Japanese Kuzushiji Font Generation Employing Differentiable Renderer

Honghui Yuan[0009−0001−4334−9363], Junwen Chen[0000−0003−1355−4128], and Keiji Yanai[0000−0002−0431−183X]

The University of Electro-Communications, Chofu, Tokyo, JAPAN
{yuan-h, chen-j, yanai}@mm.inf.uec.ac.jp

Abstract. The study of ancient documents is important for humanity to understand history. In Japan, many ancient documents were written in the Kuzushiji font. Unlike modern fonts, Kuzushiji, as a traditional handwritten font, has unique text structures, with strokes often being connected, simplified, or distorted. In recent years, most studies on font generation have focused on modern font generation, and few have focused on the generation of ancient handwritten fonts like Kuzushiji. Furthermore, vector images are resolution-independent, possess high editing flexibility, and have consistent output quality when compared to raster images. To improve the efficiency of digital data reprinting and the preservation of historical documents by taking advantage of vector images, we propose a few-shot training-free font generation method for generating Kuzushiji characters from modern fonts based on vector images. By using only modern fonts and a few Kuzushiji characters as input, this model mitigates dataset imbalance effectively. Our method integrates a differentiable rasterizer, conditional diffusion model, discriminator, and Multi-Stroke Encoder for stroke-level text control, optimized using multiple loss functions. We conducted numerous experiments on the generation of Kuzushiji fonts, and the results demonstrate that the proposed method successfully generates Kuzushiji characters in vector images, achieving better results than previous methods.

Keywords: Kuzushiji characters· differentiable renderer · font generation· vector image

1 Introduction

Texts are widely used in our daily lives, appearing in various forms such as books, advertisements, and mobile phones. In these contexts, a variety of fonts are extensively employed. However, designing and generating fonts for a large quantity of text is challenging, especially for those with multiple strokes and complex structures, such as Japanese and Chinese characters. Kuzushiji, a font commonly used in ancient Japanese books, the recognition and digital generation of these characters are crucial for the preservation of ancient documents and the understanding of Japanese culture. In recent years, with the development of deep learning, many studies on font generation have achieved excellent

results. Many research methods based on the Generative Adversarial Network (GAN) and Diffusion model use a few reference images as styles and transform content images into corresponding styles to generate new font images. However, these methods are mainly applicable to modern typography fonts. Font generation for Kuzushiji has several major difficulties compared to that for modern texts. First, the images for the Kuzushiji text are all from ancient documents, leading to an unbalanced dataset and low-quality images. Second, as a handwritten font, Kuzushiji characters have different features from typography fonts and are harder to generate. Third, summarizing the Kuzushiji style is challenging, because even the same text from the same author and work has completely different writing methods. Therefore, it is essential to conduct specific research on Kuzushiji fonts. We will discuss this in detail in Section 3. In addition, many current font generation studies are based on pixel-level generation, while there are relatively few studies on vector-based font generation. The vector-based approach can be more effective for the preservation of ancient texts like Kuzushiji characters, which were written by hand. Based on our current knowledge, there has been no research on generating vector-based images of ancient handwritten texts in the existing font generation tasks.

In this study, we proposed a few-shot generation method to generate Kuzushiji characters using vector-based images without the need for training. We used a differentiable rasterizer and conditional diffusion model based on Word-As-Image [4]. We also applied the discriminator and the encoder module to convert modern texts to Kuzushiji character fonts. Our method represented the characters as a set of control points, and by updating the parameters using several proposed loss functions, our model can transfer the Kuzushiji style to the modern font.

The contributions of this paper are summarized as follows:

1. We introduce a few-shot font generation method that can transfer modern fonts to Kuzushiji fonts without training. This method only requires the input of modern font characters and only one or more Kuzushiji characters is required for generation.
2. Our method is based on vector images for ancient handwriting font generation.
3. Experiments demonstrate that the proposed method can generate Kuzushiji-style characters while ensuring its readability.

2 Related Work

2.1 Few-shot Font Generation

Few-shot Font Generation (FFG) methods aim to generate a large number of target character fonts in a desired style from only a few reference images. In recent years, various FFG methods have been proposed and have achieved remarkable results. For example, based on the zi2zi [2] model, ZiGAN [19] proposed an end-to-end Chinese character font generation framework. This framework used

several paired samples from different character styles to generate the expected style conversion characters with only a few target Chinese characters. SCfont [6] used the stroke level semantic information as additional information, which is gradually integrated into the backbone UNET structure to constrain the stroke stability of the generated image. RD-GAN [3] introduced a Radical Extraction Module (REM) that allows one-shot generation of unseen glyphs but only transfers them to the specific target font style. DM-Font [1] introduced the component analysis module to pay more attention to the local characteristics of texts. LF-Font [9] improved upon DM-Font by separating the structural features of different parts from style features. MX-Font [10] was a weakly supervised method, automatically extracting multiple style features by multiple experts without being explicitly conditioned on component labels. It could extract style features and capture a variety of local concepts. DG-Font [20] proposed deformable convolutional blocks in the skip connection to achieve unsupervised learning. XMP-Font [8] proposed a self-supervised cross-modality pre-training strategy and used stroke labels instead of component labels as the basic expression of text structure to achieve high performance in font generation. FSFont [16] realized the few-shot font generation by learning the local style of the reference image and the spatial correspondence between the content and the reference.

