Font Representation Learning via Paired-glyph Matching

Junho Cho [1,2]
junhocho@snu.ac.kr
Kyuewang Lee[1]
kyuewang@snu.ac.kr
Jin Young Choi[1]
jychoi@snu.ac.kr

[1] Department of ECE, ASRI, Seoul National University, Seoul, Korea
[2] Samsung Advanced Institute of Technology, Samsung Electronics, Suwon, Korea

Abstract

Fonts can convey profound meanings of words in various forms of glyphs. Without typography knowledge, manually selecting an appropriate font or designing a new font is a tedious and painful task. To allow users to explore vast font styles and create new font styles, font retrieval and font style transfer methods have been proposed. These tasks increase the need for learning high-quality font representations. Therefore, we propose a novel font representation learning scheme to embed font styles into the latent space. For the discriminative representation of a font from others, we propose a paired-glyph matching-based font representation learning model that attracts the representations of glyphs in the same font to one another, but pushes away those of other fonts. Through evaluations on font retrieval with query glyphs on new fonts, we show our font representation learning scheme achieves better generalization performance than the existing font representation learning techniques. Finally on the downstream font style transfer and generation tasks, we confirm the benefits of transfer learning with the proposed method.

1 Introduction

A font, which is a graphical representation of text, delivers certain visual feelings in multimedia through its matching style set of glyphs. Professional designers carefully choose fonts to convey their design intent. However, it is challenging to search for a specific font in the vast number of fonts available. Moreover, designing fonts requires typography knowledge, and aspiring designers can take months to learn typography. To cope with these difficulties, fonts should be easier to search for and create. There has been active research on font retrieval [4, 15, 16, 19, 24], font style transfer and generation [1, 11, 40, 42].

Font retrieval is a task that allows users to find similar looking fonts. Users can browse the fonts in the latent space to find the font they want. Through recognizing font style and generating new glyphs with the corresponding style, font style transfer and generation can ease the labor-intensive job of creating numerous glyphs with a certain font style. Font retrieval, style transfer and generation have historically focused on their own specific goals. However, if a powerful font representation learning method is devised, these tasks are considered downstream tasks, and performance gains can be expected through transfer learning [21]. There-

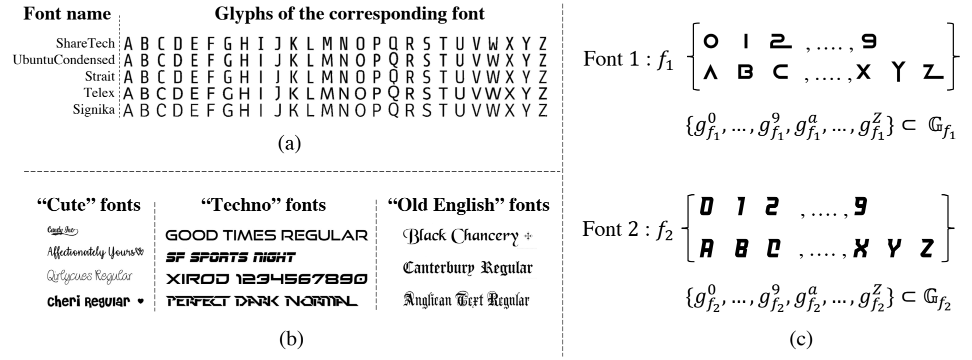


Figure 1: (a) Challenging fonts to distinguish. (b) Various fonts tagged as "Cute", "Techno" and "Old English", (c) Notations of glyphs g and glyph set $\mathbb{G}_f$ for font f

fore, we present a novel font representation learning scheme for the broader generalization on font-related downstream tasks. However, learning fonts is not as easy as one might think. Five fonts shown in Figure 1 (a), ShareTech, UbuntuCondensed, Strait, Telex, Signika are very difficult to distinguish with our eyes. Unlike general objects with textures [8], fonts have typographic elements (e.g., cap, x-height, serif, stem, stroke, descender, ascender, aperture) which are shape-based representations. Therefore, distinguishing these nuances is important for learning high-quality font representations.

In this paper, to mitigate the aforementioned difficulties, we approach how to learn these nuances through pairwise glyph similarity learning. More specifically, we try to learn the style representation of a font regardless of the shape of the character. That is, each font style keeps its unique nuance though the glyphs in the font have diverse shapes, which is referred to as *Glyph-font-consistency*. Paying attention to this unique nuance, we propose a new representation learning scheme to learn font features, keeping *Glyph-font-consistency* through a paired-glyph matching strategy. The proposed scheme attracts the font representations of glyphs in the same font to one another, but pushes away those of other fonts. We study generalization ability of our discriminative font representation learning scheme compared to existing font representation learning techniques. Finally, we evaluate performance improvement by transfer learning of our font representation learning scheme in the downstream font style transfer and generation tasks.

