Signature Embedding: Writer Independent Offline Signature Verification with Deep Metric Learning

Hannes Rantzsch, Haojin Yang, and Christoph Meinel

Hasso-Plattner-Institute, University of Potsdam, Germany
hannes.rantzsch@student.hpi.de
{haojin.yang,christoph.meinel}@hpi.de

Abstract. The handwritten signature is widely employed and accepted as a proof of a person's identity. In our everyday life, it is often verified manually, yet only casually. As a result, the need for automatic signature verification arises. In this paper, we propose a new approach to the writer independent verification of offline signatures. Our approach, named *Signature Embedding*, is based on deep metric learning. Comparing triplets of two genuine and one forged signature, our system learns to embed signatures into a high-dimensional space, in which the Euclidean distance functions as a metric of their similarity. Our system ranks best in nearly all evaluation metrics from the ICDAR SigWiComp 2013 challenge. The evaluation shows a high generality of our system: being trained exclusively on Latin script signatures, it outperforms the other systems even for signatures in Japanese script.

1 Introduction

The handwritten signature is a widely employed method to verify a person's identity in our everyday life. It plays an important role in the legitimation of legal contracts, is used to authorize transactions of money, and serves as an evidence to the provenance of documents. As a part of these processes, a large number of signatures is verified daily, often by visual, human inspection. This verification is done only casually in most cases—especially in everyday scenarios such as at the supermarket checkout counter—and the signature's correctness is not questioned until legal issues arise.

This situation motivates the creation of automatic signature verification systems. Such systems are required to be robust and accurate due to the widespread and momentous employment of handwritten signatures in our society.

In this paper, we propose a new approach to writer independent offline signature verification. Our approach is based on deep learned similarity metrics. It is able to produce a soft classification decision, which entails an application independent system design.

Offline signature verification, as opposed to *online* signature verification, describes the scenario where no additional information about the process of creating the signature is available. Such information could include the position of the pen, the inclination of the pen, or the pressure exerted onto the pen at each point in time. The approach we present does not rely on such information. It operates on static images of signature, as they could be obtained, for example, by scanning a signature.

In addition, our approach is *writer independent*, meaning it can be employed independently of the author of the signature. The system can operate without being specifically attuned to the user whose signature should be verified and does not require an underlying database of users' signatures. Instead, it is provided with a small number of *reference* signatures when it is applied. The reference signatures are then compared to a *questioned* signature.

The method we propose is designed to handle *skilled* signature forgeries. This means the forger possesses knowledge about the original signature and has sufficient time to practice the creation of a hard to detect imitation.

Our system produces a *soft decision* about the genuineness of a questioned signature, meaning it can be employed independently of the application at hand. The system conveys a measure of certainty that the given signature is genuine of forged, allowing users to condition the interpretation of the result on the particular situation.

Our approach is based on deep metric learning: The system learns to embed signatures into a high-dimensional space, in which the Euclidean distance functions as a metric of their similarity. The distance between two embedded signatures can hence be utilized to confirm or refute that both have been created by the same author. Due to the pivotal role of the embedding, we name our approach *Signature Embedding*.

In this paper, we discuss how a system that produces such embeddings can be created using a deep neural network (DNN). We describe how the distances between embedded signatures can be employed in order to derive both hard and soft decisions.

In order to evaluate the system we created, we make use of the established evaluation metrics in this domain. We compare our results to the results of the ICDAR SigWiComp 2013 [1] challenge on offline signature verification. Our system compares favorably. It outperforms the systems that participated in the challenge in near to all respects.

As we want to allow for the best possible reproducibility of our results, all source code for the creation and usage of our system is openly available[1].

2 Related Work

In 2015, Hafemann et al. [2] provided a comprehensive literature review about the domain of offline signature verification. They found that the use of deep learning approaches is not yet widely spread in this community.

