Università degli studi di Milano-Bicocca

Advanced Machine Learning

Final Project

Handwritten SigNature forgery detection: writer dependent vs writer independent approaches

Authors:
Di Lieto Gabriele - 874143 - g.dilieto@campus.unimib.it
Poveromo Marco - 830626 - m.poveromo@campus.unimib.it

February 21, 2022

Abstract

The goal of the following work is to present and analize the impact of different approaches to the task of handwritten signature forgery detection. There will be presented three models related to two kind of approaches: writer dependent (the model classify a user that already trained on) and writer independent (the model doesn't train on new user's instances), with advantages and disadvantages of the two approaches, also based on the number of signatures we have already available for new user.

1 Introduction

Handwritten signatures can be considered a form of biometric characteristic of every person, like fingerprint, face, dna, retina and voice. In the following, we will discuss how the verification process performed by graphologists can be automated by machines.

The handwritten signature forgery detection task consists in recognizing if a signature has been made by the rightful owner (**genuine**), or if it has been made by a malicious (**forge**). Recognizing the validity of a signature is a delicate task, as it concerns the safety and privacy of an individual, and it is very important in the forensic world.

The task can be approached in two ways, based on the methodology with which the signature is made, in particular it is possible to talk about **offline** signature, when the signature is done with classic methods (in this case only a signature is produced), or **online** signature, when the signature is performed with special tools (in this case, both the signature and a set of metadata like pressure or speed recorded during the process are produced).
Offline signature verification task can be also categorized in **writer dependent** and **writer independent**. If the model has to be retrained every time the system wants to add a new signature, is the case of writer dependent. On the other hand, if the system learns to model the similarity between forged and genuine signatures of a person never seen before, is the case of writer independent.

There are four main types of possible forgery that can be discovered. One of the the common forgery is the **blind forgery**, when the person forge the signature without knowing what the actual signature should look like. This type of forgery are most found in cases of money checks, expecially when the bank does not check the signature. An other type of forgery is the **simulation**, this is where the person can imitate the signature having a sample of the originals. In this case, the similarity of the forgery depends on the person's skills and time spent on practices. The last two cases are **tracing** and **optical transfer**, both are more easily discovered from the analysis of the paper, but they may be harder to check.

One of the biggest challenges of signature verification compared to other biometric characteristic like fingerprint, is the large variability between samples signatures of the same person. This is a problem specially when the **interclass variability** for one signature is high and the forged is a particular skilled signature.

2 Datasets

2.1 Data acquisition

The dataset which has been used in this project is an integration of three dataset public available, in particular:

- **CEDAR** is an offline signatures dataset for signature verification. Each of 55 individuals contributed for 24 signatures creating 1320 images. The database has 24 genuines and 24 forgeries available for each writer.
- **ICDAR 2011 - Dutch dataset** is an offline dataset of PNG images, scanned at 400 dpi in RGB color. It contains in total 2294 signatures of 69 reference writer. It has been used for the SigComp2011, the signature verification competition.
- **KAGGLE** is a dataset that contains genuine and forged signatures of 30 people. Each person has 5 Genuine signatures which they made themselves and 5 Forged signatures someone else made.

The integrated dataset was prepared by **mapping** the old ids of the three dataset into a new set of ids, and subsequently all the images have been preprocessed as described below. Since the images in the integrated dataset have different shapes, only the images with a good **quality** have been preserved. In particular, only the images with at least 128x256 pixels. In total, this dataset contains for each of the 109 individials both forged and genuine images.

2.2 Data manipulation

All the images have been preprocessed before being inserted into the new integrated dataset, firstly transforming them into **grayscale** to make the model insensitive to the color of the pen. This operation is seen in figure 1.

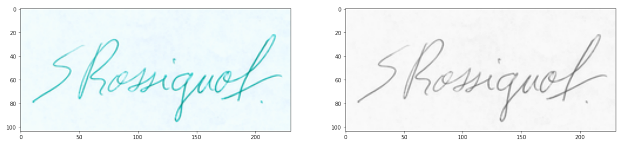


Figure 1: The original signature (left), and the grayscale signature (right)

Then a global non-binary **threshold** has been applied in order to make the background color of a uniform white, removing the paper noise while preserving the intensity of the lines in the signature. Various thresholding methods have been tried, including Otsu method [1]. Afterwards the images have been **inverted** so that the background where rapresented with 0's. This operation is seen in figure 2.

