Large-scale Tag-based Font Retrieval with Generative Feature Learning

Tianlang Chen[1]*, Zhaowen Wang[2], Ning Xu[2], Hailin Jin[2] and Jiebo Luo[1]
[1]University of Rochester [2]Adobe Research
{tchen45, jluo}@cs.rochester.edu, {zhawang, nxu, hljin}@adobe.com

Abstract

Font selection is one of the most important steps in a design workflow. Traditional methods rely on ordered lists which require significant domain knowledge and are often difficult to use even for trained professionals. In this paper, we address the problem of large-scale tag-based font retrieval which aims to bring semantics to the font selection process and enable people without expert knowledge to use fonts effectively. We collect a large-scale font tagging dataset of high-quality professional fonts. The dataset contains nearly 20,000 fonts, 2,000 tags, and hundreds of thousands of font-tag relations. We propose a novel generative feature learning algorithm that leverages the unique characteristics of fonts. The key idea is that font images are synthetic and can therefore be controlled by the learning algorithm. We design an integrated rendering and learning process so that the visual feature from one image can be used to reconstruct another image with different text. The resulting feature captures important font design details while is robust to nuisance factors such as text. We propose a novel attention mechanism to re-weight the visual feature for joint visual-text modeling. We combine the feature and the attention mechanism in a novel recognition-retrieval model. Experimental results show that our method significantly outperforms the state-of-the-art for the important problem of large-scale tag-based font retrieval.

1. Introduction

Font is one of the most important elements in digital design. Designers carefully choose fonts to convey design ideas. Since digital fonts were invented in the 1950s, millions of fonts have been created to help designers create engaging and effective designs. For instance, MyFonts.com[1], one of the many font websites, offers over 130,000 fonts with diverse designs, such as script, handwritten, decorative, just to name a few categories. With the vast number of fonts available, font selection becomes a challenge.

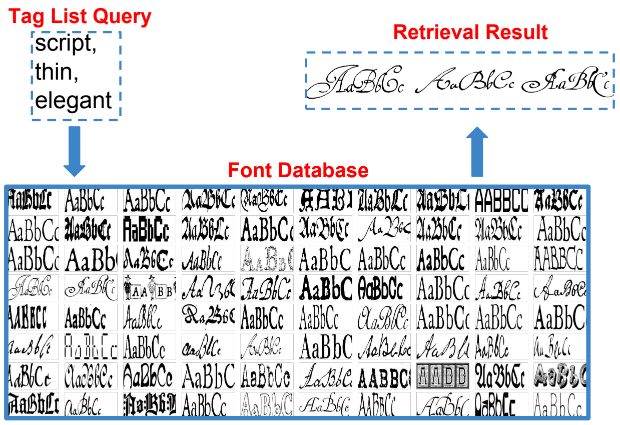

Figure 1. Overview of large-scale tag-based font retrieval.

Aspiring designers would take months to learn typography and font selection as part of their design curriculum. However, an average person who uses Microsoft Word will not have time to learn typography but may have an intuitive idea about what she wants to create. In this work, we study the problem of large-scale tag-based font retrieval (illustrated in Figure 1), which we believe is an important step toward developing intuitive font selection tools for average users.

Machine Learning approaches for tag-based font retrieval are first studied by O'Donovan et al. in [21] where they collect a tag-based font dataset and use Gradient Boosted Regression Trees [4]. However, this dataset only contains 1,278 fonts and 37 tags, which significantly limits its applicability. In this work, we are interested in a broad range of tags that a user can choose to express her design needs including properties, functionalities, attributes, to subjective descriptions, emotions, etc. To this end, we collect a large-scale font dataset from MyFonts.com, which contains nearly 20,000 high-quality professional fonts and 2,000 unique user tags. This dataset will be released as a benchmark dataset for tag-based font retrieval.

Feature plays a critical role in learning-based retrieval algorithms. The state-of-the-art image retrieval methods use convolutional neural networks to obtain image features which are typically pre-trained on large-scale image classification datasets such as ImageNet [11]. One may attempt to follow this paradigm for font retrieval. For instance, she can

*Work was done while Tianlang Chen was an Intern at Adobe.
[1]www.myfonts.com

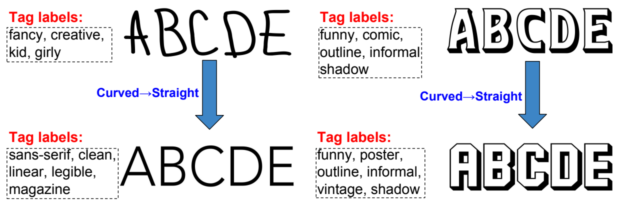
Tag labels:
fancy, creative, kid, girly
Tag labels:
funny, comic, outline, informal shadow
Curved→Straight
Curved→Straight
Tag labels:
sans-serif, clean, linear, legible, magazine
Tag labels:
funny, poster, outline, informal, vintage, shadow
ABCDE

Figure 2. An example to illustrate that a specific kind of visual feature can have different significance for tagging on different fonts.

use the DeepFont feature [27] which is trained to recognize fonts from text images. We find that although the DeepFont feature is learned to be agnostic to characters, traces of characters remain in the feature. For instance, the features of one string with two different fonts is much closer to the features of two different strings with the same font. Note that this problem may not be unique in fonts but is certainly much more severe in font retrieval than in image retrieval. The reason is that the appearance variability of text images is much larger than that of object images. In fact, there are an infinite number of text combinations. Another major problem in feature learning for tag-based font retrieval is that different parts of the font feature may have different significance for different tags in terms of tag prediction. As shown in Figure 2, the change in line type from curved to straight separates the upper two fonts from the bottom ones. At the same time, the left two have very different tags, while the right two have almost identical tags. This problem is usually addressed through the visual attention mechanism [31, 18, 30, 1, 13, 14]. However, the challenge is to learn effective attention models.

