Learning the Way We Write: A Lightweight Automatic Adjustment of Neural Classifier of Handwritten Text to Individual Users

Milan M. Čugurović
Department of Computer Science
Faculty of Mathematics
University of Belgrade
Belgrade, 11000
Serbia
milan_cugurovic@math.rs

Mladen Nikolić
Department of Computer Science
Faculty of Mathematics
University of Belgrade
Belgrade, 11000
Serbia
nikolic@math.rs

Novak Novaković
Microsoft Development Center Serbia
Belgrade, 11000
Serbia
novakn@microsoft.com

Abstract

In this paper we propose a method of improving performance of a convolution neural network (CNN) classifier of offline handwritten text by adjusting to the writing style of an individual user. It does not require retraining or altering the base CNN classifier nor transfer of user's data to the server side, which results in a lightweight privacy preserving method which does not require any specialized deep learning framework or hardware on user's side. In the training phase, performed on the server side, base CNN is trained and writing styles of different characters are defined by clustering in the feature space of the CNN. In the adjustment phase, performed on the user side, an ensemble of alternative k-nn classifiers with a ligthweight Bayesian selection mechanism is constructed to identify specific writing style of the user for each character. Essentially, the ensemble construction mechanism we propose can be seen as an automated error-space analysis of the base CNN which is used for correction of its decision making. We tested our method on two relevant real-world datasets – NIST Special Database 19 and ETH Zurich Deepwriting dataset. We achieve up to 2.7% improvement in classification precision and get state of the art results on these datasets.

1 Introduction

The problem of automatic recognition of offline handwritten characters is a very practical problem, which is a part of the area of pattern recognition. Inspired by practical use, it develops both within academia as well as within the industrial sector. The industrial sector directly commercializes solutions to this problem by including them into devices like tablets, smartphones and the like. Cause of that it is very important to have precise classifiers that rarely make mistakes.

The pioneering attempt to automatically recognize handwritten characters dates back to the 1950s C.G. Leedham (1994). After this initial attempt, a few a group of researchers worked independently on this problem Plamondon et al. (2000). Within the software and hardware limitations of that

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

time, remarkable results were achieved. In the last ten years, the intensive development of neural networks has led to a shift in boundaries in many areas, including in the area of offline handwriting recognition. The results achieved by using convolutional neural networks exceed the results of all methods developed up to then Srihari et al. (2006) Pavarez et al. (2013) Awaida et al. (2012). Additionally, convolutional neural networks are also used as parts of complex systems that deal with the recognition of offline handwritten characters, mainly as the powerful feature extractors Durjoy et al. (2015) Elleuch et al. (2016) Ding et al. (2017).

Although convolutional neural networks have significantly increased offline handwriting classification accuracy, there is still plenty of room for progress. In this paper, we presented a fast, alphabet-independent and scalable method that improves pretrained CNN without its retraining.

The new approach is based on the idea of knowing the character's writing style each of the individual users. Presented improvement of pretrained offline handwriting CNN classifier is based on the dynamic monitoring of both his mistakes and successful predictions. By this monitoring, we created the so-called user writing history. Based on user writing history, a set of several models of KNN classifiers is formed, for different values of k and the reliability of both the base classifier and these models is evaluated (in a principled way, which will be described later). Based on these ratings, the method decides which one label to anticipate.

In this paper, Section 2 describes work related to this problem while the proposed method is fully described in Section 3. Section 4 provides experimental results, and conclusions are given in Section 5.

2 Related work

This section provides background about improving offline handwritten character recognition. In addition, an overview of the results achieved so far on the datasets we use is given.

In terms of publications related to improving the offline handwritten character recognition, there is only one paper that tends to improve the offline handwritten text classifier. However, the method presented in it doesn't focus on the writing style of the individual users, which is the basic idea that motivated our improvement method. The aforementioned publication Barend et al. (2019) propose a smaller augmenting network that can be re-trained on user device using his own data. A coarse-grained reconfigurable array processor was used to reduce the energy required for network retraining. A standard way to improve the performance of a set of classifiers is to integrate them using bagging, voting and other stacking techniques. Here we mention papers of this type, although this is not essentially a work on improving a particular classifier Govindarajan (2013) Mahreen et al. (2017) Ramy et al. (2009).

