Scream painting.

be more reminiscent of blobs of colors and random artifacts, rather than the original content.

7.1 Subjective aesthetic evaluation

We compared our models that give best results with pre-trained flow models with style transfer modules being AdaIn and WCT. Results of such comparion can be seen in the figure 8. Comparing our two models in the first 2 columns, the model with 2 flow blocks transfers more accurate colours. Also, model with the 1 flow block preserves better details but performs worse in transfering style in macro scale. Therefore, model with two flow blocks has overall more aesthetically pleasant results. AdaIn and wct have comparable results to our models. However in some cases (like the "city" or "cat" image) AdaIn and WCT do not have as accurate colours taken from style image. Both preserve some colours from original content image, which if you want to do complete style transfer can be seen as a worse result. In addition, even though our proposed models should be biased by a style a bit, they preserves more accurate details from content image than the AdaIn. For example in the "city" image (content) and "pencil drawing" image (style), our models have more accurate reflections on the water.

7.2 Multi-Style transfer experiments

To verify if the value of attention based style transfer module was preserved our proposed model, we tried experimenting with interpolation of two different styles and use such new styles for the style transfer on the content image.

We took two images representing styles, extract z-vector from each style image in forward pass of neural network. Then interpolate these two vectors creating multiple new style z-vectors, where new z-vectors applied on the content show both styles with more or less biases towards one of the styles. This shows run-time flexibility of the proposed model, that can be used for more interesting experiments without need of training it from scratch, e.g.: for tasks similar to multi-style transfer. The results of this experiment can be seen in the figure 5.

The architecture with only one flow block has visibly worse coloring than the one with two flow blocks, however, the interpolation between the two styles works either way.

7.3 Reverse style transfer

As discussed in section 3.4, the AdaAttN style transfer module is not unbiased. Therefore, it is highly probable that some content information will be lost during the transfer process.

This can easily be demonstrated by styling one source image with an arbitrary style and then trying to replace this new style with the style of the original content image. ArtFlow in combination with an unbiased style transfer module such as AdaIN or WCT will be able to perfectly reconstruct the original content image. However, since AdaAttN is not unbiased, ArtFlow in combination with this particular style transfer module will not.

This reverse style transfer capatiblity can be observed in figure 6.

7.4 Content preservation via ImageNet classification

Another way to test how well the content information is preserved in the stylized images is to try to classify the objects in the images via a pre-trained network. For this purpose, we used an ImageNet [2] sample of 5000 images – 1000 classes of five examples each. This dataset was then stylized using first the original ArtFlow AdaIN model and then with our two AdaAttN

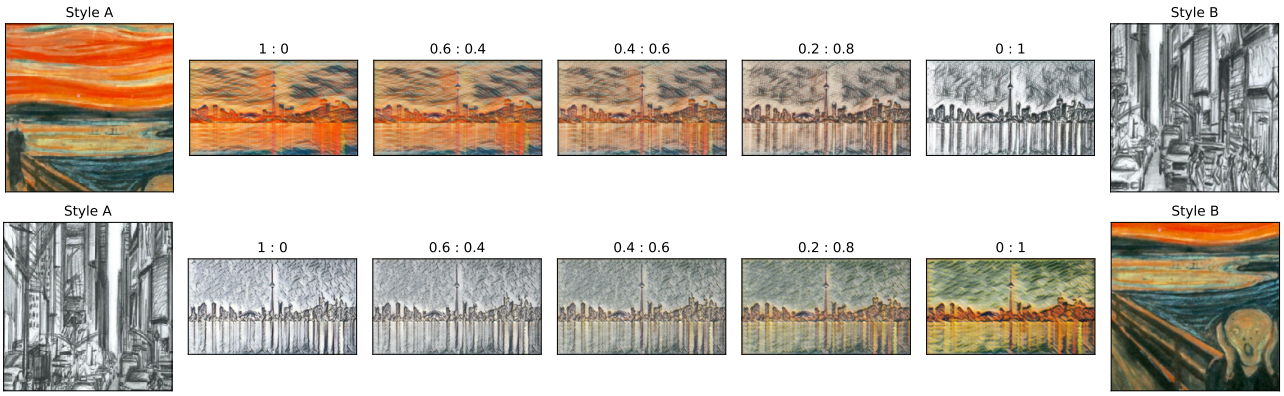

Figure 5. Interpolation of two style (far left and far right) for two different network architectures. On the top, the network has 2 flow blocks each with 8 flow steps. The bottom image was generated by an architecture consisting of 1 flow block with 16 flow steps. Numbers above images represent proportion of given style images in applied style.