2 Related Works

2.1 Font Classification & Retrieval

Font classification and recognition models [3, 31, 32, 44, 45] are used to increase performance in text detection and recognition [2, 27], to make difficult calligraphy easier for users to recognize [25]. These methods of font classification only work with fixed sets of fonts, so they lack generalization to countless number of unseen fonts. Therefore, various retrieval-based methods [4, 15, 16, 19, 24] have been proposed for learning font representation and various related applications. Before the deep learning-based method appeared, Kataria *et al*. [16] extracted the SIFT (Scale-Invariant Feature Transform) [20] feature from each glyph of the font and defined the concatenation of glyphs as the font embedding. O'Donovan *et al*. [24] defined the attributes (e.g., artistic, attractive, pretentious) of fonts and used crowdsourced way to annotate the font attributes. And by learning a model to predict the attributes of fonts, O'Donovan *et al*. predicted attributes even for unseen fonts. However, specifying font attributes and determining their values is a rather subjective task, and the cost of annotations is very high, which limited annotations for small number of fonts. In light of this, tag-based font retrieval websites with relatively low annotation costs (e.g., `dafont.com`, `myfonts.com`, `10001fonts.com`) appeared.

These websites provide a tag-based font search service that allows users to select and download selected fonts. Figure 1 (b) shows how users can search for fonts based on a query (e.g., cute, techno, Old English). However, the tag-based font search has the disadvantage that, much like the problem with tag-based image searches, the tag does not sufficiently describe the font, and even appropriate tags may be subjective. With the advent of deep learning, some tag-based font retrieval studies [4, 15, 19] have tried to associate font tags to learn font representation in a data-driven manner. These studies proposed a method to perform tag classification [4] on fonts or to share the font latent space with the tag representation through Word2vec [15, 19, 22]. They investigated how the specific glyph shape of a font was related to a specific emotional font tag. However, these methods cannot learn font embedding without font tags.

2.2 Font Style Transfer & Font Generation

The necessity of font style transfer methods comes from the tedious and labor-intensive job of creating numerous glyphs with font style. For example, Chinese contains more than 60,000 characters and Korean contains 11,172 characters. Early font style transfer methods [10, 33, 39, 40] were based on image-to-image translation models [13, 23, 29] with the advance of generative adversarial networks [9]. These methods transferred the font style of one glyph image to another glyph image. These methods typically extracted font style features from glyph images for reference via a font style encoder model. Each method focused on the structural design of the font style encoder, because the style encoder needed to learn a good font representation so the font style was represented well in the output image. That is, better font representation learning was helpful for better quality font generation.

3 Methodology

3.1 Notations and Our Research Objective

To establish appropriate context, it is important to outline how we denote characters, glyphs and fonts. A character set is defined by a class of characters, for instance, $\mathbb{C}_{\text{0-9}} = \{0,1,2,...,9\}$, $\mathbb{C}_{\text{a-Z}} = \{a,b,c,...,X,Y,Z\}$ and $\mathbb{C}_{\text{0-Z}} = \{0,1,2,...,9,a,b,c,...,X,Y,Z\}$. A glyph is an image form of a character that has a specific style in a font. For example, if a glyph describes the character "Z" with a certain font f_1, we denote the glyph as $g^Z_{f_1}$. Figure 1 (c) shows that a font f_1 includes a matched set of glyphs for a character set $\mathbb{C}$. For example, the glyph set with font f_1 of character set $\mathbb{C}_{\text{0-Z}}$ is denoted by

$$\mathbb{G}^{\mathbb{C}_{\text{0-Z}}}_{f_1} = \{g^c_{f_1} | c \in \mathbb{C}_{\text{0-Z}}\} = \{g^0_{f_1}, g^1_{f_1}, g^2_{f_1}, ..., g^X_{f_1}, g^Y_{f_1}, g^Z_{f_1}\} \subset \mathbb{G}_{f_1}. \tag{1}$$

Denoting the set of all fonts in the world by $\mathbb{F}$, two different fonts $f_i, f_{j\neq i} \in \mathbb{F}$ convey different styles through two glyph sets ($\mathbb{G}_{f_i} = \{g^c_{f_i} | c \in \mathbb{C}\}$ and $\mathbb{G}_{f_j} = \{g^c_{f_j} | c \in \mathbb{C}\}$).

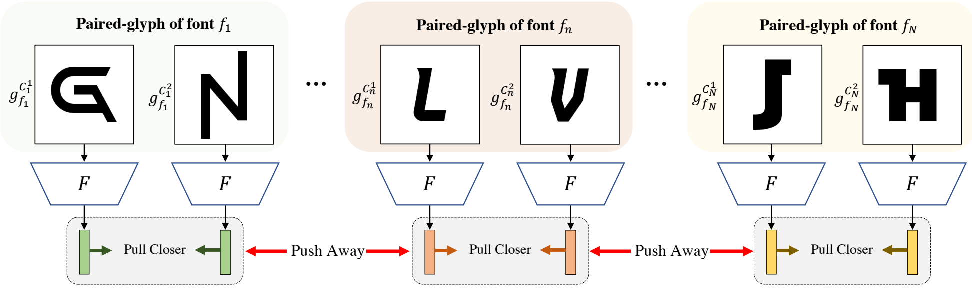


Figure 2: Overall scheme of Paired-glyph Matching learning

Based on the intrinsic relationship between fonts and glyphs, our research objective is to embed the fonts to representation space so that the glyphs in the same font are embedded into a small representation area far from those of the other fonts. To this end, we propose a *Paired-glyph Matching* learning scheme to pull the font representations of all glyphs in $\mathbb{G}_{f_i}$ closer to one another but push away from the font representations of the glyphs in the other glyph sets $\mathbb{G}_{f_{j\neq i}}$ and vice versa, as shown in Figure 2.

