Puff-Net: Efficient Style Transfer with Pure Content and Style Feature Fusion Network

Sizhe Zheng, Pan Gao*, Peng Zhou, Jie Qin*
College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics
{162100101, pan.gao, zhoupeng23, jie.qin}@nuaa.edu.cn

Abstract

*Style transfer aims to render an image with the artistic features of a style image, while maintaining the original structure. Various methods have been put forward for this task, but some challenges still exist. For instance, it is difficult for CNN-based methods to handle global information and long-range dependencies between input images, for which transformer-based methods have been proposed. Although transformers can better model the relationship between content and style images, they require high-cost hardware and time-consuming inference. To address these issues, we design a novel transformer model that includes only the encoder, thus significantly reducing the computational cost. In addition, we also find that existing style transfer methods may lead to images under-stylied or missing content. In order to achieve better stylization, we design a content feature extractor and a style feature extractor, based on which pure content and style images can be fed to the transformer. Finally, we propose a novel network termed Puff-Net, i.e., **pu**re content and style **f**eature **f**usion **net**work. Through qualitative and quantitative experiments, we demonstrate the advantages of our model compared to state-of-the-art ones in the literature. The code is available at https://github.com/ZszYmy9/Puff-Net.*

1. Introduction

As personalized expression gains popularity, people increasingly seek to transform images into new artistic styles. Imagine turning a plain landscape photo into an oil painting or a snapshot into an Impressionist-inspired image. This technology, known as Style Transfer in computer vision, offers possibilities for artistic expression. It captures and blends the essence of artistic styles into different images, creating pieces that merge original content with artistic styles.

Early style transfer methods primarily relied on optimization algorithms which minimize the differences between the input image and the reference image. However, the high computational complexity of these methods greatly limited their practical applications. With technological advancements, image style transfer techniques based on direct inference have made significant progress. The method introduced by Gatys *et al.* [11], which employs convolutional neural networks (CNN), extracts features of content and style from different layers of a pre-trained CNN model. This approach has significantly reduced computational complexity and spurred a wave of related research, including developments like AdaIN [12], Avatar [24], SANet [21], and MAST [5]. Despite the achievements of these CNN-based inference methods for image style transfer, they still face limitations. They depend on convolution operations to capture image features, and their performance is limited when the network layers are insufficient to capture global information. On the other hand, as the number of layers increases, the content details

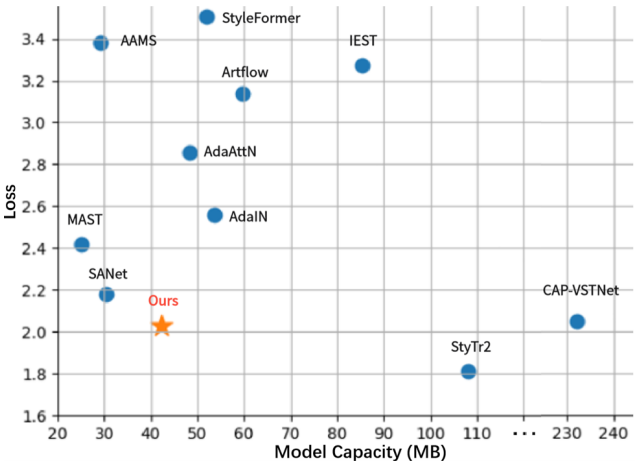

Figure 1. Comparison of different models based on loss and capacity, with the loss being a combination of 60% content loss and 40% style loss. Our model shows a favorable balance between capacity and loss. Details can be found in the Method and Experiments sections.

*Corresponding authors.

Figure 2. Some results of our Puff-Net. Our method achieves a better balance between maintaining stylized effects and reducing computational costs. The main body and background of the content image can be stylized more reasonably based on the style image.

of the synthesized image may be lost, which in turn affects the overall quality of the stylized image. Therefore, effectively transferring style while maintaining content integrity remains a challenge in the field of image style transfer.

Vision transformer (ViT) [7] offers a novel approach for utilizing transformer models [25] in visual tasks. By dividing input images into a series of small patches and rearranging them to form embedding vectors, this method has been shown to surpass the performance of traditional CNNs in several visual tasks. For example, a transformer-based model [8] has achieved significant breakthroughs in style transfer tasks. The success of transformers in handling image data can be attributed to their attention mechanism, which captures the global context of images. Additionally, this mechanism helps the model better understand the relationship between content and style in images, enhancing stylization efforts. However, the model capacity of transformer is large, with high hardware requirements and slow training speed. In order to tackle these difficulties, we design a transformer that includes only the encoder. We modify the encoder structure of the transformer so that we can obtain stylized output sequences of image patches through the encoder alone. The modified transformer is of lower complexity and has a significantly improved inference speed.