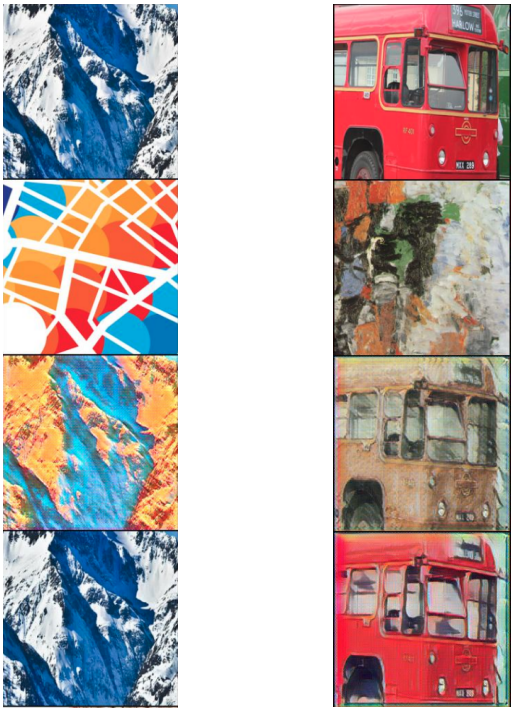

Figure 6. Reverse style transfer demonstration. The top picture is the content, second is the style, third is the stylized image and the last image is the reconstructed original content image. On the left, ArtFlow with AdaIN is perfectly capable of replacing the new style with the original content image style. On the right, the bias of AdaAttN makes it impossible to do so – the reconstructed original image retains some of the style characteristics, some detail is gone and the image is a bit distorted as a whole.

ArtFlow variants, one with 2x8 and the other with 1x16 flow steps.

We then use three models that were pre-trained on the ImageNet dataset to classify these stylized images. The results can be seen in figure 7.

Once again, it is apparent that the unbiased nature of AdaIN allows for better preservation of the content factor. Interestingly enough though, the classification accuracy for our 2x8 AdaAttN network is a bit higher than for the 1x16 network, even though we would generally consider the 2x8 model to be more 'aggressive' in the styling.

8. Conclusions

In this project, we combined two novel approaches to image style transfer, ArtFlow and AdaAttN in hopes of retaining the content clarity provided by ArtFlow and the stylistic aspects provided by the AdaAttN style transfer module. The resulting trained systems are producing decent results, held back only by some grid-shaped artifacts caused by the squeeze layers of the ArtFlow neural network. We evaluated the resulting systems both in terms of subjective aesthetic analysis and also by performing some objective experiments, namely interpolating between different styles and trying to classify the content of stylized images.

Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic.

References

[1] AN, J., HUANG, S., SONG, Y., DOU, D., LIU, W. et al. ArtFlow: Unbiased image style transfer via reversible neural flows. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.

[2] DENG, J., DONG, W., SOCHER, R., LI, L.-J., LI, K. et al. Imagenet: A large-scale hierarchical image database. In: Ieee. *2009 IEEE conference on*

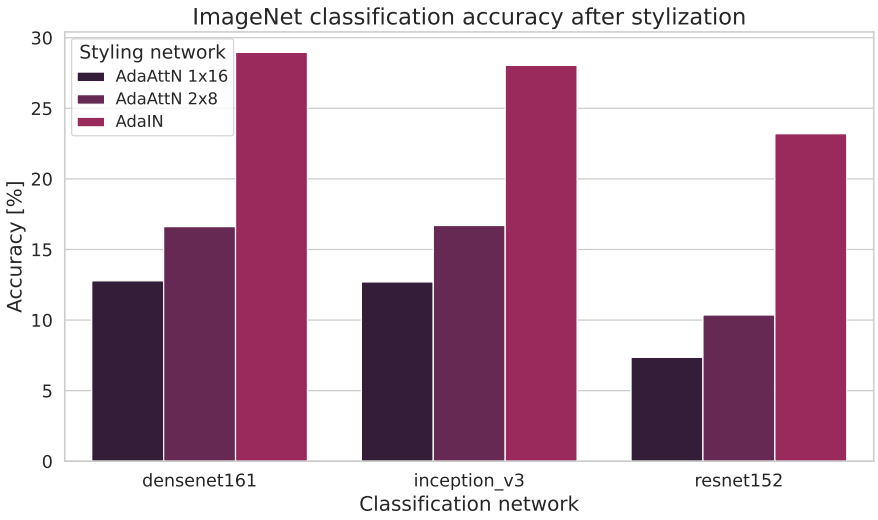

Figure 7. ImageNet classification results after stylizing the data.

computer vision and pattern recognition. 2009, p. 248–255.

[3] GATYS, L. A., ECKER, A. S. and BETHGE, M. *A Neural Algorithm of Artistic Style*. arXiv, 2015. Available at: https://arxiv.org/abs/1508.06576.

[4] HUANG, X. and BELONGIE, S. *Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization*. arXiv, 2017. Available at: https://arxiv.org/abs/1703.06868.

[5] KINGMA, D. P. and DHARIWAL, P. *Glow: Generative Flow with Invertible 1x1 Convolutions*. arXiv, 2018. Available at: https://arxiv.org/abs/1807.03039.

[6] LI, Y., FANG, C., YANG, J., WANG, Z., LU, X. et al. Universal Style Transfer via Feature Transforms. In: GUYON, I., LUXBURG, U. V., BENGIO, S., WALLACH, H., FERGUS, R. et al., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. Available at: https://proceedings.neurips.cc/paper/2017/file/49182f81e6a13cf5eaa496d51fea6406-Paper.pdf.