3.2 Paired-glyph Matching Learning

In *Paired-glyph Matching* learning, we randomly sample two fonts, f_1 and f_2, and two characters, c_1 and c_2. Then, we get a set of four glyphs $\{g_{f_t}^{c_i} | t = 1,2; i = 1,2\}$ expressing the font f_t for the character c_i. For the objective function to train F, we use cosine similarity given by $\texttt{sim}(u,v) = \frac{u^T v}{\|u\|\|v\|}$ as the dot product between L2 normalized u and v, where u,v are the font representations. We train the model F to map the glyphs from the same font into similar representations and those from different fonts into discriminative representations. That is, we maximize $\texttt{sim}(F(g_{f_t}^{c_1}), F(g_{f_t}^{c_2})), t = 1,2$ and minimize $\texttt{sim}(F(g_{f_1}^{c_i}), F(g_{f_2}^{c_i})), i = 1,2$. Glyphs of the same character look alike in the image space, even though their fonts are different from one another. However, the aforementioned objective drives the different font glyphs of the same character to be embedded far away from one another in the latent space. That is, we train the model F to focus on the font style of a glyph more than the shape of a character.

To generalize *Paired-glyph Matching* with a minibatch of N fonts, we randomly sample fonts $\{f_1, f_2, ..., f_N\}$ from the training set. We randomly sample two different glyph images for each font as $\{\{g_{f_n}^{C_n^1}, g_{f_n}^{C_n^2}\} | n = 1, \cdots, N\}$. That is, for all n in $1 \leq n \leq N$, there are N positive glyph pairs in the minibatch. Therefore for each glyph, remaining $2(N-1)$ glyphs are negative samples. Our model F maps every glyph images in the minibatch into font representation vectors in the latent space. The similarity of the embedding fonts for positive pairs and for negative pairs are defined by

$$\texttt{pos-sim}(f_n) = \exp\left(\texttt{sim}\left(F(g_{f_n}^{C_n^1}), F(g_{f_n}^{C_n^2})\right)/\tau\right), \tag{2}$$

$$\texttt{neg-sim}_k(f_n, f_{m\neq n}) = \sum_{l=1}^{2} \exp\left(\texttt{sim}\left(F(g_{f_n}^{C_n^k}), F(g_{f_m}^{C_m^l})\right)/\tau\right), \tag{3}$$

where τ is temperature scaling parameter. Then final loss $\mathcal{L}$ is sum of losses for each learning

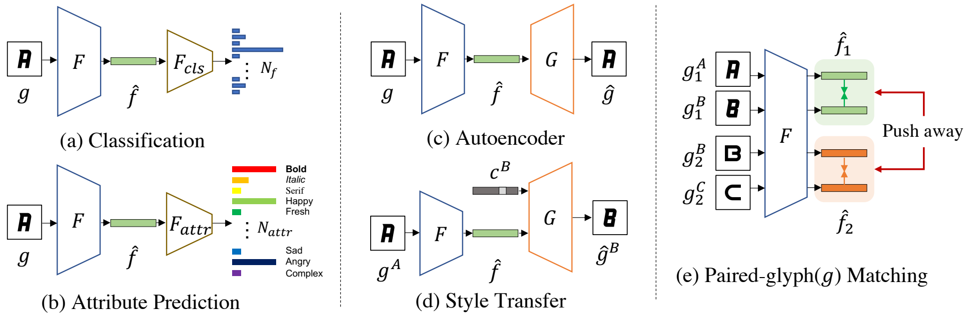


Figure 3: Font representation learning techniques. $\hat{f}$ is a font embedding in the latent space for each method.

font f_n is given by

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\left(-\sum_{k=1}^{2}\log\frac{\texttt{pos-sim}(f_n)}{\texttt{pos-sim}(f_n)+\sum_{m=1,m\neq n}^{N}\texttt{neg-sim}_k(f_n, f_{m\neq n})}\right). \tag{4}$$

The loss (4) is derived from "the normalized temperature-scaled cross entropy loss" [5].

4 Experiments

4.1 Baselines

Figure 3 shows baselines of font representation learning technique and our method *Paired-glyph Matching*. These methods all share font embedding network F as their backbone network. We consider the output of F from glyph g, $\hat{f} = F(g)$ as font embedding. Comparing font representation learning baselines (i.e., Classification [32], Style Transfer [40, 42], Autoencoder [30, 37], Attribute Prediction [4, 24] and Srivatsan *et al.* [28]) are more described in Section A of the supplementary material.

4.2 Datasets

O'Donovan *et al.* [24] dataset contains $1,088$ fonts for the training set ($\mathbb{F}_{\text{train}} = \{f_i | 1 \leq i \leq 1,088\}$) and 28 fonts for the validation set. Each font contains 62 alphanumeric characters ($\mathbb{C}_{\text{0-Z}}$). Thus, there are total $1,088 \times 62$ glyph images in training set. Among the fonts in the training set, each font in $\{f_i | 1 \leq i \leq 120\}$ is annotated by 37 attributes. Each attribute is described by a high-level expression, such as "dramatic" or "legible". Each attribute value ranges from 0 to 1. The attribute value vector of each font in $\{f_i | 1 \leq i \leq 120\}$ is denoted by $a_i \in \mathbb{A}$, where $\mathbb{A}$ is the attribute set, i.e., $\mathbb{A} = \{a_1, a_2, ..., a_{120}\}, \forall a_i \in [0,1]^{37}$. The remaining fonts of the training set (i.e., $\{f_i | 121 \leq i \leq 1,088\}$) are not annotated by any attributes.

