Face Spoofing Detection using Patch-based Deep Convolutional Neural Networks

Jonne Engelberts
MSc. Thesis
Radboud Universiteit Nijmegen

Supervisor: Tom Heskes
Second assessor: Elena Marchiori
External supervisor: Pouria Mortazavian
Special thanks to Onfido for enabling this research

May, 2018

1 Introduction

The face biometric is widely used for person identification and verification. It is present in most official documents and used in facial recognition systems at border control. Unfortunately facial recognition systems are vulnerable to face spoofing attacks [1], where the attacker presents a fake facial sample to the sensor. These attacks can be divided in 2-dimensional and 3-dimensional spoofing attacks [2]. In a 2-dimensional attack a picture or video of a legitimate user is presented to the sensor with the purpose of fooling the system. This picture attack is especially dangerous, as it is very easy to obtain a picture of any person online. In the 3-dimensional case the attacker can use a mask or make-up to fool the system, but this requires a lot more effort than obtaining a picture or video.

Face spoofing attacks are a serious problem for Onfido [3], a company specialised in identity verification and background checks. During their facial check a user takes a picture of themselves (a selfie) with their phone to verify their identity. It is crucial that this facial check can not be spoofed with a picture or video of someone else. Onfido is doing a lot of research into different methods of automatically confirming genuine selfies and only sending suspicious samples to manual review. Doing this reduces the cost of human labour and increases the speed of most facial checks, which are two important metrics for the company.

There has already been a lot of research in face spoofing detection using texture analysis methods, as detailed in the background section. The use of convolutional neural networks [4] in this area is still limited, despite its power in most image classification tasks, such as ImageNet [5] and CIFAR [6]. This thesis focused on using deep convolutional neural networks to detect 2-dimensional spoofing attempts. The third section of this thesis explains the collection and preprocessing of the data: genuine selfies and spoofing attempts. First a texture analysis method is applied to the data as a baseline. Then a deep convolutional neural network is applied to small patches extracted from the data. This patch-based approach increases the size of the dataset, allowing a deep network to be trained with limited data without overfitting. Finally the patch-based method is also applied to a public face spoofing detection dataset, with some minor modifications to suit the dataset.

2 Background

In this section a small overview of deep learning is given, including the latest advances and important regularization techniques. The problem of face spoofing attacks is explained in more detail and finally some texture analysis approaches and the few neural network approaches to face spoofing detection are reported.

2.1 Deep Learning

2.1.1 Neural Networks

A neural network is a collection of artificial neurons ordered in layers. The first neural networks [7][8] featured one or more fully-connected layers, where every neuron is connected to all neurons in the previous layer in a feed-forward fashion. Every connection is represented by a weight. To calculate the output of a neuron take the product between the value of a neuron in the previous layer and the weight of the connection, sum all incoming products and add a bias. If the weights between two layers are represented as two-dimensional matrix $\boldsymbol{W}$ the whole layer can be represented as $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, where $\boldsymbol{x}$ is the input vector, $\boldsymbol{b}$ the bias vector and $\boldsymbol{y}$ the output vector. However, multiple layers can be collapsed into one linear function. To allow the network to represent data that is not linearly separable a non-linearity is added to the output of every neuron. The most commonly used non-linearity is the REctified Linear Unit (ReLU) [9][10], which has the simple formula $f(x) = \max(x, 0)$ to clip all negative values to zero.

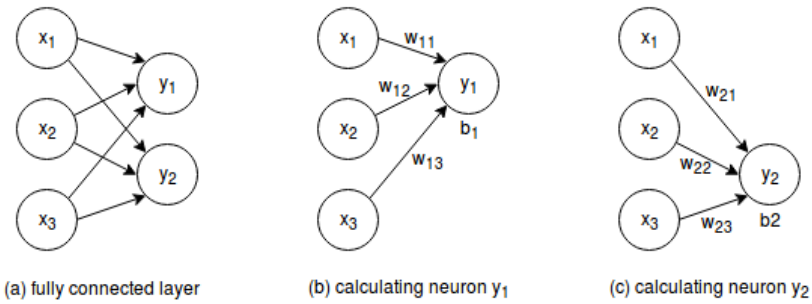

Figure 1: Fully connected layer with i=3 neurons in the first and j=2 neurons in the second layer. Every neuron in the second layer calculates its output based on the neurons in the first layer and the weights of their connections: $y_j = g(b_j + \sum_i w_{ij} * x_i)$, where g is a non-linearity such as ReLU.

