Style-Conditioned Diffusion on WikiArt

Luna Tian, Mengli Yu

1 Introduction

The goal of this project is to build a generative model that can *synthesize new artworks in the style of a given artist*, while allowing the content of the image to vary freely. Concretely, given a style reference image from WikiArt, we want to generate new images that share a similar artistic style (color palette, brush strokes, composition statistics) but depict different scenes or objects. Studying this problem is interesting from an artistic perspective and a modeling perspective. We aim to understand whether recognizable classical art styles can be learned from only a small dataset, and how such style cues can be incorporated into a pre-trained diffusion model without expensive large-scale training.

We use the WikiArt dataset as our primary source of styles. The dataset provides metadata such as artist, genre, and style labels. These labels allow us to group paintings into clusters that roughly correspond to coherent visual styles, which is crucial for constructing training pairs and evaluating whether our model has learned style-consistent transformations. In this setting, the effective "target" variable is the *style-consistent image* produced by the diffusion model, conditioned on a style reference.

There are several challenges in analyzing this dataset and building a model on top of it. First, there are resource constraints: WikiArt is relatively small compared to web-scale datasets, and we train on a single NVIDIA RTX 3080 GPU, so we cannot fully fine-tune Stable Diffusion end-to-end. Second, the metadata is *noisy*: the pre-labeled genre and style fields do not always cleanly reflect human-perceived style, and there are many `Unknown Artist` and `Unknown Genre` entries. Third, it is non-trivial to encode style information in a way that is both learnable and controllable: we want a representation that is stable enough for training, but expressive enough to capture subtle stylistic differences between artists, genres and styles.

2 Methods

To achieve high-quality artistic style transfer, we fine-tune a Stable Diffusion model rather than training a GAN or other generative models from scratch; diffusion models currently represent the state of the art and reliably produce high-fidelity, diverse images. We build on the Latent Diffusion (Stable Diffusion) architecture and introduce a *style-aware cross-attention processor*. A CLIP-Vision encoder extracts a style embedding from the WikiArt reference image, which our `ImageProjModel` converts into a small set of learnable style tokens. These tokens are appended to the text embeddings and processed by a custom `StyleAttnProcessor` that replaces the default cross-attention module in the UNet. Each attention block applies separate key/value projections for text and style, routes the style attention output through a lightweight projection layer, and injects it into the UNet hidden states through a learnable gate. Compared to simply concatenating style tokens and relying on the default attention mechanism, our processor provides explicit, layer-wise control over style injection and produces more consistent and stable style transfer.

3 Related Work

Latent Diffusion Models (LDM) [1] compress images into a VAE latent space and learn a diffusion process over these latents, conditioned on text embeddings through cross-attention. This design dramatically reduces training cost while enabling high-resolution image synthesis. Our implementation directly builds on the Stable Diffusion variant of LDM: we reuse the VAE, CLIP text encoder, and UNet backbone, but replace the standard attention processors with our style-aware processor, making LDM the architectural foundation of our method.

Adaptive Instance Normalization (AdaIN) [2] shows that the mean and variance of deep features are sufficient to describe many artistic styles. By replacing the feature statistics of a content image with those of a style image, AdaIN achieves fast, feed-forward arbitrary style transfer. Our work is inspired by the idea that style can be encoded compactly, but instead of operating in a CNN feature space, we inject style into the cross-attention mechanism of a latent diffusion model. This allows us to leverage a much more expressive generative model while still keeping the style representation low-dimensional.

IP-Adapter [3] provides image-based conditioning for Stable Diffusion by replacing the attention processor and adding a dedicated image-attention branch inside each cross-attention block. CLIP-Vision embeddings are projected into image tokens, and the model computes text-attention and image-attention separately before fusing them. We reproduced the baseline and confirmed its strong global style transfer. However, its content preservation and fine-grained style controllability are limited. Our method builds on this idea, in addition to introducing style key/value projections, we add a style-specific output projection, and a learnable gate, enabling more stable and controllable style conditioning.

