Exploring Decoder Architectures in AdaIN-based Style Transfer

ECE 285

Achyut Pillai
Electrical and Computer Engineering Department
A16646336
apillai@ucsd.edu

Abstract

Neural style transfer (NST) via Adaptive Instance Normalization (AdaIN) offers real-time stylization by aligning content features with style statistics. In this paper, I address how different decoder architectures (U-Net skip-connections, lightweight self-attention modules, and ResNet-inspired residual blocks) affect the trade-off between content reconstruction and style rendering. Starting with the baseline AdaIN pipeline, the decoders were trained on content images from the DIV2K dataset and style references selected from the "Best Artworks of All Time" dataset. Model performance was evaluated using perceptual metrics (PSNR, SSIM, LPIPS) alongside style/content loss curves and visual comparisons. By isolating the decoder structures, it was revealed that there is a clear trade off between content preservation and style weight, with the U-net architecture consistently demonstrating better results with the ability to reconstruct fine details while still applying the style.

1 Introduction

The goal of Neural Style Transfer (NST) is to create a new image by combining the content of one image and the style of another through neural networks. This problem was first addressed by Gatys et al. [2] and while the methods they used were slow and computationally intensive, it produced results showcasing how extensively it could be used in creative and artistic applications. Most subsequent NST works have focused on improving real time performance.

One key work in real time style transfer is the Adaptive Instance Normalization (AdaIN) method introduced by Huang and Belongie [4]. AdaIN matches the mean and variance of features from a content image to those in a style image. This essentially allows for style transfer without the need to retrain every new style. However, the standard AdaIN uses a mirrored VGG decoder to generate the final image, which often faces a significant trade off between preserving detailed content and fully expressing style textures and colors.

In this paper, I evaluate how different decoder architectures influence the outcome of AdaIN-based style transfer in terms of the visual quality, content correctness, and inference speed. I hypothesize that different decoder architectures can improve the trade off between content and style while still offering real time computation. Four different decoders were tested, the baseline VGG-mirror, a deeper ResNet-style decoder, a self-attention decoder to adaptively propogate features, and a U-Net decoder with skip connections to reinforce spatial details.

The results show that the U-Net consistently achieves a higher content fidelity while maintaining fast computation. The ResNet decoder also achieves a close content fidelity, but has a slightly longer inference time.

2 Related Work

This paper builds on several works in neural style transfer and perceptual metrics.

Neural Style Transfer. The introductory work by Gatys et al. [2] first demonstrated that deep features from convolutional neural networks (CNNs) could be used to separate and combine the content and style of images. This was done by matching the deeper layer (content) features of one image and the style features (correlations between features, represented by a Gram matrix) of another.

Real-Time Style Transfer. To overcome the slow optimization process of the original method, Johnson et al. [6] proposed training a feed-forward network for a single style, enabling real-time performance. This work also introduced "perceptual loss," which measures differences in the feature domain instead of the pixel domain. This paper uses the same content and style loss formulations.

Arbitrary Real-Time Style Transfer. The AdaIN method by Huang and Belongie [4] is the direct foundation of this paper. They introduced the AdaIN layer as a parameter-free way to align feature statistics, enabling a single model to apply any style in real time. This paper keeps the AdaIN layer and VGG encoder fixed while focusing on improving the decoder component they proposed.

Perceptual Similarity Metrics. Metrics like PSNR and SSIM do not always align with human perception of image quality. The Learned Perceptual Image Patch Similarity (LPIPS) metric, proposed by Zhang et al. [9], uses a pretrained neural network to measure the perceptual distance between two images. This paper uses it as a final metric in the experiments.

3 Method

My method follows the standard AdaIN pipeline, which consists of a fixed VGG-19 encoder, the AdaIN layer, and a trainable decoder. I instead experiment with four different architectures for this decoder.

