Large-Scale Visual Font Recognition

Guang Chen* Jianchao Yang† Hailin Jin† Jonathan Brandt† Eli Shechtman†
Aseem Agarwala† Tony X. Han*
†Adobe Research
San Jose, CA, USA
{jiayang, hljin, jbrandt, elishe, asagarwa}@adobe.com
*University of Missouri
Columbia, MO, USA
gc244@mail.missouri.edu hantx@missouri.edu

Abstract

This paper addresses the large-scale visual font recognition (VFR) problem, which aims at automatic identification of the typeface, weight, and slope of the text in an image or photo without any knowledge of content. Although visual font recognition has many practical applications, it has largely been neglected by the vision community. To address the VFR problem, we construct a large-scale dataset containing 2,420 font classes, which easily exceeds the scale of most image categorization datasets in computer vision. As font recognition is inherently dynamic and open-ended, i.e.*, new classes and data for existing categories are constantly added to the database over time, we propose a scalable solution based on the nearest class mean classifier (NCM). The core algorithm is built on local feature embedding, local feature metric learning and max-margin template selection, which is naturally amenable to NCM and thus to such open-ended classification problems. The new algorithm can generalize to new classes and new data at little added cost. Extensive experiments demonstrate that our approach is very effective on our synthetic test images, and achieves promising results on real world test images.*

1. Introduction

Typography is a core design element of any printed or displayed text; graphic designers are keenly interested in fonts, both in their own works, as well as in those of others. Consequently, they frequently encounter the problem of identifying "fonts in the wild." For example, a designer might spot a particularly interesting font on a restaurant menu and would like to identify it for later use. Currently, a designer's best recourse is to take a photo of the text and then seek out an expert to identify the font, such as on an online typography forum[1].

With hundreds of thousands of possible fonts to choose from, it is extremely tedious and error-prone, even for font experts, to identify a particular font from an image manually. Effective automatic font identification could therefore greatly ease this problem, and could also facilitate font organization and selection during the design process. To address this need, we propose the application of computer vision methods to automatically identify the typeface, weight and slope of the text in a photo or graphic image. We dub this problem Visual Font Recognition (VFR), in contrast to the well-studied problem of Optical Character Recognition (OCR).[2]

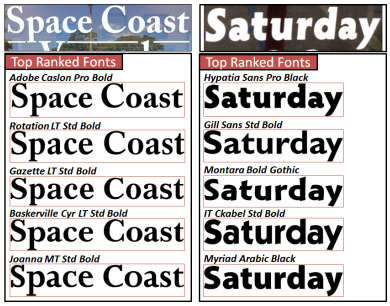


Figure 1. Visual font recognition on two real-world test images. The algorithm correctly classifies both (top of the list) and returns four other similar typefaces, despite the high level clutter and noise.

Remarkably, the computer vision research community has largely neglected the VFR problem. The few previous approaches [8, 24, 11, 17, 15, 16, 2] are mostly from the document analysis standpoint, focusing on small number of font classes on scanned document images that typically have high quality, in terms of resolution and contrast, and low geometric distortion. Consequently, tasks such as binarization, character segmentation, connected component analysis, geometric alignment, and OCR can be robustly applied. Building on these image processing steps, global texture analysis [24, 15], high-order statistical features [2] and typographical features [17, 8] have been exploited for

[1] E.g. myfonts.com or www.flickr.com/groups/type

[2] We note that the two problems are coupled in that accurate VFR has the potential to greatly improve OCR accuracy, and vice versa.

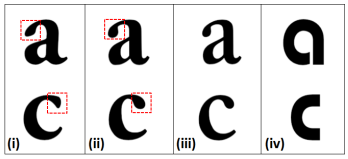


Figure 2. Example images of character a and c in different fonts: (i) Adobe Garamond Pro Bold, (ii) Adobe Calson Pro Bold, (iii) Adobe Calson Pro SemiBold, and (iv) BauhausStd-Demi.

font recognition. However, such scanned image-based techniques are less applicable to our VFR problem, which needs to be effective even on very short strings or single words from noisy web images or photos taken with mobile devices. In particular, photos in the wild suffer from noise, blur, perspective distortions, as well as complex interactions of content with background.

Even though there has been great progress in OCR from photos [4], OCR is generally designed for standard body text fonts and is not very effective for unusual fonts which are often the ones designers wish to identify. In our own experiments with state-of-the-art publicly available OCR engine [7] on our VFR-2420 dataset, which contains 2420 font classes each with 1000 clean English word images, whole words were correctly recognized only 60% of the time. Furthermore, words from more than 13% of our font classes were correctly recognized less than 30% of the time. The poor performance of OCR on wider range of fonts will adversely affect character-based font recognition algorithms. Therefore, current systems such as whatfontis.com, rely on humans interactions to solve the character segmentation and recognition problem for font recognition on photos.

