ECTI Transactions on Computer and Information Technology
Journal homepage: https://ph01.tci-thaijo.org/index.php/ecticit/
Published by the ECTI Association, Thailand, ISSN: 2286-9131

Skeleton-based Generative Adversarial Networks and Novel Evaluation Metrics for Font Structural Style Transfer

Thanaphon Thanusan[1] and Karn Patanukhom[2]

ABSTRACT

We present a novel approach for font style transfer using Generative Adversarial Networks (GANs) to enhance the scene text editing process, enabling text editing from any characters to any characters, including cross-language editing. Our GAN model utilizes pairs of sample images of the target font style and the corresponding skeleton-based features to learn their key structural details without relying on pre-trained models. Once the generator is trained, it can transform any character from the base font style to the target font style. Our approach offers the flexibility to select a base font similar to the target font for enhancing results and the ability to manipulate the stroke width of the output text. Additionally, in few-shot scenarios, we introduce a double generator scheme that integrates other existing methods with our approach. In this work, we also introduce two new evaluation metrics: Difference in Histogram of Oriented Gradients and Stroke Width Similarity. Our experimental results demonstrate that the proposed evaluation metrics can better measure font style similarity with greater robustness compared to conventional metrics. We evaluate the performance of our GAN model on style transfer for six target fonts and real scene text editing tasks, comparing it with existing methods. Our approach provides better structural similarity, readability, and visual appeal than other methods, especially for generating unseen characters.

Article information:

Keywords: Deep Learning, Generative Adversarial Network, Font Style Transfer, Font Similarity Measure, Skeleton

Article history:
Received: December 30, 2024
Revised: June 8, 2025
Accepted: August 7, 2025
Published: August 9, 2025 (Online)

DOI: 10.37936/ecti-cit.2025193.260297

1. INTRODUCTION

Scene text editing is the process of manipulating text that appears in photographs. It is primarily applied in augmented reality applications, such as translating signboards or labels in real-time using a smartphone. Additionally, it can be used for language localization tasks in the content of images or videos. However, beyond merely conveying literal meaning, font style is an integral element that communicates emotion, establishes brand identity, and ensures aesthetic consistency. For instance, in applications like media localization for films or games, simply translating text without preserving the original font's unique structural and textural style can disrupt the intended artistic expression and visual immersion. While human editors may perform this task using photo editing software, editing text in photographs remains a challenging task that heavily relies on personal skill and manual effort.

The scene text editing task can be roughly considered as three sub-problems: (1) text localization, (2) text style transferring, and (3) background generation. Text localization involves segmenting the target text in an image, distinguishing it from the background for replacement. Text style transferring is a process that creates a new text image with the same spatial transformation, structural, and textural styles as the original segmented text. Background generation is a process used to generate appropriate backgrounds or inpaint missing backgrounds in the original text area. Recently, various research efforts have aimed to reduce the need for personal skill and manual effort in scene text editing tasks. These include advancements in text localization processes [1-3], style transfer processes [4-6], background

[1,2]The authors are with The Department of Computer Engineering, Faculty of Engineering, Chiang Mai University, Thailand, Email: thanaphon_th@cmu.ac.th and karn_patanukhom@cmu.ac.th

[2]Corresponding author: karn_patanukhom@cmu.ac.th

generation processes [7], and end-to-end processes [7-9]. Recent GAN-based style transfer works increasingly focus on structure-aware and cross-domain tasks. HyperGAN-CLIP supports zero-shot synthesis via domain adaptation [10], while GAN-based text augmentation improves stylistic diversity [11]. Multilingual and low-resource applications [12] further affirm the value of structural approaches like ours.

In this study, we focus on the style transfer task. The goal of text style transfer is to change the appearance of characters from a base style to a specific target style. The differences between any two text styles can be described by three factors: (1) spatial transformation due to camera position, (2) structural style determined by font shape and stroke size, and (3) textural style influenced by text color and filling pattern. After exploring the style transfer task, we identified some limitations in existing approaches related to structural style transfer problems. The first limitation is the performance degradation when these methods are used to transfer the target style from unseen characters or to unseen characters. Because most existing style transfer approaches rely on convolutional neural networks (CNNs) [13], which require a large number of samples for learning patterns from characters in the training set (seen characters), utilizing such models to transfer the target style to or from characters not included in the training set (unseen characters) may lead to the generation of inaccurate images.

The second limitation concerns the evaluation metrics used to assess the performance of the structural style transfer process. Most existing style transfer approaches utilize Mean Absolute Error (MAE) and Structural Similarity Index Measure (SSIM) as objective evaluation metrics. These metrics do not effectively measure the style similarities or dissimilarities between different characters and between characters with different scales or positions.

The objective of this research is to enhance a scene text editing procedure to enable editing text from any characters to any characters, including cross-language editing, by refining the font structural style transfer process. Additionally, novel evaluation metrics are developed to validate the effectiveness of font structural style transfer processes.

In our previous study [5], we introduced skeleton-based features as inputs to Generative Adversarial Networks (GANs) [14] for the task of font structural style transfer. In this paper, we provide a deeper and more detailed analysis of our skeleton-based GAN models. This includes an experiment on stroke width manipulation, an experiment on varying the base font, and an experiment using different sets of given style samples. We increased the number of target styles to be tested from four to six and compared our method with additional existing methods. We also conducted tests on real scene text editing tasks using real-world images. Moreover, in this paper, we also investigate models that combine conventional methods with our proposed methods, referred to as a double generator scheme. As the final major contribution of this paper, we present two new evaluation metrics, Difference in Histogram of Oriented Gradients ($\triangle$HOG) and Stroke Width Similarity (SWS), for measuring the style similarities/dissimilarities of character images. To summarize, this paper presents two key contributions:

- A novel Skeleton-based GAN framework that enables robust style transfer, especially for unseen characters.
- The development and justification of novel evaluation metrics ($\triangle$HOG, SWS) specifically tailored for evaluating font structural style transfer.

This paper is structured as follows: Section 2 introduces relevant existing research related to this work, Section 3 presents the proposed font structural style transfer process and evaluation metrics, Section 4 provides details of our experiments and their corresponding results, and Section 5 concludes this work.

2. RELATED WORK

One of the popular techniques used in image generation tasks is the GAN [14], a framework for creating generative models proposed by Goodfellow *et al.* in 2014. GAN consists of two components: a generator network and a discriminator network. The generator's objective is to create realistic images by using a random noise as input to the network, while the discriminator's objective is to differentiate between real images and those generated by the generator. In the learning phase, these two networks are trained simultaneously. The network parameters of the discriminator are updated to minimize its loss function (increase recognition accuracy), while the network parameters of the generator are updated in the opposite direction, aiming to increase the discriminator's loss.

In the original GAN paper [14], GAN was tested on three datasets: MNIST [15] for generating digits 0-9, TFD [16] for generating human faces, and CIFAR-10 [17] for generating animals or objects such as birds, cats, dogs, airplanes, and ships. The evaluation was conducted using Parzen window-based log-likelihood. While GANs can produce satisfactory results with inputs within the learned subspace, they may encounter problems with unseen data mapped outside this space, resulting in poor outcomes [18].

There have been numerous research studies on GANs in recent years [19], including applications such as text generation, text style transfer, and scene text editing. Multi-content GAN (MC-GAN) [20], proposed in 2017 by Azadi *et al.*, is a research study that employs Conditional GAN (CGAN) [21] and PatchGAN [22] for a few-shot font style transfer task. MC-GAN divides text style transfer tasks into two mod-

ules: a glyph network and an ornamentation network. To train the MC-GAN, we need to determine the set of all possible characters (glyphs) that can be generated and collect those characters in various font styles. The function of the glyph network is to receive a stack of given grayscale character samples in the target style, along with blank images for missing characters, as input, and produce grayscale images for all missing characters as output. The glyph network learns the relationship between characters and automatically transfers their style to the missing characters across channels of the input stack. On the other hand, the ornamentation network takes the grayscale images output from the glyph network as its input and generates colored character images for each missing character according to the desired style. In their experiment, the MC-GAN model was trained using 20,000 color images of 26 uppercase English characters. SSIM was applied to investigate the dependency of the quality of model predictions on the number of given style samples. The distribution of SSIM score shows that a stable point is close to six samples. In addition, a perceptual evaluation was conducted on 33 font examples. Eleven people were asked to choose the preferred synthetic character sets from MC-GAN and the other method. The perceptual evaluation shows that the results from MC-GAN are preferred 80.0% of the time. The limitation of MC-GAN is that the network is fixed to the set of possible characters used as the style samples or generated as the outputs. For instance, if the MC-GAN model is trained with a set of 26 uppercase English characters, it can only handle those characters and cannot transfer the style from or to other characters such as lowercase characters, digits, or characters in other languages.