On the NIST dataset, several independent groups of researchers have published papers with different classification algorithms. A committee of seven CNNs obtains a precision of 88.12% Ciresan et al. (2011). For committee forming training is conducted both on the original and six preprocessed datasets (preprocessing is motivated by the purpose of handling different aspect ratios of the handwritten characters). Using a multicolumn deep neural network a precision of 88.37% was achieved Ciresan et al. (2012). This architecture was motivated by biological ideas, forcing sparsely connected neural layers like in a mammal's brain. Combining a convolutional neural network as a feature extractor and (linear) support vector machines as a classifier, an efficient method for improving the convolutional neural network is presented Darmatasia et al. (2017). By using the methods of k nearest neighbors, bagging and random forest classifier, a cheap feature selection algorithm was developed, and precision around 75% was achieved Cilia et al. (2018). Recently, on-device user customization of CNN which implies retrain of smaller augmenting network is proposed, showing a 3.5-fold reduction of the prediction error after doing user customization Barend et al. (2019).

There are currently no published results of the classification problem on the Deepwriting dataset.

3 Method

This section describes the proposed method for improving the CNN classifier of offline handwritten text. Note that while the idea behind the learning users handwriting style is rather simple, because of the complexity of the proposed method, we first give it an overview.

3.1 Method overview

The basic idea is to distinguish the reference modes of writing for each character. This is motivated by the intuition that each character can be written in finite significantly different ways. Separation the different writing styles of each character was achieved by clustering. Thus, for each of the characters, several clusters were obtained representing aggregated different writing styles of that character. This tends to separate the different writing styles of each of them.

When the proposed method is used, on the application set, for each user a set of characters that he or she authored is divided into two sets: an adaptation set and test set. On the adaptation set, we detect the user's writing style by memorizing his individual characters writing patterns. On the test set, the proposed method makes predictions by taking into account the user's writing style.

A sketch of the proposed method, which will be explained in more detail in the following sections, is given with:

- Images from the training set for the base classifier and validation set for the base classifier are grouped by labels (characters) and clustered within each group. These clusters represent the main writing styles for each of the characters.
- For each author of the application set:
  - The set of his images is stratified into an adaptation set and test set.
  - In the adaptation set, for each of the characters, the most similar writing styles (from already defined writing styles) are being identified, which makes the author's writing history.
  - Alternative classifiers (K nearest neighbor method) and their confidence vectors are created on the adaptation set.
  - For every instance of a test set (where the proposed method is used):
    - In addition to the base CNN prediction, alternative classifier predictions are also calculated.
    - Based on the confidence vectors of all the classifiers, the prediction of the most reliable of them is selected.
    - For each classifier, using the correct and its predicted labels, its confidence vector is updated.

A more detailed description of each step, as well as an explanation of terms such as writing history and the classifier confidence vector are given in more detail in the following sections.

3.2 Clustering individual character writing styles

The first step of the proposed method is to cluster the images within sets with the same labels. The main idea that motivated this step of the improvement method is the assumption that although there are variations in the ways of writing a particular character by individual authors, nevertheless this set of variations is finite. We clustered images of the training set for the base classifier and validation set for the base classifier.

We used the output of the next-to-last layer of the base CNN classifier as a set of attributes that characterize a handwritten character image Fiel et al. (2015). The next-to-last layer of the neural network gives representations of the input images in some new attribute space in which we expect the different characters to be well separated, while the last layer of the neural network only plays the role of labeling over those representations. Therefore, the last layer of the network is cut off, and for the set of attributes that describe the image from the input we used the values of the neurons of the next-to-last CNN layer Gatys et al. (2015) Johnson et al. (2016).

