Research Proposal
Offline text-independent writer identification by learning the global feature vectors via triplet CNN

Davit Kvartskhava
dkvart17@earlham.edu
Department of Computer Science
Earlham College
Richmond, Indiana

1 ABSTRACT

Writer identification based on handwriting plays an important role in forensic analysis of the documents. Convolutional Neural Networks have been successfully applied to this problem throughout the last decade. Most of the research that has been done in this area has concentrated on extracting local features from handwriting samples and then combining them into global descriptors for writer retrieval. Extracting local features from small patches of handwriting samples is a reasonable choice considering the lack of big training datasets. However, the methods for aggregating local features are not perfect and do not take into account the spatial relationship between small patches of handwriting. This research aims to train a CNN with triplet loss function to extract global feature vectors from images of handwritten text directly, eliminating the intermediate step involving local features. Extracting global features from handwriting samples is not a novel idea, but this approach has never been combined with triplet architecture. A data augmentation method is employed because training a CNN to learn the global descriptors requires a large amount of training data. The model is trained and tested on CVL handwriting dataset, using leave-one-out cross-validation method to test the soft top-N, hard top-N performance.

2 INTRODUCTION

"Handwriting is a kind of behavioral biometrics [14]." Every person has a somewhat distinct handwriting style, which makes it possible to verify or identify a person based on their handwriting [1]. Manual forensic handwriting analysis is used by law enforcement agencies to identify the writer, and it plays a considerable role in investigations [7]. However, identifying a writer based solely on their handwriting requires a lot of human expertise and experience in addition to being very time-consuming. Hence, automating this process is a research topic of interest. The research on automating writer identification methods has also become relevant to analyzing historical documents as more digitized data is now available.

Signature verification can be viewed as a specific application of the writer identification task. However, in the case of signature verification, the problem space is different, as the main focus is on distinguishing between forged and genuine signatures [5]. It should be noted that because our training data set does not include forged handwriting samples, a limitation of this study is that it will most likely fail in case of a skilled forgery.

Senior Seminar, Earlham College,
2021.

The research in writer identification is usually divided into two sub-categories – on-line and off-line writer identification. In on-line writer identification, the dynamic information about the procedure of writing is preserved using specialized devices. In off-line writer identification, such information is not available and the only input is the handwritten text itself.

The approaches for solving the problem of writer identification can also be divided into two categories – text-independent and text-dependent methods. The text-dependent method requires the input to contain the same text as the target handwriting (or at least the same set of characters). In contrast, the text-independent method tries to solve the problem regardless of the content of handwriting.

In the last decade, Convolutional Neural Networks have become a popular choice for analyzing visual documents [9]. The groundbreaking work on object detection, OCR, face verification and many other successful applications of CNNs has revolutionized the field [10]. CNNs have also been successfully used in the writer identification problem [3, 13, 14]. Such approaches have set the state-of-the-art baseline in terms of accuracy of identifying the writers based on their handwriting [3, 13, 14].

This proposal concentrates on off-line text-independent writer identification using CNNs. The rest of this paper talks about (3) the approach that I am suggesting, (4) related work that has been done using CNNs, (5) design and implementation, (6) and the results of experiments.

3 RESEARCH GOALS

My approach is to train a Convolutional Neural Network to directly learn the global representations of handwriting samples in the Euclidean space. The goal is to optimize the embeddings using the triplet loss function. This method has successfully been applied to the task of face recognition [11]. Similar work involving Triplet CNNs has been done by Keglevich et al. [7]. However, their research approach was to combine the local feature vectors through different algorithms instead of directly learning the global descriptors. The local feature vectors are produced by feeding a CNN with low dimensional patches cut out of the same handwriting sample. Hence, a set of local descriptors characterizes each handwriting sample. There are multiple methods for combining these local descriptors into a global vector that represents the handwriting style of a given sample. On the other hand, global feature vectors can be directly produced by CNNs, if instead of small patches, CNN is fed with a sizeable window of handwriting sample. Tang and Wu [13] have researched methods for optimizing the global features without aggregation of local features, but the technique that they used did not involve triplet architecture. The motivation for learning global descriptors as opposed to aggregating local ones is that the retrieval of local features from the small patches of the handwritten text images leads to the loss of information that might be key to identify the author with high accuracy. Aggregation methods that combine local descriptors to the global ones are not perfect, and the information about the spatial relationship of the patches is lost. My research is unique in that I am training CNN with a triplet loss to learn the global descriptors to tackle the writer identification problem. The downside of feeding a CNN with large patches is that it requires more data for the model to reach a satisfiable accuracy. Hence, I am also employing a data augmentation technique to enlarge the training set.

