Offline Signature Verification using Inception V3 Neural Network

Salim Benkirane , Alice Breton , Paul Naert , Hocine Taleb
Department of Computer Science, Polytechnique Montréal, Canada
{firstName, familyName}@polymtl.ca

Abstract

The goal of signature verification is to authenticate an individual using his signature, an important biometric technique still widely used nowadays despite its flaws. Namely, signatures can be forged quite easily. Our project aims to detect these forgeries using deep learning techniques, thus automating this difficult task for human beings. We used a convolutional neural network with the Inception V3 architecture to do so, after having drawn a comparison between different architectures and optimizers. The results are quite promising for we have achieved an accuracy of 99,8% when classifying signatures, under certain conditions.

1 Introduction

Authentication is a crucial aspect in security, consisting in verifying someone's identity. One of the most widely used means to do so are handwritten signatures, which are marks drawn by individuals as proof of their identity. A good signature must have two main characteristics :

- Easy to reproduce for the genuine person ; it should not vary a lot between multiple signatures of the same person, and of course be easy to remember for him/her.
- Hard to reproduce for the fraudster ; a reproduction of the signature must be very easily distinguishable among genuine signatures.

Unfortunately, these two characteristics are quite hard to find : the same individual often slightly changes his/her signature without noticing ; two signatures are never exactly the same even though they both stem from the genuine author. A minor change can occur due to the shaking of one's hands for instance. Besides, a fraudster will try to reproduce the signature as precisely as he/she can, making it very hard for someone to distinguish it among genuine signatures. Goals of fraudsters are of course against ours, which is often the problem in security.

In other words, the challenge here is to tackle two main problems:

- High intra-class variability : changes among someone's own signatures.
- High inter-class similarity : attempt by fraudsters to get as close as possible to the real signature.

2 Related work

Some papers already tried tackling this problem in their own manner, always using deep learning methods but with different architectures.

In 2014, the African Journal of Computing & ICT published a paper by [**?**] that proposed to use artificial neural networks to classify forged and genuine signatures. They simply used a multilayer perceptron, without giving much detail on how they conceived and tuned it.

Alvarez et al. proposed another architecture in 2016 [**?**] ; they used convolutional neural networks to extract interesting features from the images and classify them. Indeed, CNN are known to behave well in presence of images. They used a VGG16 architecture, which is a deep one ; it can be quite long to train it. To tackle this issue, they used transfer learning, meaning that they used pretrained layers of the CNN and tuned the others. This works because the lower layers of a CNN behave in the same way in general, extracting the same features from images databases, so using a CNN trained with ImageNet for instance can save some precious time.

In 2018, a study [**?**] appeared in Applied Sciences showed another way to classify the signatures. They used a combination of a convolutional neural network and an autoencoder. The lower layers of the CNN were used as feature extractors, dropping the fully connected layers that follow. Indeed, only these feature extractors were useful, as they were feeding the autoencoder with interesting attributes of the signature images ; the autoencoder did the job of classification afterwards.

3 Methodology

3.1 Different forgery detection systems

There are various types of problems related to forgery detection, we are going to tackle the following.

Given a set of labelled signatures, genuine and forged, of different people, our network should be able to detect if a new signature is genuine or not. As an example, if the new signature is Alice's forged by Bob, the network is trained with Alice's genuine signatures and other forgeries generated by Bob.

Other variants of the problem is to train the network with pairs of signatures. The first one is always genuine, the second one can be forged or authentic. Given two new signatures, the model should decide whether this couple is genuine-genuine or genuine-forged.

We will focus on the former problem.

3.2 Convolutional Neural Networks

As we have to deal with images and computer vision, the natural solution that can come to one's mind is to use convolutional neural networks. Indeed, one goal in computer vision is to extract interesting features from images that will help us classify them rightly ; we could then grossly decompose the work of an algorithm in two parts, the first being this feature extraction process and the second the classification. To extract these useful features, research began by hand choosing them, which is difficult and time consuming. The idea was then to feed the neural network with raw input, and let it extract the features by itself ; but it does not solve the time consuming problem, among others. This is where the convolutional neural networks come in.

Their main characteristics are :

- The first layers can extract interesting features from the raw input, letting the final fully connected layers classify the input.
- Shift invariance: features are detected in the image regardless of their position, corresponding with the intuition that humans can recognise an object disregarding of its location.