Due to the nature of our data, we used the K-means clustering algorithm Hartigan et al. (1979). Cluster centroids will represent the writing styles of the clustered character. The number of clusters for each label is given by the formula:

$$k = \min(30, 1 + \max(n/1000, 4)) \tag{1}$$

where n represents the number of images that were clustered. Previous heuristics were determined experimentally, with respect to the used datasets. The quality of clustering was evaluated by using the

K nearest neighbor algorithm (KNN) which was trained on the obtained centroids. Euclidean metric was used for clustering as well as for its evaluation. The number of neighbors in this evaluation belonged to the set $\{1, 2, 3, 4, 5, 7, 8, 9, 10, 15\}$. Clustering quality was evaluated based on the highest precision of the KNN method for various values of k. The motivation for this approach is the fact that the presented improvement method will, in the following, use the KNN algorithm in order to find the character most similar to the current one within the writing history.

3.3 Creating a user writing history

After clustering, for each character (label) a certain number of its characteristic writing styles were selected (centroids of clusters), which were represented by vectors of appropriate dimensions (the dimension of the vector is equal to the number of neurons in the next-to-last layer of the base CNN). The second part of the method, which is described in this section, relates to the model use pahse.

Like we said before, images each of individual users from the application set are stratified into two sets: an adaptation set and a test set. In the adaptation set, the method of improvement creates the writing history of each individual user. Based on the writing history, the base classifier will be improved during its usage. Based on the writing history, it is also necessary to conclude when the base classifier for the current user and his/her writing style make a mistake, and how to correct those mistakes.

The process of creating a user's writing history consists of the following. For the input image, we focus on the correct label as well as the label provided by the base classifier. Based on the output of the next-to-last layer of CNN classifier, we find the closest cluster among the clusters that correspond to the correct label (among the various writing styles of that character). We used the Euclidean metric for the search. This search attempts to identify the user's character's writing style with some of the predefined styles.

The writing history for each ordered pair *(predicted label, correct label)* remembers the average of the centroids closest to the output vector of the next-to-last CNN layer. The interpretation of this is as follows: when the base CNN classifier predicts a *predicted label* for a picture showing the *correct label*, on the basis of the output of the next-to-last layer of the base CNN, this ordered pair is assigned an appropriate cluster (a way of writing the correct character). Clusters assigned to the same pair may not always be identical. We consider their average to be a writing style for a particular character of a particular user.

3.4 Using user writing history

After creating user writing history, it is necessary to make decisions on which cases to trust the base classifier, and when to modify its predictions and change them based on the method K closest neighbors which was trained on the mentioned user writing history. The method we present, on the history of writing do not create one KNN model for correction, but a series of KNN models, where k taking values $2, 4, 6, 8$ and 10.

Based on the users writing history, a confidence vector of the base classifier and a confidence vectors of the KNN methods are created for each of the previous values of the k parameter. The classifier confidence vector is an array of beta distribution expectations, which evaluate the reliability of the classifier when predicting each of the labels. One beta distribution is created for each label l. The degree of confidence of the classifier for label l is represented by the mathematical expectation of the created beta distribution. This expectation is a function of parameters α and β. For each image of the adaptation set for which the classifier predicts label l, the parameters α and β of the beta distribution are updated, depending on whether the prediction is correct or not. Correct predictions increase the value of parameter β (thus increasing the mathematical expectation of beta distribution, that is the degree of confidence in the classifier when it predicts the label l), while negative predictions increase the value of parameter α (thus reducing the mathematical expectation of beta distribution).

The confidence vector of each of the classifiers is updated based on the original labels and their predictions for the images of the adaptation set. The KNNs training set is a subset of the user's writing history, consisting of vectors and their original labels of all writing history pairs for which the *predicted label* equals the current prediction of the base classifier.

In doing so, we created appropriate confidence vectors at the adaptation set, and we evaluate the reliability of both the base classifier and the k nearest-neighbor methods, for different values of k. Based on that, we determine which one classifier to trust during model use (we believe the classifier that for the given image has the highest reliability on the character he predicts).

