UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE

PAR
ABOLFAZL VASHEE

CLASSIFICATION DES SIGNATURES HORS LIGNE

MAI 2022

Abstract

One of the difficult tasks in biometric classifications is offline signature classifications. Main issue is the small number of samples of signatures per signer. This makes traditional methods that are based on clustering images over their similarities less efficient due to lack of sufficient data. Here, in this work we used Siamese networks to classify signatures and find the associated signer. Siamese-nets composed of pairs of CNNs with identical weights. Siamese-net maps the signatures to feature space in a way that similar signatures would be placed closer to each other. Training process is based on feeding the model with pairs of images. Input image would be a pair of similar and dissimilar images. Hence, the model learns to minimize the distance between the features of similar images and simultaneously maximizing the feature distances in case of dissimilar signature entries. At inference time, a given query signature could be compared in a pairwise manner with existing classes of signatures. Any class that has the highest similarity with the given query image would be selected as the signer of that query signature. Several experiments were conducted to determine the hyperparameters of the model. Classification results show the capabilities of our approach in signature classification. Proposed approach outperforms classification over CNN multi class structure by over 10% accuracy.

Table Of Contents

Table of figures

List of tables

Table of equations

List of Abbreviations

DT	Dichotomy Transformations
DTW	Dynamic Time Warping
HMMs	Hidden Markov Models
HOG	Histogram of Oriented Gradients
HSV	Handwritten Signature Validation
ReLU	Rectifier Linear Unit
SVMs	Support Vector Machines
WD	Writer Dependent
WI	Writer Independent
ICDAR	International Conference on Document Analysis and Recognition
ICEEOT	International Conference on Electrical, Electronics, and Optimization Techniques
ICICoS	International Conference on Informatics and Computational Sciences
Eq	Equation
CNN	Convolutional Neural Networks
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

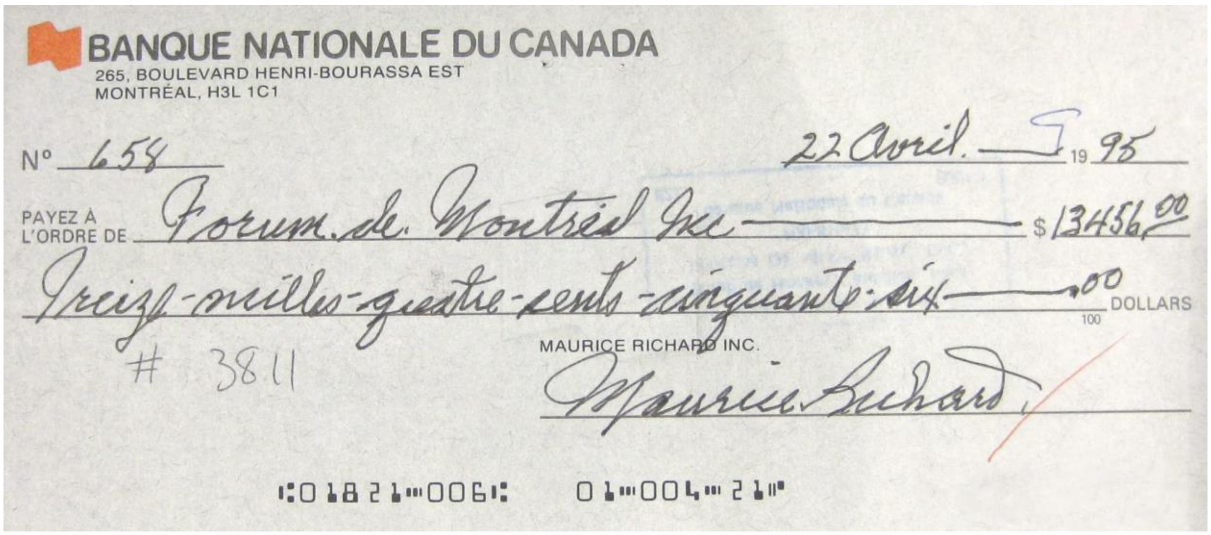
1. Introduction

Signature verification is a domain related to detecting the forged signatures of users from the genuine ones. Common problems arise when the forger's signatures have less divergence from the user's original signatures. Also, the scarcity of signature samples is a prevalent issue. A signature is a form of behavioral biometrics in the biometrics domain. Validating a signature in this domain is also more complicated in comparison with other branches like fingerprints due to good forgeries.

Forgeries can be categorized into three sections:

- Simple forgeries: the forger has information only about the writer's name (user) and doesn't see any genuine signature.
- Random forgeries: the forger has no information about the name or the signature.
- Skilled forgeries: the forger knows about both name and the genuine signature. This case results in lots of forgeries with high similarity with original signatures.

Based on how we gather, Handwritten Signature Validation (HSV) falls into two groups. Offline or static and online or dynamic. In the case of static or offline signature acquisition, usually, the signature gets scanned with different resolutions (usually 600 dpi). This scan is the only input to any verification system. A couple of examples of offline verification are cheque signatures, and vouchers at banks, such as the one shown in Figure 1.



BANQUE NATIONALE DU CANADA
265, BOULEVARD HENRI-BOURASSA EST
MONTRÉAL, H3L 1C1

N° 658 22 Avril 5 19 95

PAYEZ À L'ORDRE DE Forum de Montréal Inc $ 13456 00

Treize-milles-quatre-cents-cinquante-six 00/100 DOLLARS

3811

MAURICE RICHARD INC.

Maurice Richard

⑆01821⑈006⑆ 01⑈004⑈21⑈

Figure 1: An example of offline signature over cheques collected from the internet.

On the other hand, temporal information is also available with online signature acquisitions. This time-related information like the speed of the pen, pressure, inclination angle, number, and sequence of strokes are great help to have a better understanding through the whole signing process [11]. Such data are gathered dynamically via smart pens and pressure-sensitive pads, as shown in Figure 2.

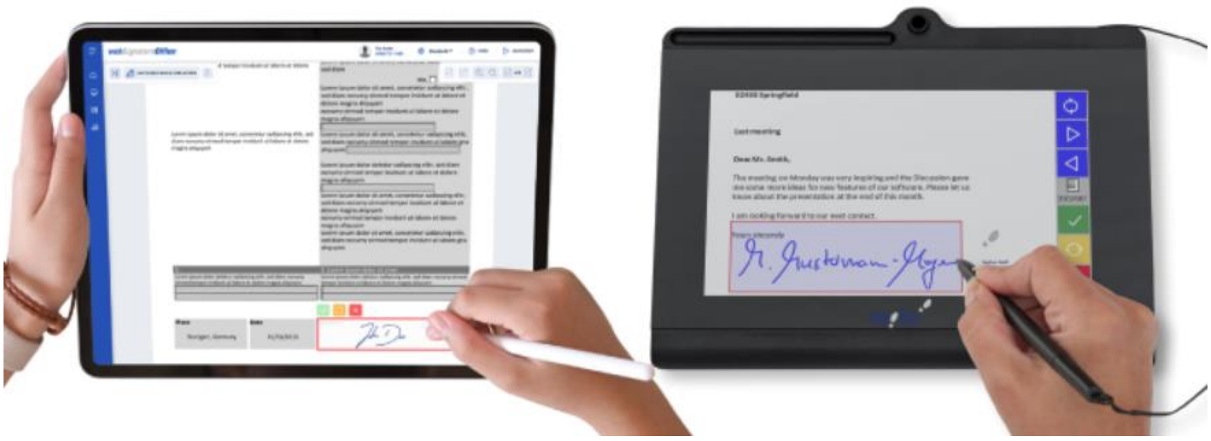


Figure 2: Two examples of an online signature capturing device [40].

The online signature acquisition is usually used for applications like the authentication of electronic documents. Different sorts of preprocessing, feature selection, and feature extractions are used depending on the offline or online approach.

Applying offline signature verification is more complicated than the online approach due to the lack of extra information caught by electronic devices. Information like trajectory tracing, speed, pressure, etc., makes it easier to detect the difference between the original and forged signatures. It should be noted that variations between the user's original signatures are edge cases and challenging ones for both online and offline approaches.

Dynamic online information increases the success rate in signature verification, which inspires some researchers to extract dynamic information reflected on the texture of the static scanned images [41]. This approach is dependent on a proper pen and paper.

In biometric authentication, in general, the user has to register his biometric, which in our case, is the signature of the system. Then, his biometrics and identification will be linked to the system. Later on, every time the user utilizes his biometrics. It is used as a query to be verified against the reference saved biometrics in the system. If the comparison between these two was above a certain similarity threshold, the verification is marked as passed, otherwise marked as failed, and in this case, considered a forgery.

A variety of approaches could be explored depending on the offline or online branches of verifications. One approach is user-based modeling, where there would be one model associated with each user. In such cases, the number of existing referential images is the bottleneck. Because here, models like Hidden Markov Models (HMMs) or Support Vector Machines (SVMs) are used, and the initial data should be more than ten samples. Having that number of initial training-sets for signatures is a bit unrealistic, while the user is usually ready to provide 2-3 signatures.
Another form of verification is template-based verification, where users have fewer initial samples (1-5). Then with every new query, it is going to be compared in the form of distance with these initial images. If the difference is below a threshold, the new query would be marked as genuine otherwise as a forgery.

In online verification, Dynamic Time Warping (DTW) was shown to be promising in utilizing the trajectory of the signature as a distinctive feature for verification [42]. However, as the number of initial samples is usually low in this problem, local feature extractions have been focused on in previous research.

We are after a powerful and precise approach that can identify and confirm a handwritten signature in a short amount of time which has a lot of usage in different corporations. So far, several works of literature have taken this issue into consideration and tackled it with different approaches [24, 32, 49]. The main focus of this thesis is to find an approach that could select specific features of static images which are distinctive enough to classify the given image automatically. That's where deep neural networks and CNNs become handy with their ability to converge to the right and distinctive features automatically.

Deep neural networks, especially CNNs (convolutional neural networks) have been used in a variety of domains like object detection, classification, etc. But one of the main limitations of such tasks is the size of the dataset. Usually, to get a reliable classification model a huge labeled dataset is needed. Which may be very hard to get in some tasks like signature classification and authentication. Here we try to tackle this task by Siamese networks. These networks, unlike normal classification networks, do not produce a probability distribution for each class. Instead, it learns the similarity between two given pairs.

In order to address the signature classification issue, we have to know the difficulties we are facing in this derivative of the signature authentication problem. Most of the issues we are facing in the main problem still persist in this subproblem too. High intraclass variation of signatures also lowers inter-class variations among some user's signatures. Also, the fact that hand-written signatures in each country can be dependent on various parameters like language and culture makes the signature classification even more challenging.

The second main issue is the lack of sufficient signature samples because either people do not find it convenient to share their signatures with others or it takes time to make such a dataset.

If we break the problem HSV into smaller problems. It would first find the user for the questioned signature and then figure out if it is a forgery or a genuine signature. Here, we focused on the first part of the problem which is the classification of the questioned signature. This stepwise approach helps us to gain more insights into feature selection and model-tuning, transfer of learning for the second part of the problem.

In the rest of the thesis, we first discuss some previous works in the literature and then explain more about CNN and its ability to automatically pick up the features based on the objective function. Then we talk more specifically about deep Siamese networks, their characteristics and why it is an excellent candidate to help us do the classification even with a small amount of training-set. Afterward, we explain our method to do the classification using the Siamese nets and compare it with the baseline CNN classification in the result section.

2. Literature Review

The first step in designing an HSV system is selecting the classification strategy. Two well-known approaches are writer-dependent vs. writer-independent. In classifications using the Writer Dependent (WD) approach, for each writer, a different classifier is trained. Which is dominantly used due to its higher accuracy and reliability. The drawbacks related to the WD approach are complexity overhead and high computational costs of the system when more writers (clients) are added to the system [43].

Considering the fact that handwritten signatures, unlike other biometric traits, have a significant dependency on a person's behavior at the time, even each signature from one person cannot be identical to another. As mentioned before, Dynamic Time Warping (DTW) showed better performance in online signature detections [30, 31].

The second approach is Writer Independent (WI) classifications, whereas one model is trained for all the writers. This approach is based chiefly on dissimilarity space generated by existing Dichotomy transformers [43]. Here, a dissimilarity metric is used by a dictomizer (two-class classifier) to determine if a given image belongs to a class or not. WI approaches seem to be less complex and also less reliable in comparison with WD approaches [44]. Existing challenges in this domain are a high intra-class variation (see Figure 3), low training samples, an imbalanced dataset, and scalability in case a new writer is introduced to the system. Overall, both systems have their advantages and disadvantages at some points. This is why some studies focused on hybrid approaches in order to achieve better results [16,17,18].

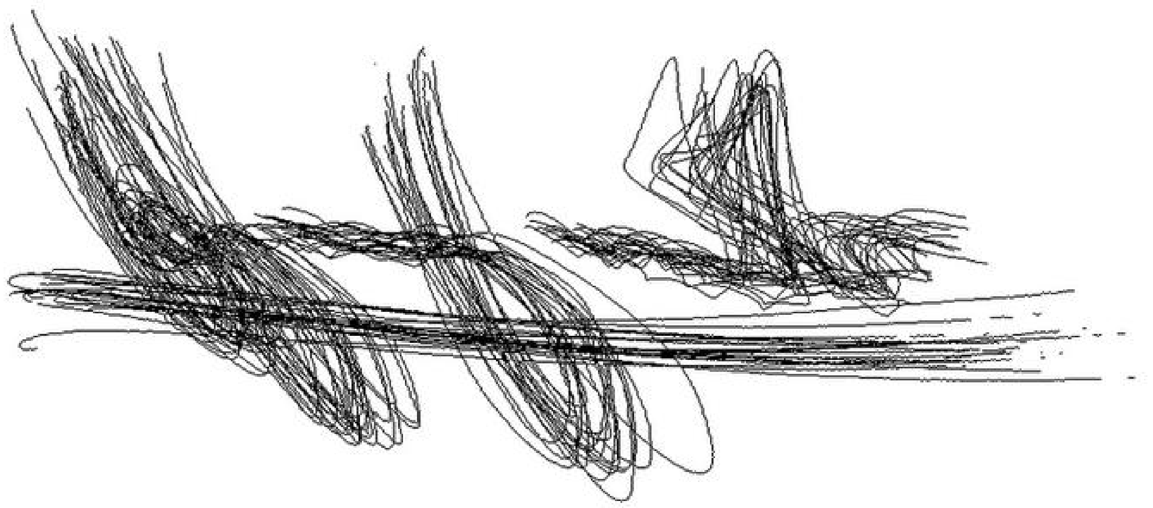


Figure 3: Overlapped signatures from the same user show a high amount of intra-class variations exists for one writer [47].