In this work, we propose a novel generative feature learning algorithm for font retrieval that addresses the first problem. The key idea is that font is synthetic and font images can be obtained through a process that can be controlled by the learning algorithm. This is a fundamental difference between real images and synthetic images. Real images cannot be controlled by the learning algorithm. We design the learning algorithm and the rendering process so that the font feature is agnostic to the characters. Specifically, we take one font and two characters and render three images. The first two images are rendered with the given font and the two characters respectively. The third image is rendered with the same character of the first image and a generic font. We train the network so that the feature extracted from the first image together with that of the third image can generate the second image using a combination of reconstruction and adversarial losses. This way, the feature from the first image has to preserve the font information but not the character information because it has to reconstruct the second image which has a different character. To address the second attention problem, we observe that there exists a strong correlation between font recognition and attention maps. Based on this observation, we design an implicit attention mechanism to help the model adaptively select useful information from the font recognition feature. In particular, we compute an attention map re-weights the feature as an attentive selection process.

1.1. Main Contributions

The main contributions of our paper are as follows: 1. We collect a large-scale font tagging dataset of high-quality professional fonts. The dataset contains nearly 20,000 fonts and 2,000 tags and a human curated evaluation set. 2. We propose a generative feature learning algorithm that leverages the unique property of font images. 3. We propose an attention mechanism to re-weight the learned feature for visual-text joint modeling. 4. We combine the two components in a recognition-retrieval model that achieves top performance for font retrieval.

2. Related Work

The combination of font understanding and machine learning starts from font recognition, where the problem is to recognize a font from a text image. Early font recognition works try to recognize a font via artificial font features. Zramdini et al. [39] identify the typeface, weight, slope, and size of a font image and use a multivariate Bayesian classifier to classify the font. Zhu et al. [38] leverage Gabor Filters to extract content-independent features for recognition. Chen et al. [3] feed local feature metric learning into a nearest class mean font classifier. As deep learning becomes popular, Wang et al. [27] build a Convolution Neural Network with domain adaptation techniques for font recognition, while Liu et al. [16] use a multi-task adversarial network to learn a disentangled representation and apply it to recognize Japanese fonts. In [17], Liu et al. leverage GAN to perform one-way transform from scene text to clean font image for better recognition. Compared with [17], we essentially perform font-specific mutual glyph transform as a bridge to connect the glyph-level input and the font-level tag. On the other hand, font generation and font style transfer has became hot spot topics in recent years. The work of Azadi et al. [2] successfully generates unobserved glyphs from restricted observed ones through an stacked conditional GAN model, Zhang et al. [34] propose an encoder-decoder EMD model which separates content and style factors for font style transfer. Following their work, we focus on tag-based font retrieval introduced in [21] and propose a deep learning approach. Compared with other attribute-based retrieval problems such as bird [25], scene [36], and animal [12], there are larger numbers of tags organized in complex semantic relationship, which significantly increases the difficulty of this problem. While most recent work [32, 23, 28] on the mentioned datasets focus on zero-shot image retrieval , we currently do not consider unseen tags for this task since there is no specialized knowledge base to describe font-related tags. But it is a problem of future interests.

Generative Adversarial Network (GAN) is porposed by Goodfellow et al. [6], which makes a generative model and a discriminative model play a minimax game to encourage the generative model to output desired synthetic images. Recently, various GAN structures are proposed that successfully achieve paired and unpaired image-to-image transformation [9, 37], and image generation conditioned on class labels [20]. The idea of GAN and adversarial learning are also applied on image retrieval task. Wang et al. [26] propose an ACMR method for cross modal retrieval, which implements adversarial learning to reduce the domain gap between the text feature and the image feature so that a shared embedding space is constructed. Gu et al. [7] achieve similar goal by directly integrating a GAN to generate corresponding images from text feature. The work of Zhang et al. [33] train an attention module and a hashing module in an adversarial way, which guides the attention module to focus on useful regions/words of an image/text.

Attention mechanism has been successfully applied to different visual-textual joint learning tasks such as image captioning [31, 18], visual question answering [30, 1], text-based image retrieval [13, 14] and semantic synthesis [19]. In most situations, attention mechanism directly guides the model to capture useful image regions or language segments, which can be visualized and considered as explicit attention. On the other hand, Kim et al. [10] propose an implicit attention mechanism, which generates the attention weight on feature maps instead of on raw data. Inspired by this work, we design an effective approach to select useful features adaptively for each input font image.

3. Large-scale Font Tagging Dataset

To the best of our knowledge, there exists no previous public large-scale dataset suitable for tag-based font retrieval. The only font-tag dataset available is the one collected by O'Donovan et al. [21]. It contains 1,278 Google web fonts and 37 attributes. To facilitate research in this direction, we collect a benchmark font tagging dataset from MyFonts.com. We select MyFonts for the following reasons: (1) Compared with free font resources such as Google fonts, the fonts on MyFonts are designed by well-known commercial font foundries which are in the typography business for decades and use by professional graphic designers. (2) MyFonts allows foundries, designers, and users to label fonts using arbitrary tags instead of selecting tags from a pre-defined set. This tremendously expands the tag vocabulary of the font retrieval system and enables semantic search of fonts. The fonts on MyFonts are originally labeled according to font families. Specifically, each font family contains a set of fonts of different design variations (e.g. regular, italic, and bold). A family is labeled by a list of