4 BACKGROUND

This section describes different methods that have been used to address the writer identification problem using CNNs. Two main methods are described below: (1) training a CNN to classify the handwriting samples and (2) learning the feature vectors via CNN. In addition to that, section 4.3 reviews the methods that have been used to address the lack of training data.

4.1 Classification into writer classes

Convolutional Neural Networks have been used in two distinct ways to identify writers based on their handwriting. The first approach treats the problem as a classification task. CNN is trained through softmax loss function, where the number of output nodes corresponds to the number of users in the database. The output of each node signifies the probability that each user is the author of the handwriting. The shortcomings of this approach are that the network is not scalable and it needs to be retrained every time a new writer is registered in the database.

Xing and Qiao [14] have taken the approach mentioned above of directly training a classifier. Such a CNN outputs a vector of probabilities for a handwriting sample belonging to a specific writer in the database. Xing and Qiao extracted the patches from the lines of handwritten text. They used a specific architecture (multi-stream structure) of a neural network comprised of two dependent CNNs that share the features in some layers. The reason for using such architecture was to take advantage of the spatial relationship between different square patches. The input for this network was a pair of two adjacent patches.

4.2 Methods for obtaining encodings

A second approach for writer identification is to produce the feature vectors or encodings associated with each input image. This approach deals with the issue of scalability of the basic classifiers. The encodings are supposed to capture the unique features of the handwriting, so that the encodings themselves are enough to differentiate between two writers. This way, a feature vector can be produced for the handwriting whose author is not in the training dataset. After the feature vector has been generated, the final step is to compare it with other encodings in the database and find the one such that some measure of similarity between the encodings is minimized.

4.2.1 Encodings produced through classification.

There are different methods for obtaining encodings. An older approach starts by training a CNN with a classification layer with a task to learn to classify the handwriting samples into the writer classes [12]. The second step is to extract the penultimate layer of the network. This layer contains the features specific enough to a writer that feature vectors can be used to distinguish between handwriting samples from different authors [12].

Fiel and Sablatnig [4] used the method described above to extract the feature vectors for the writer identification. An encoding for an entire image of handwriting was obtained by averaging the encodings generated by the small patches. These encodings were later compared using Euclidean distance.

Christlein and Maier [3] took a similar approach to extract local feature vectors - they took the penultimate layer of a CNN as an encoding. For identification, they used cosine distance between global descriptors. They combined local feature vectors using different algorithms in order to produce a global descriptor for each handwriting sample. They compared the VLAD encoding to triangulation embedding. They also compared max pooling to sum pooling in the writer identification task. The input for the CNN was the small 32x32 patches that were randomly drawn from inside of the contours of the handwriting image.

Same method was used by Tang and Wu [13] but they trained the model using large image patches. These patches included on average 15 words. They also proposed the use of joint Bayesian technique instead of square distance for the identification task.

4.2.2 Directly optimized encodings.

Another method for obtaining encodings was devised in 2015 [11], and it significantly improved the benchmark for face recognition/verification. When applied to writer identification, however, it did not improve upon the baseline set by other approaches [7]. Below, I will review both of these papers.

Schroff et al. [11] published a paper in 2015 on learning unified embeddings for face recognition. The method that they proposed produced an algorithm with a 30% lower error rate than other known approaches. They started by training a CNN with the direct aim to optimize the encodings themselves, instead of treating the problem as a classification task. They mention that the downsides of the older approach "are its indirectness and its inefficiency". The algorithm starts by picking three examples from the data — an anchor, a positive example and a negative example. Then the triplet loss function is used to maximize the distance between the encodings of the anchor and the negative example, while at the same time minimizing the distance between the anchor and the positive example. This way, the network learns to encode the images in a way that the resulting feature vector accurately represents unique features of different faces. Schroff et al. also discuss the importance of choosing the best triplets for training and propose a specific algorithm for choosing such triplets.