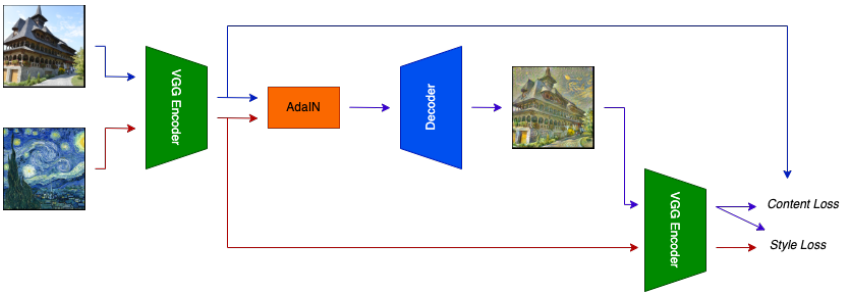


Figure 1: An overview of the style transfer pipeline. Decoders are changed while keeping the remaining structure the same.

3.1 Overall Pipeline

The process begins with a content image, I_c, and a style image, I_s.

1. A fixed VGG-19 encoder, ϕ, extracts feature maps at various depths. I use features up to layer $relu4_1$.
2. The deepest content feature map, $\phi(I_c)$, and style feature map, $\phi(I_s)$, are fed into the AdaIN layer to produce a target stylized feature map, t

$$t = \text{AdaIN}(\phi(I_c), \phi(I_s)) = \sigma(\phi(I_s)) \left(\frac{\phi(I_c) - \mu(\phi(I_c))}{\sigma(\phi(I_c))} \right) + \mu(\phi(I_s)) \tag{1}$$

3. A decoder, g, takes t as input and generates the final output image with the style, $I_{out} = g(t)$.

3.2 Loss Function

The decoder is trained by minimizing a combination of a content loss and a style loss. The output image I_{out} is passed back through the encoder ϕ to get its feature representations. The content loss encourages the output to match the content of the target feature map t. It is the Euclidean distance between the $relu4_1$ features of the output image and the target features. This differs from the original paper where they use the output of the AdaIN layer for the target features instead of the content image.

$$\mathcal{L}_c = ||\phi_4(I_{out}) - t||_2 \tag{2}$$

The style loss encourages the output to match the style of the style image I_s. This loss is computed across multiple layers ($relu1_1$, $relu2_1$, $relu3_1$, $relu4_1$) as the sum of the distances between the mean and standard deviation of the output features and the style features.

$$\mathcal{L}_s = \sum_{i=1}^{4} ||\mu(\phi_i(I_{out})) - \mu(\phi_i(I_s))||_2 + ||\sigma(\phi_i(I_{out})) - \sigma(\phi_i(I_s))||_2 \tag{3}$$

The total loss is a weighted sum: $\mathcal{L}_{total} = \mathcal{L}_c + \lambda\mathcal{L}_s$, where λ is the style weight.

3.3 Decoder Architectures

I implemented and trained four different decoder architectures.

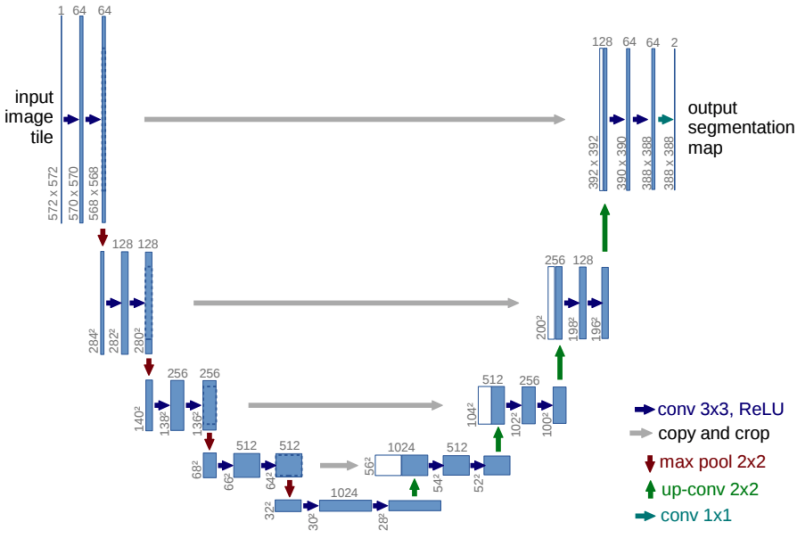


Figure 2: The original U-Net architecture from [7]

The Baseline Decoder is a simple network of convolutional and upsampling layers, mirroring the VGG encoder structure, as proposed in the original AdaIN paper [4]. It critically does not have normalization layers because normalization tends to normalize every batch or sample to be centered around one style, whereas the decoder should be able to generate images with vastly different styles.