By analyzing the generated results, we found that the generated images may have significant differences from the content images. The feature distribution of the generated images is not visually reasonable where the content features of some style images also appear. Our objective is to eliminate the style attributes from the content images, preserving only their content structure. Simultaneously, we hope that style images can focus less on content details and allow their style features to participate in the stylization process. Therefore, we preprocess the content images and style images before feeding them to the transformer for stylization. Accordingly, we develop two distinct feature extractors: one to isolate content features and the other to isolate style features from the input images.

In summary, we introduce a novel framework for efficient style transfer, namely **pu**re content and style **f**eature **f**usion **net**work (Puff-Net), which incorporates two feature extractors and a transformer equipped solely with an encoder. Our major contributions are summarized as follows:

- We enhance the structure of the encoder in the vanilla transformer so that style transfer can be performed efficiently through only the encoder, reducing computational overhead.
- We design two feature extractors that preprocess the input to obtain pure content images and pure style images, and consequently, achieving superior stylized results.
- Even with a notable reduction in model capacity, our model continues to deliver competitive performance over existing counterparts.

Figure 1 illustrates the comparison of our model with state-of-the-art models in terms of model capacity and overall loss, and Figure 2 provides visual stylized results of our Puff-Net. It is evident that the proposed Puff-Net achieves a balance between style transfer effectiveness and model efficiency.

2. Related Work

2.1. Style Transfer

Image style transfer has made significant progress. Gatys *et al.* [11] discover that when feeding an input image into a pre-trained CNN (VGG19), one can capture the content and style information of the image and integrate both information by using an optimization-based method. Then, relevant ensuing studies started emerging. AdaIN [12] aligns the mean and variance of content image features with the mean and variance of style images to implement style transfer. SANet [21] integrates local style patterns efficiently and flexibly based on the semantic spatial distribution of content images. MAST [5] enhances feature representations of content images and style images through position-wise self-attention, calculates their similarity, and rearranges the distribution of these representations. ArtFlow [1] proposes a model consisting of reversible neural flows and an unbiased feature transfer module, which can prevent content leak during universal style transfer. AdaAttN [19] designs a novel attention and normalization module and inserts it into the traditional encoder-decoder pipeline. IEST [4] utilizes external information and employs contrastive learning for style transfer. StyleFormer [27] incorporates transformer components into the traditional CNN workflow. StyTr^2 [8] proposes a model which achieves the style trasnsfer only through the vanilla transformer. CAP-VSTNet [26] adopts a reversible framework to protect content images to avoid ar-