When used, in a test set, instead of just one prediction that a base CNN classifier gives us, we get multiple of them simultaneously. For each image, based on the predictions of the base CNN classifier and the author's writing history, we obtain predictions of nearest neighbor methods for various values of k. If the base classifier predicts label l, within the writing history, we look at the style vectors of all the pairs *(predicted label, correct label)* for which the *predicted label* is precisely l. Using these vectors, for each value of k, we make the decision by the KNN method with respect to the correct labels. Based on the confidence vector of the base classifier (we use the value of expectation that relates to the prediction of label l), and based on confidence vectors of KNN methods for various values of k (we use the values of expectations related to their predicted labels) we decide which character to predict.

After the evaluation of the individual image, we update the parameters of the corresponding statistics for each of the classifiers. Depending on whether the classifier predicted the correct label or not, we are updating the beta distribution parameters which are used to evaluate the classifier's reliability on the predicted character. This corresponds to a real-world application, in which the user would have corrected his just-written character if the handwriting recognition application had failed to successfully classify it, and the application would have information about the correct label. In the absence of a correction, the model is sure to have successfully classified the character just processed. With this method of execution, the presented method achieves better results over longer, because the degree of its reliability is directly proportional to the size of the user writing history. This is a great advantage when using the presented improvement model in practice.

4 Evaluation

In this section, the results of the proposed method will be discussed. We carried out the experiments on two public databases: NIST Special Database 19 (American National Institute of Standards and Technology) and Deepwriting Dataset (ETH Zurich). In independent evaluations, our improvement method behaves very stable, and in the first of them increases the accuracy of the base classifier by approximately $2.3 - 2.5\%$. In the second dataset, the increase in precision of the base classifier is greater than 2.7%.

In addition to significantly increasing the precision of base CNN, other qualities of the proposed method should be considered. See Section 5 for detailed information.

Specifically, our source code was released on `https://github.com/MilanCugur/NNClassifierImprove`.

4.1 Used datasets

The idea of this paper is based on the adaptation of the base classifier to the individual users, and therefore a dataset with a large amount of data for each character each of individual users is needed (in order to learn the specifics that characterize each of them). This turns out to be a very big limitation and only a small number of datasets satisfy this property. As discussed in the introduction, the datasets NIST and Deepwriting were used in the paper. More information about them is given below.

4.1.1 NIST Special Database 19

The NIST Special Database 19 Patrick et al. (2016) contains handwritten characters collected from slightly less than 3600 authors in the approximately $810,000$ images, scanned from the appropriate forms, along with the corresponding labels. In addition, the database contains reference forms for possible further data collection and software tools for working with them.

The first version of this database was published as CD-ROM Grother (1995), and then re-released using a modern file format Patrick et al. (2016). The dataset contains binarized images created from 3669 forms and contains segmented, manually classified handwritten letters and numbers. These characters are presented as monochromatic images of 128×128 resolution while labeled with one of

the 62 ASCII classes corresponding to the English alphabet characters and Arabic numerals. The database is structured in five different ways, hierarchically organized according to different criteria. In this paper, the data used is structured by appropriate authors or classes.

Detailed information about dataset are given in the table 1.

4.1.2 ETH Zurich Deepwriting Database

The ETH Zurich Deepwriting DatabaseAksan et al. (2018) contains handwritten text annotated at the level of sentences, words and individual characters. About 300 authors wrote around $400,000$ characters.

It was created as an upgrade of the existing IAM Handwriting database Marti et al. (2002), which is here labeled at the level of individual characters. In addition to the data in this database, new data were collected with the help of web tools developed for iPod Pro. New authors, 94 of them, wrote the text Lancaster-Oslo-Bergen (LOB) Garside et al. (1986), since the authors in IAM Database also wrote that text. Characters in the database were initially given as online handwritten characters. In addition to the data, the authors provided software that could be transform database to offline variant. Based on the software provided, offline character images were created. Images in datased are labeled with 70 labels, which are lowercase and uppercase English alphabets, Arabic numerals, and additionally characters ' . , - ( ) / and space.

Detailed information about dataset are given in the table 1.

