复现 FontDiffusion：通过多尺度内容聚合和风格对比学习的去噪扩散进行一次性字体生成 [1]

摘要

本文介绍了基于扩散模型的字体生成方法 FontDiffuser。该方法旨在解决单样本字体生成中的难题。FontDiffuser 包含多个重要模块。多尺度内容聚合（MCA）模块可捕捉和保留复杂字符细节，通过将源图像多尺度内容特征注入模型，保障生成字体的精细笔画和布局信息。风格对比细化（SCR）模块利用 VGG 网络提取参考图像风格，通过风格投影和对比学习，让模型能准确模仿目标风格，有效处理大风格变化。参考结构交互（RSI）模块借助可变形卷积网络和交叉注意力机制，依据参考图像结构特征进行结构变形，解决源图像与目标图像结构差异。与现有基于 GAN 等框架的字体生成方法对比，FontDiffuser 在生成复杂字符和处理大风格变化方面表现优异，能生成高质量且风格准确的字体。不过，该方法存在生成速度较慢的局限。未来可优化采样算法提升速度，并拓展其应用领域，进一步优化模型结构。

关键词：扩散模型；字体生成；多尺度聚合；风格提取

1 引言

在现代社会，字体设计在各个领域都有至关重要的应用。例如，在平面设计中，不同风格的字体能够营造出各异的视觉氛围，帮助精准传达设计意图，吸引受众目光，适用于海报、宣传册、书籍封面等设计；在数字媒体领域，网页设计、移动应用界面设计都需要多样化的字体来提升用户体验，合适的字体能提升用户界面的可读性和美观度，增强用户体验，使用户更愿意停留和操作；在品牌建设方面，独特的字体能够塑造品牌形象，增强品牌辨识度。随着这些领域的不断发展，对新颖、高质量字体的需求日益增长。Canva 平台每月有超过 1500 万活跃用户，为了满足这些用户对新鲜字体的需求，平台利用自动字体生成技术每月可新增数十种字体风格。这些新字体在上线后会有一定比例的用户选择使用，例如约 20% 的专业设计师用户会尝试使用新生成的字体进行设计项目，提高了设计作品的独特性。

此外，在文化遗产保护和数字化领域，对于古代书法字体的数字化重现和衍生设计也有很大的需求。在对《兰亭集序》等经典书法作品进行数字化字体生成时，研究人员需要处理大量的书法笔画数据。自动字体生成技术可以分析书法作品中的笔画粗细、连笔方式、字间距等特征，从一个包含几百字的书法作品中，可以生成数千个风格一致的数字化字体字符，大大丰富了文化遗产数字化的成果。

但是自动字体生成在生成质量和模型训练生成等方面仍存在许多问题。在生成质量方面，会存在复杂字符生成困难和风格一致性等问题。许多字体包含复杂的字符结构，如中文字体

中的繁体字和生僻字，现有的自动字体生成方法在处理这些复杂字符时，往往会出现笔画缺失、模糊、伪影等问题 [2]。这是因为模型难以精确地捕捉和再现复杂字符的精细笔画和内部结构。例如，对于一些笔画较多且相互交错的汉字，生成的字体可能会出现部分笔画粘连或断裂的情况，导致字体的可读性和美观性受到影响。在字体生成中，风格的一致性是一个关键挑战。当目标风格与源风格存在较大差异时，现有的方法很难保证生成字体在风格上的连贯性和准确性 [3]。这包括字体的粗细、笔画的曲率、字间距等风格特征。比如，在模仿一种具有独特装饰性笔画的字体风格时，生成的字体可能会在某些字符上丢失这种装饰性特征，或者在不同字符之间出现风格不一致的现象。