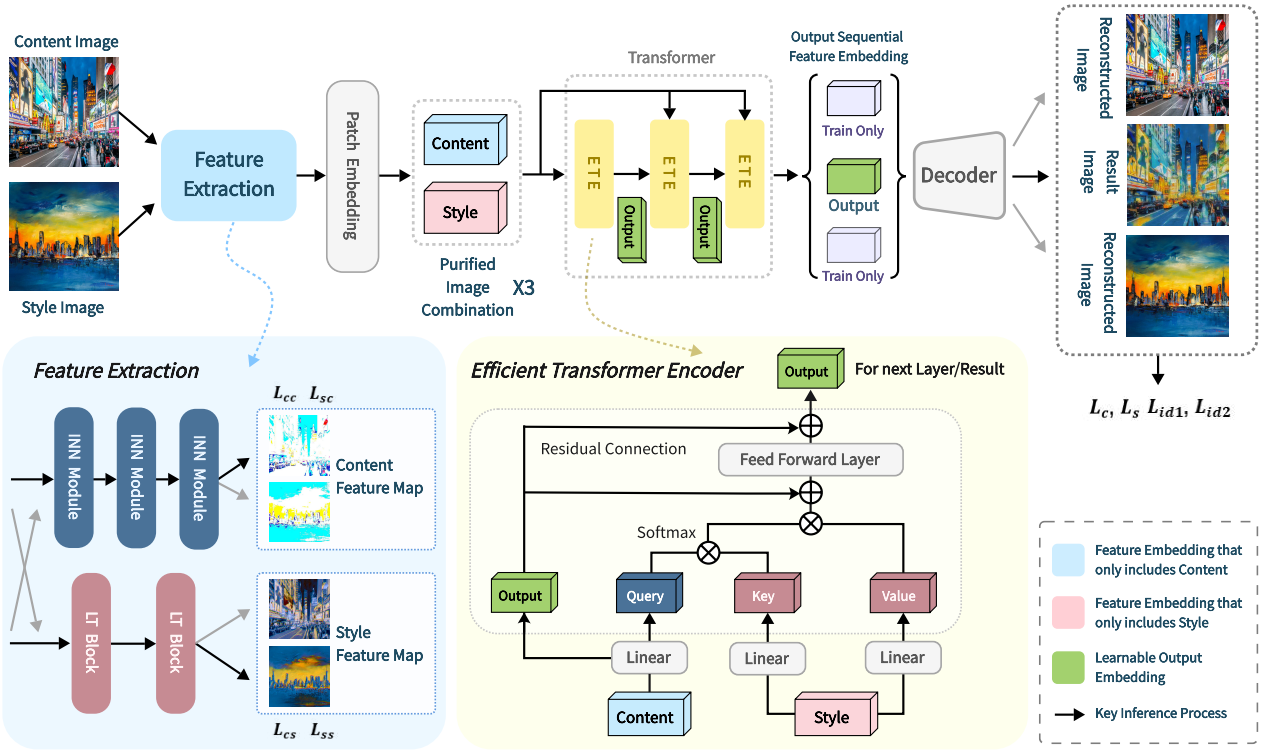

Figure 3. Schematic illustration of the Puff-Net architecture. The network begins by extracting content and style features from the input images. These features are then divided into patches and encoded into patch sequences through a linear projection. After feeding the features into the transformer for stylization, we can finally obtain the result image through the decoder. Additionally, the model leverages a reconstruction loss function during training to enhance its ability to reconstruct content and style features.

tifacts, and achieves style transfer through an unbiased linear transform module. However, it places more emphasis on retaining content, which leads to the image under-stylized. CLIPstyler [15] achieves style transfer through text injection. We aim to provide a practical solution for style transfer. Puff-Net balances the output quality and efficiency, while CNN-based methods prioritize speed. StyTr^2 focuses on higher-quality outputs at the cost of efficiency, and CAP-VSTNet retains more original content details compromising quality. More recently, diffusion-based models [9] prioritize creativity over efficiency.

2.2. Transformer

The vanilla transformer [25] is designed to tackle tasks in the field of natural language processing (NLP). The unique self-attention mechanism can effectively model the relationships between tokens. In order to apply the transformer to the field of computer vision (CV), lots of related research has been carried out. The proposal of the ViT [7] made a groundbreaking contribution to the application of transformers in CV. It segments images into patches and arranges them into embeddings, which are fed to ViT for processing. Since the advent of ViT, variants of transformers have been proposed to deal with multiple visual tasks. For example, DETR [3] and YOLOS [10] for object detection, SegFormer [29] and SETR [31] for semantic segmentation, and CTrans [16] and Swin-Transformer [20] for image classification. Transformers are also proven very effective in the area of multi-modal fusion, *e.g.*, ViLT [13]. StyTr^2 [8] adopts only the vanilla transformer for style transfer for the first time, and the improvement is very significant. Compared with the CNN, transformers can capture long-range dependencies of input images by using attention mechanisms. In this paper, we also leverage the strong global modeling capability of transformers for style transfer. However, different from prior models, we utilize the transformer encoder to associate the disentangled content feature and disentangled style feature, resulting in better stylizing results with a smaller model scale.

3. Method

In this section, we will introduce the workflow of the proposed Puff-Net. We set the dimensions of the input and output to be $H \times W \times 3$. To make use of the transformer encoder, we treat style transfer as a sequential patch generation task. We split both content and style images into patches and use a linear projection layer to project input

patches into a sequential feature embedding ε in the shape of $L \times C$, where $L = \frac{H \times W}{m \times m}$ is the length of ε, $m = 8$ is the patch size, and C is the dimension of ε. Figure 3 illustrates the overall architecture of the proposed Puff-Net.

3.1. Efficient Transformer Encoder

StyTr^2 [8] employs the transformer to implement the task of style transfer. However, their proposed model necessitates a large number of computational resources. In addition to the encoder, a transformer decoder is adopted to translate the encoded content sequence according to the encoded style sequence. The output sequential feature embedding ε_o can only be obtained through a complete transformer. This is very cumbersome, and thus we hope to obtain ε_o directly through the encoder alone. With this objective in mind, we modify the transformer encoder as follows. We append a learnable sequence feature embedding ε_o whose shape is the same as ε_c, and we process it in the encoder based on the purified content and style images. The FFN layer of the transformer consumes much of the computation, but its role in capturing context features is not significant. To this end, we make it process and transmit the information of ε_o. Through the encoder, we can obtain the required sequence feature embedding ε_o. After applying the decoder, we can obtain the result image. The overview of the Efficient Transformer Encoder (ETE) is shown in Figure 3.

Considering that the output image should be close to the content image, we initialize the learnable ε_o based on ε_c. We intend to connect ε_c and ε_s and feed them to the encoder. Each layer of the encoder consists of a multi-head self-attention module (MSA) and a feed-forward network (FFN). However, its computational complexity is $O((2L)^2 \times C + 2L \times C^2)$. Therefore, we redesign the transformer model. ε_c is encoded into a query (Q) and ε_s is encoded into a key (K) and a value (V). The computational complexity is $O(L^2 \times C + L \times C^2)$. We can also better build the connection between the content and style images in this way. Moreover, when we use the attention mechanism, the positional encoding should be included in the input. Here, we use Content-Aware Positional Encoding (CAPE) in [8], which takes image semantics into account when implementing positional encoding. We only calculate CAPE for the content image as follows:

$$Q = (\varepsilon_c + \mathcal{P}_{CA})W_q, K = \varepsilon_s W_k, V = \varepsilon_s W_v \quad (1)$$

where $W_q, W_k, W_v \in R^{C \times d_{head}}$. The multi-head attention is then calculated by