Table 1: Detailed information about used datasets

	Dataset	
Property	NIST	Deepwriting
Number of writers	3596	294
Number of images	$814,255$	$406,956$
Number of labels	62	70
Average number of images per user	226.43	1384.20
Average number of images per user $\times$ label	3.65	19.77

4.2 Images preprocessing

On both datasets preprocessing was inspired by the famous MNIST dataset LeCun (1998) and it was modeled on the work that describes the creation of the EMNIST dataset Cohen et al. (2017). In the mentioned paper, the conversion process consists of the following steps, in sequence: blurring images using Gaussian filter, cropping the character from the image, centering the previously extracted character, and resizing the image into a 28×28 pixel frame. Unlike the mentioned publication, we first cut the original image whereby the size of the margin around the character is set to one pixel. After that, a Gaussian filter is applied to the cropped character (the standard deviation of the Gaussian kernel is set to 1).

After cropping and blurring, the character is centered in a square frame. When centering, it is important to emphasize that the aspect ratio of the extracted character is preserved, because we expand the shorter dimension of the image with the empty space. This transformation from an image of dimension $w \times h$ creates an image of dimension $max\{w,h\} \times max\{w,h\}$. The original character height to width ratio is one of the specifics of the handwriting, so losing this aspect ratio would be a loosing of information. Finally, a square image of dimension $max\{w,h\} \times max\{w,h\}$, where w and h are original image width and height respectively, is converted to 28×28 pixels. This conversion was done using bi-cubic interpolation.

Datasets, created as described, are available in Zip format (as addition to this paper).

4.3 Datasets split

The dataset is divided into a training set for the base classifier, a validation set for the base classifier and an application set. Further, the application set is divided into two parts: an adaptation set and

a test set. When splitting the data into subsets, all images of one author belong to the same set. Considering the idea of the proposed approach, which implies learning the specifics of the author's handwriting style's on each individual character, a significant number of images are required for each individual character of each individual user. As the original dataset is quite unbalanced (in terms of the character count of each user), the division was made so that those who wrote more were transferred to the application set, while users with less written characters remained in the training and validation set of the base classifier.

Approximately ten percent of all images are in the application set, while the remaining images are divided into training and validation sets for the base classifier such that the number of authors in them is in a ratio of $9:1$. 90% of the images of each author of the application set make his adaptation set, while the remaining 10% of the images makes his test set.

Detailed information about datasets splits are given in the table 2.

Table 2: Breakdown of the number of samples in created splits

	Dataset	
Property	NIST	Deepwriting
No. of writers in train set for the base classifier	3057	242
No. of writers in validation set for the base classifier	339	27
No. of writers in adaptation set	200	70
No. of images in train set for the base classifier	$659,541$	$298,313$
No. of images in validation set for the base classifier	$73,209$	$33,484$
No. of images in adaptation set	$81,505$	$37,658$

It is important to point out that the NIST dataset is extremely unbalanced, in the sense that for each individual user there are many more images that are labeled with some of the numbers than with other characters. Despite this, the improved method presented in this paper has proven to be very successful. On this basis, it is realistic to expect an even better system behavior in the case of favorable datasets, or in realistic implementation.

4.4 Base CNN classificator

The base classifier whose improvement is presented in this paper is a convolutional neural network based on the *Caffe* architecture Jia et al. (2014).

4.4.1 Architecture

Our base neural network is trained on images of size 28×28, while its output has a Softmax function or a layer that predicts as a class, in the case of the NIST database one of 62 labels, and in the case of the Deepwriting dataset one of 70 labels. The base neural classifiers created for the two datasets considered in this paper differ only in the number of neurons of the last layer.

The convolutional part of the neural network consists of three successive blocks, each consisting of two consecutive convolutions, followed by Batch normalization layer, which is again followed by an Aggregation layer. The number of filters in the convolutional layers increases monotonically by blocks. Convolutions use a Rectified linear unit (ReLU) as an activation function. Aggregation is performed using the Max-pooling function, retaining the same image dimensions after aggregation. The convolutional part is followed by a fully connected block, which uses Dropout regularization. The complete network contains more than $350,000$ parameters.