2.1.2 Convolutional Neural Networks

Convolutional Neural Networks [11][4] start with convolutional layers, which are inspired by the hierarchical structure of small receptive fields in the visual cortex [12]. These layers use small kernels with weights, instead of connecting to all units in the previous layer. These kernels are convolved over the input (usually a two-dimensional image), by taking the products between the weights and corresponding inputs and summing the result (see figure 2). This way a convolutional layer uses less parameters and becomes translation invariant.

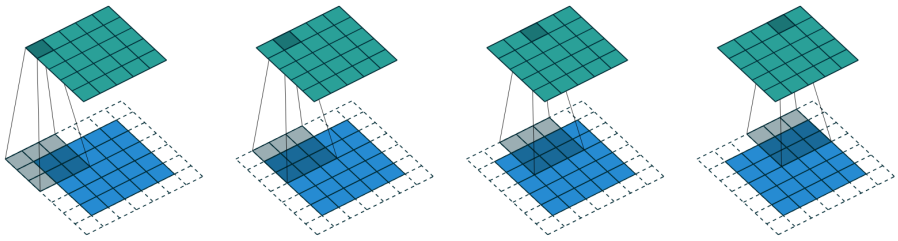

Figure 2: Convolutional layer with a 3 x 3 kernel convolving over 5 x 5 input with 1 padding (image from [13]).

2.1.3 Optimization

The last layer of the network should output a probability distribution, so instead of ReLU the Softmax function is applied: $\text{Softmax}(\boldsymbol{x}) = e^{\boldsymbol{x}} / \sum_{n=1}^{N} e^{x_n}$ where $x_{1..N}$ is the output of the last layer. This function has the property of normalizing the output of every individual neuron to a value between 0 and 1 and the total sum of all neurons to 1, like a proper probability distribution. To evaluate how well the network is performing the cross-entropy loss between the probability distribution and the preferred output of the network is calculated. If the preferred output is binary and only one output neuron is correct the Softmax and cross-entropy loss can be simplified to: $\text{loss}(\boldsymbol{x}) = \log\left(\sum_{n=1}^{N} e^{x_n}\right) - x_c$ where $x_{1..N}$ is the output of the last layer and x_c is the output of the correct neuron.

The weights and biases are randomly initialized before training. The network is trained with Stochastic Gradient Descent (SGD). This algorithm runs a random batch of training samples through the network. It uses the average cross-entropy loss of the outputs to calculate the gradients of every neuron, by differentiating all operations. This is called back-propagation, because we propagate the loss back through the network using the chain rule. Finally the gradients are multiplied with a small constant (the learning rate) to update the weights. Repeating the SGD algorithm many times forces the network to model the desired distribution.

2.1.4 State-of-the-art

The first network to win ImageNet was AlexNet [14] in the 2012 ImageNet competition. This network consisted of five convolutional layers followed by three fully connected layers. The network had sixty million parameters, but ninety-six percent of the parameters were from the fully connected layers. This network also used an important technique called Dropout [15], where during training half of the neurons are switched off. This forces the network to model redundancy and prevents overfitting to specific patterns. During test time the output of all neurons is used (but multiplied by 0.5) to take full advantage of the redundancy.

The main problem with training deeper networks is the vanishing gradient problem [16], where the gradients vanish during back-propagation. This causes the earlier layers to learn very slowly or not at all. This problem is solved by adding shortcuts to the network to create Residual Networks (or ResNets) [17]. The residual skip connections add the input of a layer to the output, so the layer only needs to learn a residual function: $x = \mathcal{F}(x) + x$. In practice it skips two layers and adds before the second non-linearity. These ResNets allow deeper networks (over a thousand layers) and faster training, likely due to a smoother loss surface [18].

