DiffCJK: Conditional Diffusion Model for High-Quality and Wide-coverage CJK Character Generation

Yingtao Tian
Google DeepMind
Tokyo, Japan
alantian@google.com

Abstract

Chinese, Japanese, and Korean (CJK), with a vast number of native speakers, have profound influence on society and culture. The typesetting of CJK languages carries a wide range of requirements due to the complexity of their scripts and unique literary traditions. A critical aspect of this typesetting process is that CJK fonts need to provide a set of consistent-looking glyphs for approximately one hundred thousand characters. However, creating such a font is inherently labor-intensive and expensive, which significantly hampers the development of new CJK fonts for typesetting, historical, aesthetic, or artistic purposes.

To bridge this gap, we are motivated by recent advancements in diffusion-based generative models and propose a novel diffusion method for generating glyphs in a targeted style from a *single* conditioned, standard glyph form. Our experiments show that our method is capable of generating fonts of both printed and hand-written styles, the latter of which presents a greater challenge. Moreover, our approach shows remarkable zero-shot generalization capabilities for non-CJK but Chinese-inspired scripts. We also show our method facilitates smooth style interpolation and generates bitmap images suitable for vectorization, which is crucial in the font creation process. In summary, our proposed method opens the door to high-quality, generative model-assisted font creation for CJK characters, for both typesetting and artistic endeavors.

Introduction

The importance of East Asian languages that use Chinese characters, including Chinese, Japanese and Korean (CJK) languages, is profound. For example, they are used by more than 1.56 billion speakers (Ethnologue Authors 2024) which account for more than 25% of global GDP (IMF 2023). Also, historically, a wide range of scripts have been heavily influenced by Chinese characters, adding more historical and culture value. In the modern era, with the proliferation of printing and information technology, writing scripts need to address the challenges typesetting: for example, the requirements on the visual appearance of characters means that a font must provide well-designed and consistent-looking glyphs across the glyph repertoire.

However, the unique aspect for CJK typesetting is the sheer number of characters that makes such requirement demande more effort: CJK contains nearly one-hundred thousand unique characters in the latest Unicode 15.1 (Unicode 2023). Furthermore, there is also an emphasis on typefaces categories with different styles[1], as they carry practical, cultural, and artistical meaning (Huang 2020; Kim 2020a; 2020b). Consequently, the font industry faces a dilemma: either making one style instance with as wide of coverage as possible, or making fonts with wider style support but with limited coverage. Despite these compromises, font creation

Figure 1: Our method generates *highly stylized* and *legitimate* CJK glyphs, For each character, our method refers to a standard font's bitmap (visualized in gray) and generates a diverse array of glyphs in various printed and calligraphy form (More details in Figure 8.) Our method is effective for both common (left) and extremely rare (right) CJK characters. The zoom-ins showcase examples of printed and calligraphy form, highlighting the method's high quality and its utility for font designers and artists alike.

[1]Here "style" is in a general sense rather than its specific meaning in font development as in "style instance".

remains an laborious and expensive task. Therefore, it remains a critical challenge to efficiently producing fonts that encompass a diverse range of *various, multi-style* yet *legitimate* CJK glyphs for the entire set of nearly one hundred thousand characters.

To address these challenges, we propose a diffusion-based method (Figure 1) capable of generating characters in diverse styles conditioned on a *single* standard reference glyph. The reference glyph only needs to outline the character's shape, obviating the need for intricate design work. As a diffusion model, it consists of two processes: a diffusion process that gradually adds noise to destroy the data, and a reverse process that generates new samples from scratch by progressively removing noise. A deep neural network is trained to estimate the noise, conditioned on the reference character. Unlike common text-to-image models which must memorize the shapes of characters and thus suffering from limited data of rare characters, our method works uniformly well for all CJK glyphs, from common to rare ones, as long as we have an available reference. Furthermore, by injecting style information in the temporal embedding for diffusion, our method can not only generate characters in various styles but also interpolate between these styles.

Experiments show that our method can generating legitimate CJK characters across a broad spectrum of styles. This includes typefaces for physical and digital typesetting, as well as artistic calligraphy styles. The later poses a greater challenge due to more limited data and significant stylistic deviations from reference glyphs. Moreover, our method achieves zero-shot generalization to CJK-inspired scripts not encountered in training (such as the under-resourced Cho Num and Tangut scripts) and can meaningfully interpolate between styles. These capabilities enable font designers to produce fonts on a large scale, even with limited resources. Additionally, our analysis highlights the efficiency of our method and its potential for vectorization, indicating its practicality for adoption and adaptation in font design workflows.

Background and Related Works

Chinese Character in CJK Typesetting

Chinese characters constitute a logographic writing system, circa 13th century BCE to present, utilized across multiple languages within the Sinosphere, or the East Asian cultural sphere (Marginson 2011). Initially developed for Chinese, it has also been adopted for Japanese, Korean, Vietnamese, and has significantly influenced lesser-known, historical scripts like the Tangut and Khitan small scripts.