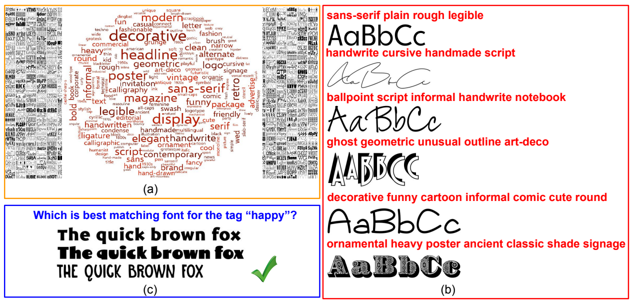
(a)
decorative
headline
display
sans-serif
poster
magazine
legible
script
modern
elegant
handwrite
(b)
sans-serif plain rough legible
AaBbCc
handwrite cursive handmade script
ballpoint script informal handwrite notebook
AaBbCc
ghost geometric unusual outline art-deco
AABBCC
decorative funny cartoon informal comic cute round
AaBbCc
ornamental heavy poster ancient classic shade signage
AaBbCc
(c)
Which is best matching font for the tag "happy"?
The quick brown fox
The quick brown fox
THE QUICK BROWN FOX

Figure 3. Overview of our large-scale font tagging dataset: (a) A word cloud of high-frequency tags; (b) Font samples with labeled tags; (c) A sample font group for AMT workers to rank.

associated tags. We find that under most conditions, a font family is labeled based on the regular version of the family. Therefore, for each font family, we use a set of pre-defined rules to find the regular version and render text images of the regular font as visual samples. In the end, we obtain in total 18,815 fonts and their corresponding labeled tag lists. We collect the complete Roman character glyph set (including both uppercase and lowercase of the 26 characters) for each font and render images of random character strings for all the fonts. We randomly split the entire dataset into training, validation, and test sets according to the proportion of 0.8/0.1/0.1.

As each font may be tagged by different foundries, designers, and users with varying quality, to guarantee the tag consistency and filter out the noisy labels, we sequentially apply the following tag pre-processing procedures: (1) convert all the words to lowercase and correct misspelling; (2) lemmatizing every word (e.g. kids $\rightarrow$ kid); (3) converting a N-Gram tag into a word with hyphens (e.g. sans serif $\rightarrow$ sans-serif); (4) combining the same tags based on the first three steps; and (5) removing infrequent tags occurring less than 10 times in the training set. In the end, we obtain 1,824 tags for our dataset. An overview of our collected dataset is shown as Figure 3.

It should be noted that the tag data from MyFonts only indicate whether a font matches a tag, but do not include the relative ranking of the matching. Moreover, tag labels collected from the web are inevitably noisy and incomplete. For the purpose of evaluating algorithms, we additionally collect a high-quality tagging set via Amazon Mechanical Turk (AMT). This additional tagging dataset contains the ranking information of different fonts for a given tag. Specifically, we choose the top-300 frequent tags of the MyFonts dataset. For each tag, we randomly generate 30 groups, each of which contains three test fonts that are already labeled with this tag. For each group of fonts, three AMT workers are requested to select the best matching font to the given tag after viewing a list of training fonts labeled with this tag as a reference. To avoid bias from text semantics, each font is represented by a font image rendered with a

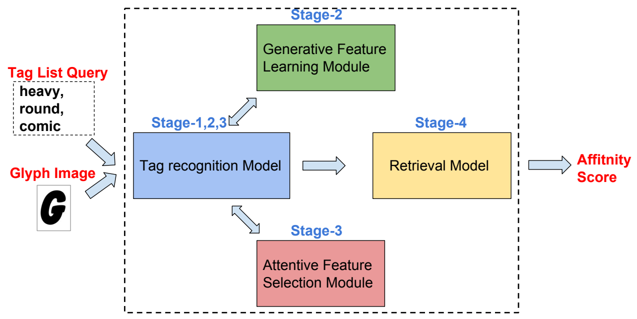

Figure 4. Overview of the proposed font retrieval system. We illustrate the stages where a specific module is trained.

standard text as in Figure 3. We add a group into the test set and label one of the fonts as the ground-truth only when all three workers select this font from this group. In the end, we collect 1,661 high confidence groups for the 300 tags, which we use as the evaluation test set in the experiments.

4. Our Method

In this paper, we follow the common setting of image retrieval tasks [29, 14, 26, 15]. We train a model to predict an affinity score between an input query and a a font as their similarity. The font is represented by 52 glyph images (a-b and A-B). The affinity score between a font and a query is averaged over all the 52 glyph images belonging to this font. When a user inputs a query, we compute the affinity scores of this query with all the fonts in the system and recommend the ones with the highest affinity scores. In contrast to the previous work [27] where each image contains multiple characters, we find the global structure of a text image (its character composition) does not provide extra information for the font retrieval problem. The overall architecture of the proposed model is illustrated in Figure 4, which is sequentially trained in four stages. In the following subsections we describe each stage in details.