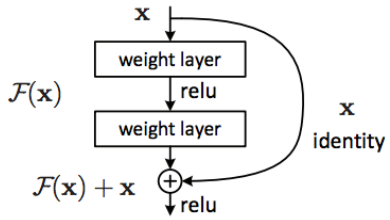

Figure 3: Residual skip connection skipping two layers (from [17]).

A combination of 152-layer ResNets won the 2015 ImageNet competition. This network also had sixty million parameters, but this time only three percent of the parameters were from the fully connected layers, the rest from convolutional layers. This network also used an important technique called Batch Normalization [19], where every batch of inputs to a neuron is normalized to zero mean and unit variance. This reduces the internal covariate shift, where small changes perturb through the network and shift the distributions in later layers. Batch Normalization allows easier learning due to the reduced covariate shift. Moreover it provides a strong regularization effect, because a sample is normalized based on batch statistics, which differ every time due to the Stochastic Gradient Descent.

2.2 Face Spoofing

2.2.1 Biometrics

Biometrics are unique human characteristics that can be used for identification and verification of people. The most used biometrics are physical (such as DNA, fingerprints and iris), but biometrics can even be behavioural (such as handwriting, voice and gait). These biometrics are widely used for different purposes. For example DNA and fingerprints are used to identify people in forensics, while the face biometric is used to verify people at border control. Today most smartphones allow access through fingerprints instead of the classic knowledge based systems, such as a pincode. Recently Apple replaced the fingerprint biometric with Face ID [20] to allow access through the face biometric. Using biometrics has certain advantages over other methods: you don't have to remember anything and you can't forget your biometrics at home. However most biometrics are not a secret, which makes the method vulnerable to attacks. Malicious users will try to fool the system by replicating the biometrics of a genuine user, called a spoofing attack.

2.2.2 Anti-spoofing methods

Anti-spoofing methods are either hardware-based or software-based [21]. Hardware-based anti-spoofing methods use sensor characteristics to detect spoofing attacks. For example measuring depth or temperature to detect human properties. These methods are very reliable, but often require extra hardware, which can be a burden for the user. Software-based anti-spoofing methods are applied after the sample has been collected. This is more user-friendly, but often less reliable than hardware-based methods. The software-based anti-spoofing methods can be further divided into static and dynamic methods [21]. Static methods require a single sample, while dynamic methods require multiple samples. When working with the face biometric the static methods require a picture, while the dynamic methods require a video. Again there is a trade-off between performance and user-friendliness, as a picture is less intrusive but a video contains more information.

2.2.3 Face spoofing attacks

There are different types of face spoofing attacks, but they can be divided in two categories: 2-dimensional and 3-dimensional spoofing attacks [2]. In a 2-dimensional attack a picture or video of a legitimate user is presented to the sensor. The picture can be printed or displayed on a digital screen. This method is very dangerous, because it is a trivial task to find a picture of anybody online. However both hardware-based methods and dynamic software-based methods are very effective against this type of attack, as a picture does not have any human properties and does not change over time. This can be countered by cutting the face out of the printed paper, creating a 2-dimensional mask and thus different depth and movement between the face and the background. Another option is wearing the 2-dimensional mask over a genuine face with the eyes or mouth cut-out. This adds distinct facial properties, such as blinking, but decreases the overall quality of the attack. Displaying a video on a digital screen is effective against dynamic software-based methods as well, because it can display realistic movement of the face. In the 3-dimensional spoofing attack a mask, make-up or plastical surgery is used to mimic a legitimate user. These methods aim to be effective to hardware-based anti-spoofing methods as well. 3-dimensional masks can add realistic depth and movement to the attack, but a realistic mask is still very expensive and hard to create. Other attacks such as make-up and plastical surgery are even harder to achieve.

2.3 Face spoofing detection

Most face spoofing detection methods use texture analysis to predict whether a sample is a spoofing attempt or not. The usual approach is detecting the face, applying a local texture descriptor, such as local binary patterns [22] or local phase quantization [23], aggregating the response and feeding it into a classifier, such as a support vector machine [24].