Some research done in WI over dissimilarity space is based on comparisons like [46] could address some of the issues like the scalability of the classifier. Such works are based on a comparison of the query signature vs reference signature in the feature space. Hence, the system is based on a comparison with the reference signature. Therefore, the system can manage writers that are not trained on which address the scalability issue.

Classification based on DT is very sensitive to the feature selection as we have only two classes positive and negative. 1) Positive class is composed of dissimilarity vectors that are computed based on existing samples in the same class. 2) The negative class has distance vectors that are calculated from other writers. Approaches based on DT try to have as little as possible overlap between these two classes.

Feature extraction is one of the essential steps in this kind of classification. For years, researchers [17, 24] have tried to tackle the issue from different angles so that they may find an optimal technique. For this phase, we can categorize their kinds of literature under handcrafted and non-handcrafted features-based methods. In the first category, the process should have been done manually, which takes lots of time and effort to design a feature extractor. On the other hand, the latter approach, with a robust ability to learn complex features and adjustability, helps us to achieve higher accuracy.

Basically, hand-crafted methods use some filters to define some simple features. These methods are categorized as Basic image features, Statistical, Content-Based, and Local and Global feature-based methods. Moreover, in this field of study (signature detection), we often come across Global and Local feature-based methods. In order to find and extract particular features from part of an image like edges or tiny patches, we divide it into a grid and then use our method on each part of the grid (which can contain a group of pixels), usually this process is known as Local feature detection. In contrast, Global feature extraction is when we take the entire image into account as one part totally and then utilize our descriptor which illustrates that image by one vector.

Although nowadays studies have focused more on deep learning methods, still hand-crafted features-based methods are part of actual research in this field. That is to say, for instance, Cavalcanti et al. (2002) [20] decided to use 3 methods: structural, invariant, and pseudo-dynamic for the feature extraction phase and chose the subset of features with the higher result according to the classifiers. They used Bayes and K-NN as classifiers. Oliveira et al. (2005) also did their research on bank cheque signatures and found out that pseudo-dynamic features along with static features are useful in this matter [21]. Histogram of Oriented Gradients (HOG) because of its great edge detection ability and immunity to noise and transfiguration in images, regardless of their shape, is one of the popular methods among these types [22, 24, 26]. Later Zhang (2010) and Harfiya et al. (2017) used the Pyramid Histogram of Oriented Gradients which is an extended form of HOG. Also, there have been some authors who analyzed the issue from a mathematical and fractal [27, 28, 29] to a Graphometry perspective [38, 39].

Machine learning (and Deep Learning) algorithms, such as Neural Networks, SVM (Support Vector Machines), and Siamese Neural Networks, can automatically do the feature learning and the classification. Among recent studies, CNN (Convolution Neural Networks), due to its robust ability for classification, has also gained lots of popularity(attention) in this matter. Using CNN for classification started with Khalajzadeh et al. [8] and was later developed by others on a larger scale [9, 6], and Kaulen and Baab did it recently on a smaller scale [10]. Also, we have to mention that there are some hybrid works that combine CNN with other approaches like the Crest-Trough method and SURF algorithm for better results [6].

Some authors used supervised classification methods and leveraged techniques like LDA (Linear Discriminant Analysis), which developed on anisotropic scaling and registration residual error [12]. LAD reduces the dimensionality of data and selects a new exercise inorder to maximize the separability (distance) between existing categories (classes). Another supervised technique is the pixel-based approach which compares the pattern to find the edges regardless of signature transformations and uses SVM for classification [13, 14]. It is worth noting that despite the fact that traditional machine learning algorithms like SVM are not sufficient to do reliable classification for large datasets, other research has shown that in smaller datasets, they have proven to have acceptable accuracies [15].

Siamese-nets are one of the recent approaches that integrate and automate the classical steps of classifications in one model. As mentioned in the previous paragraph, classical approaches mostly use the hand-crafted feature. Then classification methods like SVM would be applied on those features. Zhu et al demonstrated that for the tasks with small numbers of samples focusing on dissimilarities applying LDA [12] for verification purposes would be beneficial.

Siamese-nets follow the same steps but the feature selection is automatic and maximizing the difference between inter class images is done with a fully connected perceptron applied on top of the extracted features distances. An example of this approach in literature is the work of Sunak et al. on signature validation [1]. They used Siamese-nets to detect the forgery on signatures. They showed that Siamese-net is a reliable approach for signature validation tasks. And specifically maximizing the differences between original and forgery signatures. Here, instead of forgery detection, the Siamese-net model architecture is used to classify the given query signature and find the corresponding signer.

2.1 Literature review conclusion:

We have two methods of signature classification, Online and Offline classification. Online classification uses electronic devices to capture features related to the user. Such as, velocity of the pen, pressure, inclination angle, number, and sequence of strokes. Offline classification focuses on images of the signatures to extract meaningful and descriptive features for classifications. One of main existing issues in signature classifications are the low number of signature samples per user.

Low number of samples is a bottleneck for conventional approaches either hand crafted features or CNN based methods to extract the meaningful features. Both are mainly focused on similarity in intra-class images to create. It has been demonstrated that methods which not only focuses on capturing similarities but also simultaneously maximizing the distance between the features of dissimilar images are good candidates for classifications of images with a small number of training-set like Siamese-net.

3. Convolutional Neural Networks (CNN)

Arguably, in the field of deep learning and computer vision, we can refer to CNN (Convolutional Neural Networks) as one of the subclasses of artificial neural networks which carry a powerful technique in productive and explanatory tasks for image processing.

CNNs are deep learning models designed to process the data in the form of grid patterns, like images and videos. The animal visual system inspires this approach. It is designed to capture and learn the features and spatial relationships at low feature level like edges and high feature level like objects.

CNN usually has three building blocks. Input layer, feature-extraction layers, and Classification layers like in Figure 4. The convolutional and max-pooling layers are designed to extract features from the given image and also decrease the dimensionality of data. The last layer which is fully connected is actually the utilization of single layer neural network classification. In the nutshell, the high dimensional input data, with lots of spatial relationships, after going through feature extraction layers the image is transformed to lower dimensionality of data representations which now can be used by single layer perceptron for classifications.

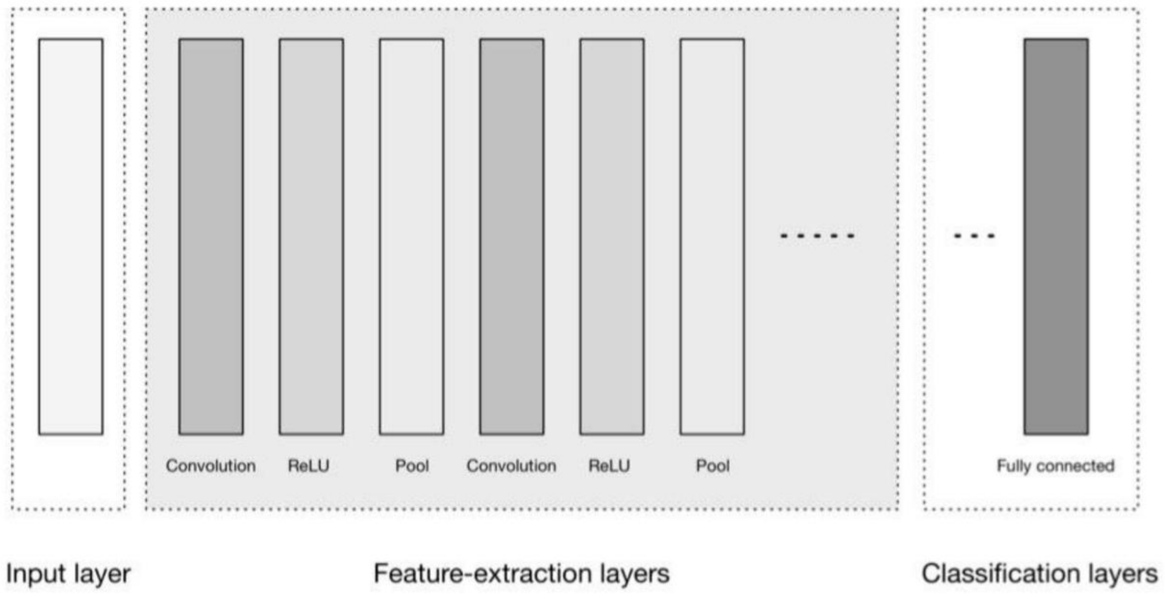


Figure 4: High level overview of CNN structures [48].

The convolutional layer is the core of the CNN architecture. Images are stored in 2-dimensional arrays that are input to a CNN, as shown in Figure 8. As the meaningful features of the image could be anywhere in the image, an optimizable small 2D array known as a kernel is going to convolve with the entire image to extract features, as shown in Figure 5. Results of this procedure from another convolutional layer that receives the same procedure, which will be applied to extract features from higher-level features. (Kernels are typically initialized with random values, then the values are optimized using a gradient descent optimizer.)

Max pooling layer: select the maximum value in a determined window over the extracted feature map, as shown in Figure 7. This layer helps the model to decrease dimensionality. Also, max pooling maintains partial invariance to small rotations because the max of a selected region depends only on the single largest feature present in that region.

After features are extracted, it is usually flattened to an array and fed to a fully connected multilayer perceptron with final cost function at the end of the pipeline. This function should be differentiable to be able to apply gradient descent back to lower levels in the pipeline and optimize the kernels. This procedure is called learning. Learning is the process to minimize the difference between outputs of the model and the ground truth target via a backpropagation algorithm. For example, let's assume the CNN model structure displayed in Figure 8 for classification of the input image into 9 different classes. If the ground truth is 1 for class 5 and the model provides a low probability for that class. In the learning process the loss function would assign a high value for the loss to this node (node 5) and would backpropagate the derivatives to previous layers in order to adjust the weights of the CNN.

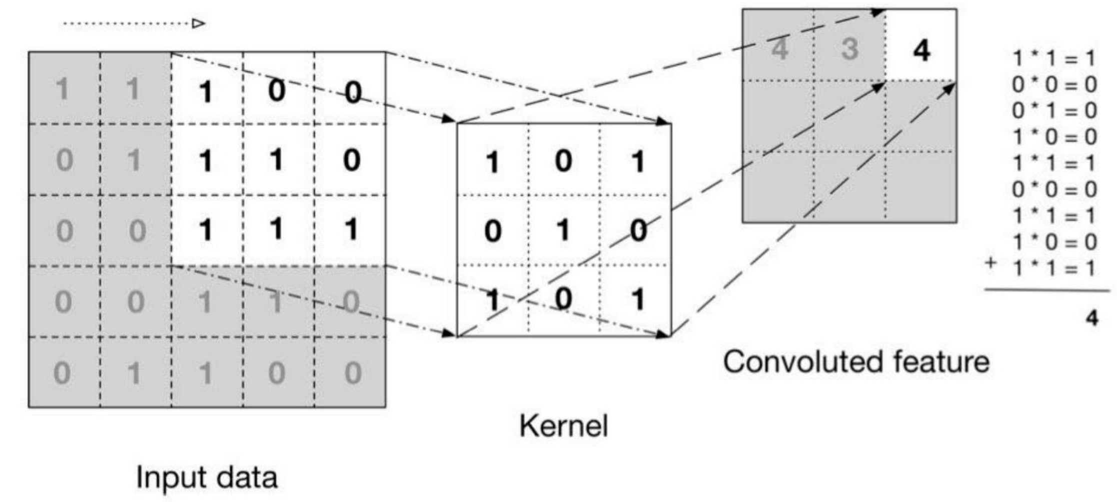


Figure 5: Convolution, Kernel, and feature maps example [48].

Each block plays a different role while the output of each layer is the input for the next layer. When we feed the image at the input layer and each Conv layer processes it with activation maps and as it goes through more layers, more convolved features are extracted. In order to stack layers, we have to pay attention to the input and output dimensions. For example, to add a dense layer after the Conv layer we commonly use a flattened layer in between to change the 4D array into 2D. As mentioned in Figure 6. Input and output shapes in CNNs are 4D arrays (batch size, height, weight, depth) where depth depends on grayscale or RGB images may vary. Pooling layers have a massive impact on reducing the computing process by decreasing the feature map size.

Usually, at the end of CNN's architecture, we have fully connected layers. The last block feeds with pooled feature maps from the previous layer. Then the output of two-dimensional pooled feature gets flattened by transforming it into a single one-dimensional vector in the Flatten layer. (This layer helps us to connect all pixel values to final layers). In the next step all neurons from one layer link to all neurons in the following layer. Each neuron correlated with a unique feature that might be existing in the image. The probability of that feature in the image passes through, the value of each neuron.

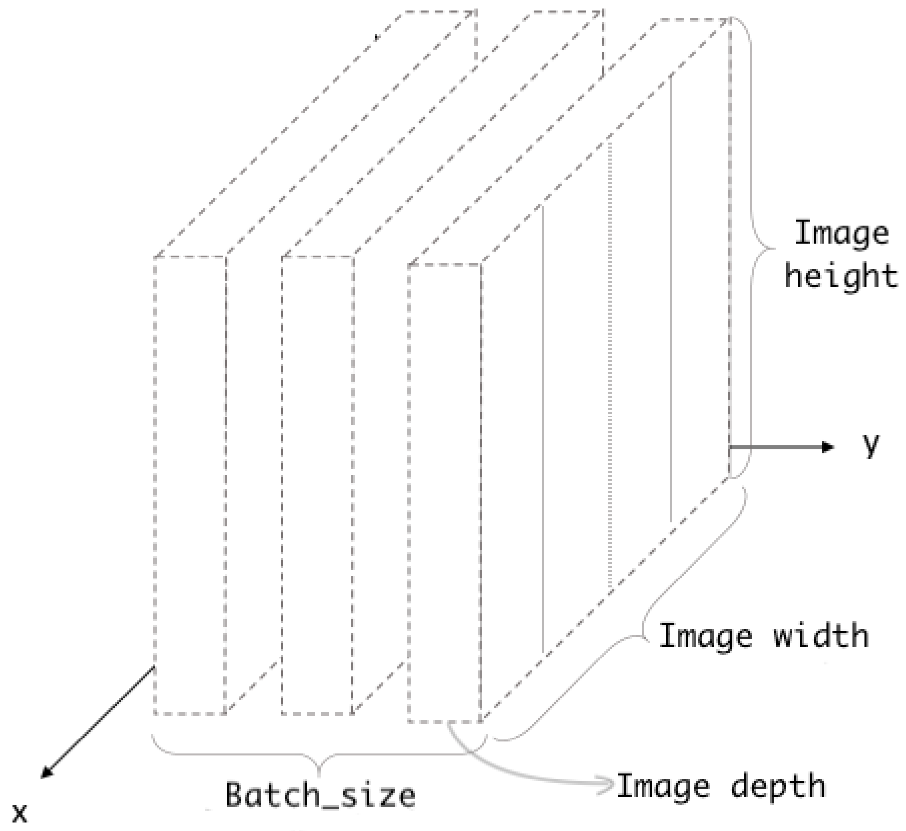


Figure 6: Input data dimensionality in CNNs [51].

