

# 通过传递相似性引导的全局样式生成少量字体，并且 量化本地样式

潘伟 Anna ZhuXinyu ZhouBrian Kenji IwanaShilin Li

<sup>1</sup> 武汉理工大学计算机科学与人工智能学院

<sup>2</sup> 九州大学人机界面实验室

{aawei, annazhu, 297932, 士林里}@whut.edu.cn iwana@ait.kyushu-u.ac.jp

## 抽象

自动小样本字体生成 (AFFG) 旨在仅使用少量字形引用生成新字体，从而减少手动设计字体的人工成本。但是，传统的样式-内容解缠 AFFG 范式无法捕获不同字体的不同局部细节。因此，提出了许多基于组件的方法来解决这个问题。基于组件的方法的问题在于，它们通常需要特殊的预定义字形组件，例如笔画和部首，这对于不同语言的 AFFG 来说是不可行的。在本文中，我们提出了一种新颖的字体生成方法，通过聚合来自字符相似性引导的全局特征和风格化组件级表示的样式。我们通过测量沿相应频道与内容特征的距离来计算目标角色和引用样本的相似度分数，并将它们分配为聚合全局样式特征的权重。为了更好地捕捉局部样式，采用基于交叉注意力的样式迁移模块将参考字形的样式传递给组件，其中组件是通过向量量化自学的离散潜在代码，无需人工定义。通过这些设计，我们的 AFFG 方法可以获得一整套组件级的样式表示，还可以控制全局字形特征。实验结果反映了所提方法在不同语言文字上的有效性和泛化性，也显示了其与其他最新方法相比的优越性。源代码可以在 <https://github.com/awei669/VQ-Font> 中找到。

但是，传统的字体设计在很大程度上依赖于专业设计人员手动渲染每个字符的字形样式，这使得字体的创建极其昂贵和劳动密集型，尤其是对于字形丰富的脚本。

近年来，随着深度学习技术的发展，人们提出了许多自动 Few-shot 字体生成 (AFFG) 方法。它们是使用卷积神经网络 (CNN) [22]、生成对抗网络 (GAN) [27]、Transformer [32] 等创建的。AFFG 方法仅使用几个参考字形图像来自动生成不同的字形。典型的策略遵循风格和内容解缠和组合范式 [38, 23]，采用全局样式表示或组件级样式表示。全局样式表示 [13, 38] 是从每种样式的所有引用中学习和提取的，这可以捕获全局特征，例如字符大小和笔画空间。但是，它缺乏各种局部细节的表示，例如局部笔画的形状和长度以及衬线大小。相反，组件样式表示类别 [11, 6, 20] 通常将共享相同字体的每个参考样本分解为预定义的组件和部首。它要么在字形图像和相应的组件标签上联合条件样式编码器，要么采用组件标签分类损失来训练样式编码器。这可能是不可行的，因为每个角色都应该手动关联一组特定的组件，这在申请新脚本时需要更多的准备。此外，对于不同的内容图像，它们与参考样本的局部关系可能会有所不同。这意味着在给定固定参考样本时，需要重新计算不同内容字符的本地样式表示，这会增加计算成本。

为了解决上述问题，我们在本文中提出了一种 AFFG 的混合全局和本地式传输方法。由于字体的全局样式表示控制了更多的样式内一致属性，例如字符的位置、大小、描边粗细和间距，而 lo-

1. 简介 字体设计技术可以使许多关键应用受益，例如徽标设计、文本相关任务的数据增强、手写模仿和识别等。

---

\*通讯作者

笔触粗细 CAL 样式表示侧重于捕获样式间不一致的组件细节，例如，笔触形状、衬线度、笔触变形。因此，我们利用全局和本地样式进行功能互补。为了获得全局样式特征表示，我们通过测量目标字形和引用样本的内容特征距离来计算它们的相似度分数，然后将它们分配为权重，用于聚合样式特征。对于本地样式特征表示，首先会自动学习字形组件，这些组件是通过矢量量化从一组字形中分解出来的离散潜在代码。然后，使用交叉注意力转换器来传输组件样式，将学习组件的表示作为查询，将引用字形的样式表示作为键和值。对比学习用于以无监督的方式学习本地风格。对于每个前向传递，参考样本中的样式可以传输到所有元件上。因此，这种本地样式提取过程独立于内容字形，避免了对不同输入进行多个组件表示计算。最后，将全局和本地样式表示与内容特征相结合，然后解码为目标字形。此外，我们采用 GAN 和自我重建策略来训练模型，而无需强大的监督。因此，它可以很容易地应用于不同的脚本字体生成。

我们证明了所提出的方法主要对中国人的有效性。实验结果反映了组合全局和局部表示的必要性，并且还表明，在参考示例非常有限的情况下，我们的方法优于其他最先进的 (SOTA) AFFG 方法。

综上所述，本文的贡献如下：

- 我们提出了一种新的 AFFG 方法，利用互补的全局和局部表示，该方法能够捕获参考字形的样式内一致属性和样式内不一致的结构。
- 内容相似性用于获取全局样式。它考虑了类似程度的字形结构。此策略可以更好地传输拥有相同组件的字形的样式。
- 采用预训练的基于向量量化的变分自动编码器 (VQ-VAE) 自动提取组件，不需要组件标签。本地样式可以通过 one-forward pass 中的交叉注意力传输到所有组件。它对于字体库的创建是有效的，因为它与内容无关。

提出了一种风格对比损失来无监督地传输组件级样式。

• 实验结果表明，我们的模型对于看不见的字体、看不见的字符和不同的脚本具有很强的泛化性。即使使用非常有限的参考样本，它也能实现字体生成的 SOTA 性能。

此外，它可以以 zero-shot 方式将样式转移到跨语言上。

## 2. 相关作品 Image-to-Image Translation。图像到图像

(I2I) 翻译旨在学习源域和目标域之间的映射，同时保留源域的内容。基于 GAN 的方法 [12, 7, 16] 在该领域得到广泛应用。Pix2pix [12] 首先将条件 GAN 引入到具有配对数据的 I2I 任务中。CycleGAN [39] 为无监督 I2I 翻译引入了圆一致性损失 [5, 39]。最近，提出了几种 I2I 方法 [2, 4] 来解决多类无监督 I2I 翻译问题，旨在同时在给定相同输入的情况下生成多种样式的输出。