Open Font Library (OFL), which is provided by Google Fonts[1], provides $1,076$ typefaces (font families). A typeface consists of several fonts that share a specific design. In this paper, we do not consider typeface, thus, fonts in a typeface are regarded as different fonts. For instance, the typeface "Bauer Bodoni" includes "regular", "bold", and "italic" fonts, which

[1] https://github.com/google/fonts

are considered different fonts in our work. Finally, we collected $3,802$ fonts for the alphanumeric character set ($\mathbb{C}_{\text{0-Z}}$). We randomly partitioned $3,702$ fonts for the training set and the remaining 100 fonts for the validation set. Since these fonts are provided in "ttf" and "otf" file formats, we converted each font file into 62 glyph images.

Capitals64 [1], which was used by Srivatsan *et al.* [28], contains capital letters ($\mathbb{C}_{\text{A-Z}}$). The dataset is split into train, validation, and test sets of $7,649$, $1,473$, and 1560 fonts, respectively. We used this dataset to compare our method with Srivatsan *et al.* method.

4.3 Implementation Details

Throughout all experiments, we used a single NVIDIA 2080ti or 1080ti gpu. We did not observe a performance boost by tuning the last dimension of the projection head, as the previous research [5]. Random sized crop augmentation was only used in our *Paired-glyph Matching* and *Attribute Prediction* as it degrades the retrieval mean accuracy of other baselines. The batch size was 64 samples for each font, and the image input size was 64×64. Bigger image size did not gain benefit on the retrieval mean accuracy score. We used glyphs representing $\mathbb{C}_{\text{a-Z}}$ for the O'Donovan and OFL datasets and $\mathbb{C}_{\text{A-Z}}$ for the Capitals64 dataset. We used the Adam [17] optimizer with a learning rate of 2e$-$4 for all models and datasets. We used ResNet18 [12] as the backbone network of font embedding network F for all models because other deeper neural network architectures were not effective. Font embedding was average pooled vector from output of the backbone network. The temperature scaling parameter τ of Equation 2 has been used as 0.1 for the OFL and Capitals64 dataset and 0.2 for the O'Donovan dataset.

Denoting feature dimension by `feat_dim`, we used $\texttt{feat\_dim} = 512$ for all models for training the O'Donovan dataset and $\texttt{feat\_dim} = 1,024$ for all models for the bigger OFL dataset. We used 5 transposed convolution layers and a last up-sample layer for the generator network G of *Autoencoder* and *Style Transfer* models to generate 64×64 dimensional images from font embedding vectors. The last 4 transposed convolutions were followed by self-attention modules [38, 43] and instance normalization [34]. The generator G of *Style Transfer* accepts ($\texttt{feat\_dim} + |\mathbb{C}|$)-dimensional vector, which is concatenation of font and one-hot character embedding. Denoting a fully connected layer of the weight matrix $d_1 \times d_2$ as $\mathrm{FC}^{(d_1 \times d_2)}$, the *Classification* head (F_{cls})is $\mathrm{FC}^{(\texttt{font\_dim} \times |\mathbb{F}_{\text{train}}|)}$, the *Attribute Prediction* head (F_{attr}) is $\mathrm{FC}^{(\texttt{font\_dim} \times 37)}$. Following previous research [5], we also used a projection head and L2 feature normalization on *Paired-glyph Matching*. The projection head is $\mathrm{FC}^{(\texttt{font\_dim} \times \texttt{font\_dim})} - \mathrm{ReLU} - \mathrm{FC}^{(\texttt{font\_dim} \times 70)}$. More details (e.g., the generator G architecture of *Style Transfer* and *Autoencoder*) are presented in Section B of the supplementary material. Codes are available at `https://github.com/junhocho/paired-glyph-matching`.

4.4 Experimental Results

To evaluate how well glyphs in a font are embedded in the latent space, we use the retrieval mean accuracy ($\underset{Ret}{\mathbf{MACC}}(\mathbb{C}_{\text{a-Z}})$) as described in Section C of the supplementary material.

4.4.1 Evaluation on Unseen Fonts (O'Donovan and OFL datasets)

Table 1 presents the performances on font embeddings $\hat{f}$ of all methods (i.e., *Paired-glyph Matching, Classification, Style Transfer, Autoencoder* and *Attribute Prediction*) depending on

Table 1: Performance evaluation ($\underset{Ret}{\texttt{MACC}}(\mathbb{C}_{\text{a-Z}})$) when trained on the O'Donovan dataset.