At the end of the fully connected layer there is an activation function. Activation functions can be divided into two categories. Linear and Nonlinear activation functions. Logistic, Softmax and ReLu (Rectified Linear Unit) are some common examples of these mathematical functions in neural networks. Logistic sigmoid activation and ReLu activation functions are used for binary classification. This function takes any real value as input and outputs values in the range of 0 to 1. Softmax is usually used in the last layer for multi-classification purposes. In the Softmax function, the raw output of the neural network transforms into a vector of probabilities. Based on the relative scale of each value in the vector.

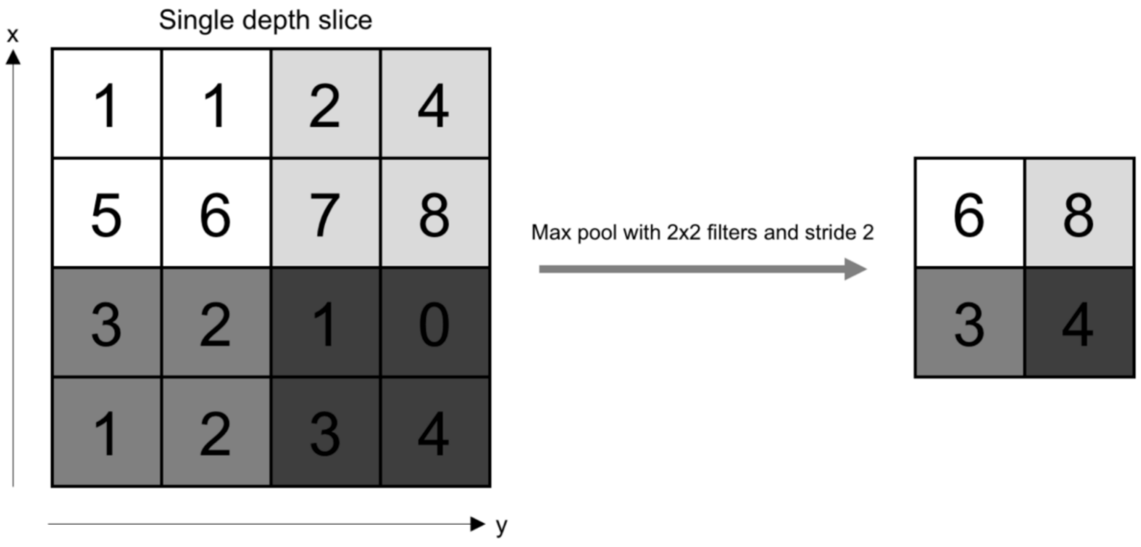


Figure 7: Max Pooling example [48].

The following terms are consistently employed throughout this master thesis so as to avoid confusion. A "parameter" in this article stands for a variable that is automatically learned during the training process. A "hyperparameter" refers to a variable that needs to be set before the training process starts. A "kernel" refers to the sets of learnable parameters applied in convolution operations.

Hyperparameters that are subject to change in this work are input image size, number of classes to be trained over, number of batches to train over, training iteration. In this work, these hyperparameters are decided over a set of experiments that has been explained in detail in Chapter 8.

3.1 Conclusion of CNN

CNN model's architectures are designed to decrease the dimensionality of the images so other machine learning techniques like multi layer perceptron. can be applied on the feature vectors. Such approaches allow the application of classifications or object detection tasks on images. Process of dimensional reduction is applied by several layers of convolutional and max-pooling operations over several kernels that are used for convolutional operations. Final Here, we used a flatten one dimensional array of features as the final output of the CNNs. We used two CNNs with shared weights in parallel to be able to extract features of a pair of images simultaneously. This is the basis of the Siamese-nets that is explained in the next chapter.

4. Siamese networks

The word Siam comes from Siamese cats that usually look identical. Here, we use two identical networks in parallel. such networks are also called sister networks. The main functionality of these networks is based on a comparison of two inputs.

Basically, these networks consist of two subnetworks (in most examples CNNs) with the same architecture, parameters, and weights. As their name represents, this type of network helps us to find the similarity between two images taken as input. In one-shot learning instead of using the Softmax or Logistic layer at the last layer, the output of fully connected layers is encoded in each subnet, and later with the help of distance metrics learning like Triplet loss, Binary cross-entropy, etc.

Siamese networks are used in a variety of verification applications, especially those with imbalanced and scarce datasets. Typical examples would be in companies for face detection and identity verification or in the medical field based on specific case studies. For instance, Liu Chin-Fu, et al. demonstrate that in a study on whole-brain MR (Magnetic resonance) images, through encoding the asymmetry in brain volumes, Siamese networks can outperform conventional prediction techniques in terms of processing time and decreasing complexity [50].

4.1. Classification and Siamese-nets

Siamese networks are designed to receive two images at the same time, they are perfect for comparison-based approaches. Previously Dichotomy Transformations (DT) approaches were shown to be a useful approach in addressing issues like a low number of samples in classes and the scalability of the model in case of adding new classes. DT is based on measuring the dissimilarities in feature space. Dichotomy Transformations are very sensitive to feature selection. This approach has been used also by Koch et al [4] for character classification.

Here, with Siamese networks, we can leverage the CNN architecture to automatically select meaningful features and build the dissimilarity model. So, a CNN model extracts the descriptive features. Then that CNN model is duplicated and used for two input images. One of the inputs is the reference image and the other would be the questioned image. The idea is that if these two images belong to the same class, then features that are going to get extracted by CNN would be very similar to one another. Hence, the distance between the features would be minimal. This helps us to calculate the dissimilarities in feature space between any query signature and existing reference signatures. Here, to classify the question image, it is compared with all the reference images of each class. Then similarly for each class it gets calculated. And lower dissimilarity or higher similarity decides which class it belongs to. This approach has been also used in different classification applications like character classification by Koch et al [4].

Siamese classification is easier to grasp when compared with normal CNN classification architecture. Usually, classification networks are composed of several layers of convolution and pooling layers. In vanilla CNN classification, when a query image is fed to the network the output would be a probability distribution over the output classes. To be noted, in this thesis the term Vanilla CNN tends to be used to refer to a typical convolutional neural network.

Considering face classification as an example for a company with five employees, the face classification for them through normal CNN classification would have five probabilities in output. To train such a network, we need large samples of each person. Another issue is that if we want to add a new class like a new employee's face, then we have to get a large number of images for this new class and retrain the whole model again. Furthermore, sometimes the classes are constantly changing. An example would be an employee leaving or getting hired if we have used the regular classification model. Not only do we need more samples of new employees' faces, but also, the whole model should be retrained, which brings lots of financial overhead in terms of time and costs. This kind of situation is where Siamese networks address such classification problems.

In a nutshell, Siamese networks would learn features to emphasize the dissimilarity between samples. By feeding a pair of images to the network and maximizing/minimizing the distance between features to determine the similarity based on whether the pairs belong to the same class or not. At inference time any given image would be compared with this reference and the similarity value would be produced by the Siamese network.

In other words, the Siamese network does not learn to classify the given image directly to output classes. Instead, it learns the similarity function between two pairs.

Figures. 8 and 9 illustrate the difference between the architectures of normal classification models by CNN and Siamese Networks. In CNN, we will have probabilities that given images belong to which output class.

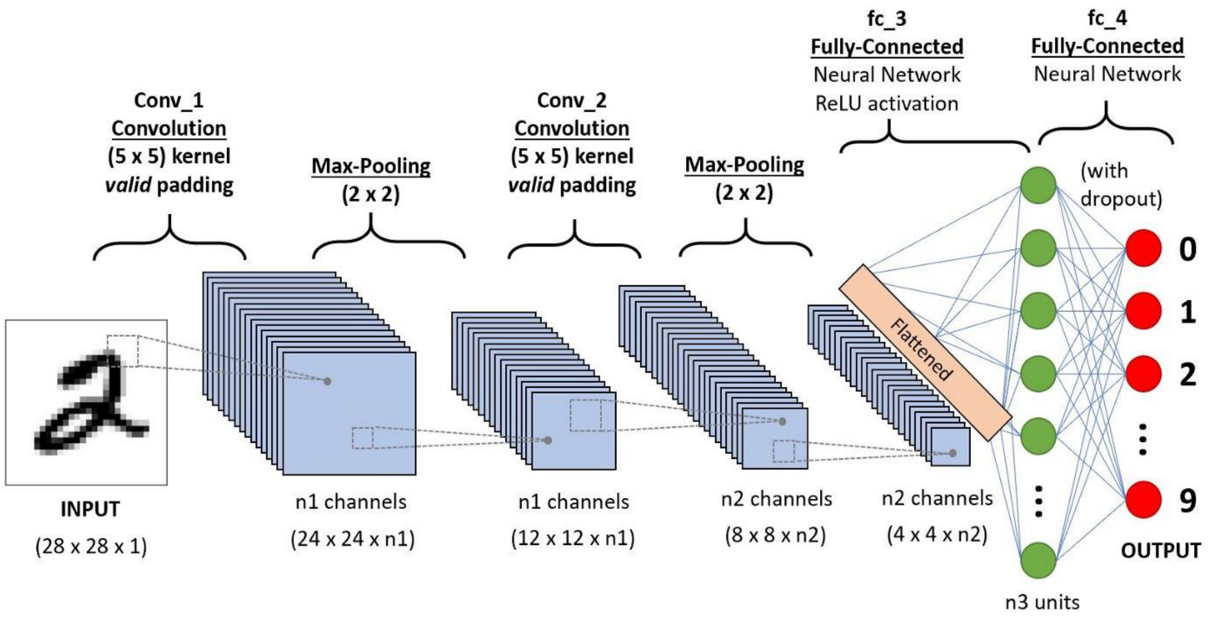
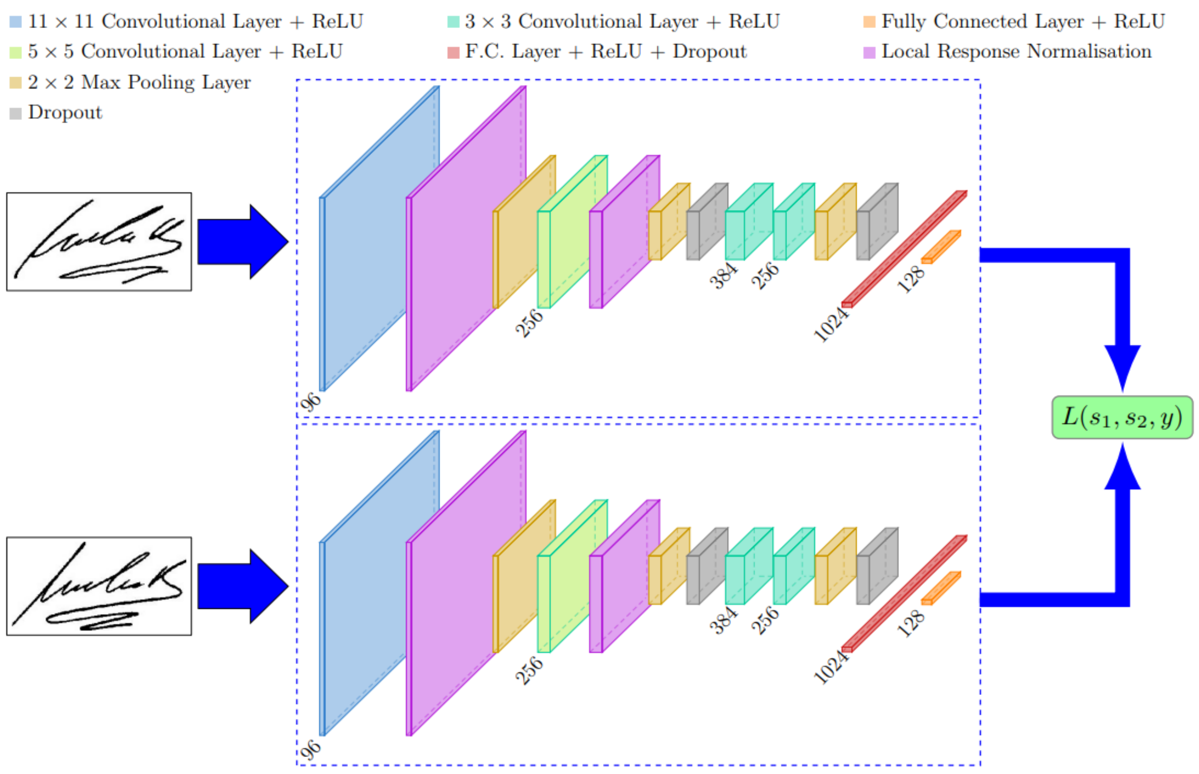


Figure 8: An example of a normal classification network based on CNN [2].

Moreover, it should be noted that a color image consists of 3 channels, known as RGB channels (Red, Green, and Blue). Therefore, the matrix dimensionality of a color image can be presented by $\boldsymbol{w} \times \boldsymbol{h} \times \boldsymbol{c}$. Respectively $\boldsymbol{w}$ and $\boldsymbol{h}$ are the width and height of the image, while $\boldsymbol{c}$ shows the number of channels (***n1*** in Figure 8). In convolving layers, after applying the filter on each channel, the output of the channels' number is different, shown by ***n2*** in this figure. In the end, before feeding the image to the hidden layers, we have a flattened layer which has been explained in more detail in chapter 3. In the flatten layer, the previous layer's output is transformed into one vector with the numbers of the neurons (shown as ***n3*** in figure 8).



11 × 11 Convolutional Layer + ReLU
5 × 5 Convolutional Layer + ReLU
2 × 2 Max Pooling Layer
Dropout
3 × 3 Convolutional Layer + ReLU
F.C. Layer + ReLU + Dropout
Fully Connected Layer + ReLU
Local Response Normalisation
96
256
384
256
1024
128
96
256
384
256
1024
128
$L(s_1, s_2, y)$

Figure 9: An example of Siamese network architecture for signature authentication [1].

Figure 9 shows examples of Siamese networks used by Sunak et al. [1]. Here, two CNN subnets extracted the features (s1 and s2) and joined by a loss function L, that compares the similarity between features s1 and s2. Additionally. In Sunak et al. [1] work, ***y*** is another function that determines if the two given samples are members of the same class or not.

4.2 Signature classification with Siamese nets

We can use Signature classification as a form of HSV (Handwritten Signature Verification). In case that query signature has low similarity with the reference signature of a client, then it can be labeled as a forgery. In large organizations like banks, we can use signature classification to determine whether a questioned signature is similar to the user's registered signature. We may have many clients, and instead of training one model for each client, we can train one model and use it to classify the given question signature even though we have a meager amount of signature samples per class.

