OneFont: A Unified Agent for End-to-End Font Creation

Yingxin Lai[1,3], **Yufei Liu**[1], **Guoqing Yang**[1], **Jiaxing Chai**[1], **Zhiming Luo**[1,2*], **Shaozi Li**[1,2]

[1]Department of Artificial Intelligence, Xiamen University, Xiamen, China
[2]Key Laboratory of Multimedia Trusted Perception and Efficient Computing, Ministry of Education of China, Xiamen University, Xiamen, China
[3]Graph Origin
laiyingxin2@gmail.com, zhiming.luo@xmu.edu.cn

Abstract

Despite recent advancements in font generation, practitioners still grapple with a laborious trial-and-error workflow. To streamline this, we propose OneFont, an end-to-end framework that interprets user intents via free-form dialogue, seamlessly integrating both glyph synthesis and refinement modules. We introduce the Font with Thought (FwT) paradigm, reframing font design as a reasoning task where the model plans actions and articulates design rationales. OneFont's core planner is trained via a two-stage regimen to master this paradigm. First, we instill reasoning abilities via Supervised Fine-Tuning (SFT) on a new, comprehensive benchmark of 1,500 font families we built. Second, we refine the model's policy with a novel reinforcement learning algorithm, Group Relative Policy Optimization (GRPO), guided by a hybrid reward that assesses visual fidelity, rationale coherence, and transformation correctness. Extensive experiments show OneFont significantly surpasses existing methods in design quality and stroke precision across diverse scripts, validated on our new benchmark.

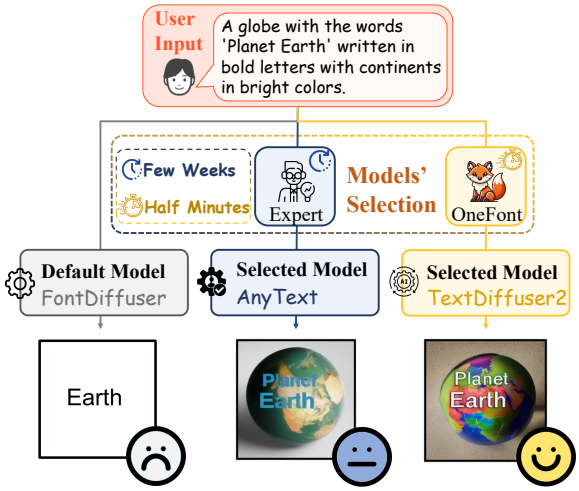

Figure 1: Illustration of the comparison between OneFont and traditional font design methods.

Introduction

Font design plays a crucial role in visual communication, with applications ranging from corporate branding to multilingual signage. Traditionally, font design is a labor-intensive process requiring meticulous stylistic and structural consistency across thousands of glyphs. This challenge is particularly pronounced in complex scripts like Chinese, where designing a complete typeface can take several months. Font generation aims to address this by synthesizing full character sets from limited reference samples. Recent advances in font processing can be categorized into three primary approaches: **(1) GAN-based methods**, which leverage adversarial training to synthesize fonts with diverse stylistic variations (Yuan et al. 2022; Hayashi, Abe, and Uchida 2019; Heusel et al. 2017); **(2) Diffusion-based methods**, which offer enhanced controllability and visual fidelity by progressively refining noise into structured glyphs (Labs 2024; Rombach et al. 2022; Dhariwal and Nichol 2021; Yang et al. 2024; He et al. 2022); and **(3) Task-specific AI-driven approaches**, which focus on localized design modifications, such as font color and texture adaptation (Liu et al. 2022; Park et al. 2021; Tuo et al. 2023) (illustrated in Fig. 1, left). Although each category exhibits distinct advantages, no single model can adequately address different font generation requirements. Achieving comprehensive language coverage, intricate stroke refinements, and style consistency often requires a strategic combination of multiple specialized models. This selection and integration process remains largely manual, requiring iterative expert intervention to achieve desired results (as depicted in Fig. 1, center). Despite recent advancements, existing font generation methods remain limited to specific tasks, such as style transfer or glyph synthesis, often requiring separate models trained for each style. This poses a critical research question: **Can users simply describe their requirements in natural language and obtain fully automated, end-to-end font processing?**

To this end, we first introduce FontBench, the benchmark specifically designed for this task. FontBench offers a substantial dataset of high-quality paired data from 2130 customized models. Each data pair comprises a user's chat input, an appropriate model answer. This comprehensive step-by-step trail enables step-wise evaluation of automatic

*Corresponding author.

font models, ensuring both quality assessment of the final font output and precise identification of potential automation bottlenecks. Furthermore, we propose a unified agent-based framework that integrates multiple font processing models within a single system. Specifically, we introduce two key innovations to improve efficiency in font design. **First, Graph-Based Planning** enables OneFont to construct a planning tree where each operation, such as font generation, refinement, or error correction is represented as a node. This allows the system to dynamically select the most appropriate model (e.g., diffusion-based, song typeface style-based) based on user input. In addition, an automatic backtracking mechanism is incorporated to prevent style drift, ensure structural consistency, and address missing radicals. **Second, Dynamic Preference Optimization:** After SFT, we apply GRPO to sharpen the model's reasoning-driven font-module selection, targeting both coherent FwT reasoning steps and high-quality font outputs. To tackle the task's complexity, we design a Dynamic Reward that jointly scores: (1) tool accuracy; (2) format alignment; (3) visual fidelity. By combining SFT-primed reasoning with this multi-objective RL, OneFont generates fonts that are precise, consistent, and visually compelling.

In summary, the contributions of this paper are as follows:

- **High-Quality Benchmark Dataset (FontBench):** We constructed a large-scale, multilingual benchmark dataset that provides a solid foundation for automated font design.
- **Unified Intelligent Framework (OneFont):** We introduce OneFont, the first framework to integrate an autonomous agent capable of planning and self-correction with a diverse suite of font tools, covering the majority of tasks from generation to editing.
- **Multi-Dimensional Quality Optimization:** We designed a novel dynamic optimization method to ensure that generated fonts achieve high standards in structure, style, and readability.