Another method performs loss-based style guidance at sampling time. Arbitrary Style Guidance for Enhanced Diffusion-based Text-to-Image Generation [4] does not modify the diffusion model; instead, it adds a style loss (e.g., VGG Gram loss [5]) during denoising and updates the current sample along the negative gradient of this loss. This can impose style without retraining, but it is slow and can be unstable because every sampling step requires backpropagation through the image decoder. Our approach moves style control into the model architecture and trains the style pathway once, making inference efficient and avoiding per-step optimization.

StyleDiffusion [6] performs style transfer by explicitly disentangling content and style in the latent space of a diffusion model. The method first applies a diffusion-based style removal module to obtain a content-preserving latent. It then defines a *style direction* in CLIP image-embedding space and fine-tunes the diffusion UNet so that the reverse denoising process reconstructs images consistent with this style direction. In contrast, our method does not rely on latent disentanglement or per-style UNet optimization. Instead, we keep the entire pre-trained Stable Diffusion UNet fully frozen and inject style information directly into its cross-attention layers through a lightweight style adapter. This design modulates attention keys/values with CLIP-Vision style tokens, while maintaining the efficiency of the training process.

4 Related Implementations (Kaggle)

In the Kaggle community, several notebooks explore style-related processing on WikiArt. Shoab Ahamed's "Arbitrary Style Transfer"[1] implements an upgraded VGG19-based neural style transfer pipeline. This method computes Gram-matrix style losses and content loss between feature maps, updating the image through gradient descent until the losses converge. In contrast, our approach conditions a latent diffusion model rather than directly optimizing pixels. Style information is injected through learnable style tokens and a custom cross-attention processor inside the UNet, enabling multi-layer, feature-level modulation throughout the generative process.

Another relevant notebook is Abdelnour Fellah's "Art images colorization from custom color palette"[2], which treats style primarily as a color palette. The notebook extracts palettes by applying KMeans clustering to image pixels (with the number of clusters representing the number of dominant colors), and then trains a U-Net–based colorization model with cross-attention to recolor grayscale images according to a provided palette. This approach captures low-level color statistics but is not designed to model high-level artistic structure such as brushwork, texture, or composition. Our method, by contrast, conditions a full latent diffusion model using CLIP-Vision embeddings and layer-wise attention modulation, allowing it to capture richer stylistic attributes beyond color distribution.

5 Data Analysis

We first inspected the WikiArt metadata to understand label quality. There are 41,914 rows with `Unknown Artist` and 16,452 rows with `Unknown Genre`, which we excluded from our style modeling pipeline to avoid noisy

[1] `https://www.kaggle.com/code/shoabahamed/arbitarystyletransfer/notebook`

[2] `https://www.kaggle.com/code/fellahabdelnour13/art-images-colorization-from-custom-color-palette`

supervision. For the remaining data, we grouped artworks by (*artist*, *genre*, *style*) and computed the number of images per group (Fig. 1). The top groups have several hundred images each; for example, the largest group $(4, 4, 12)$ contains 752 paintings, followed by $(14, 4, 24)$ with 716 images, and $(7, 3, 23)$ with 527 images. These groups form our initial candidates for style-consistent clusters.

artist	genre	style	count
4	4	12	752
14	4	24	716
7	3	23	527
17	6	12	464
10	4	21	436
22	8	21	398
42	4	12	388
22	8	20	369
2	4	12	340
11	6	21	335
25	7	17	333
8	6	21	332
48	4	21	328
5	2	12	315
74	4	21	298
104	6	9	282
12	7	15	281
17	4	12	252
22	4	20	252
4	1	12	252

Figure 1: top 20 groups(Artist,Genre,Style) Counts

To verify that these metadata-defined groups actually correspond to visually coherent styles, we sampled 5 images from each of the top 5 groups, shuffled them (25 images total), and asked a human volunteer to assign each image to one of the five groups. The volunteer achieved 23/25 correct assignments, suggesting that these clusters have clear stylistic differences that humans can reliably perceive. This informal user study supports the idea that (*artist*, *genre*, *style*) combinations provide a reasonably good proxy for style labels, at least for the larger groups.

