A Comparison of Multiple Classifiers for Font Recognition

Project Report of Group 7

Chen Hanzhao
School of Mathematics, Peking University
1500010703

Li Hongcheng
National School of Development, Peking University
1600011374

Liu Taide
School of Electronic Engineering and Computer Science, Peking University
1500012923

I. Introduction

Classification and segmentation are two main problem in the field of computer vision, and has been well discussed in the past tens years. Font recognition is similar to semantic classification problem, it hopes to tell which font the character(s) in a image it belongs to.

The possible application of font recognition is various, for example:

1) To answer a curious person who simply wants to know the name of a font he came across
2) To convert photos of handwritten texts into files in computer(such as Word or PDF) with its corresponding type of font
3) To generate the electronic version of a published book with the corresponding type of font
4) To specify the font for text recognition, and make text recognition more precise
5) Help recognize the font to give the evidence for copyright dispute

Traditionally, this work is taken by professional person, especially in the context of litigation. Human will use the serif design, strength change, angle of characters, etc. to determine which font one character belongs to. It is not a trivial job for computer, not only for the reason that font recognition needs carefully inspection, but also for the bad quality of image in most situation (low resolution, slight deflection, etc). Besides, there are many types of fonts are inherited from another, which brings more difficulty.

We decide to use font recognition as our project topic is for the reason that this is an interesting problem that is rarely discussed, while the related problem, semantic classification, has been almost solved. Therefore, we could try many methods and compare with the state-of-the-art techniques to gain a deeper insight of the methods we learn.

In this project, we tests naive Bayesian inference, multi-class SVM, decision tree, random forest, clustering algorithms and deep convolutional neural networks, and gives the conclusion that this problem has significant difference with semantic classification, and the non-triviality of this problem.

We have to admit that this work has a lot to improve. Given the limitation of hardware, we only tested seven fonts with thousands instances, which is unfair for DCNN and the scalability of methods can hardly be discussed.

II. Related Works

As far as we know, the font recognition problem firstly proposed not later than 1993. Zramdini et al. proposed using projection profiles to recognize the font of a list of characters [1]. And they proposed using typographical features [2] in 1998. In 2001, Zhu et al. [3] proposed using Gabor filter to recognize font, and reaches a pretty high accuracy. Note that these works are based on text sequence (sometimes very long). The confusion rate is not as low as modern techniques, but with a long sequence, voting could highly reduce the confusion probability.

As for recent works about this problem, Chen et al. used SVM with Local Feature Embedding (LFE)

and template selection (FE) on dataset VPR-447 [4], which has 447 classes of 105-pixel characters. The accuracy with 2048 templates is 91.35%. Wang et al. applied convolutional neural networks on font recgnition [5]. His effort, DeepFont, was accepted by Adobe as their font recognition algorithm in their product. He used a architecture called Stacked Convolutional Auto-Encoder(SCAE) to adapt CNN, and the error rate on real-world VPR dataset(105 pixels, 617 classes) was 10.87%. Xiao.C et al. developed FontCode [6], which is partially based on Chen and Wang's work. His method firstly recognizes font, then predict character based on the predicted font, making character recognition more precise.

We could found that these methods are all precisely designed for this problem, but we do not know how the traditional, generic pattern recognition methods performs in this problem, and that is what we want to discuss in this project.

III. Dataset

A. Introduction

The dataset we use is from UCI Machine Learning Repository[1], which contains 745000 instances of Unicode characters from 153 different fonts (or font families). The characters in this dataset have different sources, varies from computer generated, scanned, to handwritten, and many properties like italic, bold. All the characters are cut by the circumscribed rectangle, resized to 20×20 pixels, and stored using gray-scale image. This dataset is also used in the book 'Python Machine Learning' by Sebastian Raschka, but we do not copy or reference any part of the code.

Data selection. There are several problems about the dataset. First, the number of instances is highly unbalanced between fonts, from 824 to 93688. This is for the reason that some fonts are widely used in real life, like Arial, Calibri, etc., while some other fonts are rarely used. Second, the font families are not independent actually. Some fonts are designed based on other font(s), which brings difficulties to separate them. Third, there are many Unicode characters like '$\neq$', '$\int$', '$\leqslant$', etc. These characters can hardly provide information about font as they are all similar in each font family and only differ in size and line strength. Therefore, we have to choose a balanced, independent subset consists of regular characters in our experiement. We choose the ANSI characters from the following seven fonts: Californian, Harrington, Brush, Modern, Papyrus, Edwardian and Freestyle. We select these fonts because they all have significant difference to each other, and include many types of font like serif and sans-serif, regular and flourish. Besides, they contains the same number of instances, which means we do not need to consider the unbalance problem.

Image processing. As the height-width ratio is an important characteristic of font, the image in the raw data need to be resized using bilinear interpolation to the original size to rebuild height-width ratio. Although modern techniques do not require images of input have the same size (convolution as pooling, global pooling, etc.), to ensure the data suitable for all the methods, all the images will be zero-padded to 64×64 pixels.

Data augmentation. The subset only contains thousands of instances, which is far too smaller than the number of parameters, especially for deep convolutional neural networks. Therefore, data augmentation is needed. All the images fed into the CNN model for training will firstly be randomly cropped to 48×48 pixels and then zero-padded back to 64×64. Although there will be some broken characters, the font information, serif for example, will not lost much. The data augmentation does greatly expand the training set and help avoid overfitting.

IV. Feature Extraction

Feature extraction serves as a critical part in CV (Computer Vision) processing problems, for appropriate dimension reduction increasing computation speed and enhancing efficiency of information revelation secures a more compact and separable feature space salutary in classification problems.

A. Incremental PCA

The most common and simplest extraction strategy will be PCA (Principal Component Analysis), raised firstly by Pearson [7] and developed on computational efficiency by multiple scholars, e.g. Oja(1982 [8], 1992 [9]) and Sanger(1989 [10]).

[1] http://archive.ics.uci.edu/ml/datasets/Character+Font+Images

We utilize the incremental PCA method provided in Python's scikit-learn package to directly transform the 64×64 images acquired in Section III. To supply a rough outline of PCA extraction, we plot the seven classes mentioned in Section III under the first and the second principal components and get Figure 1(a), and likewise under the 20th and the 21st components to obtain Figure 1(b). It is readily to notice that the overlap problem is relatively severe under such a direct manner of PCA extraction, since even the first two components carrying the most significant variation cannot imply much separability as shown in Figure 1(a).