Related Work

Font Processing Methods

To achieve content fidelity and stylistic consistency, prior work has explored pixel-level translation (e.g., Pix2pix (Brooks, Holynski, and Efros 2023), CycleGAN (Zhu et al. 2017)) and encoder–decoder networks (e.g., DC-Font (Jiang et al. 2017), PEGAN (Sun, Zhang, and Yang 2018)). However, these methods often fail to handle complex glyph topologies and diverse style variations. To address this, component-based methods decompose characters into radicals or strokes to better capture fine-grained local styles. Representative systems include CalliGAN (Wu, Yang, and Hsu 2020) for radical segmentation, LFFont (Park et al. 2021) for script-specific labeling, XMP-Font (Liu et al. 2022) for image-to-stroke transformation, and FontDiffuse (Yang et al. 2024) for multiscale feature aggregation and contrastive learning. More broadly, text-driven diffusion models and LLM-based frameworks have been proposed for end-to-end font generation. For instance, multistage diffusion pipelines such as TextDiffuser (Chen et al. 2023a) and GlyphControl (Yang et al. 2023) can synthesize stylized typography from minimal input. LLM-driven frameworks such as AnyText (Tuo et al. 2023), DARLING (Zhang et al. 2024), and MetaDesigner (He et al. 2024) have further advanced content-style separation and model orchestration. Despite these advances, existing approaches still struggle to support multiple writing systems, intricate stylistic nuances, and emerging scripts.

AI Agent

Motivated by successes of LLMs (e.g., ChatGPT (OpenAI 2022), LLaMA (Touvron et al. 2023)), recent multimodal architectures such as LLaVA (Liu et al. 2023), BLIP2 (Li et al. 2023), and Flamingo (Alayrac et al. 2022) have extended LLMs to vision tasks. Agent-based frameworks have also emerged (Hong et al. 2024, 2023; Guo et al. 2025), enabling LLMs to orchestrate complex, workflows processes by autonomously selecting and invoking tools (e.g., GUI automation, code generation, APIs) according to user objectives. For instance, TooLLM (Qin et al. 2023) supports interactive segmentation, and Visual-ChatGPT (Wu et al. 2023a) integrates diverse image-processing modules, demonstrating LLMs' growing proficiency in managing complex workflows. However, typeface design imposes substantially greater requirements than code generation or standard image editing. Large-inventory scripts (e.g., Chinese) require exact stroke geometry, intact radical structures, and reliable OCR readability across thousands of glyphs. Even minor misalignments can severely degrade legibility and visual coherence.

Method

FontBench

Our goal is to relieve users from tedious steps and automatically produce the desired font images from a user's freestyle input. To achieve this objective, we first introduce FontBench, a benchmark comprising a large set of user inputs in a chatting style, built upon a foundation of over 2,000 font models with diverse functions and styles. As shown in Fig. 3, in the SFT data collection process, we first collect font images and then fine-tune different font models based on these images. During this process, we record the model functions and the multi-generation sequence of fonts. Next, we use GPT-4o (OpenAI 2023) to describe the input font models and images, and then leverage large language models to augment the image-text data. In the second stage, we document the reasoning behind model selection.

SFT Data Collection. To fine-tune our model, we collected a diverse set of real-world font resources from various global platforms (e.g., Google Fonts, Font Squirrel, DaFont). This collection includes Latin and CJK character sets, numbers, and symbols.

After constructing the dataset, to achieve real-world stylistic diversity, we gathered various advanced font processing models, primarily including: text-to-font models (e.g., WordArt Designer (He et al. 2023), TextDiffuser (Chen et al. 2023a)), few-shot font generation models