Next, we extracted style embeddings for each image using DINOv2[7]. Each image is mapped to a 768-dimensional feature vector, which we treat as a style descriptor. For the 7 largest groups, we ran PCA on these DINO features and plotted the first two principal components for visualization (Fig. 2). Most scatter plots for groups such as `(4,4,12)`, `(7,3,23)`, `(10,4,21)`, `(14,4,24)`, `(17,6,12)`, `(22,8,20)` and `(22,8,21)` show dense, compact clouds rather than random scatter, indicating that within-group style embeddings occupy a coherent region in feature space.

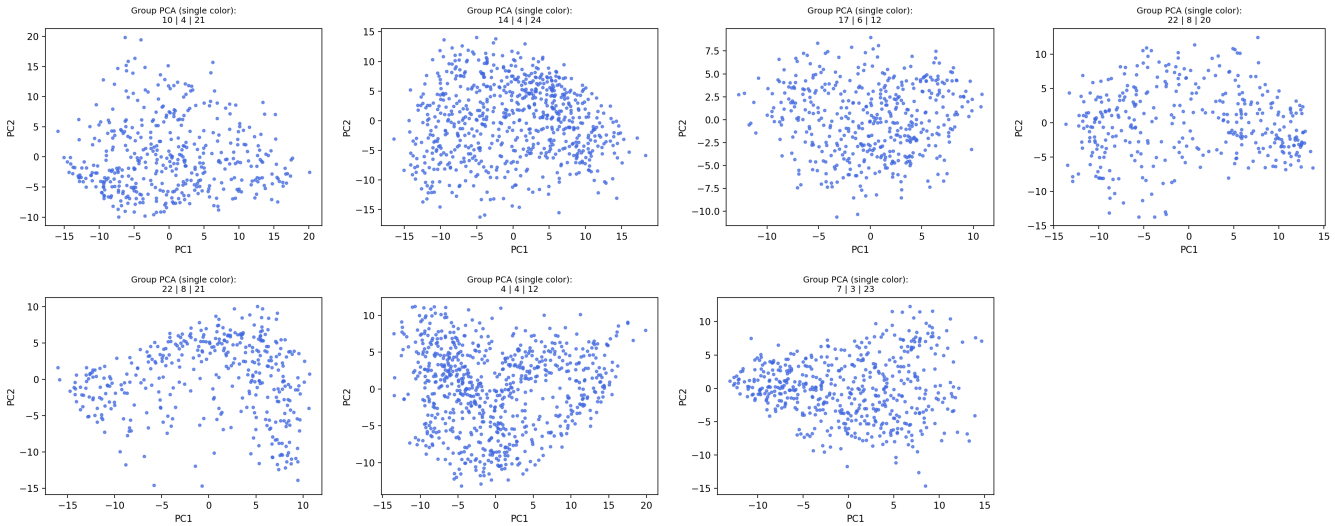

Figure 2: PC1-PC2 plot for 7 largest groups

While PCA alone cannot determine stylistic separability, these visualizations are consistent with our manual evaluation: the metadata-defined groups tend to form reasonably tight clusters, and different groups often exhibit

different cluster shapes or orientations. Together with the human classification results, these findings suggest that DINOv2 features are sensitive to stylistic attributes, not only object identity, and that using DINOv2 as a style-sensitive descriptor is appropriate for dataset analysis and preprocessing.