Although these methods have achieved great results in various font generation, they are all designed for modern text generation. Furthermore, existing methods always need the TrueType Font(TTF) file for training. However, handwritten characters do not have the TTF file. These methods cannot be used for ancient handwritten fonts like Kuzushiji. In this study, we aim to achieve font generation of handwritten Kuzushiji characters using the Few-shot Font Generation (FFG) method, which is capable of generating images from a small number of reference images selected from ancient books.

2.2 Methods with differentiable renderers

Different from previous research that generated raster images, many recent studies have shifted font generation tasks to the vector domain. CLIPFont [15] used CLIP [12] to achieve font style transfer by calculating the distance between the prompt and images. CLIP was a model that is trained on millions of image-text pairs and learned the relationship between various images and texts. Zero-shot-font [5] achieved font style transfer by combining CLIP with the differentiable renderer and distance transfer loss function. Word-As-Image [4] used CLIP and diffusion model to transfer the semantic meaning of the input word into the corresponding character in the word. This method could change the shape of the character without changing the color or texture of the text. DS-Fusion [17] could stylize the English letter font, visually convey the semantics of the input word to the letter, and ensure the output results remain readable. The Latent Diffusion Model (LDM) [13] adapted its denoising generator and a CNN-based discriminator to fit the style of the input prompt. VecFusion [18] utilized the raster diffusion model and vector diffusion model in cascade mode and made

a vector diffusion model to optimize the font images generated by the raster diffusion model.

The above methods enabled the text generation task to be implemented in the vector domain, however, they are all focused on text artistic style conversion and cannot realize the generation of normal fonts. Furthermore, most of these methods are mainly focused on English character generation and can not apply to complex characters like Japanese. Our study aims to propose a method for generating Japanese Kuzushiji characters in the vector domain without the need to train the model.

3 Japanese Classical Kuzushiji Dataset

The Kuzushiji character images are sourced from the Japanese Classical Kuzushiji Dataset[1] which is owned by the National Institute of Japanese Literature. This dataset consists of 6,151 pages of image data from 44 classical works including various documents from the Edo period. However, because these Kuzushiji images are obtained from ancient books, they often have the background color of books, such as yellow, and a significant amount of indistinct text and noisy images.

Fig. 1. Examples of Kuzushiji characters.

Fig. 1 presents some image data that were selected from the database. Fig. 1 on the left side, depending on the writing method, even different texts will display similar shapes. This indicates that due to the diversity of character types, many characters have similar shapes even though they are different characters. This similarity makes it difficult to clearly distinguish among them. Additionally, in the middle part of Fig. 1, some texts exhibit significant differences from one another despite being the same content. Furthermore, in the right part of Fig. 1, some characters have different forms depending on the work and the scribe, even if they are the same texts. These variations in Kuzushiji characters indicate that their identification and generation present a more challenging task than that of modern characters, requiring a more detailed analysis of their characteristics. These problems of the dataset have also brought challenges to training the model for Kuzushiji font generation like other methods. Thus, our few-shot font generation method can effectively solve the problems of data.

[1] http://codh.rois.ac.jp/char-shape/

4 Methodology

In this paper, we introduce a method to generate Kuzushiji characters in vector images with a small number of reference images selected from ancient books. Our model consists of a differentiable rasterizer and a conditional diffusion model, which is based on the Word-As-Image [4] approach, and a Discriminator that classifies images into two categories: Kuzushiji character images and other images (modern-type font images), and a Multi-StrokeEncoder module, which is used to control the readability of the text by using stroke level features. Furthermore, we employ several loss functions to control the font image generation process.

4.1 Basic method Word-As-Image

Firstly, we will introduce the Word-As-Image [4] that is used as our base network, which uses a pre-trained stable diffusion model to transfer the semantic features from input prompt to images. This method could select which characters to modify and which to retain in their original form.

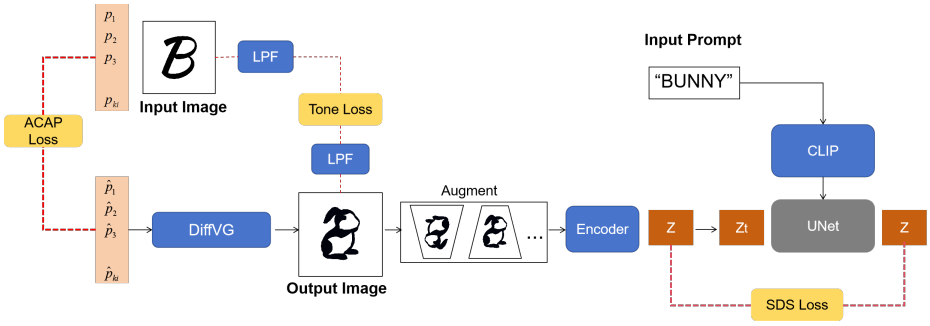

Fig. 2. The framework of Word-As-Image.