The state-of-the-art in this domain is hence defined by a method that makes use of handcrafted features: Yilmaz et al. [3] combined a histogram of oriented

[1] `http://hannesrantzsch.de/projects/signature-embedding`

gradients (HOG) and local binary patterns (LBP), which computes histograms of a pixels neighborhood. With this approach and support vector machine (SVM) classification, they achieved the highest score in the ICDAR SigWiComp challenge in both the years 2013 [1] and 2015 [4].

Khalajzadeh et al. [5] made use of a DNN for feature extraction on offline signatures. However, they did not consider skilled forgeries. Instead, they distinguished the signatures of the 22 users in their experiment. In order to do so, they trained one writer dependent classifier for each of the users.

A more general, writer independent approach has been proposed very recently by Hafemann et al. [6]. In order to allow for writer independent classification, the authors trained a DNN on a training set that does not include any authors from the evaluation set. The trained DNN is then used to obtain a feature representation of each signature in the evaluation set. As the DNN is trained as a classification task, no similarity metric can be obtained. Instead, an additional binary ("genuine" or "forged") classifier is trained for the samples in the evaluation set. The binary classifier provides a hard decision, rather than a soft decision. In addition to the writer independent component, the system of Hafemann et al. is equipped with a writer dependent component, which is able to leverage the feature representations obtained by the DNN.

An approach closely related to ours, though not based on deep learning, was proposed by Bromley et al. [7]. The authors trained a neural network to learn a similarity metric based on handcrafted features. The system they proposed is a writer dependent online signature verification system. However, their *Siamese* classifier architecture is the first application of similarity metrics to the problem of signature verification that we are aware of.

Schroff et al. [8] applied an approach similar to ours to the problem of face recognition. They used a DNN in order to learn a similarity metric of faces. Just like in our system, the training of their DNN is based on embedding triplets. However, rather than distinguishing different users, the Signature Embedding system needs to handle purposefully forged signatures for each of the users. Our system hence has to cope with a very low inter-class variability [2].

3 Method

The key concept of Signature Embedding is to learn a similarity metric for signatures. Signatures are embedded into a high-dimensional space, in a way that their Euclidean distance in that space can be employed as an estimation of their similarity. Hence, genuine signatures of the same author, which are most similar, are embedded close to each other, while forgeries are embedded further away from them.

In our system, a DNN is used for the embedding of samples. The DNN computes a function f_w, parameterized by its weights w. Thus, the similarity metric is defined as

$$M_w(\mathbf{x_1}, \mathbf{x_2}) = \|f_w(\mathbf{x_1}) - f_w(\mathbf{x_2})\|_2,$$

where M_w should be small if $\mathbf{x_1}$ and $\mathbf{x_2}$ are genuine signatures of the same author, and large otherwise.

The four major steps involved in the creation and application of our system:

- Preparing input data for training the DNN
- Training the DNN to compute the function f_w with the desired properties, and hence embedding the input samples
- Calculating the distance between embedded signatures
- Making a classification decision based on the distance

3.1 Preparing the Data

Prior to training the DNN, signature samples are augmented and preprocessed. Preprocessing involves cropping white boarders from the signature sample and resizing it to the input size expected by our DNN (192 × 96 pixel). The choice of the input size is based on a trade-off: larger input samples require a larger DNN (and hence more resources); smaller input sizes affect the recognizability of features.

Augmenting the training data is particularly important in order to apply deep learning technologies within the domain of signature verification. Creating a dataset of labeled signature samples requires a large amount of manual effort, as many authors are required to contribute numerous of their genuine signatures. In addition, obtaining skilled forgeries is even more laborious, as authors first need to practice to forge the signatures. Very large datasets are hence not available for offline signature verification. Consequently, we augment the available training data by applying different rotations and perspective transforms to the samples.

