APRNet: Attention-based Pixel-wise Rendering Network for Photo-Realistic Text Image Generation

Yangming Shi , Haisong Ding, Kai Chen, and Qiang Huo

Abstract—Style-guided text image generation tries to synthesize text image by imitating reference image's appearance while keeping text content unaltered. The text image appearance includes many aspects. In this paper, we focus on transferring style image's background and foreground color patterns to the content image to generate photo-realistic text image. To achieve this goal, we propose 1) a content-style cross attention based pixel sampling approach to roughly mimicking the style text image's background; 2) a pixel-wise style modulation technique to transfer varying color patterns of the style image to the content image spatial-adaptively; 3) a cross attention based multi-scale style fusion approach to solving text foreground misalignment issue between style and content images; 4) an image patch shuffling strategy to create style, content and ground truth image tuples for training. Experimental results on Chinese handwriting text image synthesis with SCUT-HCCDoc and CASIA-OLHWDB datasets demonstrate that the proposed method can improve the quality of synthetic text images and make them more photo-realistic.

Index Terms—Style transfer, text image generation, attention

1 Introduction

Style-guided text image generation is a challenging task, which tries to synthesize text images by imitating reference style image's appearance while keeping text content unaltered (*e.g.* [1]–[3]). The appearance of a text image includes foreground and background color patterns, spatial transformations and deformations, typography (for printed text), writing styles and stroke thickness (for handwriting text), etc [2]. In this paper, we propose an **A**ttention-based **P**ixel-wise **R**endering **Net**work (APRNet) to transfer the foreground and background color patterns from a style text image to a new image without altering its text content.

Previously, text image generation is mainly based on traditional signal processing techniques, which has been successfully applied to augment optical character recognition (OCR) model training in natural scene scenarios (*e.g.* [4]–[6]). These kinds of methods synthesize text images by rendering text transcriptions with given typeset fonts, followed by colorization, distortion, and natural scene image blending. The synthetic image style is controlled by predefined rules and not photo-realistic.

Recently, neural network based style-guided approaches have been applied to printed [7], [8] and handwriting text image generation [1]–[3]. Equipped by the generative adversarial network (GAN) [9], style transfer [10]–[15] and image-to-image translation [16]–[19], the quality of synthetic text images has been improved greatly. However, transferring varying foreground and background color patterns from camera-captured style images to content images at line-level is still an unsolved problem. It is mainly caused by complicated backgrounds, various lighting conditions and foreground (text) misalignment.

To improve the color pattern transfer quality of text image generation, especially at line-level, we make the following technical contributions:

1) An **Atten**tion-based **Pix**el S**amp**ling (**AttnPixamp**) module is proposed to roughly mimicking style text image's background;
2) A **Pix**el-wise St**y**le **Mod**ulation (**PixyMod**) approach is derived based on StyleGANv2 [15] to transfer spatial-varying color patterns;
3) An **Atten**tion-based **Mul**ti-scale **S**tyle **F**usion (**AttnMuSF**) module is designed based on spatial pyramid pooling [20] and cross attention to solve text foreground misalignment issue between style and content images;
4) An image patch shuffling strategy named *Single Crop* is used to train this well-designed APRNet in a self-supervised manner;
5) Experimental results on Chinese handwriting text image synthesis with SCUT-HCCDoc [21] and CASIA-OLHWDB [22] datasets demonstrate that the proposed method can improve the quality of synthetic text images and make them more photo-realistic.

The rest of this paper is organized as follows. In Sec. 2, we review previous style transfer and text image generation works. In Sec. 3, we introduce our APRNet design in detail. Experimental results are presented in Sec. 4 and limitations of our method are discussed in Sec. 5. Finally we conclude the paper in Sec. 6.

Yangming Shi is with the Automation Department of University of Science and Technology of China. (E-mail: ymshi@mail.ustc.edu.com)
Haisong Ding, Kai Chen and Qiang Huo are with the Speech group of Microsoft Research Asia, Beijing, China (E-mail: {hadin, kaic, qianghuo}@microsoft.com).
This work was done when Yangming Shi worked as the intern in Speech Group, Microsoft Research Asia, Beijing, China.

2 Related Work

2.1 Style transfer and exemplar-guided generation

Style-guided handwritten text generation is a sub-field of style transfer, whose goal is to mimic the reference image's appearance and transfer expected styles into given content images. Gatys *et al*. [10] perform artistic style transfer by jointly minimizing the feature and style reconstruction loss (using Gram Matrix to represent styles). StyleGAN [14] employs AdaIN for style integration, while StyleGANv2 [15] further improves generation quality by activation standard deviation (std) modulation and normalization. The latest version of StyleGANv3 [23] presents a principled solution to aliasing caused by pointwise nonlinearities by considering their effects in a continuous domain. Besides, GuaGAN [24] proposes a spatially-adaptive normalization residual block to catch spatial-varying styles.

Another similar task is exemplar-guided image synthesis, whose applications include pose-guided person generation, face attributes editing, virtual try-on, sketch image colorization, etc. The algorithms [25]–[33] range from explicit image matching by giving additional conditions to implicit learning using flexible cross-attention models. However, unlike these methods which pay more attention to the global similarity between references and outputs, text image generation cares both the style realism of global and local regions and the rightness of content.

2.2 Text image generation

Handwritten text image generation. Handwritten text generation can be traced to 2014, when the generative adversarial network [9] was proposed to generate handwritten digital symbols from a noise vector. Afterwards, the cGAN [34] allows controlling the class of generated contents by inputting an additional class condition. The latest techniques [3], [35]–[40] focus on line-level style-guided text image generation. These systems synthesized glyph-realistic text line images with different writer styles, usually optimized by discriminators, writer classifiers, and character recognizers. Besides, the metaHTR [41] attempted to use a meta-learning framework to achieve writing style adaptive generation with few data. However, most previous works pay more attention to handling diverse glyphs while not considering complex style patterns in reality. In this regard, skeleton-based approaches avoid the glyph burden and focus more on style transfer. SC-GAN [1] employs the StyleGAN's decoder to integrate style code into content code and render photo-realistic text images in simple cases. Coincidentally, TextStyleBrush [2] distills text image content from its appearance, which can be applied to the new content with self-supervised ways. However, these methods still cannot handle complex foreground and background color patterns due to the limitation of employed style vectors.

Printed text image generation. Printed image generation remains a hot topic. Early studies [4]–[6] synthesized images by traditional signal processing techniques such as colorization, distortion, and natural scene image blending. However, these kinds of methods suffered none photo-realistic from pre-defined rules. The latest advances also benefit from GANs. Wu *et al*. [7] proposed an end-to-end style retention network (SRNet) with text conversion, background inpainting, and fusion for editing text in the wild. Shimoda *et al*. [8] suggested a text vectorization technique by leveraging the advantage of differentiable text rendering to accurately reproduce the input raster text in a resolution-free parametric format. On the other hand, the printed character image generation [42]–[48] has been comprehensively studied. These methods simulate writing styles such as fonts, thickness, and artistic style while preserving the base structure. However, it is not recommended to directly employ related techniques for handwriting image generation task considering different application scenarios.

3 Methodology

3.1 Overview

In this paper, we explore solutions to render photo-realistic text images with style guidance for handwriting text image rendering. As shown in Fig. 1, our model takes a binary handwriting skeleton image (denoted as $\mathbf{I}_c$) as content input, and a real handwriting text line image (denoted as $\mathbf{I}_s$) as style reference input. The sizes of $\mathbf{I}_c$ and $\mathbf{I}_s$ are all $H \times W$. In practice, the skeleton images can be rendered from online handwriting texts (*e.g*. [1]), synthesized from generative models (*e.g*. [49]), or extracted from real handwriting images (*e.g*. [50]).

We want to generate a rendered image $\mathbf{I}_r$, which shares the same foreground and background color patterns with $\mathbf{I}_s$, and its text content and character glyphs keep the same with $\mathbf{I}_c$. This is a challenging style transfer task due to complex backgrounds, colors, textures, noises, stroke thicknesses, and writing styles in $\mathbf{I}_s$. Besides, text foreground misalignment between $\mathbf{I}_s$ and $\mathbf{I}_c$ brings in more difficulties to achieve this goal.