字体生成属于典型的 I2I 翻译任务，因此通用的 I2I 翻译方法可以自适应地修改字体生成 [17, 9]。

### 字符样式转移。字符样式迁移

主要集中在排版转移上。传统方法通过形状建模 [31] 和统计建模 [30] 构建了转换。最近的方法将深度学习技术应用于字符合成 [3]。一些方法能够合成字形以及纹理效果 [1, 36, 8]。但是，字体生成任务更强调字符形状的一致性，而不是文本效果。

### 字体生成的全局样式内容解缠

eration. 全局样式-内容解缠和重组是字体生成的一种流行策略。它将每种字体样式建模为通用表示形式。EMD [35] 提出了一种具有两个编码器的网络架构，一个用于内容，另一个用于样式。他们通过混合内容和样式特征，然后解码混合特征来生成任意字体。MDM [37] 构建了一个框架，将文本图像分解为三个因素：文本内容、字体和样式特征，然后重新混合不同图像的因素以传递新的样式。ELDF [15] 利用了新的一致性损失，它迫使堆叠输入的编码特征的任意组合拥有文本内容和字体样式的一致特征。

但是，字体的局部细节更加多样化，仅通过全局样式表示很难捕获。

### 基于组件的字体生成。自从性格

汉字等 ters 可以分解，最近出现了一些基于组件的字体生成方法 [6, 19, 20]。它们不会对整个图像进行编码以提取样式信息，而是提取构成音节的每个组件的样式信息并使用它

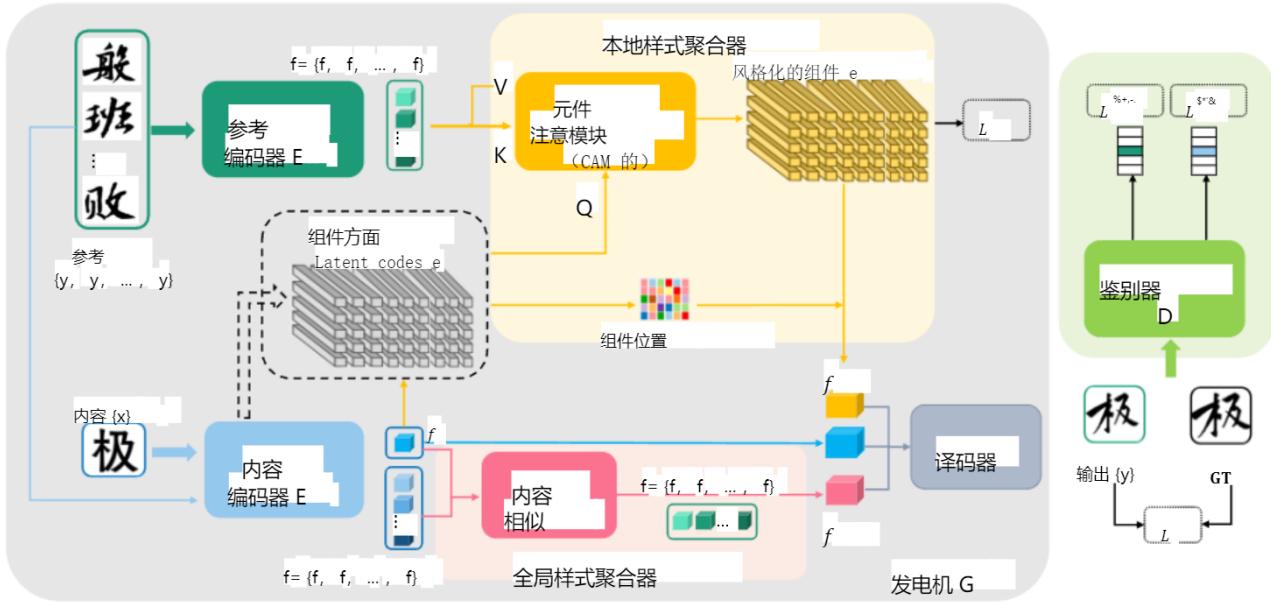


图 1. 我们模型的架构。该生成器由五个部分组成：预先训练的内容编码器、参考样式编码器（标记为深绿色）、通过 CAM 的本地样式聚合器（标记为黄色）、具有内容相似性指南的全局样式聚合器（标记为粉红色）以及结合内容和样式特征的解码器以及用于字体生成的样式特征（标记为灰色）。遵循判别器（标记为绿色）来区分真实和虚假图像，并同时对生成字符的内容和样式类别进行分类。

生成字体。例如，CalliGAN [28] 将字符分解为组件，并提供低级结构信息，包括笔画的顺序，以指导生成过程。SA-VAE [24] 将文本和内容解开为两个不相关的领域，将汉字编码为高频字符结构构型和部首。Tang et al. [25] 提出了 FS-Font 模型，该模型采用交叉注意力机制将参考样式聚合成细粒度的样式表示。XMP-Font [18] 提出了一种自我监督的跨模态预训练跨模态转换器网络，用于对笔画级、组件级和字符级的风格表示进行建模。

基于组件的字体生成方法显著提高了生成的字形的质量，并且使用少量引用。但是，它们中的大多数要求您手动预定义格式或组件。

3. 方法 在本节中，我们将详细介绍整体框架、不同的模块和训练策略。

3.1. 整体管道给定一个内容图像  $x$  和一组共享相同字体的参考字符图像  $y = \{y_1, y_2, \dots, y_n\}$ ，我们的模型旨在使用  $x$  和参考图像的字体样式生成具有相同内容的角色。

整体框架如图 1 所示。它由一个生成器 G 和一个多任务判别器 D 组成。

生成器 G 包含五个部分，如图 1 所示。内容编码器 Etakes 内容图像作为输入以提取其结构表示  $f$ 。它通过 VQ-VAE 进行预训练，以获得组件级潜在代码  $e$ 。样式编码器旨在从参考图像  $y$  中学习样式表示。具体来说，它独立输入每个参考图像，以将它们映射到样式潜在向量  $f = \{f_1, f_2, \dots, f_n\}$ 。之后，将遵循本地样式聚合器 (LSA) 将样式传输到学习的组件上。它通过交叉注意力模块 (CAM) 工作，其中包含所有组件的潜在代码、eas 查询以及引用字符的表示，以及键和值。在组件风格化之后，可以通过在  $e = \{e_1, e_2, \dots, e_n\}$  中搜索最相似的代码及其表示来获得与内容相关的本地样式  $f$ 。全局样式聚合器 (GSA) 对样式表示  $f$  重新加权，并在通道上对所有样式表示  $f$  求和，得到全局表示  $f$ 。权重是每个参考字符和输入字符之间内容表示的标准化距离，分别显示  $f = \{f_1, f_2, \dots, f_n\}$  和  $f_{\text{in}}$  图 1。然后，使用解码器生成目标图像  $y$ ，其中包含内容特征  $f$ 、本地样式表示  $f$  和全局样式表示的串联特征