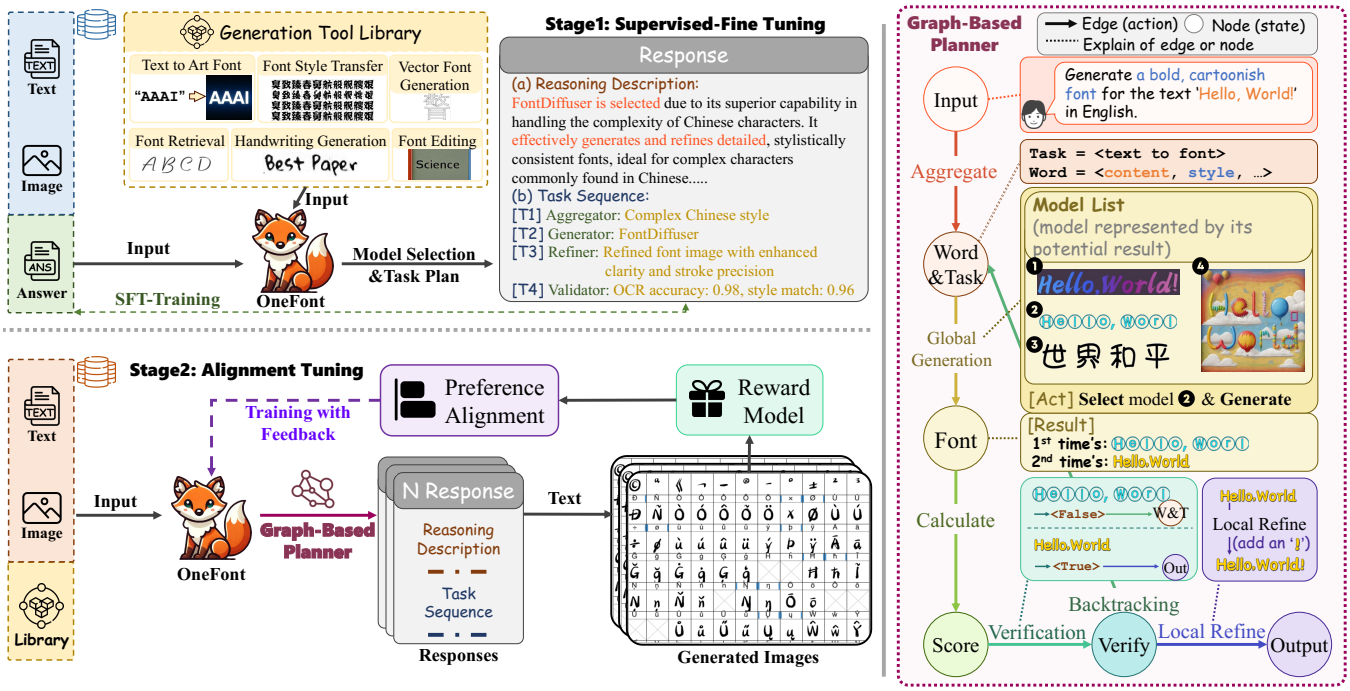

Figure 2: Overview of OneFont. An MLLM-based agent serves as the "brain" using a graph-based planner to decompose the user's requests into an intelligent workflow and invoking specialized generation or editing tools as needed. **Stage 1**: Supervised Fine-Tuning produces the initial OneFont response, leveraging the Generation Tool Library (details and an example are shown in the figure). **Stage 2**: Alignment Tuning refines OneFont's outputs, guided by the Graph-Based Planner to ensure stylistic consistency and comprehensive coverage.

Model Name	Task
AnyText	Text font generation
TextDiffuser	Text font generation
...	...
FontDiffuser	Few-shot font generation
DG-Font	Few-shot generation
Attribute2Font	Font retrieval
SVGformer	Vector font generation
DualVector	Vector font generation

Table 1: Examples of the model library used by OneFont.

(e.g., LF-Font (Park et al. 2021), FontDiffuser (Yang et al. 2024)), font retrieval models (e.g., Attribute2Font (Tuo, Geng, and Bo 2024)), and vector font generation models (e.g., SVGformer (Cao et al. 2023), DeepVecFont-v2 (Wang et al. 2023)). An example is provided in Table 1. Each node in the agent's execution process involves invoking an external tool to fulfill the font requirements. During this process, it's crucial to convey task specifics, input needs, characteristics/advantages of each model to the MLLM agent. Model captions encapsulate the model name (it serves as a unique identifier, enabling the agent to differentiate among the utilized models), required inputs, and applicable tasks/advantages (e.g., suitable for generating cute, simple fonts). For each generation process, we meticulously recorded user inputs and intermediate transformation sequences. To facilitate fine-grained glyph adjustment learning, we curated a dataset of 10k high-quality local deformation sequences. After obtaining the basic image-text pairs, we used models such as QwenVL 72B (Bai et al. 2023), GPT4o (OpenAI 2023), to rewrite and expand the model/font caption.

Preference Alignment Data. To achieve model preference alignment, we used DeepSeek R1 (Guo et al. 2025)/Gemini2 pro (Team et al. 2023) to record the model's Chain-of-Thought (CoT), i.e., detailed intermediate operations, internal logic, and decision branches during font processing and decision-making. We collected approximately 10k thought processes, formatted with clear `<think>...</think>` and `<answer>...</answer>` tags to ensure reasoning correctness. Each sample was validated across dimensions. Poorly structured or ambiguous samples were either corrected or discarded.

Data Filtering and Composition. The FontBench dataset was created by filtering 320k fonts down to 260k high-quality pairs through copyright checks, deduplication, and expert review. It features a balanced mix of languages (Chinese, English, Japanese, Korean) and is split 8:1:1 into training, validation, and test sets in JSON format.

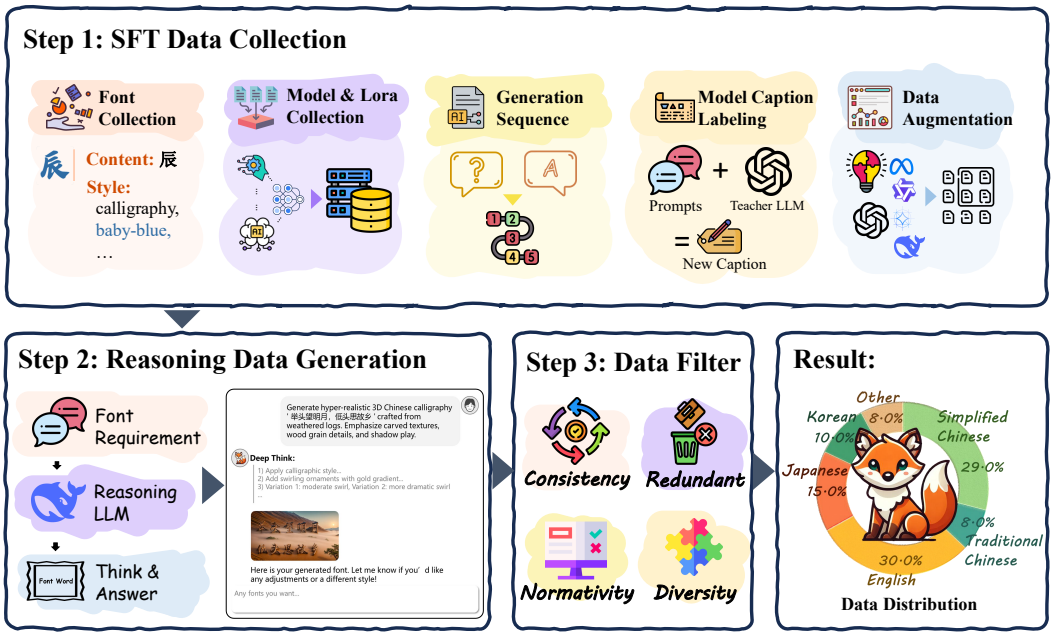

Figure 3: FontBench Dataset Construction Pipeline: The dataset is built through three main stages: (1) *SFT Data Collection* (2) *Reasoning Data Generation* (3) *Data Filtering*

OneFont: Achieving Automatic Text to Font

Our goal is to train the model capable of processing freestyle user inputs and generating the necessary components for font processing, thereby achieving automatic text to font. To achieve this, we propose OneFont, a two-stage agent-based framework, designed to automate and optimize the font design process (Fig. 2).