During rendering, $\mathbf{I}_c$ and $\mathbf{I}_s$ are encoded first by a fully convolutional network (FCN) based content and style encoders, respectively. Extracted content and style feature maps are stored into content and style banks and serve as the inputs of the following 2-stage rendering procedure. The first rendering stage is called *AttnPixamp*, which samples pixels from style images directly and aggregates them with a well-designed content-style cross attention mechanism to generate preliminary rendering results. The second stage consists of two modules: an *AttnMuSF* module, which fuses multi-scale style features to construct a better style encoding, and a *PixyMod* module, which applies style modulation to content features spatial-adaptively to capture varying style patterns. The outputs of these two stages will be fused by addition to synthesize a final rendered image $\mathbf{I}_r$.

3.2 Content and style encoders

The content and style encoders in our model share the same FCN architecture as shown in Fig. 2. This architecture consists of 4 stages without using any max-pooling layers, and the output sizes of each stage are $\frac{H}{2} \times \frac{W}{2} \times 32$, $\frac{H}{4} \times \frac{W}{4} \times 64$, $\frac{H}{8} \times \frac{W}{8} \times 128$ and $\frac{H}{8} \times \frac{W}{8} \times 256$, respectively. These output feature maps of the content/style encoder are stored in the aforementioned content/style bank as shown in Fig. 1 for the next rendering procedures.

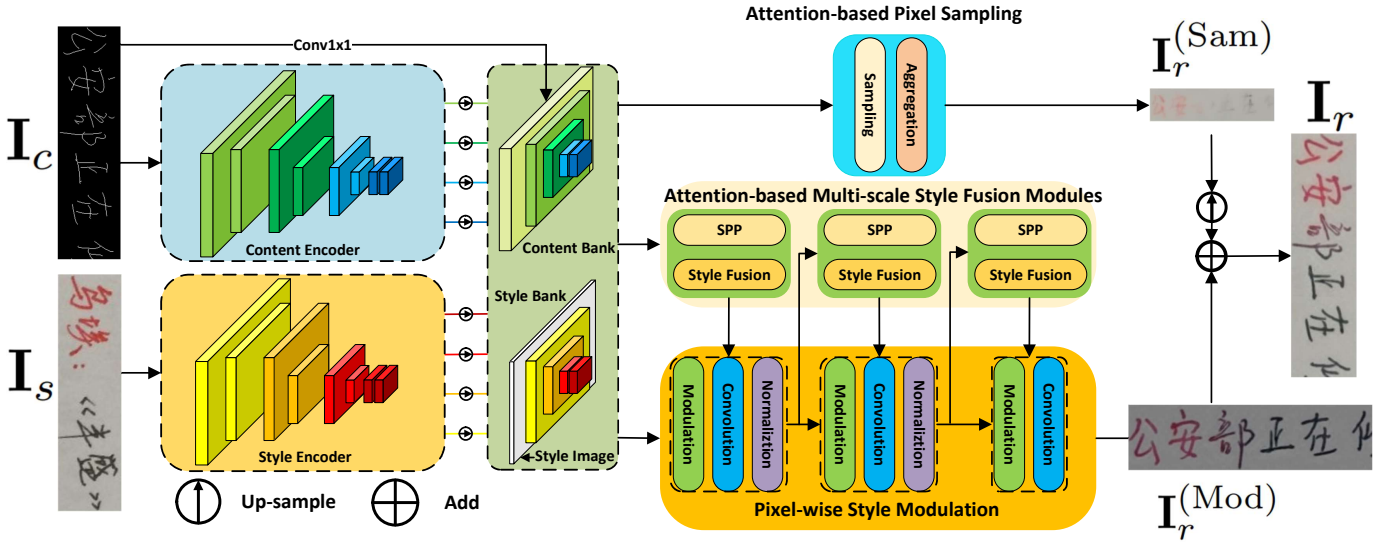

Fig. 1. Overview of Attention-based Pixel Rendering Network. It consists of a style/content encoder to extract style/content encodings, an AttnPixamp module to render background, an AttnMuSF module to generate scale-fused style encodings, and a PixyMod module to render foreground and refine background spatial-adaptively. These designs will help to synthesize photo-realistic text images with spatial-varying color patterns.

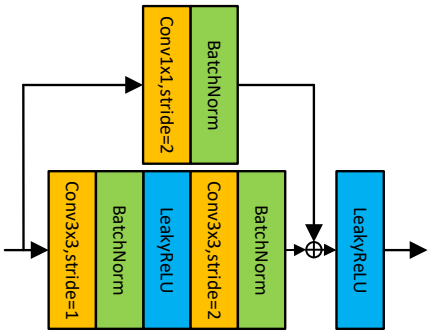

Fig. 2. Blocks of one FCN encoder stage. Each encoder consists of 4 stages. All strides in convolution layers of the last stage are 1. For content encoder, the kernel size of highway convolution (the upper one) is 2×2.

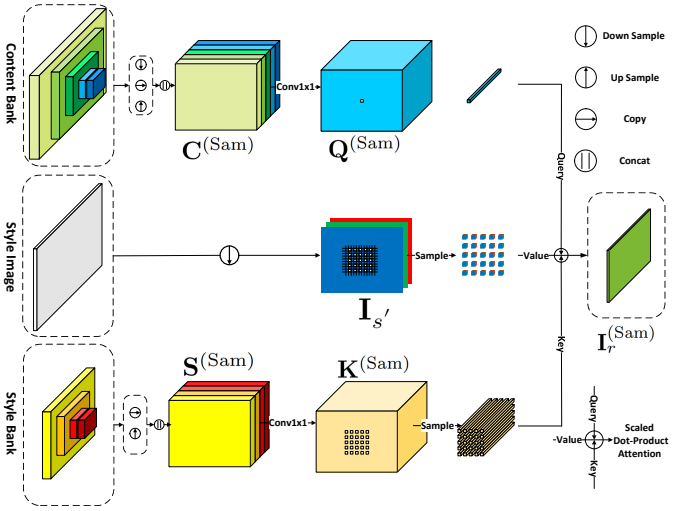

Fig. 3. Architecture of attention-based pixel sampling module.

Compared with SC-GAN [1], which only uses the content encoder's last output as input to decoder and style encoder's last output to extract style vector, our rendering modules leverage early stage low-level features such as edge, texture, etc., and later stage high-level semantic features simultaneously. Low-level features are helpful for stroke generation, and high-level features are essential for background rendering.

In the content bank, we also store an $H \times W \times 256$ tensor, which is extracted from $\mathbf{I}_c$ by a 256-channel 1×1 convolution (Fig. 1). This redundant operation is quite helpful to avoid stroke loss and get continuous strokes.

Since $\mathbf{I}_s$ is used in AttnPixamp module, it is also stored in the style bank (Fig. 1).

3.3 Rendering stages

3.3.1 *Attention-based pixel sampling*

Generally speaking, the majority of pixels in a text line image belong to background. Given a background pixel of $\mathbf{I}_c$, we can usually find background pixels from its respective neighborhood in $\mathbf{I}_s$. Based on this observation, it would be a more direct way to render $\mathbf{I}_r$'s background pixels by sampling from their respective neighborhoods in $\mathbf{I}_s$ compared with regressing RGB vectors through complex operations.

In practice, we first get a $\frac{H}{2} \times \frac{W}{2}$ style image $\mathbf{I}_{s'}$ by down-sampling $\mathbf{I}_s$ to reduce following memory cost. For coordinate (i, j) in $\mathbf{I}_{s'}$, k^2 pixels from its $(km + k - m) \times (km + k - m)$ square neighborhood in $\mathbf{I}_{s'}$ are sampled uniformly, where m is the horizontal/vertical distance between 2 successively sampled pixels (Fig. 3). These sampled pixels are then aggregated by a content-style cross attention mechanism. We denote the rendered image in this step as $\mathbf{I}_r^{(\mathrm{Sam})}$, whose size is also $\frac{H}{2} \times \frac{W}{2}$.

The attention mechanism is shown in Fig. 3, and it works as follows:

1) Down-sample or up-sample all feature maps in content and style banks to $\frac{H}{2} \times \frac{W}{2}$, and concatenate them respectively to get content tensor $\mathbf{C}^{(\mathrm{Sam})} \in$

$\mathbb{R}^{\frac{H}{2}\times\frac{W}{2}\times 736}$ and style tensor $\mathbf{S}^{(\mathrm{Sam})} \in \mathbb{R}^{\frac{H}{2}\times\frac{W}{2}\times 480}$;

2) Modify channel numbers of $\mathbf{C}^{(\mathrm{Sam})}$ and $\mathbf{S}^{(\mathrm{Sam})}$ to d_s by 1×1 convolutions respectively, and the resulted tensors are denoted as $\mathbf{Q}^{(\mathrm{Sam})}$ and $\mathbf{K}^{(\mathrm{Sam})}$.