4.1. Basic Tag Recognition Model

In the first stage, different from typical retrieval models [22, 26, 35] that directly learn a joint embedding for image and tag features, we train a tag recognition model which predicts the probability of an input glyph image with respect to each tag. Overall, this is a multi-label learning task, since one image may correspond to multiple tags. Specifically, let $\{F_1, ..., F_M\}$ be the training font set and $\{L_1, ..., L_{52}\}$ be the glyph set. We denote the text image containing glyph L_j of font F_i as I_i^j. Our basic tag recognition model first extracts a hidden feature f_i^j using a convolutional neural network. The feature is then fed into a fully-connected layer with N output nodes, where N is the total tag vocabulary size. In the end, a sigmoid unit maps the value of each node to the range of $(0, 1)$, which corresponds to the tag probability for the input image. We employ a cross-entropy

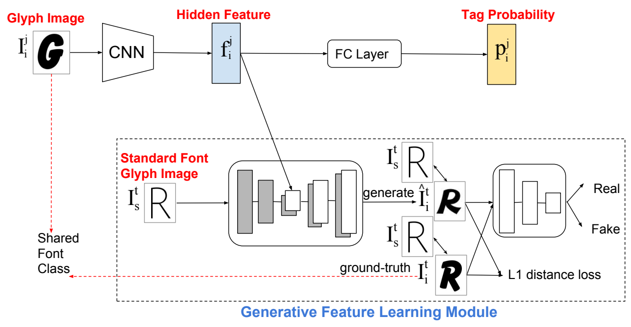

Figure 5. Generative Feature Learning: we transform input glyph image's character with a generative adversarial network.

loss to train the model:

$$L_c = \sum_{i,j} \sum_{k=1}^{N} (y_i^k \log(p_i^{j,k}) + (1-y_i^k) \log(1-p_i^{j,k})), \quad (1)$$

where $p_i^{j,k}$ is the predicted probability for I_i^j wrt the k^{th} tag; y_i^k is 1 if F_i is labeled with the k^{th} tag, and 0 otherwise.

4.2. Generative Feature Learning

After training the tag recognition model, in the second stage, we further encourage the extracted latent feature of a glyph image to better capture the information about the font instead of the character content. Otherwise, it may bias tag prediction with irrelevant information. Our idea is that we require a glyph image to be accurately generated via the latent feature of another glyph image with a different character but the same font. In particular, we design a conditional generative adversarial network [6] consisting of a generator and a discriminator as in Figure 5. Conditioned on the feature f_i^j of image I_i^j, the generator G is trained to convert an input image I_s^t rendered with a fixed standard font F_s and an arbitrary character L_t to a target image $\hat{I}_i^t$ of the same font style as I_i^j and same character as I_s^t (i.e. $\hat{I}_i^t$ = G(f_i^j, I_s^t)). The discriminator D is trained to discriminate between the generated image $\hat{I}_i^t$ and the real image I_i^t. The objective of the GAN is expressed as:

$$\begin{aligned} L_{gan} = \min_G \max_D (&\mathbb{E}_{I_i^t, I_s^t}[\log D(I_i^t, I_s^t)] + \\ &\mathbb{E}_{f_i^j, I_s^t}[\log(1 - D(G(f_i^j, I_s^t), I_s^t))]) \end{aligned} \quad (2)$$

Following [9], we design a PatchGAN architecture for the discriminator to enhance the local details of the generated image. Random noise is provided in the form of dropout. Compared with DR-GAN [24], we do not require the discriminator to explicitly recognize the character class of the generated glyph image, which is irrelevant in our case. To further encourage the generated image to have the same font style as the targeted one, we add an L_1 distance loss as [9]:

$$L_{L_1} = \mathbb{E}_{I_i^t, \hat{I}_i^t}(\|I_i^t - \hat{I}_i^t\|_1). \quad (3)$$

Our final objective of the GAN branch is:

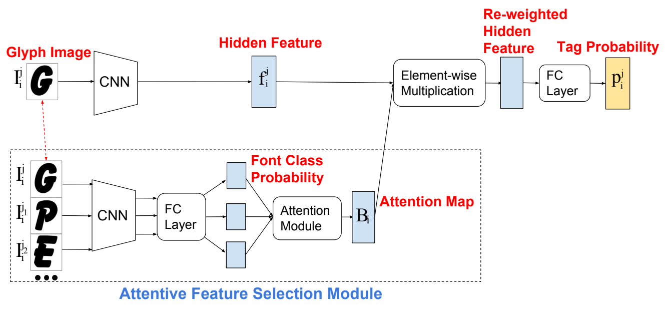

Figure 6. Attentive Feature Selection: an implicit attention mechanism adaptively selects useful features of the glyph image.

$$L_{lgan} = L_{gan} + \lambda \cdot L_{L_1} \tag{4}$$

The training objective of the recognition model is:

$$L_{rec} = L_c + \beta \cdot L_{lgan}, \tag{5}$$

where λ and β are hyper-parameters to adjust the relative weights among different loss terms. In this training stage, we first train GAN using Equation 4, and the features f_i^j are taken from the pre-trained tag recognition model without back-propagation. After convergence, the tag recognition model is fine-tuned by the ground-truth tags and GAN alternately. More concretely, we split each training epoch into two sub-epochs. In the first sub-epoch, we optimize the parameters of the CNN by $\beta \cdot L_{lgan}$ while fixing the last fully-connected layer. In the second sub-epoch, we optimize the parameters of the CNN and the fully-connected layer by L_c. This training strategy yields better performance than a multi-task training that jointly updates all the parameters.

4.3. Attentive Feature Selection

In the third stage, as motivated by the problem shown in Figure 2, our model is further guided to select relevant information based on the font class prediction of the input image. To this end, we train a font classification model to predict the font type of the input glyph image as in Figure 6. The structure of the font classification model is similar to that of the tag recognition model. It integrates a CNN to extract the hidden feature of the image and feeds it into a fully-connected layer with an output node size equal to the total number of font classes. Since one glyph image uniquely belongs to one font, this task is a single-label classification task. We thus replace the sigmoid unit with the softmax unit to map the outputs of the fully-connected layer, and train the font classification model using the cross-entropy loss. We feed the predicted font class probability distribution c_i^j for an input glyph I_i^j into an attention module which contains a fully-connected layer followed by a sigmoid unit. The distribution is transformed into an attention map B_i^j where B_i^j has the same dimension as the hidden feature f_i^j, and its values are in the range of $(0, 1)$.

