DiffVecFont: Fusing Dual-Mode Reconstruction Vector Fonts via Masked Diffusion Transformers

Yu Liu
Dalian Chinese Font Design Technology Innovation Center
Dalian Minzu University, 116600 Dalian, China
Faculty of Computer Science and Information Technology
University Putra Malaysia, 43400 UPM Serdang, Malaysia
ethanliuyu@foxmail.com

Fatimah Binti Khalid
Faculty of Computer Science and Information Technology
University Putra Malaysia, 43400 UPM Serdang, Malaysia
fatimahk@upm.edu.my

Cunrui Wang*
Dalian Chinese Font Design Technology Innovation Center
Dalian Minzu University, 116600 Dalian, China
wcr@dlnu.edu.cn

Mas Rina Binti Mustaffa
Faculty of Computer Science and Information Technology
University Putra Malaysia, 43400 UPM Serdang, Malaysia
MasRina@upm.edu.my

Azreen Bin Azman
Faculty of Computer Science and Information Technology
University Putra Malaysia, 43400 UPM Serdang, Malaysia
azreenazman@upm.edu.my

Abstract

Vector fonts are favored by font designers for their editability. However, reconstructing vector images from raster glyph images is a cross-modal process. Existing methods either struggle to reconstruct fine contours or fail to represent glyphs concisely. This paper proposes a dual-modal vector denoising diffusion model that integrates vector and raster images for reconstructing quadratic Bézier curves from raster images. Specifically, we pre-calculate the signed distance function (SDF) values for each input glyph to effectively capture the geometric information of the glyph. Then, we integrate the image and vector dual modalities to enhance the stability of the reconstruction process. By incorporating a variable masking mechanism, the model gradually reduces its dependency on the vector modality, enabling end-to-end cross-modal vector reconstruction during the inference process. Our vector diffusion model employs a transformer architecture and an innovative vector representation method, capable of modeling various vector geometric shapes. In the realm of vector font reconstruction, our approach outperforms existing techniques. The font images generated by our method can be readily converted into TrueType fonts, highlighting the significant practical value of this research. Keywords: Vector Font Generation, Multi-Modal Representation, Variable-Rate Masking,Vector Quantization.

1. Introduction

Vector fonts define outlines through a set of parameterized shapes, allowing for easy adjustments and ex-

pansion of glyph appearances. However, developing vector fonts is a time-consuming task, as it requires font designers to create glyph outlines for 9,169 Chinese characters. This process relies on the expertise and experience of the designers, setting a high threshold for font development and making the fusion of art and technology a valuable endeavor. With advancements in image generation technology, deep learning-based font reconstruction methods can generate pixelated glyph images. The reconstruction of vector glyph images from raster ones helps further simplify and expedite the font design process. However, due to the topological structure, parameter sequence length, and stylistic diversity of vector fonts, reconstructing vector glyphs from pixelated glyph images remains a challenging and ongoing problem.

The development of deep generative models has led to the emergence of several effective and powerful methods for reconstructing aesthetically pleasing raster glyph images. The development of Chinese font libraries is gradually transitioning from manual creation to automation, significantly enhancing the efficiency of Chinese font creation. Some methods have been proposed that through deep generative models[12, 37], a small portion of reference style images designed by designers is learned to generate raster glyph images of 9,169 characters in an end-to-end manner. Other methods have been proposed to learn separate representations for Chinese character style and content in an unsupervised manner, enabling the generation of raster images for arbitrary style-content combinations [33, 29, 17, 6]. However, after the model generates raster images for all Chinese characters, these raster glyphs need to be converted into vector glyph images. However, due to the complexity of glyph shapes, this conversion process still requires significant manual intervention.

In recent years, researchers have paid extensive attention to font reconstruction and recognition based on sequential representations. FontRNN[27] and Write Like You[26] encode and decode sequences of handwritten trajectories using RNN or LSTM, achieving vector reconstruction of handwritten strokes. The Deep imitator[38] accurately mimics handwritten trajectories by introducing attention mechanisms, resulting in the generation of editable handwritten fonts. Recently, Liu et al.[11] and Yang et al.[35] proposed online stroke analysis tasks during the author's writing process. However, unlike conventional TrueType Font (TTF), these methods treat handwriting as time-series data, focusing on capturing dynamic stroke variations during the writing process rather than static glyph outlines. TTFs precisely describe glyph outlines using mathematical formulas and coordinate parameters, forming complex closed shapes composed of quadratic Bézier curves and lines[8]. This representation method is widely used in the Scalable Vector Graphics (SVG) format[39]. With the advancement of sequence generation models, Lopes et al.[14] and Deepsvg[1] proposed transforming the vector font reconstruction problem into generating SVG drawing parameters, making it possible to reconstruct vector fonts based on these parameters. Researchers have made much effort to directly reconstruct vector fonts in recent years. A typical approach is to model each glyph as a shape primitive composed of several Bézier curves, mapping from raster images to vector images through combination and adjustment of vector shapes [32, 13]. However, deep neural networks struggle to directly learn and understand explicit representations of vector glyphs. The main challenges lie in the long-sequence dependencies exhibited by vector graphics and the ambiguity in depicting glyph contours.

In this study, we aim to address the challenging problem of reconstructing vector glyphs with compact contours from rasterized glyph images. To capture long-range dependencies and accommodate the irregularity of vector representations, we introduce a transformer-based vector diffusion model. We convert vector glyph data into SVG representations, making it more suitable for the denoising diffusion model and the learning process of the Transformer. Given the heterogeneity between image modality and vector modality data, establishing a direct connection between the two modalities is challenging. Therefore, we precompute the signed distance function (SDF) values for each input glyph, effectively capturing the geometric information of the glyph while bridging the differences between the modalities. We then apply a modality alignment strategy. During the joint training of image modality and vector modality, we employ a masking mechanism that gradually increases over training time to reduce the model's reliance on vector modality and enhance its focus on image modality features. Next, we quantize the features of the vector modality to obtain discrete indices for each feature. Subsequently, we use a self-supervised learning approach to train the image modality to predict the corresponding feature indices of the vector modality, thereby enhancing the consistency between the two modalities. Finally, during inference, the model predicts the quantized indices of unmasked vector features solely through the image modality, achieving the cross-modality task of vector reconstruction during the inference process. Our proposed model is capable of reconstructing vector glyphs from raster glyph images in an end-to-end manner.

Furthermore, it excels in generating various types of icons. In summary, our main contributions are as follows

- We propose a masked diffusion transformer model that integrates vector and image modalities, effectively "upsampling" low-resolution raster glyph images into vector representations.
- We condition the vector diffusion model on the SDF values of raster glyphs, allowing the model to "denoise" noisy vector glyph representations into structured Bézier curves of the glyph.
- We introduce a new framework for reconstructing vector glyphs. During training, we gradually reduce the model's dependence on vector modality through variable-rate masking and predict the missing vector modality information based on image modality, thereby achieving the cross-modality task of vector reconstruction.
- We conducted extensive experiments and achieved state-of-the-art performance in generating multi-language vector fonts and vector icons, producing outputs that are comparable in quality and compactness to those designed by humans. The outputs can be easily converted into common font formats.

2. Related Work

2.1. Vector Font Generation

Vector font reconstruction methods can be categorized into supervised and unsupervised approaches. In unsupervised methods, Im2vec [21] maps raster images to drawing commands by adjusting control points on the unit-circle contour. VecFontSDF [32] fits glyph images by combining and adjusting multiple Bézier curves forming closed shapes. DualVector [13] treats each glyph as a collection of closed paths and obtains glyph contours through Boolean operations on these closed paths. While these methods are suitable for reconstructing vector glyphs of simple characters, the resulting vector glyphs often suffer from redundant curves and non-smooth contours. Supervised vector font reconstruction methods rely on annotated vector glyphs as training data. SVG-VAE [14] was the first to build a sequential generative model for SVG fonts. DeepSVG [1] developed a hierarchical transformer-based generative model for generating complex SVG icons and interpolations. DiffVG [9] made vector curves differentiable, establishing the relationship between vector graphics and bitmaps, making them suitable for optimization through networks, and gradually becoming the theoretical basis for many vector shape generation works. DeepVecFont [30] and DeepVecFont-v2 [31] jointly represent features of font images and sequences, establishing relationships between vector graphics and bitmaps, and leveraging the Transformer's ability to model long sequences to generate drawing commands. However, the vector reconstruction process is a cross-modal task, and the heterogeneity of the data makes it challenging to establish direct connections between these two modalities. To address this issue, we precompute the SDF values for each input glyph, which effectively captures the geometric information of the glyph while coordinating the differences between the modalities.