3) Denote k^2 sampled style pixels as value tensor $\mathbf{V}_{i,j}^{(\mathrm{Sam})}$, k^2 respective vectors in $\mathbf{K}^{(\mathrm{Sam})}$ as key tensor $\mathbf{K}_{i,j}^{(\mathrm{Sam})}$, and the (i,j)-th vector in $\mathbf{Q}^{(\mathrm{Sam})}$ as query vector $\mathbf{q}_{i,j}^{(\mathrm{Sam})}$, we can get the rendered $\mathbf{I}_{r_{i,j}}^{(\mathrm{Sam})}$ through scaled dot-product cross attention:

$$\mathbf{I}_{r_{i,j}}^{(\mathrm{Sam})} = \left[\mathbf{Softmax}(\frac{\mathbf{q}_{i,j}^{(\mathrm{Sam})^T}\mathbf{K}_{i,j}^{(\mathrm{Sam})}}{\sqrt{d_s}})\right]^T \mathbf{V}_{i,j}^{(\mathrm{Sam})} \tag{1}$$

.

Repeat the above procedure for all coordinates to get $\mathbf{I}_r^{(\mathrm{Sam})}$.

Since most sampled style image pixels are background ones, it is predictable that this module will focus on rendering the content image's background and not suitable for foreground stroke rendering. We design other two modules to render foreground as well as fine-tune background pixels.

3.3.2 Pixel-wise style modulation

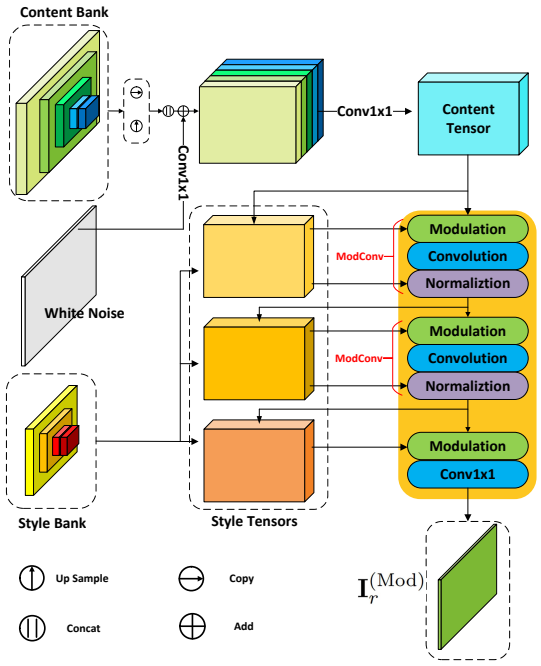

Fig. 4. Details of pixel-wise style modulation.

Previous works on text image style transfer usually encode a style input as a vector, and apply it to a generation procedure through AdaIN [13]. It has been shown in [15] that instance normalization may cause water droplet-like artifacts in generated images. StyleGANv2 [15] proposed to transfer styles through activation standard deviation (std) modulation and normalization to solve this problem and improve image quality. However, it is still a style vector based approach, which means the transferred styles are uniform across spatial coordinates [24]. Experimental results in [1] and this paper also verify that for text image synthesis scenario, style vector based approach can only synthesize images with uniform color patterns.

It has been proven by SPADE [24] that applying style encoding spatial-adaptively is essential to catch spatial-varying styles. Inspired by StyleGANv2 and SPADE, we propose a PixyMod module to transfer spatial-varying color patterns of the style image.

In StyleGANv2, the modulate and normalize activation std with style vector is equivalent to modulate and normalize weights of convolution layer, because the weights and style vector are all shared across spatial coordinates. For PixyMod, we have to back-off to applying modulation and normalization on input and output tensors of convolution layer pixel-wisely.

Denote the content tensor as $\mathbf{C} \in \mathbb{R}^{H\times W\times I}$. Before input into convolution layer, it is first modulated by style tensor $\mathbf{S} \in \mathbb{R}^{H\times W\times I}$ element-wisely to get $\mathbf{M} \in \mathbb{R}^{H\times W\times I}$ as

$$\mathbf{M} = \mathbf{S} \otimes \mathbf{C} \tag{2}$$

where $\otimes$ is an element-wise multiplication.

Then, $\mathbf{M}$ is convoluted by $\mathbf{W} \in \mathbb{R}^{O\times k_h\times k_w\times I}$ to get $\mathbf{U} \in \mathbf{R}^{H\times W\times O}$:

$$\mathbf{U} = \mathbf{W} * \mathbf{M} \tag{3}$$

where $*$ is a convolution operation.

To conduct pixel-wise normalization, for simplicity, let's take 3×3 convolution for an example. $\mathcal{P} = \{(-1,-1),(-1,0),\cdots,(0,1),(1,1)\}$ defines a 3×3 kernel, so

$$u_{j,k,o} = \sum_{\mathbf{p}\in\mathcal{P},i} w_{i,\mathbf{p},o}\cdot m_{(j,k)+\mathbf{p},i} \tag{4}$$

$$= \sum_{\mathbf{p}\in\mathcal{P},i} w_{o,\mathbf{p},i}\cdot s_{(j,k)+\mathbf{p},i}\cdot c_{(j,k)+\mathbf{p},i}\ . \tag{5}$$

Based on the i.i.d. and unit std assumption of each element in $\mathbf{C}$, after modulation and convolution, the std of $u_{j,k,o}$ is

$$\sigma_{j,k,o} = \sqrt{\sum_{\mathbf{p}\in\mathcal{P},i} w_{o,\mathbf{p},i}^2\cdot s_{(j,k)+\mathbf{p},i}^2}\ . \tag{6}$$

It is equivalent to convolute $\mathbf{S}^2$ with $\mathbf{W}^2$ and then get square roots element-wisely as

$$\mathbf{\Sigma} = \sqrt{\mathbf{W}^2 * \mathbf{S}^2} \tag{7}$$

where $(\cdot)^2$ and $\sqrt{(\cdot)}$ are all element-wise operations.

Finally, we normalize the output tensor $\mathbf{N} \in \mathbf{R}^{O\times H\times W}$ as

$$\mathbf{N} = \frac{\mathbf{W} * (\mathbf{S}\otimes\mathbf{C})}{\sqrt{\mathbf{W}^2 * \mathbf{S}^2}}\ . \tag{8}$$

We call Eq. (8) as *ModConv*. In practice, as shown in Fig. 4, several ModConv layers are stacked and then followed by a modulation and 1×1 convolution layer to generate a style-transferred image denoted as $\mathbf{I}_r^{(\mathrm{Mod})}$, which will then be fused with $\mathbf{I}_r^{(\mathrm{Sam})}$ by addition to generate a final rendered image $\mathbf{I}_r$. Due to computation and memory limitations, the size of $\mathbf{I}_r^{(\mathrm{Sam})}$ is set to $\frac{H}{2}\times\frac{W}{2}$. Before addition, we need to up-sample it to $H\times W$. As shown in Fig. 1, we get

$$\mathbf{I}_r = \mathbf{I}_r^{(\mathrm{Mod})} + \mathrm{UpSample}(\mathbf{I}_r^{(\mathrm{Sam})})\ . \tag{9}$$

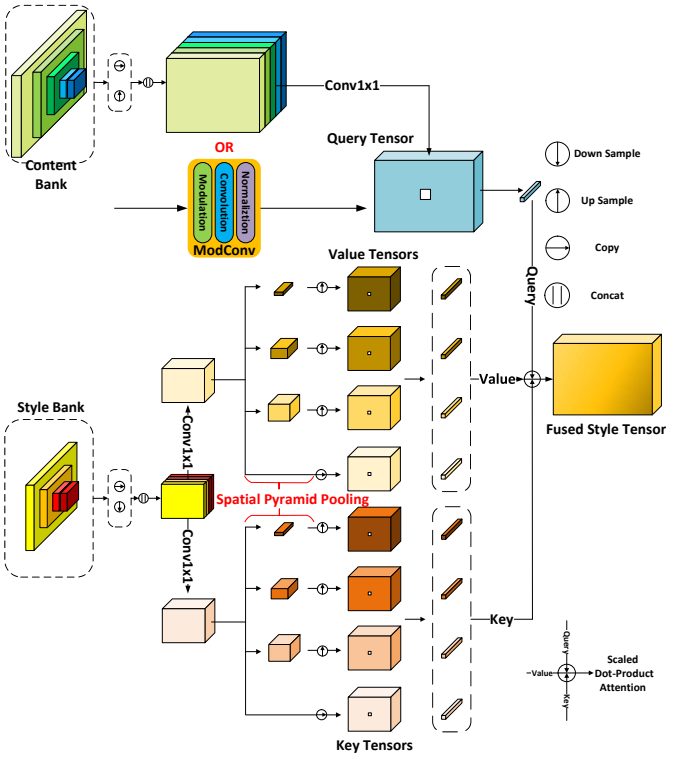

Fig. 5. Details of attention-based multi-scale style fusion.