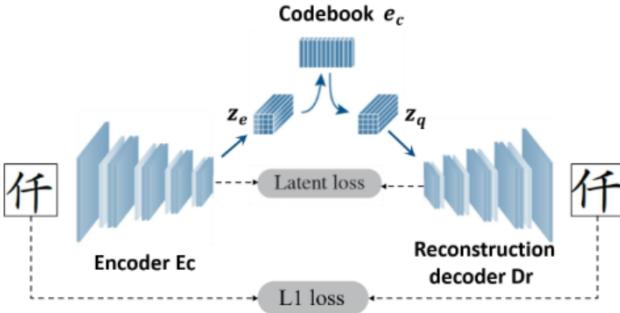


图 2. 字形功能 decomcomposition network 用于预训练内容编码器和获取组件表示。

tation fas input.

在训练过程中，多任务判别器被改造成与生成器进行最小-极大博弈，以区分真实的  $\hat{y}$  和虚假的  $y$ 。为了确保生成的字形具有正确的参考样本样式，并且仍保留输入内容字符的结构，鉴别器会输出每个字符的样式和内容类别的二进制分类。

### 3.2. 字形编码器和组件分解

内容编码器在特征分解网络上进行了预训练，并在字体生成任务中保持冻结状态。在预训练步骤中，用于将字形图像分解成分量级的潜在代码，并在一定的字符模板集上进行训练以进行重建。特征分解网络的概述如图 2 所示。它包含一个内容编码器和一个重建解码器，由使用潜在码本  $e \in R$  的离散潜在空间连接。

编码器基于 CNN 构建，并将输入字形图像映射到潜在表示形式  $z \in R$ ，其中  $d$  是通道数， $h$  和  $w$  分别表示特征图的高度和宽度。随后，矢量量化 [26] 用于离散  $z_{as}$ ，如下所示：

$$z = e, \quad s.t. \quad n = \arg \min_{n \in \{1, 2, \dots, N\}} \|z - e\|, \quad (1)$$

其中，通过在由  $ddimensional$   $n$  代码向量组成的 Codebook  $e$  上执行最近邻查找，将每个空间元素向量  $z \in R^{in}$  替换为最近的代码向量。最后，重建解码器使用匹配的代码  $z \in R^{as}$  作为输入并输出重建的字形。

为了优化此过程，编码器和码本被更新以最小化目标  $Lin$  方程 (2)，它指的是基于  $L$  的重建损失和潜在损失  $L$

如下：

$$L = L + L$$

$$= \|x - \hat{x}\| + \alpha \|sg[z] - e\| + \beta \|z - sg[e]\|,$$

(2) 其中  $sg$  是一个停止梯度算子，在前向计算时被定义为恒等式，偏导数为零。 $\alpha$  和  $\beta$  是平衡超参数。在实验中，我们分别将它们设置为 1 和 0.1。方程 (2) 中的第一项确保在前向传递中没有信息丢失。第二项通过将嵌入向量  $e$  移向编码器输出  $z$  来更新 codebook 变量。同时，第三项鼓励编码器输出匹配目标代码向量。编码器和解码器之间没有旁路连接，因此编码器处理的字形集的所有组件信息都可以在潜在空间中压缩 [21]。

在这个预训练任务之后，我们修复内容编码器  $Eas$  以及组件 codebook  $e$ ，以构建字体生成模型。样式编码器  $E$  具有与  $E$  相同的架构，但它是从头开始训练的。

3.3. 本地样式聚合器最近提出的方法 [18, 19] 通过从输入内容和参考样式中获得注意力来提取字符的本地样式表示。它效率低下，尤其是对于创建包含大量字形的新字体库。由于每个字形都必须经过一次 attention 模型才能进行特征聚合。但是，如果它们具有相同的笔画或部首，它们中的大多数实际上共享一些本地表示。从这一点开始，我们设计了一个 LSA 模块，核心是一个 CAM 模块，用于将样式转移到所有组件代码  $e$  而不是输入字形上。

CAM 块建立在堆叠的多头变压器之上，具有组件代码、 $eas$  查询和参考样式向量、 $fas$ 、键和值。形式上，特征图  $f \in Rof$  第  $k$  个参考样本分别沿通道轴重塑，形成顺序参考标记  $f \in R$ 。对于  $ihead$ ，对其应用两个线性投影  $W \in Rand$   $W \in Rare$ ，分别获得键  $K = fW$  和值  $V = fW$ 。同时，我们获得学习权重为  $W \in R$  的查询矩阵  $Qby$  线性投影  $e$ 。

注意力模块对查询  $Q$ 、键  $K$  和值  $V$  进行操作，并生成加权平均向量

$\hat{V}$ ，可以表述为：

$$\hat{V} = \text{注意}(Q, K, V) = \text{softmax} \frac{\sqrt{-\frac{QK}{c}}}{\sqrt{-\frac{V}{c}}}$$

(3) 在获得每个 head 中的表示后，我们沿通道维度连接所有  $\hat{V}$ ，并使用

线性投影得到多头注意力结果  $E$ , 如下所示:

$$e = \text{Multi-head} (Q, K, V) = \text{连接 } \hat{V}, \hat{V}, \hat{V}, W, \quad (4)$$

(4) 其中  $m$  是总注意力头数,  $W$  是学习权重。

在我们得到程式化的组件代码  $e = \{e_1, e_2, \dots, e_m\}$  之后。对于每个输入字形, 其局部表示  $f$  可以通过  $e$  通过在  $e$  中搜索最接近的标签  $n$  来聚合, 对于方程 (1) 中表示的  $fas$  中的每个空间元素, 然后选择相应的样式表示  $e$  进行替换。  
3.4. 全局样式聚合器每个参考样本的原始样式表示  $f$  能够为字体生成提供更多的全局信息, 可以控制大小、笔画间距等。因此, 我们直接将它们的特征聚合为全局表示。