STEFANN [8], proposed by Roy *et al.*, is another work focused on the text style transfer task. Similar to MC-GAN, STEFANN is composed of two generative networks: FANnet for shape transfer and Colornet for color transfer. STEFANN is a non-GAN architecture. It trains two generative networks by minimizing MAE, without employing adversarial loss as in MC-GAN. Consequently, there are no discriminators in STEFANN. The inputs of FANnet are a single style sample and a one-hot vector representing the target character class to be generated. The limitation of STEFANN is similar to that of MC-GAN in that STEFANN cannot generate characters outside the predefined set of character classes.

In 2019, Artistic Glyph Image Synthesis Network (AGIS-Net) [4] was proposed by Gao et al. AGIS-Net is a GAN-based few-shot learning model for text style transfer. The model comprises one generator and three discriminators. The model consists of a generator and three discriminators. Within the generator, there are two parallel encoders: a context encoder and a style encoder. It also includes two parallel decoders: a shape decoder and a texture decoder. The context encoder extracts features from a target character in a base (standard) font style, while the style encoder extracts features from given character samples in the target style. The features extracted from both encoders are input to the shape decoder to generate a grayscale character image. Then, the output from the shape decoder, along with the features from the encoders, is input to the texture decoder to generate a desirable colored character image. Three discriminators, including a shape discriminator, a texture discriminator, and a local discriminator, are used to calculate losses. AGIS-Net can be pre-trained with various characters in different styles and then fine-tuned with a specific target text style using only a few given characters. In the experiment, evaluation metrics including Inception Score [23], Fréchet Inception Distance [24], SSIM, and pixel-level accuracy are used to evaluate the performance of AGIS-Net. The results show that AGIS-Net outperforms similar models such as MC-GAN. The researchers found that with more training samples, the quality of the synthesized images improves significantly across various content input font styles. The limitation of AGIS-Net is its performance with unseen characters (outside of the pre-trained samples), either in the given style samples or in the target characters to be generated as the output. Fig. 1 illustrates the issues encountered when the model attempts to generate unseen characters. In this example, the model is pre-trained by a set of English characters. In this example, the model is pre-trained on a set of uppercase English characters. It provides good results when generating the text "HAPPY," which consists of all seen characters, but the results are worse when generating numeric text like "1234" or Thai text which are unseen characters.

Shape Matching GAN [6], proposed by Yang *et al.*, is another framework that utilizes GAN for artistic text style transfer. Shape Matching GAN can transfer any structural and textural styles from non-text images, such as fire, ice, or leaves, to any text image. The process of Shape Matching GAN also involves two GANs: the glyph network and the texture network. Shape Matching GAN can learn shape deformation information from style input and transfer it to target text images with an adjustable stylistic degree of the glyph.

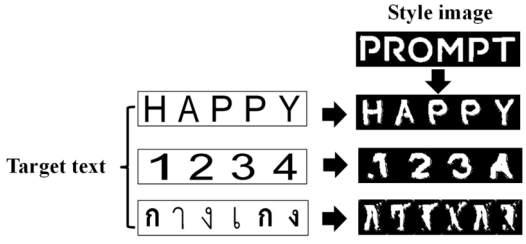

Fig.1: *Samples of failed cases of AGIS-Net.*

Wu *et al.* [7] studied an end-to-end trainable scene text editing model designed to modify a word in the source image with another word, while preserving its realistic appearance. They proposed a style retention network (SRNet) consisting of three modules: a text conversion network, a background inpainting network, and a fusion network. The role of the text conversion network is the same as MC-GAN, STEFANN, and AGIS-Net, which is to transfer the style from the reference text to another text. The structure of SRNet's text conversion network is similar to AGIS-Net, as it consists of two parallel encoders and two parallel decoders. For SRNet, the text conversion network not only learns to transfer font structure and color, but also geometric deformations. The background inpainting network, which is structured like a U-Net, is utilized for removing text and generating a pure background image. The generated foreground text and background images are input into the fusion network to create the final output image. Two PatchGAN discriminators are applied for training the background inpainting network and the fusion network. This method allows for training the model for cross-language editing. The experiment is conducted on English-to-Chinese translation tasks, where large amounts of English and Chinese fonts are used as training data. The results demonstrate that the font style, color, geometric deformation, and background texture can be accurately transferred to the synthetic Chinese characters. However, the performance of SRNet also drops when handling unseen characters in the training data or when the lengths of the original text in the scene and the target text to be replaced are significantly different. Fig. 2 shows the example of SRNet's failure cases. This SRNet model is trained on an English training set. From these examples, we can see that the model can perform well in editing "Cream" to "HAPPY" because both of them are in English and their lengths are similar. On the other hand, its performance is lower when editing to unseen text (numbers), like changing "Cream" or "plus" to "1234", or when editing to text with different length like changing "plus" to "HAPPY".

In 2020, Yang *et al.* [9] proposed a new scene text editing model named SwapText. The model consists of three modules similar to SRNet, but it replaces the text conversion network with a text swapping network. In the text swapping network, control points representing geometrical attributes of text are extracted from the style image. Using the geometrical attributes of the style image, a transformation is applied to the content image. Two encoders are utilized to extract feature maps from both the style image and the geometrically transformed content image. A self-attention network is employed to learn the correspondences between the two feature maps. The output of the self-attention network is then passed through the decoder to obtain the final output of the text swapping network.The researchers present experimental results indicating that SwapText can enhance the Mean Squared Error (MSE) and SSIM of synthetic English and Chinese texts. However, the problem of dealing with unseen characters still remains a challenge in this approach.

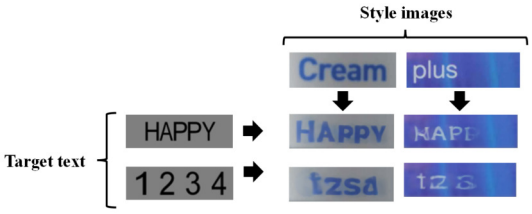

Fig.2: *Samples of failed cases of SRNet.*

In 2023, we introduce a method that employs skeleton-based features as inputs to a GAN model for font structural style transfer [5]. This approach enables the transfer of local structures from or to unseen characters or characters in different languages. Furthermore, we incorporate an evaluation technique using the Optical Character Recognition (OCR) tool to assess the model's performance. The accuracy of character recognition (OCRA) in the generated text helps determine how accurately the model generates characters while maintaining the stylistic elements of the target font. In the paper, we present preliminary results comparing the performance of the skeleton-based GAN with AGIS-Net for generating unseen characters. The results demonstrate that the skeleton-based GAN outperforms AGIS-Net in terms of readability for generating unseen characters.

To better position our proposed method within the existing landscape of text style transfer, Table 1 provides a detailed comparative analysis of its key features against prominent prior works, including MC-GAN, AGIS-Net, and SRNet. This comparison highlights crucial distinctions in types of input and output for the style transfer models, sensitivity to target text and style sample length variations, and capabilities regarding both seen and unseen characters, as well as their respective large dataset requirements. Both MC-GAN and AGIS-Net, similar to our approach, operate on a character-wise generation model, which offers fine-grained control over individual characters. SRNet, conversely, focuses on word or phrase-level generation, which can introduce sensitivity to varying text lengths. All surveyed methods, achieve high to very high performance on seen characters, indicating a strong baseline capability across the board for familiar text. However, the major limitations of existing methods, and consequently the core contributions of our work, become evident when considering unseen characters and dataset requirements. Prior works like MC-GAN have no ability to generate characters outside their predefined set, which is fixed by the network architecture. AGIS-Net and SRNet, while offering

some degree of generalization, still exhibit low performance on unseen characters. Moreover, all these methods traditionally require a large dataset for effective training. In contrast, our proposed method makes a significant academic contribution by demonstrating high performance on unseen characters without need for a large pre-trained dataset.