Even with human intervention, these systems are still not robust enough for practical use. Equipped with recent advances in machine learning and image categorization [23, 18, 1, 22, 21, 12], we propose to develop an automatic algorithm that does not rely on character segmentation or OCR, and therefore, is applicable to the range of inputs needed for VFR. However, compared with previously studied image categorization and fine-grained recognition problems, visual font recognition has the following additional challenges:

- VFR is an extremely large-scale recognition problem. For instance, myfonts.com alone claims that they have more than 100,000 fonts in their collection, which is much larger than most image recognition problems the vision community has so far addressed. This is also dramatically different from previous font recognition works, where, limited to the scope of scanned documents, they have only investigated the problem on a scale of tens of fonts.
- VFR is inherently dynamic and open-ended in that new font classes need to be continually added to the system. It is therefore critical that the algorithm is able to adapt to new classes with low added cost, and to scale elegantly with the number of classes.
- VFR is a combination of super fine-grained recognition and coarse-grained image categorization, and it is character-dependent. For example, consider Fig. 2, where fonts (i) and (ii) only differ slightly at the letter endings (in red rectangles), (ii) and (iii) only differ on font weight, while all three fonts (i), (ii) and (iii) are visually distinct from font (iv). However, in all cases, without knowing the characters, it is very hard to find a feature space where character a is closer to c from the same font than to character a from another font.

In this paper, we develop a scalable data-driven approach to VFR that does not depend on character segmentation or OCR to address the above challenges. Therefore, our algorithm does not need to know the context of the word when recognizing the font. Specifically, our contributions are as follows:

1. Inspired by the recent object recognition and fine-grained recognition work [18, 22, 21], we propose an image feature representation called local feature embedding (LFE) that captures salient visual properties to address the simultaneous fine-grained and coarse-grained aspects of VFR.
2. Adopting the nearest class mean (NCM) classifier, we build a scalable recognition algorithm with metric learning and max margin template selection based on the LFE representation. Similar to [12], the new algorithm can generalize to new classes at very little added cost.
3. We have synthesized a large-scale dataset for visual font recognition, consisting of 2,420 font classes each with 1,000 English word images. We also collected a small test set of real world images, each with a known font class label in the training set.

Fig. 1 shows our visual font recognition on two real world images, where our algorithm correctly classifies both (top of the list) and returns four similar other typefaces, even though the text images have high levels of clutter and noise, and some perspective distortion (right).

2. A Database for Visual Font Recognition

In this work, we focus on visual font recognition for the Roman alphabet. As the most common case for font recognition is to identify the font class of short texts in images, we want to collect images that contain short strings or single English words as the training data. However, collecting real world examples for a large collection of font classes turns out to be extremely difficult because many attainable real world text images do not have font label information. Furthermore, the error-prone font labeling task requires font

expertise that is out of reach of most people. Instead, we turn to synthesizing these images for the given fonts to obtain the training data.

In order to get a representative English word set, we randomly select 1,000 English words from the most common 5,000 English words sampled from a large corpus. The English words have variable lengths, resulting in word images of different sizes. To capture the variations caused by letter cases, we randomly divide the selected 1,000 English words into lower and uppercases with equal probability. We collect in total 447 typefaces, each with different number of variations resulting from combinations of different styles, *e.g.*, regular, semibold, bold, black, and italic, leading to 2,420 font classes in the end.

For each font class, we generate one image per English word, which gives 2.42 million synthetic images for the whole dataset. To normalize the text size[3], we add two lower case letters "fg" in front of each word when synthesizing the image. This helps us to find the ascender and descender lines of the text. We then normalize the image size by fixing the distance between the ascender line and descender line. The two letters "fg" are then removed from the synthesized images. After normalization, we obtain all word images with the same height of 105 pixels.

Besides the synthetic data, we also collected 325 real world test images for the font classes we have in the training set These images were collected from typography forums, such as myfonts.com, where people post these images seeking help from experts to identify the fonts. Compared with the synthetic data, these images typically have much larger appearance variations caused by scale, background, lighting, noise, perspective distortions, and compression artifacts. We manually cropped the texts from these images with a bounding box to normalize the text size approximately to the same scale as the synthetic data. Figure 6 in the experiment section shows some real-world test images from different font classes.

3. Our Approach

In this section, we first present our image feature representation for visual font recognition, and then we describe our large-scale classification algorithm based on local feature metric learning and max-margin template selection.