Fig. 2 shows the framework of Word-As-Image. The Word-As-Image model uses Bezier Curves to deform the target characters by specifying the style from the input prompt. Since the Stable Diffusion model [11] operates on raster images, it uses DiffVG [7], a differentiable rasterizer that can back-propagate gradients using raster-based loss to optimize shape parameters. Given a word W represented as a string of n characters l_1,...l_n, Word-As-Image applies the semantic features of word W to all characters l_i individually to produce a semantic visual depiction of the characters. Specifically, it utilizes the FreeType font library[2] to extract the outline of the character. This step can ensure a consistent representation across different fonts and characters and facilitate the use of DiffVG for differentiable rasterization. Furthermore, the extracted outline is defined by a different number of control points. A set of control points $P_i = \{P_j\}_{j=1}^{k_i}$ is defined for the shape of the target character l_i. For characters with a small

[2] https://freetype.org/

number of control points, a further subdivision method is applied to increase the control points.

During the generation process, the model incorporates three loss functions to optimize the above parameters. The SDS loss $\nabla_\theta \mathcal{L}_{LSDS}$ could convey semantic concepts from the prompt to the images using CLIP [12] and the Stable Diffusion model [11]. This loss function can be defined as follows:

$$\nabla_\theta \mathcal{L}_{LSDS} = \mathbb{E}_{t,\epsilon}\left[w(t)\left(\hat{\epsilon}_\phi(\alpha_t x_t + \sigma_t \epsilon, y) - \epsilon\right)\frac{\partial z}{\partial z_{aug}}\frac{\partial x_{aug}}{\partial \theta}\right] \tag{1}$$

The As-Conformal-As-Possible Deformation Loss $\mathcal{L}_{acap}$ is designed to preserve the structure of the font. This loss function encourages the optimized shape $\hat{P}$ to have an induced angle α that does not deviate significantly from the angle of the original shape P.

$$\mathcal{L}_{acap}(\mathbf{P}, \hat{\mathbf{P}}) = \frac{1}{k}\sum_{j=1}^{k}\left(\sum_{i=1}^{m_j}(\alpha_j^i - \hat{\alpha}_j^i)^2\right) \tag{2}$$

where α_j^i denotes the angles between the connection line of control points.

The Tone Preserving Loss $\mathcal{L}_{\text{tone}}$ is designed to maintain the consistency of the font's tone (the total amount of black and white areas) between the generated images and the input images.

$$\mathcal{L}_{\text{tone}} = \left\|\text{LPF}(R(P)) - \text{LPF}(R(\hat{P}))\right\|_2^2 \tag{3}$$

where LPF represents a low-pass filter and P and $\hat{P}$ are the sets of control point parameters that constitute the vector image.

4.2 Proposed method

Word-As-Image achieved the transformation from the semantic features of the prompts into character images of the corresponding words and yielded excellent results. However, this method has several limitations. It mainly focuses on English characters and cannot produce satisfactory results for complex text with many strokes such as Japanese characters. Moreover, when presented with sparse prompts like "Japanese Kuzushiji," the model cannot fully understand the semantic meaning of the prompt. This method is also limited to converting semantic styles to the characters that belong to the input semantic words, which does not allow for the free selection of characters.

Therefore, we propose a new model to generate vector images for Kuzushiji characters using a few reference images without training. We use "Japanese Kuzushiji" as the input prompt and our method can select any text to be converted without the limitation of the generation in the corresponding words. The framework of the proposed method is illustrated in Fig. 3. When using the prompt as a condition to control the generation of fonts, it is challenging

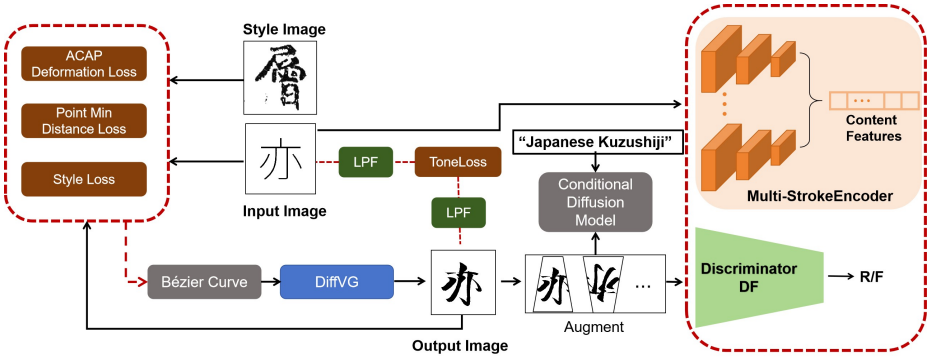

Fig. 3. The network of the proposed method.