Overview First, given the input

$$u = \big(\{I_1, \ldots, I_k\}, t_{\text{task}}, c_{tx}\big), \tag{1}$$

where $\{I_k\}$ are reference glyphs or text snippets, t_{task} is a textual description of the desired style and c_{tx} provides optional contextual cues, our goal is to produce a set of glyphs $\mathcal{C} = \{c_1, \ldots, c_n\}$ that align with the user-specified style, legibility and completeness.

Graph-Based Planner with Verification. The first step in the process is to accurately understand the user requirements. The aggregator module Ψ_A is responsible for this initial interpretation. The $\Psi_A(u)$ interprets the input u:

$$s_u, \mathcal{C} = \Psi_A(u), \tag{2}$$

where s_u encodes the specified style and $\mathcal{C}$ denotes the set of glyphs to be generated or modified. Unlike standard image-related tasks, this aggregator step must interpret font-specific prompts (e.g., retro serif style with consistent stroke thickness) to effectively guide subsequent modules. In text-driven scenarios, such as text-to-image or text-to-font tasks, Ψ_A further parses the user's prompt to separate content from style. For instance, given the request *"Generate a bold, cartoonish font for the text "Hello, World!' in English"*, the aggregator first categorizes it as a *"text-to-font"* task, and then extracts detailed instructions: the *content* $\mathcal{C}$ as $\{\texttt{H}, \texttt{e}, \texttt{l}, \texttt{l}, \texttt{o}, \texttt{,}, \texttt{W}, \texttt{o}, \texttt{r}, \texttt{l}, \texttt{d}, \texttt{!}\}$, the *style* s_u as $\{\texttt{bold}, \texttt{cartoonish}\}$, and the target *language* as $\{\texttt{English}\}$. These parsed elements clearly direct subsequent modules regarding both the visual characteristics and the precise glyphs to process.

Global Generation Ψ_B. Given each character $c \in \mathcal{C}$ and the style specification s_u, the function shows below:

$$S^{(0)}(c) = \Psi_B\big(C,\, s_u\big), \tag{3}$$

Here, the glyph generation model Ψ_B creates glyph appearances based on the style s_u specified by the user. For example, as shown in Fig. 2 right, we select model 2 to produce the initial draft glyphs, which serve as the basis for subsequent validation steps. During the validation process, if certain glyphs fail to meet the desired criteria, our Directed Acyclic Graph (DAG) structure enables selective reinvocation of Ψ_B (e.g., by adjusting the random seed) solely for those specific glyphs, avoiding the need to regenerate the entire glyph set.

Local Refinement Ψ_C. When finer adjustments are needed, a local refiner applies transformations:

$$S^{(\text{ref})}(c) = \Psi_C\big(S^{(0)}(c),\, s_u\big), \tag{4}$$

The refiner model can collaboratively refine glyphs using multiple parallel refiners. For example(Fig. 2,right), model initially generates draft glyphs (Hello, World) based on the user-specified style. These preliminary glyphs form the basis for subsequent validation. If validation identifies specific glyphs as unsatisfactory, for example, a missing or incorrect glyph, the agent receives instructions from validation and backtracking modules to perform localized adjustments. can

then selectively refine only the problematic glyphs, such as regenerating the missing glyph by adjusting parameters or the random seed, without regenerating the entire sequence. This graph-based collaborative refinement process is significantly more efficient and flexible than the redundant regeneration commonly seen in traditional tree-structured approaches.

Verification and Backtracking. After each generation or refinement step, our quality verifier module, Ψ_V, assesses the current glyph set $\widehat{\mathcal{C}}$. This is achieved by performing three concurrent checks: content completeness (S_{cov}), text readability (S_{ocr}), and aesthetic quality (S_{style}). Specifically, S_{cov} confirms all requested characters are present. S_{ocr} uses EasyOCR to ensure perfect legibility. For aesthetic quality, S_{style} employs the HPSv2 (Wu et al. 2023b) model to provide a human-preference score. The verifier returns 'True' only if all three conditions are met, as formally defined:

$$\text{Verify}(\widehat{\mathcal{C}}) = \begin{cases} \text{True}, & S_{\text{style}}(\widehat{\mathcal{C}}) \geq \tau_{\text{style}}, \\ & S_{\text{ocr}}(\widehat{\mathcal{C}}) = \text{True}, \\ & S_{\text{cov}}(\widehat{\mathcal{C}}) = \text{True} \\ \text{False}, & \text{otherwise} \end{cases} \tag{5}$$

If the result is False, the agent receives specific feedback and initiates a backtracking mechanism. For instance, an OCR failure on a specific glyph would cause the agent to invoke the local refiner Ψ_C to fix only that problematic glyph. This targeted feedback loop ensures both high efficiency and high-quality final output.

Two-Stage Training To train the OneFont agent for complex planning and high-quality generation, we devise a two-stage training procedure. Crucially, our training objective is not to fine-tune the tool modules (Ψ_A, Ψ_B, etc.) themselves, which are pre-defined, but rather the agent's policy network π_Θ that learns to orchestrate these tools.