Like for any writing system, both physical (Needham 1974) and digital (Liu 2022) typesetting has a profound impact in East Asian culture. In such typesetting practices, alongside *how characters are arranged*, an equally important aspect is *what characters look like* (Chiba and others 2020; Tung and others 2023; Lim and others 2020) which necessitates *glyphs* that are legitimate, consistent, and available for the vast number of characters. This requirement is deeply rooted in pre-typesetting traditions, particularly the established script styles for handwriting. As shown in

Oracle bone script (甲骨文)	Seal script (篆書)	Clerical script (隸書)	Semi-cursive script (行書)	Cursive script (草書)	Regular script (楷書)

Figure 2: Example of hand-written Chinese script styles for "馬" (horse) (Wikipedia contributors 2024).

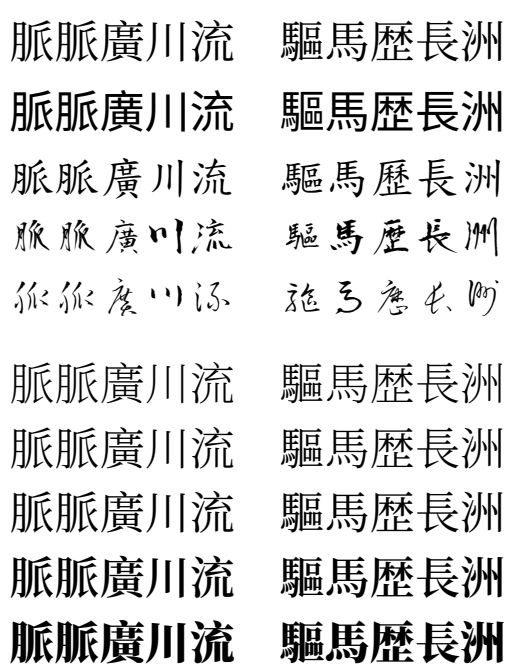

Figure 3: Example of CJK typefaces (above) and weights (below). **Typefaces**: (1) *Ming* a.k.a. *Song*, (2) *Gothic typefaces*, (3) *Regular script*, (4) *Semi-cursive* and (5) *cursive* script. **Widths**: Noto Serif CJK of different width: *ExtraLight*, *Light*, *Normal*, *SemiBold* and *Black* (i.e. *Bold*.)

Figure 2, there are five major script styles — *seal*, *clerical*, *semi-cursive*, *cursive* and *regular* style — traditionally used to write Chinese characters, in additional to oracle bone script, the oldest known form of Chinese characters. These styles naturally transitioned into typesetting (Lisa Huang 2020) practice which will be elaborated below. In the context of typesetting, the term **CJK** characters is used to collectively describe the glyphs for Chinese, Japanese, and Korean languages, all of which incorporate Chinese characters into their writing systems. Technically, CJK also encompasses derivatives of Chinese characters, such as the Kana scripts (Hiragana and Katakana) used in Japanese, although these are beyond the scope of this paper.

The importance of glyphs in typesetting is multifaceted, with style (typefaces) and weight (Stocks 2020a; 2020b) being two crucial aspects. In Latin typesetting, well-recognized styles include the serif ("brown fox") and sans-serif ("brown fox"), while weight variations are exempli-

fied by bold (**"brown fox"**) and italics (*"brown fox"*). CJK typesetting is not an exception, and even encompasses more delicacy regarding this matter: it requires a consistent appearance across a vast number of characters combined with unique cultural practices. Also, CJK typesetting places a significant emphasis on typeface categories, which carry practical, cultural, and artistic significance (Huang 2020; Kim 2020a; 2020b).

Notably, styles in CJK typesetting, while similar, do not entirely align with Latin typesetting practices. They are better shown by several key styles unique to CJK typography in Figure 3: (1) *Ming/Song*, a printed form style that has evolved from the regular script over centuries, is the most widely used and is akin the serif style in Latin typography. (2) *Gothic typefaces*, printed form styles that convey a sense of modernity and are akin to sans serif styles in Latin typography. (3) *Regular* script, a printed form style that mimics handwritten forms, is primarily used for educational purposes and also serves a role similar to italics in Latin typography, although italics per se do not exist in CJK typesetting. (4) Calligraphy form, such as *semi-cursive* and *cursive* script, that are rarely used in standard typesetting but are favored for artistic expression. Regarding font width, there exists a broad spectrum from light to black (which is akin to bold in Latin typography). However, adjusting font weight in CJK characters involves more than merely altering stroke width: The exact boldness of each stroke must be carefully tuned to ensure visual consistency across characters with varying stroke counts, thus requiring more effort.

For the sake of completeness, we also want to emphasize the regional difference of CJK characters (Whistler 2023). These regional variations in typesetting are handled by both font design and encoding. Another point of distinction is the use of simplified versus traditional characters, which are represented by different code points. However, these aspects, while important, fall outside the scope of this paper and and we leave them for further study.

Computation and Creative Approach to CJK Glyphs

The advances of generative models have led to image generation models capable of producing high quality images comparable to professional photography and artwork. Noteworthy examples of such models that are public-available and state-of-the-art include Stable Diffusion XL (Podell et al. 2023) and MidJourney v6 (MidJourney Authors 2024). The former is open-source while the later is offered as a commercial product. Given their powerful capabilities, it is natural to apply these tools for producing CJK glyphs. However, generating valid CJK characters turns out to remain a challenging task even for these advanced models. As illustrated in Figure 4, these state-of-the-art image generation models fail to create legitimate CJK characters despite their ability to produce valid scenes and fine-grain details.