Keglevich et al. [7] applied this recent version of obtaining the encodings to writer identification. Again, the objective was to learn the encodings of the handwriting samples where the square distance (L2 measure) between encodings obtained from two different classes is maximized and the same measurement for the identical classes is minimized. In this paper, they incorporated an interesting algorithm for extracting the patches. They retrieved the patches around the SIFT keypoints. As they claim, based on previous research, SIFT points are such that there is enough information around them for the network to learn useful encodings. After feeding the CNN with these patches, they aggregated the vectors from different patches into one encoding. For this process of creating one feature vector per entire image of handwriting, they use VLAD [6] encodings. This approach was tested on ICDAR 2013 dataset.

4.3 Methods addressing the lack of data

Tang and Wu [13] proposed a novel data augmentation technique because of the necessity of large amounts of data to train a CNN. All the previous research that has been done in this area has focused on training the CNN on small image patches; however, the problem of this approach is that when local features are extracted from patches, some details about a person's writing style are lost. Learning the global features requires a lot more data, so they first extracted the words from the images of handwritten texts and then randomly permuted each word in a line. As a result, they were able to accumulate thousands of handwriting images for each writer in the dataset. They reported the best results on the CVL dataset and near state-of-the-art on ICDAR 13.

Chen et al. [2] also pointed out that CNNs need a lot of training data to achieve satisfactory accuracy in real-world applications. Data augmentation techniques do generate more data, but the downside of using such techniques is the risk of overfitting to the repeated data. Instead, they proposed a semi-supervised deep learning algorithm that learns to extract the writing style features from the mixture of labeled and unlabeled data. The patches were obtained from the original images, and VLAD encodings produced global descriptors from the local feature vectors.

5 DESIGN AND IMPLEMENTATION

This research aims to improve upon the approach taken by Tang and Wu. The data augmentation technique and the idea of training a CNN to learn the global embeddings were adopted without significant changes. There are three key differences in comparison with their approach: 1) Instead of extracting the feature vector from the network's penultimate layer, I am using a CNN that directly outputs the feature vector. This network is trained with triplet loss function. 2) I am using the Euclidean distance as a measure of similarity between the feature vectors, as opposed to using the joint Bayesian technique. 3) In the data augmentation step, multiple patches are produced per handwriting sample. I am taking the median values across each component of the vectors associated with patches in order to produce a single vector per sample. My workflow consists of three main parts - data augmentation, training of the model, and evaluating the model. These parts are discussed in detail below.

5.1 Data Augmentation and pre-processing

The first step in the workflow is data augmentation and pre-processing, which is almost identical to the method used by Tang and Wu. Each handwriting sample goes through the same set of steps:

(1) Original handwriting sample.
(2) Sample is segmented into words (both CVL and IAM datasets already provide word segmentation). See Figure 1 (2).

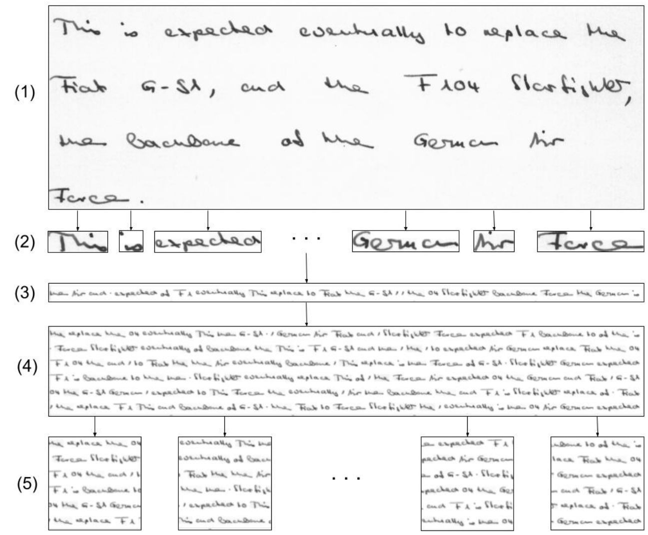

Figure 1: Data Augmentation Pipeline
(1) Original handwriting sample. (2) Word segmentation. (3) A random line produced by concatenating the words. (4) A page that was produced stacking 6 lines from the output of step (3). (5) Square patches given by splitting a page. In this example, the binarization has not yet been applied.