3.3.3 Attention-based multi-scale style fusion

Given extracted style feature maps from different stages, generating style tensors for PixyMod is another contribution of this paper. A natural way to achieve this goal is to upsample all feature maps in style bank to $H \times W$ and concatenate them together, then feed this concatenated tensor into a 1×1 convolution layer to make its channel number the same as the content tensor in PixyMod.

This strategy works well in cases where the text regions in content and style image align well. However, when misalignment happens, for example in Fig. 8, artifacts happen on the rendered image because respective regions on the style image only contain background pixels.

To eliminate misalignment-caused artifacts, we design an AttnMuSF module to generate multi-scale style tensors through spatial pyramid pooling (SPP) [20] and fuse them with a cross channel attention mechanism spatial-adaptively.

As shown in Fig. 5, content tensor, which works as queries in cross channel attention mechanism, is obtained by concatenation and 1×1 convolution, and we denote it as $\mathbf{Q}^{(\mathrm{SF})} \in \mathbb{R}^{H \times W \times d_f}$.

Feature maps in style bank are also concatenated first. Due to memory limitation, the shape of this concatenated tensor $\mathbf{S}^{(\mathrm{cat})}$ is $\frac{H}{8} \times \frac{W}{8} \times 480$. To get key tensors, a 1×1 convolution is then applied to $\mathbf{S}^{(\mathrm{cat})}$ to generate $\mathbf{K}_0^{(\mathrm{SF})} \in \mathbb{R}^{\frac{H}{8} \times \frac{W}{8} \times d_f}$. To capture larger scale style encoding, SPP is applied to $\mathbf{K}_0^{(\mathrm{SF})}$ and get $\mathbf{K}_1^{(\mathrm{SF})} \in \mathbb{R}^{4 \times \frac{4W}{H} \times d_f}$, $\mathbf{K}_2^{(\mathrm{SF})} \in \mathbb{R}^{2 \times \frac{2W}{H} \times d_f}$, $\mathbf{K}_3^{(\mathrm{SF})} \in \mathbb{R}^{1 \times \frac{W}{H} \times d_f}$. These 3 tensors are then up-sampled to $\frac{W}{8} \times \frac{H}{8}$. Finally we get 4 key tensors denoted as $\mathbf{K}^{(\mathrm{SF})} = \{\mathbf{K}_i^{(\mathrm{SF})} | i = 0, 1, 2, 3\}$. Similarly, we can get multi-scale value tensors from $\mathbf{S}^{(\mathrm{cat})}$, denoted as $\mathbf{V}^{(\mathrm{SF})} = \{\mathbf{V}_i^{(\mathrm{SF})} \in \mathbb{R}^{\frac{H}{8} \times \frac{W}{8} \times I} | i = 0, 1, 2, 3\}$, where I equals to the channel number of content tensor to be modulated in PixyMod.

Given a query vector $\mathbf{q}_{j,k}^{(\mathrm{SF})}$, its respective key and value vectors will be $\mathbf{K}_{\lfloor \frac{j}{8} \rfloor, \lfloor \frac{k}{8} \rfloor}^{(\mathrm{SF})} = \{\mathbf{k}_{i, \lfloor \frac{j}{8} \rfloor, \lfloor \frac{k}{8} \rfloor}^{(\mathrm{SF})} | i = 0, 1, 2, 3\}$ and $\mathbf{V}_{\lfloor \frac{j}{8} \rfloor, \lfloor \frac{k}{8} \rfloor}^{(\mathrm{SF})} = \{\mathbf{v}_{i, \lfloor \frac{j}{8} \rfloor, \lfloor \frac{k}{8} \rfloor}^{(\mathrm{SF})} | i = 0, 1, 2, 3\}$. The fused style vector on this coordinate is as follows:

$$\mathbf{s}_{j,k} = \left[\mathbf{Softmax}\left(\frac{\mathbf{q}_{j,k}^{(\mathrm{SF})^T} \mathbf{K}_{\lfloor \frac{j}{8} \rfloor, \lfloor \frac{k}{8} \rfloor}^{(\mathrm{SF})}}{\sqrt{d^{(\mathrm{SF})}}} \right) \right]^T \mathbf{V}_{\lfloor \frac{j}{8} \rfloor, \lfloor \frac{k}{8} \rfloor}^{(\mathrm{SF})}. \quad (10)$$

Each modulation operation in PixyMod has its own AttnMuSF module to generate style tensor, whose query tensor is from content bank or previous ModConv layer's output (Fig. 4 and Fig. 5).

3.4 Training strategy

3.4.1 Training data generation

To train APRNet, a dataset containing matched content, style, and ground truth images is required. However, this kind of dataset is difficult to collect. SC-GAN [1] crops 2 non-overlapping image patches from a real text line image randomly. One serves as the style image, and another as the ground truth image. A skeleton extracted from the ground truth image is used as the content image. This strategy is based on an assumption that the style within a text line image is uniform. Apparently, this assumption does not always hold.

We propose a new training data generation strategy named *Single Crop* to solve this problem. Different from SC-GAN, given a real text line image, we crop 1 patch from it randomly. This patch serves as the ground truth image and the skeleton extracted from it as the content image. While the style image is generated from ground truth image by the following procedure:

1) Divide the image into 16×16 patches;
2) For each patch, randomly rotate it by $0°$, $90°$,$180°$ or $270°$;
3) For each patch, randomly swap it with one of its 8 neighbor patches.

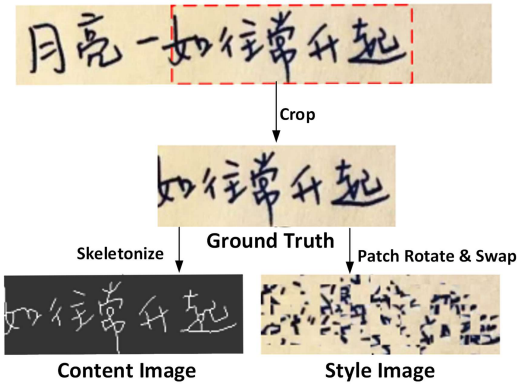

Fig. 6. Illustration of *Single Crop* strategy.

This *Single Crop* method is illustrated in Fig. 6. With this method, we can create roughly matched content, style, and ground truth images. Patch rotation and shuffling can avoid overfitting and simulate text region misalignment phenomenon in real scenarios.

3.4.2 Loss function

The loss function we used is general, which consists of content loss $\mathcal{L}_c$, perceptual loss $\mathcal{L}_p$ and adversarial loss $\mathcal{L}_a$:

$$\mathcal{L} = \lambda_c \mathcal{L}_c + \lambda_p \mathcal{L}_p + \lambda_a \mathcal{L}_a \tag{11}$$

where $\lambda_{c,p,a}$ are hyper-parameters to control weights of loss components.

We use pixel-wise L_1 distance between $\mathbf{I}_r$ and ground truth image $\mathbf{I}_g$ as $\mathcal{L}_c$, and mean square error between $\mathbf{I}_r$ and $\mathbf{I}_g$'s 5-stage CNN feature maps extracted by a pretrained VGG-19 [51] as $\mathcal{L}_p$. For $\mathcal{L}_a$, we compute adversarial loss for every pixel in discriminator's output [16] to make the synthetic image photo-realistic.

4 Experiments

4.1 Experimental Setup

To verify the effectiveness of our design, we evaluate it on a Chinese handwriting text image synthesis task with SCUT-HCCDoc dataset [21] and CASIA-OLHWDB dataset [22]. The SCUT-HCCDoc dataset contains camera-captured document text line images with diverse appearances, backgrounds, and resolutions, which are captured from different application scenarios. We use the SCUT-HCCDoc dataset to train our models and serve as the style images in testing. The CASIA-OLHWDB dataset contains online ink trajectories of Chinese text line images. We use the CASIA-OLHWDB2.0-2.2 dataset to create skeleton text line images, which serve as the content images in testing.