To briefly evaluate the efficacy of direct PCA transformation, we apply SVM multi-classifier to test the accuracy the extraction method is capable to reach. We select the top 50 principal components as training features, with RBF kernel function and regularization parameter $C = 20$ as initiation of SVM classifier. We replicate seven times of training where during each time 10-fold cross-validation is utilized to gain stable accuracy level. The result is presented in Figure 2. From this presentation, we notice the lowest accuracy of seven fonts is lingering about 70% whereas the highest merely stands on 90%. Tolerable as it appears to be, more efficient methods are required.

B. Features for CV

On the concern of optimizing the quality of feature extraction, we consider methods specialized in CV area. Classical local region descriptors such as SIFT [11], [12] and GLOH [13] have been very well understood. The main idea of SIFT(Scale-Invariant Feature Transformation) is to sum up the local region pixels with weights positively related to the distance from a reference point to each specific pixel and negatively related to the angular separation of the orientation of the image gradient of these two points, and by choosing different orientation of the reference point and moving the reference point to different location, a three-order tensor, in which each component represents a weighted sum, is derived. However, computational inefficiency and inadequate rotational robustness remain as severe problems. We further peruse the introduction of DAISY descriptor by Tola, Leptit and Fua [14], in which they justify the preponderance of DAISY over traditional descriptors. Moreover, we investigate into HOG feature elaborated by Dalal and Triggs [15].

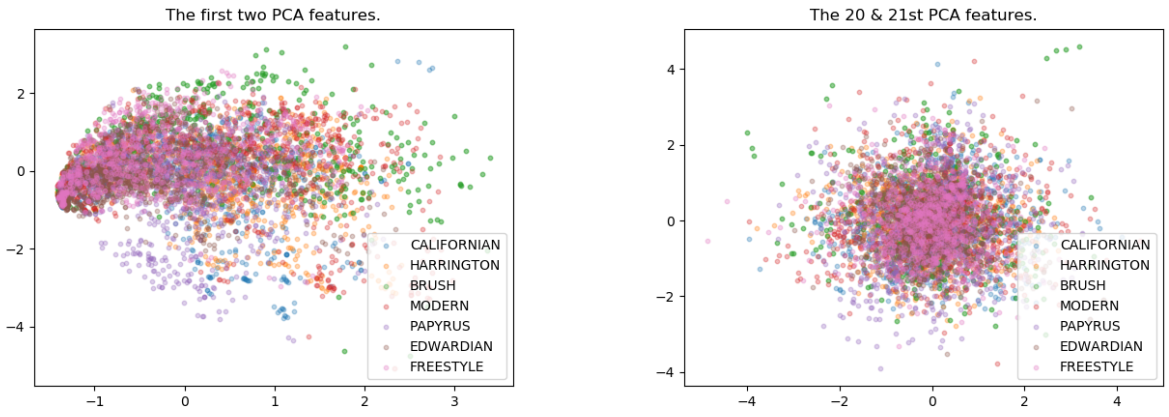
The main methods we utilize include DAISY descriptor and HOG descriptor. We firstly apply them to process images(64×64) and secure a more sparse three-order tensors than the original figures, and after that we still use PCA method to extract information. Likewise, we depict the seven fonts in feature space respectively composed of the first two and the 20th with the 21st principal components in Figure 3(a) and 3(b). In comparison with direct transformation, principal components extracted from DAISY descriptor seems to be more separable, since the focal region in Figure 3 is more extensive than that in Figure 1 and overlap is obviously alleviated.

To test the efficiency improvement DAISY feature is able to accomplish, we illustrate a comparison on classification accuracy between the top 50 PCA features of raw images and the counterpart of DAISY-transformed tensors in Figure 4, where we still use SVM initialized with RBF kernel and $C = 20$. The enhancement turns out to be prominent, as the lowest accuracy of DAISY method is 5-10% higher than mere PCA strategy while the highest level almost reaches 95%. And the internal differences among DAISY accuracy of fonts are considerably compressed, which is also robust when it comes to two-class classification problem where direct PCA transformation takes on *ill domination*, namely, while accuracy of one font will be high, that of the other tend to be extremely low, and the severity of such domination will be compounded if the two fonts look alike. Thus, based on the predominant efficiency and robustness, we only utilize CV features like DAISY in our later discussions and the direct PCA method is abandoned.

V. Evaluation

A. Experiment Setup

- Programming Language: Python 3.5
- Used Libraries: numpy, scikit-learn, scikit-image, tensorflow-gpu
- CPU used: Intel Core i7-6700HQ (KNN, Mean-shift, DCNN), Intel Core i5-6200U (NB, SVM), Intel Core i5-5200U (DT,RF,K-means)



(a) 1st and 2nd principal components

(b) 20th and 21st principal components

Fig. 1: Seven fonts depicted in the feature space composed of principal components derived directly from 64×64 images.

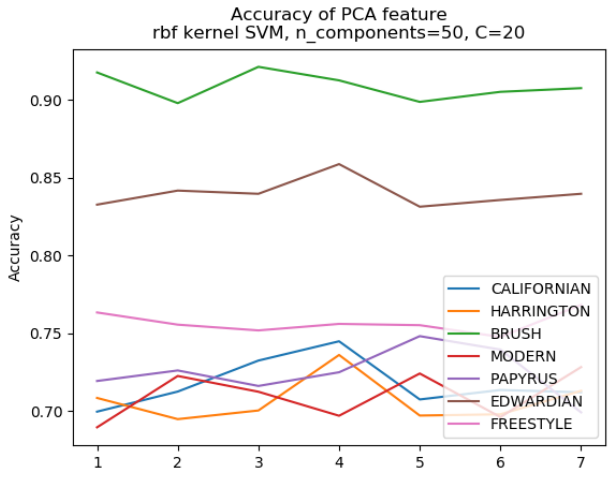


Fig. 2: The classification accuracy realized using the top 50 directly transformed principal components, under the initiation of RBF kernel and regularization parameter $C = 20$.