2.2. Diffusion Model

Diffusion models are a novel type of generative model that use an iterative reverse diffusion process to generate high-quality images. Sohl-Dickstein et al. [24] first elucidated the concept of diffusion probabilistic models and denoising diffusion probabilistic models (DDPM). DDPM [4] is a latent variable model where a denoising autoencoder gradually converts Gaussian noise into the original data. Compared to GANs, diffusion models produce higher quality images and exhibit better training stability. Consequently, they have garnered increasing attention in the field of image reconstruction. Building on this, the introduction of conditional constraints has further expanded their application scope, allowing models to generate corresponding outputs based on specific input conditions. Currently, conditional diffusion models are widely used in tasks ranging from text-to-image generation. CTIG-DM [40] introduces four types of text-image generation modes in the diffusion model, using images, text, and styles as conditions. DEADiff [18] proposes a mechanism to decouple the style and semantics of the reference image, achieving an optimal balance between the inherent text controllability of text-to-image models and the style similarity to the reference image. DiffuSeq[3] and DiffuSeq-v2[3] propose a diffusion model specifically designed for sequence-to-sequence (Seq2Seq) text generation tasks, demonstrating the enormous potential of diffusion models in complex conditional sequence generation tasks. The task of vector font reconstruction is viewed as a problem of reconstructing SVG drawing parameter sequences. We condition the vector diffusion model on the SDF values of raster glyph images, using the diffusion model to "denoise" the vector glyph representation into structured Bézier curves that represent the glyph.

2.3. Multi-Modality Representation Learning

Multimodal models involve the integration of various types of data and information, such as text, images, etc., to complement or align these pieces of information, thereby enhancing model performance and effectiveness. Multimodal representation can be divided into Joint Representation and Aligned Representation. In Joint Representation studies, models like BLIP [7], ViLBERT [15], and ALIGN [5] merge information from different modalities into a shared latent space through methods like feature vector concatenation and averaging, enabling joint representation for tasks like visual question answering and visual commonsense reasoning. Models such as CLIP [19]and A-CLP [34] align data from different modalities in the latent space to more effectively learn their correlations and mutual influences. This alignment method unifies visual and language tasks under a single framework, facilitating the fusion and modeling of multimodal information. Aligned Representation can handle both understanding-based tasks (like image-text retrieval) and generation-based tasks (like image captioning) within the model architecture. However, due to potential significant differences in the distributions of feature sets from different modalities, the representational capacity of fused features may be insufficient, thereby limiting model performance. In our approach, we leverage features from both glyph image and vector modalities, sharing a learned codebook across modalities, and encoding glyph image and sequence modalities into a unified embedding space. Using self-supervised learning, we predict the corresponding vector feature's discrete index for image features to achieve effective alignment between the two modalities.

3. Method Description

3.1. Problem Statement and Method Overview

Our goal is to develop a model capable of reconstructing vector images from raster glyph images. Since this conversion process involves cross-modal generation, it is highly challenging to apply traditional autoregressive or non-autoregressive models to generate outputs in a single step. Therefore, we employ a diffusion model that iteratively optimizes the generated sequence of drawing parameters, using a Transformer to capture the long-range dependencies and adapt to the irregularities of vector representations. To make the vector data suitable for the denoising diffusion model and the Transformer learning process, we convert the unstructured glyph vector image into structured tensor data during data preparation, which is then used for feature embedding. Additionally, we incorporate glyph image and vector modalities as guiding conditions in the model. We precompute the SDF values for each input glyph, enriching the model's learning capability by combining the global information from the glyph image's SDF values with the structural information from the vector modality.

To accomplish the cross-modal vector reconstruction task, we propose a novel cross-modal generation strategy. During training, we employ a variable-rate masking mechanism, gradually increasing the masking rate as training progresses, focusing on different modal information at different stages. Initially, the model emphasizes learning the features of the vector modality. As the masking rate increases, the model shifts its focus to learning the features of the image modality. During inference, the model uses the image modality features to predict the indices of the unmasked vector features, thereby substituting the features from the vector modality. Through this strategy, the model can fully rely on the image modality to complete the vector reconstruction task, achieving the conversion from glyph images to vector glyphs.

3.2. Data Preprocessing

3.2.1 Vector Glyph

A vector glyph is defined by a series of drawing parameters and is considered unstructured data (see Appendix A for more details). Due to varying levels of glyph complexity, the length of these drawing parameters may differ. To enable parallel processing by the model, we convert the unstructured drawing parameters into structured tensor data. Each vector glyph image consists of N drawing parameters S, where each drawing parameter $s_i = (c, p)$ is composed of a drawing command type and drawing coordinate parameters. The drawing command type is represented by commands $c \in \{< \text{SOS} >, \text{M}, \text{L}, \text{V}, \text{H}, \text{Q}, \text{Z}, < \text{EOS} >\}$, where $< \text{SOS} >$ and $< \text{EOS} >$ represent the start and end of the sequence, respectively. The drawing coordinate parameters are given as $p = (x_1, y_1, x_2, y_2)$. If the length of S is less than N, it is padded with -1 (see Appendix B for more details).

3.2.2 Signed Distance Function

The SDF value is calculated as the distance from each pixel to the nearest contour, with a sign indicating whether the pixel is inside or outside the contour. For a vector glyph image, it is first converted into a binary image B, where a pixel value of 1 indicates that it is outside the glyph, and 0 indicates that it is inside. The SDF value $X_{sdf}(x, y)$ at position (x, y) is then defined as

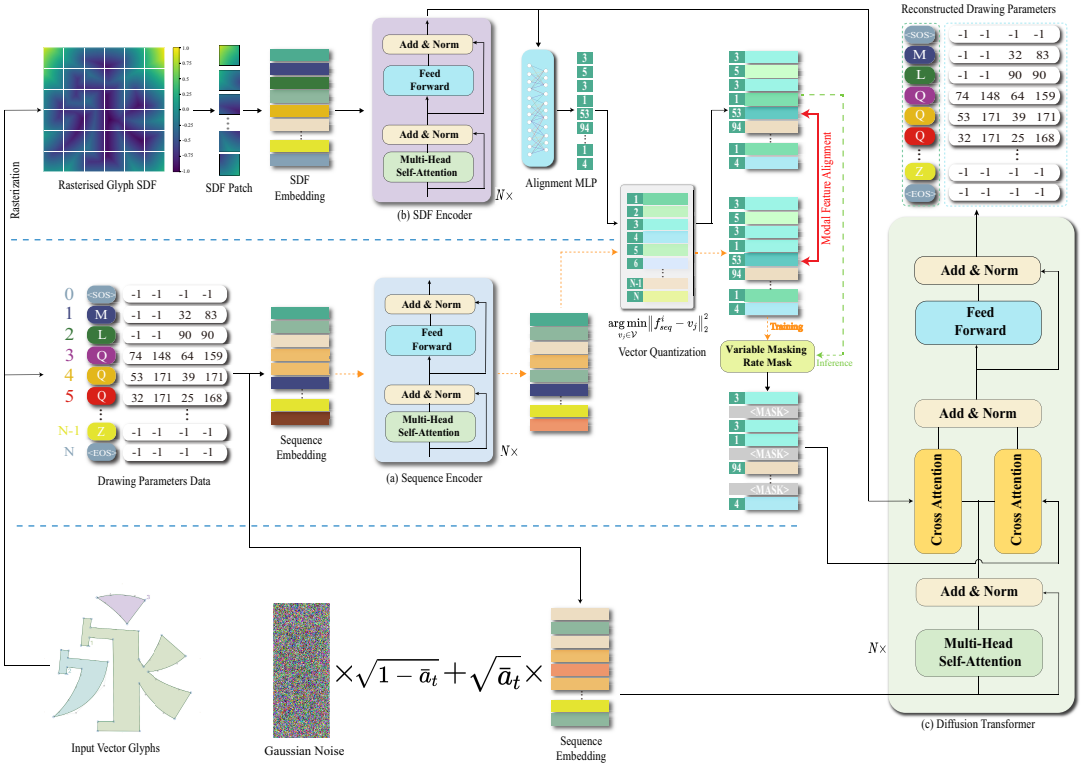


Figure 1. Detailed data flow of the SDF encoder and masked diffusion transformer model. Given a Chinese character from a font, the model receives its drawing parameters and SDF image as input, which are processed separately by the SDF decoder and sequence decoder to extract image features and vector features, respectively. The vector features, after vector quantization, are partially masked by 75%, and this masked part, along with the image features, is fed into the diffusion transformer model for reconstructing vector drawing parameters. Image features predict the discrete indices of corresponding vector features through an MLP layer to align the modalities. In the inference phase, the model uses the image modality to predict the quantized indices of unmasked vector features, thus enabling the model to generate corresponding vector drawing parameters relying solely on raster images.