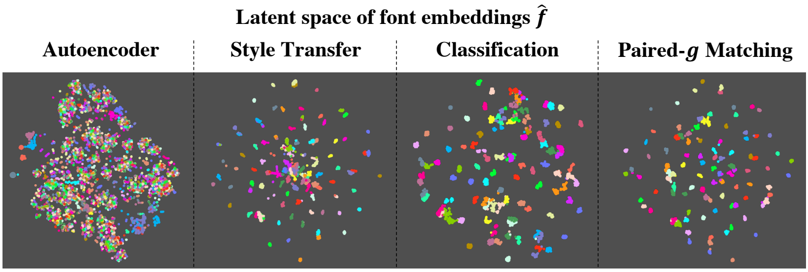
Methods	Data portion	O'Donovan $\underset{Ret}{\texttt{MACC}}(\mathbb{C}_{\text{a-Z}})$	OFL $\underset{Ret}{\texttt{MACC}}(\mathbb{C}_{\text{a-Z}})$
$\hat{f}$ of Paired-g Matching	$\{f_i \mid 1 \le i \le 1{,}088\} + \mathbb{A}$	**89.91**	**66.46**
	$\{f_i \mid 1 \le i \le 1{,}088\}$	**89.60**	**64.53**
	$\{f_i \mid 1 \le i \le 120\}$	**72.03**	**45.06**
$\hat{f}$ of Classification [32]	$\{f_i \mid 1 \le i \le 1{,}088\} + \mathbb{A}$	83.11	58.56
	$\{f_i \mid 1 \le i \le 1{,}088\}$	83.90	57.36
	$\{f_i \mid 1 \le i \le 120\}$	63.08	35.33
$\hat{f}$ of Style Transfer [40, 42]	$\{f_i \mid 1 \le i \le 1{,}088\} + \mathbb{A}$	76.71	36.71
	$\{f_i \mid 1 \le i \le 1{,}088\}$	71.84	36.88
	$\{f_i \mid 1 \le i \le 120\}$	65.07	30.00
$\hat{f}$ of Autoencoder [30, 37]	$\{f_i \mid 1 \le i \le 1{,}088\} + \mathbb{A}$	57.87	31.97
	$\{f_i \mid 1 \le i \le 1{,}088\}$	27.13	13.96
	$\{f_i \mid 1 \le i \le 120\}$	29.43	12.31
$\hat{f}$ of Attribute Pred. [4, 24]	$\{f_i \mid 1 \le i \le 120\} + \mathbb{A}$	64.08	38.02

Table 2: Performance evaluation ($\underset{Ret}{\texttt{MACC}}(\mathbb{C}_{\text{a-Z}})$) when trained on the OFL dataset. Paired-glyph Matching † and ‡ are each trained with different similarity-based losses [14, 26].

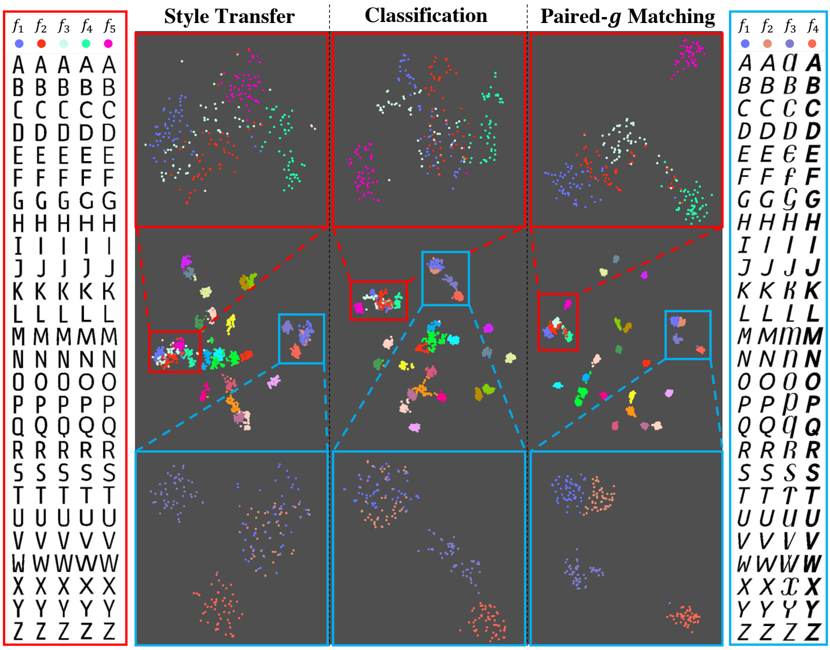
Methods	OFL valset	O'Donovan valset
$\hat{f}$ of Paired-glyph Matching	**91.82**	**75.44**
$\hat{f}$ of Classification [32]	83.67	68.48
$\hat{f}$ of Style Transfer [40, 42]	82.24	46.23
$\hat{f}$ of Autoencoder [30, 37]	15.55	26.66
$\hat{f}$ of Paired-glyph Matching †	88.93	72.98
$\hat{f}$ of Paired-glyph Matching ‡	82.70	54.28

training data portion in the O'Donovan dataset. For every 100 epochs until $15{,}000$ epochs, we evaluated the models on the O'Donovan validation set with the retrieval mean accuracy. Note that models had not seen fonts in the validation set. We found and reported the best score on the O'Donovan validation set and then evaluated the model with same weights on the OFL validation set. First of all, we compare when training only small portion ($\{f_i \mid 1 \le i \le 120\}$) of fonts in Table 1 and the performance was excellent in the order of *Paired-glyph Matching* (72.03), *Style Transfer* (65.07), *Attribute Prediction* (64.08), *Classification* (63.08), *Autoencoder* (29.43). It is notable that *Paired-glyph Matching* outperformed *Attribute Prediction* even without richer font annotations $\mathbb{A}$. To see the effectiveness of font attribute data $\mathbb{A}$, we jointly trained *Paired-glyph Matching, Classification, Style Transfer, Autoencoder* with *Attribute Prediction* (F_{attr} in Figure 3 (b)) and reported as $\{f_i \mid 1 \le i \le 120\} + \mathbb{A}$ in the data portion column of Table 1. We found training font attributes ($+\mathbb{A}$) to have no significant difference in *Paired-glyph Matching* and *Classification*. This indicates that font attribute data may not be worth the high annotation cost to train font representations.