For text-to-image models, semi-supervised training is important for generating high-fidelity images (Zhou et al. 2023). However, we argue that the fundamental issue with the inability of state-of-the-art model to produce legitimate CJK characters lies primarily in the very distribution of characters. Specifically, the challenges are two fold, involving both the amount and complexity of CJK characters: Firstly, regarding number of characters, the latest version of Unicode, Unicode 15.1, encodes a total of **97,680** characters. Secondly, a Figure 5 show, the frequency distribution of characters, which follows power-law, suggests that only a small subset of characters appears frequently enough to enable effective learning by any model. As a result, a vast number of CJK characters remains underrepresented, introducing difficulty for any model dealing with them.

Another line of work in generating CJK characters involves transferring from a standardized glyph. Specifically, it has been proposed to generate the glyph in the desired style based on a standard glyph, usually using a reference font that covers CJK code points comprehensively. This approach becomes promising since only a small number of fonts covers a complete set of CJK code points due to

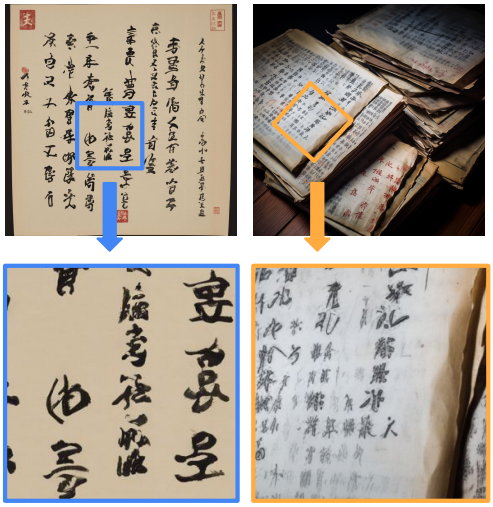

Figure 4: Example of generating CJK characters using state-of-the-art text-to-image models. From left to right: Stable Diffusion XL (Podell et al. 2023), MidJourney v6 (MidJourney Authors 2024), and zoomed-in views are on the lower row. While these models are powerful and expressive, the generated characters are illegitimate to native speakers.

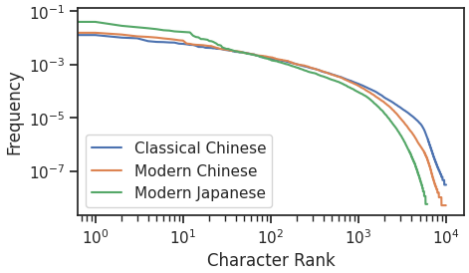

Figure 5: Distribution of CJK characters in Classical Chinese, Modern Chinese and Modern Japanese. Besides frequent characters, most of the characters are underrepresented in the text data. This plot uses data aggregated from corpus statistics (Da 2010; NINJAL 2015).

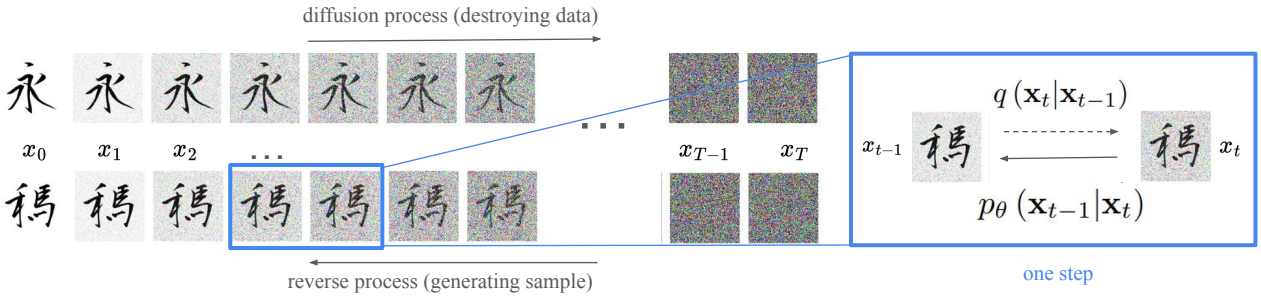

Figure 6: The diffusion model consists of two processes: (1) the diffusion process that gradually adds noise to destroy the data, and (2) the reverse process that gradually removes the noise through a Markov process (sampling from $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ at step $t-1$) for generating new samples. The estimation of the noise is parameterized by a deep learning model.

significant effort required to produce a consistent appearance for all CJK characters. Examples of such reference fonts that are free to use include Noto CJK (Noto Authors 2020) and Jigmo (Kamichi 2023). The availability of these comprehensive fonts enables the training of a conditional generative model on a limited set of paired glyphs, which can be applied to the entire set of CJK in Unicode. Many of these models are based on Generative Adversarial Networks (GAN) (Zhang, Zhang, and Cai 2018; Tian 2018; Gao et al. 2019; Yun-Chen Lo 2019; Zhu et al. 2020; Park et al. 2021; Liu and Lian 2023). Some of them leverage additional information about components and composition. While these efforts has seen promising results, they still face quality issues observable upon closer inspection, likely due to the inherent challenges of training GANs.