to describe Kuzushiji features in terms of the prompt. Therefore, to generate the result with the feature of Kuzushiji characters, we use the text image of Kuzushiji to control the generated result. In addition to using the Conditional Stable Diffusion model [11] and DiffVG used by Word-As-Image, we utilize a Discriminator(DF) that is trained to discriminate two classes of images, Kuzushiji character images and modern font images. Specifically, we utilize the Kuzushiji dataset and modern text to train the discriminator in advance, Fig. 4 shows some Kuzushiji samples of our training data. In the generation process, we utilize the trained discriminator to control the generation of Kuzushiji characters. In addition, inspired by the MX-Font [10], which employed multiple encoders to extract the features of the input images for each component of the character, we applied the Multi-Stroke encoders to control the generation at the detail level.

Furthermore, because our method is FFG and Kuzushiji has more deformation and consecutive strokes, it is difficult to learn Kuzushiji features only using the above model. Thus, we proposed some new loss functions to realize font transfer while keeping font readability and Kuzushiji font features. Specifically, we used Point Min Distance Loss, which imposes a penalty when the distance between the closest points is less than the determined distance, and the Style Loss controls the similarity of style features between the Kuzushiji character and generated images. We have also improved As-Conformal-As-Possible(ACAP) Deformation Loss to make it applicable to more complex characters with multiple strokes. Additionally, we also utilized the Tone Preserving Loss and SDS Loss, which was used in the basic method Word-As-Image [4] to maintain the original font shape and transfer the semantic features from prompt to generated image. In the next section, we will introduce the loss function in detail.

Style Loss To better achieve the style transfer of the Kuzushiji features for the generated characters, we used the style loss L_{style} in our method. Specifically, we compute the Gram matrix for both the features of the generated images and the style images. After calculating the Gram matrix, we compute the mean squared error between them. The style loss allows us to quantitatively measure and minimize the stylistic differences between the two images. The style images

Fig. 4. The example Kuzushiji images of the dataset to train the discriminator.

were randomly selected from the Kuzushiji dataset, and to better obtain style features and reduce the influence of content features, in this study, we used 20 style images to extract the mean style features. The calculation of the style loss is as follows.

$$L_{\text{style}} = \sum \|G_{l,i,j}(x_s) - G_{l,i,j}(x_g)\|^2 \tag{4}$$

$$G_l(x) = F_l(x)F_l^T(x) \tag{5}$$

where G_l is the Gram matrix. $F_l(x)$ is the feature map of image x from the VGG19 [14]. x_s is the reference style image and x_g is the generated image.

Tone Preserving Loss and SDS loss The Tone Preserving Loss L_{tone} involves applying a low-pass filter to the rendered vector image and calculating the $L2$ distance between the input image and the generated image, which preserves the tones of the image. This process helps the adjusted image not deviate significantly from the input image. In our study, we employed the Tone Preserving Loss, which is based on the Word-As-Image approach to restrict the tone of generated Kuzushiji characters to maintain the text structure. Furthermore, to enable the Kuzushiji style features of the input prompt to transfer to the generated image, based on Word-As-Image, we also use SDS loss to transfer the style features into the latent space.

Multi-stroke ACAP Loss In Word-As-Image [4], the As-Conformal-As-Possible Deformation Loss function helps control the angle between control points, preventing large deformations. To more precisely control the font structure of components like strokes, the angle calculation for each stroke is necessary. The original loss function performs very well for simple English letters, but due to the multi-stroke structure of Kuzushiji characters, it is sometimes impossible to calculate the angle for each stroke. Thus, it does not perform well in complex font

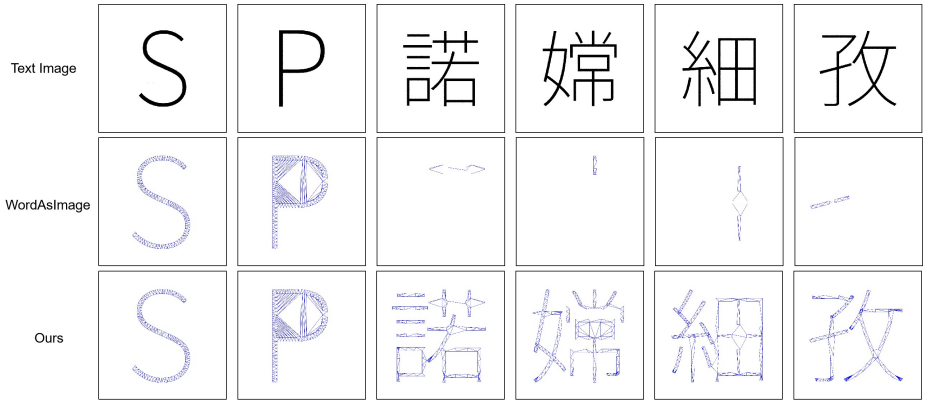

Fig. 5. Visualization images of the connecting lines between the control points for calculating the angle for ACAP Loss.