如 [36] 所示, 参考文献的样式特征和内容特征并不是完全独立的。从人类的感知观察来看, 不同的角色之间存在一些相关性。如果输入字形具有完全相同的部首或结构, 并且有一些引用, 则在传输过程中应高度引用它们的样式表示。因此, 我们根据内容特征相似性对每个引用字形的各个样式特征进行重新加权。我们提取内容特征  $f = \{f_1, f_2, \dots, f_m\} \in Rof$  每个引用并将它们重塑为  $f \in R$ , 同时, 将内容特征  $f \in Rof$  内容图像重塑为

$\tilde{f} \in R$ . 字形相似度由归一化互相关度量确定, 如下所示:

$$W = \frac{\tilde{f} : f}{\tilde{f} * f}, \quad j \in \{1, 2, \dots, d\}. \quad (5)$$

$Wis$  是一个标量, 表示  $k$  引用与  $j$  channel 上的输入字形之间的相似性。然后, 我们将每个通道上的  $k$  个样本视为:

$$\bar{W} = \text{softmax} \left( \frac{\exp(aW)}{\sum_{k=1}^K \exp(\text{宽})} \right). \quad (6)$$

然后, 将此权重应用于参考样式表示  $f$ , 其中  $jh$  通道特征  $f$  由  $W$  加权, 以聚合全局表示  $fas$ , 表示为:

$$f = \text{连接} \left( \prod_{k=1}^K \bar{W}^k f \right). \quad (7)$$

此全局样式表示  $fas$  与内容特征  $f$  和本地样式表示  $f$  连接, 然后输入到解码器以生成目标字体。

3.5. 训练 我们训练模型从输入内容图像  $x$  和一组参考字形图像  $y = \{y_1, y_2, \dots, y_N\}$  生成图像  $\hat{y}$ 。内容编码器是预先训练的, 并在此期间保持固定。损失由三部分组成, 对抗性损失、匹配损失和风格对比损失。对抗性损失和匹配损失在生成的结果  $y$  和真实值  $\hat{y}$  之间使用。而 style contrast loss 旨在学习组件的区分样式。

对抗性损失。为了使模型生成合理的图像, 我们在框架中采用了多头投影判别器  $D$  为样式标签  $s$  和字符标签  $c$ 。损失函数被实现为铰链 GAN 损失 [33]:

$$\begin{aligned} L &= -E \min (0, -1 + D(\hat{y})) \\ &\quad - \text{艾敏} (0, -1 - D(y)) \end{aligned} \quad (8)$$

$$L = -ED(y),$$

其中  $p$  表示生成的图像集,  $p$  表示真实字形图像集。

匹配损失。要使生成的字符  $y$  在像素级和特征级与地面实况保持一致

$\hat{y}$ , 我们对图像和图像特征都使用  $L_{loss}$ , 如下所示:

$$\begin{aligned} L &= E[\|y - \hat{y}\|] \\ L &= E[\left\| \frac{x}{\|x\|} f(y) - f(\hat{y}) \right\|_1], \end{aligned} \quad (9)$$

其中  $f(y)$  和  $f(\hat{y})$  表示  $D$  的  $lh$  层中的中间特征。

样式对比度损失。给定两个不同的引用集但共享相同的字体样式, 则这些集的关联样式化组件应相同。但是, 如果参考字符集具有不同的字体样式, 则应区分组件样式。因此, 我们将风格对比损失表述如下:

$$\begin{aligned} L &= \left( \sum_{n=1}^N \exp(-\log \left( \prod_{s \in S_n} \exp(ee_s) \right) \right)^{-1} \\ &\quad \times \left( \sum_{n=1}^N \exp \left( \sum_{s \in S_n} \exp(ee_s) \right) \right)^{-1} \right)^{-1} \end{aligned} \quad (10)$$

详细地说, 对于每个特定的字体样式  $s$  和  $e$  是对应的码簿  $e$ , 其中  $e$  是的正码簿对

$e$ 的引用具有相同的字体样式但内容不同，而  $eis$  负码簿对具有不同的字体样式。 $N$  是负样式样本的数量，对于第  $n$  个码本条目  $ein$  codebook  $e \in R$ ，我们总能在相应位置找到正对  $e$  和负对集  $\{e, e, \dots, e\}$ 。

总体目标损失。最后，我们通过以下全目标函数对整个模型进行优化：

$$\min_{E,G} \max_D L + L + \lambda L + \lambda L + \lambda L. (11)$$

$\lambda$ 、 $\lambda$  和  $\lambda$  是加权超参数。我们分别将它们设置为 1、1 和 0.1。

## 4. 实验结果

4.1. 实验设置数据集。我们收集了 386 种中文字体，并根据一级汉字表为每种字体生成了 3500 个汉字。所有角色图片均标准化为  $128 \times 128$ 。从 386 种字体中随机选择一个字体模板来提取字形内容特征，并在整个训练和测试过程中保持固定 [1]。该字体集也用于预训练 VQ-VAE，以获得一组常用零件码本。

中文训练集包含 370 种字体和 3,000 个中文字符，表示为 seen font seen character (SFSC) 集。为了验证我们的方法，我们在两个测试集上评估了字体生成能力：一个是 15 种未见过字体的其余部分，每种字体有 3,000 个见过的字符，表示为见过字体未见过的字符 (UFSC) 集；另一种是剩余的 15 种未见过的字体，每种字体有 500 个未见过的字符，表示为未见过的字体未见过的字符 (UFUC) 集。

实现细节。我们方法的主要训练过程分为两部分。首先，使用 3,000 个共享相同字体的中文字符训练 componentwise codebook。我们将嵌入维度  $d$  设置为 256，将批量大小设置为 256，并将迭代步数设置为 50,000。然后，为了训练整个字体生成模型，我们将 batch size 设置为 48，将三个堆叠 transformer 层中的 attention head 数量设置为 8，迭代步数设置为 500,000。