Stage 1: Supervised Fine-Tuning (SFT). In the first stage, we train the agent to imitate expert workflows using our FontBench SFT dataset, D_{SFT}. Each sample in this dataset consists of a user input u and the corresponding expert action sequence $\mathcal{A}^* = \{a_1^*, \ldots, a_T^*\}$, where T is the total number of actions. The policy is trained by minimizing the negative log-likelihood of the expert actions:

$$\mathcal{L}_{\text{SFT}} = -\sum_{(u,\mathcal{A}^*)\in D_{\text{SFT}}} \sum_{t=1}^{T} \log p_\Theta\big(a_t^* \mid u,\ a_{1:t-1}^*\big), \tag{6}$$

where p_Θ is the agent's policy parameterized by Θ.

Stage 2: Preference Alignment via Reinforcement Learning. While SFT provides a strong initial policy, it restricts the agent to mimicking known solutions. To enable the agent to surpass expert demonstrations and develop a nuanced understanding of quality, we introduce a second stage of preference alignment using reinforcement learning (RL). We employ Group Relative Policy Optimization (GRPO). For a given input u, the current policy π_Θ, initialized from the SFT model, generates a set of G diverse candidate solutions $\{C_g\}_{g=1}^G$. Each candidate C_g contains both the generated glyphs C_g^{output} and the reasoning trace C_g^{trace}.

The quality of each candidate is then assessed by a hybrid reward function R_{hybrid}. It is defined as a weighted sum of four key criteria, with the weights λ serving as tunable hyperparameters:

$$\begin{aligned} R_{\text{hybrid}}^{(g)}(u) = \lambda_{\text{think}}&\mathcal{R}_{\text{think}}(C_g^{\text{trace}}, u) \\ &+ \lambda_{\text{tool}}\mathcal{R}_{\text{tool}}(C_g^{\text{trace}}, C_g^{\text{output}}) \\ &+ \lambda_{\text{style}}\mathcal{R}_{\text{style}}(C_g^{\text{output}}, u) \\ &+ \lambda_{\text{visual}}\mathcal{R}_{\text{visual}}(C_g^{\text{output}}). \end{aligned} \tag{7}$$

The four reward components are defined as follows:

- $\mathcal{R}_{\text{think}}$ **(Reasoning Quality):** Evaluates the structural integrity of the reasoning trace. It rewards methodical planning by detecting the presence of predefined stage tags (e.g., `<Planning>`, `<Generate>`).
- $\mathcal{R}_{\text{tool}}$ **(Tool Utilization):** A rule-based reward assessing the compliance of tool calls, checking if the chosen tool is appropriate for the task and its parameters are valid.
- $\mathcal{R}_{\text{style}}$ **(Style & Content Alignment):** Measures alignment with user requirements by combining two metrics: (1) a CLIP score [41] for style similarity between the image and text prompt, and (2) an OCR check for content accuracy.
- $\mathcal{R}_{\text{visual}}$ **(Visual Aesthetics):** Assesses the visual appeal of the generated glyphs. We employ the HPSv2 [67] model to score the aesthetic quality of the image, using its score directly as the reward value.

The resulting rewards are normalized across the G candidates to compute the relative advantages $\hat{A}_g$:

$$\hat{A}_g = \frac{R_{\text{hybrid}}^{(g)}(u) - \text{mean}(\{R_{\text{hybrid}}^{(j)}(u)\}_{j=1}^G)}{\text{std}(\{R_{\text{hybrid}}^{(j)}(u)\}_{j=1}^G) + \delta}, \tag{8}$$

where δ is a small constant for numerical stability. Finally, the agent's policy π_Θ is updated using these advantages via the GRPO objective, which maximizes the expected advantage over the policy $\pi_{\Theta_{\text{old}}}$ that collected the data:

$$\nabla_\Theta J = \mathbb{E}_u \mathbb{E}_{C_g\sim\pi_{\Theta_{\text{old}}}}[\hat{A}_g \log \pi_\Theta(C_g|u)]. \tag{9}$$

This two-stage process, combining imitation learning with reward-driven optimization, equips the OneFont agent with both foundational skills and the ability to refine its strategies towards generating higher-quality and more creative results.

Experiment

Evaluation Protocols and Metrics.

To comprehensively evaluate *OneFont* and to fairly benchmark it against 31 related models, we designed three complementary evaluation protocols, each targeting distinct aspects of font processing performance. *To ensure fairness, all open-source models were retrained on the FontBench training set and tested on the FontBench test set.*

Method	Detector	Backbone	Model ACC (%) ↑	Config ACC (%) ↑	FID ↓	CLIP Score ↑	Font ACC (%) ↑
Random	–	–	10.3	–	85.4	22.1	10.2
Close Source	GPT-4o (OpenAI 2023)	–	55.2	–	41.2	68.5	45.3
Close Source	Gemini2 (Team et al. 2023)	–	58.7	–	38.4	69.6	46.0
Close Source	DeepSeek R1 (Guo et al. 2025)	–	60.1	–	37.5	70.2	46.9
LLM	mPLUG-Owl2 (Ye et al. 2023)	LLaMA-2 7B	67.3	65.5	29.4	72.7	51.5
LLM	BLIP2 (Li et al. 2023)	OPT 7B	69.5	66.2	28.9	73.3	52.1
LLM	InternVL2 (Chen et al. 2024)	LLaMA-2 7B	70.1	67.1	27.5	73.9	53.0
LLM	LLaVA-v1.6 (Liu et al. 2023)	Vicuna 7B	72.8	69.3	26.3	74.5	54.2
LLM	Phi3.5 (Abdin et al. 2024)	Phi-3.8B	<u>73.0</u>	<u>69.8</u>	<u>25.9</u>	74.6	<u>54.7</u>
LLM	Qwen-VL (Bai et al. 2023)	Qwen-VL 7B	73.2	69.1	25.7	<u>74.7</u>	54.4
Agent	ChatGen (Jia et al. 2024)	–	42.0	38.9	48.3	57.9	37.9
Agent	MODEL SPIDER (Zhang et al. 2023)	–	53.3	40.5	47.8	58.4	34.3
Agent	**Ours**	–	**81.7**	**76.4**	**23.2**	**76.2**	**59.4**

Table 2: Comprehensive benchmark comparison. **Bold** indicates best results, <u>underline</u> denotes second-best. All LLM based are trained on FontBench, and is evaluated on FontBench test. All open-source competing methods are evaluated using their officially released models.