The ResNet Decoder replaces the standard convolutional layers with 2 layer residual blocks [3]. Each block is

$$\text{out} = x + \left[\text{Conv}_{3\times3}(\,\text{ReLU}(\text{Conv}_{3\times3}(x)))\right]$$

and three of these blocks are placed after every upsample stage for every spatial resolution. This allows for deeper networks that are easier to train and can potentially learn more complex mappings from features to pixels.

The Attention Decoder incorporates lightweight self-attention modules [8] before the upsampling stages given by

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T)\,V, \quad \text{where } Q, K \in \mathbb{R}^{(HW)\times(C/8)} \text{ and } V \in \mathbb{R}^{(HW)\times C}$$

The scaling parameter lets the network gradually decide how much mixing to apply from the nonlocal means. With the use of both early and late attention, the goal is to allow the network to model long-range dependencies and propagate style information more effectively across the image.

The UNet Decoder utilizes a U-Net architecture 2[7] with skip connections that feed feature maps from the encoder ($e_1 : 256 \times 256,\ e_2 : 128 \times 128,\ e_3 : 64 \times 64$) directly to corresponding layers in the decoder. This provides high-frequency information that is often lost in the deep feature maps, helping to preserve fine content details like edges and textures.

3.4 Training Strategy

Each decoder is trained independently using a two dataset loop over content and style images. Certain hyperparameters were fixed for the entire experiment: an image size of 256, a learning rate of 1×10^{-4}, and a total of 150 epochs.
Different style weights were experimented on for all decoders: $[3,\ 7,\ 10,\ 20,\ 30]$

Content images are drawn from the DIV2K training set, while style images come from a curated list of 15 artist subdirectories; both datasets are resized and center-cropped to 256×256, converted to tensors, and shuffled in separate dataloaders.

For each decoder, a fresh VGG-19 encoder and the specified decoder is initialized, then only the decoder parameters are optimized via Adam.

At every training step:

1. Extract multi-layer features from content and style via the encoder.
2. Fuse the content and style features using AdaIN.
3. Decode the stylized feature into an image.
4. Re-encode the generated image and compute the sum of content loss and the weighted style loss.
5. Backpropagate and update decoder weights.

4 Experiments

Each decoder was trained independently across five style weight (SW) settings (3, 7, 10, 20, and 30). Their performance was then evaluated on the content validation set using four quantitative metrics, PSNR, SSIM, LPIPS, and average inference time. These results were complemented with a qualitative assessment by visually inspecting the stylized outputs.

4.1 Datasets

Two datasets were used, the DIV2K dataset for content images and the Best Artworks of All Time dataset for the style images.

4.1.1 DIV2K Dataset

The DIV2K dataset [1] is a diverse collection of high resolution images, specifically curated for image super-resolution tasks.

- **Data Format**: Images are in PNG format with 2K resolution.
- **Dataset Split**: 800 Training Images, 100 Validation Images

In the interest of computation resources and time, only 300 training images and 50 validation images were used.

4.1.2 Best Artworks of All Time Dataset

The Best Artworks of All Time dataset [5] is a collection of paintings of 50 of the most influential artists of all time curated for a classification task.

- **Data Format**: Images are in JPG format with varying resolution.
- **Dataset Split**: 50 Artists, 8355 total artworks

In the interest of computation resources and time, the artworks of 15 artists for a total of 2788 images were used.

4.2 Results

Table 1 and Figure 3 summarize the quantitative evaluation of the four decoder architectures (Baseline, ResNet, Attention, UNet) over the five SW settings. Qualitative examples appear in Figures 8–7.