Table 1: *A comparative summary of existing text style transfer methods and the proposed approach.*

Feature	MC-GAN	AGIS-Net	SRNet	This work
Input and output	Character	Character	Word or Phrase	Character
Sensitivity to text length variation	Low	Low	High	Low
Performance on seen characters	Very high	Very high	Very high	High
Performance on unseen characters	No	Low	Low	High
Need for a large dataset	Yes	Yes	Yes	No

3. PROPOSED METHOD

As mentioned earlier, the model used in this research is the skeleton-based GAN developed in our previous work [5]. The process of obtaining results involves two distinct phases. Firstly, the training phase aims to develop a robust generator model capable of producing the desired outputs. Secondly, the inference phase involves using the trained generator model to generate the final output based on the given inputs. This model is specifically trained to generate a particular target font style without relying on any pre-trained models. To train the model to generate the target font style, we need to prepare a set of samples of character image pairs $(B_c, T_c)|c \in C$ where T_c and Bc are the binary images of the character c in the target style and the base style, respectively. C is a set of given characters in the training process. The target style refers to the structural style of the font we aim to generate, while the base style refers to another font style used as the input to the generator. In the first step of the training process, all target font images Tc are input into a skeleton extraction module to extract skeleton-based features $S(T_c)$. The set of $\{(S(T_c), T_c)|c \in C\}$ is utilized to train the generator G while the set of $\{(G(S(T_c)), 0)$ and $(T_c, 1)|c\}$ is utilized to train the discriminator D. The learning process is illustrated in Fig. 3.

After the generator is trained for the specific font style, we can use it to transform any character from the base font style to the target font style using the pipeline illustrated in Fig. 4. We incorporate a resizing module to adjust the size of a target text image before extracting the skeleton-based features. Variations in the size of the input image can result in differences in the stroke width of the output image once it is processed by the model. This resizing module allows us to manipulate the stroke width of the output image by inputting a stroke width adjustment parameter s.

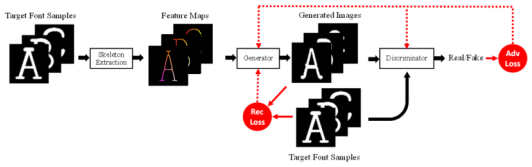

Fig.3: *Learning process of the proposed skeleton-based GAN for font structural style transfer.*

Next, the skeleton-based features are input into the generator, which produces the character image in the target font style as the output. The final output is obtained by resizing the generator's output to the original scale.

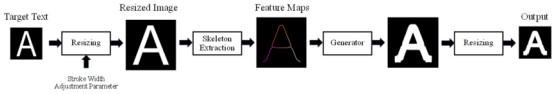

Fig.4: *Inference process of the proposed skeleton-based GAN for font structural style transfer.*

3.1 Model Architecture

The proposed GAN (Fig. 5) includes a generator G and a PatchGAN discriminator D. G (3.83M params) adopts an encoder–decoder from Shape Matching GAN [14], with stride-2 convolutions and transposed convolutions for structural encoding and glyph reconstruction, plus skip connections for fine detail. D (0.70M params) applies six convolutions to classify local patches, encouraging sharp edges and junctions. This setup ensures style realism and structural fidelity, supporting generalization to unseen characters without pretraining.

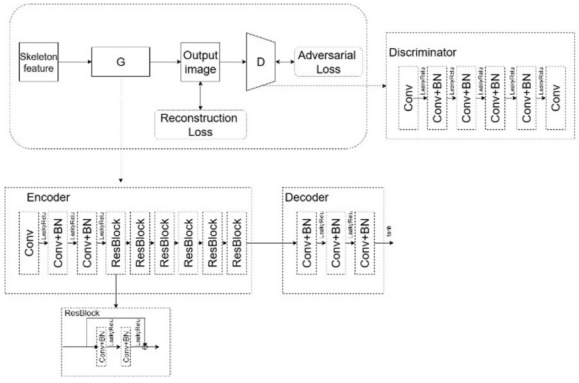

Fig.5: *Network architecture of our model.*

3.2 Skeleton-based Feature Extraction

We extract four skeleton-based features from character images in a base font and utilize them as inputs for the generator. These features, as described in [5], include: (1) skeleton image preserves the medial axis for core structure, (2) normalized vertical position, (3) normalized horizontal position maps encode stroke placement and proportions, and (4) line-end position map marks stylistic stroke terminations. The skeleton image can be obtained using a morphological thinning operator. Example of skeleton-based features is illustrated in Fig. 6.

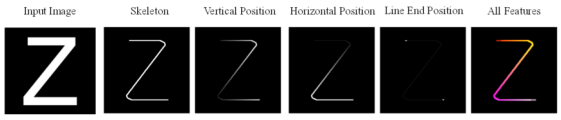

Fig.6: *Visualization of the skeleton-based features.*

3.3 Loss Function

Two loss components are used for training our GAN model: (1) an adversarial loss L_A, promoting realistic style and stroke continuity; and (2) a reconstruction loss L_R, using L_1 distance to preserve character structure. which are defined as:

$$L_A = \underset{c\in C}{E}\{\log(D(T_c)\} + \underset{c\in C}{E}\{\log(1 - D(G(S(T_c))))\} \tag{1}$$

$$L_R = \underset{c\in C}{E}\{\|G(S(T_c)) - T_c\|_1\} \tag{2}$$

where G is a generator, D is a discriminator, C is a set of given characters (target style) used in the training process, T_c is an image of a character c in C, S represents the skeleton extraction module, and $E\{\}$ denotes expectation. The optimal generator and discriminator are defined as $\min_G \max_D (\lambda_A L_A + \lambda_R L_R)$ where λ_A and λ_R are tune-able loss weights for optimization. This combined loss guides the model to preserve style and structure, improving generalization to unseen characters.

3.4 Image Resizing

As shown in Fig. 4., the inference pipeline incorporates image resizing processes at both the initial and final stages. These resizing modules serve two primary purposes: (1) adjusting the aspect ratio to match the target style and (2) manipulating the stroke width.

Due to the mismatch in aspect ratios between the target style characters and the base style characters, it is necessary to resize the input image (base style) to match the aspect ratio of the target style. To achieve this, an aspect ratio scaling parameter r is calculated as follows:

$$r = \frac{1}{|C|}\sum_{c\in C}\frac{W(t_c)H(B_c)}{H(T_c)W(B_c)} \tag{3}$$

where $|C|$ denotes the number of given characters in the training process. $W(I)$ and $H(I)$ refer to the width and height of the bounding box enclosing characters in the image I. While maintaining the same height of character's bounding box, we use the parameter r to adjust the width of the base image, ensuring its aspect ratio matches that of the style image.

The second resizing parameter is a stoke width adjustment parameter s. One feature of the generator in the proposed model is that it can accept inputs of varying sizes. Using a larger character size as the input to the generator results in narrower stroke widths relative to the character size. Conversely, using a smaller character as the input to the generator leads to wider stroke widths. Based on this characteristic, we can adjust the stroke width by a factor of s while preserving the overall appearance of the font characters by resizing the generator input to $1/s$ times its original size.

By combining these two resizing parameters, the resizing process can be described as follows:

$$a = \frac{H_M}{SH_I} \tag{4}$$

$$J = R_{(a^{-1},a^{-1})}(G(S(R_{r\cdot a,a}(I)))) \tag{5}$$

where I is the input image of the inference process with a shape of $W_I \cdot H_I$, J is the final output of the inference process with a shape of $rW_I \cdot H_I$, $W_M \cdot H_M$ is the size of sample images (B_c and T_c) used in training process, and $R_{(w,h)}$ represents an image resizing function with a scaling factor w in the horizontal direction and h in the vertical direction. Fig. 7 demonstrates samples of the generated images obtained by varying the stoke width adjustment parameter s.