As mentioned before, Siamese classification is based on comparison with reference signatures. Classification is done in dissimilarity space which makes it scalable when a new class is introduced and also addresses issues mentioned above. Siamese networks automatically extract meaningful features that help to maximize the interclass difference. So, at inference time, given a new signature, we would be able to compare it with all previous client's signatures and find the most similar signatures and dissimilar signatures. The structure of the network would be as in Figure 10. At training phase, the ConvNet extracts the features from given images, and then the difference between these features is minimized if two images belong to the same class and the difference gets maximized if the two given images do not belong to the same class.

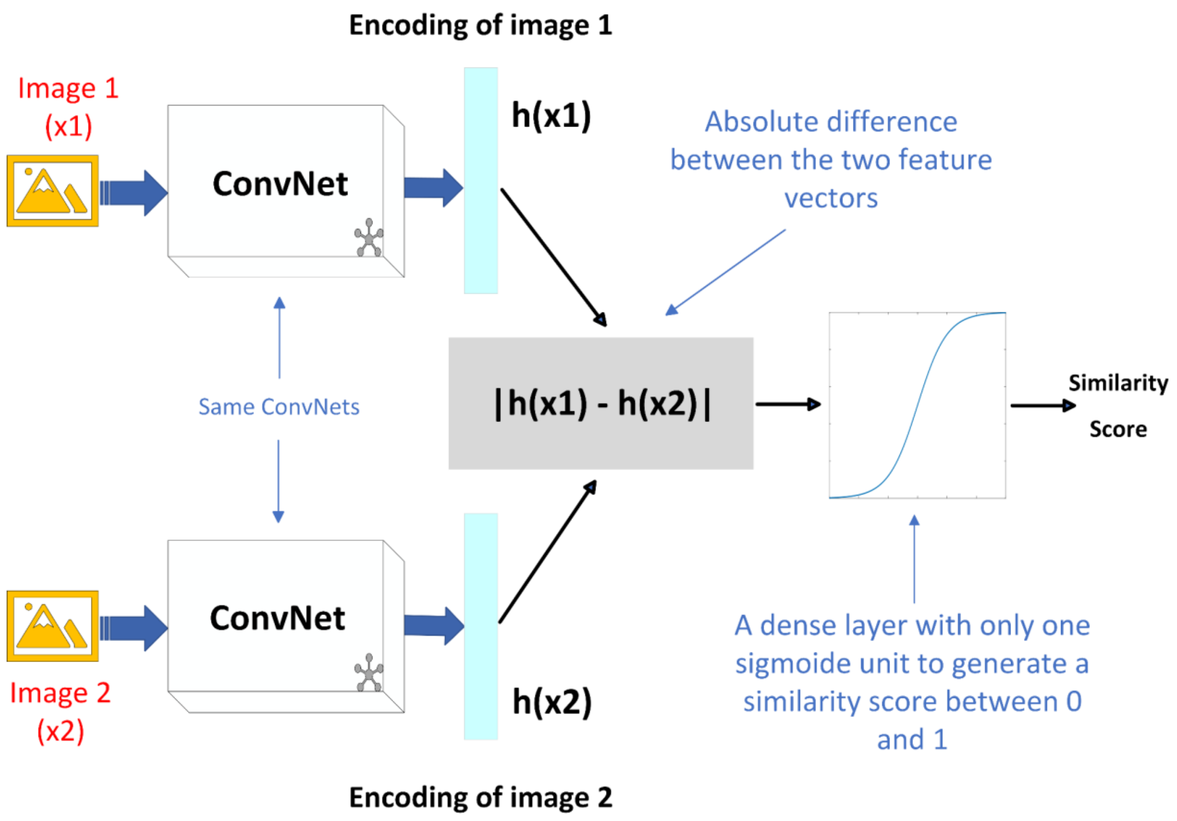

Encoding of image 1
Image 1 (x1)
ConvNet
h(x1)
Absolute difference between the two feature vectors
Same ConvNets
|h(x1) - h(x2)|
Similarity Score
ConvNet
Image 2 (x2)
h(x2)
A dense layer with only one sigmoide unit to generate a similarity score between 0 and 1
Encoding of image 2

Figure 10: A Siamese network architecture in nutshell for similarity calculation.

If each image is considered as x1 and x3 the corresponding extracted features by twin CNNs would be H(x1) and H(x2). The elementwise absolute distance between two features will be fed to a fully connected perceptron with final sigmoid function. During the training the output of the sigmoid function would be set to 1 if two input signature pairs belong to the same class (same signer) otherwise the output of the sigmoid function would be set to zero. So, the model learns how to determine the similarity between two images in the form of a value between zero to one.

4.3 Conclusion of Siamese-net

Siamese-nets are composed of two identical CNN architectures to extract the features from images and vectorize them in a one-dimensional array. In another word input images are mapped into a feature space. The idea of Siamese nets is the fact that similar images would be placed near each other in this feature space. So, having pairwise features their elementwise distance is calculated. Then their distance is fed to a fully connected perceptron with a final sigmoid function to map the similar images to 1 and dissimilar images to 0 in the training phase. This model structure would extract the features in a way that minimizes the distance between the similar signatures and simultaneously maximizes the distance between features of dissimilar input pairs.

In the inference phase, when a new query signature is given to classify, the trained model would be able to assign the similarity value between two input images. This similarity value would be assigned to each existing reference class of signatures. The class with maximum similarity would be selected as the signer of the given query signature.

5. Methodology

Our approach is based on learning similarities between pairs of signatures. Hence, at inference time, we can compare the given query signature with each existing class of reference signatures. Then we can select the class that has the highest similarity with the given query signature as the signer of the query signature. This process is explained in detail in this section.

The key to solving data scarcity in signature classification is good feature engineering. Siamese-nets, helps to obtain features that maximizes the distance between features for dissimilar images and minimize the distance between features for similar features.

Siamese Nets or twin nets are practically two different convolutional networks but share the same weights and parameters. Hence, we can hypothesize that if we pass two images that belong to the same class to twin models, as the conv-nets share the same weights and the images are similar, the extracted features should be also similar. On the other hand, if the images belong to different classes the produced features should be different. For example, consider *Figure 10*. Two images ***x1*** and ***x2*** are passed to each of the twin sisters and their corresponding features are automatically extracted as ***h(x1)*** and ***h(x2)***. Each feature flattens (Figure 8 flatten array) and then in another layer, the distance between them is calculated.

The feature difference is calculated based on the absolute difference between ***h(x1)*** and ***h(x2).*** In our case, as explained in Figure 10, each feature vector would be of size 4096.

$$dif^{j} = |h^{j}(x1) - h^{j}(x2)| \qquad \textit{(Eq. 1)}$$

The ***dif*** vector, also of size 4096, would be given to a single sigmoidal output unit to calculate the prediction as follow:

$$p = \sigma(\sum_{j}^{4096} dif^{j} w^{j}) \qquad \textit{(Eq. 2)}$$

Here, ***p*** is similarity value and ***j*** is the index of each value in the feature vector (flatten layer in Figure 8). The value of ***j*** varies between 1 to 4098. In eq.2, ***σ*** is the sigmoid activation function. And ***w*** are the weights specified to each element of the feature vector. These weights are adjusted during the training phase. In the training phase we set the target value of 1 to each similar input pair of signatures (signatures that belong to the same person). And target value equal 0 for pairs of input signatures that are not similar (belong to two different persons).

Hence, later on, at the inference phase, the output of the sigmoid activation function would be a probability between 0 to 1. This probability is an indication of similarity of two given pairs of images. This is the main idea behind this methodology.

The structure of the network with three convolutional layers was followed by three max-pooling layers and flattening of the final feature vector of size 4096 is demonstrated at a high level as follows.

Here, for sake of simplicity first we down-sampled our images to 70x70 so the shape of the vectors passed at different layers could be visualized by the model plot in Keras as shown in Figure 11.

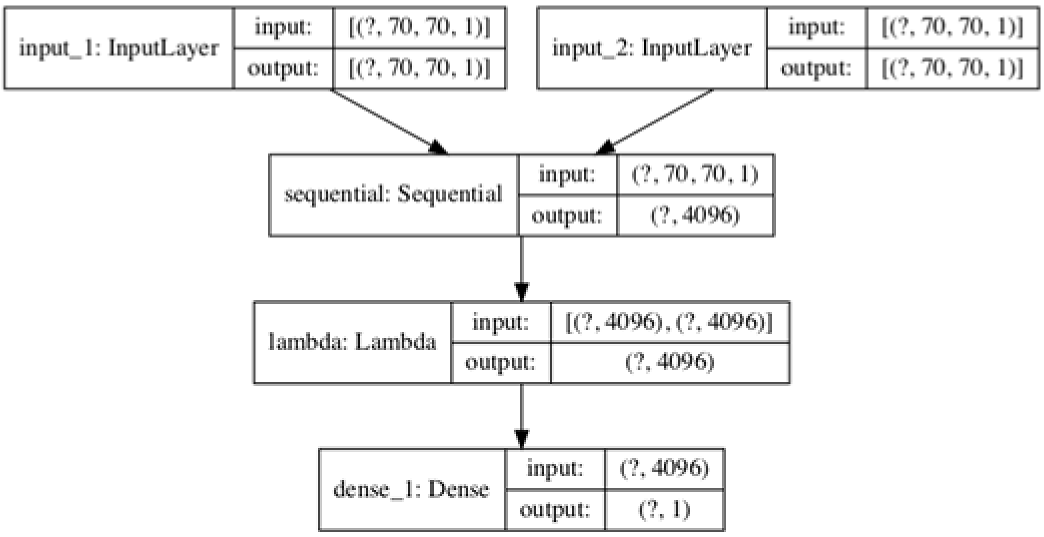


Figure 11: The shape of the vectors that passed in different layers of the network.

Figure 11 shows the summary of the structure of Siamese net model architecture where the input shapes of the image is 70x70 then the extracted features are of 4096. Two features of size 4096 go to a lambda function to calculate the absolute elementwise distance between them and the result would be another 4096 vector that is fully connected to the final last layer with a sigmoid function.

5.1 Conclusion of methodology

Core of this methodology is based on comparison of pairs of images. The Result of the comparison is a similarity value. To get the similarity of pairs of input signatures, first features are extracted by a twin CNN network with shared weights between them. Idea is that if the input images are visually similar the output extracted features would be similar too.

Then, absolute element wise distance between feature vectors creates another feature vector (eq 1) with the same size Figure 11. This feature vector is fully connected to a sigmoid function Figure 10. Sigmoid function gets trained on this distance vector. If the input pairs of signatures are similar, target value of the sigmoid would be set as 1 otherwise 0. Hence, Siamese-net learns to map the similarity of input images to a value between 0 to 1. Also learns how to extract features that minimizes the distance between the similar signatures and simultaneously maximizes the distance between features of dissimilar input pairs.

6. Dataset

The data set is composed of 55 different people's signatures. Examples of such signatures are shown in Figure 12. Each of these 55 classes has around 12 images of signatures for a specific individual.

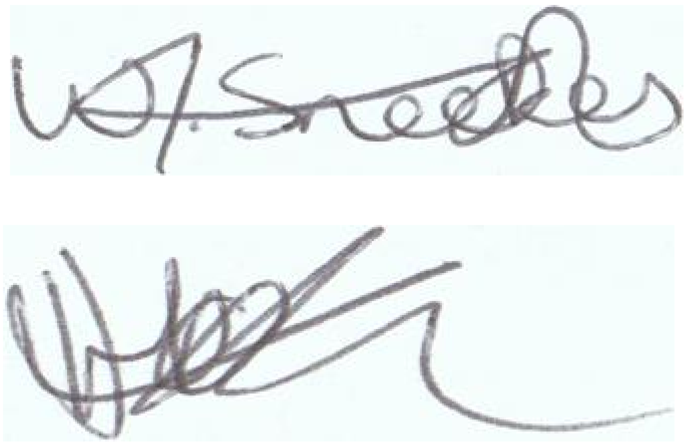


Figure 12: Signature examples.

For sake of simplicity and finding the best hyperparameters, we first decided to do the classifications for 10 classes. Then after finding the best hyperparameters for training the model, we test the trained model on 55 classes of different signatures. The number of images we had per class was 12 (except for two classes with 24 images).

Dataset was split into three subsets: train, validation, and test. The training dataset is used to train the model. The test dataset is used for testing the model after it is trained. If the performance of the model on the test dataset was acceptable then the model performance on validation is calculated. Here, all the reports are done on the validation dataset. As mentioned above this dataset is not seen during the training. The split was 50% for the training subset, 25% for the test subset and 25% for validation subset. For example, for all 10 classes except for class 1. We had 6 images of signatures in the training subset, 3 images for the test subset and 3 images in the validation subset.

Same split for train, test and validation is done on a second dataset that was collected from Chinese signatures [54]. It has 20 different classes, which contain 240 samples. A sample of it is shown in Figure 19.

6.1. Preprocessing

Here, images are converted to gray level and then scaled. Usually, for faster-fine tuning purposes, we began with low image dimensions 32×32 pixels for the first experiment and gradually increased it to 100×100 pixels. We found out that with larger images than 100 pixels, like 200 pixels we begin to lose some accuracy.

6.1 Conclusion of dataset

The Dutch signature dataset contains 55 classes. Each class contains signatures of an individual. Each class is divided into 3 subsets to be able to apply training, validation and test on this dataset. Training usually contains 6 images of each class and test and validation has 3 samples of each class images.

We also apply the same preprocessing on the second dataset (Chinese signature dataset). The Chinese dataset contains 20 different classes and the same split of number of images for train (6 images), and 3 images for test-set and 3 for validation-set.

7. Training and validation

As in our case we have two target values, 0 and 1, we used *binary cross-entropy* loss function to train the model. In binary cross-entropy loss function each of the predicted probability is compared to the actual class output which can be either 0 or 1. Then the score that penalizes the probability is calculated. This score is based on the distance between the calculated probability and the actual expected value.

$$\text{Log loss} = \frac{1}{N} \sum_{i=1}^{N} -(y_i * \log(p_i) + (1-y_i) * \log(1-p_i)) \quad \textit{(Eq. 3)}$$

In the above equation, ***N*** is the number of samples, ***P*** is the prediction and ***Y*** is the actual value.

It was trained for 1000 iterations of training the model over a batch size of 32 randomly selected pairs of signatures and their target values. So, at each iteration we would select another random batch of 32 images from the training-set. The model gets evaluated after 5 iterations training or fitting the model over a given 32 pairs of images. This number is also a parameter to set as if we begin to evaluate more often the training takes a longer time. Later on, we set it to 20.