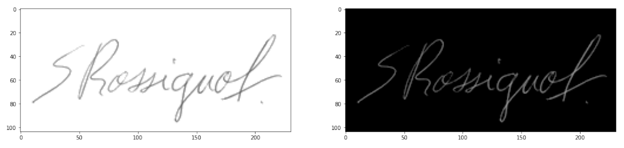


Figure 2: The thresholded signature (left), and the inverted signature (right)

Since the training of a neural network tipically requires images of the same size, but the dimensions of the signaures before the integrations goes from (36x72) to (1089x2857), images have been **resized** to (128x256) with a linear interpolation.

The last step in the data manipulation is the data augmentation. In order to increase the number of signatures for each user, all the images have been **rotated** by 10 degrees and -10 degrees. Before the rotation, the signature is cropped, and after the rotation the necessary padding is added to reach the previous 128x256 shape. An example of this rotation is present in figure 3.

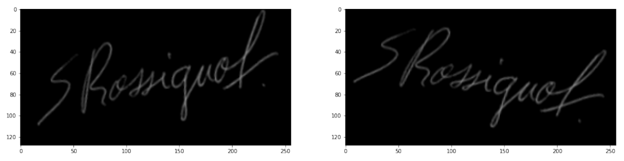


Figure 3: The rotation of -10 degrees (left), the rotation of 10 degrees (right)

3 The Methodological Approach

The main objective of this work is to find out a way to automatically adapt the model to a signature of a new user, and to evaluate the impact of the number of

genuine signatures that a user can provide. Let's suppose we want to verify a signature on a user that a network has never seen before. In a realistic case we have a relatively small set of official signatures of a user and we want to understand if a new signature is genuine or not. Since the dataset is small, it is necessary to start from a **basic model**, that will be refined for the single user.
It is possible to proceed in two ways: by **fine-tuning** the model by continue the training with new examples, or making **transfer learning** on the basic network, training only the latest layers while locking others. In the first case we have that the model can adjust the weights of the whole network, but layers are also influenced by previous trained examples and, since we expect the new dataset to be quite small, this could end up with poor results and, moreover, also a signature of a different user that the basic model has already trained on could be recognized as a genuine signature, so a malicious could use this trapdoor to elude the model. This means that the uncutted model should be combined with other algorithms to ensure that the target signature is related to the new user's signature, and not to others. However, these two methods turn out to be writer dependent, as they assume that the models have previously seen a certain number of a user's signatures before being able to decide whether that user's new signature is genuine or forged. We can actually think of developing a writer independent model, for example based on the approach used by SigNet [2], that is, using Convolutional Siamese networks, where the model can recognize the signature of a new user without ever being trained on that specific user, but only through at least one of his genuine signatures (called **anchors**).

The hypothesis is that the writer dependent model, being specific to the user, given a good number of signatures for the single user, can have better performances. If, on the other hand, the number of signatures of a user is low (or even 1), the writer independent model will probably perform better (it would be difficult to pull writer dependent models with few instances).

3.1 Writer dependent approach

3.1.1 Basic network

For this basic model a convolutional neural networks **CNN** have been used. The CNN has been used since it allows the network to get a representation that is robust and flexible to variance on the signature positions. Related works [3] also shown that CNN is an accurate approach for this problem.

What is expected from the model is the ability to extract features from the signatures in order to recognize, for each user, what patterns are related to the genuine signature. The callback function has been used to retrive the best model,in terms of f1_score. Since the output is a binary output, the last activation is a sigmoid and the loss function is a "binary_crossentropy".

3.1.2 Fine-tuning approach

In the fine-tuning approach, what has been done is, given a set of training instances of a new user, loading the basic model and training on the specific user. So the model's architecture is the same of the basic model's one. This approach could guarantee better performance in case there are a large number of instances of the new user, in this way the model can better adapt the internal representations and obtain excellent performance on the individual user. On the other hand, it is more difficult to train, as it requires many weights to be changed. Another problem is the possibility that the model recognizes images that belong to other users (those on which the basic model was drawn), for this reason it is possible to think of assisting the model with an identification network. 4.