$$\begin{aligned} \mathcal{F}_{MSA} &= Concat(head_1, \ldots\ldots, head_n)W_o; \\ head_i &= Attention_i(Q, K, V) \end{aligned} \quad (2)$$

where $W_o \in R^{C \times C}$, n is the number of attention heads, and $d_{head} = \frac{C}{N}$. In each encoder layer, the output Y is calculated as

$$\begin{aligned} Y' &= \mathcal{F}_{MSA}(Q, K, V) + \varepsilon_o; \\ Y &= \mathcal{F}_{FFN}(Y') + Y' \end{aligned} \quad (3)$$

where $\mathcal{F}_{FFN}(Y') = max(0, Y'W_1 + b_1)W_2 + b_2$. Layernorm [2] is applied in each layer.

Through the encoder, we can obtain the output sequence in the form of $\frac{H \times W}{m \times m} \times C$. To obtain the final result, we employ a three-layer CNN decoder proposed in [31]. In each layer, we expand the scale by executing a series of operations including 3 × 3 Conv + ReLU + 2 × Upsample. Finally, we can save the resultant image in the form of H × W × 3.

3.2. Feature Extraction

Since we aim to transfer the style of the content image, we hope that the underlying model can change the color and other characteristics. Meanwhile, we ought to avoid damaging content images as much as possible. Therefore, we try to increase the proportion of the content loss when designing the loss function. Doing so may result in a small difference between the results and content images, yet still including style features such as color. In order to ensure that the content is not missing while the style of the result image is closer to that of the style image, we preprocess the content and style images to extract their distinct features. To extract different kinds of features from the content and style images, we assume the two images contain information from two modalities, from which we can capture their unique features. Therefore, we handle the two types of images separately to obtain pure content and style images. Towards this end, we employ different feature extractors for content images and style images, respectively.

To process content images with minimal loss of detail, we select the INN module [6] as the backbone for our content extractor, aiming for utmost content preservation. It can better preserve the content by making its input and output features mutually generated. Therefore, we adopt the INN block [6, 32] with affine coupling layers. In each invertible layer, the transformation is written as follows:

$$\begin{aligned} Y_{k+1}[c+1:C] &= Y_k[c+1:C] + \phi_1(Y_k[1:c]) \\ Y_{k+1}[1:c] &= Y_k[1:c] \odot \exp(\phi_2(Y_{k+1}[c+1:C])) \\ &\quad + \phi_3(Y_{k+1}[c+1:C]) \\ Y_{k+1} &= Concat(Y_{k+1}[1:c], Y_{k+1}[c+1:C]) \end{aligned} \quad (4)$$

where $\odot$ is the Hadamard product, Y_k (k=1,2,$\cdots$) is the output of the k-th layer, [1:c] represents the 1st to the c-th channels, and ϕ_i (i=1,2,3) are the arbitrary mapping functions. To balance the feature extraction ability and computational complexity, we employ the bottleneck residual block (BRB) in MobileNetV2 [23].

For style images, our focus is on capturing the general style, rather than the local details. It requires the extractor to grasp the global information and long-distance dependency features well. Meanwhile, considering the computational complexity of the model, we choose the LT block [28] as the basic unit of the style extractor. It flattens the bottleneck of transformer blocks by flattening the feed-forward network, which saves substantial computation. Please refer to the supplementary material for the network details of the two extractors.

3.3. Loss Function

The generated image requires a fusion of content and style. Therefore, we need a content loss function and a style loss function, respectively. Following [1, 12], we obtain feature maps through a pretrained VGG model and use them to construct the content perceptual loss $\mathcal{L}_c$ and the style perceptual loss $\mathcal{L}_s$ as follows:

$$\begin{aligned}\mathcal{L}_c =& \frac{1}{N_l}\sum_{i=0}^{N_l} \parallel \psi_i(I_o) - \psi_i(I_c) \parallel_2 \\ \mathcal{L}_s =& \frac{1}{N_l}\sum_{i=0}^{N_l} \parallel \mu(\psi_i(I_o)) - \mu(\psi_i(I_s)) \parallel_2 \\ &+ \parallel \sigma(\psi_i(I_o)) - \sigma(\psi_i(I_s)) \parallel_2\end{aligned} \tag{5}$$

where I_o represents the output of the model, I_c is the content image and I_s is the style image, $\psi_i(\cdot)$ denotes the features extracted from the i-th layer in a pretrained VGG19, and N_l is the number of layers. $\mu(\cdot)$ and $\sigma(\cdot)$ denote the mean and variance of the extracted features, respectively.