在模型训练生成方面，会存在数据标注成本高、模型训练不稳定、生成速度较慢、资源消耗大等问题。一些字体生成方法依赖于有标注的数据，例如需要对字体的笔画、组件等进行标注。对于复杂的字体，尤其是包含大量字符的字体集，这种数据标注工作需要耗费大量的人力和时间成本。例如，在对中文字体进行笔画标注时，由于汉字数量庞大且结构复杂，标注工作的工作量极大，限制了基于标注数据的字体生成方法的应用和发展。基于深度学习的字体生成模型，如基于 GAN 的模型 [4]，在训练过程中往往存在不稳定的问题。GAN 模型的训练需要平衡生成器和判别器之间的博弈，稍有不慎就会导致训练失败，出现模式崩溃等问题。例如，在训练过程中，生成器可能会生成非常相似甚至相同的字体样本，失去了生成多样化字体的能力，这是由于判别器对生成器的反馈出现偏差，导致生成器无法正确学习数据的分布。一些先进的字体生成方法，虽然在生成质量上有一定优势，但往往需要较长的时间来生成字体。这对于需要实时生成字体的应用场景，如在线字体设计工具、实时文字处理软件等，是一个很大的限制。例如，基于扩散模型的字体生成方法 [5]，由于其生成过程通常需要多步迭代，导致生成速度相对较慢，在实际应用中可能无法满足用户对快速生成字体的需求。复杂的字体生成模型在训练和生成过程中需要消耗大量的计算资源，包括 CPU、GPU 的运算资源和内存资源等。这不仅增加了研究和应用的成本，也限制了这些方法在资源受限环境下的应用。例如，训练一个大规模的字体生成模型一般需要多块高端 GPU 并行计算，并且在生成过程中也需要较高的硬件配置来保证运行效率，这对于一些小型研究团队或低成本应用场景来说是难以承受的。

2 相关工作

2.1 早期方法

早期的字体生成尝试多基于规则和统计手段，例如通过分析已有字体的笔画结构、字符间距等规则，结合概率统计模型来生成新字体。然而，这些方法往往过于依赖人工设定的规则和简单的统计规律，缺乏对字体风格的深度理解和灵活变化能力，生成的字体样式较为局限，很难产生富有创意和独特风格的字体，并且在处理复杂字符结构时效果不佳。

2.2 深度学习方法

随着深度学习的发展，GAN 在字体生成领域得到了广泛应用。GAN 由生成器和判别器组成 [6]，生成器负责生成字体图像，判别器则判断生成的图像是否与真实字体图像相似，通过二者的对抗训练不断优化生成器的生成能力。不过，GAN 框架的字体生成方法存在诸多问

题。一方面，训练过程不稳定，由于生成器和判别器之间的博弈关系微妙，很容易出现训练难以收敛甚至模式崩溃的情况，导致生成器最终只能生成有限几种相似的字体样式，失去了多样性。另一方面，这类方法大多通过单尺度高级特征感知内容信息，对于复杂字符（如中文里笔画繁多、结构复杂的汉字）来说，难以有效保留其源内容至关重要的细粒度细节，常出现笔画缺失、伪影、模糊、布局错误等问题，严重影响字体质量。

除了 GAN，还有一些利用其他深度学习架构的字体生成方法，比如利用变分自编码器（VAE）[7] 等。但这些方法同样面临挑战，例如在处理大风格变化时表现欠佳，难以准确地将一种风格鲜明的字体特征迁移到另一种截然不同的风格上，导致生成的字体风格不一致，不能很好地满足实际应用中对多样化、高质量字体风格转换的需求。另外，部分方法若想取得较好效果需要大量的标注数据作为先验知识，像标注字体的笔画、组件等信息，但对于复杂字体集而言，标注成本高昂，极大地限制了这些方法的应用范围和实用性。

2.3 最新成果

近几年，国内外许多学者企业都取得了重要的结果。在国外，筑波大学 AideaLab 于 2022 年制作出字体生成系统，利用生成对抗网络（GAN），从几个人工设计的字体中提取，依据粗细、线条等生成 14,000 个字以上的字体，可通过调整参数值设计新字体，输出字体能存储为矢量文件；皮卡智能在基于生成对抗网络和风格迁移的方法中，于 GAN 模型新增字体风格特征提取器，能让生成器生成任意风格的字体，如精确还原“王羲之”的书法风格特征形成“李白实验室出品-兰亭体”。在国内，北京大学王选计算机研究所团队成果丰硕：用偏传统图形学的方法首次解决大规模手写体中文字库的自动生成问题；提出基于 Image2image 框架的中文字库生成模型；研发的 AGIS-Net 模型能依据极少量输入汉字字形样本，建模字体风格并迁移其纹理特征，解决小样本学习的特效字库自动生成问题；Attribute2Font 模型可给定任意字体属性值自动生成个性化订制风格的完整中英文字库；EasyFont 模型首次实现基于风格学习的大规模手写体中文字库自动生成系统。