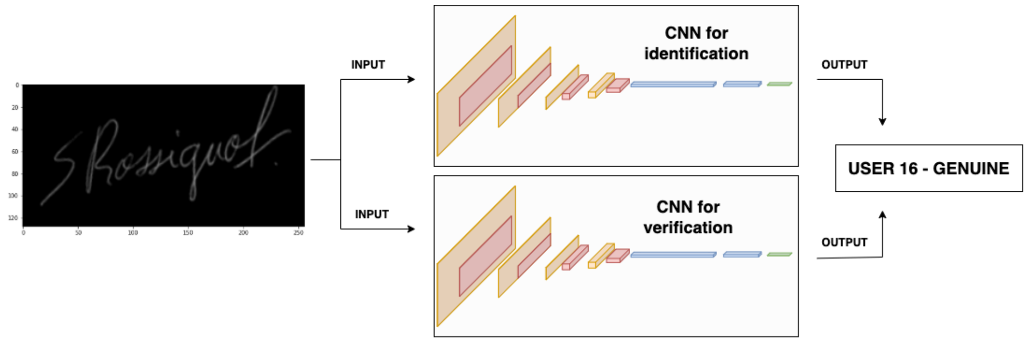


Figure 4: Writer dependent system

For the CNN related to the signature user identification, the output of the model must be multiclass, one class for each user present in the dataset. The architecture is similar to the CNN presented above with slight differences. It's composed of 4 convolutional layers, each of that is followed by a max-pulling layer. The last 2 are fully connected, and the last output layer has a softmax activation function, this is due to the multi-class problem. The complete system of the writer dependent approach is composed by the two CNN presented. The two networks are trained on the same dataset, but with different labels. In the picture 4 is shown the complete system.

3.1.3 Tailored approach

In the tailored approach what is done is to carry out a transfer learning operation from the basic network: given a set of training instances of a new user, we load the basic model, we cut the last layers (since the tasks are closely related), we introduce a new final set of layers of the model and we train only this set (locking the previous layers). It is expected that this approach could guarantee a good generalization, but should cost less, in term of time performances, than the previous one.

3.2 Writer independent approach

The signature verification task can also be categorized in writer indipendent, if the model can learn to distinguish a real signature from a forged one of a person never seen before. This approach is implemented through the **siamese network** and is closely related to the **one-shot learning** task, in which there is one or a few samples needed to learn. Related works includes [2].

The siamese network is composed of two identical CNNs that share parameters and weights. The CNNs are joined in a layer that calculates the similarity, with purpose of producing embeddings with a small distance d between positive pairs (genuine, genuine), and with a greater distance in the case of (genuine, forged) pairs. This layer is implemented with the **euclidean distance** as in equation 1.

$$d(r_0, r_1) = \|r_0 - r_1\|_2 \tag{1}$$

The loss function used is the **pairwise contrastive loss** [4] also called ranking loss, that forces representations to have 0 distance for positive pairs, and a distance greater than a margin for negative pairs. Being r_0, r_1 the pairs elements, y a binary label that indicates 0 for negative pairs and 1 for positive pairs, m the margin, the loss can be written as in equation 2:

$$L(r_0, r_1, y) = y(d(r_0, r_1)) + (1 - y)max(0, m - d(r_0, r_1)) \tag{2}$$

The complete siamese architecture is rapresented in figure 5.

The architecture used for the shared CNN is composed by convolutional layers and max pooling layers. An increasing percentage of dropout is also added after convolutional layers to reduce overfitting.

The weights and the bieses have been initialized with the **HeNormal initialization**, the weights are initialized according to the size of the previous layer, which helps in attaining a global minimum of the cost function faster and more efficiently [5].