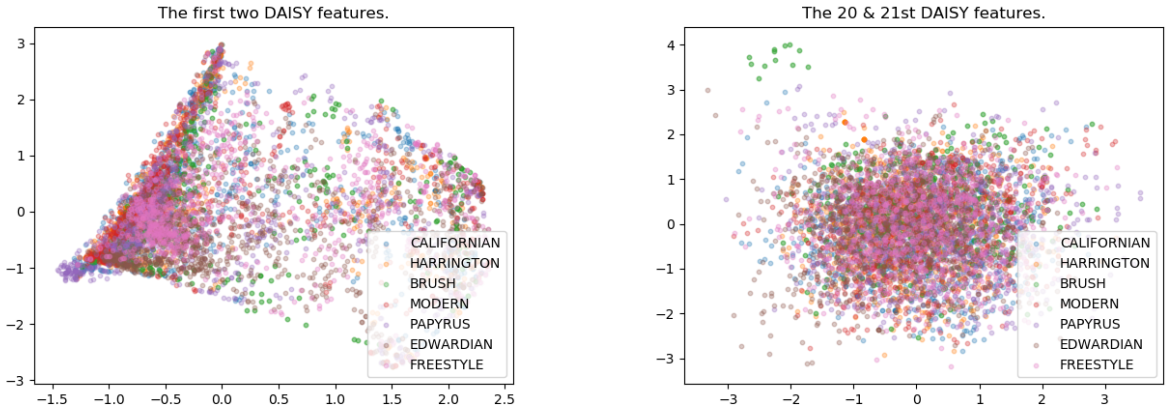
- GPU used: Nvidia Geforce GTX 960M (vmem: 2GB)

B. Naive Bayesian Inference

Simple probabilistic classifiers have been extensively discussed and very well understood since 1950s, and among them NB(Naive Bayes) classifier is one of the most eminent utilized in text categorization at its incipient stage. For reference, we learn about the discussions, among oceans of literature, by McCallum and Nigam(1998 [16]), Lewis(1998 [17]) and Zhang(2004 [18]).

Introduction. The naive Bayesian classifier is based on probabilistic theory and Bayesian principle along with the critical assumption that all input features are independent to each other in the sense of conditional probability. To formulate this, we denote $\boldsymbol{x}$ as the input vector where x_k represents its kth component($k = 1, 2, \ldots, m$), and ω_i is noted as the ith class($i = 1, 2, \ldots, c$). The conditional probability of x_k conditional on ω_i is written as $p(x_k|\omega_i)$, and the prior probability of each ω_i is $P(\omega_i)$. Thus, according to Bayesian principle and independence assumption, we can derive the post probabilty of each class $P(\omega_i|\boldsymbol{x}) \propto P(\omega_i) \prod_{k=1}^{m} p(x_k|\omega_i)$, from which the maximal post probability criterion is able to give the classification result. In Python's scikit-learn package, three different kinds of NB classifiers are provided, among which the only distinction remains in the way of modelling the conditional probability, for instance, $GaussianNB$ method assumes that $p(x_k|\omega_i)$ takes on the form of normal distribution. In our practice, we notice that different modellings of conditional probability only generate nuance in classification efficacy, so we only apply Gaussian NB method hereafter.

Evaluation of NB. It is apparent that the independence assumption cannot hold in font recognition problem, for pixels in an image always correlate



(a) 1st and 2nd principal components

(b) 20th and 21st principal components

Fig. 3: Seven fonts depicted in the feature space composed of principal components derived directly from 64×64 images.

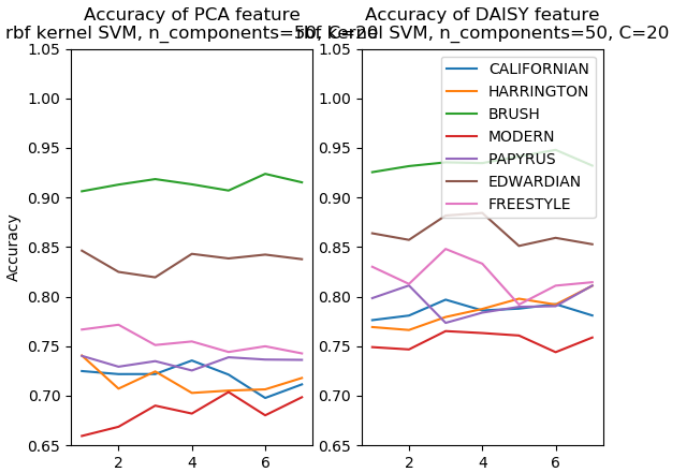


Fig. 4: Comparison of classification accuracy between the top 50 principal components of raw images and that of DAISY-transformed images, under the initiation of RBF kernel and regularization parameter $C = 20$.

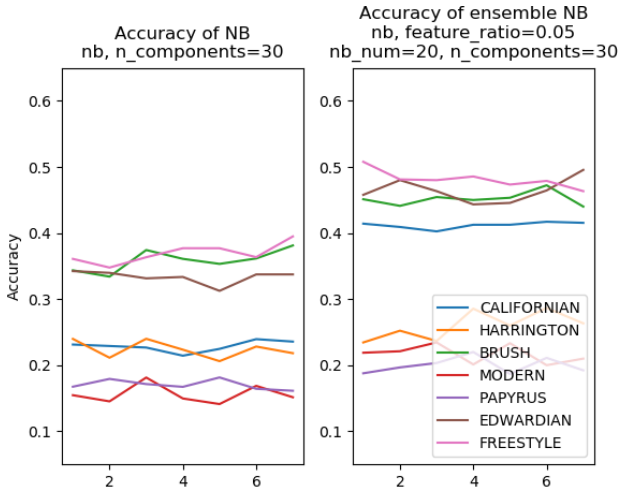


Fig. 5: Comparison of normal Gaussian NB method and ensemble Gaussian NB method by the top 30 DAISY-transformed principal components, with ensemble parameters feature ratio=0.05 and classifier number=20.

in a nonlinear manner as long as the content of the image makes sense and varies according to some regularity. Applying DAISY descriptor may alleviate this problem, but its convolutional nature of transformation will for sure retain most of the confounding. Consequently we predict an unsatisfactory outcome. Indeed the actual result is far from ideal which is depicted on the left side of Figure 5, in which we use the top 30 principal components from DAISY descriptor as input vector. It is noticeable that the lowest level of accuracy is only slightly higher than the bottom line, $1/7$, namely the accuracy of completely random guess in this seven-font problem, while the highest is less than 40%.