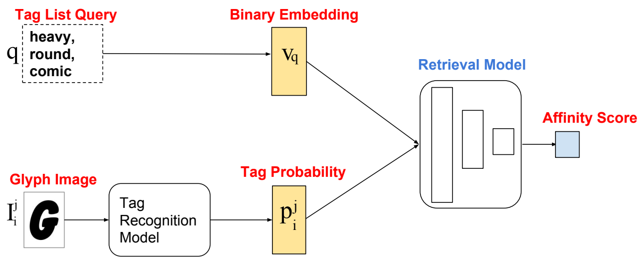

Figure 7. Combined recognition-retrieval model: the retrieval model is trained on top of the tag recognition model with query tag combinations.

The attention map estimated based on one image I_i^j may not be reliable. Therefore, during training, we aggregate the attention maps from a set of J images $\{I_i^{j_1}, ..., I_i^{j_J}\}$ of the same font F_i and randomly selected character set $L_{j_1}, ..., L_{j_J}$. The final attention map B_i for font F_i is computed as:

$$B_i = B_i^{j_1} \odot B_i^{j_2} \odot ... \odot B_i^{j_J}, \tag{6}$$

where $\odot$ represents the element-wise multiplication. This random selection of multiple glyph images improves the accuracy and selectivity of the attention map for a specific font. In the end, we take the element-wise multiplication between f_i^j and B_i to obtain the re-weighted feature of I_i^j, which is then fed into the top fully-connected layer of the tag recognition model. As in [10], our node-level feature re-weighting enforces an implicit attention mechanism. Figure 6 shows the overall structure of our tag prediction model with the attentive feature selection module integrated. When we train the model in this stage, the same training objective as Section 4.1 is employed to update only the parameters of the attention module and the last tag recognition fully-connected layer. Given a glyph image at test time, we only extract one attention map from this single image without further aggregation, leading to a significantly faster retrieval speed as well as competitive accuracy.

4.4. Combined Recognition-Retrieval Model

For a single-tag query, we define its affinity score to a glyph image as the predicted probability of the tag recognition model. Indeed, the model can also be used for retrieving fonts from multi-tag queries by computing the affinity score as the product or sum of the predicted probabilities of all the tags in the query. However, due to the imbalance of tag occurrence, the predicted probabilities of popular tags are usually much higher than those unpopular ones, leading to the tag dominance problem. The top recommended fonts may just match few tags of a multi-tag query.

Therefore, for multi-tag query, in the fourth stage, we introduce a retrieval model (as Figure 7) on top of the tag recognition model that maps the predicted tag probabilities to a comprehensive affinity score with unbiased consideration for each query tag. Given a pair of input image I_i^j and query q, the tag recognition model first predicts the tag probability distribution $p_i^j \in \mathbb{R}^N$. On the other hand, the query is encoded as a binary vector $v_q \in \mathbb{R}^N$, whose t-th entry is set to 1 if the query contains the t-th tag. The retrieval model takes the element-wise multiplication of p_i^j and v_q to generate a query-based tag probability vector, and feeds it into two-layer fully-connected network to get the affinity score between I_i^j and q. Before fed into the fully-connected layer, the query-based tag probability vector is first mapped by a power activation function $x \to (x+\epsilon)^\alpha$, where α is a hyper-parameter and ϵ is used to prevent infinite gradient. It shows better performance in transforming probabilities than other common activation functions. At the top of the second layer there has a sigmoid activation function which maps the output score in the range of $(0, 1)$. Following [29], we train the retrieval model as a relation network by sampling triplets of query, positive image and negative image. The query is composed of 2 to 5 randomly selected tags, all/not all of which are included in the ground-truth tags of the positive/negative image. We train the retrieval model by minimizing the following pairwise soft ranking loss:

$$L_{ret} = \mathbb{E}_{q,I^+,I^-}\left[\log(1+\exp(\gamma(s(q,I^-)-s(q,I^+))))\right], \tag{7}$$

where I^+ and I^- are positive and negative samples for query q, and $s(q, I)$ is the affinity score between q and I outputted by the retrieval model. The parameters of the tag recognition model are fixed in this stage.

5. Experiments

In this section, we compare our proposed method with state-of-the-art image retrieval methods adapted to our font retrieval dataset. An ablation study is presented to demonstrate the effectiveness of each component in our method.

5.1. Dataset and Experiment Settings

We train all our models on the MyFonts training set and evaluate them on the two test sets presented in Section 3, which are named MyFonts-test and AMT-test, respectively. MyFonts-test contains 1,877 fonts split from the original MyFonts dataset and is labeled with 1,684 different tags. We construct three query sets for evaluation. The first query set focuses on single-tag queries. It contains all the 1,684 tags with each tag considered as a query. The second query set focuses on multi-tag queries. For each font in MyFonts-test, we randomly generate 3 subsets of its ground-truth tags as 3 multi-tag queries. The query size ranges from 2 to 5 tags. After filter out repeating queries, this query set contains 5,466 multi-tag queries. The third query set focuses on testing the retrieval performance of the model on frequent tags that are more likely be searched. It contains the top-300 frequent tags in the training set with each tag considered as a query. For the first and third query sets, a font matches to a query (i.e. a positive font) if its ground-truth tag list contains the corresponding tag. For the second query set, a font matches to a query if its ground-truth tag list contains all the tags of the query. For each query set, we adopt two well-known retrieval metrics, *i.e.*, Mean Average Precision (mAP) and Normalized Discounted cumulative gain (nDCG) for model evaluation. Please refer to our supplementary material for a detailed description of the two metrics. AMT-test contains 1,661 groups, with each group associating a tag with one most relevant ground-truth font. Given a tag from a group, we compute and rank the affinity scores of all the fonts with respect to the tag for different models. We compare both the accuracy of each model selecting the ground-truth font in each group and the average rank of the ground-truth.