Specifically, in the training phase, we randomly split SCUT-HCCDoc dataset into training and validation subsets, which contains 83,926 and 9,325 images, respectively. When constructing a mini-batch, images are first resized to $H = 128$ while keeping aspect ratio, then randomly cropped to 128×384 patches, which serve as style images. The corresponding content images are obtained using adaptive threshold binarization and skeletonization methods. In the testing phase, we first use the content and style images in the validation set to reconstruct the ground-truth style images. Then we measure their similarities using Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) [52], and LPIPS distance [53] metrics. We also use CASIA-OLHWDB2.0-2.2 as content images and SCUT-HCCDoc as style images to generate a total of 20,000 synthetic text line images. We measure the similarities between rendered images and corresponding style images using Fréchet inception distance (FID) [54] and Kernel Inception Distance (KID) [55] metrics.

4.2 Baseline results

We first compare AdaIN-based SC-GAN [1] model with a StyleGANv2-based [15] baseline model. Both models are style vector based. To make the baseline model comparable with APRNet, it also consists of two stages, which generate

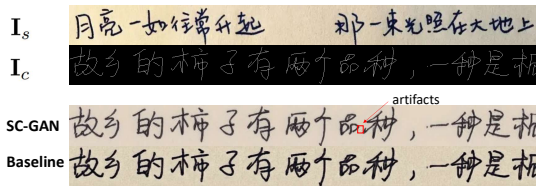

Fig. 7. Comparison results between SC-GAN and our StyleGANv2 baseline model. Both models are only able to generate uniform styles across coordinates. There is a "water droplet" artifact in the SC-GAN rendered image which is also observed in [15]. The StyleGANv2 based model eliminates this artifact and the generated image is more style-consistent with the style reference.

$\frac{H}{2} \times \frac{W}{2}$ and $H \times W$ images, respectively. In both stages, the style vectors are obtained by directly flattening the last stage's style bank to a vector with global average pooling, followed by a 1×1 convolution. These vectors are then integrated in the rendering procedure using StyleGANv2's "demodulation" operation. As shown in Fig. 7, the lack of spatial information in style encoding makes SC-GAN and baseline model only generate uniform styles across coordinates. "Water droplet" artifacts observed in [15] also happen in our SC-GAN results. Figure 7 also shows that images rendered by baseline model are more style-consistent with style reference than SC-GAN.

4.3 Effect of pixel-wise style modulation

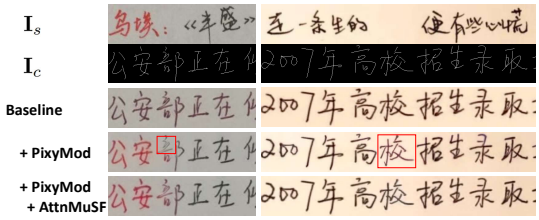

Fig. 8. By replacing style vectors with style tensors and integrating style using PixyMod, the spatial-varying styles can be successfully transferred. Equipping PixyMod with AttnMuSF, the rendering quality on misaligned content-style regions is improved.

In this section, we replace the style vectors in baseline model with style tensors and transfer spatial-varying color patterns using PixyMod module. This model also consists of 2 rendering stages, and both are based on PixyMod. The style tensors are obtained by up/down-sampling all tensors in style bank to the same spatial size as content tensor and then concatenating them together. The concatenated tensors are fed into 1×1 convolutions to match the channel numbers of content tensors. In this way, these style tensors are able to preserve the spatial-varying styles. The comparison results of PixyMod and baseline model are shown in Fig. 8. Clearly, compared with baseline, with the help of PixyMod, the spatial-varying styles can be successfully transferred. However, since the style tensors only encode local style information, when the model is trying to render a text region of content image while its corresponding style region only contains background pixels, artifacts will happen, as shown in the red rectangle area of Fig. 8. Such misalignment problems will become severe especially when the style image contains wide spaces.

TABLE 1
Quantitative comparisons of different methods. ↑ indicates that higher is better, and ↓ indicates lower is better. The best results are in bold.

Methods	Validation (9,325 pairs)			Test (20,000 pairs)	
	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	KID ↓ ($\times 10^{-2}$)
SC-GAN (StyleGANv1-based)	22.03	0.826	0.130	41.17	2.54±0.17
Baseline (StyleGANv2-based)	24.54	0.862	0.109	40.71	2.34±0.13
+ PixyMod	27.14	0.886	0.083	39.41	2.48±0.13
+ PixyMod + AttnMuSF	26.60	0.879	0.090	33.19	1.80±0.11
+ PixyMod + AttnPixamp	**27.41**	0.891	**0.078**	33.18	1.94±0.12
+ PixyMod + AttnMuSF + AttnPixamp (APRNet)	27.09	**0.892**	0.084	**32.96**	**1.74±0.10**

4.4 Effect of attention-based multi-scale style fusion

In order to eliminate the misalignment-caused artifacts, we use the proposed AttnMuSF module to fuse multi-scale style encodings into style tensor. Ideally, at misaligned regions, AttnMuSF will pay more attention to larger scale style encoding.

As shown in Fig. 8, the rendering quality on the misaligned regions has been improved greatly. We also visualize the attention map of APRNet's AttnMuSF module on different encoding scales in Fig. 10. It's shown that if a text region in content is aligned to background one in style, AttnMuSF will assign more weights on $1 \times \frac{W}{H}$ encoding, otherwise local style encodings will be attended, which verifies that AttnMuSF works just as expected.

4.5 Effect of attention-based pixel sampling

However, a 2-stage "PixyMod+AttnMuSF" model still cannot handle complex background rendering well. As shown in Fig. 9, when the background styles become complex, both stages generate images with a blurry background. Therefore we replace the 1st stage with an AttnPixamp module. For each coordinate, we sample 25 pixels from its 17×17 neighborhood grid ($k = 5, m = 4$) uniformly, and aggregate these pixels by style-content cross attention to get render result. Fig. 9 shows that AttnPixamp can improve background rendering quality greatly. Intermediate results in Fig. 9 and Fig. 10 also show that with AttnPixamp focusing on background rendering, PixyMod can pay more attention to the foreground. This disentanglement effect improves final rendering quality and makes the output more photo-realistic.

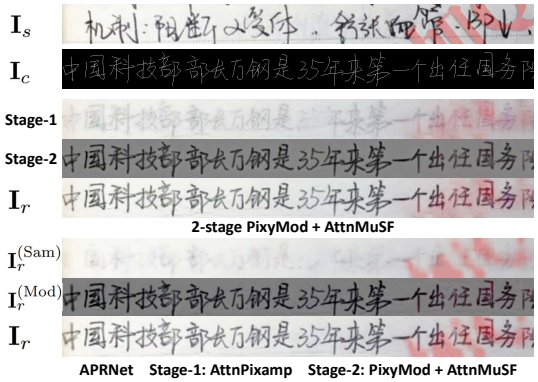

Fig. 9. Effect of AttnPixamp module. 2-stage "PixyMod+AttnMuSF" generates blurry background. Replacing the 1st stage with AttnPixamp, the background quality of rendered image can be improved.

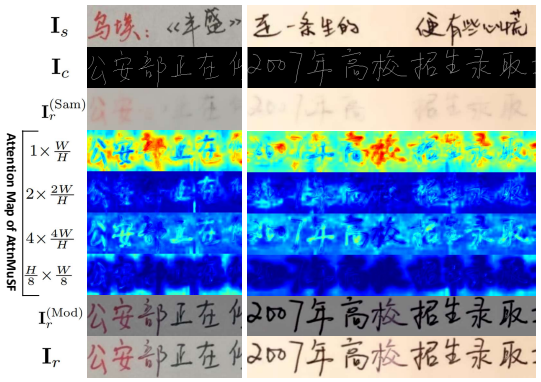

Fig. 10. APRNet's 1st stage AttnPixamp ($\mathbf{I}_r^{(\mathrm{Sam})}$) and 2nd stage "PixyMod+AttnMuSF" ($\mathbf{I}_r^{(\mathrm{Mod})}$) outputs, and the attention map visualizations of AttnMuSF in the 2nd stage. The AttnPixamp stage mainly focuses on background and the "PixyMod+AttnMuSF" stage focuses on content foreground. When a text region is aligned to background in style, AttnMuSF will pay more attention to the $1 \times \frac{W}{H}$ encodings.