评估指标。为了评估字体生成质量，我们使用以下五个指标 [14]、均方根差 (RMSE)、结构相似性 (SSIM)、学习感知图像补丁相似度 (LPIPS) [34]、Fr'echet 起始距离 (FID) [10] 和用户研究。用户研究由 10 名志愿者进行，他们观察参考样本并从所有比较方法中投票选出最佳生成结果。

表 1. 组件码本的烧蚀研究。

码本大小	SSIM↑	RMSE↓	LPIPS↓	FID↓	
50	0.512	0.422	0.343	144.2	
80	0.546	0.407	0.296	120.7	
100	0.566	0.390	0.282	110.1	
120	0.568	0.392	0.278	109.5	
150	0.567	0.388	0.282	112.4	

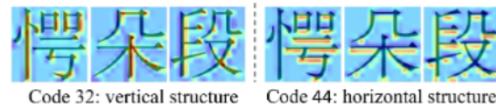


图 3. 潜在代码的可视化。

4.2. 消融研究 我们进行了三次消融研究，以评估我们提出的模型对少数镜头中文字体生成任务的有效性。这些实验是在 UFUC 数据集上进行的。

### 4.2.1 组件码簿大小的影响

组件 codebook 大小反映了脚本的复杂程度。我们将其设置为不同的值，用于训练内容编码器并在 VQ-VAE 中获取组件表示。然后，它们被固定以生成字体。结果显示在表 1 中。我们可以看到，当 codebook 大小超过 100 时，每个指标的性能改进有限。这说明使用 100 个分量就足以描述汉字。当我们用英文脚本训练模型时，我们发现仅使用 15 个代码，模型可以产生非常好的结果。因此，组件 codebook 的大小与脚本有关。更复杂的脚本需要更大的码本大小。在下面的实验中，我们将中文组件码簿大小设置为 100。

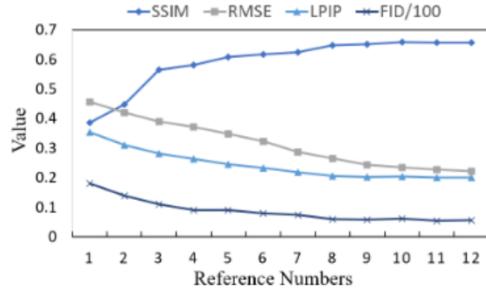
我们从码本中选择某个代码，在空间上计算它与不同字形特征的相似度。32 代码和 44 代码的可视化结果如图 3 所示。我们可以看到 32code 主要提取垂直分量，44 提取水平分量。它演示了 VQ-VAE 策略可以自动生成“组件”。

### 4.2.2 每个模块的效果

为了说明我们方法的 GSA 和 LSA 模块的影响，我们在模型中进行了有和没有它的消融研究。通过删除 LSA，不需要 componentwise 表示。该模型退化为全局内容风格的解纠缠架构，聚合风格表示仅以内容相似性为条件。移除 GSA 零件时，模型聚焦

Reference	雳 枪 嫌	惧 蔡 蔬	鼠 刨 贻
w/o GSA	盈 内 弗	情 息 锐	厕 奸 纠
w/o LSA	盈 内 弗	情 息 锐	厕 奸 纠
Full model	盈 内 弗	情 息 锐	厕 奸 纠
GT	盈 内 弗	情 息 锐	厕 奸 纠

图 4. 带和不带 LSA 和 GSA 模块的定性字体生成结果。第一行和最后一行分别是三个引用和每种字体的真实值。中间的行是相应地生成的结果。红色虚线框指出了我们的完整建议模型更好地生成的细节。



但自动由多个专家设计到 图 5。我们模型的性能会因参考样本数量的变化而变化。

有关描边细节的更多信息，但会丢失 Global Style 控件。我们在收集的 UFUC 数据集上评估了这两个模型和提出的完整模型。图 4 显示了一些比较结果。有和没有这些模块的结果有明显的差异。具有 LSA 的模型可以更好地捕获参考笔画的细节，而生成的字符的大小和内部空格并不总是与给定的参考一致。仅具有 GSA 的模型具有相反的功能。大多数描边都不能很好地生成。完整模型结合了 LSA 和 GSA，实现了更好的性能（参见图 4 红色虚线框中的详细信息）。表 3 中的定量结果进一步证实了这两个模块的重要性。LSA 模型的帮助比 GSA 更多，这表明字形的风格主要在于局部细节。从这些结果中，我们得出结论，LSA 和 GSA 在我们的模型中都是必不可少的，它们使模型能够捕获不同的本地样式并控制全局样式。

#### 4.2.3 参考编码的影响

直观地说，给定更多的参考样本，生成的字体更类似于真实的字体。图 5 显示了参考编号的影响。我们可以看到，当引用的数量从 1 增加到 8 时，生成的字符性能有增量提升。该图显示，当给出更多参考时，它会变得稳定。然而，在最初的几次引用之后，回报递减。详情请参阅附录。

4.3. 与 SOTA 方法比较 我们将我们的方法与五种 SOTA AFFG 方法进行比较。1) FUNIT [17] 使用 AdaIN 模块的两种不同的编码器来生成具有混合内容和样式的新的图像。它属于全局样式-内容解缠 FFG 方法。2) MX-Font [20] 提取多个未明确以组件标签为条件的样式特征

表示不同的本地概念，例如，左侧子字形。它显示出对看不见的语言的良好泛化能力。3) LF-Font [19] 提出了受低秩矩阵分解启发的本地化样式表示，它也提取了组件级特征。4) DG-Font [29] 是一个无监督网络，它使用生成器中的可变形卷积将一种字体的字符变形并转换为另一种字体。它对草书字符产生了很好的效果。5) AGIS-net [8] 使用两种不同的解码器依次生成具有字符形状的图像，然后生成纹理信息。它可以传输字体和文本效果。6) FS 字体 [25] 需要固定的内容引用映射来传输细粒度本地样式。我们使用原始训练超参数和策略通过数据集重新训练所有模型。在所有方法的训练和测试阶段，参考样本都设置为 3。定性和定量比较结果分别如图 6 和表 2 所示。