$$X_{sdf}(x,y) = \begin{cases} d(x,y), & B(x,y) = 1 \\ -d(x,y), & B(x,y) = 0 \end{cases}. \quad (1)$$

Let $\mathcal{C}$ be the set of contour points. The SDF value for any pixel point (x, y) is

$$d(x,y) = \min_{(x_c,y_c)\in\mathcal{C}} \sqrt{(x-x_c)^2 + (y-y_c)^2}. \quad (2)$$

3.2.3 Embedding of Drawing Parameters Features

Given the discrete nature of the drawing parameters, we apply feature embedding techniques commonly used in natural language processing. For each drawing parameter s_i, we embed the drawing command type c. An 8-dimensional one-hot vector δ_c represents c, where only one dimension corresponds to the current command type index. This is then mapped to a vector of dimension d using a learnable matrix $W_{\text{cmd}} \in \mathbb{R}^{d\times 8}$, resulting in the embedded vector $e^i_{\text{cmd}} = W_{\text{cmd}}\delta_c$.

The size of the vector image is set to 128×128, and it includes a padding flag of -1. For the coordinates in the drawing parameter, a one-hot vector is first used to represent each coordinate, which is then mapped to a d-dimensional vector via a learnable weight matrix $W_x \in \mathbb{R}^{d\times 130}$. The embedded vector is computed as $e^i_{\text{coord}} = W_x\delta_{x_1} \oplus W_x\delta_{y_1} \oplus W_x\delta_{x_2} \oplus W_x\delta_{y_2}$. Finally, an absolute position encoding e^i_{pos} is introduced to help the model capture positional relationships within the sequence. The final feature embedding for each s_i is given by $z^i = (e^i_{\text{cmd}} \oplus e^i_{\text{coord}}) + e^i_{\text{pos}}$.

3.3. Vector Diffusion

In the forward process, noise is injected into the feature-embedded vector drawing parameters. After T steps of forward random perturbation, the parameters eventually become Gaussian noise.

$$q(z_t \mid z_{t-1}) = \mathcal{N}\left(z_t; \sqrt{1-\beta_t} z_{t-1}, \beta_t I\right), \quad (3)$$

where $t = 1, 2, \ldots, T$ and $\beta_1 < \ldots < \beta_T$ are predefined parameters that control the mean and variance of the noise. Let $\alpha_t = 1 - \beta_t$, and $\prod_{i=1}^{t} \alpha_i$, so that $\mathbf{z}_t$ at any time can be obtained as

$$z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon, \tag{4}$$

where ϵ stands for Gaussian noises. For the reverse process, conditioned on the information obtained from the image modality and vector modality c, the isotropic Gaussian noise is removed from z_T.

$$p_\theta \left(z_{t-1} \mid z_t, c \right) = \mathcal{N} \left(z_{t-1}; \mu_\theta \left(z_t, t, c \right), \sigma_\theta \left(z_t, t, c \right) \right), \tag{5}$$

where $\mu_\theta(\cdot)$ and $\sigma_\theta(\cdot)$ are the predicted parameterizations of the mean and standard deviation of $q\left(\mathbf{z}t \mid \mathbf{z}t - 1\right)$ in the forward process. Using Bayes' rule, we can obtain the parameterized mean of $q(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{z}_0)$ as

$$\mu_t \left(z_t, z_0 \right) = \frac{\sqrt{\alpha_t} \left(1 - \bar{\alpha}_{t-1} \right)}{1 - \bar{\alpha}_t} z_t + \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1 - \bar{\alpha}_t} z_0. \tag{6}$$

We use a Transformer architecture to model f_θ. The model estimates the final target result directly at each time step, predicting the vector drawing parameters in an iterative non-autoregressive manner. By applying the Clamping Trick [10] to map back from continuous latent space representations to discrete vector embeddings, we reduce errors caused by rounding or floating-point precision.

$$z_{t-1} = \sqrt{\bar{\alpha}} \cdot \text{Clamp} \left(f_\theta \left(z_t, t \right) \right) + \sqrt{1 - \bar{\alpha}} \cdot \epsilon. \tag{7}$$

The Clamp$(\cdot)$ operation maps $f_\theta\left(\mathbf{z}t, t\right)$ to the nearest vector embedding, ensuring that the generated $\mathbf{z}t - 1$ is precisely aligned with the discrete vector embedding. The objective function of the model is

$$\begin{aligned} \mathcal{L}_{\text{diff}} = \min_\theta \Bigg[& \sum_{t=2}^{T} \left\| z_0 - f_\theta \left(\hat{z}_t, t \right) \right\|^2 \\ & + \left\| \text{EMB} \left(S \right) - f_\theta \left(\hat{z}_1, 1 \right) \right\|^2 \\ & + \mathcal{R} \left(\left\| z_0 \right\|^2 \right) \Bigg]. \end{aligned} \tag{8}$$

where , the mathematically equivalent regularization term $\mathcal{R}\left(\left\|z_0\right\|^2\right)$ regularize the embedding learning.

3.4. Sequence Encoder

As shown in Figure 1 (a), the sequence encoder consists of six layers of multi-head self-attention Transformer modules . To enable the vector image V composed of drawing parameters to participate in model operations, it is first converted into structured tensor data S. After the feature embedding operation, it is input into the encoder to obtain the feature F_{seq} for the vector modality. We then introduce the vector modality into the diffusion model using cross-attention.

3.5. SDF Encoder

As shown in Figure 1 (b), we input the SDF values $Xsdf$ of the glyph vector image V into the image modality encoder, which uses a six-layer multi-head attention transformer as the encoder. We flatten it into SDF patches $X^{\text{p}} \in \mathbb{R}^{L \times P^2}$, where $L = \frac{H \times W}{P^2}$ is the number of SDF patches, and P is the size of each SDF patch.

Additionally, we extend the boundaries of the patches by o pixels to a size of $(P+2o) \times (P+2o)$. This extra extension ensures overlap between each patch, allowing the encoder to better understand adjacent patches and perceive positional information. Simultaneously, positional embeddings $E_{pos} \in R^{L \times d}$ are added to form patch embeddings. The patch embedding E_{sdf} is input into the SDF decoder to obtain the latent vector F_{sdf}. We then introduce F_{sdf} into the diffusion model using cross-attention.

3.6. Dual-modal Fusion

During the model training phase, we utilize features from both the glyph image modality and the vector modality for training. To ensure that during the inference phase, the model can solely rely on the image modality to complete the vector reconstruction task, we design a bimodal feature alignment strategy.

First, the features of the vector modality are quantized. A discrete latent space $V \in \mathbb{R}^{d \times K}$ is defined, where K is the size of the latent space, and d is the dimension of the latent embedding vector.

There exist K embedding vectors $v_j \in V$, $j \in \{1, 2, \ldots, K\}$. For each latent feature $f_{\text{seq}}^i \in F_{\text{seq}}$ in the vector modality, f_{seq}^i is mapped to the nearest embedding vector v_{seq}^j based on the L2 distance, as follows

$$v_{\text{seq}}^j = \arg\min_{v_j \in \mathcal{V}} \left\| f_{\text{seq}}^i - v_j \right\|_2^2. \tag{9}$$

The latent features F_{seq} in the sequence modality are mapped to a discrete latent space V through vector quantization, denoted as $\hat{F}_{\text{seq}}$. $Y_{\text{seq}} = [y_i]_{i=1}^N$ represents the discrete indices of the latent features in the sequence modality. The features $\hat{F}_{\text{seq}}$ are input to the diffusion model after masking 75% of them using the variable masking rate module.