GT	Metric	Text	Method									
			SDXL	AGIS-Net	DS-Fusion	TextDiff	TextDiff2	AnyText	Flux.1	MetaD.	Ours*	Ours
P	SSIM ↑	Eng	0.158	0.151	0.166	0.231	0.249	0.268	0.261	0.286	<u>0.295</u>	**0.331**
		CJK	0.181	0.174	0.187	0.233	0.250	0.261	0.257	0.279	<u>0.242</u>	**0.302**
		All	0.170	0.164	0.181	0.237	0.248	0.265	0.260	0.282	<u>0.289</u>	**0.304**
	LPIPS ↓	Eng	0.761	0.778	0.769	0.736	0.714	0.706	0.695	0.687	<u>0.683</u>	**0.641**
		CJK	0.749	0.755	0.747	0.719	0.701	0.693	0.684	0.681	<u>0.673</u>	**0.632**
		All	0.755	0.765	0.758	0.728	0.709	0.702	0.689	0.684	<u>0.678</u>	**0.645**
D	SSIM ↑	Eng	0.203	0.197	0.205	0.262	0.276	0.289	0.287	0.297	<u>0.305</u>	**0.319**
		CJK	0.192	0.185	0.191	0.254	0.263	0.272	0.274	0.281	<u>0.286</u>	**0.299**
		All	0.198	0.191	0.200	0.258	0.269	0.280	0.279	0.289	<u>0.294</u>	**0.326**
	LPIPS ↓	Eng	0.792	0.811	0.804	0.756	0.735	0.724	0.716	0.709	<u>0.698</u>	**0.672**
		CJK	0.805	0.812	0.818	0.749	0.733	0.728	0.722	0.715	<u>0.709</u>	**0.653**
		All	0.799	0.806	0.810	0.753	0.739	0.732	0.719	0.713	<u>0.704</u>	**0.659**

Table 3: Quantitative comparison of font generation methods using SSIM (↑) and LPIPS (↓). The best results are highlighted in **bold**, and the second best are <u>underlined</u>. All methods were evaluated using their officially released models, following the experimental setup of MetaDesigner (He et al. 2024). "Ours*" denotes the version without preference alignment.

Protocol-1 (Model Selection and Visual Fidelity) This protocol compares *OneFont* against sota MLLMs (e.g., Qwen-VL (Bai et al. 2023), LLaVA-v1.6 (Liu et al. 2023)) and agent-based methods (e.g., ChatGen (Jia et al. 2024)). Following fine-tuning on the FontBench training set, all methods are presented with identical user input textual prompts (e.g., "bold, cartoonish font") and style references. We assess their autonomous capability to select an appropriate model pipeline and configuration for glyph generation. The evaluation is quantified by key metrics such as Model Accuracy (percentage of correct pipeline selections), Config Accuracy (accuracy of applied style), FID, CLIP Score, and Font Accuracy (glyph recognition accuracy using EasyOCR).

Protocol-2 (User Study) To evaluate real-world usability, a subjective assessment was conducted with 70 participants from diverse backgrounds. Participants were instructed to rate the generated fonts on a 0-100 scale according to three key dimensions: Accuracy (measuring text legibility and correctness), Aesthetics (encompassing visual appeal and style coherence), and Creativity (reflecting novelty and design innovation).

Training details. We built upon Qwen-VL 2.5 as the base backbone, performing full fine-tuning through a two-stage process: initial SFT and GRPO training. The AdamW optimizer was used consistently across both stages with a learning rate of 4×10^{-5} and a weight decay of 1.0, with each stage lasting 5 epochs. Experiments were conducted on 8 NVIDIA H100 GPUs.

Method	Venue	Accuracy	Aesthetics	Creativity
SDXL	Arxiv'23	3.1	0.2	0.7
FLUX.1	-	8.3	6.7	1.4
TextDiff	NIPS'24	66.1	1.3	0.8
TextDiff-2	ECCV'24	69.2	3.9	1.7
AnyText	ICLR'24	73.9	1.1	0.9
MetaDesigner	ICLR'25	<u>69.8</u>	<u>61.5</u>	<u>63.4</u>
Ours	-	**81.4**	**73.1**	**72.9**

Table 4: User Study Results on Multiple Datasets. We evaluate Text Accuracy, Aesthetics, and Creativity. Scores range from 0 to 100, higher is better. **Bold** indicates best; <u>underlined</u> indicates second-best.

Quantitative Results

Protocol 1: Model Selection and Visual Fidelity Protocol 1 highlights *OneFont*'s superior performance sota MLLM in model selection and visual fidelity, outperforming baselines trained on FontBench (Table 2). *OneFont* achieves key metric gains, reaching 81.7% Model ACC and 23.2 FID, significantly surpassing Phi3.5 (Abdin et al. 2024) (73.0%

Figure 4: Qualitative comparison with existing state-of-the-art methods for font generation tasks.

Components			FID ↓	CLIP ↑	Font ACC(%) ↑
GBP	SA	PA			
✗	✗	✗	29.4	72.1	51.5
✓	✗	✗	27.1	73.3	53.8
✓	✓	✗	26.3	74.4	55.7
✓	✗	✓	25.9	74.6	56.4
✓	✓	✓	**23.2**	**76.2**	**59.4**

Table 5: Ablation study regarding the effectiveness of each proposed module via FontBench test evaluations. The results show an incremental benefit in each module.