To construct training pairs and test reference images, we removed groups with very few samples (e.g., fewer than 7 images), since these small clusters offer little statistical support and may lead to overfitting or unstable training. For each remaining group, we reserved 5% of images as a style reference test set and used the rest for training pairs. In order to encourage content diversity, we formed pairs of images within the same (*artist*, *genre*, *style*) group and computed their CLIP-Vision cosine similarity. Pairs whose similarity exceeded a threshold 0.8 were excluded, because they likely depict near-duplicate content rather than diverse scenes with a shared style. Formally, for embeddings e_i, e_j, we keep the pair only if

$$\cos(e_i, e_j) = \frac{e_i^\top e_j}{\|e_i\| \, \|e_j\|} < 0.8.$$

This filtering step helps the model avoid overfitting to trivial style copies of nearly identical compositions. Overall, this preprocessing and analysis pipeline gave us a set of reasonably clean, style-consistent groups and diverse within-group pairs that are well-suited for learning style conditioning.

6 Detailed Proposed Method

Our model builds on the pre-trained Stable Diffusion (v1.5) and augments only its cross-attention mechanism with a learnable style pathway. All backbone components—the VAE, UNet weights, CLIP text encoder, and CLIP-Vision encoder—remain frozen. Training therefore focuses exclusively on two lightweight modules: a small projector that maps CLIP-Vision embeddings into style tokens, and a custom attention processor that injects these tokens into the UNet. During training, a target image is encoded into a latent z via the VAE, Gaussian noise ϵ and a timestep t are sampled, and the noisy latent $\tilde{z}_t = \text{add\_noise}(z, \epsilon, t)$ is fed into the UNet. The UNet learns to predict the noise $\hat{\epsilon}_\theta(\tilde{z}_t, t, c)$, where c is the conditioning (empty text tokens plus our style tokens), using the standard MSE objective

$$\mathcal{L}_{\text{noise}} = \mathbb{E}_{z,\epsilon,t}\left[\|\epsilon - \hat{\epsilon}_\theta(\tilde{z}_t, t, c)\|_2^2\right].$$

To encode style, we use CLIP-Vision to extract a global embedding from the style reference image. This embedding is first normalized and then passed into an `ImageProjModel`, which consists of a LayerNorm followed by a two-layer MLP with a GELU activation. Given an input embedding $e \in \mathbb{R}^{768}$, the projector outputs N style tokens $T_{\text{style}} \in \mathbb{R}^{N \times D}$, where N (e.g., 4) is the number of style tokens, D matches the UNET cross-attention dimension. This provides sufficient capacity to transform the CLIP embedding into an attention-compatible token representation.

The core idea lies in the `StyleAttnProcessor`, which replaces the default attention processor for all cross-attention layers in the UNet. For each attention block, we split the encoder hidden states into text tokens and style tokens:

$$\text{text\_context} = c_{1:L_{\text{text}}}, \quad \text{style\_context} = c_{L_{\text{text}}+1:L_{\text{text}}+N}.$$

Queries Q are always computed from the UNet hidden states via the original projection. For the text pathway, keys and values are computed using the pre-trained projections:

$$K_{\text{text}} = \text{text\_context} \cdot W_k^{\text{text}}, \quad V_{\text{text}} = \text{text\_context} \cdot W_v^{\text{text}}.$$

For the style pathway, we introduce new learnable projections

$$K_{\text{style}} = \text{style\_context} \cdot W_k^{\text{style}}, \quad V_{\text{style}} = \text{style\_context} \cdot W_v^{\text{style}},$$

where $W_k^{\text{style}}, W_v^{\text{style}} \in \mathbb{R}^{D \times C}$ are independent from the text projections and trained from scratch.