2.3.1 Texture analysis

The paper "Face Spoofing Detection From Single Images Using Micro-Texture Analysis" [25] applied the local binary patterns (LBP) algorithm to the problem of face spoofing. This algorithm processes every pixel by comparing it with close neighbouring pixels and encoding this information. The

paper showed the power of the computationally fast and intensity invariant LBP algorithm in the field of face spoofing detection. Figure 4a shows how to calculate the pattern of a single pixel, given its neighbouring pixels. Every neighbour is thresholded against the center pixel resulting in eight binary values. These binary values form the pattern for the center pixel. With eight neighbouring pixels there are 2^8 = 256 unique patterns. This process is repeated for every pixel in the image, except those at the edges. Finally the occurrence of every pattern is counted, resulting in a histogram of patterns. The normalized frequencies of the histogram are then used as input to a classifier. Thresholding the intensities of the neighbouring pixels against the intensity of the center pixel makes this algorithm robust against global changes in intensity, such as illumination changes. Furthermore the simplicity of the algorithm makes it relatively fast.

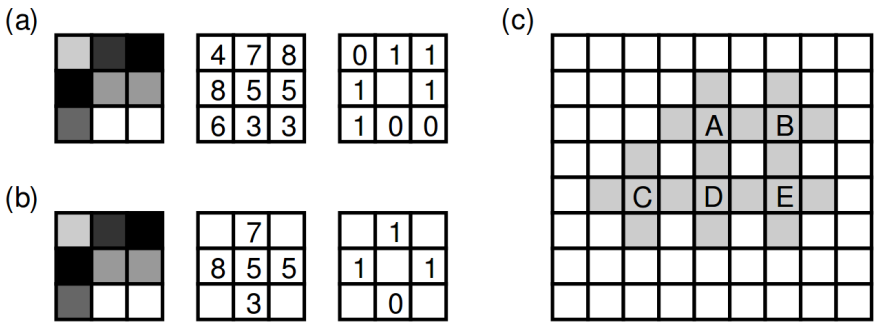

Figure 4: (a) Regular local binary pattern. (b) Reduced local binary pattern used in CoALBP. (c) CoALBP counts the co-occurrences between the pattern at A and the patterns at B, C, D and E for every pixel A.

The Co-occurrence of Adjacent Local Binary Patterns (CoALBP) algorithm [26] is an extension of the original LBP algorithm. Instead of counting how often a pattern occurs the CoALBP algorithm counts how often two patterns occur together in close proximity. These co-occurrences are able to represent more complex patterns than the original patterns. To reduce the amount of possible combinations the CoALBP algorithm only considers the four direct neighbours (see figure 4b), resulting in 16 distinct patterns and 256 possible co-occurrences between two patterns. Figure 4c shows how to count the co-occurrences after calculating the pattern for every pixel. For every pattern at A count the co-occurrence with the patterns at location B, C, D and E in a separate histogram. Finally concatenate the four histograms into a single one with length 4*256 = 1024.

The paper "Face Spoofing Detection Using Colour Texture Analysis" [27] tested five different texture descriptors on multiple face spoofing datasets. The CoALBP algorithm performed best on two out of three datasets. To improve performance these descriptors were applied to every colour band separately instead of only to the greyscale. They also considered different colour spaces besides RGB, such as HSV and YCbCr, because the colour channels of RGB are very correlated. Their results show the YCbCr colour channel performed best on two out of three datasets, but a combination of both the HSV and YCbCr channels resulted in the best performance.

2.4 Convolutional Neural Networks

So far there has not been much research into using convolutional neural networks for face spoofing detection. The first paper to use this approach is "Learn Convolutional Neural Network for Face Anti-Spoofing" [28], where they train an eight-layered convolutional neural network to classify samples as genuine or spoof. After training they use the features from the last layer as input to a support vector machine. A more competitive paper using convolutional neural networks is the "Face Anti-Spoofing Using Patch and Depth-Based CNNs" [29] paper, where two methods are combined. The first method trains an eight-layered convolutional neural network on patches extracted from the face area to classify samples as genuine or spoof. The second method tries to predict the depth map of a sample using an estimated depth map as target for genuine samples and a flat depth map for spoofed samples. The resulting depth map is used for feature extraction and classification using a support vector machine. The power of the patch and depth-based approach

is that there is no need for resizing, due to the same size of the patches and fully convolutional structure in the depth-based network.