5.2. Implementation Details

For both tag recognition model and font classification model, we select ResNet-50 [8] as the base CNN architecture, where the hidden feature of an image is extracted from the pool5 layer of ResNet-50. We follow the pix2pix GAN [9] to build the generator as an encoder-decoder structure with skip connections. It receives a standard 128×128 glyph image with a fixed font F_s from its encoder, and concatenates the hidden feature computed by the tag recognition model before the bottleneck layer. For the discriminator, we apply a 14×14 local discriminator with three convolutional layers, to differentiate between real and fake local patches. We use hyper-parameters λ=10 in Equation 4, β=0.04 in Equation 5, J=4 in Section 4.3, and α=0.1, γ=100 in Section 4.4. When being trained in a particular stage, the learning rates of the CNN, the GAN, the last fully-connected layer, and the top retrieval model are set to 5×10^{-4}, 2×10^{-3}, 5×10^{-3} and 5×10^{-3}. We use batch size 20 in all the training experiments. Weight initialization details are provided in the supplementary material.

5.3. MyFonts Test

Since there are no prior works for large-scale font retrieval, we adapt and compare against several state-of-the-art image retrieval methods [15, 5, 29] for our task. The implementation details are as follows.

GNA-RNN [15]: the modified GNA-RNN model uses ResNet-50 [8] to extract the font image feature. Because there is no order among query tags, we use two fully-connected layers to extract the query feature from the binary encoding of the query input instead of using a RNN. After that, the image feature and query feature are mapped into the same embedding. Their inner product is further computed to obtain the affinity score. We use random positive

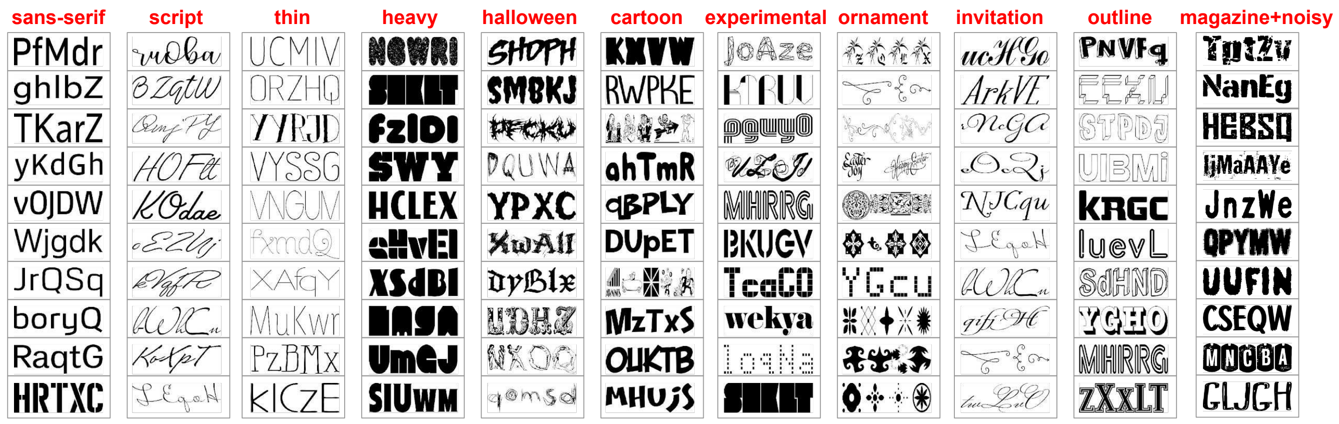

Figure 8. Retrieval results of our model on typical single-tag and multi-tag queries (font is represented by a random 5-character image).

Methods			Single tag (300)		Single tag (full)		Multi tag	
			mAP	NDCG	mAP	NDCG	mAP	NDCG
GNA-RNN [15]			14.90	56.97	5.03	28.41	7.01	27.49
DeViSE [5]			13.43	55.98	3.73	26.04	5.48	25.80
RelationNet [29]			15.33	57.49	5.66	29.27	7.52	28.05
Ours								
GAN	Att	Retr						
✗	✗	✗	26.29	68.67	16.77	42.63	14.93	35.52
✓	✗	✗	26.99	69.18	17.30	43.15	15.37	35.87
✗	✓	✗	27.75	69.81	17.76	43.65	15.78	36.40
✓	✓	✗	**28.08**	**70.04**	**18.02**	**43.95**	16.06	36.72
✓	✓	✓	**28.08**	**70.04**	**18.02**	**43.95**	**16.74**	**37.57**

Table 1. Comparison of different retrieval models on MyFonts-test set. "GAN" denotes our generative feature learning module. "Att" denotes our attentive feature selection module. "Retr" denotes our retrieval model. "✓" denotes with while "✗" denotes without. For a framework without the retrieval model, it computes the affinity score between a glyph image and a query as the product of the image's predicted probabilities of all the tags in the query.

and negative pairs of font images and query tags as training samples to train the model using a cross-entropy loss.