During the model inference phase, the sequence encoder is removed. To compensate for the information loss caused by removal of the sequence decoder, we

employ self-supervised learning to predict the discrete indices of vector features from image features, and then use a shared codebook to restore the indices to original features. Specifically, the image modality features F_{img} are used to predict indices $Y_{\text{img}} = [y_i]_{i=1}^{N}$ through an MLP layer. The model minimizes the difference between Y_{img} and Y_{seq} using cross-entropy loss, enabling features from both modalities to be transformed into sequences of discrete indices in the same latent space.

The codebook, shared across modalities, is trained end-to-end through the loss function to optimize the differences between each latent feature in terms of codebook and sequence modality.

$$\mathcal{L}_{\text{VQ}} = \|\text{sg}[F_{\text{seq}}] - V\|_2^2 + \beta \|\text{sg}[V] - F_{\text{seq}}\|_2^2, \quad (10)$$

where $\text{sg}[\cdot]$ denotes the stop gradient operation, and β represents the loss weight for the commitment loss.

To align features in the discrete space between image and vector modalities, we use the sequence indices Y_{seq} as predicted target labels, aligning them with the indices of the image modality Y_{img} using cross-entropy loss.

$$\mathcal{L}_{\text{align}} = \sum_{i=1}^{N} \ell_{CE}\left(y_{\text{img}}^{i}, y_{\text{seq}}^{i}\right). \quad (11)$$

3.7. Variable Masking Rate

Initially, the model relies on the vector modality, but as the training progresses, it increasingly concentrates on learning image modality features. During the model training process, we apply a masking operation, denoted as $M = [m_i]_{i=1}^{L}$, to extract a subset by replacing part of the features in $\hat{F}$seq with a special $<$ MASK $>$ tag.

The mask is created using a variable-rate masked generation function. For each training sample, a masking rate $r \in [0, 1]$ is sampled from a truncated cosine distribution with density function $\gamma(r)$

$$\gamma(r) = \frac{\pi}{4} \cos\left(\frac{\pi}{2}(1-r)^2\right). \quad (12)$$

This distribution truncates at a masking rate of 0.75, tending to produce higher masking rates[36, 2]. We uniformly select $\gamma(\frac{t}{T}) \cdot L$ markers in $\hat{F}_{\text{seq}}$ at intervals linearly related to the training time, denoted by $\frac{t}{T}$, to place masks .

4. Experiments

4.1. Dataset

Vector Font Dataset: We collected 300 different styles of vector fonts from the official websites of Hanyi and Founder fonts for model training and testing purposes. A random selection of 250 fonts was used for training, while the remaining 50 fonts were used for testing.

Character Dataset: We selected a set of 3,500 commonly used Chinese characters based on their frequency of usage. Of these, 3,000 characters were used to train the model, and 500 characters were reserved for testing.

Data Augmentation: Each glyph vector image is defined by graphical parameters, where each parameter includes a drawing command type c and drawing coordinate parameters $p = (x_1, y_1, x_2, y_2)$. We introduce small offsets Δ_x and Δ_x to the coordinates, where $\Delta_x, \Delta_y \in [-10, 10]$, to translate the glyphs horizontally and vertically.

4.2. Comparison with Other Methods

Vector glyph reconstruction methods can be divided into two categories: vector-supervised and vector-unsupervised approaches. Among them, DeepSVG [1],VecFusion[28], DeepVecFont [30], and DeepVecFont-v2 [31] use self-supervised approaches to minimize the difference between input data and data reconstructed through encoders and decoders. These methods learn the mapping of vector glyph data to latent space and generate vector glyphs by sampling in latent space. Another category of methods adjusts the coordinate parameters of Bézier curves to reconstruct vector glyphs for raster images of glyphs. Examples include Im2Vec [21] and VecFontSDF [32].

4.2.1 English Vector Font Generation

In this experiment, we compared the performance of different methods in reconstructing vector images of English characters, with a qualitative comparison as shown in Figure 2. Among the two categories of methods compared, vector-supervised methods show significantly better quality in generating English vector glyphs compared to the other category. Im2Vec deforms any original closed shape as a deformation of the unit circle, which can only preserve rough shapes. VecFontSDF fits curves to all right angles, resulting in a lack of detail at glyph corners. DeepSVG overcomes the challenge of generating complex vector lines by decomposing vector graphics into smaller components; however, characters are usually composed of many continuous parts that can not be easily decomposed. DeepVecFont and DeepVecFont-v2 reconstruct vector glyphs in a self-supervised manner, capable of reconstructing complex vector glyphs, but the generated vector graphics have less smooth edges. Com-

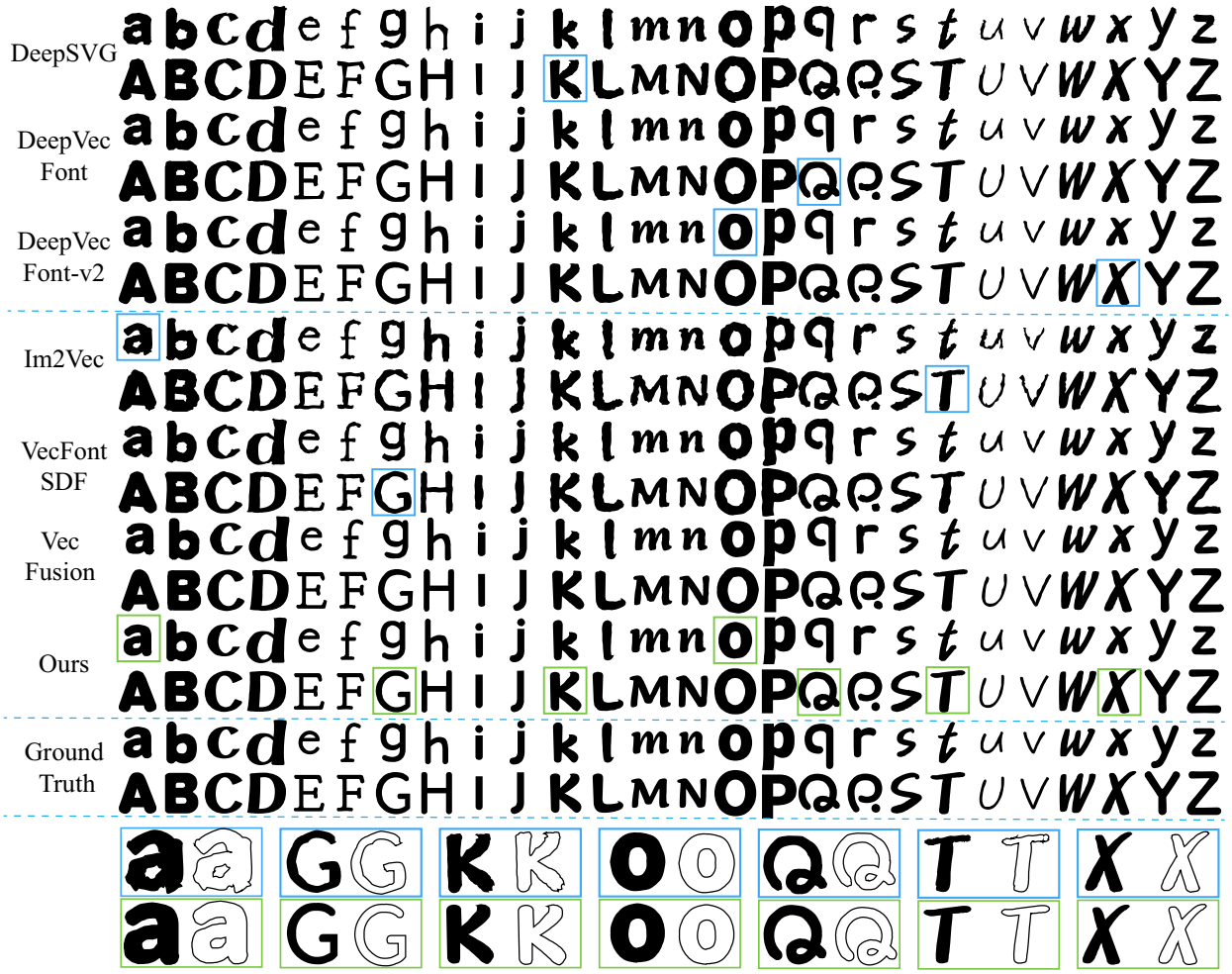


Figure 2. Qualitative comparison of English vector glyphs generated by our method and five other methods.

pared to the above methods, our method can generate English vector glyphs with clear outlines and sharp edges.