For the feature extraction module, we can train the two extractors with these loss functions. We adopt the content perceptual loss for the input and output of the content extractor ($\mathcal{L}_{cc}$: the content perceptual loss w.r.t. the content image; $\mathcal{L}_{sc}$: the content perceptual loss w.r.t the style image), and the style perceptual loss for the input and output of the style extractor ($\mathcal{L}_{cs}$: the style perceptual loss w.r.t. the content image; $\mathcal{L}_{ss}$: the style perceptual loss w.r.t. the style image). $\mathcal{L}_{fe}$ is used to calculate the total loss of feature extractors. In order to enhance the learning ability of the extractor, we implement two extractors on both the content and style images. We reconstruct the result image through the content and style features extracted from the same image, and the result image should be consistent with the original image. Here we employ two identity losses [21] to increase the severity of the penalty as follows:

$$\begin{aligned}\mathcal{L}_{fe} =& \lambda_1\mathcal{L}_{cc} + \lambda_2\mathcal{L}_{cs} + \lambda_1\mathcal{L}_{sc} + \lambda_2\mathcal{L}_{ss} \\ \mathcal{L}_{id1} =& \parallel I_{cc} - I_c \parallel_2 + \parallel I_{ss} - I_s \parallel_2 \\ \mathcal{L}_{id2} =& \frac{1}{N_l}\sum_{i=0}^{N_l} \parallel \psi_i(I_{cc}) - \psi_i(I_c) \parallel_2 \\ &+ \parallel \psi_i(I_{ss}) - \psi_i(I_s) \parallel_2\end{aligned} \tag{6}$$

where λ_1, λ_2 are the weights set to 0.7 and 1, respectively. I_{cc} and I_{ss} are the reconstructed images.

In summary, the entire network is optimized by minimizing the following function:

$$\mathcal{L} = \lambda_c\mathcal{L}_c + \lambda_s\mathcal{L}_s + \lambda_{fe}\mathcal{L}_{fe} + \lambda_{id1}\mathcal{L}_{id1} + \lambda_{id2}\mathcal{L}_{id2}. \tag{7}$$

We set $\lambda_c, \lambda_s, \lambda_{fe}, \lambda_{id1}, \lambda_{id2}$ to 7, 10, 20, 70, 1 so as to alleviate the impact from magnitude differences.

4. Experiments

4.1. Implementation Details

We adopt MS-COCO [18] as the content dataset and WikiArt [22] as the style dataset. In the training stage, all the images are randomly cropped into a fixed resolution of 256 $\times$ 256, while any image resolution is supported at the test time. We choose the Adam optimizer [14] with a learning rate of 0.0005 and use the warm-up adjustment strategy [30]. The batch size is set to 1 and we train our network with 100,000 iterations. Our model is trained on the NVIDIA Tesla A40 for about half a day.

During the training stage, we found that the style extractor trained for 12,000 iterations produced the best image style. Some style features may disappear after more iterations. We believe this is because the difference between the result image and the content image accounts for a larger proportion of the total loss, as we tend to preserve content details as much as possible. In order to reduce the total loss, the extracted style features will decrease after more rounds of training. Without stylization, the content perceptual loss will be very low, and so will the total loss. Therefore, we freeze the parameters of the style extractor after 12,000 iterations of training, while the other parts continue to participate in the training. Maybe we can also use two-stage training scheme [17].

4.2. Comparison with State-of-the-Art Methods

Transformer networks have proven their powerful performances in numerous computer vision fields. So far, state-of-the-art models, such as StyTr^2 [8], have utilized the attention mechanism. CNN-based models, despite their fast inference, can result in missing details due to their limitations of kernel weight sharing. We have chosen the mainstream style transfer models CAP-VSTNet [26], StyTr^2 [8], StyleFormer [27], and IEST [4] for comparison. We conduct both qualitative and quantitative comparisons.

4.2.1 Qualitative comparison

Figure 4 shows the visual results of the qualitative comparisons. The IEST retains more content features, but the degree of stylization is insufficient. CAP-VSTNet also has the same drawback, but its original content is more protected.