4.4.2 Training and results

The basic neural classifier was trained by the Adam optimization method with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and with a learning rate of 0.001.

In the case of the NIST dataset, the base classifier was trained in 35 epochs in three series of 20, 10 and 5 epochs each. Each successive series uses a larger batch size. This kind of training is inspired by the actual work in this area Smith et al. (2017). The results achieved by the base classifier on the

NIST dataset are 88.36%, 87.39% and 87.35% respectively on the training set for the base classifier, validation set for the base classifier and application set.

The base classifier on the Deepwriting dataset was trained for 20 epochs, after which training was stopped by the early stopping technique. The results of such a trained classifier are are 87.65%, 83.12% and 82.12% respectively on the training set for the base classifier, validation set for the base classifier and application set.

4.5 Evaluation results

The evaluation method includes both data stratification and resampling techniques.

The stratified data split of each of the users from the application set into his adaptation and testing sets are of great importance. The stratification is performed with respect to the corresponding labels, thereby achieving approximately equal distribution of the labels in the adaptation set and in the test set. This avoids the situation of character occurrence in evaluation, where the corresponding label was not seen in the user writing history (in such a scenario, our model cannot give an improvement but behaves the same as a base classifier). This situation, however, cannot be completely eliminated, but its likelihood can be reduced. This partially eliminates the bad properties of the datasets we work with. In the case of an ideal dataset, or real use scenario, stratification would not be required. The stratified split of the application set images was performed by sending 90% of the images to the adaptation set, while 10% of them ended up in the test set.

In the evaluation, resampling is used so that for an arbitrary user precision is not only calculated once on the application set, but the same procedure is repeated independently n[1] times, each time with a stratified split of data into a section intended to create a user writing history and a section intended for the evaluation.

The presented method goes beyond the results reported so far on the NIST Special Database 19, using the transformation of images into resolution 28×28 promoted by the MNIST and EMNIST data sets. It is important to note that the data split we used does not follow the recommended data partition for the training and test set of the NIST dataset, due to the specificity of the proposed method (the division used in the evaluation create application set of those authors who have written the most characters, cause of the great imbalance and difference in the number of written characters between the authors of the NIST dataset and the fact that the proposed method requires a certain amount of data to learn information about the writing style of an individual user). Therefore, there is no easy way to directly compare our results with the previous ones. The presented improvement method achieves a precision of 89.60%, which outperforms the previously published results. This result was achieved by improuving a base network whose results were worse than the best published on NIST so far (by increasing the accuracy of the base network for 2.4%, we have exceeded the best published results on this dataset so far).

On the Deepwriting dataset, there are no published results of the classification problem so far, and therefore our results are state of the art. The presented improuvement method increases the precision of the base classifier for 2.72% thereby achieving a precision of 84.77%.

More detailed evaluation results (whereby the number of resamples was set to 10) are given in the paragraphs below.

During the evaluation, for each user of the application set, we monitored the precision of the base classifier as well as the precision of the improvement method. The number/percentage of images on which the base classifier correctly predicted the labels, the number/percentage of images on which the improuvement method gave the correct labels, and the number/percentage of images where at least one of them was correct were monitored. The last one is included for the purpose of assessing the quality of improvement method, which is a marginal case or scenario that could happen if we in any time knows which classifier to believe, whether the the base or one of the methods of the K nearest neighbor simultaneously performed with him. Previous represents the upper limit for the improvement method quality.

In the case of NIST dataset the model was tested on $81,685$ images written by 200 different authors. The precision of the base classifier on them is 87.24%, while the precision achieved by the improve-

[1] where n represents the resampling parameter

ment method on the same set is equal to 89.60%. The upper limit of precision achievable by the our method is 91.42%. There are 24 authors on whose the base classifier performs better than the presented improvement method, on 9 authors they have the same performance, while as many as on 167 of them a new improvement method has better performance than the base CNN. In the performed evaluation, the improvement method gives an increase in the precision of the base classifier of 2.36%. Theoretically, the highest possible increase in precision (by this evaluation) is 4.18%.