To rescue the extremely awful performance of NB classifier seems imperative, and after several trials we eventually consider the ensemble strategy. Rodriguez and Kuncheva(2007 [19]) offer an ensemble method called random oracle designed

for NB classifier, in which each weak classifier is trained based on both proportional selection of samples and proportional selection of features. Nonetheless, given that our scale of data is comparatively small and each of DAISY feature does not convey direct and specific practical meaning, we simplify this strategy by only randomly choosing features according to a ratio. On the right side of Figure 5, we set the feature ratio to be 0.05, namely five will be randomly selected out of a hundred, with the number of weak NB classifiers being 20, and likewise using the top 30 DAISY features, we trained the ensemble NB classifier for seven-font problem. The improvement is significant, as the accuracy of every font is increased by 5-10%. What remains undesirable is that the absolute level of classification efficacy is still low and no font's accuracy even surpasses the 50% line.

NB as Font Similarity Measure. Even though naive Bayes is no efficient categorization method, the distributive pattern of accuracy of different fonts still provides important information, and we find that NB model tends to enlarge the range of accuracy of different fonts(in seven font case, 35% for NB while 20% for SVM in seven-font problem), which reinforces the presence of such pattern.

We show this insight via Figure 6 where we can overtly observe a division between the upper three sorts of fonts and the lower four, and their levels of accuracy stand very closely if they are on the same side to the division but distantly if separated on different sides. This ranking seems robust even if we alternate the parameters. Now we explain why this happens by first showing how the seven fonts look like, and this is exhibited in Figure 7 where the left three fonts correspond exactly to the upper three in Figure 6 and the fonts on the right the four at the bottom. Such a phenomenon suggests that there exists a positive correlation between font characteristics and classification accuracy. Along with more tests, we conclude that fonts with inclined posture, bold and thick lines or tall and thin shape tend to obtain relatively high accuracy. Naturally through this property, we conjecture that fonts sharing similar traits are prone to manifest accuracy of similar level via NB classifier, and consequently NB classifier may serve as a kind of font similarity

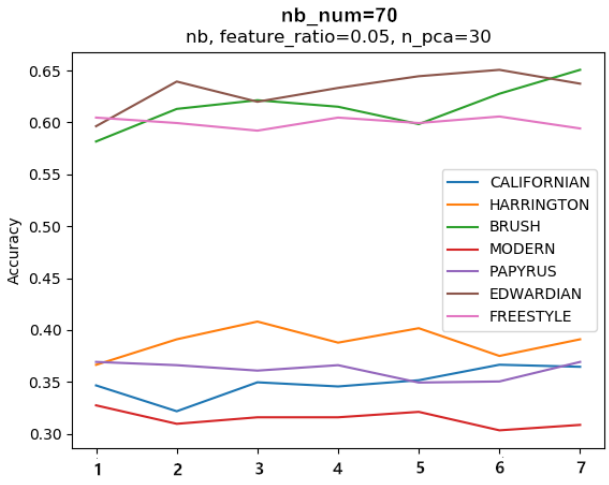


Fig. 6: Classification outcome of seven fonts, using the top 30 DAISY features and ensemble strategy, with feature ratio=0.05 and classifier number=70.

Fig. 7: The three fonts on the left side are inclined squiggles while the four on the right stand erectly and have few ornamental branches.

measure.

To partially justify this hypothesis, we consider a fourteen-font problem's as robustness check by adding another seven randomly selected fonts to the original seven-font problem and re-operating NB classification, and we attain Figure 8. What is interesting, the accuracy lines within the red box in Figure 8 correspond to five and the only five squiggles in these fourteen fonts. And lines within blue box correspond to all the erectly standing and ornament-free fonts. Furthermore, the font with the highest accuracy is a sort with exaggeratedly thick lines called *Bauhaus*, while the one with the lowest accuracy turns out to be font with common look

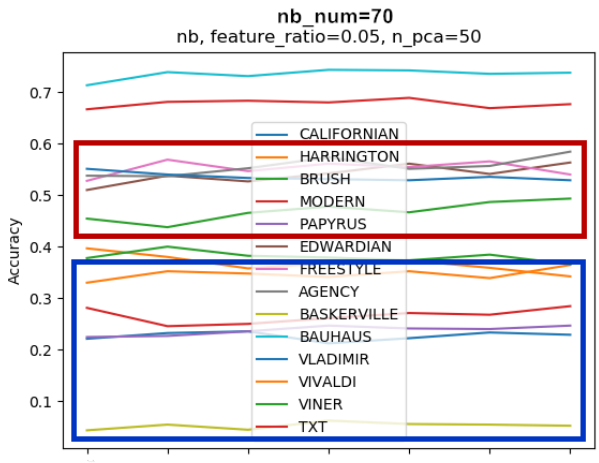


Fig. 8: Classification outcome of fourteen fonts, using the top 30 DAISY features and ensemble strategy, with feature ratio=0.05 and classifier number=70.

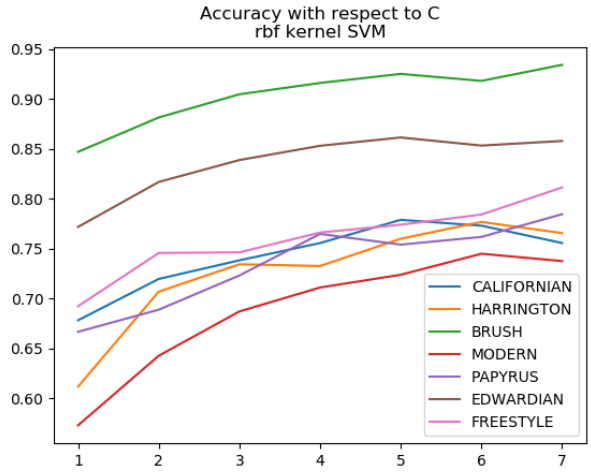


Fig. 9: The increasing trend of SVM classification accuracy with respect to C varying from 1 to 7, with RBF kernel, using the top 50 DAISY features.

resembling the widely used *Times New Roman*, and this one is called *Baskervile*.