3 本文方法

3.1 本文方法概述

扩散模型是一种用于生成任务的概率模型。其基本思想是通过设计一个正向过程，逐步向目标分布（如图像数据分布）中添加噪声，将原始数据逐渐变换为噪声，这个过程可以用马尔可夫链来描述。然后，通过学习反向映射，即从噪声到原始数据分布的过程，来实现生成任务。在反向过程中，模型需要预测每一步添加的噪声，通过不断迭代去噪，最终生成目标样本。

本文介绍的模型 FontDiffuser 是由一个字体生成扩散模型和一个风格对比细化模块组成，如图 1所示：

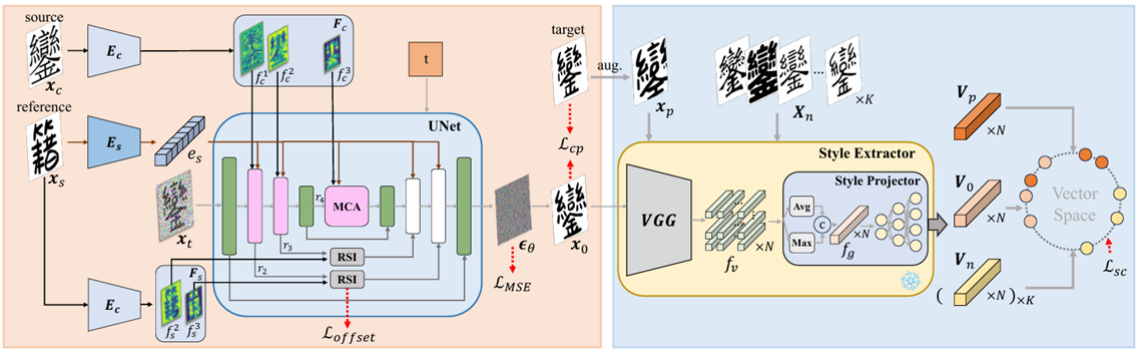

图 1. 模型 FontDiffuser 总体框架

在该字体生成扩散模型中，给定一个源图像 $\boldsymbol{x}_c$ 和一个参考图像 $\boldsymbol{x}_s$，最终输出图像不仅应该具有与 $\boldsymbol{x}_c$ 中相同的内容，而且应该与参考风格一致。风格对比细化模块的作用是从一组图像中梳理出不同的风格，并通过风格对比损失对扩散模型进行指导。

3.2 字体生成扩散模型

首先，图 1左边的字体生成扩散模型主要是基于 DDPM [8]，其正向过程是一个马尔可夫链，目标图像按照公式（1）逐渐变为噪声，其中，t 为扩散步数，ϵ 为添加的高斯噪声，$\bar{\alpha}_t = \prod_{i=0}^{t}(1-\beta_i)$，$\beta_i$ 为固定方差的超参数。反向过程则通过模型预测噪声 $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t, \boldsymbol{x}_c, \boldsymbol{x}_s)$，并按照公式（2）逐步还原为目标图像，其中，$\sigma_t$ 为超参数，z 为噪声。

$$\boldsymbol{x} = \sqrt{\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \tag{1}$$

$$\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t, \boldsymbol{x}_c, \boldsymbol{x}_s)\right) + \sigma_t \boldsymbol{z} \tag{2}$$