The model has been trained with the Adam **optimizer** with a learning rate equal to $6 * 10^{-5}$, a **batch size** of 32 samples and a **margin** equal to 1 for the contrastive loss.
The model has been trained for 15 epochs and the **best model** in terms of vali-

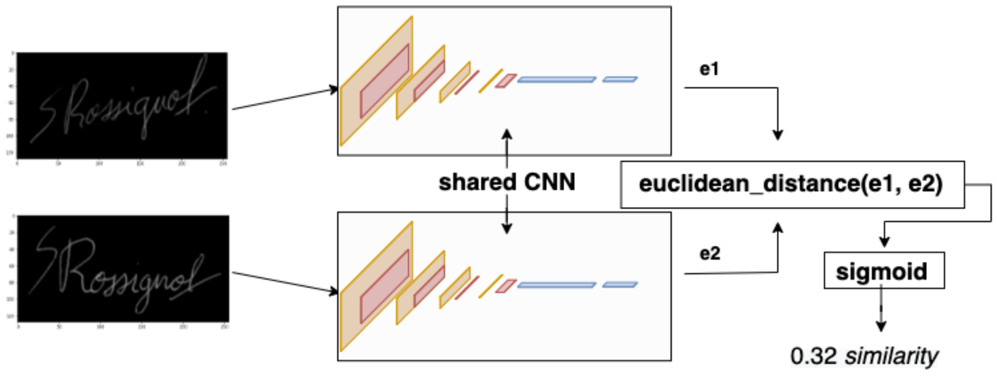


Figure 5: Siamese network structure

dation accuracy is then retrived by a callback.

The nework calculates the euclidean distance of the input pairs (r_0, r_1) and the sigmoid gives in output a distance measure $d \in [0, 1]$. To decide if the pairs (r_0, r_1) belongs to the negative class (0) or to the positive class (1), the basic strategy is to set a threshold t, for instance 0.5, that binarize the output. The strategy used in this project is to find the **threshold** $t* \in [0, 1]$ that maximizes the accuracy, as in equation 3. The threshold is calculated on the validation set with a grid search in $[0, 1]$ with a step-size of 0.01.

$$t* = \arg \max_{t \in [0,1]} Accuracy(t) \tag{3}$$

The threshold t* is then applied on the output of the siamese model and used to calculate the accuracy on the test set.

We expect that this model allows us to have better performances when classifying a new user with down to one single genuine signature available, and to be more flexible and fast (we don't need to retrain it every time we want to classify a new user).
Although it could have worse performances, compared to the writer dependent approaches, in case we have a bigger dataset, where retraining the model could led us to a better classification.
In case we have a bigger dataset for the new user (**more than one anchor**), we introduced an **election** approach to improve the siamese performances.

3.2.1 Election model

Given the model S, an instance to classify c and a set of anchors $A = A_1, A_2, ..., A_n$, with cardinality n, we defined an **election** model that classifies c as positive if

and only if $E_A = \sum_{i=1}^{n} S(A_i, c) > n/2$. So let's say that we set $S(A, c) = 1$ if and only if $E_A > n/2$, otherwise we set $S(A, c) = 0$.
That means that the election model will classify the instance as genuine if the model S finds a positive similiarity between c and more than half of the anchors in A, otherwise it will classify the instance as forged. Let L(c) be the correct label of c, we would like to understand which is the probability $P_A = P(S(A, c) = L(c))$, so which is the accuracy of the election model. We could suppose the comparisons to have a given degree of independence with respect to different anchors. Let's define $P_i = P(S(A_i, c) = L(c))$ as the probability that the instance c is classified correctly by S given A_i (we can assume P_i to be approximized by the accuracy **p** of the model). So if we assume that $P_i = p$ is independent from $P_j = p$ for each $i \neq j$, it is possible to check that

$$P_A = \sum_{k=\lfloor \frac{n}{2} \rfloor + 1}^{n} \binom{n}{k} p^k (1-p)^{n-k} \tag{4}$$

Moreover, it is possible to verify that (for odd cardinalities) if $1/2 < p < 1$ then $P_A > p$, so if the accuracy of the siamese model is greater than 0.5 we should expect that the accuracy of the election model increases given more anchors.
The independence is a strong assumption, but we decided to try to verify experimentally how, using the election, the accuracy of the model increases by testing it with increasingly cardinalities of A, and then to compare it to the results of the theoretical assumptions.