C. Multiclass-SVM

Support vector machine was firstly raised by Vapnik and Chervonenkis [20], and subsequently developed by Vapnik et al.(e.g., 1999 [21] and 2000 [22] [23]). The SVM multi-classifier earns the reputation of being efficient, robust and especially resistant to over-fitting, and we have full reason to expect its excellent performance in font recognition.

Evaluation of SVM. Through parameter adjustment, we optimize the classification on both concerns of efficiency and economy, test SVM's robustness with respect to parameters, and disclose some peculiar properties of font recognition. All the accuracy values derived are stabilized via 10-fold cross validation.

We firstly find that SVM classifier generally exhibits the best outcome with RBF kernel and default-valued $\gamma = 1$ parameter. Nonetheless, for the sake of severe overlap in feature space, parameter selections of these two aspects do not engender sensible variation in recognition accuracy.

The critical regularization parameter, C, is subsequently considered. We depict the accuracy levels of seven fonts respectively with respect to different c values varying from 1 to 50, using the top 50 DAISY features. To exhibit the significant increasing trend, we only partially demonstrate the curves corresponding to C from 1 to 7 in Figure 9. We can easily see that increasing regularity parameter leads to sharp increments in accuracy by on average 10-12%, whereas the marginal effect tends to decrease. When we further extend this figure toward its right side, each curve converges to a vertical line roughly at the level $C = 20$, however, the total improvement in this extended part is comparatively less significant than that in Figure 9. The insight of this improvement and convergence lies in the fact of serious overlap problem which seems to be inevitable in font recognition along with the presence of numerous outlier points, both indicating that more slackness will be salutary and necessary.

Besides, we investigate the most apt number of DAISY features that we should apply. We choose the parameters stated above(RBF kernel, $C = 20$) and depict the variation of accuracy with respect to feature numbers varying from 10 to 70 in Figure 10. We find that the overall trend is increasing while the marginal effect decreasing, and the best number to choose lies around 50. Actually when we further increase the feature number beyond 70 to 100 or even 200, the accuracy curves will sustain as flat lines and still do not go downward, which suggests excellent resistance against over-fitting. Such property of resistance will be explained not only by the

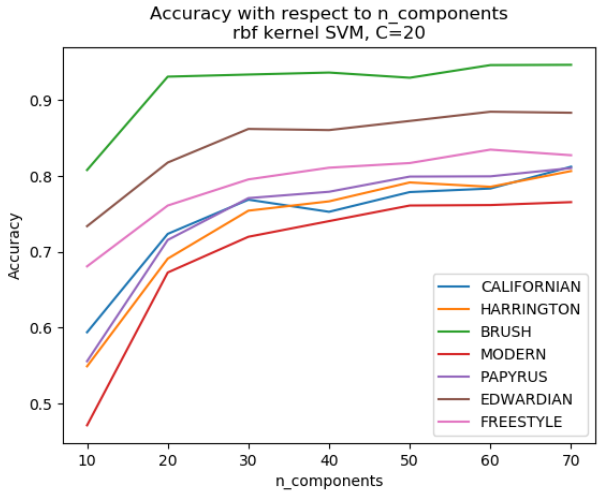


Fig. 10: The increasing trend of SVM classification accuracy with respect to number of DAISY features varying from 10 to 70, with RBF kernel and $C = 20$.

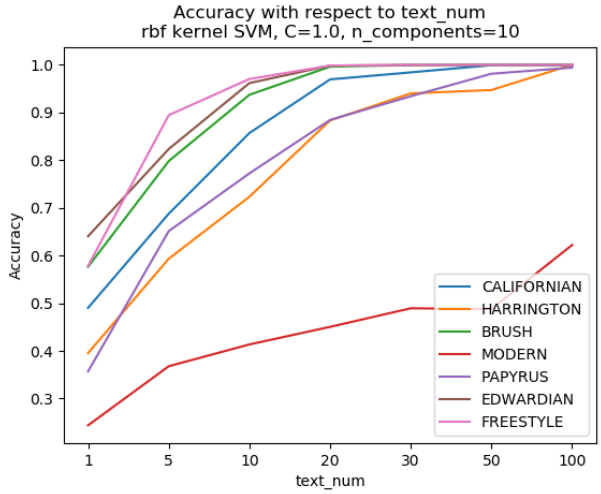


Fig. 11: The accuracy of text font recognition with respect to the length of text. A weak SVM is utilized, with $C = 1$, using only top 10 principal components.

intrinsic robustness of SVM, but also by the fact about font recognition that the information useful of telling the font sort of a specific image tends to appear in higher order principal components but not the most top ones. To clarify this, we shall consider two images between which there exist two kinds of variation, namely the variation of the content and variation of the type of font, and we need to point out that the former seems to always overshadow the latter and as a result the top principal components carrying more variation tends to only include information about the differences of image content, which appears useless and confounding in our problem, whereas the components of higher order will bring more details that we are concerned. Therefore, the best feature scale, 50, is higher than the suggested *turning point of variance curve*, and also the accuracy will not take on decreasing trend very quickly as the feature scale increases.

Discussion. Despite the comparative outstanding performance of SVM, the intrinsic difficulty elaborated above that the information concerning type of font is dominated by that concerning image content will eventually prevent most of classifiers from being sufficiently accurate(e.g., accuracy of every class stands on 95% line). However, our luck is that in reality we do not always confront font recognition problems focusing only on one single image, but on the contrary, it is pieces of text consisting of many letters that we tend to consider.

Following this thought, we firstly train the SVM classifier still using independent images containing only one letter each, whereas we evaluate the outcome by generating accuracy of *text* font recognition, namely we predict the type of every letter in the sentence and then the prediction of the entire text will be decided by voting. By varying the length of text, we can examine the variation of classification efficiency.

To manifest the excellence of SVM in the new problem, we only utilize weak classifier with poor parameters, $C = 1$ and 10 principal components, and the text length varies from 1 to 100. The result is satisfactory, and is presented in Figure 11. When the length gets sufficiently long, say about 100, the accuracy of most classes have reach arbitrarily close to 100%, and those temporarily falling behind will eventually converge to this level. The remarkable result of text font recognition is thus guaranteed. In this way, we provide a practical manner in which we are to evaluate different classifiers, and also the justification of the accuracy that we get from them.