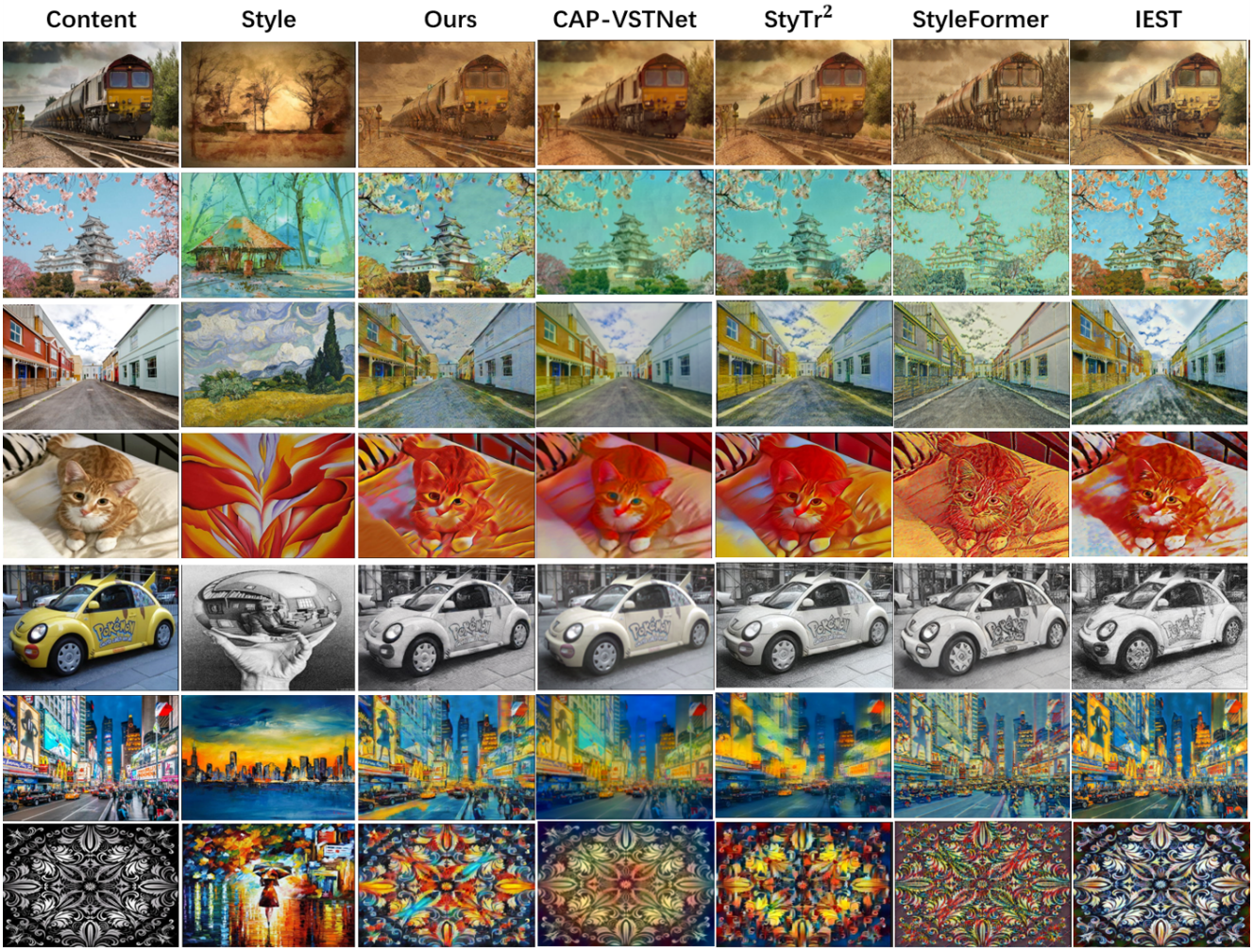

Figure 4. The visual results of qualitative comparisons

Resolution	Ours	CAP-VST	StyTr^2	StyleFormer	IEST
256 × 256	0.098	0.107	0.116	0.013	0.065
512 × 512	0.134	0.162	0.661	0.026	0.092

Table 1. Average inference time (in seconds) of the comparison methods at two output resolutions.

The StyleFormer sometimes exaggerates details, resulting in some unreasonable stylization. The local details of some results generated by the StyTr^2 are not obvious, leading to the content missing. By contrast, our model can effectively utilize the content and style features of input images and exploit their relationships. It can extract the main content lines of the original image and adopt attention mechanism to stylize these main features, which can maintain the global structure of the content image and make the stylized image look very coordinated. But we also observe that when the input content images and style images are more complex, sometimes there may be unreasonable stylization.

4.2.2 Quantitative comparison

In Table 1, we compare the inference time of these models at two output resolutions using one NVIDIA Tesla P100. As can be seen from the table, our model's inference speed is at the forefront of these mainstream models.

To quantitatively analyze the effect of generating stylized images, we randomly select 20 content images and 20 style images, and then use the mainstream models to generate 200 stylized images. We calculate the content difference and the style difference using (5). Table 2 shows that our model's comprehensive performance is at the forefront. In terms of content difference, we have a small gap compared to the StyTr^2. Our method also achieves the second-lowest style loss. Although the style difference is slightly greater, we do not pay much attention to the local detail differences between the result image and style image. What's more, we can see that CAP-VSTNet has the lowest content loss, but its degree of stylization is lacking. Through the quantitative analysis, one can see that our proposed model still retains a good performance despite significantly reduced model ca-

Model	Ours	CAP-VST	StyTr^2	StyleFormer	IEST
$\mathcal{L}_c$	1.92	0.86	1.89	2.87	1.97
$\mathcal{L}_s$	2.21	4.42	1.69	3.34	3.99

Table 2. Quantitative comparison on the content perceptual loss and style perceptual loss.