3.1. Local Feature Embedding

Most of the current state-of-the-art generic image classification systems [9, 20, 18] follow the pipeline of first encoding the local image descriptors (*e.g.*, SIFT [10] or LBP [1]) into sparse codes, and then pooling the sparse codes into a fixed-length image feature representation. With each image represented as a collection of local image descriptors $\{\mathbf{x}_i\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$, the first coding step encodes each local descriptor into some code (typically sparse),

$$\mathbf{y}_i = f(\mathbf{x}_i, T), \tag{1}$$

where $T = \{\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_K\}$ denotes the template model or codebook of size K and $\mathbf{t}_i \in \mathbb{R}^d$, f is the encoding function (*e.g.*, vector quantization [9], soft assignment [13], LLC [18], or sparse coding [20]), and $\mathbf{y}_i \in \mathbb{R}^K$ is the code for $\mathbf{x}_i$. Then the pooling step obtains the final image representation by

$$\mathbf{z} = g(\{\mathbf{y}_i\}_{i=1}^n), \tag{2}$$

where g is a pooling function that computes some statistics from each dimension of the set of vectors $\{\mathbf{y}_i\}_{i=1}^n$ (*e.g.*, average pooling [9], max pooling [20]), and $\mathbf{z} \in \mathbb{R}^K$ is the pooled feature vector that will later be fed into a classifier.

While the above feature extraction pipeline is effective at distinguishing different categories of objects, it is not sufficient to capture the subtle differences within a object category for fine-grained recognition, *e.g.*, the letter endings in Fig. 2. Inspired by the recent fine-grained recognition work [6, 22, 21], we extend the above feature extraction pipeline by embedding the local features into the pooling vector, in order to preserve the details of the local letter parts. Specifically, using max pooling in Eqn. (2), we not only pool the maximum sparse coefficients, but also record the indices of these max pooling coefficients:

$$\{\mathbf{z}, \mathbf{e}\} = \max(\{\mathbf{y}_i\}_{i=1}^n), \tag{3}$$

where $\mathbf{z}$ contains the max coefficients pooled from each dimension of the set $\{\mathbf{y}_i\}_{i=1}^n$ and $\mathbf{e}$ is its index vector. Denoting $e_k = \mathbf{e}(k)$ and $z_k = \mathbf{z}(k)$, it is easy to see that $z_k = \mathbf{y}_{e_k}(k)$. Instead of using the max pooling coefficients as the final image feature representation [18], we obtain the pooling coefficients together with the local descriptor that fires each of them $\{z_k, \mathbf{x}_{e_k}\}_{k=1}^K$. We construct the final feature representation by concatenating these local descriptors weighted by their pooling coefficients:

$$\mathbf{f} = [z_1\mathbf{x}_{e_1}; z_2\mathbf{x}_{e_2}; ...; z_K\mathbf{x}_{e_K}]. \tag{4}$$

The max pooling procedure introduces a competing process for all the local descriptors to match templates. Each pooling coefficient z_k measures the response significance of $\mathbf{x}_{e_k}$ with respect to template $\mathbf{t}_k$, which is effective at categorizing coarse object shapes [18, 20], while the pooled local descriptor $\mathbf{x}_{e_k}$ preserves the local part details that are discriminative for classifying subtle fine-grained differences when the pooling coefficients are similar [6]. Therefore, our feature representation in Eqn. (4) can capture both coarse level object appearance changes and subtle object part changes, and we call this feature representation local feature embedding (LFE).

[3] In font rendering, the same font size may not generate the same text size in pixels for different fonts.

Our local feature embedding embeds the local descriptors from max pooling into a much higher dimensional space of $\mathbb{R}^{Kd}$. For instance, if we use 59-dimensional LBP descriptors and a codebook size of 2048, the dimension of $\mathbf{f}$ without using SPM is already 120,832. Although embedding the image into higher dimensional spaces is typically amenable to linear classifiers [23, 18, 20], training classifiers for very large-scale applications can be very time-consuming. What's more, a major drawback of training classifiers for large-scale classification is that, when images of new categories become available or new images are add to the existing categories, new classifiers have to be retrained at a very high computational cost [12]. In the following section, we propose a new large-scale classification algorithm based on local feature metric learning and template selection, which can be easily generalized to new classes and new data at very little cost. For this purpose, we modify the LFE feature in Eqn. (4) into a local feature set representation:

$$\mathbf{f} = \{(z_k, \mathbf{x}_{e_k})\}_{k=1}^{K}. \tag{5}$$

3.2. Large-Scale Classification

In our large-scale visual font recognition task, the dataset is typically open-ended, *i.e.*, new font categories appear over time and new data samples could be added to the existing categories. It is, therefore, important for a practical classification algorithm to be able to generalize to new classes and new data at very little cost. Nearest class mean (NCM) together with metric learning [12] has been explored for large-scale classification tasks, where each class is represented by their mean feature vector that is efficient to compute. In this paper, we generalize this idea to NCM based on pooled local features to form a set of weak classifiers. Then we propose a max-margin template selection scheme to combine these weak classifiers for the final classification.