DeViSE [5]: Image features are extracted using the same ResNet-50 model. Tag embedding is trained from scratch because there is no appropriate text dataset with font related words. The method directly transforms an image and a query feature into an affinity score using a transformation matrix. Hinge ranking loss is applied during training.

RelationNet [29]: We use the same networks as in GNA-RNN to extract and map the font features and the tag features. The two features are concatenated and mapped to affinity scores using two fully-connected layers. Mean square error loss is employed during training.

The results of our method compared with other retrieval methods on the MyFonts-test set are presented in Table 1. It is clear that our method outperforms all the other methods by a large margin. Surprisingly, our basic tag recognition model described in Section 4.1 (the 4th result in Table 1) is already better than the other image retrieval based methods. We believe that the recognition based model is more suitable for the font retrieval task than the retrieval models

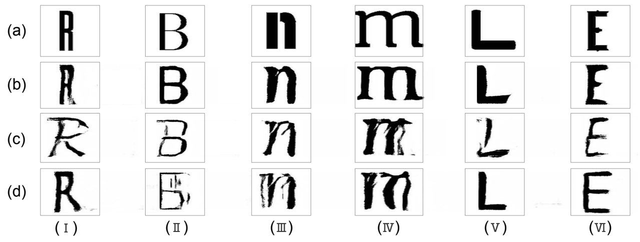

Figure 9. Comparison of the reconstructed glyph images using different feature nodes: (a) ground-truth (input) glyph, (b) reconstructed glyph from the top-100 feature nodes of the input's own attention map, (c) reconstructed glyph from the bottom-100 feature nodes of the input's own attention map, (d) reconstructed glyph from the top-100 feature nodes of the adjacent glyph's attention map ("I" ↔ "II", "III" ↔ "IV", "V" ↔ "VI"). Each column represents an input/reconstructed group.

which usually learn a joint embedding between the image feature and the tag feature. This can be attributed to the fact that font tags have very subtle meaning which is hard to capture by a joint embedding. In addition, the training data we collected from the web may not be complete. For example, some tags may be omitted by web users for a given font. In comparison to commonly used ranking loss and triplet loss for image retrieval, the multi-label cross-entropy loss can be considered as a separately training classifier for each tag, which is more robust to handle such annotation noise, and thus achieves better results.

The bottom part of Table 1 shows the ablation study of our method. With individual modules added one by one, our method gets steady improvements, which demonstrates the effectiveness of each component. The generative feature learning module and the attentive feature selection module capture font specific features and thus learn a better font feature representation, which leads to improved retrieval results. The retrieval model handles the imbalanced tag frequency in multi-tag queries, so that the top ranked fonts can match all the tags in the query. We show some font retrieval results based on our full model in Figure 8.

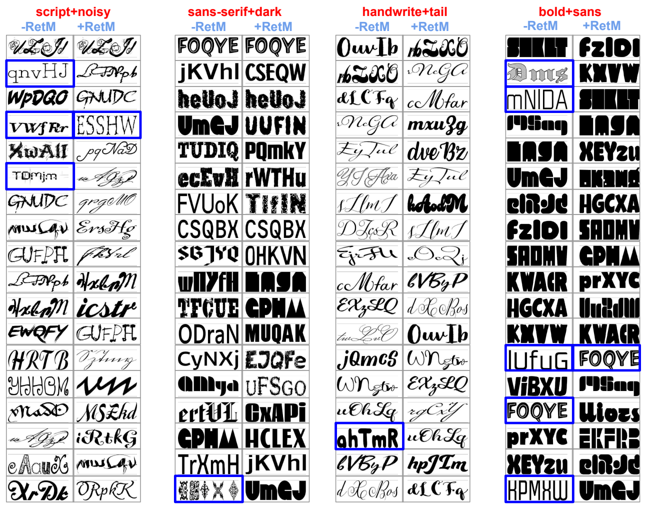

Figure 10. Comparison of the models with ("+RetM") and without ("-RetM") the retrieval model on typical queries. The obvious failure cases are outlined by blue boxes.

We show some qualitative results to further demonstrate the effectiveness of the proposed modules. We first verify that the attentive selection is effective in guiding the model to select useful and discriminated features for different fonts. Because the implicit attention mechanism is implemented at the node level, it is difficult to visualize as explicit attention mechanism. Therefore, we design a novel way for visualization. First, we train a GAN in the same way as Section 4.2 to reconstruct a glyph image from its extracted hidden feature. Given an input glyph image, we further restrict the GAN to reconstruct the glyph image by only using its top-100/bottom-100 hidden feature nodes with the highest/lowest attention weights from this attention map. We manually set the rest nodes to 0. For the last row in Figure 9, we reconstruct each glyph image using its 100 feature nodes correspond to the highest attention weights on the attention map of another glyph image. From Figure 9, we find that a glyph image reconstructed by the top-100 nodes based on its own attention map is more similar to the original input image. This serves our motivation for attentive feature selection, which is to adaptively select features that have effect on the input glyph image's tags and filter out the useless ones that cause error. In addition, the cross-attention generation results in the last row indicate that the model pays attention to different features for different fonts, instead of simply focusing on a uniform set of features.

Finally, Figure 10 shows the different multi-tag retrieval results obtained with and without the retrieval model. Given a multi-tag query including a frequent tag "script" and an infrequent tag "noisy", our retrieval model can retrieve fonts which match both tags. While without the retrieval model, our retrieval results are less satisfactory, it recommends some fonts that don't have attribute "script" because of the tag dominance problem.

Methods	Accuracy	Ave. Rank
Ours-Basic	44.49	1.81
Ours	**47.50**	**1.75**

Table 2. Comparison of our basic tag recognition model and full model on the AMT-test set.