3.2 Training the Deep Neural Network

In order to embed the signatures into the high-dimensional Euclidean space, they are forwarded through a DNN. The DNN we employ is based on the *VGG-16* network [9]. Table 1 provides an overview of our network layout. It is slightly reduced compared to the original: Three convolutional layers, one pooling layer, and one fully connected layer have been removed. Furthermore, layer parameters, such as input size, output size, and the size of the kernel, have been adjusted.

As our reduced version of VGG still has comparably many layers, we pretrain the model to perform the simpler task of distinguishing authors of signatures. As a result, we obtain convolutional layers which are already trained to extract features related to the task of signature verification. Of the pretrained model, the convolutional layers are used to initialize the main model, while the fully connected layers are discarded.

The DNN should embed samples in a way that the Euclidean distance in the embedding space can be used as a similarity metric. This embedding is learned from relative comparisons [10]. In other words, the embedding function is not evaluated based on the absolute positions of embedded samples, but on their position relative to each other.

Table 1. Layout of the deep neural network

layer type	*kernel size, stride,* *padding width*	*output size* ($dim \times w \times h$) *or number of neurons*
(input data)	*(does not apply)*	$1 \times 192 \times 96$
convolution	$11, 3, 1$	$96 \times 62 \times 30$
convolution	$3, 1, 1$	$96 \times 62 \times 30$
max pooling	$2, 2, 1$	$96 \times 32 \times 16$
convolution	$5, 1, 1$	$128 \times 30 \times 14$
convolution	$3, 1, 1$	$128 \times 30 \times 14$
max pooling	$2, 2, 1$	$128 \times 16 \times 8$
convolution	$3, 1, 1$	$256 \times 16 \times 8$
convolution	$3, 1, 1$	$256 \times 16 \times 8$
convolution	$3, 1, 1$	$256 \times 16 \times 8$
max pooling	$2, 2, 0$	$256 \times 8 \times 4$
convolution	$3, 1, 1$	$512 \times 8 \times 4$
convolution	$3, 1, 1$	$512 \times 8 \times 4$
convolution	$3, 1, 1$	$512 \times 8 \times 4$
max pooling	$2, 2, 0$	$512 \times 4 \times 2$
fully connected	*(does not apply)*	1024
fully connected	*(does not apply)*	128

Therefore, each batch forwarded through the DNN consists of *triplets* of three samples. Each triplet consists of *anchor*, *positive*, and *negative* samples. Both anchors and positives are genuine signatures of the same author. The negative samples are skilled forgeries for the respective author or other authors' signatures.

We call triplets whose negative sample is a skilled forgery *hard triplets*. The ratio of hard triplets is determined by a hyperparameter passed to our system initial to the training.

The DNN is trained by computing a loss and propagating it back through the network. The complete process of embedding the samples and computing the loss is illustrated in Figure 1.

In order to obtain the loss, anchor, positive, and negative sample are embedded by the DNN. Subsequently, the distance between anchor and positive is compared to the distance between anchor and negative. The target of the loss function is to minimize the anchor-positive distances, while maximizing the anchor-negative distances. Hence, the Euclidean distances between anchors and positives, as well as between anchors and negatives are computed. Thereafter, the *softmax* function is employed as a ratio measure between these distances, normalizing the distances to real values in the range of 0 to 1 that add up to 1.

The distance between anchors and negatives is desired to be the larger of the two. Consequently, *mean squared error (MSE)* is used to compare the softmax

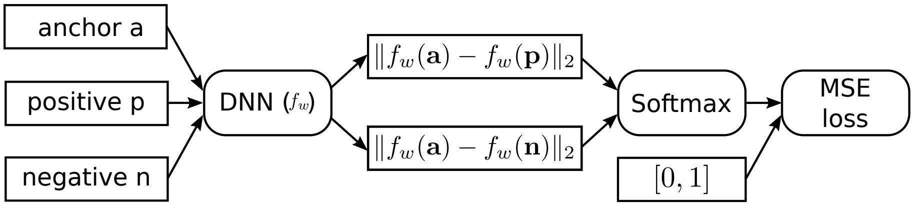
anchor a
positive p
negative n
DNN (f_w)
$\|f_w(\mathbf{a}) - f_w(\mathbf{p})\|_2$
$\|f_w(\mathbf{a}) - f_w(\mathbf{n})\|_2$
Softmax
[0, 1]
MSE loss