characters. In this study, we proposed Multi-stroke ACAP Loss L_{ACAP} to extend this loss function to apply to multiple stroke structures. Specifically, instead of building control points for the whole character at once, we use Diffvg to get the number of strokes in the character and then construct control points for each stroke in the character. Finally, we combine them to get the final control points. Fig. 5 shows some visualizations of the character for calculating the angle using the connecting lines between the control points. The first row shows the original text, and the second and third rows show Word-As-Image and the results of our method. The results indicate that our proposed loss function can effectively get the connection line for each stroke. Thus, our method could compute the loss function between individual strokes. In contrast, when dealing with complex fonts, Word-As-Image has a large number of missing strokes and therefore is unable to do an effective calculation for complex characters.

Point Min Distance Loss In addition to constraints on the overall appearance of the font and the angle between the control points, we also utilized the minimum distance loss L_{point} to restrict the length between control points. We prevent sudden and significant changes in the connecting lines between control points to ensure that the distances of the characters change gradually, thereby maintaining the continuity of the strokes in the character. Specifically, we calculated the distance between adjacent points of the control point and controlled the distance within a certain value.

Multi-Stroke Module Although we employ the aforementioned loss function to control style and font structure, they are all grounded in the overall structure of the font and lack component-level control. Furthermore, because kuzushiji is often accompanied by large deformation of font structure, without detailed control, it is easy to cause the generated results to be accompanied by the collapse of structure. We will prove this in ablation studies. Therefore, to add the

detailed features, we added the Multi-Stroke Module to our network and calculate the loss function L_{SLS} to ensure the readability and integrity of the font. Specifically, as shown in Fig. 3, drawing inspiration from MX-Font [10] that uses multiple encoders to automatically extract the component features of characters, we introduced the multiple-stroke encoder to extract component-level features. We use 4 encoders to extract the detailed features and then concatenate these features as the final features. After getting the whole features, we calculate the Mean Squared Error (MSE) loss between the generated image and the original content image.

The total loss function of our method is as follows:

$$\begin{aligned} L_{total} =& \lambda_{dis} L_{\text{discriminator}} + \lambda_t L_{\text{tone}} + \lambda_a L_{\text{ACAP}} + \lambda_p L_{\text{point}} \\ &+ \lambda_s L_{\text{style}} + \lambda_{sls} L_{\text{SLS}} + \lambda_{sds} L_{\text{SDS}} \end{aligned} \tag{6}$$

By utilizing the loss functions and module described above, we can precisely manipulate various aspects of the vector image, including controlling the angles between control points, the distances between these points, the overall structure and detailed strokes of the font. Thus, our model can transfer the Kuzushiji style features to the modern font.

Fig. 6. The results of our proposed method compared with ground truth. The first row shows the original font image(modern text font). The second row is the Kuzushiji characters obtained from our method, and the last row shows the real Kuzushiji images.

5 Experiments

5.1 Experimental Settings and Results

To demonstrate the effectiveness of our method, we conducted experiments on all types of Japanese characters, including kanji, hiragana, and katakana. Experiments were conducted with the following settings for the parameters used in the evaluation. The number of Iterations: 300. The loss weights: Discriminator Loss λ_{dis}: 0.01 , Tone Loss λ_t: 1, SDS Loss λ_{sds}: 0.001, Multi-stroke As-Conformal-As-Possible Deformation Loss λ_a: 1, Point Min Distance Loss λ_p: 1, Point Min Distance Loss Distance: 1, Style Loss λ_s: 0.002, Multi-Stroke Module Loss λ_{sls}: 1. We used version 1.5 of the Stable Diffusion model. The input in the reference

phase only needs the specified characters. The reference time of our method is about 2 to 3 minutes. The experiments are conducted on a single RTX 3090 GPU.

As shown in Fig. 6, the experimental results of the proposed method show that our method can realize Kuzushiji character generation from modern font characters while ensuring the readability of the text. Compared to the real Kuzushiji characters, our results could generate the same features at the stroke level. Fig. 7 displays additional results produced by our model. The results show that our model is capable of processing various types of Japanese characters in Kuzushiji font like Hiragana and Katakana. Additionally, since our model only requires the input of desired characters, large-scale generation of Kuzushiji data becomes feasible.