In the case of the Deepwriting dataset we tested our model on $37,672$ images, which was written by 25 different authors. The base classifier achieves an accuracy of 82.05%, improvement method of 84.77%, while the upper limit of improvement method is 86.38%. Our model increases the precision of the base classifier by 2.72%, while in theory the largest possible increase in precision equals 4.33%. On the images of 24 authors, our method achieves greater precision than the base network, while on the images of one author, the base network achieves greater precision.

5 Conclusion

In this paper, we present a conceptually simple yet powerful method, by which we were able to increase the precision of the base neural network up to 2.7%. The proposed method exhibits many positive properties, such as stability (in terms of showing good performance with multiple independent evaluations) and scalability (in terms of improving performance with more data), which is very important for real application because then we are expecting a really large amount of information about the handwriting of the device user. The presented method is also alphabet independent, which greatly expands its application domain. In this paper, we have been able to improve the neural network without re-training it by creating a system that adapts to it the right way, learning its mistakes and making the right predictions. Therefore, our system is very fast and does not require powerful graphics cards for its execution, which makes it suitable for installation on smaller devices, which again qualifies it for practical use.

The main contribution of this paper is the development of a new method that uses basic algorithms and machine learning techniques such as K-means clustering and K nearest neighbor classification, which outperforms the performance of leading classifiers of offline handwritten text. The proposed model of improvement at the time of use of the base classifier provides prediction by combining the prediction of the base classifier with the predictions of a series of auxiliary classifiers. This delegates decision making, thus creating space to isolate the handwriting style of each user. It is important to note that our work is based on learning the writing style of individual users and is the first work of this kind. No work before ours attempts to improve the performance of an offline handwritten text classifier by focusing on the writing style itself.

Future research may include meta-learning and few-shot learning techniques to prepare a base classifier for quick adaptation to each new user.

References

Patrick, Grother & Kayee, Hanaoka (2016) NIST Special Database 19 Handprinted Forms and Characters, 2nd Edition

Aksan, Emre & Pece, Fabrizio & Hilliges, Otmar (2018) DeepWriting: Making Digital Ink Editable via Deep Generative Modeling In *SIGCHI Conference on Human Factors in Computing Systems*: CHI '18, ACM, New York, NY, USA

C.G. Leedham (1994) Historical perspectives of handwriting recognition systems In *IEE Colloquium on Handwriting and Pen-Based Input*: 1-3

Plamondon, R., & Srihari, S.N. (2000). On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. IEEE Trans. Pattern Anal. Mach. Intell., 22, 63-84.

Srihari, Sargur N., & Xuanshen Yang & and Gregory R. Ball. (2006) "Offline Chinese Handwriting Recognition : A Survey."

Parvez, Mohammad & Mahmoud, Sabri. (2013). Offline Arabic Handwritten Text Recognition: A Survey. ACM Computing Surveys (CSUR). 45. 10.1145/2431211.2431222.

Awaida, Sameh & Mahmoud, Sabri. (2012). State of the art in off-line writer identification of handwritten text and survey of writer identification of Arabic text. Educational Research and Reviews. 7. 10.5897/ERR11.303.

Fiel, Stefan & Sablatnig, Robert. (2015). Writer Identification and Retrieval Using a Convolutional Neural Network. 26-37. 10.1007/978-3-319-23117-4_3.

Gatys, Leon & Ecker, Alexander & Bethge, Matthias. (2015). A Neural Algorithm of Artistic Style. arXiv. 10.1167/16.12.326.

Johnson, Justin & Alahi, Alexandre & Li, Fei Fei. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution.

Hartigan, J. A., & M. A. Wong. (1979) "Algorithm AS 136: A K-Means Clustering Algorithm." Journal of the Royal Statistical Society. Series C (Applied Statistics) 28, no. 1: 100-08. doi:10.2307/2346830.

Grother, Patrick J. (1995) "NIST special database 19." Handprinted forms and characters database, National Institute of Standards and Technology.

Marti, U. V., & Bunke, H. (2002). The IAM-database: an English sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition, 5(1), 39-46.