4.6 Qualitative comparison

Fig. 11 shows the generation results of the APRNet and other models when inputting complex style images. The APRNet significantly improves the generation quality compared with the SC-GAN and the baseline model, and performs better than other models in terms of the details. Fig. 12 presents the synthesis results of APRNet using the same content image and different style images. Clearly, the APRNet is able to imitate style images with simple/uniform and complex/spatial-varying styles. The synthetic results in Fig. 12 demonstrate that the stroke width of handwritten text can also be transferred. However, there can be a "blob-like" artifact when the style image contains thick and dense text strokes. After careful analysis, we find that it is caused by the AttnPixamp module. Because we are using uniform-sampled pixels as candidates for each pixel in content bank, there are chances when the pixel in content bank is background while all sampled candidates in the style image lie in foregrounds. This issue can be solved if the candidates are sampled adaptively. We leave this as our future works.

4.7 Quantitative comparison

Finally, we compare these models using quantitative measurements. Results are listed in Tab. 1. Besides "PixyMod", "PixyMod+AttnMuSF" and APRNet, we also list the result of "PixyMod+AttnPixamp" in which we remove the AttnMuSF in APRNet. According to Tab. 1, the PSNR, SSIM, and LPIPS scores are greatly improved using PixyMod. It indicates that using PixyMod, models are able to imitate

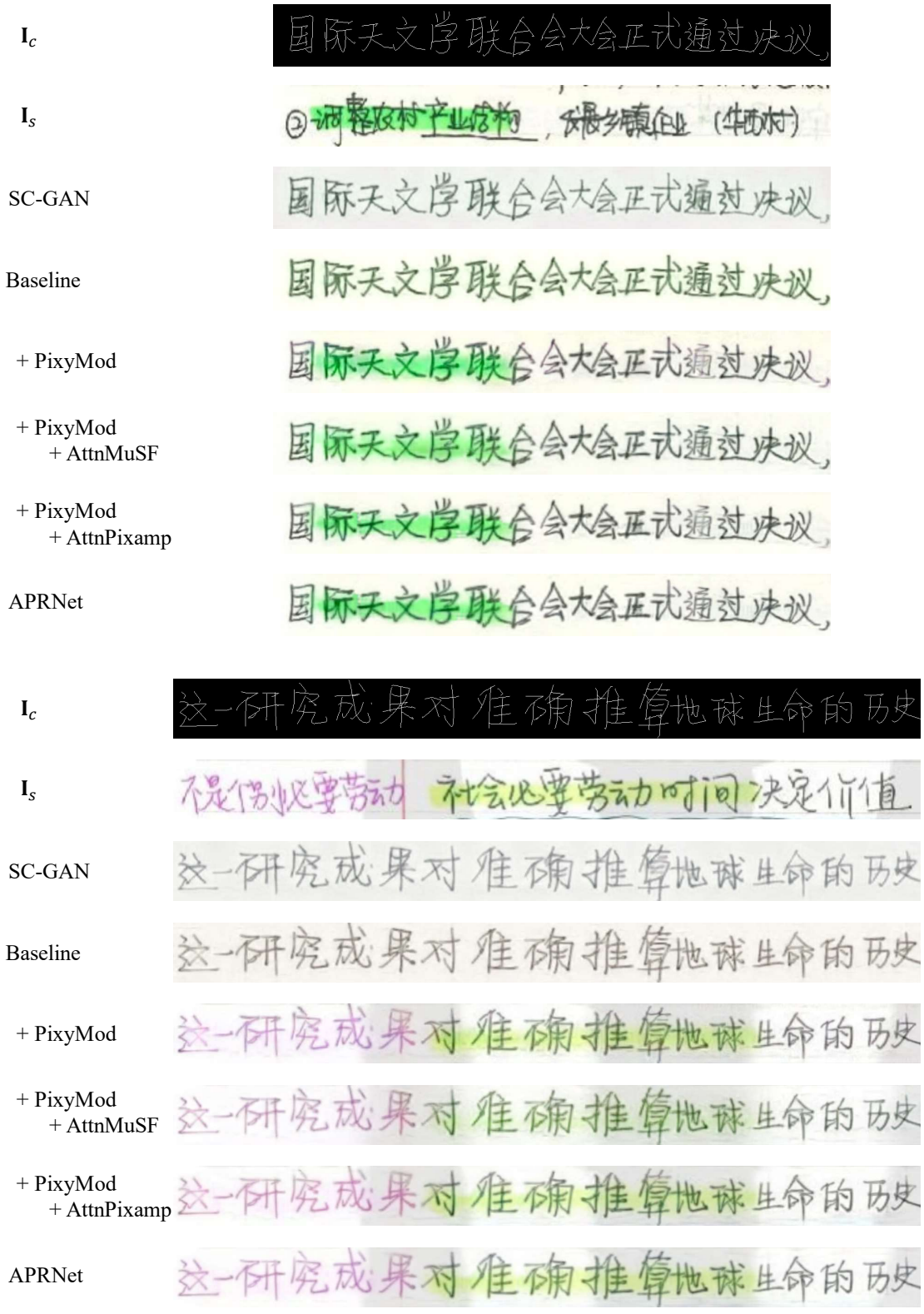

Fig. 11. Additional results with comparison to other models.

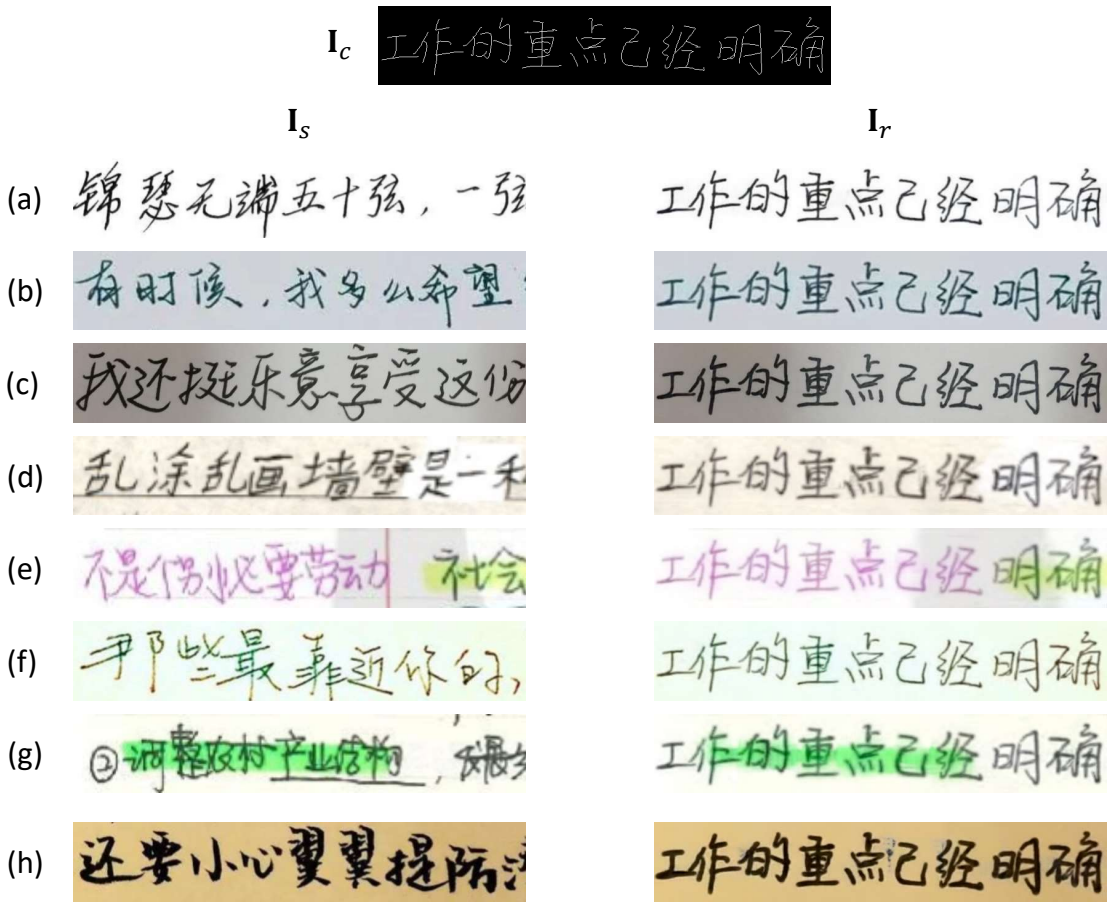

Fig. 12. Additional results of APRNet using the same content image and different style images. APRNet is able to imitate reference images with (a-b) simple and uniform foreground and backgrounds, (c-f) spatial-varying styles, and (g) complex backgrounds. However, there can be a "blob-like" artifact cause by AttnPixamp, when all candidates pixels from style images of AttnPixamp lie in foregrounds.

the spatial-varying styles in reference style images much better, which achieve lower reconstruction differences. The FID and KID scores on the testing set are improved a lot using AttnMuSF and AttnPixamp. It shows that using these two methods, the generated images are more photo-realistic and visually similar to corresponding style reference. Combining PixyMod, AttnMuSF and AttnPixamp, our proposed APRNet achieves much better results than SC-GAN and baseline models.