D. Decision Tree and Random Forest

Decision Tree. Decision tree is a traditional method which use entropy of information to extract the best feature, and generalize a tree of decisions. We

divide train and test set by 8:2, and the division was randomized.

A sample of a character has 4096 binary features after DAISY. We can expect that decision tree's performance won't be too bad.

To train a decision tree thoroughly is a waste of time. There are two parameters which decides how a decision tree will stop training:

1) MD. The max depth of a decision tree. Any slot of this depth is decided to be a leaf. Smaller MD causes bigger error,and bigger MD is a waste of time and space and may cause over-fitting.
2) MID. A threshold for min impurity decrease. If a slot's impurity decreases too little to overpass this value, it will be considered needless to split, and become a leaf. Smaller MID may cause over-fitting, and bigger MID might make the tree too short.

Fig 12 shows how MD and MID affects the accuracy of the decision tree on test set.

Random Forest. Random forest [24] consists of many different decision trees. Each tree is trained on random features and samples are assigned are assigned with random weights. Thus these decision trees are diverse and depend little on each other. As a result, the variance caused by the uncertainty of a single decision tree decreases notably. As for predicting, these trees vote and the majority wins.

Since we have found a best number of features (Md=30), all decision trees will be trained on 30 random features.

The performance of Random forest depends heavy on the number of decision trees N. Fig 13 shows how accuracy increases while N boosts. Since we have decided all the hyperparameters, we can evaluate the two methods by its accuracy, and its accuracy with recall one by one, as shown in fig 14.

E. K-Nearest Neighbour

K-Nearest Neighbour (KNN) is a traditional method for classification which does not require linear separability of data. It computes the distance between input and each samples, and use the k-nearest samples as vote to decide which class the input belongs to. The time complexity of simple KNN is $O(N + K \log K)$. In our scenario, The size of sample N is relatively small, therefore we do not need to do other performance optimization here. The result of KNN shows in table I.

TABLE I: Accuracy of random test set

K	Descriptor	
	Daisy	HOG
1	0.752	0.748
4	0.679	0.673
8	0.617	0.566
16	0.502	0.477
24	0.494	0.490
32	0.472	0.445

TABLE II: Accuracy of selected test set

K	Descriptor	
	Daisy	HOG
1	0.661	0.488
4	0.0685	0.601
8	0.696	0.595
16	0.667	0.601
24	0.661	0.583
32	0.631	0.548

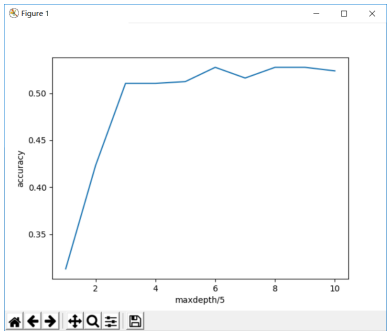
In general, the accuracy of KNN method is a concave function of K (firstly increase with K increasing, then decrease with K increasing). However, our result shows that the accuracy is monotonically decreasing as K increasing. This indicate that the result is almost depend on the nearest sample. Showing the nearest several samples out, we found that the nearest several samples are usually of the same character, but not the same font. This indicate that the similarity of same character is much higher than the similarity in the same font, which is apparently self-evident.

To overcome this problem, instead of picking the test set randomly, we use the character 's', 't' and 'u' (lower and upper case both) as test set, and get the result shown in table II. Note that this test set is only about 10 percent of the whole dataset, but we hope the samples could provide a little more information about the fonts.

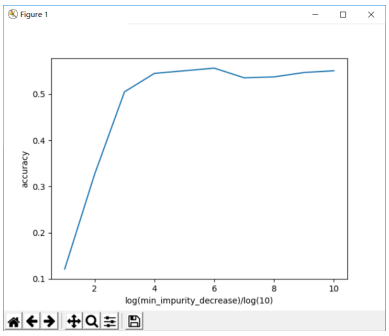
The result meets our expect now, and we could give the conclusion that the Daisy descriptor does extract more features about the font than HOG descriptor, and the similarity difference is the reason of the abnormal result shown before. All the tests are done in several seconds, the accuracy seems not bad given the performance is pretty good.

F. K-Means and Mean-shift

K-means and mean-shift are two clustering algorithms. These two algorithms performs both terrible in this problem, and we will give an explain about it. **Introduction.** K-means is a classical clustering algorithm. Given a set of pre-defined (or random)



(a) Respect to max depth, MD can be 30



(b) Respect to min impurity decrease, MID can be 10^{-6}

Fig. 12: Accuracy of decision tree on the test set with respect to different factors

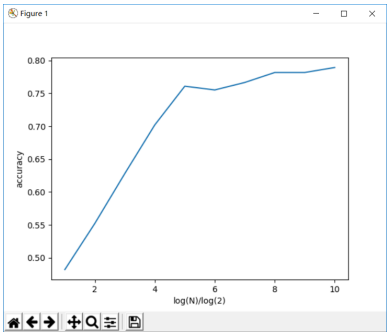


Fig. 13: Accuracy on the test set with respect to the number of decision trees. According to this figure, N can be 256.

	precision	recall	f1-score
CALIFORNIAN.csv	0.43	0.48	0.46
HARRINGTON.csv	0.47	0.54	0.51
BRUSH.csv	0.87	0.73	0.79
MODERN.csv	0.58	0.54	0.56
PAPYRUS.csv	0.47	0.45	0.46
EDWARDIAN.csv	0.52	0.49	0.51
FREESTYLE.csv	0.45	0.49	0.47

(a) Decision Tree

	precision	recall	f1-score
CALIFORNIAN.csv	0.73	0.70	0.72
HARRINGTON.csv	0.87	0.69	0.77
BRUSH.csv	0.97	0.89	0.93
MODERN.csv	0.65	0.75	0.70
PAPYRUS.csv	0.78	0.85	0.81
EDWARDIAN.csv	0.80	0.89	0.84
FREESTYLE.csv	0.71	0.73	0.72

(b) Random Forest

Fig. 14: Evaluation of decision tree and random forest. The average of decision tree is 55%, of random forest is 78%

class center points, K-means method iteratively classify the points using current center points and compute new center points for next iteration. The result of K-means highly depend on the starting points, and needs priori-knowledge about the clustering pattern of samples.