5.2 Comparison with Previous Method

In this section, we will compare the results of our method with those of previous methods. The objective of our experiment is to generate black-and-white Kuzushiji vector images from modern type font images. Word-As-Image [4] and CLIPFont [15] are methods that, like ours, utilize vector images to transfer font style based on input prompt. Thus, we conducted the experiments between our method and these methods. In both experiments, Kuzushiji characters were not learned in advance. The input prompt for the other methods was the same as ours, which is "Japanese Kuzushiji". Fig. 8 shows the comparison results. The results from CLIPFont are similar to the original font with few Kuzushiji features. Word-As-Image exhibits only irregular deformation of characters without distinct Kuzushiji features. In contrast, our method successfully generates Kuzushiji characters from the source modern font characters.

Fig. 7. The results generated by our model with different types of Japanese characters.

Furthermore, Fig. 9 presents comparisons with recent raster-based few-shot methods, including diffusion-based models. These baselines use officially trained

weights with Kuzushiji as reference input. Despite this, our method achieves better results, even without training.

Fig. 8. The results of our method compared to the previous methods. The first row shows the modern text fonts as the original input text, the other rows show the results of the previous methods and our method.

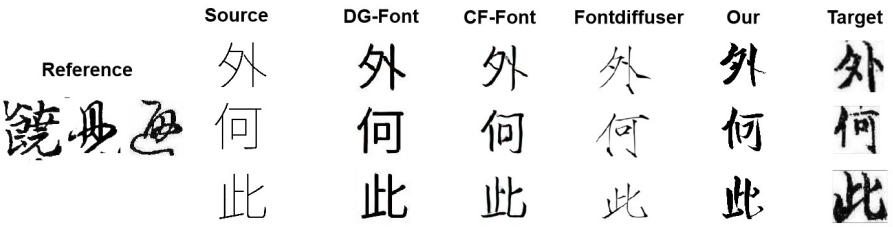

Fig. 9. Comparison results with all types of font generation methods.

Table 1 shows the quantitative results between our method and previous methods. The bold text displays the best results, while the underlined text displays the second-best results. We compared the results of each method with 100 randomly selected corresponding Kuzushiji characters. Specifically, we employed RMSE to measure the similarity between the generated image and the target style image at the pixel level, and LPIPS could discriminate the similarity of the images from the human perspective. FID and SSIM are used to calculate the similarity between two whole images. The evaluation metrics are computed as follows. RMSE: the squared differences for each data point. LPIPS: utilizes VGG to extract features and calculate L2 loss. FID: calculate the Fréchet distance between two images. SSIM: compares local brightness, contrast, and structural information of two images. The experimental results show that our model achieves the best scores in LPIPS, SSIM, and FID. The last line shows that our results have achieved an improvement of more than 6% in these scores. We also achieved the second score in RMSE.

Table 1. Quantitative evaluation results with previous methods.

	LPIPS↓	SSIM↑	FID↓	RMSE↓
CLIPFont	0.4137	0.5002	1.8965	**0.4318**
WordAsImage	0.3903	0.4887	1.8531	0.4498
Ours	**0.3634**	**0.5592**	**1.5494**	0.4379
	6.89%	11.79%	16.38%	(-)

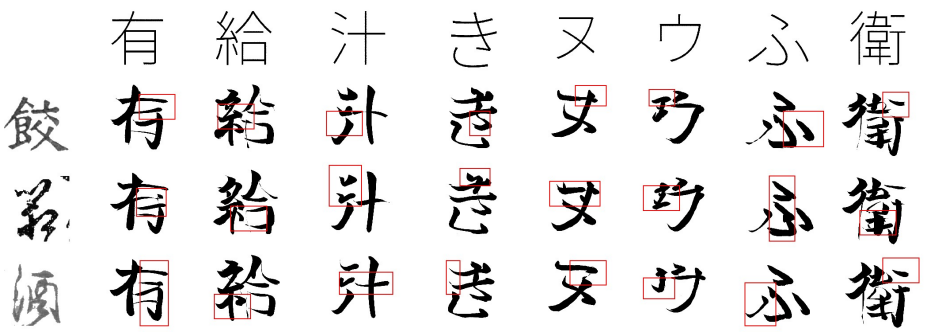

Fig. 10. The comparison results with changing different style images.

The quantitative results show that our generated font images are more similar to the real Kuzushiji character images in terms of the overall structure and the stroke of the image. In contrast, Word-As-Image received the lowest score due to its irregular changes in structure. Although CLIPFont performed better in RMSE, this method almost has the same structure as the original content image and failed to reflect the Kuzushiji style.

In addition, to confirm the effect of the reference style image, we conducted an experiment on how the results would change when using different reference style images used in the style loss. Fig. 10 shows the results of this experiment that uses one style image differently. The content image is in the first row, and the images of different styles are in the first column. The first style has more regular and straight strokes, the second has connected and more curves in stroke, and the third has sharp and slender connected strokes. In the middle are the results generated by our model. The red part highlights the differences in the results that correspond to each part that is similar to the reference style features. The results indicate that by changing the reference style image, our model can obtain the corresponding Kuzushiji font images that are more similar in reference style. Furthermore, we conducted experiments on the performance of our model in other languages. Fig. 11 shows the results of our model for Korean characters. The results indicate that our model can also perform Kuzushiji style conversion for other languages. This demonstrates the effectiveness of our few-shot generation model in Kuzushiji font generation.