5 Limitations

Our APRNet still has the following limitations:

1) Sharp edges in background are not rendered very well;
2) Memory requirement of APRNet is pretty large;
3) *Single Crop* strategy releases us from collecting matched tuple images for training, but it relies on high quality binarization/skeletonization techniques to achieve satisfactory results;
4) APRNet can only transfer color patterns currently. It can not imitate style image's typography/writing styles, spatial transformations/deformations, etc.

6 Conclusion

We propose an APRNet equipped with AttnPixamp, PixyMod and AttnMuSF modules to transfer spatial-varying color patterns from style text image to content image and a *Single Crop* strategy to train this model. Experimental results on Chinese handwriting text image synthesis show that our design can improve the quality of synthetic text images and make them more photo-realistic.

References

[1] M. Guan, H. Ding, K. Chen, and Q. Huo, "Improving handwritten ocr with augmented text line images synthesized from online handwriting samples by style-conditioned GAN," in *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2020, pp. 151–156.

[2] P. Krishnan, R. Kovvuri, G. Pang, B. Vassilev, and T. Hassner, "Textstylebrush: Transfer of text aesthetics from a single example," *arXiv preprint arXiv:2106.08385*, 2021.

[3] L. Kang, P. Riba, M. Rusinol, A. Fornes, and M. Villegas, "Content and style aware generation of text-line images for handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[4] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic data and artificial neural networks for natural scene text recognition," in *Workshop on Deep Learning, Proceedings of NIPS*, 2014.

[5] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proceedings of CVPR*, 2016, pp. 2315–2324.

[6] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, 2016.

[7] L. Wu, C. Zhang, J. Liu, J. Han, J. Liu, E. Ding, and X. Bai, "Editing text in the wild," in *Proceedings of the 27th ACM international conference on multimedia*, 2019, pp. 1500–1508.

[8] W. Shimoda, D. Haraguchi, S. Uchida, and K. Yamaguchi, "De-rendering stylized texts," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 1076–1085.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[10] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[11] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.

[12] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural style transfer: A review," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 11, pp. 3365–3385, 2019.

[13] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1501–1510.

[14] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[15] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[16] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[17] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[18] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in neural information processing systems*, 2017, pp. 700–708.

[19] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 172–189.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[21] H. Zhang, L. Liang, and L. Jin, "Scut-hccdoc: A new benchmark dataset of handwritten chinese text in unconstrained camera-captured documents," *Pattern Recognition*, p. 107559, 2020.

[22] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang, "Casia online and offline chinese handwriting databases," *2011 International Conference on Document Analysis and Recognition*, pp. 37–41, 2011.

[23] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, "Alias-free generative adversarial networks," *arXiv preprint arXiv:2106.12423*, 2021.

[24] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[25] L. Ma, X. Jia, S. Georgoulis, T. Tuytelaars, and L. Van Gool, "Exemplar guided unsupervised image-to-image translation with semantic consistency," *Proceedings ICLR 2019*, 2019.

[26] Y. Wang, L. Qi, Y.-C. Chen, X. Zhang, and J. Jia, "Image synthesis via semantic composition," in *ICCV*, 2021.

[27] S. Liu, T. Lin, D. He, F. Li, M. Wang, X. Li, Z. Sun, Q. Li, and E. Ding, "Adaattn: Revisit attention mechanism in arbitrary neural style transfer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6649–6658.

[28] P. Zhang, B. Zhang, D. Chen, L. Yuan, and F. Wen, "Cross-domain correspondence learning for exemplar-based image translation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5143–5153.

[29] X. Zhou, B. Zhang, T. Zhang, P. Zhang, J. Bao, D. Chen, Z. Zhang, and F. Wen, "Cocosnet v2: Full-resolution correspondence learning for image translation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 11 465–11 475.

[30] H. Tang, S. Bai, and N. Sebe, "Dual attention gans for semantic image synthesis," in *ACM MM*, 2020.

[31] H. Tang, D. Xu, N. Sebe, Y. Wang, J. J. Corso, and Y. Yan, "Multi-channel attention selection gan with cascaded semantic guidance for cross-view image translation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2417–2426.

[32] B. AlBahar and J.-B. Huang, "Guided image-to-image translation with bi-directional feature transformation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[33] D. Y. Park and K. H. Lee, "Arbitrary style transfer with style-attentional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[34] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[35] J. Gan and W. Wang, "Higan: Handwriting imitation conditioned on arbitrary-length texts and disentangled styles," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 7484–7492.

[36] A. K. Bhunia, S. Khan, H. Cholakkal, R. M. Anwer, F. S. Khan, and M. Shah, "Handwriting transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 1086–1094.

[37] J. Zdenek and H. Nakayama, "Jokergan: Memory-efficient model for handwritten text generation with text line awareness," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 5655–5663.

[38] S. Fogel, H. Averbuch-Elor, S. Cohen, S. Mazor, and R. Litman, "Scrabblegan: Semi-supervised varying length handwritten text generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4324–4333.

[39] L. Kang, P. Riba, Y. Wang, M. Rusiñol, A. Fornés, and M. Villegas, "Ganwriting: Content-conditioned generation of styled handwritten word images," in *European Conference on Computer Vision*. Springer, 2020, pp. 273–289.

[40] A. Mattick, M. Mayr, M. Seuret, A. Maier, and V. Christlein, "Smartpatch: Improving handwritten word imitation with patch discriminators," in *Document Analysis and Recognition – ICDAR 2021*, J. Lladós, D. Lopresti, and S. Uchida, Eds. Cham: Springer International Publishing, 2021, pp. 268–283.

[41] A. K. Bhunia, S. Ghose, A. Kumar, P. N. Chowdhury, A. Sain, and Y.-Z. Song, "Metahtr: Towards writer-adaptive handwritten text recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 830–15 839.

[42] S. Park, S. Chun, J. Cha, B. Lee, and H. Shim, "Multiple heads are better than one: Few-shot font generation with multiple localized experts," in *International Conference on Computer Vision (ICCV)*, 2021.

[43] J. Cha, S. Chun, G. Lee, B. Lee, S. Kim, and H. Lee, "Few-shot compositional font generation with dual memory," in *European Conference on Computer Vision (ECCV)*, 2020.

[44] S. Park, S. Chun, J. Cha, B. Lee, and H. Shim, "Few-shot font generation with localized style representations and factorization," in *AAAI Conference on Artificial Intelligence*, 2021.

[45] S. Yang, Z. Wang, and J. Liu, "Shape-matching gan++: Scale controllable dynamic artistic text style transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

[46] S. Yang, J. Liu, W. Yang, and Z. Guo, "Context-aware text-based binary image stylization and synthesis," *IEEE Transactions on Image Processing*, vol. PP, no. 99, pp. 1–13, 2018.

[47] S. Yang, W. Wang, and J. Liu, "Te141k: Artistic text benchmark for text effect transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[48] S. Yang, Z. Wang, Z. Wang, N. Xu, J. Liu, and Z. Guo, "Controllable artistic text style transfer via shape-matching gan," in *International Conference on Computer Vision*, 2019.

[49] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[50] T.-Q. Wang and C.-L. Liu, "Fully convolutional network based skeletonization for handwritten chinese characters," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr.

2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11868
[51] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of ICLR*, 2015.
[52] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*. IEEE, 2010, pp. 2366–2369.
[53] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018.
[54] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.
[55] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD GANs," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=r1lUOzWCW

APPENDIX A
ADDITIONAL IMPLEMENTATION DETAILS

We add more details of all systems we compared including StyleGANv2-based baseline system (Fig. 13), "Baseline+PixyMod" system (Fig. 14), "Baseline+PixyMod+AttnPixamp" system (Fig. 15) and "Baseline+PixyMod+AttnMuSF" system (Fig. 16).