Compared with K-means, mean-shift is a far more stable, self-organizing, but also far more time consuming clustering method. It does not require starting points or number of classes. Instead, it needs a hyper-parameter called 'bandwidth', or 'radius'. Mean-shift was firstly proposed in 1975 [25] and got promoted in 1995 [26]. It treats each sample point as a class center, and all the samples closer than the 'radius' belongs to this class, and then move the center sample to the mean position. Intuitively, the samples close to each other will tend to gather together, and will not affect (or affected by) the samples far away. The result of mean-shift is some points far from each other in the sample space, and each point may contains samples from multiple classes. After mean-shift, using KNN method and voting, the test sample could easily be classified.

Experiment Result. The performance of K-means and mean-shift are both terrible. It could hardly perform better than randomly predicting. Mean-shift method gives the result that even the 'radius' is pretty small (0.1), most (about 90%) samples will clustered to the same point using HOG descriptor. As for Daisy descriptor, although the 'radius' is bigger (about 0.5), only about 70% samples will clustered to the same point. However, no matter how to change the radius in a proper range, there

will always be hundreds to thousands samples cannot be clustered (only 1-3 samples belong to the class). This indicate that there is one center which most samples are really close to, and many outliers scattered in the space. Therefore, we are confident to give the conclusion that clustering method is not suitable for this problem.

G. Deep Convolutional Neural Networks

Deep convolutional neural networks is the state-of-the-art technique in the field of computer vision, and considered to be the silver bullet of the traditional CV problem — classification and segmentation. Thusly, we want to test if DCNN can reach a much higher accuracy. We use the traditional CNN LeNet-5 [27] and state-of-the-art ResNet [28] to solve this problem.

Introduction. Convolutional neural networks use convolution operation to extract the feature of a small area of the image. The convolution operation of each small area shares the same parameters, which significantly reduces the number of parameters and make the training of high dimensional input possible.

LeNet is a typical, representative basic model of CNN. It uses three convolutional layer for feature extraction, two max-pooling layer for downsampling (or expanding respective field), and one dense layer for classification. The activation function of each convolutional layer was not clearly proposed in the original paper, but people usually uses sigmoid (or tanh) function as the activation function. This model was firstly proposed in 1998 and reached a reletive high accuracy in handwritten digits recognition. The learning ability seems to be insufficient nowadays, however, the small number of trainable parameters seems to fit this problem.

ResNet proposed a structure called 'residual block', which will use the sum of convolutional layer output and its input as output, instead of simply using the output of convolutional layer like VGG Net [29] (the most popular and the deepest DCNN model before ResNet). It makes the learning of identity mapping more easier (as the activation function always non-linear but usually go through the origin point), and the gradient back-propagation much more fluent, which means the depth of network could go much deeper without increasing the training difficulty. With ResNet, we could easily use a deep enough network to solve the problem, and the ResNet will just set the excess parameters to zero.

Experiment. We use modified LeNet-5, ResNet-18 and a shallower ResNet called ResNet-10. Note that even for LeNet-5, there are already 51397 parameters, which is far more than the number of instances. Considering the limitation of hardware also, we decide not to test deeper networks. The main structure of networks are all the same, therefore we just list the modified part here:

1) All the networks use ReLU-6[2] as activation function of convolutional layer
2) ResNet-10 is similar to ResNet-18 but use single convolutional layer in each residual block
3) Output of dense layer is 7 classes, not 1000 classes for ImageNet
4) Did not include other training techniques like batch normalization, randomly dropout (hardware limitation)

The tests included LeNet, ResNet-10 and ResNet-18 with/without data augmentation, and recorded the absolute training speed, convergence speed, test accuracy, and memory usage. The result shows in table III

Discussion. There are some conclusions worth discussing. The first one is over-fitting problem. As mentioned before, the number of parameters is far more than the size of training set, the over-fitting problem is really severe. As shown in fig 15, the accuracy on training set can go up to 98%-99%, but the accuracy on validation set is only about 70%-80%, which means over-fitting is a significant problem using DCNN.

The second one is that the data augmentation did not meet our expectation. As in fig 15 and table III, the data augmentation greatly slow the convergence speed, and force the ResNet-18 choose a much smaller learning rate (otherwise the loss do not decrease). However, it did not improve the accuracy on validation/test set, but even decrease it at an earlier time (with respect to training accuracy). This may be for the reason that the character integrity

[2]ReLU-6(x)=min(ReLU(x),6)

TABLE III: Test result of DCNN

Method		Training speed[1] (s/5epochs)	Convergence speed[2] (epochs)	Test accuracy	Model size[3] (MB)
LeNet-5	Without aug.	3	170	0.722	163
	With aug.	13	800+[4]	0.567	163
ResNet-10	Without aug.	27	30	0.740[5]	225
	With aug.	30	235	0.608	225
ResNet-18	With aug.	70	600	0.629	297

[1] Model saving time excluded. In fact, the model saving time is the most time consuming part for small model.

[2] The loss of training time usually still decreasing when stop training, but the accuracy of validation set is going to decrease also. The learning rate is 5×10^{-5} for ResNet-18 and 2×10^{-4} for other models. Optimizer is Adam [30]

[3] Model size is not the memory usage when training/predicting. The batch size and network structure (when back-propagating) are also important factors of memory usage.

[4] The '+' means that it did not meet our expectation on the training loss after 800 epochs.

[5] The highest accuracy of ResNet-10 in our test is 0.797, but we accidentally lost the model file, and the 0.740 here is another training result.

is also important rather than not important as we believed before.

The third one is that the DCNN does imporve the accuracy compared with the previous methods, and the training time does not increase much. But this method is still far from being called 'problem solved'. Despite the potential difficulty of this problem, the most probable reason is that the model is still shallow and the dataset is too small. Note that we do not test the performance on larger dataset, for the strange behaviour that some fonts will cause gradient become NaN, and tensorflow do not allow data pre-load into memory larger than 2GB, which cause the training time increases dramatically.