4.2.2 Chinese Vector Font Generation

To evaluate the effectiveness of these methods in generating Chinese vector fonts, we selected three simple Chinese characters and three complex Chinese characters for each font style. The quantitative comparisons are illustrated in Figure 3. Overall, vector-supervised methods demonstrate strong performance in generating vector glyphs. These methods, which are based on the AE model, generate vector glyph plotting parameters in an autoregressive manner, avoiding the complexities of cross-modal transformations from raster to vector, thereby achieving more precise results. While DeepVecFont and DeepVecFont-v2 produce satisfactory vector glyphs through vector supervision, they struggle to reconstruct vector images from raster glyphs. Our method accurately reproduces both the overall shape and finer details of raster glyph images. However, issues with contour line details persist across all methods. Even small deviations in each parameter coordinate can result in non-smooth contours in the rendered raster images, especially when a curve is composed of multiple Bézier curves .

4.2.3 Quantitative Comparison of Chinese Vector Fonts

Since the generated vector drawing parameters from different methods can not be aligned with real vector drawing parameters, we converted the generated vector glyphs into raster images for comparison. To quantitatively evaluate the performance of the models, we used L1, RMSE, SSIM, and LPIPS pixel-level evaluation metrics to measure the differences between the generated vector glyphs and real ones. Additionally, we conducted a user study with 30 volunteers. Each

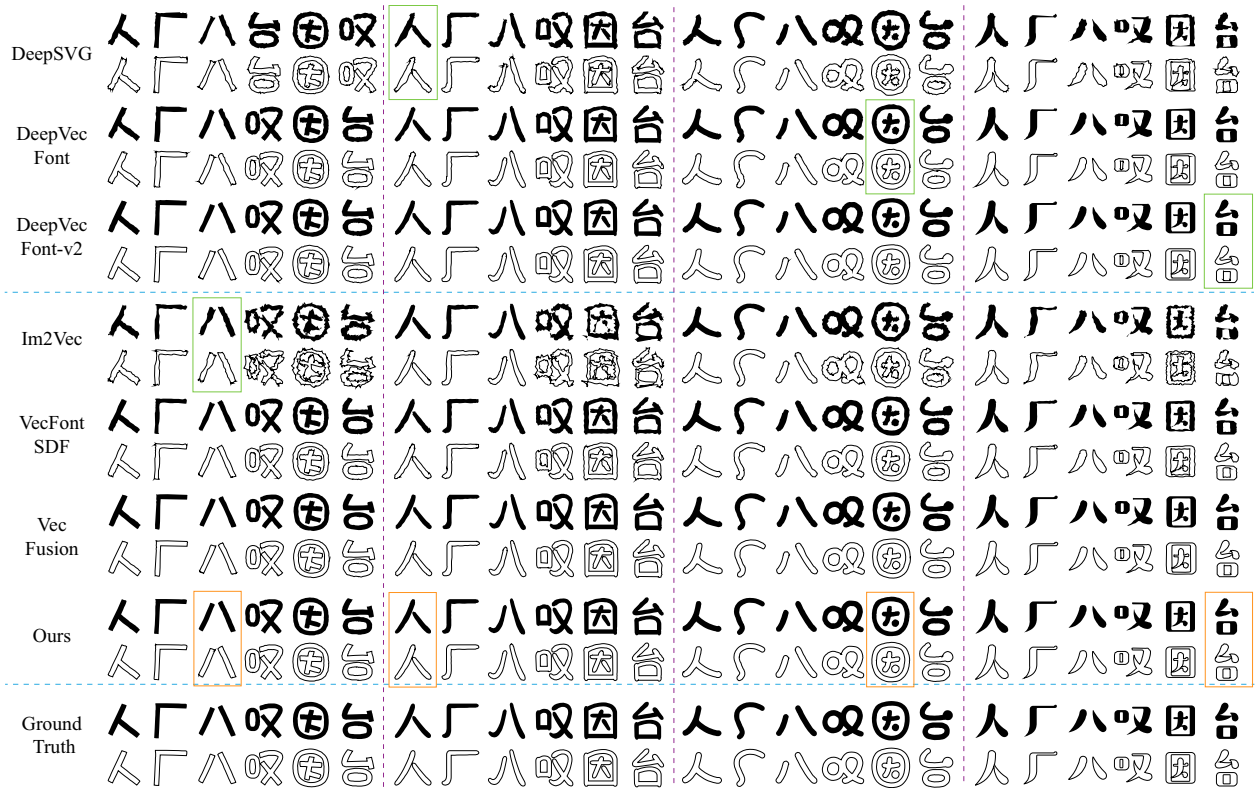


Figure 3. Qualitative comparison of our method with five other methods in Chinese vector glyph generation. Compared to other methods, our bimodal model-based approach achieves higher-quality outputs. Our method demonstrates advantages in handling complex characters and obtaining more accurate contours.

participant was presented with 50 characters generated by various methods mixed with 50 randomly selected characters from real fonts, and asked to identify the model-generated characters. The quantitative results are presented in Table 1.

Vector-unsupervised methods, which adjust vector plotting parameters to fit raster glyphs, can not reconstruct high-quality vector glyphs for complex Chinese characters. The other three methods, where the input and output of the model are of the same modality, are more stable and produce higher-quality results. Our method achieves results comparable to human-designed vector fonts, but due to the characteristics of Bézier curves, even small errors in the plotting parameter coordinates can lead to edge noise in the rendered raster images.

4.2.4 Compactness of Drawing Commands Analysis

We calculated the number of drawing command types used in reconstructing vector glyphs for each method, where a smaller number indicates a more compact representation of the generated vector glyphs. For each method, we selected ten different font styles and calculated the number of drawing commands used to reconstruct 200 characters for each font, then averaged the results. The results are shown in Table 2. Methods that adjust Bézier curve coordinates to fit glyph raster images generate vector drawing parameters with a considerable amount of redundancy. Im2Vec, which uses a fixed number of drawing parameters, contains more redundancy than VecFontSDF. Vector-supervised models sample from the original plotting parameters and are supervised using vector data, resulting in a generated number of plotting parameters closer to the real number of plotting parameters. Our method, which combines multimodal data with vector data supervision, achieves a level of compactness comparable to that of human-designed fonts.

4.2.5 Comparison with Vectorisation Tools

In addition to the deep learning-based methods discussed earlier, several computer graphics techniques are widely used for raster-to-vector image conversion. To comprehensively evaluate the effectiveness of our method, we qualitatively compare it with Potrace [23] and Adobe Image Trace (AIT). Potrace approximates

Table 1. Quantitative comparison between our method and the state-of-the-art methods for vector font generation.

Methods	Vector Supervision	Image-to-vector	L1 loss ↓	RMSE ↓	SSIM ↑	LPIPS ↓	User ↓
Im2Vec[21]	×	✓	0.0578	0.652	0.421	0.823	0.98
VecFontSDF[32]	×	✓	0.0451	0.571	0.551	0.601	0.81
DeepSVG[1]	✓	×	0.0614	0.623	0.483	0.782	0.86
DeepVecFont[30]	✓	×	0.0442	0.587	0.542	0.584	0.79
DeepVecFont-v2[31]	✓	×	0.0402	0.501	0.621	0.493	0.67
VecFusion[28]	✓	✓	0.0395	0.492	0.664	0.486	0.66
Ours	✓	✓	0.0312	0.443	0.711	0.431	0.60

Table 2. The average number of commands used per glyph for different methods in the font generation task.

Methods	Vector Supervision	Image-to-vector	M	L	H	V	Q	C	Total ↓
Im2Vec[21]	×	✓	8	0	0	0	0	160	168
VecFontSDF[32]	×	✓	8.61	0	0	0	78.8	0	87.41
DeepSVG[1]	✓	×	7.12	26.8	0	0	40.2	0	74.12
DeepVecFont[30]	✓	×	6.12	21.7	0	0	0	34.8	62.62
DeepVecFont-V2[31]	✓	×	5.31	17.2	0	0	0	29.5	53.01
VecFusion[28]	✓	✓	8.34	16.1	0	0	48.4	0	72.84
Ours	✓	✓	5.42	14.6	7.4	6.8	24.2	0	58.42
Human-Designed	N/A	N/A	4.41	12.4	5.4	5.3	24.2	0	51.71

polygons with Bézier curves to generate vector images. The qualitative results are shown in Figure 4, with different colors used to identify each vector contour. Potrace tends to use more Bézier curves to finely fit bitmap images. While AIT produces vector glyphs that avoid most of the artifacts found in Potrace's output, it still introduces a significant amount of redundant drawing parameters. Our method, by learning a substantial amount of prior information about vector fonts, produces vector glyphs that are more accurate and closer in quality to manually vectorized glyphs than those generated by other methods.