Convolutional neural networks differ in their architectures, and do not yield the same results depending on theirs (number of layers, how they are connected to each other and so on).

3.3 VGG16 and Inception V3 Architectures

In our work, we drew a comparison between two main architectures used when designing convolutional neural networks; VGG16 and Inception V3. While the former has already been used in [**?**], the latter has not, to the best of our knowledge.

The VGG16 architecture was introduced by [**?**] and is well represented in figure 1. The use of very small convolutional filters (3x3) and deep weight layers (16) achieved very good results on the ImageNet Challenge 2014.

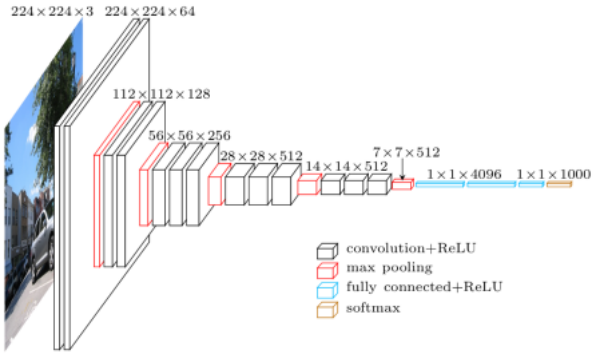

Figure 1: A visualization of the VGG architecture.

The Inception V3 architecture on the other hand, arrived first in ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2015, and was introduced by [**?**]. Making a bigger model to improve performance (bigger meaning more layers and more neurons in each layer) can bring some problems such as increasing needs for computational resources and overfitting. The main idea of the Inception architecture is to create deep networks without augmenting the need for computational resources. This is done through sparsely connected layers called "inception layers".

The results made us opt for an Inception V3 architecture, as it has yielded the best accuracies (see Section 4).

3.4 Dataset

We used the dataset from the International Conference on Document Analysis and Recognition (ICDAR) 2011 SigComp international signature verification competition [**?**]. It contains online and offline signatures (of which we only use offline ones) for both Chinese and Dutch signers.

The dataset is split into a training set and testing set of nonoverlapping IDs. The training sets include a total of 942 images for 10 IDS, with about 50 genuine signatures and 41 forgeries for each ID, Chinese and Danish sets combined. The testing sets had a lot more images, 2529 to be exact split between "reference" and "questioned" sets. The first one contains known genuine signatures and the "questioned" set includes genuine and forged signatures.

We also added our own signatures, with each member of the team producing 10 genuine signatures and 10 forged ones for each other member, counting for a total of 40 genuine and 120 forged signatures.

Finally, to have a bigger dataset for training, we merged all the images and split them afterwards into training, validation and testing sets.

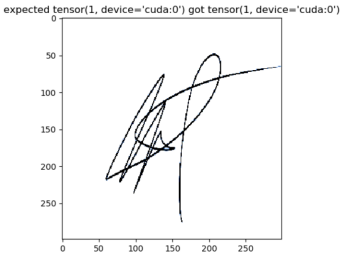

Figure 2: A genuine signature we added to the dataset.

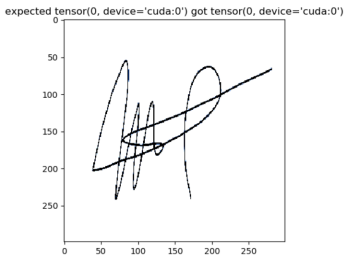

Figure 3: A forged signature we added to the dataset.

3.5 Preprocessing and training

One main advantage of convolutional neural networks is that they require minimal preprocessing work, but minimal does not mean none existing. The first task was to normalize images so that they all have the same size. The use of 224x224 images was necessary to feed the VGG network, and 299x299 for the inception network.

We then wrote Python scripts to merge our datasets into one (to be split afterwards) and create a CSV file that will be managed through the library Pandas.

Finally, using PyTorch mainly, we tested different architectures (VGG16 and Inception V3) and optimizers (SGD momentum, Adam and Nesterov) for which we yield the results in Section 4. The use of pretrained networks (for the first convolutional layers) greatly reduced training time; this is a common way of doing called "transfer learning".

The GitHub project can be found at https://github.com/Hocine958/INF8225-Projet for more details.

4 Results

4.1 Models comparison

Regarding our signatures classification problem, we compared the results of VGG16 and Inception v3 models with mainly three different optimization methods. We considered stochastic gradient descent SGD, Adam stochastic optimization method and Nesterov momentum, for a total of six different architectures. The results for training and validation accuracy after 5 epochs are shown on the figures 2 and 3.