3.3 Training and evaluation methods

To train each model we divided the starting dataset D in two shards: D_1 and D_2, where D_1 with a ratio 80/20. It is important to underline that no user is both in D_1 (containing users from id 1 to 86) and D_2 (containing users from id 87 to 109).
The idea for the writer dependent approach is to build the basic model using D_1, and then simulate the task of adapting this model with a new users, using users in shard D_2.
To train the basic model we considered to divide D_1 in train, validation and test set. After that we use both the fine tuning model and the tailored model this way: for each user in D_2, we consider train, validation and test sets, and we train n new sub-models, starting the train from the basic model each sub-model. This way we can obtain for n new user, n new models adapted specifically for the single user.

For the siamese nework has been used the same dataset, a sample of pairs (genuine, genuine) and (genuine, forged) are generated for each user, with a particular **selection strategy** for the negatives. The easy negatives have been avoided, since their result loss is likely to be 0. This means for instance that a pair with two different users were not generated. The validation users are not present in the train set, this has been made for being sure that the validation set is a statistically significant sample of the test set, that also contains users never seen before by the model.

To evaluate the generalization capacity of the models:

- for the writer dependent approach, we get the n sub-models and evaluate them on the test sets of the relative users
- for the writer independent approach, we get the single model and we test it on all the test set of the users

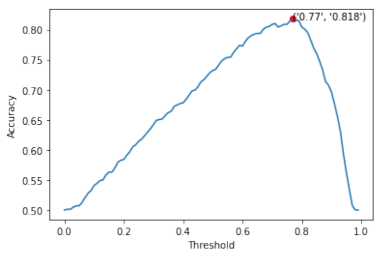
Then we sum the predictions obtained for each user, and we calculate measures that are independent from the single users. This way we can compare the three approaches the same way, by comparing measures obtained on the whole shard D_2.

4 Results and Evaluation

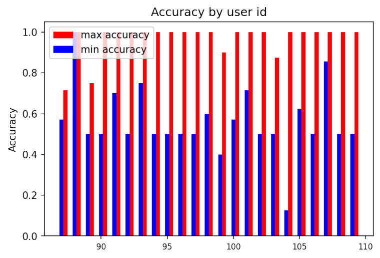
4.1 Siamese nework

For the siamese model, the **threshold t*** is calculated as seen in section 3.2. The best threshold is stimated $t^* = 0.77$ that on validation set produce an accuracy of 82%, as shown in figure 6 (a).

For all users in test set have been calculated the **anchors** that produced the **maximum accuracy** and the anchors that produced the **minimum accuracy**, using t^* to binarize the output. The result is shown in figure 6 (b).



(a) Threshold that maximizes the accuracy



(b) Min and best accuracy of all anchors

Figure 6

According with the figure 6 (b) an example of the best and worst anchors are visually showed.

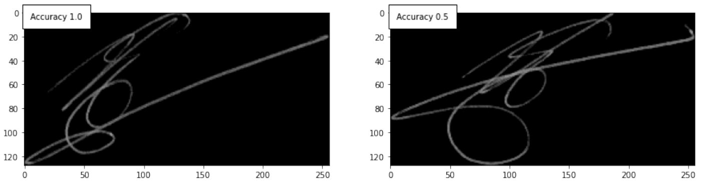
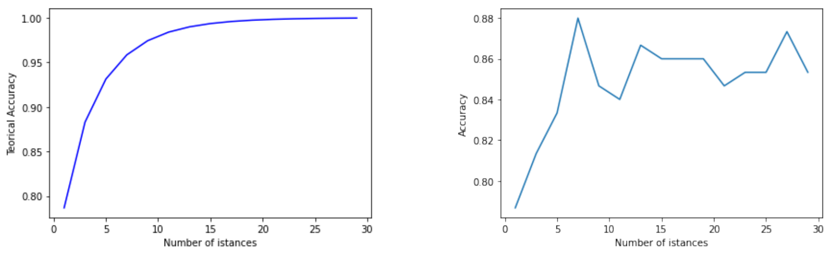


Figure 7: An example of a best anchor (left) and a worst anchor (right)