4.2.1 Evaluation

PSNR. Across all SW values, the UNet decoder consistently achieves the best results with the highest reconstruction fidelity. It peaks at 14.36 dB for SW = 3 and only gradually degrades to 12.21 dB at SW = 30. In contrast, the Baseline decoder hovers around 10 dB and drops by approximately 0.7 dB as style weight increases. The ResNet and Attention decoders perform comparably in the 11–12 dB range, with ResNet slightly outperforming Attention. It is also interesting to note that ResNet has very little degradation as style weight increases compared to the other decoders. (Fig. 3a)

SSIM. The UNet decoder also outperforms the rest in structural similarity (SSIM = 0.67 at SW = 3), retaining more content texture as SW increases (SSIM = 0.42 at SW = 30). Baseline and ResNet begin near SSIM = 0.26 but decline to 0.17 and 0.20, respectively, by SW = 30. Attention lags slightly behind ResNet at higher SWs, indicating its global mixing may be a trade off for detailed structure. It is important to note that the UNet significantly outperformed the other decoders in terms of SSIM. (Fig. 3b)

Perceptual Quality (LPIPS). Lower LPIPS scores indicate outputs closer in perceptual space to the target style–content mix. The UNet consistently achieves the lowest LPIPS (0.25 at SW = 3), rising to 0.51 at SW = 30. Baseline, ResNet, and Attention exhibit similar LPIPS trends (0.44 to 0.58, 0.42 to 0.57, and 0.44 to 0.57 respectively). Once again the UNet appears to greatly outperform the other decoders. (Fig. 3c)

Inference Time. The Baseline and UNet decoders are the fastest (approximately in the 1.7 ms per image range), reflecting their lightweight convolutional and skip-connection designs. Attention adds only a modest overhead (2.3 ms), whereas ResNet is the slowest at 3.2 ms due to repeated residual blocks. (Fig. 3d)

Qualitative Comparisons. Based on the quantitative metrics, the UNet appears to be the best decoder, but the best indicator of performance is the qualitative assessments. Figures 4 and 6 compare the decoders side by side on a Van Gogh and Monet style image respectively. The UNet in both cases appears to retain the content's structure the best, the Attention seems to have smoother textures, the ResNet is similar to the UNet with slightly smoother textures, and the Baseline looks to be the most stylized.

Figures 5 and 7 show how different style weights affect the resulting stylized outputs. The larger the style weight, the more similar it is to the style image, especially in terms of color and brushstrokes. An SW of 10 seems to be a good balance of content fidelity and style. Figure 8 shows how different style images greatly affect the colors, textures, and content fidelity, but they all still maintain enough content fidelity to appear to be stylized versions of the original.

4.2.2 Summary

Overall, the UNet decoder achieves the best trade-off between content preservation, style transfer quality, and runtime. The ResNet offers similar quality to UNet but has relatively high computational costs. The Attention decoder offers intermediate performance with intermediate computational costs, while the Baseline decoder, although fast, yields the lowest perceptual quality. Style weight provides a controllable knob: values around 7–10 offer balanced results, whereas very low or very high weights favor content or style dominance, respectively.

Table 1: Metrics by Decoder and Style Weight (SW)

(a) PSNR

SW	Baseline	ResNet	Attention	UNet
3	10.72	11.57	11.22	14.36
7	10.69	12.01	11.27	13.38
10	10.35	11.87	11.59	13.06
20	10.21	11.69	10.39	12.50
30	10.03	11.31	10.83	12.21

(b) SSIM

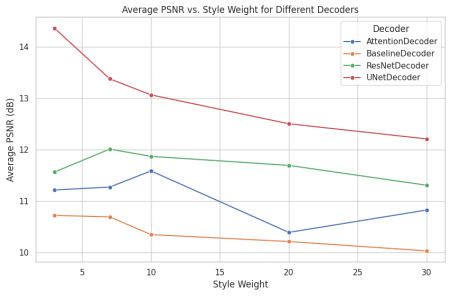
SW	Baseline	ResNet	Attention	UNet
3	0.259	0.259	0.265	0.669
7	0.232	0.253	0.247	0.584
10	0.215	0.251	0.245	0.542
20	0.189	0.217	0.180	0.413
30	0.172	0.196	0.191	0.420