We then compute two scaled dot-product attentions, one for text and one for style, using the same queries:

$$A_{\text{text}} = \text{softmax}\left(\frac{QK_{\text{text}}^\top}{\sqrt{d}}\right)V_{\text{text}}, \quad A_{\text{style}} = \text{softmax}\left(\frac{QK_{\text{style}}^\top}{\sqrt{d}}\right)V_{\text{style}}.$$

The text attention is passed through the original output projection W_o, while the style attention is passed through a style-specific output projection W_o^{style}:

$$O_{\text{text}} = A_{\text{text}} W_o, \quad O_{\text{style}} = A_{\text{style}} W_o^{\text{style}}$$

Finally, we combine them using a learnable scalar gate g and a global scale factor α:

$$O = O_{\text{text}} + \alpha g \, O_{\text{style}}.$$

The gate is initialized to a small value (0.1), and the style output projection is zero-initialized, ensuring that the style pathway has no effect at the beginning of training. Since this projection is a single linear layer, zero-initialization does not hinder learning, but it guarantees that the pretrained UNet is not disturbed during the initial steps. As training progresses, the gate and the style projections gradually learn how strongly each attention block should respond to style.

Although we experimented with adding the style loss into the training objective, we ultimately used it only during evaluation, since naively combining it with noise prediction loss made optimization unstable under our limited compute. However, the evaluation module is still important: every few steps we generate a stylized image for a held-out style reference, compute the style loss between the generated image and reference, and log the curve to monitor whether style consistency improves over training.

7 Analysis

The chosen architecture and features are well-suited to the WikiArt setting for several reasons. First, by building on a pre-trained LDM, we inherit a strong prior over natural images and only need to learn a relatively small number of additional parameters (the style projector and attention processors). This is crucial when training on a modest dataset such as WikiArt with limited GPU resources. The model does not need to learn how to draw from scratch; instead, it learns how to route style information through the existing cross-attention mechanism.

Second, CLIP-Vision embeddings serve as effective style descriptors. They capture high-level visual semantics and artistic attributes learned from large-scale data, which can generalize beyond the specific images in WikiArt. Projecting these embeddings into a small set of style tokens allows us to reuse the same attention infrastructure that Stable Diffusion uses for text, making style conditioning naturally compatible with the rest of the architecture.

Third, injecting style at the attention level rather than at sampling time (as in Arbitrary Style Guidance) brings both controllability and efficiency. Each cross-attention block can independently learn how much style it needs via the gate parameter, and we can visualize and interpret the effect by setting processor's scale to zero to recover the baseline model. Because style is baked into the UNet forward pass, inference remains as fast as standard Stable Diffusion sampling and does not require per-step gradient updates.

This method is generalizable to other artistic datasets. Any collection where style is important, such as art, manga, or stylized game assets, could benefit from the same approach: cluster images into style groups, build styled pairs, and train a style-attention pathway on top of a pre-trained diffusion model.

8 Experimental Setup

We started from the full WikiArt training split and filtered out all entries with `Unknown Artist` or `Unknown Genre`. We then grouped the remaining paintings by (*artist*, *genre*, *style*) and kept only groups with at least 7 images. For each group, we constructed pairs of images for training by sampling two distinct images within the group and discarding pairs whose CLIP-Vision cosine similarity exceeded 0.8, in order to avoid near-duplicate content. For each surviving group, we held out 5% of images as a style reference test set and used the rest to form training pairs. Training is performed on a single NVIDIA RTX 3080 GPU using the Stable Diffusion v1.5 [3]. All base components (VAE, text encoder, UNet, CLIP-Vision) are frozen; only the `ImageProjModel` and `StyleAttnProcessor` parameters are updated. We adopt AdamW as the optimizer and trained for four epochs over the constructed target–style pairs.

[3] `https://huggingface.co/stable-diffusion-v1-5/stable-diffusion-v1-5`

In order to monitor training, we periodically run an evaluation loop on a fixed mini-batch every 250 steps. For one style reference image, we sample a stylized image using a shortened sampling schedule (e.g., 30 DDPM steps) and compute the style loss between the generated image and the style reference. This produces the *Step vs. Style Gram Loss* line chart shown in our results. In addition, after each epoch we perform a qualitative A/B test: using a fixed random seed and style reference images, we generate images with and without the style processor (by swapping our processor with the default attention processor) to ensure the effectiveness of our setup.

