# Learn the Way We Write: Automatic Adjustment of Handwriting Neural Classifier 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

## Abstract

In this paper, we consider improving convolution neural network (CNN) classifier of offline handwritten text. We focus on the style of handwriting each of the individual users, causing of its great variability and huge impact on the ability to successfully recognize written characters. We present fast, scalable and a no-retrain method for improving CNNs classifier. Using only basic machine learning techniques, such as K nearest neighbor classifier and K means clustering, we achieve up to +2.7% improvement in neural classifier precision, and get state of the art results on the dataset we use. We evaluate our method on two dataset: NIST Special Database 19 and ETH Zurich Deepwriting dataset.

## 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 of 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 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).

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, an alphabet-independent and scalable method which improves pretrained CNN without its retraining.

[*]poincare.matf.bg.ac.rs/∼milan_cugurovic
[†]poincare.matf.bg.ac.rs/∼nikolic

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

pass

### 2.1 Previous work in offline handwritten character recognition

pass

### 2.2 Previous work in improving offline handwriting classifiers

pass

## 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 classfier predict a *predicted label* for a picture showing the *correct label*, on the basis of the output of 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 is 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 user's 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 $\alpha$ and $\beta$. For each image of the adaptation set for which the classifier predicts label $l$, the parameters $\alpha$ and $\beta$ of the beta distribution are updated, depending on whether the prediction is correct or not. Correct predictions increase the value of parameter $\beta$ (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 $\alpha$ (thus reducing the mathematical expectation of beta distribution).

The confidence vector of each of the classifiers is updated based on the original labels and its 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 can give us a base CNN classifier, 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 the first of them, our proposed method 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 $814,255$ segmented, manually classified, handwritten letters and numbers. These characters are presented as monochromatic images of $128 \times 128$ resolution while labeled with one of the 62 ASCII classes corresponding to the English alphabet characters and digits. 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.

### 4.2 Images preprocessing

On both datasets preprocessing was inspired by 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: adding a

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 | $3.65$ | $19.77$ |

Gaussian blur, cropping the character from the image, centering the previously cropped character, and resizing the image in $28 \times 28$. 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 resolution of the character does not change, but the shorter dimension of the image expands 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 ratio would be a loosing of information. Subsequently, 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 a square image of dimension $28 \times 28$. This conversion was done using bicubic 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 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 $9 : 1$.

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 | $81,505$ | $37,658$ |

$90\%$ 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.

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, and 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 $28 \times 28$, 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 data stratification and resampling techniques.

The stratified data split of each of the users of 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 $n$[3] times, each time with a

[3]where $n$ represents the resampling parameter

stratified split of data into a section intended to create a user writing history and a section intended for the evaluation.

## 5 Conclusion

pass

## 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.