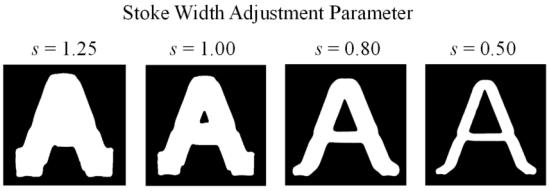

Fig.7: *Visualization of the effect of the stroke width adjustment parameter s.*

3.5 Double Generator Scheme

In practical situations, there may be only a few samples available for a model to learn a target style. The performance of the proposed skeleton-based model, as illustrated in Fig. 8, may be reduced due to the limited number of available samples. Let's refer to our proposed skeleton-based style

transfer method, described from the beginning until now, as a single generator scheme. In this paper, we also develop another scheme, especially designed for few-shot learning situations, referred to as a double generator scheme. In this scheme, the process consists of two generators as shown in Fig. 9. The first generator (Generator 1) is designed based on conventional methods such as MC-GAN, a conditional GAN that learns joint content-style features; AGIS-Net, which encodes shape and texture separately for flexible adaptation; and SRNet, an end-to-end scene text editor, which have already learned from multiple characters in a large number of font styles. As mentioned earlier, these methods perform well when used to transfer seen characters to target styles similar to the styles they have seen in the training set. In the first stage, we use Generator 1 to transform all of their seen characters from the base style to the target style. The result from this stage is a set of synthetic samples of the target style. In the second stage, our skeleton-based generator is used as the second generator to transfer the target style to the target text. Instead of using only the original available style samples to train the model, we also include the synthetic samples in the training set to train Generator 2. After Generator 2 is trained, we can use it for font style transferring, similar to a single generator scheme.

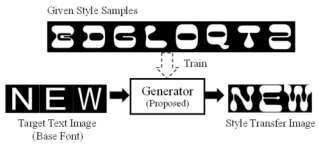

Fig.8: *Single generator scheme.*

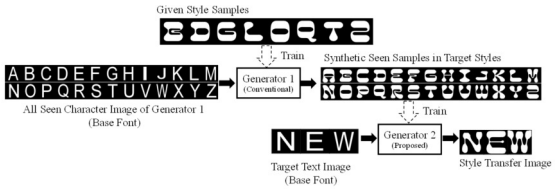

Fig.9: *Double generator scheme.*

3.6 Evaluation Metrics

In this section, we will discuss the evaluation metrics, including MAE, SSIM, △HOG, SWS, and Recognition Accuracy Ratio (RAR), which will be used in this study. All five evaluation metrics used here are quantitative measures that compare the synthetic results with the known ground truth. △HOG and SWS are original evaluation metrics proposed for font structural style transfer and are applied for the first time in this paper.

3.6.1 Mean Absolute Error (MAE)

MAE is a simple and understandable metric used to measure how well a machine learning model, such as an object or image recognition model, performs in terms of prediction accuracy. It is used to calculate the average absolute difference between actual values and predicted values. In text generation tasks, metrics such as MAE, MSE, or RMSE can be used to measure the similarity between the image output from models and the ground truth image [7-9]. The MAE can be calculated using the following equation:

$$\mathrm{MAE}(I_1, I_2) = \frac{1}{N} \sum_{(x,y)} |I_1(x,y) - I_2(x,y)| \qquad (6)$$

where I_1 represents the ground-truth image, I_2 represents the generated image, and N is the number of pixels in the images. The MAE of any pair of 8-bit grayscale images ranges from 0 to 255, with lower values indicating greater image similarity.

3.6.2 Structure Similarity Index Measure (SSIM)

SSIM was introduced by Zhou Wang [25] for assessing perceptual image quality. SSIM quantifies the similarity between two images by comparing their structural information. The similarity measure is calculated based on three comparisons, including luminance, contrast, and structure comparisons. The SSIM score ranges from -1 to 1, with higher values indicating greater image similarity, and a score of 1 representing perfect similarity according to human perception. SSIM is one of the popular quantitative measures for text generation tasks that is used in many studies [4, 7-9, 20]. SSIM of any pair of images I_1 and I_2 is defined as:

$$\mathrm{SSIM}(I_1, I_2) = \frac{(2\mu_1\mu_2 + C_1)(2\sigma_{1,2} + C_c)}{(\mu_1^2 + \mu_2^2 + C_1)(\sigma_1^2 + \sigma_2^2 + C_c)} \qquad (7)$$

where μ_1 and μ_2 are mean of pixel values of I_1 and I_2, respectively, σ_1 and σ_2 are standard deviation of pixel values of I_1 and I_2, $\sigma_{1,2}$ is a covariance of pixel values of I_1 and I_2, C_l and C_c are the constant to avoid instability due to zero division.

3.6.3 Difference in Histogram of Oriented Gradients (△HOG)

HOG is one of the popular shape features proposed by Dalal *et al.* [26] for human detection. Its usage has become widespread across many computer vision tasks. In this work, we introduce the use of HOG for measuring the similarity between the generated character and the ground truth image. Compared to MAE and SSIM, HOG possesses scale and translation invariance, making it suitable for comparing the similarity of shapes or fonts. If only MAE and SSIM are

used, which lack these properties, even slight variations in character position or size would lead to significantly different MAE or SSIM values when measured using conventional methods. Therefore, using HOG can help address this limitation. A difference in HOG between any pair of images I_1 and I_2, denoted as $\triangle\text{HOG}(I_1, I_2)$, is computed by dividing both images into n cells and extracting b-bin HOG from every cells. Fig. 10 illustrates an example of HOG extracting process when $n = 4$. These histograms capture the edge orientation of the character, representing the shape or structural information of the character. After HOGs are extracted from all cells of both images, $\triangle$HOG can be calculated as follows:

$$\triangle\text{HOG}(I_1, I_2) = \frac{1}{nb}\sum_{i=1}^{n}|h_i(I_1) - h_i(I_2)| \quad (8)$$

where $h_i(I)$ represent the b-bin HOG extracted from the i-th cell of the image I. A lower $\triangle$HOG value indicates a higher similarity between two images.

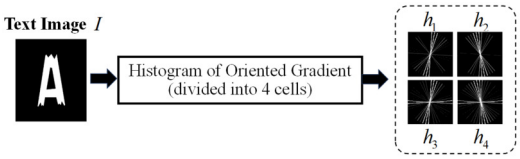

Fig.10: *Visualization of HOG feature extraction in the case of $n = 4$.*

3.6.4 Stroke Width Similarity (SWS)

A stroke width refers to the thickness or width of the lines drawn to create the characters. Stroke width is one of the distinctive characteristics in each font, which affects the appearance of that font. Some fonts may have inconsistent stroke widths across their characters. Therefore, in this study, we propose to use stroke width as one metric to measure the similarity between the generated character and the ground truth image.

To calculate the SWS for a pair of binary character images I_1 and I_2, we first need to apply the stroke width transform (SWT) [1] to both images. The SWT was first proposed by Epshtein *et al.* in 2010 for text detection tasks. It involves transforming pixel values in an input binary image to represent the thickness of the character strokes. The SWS is a metric used to compare two images based on the correlation of their SWT histograms. Let $swt(I)$ represents the stoke width transform of any image I. The stoke width image $swt(I)$ is then divided into n cells, resulting in $swt_i(I)$ for $i = 1, 2, 3, \ldots, n$. The SWS of the i-th cell of a pair of images I_1 and I_2, denoted as $sws_i(I_1, I_2)$, can be calculated as follows:

$$sws_i(I_1, I_2) = MC\left(HT(swt_i(I_1)), HT(swt_i(I_2))\right) \quad (9)$$

where HT is a histogram function and MC is the maximum of the cross-correlation between any two input sequences. Next, the final stroke width similarity, denoted as $\text{SWS}(I_1, I_2)$, is obtained by finding the mean of the maximum correlation obtain from all n cells, as follows:

$$\text{SWS}(I_1, I_2) = \frac{1}{n}\sum_{i=1}^{n} sws_i(I_1, I_2) \quad (10)$$

A higher SWS value indicates greater similarity between two images. Ideally, our proposed SWS, when $n = 1$, possesses both translation and scale invariance. This is because the calculation of the normalized histogram of SWT inherently offers these properties. Calculating the correlation of histograms instead of the difference of histograms also enhances robustness in measuring font similarity across different font sizes. Fig. 11 illustrates an example of the SWS calculation process for $n = 4$.