Several implementation details were important for stability. We found that a sigmoid-gated of style projection was unstable, so we replaced it with a linear gate initialized at 0.1, which produced smoother training. Finally, we observed that using the target image as its own style reference for about 80% of steps (self-style pairing) improved content preservation and style consistency.

At inference time, we keep the text prompt empty so that the output is controlled entirely by the style image. We convert the CLIP-Vision embedding into style tokens and append them to the empty-prompt text tokens before sending them into the UNet's cross-attention layers. For classifier-free guidance, we also prepare an unconditional branch where the same text tokens are used but the style tokens are replaced with zeros. The UNet predicts noise for both branches, and guidance combines them in the usual way. We introduce a style scale parameter to control the strength of stylization, and we fix the random seed so that the content structure remains identical while the style changes.

9 Results

To quantify early training behavior, we record the VGG Gram style loss across the first epoch every 250 steps. As shown in Fig. 3, the curve displays a clear downward trend, decreasing from nearly 30 to the low 20s by the end of the epoch. Because the later curves show little additional structure, we only report the first-epoch plot here and instead rely on qualitative evaluations to assess longer-term learning.

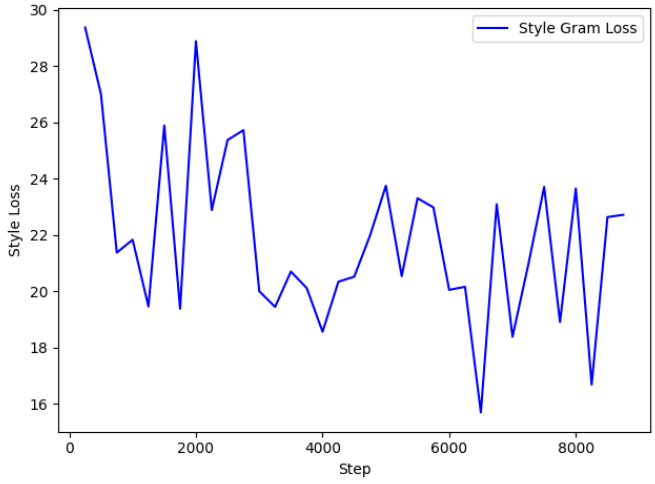

Figure 3: style loss line chart (1st epoch)

Beyond this initial quantitative signal, our qualitative A/B tests after each epoch provide much stronger evidence of progressive learning. Using a fixed seed and fixed style scale (0.5), we regenerate the same content image with four different style references. With each epoch, the model reproduces increasingly faithful stylistic cues: global palette shifts become more consistent, brush-like textures emerge more clearly, and fine structures such as color boundaries or shadow shapes align more closely with the reference style. This effect is visible in Fig. 4, where the leftmost column shows a fixed content image (top) and four fixed style references (below), and the right four columns display the corresponding stylized outputs at epochs 1–4. Stylization begins coarse but becomes significantly more coherent after 3–4 epochs, with clearer textures and more consistent palette transfer.

Figure 4: Visualization of stylization across training, top: content image (seed 1234)

After completing all four epochs of training, we also evaluate controllability by sweeping the style scale from 0 to 1. As shown in Fig. 5, the outputs transition smoothly: scale 0 resembles baseline Stable Diffusion, while scale 1 strongly adopts the reference style including color blocks, pigment patterns, and painterly textures, while still preserves the geometry of image generated by random seed. This also showed that learned attention modulation integrates cleanly with classifier-free guidance without collapsing the content.

Figure 5: Output images adjusted scale value from 0 to 1.