4.3. Ablation Studies

4.4. Impact of Masked Generation Function

In our method, the quality of generated vector glyphs is significantly influenced by the Masked Generation function. This function calculates the mask ratio of latent features at different training times $0/T, 1/T, \cdots, (T-1)/T$. The Masked Generation function $\gamma(r) \in [0, 1]$ must be a continuous function that is monotonically increasing with respect to r, satisfying $\gamma(0) \to 0$ and $\gamma(1) \to 0.75$. We considered some common functions and made slight modifications to meet these properties. These functions fall into three categories, as shown in Figure 5. The first category is linear functions, where the masked token count increases linearly with r. The second category consists of concave functions, where the model needs to mask a small number of features initially and sharply increases the masking rate towards the end. The third category comprises convex functions, which are the opposite of concave functions, sharply increasing the masking rate initially and gradually increasing it later on.

In the ablation study, we conducted experiments using only different Masked Generation functions while keeping other settings constant. After each training iteration t, an inference experiment was performed. In each inference, drawing parameters were generated for the same 100 glyphs, rendered into raster images, and the L1 loss with respect to the ground truth raster images was calculated and visualized in Figure 6. Generally, all functions achieve similar performance midway through training, but the performance of concave functions deteriorates as training continues. Initially, lower mask rates enable the model to achieve higher performance levels because it can extract sufficient information from the sequence modality. However, with the rapid increase in mask rates later on, the model's performance deteriorates due to the introduction of more uncertain information. Further research on Masked Generation functions will be a key direction for future work.

4.5. Feature Visualization Validation

To validate the effectiveness of our proposed cross-modal vector reconstruction strategy, we conducted a feature visualization experiment. As is shown in Figure 7, we presented the visualization of attention scores at $t/T = 0.1$, $t/T = 0.3$, and $t/T = 1.0$ respectively. In the initial phase at $t/T = 0.2$, the model primarily focuses on the features of the vector modality, with relatively less attention to the image modality. As the training progresses, the model's dependency on vector modality features gradually decreases, shifting its fo-

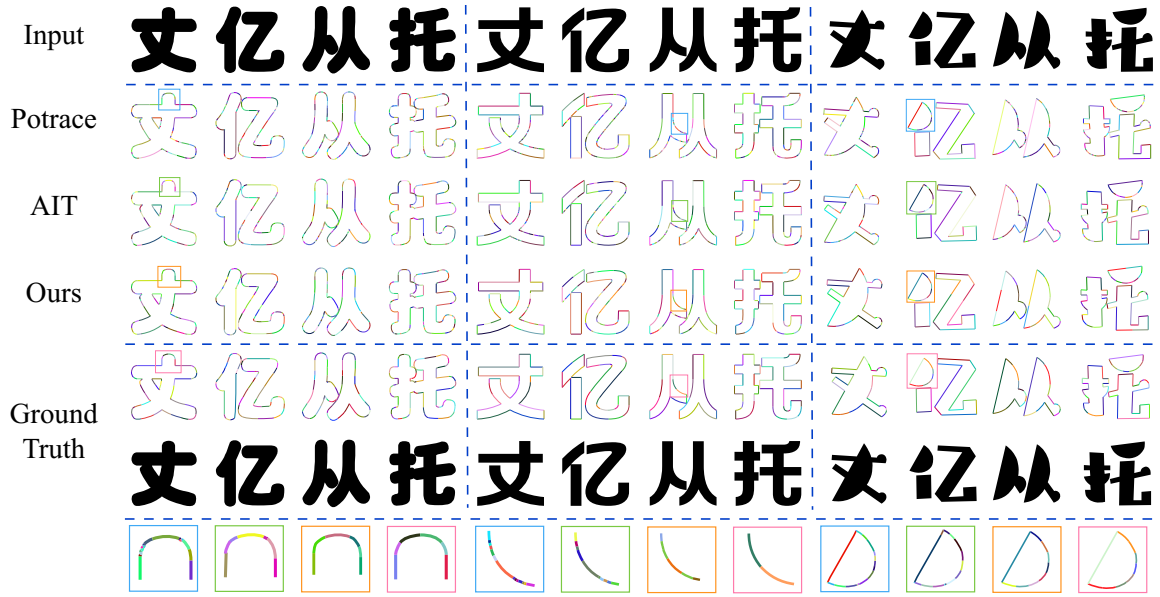


Figure 4. A comparison between our method and AIT and Potrace. Different colors are used to annotate each drawing command.

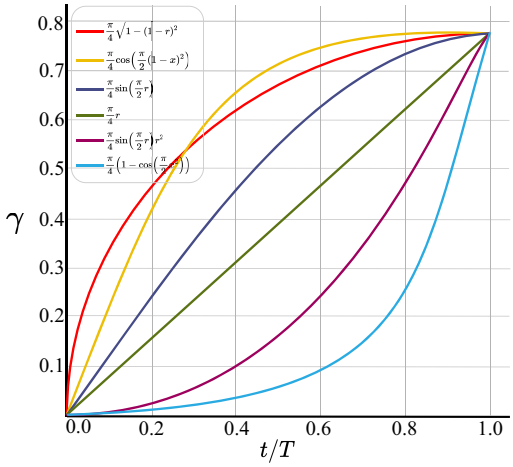


Figure 5. The six masked generation functions we selected.

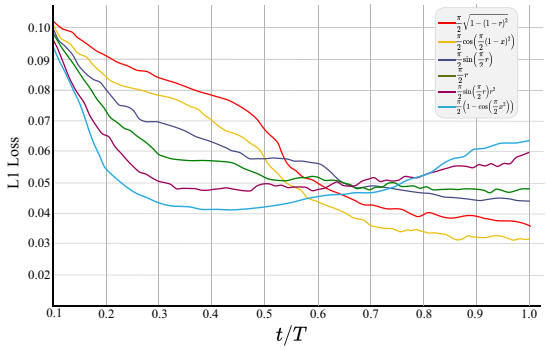


Figure 6. Relationship between L1 Loss and model iteration T under different masked generation function settings.

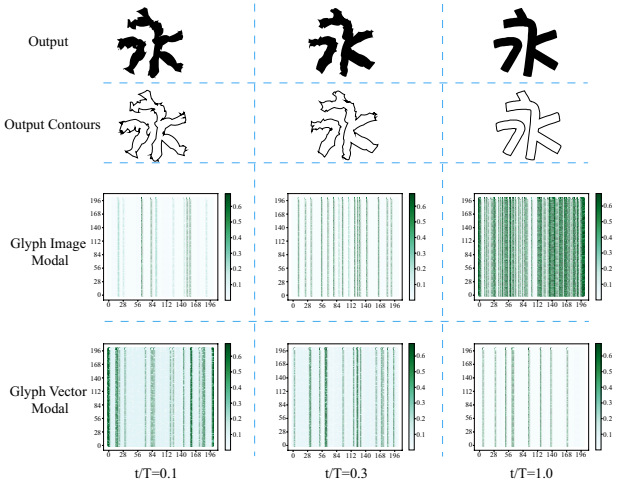


Figure 7. Visualization of attention scores across different training phases to demonstrate the effectiveness of the proposed cross-modal vector reconstruction strategy

cus more towards the image modality. By the time it reaches $t/T = 1.0$, 75% of the features in the vector modality are masked, and the model extracts fewer features from the vector modality, significantly enhancing the focus on image modality features. This shift validates the effectiveness of our proposed cross-modal vector reconstruction strategy.

4.6. Multi-Language Vector Font Generation

In this experiment, we further validated the ability to generate vector glyphs for different languages. We focused on Japanese and Korean, which are two promi-

Figure 8. Japanese and Korean vector glyph image generation experiments.

nent Asian languages written in two-dimensional forms and have more diverse styles compared to alphabetic languages. The generation results, shown in Figure 8, indicate that our method produces vector fonts for Japanese and Korean with results comparable to the target glyphs. This further underscores the effectiveness of our method in cross-language scenarios.