Perhaps the work that is mostly closely related to ours, in that it uses a Denoising Diffusion Probabilistic Model (DDPM) (Ho, Jain, and Abbeel 2020) is (Gui et al. 2023), which introduces Glyph Conditional DDPM (GC-DDPM) that is built on a UNet architecture (Ronneberger, Fischer, and Brox 2015) and generates glyphs from a reference glyph. The key distinction lies in that our method focuses on both typesetting style and brush-based calligraphy generation for font designing and artist, aiming at publisher-level quality suitable for professional font design. In contrast, GC-DDPM aims to synthesize data to enhance the performance of downstream classification on stroke-based hand-writing recognition tasks through data augmentation. Nonetheless, the design decisions made in GC-DDPM have provided valuable insights for our approach.

Diffusion Model

Diffusion models (Sohl-Dickstein et al. 2015; Ho, Jain, and Abbeel 2020; Nichol and Dhariwal 2021; Song et al. 2020) represent a broad spectrum of deep generative models that have established the state-of-the-art in a wide range of challenging tasks. A very limited set of examples from the large body of diffusion works includes computer vision (Nichol and Dhariwal 2021; Song et al. 2020), image synthesis (Ruiz et al. 2023; MidJourney Authors 2024; Podell et al. 2023), natural language processing (Lovelace et al. 2024; Wu et al. 2024), and signal processing (Engel et al. 2020; Goel et al. 2022). Given the expansive growth and wide applications, this emerging field is best navigated through comprehensive surveys (Yang et al. 2023; Croitoru et al. 2023). The specifics of how we leverage diffusion models are elaborated upon in the subsequent section.

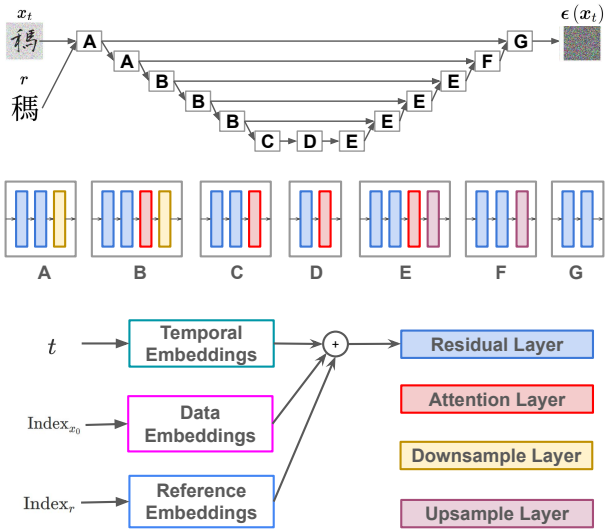

Figure 7: The UNet used in our proposed method.

Method

Our proposed method employs a diffusion model (Sohl-Dickstein et al. 2015) and we adopt the notation used DDPM (Ho, Jain, and Abbeel 2020) here. As illustrated in Figure 6, the diffusion model consists of two processes. Denoting data as $\mathbf{x}_0 \sim q(\mathbf{x_0})$, the diffusion process, also known as the forward process, gradually adds noise to destroy data $\mathbf{x}_0$, producing a sequence of progressively more noisy samples $\mathbf{x}_1, \mathbf{x}_2, \cdots$ until the entirely noisy sample $\mathbf{x}_T$. Given a schedule of variance, the forward process is a Markov process $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}\right)$, and approximates the posterior.

On the other hand, the reverse process gradually removes the noise ("denoising") through a Markov process $\mathbf{x}_{t-1} \sim$

$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_\theta(\mathbf{x}, t), \boldsymbol{\Sigma}_\theta((\mathbf{x}, t))$. As the reverse process approximates the prior of data $\mathbf{x}_0$, it can be used for generating samples. Note that if β_t is small enough, $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ would be Gaussian, so we have $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\overline{\alpha}_t}\mathbf{x}_0, (1-\overline{\alpha}_t), \mathbf{I}\right)$ where $\overline{\alpha} = \prod_{s=1}^{t} \alpha_s$ and $\alpha_t = 1 - \beta_t$. This means we can approximate q in the following way: $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0), \tilde{\beta}_t \boldsymbol{I}\right)$ where $\tilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}\boldsymbol{x}_0 + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\boldsymbol{x}_t$ and $\tilde{\beta}_t = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t$.

The training in practice has a few notable realizations as suggested in the DDPM literature (Ho, Jain, and Abbeel 2020; Nichol and Dhariwal 2021). First, the training process can sample arbitrary t instead of going from T to 1. Also, a deep neural network is tasked with predicting the noise added to $\boldsymbol{x}_0$ from partially corrupted $\boldsymbol{x}_t$, since we have $\boldsymbol{x}_0 = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}(\boldsymbol{x}_t)\right)$. We follow the suggestions (Ho, Jain, and Abbeel 2020) that the network should predict $\boldsymbol{\epsilon}(\boldsymbol{x}_t)$. Although, it is recommended by (Nichol and Dhariwal 2021) DDPM can be improved with learned variance, our empirical findings suggest that the base DDPM model's performance is sufficiently robust, and gains from such modifications could be minimal.