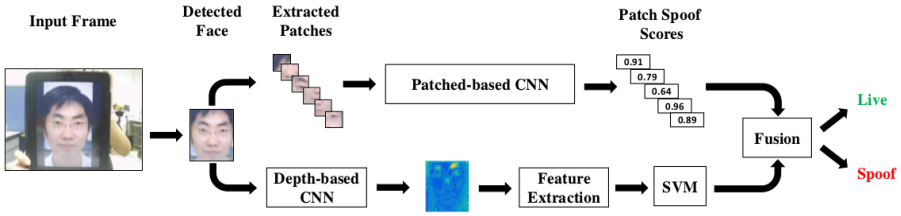

Figure 5: Architecture of the Patch and Depth-Based CNN (image from [29]).

3 Data

Onfido is currently processing hundreds of thousands of facial checks per month. To ensure fair evaluation and minimize overlap data from 2017 was used as training and validation data, while data from 2018 was used as testing data. The biggest limiting factor was the storage space, so only thirty thousand genuine selfies were used for training and validation. Inspection of the data identified three distinct types of 2-dimensional spoofing attempts: taking a picture of a face on a picture or print-out (picture attack), taking a picture of a face on a screen, such as a computer screen or phone screen (screen attack) and taking a picture of the headshot on a document (document attack). Unfortunately there is some overlap between these classes, for example screen attacks displaying a document headshot and pictures that are glued to a document instead of plasticized. Figure 6 shows an original picture being used as a picture and screen attack, furthermore an example of a document attack is included. It shows how hard to detect the picture attack and screen attack can be, especially when viewed in isolation and restricted to the area of the face. The document attack is more obvious as the headshots on a document are usually rather small and contain watermarks.

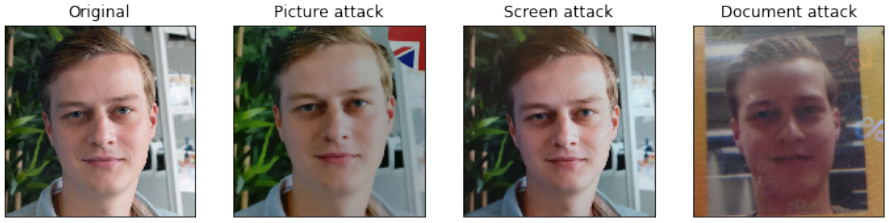

Figure 6: Examples of spoofing attempts (not actual client data).

3.1 Training and validation data

As the vast majority of facial checks contain a genuine selfie there is a lot of positive data. The problem is the limited supply of spoofing attempts: a rough estimate of spoofing attempts per facial check would be one percent. It was important to find as many spoofing attempts as possible to counter this class imbalance. In the first round of manual labeling around five thousand potential spoofing attempts were identified. This data was used to quickly train a preliminary model, which was then used to filter more potential spoofing attempts from the rest of the data. The potential spoofing attempts were sent to external labelers to filter out most genuine selfies. Finally I went through all spoofing attempts myself, sorting them in picture attack, screen attack, document attack and filtering out everything else.

	Genuine selfies	Picture attacks	Screen attacks	Document attacks	Total samples
Available data	29,259	1,044	1,051	1,200	32,554
Training data	28,809	894	901	1,050	31,654
Validation data	450	150	150	150	900

Table 1: Number of training and validation samples per class.

In total there were 32,554 samples available for training at the time, of which about ten percent were spoofing attempts. These samples were split into training and validation data. To keep the validation data balanced 450 genuine selfies, 150 picture attacks, 150 screen attacks and 150 document attacks were separated from the training data. It is important to have a separate validation set to track performance and generalization to unseen samples. The validation set is used to select the best hyperparameters before training, while it is also used after training to select the best model and set a good threshold.