The way we evaluate the model is based on a comparison of the questioned image with current existing classes of signatures. In this work, we are going to call this kind of comparison the **N-Way comparison**. For example, if we compare the given questioned signature with the current 10 reference signatures, we are doing the 10-way comparisons. Knowing that N can vary up to a number of existing classes of signatures. It is explained in detail in the next sections

7.1. N-Way comparison:

To validate the model performance, we have to see in a pair, if the model could find high similarity intra-class and low similarity for inter-class signatures. To verify that we used a notion called N-way comparison. It is basically comparing the questioned signature with other

N reference signatures including its own class samples. For example, a 4-way comparison would be like the image 1 in Figure 13.

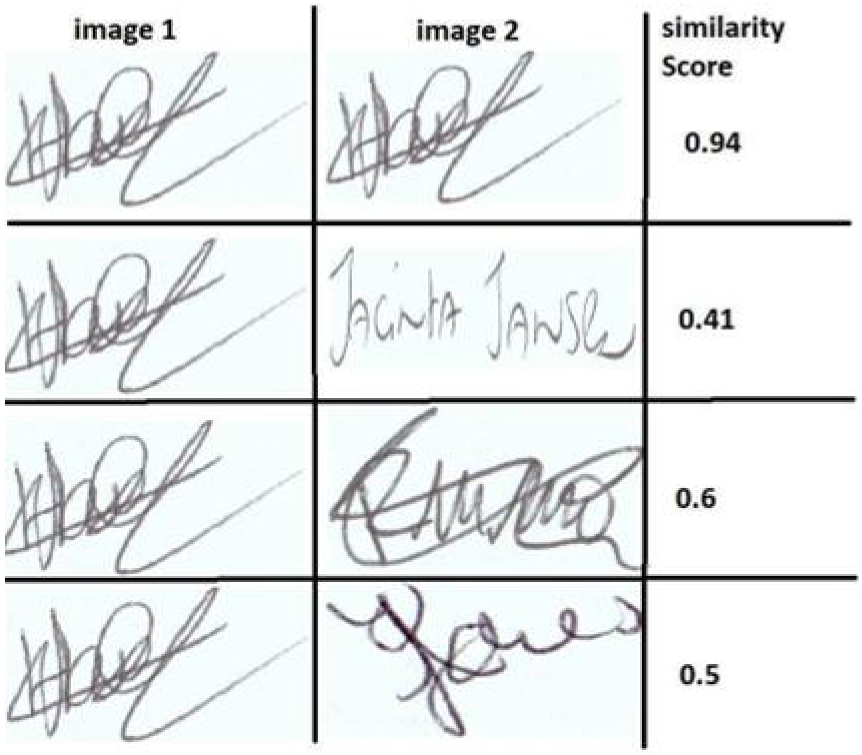


Figure 13: A 4-way comparison pair and its results.

In Figure 13 the questioned signature (in the first column) is compared with **4 (N)** other different classes (second column) and corresponding similarity is calculated (third column). So, if the maximum similarity belongs to the pairs with signatures from the same person, we consider it a correct prediction and if not, we consider this as a miss. It should be noted that the **N-way** comparison has only one pair of signatures belonging to the same person (row one). When we create the pairs to evaluate the model, we select them randomly from the validation-set.

If we repeat creating an N-way comparison for ***'k'*** times, then the accuracy of the model would be obtained as the ratio of the correct predictions over the validation-set, as follow:

$$Validation\_accuracy = (100 * n\_correct) / k \qquad (Eq.\ 4)$$

In the equation 4 ***k*** is the number of trials and ***n_correct*** is the number of correct predictions in all the trials. We call this metric ***Validation_accuracy.*** Which is used during training to check the model performance on our validation-set. This will give us an idea of where the model is in terms of quality of classification during training.

It should be noted that this is the validation accuracy and is different from the final classification accuracy that is used to evaluate the model performance on the test-set (see eq.11). Because, at the validation step, the goal is just to select the models that are doing well on the classification of the validation-set. So later on, after we save those good candidate models during the training, we can do a more detailed analysis of the models over the test-set. The evaluation on the test-set is in terms of classification metrics like Precision, Recall, and Accuracy. The accuracy we used at the test level is the ratio of all True Positives to all (true positives and negatives) which is mentioned in eq.11 Whereas, the accuracy used in the validation phase is just how many True Positives we get over the validation-set. This accuracy is called validation-accuracy.

So now we can evaluate the model during training. It tells us which model we should pick as the final model for the testing phase. To select the best model, we followed the following logic. If the model accuracy is higher than a threshold (**60%**) over the validation-set during training, then we save the model. We follow this approach but each time our model gets better accuracy we raise the threshold and save the new model too.

For example, if in the next run the accuracy of the model over the validation-set becomes 70% the threshold would be raised to 70%, and we save the new model too, and so on and so forth. Then finally we would select models that have higher accuracy on validation to test them on the test-set. It should be noted that if we have multiple models with the same validation accuracy (i.e., 100%) then we would keep saving the last 5 models during the training. We explain this procedure in the form of a pseudo code in the next section.

7.2 Pseudocode for training and validation:

Here, we modularized the training the model and validation of the model in two functions. First function is **N_way_validation** assigns an accuracy to the trained model. The trained model is obtained by **train_and_validate** function which is the second function we mention here. **train_and_validate** function trains the model, passes it to the **N_way_validation** function to evaluate its accuracy over the **validation-set, saves** the model if its accuracy is above the threshold, and increases the threshold incrementally to keep the models with highest accuracy.

Pseudo code 1

```
# N is number of classes of signatures exists in validation-set
# The model is already trained over the training-set.
# k is a fixed number like 20 or 100.

Function N_way_Validation (Number_of_classes = N, model):
  correct_classification_counter = 0
  Repeat K times:
      -  Randomly select an image A in the validation-set.
      -  Create N pairs from N different classes of validation-set.
      -  Evaluate each pair with the currently trained model.
      -  if model classify image A correctly among N classes:
           o  correct_classification_counter += 1
  validation_accuracy = correct_classification_counter / K
  Return validation_accuracy
```

Pseudo code 2

# This function trains the model on training set and keeps track of the model accuracy, if newly trained model at every 5 iterations (user can change this value) has higher accuracy than the previous trained model's accuracy, it is saved and the accuracy threshold is updated to the latest highest accuracy.

# **number_of_iterations**, is how many times we run and fit the model over randomly selected 32 batches of pairs of images from the training-set.

# **N** is the number of classes of signatures that the model is training on.

```
Function train_and_validate (number_of_iterations, N = number_of_classes):
  accuracy_threshold = 60%
  For 0 to number_of_iterations do:
      - Randomly create a batch of 32 pairs of images from training_set.
      - Train the model with these 32 pairs of images.
      - Every 5 iterations do:
          - validation_accuracy = N_way_Validation(N, model)
          - If (validation_accuracy > accuracy_threshold):
            then:
              o A good candidate model is trained, save this model.
              o Update the accuracy_threshold = validation_accuracy
```

7.3 Conclusion of training and validation

Siamese-nets are pairwise models. Hence, to train them we need a pair of images. Output or target is set to either one or zero, if the input images respectfully belong to the same class or not. To be able to train, test and validate, we advise an approach (N-Way comparison) to measure the performance metric. To see if a model can classify a given query image, we need to have a reference sample of each of the existing classes to compare the query image against it. Hence, N-way comparison is used to compare the query image with N other classes. In this Approach N pairs of images have been constructed. Among those N pairs, only one pair has both the query and the reference image from the same class of signatures. The rest of the N-1 pairs have pairs of different classes. So, the target values used to train these sets would have only 1 value equal to one and the rest of the N-1 target values are equal to zero.

For example, in case we want to train over the total number of classes (N = 55), first we make 55 pairs of images where only one pair of classes matches. Then we randomly select a batch. For example, for a batch of 32 we would randomly create 55 pairs of images then select 32 of these pairs randomly.

8. Results and analysis

8.1 Baseline

To evaluate the performance of the method we used the Convolutional Neural Network as the baseline classification model for our signature's classification. CNN can be one of the efficient tools for extracting the common features among different classes [53]. So, we decided to run a test on a simple CNN structure for classification purposes. Training is done from scratch like the Siamese network.

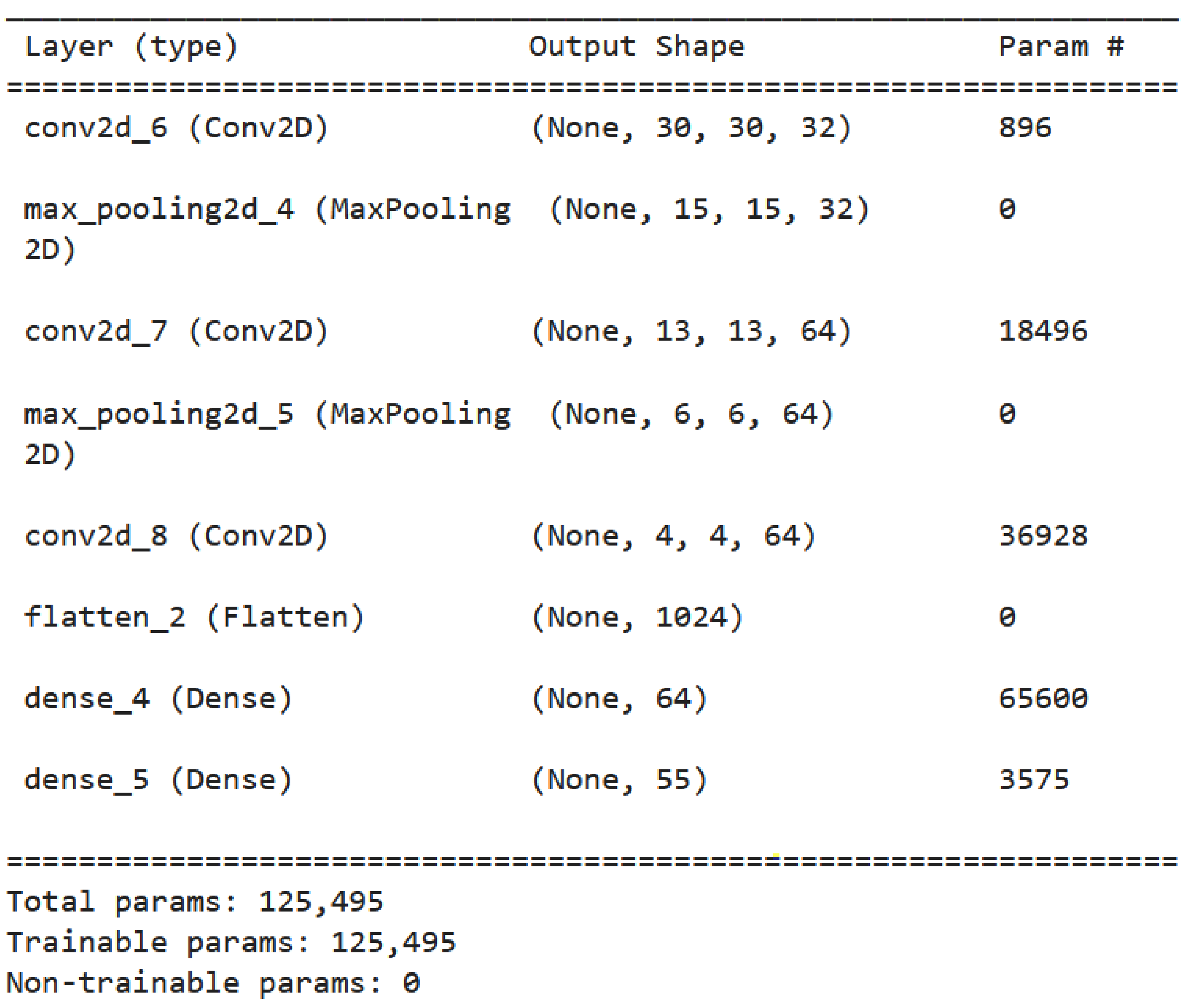
8.1.1 - Dataset

The Dutch dataset [52] has been used in this case. Here, we selected 55 different classes of signatures. In other words, it's a dataset of signatures of 55 different people. Number of signature samples for each class of signatures is around 12 images. The dataset containing the 55 different classes of signatures is divided into 3 sets. Train, test, and validation. We assigned 6 to train, 3 to validate, and 3 to test.

8.1.2 - Structure of the network

In Figure 14 A simple CNN's structure is used for classification of 55 classes of signatures. The input sizes of the model changed for several experiments we did over input image sizes. We did experiments with input size of 32 pixels up until 100 pixels.

This CNN structure is composed of the first two conv layers (conv2d_2 and conv_2d_3) that each are followed by a separate max pooling layer (max_pooling2d_2 and max_pooling2d_3). A third conv layer (conv2d_5) is used on top of the previous max pooling layers and its output is flattened to a one-dimensional vector. The length of this vector is again decreased over two dense layers (dence_2 and dence_3) of sizes of 64 and 55 to be able to classify the given images to one of 55 classes.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_4 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_7 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_8 (Conv2D)           (None, 4, 4, 64)          36928