(3) The words from a single sample are randomly permuted into a line of handwriting. The words are centered vertically. See Figure 1 (3).
(4) Step 2 is repeated L times to get L lines of handwriting. These lines are concatenated vertically to produce a page. See Figure 1 (4).
(5) A page is then broken up into non-overlapping square patches. The remainder of the page is discarded. The resulting patches are resized to 224x224 pixels. See Figure 1 (5).
(6) Steps (4) and (5) are repeated N times.
(7) Finally we apply binarization. The patches are thresholded using adaptive Gaussian Thresholding with 37x37 kernel.

In the process described above, we have two hyper-parameters, L and N. L denotes how many lines are in a page and, therefore, in patches. L indirectly controls how many words are each patch. I used L=6 throughout the experiments, which ensured that there were at least 15 words per patch.

The number of patches per sample depends on N, L, and the sample's text size. If N=1, L=6 is set, each sample produces 20 patches on average. N controls how many pages are produced; therefore, it is roughly a factor by which the dataset is enlarged. A high value for N will result in a bigger training set but will also increase the risk of overfitting.

N=6 was used for the training set. As a result, over 50,000 patches were generated from the CVL dataset. The samples that go through this process yield patches that are ready to be fed to the CNN model. Enlarging the test set serves no purpose, but we still want to standardize the input for the CNN. So N=1 was used for the test set.

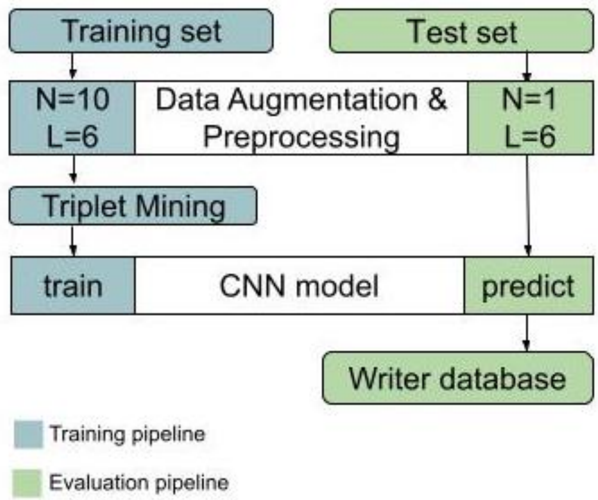

Figure 2: Workflow

5.2 Training, Validation and Test sets

The CVL database [8] consists of handwritten texts from 310 writers. This dataset provides a default split into training and test sets, with 27 and 283 writers in each, respectively. The writers in the training set contributed with 7 handwriting samples (1 in German and 6 in English), whereas the writers in the test set only provided 5 samples (1 in German and 4 in English).

I randomly chose the validation subset from the default test set. Validation loss can only provide valuable information if the validation set is disjoint (in terms of writers) from the training set. The model that minimized the loss on validation set was evaluated on CVL's test set. The validation set was not included in the final evaluation.

CVL also provides the word segmentation of all samples in the database. These images contain single words and were directly used in this research to generate patches as discussed in section 5.1.

5.3 Training

Initially, the architecture that I chose for the CNN was similar to the one used by Tang and Wu. During the experiments, difference between training and validation losses was quite significant, so I simplified the model to just 3 convolutional blocks followed by a single fully connected layer. Each convolutional block includes a 2D convolutional, batch normalization, max pooling and dropout layers. See Figure 3..

Batch normalization layers help speed up the training process and the dropout layers (dropout rate=0.4) alleviate the risks of overfitting. L2 regularization was applied to the fully connected layer, with lambda=0.0001.The model was trained for 15 epochs with batch gradient descend and Adam optimizer, with an initial learning rate of 0.0003. I implemented this CNN framework in keras with tensorflow backend.