3.2.1 Within-Class Covariance Normalization

Given the LFE feature $\mathbf{f} = \{(z_k, \mathbf{x}_{e_k})\}_{k=1}^{K}$ for each image, we would like to learn a Mahalanobis distance metric for each pooled local feature space, under which we formulate the NCM classifier using multi-class logistic regression [12], where the probability for a class c given a pooled local feature $\mathbf{x}_{e_k}$ is defined by

$$p(c|\mathbf{x}_{e_k}) = \frac{\exp(-\|\mu_k^c - \mathbf{x}_{e_k}\|_{W_k}^2)}{\sum_{c'=1}^{C} \exp(-\|\mu_k^{c'} - \mathbf{x}_{e_k}\|_{W_k}^2)}, \tag{6}$$

where μ_k^c is the class mean vector for the k-th pooled local features in class c, and

$$\|\mu_k^c - \mathbf{x}_{e_k}\|_{W_k}^2 = (\mu_k^c - \mathbf{x}_{e_k})^T W_k^T W_k (\mu_k^c - \mathbf{x}_{e_k}). \tag{7}$$

Denoting $\Sigma_k^{-1} = W_k^T W_k$, we can see the k-th pooled feature space (or its projected subspace) is modeled as a Gaussian distribution with an inverse covariance matrix Σ_k^{-1}.

To learn the metric W_k for the k-th pooled feature space, we use a simple metric learning method called within-class covariance normalization (WCCN). First, interpreting z_k as the probabilistic response of $\mathbf{x}_{e_k}$ to template $\mathbf{t}_k$, we can compute the class mean vector μ_k^c by

$$\mu_k^c = \frac{1}{Z^c} \sum_{i \in \mathcal{I}_c} z_k^i \mathbf{x}_{e_k}^i, \tag{8}$$

where i is the index for the i-th training image with LFE feature $\mathbf{f}^i = \{z_k^i, \mathbf{x}_{e_k}^i\}_{k=1}^{K}$, $\mathcal{I}_c$ denote the sample index set for class c, and $Z^c = \sum_{i \in \mathcal{I}_c} z_k^i$ is a normalization factor. Then, we compute Σ_k as the expected within-class covariance matrix over all classes:

$$\Sigma_k = E\left[\Sigma_{c'k}\right] \approx \sum_{c'=1}^{C} p(c') \Sigma_k^{c'}, \tag{9}$$

where

$$p(c') = \frac{\sum_{i \in \mathcal{I}_{c'}} z_k^i}{\sum_i z_k^i} \tag{10}$$

is the empirical probability of class c', and $\Sigma_k^{c'}$ is the within-class covariance for class c' defined as

$$\Sigma_k^{c'} \approx \frac{1}{Z^{c'}} \sum_{i \in \mathcal{I}_{c'}} z_k^i (\mathbf{x}_{e_k}^i - \mu_k^{c'})(\mathbf{x}_{e_k}^i - \mu_k^{c'})^T, \tag{11}$$

with $Z^{c'} = \sum_{i \in \mathcal{I}_{c'}} z_k^i$. In practice, empirical estimates of Σ_k may be noisy; therefore, we add a certain amount of smoothness by shrinking it towards the scalar covariance as

$$\hat{\Sigma}_k = (1 - \alpha)\Sigma_k + \alpha \sigma^2 I, \ \alpha \in [0, 1), \tag{12}$$

where $\hat{\Sigma}_k$ represents a smoothed version of the empirical expected within-class covariance matrix, I is the identity matrix, and σ^2 can take the value of $trace(\Sigma_k)$. Suppose we compute the eigen-decomposition for each $\hat{\Sigma}_k = U_k D_k U_k^T$, where U_k is orthonormal and D_k is a diagonal matrix of positive eigenvalues. Then the feature projection matrix W_k in Eqn (6) is defined as

$$W_k = D_k^{-1/2} U_k^T, \tag{13}$$

which basically spheres the data based on the common covariance matrix. In the transformed space, nearest class mean can be used as the classifier, which lays the foundation for the multi-class logistic regression in Eqn. (6).

To further enhance the discriminative power of W_k, we can depress the projection components with high within-class variability, by discarding the first few largest eigenvalues in D_k, which corresponds to the subspace where the

feature similarity and label similarity are most out of sync (large eigenvalues correspond to large within-class variance). In this case, it can be shown that the solution of WCCN can be interpreted as the result of discriminative subspace learning [19].