此外，为了增强复杂字符的保存性，模型采用了 MCA 模块将全局和局部内容线索注入到模型的 UNet 中。同时，采用 RSI 模块来促进参考特征的结构变形。

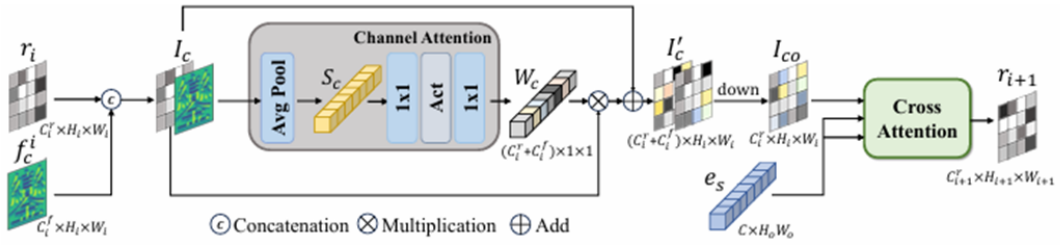

图 2. MCA 模块

MCA 模块结构如图 2所示。源图像 x_c 首先被内容编码器 E_c 编码提取特征，得到不同层的多尺度内容特征 $F_c = \{f_c^1, f_c^2, f_c^3\}$。每个内容特征 f_c^i 分别与风格编码器 E_s 编码提取的风格特征 e_s 一起，通过三个 MCA 模块注入 UNet。内容特征 f_c^i 与之前的 UNet 块特征 r_i 相连接，从而产生通道信息特征 I_c。为了增强自适应选择通道融合的能力，模型在 I_c 上应用了通道注意力机制 [9]，其中包含了平均池化，两个 1×1 卷积和一个激活函数。这种注意力机制会

产生一个全局通道感知向量 W_c，该向量通过通道来对通道信息特征 I_c 进行加权。然后，在残差连接后，使用了 1×1 卷积来减少 I_c' 的通道数，得到输出 I_{co}。最后，使用一个交叉注意力模块来插入风格特征 e_s，其中 e_s 作为键和值，I_{co} 作为查询。

为了解决源图像和目标图像之间的结构差异，论文提出了 RSI 模块。该模块采用可变形卷积网络（DCN）[10]，在 UNet 的跳跃连接上进行结构变形。具体来说，参考图像 x_s 首先通过内容编码器 E_c 提取结构图 $F_c = \{f_c^1, f_c^2, f_c^3\}$，其中每个 f_c^i 分别作为两个 RSI 模块的输入。由于 UNet 特征与参考特征在空间位置上存在不对齐现象，因此，与传统 DCN 中使用 CNN 来获得形变偏移的做法不同，模型引入了一种跨注意力机制来实现长距离交互。该交互过程可通过方程 3 进行总结。其中，r_i 是 UNet 的特征，而 f_s^i 是结构图。在 softmax 操作中，主要计算了相对于每个查询点的兴趣区域，Φ_q、Φ_k、Φ_v 是线性投影，FFN 表示前馈网络。I_R 是 RSI 的输出。

$$
\begin{aligned}
S_s \in \mathbb{R}^{C_f^i \times H_i W_i} &= \text{flatten}\left(f_s^i\right), \\
S_r \in \mathbb{R}^{C_r^i \times H_i W_i} &= \text{flatten}\left(r_i\right), \\
Q = \Phi_q\left(S_s\right), K &= \Phi_k\left(S_r\right), V = \Phi_v\left(S_r\right), \\
F_{\text{attn}} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V, &\quad \delta_{\text{offset}} = FFN\left(F_{\text{attn}}\right), \\
I_R &= DCN\left(r_i, \delta_{\text{offset}}\right),
\end{aligned}
\tag{3}
$$

3.3 风格对比细化模块

为了更好的实现预期的风格模仿效果，论文提出了 SCR 模块（图 1右），这是一个从一组样本图像中提取风格的字体风格表示学习模块，并结合一种风格对比损失来监督前面提到的的扩散模型，确保生成的字体在全局和局部级别上与目标对齐。SCR 采用 VGG 网络将字体图像嵌入到提取器中。为了有效捕获全局和局部风格特征，模型从 VGG 网络中选择 N 个特征图 $F_{\text{w}} = \left\{f_w^1, f_w^2, ..., f_w^N\right\}$，并将它们用作输入到风格投影器中。投影器通过平均池化和最大池化分别提取不同全局通道的特征，然后将它们在通道维度上拼接，得到特征 $F_{\text{g}} = \left\{f_g^1, f_g^2, ..., f_g^N\right\}$。最后，经过几次线性投影后，得到风格向量 $V = \left\{v^0, v^1, ..., v^N\right\}$。