Garside Rodger & Stig Johhanson & Atwel Eric & Leech Georey (1986) "The tagged LOB Corpus: User's manual"

Cohen, G. & Afshar, S. & Tapson, J. & van Schaik, A. (2017). "EMNIST: an extension of MNIST to handwritten letters." arXiv preprint arXiv:1702.05373.

LeCun, Y. (1998). "The MNIST database of handwritten digits." http://yann.lecun.com/exdb/mnist/.

Jia, Y. & Shelhamer, E.& Donahue, J. & Karayev, S. & Long, J.& Girshick, R., ... & Darrell, T. (2014, November). "Caffe: Convolutional architecture for fast feature embedding." In Proceedings of the 22nd ACM international conference on Multimedia (pp. 675-678). ACM.

Smith, S. L. & Kindermans, P. J. & Ying, C. & Le, Q. V. (2017). "Don't decay the learning rate, increase the batch size." arXiv preprint arXiv:1711.00489.

Ciresan, D.C. & Meier, Ueli & Gambardella, L.M. & Schmidhuber, Juergen. (2011). "Convolutional Neural Network Committees For Handwritten Character Classification. Proceedings of the International Conference on Document Analysis and Recognition", ICDAR. 1135 - 1139. 10.1109ICDAR.2011.229.

Cireşan, Dan & Meier, Ueli & Schmidhuber, Juergen. (2012). "Multi-column Deep Neural Networks for Image Classification." Proceedings CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 10.1109CVPR.2012.6248110.

Palehai, Darmatasia & Fanany, Mohamad Ivan. (2017). "Handwriting Recognition on Form Document Using Convolutional Neural Network and Support Vector Machines (CNN-SVM)." 10.1109ICoICT.2017.8074699.

Cilia, Nicole & De Stefano, Claudio & Fontanella, Francesco & Scotto di Freca, Alessandra. (2018). "A ranking-based feature selection approach for handwritten character recognition." Pattern Recognition Letters. 10.1016j.patrec.2018.04.007.

Harris, Barend & Bae, Inpyo & Egger, Bernhard. (2018). "Architectures and algorithms for on-device user customization of CNNs." Integration. 10.1016j.vlsi.2018.11.001.

Sen Maitra, Durjoy & Bhattacharya, Ujjwal & Parui, Swapan. (2015). "CNN based common approach to handwritten character recognition of multiple scripts." 1021-1025. 10.1109ICDAR.2015.7333916.

Elleuch, Mohamed & Maalej, Rania. (2016). "A New Design Based-SVM of the CNN Classifier Architecture with Dropout for Offline Arabic Handwritten Recognition." Procedia Computer Science. 80. 1712-1723. 10.1016j.procs.2016.05.512.

Zhong, Zhuoyao & Jin, Lianwen & Xie, Zecheng. (2015)." High performance offline handwritten Chinese character recognition using GoogLeNet and directional feature maps." 846-850. 10.1109ICDAR.2015.7333881.

Ding, Haisong & Chen, Kai & Yuan, Ye & Cai, Meng & Sun, Lei & Liang, Sen & Huo, Qiang. (2017). "A Compact CNN-DBLSTM Based Character Model for Offline Handwriting Recognition with Tucker Decomposition." 507-512. 10.1109ICDAR.2017.89.

Govindarajan, M.. (2013). "Evaluation of Ensemble Classifiers for Handwriting Recognition." International Journal of Modern Education and Computer Science. 5. 11-20. 10.5815ijmecs.2013.11.02.

Ahmed, Mahreen & Rasool, Asma & Afzal, Hammad & Siddiqi, Imran. (2017). "Improving Handwriting based Gender Classification using Ensemble Classifiers." Expert Systems with Applications. 85. 10.1016j.eswa.2017.05.033.

Mohamad, Ramy & Likforman-Sulem, Laurence & Mokbel, Chafic. (2009). "Combining Slanted-Frame Classifiers for Improved HMM-Based Arabic Handwriting Recognition." IEEE transactions on pattern analysis and machine intelligence. 31. 1165-77. 10.1109TPAMI.2008.136.