Fig. 1. After samples have been forwarded through the DNN, the loss is computed based on the softmax ratio between the anchor-positive distances and the anchor-negative distances.

ratio to the vector $[0, 1]$, producing a loss. This loss function is based on the function Hoffer and Ailon [11] proposed.

3.3 Calculating the Distance

The Euclidean distance between two embedded samples can be calculated as

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

In order to estimate whether or not a questioned signature is genuine, it is beneficial if more than one reference signature is available. In this case, the embedding of the questioned signature is compared to the centroid of the embedded reference signatures. Note that we use the squared Euclidean distance in practice, since distances only need to be compared to each other. This saves the calculation of the square root.

3.4 Making a Decision

After signatures have been embedded by the neural network and the distance of the questioned signature to its reference signatures has been calculated, the final step is the classification decision based on that distance. This decision can be done either in the form of a hard decision, applying a threshold to the distance in order to classify the questioned signature as “genuine” or “forged”; or in the form of a soft decision, producing a relative value that expresses the system’s confidence in either of the two hypotheses.

Setting a threshold in order to obtain a hard decision results in a trade-off between mistakenly accepted forgeries and mistakenly rejected genuine signatures. The decision about this threshold depends on the severity (*cost*) of each of these errors in the specific application scenario of the system. Consequently, systems that produce hard decisions are termed *application dependent* [12].

In recent years, however, *application independent* [12] systems, which produce soft decisions, have been considered more desirable in the signature verification community, most notably in ICDAR Signature Verification challenges starting from the year 2011 [13].

In our system, a soft decision is retrieved based on *log-likelihood-ratios*, as described by Van Leeuwen and Brümmer [14]. Therefore, a *score* s is computed for each distance, where a smaller distance results in a higher score. The log-likelihood-ratio $\mathcal{L}(s)$ for a score is then calculated as

$$\mathcal{L}(s) = \log \frac{P(s|\text{genuine signature})}{P(s|\text{forged signature})}.$$

$\mathcal{L}(s)$ can be interpreted as "expressing the degree of support that the raw score s gives to one or the other hypothesis"[14], where the hypotheses are "questioned sample is genuine" and "questioned sample is a forgery".

4 Evaluation

We evaluated our system with regards to the metrics that have been employed in the ICDAR SigWiComp2013 challenge. These metrics include application dependent as well as independent methods.

The application dependent metrics we use are *accuracy*, *false reject rate (FRR)*, and *false accept rate (FAR)*. All of these methods depend on the system to produce hard decisions. Consequently, the chosen threshold has an impact on these metrics. For example, changing the threshold for the benefit of an improved FRR will result in a worse FAR. In order to obtain comparable metrics over different systems, the threshold is set to the value that produces the *equal error rate (EER)*, the point where FRR and FAR are (about) equal. Hence, accuracy, FRR, and FAR in our evaluation refer to the value of each metric at EER.

Another commonly employed application dependent indicator of a system's performance is the ROC-curve. However, in order to allow for better comparability of results we employ the same metrics as the ICDAR challenge.

The application independent metrics we use are the log-likelihood-ratio cost (C_{llr}) and the *optimized* C_{llr} (C_{llr}^{min}), both of which are based on the log-likelihood-ratio described above.

4.1 Experimental Setup

In order to train and evaluate our system, we made use of the following datasets: We trained the model using the datasets *MCYT 100* [15], *GPDSsyntheticSignature* [16], and a subset of the *Dutch Offline Signatures* dataset from the ICDAR SigWiComp2013 challenge [1], which is the most recent SigWiComp dataset that is publicly available.