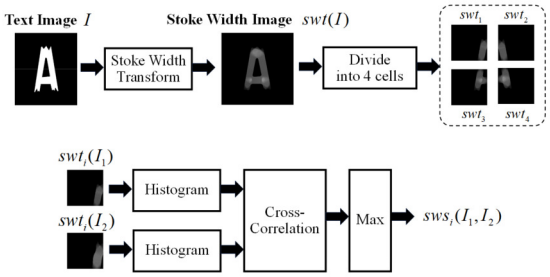

Fig.11: *Visualization of SWS calculation for $n = 4$.*

3.6.5 Recognition Accuracy Ratio (RAR)

Text recognition accuracy serves as another metric for evaluating the perceptual quality of generated text in text generation studies [7-9], as it provides insights into the readability or interpretability of the generated text. In this study, to assess the quality of generated text, we compare the recognition accuracy of the generated texts to their corresponding ground truth. In this paper, we refer to this metric as Recognition Accuracy Ratio (RAR), which is the new name for the evaluation metric formerly named OCRA in our previous work. The definition of RAR is as follows:

$$RAR = \frac{NCR_{GEN}}{NCR_{GT}} \quad (11)$$

where NCR_{GEN} is the number of correctly recognized generated character images and NCR_{GT} is the number of correctly recognized ground truth images. A higher RAR indicates greater readability of

the generated text. The value of RAR can be greater than 1, indicating that the generated text achieves a higher recognition rate than the real one, but it does not necessarily mean that the generated text is similar to the real one. The closer the value of RAR is to 1, the more the readability level of the generated text resembles that of the original text.

4. EXPERIMENT

In this section, we segment the experiment into four parts: (1) Evaluation Metrics, (2) Font Structural Style Transfer, (3) Variations of the Proposed Methods, and (4) Scene Text Editing.

4.1 Evaluation Metrics

In this section, we explore the characteristics of our evaluation metrics by analyzing their ability to measure image similarity, as shown in Table 2. We compare the values of MAE, SSIM, $\triangle$HOG, and SWS across 12 examples of image pairs. In each pair of images, there will be differences in various aspects, such as the size, style, or type of the characters. $\triangle$HOG is calculated using $b = 24$ bins with two values of n. $\triangle\text{HOG}_1$ denotes the value of $\triangle$HOG when $n = 1$ cell, while $\triangle\text{HOG}_4$ denotes the value of $\triangle$HOG when $n = 4$ cells. SWS is also calculated using two values of n. SWS_1 denotes the value of SWS when $n = 1$ cell, while SWS_4 denotes the value of SWS when $n = 4$ cells. An example in the first row demonstrates a scenario where two images are identical. The MAE, $\triangle\text{HOG}_1$, and $\triangle\text{HOG}_4$ values are all equal to zero, while the SSIM, SWS_1, and SWS_4 values are all equal to one.

Table2

Table 2: *The values of MAE, SSIM, $\triangle$HOG, and SWS across 12 examples of image pairs (↑ indicates higher is better; ↓ indicates lower is better).*

I_1 ,I_2	MAE ↓	SSIM ↑	$\triangle\text{HOG}_1$ ↓	SWS_1 ↑	$\triangle\text{HOG}_4$ ↓	SWS_4 ↑
	0.00	1.00	0.00	1.00	0.00	1.00
	34.51	0.80	0.03	0.90	0.03	0.88
	42.01	0.78	0.04	0.70	0.04	0.83
	62.12	0.69	0.01	0.94	0.05	0.73
	32.57	0.81	0.05	0.75	0.04	0.79
	47.95	0.74	0.07	0.70	0.07	0.71
	20.89	0.85	0.01	0.46	0.06	0.79
	36.94	0.79	0.08	0.83	0.09	0.71
	40.90	0.78	0.05	0.76	0.05	0.53
	63.42	0.67	0.16	0.78	0.15	0.85
	61.02	0.69	0.15	0.86	0.18	0.87
	53.30	0.72	0.14	0.75	0.14	0.84

When examining the MAE values across the examples, we note that the highest MAE values occur when comparing images of different characters (rows 10-12), with MAE values ranging from 53.30 to 63.42. However, when comparing the same character in the same font but with a shifted position, a high MAE value is obtained (row 4, MAE 62.12). This value is similar to that obtained when comparing different characters but higher than the MAE values obtained when comparing the same character in different fonts (rows 7-9, MAE 20.89-40.90). These results highlight a limitation of MAE for measuring the similarity between the generated image and the ground truth: its sensitivity to the alignment of the characters' positions in both images.

For SSIM, comparing images of different characters yields low values (rows 10-12, SSIM 0.67-0.72). However, similar to MAE, SSIM is sensitive to translation, scaling, and rotation. Therefore, comparing characters in the same font but with different sizes or positions results in lower SSIM values (rows 2-4, SSIM 0.69-0.80) compared to when comparing the same character in different fonts (rows 7-9, SSIM 0.78-0.85).

For $\triangle$HOG, the least similar groups are different characters (rows 10-12, $\triangle\text{HOG}_1$ 0.14-0.16, $\triangle\text{HOG}_4$ 0.14-0.18), similar to the results for MAE and SSIM. $\triangle$HOG is less sensitive to scaling (row 2-3, $\triangle\text{HOG}_1$ 0.03-0.04, $\triangle\text{HOG}_4$ 0.03-0.04) than MAE and SSIM. $\triangle\text{HOG}_4$ can capture font differences better than $\triangle\text{HOG}_1$ (row 7-9, $\triangle\text{HOG}_1$ 0.01-0.08, $\triangle\text{HOG}_4$ 0.05-0.09), especially in row 7 where $\triangle\text{HOG}_1 = 0.01$. In contrast, the result in row 4 shows that $\triangle\text{HOG}_1$ has higher robustness to translation than $\triangle\text{HOG}_4$ ($\triangle\text{HOG}_1 = 0.01\$$ while $\triangle\text{HOG}_4 = 0.05\$$).

Unlike the other metrics, SWS can offer higher sensitivity to font styles (rows 7-9, SWS_1 0.46-0.84, SWS_4 0.53-0.79) than to character classes (rows 10-11, SWS_1 0.75-0.86, SWS_4 0.84-0.87). Compared to SSIM, SWS demonstrates significantly greater robustness to medium scaling in rows 3 and translation in row 4.

We summarized the sensitivity levels of each evaluation metric to scale, translation, character, and font style variations in Table 3. The results in this section demonstrate that our proposed evaluation metrics ($\triangle$HOG and SWS) can effectively measure differences in font style. These metrics exhibit lower sensitivity to translation and scaling compared to conventional evaluation metrics like MAE and SSIM. Moreover, SWS can be utilized to measure differences in font style across different characters.

Table3

4.2 Font Structural Style Transfer

In this section, we conduct an experiment to evaluate the performance of the font structural style transfer task. The experiment involves transferring six target font styles: Angry Birds, Broadway, Bernard, Datefruitdays, Lymantria, and Winwriter. The 26

Table 3: *Sensitivity of Evaluation Metrics (MAE, SSIM, △HOG, SWS) to Scale, Translation, Character, and Style Variations (VL = Very Low, L = Low, M = Moderate, H = High, VH = Very High).*

Sensitivity	MAE	SSIM	$\triangle HOG_1$	SWS_1	$\triangle HOG_4$	SWS_4
Scale	M	L	M	M	L	L
Translation	VH	VH	L	VL	L	M
Character	VH	VH	VH	M	VH	L
Font Style	L	L	H	H	M	H

upper-case characters of six target fonts are illustrated in Fig. 12. We use the five metrics explained in Section 3.6 to compare the generated character images with the actual character images. In this experiment, we use $n = 4$ to calculate SWS and △HOG For RAR calculation, we use CRNN [27] as character recognition model. This experiment is split into two scenarios. The first scenario is a scenario when the target characters to be generated is 26 upper-case characters. In this scenario, we compare our proposed method with three existing methods: MC-GAN [20], AGIS-Net [4], and SRNet [7]. The 26 upper-case characters are seen as characters for the pre-trained models of MC-GAN, AGIS-Net, and SRNet. The second scenario is a scenario when the target characters to be generated is lower-case characters, Arabic numbers, symbols or characters from other languages. In this scenario, we compare our proposed method with two existing methods: AGIS-Net and SRNet. Those target characters are unseen or rarely seen in the training phase of AGIS-Net and SRNet. We exclude MC-GAN from this scenario because it cannot generate unseen characters.