For the network architecture, we train a U-Net (Ronneberger, Fischer, and Brox 2015) to predict $\boldsymbol{\epsilon}(\boldsymbol{x}_t)$ as shown in Figure 7. The overall network structure is a typical UNet similar to that of GC-DDPM (Gui et al. 2023). This UNet, taking $\boldsymbol{x}_t$ as the input and predicting the noise $\boldsymbol{\epsilon}(\boldsymbol{x}_t)$, consists of several blocks that first downsample and then upsample. These blocks are connected by skip-connections. Each block is composed of residual, attention, downsample and/or upsample layers. To allow controlling of the generation, we feed the reference image r by concatenating it with $\boldsymbol{x}_t$. Also, the timestep t, index of $\boldsymbol{x}_0$, and r are injected through embeddings. After training, we can sample images by iteratively reducing the noise predicted by the U-Net.

Experiments

Experiment Setup

Dataset For training and inference, we use publicly-available fonts since they provide a large set of glyphs with a consistent look. Concretely, we use a wide range of free fonts in both printed and calligraphy forms, detailed in Table 1. During training, we construct glyph pairs by looking at the the shared glyphs between the two fonts.

Model and Training Details Our network configuration is detailed as follows: each blocks in Figure 7 has respectively 128, 128, 256, 256, 512, 512, 512, 512, 512, 256, 256, 128, 128 channels respectively in their residual layers. All embeddings utilized are uniformly set to 512 dimensions. The network accepts inputs with spatial dimensions of 128 by 128, which are halved in each downsample block, reaching a minimum dimension of 2×2 pixels at block D.

For training, Noto Serif TC is employed as the reference font, with all other fonts serving as target fonts for the model to learn from. The training process contains 3000 epochs, and spans a total duration of one week with 16 A100 GPUs.

Style	# / Form [a]	Font
Serif (Ming/Song)	43062 / P	Noto Serif TC
Gothic Typefaces	43098 / P	Noto Sans TC
Regular Script [b]	83534 / P	TW-Kai
Serif (Ming/Song) [b]	83534 / P	TW-Sung
Clerical Script	7349 / C	AoyagiReisho
Semi-cursive Script	8865 / C	KouzanMouhitu
Semi-cursive Script	7360 / C	KouzanGyousho
Cursive Script	7741 / C	KouzanSousho
Serif (Ming/Song) [c]	50217 / P	Noto Serif Tangut
Serif (Ming/Song) [d]	22741 / P	NomNaTong

[a] P for Printed From and C for Calligraphy Form.
[b] CNS 11643 Standard. [c] Only for Tangut Script.
[d] Contains CJK, including Chu Nom

Table 1: Fonts we used for training and/or inference. We focuses on two common types in CJK: typefaces for typesetting and calligraphy for artistical purpose. All fonts are open fonts that are free to use.

Overview of Generation Capability

To demonstrate the overall generation capability of our proposed method, we present a matrix of generated results in Figure 8, featuring a wide range of fonts from Table 1. Our method applies to both common and rare characters, the latter not present in the training dataset. Specifically, we using Noto Serif TC, a Serif (*Ming/Song*) font as the reference to generate characters in the desired fonts. These desired fonts include a wide range of styles such as (1) Noto Sans TC (*Gothic Typefaces*), (2) TW-Kai (*Regular Script*), (3) TW-Sung (*Serif (Ming/Song)*), along with calligraphy styles including (4) AoyagiReisho (*Clerical Script*), (5-6) KouzanMouhitu and KouzanGyousho (*Semi-cursive Script*), (7) KouzanSousho (*Cursive Script*). The selection of common characters is based on statistics (Da 2010), while less common characters are randomly sampled from Unicode block *CJK Unified Ideographs* (Unicode 2023), which contains the first batch of $20,992$ CJK characters in Unicode. It is shown that our proposed method consistently generates high-quality characters across this diverse range of styles, which are coherent and recognizable to native speakers.

Converting between Printed and Calligraphy Form

Here we explore the model's ability to generate printed form glyphs, focusing on the subtle distinctions found in fine-grain details. As shown in Figure 9. we demonstrate generating *Gothic typefaces*, *Regular script* and *Song/Ming* with different characters: first two are most common characters from statistics on Classical Chinese, middle two are less common ones from Unicode block *CJK Unified Ideographs*, and last two are extremely rare ones sampled from Unicode block *CJK Unified Ideographs Extension B* (Unicode 2023) that contains $42,720$ characters extremely rare and historical characters. Our method is capable of accurately generating

Figure 8: Our method generating CJK characters in a wide range of styles. The upper block is very-commonly-used characters, while the lower block are characters drawn randomly from a collection of less common ones. In each block, we show reference characters in gray, and generated characters in black. Each row shows different styles our method could generate: they are respectively (1) *Gothic typefaces*, (2) *Regular script*, (3) *Song/Ming* in a different standard (CNS 11643), (4) *Clerical script*, (5-6) two different *Semi-cursive scripts* and (7) *Cursive script*. See Table 1 for font data.

characters in a variety of styles, and works for a wide range of characters uniformly.

Furthermore, we examine our proposed method's capability to produce hand-written style, a significantly more challenging task due to the considerable differences from typed fonts. In Figure 9 we showcase the generation of characters in *Clerical script*, *Semi-cursive script* and *Cursive script*. It is revealed that our method can adeptly produce characters in calligraphy form, which are markedly distinct from printed form. This success underscores the model's advanced understanding and flexibility, and makes it a potent tool for generating handwritten character styles for all CJK characters, a task not done by any human calligrapher yet.