I used relu activation function for all convolution layers. The dense layer was compiled with no activation function and the final 256 dimensional output vector was normalized. The model was compiled with triplet loss function. Given the embeddings of the triplets (anchor, positive and negative examples), triplet loss is calculated using the formula:

$$L = max(d(anchor, positive) - d(anchor, negative) + margin, 0)$$

The process described in section 5.1 yielded 224x224 grayscale patches that were fed to the CNN in batches of 256. I combined the semi-hard negative triplet mining with hard negative and hard positive mining. For the initial 10 epochs, an online semi-hard negative triplet mining strategy was used to pick the triplets from each batch. The training is accelerated by choosing those triplets where the positive is closer to the anchor than the negative, but the distances do not differ by a defined margin. Semi-hard negative triplets are the ones that satisfy the following property:

$$d(a, p) < d(a, n) < d(a, p) + margin$$

10 epochs of training with semi-hard negative triplet mining was followed by 5 epochs of hard negative triplet mining. Hard negative triplets are the ones that satisfy the inequality given below:

$$d(anchor, positive) > d(anchor, negative)$$

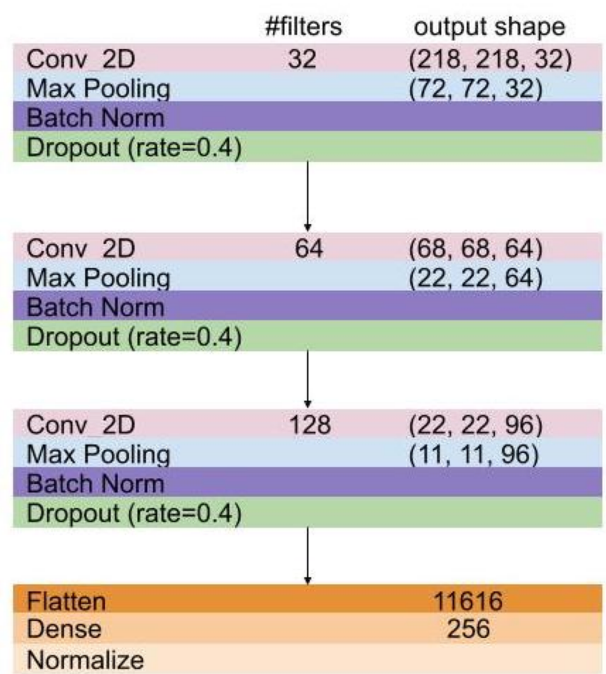

Figure 3: CNN framework

As I already mentioned, the triplets are chosen from each batch, not the entire dataset. Hence, the size of the mini-batches has an impact on the performance of the final model. On the other hand, the validation loss tends to increase as we increase the batch size, because there is a higher chance that we encounter harder triplets in larger batches. This poses an issue for evaluating which model performs best on the validation set. Therefore, two models that were trained with different batch sizes were evaluated on the test set.

N	Soft criterion	Hard Criterion
1	0.91	0.91
2	0.95	0.82
3	0.97	0.71
4	0.97	0.51
5	0.98	N/A
10	0.99	N/A

Table 1: Experimental results

I experimented with different CNN architectures (number of layers, number of filters in convolutional layers) as well as with different hyperparameters (initial learning rate, N - the augmentation factor, dropout rate). The model that performed best on the validation set was evaluated on the test set to obtain the final results.

5.4 Evaluation of the model

Once the training of the CNNs completed, pre-processing steps discussed in 5.2 were applied to the test set. A single page of handwriting was produced (with N=1, L=6) from each sample. Each page yielded 20 patches on average. These patches were then fed to the CNN to obtain the embeddings. The result of training the model with triplet loss is that the model learns the mapping from the input samples to the 256-dimensional Euclidean space, where the measure of similarity is simply the distance between the vectors.

For the evaluation to be comparable to other approaches, each sample needs to have a single embedding. However, each sample yields multiple patches and, therefore, multiple feature vectors. For this reason, the feature vectors from the patches of the same sample need to be combined to produce a single embedding. I took the median of the vectors (median value in each dimension across multiple vectors) and built a database of (handwriting sample - embedding) pairs.

I used the leave-one-out cross-validation strategy to evaluate the model on soft Top-N and hard Top-N criteria. This is one of the most popular approaches taken by other researchers to measure how well the model clusters the samples from the same class.