3.2.2 Max-Margin Template Selection

After we learned the metric for each pooled local feature space, and assuming the templates in T are independent, we can evaluate the posterior of a class c for the input image feature representation $\mathbf{f}$ by combining the outputs of Eqn. (6) using a log-linear model:

$$p(c|\mathbf{f}) = \frac{1}{H} \exp\left(a + \sum_k w_k \log p(c|\mathbf{x}_{e_k})\right). \quad (14)$$

where H is a normalization factor to ensure the integrity of $p(c|\mathbf{f})$, w_k weights the contribution of each pooled local feature to the final classification, and a is a small constant offset. Here, the weight vector $\mathbf{w} = [w_1, w_2, ..., w_K]^T$, shared by all classes, acts to select the most discriminative templates from the template model $T = \{\mathbf{t}_k\}_{k=1}^K$ for the given classification task. Then classification for $\mathbf{f}$ is simply to choose the class with the largest posterior:

$$c^* = \arg\max_{c'} p(c'|\mathbf{f}). \quad (15)$$

Alternatively, we can treat the multi-class logistic regression for each pooled local feature as a weak classifier, and then linearly combine them to obtain a strong classifier:

$$s(c|\mathbf{f}) = \sum_{k=1}^K w_k p(c|\mathbf{x}_{e_k}). \quad (16)$$

In this way, we can avoid the numerical instability and data scale problem of logarithm in Eqn. (14). The score function $s(c|\mathbf{f})$ does not have a probabilistic interpretation any more, but classification is again simply to find the class with the largest score output. In practice, we find this formulation works slightly better than the previous log-linear model, and we adopt this linear model for all the experiments.

Given the training samples $\{\mathbf{f}^i, c^i\}_{i=1}^N$, where $c^i \in \{1, ..., C\}$ is the class label for the i-th data sample, we want to find the optimal weight vector $\mathbf{w}$ such that the following constraints are best satisfied,

$$s(c^i|\mathbf{f}^i) > s(c'|\mathbf{f}^i),\ \forall i, c' \neq c^i, \quad (17)$$

which translates to

$$\sum_{k=1}^K w_k \left(p(c^i|\mathbf{x}^i_{e_k}) - p(c'|\mathbf{x}^i_{e_k})\right) > 0,\ \forall i, c' \neq c^i. \quad (18)$$

In order to learn $\mathbf{w}$, we define a cost function using a multi-class hinge loss function to penalize violations of the above constraints

$$\mathcal{L}(\mathbf{f}^i, c^i; \mathbf{w}) = \sum_{c' \neq c^i} \max\{0, -\gamma^i(c') + 1\}, \quad (19)$$

where

$$\gamma^i(c') = \sum_{k=1}^K w_k \left(p(c^i|\mathbf{x}^i_{e_k}) - p(c'|\mathbf{x}^i_{e_k})\right). \quad (20)$$

Then learning $\mathbf{w}$ is simply to solve the following optimization:

$$\min_{\mathbf{w}} \lambda \sum_{i=1}^N \mathcal{L}(\mathbf{f}^i, c^i; \mathbf{w}) + \rho(\mathbf{w}), \quad (21)$$

where $\rho(\mathbf{w})$ regularizes the model complexity. In this work, we use $\rho(\mathbf{w}) = \|\mathbf{w}\|_2^2$, and Eqn. (21) is simply the classical one-class SVM formulation. To see this, denoting

$$\mathbf{p}^i(c) = \left[p(c|\mathbf{x}^i_{e_1}); p(c|\mathbf{x}^i_{e_2}); ...; p(c|\mathbf{x}^i_{e_K})\right], \quad (22)$$

and $\mathbf{q}^i(c') = \mathbf{p}^i(c^i) - \mathbf{p}^i(c')$, Eqn. (19) can then translate to

$$\mathcal{L}(\mathbf{f}^i, c^i; \mathbf{w}) = \sum_{c' \neq c^i} \max\{0, -\mathbf{w}^T \mathbf{q}^i(c') \cdot 1 + 1\}, \quad (23)$$

where $\mathbf{q}^i(c')$ can be regarded as feature vectors with only positive label $+1$. Therefore, the optimization in (21) is the classical SVM formulation with only positive class and thus can be readily solved by many existing SVM packages, *e.g.*, [5]. The regularization term $\rho(\mathbf{w})$ here may also take other forms, such as $\|\mathbf{w}\|_1$, where the ℓ^1-norm promotes sparsity for template selection, which typically has better generalization behavior when the size K of the template model T is very large.

After we learn the WCCN metric for all pooled local feature spaces and the template weights based on LFE, classification for a given $\mathbf{f}$ is straightforward: first compute the local feature posteriors using Eqn. (6), combine them with the learned weights $\mathbf{w}$, and then predict the class label by selecting the largest score output $c^* = \max_{c'} s(c'|\mathbf{f})$. When new data or font classes are added to the database, we only need to calculate the new class mean vectors, and estimate the within-class covariances to update the WCCN metric incrementally. As the template model is universally shared by all classes, the template weights do not need to be retrained.[4] Therefore, our algorithm can easily adapt to new data or new classes at little added cost.

[4] Same as other supervised learning algorithms, if the new added data or classes change the data distribution substantially, the template model and their weights need to be retrained.