VI. Conclusion

We have tested most of the traditional, generic methods for pattern recognition and the state-of-the-art techniques in computer vision. All the performance are pretty bad compared with the related works. This shows the nontrivial of this problem. Actually, the most difficult part of this problem is recognize the similarity of small features while not being affected by the significant features. Thusly, this problem is similar in description, but dramatically different from semantic classification that hopes to extract significant features while keeping robust for differences of small features.

Our work also shows that the descriptor design is one of the most important part in computer vision, which is the shortage of our work. Besides, it shows that SVM and DCNN are two really powerful method in such classification problem, which is consistent with the fact that other methods are rarely used in CV area.

References

[1] Abdel Wahab Zramdini and Rolf Ingold. Optical font recognition from projection profiles. *Electronic Publishing*, 1993.

[2] Abdelwahab Zramdini and Rolf Ingold. Optical font recognition using typographical features. *TPAMI*, 1998.

[3] Yong Zhu, Tieniu Tan, and Yunhong Wang. Font recognition based on global texture analysis. *IEEE TPAMI*, 2001.

[4] Chen Guang, Yan Jianchao, Jin Hailin, Brandt Jonathan, Shechtman Eli, Agarwala Aseem, and Han Tony X. Large-scale visual font recognition. *CVPR*, 2014.

[5] Wang Zhangyang, Yang Jianchao, Jin Hailin, Shechtman Eli, Agarwal Aseem, Brandt Jonathan, and Huang Thomas S. Deepfont: Identify your font from an image. *arXiv preprint 1507.03196*, 2015.

[6] Xiao Chang, Zhang Cheng, and Zheng Changxi. Fontcode: Embedding information in text documents using glyph perturbation. *ACM Trans. Graph.*, 2017.

[7] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 1901.

[8] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 1982.

[9] Erkki Oja. Principal components, minor components, and linear neural networks. *Neural networks*, 1992.

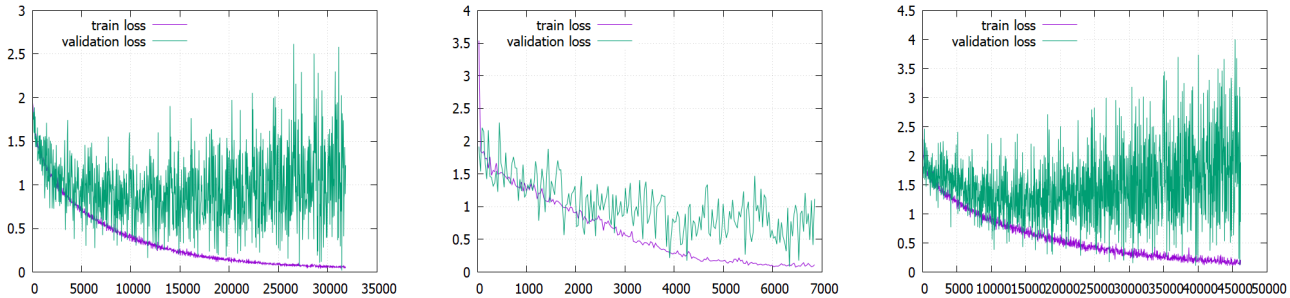
[10] Terence D Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 1989.

[11] David G Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

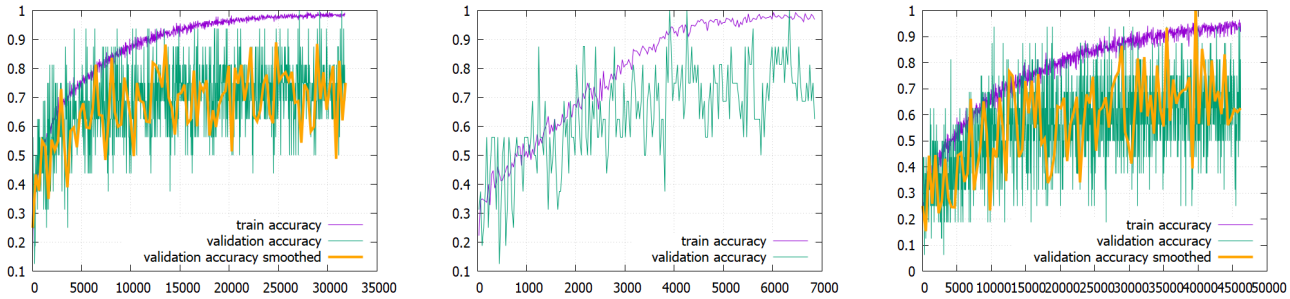
[12] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 2004.

[13] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE TPAMI*, 2005.

[14] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE TPAMI*, 2010.



(a) LeNet-5 Loss Without aug. (b) ResNet-10 Loss Without aug. (c) ResNet-10 Loss With aug.



(d) LeNet-5 Acc Without aug. (e) ResNet-10 Acc Without aug. (f) ResNet-10 Acc With aug.

Fig. 15: Loss and Accuracy curve of LeNet-5 and ResNet-10 with/without data augmentation (x-axis is training steps)

[15] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[16] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*. Citeseer, 1998.

[17] David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning*. Springer, 1998.

[18] Harry Zhang. The optimality of naive bayes. *AA*, 2004.

[19] Juan J Rodrïguez and Ludmila I Kuncheva. Naive bayes ensembles with a random oracle. In *International Workshop on Multiple Classifier Systems*. Springer, 2007.

[20] Vladimir N Vapnik and Alexey J Chervonenkis. Theory of pattern recognition. 1974.

[21] Olivier Chapelle, Patrick Haffner, and Vladimir N Vapnik. Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks*, 1999.

[22] Vladimir Vapnik and Sayan Mukherjee. Support vector method for multivariate density estimation. In *Advances in neural information processing systems*, 2000.

[23] Olivier Chapelle and Vladimir Vapnik. Model selection for support vector machines. In *Advances in neural information processing systems*, 2000.

[24] Tin Kam Ho. Random decision forests. In *3rd International Conference on Document Analysis and Recognition*, 1995.

[25] Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Transactions on information theory*, 1975.

[26] Yizong Cheng. Mean shift, mode seeking, and clustering. *TPAMI*, 1995.

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.