Table 2 presents the performances of all font embedding methods trained on the OFL dataset. We found and reported the best score on the OFL validation set until $25{,}000$ epochs and then evaluated the model with same weights on the O'Donovan validation set. Since,



(a)



(b)

Figure 4: (a) Font latent space of (a) the OFL dataset and (b) the O'Donovan dataset annotated by font classes for each embedding methods. The Red and the cyan boxes respectively include font classes $\{f_1, f_2, f_3, f_4, f_5\}$ and $\{f_6, f_7, f_8, f_9\}$.

there are more possible solutions (e.g, triplet loss [26] or other self-supervised methods [6, 14, 35]) to learn similarities in paired-glyph matching learning, we include Paired Glyph Matching †, ‡ which are respectively trained with losses based on deep clustering algorithm (PICA) [14] and triplet loss [26]. We observed that paired-glyph matching learning with the loss (4) performed the best compare to other similarity learning approaches †, ‡ [14, 26].

To visually understand how comparing methods perform, we used T-SNE [36] projection on the font latent space as in Figure 4. From observations in the font latent space of the OFL dataset (Figure 4 (a)) and the O'Donovan dataset (Figure 4 (b)), glyphs in a font were better clustered in the order of *Paired-glyph Matching*, *Classification*, and *Style Transfer*. In particular, note the red and cyan boxes in Figure 4 (b). *Style Transfer* and *Classification* methods do not distinguish the glyphs of the fonts f_1, f_2, f_3 in the red box and f_6, f_7 in the cyan box, but our method distinguished them relatively well.

Table 3: Performance evaluation ($\underset{Ret}{\texttt{MACC}}(\mathbb{C}_{\text{A-Z}})$ and Font attribute prediction) on the Capitals64 dataset. All methods are trained on Capitals64, validated and tested on Capitals64. Font attribute prediction is evaluated on O'Donovan dataset with L_1-error.

Methods	Captials64 valset $\underset{Ret}{\texttt{MACC}}(\mathbb{C}_{\text{A-Z}})$	Captials64 testset $\underset{Ret}{\texttt{MACC}}(\mathbb{C}_{\text{A-Z}})$	O'Donovan L_1-error
$\hat{f}$ of Paired-g Matching	**61.38**	**62.66**	**0.09589**
$\hat{f}$ of Classification [32]	55.27	56.31	0.1275
$\hat{f}$ of Style Transfer [40, 42]	32.22	32.53	0.1217
$\hat{f}$ of Autoencoder [30, 37]	13.60	14.16	0.1312
$\hat{f}$ of Srivatsan *et al.* [28]	11.72	11.56	0.1097

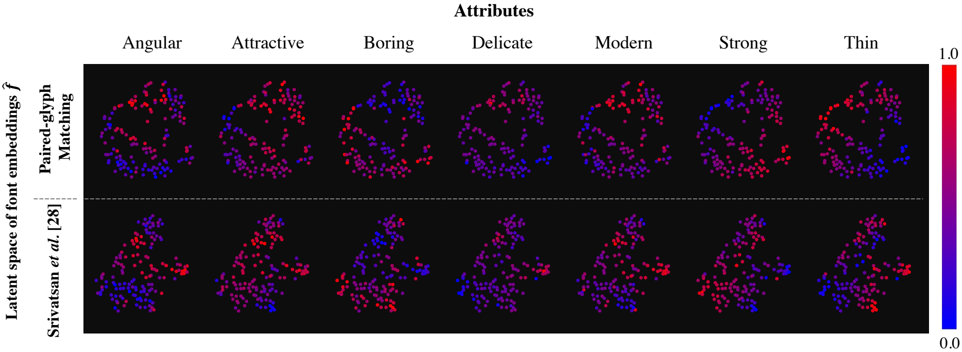


Figure 5: Font latent space of *Paired-glyph Matching* and Srivatsan *et al.* [28] annotated by font attributes (i.e., Angular, Attractive, Boring, Delicate, Modern, Strong, Thin). Both methods are trained on Capitals64 and tested on the O'Donovan dataset.

4.4.2 Evaluation on Unseen Fonts (Capitals64 dataset)

Table 3 presents the performances of font representation learning methods (i.e., *Paired-glyph Matching, Classification, Style Transfer, Autoencoder* and Srivatsan *et al.* [28]) on the Capitals64 dataset and the O'Donovan dataset. Similar to Table 1 and 2, our method performs the best in the retrieval mean accuracy $\underset{Ret}{\texttt{MACC}}(\mathbb{C}_{\text{A-Z}})$ measure. To more quantitatively evaluate representation power of $\hat{f}$, we trained font attribute ($\mathbb{A}$) prediction task, which is similar to linear evaluation protocol [5]. That is, we train a linear mapping from font embedding $\hat{f}$ of each method to 37 font attributes and validate with L_1-prediction error. We trained 120 fonts and validated 28 fonts in O'Donovan dataset, varying learning rate in range of $[1\text{e}{-6}, 1\text{e}{-5}, 1\text{e}{-4}, 1\text{e}{-3}, 1\text{e}{-2}]$ and reported the lowest L_1-error in Table 3 last column. Our method outperformed Srivatsan *et al.* by predicting font attributes with lower error.