Finally, we compare our method against the IP-Adapter method (Fig. 6), the image-guided adapter for Stable Diffusion. On ten randomly sampled WikiArt styles, the IP-Adapter baseline achieves impressive global style transfer, including color palette changes and high-level stylistic cues. However, we observed that IP-Adapter often

overwrites content structure, and provides limited control over strength of stylistic attributes. In contrast, our method of introducing additional style projection, and a learnable coefficient preserves content significantly better while still capturing high-level style cues. Remarkably, our model achieves this using only a lightweight adapter trained for a few epochs. These results suggest that explicitly injecting style features inside cross-attention is an efficient and scalable approach to style-aware latent diffusion.

Looking forward, we also evaluated the model's behavior on styles unseen during training. While the model generalizes reasonably well to styles belonging to the same broad family (e.g., Impressionism variants), performance degrades for rare styles with highly distinctive textures. This suggests that improving the style embedding or using larger dataset could meaningfully enhance generalization capability. With stronger style features and more diverse clusters, we expect better robustness on long-tail artistic styles. Due to limited time and GPU resources in this project, we were unable to explore higher-capacity style encoders or larger-scale clustering, but we believe these directions would further strengthen model performance.

10 Conclusion

In this project, we demonstrate that a lightweight style adapter inserted into Stable Diffusion's cross-attention layers can effectively transfer artistic styles from WikiArt, while preserving Stable Diffusion's ability to generate diverse content. Despite training for only four epochs on a modest GPU, the resulting model demonstrates great controllability, coherent stylization, and competitive quality. Our qualitative and quantitative evaluations suggest that directly modulating key–value projections within cross-attention is a simple yet powerful mechanism for style conditioning.

For practitioners interested in deploying style-aware generation, our method offers an attractive balance of efficiency, interpretability, and control. Because the base diffusion model remains fully frozen, training is low-cost and can be repeated for new style clusters with minimal compute. The controllable style scale also makes the method compatible with creative pipelines that require fine-grained adjustment of stylistic strength. If computation permits and large-scale training is not required, we would recommend this adapter-based approach over requiring full UNet fine-tuning.

With a larger and more diverse dataset, the model's generalization ability would naturally improve. For deployment, two strategies are most practical. Companies can adopt more powerful or domain-specific style embedding methods to improve robustness, and they can maintain multiple lightweight per-style adapters, each trained on a distinct style family. This modular design enables flexible stylistic control without retraining the diffusion model. Overall, our results show that adapter-based style conditioning is efficient, scalable, and well-suited for real-world generative applications.

References

[1] Rombach, Robin, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-resolution image synthesis with latent diffusion models." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10684-10695. 2022.

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

[3] Ye, Hu, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. "Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models." arXiv preprint arXiv:2308.06721 (2023).

[4] Pan, Zhihong, Xin Zhou, and Hao Tian. "Arbitrary style guidance for enhanced diffusion-based text-to-image generation." In Proceedings of the IEEE/CVF winter conference on applications of computer vision, pp. 4461-4471. 2023.

[5] Gatys, Leon A., 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, pp. 2414-2423. 2016.

[6] Wang, Zhizhong, Lei Zhao, and Wei Xing. "Stylediffusion: Controllable disentangled style transfer via diffusion models." In Proceedings of the IEEE/CVF international conference on computer vision, pp. 7677-7689. 2023.

[7] Oquab, Maxime, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, et al. "DINOv2: Learning Robust Visual Features without Supervision." arXiv preprint arXiv:2304.07193, 2023.