ACC, 25.9 FID). Unlike general models like GPT-4o (OpenAI 2023), which lack the specialized reasoning for font design challenges (glyph completeness, style consistency), *OneFont*'s Graph-Based Planner (GBP) ensures end-to-end control. The GBP systematically optimizes pipeline selection, driving higher visual fidelity and validating our approach as confirmed by Table 3 for structural integrity and perceptual quality.

Protocol 2: User Study The user study (Table 4) confirms *OneFont*'s real-world practicality, showcasing its superiority in user perception. With 70 participants, *OneFont* leads across all key dimensions: Accuracy (81.4%), Aesthetics (73.1%), and Creativity (72.9%). Critically, it widens the gap by 10-12 percentage points over MetaDesigner (He et al. 2024) in Aesthetics (61.5%) and Creativity (63.4%), suggesting a deeper user appreciation for its design. While traditional methods (SDXL (Podell et al. 2023), FLUX.1 (Labs 2024)) and even diffusion models like TextDiff (Chen et al. 2023a) struggle to balance accuracy with aesthetic appeal and creative novelty, *OneFont*'s Preference Alignment (PA) training proves instrumental. PA directly enhances user expectation alignment, yielding fonts that are not only legible but also visually compelling and innovative.

Ablation Study

This section analyzes the impact of our core components. **Effectiveness of GBP, SA, and PA.** First we investigate how each component (GBP, SA, PA) affects performance by progressively enabling them, as shown in Table 5. Removing all three modules (first row) leads to an FID of 29.4%, a CLIP score of 72.1%, and a Font ACC of 51.5%. Introducing the Graph-based Planner immediately improves FID, CLIP, and Font ACC, confirming that a globally structured planning mechanism helps mitigate coarse layout errors. Subsequently adding the StrokeMicro Adjuster refines local stroke generation, further boosting Font ACC from 53.8% to 55.7%. Substituting SA with Preference Alignment increases CLIP and Font ACC, highlighting that aligning the generation process with user-driven cues can enhance semantic coherence. Finally, enabling all three achieve the best performance across all metrics (FID 23.2%, CLIP 76.2%, Font ACC 59.4%), indicating that global planning, local refinement, and preference alignment jointly produce visually faithful and user-aligned font outputs.

Visualization and Analysis

In Fig. 4, OneFont proves to be highly effective in font image generation, outperforming Stable Diffusion2 (Rombach et al. 2022), PixArt-α (Chen et al. 2023b), AnyText (Tuo et al. 2023), TextDiffuser2 (Chen et al. 2023a), and MetaDesign (He et al. 2024). Its superior attribute binding ensures that the "cute and clear" and "solid metal texture" requirements are met with precision. The method's position accuracy and ability to handle complex and diverse requirements result in visually appealing, prompt-faithful outputs, such as cheerful "Hello, World!" and Chinese metallic. By integrating tools and continuously verifying results, Ours extracts both style and content to generate the most compatible font images, making it the most robust solution for these tasks.

Conclusion

In this paper, we introduce OneFont, an agent that unifies multiple, disparate font generation tools using a smart planner. Its core components, featuring intelligent planning and self-correction mechanisms, actively fix common issues like style inconsistencies and structural errors, ensuring high-quality. Comprehensive experiments show OneFont significantly outperforms single-model methods, not only in quantitative benchmarks but also in qualitative user studies. As future work, we plan to expand this unified framework to support more languages and complex creative tasks.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant Nos. 62276221 and 62376232.

References

Abdin, M.; Aneja, J.; Awadalla, H.; Awadallah, A.; Awan, A. A.; Bach, N.; Bahree, A.; Bakhtiari, A.; Bao, J.; Behl, H.; et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone, 2024. *URL https://arxiv.org/abs/2404.14219*.

Alayrac, J.-B.; Donahue, J.; Luc, P.; Miech, A.; Barr, I.; Hasson, Y.; Lenc, K.; Mensch, A.; Millican, K.; Reynolds, M.; et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35: 23716–23736.

Bai, J.; Bai, S.; Yang, S.; Wang, S.; Tan, S.; Wang, P.; Lin, J.; Zhou, C.; and Zhou, J. 2023. Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond. *arXiv preprint arXiv:2308.12966*.

Brooks, T.; Holynski, A.; and Efros, A. A. 2023. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18392–18402.

Cao, D.; Wang, Z.; Echevarria, J.; and Liu, Y. 2023. Svgformer: Representation learning for continuous vector graphics using transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10093–10102.

Chen, J.; Huang, Y.; Lv, T.; Cui, L.; Chen, Q.; and Wei, F. 2023a. Textdiffuser: Diffusion models as text painters. *Advances in Neural Information Processing Systems*, 36: 9353–9387.

Chen, J.; Yu, J.; Ge, C.; Yao, L.; Xie, E.; Wu, Y.; Wang, Z.; Kwok, J.; Luo, P.; Lu, H.; et al. 2023b. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*.

Chen, Z.; Wu, J.; Wang, W.; Su, W.; Chen, G.; Xing, S.; Zhong, M.; Zhang, Q.; Zhu, X.; Lu, L.; et al. 2024. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 24185–24198.

Dhariwal, P.; and Nichol, A. 2021. Diffusion models beat gans on image synthesis. *Proc. NeurIPS*, 34: 8780–8794.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Hayashi, H.; Abe, K.; and Uchida, S. 2019. GlyphGAN: Style-Consistent Font Generation Based on Generative Adversarial Networks. *Knowledge-Based Systems*, 186: 104927.

He, H.; Chen, X.; Wang, C.; Liu, J.; Du, B.; Tao, D.; and Qiao, Y. 2022. Diff-Font: Diffusion Model for Robust One-Shot Font Generation. *arXiv preprint arXiv:2212.05895*.