表 2 显示，我们的方法优于以前的 SOTA 方法，在所有指标上都有很大差距。我们可以观察到 FUNIT 无法捕捉到不同的风格。只有当内容样式和引用样式在视觉上接近时，它才能生成结构化字符。这说明了通用样式表示策略不是 AFFG 的好选择。LF-Font 生成的字体视觉质量较差，因为参考编号非常有限。它的性能不稳定，因为它们容易丢失某些样式的笔触或组件失真。DG-Font 可能无法执行样式传输，尤其是当引用样式与内容的样式明显不同时。此外，它还会生成包含特征伪影的字形。AGISNet 更稳定地生成不同样式的字体和补码结构。它可以更好地传递全局特征，例如笔画的大小和粗细。MXFont 生成的字符比其他方法更好，但与我们的方法相比，它无法生成细粒度样式特征的概率更高。FS-Font 的生成结果也不令人满意，尤其是在看不见的引用上，由于随机选择

表 2.UFSC 和 UFUC 数据集的定量比较结果

数据	方法	SSIM↑	RMSE↓	LPIPS↓	FID↓	用户研究↑
UFSC	FUNIT [17]	0.512	0.426	0.361	161.8	1.2%
	LF 字体 [19]	0.548	0.406	0.334	123.9	3.9%
	AGIS-NET [8]	0.529	0.409	0.302	145.0	10.3%
	DG 字体 [29]	0.517	0.419	0.312	149.0	11.6%
	MX 字体 [20]	0.523	0.407	0.297	94.5	14.4%
	FS 字体 [25]	0.570	0.379	0.327	172.6	21.5%
UFUC	Ours	0.636	0.341	0.225	93.7	58.6%
	FUNIT [17]	0.468	0.472	0.379	153.4	3.4%
	LF 字体 [19]	0.481	0.462	0.377	136.7	4.8%
	AGIS-NET [8]	0.550	0.400	0.310	137.2	11.5%
	DG 字体 [29]	0.493	0.397	0.312	136.7	10.8%
	MX 字体 [20]	0.478	0.447	0.333	114.8	15.2%
Ours	FS 字体 [25]	0.418	0.474	0.377	173.7	19.5%
	Ours	0.566	0.390	0.282	110.1	54.3%

Dataset	UFUC					UFSC				
Reference	嘉明主	泛懂隘	腰毡携	镑符挂	塞暗室	原初忍	香贬刷	矩财需	斧维仇	倡悲捕
FUNIT	想奠舱	御僻拌	牲蚂耿	拘舟净	凄揭蛮	政姜避	轧倍沉	牛学	少白内	乳咙游
LF-Font	想奠舱	御僻伴	牲蚂耿	拘舟净	凄揭蛮	政姜避	轧倍沉	维洪学	英查贞	乳咙游
DG-Font	想奠舱	御僻拌	牲蚂耿	拘舟净	凄揭蛮	政姜避	轧倍沉	维洪学	英查页	乳咙游
AGIS-Net	想奠舱	御僻拌	牲蚂耿	拘舟净	凄揭蛮	政姜避	轧倍沉	维洪学	英查页	乳咙游
MX-Font	想奠舱	御僻拌	牲蚂耿	拘舟净	凄揭蛮	政姜避	轧倍沉	维洪学	英查页	乳咙游
FS-Font	想奠舱	御僻拌	牲蚂耿	拘舟净	凄揭蛮	政姜避	轧倍沉	维洪学	英查页	乳咙游
Ours	想奠舱	御僻拌	牲蚂耿	拘舟净	凄揭蛮	政姜避	轧倍沉	维洪学	英查页	乳咙游
GT	想奠舱	御僻拌	牲蚂耿	拘舟净	凄揭蛮	改善遍	轧倍沉	维洪学	英查页	乳咙游

图 6. 每种方法在 UFUC 和 UFSC 数据集上的 FFG 结果。我们表示五种不同字体的生成样本，每种字体给定 3 个参考和 3 个内容图像。红色框表示更好的生成详细信息。

表 3.UFUC 数据集中不同模型的消融研究。

方法	SSIM↑	RMSE↓	LPIPS↓	FID↓	用户研究↑	无 LSA 0.496 0.422
0.318	145.0	7.3%	无 GSA 0.532	0.402	0.293	132.1 20.3% 完整模型
0.566	0.390	0.282	110.1	72.4%	—	—

content-reference 对。

4.4. 应用于其他语言为了验证我们的方法对其他语言的通用性，我们收集了 125 个日语字符的 60 种字体。随机选择 50 种字体，每种字体有 100 个字形进行训练，其余字体进行测试。一些结果如图 7 所示。由于日本人的字符结构比中国人的字符结构简单得多，因此生成的结果更接近基本事实。此外，我们的方法可以将样式转移到看不见的内容上，包括跨语言的字形，如图 8 所示。

5. 结论 我们在本文中提出了一种新的 AFFG 方法

Content	Reference
えゑぎ	サゼとめはびほまめらわ力もぐべん サゼとぬはびほまめらわ力もぐべん サゼとぬはびほまめらわ力もぐべん
むのお	サゼとぬはびほまめらわ力もぐべん サゼとぬはびほまめられ力もぐべん
るぎあ	サゼとぬはびほまめらわ力もぐべん サゼとぬはびほまめらわ力もぐべん

它可以聚合有限引用的全局和局部样式图 7. 日语脚本的 FFG 结果。对于每种样式，上行和下行分别表示模型生成结果和 GT。

ences.设计的 LSA 和 GSA 模块被分配了不同的任务，以分别捕获字体的局部细节和全局结构。LSA 采用了一种 crossattention 机制，无需手动定义即可将样式转移到所有自学组件上。而 GSA 利用了不同字形的相似性，然后用它来指导所有引用的全局样式组合。实验证明了这两个模块的有效性，并且还表明我们提出的方法在定量和定性方面明显优于其他方法。