 flatten_2 (Flatten)         (None, 1024)              0

 dense_4 (Dense)             (None, 64)                65600

 dense_5 (Dense)             (None, 55)                3575

=================================================================
Total params: 125,495
Trainable params: 125,495
Non-trainable params: 0
```

Figure 14: Structure of the simple CNN network for classification of 55 classes of signatures. for input images of 32x32 pixels.

Also, for the sake of not losing any details in our images, we kept the color channels and avoided converting the images to grayscale mode.

8.1.3 - Training

We divided the dataset to train, test and validation. In the train-set, each class has 6 images and the test and validation-set have 3 images. Here, we performed 3 different experiments and altered the training hyperparameters in terms of image input size, number of classes, and activation function to check accuracy of the model.

We stopped the training after the accuracy of validation converged after 100 iterations. Here, iteration means the number of times the model is fitted over the given training-set. We see that in almost 40 iterations the model converges on its performance. We didn't go after 100 to prevent the model from beginning to remember the training-set and overfit. Overfitting is a concept in machine learning which happens when a model learns very well on the training set but performs poorly on test data. We were careful here knowing the training-set had only 6 samples per class so stopping it early as soon as it reached its accuracy plateau to prevent the overfitting impact on the model.

Experiment_1: image size 32x32 pixels

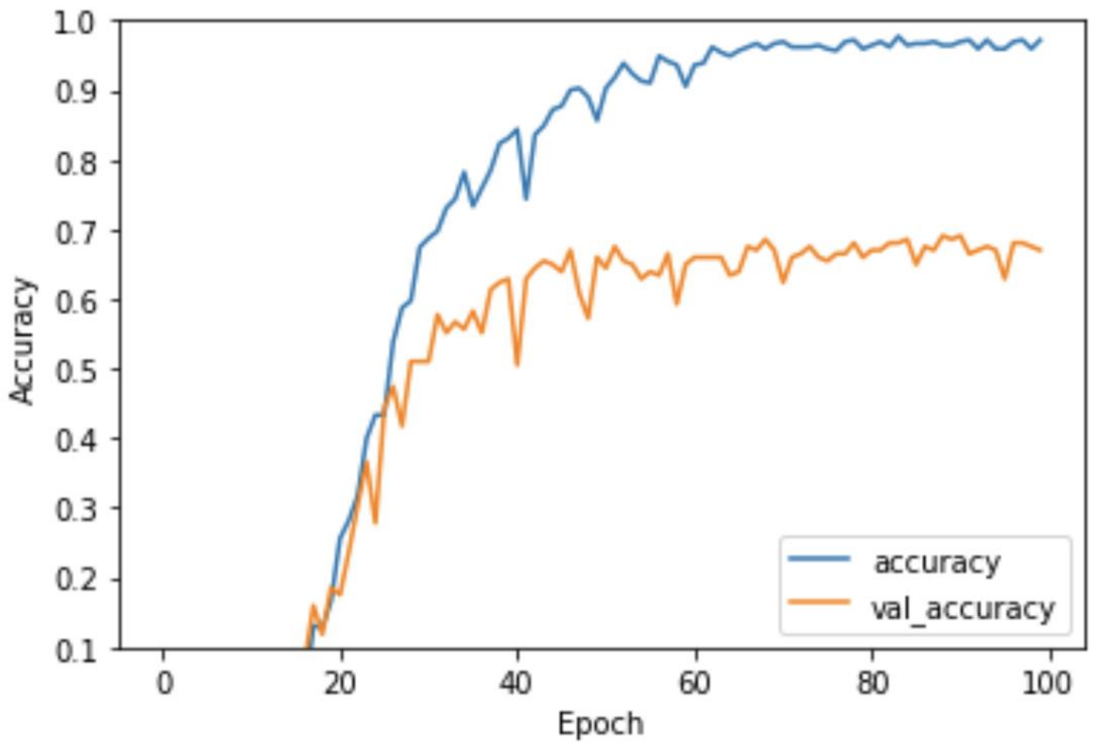


Figure 15: The accuracy of the CNN model during the training for the classification of 55 classes of signatures. Blue is the model accuracy. on the training-set and yellow is the model accuracy on the validation-set. Image size: 32x32, activation: Relu.

Experiment 2: Image size: 32 x 32 pixels, activation function: Sigmoid.

In the model structure we had two dense layers dense_6 and dense_7 shown in Figure 14. at the end of the pipeline. For the dense layer right before the last layer (dense 6) we were using the ***Relu*** activation function. In the second experiment (next chapter), we changed it to change that to ***sigmoid*** to check the performance of the model accuracy. The last layer (dence_7) has a ***softmax*** activation function. Given an image as an input the Softmax function calculates the probability of classification of that image for each of 55 classes.
We see an improvement of around 10% in validation accuracy while training the model with a sigmoid activation function.

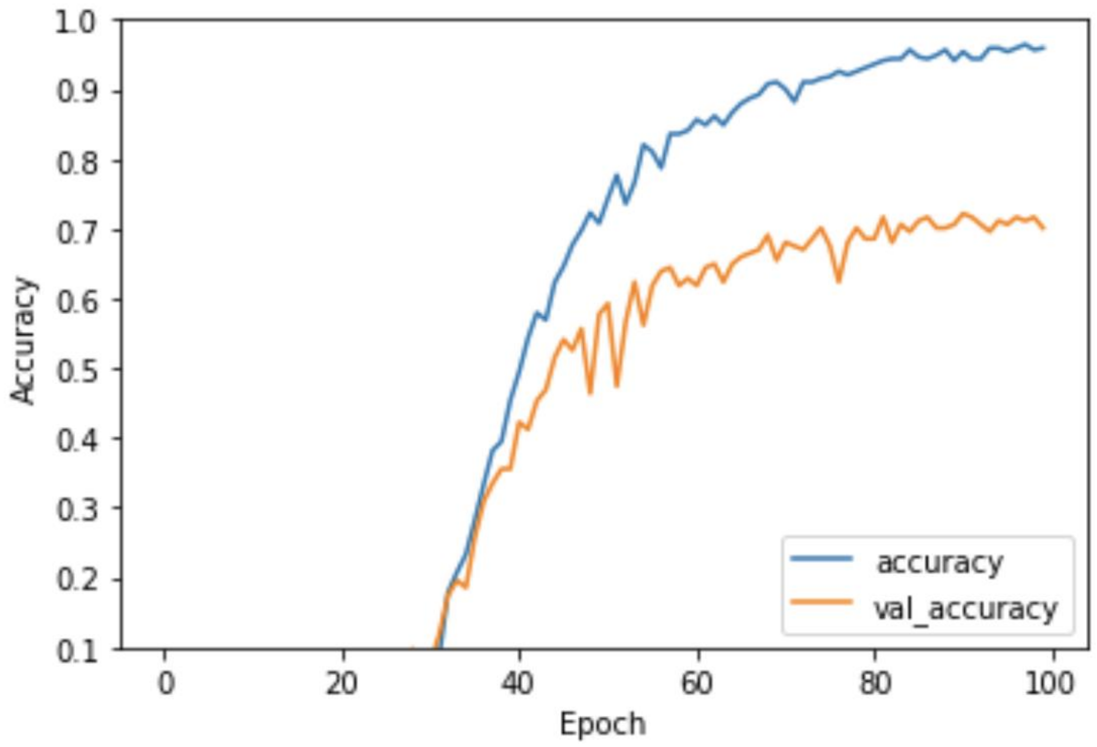


Figure 16: Mode's accuracy of the model on training-set and validation-set on the classification of 55 classes with CNN. Image size 32x32 and activation function: sigmoid.

Experiment 3: Image size: 100X100 pixels, activation function: Sigmoid

Now that we have selected our activation function, we can do an experiment on another parameter which is the input size of the CNN model. We increased the signature image to 100x100 pixels. The training was done on the same dataset as before with 55 classes. Training accuracy on both training-set and validation-set has been converged after 30 iterations.

The Validation accuracy this time increased to 78%. That's the point where we stopped the training at the 100 iterations. The model reached its plateau of 78% accuracy on validation at around iteration 40 as shown in Figure 17.

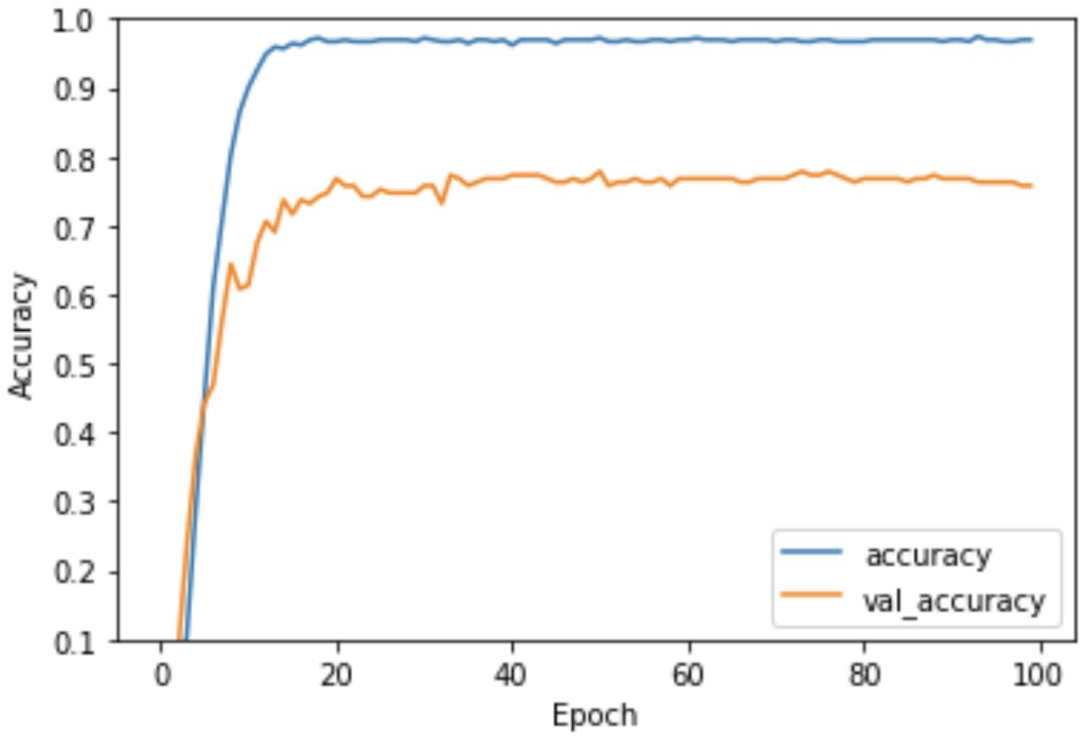


Figure 17: The model's accuracy on training-set and validation-set on the classification of 55 classes with CNN. Activation: sigmoid, size: 100x100.

8.1.4 - Test CNN on 55 classes

In previous sections we trained classification models. Now we are going to evaluate their performance on the test-set. The evaluation metric is accuracy, precision-recall, and F1 score. This accuracy is the ratio of all True positives over (TPs + TNs) that is mentioned in eq. 11. We calculate the metrics on all the previous experiments to select the best hyperparameters mentioned above.

Experiment_1: Image size: 32x32 pixels, activation function: Relu

Testing on the trained CNN model with 32x32 images results in an accuracy of 62% (Table 1). This makes sense as the number of training on images was so low that the model couldn't pick up meaningful features to discriminate the classes from one another.

Table 1. On 32x32 images, the Precision, and Recall related to the classification of 55 classes with CNN.

	Precision	Recall	F1-Score
Accuracy			62%
Macro Average	66%	62%	61%
Weighted Average	67%	62%	61%

Experiment_2: image size:32x32 pixels, activation function: Sigmoid

Test results on the model that trained with 32x32 images but with activation function of sigmoid shows 10% improvement on classification accuracy over test-set (Table 2) in comparison with Relu activation function.

Table 2. On 100x100 images the precision-Recall related to the classification of 55 signatures with CNN.

	Precision	Recall	F1-Score
Accuracy			70%
Macro Average	74%	69%	68%
Weighted Average	73%	70%	67%

Experiment_3: image size 100x100, activation function softmax

Here, the model that is trained with images of sizes of 100x100 pixels and softmax activation function is tested. The accuracy was improved by 6% from 70% to 76% as shown in Table 3.

Table 3. On 100x100 images and activation function of sigmoid the precision-Recall related to the classification of 55 signatures with CNN.

	Precision	Recall	F1-Score
Accuracy			76%
Macro Average	82%	76%	76%
Weighted Average	82%	76%	76%

8.1.5 - Baseline conclusion

Based on the results, we found out that a CNN architecture has the potential of picking up meaningful and distinguishable features if the image size is increased to 100x100 pixels and softmax is used as an activation function. We could get up to 76% classification accuracy. It also shows that the limitations of CNN are that if we focus on similar features to classify the signatures, we need more images than just 6 samples per class. Also, we found out that by using the Sigmoid function instead of Relu in the last dense layers could increase the classification accuracy.

8.2 - Our approach

Our classification approach is based on training the model with pairs of images in the training-set. The trained model would be a binary classification for each pair. Indicating if these two pairs belong to the same class or not. To find the best hyperparameters for training the model we ran several experiments that are explained in detail.

On the inference side and testing, knowing every image's true class, we would compare each class image with the query image. Average of the similarity for each class would be the similarity of the quarry image with that class. Finally, the query image would be classified based on the highest similarity with each class. Here, this process is explained in detail.

To find the best setup and hyperparameters for the Siamese nets, several experiments have been performed. We began with low values in hyperparameters (i.e. image and class sizes) and then increased them to track the performance of the model. For example, we began classification on only 10 classes and then increased it to the maximum existing number of classes in our dataset (55).

Here, we began with 32x32 then 70x70, and then 100x100 size for input images. Also, we did some experiments in terms of feeding the model with more pairs of the same class (sequential batches) vs more random pairs (random batches). It turns out that to get the optimal performance the best hyperparameters to train the model with are 100x100 pixels images in random pairs.

8.2.1 Training

We did our training based on what has been explained in Chapter 7. First, we trained our model with 32x32 pixels images. We began to track the accuracy of the model on validation-set while we were training the model on the training-set. This way we let the model continue learning and also continue the generalization over the validation-set. The metric used to evaluate the model on the validation-set is based on **N_way_comparision** which is explained in chapter 7.

Also, for sake of simplicity, we began our experiments with only 10 classes to classify with siamese-net then increased it to 55 classes after fine-tuning the model. Hence, we used 10_way_comparision for the first iterations and then increased it to 55_way_comparision.

As a reminder, N_Way_comparision was a method developed in previous sections to validate a siamese model. of sampling from the validation-set and calculating the model performance over the classification of N classes in the validation-set during the training phase. Obtained accuracy is called ***validation_accuracy*** and if the model trained at the designated epoch (i.e., every 20 epochs) has a higher than 60% threshold ***validtion_accuracy***, the model is a good candidate and would be saved for the testing phase.

We stopped the training procedure when the ***validation_accuracy*** reached 100% and we already saved 5 models with 100% ***validation_accuracy*** or we reached the maximum number of epochs. Meaning the model is doing well on the validation-set and we can stop training and proceed to the testing phase.

Below we explain how we used similarity values for classification and later on we explained in detail how we have evaluated these classifications in terms of Precision-Recall and F-score.

8.2.1 - Similarity used for classification

Having the Siamese model trained and saved in the training phase, we can evaluate its performance on unseen signatures in the test-set at the inference phase.

For example, to decide if a query image belongs to class ***k*** or not, we compare the query signature with all the sample signatures in reference class ***k*** as follows.

$$Similarity_k = \frac{\sum_{j=1}^{n} \quad similarity(query\ image_j)}{n} \qquad \textit{(Eq. 5)}$$

Here, k is the class and j is the number of signature images that exist in the class k (1…n). In other words, similarity of a query image with a class is the average of similarity of query image with all the available images from that class. This strategy comes from the fact that in reality, we don't have lots of reference signatures of a person (a class) in our datasets (i.e., bank's client's signatures). Hence, to see if a new query signature belongs to a person (a class of signatures), we take the average of the similarity of that query image with all the signatures available for that specific person. In our cases the number of signatures per class is low, around 3-6 images.

After being able to assign a similarity of a query signature to a specific class. We followed the same procedure for all the 55 classes of signatures. We have to apply the eq.5 on all the 55 classes (k =1…55). Then a class with the highest average of similarity is selected as the final class of the query image. This procedure is done by applying an argmax over calculated similarities for existing classes as shown in eq. 6.

$$class_{query} = argmax(similarity_k)_{k=1..55} \qquad \textit{(Eq. 6)}$$

In the test-set we have the actual class for every signature image which we use as the ground truth when we evaluate our model classification performance. We test the previously trained model (on training-set) against the test-set images by randomly selecting the images from the different classes. Then using the eq.5 and eq.6 to see if the model could classify the query correctly or not. Using this statistic we can calculate the model performance with a confusion matrix, the precision, and recall for each of the 10 classes of signatures. Below we directly mention the iterations we used for fine tuning the model in the test phase with the hyperparameters we used for training in different experiments. Also, we explain the evaluation metrics.

Experiment_1: 10 classes and image size 32x32

We began our experiments with the small signature classes and signature sizes. This way it takes less time (in terms of training and evaluating each model) to find the best hyperparameters to get better accuracy for the classification models. The hyperparameters that are selected and changed are explained in the following experiments along with the model performance metrics. In this experiment, we trained a model to classify 10 randomly selected classes with signatures of 32x32 pixels. The model had around 60% of accuracy (eq.11). Then, we increased the image size to 70x70 pixels and redo the same experiment.

Experiment_2: 10 classes and image size 70x70

We increased the size of input images to 70x70 and redo the training with the previous setting. 32 batch size, 1000 iterations, and ***10_way_comparision*** at training time, also do the validation_accuary every 5 iterations. The accuracy obtained at this ***Experiment_2*** was higher than ***Experiment_1***. Hence, we performed a more detailed evaluation in terms of confusion matrix, Recall, and Precision too.

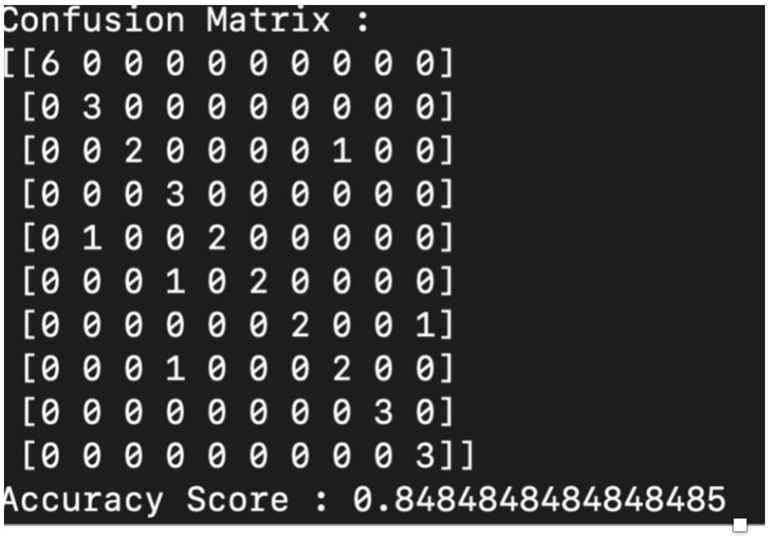
8.2.2. Performance Metrics

After classifying the test-set, to analyze the performance of the classification we adopt the following metrics.

Confusion matrix

Confusion matrix is a table that explains the performance of a model in classification. Each row indicates the classes to classify and each column shows the classes that model predicted. In our case we intend to classify samples of 10 classes. Hence, our confusion matrix is 10 by 10. For example, for images that belong to class 1 (row 1), shows that there was a total of 6 images in class 1 and the model classifies all the 6 images correctly as class 1 (put all the 6 images in column 1). Another example, for images belonging to class 3 (row 3) we see that the model classifies 2 of them as class 3 (put them in column 3) and 1 of the images as class 8 (1 record is mentioned in column 8). So, if the model is more populated in the diagonal part of the confusion matrix, it says that the model is less confused and could predict most of the classes correctly as we see in the Table. 4.

Table 4. Confusion matrix over 10 different classes with batch training of (70x70) samples.