In Figure 5, we observed the latent space of the O'Donovan fonts with attribute annotations $\mathbb{A}$. Refer to Srivatsan *et al.*, we took max-pooling operation on embeddings of glyphs in a font and regarded it as the font embedding. Each font in the O'Donovan dataset is colored with respective attribute value in Figure 5. Despite not training on font attribute data ($\mathbb{A}$), both methods gathered fonts according to values of the font attributes.

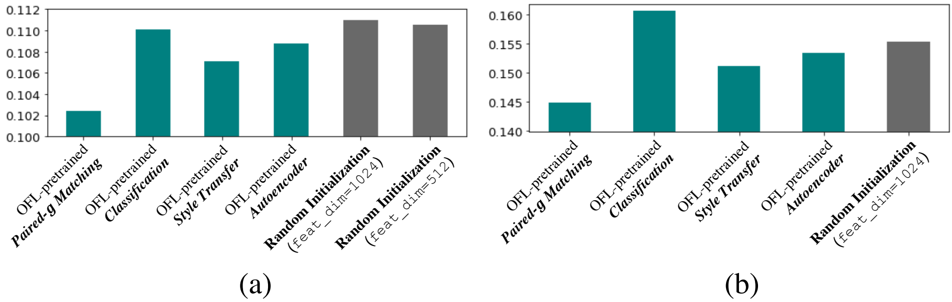


Figure 6: Image `L1-error` measured in (a) font style transfer and (b) font generation (Attr2Font [41]) with the OFL dataset pretrained and random initialization.

4.4.3 Transfer Learning to Font Style Transfer & Generation.

In this experiment, we checked the transfer learning performance in font style transfer (See Section A.4) and font generation (Attr2Font [41]) as downstream tasks. We used pretrained weights from the best-performing models F (i.e., *Paired-glyph Matching, Classification, Style Transfer* and *Autoencoder* from Table 2) on the OFL dataset and applied transfer learning to O'Donovan dataset, which is smaller the OFL dataset. To evaluate the generation quality of font style transfer model, we calculated average L_1 errors for all images generated from a input glyph and an one-hot character embedding as follows:

$$\texttt{L1-error} = \frac{1}{|\mathbb{F}_{\text{val}}| \times I_{\text{dim}}} \sum_{f \in \mathbb{F}_{\text{val}}} \sum_{\mathrm{C}_i, \mathrm{C}_j \in \mathbb{C}} \| G\left(F(g_f^{\mathrm{C}_i}), c^{\mathrm{C}_j}\right) - g_f^{\mathrm{C}_j}) \|_1,$$

where $I_{\text{dim}} = H \times W \times C$ is number of pixels in an image. For the Attr2Font model [41], which performs attribute-based font generation as a downstream task, we initialized the "style encoder" with the aforementioned pretrained weights, and `L1-error` is similarly defined. Note that we scratch-train the generator G weights of *Autoencoder* and *Style Transfer*. In Figures 6, we measured performance gains of pretrained models over random initialized baseline. Interestingly, the models trained in the generative way (i.e., *Autoencoder, Style Transfer*) on the OFL dataset seemed to be better in the downstream generative tasks than the model trained through *Classification*. As a result, we determined that *Paired-glyph Matching* performed the best, showing that our method can be useful as transfer learning to the generative tasks.

5 Conclusion

In this paper, we proposed a new discriminative font embedding method that attracts the representations of glyphs in the same font to one another but pushes away glyphs in other fonts. Our method needed neither a generator network nor font attribute tags because we actively take advantage of *Glyph-font-consistency*. Through extensive evaluation, we show our model outperformed the conventional representation learning techniques for generalization to unseen fonts. Finally, we confirmed the benefits of our method for transfer learning in the font style transfer and generation tasks.

Acknowledgement

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [NO.2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)] and IITP grant funded by Korea government(MSIT) [No.B0101-15-0266, Development of High Performance Visual BigData Discovery Platform for Large-Scale Realtime Data Analysis].

References

[1] Samaneh Azadi, Matthew Fisher, Vladimir Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. Multi-content gan for few-shot font style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 11, page 13, 2018.

[2] Henry S. Baird and George Nagy. Self-correcting 100-font classifier. In Luc M. Vincent and Theo Pavlidis, editors, *Document Recognition*, volume 2181, pages 106 – 115. International Society for Optics and Photonics, SPIE, 1994. doi: 10.1117/12.171098. URL https://doi.org/10.1117/12.171098.

[3] Sami Ben Moussa, Abderrazak Zahour, Abdellatif Benabdelhafid, and Adel Alimi. New features using fractal multi-dimensions for generalized arabic font recognition. *Pattern Recognition Letters*, 31:361–371, 04 2010. doi: 10.1016/j.patrec.2009.10.015.

[4] Tianlang Chen, Zhaowen Wang, N. Xu, Hailin Jin, and Jiebo Luo. Large-scale tag-based font retrieval with generative feature learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9115–9124, 2019.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[6] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. *Advances in neural information processing systems*, 27, 2014.

[7] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001. doi: 10.1214/aos/1013203451. URL https://doi.org/10.1214/aos/1013203451.