参考	内容	アウラえおがセヒふゐ
别福混	アウラえおがセヒふる	アウラえおがセヒふる
伏桔快	アウラえおがセヒふる	アウラえおがセヒふる
(a)		
参考	内容	斗 双 𠂔 𠂔 𠂔 𠂔 𠂔 𠂔
痴佳体	斗 双 𠂔 𠂔 𠂔 𠂔 𠂔 𠂔	斗 双 𠂔 𠂔 𠂔 𠂔 𠂔 𠂔
版贾句	斗 双 𠂔 𠂔 𠂔 𠂔 𠂔 𠂔	斗 双 𠂔 𠂔 𠂔 𠂔 𠂔 𠂔
(b)		
参考	生成的中文样本	生成的 Japanese 样本
宣 呢 除	绒 渠 膨 捻 陵	ライイアあ
医 脸 别	聘 惩 懒 傍 管	おエえウカ
受 追 坐	黔 穆 橘 澄 嚎	うせむずめ
筏 槽 播	黎 罐 覆 翻 鞍	げぎキきが
预 缺 范	潘 蹤 蔑 骤 黯	るほおびぜ
街 登 和	舆 聲 膝 殿 祷	なばゆあぬ
旁 骨 板	畸 蹦 蹤 恐 鞍	れはほのせ
释 旅 盖	廉 藉 噎 癇 濕	ぽぐすサ五
课 急 放	貳 瞳 窥 稽	あぬをゆば
智 蒙 弹	澜 雌 襟 滚 霽	づゑばぶせ

(c)

图 8. 对 (a) 日语字形和 (b) 韩语字形进行跨语言推理。 (c) 在少发模式下生成的中国结果和在零发模式下生成的日本结果。

itatively.但是，我们的方法仅限于两个方面。如果只给出一个参考或非常花哨的字体，例如，decoration 或 shadow，生成的结果是不令人满意的。

## 确认

这项工作得到了模式识别国家实验室（NLPR）开放项目计划（No.20220004）的支持。我们感谢所有匿名审稿人对本文的建设性评论。

## 附录

除了原始论文中介绍的结果和讨论外，我们还在本附录中提供了更多的实验和结果，进一步解释了我们方法的有效性和通用性。

### 不同引用字符对生成结果的影响

字体样式是一个美学概念，与特定字符无关。在少量 shot 字体生成过程中，给定具有不同内容的参考字符，提供的样式表示应该是相同的。

参考	GT	鞠 肖 剥 频 裙 翔 祥 凝 挨
搗 册 被	鞠 肖 剥 频 裙 翔 祥 凝 挨	鞠 肖 剥 频 裙 翔 祥 凝 挨
海 梗 畅	鞠 肖 剥 频 裙 翔 祥 凝 挨	鞠 肖 剥 频 裙 翔 祥 凝 挨
(a)		
参考	GT	揍 烩 寝 漂 菊 镜 筷 壁 德
榆 膀 蔡	揍 烩 寝 漂 菊 镜 筷 壁 德	揍 烩 寝 漂 菊 镜 筷 壁 德
燥 稀 魁	揍 烩 寝 漂 菊 镜 筷 壁 德	揍 烩 寝 漂 菊 镜 筷 壁 德

图 9. 生成相同的目标字符，引用来自相同样式字体的不同内容的字符。

参考集	苏 横 笑 陈 洁 伟 豪 科 腋
Ref1	床 亚 哇 泉 责 度 昔 层 宇
编号1-3	床 亚 哇 泉 责 度 昔 层 宇
编号1-5	床 亚 哇 泉 责 度 昔 层 宇
参考文献	床 亚 哇 泉 责 度 昔 层 宇
编号1-9	床 亚 哇 泉 责 度 昔 层 宇
GT	床 正 哇 泉 责 度 昔 层 宇

图 10. 通过改变参考集大小生成的样本。每个中间行显示由给定不同数量 (1, 3, 5, 7, 9) 的引用生成的样本。数字 (1 到 9) 表示要使用的每个字符的顺序。目标字形显示在底行中。

在我们的训练过程中，我们随机选择 k 样式的参考字符来生成字符。在推理阶段，我们选择与参考集样式相同的内容不同的字符。生成的结果如图 9 所示。生成结果的整体样式和局部样式之间存在细微的差异。

### 使用不同参考文献数量的定性结果

直观地说，给定参考字符的数量越多，提取的样式特征应该越丰富、越准确。在推理阶段，我们更改参考字符的数量并观察对生成结果的影响。从图 10 中可以观察到，生成字符的整体和局部质量与参考字符的数量大致呈正相关。当数字从 1 增加到 5 时，质量提升更加明显，超出此数字的改进是有限的。在我们的论文中，参考图像的数量设置为 3，模型训练和其他方法的定性和定量比较都是用这个设置完成的。