4.7. Vector Icon Generation

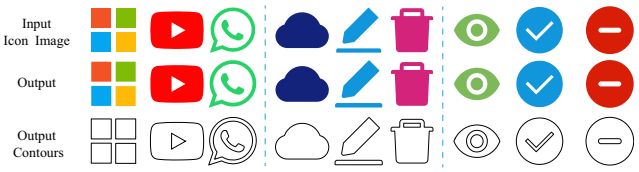


Figure 9. Experiments on generating vector icons.

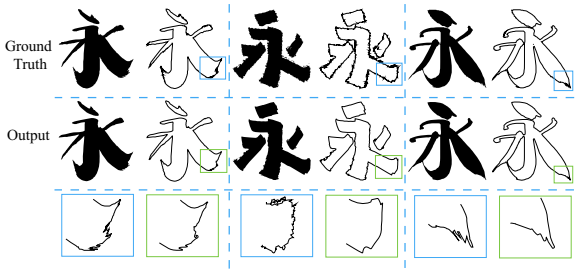


Figure 10. Tend to use smooth curve fitting for complex shapes.

To further showcase its capability in reconstructing vector icons, we collected samples of vector icons from different application domains on the Icons8 website, including commonly used vector icons in communication, entertainment, social media, and utility tools. After fine-tuning our model on these datasets, the generated vector icon results are presented in Figure 9. Our model can accurately handle icons filled with details and smoothly curved lines. This further validates the versatility and practical value of our method.

5. Conclusion and Future Work

Our proposed cross-modal generation strategy excels not only in technical aspects but also lays the groundwork for future work and provides strong references for researchers in the field. This report introduces a masked diffusion transformer model that integrates vector and image modalities, effectively "upsampling" low-resolution raster glyph images into vector representations. Our proposed cross-modal generation strategy not only excels in technical aspects but also lays a solid foundation for future research, providing valuable insights for scholars in the field. Limitations: As shown in Figure 10, our method tends to use smooth curve fitting when generating glyph vector outlines for characters with many curved paths, overlooking their detailed shapes. Future work: As the research progresses, we expect our method and its derivative techniques to find broader applications across various domains. Future work will focus on further improving model performance and expanding to reconstruction in more languages and vector graphics.

Acknowledgement

This study was supported in part by Liaoning Provincial Science and Technology Plan Joint Program (Technology R&D Program Project) under Grants 2024JH2/102600108, the Science and Technology Innovation Fundation of Dalian under Grant 2023JJ12GX026, and in part by the Foundation of Key Laboratory of Education Informatization for Nationalities (Yunnan Normal University, Ministry of Education) under Grant EIN2024B002.

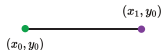
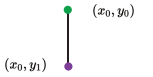
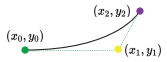
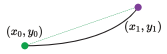
A. Appendix

A.1. Vector Font

Vector fonts define glyphs precisely using mathematical equations, drawing contours of Chinese characters using lines and Bezier curves. This approach grants vector fonts scalability and editability, showcasing high flexibility in design and application. In recent years, researchers have extensively explored glyph reconstruction in vector glyphs. The method based on Bezier curves is a commonly used approach [20]. Bezier curves in vector fonts are described by general SVG drawing commands. SVG, as an open standard vector graphics format, accurately and flexibly presents the shapes of Chinese characters [22]. SVG uses a series of commands to describe the shapes to create, and through the drawing commands in Table 3, each glyph in the

font can be drawn. As shown in Figure 11, in vector fonts, each glyph consists of a different number of SVG drawing commands, with each command representing a parameterized shape contour.Unlike traditional raster font image generation, the creation of vector glyphs resembles a cross-modal generation task from image to text[25, 16]. However, compared to the flexibility of natural language generation, generating SVG drawing parameters requires strict syntax rules and structures, making it more complex and challenging for vector glyph generation.

Table 3. Vector glyphs consist of a different number of SVG drawing commands, with each drawing command representing a parameterized shape contour.

Commands	Arguments	Example	Description
M	x, y		Move from (x_0, y_0) to (x_1, y_1)
L	x, y		Draw a line from (x_0, y_0) to (x_1, y_1)
H	x, y		Draw horizontal line from (x_0, y_0) to (x_1, y_0)
V	x, y		Draw vertical line from (x_0, y_0) to (x_0, y_1)
Q	x_1, y_1 x_2, y_2		Draw quadratic Bessel curve from (x_0, y_0) to (x_2, y_2) ,Control point (x_1, y_1)
Z	$\varnothing$		Close path from (x_0, y_0) to (x_0, y_1)

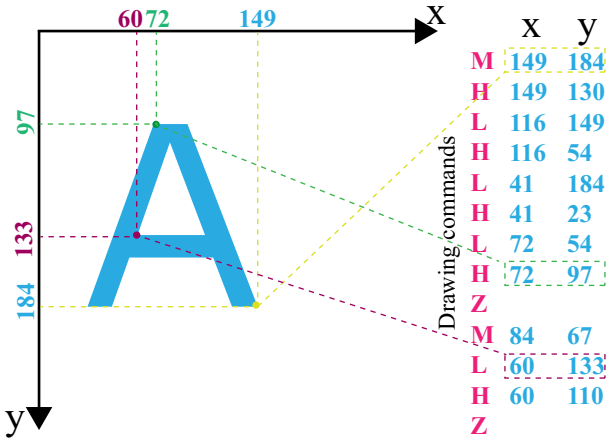
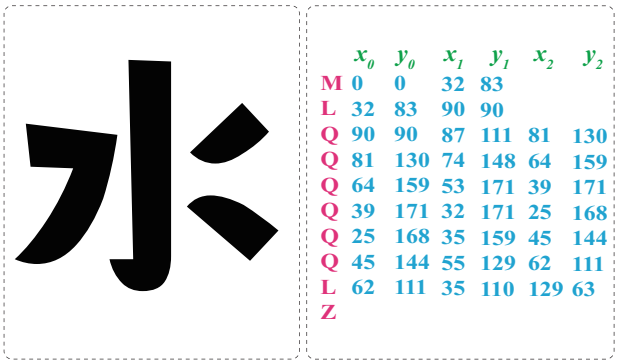
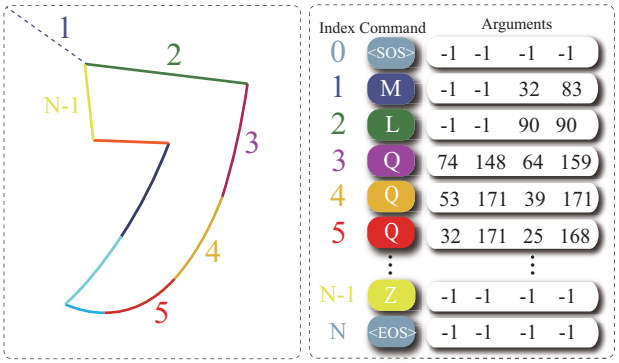


Figure 11. A vector glyph is composed of a varying number of SVG drawing commands, with each command representing a parameterized shape contour.



(a) Glyph Vector Image (b) SVG Drawing Parameters



(c) Stroke Example (d) Structured Drawing Parameters

Figure 12. Visualization of the drawing parameter data structure. Left: glyph vector images and their SVG representations. Right: Vector data tensor representation with an example stroke.

A.2. Vector Data

Glyph raster images are represented as fixed-size arrays of pixels, while vector glyphs are defined by a series of drawing parameters and appear as unstructured data. Due to variations in glyph complexity, the lengths of drawing parameters can differ. To enable parallel processing by the model, we convert unstructured drawing parameters into structured tensor data.

Each glyph vector image consists of N drawing parameters S. Each drawing parameter $s_i = (c, p)$ consists of its drawing command type and drawing coordinate parameters. Drawing command types such as those listed in Table 3 are represented by commands $c \in \{< \text{SOS} >, \text{M}, \text{L}, \text{V}, \text{H}, \text{Q}, \text{Z}, < \text{EOS} >\}$, where $< \text{SOS} >$ and $< \text{EOS} >$ denote special markers indicating the beginning and end of the sequence, respectively. These markers help the model identify the boundaries of drawing parameters during data processing, aiding the model in correctly initiating and terminating the generation process. We use fixed-length drawing coordinate parameters $p = (x_1, y_1, x_2, y_2)$. If the length of the drawing command S is less than N,

we pad it with -1. Figure 12 illustrates an example of a Chinese glyph vector image and its tensor representation, focusing on one stroke. The stroke begins with the special marker $<$ SOS $>$ denoting the start of the drawing command, followed by the command M for moving to the starting point, then drawing a straight line L, followed by a series of Bezier curve Q, and finally, the command Z to close the path, with the $<$ EOS $>$ command indicating the end of the drawing process.