He, J.-Y.; Cheng, Z.-Q.; Li, C.; Sun, J.; He, Q.; Xiang, W.; Chen, H.; Lan, J.-P.; Lin, X.; Zhu, K.; et al. 2024. MetaDesigner: Advancing Artistic Typography through AI-Driven, User-Centric, and Multilingual WordArt Synthesis. *arXiv preprint arXiv:2406.19859*.

He, J.-Y.; Cheng, Z.-Q.; Li, C.; Sun, J.; Xiang, W.; Lin, X.; Kang, X.; Jin, Z.; Hu, Y.; Luo, B.; et al. 2023. WordArt designer: user-driven artistic typography synthesis using large language models. *arXiv preprint arXiv:2310.18332*.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. GANs trained by a two time-scale update rule converge to a local nash equilibrium. *Proc. NeurIPS*, 30.

Hong, S.; Zheng, X.; Chen, J.; Cheng, Y.; Wang, J.; Zhang, C.; Wang, Z.; Yau, S. K. S.; Lin, Z.; Zhou, L.; et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 3(4): 6.

Hong, W.; Wang, W.; Lv, Q.; Xu, J.; Yu, W.; Ji, J.; Wang, Y.; Wang, Z.; Dong, Y.; Ding, M.; et al. 2024. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14281–14290.

Jia, C.; Xia, C.; Dang, Z.; Wu, W.; Qian, H.; and Luo, M. 2024. ChatGen: Automatic Text-to-Image Generation From FreeStyle Chatting. *arXiv preprint arXiv:2411.17176*.

Jiang, Y.; Lian, Z.; Tang, Y.; and Xiao, J. 2017. DCFont: an end-to-end deep Chinese font generation system. In *SIGGRAPH Asia Technical Briefs*, 1–4.

Labs, B. F. 2024. FLUX. https://github.com/black-forest-labs/flux.

Li, J.; Li, D.; Savarese, S.; and Hoi, S. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In *ICML*.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*.

Liu, W.; Liu, F.; Ding, F.; He, Q.; and Yi, Z. 2022. XMP-Font: self-supervised cross-modality pre-training for few-shot font generation. In *Proc. CVPR*, 7905–7914.

OpenAI. 2022. ChatGPT: A Large-Scale Transformer-Based Language Model. https://www.openai.com/chatgpt.

OpenAI. 2023. GPT-4: A Large-Scale Transformer-Based Language Model. https://www.openai.com/gpt-4.

Park, S.; Chun, S.; Cha, J.; Lee, B.; and Shim, H. 2021. Few-shot font generation with localized style representations and factorization. In *Proc. AAAI*, 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*.

Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; Zhao, S.; Tian, R.; Xie, R.; Zhou, J.; Gerstein, M.; Li, D.; Liu, Z.; and Sun, M. 2023. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. arXiv:2307.16789.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.

Sun, D.; Zhang, Q.; and Yang, J. 2018. Pyramid embedded generative adversarial network for automated font generation. In *Proc. ICPR*, 976–981. IEEE.

Team, G.; Anil, R.; Borgeaud, S.; Alayrac, J.-B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A. M.; Hauth, A.; Millican, K.; et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Tuo, Y.; Geng, Y.; and Bo, L. 2024. AnyText2: Visual Text Generation and Editing With Customizable Attributes. *arXiv preprint arXiv:2411.15245*.

Tuo, Y.; Xiang, W.; He, J.-Y.; Geng, Y.; and Xie, X. 2023. Anytext: Multilingual visual text generation and editing. *arXiv preprint arXiv:2311.03054*.

Wang, Y.; Wang, Y.; Yu, L.; Zhu, Y.; and Lian, Z. 2023. Deepvecfont-v2: Exploiting transformers to synthesize vector fonts with higher quality. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 18320–18328.

Wu, C.; Yin, S.; Qi, W.; Wang, X.; Tang, Z.; and Duan, N. 2023a. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*.

Wu, S.-J.; Yang, C.-Y.; and Hsu, J. Y.-j. 2020. CalliGAN: Style and structure-aware Chinese calligraphy character generator. *arXiv preprint arXiv:2005.12500*.

Wu, X.; Hao, Y.; Sun, K.; Chen, Y.; Zhu, F.; Zhao, R.; and Li, H. 2023b. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*.

Yang, Y.; Gui, D.; Yuan, Y.; Liang, W.; Ding, H.; Hu, H.; and Chen, K. 2023. Glyphcontrol: glyph conditional control for visual text generation. *Advances in Neural Information Processing Systems*, 36: 44050–44066.

Yang, Z.; Peng, D.; Kong, Y.; Zhang, Y.; Yao, C.; and Jin, L. 2024. FontDiffuser: One-Shot Font Generation via Denoising Diffusion with Multi-Scale Content Aggregation and Style Contrastive Learning. In *Proceedings of the AAAI conference on artificial intelligence*.

Ye, Q.; Xu, H.; Xu, G.; Ye, J.; Yan, M.; Zhou, Y.; Wang, J.; Hu, A.; Shi, P.; Shi, Y.; et al. 2023. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*.

Yuan, S.; Liu, R.; Chen, M.; Chen, B.; Qiu, Z.; and He, X. 2022. SE-GAN: Skeleton Enhanced GAN-Based Model for Brush Handwriting Font Generation. In *IEEE International Conference on Multimedia and Expo (ICME)*.

Zhang, B.; Xie, H.; Gao, Z.; and Wang, Y. 2024. Choose what you need: Disentangled representation learning for scene text recognition removal and editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 28358–28368.

Zhang, Y.-K.; Huang, T.-J.; Ding, Y.-X.; Zhan, D.-C.; and Ye, H.-J. 2023. Model spider: Learning to rank pre-trained models efficiently. *Advances in Neural Information Processing Systems*, 36: 13692–13719.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. ICCV*, 2223–2232.