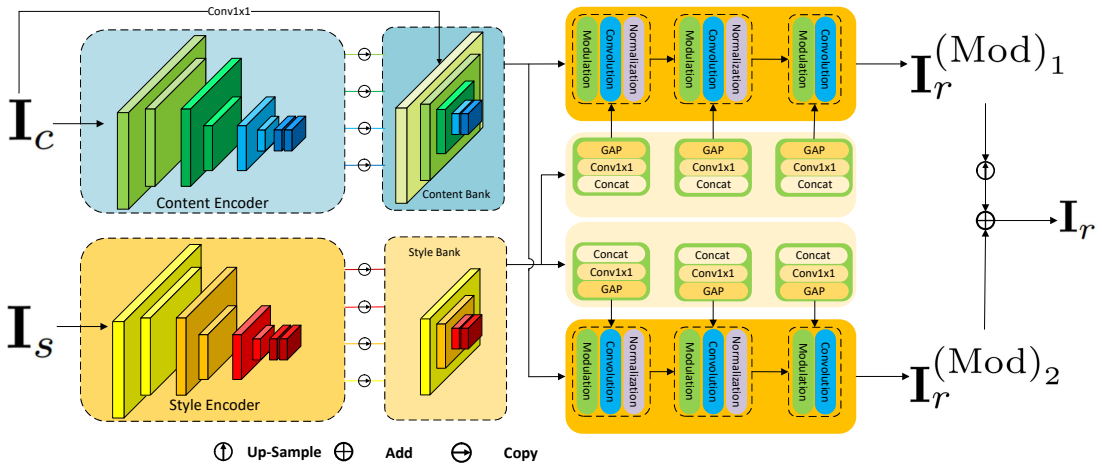

Fig. 13. StyleGANv2-based baseline system. This system consists of 2 rendering stages, which will generate $\frac{H}{2} \times \frac{W}{2}$ and $H \times W$ images respectively. The style **vectors** used for modulation and normalization are generated by 1) down/up-sampling all feature maps in style bank to same size, 2) concatenating them to form a raw style tensor, 3) feeding this tensor to a 1×1 conv layer to fit the channel number of content tensors, and 4) generating style vectors through global average pooling (**GAP**).

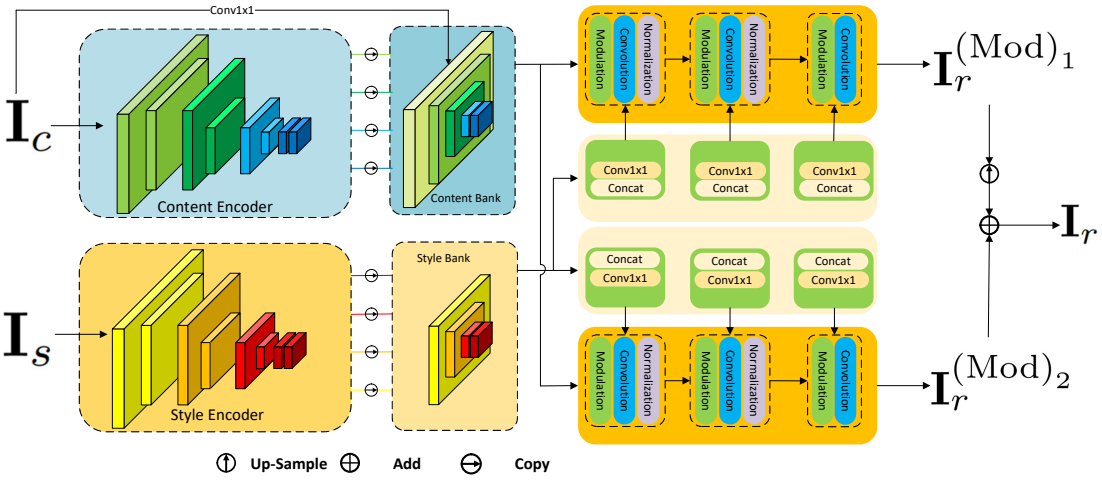

Fig. 14. "Baseline+PixyMod" system. Compared with Fig. 13, this system replaces style vectors with style tensors. It is implemented by abandoning the GAP operations.

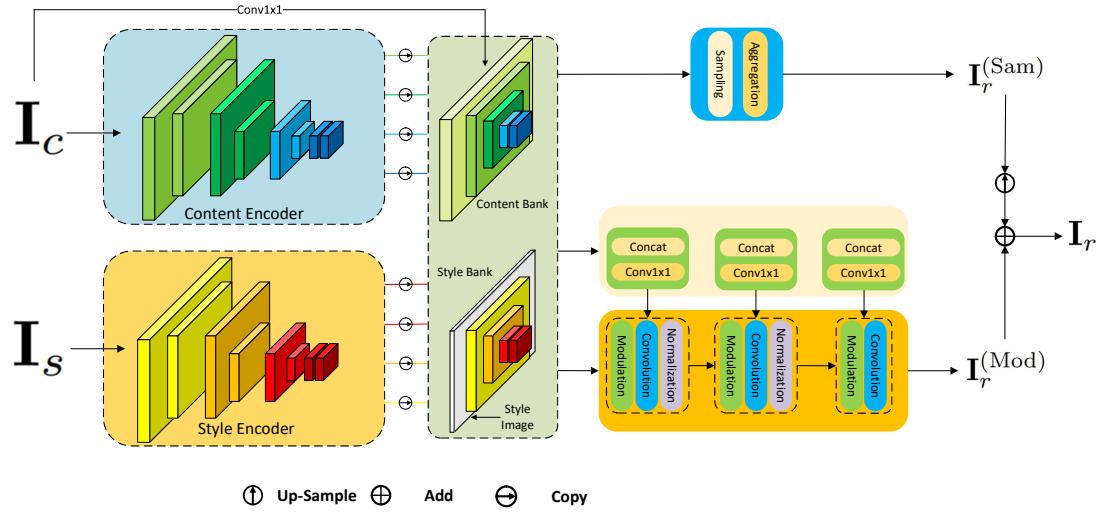

Fig. 15. "Baseline+PixyMod+AttnPixamp" system. Compared with Fig. 14, this system replaces first stage with AttnPixamp Module.

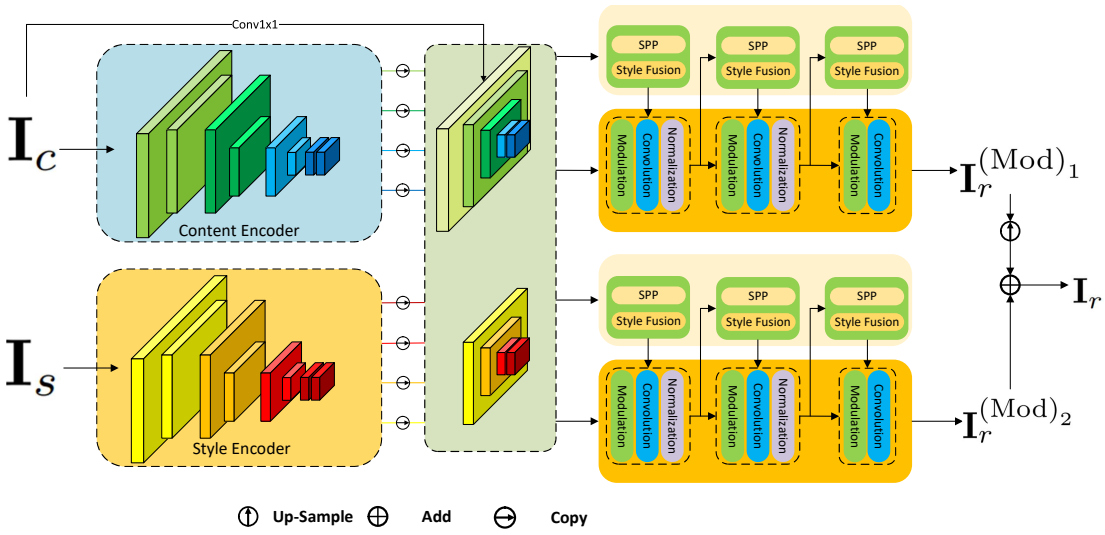

Fig. 16. “Baseline+PixyMod+AttnMuSF” system. Compared with Fig. 14, this system generates style vectors via AttnMuSF module.