4. Experiments

We now evaluate our large-scale recognition algorithm on the collected VFR database. We implement and evaluate two baseline algorithms: 1) a representative font recognition algorithm on scanned documents [2]; and 2) a widely used image recognition algorithm LLC [18]. To get the local feature embedding (LFE) representation, we evaluated several state-of-the-art texture or shape descriptors including covariance feature [14], shape context [3], local binary pattern (LBP) [1] and SIFT [10], which have been extensively verified in many computer vision applications. We find that SIFT works the best. For the local descriptor encoding, we use LLC coding [18] to compare fairly with the LLC baseline, although other coding schemes can also be used, such as soft assignment and sparse coding.

Our algorithm has very few parameters. The local descriptors(e.g., SIFT, LBP) are extracted from 12×12 image patches sampled from a regular grid on the image with step size of 6 pixels. The template model T is learned by kmeans with size 2048. To compute the smoothed version of the within-class covariance in Eqn. (12), we set $\alpha = 0.1$ as a constant. When calculating the projection matrix W_k in Eqn. (13), we throw away the first two largest eigenvalues from D_k to depress the components with high within-class variability.

4.1. Dataset Preparation

Our experimental dataset consists of three distinct sets: VFR-447, a synthetic dataset containing 447 typefaces with only one font variation for each typeface; VFR-2420, a large synthetic dataset containing typefaces with all variations; and VFR-Wild, which has 325 real world test images for 103 fonts out of the 2420 classes, each class with the number of images ranging from 1 to 17. Each class in VFR-447 and VFR-2420 has 1,000 synthetic word images, which are evenly split into 500 training and 500 testing. There are no common words between the training and testing images.

To model the realistic use cases, we add moderate distortions and noise to the synthetic data. First, we add a random Gaussian blur with standard deviation from 2.5 to 3.5 to the image. Second, a perspective distortion is added by moving each corner of the image randomly by ± 5 pixels along x and y directions (images are pre-normalized to have the same height of 105 pixels), which defines a perspective transformation. Third, the foreground (text) and background intensities are randomly perturbed between [0, 255] with the constraint that the intensity of foreground is at least 20 intensity levels smaller than the background. Finally, some small Gaussian noise is added to the image, where we assume the noise of a test images will be reasonably reduced with some simply preprocessing.

Codebook	512	1024	1536	2048
LLC + SVM	64.63	73.04	78.13	82.23
LFE + Naive	76.04	80.20	81.77	84.21
LFE + FS	**80.44**	**85.26**	**87.73**	**91.35**
Improvements	18.36%	25.56%	32.69%	45.22%

Table 1. The top 1 accuracy of different algorithms with different template model sizes on VFR-447. "Improvements" shows error reduction percentage of LFE+FS against LFE + Naive.

Covariance [14]	Shape C. [3]	LBP [1]	SIFT [10]
44.13	54.13	89.02	**91.35**

Table 2. The top 1 accuracies of different local descriptors with our proposed LFE+FS algorithm on VFR-447.

4.2. Results on VFR-447

Table 1 shows the recognition results on the VFR-447 synthetic dataset under different template model sizes in terms of top 1 accuracy. "LFE+Naive" denotes our method without template selection, *i.e.*, equal weights in Eqn. (16), and "LFE+FS" denotes our method with template selection. In all cases, both our methods significantly outperform the baseline algorithm LLC [18]. Table 2 lists the recognition results with different local descriptors. We can see that SIFT performs the best. LBP is slightly worse than SIFT, but its efficiency justifies itself as a good alternative to SIFT.

In Table 3, we evaluated the top 1, 5, and 10 accuracies on the VFR-447 dataset with 2048 templates, in comparison with the two baseline algorithms [2] (denoted by "STAT") and LLC [18]. In all cases, our algorithm works the best. The previous font recognition algorithm [2] focuses on scanned documents. It depends on a large text sample to extract stable texture statistics for recognition. Therefore, it won't work well in our case where the test images have very short texts with noisy background. Since the VFR-447 dataset does not have font variations within each typeface, *i.e.*, most font classes are visually distinct, coarse-grained techniques such as LLC is still working reasonably well.

Figure 3 depicts the max-margin template selection results on the 2,048 template model. The left figure plots the sorted weights for the 2,048 templates after max-margin template selection using optimization in Eqn. (21). Although all templates are selected (using $\|\mathbf{w}\|_2^2$ regularization), only a small portion of the templates are selected with large weights. The right figure illustrates the top 81 selected templates[5], most of which correspond to letters endings and curvature strokes that are most informative for font recognition.

[5]The templates are visualized by image patches whose local descriptors are the nearest neighbors to the selected templates

Methods	Top 1	Top 5	Top 10
STAT [2]	25.12	30.30	33.41
LLC + SVM [18]	82.23	93.39	95.32
LFE + Naive	84.20	94.14	95.53
LFE + FS	**91.35**	**98.90**	**99.62**

Table 3. The top 1, 5, 10 accuracies of different algorithms with template model size 2,048 on VFR-447 synthetic dataset.