We evaluated the model using the signatures of 20 authors from the Dutch Offline Signatures that have been excluded from training. In addition, we used "Japanese Offline Signatures", also from the ICDAR SigWiComp2013 challenge, exclusively for evaluation.

The system is implemented using the deep learning framework *Chainer* [17]. We trained the DNN using an Nvidia GTX 980 GPU for 48 hours (55 epochs). As gradient update method, we used *Momentum SGD* [18]. The learning rate was initialized with 0.001 and halved every five epoch, where one epoch corresponds to processing one batch per author. The batch size was set to 180 triplets, or 540 samples. We made use of *weight decay* regularization with a factor of 0.001. The ratio of hard triplets was set to 90 %.

In the evaluation, we allowed the system to make use of 12 reference signatures in order to estimate their similarity to a questioned signature. This corresponds to the number of reference signatures provided per author in the "Dutch Offline Signatures" dataset.

4.2 Results

The results of our evaluation compare very favorable to the results of the ICDAR SigWiComp2013 challenge [1], which used the same datasets. Tables 2 and 3 show the Signature Embedding system in comparison with the three best ranked[2] competitors for each of the two tasks of the challenge.

In the first task of the challenge, the verification of Dutch offline signatures (Table 2), our system achieves the best scores in all metrics, except for the C_{llr}^{min}. In the second task, the verification of Japanese offline signatures (Table 3), Signature Embedding improves on the results in all of the employed metrics.

Please note that the evaluation of our system is based on a different subset of the datasets than the results we compare them to. The reason is that the complete datasets were not available to us anymore. Thus, as described above, we reserved part of the Dutch training dataset and the complete Japanese training dataset for evaluation purposes and did not use them in order to train our system. This process should provide a very good indication of the system's performance.

4.3 Discussion

Even though Signature Embedding was trained using Latin script signatures only, it performs better on the Japanese signatures than it does on the Dutch signatures. The ability to verify signatures of a script that the system has never been trained on indicates a very good generalizability of our approach. A possible reason why the system performs even better on the Japanese signatures is that the task of verifying Japanese signatures—at least on the given data—is easier. This explanation finds support in the fact that the overall results in the SigWiComp2013 challenge are better on this dataset as well.

[2] The "best ranked" systems from the challenge were selected based on their C_{llr}^{min} value. Other participants partly ranked higher in other values. Please refer to the original results in [1].

Table 2. Comparison of Signature Embedding to participants of ICDAR SigWiComp2013 Task 1: Dutch Offline Signature Verification

ID in [1] *or our system*	*Accuracy (%)*	*FRR (%)*	*FAR (%)*	C_{llr}	C_{llr}^{min}
2	76.83	23.70	23.10	0.880 048	**0.642 632**
Signature Embedding	**81.76**	**18.24**	**18.24**	**0.705 924**	0.653 741
4	74.93	25.19	25.05	0.979 237	0.698 044
3	75.56	24.44	24.44	1.086 197	0.706 733

Table 3. Comparison of Signature Embedding to participants of ICDAR SigWiComp2013 Task 2: Japanese Offline Signature Verification

ID in [1] *or our system*	*Accuracy (%)*	*FRR (%)*	*FAR (%)*	C_{llr}	C_{llr}^{min}
Signature Embedding	**93.39**	**6.66**	**6.57**	**0.421 014**	**0.316 642**
2	90.72	9.74	9.72	0.796 040	0.339 265
3	89.82	10.23	10.14	0.814 598	0.349 146
4	86.95	13.04	13.06	0.831 630	0.400 977

5 Conclusion

In this paper we presented a new approach to writer independent offline signature verification that is based on a deep learned similarity metric. Our approach compares two given signatures based on an embedding in a high-dimensional space, in order to confirm or to refute that both signatures are created by the same author. We showed how this can be achieved by training a DNN using a triplet-based loss function and discussed how our approach can be utilized in order to obtain an application independent, soft classification decision.