Fig.12: *Six target font styles used in the experiments.*

4.2.1 Seen Target Characters

In this scenario, for each target style, a set of eight given characters, $\{B, D, L, O, G, Q, T, Z\}$, is provided to each method to generate a complete set of images for all 26 upper-case characters. The target size of the each character image to be generated and evaluated is 408×408 pixels. The implementation details of each method are as follows:

MC-GAN: The experiment utilizes the pre-trained MC-GAN model provided by the authors (`https://github.com/azadis/MC-GAN`). The pre-trained MC-GAN model was trained with 10K grayscale Latin fonts, each with 26 uppercase characters of 64×64 pixels. Eight sample character images in the target style are resized to 64×64 pixels and input into the MC-GAN model. The textural style of the input is a binary image. The generated outputs, with a size of 64×64, are resized back to 408×408 for evaluation.

AGIS-Net: The pre-trained model provided by the authors (`https://github.com/hologerry/AGIS-Net`) is used in the experiment. The pre-trained AGIS-Net model was trained with the same dataset as MC-GAN, using a batch size of 100 for 20 epochs. This pre-trained model is then fine-tuned using eight pairs of sample character images in the target font style and their corresponding base font style, which is Arial. The target textural style is set as a binary image. For each target font, all images are resized to 64×64 pixels for training each AGIS-Net model. The generated outputs, with a size of 64×64, are resized back to 408×408 for evaluation.

SRNet: The pre-trained model provided by the authors (`https://github.com/youdao-ai/SRNet`) is used in the experiment. The pre-trained SR-Net model was trained by using 50,000 synthetic texts with a height of 64 pixels. Both a style image and an image of the target text are required as inputs. The style image used in this experiment is a sequence of eight characters (B, D, L, O, G, Q, T, Z) in one row with the texture style as white text on a black background. As mentioned earlier, the performance of the SRNet model is sensitive to the difference in the length of text in both images. As a result, the target text image is created by repeating the same target character eight times to match the length of text in both images. The font style used in the target text image is Arial, and the textural style is black text on a gray background. Based on this setting, the output from SRNet is eight generated versions of the same target character. Only the first generated character is cropped and resized to 408×408 for evaluation.

Proposed: The proposed model used in this experiment is a single-generator scheme that's trained without a pre-trained model. In the training phase, the model is trained for 1,000 epochs using the Adam optimizer [28] with a learning rate of 0.002. All given style samples (eight characters) are used as the training set while the other uppercase characters (18 characters) are used as the test set. The parameters in the loss function in Eq. (3) are set as $\lambda_R = 100$ and $\lambda_A = 0.1$. The base font style used as input to the generator is Arial. The skeleton is extracted from the input using the Guo-Hall thinning algorithm [29]. The image size for both the input and the output is 408×408 pixels. The stroke width adjustment parameter is set to 1. Batch training is employed in this method with a batch size of 8. For each training batch, we randomly select 204×204 image patches from random positions within the whole character images.

The generated images are illustrated in Fig. 13 and their corresponding objective evaluation is presented in Table 4. The value of each evaluation metric shown in the table is the average from all target characters of all target fonts. We employed paired t-tests to compare the performance of the methods, as our data are paired observations on the same test set. The experimental results show that MC-GAN generates the most similar characters to the ground truth in terms of MAE and SSIM. The proposed method achieves the highest similarity to the ground truth via △HOG and SWS and the highest readability via RAR. The experimental results show that MC-GAN significantly outperformed all other methods in terms of MAE and SSIM ($p < 0.01$). The proposed method outperformed AGIS-Net and SRNet in terms of SWS_4 ($p < 0.05$) and outperformed all other methods in terms of $\triangle HOG_4$ and RAR ($p < 0.01$).

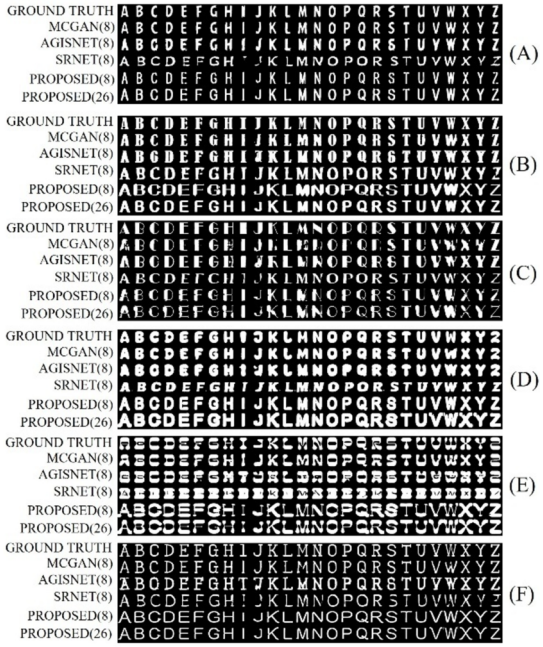

Fig.13: *Generated results of seen target characters from various methods of different target fonts (A) Angry Birds, (B) Bernard, (C) Broadway, (D) Datefruitdays, (E) Lymantria, (F) Winwriter.*

Table 4: *The average objective evaluation of the generated results of seen target characters from various methods (↑ indicates higher is better; ↓ indicates lower is better).*

Method	MAE ↓	SSIM ↑	$\triangle HOG_4$ ↓	SWS_4 ↑	RAR ↑
MC-GAN	**17.97**	**0.87**	0.10	0.66	1.08
AGIS-Net	23.50	0.84	0.11	0.64	0.86
SRNet	49.28	0.74	0.10	0.58	1.06
Proposed	51.27	0.73	0.08	0.66	1.42

4.2.2 Unseen Target Characters

In this scenario, we evaluate the performance of our proposed method in comparison to AGIS-Net and SRNet for transferring font styles to unseen characters. The same set of eight given characters $C = \{B, D, L, O, G, Q, T, Z\}$, is used as style samples for AGIS-Net, SRNet, and the proposed method, as in the previous part of the experiment. In addition, in this scenario, we also test the performance using a set of all 26 upper-case characters $C = \{A, B, C, D, \ldots, X, Y, Z\}$, as style samples for AGIS-Net and the proposed method. The implementation details of each method are the same as described in the previous scenario.

The generated images are illustrated in Fig. 14 and their corresponding objective evaluation are presented in Table 5. The value of each evaluation metric shown in this table is the average from characters with known ground truth, as indicated in the 'Target Characters' column. For example, in Angry Birds, the evaluation metric values are averaged from ten characters (0-9), while in Broadway, the values are averaged from 36 characters (a-z and 0-9). The brackets following each method name refer to the number of given characters, where (8) means C is $\{B, D, L, O, G, Q, T, Z\}$ and (26) means $C = \{A, B, C, D, \ldots, X, Y, Z\}$.

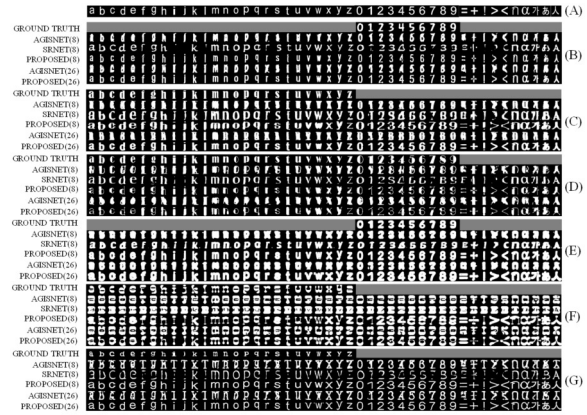

Fig.14: *Generated results of unseen target characters from various methods of different target fonts (A) Base Style, (B) Angry Birds, (C) Bernard, (D) Broadway, (E) Datefruitdays, (F) Lymantria, (G) Winwriter.*

The results show that the proposed methods provide the best of all five evaluation metrices in both eight and 26 given style characters In the case of eight characters, the proposed method significantly outperformed AGIS-Net in terms of MAE, SSIM, $\triangle HOG_4$ and RAR ($p < 0.05$) while it significantly outperformed SRNet in terms of MAE, SSIM and SWS_4 ($p < 0.01$). In the case of 26 characters, the proposed method significantly outperformed AGIS-Net in all evaluation metrics ($p < 0.05$). Fig. 15 presents samples of generated character images. As observed, the