```
Confusion Matrix :
[[6 0 0 0 0 0 0 0 0 0]
 [0 3 0 0 0 0 0 0 0 0]
 [0 0 2 0 0 0 0 1 0 0]
 [0 0 0 3 0 0 0 0 0 0]
 [0 1 0 0 2 0 0 0 0 0]
 [0 0 0 1 0 2 0 0 0 0]
 [0 0 0 0 0 0 2 0 0 1]
 [0 0 0 1 0 0 0 2 0 0]
 [0 0 0 0 0 0 0 0 3 0]
 [0 0 0 0 0 0 0 0 0 3]]
Accuracy Score : 0.8484848484848485
```

Table 5. Performance of the model over the 10 classes with sequential batch training of (70x70) samples.

	Precision	Recall	F1-Score
Accuracy			85%
Macro Average	88%	83%	83%
Weighted Average	89%	85%	85%

Precision

Precision shows how precise is the model in classification and is calculated as follows:

$$Precision = TP/(TP+FP) \qquad (Eq.\ 7)$$

In this equation, TP is True Positive (in this case signature is correctly classified) and FP stands for False Positive (cases that model wrongly classified as a correct class) in other words misclassifications. As we see in Table 5 the precision is above 80%. Overall precision is depicted by macro and weighted average precisions of all the classes. Weighted average precisions consider the number of the images in the dataset too. This means that the model is accurate at 83% of its classification.

Recall

The Recall is basically the detection rate or the ability of the model to identify and classify all the samples of relevant class and is calculated as follows.

$$Recall = TP/(TP+FN) \qquad (Eq.\ 8)$$

In this formula, FN is False Negative and are those positive samples of each class that the model missed or couldn't classify correctly. Table 5 shows that the model performs well in finding all the samples. In another word, we have fewer False Negatives or missing some cases by misclassifying them to another class. We observe that the model could detect 83% of the signatures of each person on average.

Macro Average:

Macro average here is basically the simple average or arithmetic mean of all the values of Precision and Recalls.

Weighted Average:

Macro average is a simple average of all the precision and recalls over 10 classes. Whereas, Weighted average is a weighted average. This metric considers how many samples of each class were present in its calculation. So, fewer samples of one class means that its precision/recall/F1 score has less of an impact on the weighted average for each of those metrics. The Weighted average is the average calculation based on the number of images (support) as follows:

$$W = \frac{\sum_{i=1}^{n} w_i X_i}{\sum_{i=1}^{n} w_i} \qquad (Eq.\ 9)$$

Here, in this equation w_i is the number of images in the class i and $\boldsymbol{X}$ is the Precision or Recall.

F1 score

F1 score is the harmonic average of recall and precision. F1 score shows the balance between Precision and Recall. It tells you how precise your classifier is (how many instances it classifies correctly). F1 Score tries to find the balance between precision and recall. Usually, there is a trade-off between precision and recall. the F1 Score is a combination of these two as follows:

$$F1\ score = 2\ x\ [\ (precision\ x\ recall)\ /\ (precision+recall)] \qquad (Eq.\ 10)$$

A higher value of F1 Score means that the model performs well in both finding all the samples and also performs well in distinguishing them from other classes. The range for F1 Score is **[0, 1]**. When both precision and recall is zero, we have minimum value for F1 score and when both precision and recall are 1, meaning the model has neither False Positive nor False Negative.

Accuracy

Accuracy simply is the ratio of correctly predicted observations to the total observations. One For our model, we have got 84% which means our model is 84% accurate in drawing the line between classes.

$$Accuracy\ score = \frac{TP+TN}{TP+TN+FP+FN} \qquad (Eq.\ 11)$$

8.2.3. Experiment_3: Sequential batch training

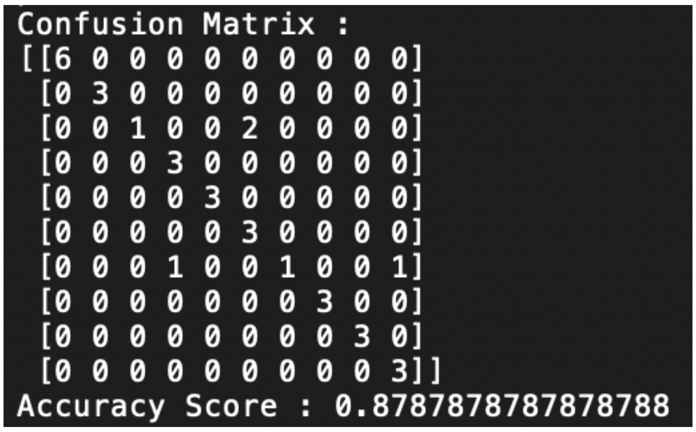
Here, we made another experiment that instead of randomly creating the pairs for Siamese net to train on, for a 32-batch of pairs, we made sure that half of it contains pairs of the same class and the other half pairs of negative classes. But the accuracy of the model over the test-set decreased to around 70%. So, we went back to training over random pairs and the performance increased. This is explained in the following section in detail.

8.2.3. Experiment_4: Random batch training

Unlike the previous section here we completely randomly select the 32 batches of pairs over all the classes of signatures. As our dataset is balanced, we got better results.

The confusion matrix over 10 classes of signatures is as follows:

Table 6. Confusion matrix over 10 classes of signatures with random batch training of (70x70) samples.



```
Confusion Matrix :
[[6 0 0 0 0 0 0 0 0 0]
 [0 3 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 2 0 0 0 0]
 [0 0 0 3 0 0 0 0 0 0]
 [0 0 0 0 3 0 0 0 0 0]
 [0 0 0 0 0 3 0 0 0 0]
 [0 0 0 1 0 0 1 0 0 1]
 [0 0 0 0 0 0 0 3 0 0]
 [0 0 0 0 0 0 0 0 3 0]
 [0 0 0 0 0 0 0 0 0 3]]
Accuracy Score : 0.8787878787878788
```

Table. 6 shows that the model is less confused in the classification of a given new signature in comparison with sequential batch training. Also, the accuracy score has improved from 3% to 87%.

Table 7. Precision and recall for all of the 10 classes of signatures with random batch training of (70x70) samples.

	Precision	Recall	F1-Score
Accuracy			88%
Macro Average	91%	87%	85%
Weighted Average	92%	88%	86%

As shown in Table 7 the average micro-precision of this model has also improved to around 91%. Meaning that model is doing better in terms of encountering fewer False Positives and classifying each new signature. We also observe an increase in the macro average of recalls over all the 10 classes to 87%.

Results have improved in comparison with the previous iteration. This indicates that the model is doing well with maximizing the distance between dissimilar images. Because in the case of random pairs, the model is presented mostly with the different class pairs and the whole Siamese net approach tries to classify the pairs over the distance space. It shows that the model is doing better on learning the distances between signatures rather than similarities.

8.2.4. Experiment_4: image size 100x100

Following the trend in previous iterations, increasing the image dimensions was improving the model's accuracy. Hence, we did another iteration with **(100 x 100)** pixels and we trained them with the same training hyperparameters like **1000** epochs and random **32** batch size. The model loss at each iteration is depicted in Figure 18.

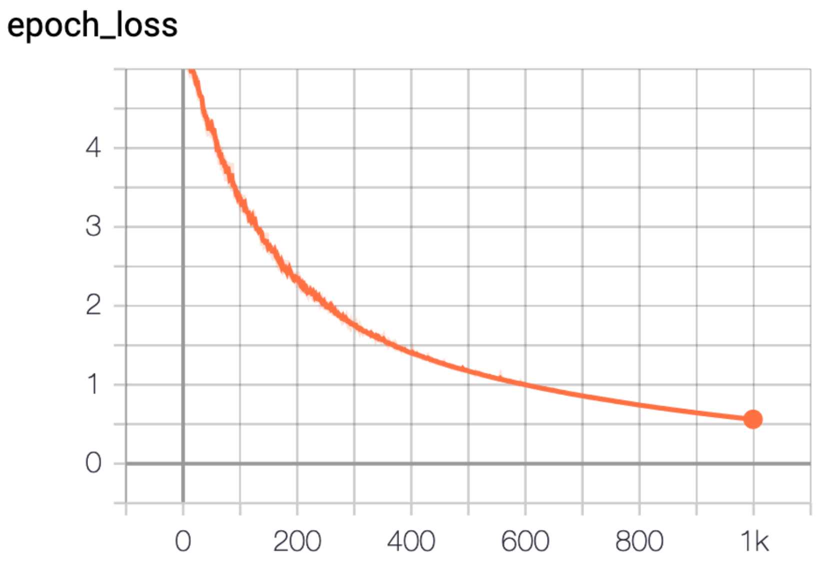
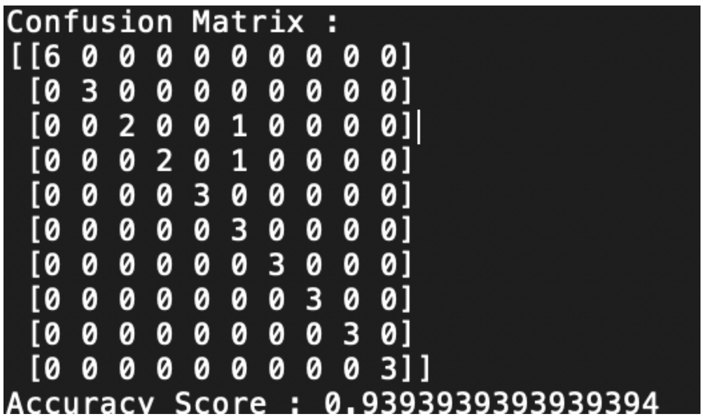


Figure 18: Training loss for 10 classes.

Results show that increasing the sample sizes helps the model to get less confused between the classes of signatures. This yields higher accuracy in terms of classification to 94% as shown in Table 8.

Table 8. Confusion matrix over 10 classes of signatures with random batch training of (100x100) samples.