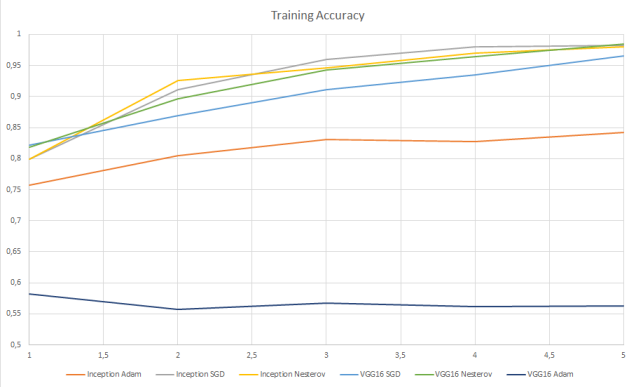

Figure 4: Training accuracy models comparison.

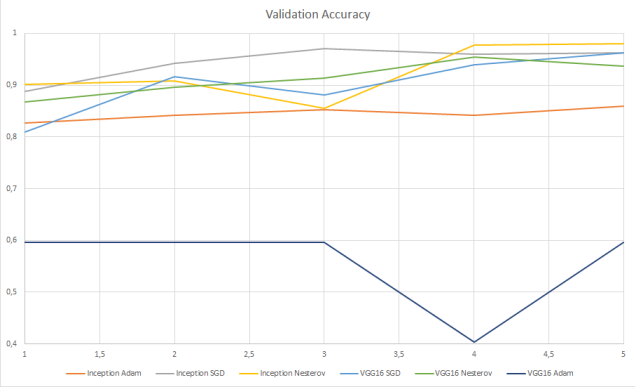

Figure 5: Validation accuracy models comparison.

The different results yielded by choosing different optimizers are displayed in Table 1. We can see that the best model is the Inception V3 with Nesterov optimizer, which gives a validation accuracy of 98,0% after 5 training epochs. After 25 epochs, we were able to generate a validation accuracy of 99.8% on a validation set of 362 entries (10% of the total), and 99.8% on the test set of the same size.

Optimizer	VGG16	Inception V3
SGD Mom 0.9	96,2%	96,2%
Adam	59,6%	85,8%
Nesterov	95,3%	98,0%

Table 1: Accuracies for different optimizers and architectures for validation after 5 epochs

Optimizer	Inception V3
Nesterov	99,7%

Table 2: Accuracies for different optimizers and architectures for validation after 5 epochs

For some reason, and just like the authors of the original article, we were not able to get Adam to converge properly, even after manipulating the hyperparameters.

5 Discussion

Our predictions for the best model are very accurate. However we could discuss the limitation of our problem: we have to train our model with genuine and forged signatures of the person whose new signature we want to classify. Moreover, if this new signature is forged, our model also has to be trained with other forgeries of the same counterfeiter, or else the predictions will not be as good.

For example we have tested the following: we removed all the forged signatures of the counterfeiter A imitating B's signature from the training set. The network is then incapable of deciding whether a new signature produced by A imitating B is a forgery or not.

This is a problem because in real-life signature forgery detection we do not have access to all the forged signatures of a given person. Maybe with enough data we could work around this limitation.

Another problem we need to consider more is data normalization. We apply basic normalization to the images, but images from the original data set and the ones we generated can be easily told apart, so are the signatures we generated on different tools (computer, telephone, ...). These differences are probably taken into account by the network, and they should not be present in an ideal data set.

Finally, we did not have enough data to generate meaningful results on the validation and test sets while maintaining enough images for training. Using 95% of our data for training, we were able to generate 100% accuracy on the validation (2% - 72 signatures) and test (3% - 108 signatures) sets, but it could be due to them not being large enough.

6 Conclusion

We experimented two architectures in this work to classify genuine and forged signatures; VGG16 and Inception V3. As convolutional neural network already showed their efficiency in dealing with images, their usage was natural for us. Indeed, both VGG16 and Inception V3 architectures displayed excellent results regarding our task in hand, with a slight advantage

for Inception V3 (99,8% test accuracy). The use of this architecture is a novelty compared to previous work in the same problem, and reaffirms the great performance of recent deep learning architectures. Future work can focus on the online version of this problem while testing new architectures.