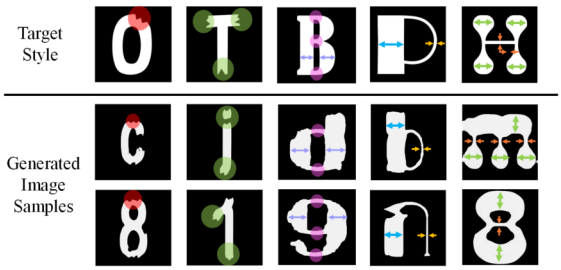

Fig.15: *Detailed comparison between some sample images in the original font styles and the transferred styles by the proposed method.*

Table 5: *The average objective evaluation of the generated results of unseen target characters from various methods (↑ indicates higher is better; ↓ indicates lower is better).*

Method	MAE ↓	SSIM ↑	△HOG$_4$ ↓	SWS$_4$ ↑	RAR ↑
AGIS-Net$_{(8)}$	38.16	0.79	0.12	0.59	0.38
SRNet$_{(8)}$	50.03	0.74	0.10	0.55	0.69
Proposed$_{(8)}$	**37.69**	**0.79**	**0.08**	**0.60**	**0.97**
AGIS-Net$_{(26)}$	36.59	0.80	0.12	0.58	0.35
Proposed$_{(26)}$	**35.98**	**0.80**	**0.08**	**0.63**	**0.89**

models effectively transfer the wavy patterns found at the line ends and corners of the target styles to the generated images. Additionally, they accurately replicate the non-uniform stroke width patterns from the target font styles in the generated characters.

4.3 Variants of the Proposed Method

In this section, we explore several variants of the proposed method to understand its performance under different configurations. These experiments provide insights into how different parameters, inputs, and schemes of the proposed method affect the generated results.

4.3.1 Stoke Width Manipulation

In this part, we demonstrate the stroke width manipulation capability of the proposed method. We conduct an experiment by adjusting the stroke width parameter s to four different values: 0.50, 0.67, 1.00, and 1.25, and use $C = \{A, B, C, D, \ldots, X, Y, Z\}$. Samples of generated characters from six target fonts, each with these four different stroke width settings, are visually presented in Fig. 16. The results show that the proposed method can successfully manipulate the stroke width of the generated characters while preserving the structural styles of the fonts.

4.3.2 Base Font Styles

We compare four different base font styles, including Arial, Calibri, Courier New, and Times New Roman, when they are used as inputs to the proposed model. In this experiment, we use $C = \{A, B, C, D, \ldots, X, Y, Z\}$ for generating the unseen

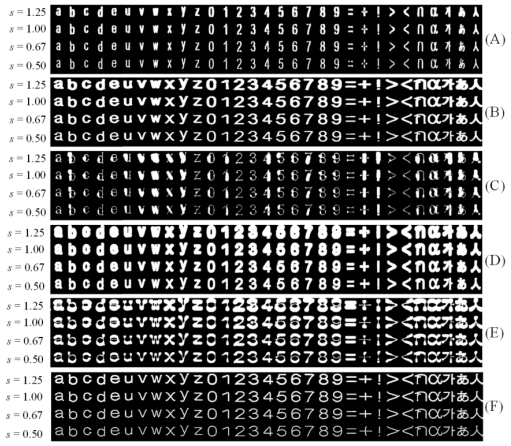

Fig.16: *Generated results from the proposed methods with various stoke width adjusting parameter s of different target fonts (A) Angry Birds, (B) Bernard, (C) Broadway, (D) Datefruitdays, (E) Lymantria, (F) Winwriter.*

characters. The images of the four base styles are illustrated in Fig 17.

Fig.17: *Four base font style used in the experiments.*

Table 6: *The average objective evaluation of the generated results from the proposed methods with various base font styles (↑ indicates higher is better; ↓ indicates lower is better).*

Base Style	MAE ↓	SSIM ↑	△HOG$_4$ ↓	SWS$_4$ ↑	RAR ↑
Arial	**36.54**	0.79	0.08	0.62	0.78
Calibri	50.99	0.74	0.09	0.62	0.71
Courier New	61.93	0.69	0.09	0.62	0.76
Times New Roman	49.00	0.75	0.10	0.62	0.56

We evaluate the results using five evaluation metrics and the same set of generated images with known ground truth as used in Table 5. The evaluation metrics comparing different base styles are displayed in Table 6. The value of each evaluation metric shown in the table is the average from all target characters of all target fonts. The results show that the base styles input to the model affect the output quality. Arial significantly provides the best similarity in terms of MAE, SSIM, and △HOG$_4$ ($p < 0.01$).

4.3.3 Set of Given Style Samples

We conduct the experiment to explore the effects of a set of given style samples on the quality of generated characters. Six different sets of given style samples selected from uppercase characters are compared as follows: $C_1 = \{A, B, C, D, \ldots, X, Y, Z\}$ (26 characters), $C_2 = \{A, C, E, G, \ldots, U, W, Y\}$ (13 characters),

$C_3 = \{B, D, F, H, \ldots, V, X, Z\}$ (13 characters), $C_4 = \{S, T, Y, L, E\}$ (5 characters), $C_5 = \{W, O, R, D\}$ (4 characters), and $C_6 = \{C, M, U\}$ (3 characters). In this experiment, we set the base style to Arial and the stroke width parameter to $s = 1$.

We evaluate the results using five evaluation metrics and the same set of generated images with known ground truth as used in Table 5 and 6. The evaluation metrics are displayed in Table 7. The value of each evaluation metric shown in the table is the average from all target characters of all target fonts. Analysis across five evaluation metrics revealed that a greater quantity of given samples generally leads to improved performance. For instance, the 26 given samples achieved significantly superior results compared to 5, 3, or 2 samples for MAE, SSIM, and SWS4 ($p < 0.05$). This indicates that a greater diversity and quantity of style samples are beneficial for generalization. Similarly, for seen target characters, increasing the number of style samples in C generally leads to improved experimental outcomes.

Table 7: *The average objective evaluation of the generated results from the proposed methods with various sets of given style samples.*

Given Style Samples	MAE ↓	SSIM ↑	$\triangle$HOG$_4$ ↓	SWS$_4$ ↑	RAR ↑
ABCDE...XYZ	36.01	0.80	0.08	0.64	0.81
ACEGI...UWY	36.16	0.80	0.08	0.62	0.79
BDFHJ...VXZ	36.51	0.79	0.08	0.63	0.80
STYLE	38.86	0.78	0.08	0.62	0.79
WORD	37.10	0.79	0.09	0.62	0.79
CMU	37.58	0.79	0.09	0.59	0.60

4.3.4 Double Generator Scheme

In this part, we conduct an experiment to compare the double generator scheme (Fig. 9) with the single generator scheme (Fig. 8), as described in Section 3.5. For the double generator scheme, we first fine-tune the AGIS-Net model (Generator 1) for each target style using $C = \{$B, D, L, O, G, Q, T, Z$\}$. We then use the fine-tuned model to generate all 26 uppercase characters. Next, all 26 generated uppercase characters are used to train the proposed skeleton-based GAN model (Generator 2) for generating lowercase characters, numbers, symbols and characters in other languages. The base style used for both generators in this experiment is Arial.

The results are compared with the single generator scheme using $C = \{$B, D, L, O, G, Q, T, Z$\}$. Table 8 shows averages values of five evaluation metrics. Results from the table show that the single generator significantly outperforms the double generator in terms of MAE and SSIM ($p < 0.01$). However, this depends on the target fonts. The double generator scheme can only improve the quality of the generated characters for some font styles. Bernard is the only font where the double generator scheme improves all five evaluation metrics compared to the single generator scheme, while Datefruitdays is the only font where the double generator scheme does not improve any of the five evaluation metrics. In the other font styles, the double generator scheme improves only some of the evaluation metrics. Since AGIS-Net is used as Generator 1 of the double generator scheme in this experiment, the quality of the AGIS-Net generation for 26 uppercase characters affect to the performance of the double generator scheme. AGIS-Net achieves the highest SWS for Bernard (0.717) and the lowest SWS for Datefruitdays (0.618) when generating the 26 uppercase characters. Following this observation, we analyzed the correlation between the improvement offered by the double generator scheme in each metric and the performance of AGIS-Net when generating the 26 uppercase characters. Our analysis revealed a strong relationship between the improvement in RAR using the double generator scheme and the MAE of AGIS-Net for generating the 26 uppercase characters, yielding a correlation coefficient of 0.855. Furthermore, a strong correlation also existed between the improvements in MAE and SSIM within the double generator scheme and the RAR of AGIS-Net for these characters, with coefficients of 0.747 and 0.728, respectively. These correlations empirically demonstrate a direct relationship between the double generator's improvements in specific metrics and the baseline performance of AGIS-Net, offering crucial insights into its targeted benefits.