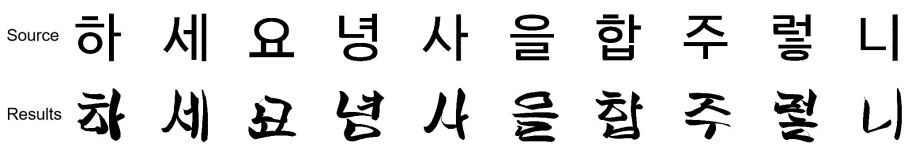

Fig. 11. The results of our model for other languages.

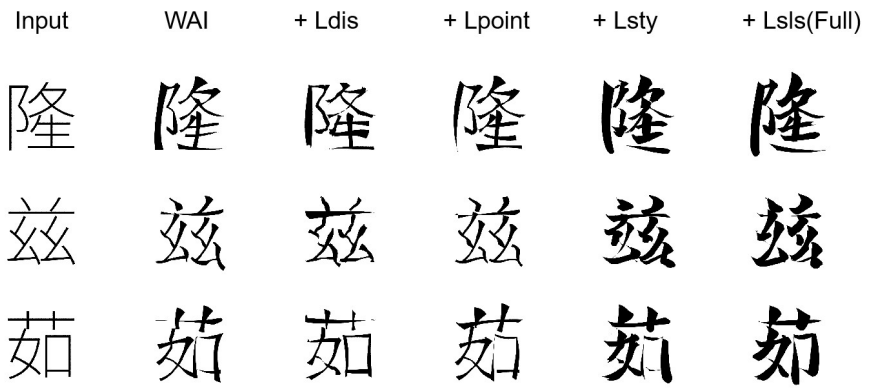

Fig. 12. The results of ablation study. WAI represents the results generated using SDS, original ACAP, and Tone loss that were used in the basic method Word-As-Image.

5.3 Ablation Studies

We conducted ablation studies to verify the effectiveness of each part of the proposed method. The results of the ablation study are shown in Fig. 12. Since we use the same loss function as the baseline method, we only investigated the new loss functions and modules that were proposed in this study. When we incorporate the discriminator into the model, the generated results have some features of Kuzushiji, but the features are not uniformly distributed among the strokes and the effect is not obvious. By utilizing Point Min Distance Loss, the stylistic characteristics of Kuzushiji can be rendered more apparent. The results from utilizing style loss indicate that style loss effectively transfers the style of the Kuzushiji style to the whole image but causes distortions that appear at the stroke level. Especially unnecessary deformations at the ends of strokes. As shown in the last column, after adding the Multi-Stroke Module, the strokes of the characters are successfully preserved in the generated results while maintaining the feature of Kuzushiji. Therefore, by combining the aforementioned loss functions and modules, our method can generate natural-looking Kuzushiji characters while ensuring the complete structural integrity of the font.

Table 2 shows our quantitative evaluation of the ablation study. We used the same metrics and we calculated the score between the generated results and the real Kuzushiji characters. Our full model received the highest and second scores. The absence of any module will lead to a deduction of scores in LPIPS,

Table 2. The quantitative evaluation results of the ablation study.

	LPIPS↓	SSIM↑	FID↓	RMSE↓
WAI	<u>0.4158</u>	0.4841	1.9022	0.4113
+ Ldis	0.4305	<u>0.4910</u>	1.9084	**0.3733**
+ Lpoint	0.4242	0.4849	<u>1.8382</u>	0.3783
+ Lsty	0.4258	0.4825	1.8429	0.3796
+ Lsls(Full)	**0.3685**	**0.4930**	**1.7724**	<u>0.3772</u>

SSIM, and FID. Considering the rationality and integrity of the font structure, this proves that our generated results perform best using the full model.

6 Discussion

Our method has achieved the few-shot Kuzushiji font generation task in vector images. However, Kuzushiji characters can have diverse writing styles even for the same character that is written by the same writer, and the ancient image data always have poor quality, thus it remains challenging to achieve an exact match with real samples. At the present stage, we generate the Kuzushiji style as one category. However, Kuzushiji, as a large font system, has produced many different works by different authors, and a more detailed division is necessary in the future. Moreover, due to limitations of the Bezier Curve in capturing texture features, our method currently struggles to accurately render the texture of Kuzushiji characters, further improvement of the method, like adding texture modules, is needed to improve the result's similarity to Kuzushiji characters.