```
Confusion Matrix :
[[6 0 0 0 0 0 0 0 0 0]
 [0 3 0 0 0 0 0 0 0 0]
 [0 0 2 0 0 1 0 0 0 0]
 [0 0 0 2 0 1 0 0 0 0]
 [0 0 0 0 3 0 0 0 0 0]
 [0 0 0 0 0 3 0 0 0 0]
 [0 0 0 0 0 0 3 0 0 0]
 [0 0 0 0 0 0 0 3 0 0]
 [0 0 0 0 0 0 0 0 3 0]
 [0 0 0 0 0 0 0 0 0 3]]
Accuracy Score : 0.9393939393939394
```

Model performance in terms of precision and recall is also improved as shown in Table 9. The macro average of both is over 93% which yields the F1 Score to improve also to 93%.

Table 9. Precision and recall for all of the 10 classes of signatures with random batch training of (100x100) samples.

	Precision	Recall	F1-Score
Accuracy			94%
Macro Average	96%	93%	93%
Weighted Average	96%	94%	94%

8.2.5. Experiment_5: Increasing the number of classes to 55

In previous sections we were fine tuning a model with smaller sample size, and smaller iterations. After finding the hyperparameters that result in higher performance, the model will be trained for the maximum number of classes to see how our classification method performs over 55 classes and 4000 iterations.

So, we increase the ***N_way comparison*** to ***55_way_comparison*** at the validation phase. The validation process repeats every 20 iterations and the model with the best performance would be saved (eq.4) as explained in Chapter 7.

Results of the signature classification over Dutch dataset with 55 classes are shown in Table 10. As we can see the accuracy (eq.11) is 90% whereas baseline has the accuracy of 76% in chapter 8.1.4.

Table 10. Model performance of classification over 55 classes.

	Precision	Recall	F1-Score
Accuracy			89%
Macro Average	85%	89%	86%
Weighted Average	85%	89%	86%

8.2.6 Test on Chinese dataset and comparisons

We tested our approach with another dataset that was collected from Chinese signatures [54]. It has 20 different classes, which contain 240 samples. A sample of it is shown in Figure 19.

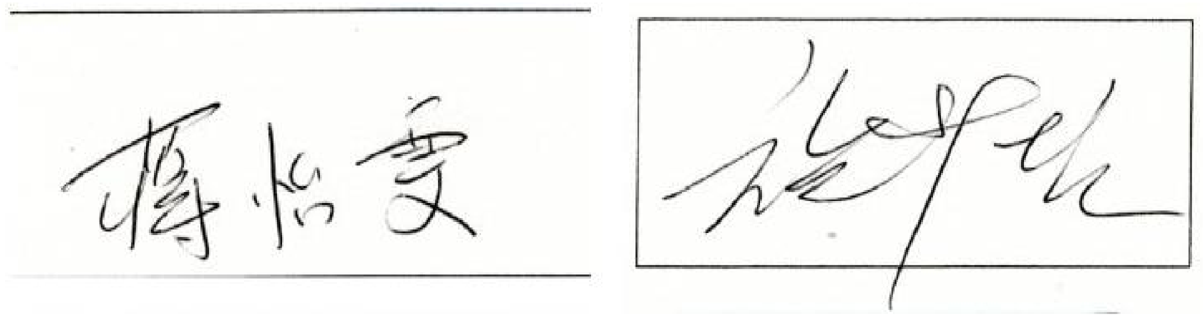


Figure 19: Samples of signatures from the Chinese signature dataset.

For each class, we assigned 6 images for the training and 3 for tests, and 3 images per class for validation. Our model's classification performance is presented in Table 11.

Table 11. Model performance of classification Chinese signatures dataset

	Precision	Recall	F1-Score
Accuracy			84%
Macro Average	76%	84%	79%
Weighted Average	76%	84%	79%

Table 11 shows Siamese-net performance on classification of the second dataset (Chinese signature dataset). The model had a high accuracy of 84% and F1 score of 79%.

8.3 Conclusion of our approach

On the training side, best hyperparameters for training the model are found over an experiment of classifying only 10 classes of signatures. Using these hyperparameters we trained the model over the whole 55 classes on Dutch dataset and 20 classes on the Chinese dataset.

On inference phase, the results of the model's accuracy on test-set for Dutch dataset was 89% and 84% on the Chinese test-set. Process of classification over N classes is based on comparing the query image with those classes. Each class contains several (3 - 6) images. So, the similarity of the query signature is determined by the average of the similarity of the query with all the images in that class (3-6) eq 5. Finally, the class with maximum similarity would be selected as the signer of the query image, like Figure 13, eq 6. The performance of the Siamese-net is compared with the base model in next chapter.

9. Comparison

In previous experiments the models have been fine tuned so best hyperparameters in training were selected. Here, to simplify the comparison of baseline (CNN classification) and our approach we summarize the performances mentioned in previous tables (Tables 3,10,11). Among the metrics in those tables the accuracy metric has been selected to summarize and compare the performance of the models in previous experiments. Accuracy metric explains the model performance as the ratio of the all the True classifications (True Positives + True Negatives) to all the dataset which contains the False Negatives and False Positives (TP + TN + FP + FN) eq.11.

Table 12. Comparison results between Dutch and Chinese dataset.

	Accuracy	
	Dutch	Chinese
Siamese-net	**89%**	**84%**
CNN (Base line)	76%	78%

In Table 12 we observe that Siamese-net could improve the performance of classification on both datasets. Reason is that simple CNN architecture only relies on similar features of intra class signatures. On the other hand, Siamese networks leverage the pairwise training to focus on extracting features that minimizes the distance for intra class samples (similar signatures) and maximizes the distance between inter signatures (non similar signatures). Hence not only focuses on similarity of images but also leverages the dissimilarity of them for classification.

10. Conclusion and Future works

Our experiments show that Siamese networks could perform well in the classification of offline signatures with a low (3 to 6) number of samples per class. In this thesis, we could reach 89% accuracy in offline signature classification using Siamese nets on Dutch dataset and 84% on Chinese dataset. Model is trained to extract features that maximises the distance between features of inter-class images and minimizes the feature distances between intra-class images. Tuning the model plays an important role in optimal performance. Experiments show that having larger sizes of samples around 100x100 pixels with random sampling to train the pairs improves the performance of classification.

Here, it has been demonstrated that simple CNN (base line) has a problem picking meaningful similarities for classification purposes. Whereas, the Siamese network is a more reliable method to do classification over a small or large number of classes with a low amount of training pics like 6 images per class.

Advantage of Siamese-net classification is that one model can be trained for all the users, one model per user and still have a high level of accuracy around 90%. Also, as the classification is decided based on distances between two signatures (reference and query signatures), models trained with Siamese-net would be scalable. In this sense that in case a new class is added to the dataset that is not trained on we can still use the Siamese net to calculate the similarity of the query image with the new class.

For future works, one can use this approach to measure the similarity of the query signatures with the reference signatures to determine the authenticity of the query signature. Also, one can use the transfer of learning of currently learned weights across all the classes. This means that we can use the weights of the model trained in the first task (signature classification) as an initialization of the second model (signature authentication) in case of training a writer dependent model per user, instead of training it from scratch.

References

[1] Dey, Sounak, et al. "Signet: Convolutional Siamese network for writer independent offline signature verification." *arXiv preprint arXiv:1707.02131* (2017).

[2] Chen, Yunpeng, et al. "Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019. Retrieved from https://www.slideshare.net/ShunYUKo/octave-convolution

[3] towardsdatascience.com/one-shot-learning-with-siamese-networks-using-keras-17f34e75bb3d

[4] Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." *ICML deep learning workshop*. Vol. 2. 2015.

[5] Keri L. Kettle, Gerald Häubl, The Signature Effect: Signing Influences Consumption-Related Behavior by Priming Self-Identity, Journal of Consumer Research, Volume 38, Issue 3, 1 October 2011, Pages 474–489, https://doi.org/10.1086/659753

[6] Poddar, J., Parikh, V., & Bharti, S. K. (2020, April 14). Offline signature recognition and forgery detection using Deep Learning. sciencedirect. Retrieved January 2, 2022, from https://www.sciencedirect.com/science/article/pii/S1877050920305731

[7] Kennard, D. J., Barrett, W. A., & Sederberg, T. W. (2012, November). Offline signature verification and forgery detection using a 2-D geometric warping approach. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012) (pp. 3733-3736). IEEE.

[8] Khalajzadeh, Hurieh, Mohammad Mansouri, and Mohammad Teshnehlab. "Persian signature verification using convolutional neural networks." International Journal of Engineering Research and Technology 1.2 (2012): 7-12.

[9] Calik, Nurullah, et al. "Large-scale offline signature recognition via deep neural networks and feature embedding." Neurocomputing 359 (2019): 1-14.

[10] Kaulen, Diego, and Kaitlyn Baab. "Offline Signature Recognition with Convolutional Neural Networks."

[11] Padmajadevi, G., and K. S. Aprameya. "A review of handwritten signature verification systems and methodologies." 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). IEEE, 2016.

[12] Zhu, Guangyu, et al. "Signature detection and matching for document image retrieval." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.11 (2008): 2015-2031.

[13] Ashwin, C. S., et al. "PIXBAS" Pixel Based Offline Signature Verification"." *Adv. Inf. Sci. Serv. Sci.* 2.3 (2010): 14-17.

[14] Shekar, B. H., Bharathi Pilar, and K. D. S. Sunil. "Blockwise binary pattern: a robust and an efficient approach for offline signature verification." *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 42 (2017): 227.

[15] Wang, Pin, En Fan, and Peng Wang. "Comparative analysis of image classification algorithms based on traditional machine learning and deep learning." *Pattern Recognition Letters* 141 (2021): 61-67.

[16] Eskander, George S., Robert Sabourin, and Eric Granger. "Hybrid writer-independent–writer-dependent offline signature verification system." *IET biometrics* 2.4 (2013): 169-181.

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

[18] Yılmaz, Mustafa Berkay, and Berrin Yanıkoğlu. "Score level fusion of classifiers in off-line signature verification." *Information Fusion* 32 (2016): 109-119.

[19] Taigman, Yaniv, et al. "Deepface: Closing the gap to human-level performance in face verification." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

[20] Cavalcanti, G. D. D. C., Rodrigo C. Doria, and E. Cde BC Filho. "Feature selection for off-line recognition of different size signatures." *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*. IEEE, 2002.

[21] Oliveira, Luiz S., et al. "The graphology applied to signature verification." *12th conference of the international graphonomics society*. 2005.

[22] Patil, Pallavi, et al. "Offline signature recognition system using histogram of oriented gradients." *2017 International Conference on Advances in Computing, Communication and Control (ICAC3)*. IEEE, 2017.

[23] Harfiya, Latifa Nabila, Agus Wahyu Widodo, and Randy Cahya Wihandika. "Offline signature verification based on pyramid histogram of oriented gradient features." *2017 1st International Conference on Informatics and Computational Sciences (ICICoS)*. IEEE, 2017.

[24] Yilmaz, Mustafa Berkay, et al. "Offline signature verification using classifier combination of HOG and LBP features." *2011 international joint conference on Biometrics (IJCB)*. IEEE, 2011.

[25] Zhang, Bailing. "Off-line signature verification and identification by pyramid histogram of oriented gradients." *International Journal of Intelligent Computing and Cybernetics* (2010).

[26] Soleimani, Amir, Babak N. Araabi, and Kazim Fouladi. "Deep multitask metric learning for offline signature verification." *Pattern Recognition Letters* 80 (2016): 84-90.

[27] Bouletreau, Viviane, et al. "Handwriting and signature: one or two personality identifiers?." *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No. 98EX170)*. Vol. 2. IEEE, 1998.

[28] Zouari, Ramzi, Raouia Mokni, and Monji Kherallah. "Identification and verification system of offline handwritten signature using fractal approach." *International Image Processing, Applications and Systems Conference*. IEEE, 2014.

[29] Pourshahabi, Muhammad Reza, Mohamad Hoseyn Sigari, and Hamid Reza Pourreza. "Offline handwritten signature identification and verification using contourlet transform." *2009 International Conference of Soft Computing and Pattern Recognition*. IEEE, 2009.

[30] Mohsen, Heba. "Signature identification and verification systems: a comparative study on the online and offline techniques." *Future Computing and Informatics Journal* 5.1 (2020): 3.

[31] Kholmatov, Alisher, and Berrin Yanikoglu. "Identity authentication using improved online signature verification method." *Pattern recognition letters* 26.15 (2005): 2400-2408.

[32] Shanker, A. Piyush, and A. N. Rajagopalan. "Off-line signature verification using DTW." *Pattern recognition letters* 28.12 (2007): 1407-1414.

[33] Chen, Siyuan, and Sargur Srihari. "Use of exterior contours and shape features in off-line signature verification." *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*. IEEE, 2005.

[34] Güler, Inan, and Majid Meghdadi. "A different approach to off-line handwritten signature verification using the optimal dynamic time warping algorithm." *Digital Signal Processing* 18.6 (2008): 940-950.

[35] Stauffer, Michael, et al. "Offline signature verification using structural dynamic time warping." *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019.

[36] Kanwal, Saira, Muhammad Uzair, and Habib Ullah. "A Survey of Hand Crafted and Deep Learning Methods for Image Aesthetic Assessment." *arXiv preprint arXiv:2103.11616* (2021).

[37] Coetzer, Johannes. *Off-line signature verification*. Diss. Stellenbosch: University of Stellenbosch, 2005.

[38] Bertolini, Diego, et al. "Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers." *Pattern Recognition* 43.1 (2010): 387-396.

[39] Oliveira, Luiz S., et al. "The graphology applied to signature verification." *12th conference of the international graphonomics society*. 2005.

[40] https://www.stepover.com/us/solutions/

[41] J. F. Vargas, M. A. Ferrer, C. M. Travieso, and J. B. Alonso, "Off-line signature verification based on grey level information using texture features, Pattern Recogn., vol. 44, pp. 375–385, February 2011.

[42] A. Kholmatov and B. Yanıko˘glu, "Identity authentication using improved online signature verification method," Pattern Recognition Letters, vol. 26, no. 15, pp. 2400–2408, 2005.

[43] ESKANDER, G. S.; SABOURIN, R.; GRANGER, E. Hybrid writer-independent–writer-dependent offline signature verification system. IET biometrics, IET, v. 2, n. 4, p. 169–181, 2013. ISSN 2047-4938.

[44] HAFEMANN, L. G.; SABOURIN, R.; OLIVEIRA, L. S. Offline handwritten signature verification — literature review. In: 2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA). [S.l.: s.n.], 2017. p. 1–8. ISSN 2154-512X.

[46] SHAO, L.; ZHU, F.; LI, X. Transfer learning for visual categorization: A survey. IEEE Transactions on Neural Networks and Learning Systems, v. 26, n. 5, p. 1019–1034, May 2015. ISSN 2162-237X.

[47] Hafemann, Luiz Gustavo & Sabourin, Robert & Soares de Oliveira, Luiz. (2017). Offline Handwritten Signature Verification-Literature Review. 10.1109/IPTA.2017.8310112.

[48] Josh Patterson and Adam Gibson. Deep learning: a practitioner's approach. OReilly, 2017.

[49] Rosso, Osvaldo A., Raydonal Ospina, and Alejandro C. Frery. "Classification and verification of handwritten signatures with time causal information theory quantifiers." PloS one 11.12 (2016): e0166868.

[50] Liu, Chin-Fu et al. "Using deep Siamese neural networks for detection of brain asymmetries associated with Alzheimer's Disease and Mild Cognitive Impairment." Magnetic resonance imaging vol. 64 (2019): 190-199. doi:10.1016/j.mri.2019.07.003

[51] https://towardsdatascience.com/understanding-input-and-output-shapes-in-convolution-network-keras-f143923d56ca

[52] https://www.etsmtl.ca/unites-de-recherche/livia/recherche-et-innovation/projets/signature-verification

[53] Jogin, Manjunath, et al. "Feature extraction using convolution neural networks (CNN) and deep learning." *2018 3rd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT)*. IEEE, 2018.

[54] http://www.iapr-tc11.org/mediawiki/index.php/ICDAR_2011_Signature_Verification_Competition_(SigComp2011)