Zero-shot generation to Chinese Character-inspired Writing Systems

The cultural and historical influence of Sinosphere has lead to other Chinese Character-inspired writing systems beyond

Figure 9: Generating Printed form. The generated characters are in black, and three rows show *Gothic typefaces*, *Regular script*, *Song/Ming* in a different standard (CNS 11643) respectively. First two characters are most common, middle two less common and final two extremely rare as detailed in the main text.

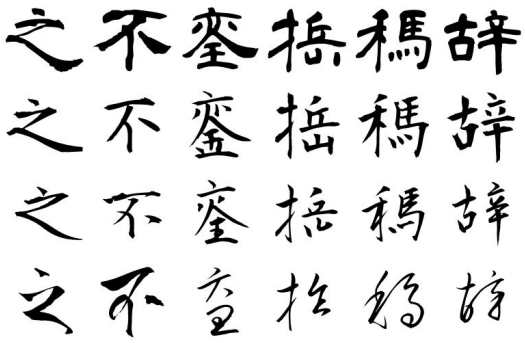

Figure 10: Generating the same set of characters in Figure 9 in (4) *Clerical script*, (5-6) two examples of *Semi-cursive script* and (7) *Cursive script*.

the core CJK (Chinese, Japanese and Korean). Notably, one such system is Chu Nom (omniglot 2023a), consisting of complex and newly created characters that adhere to Chinese Character system. Historically, Chu Nom was used along side Chu Han (Chinese Character) to write Vietnamese from the 13th to the 20th century. Thanks to significant efforts towards its revival, (HNRCV Authors), we now have a workable references for Chu Nom in the digital publishing era, including NomNaTong font used in this paper. Another example is the Tangut script (omniglot 2023b), created by modelling Chinese Characters loosely and employed to write the now-extinct Tangut language from the 10th to the 15th century. A great challenge of such scripts is the extremely limited resource due to lack of active users, as Vietnamese nowadays is written in Latin-based alphabet and Tangut language has no living speakers. Two fonts in Table 1 represent almost the entirety of free resources available for these scripts, and there is neither demand nor resources to create fonts in other styles for these writing systems.

To bridge this gap, we apply our proposed method to the setting of zero-shot generation of new styles for these scripts. Concretely, our method generate *regular script* based on *Ming/Song* that is trained only with pairs of data in CJK. Remarkably, even our proposed method has not seen these out-of-domain characters, it succeeds in producing high-quality results in a new style for both Chu Nom and Tangut scripts.

Figure 11: Zero-shot generation of new styles for Chu Nom (upper block) and Tangut scripts (lower block). Gray characters are reference ones and black characters are generated by our method in a new style.

Model Analysis

Comparison with GAN-based Model

Diffusion models have recently outperformed GAN-based counterparts in many applications, which have long been the predominant approach in generative modeling. Our work is a clear example of this paradigm shift. As shown in Figure 13, our method could generate characters with visually higher quality. Crucially, it also succeeds in capturing the more global structure of the desired style, a task for which prior models have failed.

Style Interpolation The design of the architecture in our method allows free interpolation between different styles through a weighted mixture of data embeddings. As we show in Figure 12, our method facilitates smooth transitions across different styles (printing or handwriting), and the same for font weight from *Ultra Thin* to *Bold* and the generated intermediate results are coherent and meaningful. This capability is particularly helpful for creating stylized fonts that blend various styles, offering a painlessly pipeline for all CJK characters.

Performance Study: Diffusion Steps v.s. Quality.

The diffusion model is trained and inferred with discretization of time steps from the continuous timespace $[0, 1]$ into T steps. Although we train the model with a discretization of $T = 1000$ steps, it is suggested (Lu et al. 2022; Xue et al. 2024) that effective inference can be achieved with significantly fewer steps. While we do not explicitly employ techniques to optimize for few-step sampling, exploring the minimum number of steps required for accurate generation remains valuable. As shown in Figure 14, five steps are sufficient and could be treated as feasible approximation. This efficiency allows for generating a character

Figure 12: Interpolation of the same character. **Above**: interpolating from *Gothic typefaces* to *Regular script* (upper block), then to *Semi-cursive script* (middle block), then to *Cursive script* (lower block). **Lower**: interpolating from *Ultra Thin* to *Bold*. In each block above row shows superficially mixing character at pixel level and the lower row is generated by our method.

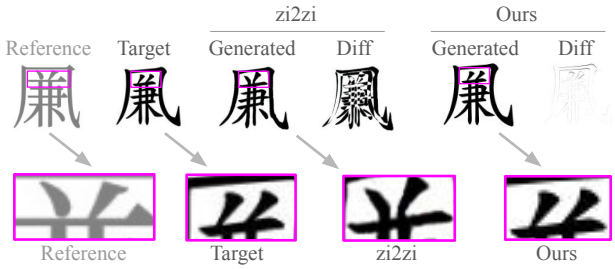

Figure 13: Comparing our method with zi2zi(Tian 2018). Ours methods generated characters that are much closer to the target (compare two "Diff") and visually more smoothing (compare two "Generated"). Furthermore, in the zoom-ins below show that our method successfully transfers into a more global structure in the target style, while zi2zi fails and instead copies the reference.

in under one second on a T4 GPU, which is considered less powerful and more suited for inference tasks.