Table 8: *The average objective evaluation of the generated results of the proposed methods compared between the double generator scheme and the single generator scheme (↑ indicates higher is better; ↓ indicates lower is better).*

Methods	MAE ↓	SSIM ↑	$\triangle$HOG$_4$ ↓	SWS$_4$ ↑	RAR ↑
Single Generator	**37.69**	**0.79**	**0.08**	0.60	0.97
Double Generator	40.50	0.78	0.08	**0.62**	**0.98**

4.4 Scene Text Editing

For the final experiment, we test our proposed model on the real scene text editing problem by combining our font structural transfer model with a textural transfer model. Examples of scene text editing results are illustrated in Fig 18. The figure presents the outputs from three methods: SRNet, AGIS-Net, and the proposed method. The results show that our proposed method can produce better results compared to SRNet and AGIS-Net, especially when the target text is unseen by the SRNet and AGIS-Net models.

When the target text is "HAPPY," a seen character for both SRNet and AGIS-Net, we observe varying performance. SRNet generates legible text in the second and third examples. However, in the first example, remnants of the original image persist due to a significant length difference between the original and

Fig.18: *Sample results of real scene text editing task.*

target text. AGIS-Net, on the other hand, produces clearly legible characters only in the second example. In contrast, our proposed method consistently generates legible characters across all three examples.

Furthermore, when the target text consists of unseen characters, such as numbers, symbols, or Thai characters, our proposed method accurately generates the desired characters. Conversely, SRNet and AGIS-Net often produce illegible characters or generate characters that merely approximate English letters previously encountered during training.

5. CONCLUSION

In this study, we developed a text structural style transfer model using GANs with skeleton input and proposed new evaluation metrics for this task. △HOG can measure the similarity of font styles with greater robustness to translation and scaling compared to MAE, while SWS can measure the similarity of font styles better than SSIM and can assess style similarity across different characters. In the style transfer experiment with six sample fonts, it was found that our approach produces better results than conventional methods, especially on unseen characters. Since our approach does not require a pretrained model, it provides greater flexibility for users to choose an input base font that is more similar to the target font, thereby improving the results. It also has the capability to manipulate the stroke width of the output text. In a few-shot scenario, we also provide the double generator scheme that incorporates existing methods with our approach. This scheme is useful when existing methods can generate seen characters well for a few samples but struggle with generating unseen characters. The existing methods can assist our proposed model by increasing the number of given style samples, thereby allowing our model to generate better unseen characters. In addition, we also test our approach on real scene text editing tasks. The experimental results show that our approach can successfully perform cross-language text style transfer.

ACKNOWLEDGEMENT

This work was supported by Faculty of Engineering, Chiang Mai University, Thailand.

AUTHOR CONTRIBUTIONS

TT conducted the programming and experiments and wrote the manuscript. KP supervised the project and provided critical feedback on the manuscript.

References

[1] B. Epshtein, E. Ofek and Y. Wexler, "Detecting text in natural scenes with stroke width transform," *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, USA, pp. 2963-2970, 2010.

[2] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao, "TextSnake: A flexible representation for detecting text of arbitrary shapes," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pp. 20–36, 2018.

[3] X. Zhou *et al.*, "EAST: An Efficient and Accurate Scene Text Detector," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, pp. 2642-2651, 2017.

[4] Y. Gao, Y. Guo, Z. Lian, Y. Tang, and J. Xiao, "Artistic glyph image synthesis via one-stage few-shot learning," ACM Transactions on Graphics (TOG), vol. 38, no. 6, pp. 1–12, 2019.

[5] T. Thanusan and K. Patanukhom, "Skeleton-based generative adversarial networks for font shape style transfer," in *Proceedings of the 2023 Asia Conference on Computer Vision, Image Processing and Pattern Recognition*, no.2, pp. 1–7, 2023.

[6] S. Yang, Z. Wang, Z. Wang, N. Xu, J. Liu and Z. Guo, "Controllable Artistic Text Style Transfer via Shape-Matching GAN," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), pp. 4441-4450, 2019.

[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*, pp. 1500–1508, 2019.

[8] P. Roy, S. Bhattacharya, S. Ghosh and U. Pal, "STEFANN: Scene Text Editor Using Font Adaptive Neural Network," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, pp. 13225-13234, 2020.

[9] Q. Yang, J. Huang and W. Lin, "SwapText: Image Based Texts Transfer in Scenes," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, pp. 14688-14697, 2020.

[10] A. B. Anees, A. C. Baykal, M. B. Kizil, D. Ceylan, E. Erdem, and A. Erdem, "HyperGAN-CLIP: A unified framework for domain adaptation, image synthesis and manipulation," in *Proc. SIGGRAPH Asia Conf. Papers (SA '24)*, Art. 105, pp. 1–12, 2024.

[11] V. Singh, "Data augmentation techniques using generative adversarial networks: Employing GANs to create synthetic data for enhancing machine learning model training," *Journal of Engineering Research and Reports*, vol. 27, no. 2, pp. 228–248, 2025.

[12] M. Ayadi *et al.*, "Empowering accessibility in handwritten Arabic text recognition for visually impaired individuals through optimized generative adversarial network (GAN) model," *Journal of Disability Research*, vol. 4, no. 1, 2025.

[13] Y. LeCun, P. Haffner, L. Bottou and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*, pp. 319–345, Springer, 1999.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 27, 2014.

[15] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[16] J. Susskind, A. Anderson and G. E. Hinton, "The Toronto Face Dataset," *Univ. Toronto, Tech. Rep.*, UTML TR 2010-001, 2010.

[17] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Univ. Toronto, Tech. Rep.*, 2009.

[18] Y. Tian, X. Peng, L. Zhao, S. Zhang, and D. N. Metaxas, "CRGAN: Learning complete representations for multi-view generation," *arXiv preprint*, arXiv:1806.11191, 2018.

[19] G. Iglesias, E. Talavera, and A. Díaz-Álvarez, "A survey on GANs for computer vision: Recent research, analysis and taxonomy," *Computer Science Review*, vol. 48, p. 100553, 2023.

[20] S. Azadi, M. Fisher, V. Kim, Z. Wang, E. Shechtman and T. Darrell, "Multi-content GAN for Few-Shot Font Style Transfer," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, pp. 7564-7573, 2018.

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

[22] P. Isola, J. -Y. Zhu, T. Zhou and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, pp. 5967-5976, 2017.

[23] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen, "Improved techniques for training GANs," in *NIPS'16: Proceedings of the 30th International Conference on Neural Information Processing Systems* , vol. 29, pp. 2234 - 2242, 2016.

[24] 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," in *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems* , vol. 30, 2017.

[25] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, April 2004.

[26] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, vol. 1, pp. 886-893, 2005

[27] B. Shi, X. Bai and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298-2304, 1 Nov. 2017.

[28] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015.

[29] Z. Guo and R. W. Hall, "Parallel thinning with two-subiteration algorithms," *Communications of the ACM*, vol. 32, no. 3, pp. 359–373, 1989.

Thanaphon Thanusan received his B.Eng. degree in Computer Engineering from Chiang Mai University, Thailand, and has completed all academic requirements for the M.Eng. degree in the same field. His research interests include deep learning, font synthesis, and generative adversarial networks (GANs).

Karn Patanukhom was born in 1981. He received the B.Eng. degree in Electrical Engineering from Chiang Mai University in 2003 and received the M.Eng. and Ph.D. degrees in electronics from Tokyo Institute of Technology in 2006 and 2009, respectively. He is an assistant professor at the Department of Computer Engineering, Chiang Mai University. His research interests include image processing, machine learning, computer vision, and natural language processing.