References

[1] A. Carlier, M. Danelljan, A. Alahi, and R. Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. Advances in Neural Information Processing Systems, 33:16351–16361, 2020. 2, 3, 7, 10

[2] H. Chang, H. Zhang, L. Jiang, C. Liu, and W. T. Freeman. Maskgit: Masked generative image transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11315–11325, 2022. 7

[3] S. Gong, M. Li, J. Feng, Z. Wu, and L. Kong. Diffuseq-v2: Bridging discrete and continuous text spaces for accelerated seq2seq diffusion models. In Findings of the Association for Computational Linguistics: EMNLP 2023, pages 9868–9875, 2023. 3

[4] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. Advances in neural information processing systems, 33:6840–6851, 2020. 3

[5] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In International conference on machine learning, pages 4904–4916. PMLR, 2021. 4

[6] C. Li, Y. Taniguchi, M. Lu, S. Konomi, and H. Nagahara. Cross-language font style transfer. Applied Intelligence, 53(15):18666–18680, 2023. 2

[7] J. Li, D. Li, C. Xiong, and S. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In International Conference on Machine Learning, pages 12888–12900. PMLR, 2022. 4

[8] Q. Li, J.-P. Li, and L. Chen. A bezier curve-based font generation algorithm for character fonts. In 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pages 1156–1159. IEEE, 2018. 2

[9] T.-M. Li, M. Lukáč, M. Gharbi, and J. Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. ACM Transactions on Graphics (TOG), 39(6):1–15, 2020. 3

[10] X. Li, J. Thickstun, I. Gulrajani, P. S. Liang, and T. B. Hashimoto. Diffusion-lm improves controllable text generation. Advances in Neural Information Processing Systems, 35:4328–4343, 2022. 6

[11] J.-Y. Liu, Y.-M. Zhang, F. Yin, and C.-L. Liu. Transformer-based stroke relation encoding for online handwriting and sketches. Pattern Recognition, 148:110131, 2024. 2

[12] Y. Liu, F. binti Khalid, C. Wang, M. R. binti Mustaffa, and A. bin Azman. An end-to-end chinese font generation network with stroke semantics and deformable attention skip-connection. Expert Systems with Applications, 237:121407, 2024. 2

[13] Y.-T. Liu, Z. Zhang, Y.-C. Guo, M. Fisher, Z. Wang, and S.-H. Zhang. Dualvector: Unsupervised vector font synthesis with dual-part representation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14193–14202, 2023. 2, 3

[14] R. G. Lopes, D. Ha, D. Eck, and J. Shlens. A learned representation for scalable vector graphics. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 7930–7939, 2019. 2, 3

[15] J. Lu, D. Batra, D. Parikh, and S. Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. Advances in neural information processing systems, 32, 2019. 4

[16] Y. Ma, J. Ji, X. Sun, Y. Zhou, and R. Ji. Towards local visual modeling for image captioning. Pattern Recognition, 138:109420, 2023. 13

[17] S. Park, S. Chun, J. Cha, B. Lee, and H. Shim. Few-shot font generation with localized style representations and factorization. In Proceedings of the AAAI conference on artificial intelligence, volume 35, pages 2393–2402, 2021. 2

[18] T. Qi, S. Fang, Y. Wu, H. Xie, J. Liu, L. Chen, Q. He, and Y. Zhang. Deadiff: An efficient stylization diffusion model with disentangled representations. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024. 3

[19] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In International conference on machine learning, pages 8748–8763. PMLR, 2021. 4

[20] N. K. B. Razali, N. N. B. C. Draman, S. M. B. Nor-Al-Din, and N. B. M. Sukri. Cubic curve fitting method in reconstruction of chinese calligraphy outline. In Journal of Physics: Conference Series, volume 2084, page 012020. IOP Publishing, 2021. 12

[21] P. Reddy, M. Gharbi, M. Lukac, and N. J. Mitra. Im2vec: Synthesizing vector graphics without vector supervision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7342–7351, 2021. 3, 7, 10

[22] Y. Ren, S. Liu, Y. Jiang, J. Yan, and Y. Gao. Design and draw patterns based on bezier curves. In 2022 International Symposium on Intelligent Robotics and Systems, pages 13–15. IEEE, 2022. 12

[23] P. Selinger. Potrace: a polygon-based tracing algorithm, 2003. 9

[24] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In International conference on machine learning, pages 2256–2265. PMLR, 2015. 3

[25] J. H. Tan, C. S. Chan, and J. H. Chuah. End-to-end supermask pruning: Learning to prune image captioning models. Pattern Recognition, 122:108366, 2022. 13

[26] S. Tang and Z. Lian. Write like you: Synthesizing your cursive online chinese handwriting via metric-based meta learning. In Computer Graphics Forum, volume 40, pages 141–151. Wiley Online Library, 2021. 2

[27] S. Tang, Z. Xia, Z. Lian, Y. Tang, and J. Xiao. Fontrnn: Generating large-scale chinese fonts via recurrent neural network. In Computer Graphics Forum, volume 38, pages 567–577. Wiley Online Library, 2019. 2

[28] V. Thamizharasan, D. Liu, S. Agarwal, M. Fisher, M. Gharbi, O. Wang, A. Jacobson, and E. Kalogerakis. Vecfusion: Vector font generation with diffusion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7943–7952, 2024. 7, 10

[29] C. Wang, M. Zhou, T. Ge, Y. Jiang, H. Bao, and W. Xu. Cf-font: Content fusion for few-shot font generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1858–1867, 2023. 2

[30] Y. Wang and Z. Lian. Deepvecfont: Synthesizing high-quality vector fonts via dual-modality learning. ACM Transactions on Graphics (TOG), 40(6):1–15, 2021. 3, 7, 10

[31] Y. Wang, Y. Wang, L. Yu, Y. Zhu, and Z. Lian. Deepvecfont-v2: Exploiting transformers to synthesize vector fonts with higher quality. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 18320–18328, 2023. 3, 7, 10

[32] Z. Xia, B. Xiong, and Z. Lian. Vecfontsdf: Learning to reconstruct and synthesize high-quality vector fonts via signed distance functions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1848–1857, 2023. 2, 3, 7, 10

[33] Y. Xie, X. Chen, L. Sun, and Y. Lu. Dg-font: Deformable generative networks for unsupervised font generation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 5130–5140, 2021. 2

[34] Y. Yang, W. Huang, Y. Wei, H. Peng, X. Jiang, H. Jiang, F. Wei, Y. Wang, H. Hu, L. Qiu, and Y. Yang. Attentive mask clip. In 2023 IEEE/CVF International Conference on Computer Vision (ICCV), pages 2771–2781, 2023. 4

[35] Y.-T. Yang, Y.-M. Zhang, X.-L. Yun, F. Yin, and C.-L. Liu. Dygat: Dynamic stroke classification of online handwritten documents and sketches. Pattern Recognition, 141:109564, 2023. 2

[36] L. Yu, Y. Cheng, K. Sohn, J. Lezama, H. Zhang, H. Chang, A. G. Hauptmann, M.-H. Yang, Y. Hao, I. Essa, et al. Magvit: Masked generative video transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10459–10469, 2023. 7

[37] J. Zeng, Q. Chen, Y. Liu, M. Wang, and Y. Yao. Strokegan: Reducing mode collapse in chinese font generation via stroke encoding. In Proceedings of the AAAI conference on artificial intelligence, volume 35, pages 3270–3277, 2021. 2

[38] B. Zhao, J. Tao, M. Yang, Z. Tian, C. Fan, and Y. Bai. Deep imitator: Handwriting calligraphy imitation via deep attention networks. Pattern Recognition, 104:107080, 2020. 2

[39] S. Zhou, D. Gao, and D. Zhou. Quick draw of the original handwriting base on quadratic bezier curve. International Journal of Wavelets, Multiresolution and Information Processing, 14(04):1650025, 2016. 2

[40] Y. Zhu, Z. Li, T. Wang, M. He, and C. Yao. Conditional text image generation with diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14235–14245, 2023. 3