[8] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bygh9j09KX.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In

Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[10] Ammar Ul Hassan, Hammad Ahmed, and Jaeyoung Choi. Unpaired font family synthesis using conditional generative adversarial networks. *Knowl. Based Syst.*, 229:107304, 2021.

[11] Hideaki Hayashi, Kohtaro Abe, and Seiichi Uchida. Glyphgan: Style-consistent font generation based on generative adversarial networks. *ArXiv*, abs/1905.12502, 2019.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[14] Shaogang Gong Jiabo Huang and Xiatian Zhu. Deep semantic clustering by partition confidence maximisation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[15] Jihun Kang, Daichi Haraguchi, Akisato Kimura, and Seiichi Uchida. Shared latent space of font shapes and impressions. *arXiv preprint arXiv:2103.12347*, 2021.

[16] Saurabh Kataria, Luca Marchesotti, and Florent Perronnin. Font retrieval on a large scale: An experimental study. In *2010 IEEE International Conference on Image Processing*, pages 2177–2180, 2010. doi: 10.1109/ICIP.2010.5650155.

[17] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[19] Tugba Kulahcioglu and Gerard de Melo. *Fonts Like This but Happier: A New Way to Discover Fonts*, page 2973–2981. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450379885. URL https://doi.org/10.1145/3394171.3413534.

[20] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999. doi: 10.1109/ICCV.1999.790410.

[21] Pedro Marcelino. Transfer learning from pre-trained models. *Towards Data Science*, 10:23, 2018.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[23] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2642–2651. JMLR.org, 2017.

[24] Peter O'Donovan, Janis Libeks, Aseem Agarwala, and Aaron Hertzmann. Exploratory Font Selection Using Crowdsourced Attributes. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 33(4), 2014.

[25] Gao Pengcheng, Gu Gang, Wu Jiangqin, and Wei Baogang. Chinese calligraphic style representation for recognition. *Int. J. Doc. Anal. Recognit.*, 20(1):59–68, mar 2017. ISSN 1433-2833. doi: 10.1007/s10032-016-0277-z. URL https://doi.org/10.1007/s10032-016-0277-z.

[26] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[27] Hongwei Shi and T. Pavlidis. Font recognition and contextual processing for more accurate text recognition. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, volume 1, pages 39–44 vol.1, 1997. doi: 10.1109/ICDAR.1997.619810.

[28] Nikita Srivatsan, Jonathan Barron, Dan Klein, and Taylor Berg-Kirkpatrick. A deep factorization of style and structure in fonts. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2195–2205, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1225. URL https://aclanthology.org/D19-1225.

[29] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[30] Shancheng Tang, Puyue Zhang, Xiongxiong Chen, Hanbo Wang, and Ming Chen. A word representation method based on glyph of chinese character. In *2020 International Conference on Intelligent Transportation, Big Data Smart City (ICITBS)*, pages 954–957, 2020. doi: 10.1109/ICITBS49701.2020.00212.

[31] Dapeng Tao, Xu Lin, Lianwen Jin, and Xuelong Li. Principal component 2-d long short-term memory for font recognition on single chinese characters. *IEEE Transactions on Cybernetics*, 46(3):756–765, 2016. doi: 10.1109/TCYB.2015.2414920.

[32] Chris Tensmeyer, Daniel Saunders, and Tony Martinez. Convolutional neural networks for font classification. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 985–990, 2017. doi: 10.1109/ICDAR.2017.164.

[33] Yuchen Tian, Apr 2017. URL https://kaonashi-tyc.github.io/2017/04/06/zi2zi.html.

[34] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv:1607.08022 [cs]*, July 2016. URL http://arxiv.org/abs/1607.08022. arXiv: 1607.08022.

[35] Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv e-prints*, pages arXiv–1807, 2018.

[36] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[37] Chen Wang, Yani Zhu, Zhangyi Shen, Dong Wang, Guohua Wu, and Ye Yao. Font transfer based on parallel auto-encoder for glyph perturbation via strokes moving. In Yongxuan Lai, Tian Wang, Min Jiang, Guangquan Xu, Wei Liang, and Aniello Castiglione, editors, *Algorithms and Architectures for Parallel Processing*, pages 586–602, Cham, 2022. Springer International Publishing. ISBN 978-3-030-95388-1.

[38] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[39] Yankun Xi, Guoli Yan, Jing Hua, and Zichun Zhong. Jointfontgan: Joint geometry-content gan for font generation via few-shot learning. *Proceedings of the 28th ACM International Conference on Multimedia*, 2020.

[40] Yangchen Xie, Xinyuan Chen, Li Sun, and Yue Lu. Dg-font: Deformable generative networks for unsupervised font generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[41] Zhouhui Lian Yizhi Wang*, Yue Gao*. Attribute2font: Creating fonts you want from attributes. *ACM Trans. Graph.*, 2020.

[42] Yexun Zhang, Ya Zhang, and Wenbin Cai. Separating style and content for generalized style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, 2018.

[43] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, 2018.

[44] Yong Zhu, Tieniu Tan, and Yunhong Wang. Font recognition based on global texture analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10): 1192–1200, 2001. doi: 10.1109/34.954608.

[45] Abdelwahab Zramdini and Rolf Ingold. Optical font recognition using typographical features. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):877–882, aug 1998. ISSN 0162-8828. doi: 10.1109/34.709616. URL https://doi.org/10.1109/34.709616.