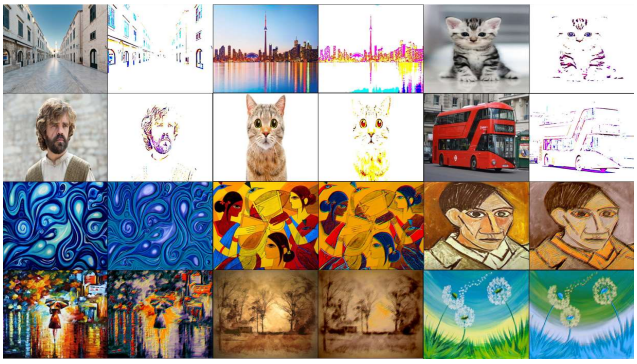

Figure 5. The features extracted by our model. The first and second rows display renderings of the content extractor, while the third and fourth rows display renderings of the style extractor.

pacity. This provides a better method for the application of style transfer.

4.3. Ablation Experiment

4.3.1 Feature Extraction

Attention mechanisms are proven to be effective in the field of style transfer. We intend to investigate whether the feature extractors work. Figure 5 plots the results generated by these extractors. By using a content feature extractor, the structure, lines, and other content features of an input image are extracted, which meet our expectations. Some background and less important contents are blurred. Through a style feature extractor, the color, texture, and other aspects of the input image are extracted. Although the feature distribution of some images has changed, we do not pay attention to the details of the style image.

In order to further investigate the efficacy of the feature extractors, we conduct ablation experiments. We remove the content feature extractor and style feature extractor from the model and do not extract the pure features of the content and style images. We directly project their patches into sequential feature embeddings and feed them into the encoder-based transformer. In order to offset the impact of different network depths, we add the encoder layers from 3 to 6. Figure 6 shows the results generated by pure encoder without those extractors.

As can be observed from Figure 6, some generated images have lost their original content structure, resulting in visual distortions (second row, first group). There are also some content features such as lines in the style image appearing in the resultant image (first row, second group). Some images have insufficient stylization in terms of details (third row, second group). Therefore, it can be concluded that our content and style feature extractors are of good performance.

Figure 6. The results without extractors. Each group has three images, the first one is a style image, the second one is a content image, and the last one is the result image.

4.3.2 Content-Aware Positional Encoding

Content-Aware Positional Encoding (CAPE) is a learnable position encoding method based on image semantic information proposed by [8]. Since our model extracts features from the content picture image, the content image will lose some semantic information. We have already shown some extracted feature maps in Figure 5. Through the content extractor, we can extract the structure, lines, and other features of the input image, but we discard style features such as colors, which destroys some of the semantic information of the image. Therefore, in order to verify whether CAPE can play a better role in our model, we carry out a ablation study. In the ablation experiment, we replace CAPE with traditional sinusoidal positional encoding and trained the model. We present the results of this ablation experiment in Figure 7. As can be seen from the experimental results in Figure 7, the results using CAPE are better than those using traditional sinusoidal positional encoding. The results without using CAPE may have unreasonable stylization, and some originally similar areas may have significant differences after stylization. We believe that although extracting features may cause losses to the semantic information of the image, we still need positional encoding to exploit the remaining information for stylization. Some features such as the background need to be similarly stylized, and different detail features can be stylized differently. Therefore, we still employ CAPE as the positional encoding method.

4.4. Output Sequential Feature Embedding

In our model, we can obtain output sequential feature embedding ε_o only through the encoder. So, we modify the

Figure 7. Ablation experiments for CAPE. From the first to the last column: style images, content images, result images using sinusoidal positional encoding, and result images using CAPE.

transformer encoder, appending a learnable sequence feature embedding ε_o to the input. Its shape is the same as content sequential feature embedding ε_c. During the training stage, as we know the resultant image should be closer to the content image, we use ε_c to initialize it. In order to further investigate its role in the model, we experiment with other initialization methods.

We first initialize it using style sequence feature embedding ε_s. It can be observed that the resultant image is very similar to the style image, which does not meet expectations. We believe that our model realize stylization based on ε_o, using the attention mechanism to calculate each part's stylization approach. Since we use ε_s to initialize ε_o, it is in the stylized state from the beginning, and the subsequent stylization effect will not be significant. We also use random initialization and zero initialization, and find that the generated stylized images are blank. We believe that our model cannot find a suitable way to stylize images without the content. The qualitative results using different output feature embedding initialization methods are presented in supplementary material due to space constraint.

In summary, ε_o is the basis for style transfer in our model, and the calculation results of the attention mechanism determine the way of stylization for each patch. Therefore, we choose to initialize it with ε_c, which is more in line with the goal of style transfer.

4.5. User Study

In order to better evaluate the performance of our model, we conduct a user study. The comparison resources come from Figure 4. We invited 45 college students and 10 middle-aged people to conduct this survey. We have set three types of questions for the purpose of style transfer task. The first question is which model can better maintain the original image's content structure. The second question is which model's result is closer to the target style image. The third question is which model's result after stylization looks the most harmonious. We will provide two examples for each type of question. The results of the survey are shown in Figure 8. In terms of the example we provided, from the results, we can see that our model's ability to maintain the original image content structure is similar to that of CAP-VSTNet, and its ability to achieve stylization is optimal, followed closely by StyTr^2 and IEST. As for the ability to achieve reasonable stylization, our model is also outstanding. In order to further demonstrate the performance of the model and reduce randomness, we hope that more people can use our model to produce the expected results.