[7] LIN, T.-Y., MAIRE, M., BELONGIE, S., BOURDEV, L., GIRSHICK, R. et al. *Microsoft COCO: Common Objects in Context*. arXiv, 2014. Available at: https://arxiv.org/abs/1405.0312.

[8] LIU, S., LIN, T., HE, D., LI, F., WANG, M. et al. AdaAttN: Revisit Attention Mechanism in Arbitrary Neural Style Transfer. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2021.

[9] LUGMAYR, A., DANELLJAN, M., VAN GOOL, L. and TIMOFTE, R. *SRFlow: Learning the Super-Resolution Space with Normalizing Flow*. arXiv, 2020. Available at: https://arxiv.org/abs/2006.14200.

[10] SHENG, L., LIN, Z., SHAO, J. and WANG, X. Avatar-Net: Multi-scale Zero-shot Style Transfer by Feature Decoration. In: *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. 2018, p. 1–9.

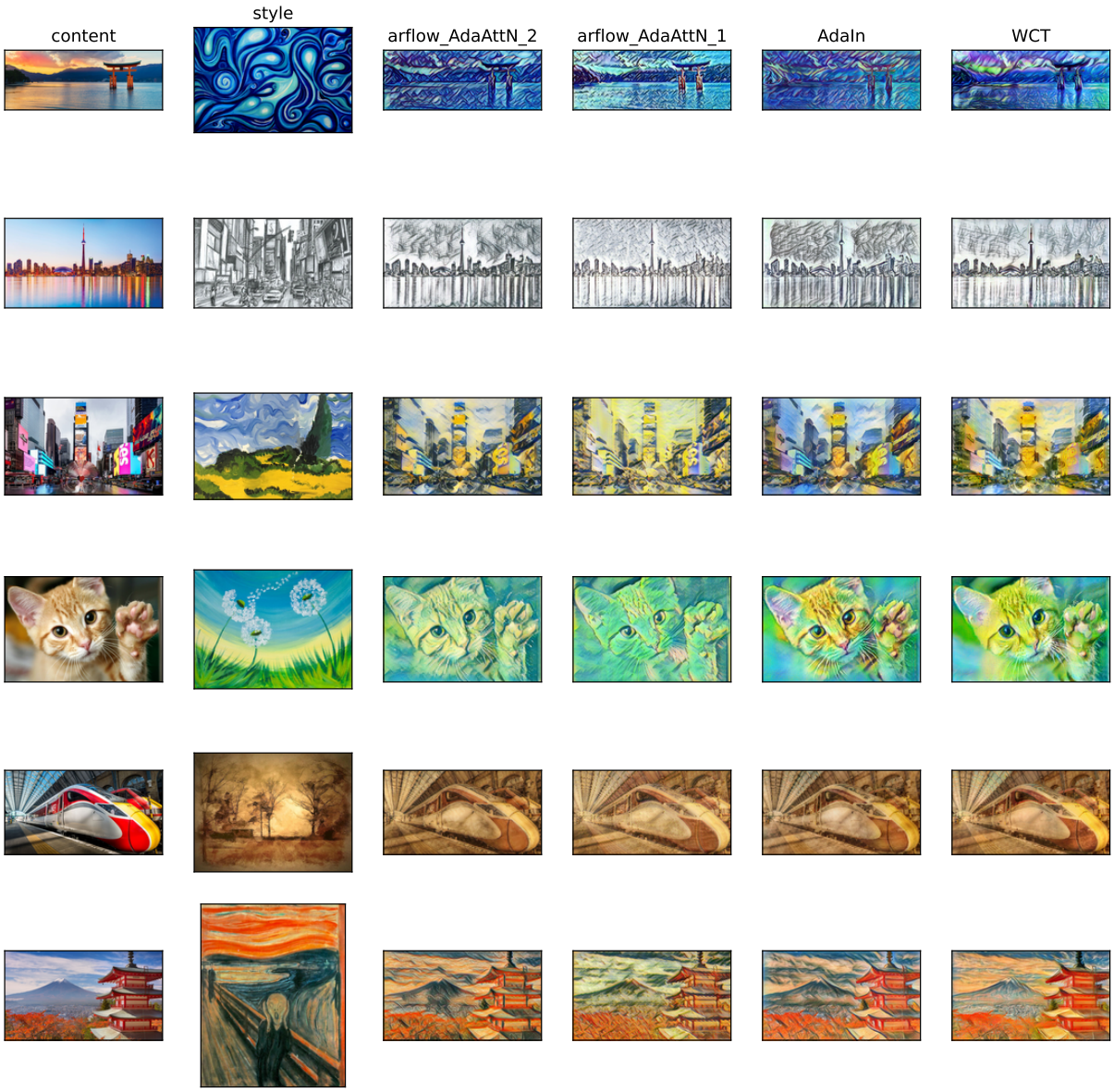

Figure 8. Comparison of proposed models on sub-sample of content and style images.