Vectorization of Generated Characters

We finally show the vectorization of generated characters in Figure 15 using the autotrace tool (Yamato et al. 2020). This process is necessary since our method produces bitmap while a font designer would expect a vectorized glyph. The result is satisfactory. It's important to note that vectorization parameters are highly sensitive to specific fonts and must be carefully adjusted for each font.

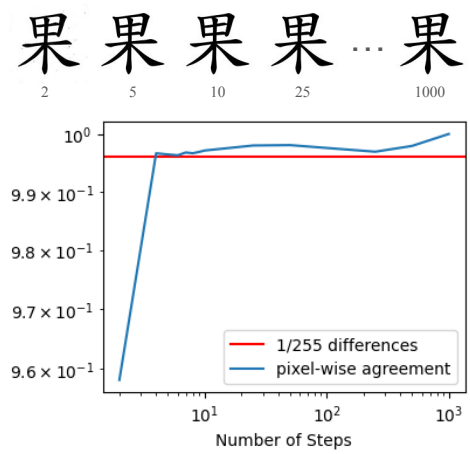

Figure 14: Inference using different number of steps for diffusion. We here show the results with different number of steps. It could be seen that only the most extreme case of two steps we may see a visible artifact. We also compare a wide range of steps by looking at the mean of pixel-wise matching between them and the result from 1000 steps.

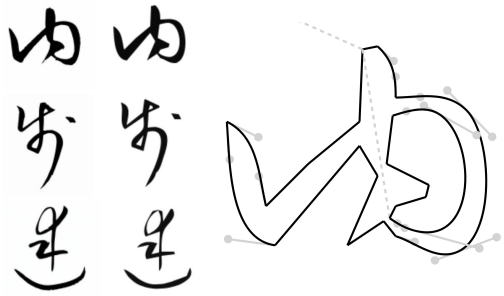

Figure 15: Example of vectorziation. On the left we show two columns: generated images and vectorized SVG files. On the right are the visualized SVG paths that shows control commands, where solid grey lines for Bezier curve and dash grey lines for moving pen. Painted using SVG Path Visualizer (Dutour 2020).

Conclusion

In this paper, we propose a diffusion based method that generates glyph in a wide range of target style from a single reference glyph. Our experiments show that our model does well for different styles, making it useful for both font design and artistical purpose.

Future work directions could include (1) deepening the integration into font design pipeline, (2) extending to multiple character typesetting and calligraphy where the interaction *between* the characters could be modeled, and (3) more ambiguously, integrating into powerful text-to-image models to enable accurate label generation.

Acknowledgement

We thank Marco Raymond Cognetta and Chris Simpkins for their valuable feedback.

References

Chiba, H., et al. 2020. Requirements for japanese text layout. W3c working group note, W3C. https://www.w3.org/TR/jlreq/.

Croitoru, F.-A.; Hondru, V.; Ionescu, R. T.; and Shah, M. 2023. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Da, J. 2010. Chinese text computing. https://lingua.mtsu.edu/chinese-computing/. Accessed: 2024-01-01.

Dutour, M. 2020. Svg path visualizer. https://svg-path-visualizer.netlify.app/. Accessed: 2024-01-01.

Engel, J.; Hantrakul, L.; Gu, C.; and Roberts, A. 2020. Ddsp: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643*.

Ethnologue Authors. 2024. Ethnologue: Languages of the world. https://www.ethnologue.com/. Accessed: 2024-02-01.

Gao, Y.; Guo, Y.; Lian, Z.; Tang, Y.; and Xiao, J. 2019. Artistic glyph image synthesis via one-stage few-shot learning. *ACM Transactions on Graphics (TOG)* 38(6):1–12.

Goel, K.; Gu, A.; Donahue, C.; and Ré, C. 2022. It's raw! audio generation with state-space models. In *International Conference on Machine Learning*, 7616–7633. PMLR.

Gui, D.; Chen, K.; Ding, H.; and Huo, Q. 2023. Zero-shot generation of training data with denoising diffusion probabilistic model for handwritten chinese character recognition. *arXiv preprint arXiv:2305.15660*.

HNRCV Authors. Han-nom Revival Committe of Vietnam. https://www.hannom-rcv.org/.

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

Huang, L. 2020. Type classification in cjk: Chinese. https://fonts.google.com/knowledge/type_in_china_japan_and_korea/type_classification_in_cjk_chinese. Accessed: 2024-01-01.

IMF. 2023. "world economic outlook database, october 2023". https://www.imf.org/en/Publications/WEO/weo-database/2023/October. Accessed: 2024-02-01.

Kamichi, K. 2023. Jigmo（字雲）font. https://kamichikoichi.github.io/jigmo/. Accessed: 2024-01-01.

Kim, M.-Y. 2020a. Type classification in cjk: Japanese. https://fonts.google.com/knowledge/type_in_china_japan_and_korea/type_classification_in_cjk_japanese. Accessed: 2024-01-01.

Kim, M.-Y. 2020b. Type classification in cjk: Korean. https://fonts.google.com/knowledge/type_in_china_japan_and_korea/type_classification_in_cjk_korean. Accessed: 2024-01-01.

Lim, S.-B., et al. 2020. Requirements for hangul text layout and typography. W3c working group note, W3C. https://www.w3.org/TR/klreq/.