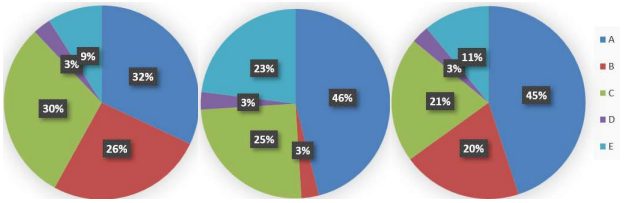

Figure 8. Results of User Study. The above three figures correspond to questions one, two, and three, respectively. A-Puff-Net. B-CAP-VSTNet. C-StyTr^2. D-StyleFormer. E-IEST.

5. Conclusion

In this paper, we proposed a novel style transfer model dubbed the Puff-Net. The proposed model consists of two feature extractors and a transformer that only contains the encoders. We first obtained pure content images and pure style images through the two feature extractors. Then we fed them into an efficient encoder-based transformer for stylization, in which a sequence of learnable tokens were added to interact with pure content and style tokens. Our model solves the problem of huge capacity in existing transformer-based models. We also verified its good performance through extensive experiments and demonstrated the potential application of style transfer in practice.

Acknowledgement

This work was partially supported by the National Natural Science Foundation of China (No. 62276129 & 62272227), and the Natural Science Foundation of Jiangsu Province (No. BK20220890).

References

[1] Jie An, Siyu Huang, Yibing Song, Dejing Dou, Wei Liu, and Jiebo Luo. Artflow: Unbiased image style transfer via reversible neural flows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 862–871, 2021. 2, 5

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas

Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 3

[4] Haibo Chen, Zhizhong Wang, Huiming Zhang, Zhiwen Zuo, Ailin Li, Wei Xing, Dongming Lu, et al. Artistic style transfer with internal-external learning and contrastive learning. *Advances in Neural Information Processing Systems*, 34:26561–26573, 2021. 2, 5

[5] Yingying Deng, Fan Tang, Weiming Dong, Wen Sun, Feiyue Huang, and Changsheng Xu. Arbitrary style transfer via multi-adaptation network. In *Proceedings of the 28th ACM international conference on multimedia*, pages 2719–2727, 2020. 1, 2

[6] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 4

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 3

[8] Deng et al. Stytr2: Image style transfer with transformers. In *CVPR*, pages 11326–11336, 2022. 2, 3, 4, 5, 7

[9] Ho et al. Denoising diffusion probabilistic models. *NIPS*, 33:6840–6851, 2020. 3

[10] Yuxin Fang, Bencheng Liao, Xinggang Wang, Jiemin Fang, Jiyang Qi, Rui Wu, Jianwei Niu, and Wenyu Liu. You only look at one sequence: Rethinking transformer in vision through object detection. *Advances in Neural Information Processing Systems*, 34:26183–26197, 2021. 3

[11] 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. 1, 2

[12] 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. 1, 2, 5

[13] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*, pages 5583–5594. PMLR, 2021. 3

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[15] Gihyun Kwon and Jong Chul Ye. Clipstyler: Image style transfer with a single text condition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18062–18071, 2022. 3

[16] Jack Lanchantin, Tianlu Wang, Vicente Ordonez, and Yanjun Qi. General multi-label image classification with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16478–16488, 2021. 3

[17] Hui Li, Xiao-Jun Wu, and Josef Kittler. Rfn-nest: An end-to-end residual fusion network for infrared and visible images. *Information Fusion*, 73:72–86, 2021. 5

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 5

[19] Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Meiling Wang, Xin Li, Zhengxing Sun, Qian Li, and Errui Ding. Adaattn: Revisit attention mechanism in arbitrary neural style transfer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6649–6658, 2021. 2

[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 3

[21] Dae Young Park and Kwang Hee Lee. Arbitrary style transfer with style-attentional networks. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5880–5888, 2019. 1, 2, 5

[22] Fred Phillips and Brandy Mackintosh. Wiki art gallery, inc.: A case for critical thinking. *Issues in Accounting Education*, 26(3):593–608, 2011. 5

[23] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 4

[24] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8242–8250, 2018. 1

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 3

[26] Linfeng Wen, Chengying Gao, and Changqing Zou. Cap-vstnet: Content affinity preserved versatile style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18300–18309, 2023. 2, 5

[27] Xiaolei Wu, Zhihao Hu, Lu Sheng, and Dong Xu. Styleformer: Real-time arbitrary style transfer via parametric style composition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14618–14627, 2021. 2, 5

[28] Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. *arXiv*, 2020. 5

[29] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34:12077–12090, 2021. 3

[30] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the trans-

former architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020. 5

[31] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6881–6890, 2021. 3, 4

[32] Man Zhou, Jie Huang, Yanchi Fang, Xueyang Fu, and Aiping Liu. Pan-sharpening with customized transformer and invertible neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3553–3561, 2022. 4