(c) LPIPS

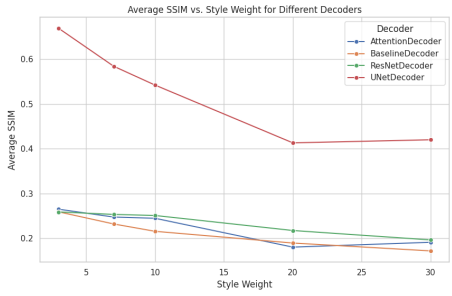
SW	Baseline	ResNet	Attention	UNet
3	0.442	0.419	0.444	0.249
7	0.490	0.470	0.495	0.364
10	0.523	0.500	0.513	0.414
20	0.551	0.547	0.568	0.492
30	0.581	0.569	0.572	0.514

(d) Inference Time[1](ms)

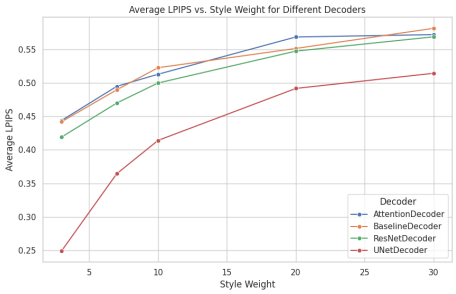
SW	Baseline	ResNet	Attention	UNet
3	2.336	3.195	2.287	1.782
7	1.709	3.185	2.287	1.806
10	1.700	3.185	2.294	1.806
20	1.697	3.186	2.282	1.828
30	1.702	3.187	2.284	1.795



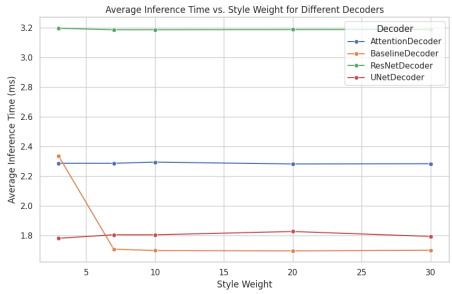
(a) PSNR vs. Style Weight



(b) SSIM vs. Style Weight



(c) LPIPS vs. Style Weight



(d) Inference Time vs. Style Weight

Figure 3: Comparison of metrics across decoders as a function of style weight.

5 Supplementary Material

Link to Codebase: `https://github.com/achyutpillai/AdaIN_Style_Transfer/tree/main`

Link to Presentation Video: `https://youtu.be/2yMui66qack`

[1]Using NVIDIA RTX A5000

5.1 Future Work

It would be beneficial to use the entire DIV2K and Best Artworks of All Time datasets during training as that would provide more diversity and prevent any potential overfitting. Some other techniques that could provide different results that should be explored is to use the output of the AdaIN layer for content loss like in the original AdaIN paper, trying different encoder architectures, and not fixing the encoder. These changes paired with the UNet Decoder could result in even better stylized outputs while still maintaining fast computation.

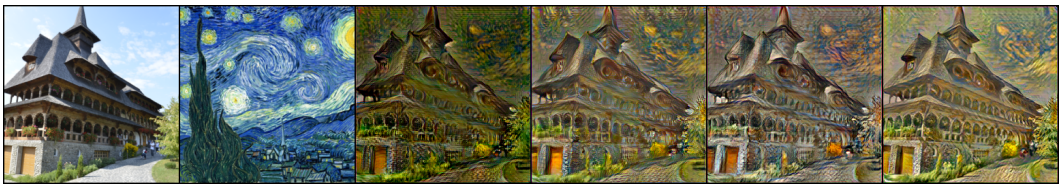


Figure 4: Comparison of different decoders on content image 0834 and Van Gogh style with SW = 10. (From left to right: Content, Style, Baseline, Resnet, Attention, UNet)

Figure 5: Comparison of different style weights on content image 0834 and Van Gogh style using UNet Decoder. (From left to right: Content, Style, 3, 7, 10, 20, 30)

Figure 6: Comparison of different decoders on content image 0820 and Monet style with SW = 10. From left to right: Content, Style, Baseline, Resnet, Attention, UNet.

Figure 7: Comparison of different style weights on content image 0820 and Monet style using UNet Decoder. (From left to right: Content, Style, 3, 7, 10, 20, 30)

Figure 8: Comparison of 7 different styles images on 7 different content images using the UNet Decoder with SW = 10.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 126–135, 2017.

[2] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[5] Ikarus777. Best artworks of all time, 2019.

[6] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[8] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[9] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.