风格向量 V 可以为扩散模型提供监督，引导其模仿风格。因此，论文采用一种对比学习策略，在该策略中，利用预训练的 SCR 模块，并结合风格对比损失 L_{sc} 来监督生成的样本 x_0 的风格是否与目标样本风格一致，同时与负样本区分开来。为了确保内容无关性和风格相关性，模型将目标图像作为正样本，选择 K 个与其内容相同但风格不同的负样本。因此，SCR 的监督过程可总结如下：

$$
\begin{aligned}
&\boldsymbol{V}_0 = \text{Extrac}\left(\boldsymbol{x}_0\right), \quad \boldsymbol{V}_p = \text{Extrac}\left(\boldsymbol{x}_p\right), \quad \boldsymbol{V}_n = \text{Extrac}\left(\boldsymbol{x}_n\right) \\
&\mathcal{L}_{sc} = -\sum_{l=0}^{N-1} \log \frac{\exp\left(v_0^l \cdot v_p^l / \tau\right)}{\exp\left(v_0^l \cdot v_p^l / \tau\right) + \sum_{i=1}^{K} \exp\left(v_0^l \cdot v_{n_i}^l / \tau\right)}
\end{aligned}
\tag{4}
$$

其中，$Extrac$ 表示风格提取器的作用。K 是负样本的数量，V_0、V_p 和 V_n 分别表示生成的正样本、正样本和负样本的风格向量。v_0^l, v_p^l, $v_{n_i}^l$ 分别表示第 l 层生成的正样本、目标样本和负样本的向量特征。τ 是温度超参数，设置为 0.07。

3.4 评估指标

对于实验结果，我采用了 4 个不同的量化评估指标来全面衡量生成字体的质量，具体包括：FID、SSIM、LPIPS 和 L1 损失。FID 是一种衡量生成图像与真实图像分布差异的指标，值越小意味着生成图像在整体分布上越接近真实图像，从宏观角度评估生成字体与目标字体在视觉上的整体相似度和质量，能反映模型生成的字体是否符合真实字体的整体风格和视觉特征。SSIM 用于考量生成样本与目标样本的结构相似性，通过对比图像的亮度、对比度和结构信息等方面，来确定生成字体在结构上与目标字体的一致性程度，其取值范围在-1 到 1 之间，越接近 1 表示结构相似性越高，能直观体现模型对字体结构的把握和还原能力。LPIPS 是基于学习到的感知特征来计算图像之间的相似性，更贴合人类的视觉感知特点，可有效判断生成字体在视觉感知层面与目标字体的接近程度，有助于评估生成字体是否在人眼看起来足够自然、真实且符合预期风格。L1 损失可以衡量生成样本与目标样本的像素级差异，即计算每个像素点的绝对差值之和，能精确反映生成字体在像素层面与目标字体的精确程度，值越小表示生成字体与目标字体的像素差异越小，意味着模型在细节上的还原和生成能力越强。

4 复现细节

4.1 与已有开源代码对比

原论文虽然对代码进行了开源，但是数据集并没有开源。我使用的数据集主要是在 github 上开源的 free-font 数据集，它包含了 199 种免费商用的中文字体，其收录的字体都能找到其开源或者开放免费商用授权的出处。这里我随机选择 150 种字体作为训练集，共包含 400 个中文字符。同时设置了两个测试集，SFUC 测试集包含 100 种随机选择的训练时用过的字体（100 个训练期间未见过的字符），UFUC 测试集包含 49 种训练时未用过的字体和 100 个训练期间未见过的字符，另外还有 UFSC 测试集（49 种训练时未用过的字体和 400 个训练时用过的字符）用于额外对比。同时根据字符笔画数将字符分为三个复杂度级别（易、中、难），具体分类标准为：笔画数在 6-10 之间为易，11-20 之间为中，大于 21 为难（难度设置参考原论文）。这里计算中文字符笔画数我用的是 github 上的 chinese-xinhua 数据集，它收录了 16142 个汉字，每个汉字都包含其笔画信息，但经过验证，其笔画数存在一定的误差，因此在确定字符笔画数时，我先用 chinese-xinhua 中的笔画数初步确认字符的笔画，然后通过新华字典 API 进一步确认该字符的笔画数。