We demonstrated that the system we created outperforms the state-of-the-art from the ICDAR SigWiComp 2013 challenge on offline signature verification. Our results also show that our approach generalizes well, even to signature in a script unknown to the system.

In future investigations, we want to explore how our system can be employed in domains other than offline signature verification, such as the related problem of writer identification based on handwriting recognition.

References

1. Malik, M.I., Liwicki, M., Alewijnse, L., Ohyama, W., Blumenstein, M., Found, B.: Icdar 2013 competitions on signature verification and writer identification for on-and offline skilled forgeries (sigwicomp 2013). In: 2013 12th International Conference on Document Analysis and Recognition, IEEE (2013) 1477–1483

2. Hafemann, L.G., Sabourin, R., Oliveira, L.S.: Offline handwritten signature verification-literature review. arXiv preprint arXiv:1507.07909 (2015)
3. Yilmaz, M.B., Yanikoglu, B., Tirkaz, C., Kholmatov, A.: Offline signature verification using classifier combination of hog and lbp features. In: Biometrics (IJCB), 2011 International Joint Conference on, IEEE (2011) 1–7
4. Malik, M.I., Ahmed, S., Marcelli, A., Pal, U., Blumenstein, M., Alewijns, L., Liwicki, M.: Icdar2015 competition on signature verification and writer identification for on-and off-line skilled forgeries (sigwicomp2015). In: Document Analysis and Recognition (ICDAR), 2015 13th International Conference on, IEEE (2015) 1186–1190
5. Khalajzadeh, H., Mansouri, M., Teshnehlab, M.: Persian signature verification using convolutional neural networks. In: International Journal of Engineering Research and Technology. Volume 1., ESRSA Publications (2012)
6. Hafemann, L.G., Sabourin, R., Oliveira, L.S.: Writer-independent feature learning for offline signature verification using deep convolutional neural networks. arXiv preprint arXiv:1604.00974 (2016)
7. Bromley, J., Bentz, J.W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., Shah, R.: Signature verification using a ’’siamese‘’ time delay neural network. International Journal of Pattern Recognition and Artificial Intelligence **7** (1993) 669–688
8. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2015)
9. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
10. Schultz, M., Joachims, T.: Learning a distance metric from relative comparisons. Advances in neural information processing systems (NIPS) (2004) 41
11. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: International Workshop on Similarity-Based Pattern Recognition, Springer (2015) 84–92
12. Brümmer, N., du Preez, J.: Application-independent evaluation of speaker detection. Computer Speech & Language **20** (2006) 230–275
13. Liwicki, M., Malik, M.I., van den Heuvel, C.E., Chen, X., Berger, C., Stoel, R., Blumenstein, M., Found, B.: Signature verification competition for online and offline skilled forgeries (sigcomp2011). In: 2011 International Conference on Document Analysis and Recognition, IEEE (2011) 1480–1484
14. Van Leeuwen, D.A., Brümmer, N.: An introduction to application-independent evaluation of speaker recognition systems. In: Speaker classification I. Springer (2007) 330–353
15. Ortega-Garcia, J., Fierrez-Aguilar, J., Simon, D., Gonzalez, J., Faundez-Zanuy, M., Espinosa, V., Satue, A., Hernaez, I., Igarza, J.J., Vivaracho, C., et al.: Mcyt baseline corpus: a bimodal biometric database. IEE Proceedings-Vision, Image and Signal Processing **150** (2003) 395–401
16. Vargas, J.F., Ferrer, M.A., Travieso, C.M., Alonso, J.B.: Off-line handwritten signature gpds-960 corpus. In: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007). (2007)
17. Tokui, S., Oono, K., Hido, S., Clayton, J.: Chainer: a next-generation open source framework for deep learning. In: Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS). (2015)
18. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Cognitive modeling **5** (1988) 1