7 Conclusion

In this study, we proposed a few-shot font generation model for generating Kuzushiji characters in vector images without training. The proposed method can generate Kuzushiji character fonts from modern fonts according to the reference Kuzushiji characters. Our experimental results demonstrate that this approach can maintain the structural integrity of the text while accurately reflecting the Kuzushiji style without learning Kuzushiji characters in advance. We have analyzed the effectiveness of each loss function and module in our method. The results show that with all of the advantages of our component, we could generate characters that captured the features of Kuzushiji characters and generate the closest results to the real Kuzushiji image. Considering that ancient texts such as Kuzushiji often suffer from a lack of sufficient and high-quality datasets, our method provides a convenient means for generating Kuzushiji characters without the need for a large amount of text data and thus can contribute to enriching the dataset as well as recovering ancient texts in other languages. Furthermore, because our model only needs to input text to generate Kuzushiji characters, it can be easily implemented for large-scale generation tasks. In the future, we can improve our model by doing work such as texture generation and precise generation.

References

1. Cha, J., Chun, S., Lee, G., Lee, B., Kim, S., Lee, H.: Few-shot compositional font generation with dual memory. In: Proc. of European Conference on Computer Vision. pp. 735–751. Springer (2020)
2. Chang, B., Zhang, Q., Pan, S., Meng, L.: Generating handwritten chinese characters using cyclegan. In: 2018 IEEE winter conference on applications of computer vision (WACV). pp. 199–207. IEEE (2018)
3. Huang, Y., He, M., Jin, L., Wang, Y.: Rd-gan: Few/zero-shot chinese character style transfer via radical decomposition and rendering. In: Proc. of European Conference on Computer Vision. pp. 156–172. Springer (2020)
4. Iluz, S., Vinker, Y., Hertz, A., Berio, D., Cohen-Or, D., Shamir, A.: Word-as-image for semantic typography. ACM Transactions on Graphics (TOG) **42**(4), 1–11 (2023)
5. Izumi, K., Yanai, K.: Zero-shot font style transfer with a differentiable renderer. In: Proceedings of the 4th ACM International Conference on Multimedia in Asia. pp. 1–5 (2022)
6. Jiang, Y., Lian, Z., Tang, Y., Xiao, J.: Scfont: Structure-guided chinese font generation via deep stacked networks. In: Proc. of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4015–4022 (2019)
7. Li, T.M., Lukáč, M., Gharbi, M., Ragan-Kelley, J.: Differentiable vector graphics rasterization for editing and learning. ACM Transactions on Graphics (TOG) **39**(6), 1–15 (2020)
8. Liu, W., Liu, F., Ding, F., He, Q., Yi, Z.: Xmp-font: Self-supervised cross-modality pre-training for few-shot font generation. In: Proc. of IEEE Computer Vision and Pattern Recognition. pp. 7905–7914 (2022)
9. Park, S., Chun, S., Cha, J., Lee, B., Shim, H.: Few-shot font generation with localized style representations and factorization. In: Proc. of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 2393–2402 (2021)
10. Park, S., Chun, S., Cha, J., Lee, B., Shim, H.: Multiple heads are better than one: Few-shot font generation with multiple localized experts. In: Proc. of IEEE International Conference on Computer Vision. pp. 13900–13909 (2021)
11. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. In: International Conference on Learning Representations (2023)
12. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: Proc. of the International Conference on Machine Learning. pp. 8748–8763. PMLR (2021)
13. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proc. of IEEE Computer Vision and Pattern Recognition. pp. 10684–10695 (2022)
14. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
15. Song, Y., Zhang, Y.: Clipfont: Text guided vector wordart generation. In: Proc. of the British Machine Vision Conference. BMVA Press (2022)
16. Tang, L., Cai, Y., Liu, J., Hong, Z., Gong, M., Fan, M., Han, J., Liu, J., Ding, E., Wang, J.: Few-shot font generation by learning fine-grained local styles. In: Proc. of IEEE Computer Vision and Pattern Recognition. pp. 7895–7904 (2022)
17. Tanveer, M., Wang, Y., Mahdavi-Amiri, A., Zhang, H.: Ds-fusion: Artistic typography via discriminated and stylized diffusion. In: Proc. of IEEE International Conference on Computer Vision. pp. 374–384 (2023)

18. Thamizharasan, V., Liu, D., Agarwal, S., Fisher, M., Gharbi, M., Wang, O., Jacobson, A., Kalogerakis, E.: Vecfusion: Vector font generation with diffusion. In: Proc. of IEEE Computer Vision and Pattern Recognition. pp. 7943–7952 (2024)
19. Wen, Q., Li, S., Han, B., Yuan, Y.: Zigan: Fine-grained chinese calligraphy font generation via a few-shot style transfer approach. In: Proc. of ACM International Conference Multimedia. pp. 621–629 (2021)
20. Xie, Y., Chen, X., Sun, L., Lu, Y.: Dg-font: Deformable generative networks for unsupervised font generation. In: Proc. of IEEE Computer Vision and Pattern Recognition. pp. 5130–5140 (2021)