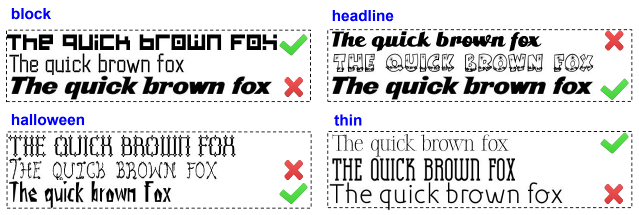

Figure 11. Qualitative comparison of our basic recognition model and full model on typical groups of AMT-test. The fonts with "✗" are falsely predicted by the basic recognition model, the ground-truth fonts with "✓" are correctly predicted by the full model.

5.4. AMT Test

Table 1 shows that our basic recognition model is a strong baseline and much better than previous retrieval methods. We only compare our full model with the basic recognition model on the AMT-test set. Overall, the task on AMT-test is difficult because all fonts in the group of a tag is originally labeled with the tag. The model needs to select the fonts that are more relevant. As shown in Table 2, our full model achieves better performance than the basic version. It indicates that for the top-related fonts toward one tag, the ranking of the full model is still more consistent with the human judgment. Some qualitative results are illustrated in Figure 11.

6. Conclusion

In this paper, we study the problem of tag-based font retrieval. We collect a large-scale font tagging dataset. We propose a joint recognition-retrieval model. In particular, a recognition model is first trained to predict the tag probabilities of a font. With the guidance of the generative feature learning and attentive feature selection mechanisms, the model adaptively selects the general and significant information of the font and makes better tag probability prediction. A retrieval model is further integrated to map tag probabilities to font-query affinity scores. Extensive qualitative and quantitative evaluations validate the effectiveness of our model for font retrieval.

7. Acknowledgment

This work is partially supported by NSF award #1704337.

References

[1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and vqa. *arXiv preprint arXiv:1707.07998*, 2017.

[2] 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.

[3] Guang Chen, Jianchao Yang, Hailin Jin, Jonathan Brandt, Eli Shechtman, Aseem Agarwala, and Tony X Han. Large-scale visual font recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3598–3605, 2014.

[4] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[5] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[7] Jiuxiang Gu, Jianfei Cai, Shafiq Joty, Li Niu, and Gang Wang. Look, imagine and match: Improving textual-visual cross-modal retrieval with generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7181–7189, 2018.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[10] Jin-Hwa Kim, Sang-Woo Lee, Donghyun Kwak, Min-Oh Heo, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Multimodal residual learning for visual qa. In *Advances in Neural Information Processing Systems*, pages 361–369, 2016.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[12] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 951–958. IEEE, 2009.

[13] Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. Stacked cross attention for image-text matching. *arXiv preprint arXiv:1803.08024*, 2018.

[14] Shuang Li, Tong Xiao, Hongsheng Li, Wei Yang, and Xiaogang Wang. Identity-aware textual-visual matching with latent co-attention. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 1908–1917. IEEE, 2017.

[15] Shuang Li, Tong Xiao, Hongsheng Li, Bolei Zhou, Dayu Yue, and Xiaogang Wang. Person search with natural language description. *arXiv preprint arXiv:1702.05729*, 2017.

[16] Yang Liu, Zhaowen Wang, Hailin Jin, and Ian Wassell. Multi-task adversarial network for disentangled feature learning. In *CVPR*, 2018.

[17] Yang Liu, Zhaowen Wang, Hailin Jin, and Ian Wassell. Synthetically supervised feature learning for scene text recognition. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[18] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 6, page 2, 2017.

[19] Shuang Ma, Jianlong Fu, Chang Wen Chen, and Tao Mei. Da-gan: Instance-level image translation by deep attention generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5657–5666, 2018.

[20] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[21] Peter O'Donovan, Jānis Lībeks, Aseem Agarwala, and Aaron Hertzmann. Exploratory font selection using crowd-sourced attributes. *ACM Transactions on Graphics (TOG)*, 33(4):92, 2014.

[22] Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–58, 2016.

[23] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013.

[24] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1415–1424, 2017.

[25] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[26] Bokun Wang, Yang Yang, Xing Xu, Alan Hanjalic, and Heng Tao Shen. Adversarial cross-modal retrieval. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 154–162. ACM, 2017.

[27] Zhangyang Wang, Jianchao Yang, Hailin Jin, Eli Shechtman, Aseem Agarwala, Jonathan Brandt, and Thomas S Huang. Deepfont: Identify your font from an image. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 451–459. ACM, 2015.

[28] Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning-the good, the bad and the ugly. *arXiv preprint arXiv:1703.04394*, 2017.

[29] Flood Sung Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. 2018.

[30] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–29, 2016.

[31] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659, 2016.

[32] Li Zhang, Tao Xiang, Shaogang Gong, et al. Learning a deep embedding model for zero-shot learning. 2017.

[33] Xi Zhang, Hanjiang Lai, and Jiashi Feng. Attention-aware deep adversarial hashing for cross-modal retrieval. In *European Conference on Computer Vision*, pages 614–629. Springer, 2018.

[34] 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.

[35] Zhedong Zheng, Liang Zheng, Michael Garrett, Yi Yang, and Yi-Dong Shen. Dual-path convolutional image-text embedding. *arXiv preprint arXiv:1711.05535*, 2017.

[36] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2018.

[37] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*, 2017.

[38] 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.

[39] Abdelwahab Zramdini and Rolf Ingold. Optical font recognition using typographical features. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8):877–882, 1998.