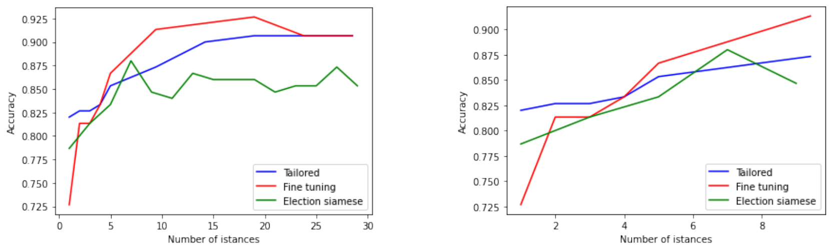
(a) Theoretical election accuracy

(b) Experimental election accuracy

Figure 8: Accuracy of the election model, with respect to the (b) approximated theoretical results, and (a) the experimental results with increasing anchor set

4.2 Comparison of the models

To compare the models the accuracy for different ranges of the number of instances available has been taken into consideration.



(a) Accuracy comparison in range **[1,30]**

(b) Accuracy comparison in range **[1,10]**

Figure 9: Accuracy of three approaches with respect to different ranges of instances available (as training set or as anchor set)

Model	Training time
Fine tuning model	968,5 s
Tailored model	39,7 s
Siamese model	None

Table 1: Training time per single user (the number of genuine instances considered were 5)

5 Discussion

The result in the figure 6(b) shows how the problem of intraclass variance is a crucial factor in the Siamese network. In fact, a user who does not sign consistently could produce anchors that are not useful for prediction. On the contrary, for most users, there are anchors that allow to get 100% accuracy, they are examples that adapt better to the intraclass variability. An example of the two anchors is shown in figure 7, where is visually seen that the two circles of the signatures are quite different, and the horizontal line is placed in an inconsistent way.

For the Siamese network, it can be seen that the theoretical accuracy (calculated with the expression (4)) has a similiar trend to the experimental election accuracy 8, even though the latter reaches a lower plateau. It could be a good starting point for further analysis. For example we could consider to unbalance election threshold (it is possbile to consider to increase/decrease the number of positive votes to accept an image as genuine) to see which measure improves.

Looking at results for the different models, we can than confirm part of our starting hypotesis on how to choose the right model to classify new signatures, given a set of genuine signatures of a user. It emerges that, to choose the right model, we have to take into consideration the following factors:

1. How many instances do we have for this user? If we have just one genuine signature, the fine tuning approach isn't the right choice (as it's possible to see in figure 9(b)).

2. Is time to train the model an issue, or can't you train? If time to train is critical (see Table 1), then the fine tuning approach is to avoid. If it is acceptable some time to train, the tailored approach could be the best choice. If you need a static model that doesn't need to train, the siamese is the right approach.

3. Can you generate a good amount of forgery signatures for the user? If not, then the only approach we can take is to go with the siamese/election model, since they just need genuine anchors.

6 Conclusions

We have presented different kind of models and a way to guide the possible choice of a model to classify a user's signature, given different factors to take into consideration. In real tasks, further questions could certainly arise. For example, there could be the need to minimize false positives (one could think for a Siamese model to modify the threshold based on this need), to evaluate how the model adapts to other types of forgery signatures (not only simulation forgery) and to understand if the proposed approaches can be applied for similar tasks, or assist more complex tasks (such as those related to the online signature).

Certainly, however, some ideas can be evaluated and possibly applied in real contexts to evaluate their robustness and effective flexibility. It would also be interesting to try other variants and different configurations of the models presented, which do not claim to generate an optimal model, but helped explore the pros and cons of different approaches to the problem of handwritten signature forgery detection.

References

[1] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[2] J. I. T. S. K. J. L. U. P. Sounak Deya, Anjan Duttaa, "SigNet: Convolutional Siamese Network for Writer Independent Online Signature Verification," *Elsevier*, 2017.

[3] A. D. J. Gideon, A. Kaldulna, "Handwritten Signature Forgery Detection using Convolutional Neural Networks," *Procedia Computer Science*, vol. 143, pp. 978–987, 2018.

[4] Y. L. Raia Hadsell, Sumit Chopra, "Dimensionality Reduction by Learning an Invariant Mapping," 2005.

[5] K. H. X. Z. S. R. J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *Microsoft Research*, 2015.