Lisa Huang, M.-Y. K. 2020. Type in china, japan, and korea. https://fonts.google.com/knowledge/type_in_china_japan_and_korea. Accessed: 2024-01-01.

Liu, Y., and Lian, Z. 2023. Fonttransformer: Few-shot high-resolution chinese glyph image synthesis via stacked transformers. *Pattern Recognition* 141:109593.

Liu, Y. 2022. Deciphering the hanzisphere. https://www.sixthtone.com/news/1011502. Accessed: 2024-01-01.

Lovelace, J.; Kishore, V.; Wan, C.; Shekhtman, E.; and Weinberger, K. Q. 2024. Latent diffusion for language generation. *Advances in Neural Information Processing Systems* 36.

Lu, C.; Zhou, Y.; Bao, F.; Chen, J.; Li, C.; and Zhu, J. 2022. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems* 35:5775–5787.

Marginson, S. 2011. Higher education in east asia and singapore: Rise of the confucian model. *Higher education* 61:587–611.

MidJourney Authors. 2024. Midjourney. https://www.midjourney.com/home. Accessed: 2024-02-09.

Needham, J. 1974. *Science and civilisation in China*, volume 5. Cambridge University Press.

Nichol, A. Q., and Dhariwal, P. 2021. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, 8162–8171. PMLR.

NINJAL. 2015. 『現代日本語書き言葉均衡コーパス』語彙表 (vocabulary list for "the balanced corpus of contemporary written japanese"). https://clrd.ninjal.ac.jp/bccwj/freq-list.html. Accessed: 2024-01-01.

Noto Authors. 2020. Noto fonts. https://fonts.google.com/noto/use. Accessed: 2024-01-01.

omniglot. 2023a. Chu-non script. https://www.omniglot.com/writing/chunom.htm. Accessed: 2024-01-01.

omniglot. 2023b. Tangut. https://www.omniglot.com/writing/tangut.htm. Accessed: 2024-01-01.

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

Podell, D.; English, Z.; Lacey, K.; Blattmann, A.; Dockhorn, T.; Müller, J.; Penna, J.; and Rombach, R. 2023. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference,*

Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, 234–241. Springer.

Ruiz, N.; Li, Y.; Jampani, V.; Pritch, Y.; Rubinstein, M.; and Aberman, K. 2023. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 22500–22510.

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

Song, Y.; Sohl-Dickstein, J.; Kingma, D. P.; Kumar, A.; Ermon, S.; and Poole, B. 2020. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Stocks, E. J. 2020a. Introducing weights & styles. https://fonts.google.com/knowledge/introducing_type/introducing_weights_styles. Accessed: 2024-01-01.

Stocks, E. J. 2020b. Making sense of typographic classifications. https://fonts.google.com/knowledge/introducing_type/making_sense_of_typographic_classifications. Accessed: 2024-01-01.

Tian, Y. 2018. zi2zi: Master chinese calligraphy with conditional adversarial networks. https://github.com/kaonashi-tyc/zi2zi. Accessed: 2024-01-01.

Tung, B., et al. 2023. Requirements for chinese text layout. W3c group draft note, W3C. https://www.w3.org/TR/clreq/.

Unicode. 2023. Unicode 15.1 character code charts. https://unicode.org/charts/. Accessed: 2024-01-01.

Whistler, K. 2023. On the encoding of latin, greek, cyrillic and han. Unicode technical note, Unicode Consortium. https://www.unicode.org/notes/tn26/.

Wikipedia contributors. 2024. Wikimedia commons:ancient chinese characters project. Accessed: 2024-01-01.

Wu, T.; Fan, Z.; Liu, X.; Zheng, H.-T.; Gong, Y.; Jiao, J.; Li, J.; Guo, J.; Duan, N.; Chen, W.; et al. 2024. Ar-diffusion: Auto-regressive diffusion model for text generation. *Advances in Neural Information Processing Systems* 36.

Xue, S.; Yi, M.; Luo, W.; Zhang, S.; Sun, J.; Li, Z.; and Ma, Z.-M. 2024. Sa-solver: Stochastic adams solver for fast sampling of diffusion models. *Advances in Neural Information Processing Systems* 36.

Yamato, M.; Lemenkov, P.; Weber, M.; et al. 2020. autotrace. https://github.com/autotrace/autotrace. Accessed: 2024-01-01.

Yang, L.; Zhang, Z.; Song, Y.; Hong, S.; Xu, R.; Zhao, Y.; Zhang, W.; Cui, B.; and Yang, M.-H. 2023. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys* 56(4):1–39.

Yun-Chen Lo, Yi-Ren Chen, J. Z. J. 2019. Font2font. https://github.com/yunchenlo/Font2Font. Accessed: 2024-01-01.

Zhang, Y.; Zhang, Y.; and Cai, W. 2018. Separating style and content for generalized style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8447–8455.

Zhou, Y.; Liu, B.; Zhu, Y.; Yang, X.; Chen, C.; and Xu, J. 2023. Shifted diffusion for text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10157–10166.

Zhu, A.; Lu, X.; Bai, X.; Uchida, S.; Iwana, B. K.; and Xiong, S. 2020. Few-shot text style transfer via deep feature similarity. *IEEE Transactions on Image Processing* 29:6932–6946.