4.2 实验环境搭建

本文的复现工作主要是部署在 Linux 系统的服务器上，内含 4 张 V100 显卡。实验环境具体包括 Python 3.9，Pytorch 1.13.1，CUDA 11.7 等。

5 实验结果分析

具体实验结果如表 1 所示，可以看到在不同的测试集中，简单难度的字体生成效果在各项评估指标中均表现最佳，中等难度的次之，这体现出模型对于笔画数较少的字符生成的来

表 1. 实验结果（粗体表示最优, 下划线表示次优）

	SFUC			UFUC			UFSC		
	easy	medium	hard	easy	medium	hard	easy	medium	hard
FID↓	**38.1663**	38.2489	48.0365	**42.0324**	43.6236	51.2141	**35.1481**	38.2817	42.8467
LPIPS↓	**0.1416**	0.1485	0.1631	**0.1513**	0.1592	0.1821	**0.1526**	0.1692	0.1823
L1↓	**0.1508**	0.1826	0.2217	**0.1789**	0.2482	0.2592	**0.1828**	0.2014	0.2407
SSIM↑	**0.6569**	0.5922	0.5154	**0.6923**	0.6322	0.5928	**0.6052**	0.5532	0.4982

图 3. 日文生成样例こんにちは

的效果更好。而对于不同的测试集，UFSC 生成的结果相对最好，UFUC 生成的结果相对最差，这体现了模型对于训练时使用过的字符更加敏感，生成出来的效果更好。

此外，图 3是使用 FontDiffusers 生成的跨语言文字，可以看到其生成出来的效果直观来看还是很不错的, 这表现出其语言生成方面是灵活的，具有一定的跨域能力。

6 总结与展望

未来可进行探索如何生成手写风格的字体以及一次生成多个字符。

参考文献

[1] Z. Yang, D. Peng, Y. Kong, Y. Zhang, C. Yao, and L. Jin. Fontdiffuser: One-shot font generation via denoising diffusion with multi-scale content aggregation and style contrastive learning. In *Proc. AAAI Conference*, pages 6603–6611, 2024.

[2] Y. Xie, X. Chen, L. Sun, and Y. Lu. Dg-font: Deformable generative networks for unsupervised font generation. In *Proc. CVPR*, pages 5130–5140, 2021.

[3] Y. Kong, C. Luo, W. Ma, Q. Zhu, S. Zhu, N. Yuan, and L. Jin. Look closer to supervise better: one-shot font generation via component-based discriminator. In *Proc. CVPR*, pages 13482–13491, 2022.

[4] C. Wang, M. Zhou, T. Ge, Y. Jiang, H. Bao, and W. Xu. Cf-font: Content fusion for few-shot font generation. In *Proc. CVPR*, pages 1858–1867, 2023.

[5] C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, and M. Norouzi. Palette: Image-to-image diffusion models. In *Proc. SIGGRAPH*, pages 1–10, 2022.

[6] H. He, X. Chen, C. Wang, J. Liu, B. Du, D. Tao, and Y. Qiao. Diff-font: Diffusion model for robust oneshot font generation. *arXiv preprint arXiv:2212.05895*, 2022.

[7] D. Sun, T. Ren, C. Li, H. Su, and J. Zhu. Learning to write stylized chinese characters by reading a handful of examples. *arXiv preprint arXiv:1712.06424*, 2017.

[8] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Proc. NeurIPS*, volume 33, pages 6840–6851, 2020.

[9] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proc. CVPR*, pages 7132–7141, 2018.

[10] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *Proc. ICCV*, pages 764–773, 2017.