## 引用

- [1] 萨曼·阿扎迪、马修·费舍尔、弗拉基米尔·金、王昭文、伊莱·谢特曼和特雷弗·达雷尔。多内容 gan, 用于 few-shot 字体样式传输。在 CVPR, 2018 年。2, 6 [2] Kyungjune Baek, Yunjey Choi, Youngjung Uh, Jaejun Yoo 和 Hyunjung Shim.重新思考真正无监督的图像到图像的转换。在 CVPR, 2021 年。2 [3] 舒米特·巴鲁贾。学习排版风格: 从 dis-
- 对综合的犯罪。机器视觉与应用, 28:551–568, 2017.2 [4] Deblina Bhattacharjee, Seunghyong Kim, Guillaume Vizier 和 Mathieu Salzmann。Dunit: 基于检测的无监督图像到图像转换。在 CVPR, 2020 年。2 [5] 康斯坦丁·诺斯·布斯马利斯、内森·西尔伯曼、大卫·多汉、杜米特鲁·埃尔汉和迪利普·克里希南。使用生成对抗网络进行无监督像素级域适应。在 CVPR, 2017 年。2 [6] Junbum Cha, Sanghyuk Chun, Gayoung Lee, Bado Lee, Seonghyeon Kim 和 Hwalsuk Lee。具有双重内存的 Few-shot 组合字体生成。在 ECCV, 2020 年。1, 2 [7] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, 和 Jaegul Choo. Stargan: 用于多域图像到图像转换的统一生成对抗网络。在 CVPR, 2018 年。2 [8] 高明、郭元、连周辉、唐英民和肖建国。通过一阶段 fewshot 学习进行艺术字形图像合成。ACM Trans. Graphics, 38 (6) : 185, 2020.2, 7, 8 [9] 高一鸣和吴江琴.通过骨架变换和笔触渲染进行基于 GAN 的未配对汉字图像翻译。在 AAAI, 2020 年。2
- [10] 马丁·豪塞尔、休伯特·拉姆绍尔、托马斯·温特辛纳、伯恩哈德·内斯勒、Günter Klambauer 和 Sepp Hochreiter。由两个时间尺度更新规则训练的 gans 收敛到纳什均衡。在 NeurIPS 中, 2017 年。6
- [11] 黄耀雄、何梦超、金连文和永攀 王。Rd-gan: 通过根式分解和渲染进行少量/零镜头的汉字风格迁移。在 ECCV, 2020 年。1
- [12] 菲利普·伊索拉、朱俊燕、周廷辉和阿列克谢 A 埃夫罗斯。使用条件对抗网络进行图像到图像的翻译。在 CVPR, 2017 年。2
- [13] 江岳、连周辉、唐英民和肖建国。Scfont: 通过深度堆栈网络生成结构引导的中文字体。在 AAAI, 2019 年。1
- [14] 孔玉欣, 罗灿杰, 马伟宏, 朱启元, 盛-gao Zhu, Nicholas Yuan, 和 Lianwen Jin.仔细观察以更好地监督: 通过基于组件的鉴别器生成 One-shot 字体。在 CVPR 中 2022 年。6
- [15] Jeong-Sik Lee, Rock-Hyun Baek 和 Hyun-Chul Choi。通过编码器学习解纠缠特征生成 Bitrary 字体。传感器, 第 2374 页, 2022 年。2
- [16] Alexander H Liu, Yen-Cheng Liu, Yu-Ying Yeh, and Yu-Chiang Frank Wang.用于多域图像转换和操作的统一特征解缠器。神经IPS, 31,2018 年。2
- [17] 刘明宇, 黄勋, Arun Mallya, Tero Karras, Timo 艾拉、Jaakko Lehtinen 和 Jan Kautz。少镜头无监督图像到图像的转换。在 CVPR, 2019 年。2, 7, 8
- [18] Wei Liu, Fangyue Liu, Fei Din, Qian He, 和 Zili Yi. Xmp-font: 用于 fewshot 字体生成的自我监督跨模态预训练。在 CVPR 中, 2022 年。3, 4
- [19] Song Park, Sanghyuk Chun, Junbum Cha, Bado Lee 和 沈贤贞。使用本地化样式表示和分解生成 Few-shot 字体。在 AAAI, 2021 年。2, 4, 7, 8
- [20] Song Park, Sanghyuk Chun, Junbum Cha, Bado Lee 和 沈贤贞。多个头比一个头好: 由多个本地化专家生成 Fewshot 字体。在 CVPR, 2021 年。1, 2, 7, 8
- [21] Ali Razavi, Aaron van den Oord 和 Oriol Vinyals。一般- 使用 VQ-VAE-2 处理各种高保真图像。在 NeurIPS, 2019 年。4
- [22] 埃文·谢尔哈默、乔纳森·朗和特雷弗·达雷尔。完全地 用于语义分割的卷积网络。IEEE 的 Trans. Pattern Anal. Mach. Intell., 39 (4) : 640–651, 2017.1
- [23] 尼基塔·斯里瓦桑、乔纳森·巴伦、丹·克莱因和泰勒·伯格·柯克帕特里克。字体中样式和结构的深度分解。在 EMNLP, 2019 年。1
- [24] 孙丹阳, 任同正, 李崇勋, 苏航, 和 朱军.通过阅读一些示例来学习书写程式化的汉字。在 IJCAI, 2018 年。3
- [25] 唐立成, 蔡一阳, 刘佳明, 洪志斌, 明- 龚明、范敏虎、韩俊宇、刘静拓、丁蕊瑞和婉吉。通过学习细 粒度的本地样式生成 Few-shot 字体。在 CVPR 中, 2022 年。3, 7, 8
- [26] Aaron van den Oord, Oriol Vinyals 和 Koray Kavukcuoglu。神经离散表示学习。 在 NeurIPS 中, 2017 年。4 [27] 向伟、龚伯青、刘子霞、陆伟和 李强 王。改善 wasserstein gans 的改进训练: 一致性项及其双重效 应。在 ICLR, 2018 年。1
- [28] Shan-Jean Wu, Chih-Yuan Yang 和 Jane Yung-jen Hsu。 书法: 风格与结构意识强的中国书法字符 Acter Generator 的 SarXiv 预印本 arXiv: 2005.12500, 2020 年。3
- [29] 谢阳辰、陈欣元、孙丽和卢悦。DG- font: 用于无监督字体生成的可变形生成网络。在 CVPR, 2021 年。7, 8
- [30] 徐松华, 江浩, 金涛, 刘志明, 和 潘云鹤。自动生成具有风格模仿的中国书法作品。IEEE 智能系 统, 24 (2) : 44–53, 2009 年。2
- [31] 徐松华, 金涛, 江浩, 和 F. Lau.自动 通过捕获个人笔迹的特征来生成个人中文笔迹。在 IAAI, 2009 年。2
- [32] 李媛, 陈云鹏, 王涛, 于伟浩, 石玉军, Francis E. H. Tay, Jiashi Feng 和 Shuicheng Yan。 tokensto-token vit: 在 imagenet 上从头开始训练视觉转 换器。在 ICCV, 2021 年。1
- [33] H. Zhang, I. Goodfellow, D. Metaxas, 和 A. Odena.自 我- 注音生成对抗网络 在 ICLR 2019 年 5 ICLR 研讨会.24 会议上呈现. man 和 Oliver Wang。深度特征作为感知指标的不合理有效 性。在 CVPR, 2018 年。6

在 CVPR [35] 中, Yexun Zhang、Ya Zhang 和 Wenbin Cai。分离风格

和内容进行广泛样式转移 在 CVPR 2018 年 21361 Anna Brian Kenji Iwana 和 Shengwu Xiong。通过深度特征相似性进行小样本文本样式传输。IEEE Trans. Image Process., 29: 6932–6946. 2020. 2 5

[37] 朱安娜, 尹占辉, 布莱恩·岩名健二, 周欣宇, 和 Shengwu Xiong。基于多因素解缠和混合的文本样式迁移。在 ACMM, 2022 年。2

[38] Anna Zhu, Qiyang Zhang, Xiongbo Lu, 和 Shengwu Xiong. 基于所选内容和引用样式嵌入的字符图像合成。在 ICME, 2019 年。1

[39] Jun-Yan Zhu, Taesung Park, Phillip Isola 和 Alexei A 埃夫罗斯。使用 cycleconsistent 对抗网络的未配对图像到图像转换。在 ICCV, 2017 年。2