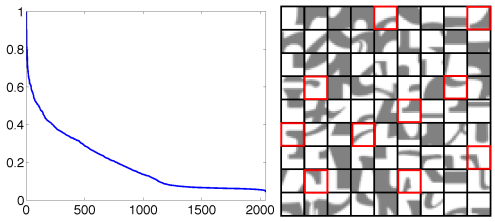


Figure 3. Sorted template weights (left) and top 81 selected templates (right) with largest weights. Many of these selected templates correspond to letter endings.

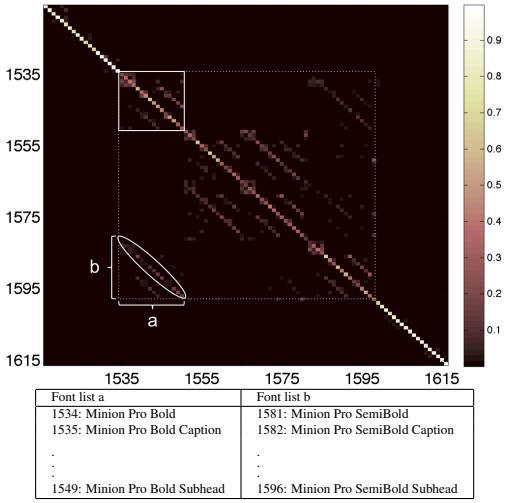
4.3. Results on VFR-2420

In Table 4, we report the top 1, 5, and 10 accuracies on the VFR-2420 dataset with template model of size 2,048 and LBP as local descriptor. Compared with the results on VFR-447 in Table 1, the top 1 accuracy of our algorithm for VFR-2420 drops notably, from 91.35% to 72.50%, which is expected as the problem becomes super fine-grained with font variations within each typeface. However, the top 5 and 10 accuracies are much better, suggesting that our algorithm is effective at retrieving similar font classes, even though it is confused by subtle font variations for top 1 classification. In contrast, LLC works much worse than our algorithm on this dataset, due to its ineffectiveness in handling fine-grained recognition tasks. Again, STAT [2] performs the worst.

To see that the top-1 accuracy of our algorithm is mainly affected by font variations within each typeface, fig. 4 plots a sub-confusion matrix for 100 fonts indexing from 1515 to 1615. The sub-confusion matrix demonstrates a hierarchical block structure, where large blocks correspond to font variations within a typeface, *e.g.*, the large dotted rectangle corresponds to all font variations within typeface Minion Pro, and smaller blocks correspond to font variations within a subfamily of a typeface, *e.g.*, the small rectangle corresponds to variations within Minion Pro Bold. Interestingly, there are many periodic off-diagonal lines inside the blocks, which are typically caused by one particular font variation. For example, the line structure in the ellipse is caused by the similarity of weight variation between bold and semibold, as indicated by their font names listed in the accompanying table. Such observations suggest that our confusion matrix is a good reflection of font similarity, which may be useful for font organization, browsing, and recommendation. Fig. 4.3 shows a clean example of the top-10 predicted fonts for "MinionPro-MediumCnItCapt".These top-ranked predictions are visually very similar to the input font and some are even indistinguishable by human eye.

Methods	Top 1	Top 5	Top 10
STAT [2]	15.25	20.50	26.68
LLC + SVM [18]	50.06	72.48	78.49
LFE + Naive	65.20	85.36	89.93
LFE + FS	**72.50**	**93.45**	**96.87**

Table 4. The top 1, 5, 10 accuracies of different algorithms with template model size 2,048 on VFR-2420 synthetic dataset.



Font list a	Font list b
1534: Minion Pro Bold	1581: Minion Pro SemiBold
1535: Minion Pro Bold Caption	1582: Minion Pro SemiBold Caption
.	.
.	.
.	.
1549: Minion Pro Bold Subhead	1596: Minion Pro SemiBold Subhead

Figure 4. The sub-confusion matrix for 100 font classes indexing from 1515 to 1615.

Figure 5. A clean example of the top-10 predicted fonts for "MinionPro-MediumCnltCapt". Most of them are variations from the same typeface. From top left to bottom right (prefix "MinonPro-" omited where appropriate): MediumCnltCapt, CnItCapt, CnIt, MediumIt, ItCapt, MediumItCapt, MeliorLTStd-Italic, It, SemiboldIt.