Appendix

Code repository: `https://github.com/Lunasocute/style-attention-diffusion/`

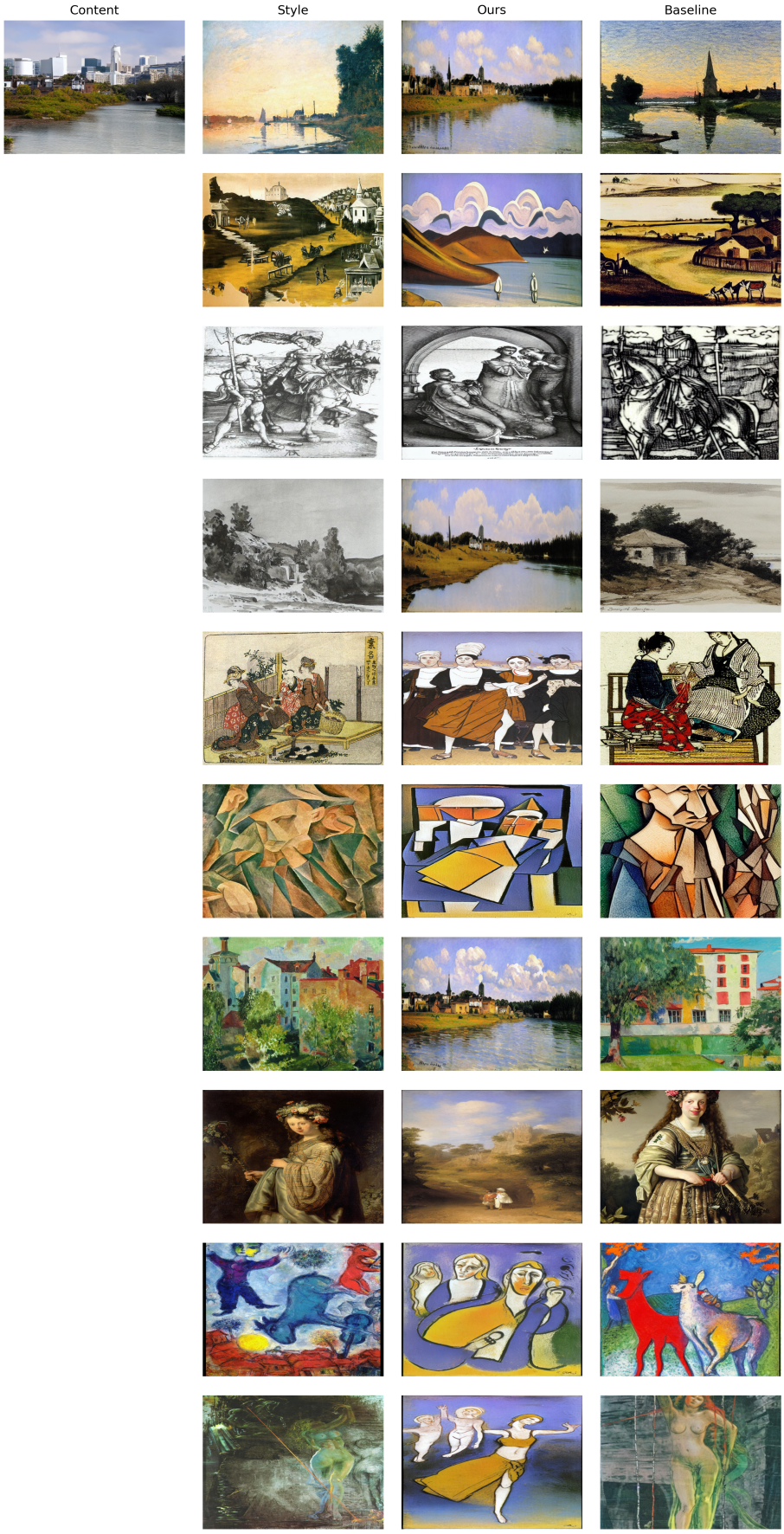

Figure 6: Ours generated images vs. IP-Adapter's, fixing content & style reference

Statement of Contributions

All group members contributed equally to this project.
Luna: Led the overall project direction, conducted literature review and research-paper analysis, designed the diffusion-based style conditioning method, implemented the core training pipeline (including UNet integration, style encoder, and attention modifications), ran experiments, and wrote the main sections of the report.
Mengli: Contributed to dataset processing, organized experiment logs, coordinated qualitative user inspection, and provided feedback for model evaluation and report polishing.