The first step in measuring soft Top-N, hard top N is finding N nearest neighbors of a single embedding. If all N neighbors have the same label as the anchor, then it is considered a hit for hard evaluation. For soft evaluation, at least one neighbor has to have the same label. Soft Top-1 and hard Top-1 always have the same value.

6 RESULTS

The method proposed in this paper was evaluated on CVL's test set. The embeddings for each handwriting sample were produced by combining the embeddings of the patches generated from those samples. The final feature vectors were evaluated on soft Top-N, hard Top-N criteria. The experimental results are given in Table 2. These results look promising, given that the CNN was trained on handwriting samples from only 27 writers.

The state-of-the-art methods proposed by Tang and Wu [13] and Christlein at al. [3] report far better results, but they used much more training data. Tang and Wu reported 0.93 hit accuracy for hard Top-4, whereas Christlein and Meier reported 0.945 for hard top-3. Both of these teams of researchers used ICDAR13 dataset, which contains samples from 100 writers in the training set.

Overfitting was one of the main issues during the training due to the lack of writer classes. Surprisingly, adding the random rotations during pre-processing worsened the results. The augmentation factor N=10 yielded the best results in the experiments. Increasing N further introduced more overfitting. The optimal mini-batch size for training with triplet loss turned out to be 256, and the dropout rate for each dropout layer at 0.4 produced the best results. I experimented with increasing the dropout rates in consecutive layers, proportional to the filter size; however, this approach did not yield better results.

7 CONCLUSION

This paper proposes the use of triplet architecture to obtain the global embeddings for handwriting samples to tackle offline text-independant writer identification. The features are learned from large patches of handwritings as proposed by Tang and Wu. The data augmentation method provides a CNN with enough input to be trained on large patches that capture a lot of information about the handwriting style. Triplet loss as a cost function provides more direct way to learn a mapping of a sample to its embedding.

Overfitting remains a big issue in the proposed method. Augmentation provides more samples in each writer class but it is the number of writers in the database that would make more difference.

ACKNOWLEDGMENTS

I would like to thank David Barbella and Charles Peck.

REFERENCES

[1] Marius Bulacu and Lambert Schomaker. 2007. Text-independent writer identification and verification using textural and allographic features. *IEEE transactions on pattern analysis and machine intelligence* 29, 4 (2007), 701–717.

[2] Shiming Chen, Yisong Wang, Chin-Teng Lin, Weiping Ding, and Zehong Cao. 2019. Semi-supervised feature learning for improving writer identification. *Information Sciences* 482 (2019), 156–170.

[3] Vincent Christlein and Andreas Maier. 2018. Encoding CNN activations for writer recognition. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE, 169–174.

[4] Stefan Fiel and Robert Sablatnig. 2015. Writer identification and retrieval using a convolutional neural network. In *International Conference on Computer Analysis of Images and Patterns*. Springer, 26–37.

[5] Luiz G Hafemann, Robert Sabourin, and Luiz S Oliveira. 2017. Learning features for offline handwritten signature verification using deep convolutional neural networks. *Pattern Recognition* 70 (2017), 163–176.

[6] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. 2010. Aggregating local descriptors into a compact image representation. In *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, 3304–3311.

[7] Manuel Keglevic, Stefan Fiel, and Robert Sablatnig. 2018. Learning features for writer retrieval and identification using triplet CNNs. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 211–216.

[8] Florian Kleber, Stefan Fiel, Markus Diem, and Robert Sablatnig. 2013. Cvl-database: An off-line database for writer retrieval, writer identification and word spotting. In *2013 12th international conference on document analysis and recognition*. IEEE, 560–564.

[9] YD Li, ZB Hao, and Hang Lei. 2016. Survey of convolutional neural network. *Journal of Computer Applications* 36, 9 (2016), 2508–2515.

[10] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. 2017. A survey of deep neural network architectures and their applications. *Neurocomputing* 234 (2017), 11–26.

[11] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.

[12] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2015. Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2892–2900.

[13] Youbao Tang and Xiangqian Wu. 2016. Text-independent writer identification via CNN features and joint Bayesian. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 566–571.

[14] Linjie Xing and Yu Qiao. 2016. Deepwriter: A multi-stream deep CNN for text-independent writer identification. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 584–589.