4.4. Results on VFR-Wild

Table 5 shows the performance of our algorithm on the real world test images. As introduced in Section 2, all the real-world images were roughly cropped and oriented manually. Then we only did minimum preprocessing of denoising using a bilateral filter with fixed parameters for all images. For heavier distortions, more complicated preprocessing is needed, which we leave as future work. As shown in Table 5, compared to the synthetic data, the performance drops notably, which is expected because of the mismatch between training and testing data. Nevertheless, our LFE

Methods	Top 1	Top 5	Top 10	Top 20
STAT [2]	4.13	7.30	9.08	10.17
LLC + SVM [18]	26.46	44.00	51.08	57.23
LFE + Naive	29.54	46.15	54.46	62.46
LFE + FS	**52.61**	**58.40**	**62.14**	**64.16**

Table 5. The top 1, 5, 10, and 20 accuracies of different algorithms with 2,048 templates on real world dataset.

(a) (b)

Figure 6. (a) Real world images that are correctly classified (rank one). (b) Real world images that are wrongly classified (fall out of top rank 20).

combined with template selection again significantly outperforms both LLC and LFE without template selection. STAT was developed for scanned documents and performs very poorly on the wild data. Fig. 6 shows some example real world test images that are (a) correctly and (b) wrongly classified by our algorithm. Remarkably, our model, although trained on the synthetic dataset, is robust to cluttered background and noise to a large extent shown by (a). In cases of decorated texts, very low-resolution, extremely noisy input, and very cluttered background shown in (b), the algorithm will fail. Better image preprocessing techniques will definitely help in such cases, which we leave as future work. Considering all these challenging factors in real world VFR, a recognition rate of 52.61% at top 1 accuracy is very promising. Note that most of our real world images contain very short strings. In cases where the input may contain long strings of text, we can cut the them into short strings and combine the algorithm's inferences on all of them.

5. Conclusion

In this paper, we focus on the large-scale VFR problem that has long been neglected by the vision community. To address this problem, we have constructed a large-scale set of synthetic word images for 2,420 font classes, and collected a small set of real world images. Experiments on synthetic test data demonstrate the effectiveness of our approach, and experiments on real test images show very promising results. For future work, we will continue to grow the dataset of real test images. We will also explore the effects of different levels of distortions and noise added to synthetic data on the final performance on real test images. Machine learning and computer vision techniques, *e.g.*, transfer learning and robust local descriptors, will be exploited to close the gap between synthetic training and real world testing.

References

[1] T. Ahonen, A. Hadid, and M. Pietikinen. Face description with local binary patterns: Application to face recognition. *PAMI*, 2006.

[2] C. Aviles-Cruz, R.Rangel-Kuoppa, M. Reyes-Ayala, A. Andrade-Gonzalez, and R. Escarela-Perez. High-order statistical texture analysisfont recognition applied. *Pattern Recognition Letter*, 2005.

[3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 2002.

[4] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *ICCV*, 2013.

[5] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classificatio. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[6] R. Farrell, O. Oza, N. Zhang, V. Morariu, T. Darrell, and L. Davis. Birdlets: subordinate categorization using volumetric primitives and pose-normalized apparances. In *ICCV*, 2011.

[7] Google Tesseract-OCR Eigine. https://code.google.com/p/tesseract-ocr/.

[8] M.-C. Jung, Y.-C. Shin, and S. N. Srihari. Multifont classification using typographical attributes. In *Proc. of Intl. Conf. on Document Analysis and Recognition*, 1999.

[9] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[10] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[11] H. Ma and D. Doermann. Gabor filter based multi-class classifier for scanned document images. In *Proc Intl Conf on Document Analysis and Recognition*, 2003.

[12] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric learning for large-scale image classification: generalizing to new classes at near-zero cost. In *ECCV*, 2012.

[13] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[14] F. Porikli and T. Kocak. Robust license plate detection using covariance descriptor in a neural network framework. In *AVSS*, 2006.

[15] R. Ramanathan, L. Thaneshwaran, and V. Viknesh. A novel technique for english font recognition using support vector machines. In *Proc Intl Conf on Advances in Recent Technologies in Communications and Computing*, 2009.

[16] M. Solli and R. Lenz. Fyfont: find-your-font in large font databases. In *SCIA*, 2007.

[17] H.-M. Sun. Multi-linguistic optical font recognition using stroke templates. In *ICPR*, 2006.

[18] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[19] S. Yan, X. Zhou, M. Liu, M. Hasegawa-Johnson, and T. S. Huang. Regression from patch-kernel. In *CVPR*, 2008.

[20] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[21] S. Yang, L. Bo, J. Wang, and L. G. Shapiro. Unsupervised template learning for fine-grained object recognition. In *NIPS*, 2012.

[22] B. Yao, G. Bradski, and L. Fei-Fei. A codebook-free and annotation-free approach for fine-grained image categorization. In *CVPR*, 2012.

[23] X. Zhou, K. Yu, T. Zhang, and T. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010.

[24] Y. Zhu, T. Tan, and Y. Wang. Font recognition based on global texture analysis. *PAMI*, 2001.