3.2 Testing data

The testing data is used to test the performance of the final model given the threshold set on the validation data. As the validation data is used multiple times before and after training to change the model, it is no longer independent. That is why it is important to have separate testing data and limit its usage. All available spoofing attempts were collected and supplemented with genuine samples to a total of 5000. Again the spoofing attempts were divided in three categories by myself, to the best of my abilities. Unlike the training and validation data the spoofing attempt categories were quite unbalanced, with mostly screen attacks.

	Genuine selfies	Picture attacks	Screen attacks	Document attacks	Total samples
Testing data	3,709	271	755	265	5000

Table 2: Number of testing samples per class.

3.3 Facial crops

In every sample the face was detected and cropped, using the dlib [30] face detector [31]. This ensured the method focused on texture details instead of peripheral attributes (such as the edges of a picture or pictograms on a computer screen) which might not be present in more sophisticated spoofing attempts. Moreover it prevents the model getting confused by background materials, such as posters or screens. These detected and cropped faces will be referred to as 'facial crops' from this point on.

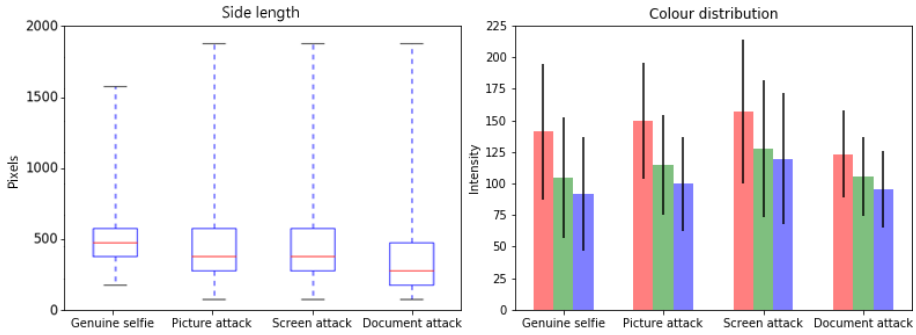

Figure 7: The size and colour distribution of the facial crops from the training and validation data. The boxplots represent the length of the side of a facial crop, which was always square. The bars in the histogram represent the average intensity of that colour within that category, while the horizontal line represents the standard deviation.

A quick analysis of the training and validation data is shown in figure 7. The side length indicates the height and width of the facial crop, which was always square. Most facial crops were between 128 and 512 pixels. Genuine selfies were a bit larger on average and some document attacks were especially small, but overall the distributions were similar. The colour distribution histograms don't show any notable differences, except perhaps a smaller standard deviation in document attacks.

3.4 Datasets

The different sizes of the facial crops are not a problem for both the texture analysis and the patch-based deep learning approach. The histogram from the texture analysis method is normalized, so it only represents the relative frequency of a pattern or co-occurrence. Most convolutional neural networks do require a fixed-sized input. However the patch-based approach alleviates this problem, because it selects a fixed-sized patch as input, no matter the size of the facial crop. Using the original resolution might increase performance as resizing can decrease important texture artifacts, such as moire patterns which are common in screen attacks. However the range of the sizes of the face crops is sufficiently large to cause concern, as some are almost a hundred times larger than others.

To test the assumption that it is best to use the original resolution five different datasets were created from the training and validation data. The first dataset consisted of the facial crops in their original resolution, while the other datasets contained the same facial crops, but resized to fixed resolutions:

- **original size:** all facial crops in their original resolution
- **512 by 512:** all facial crops resized to 512 by 512
- **256 by 256:** all facial crops resized to 256 by 256
- **128 by 128:** all facial crops resized to 128 by 128
- **64 by 64:** all facial crops resized to 64 by 64

4 Methods

The goal of this thesis was to create a method that automatically accepts as many genuine selfies as possible, without accepting too many spoofing attempts. To quantify this the method was not allowed to accept more than one percent of the spoofing attempts. The problem was treated as a simple classification task with two classes: genuine selfie and spoofing attempt. The distinction between different spoofing attempts (picture, screen and document attack) was ignored, as subdividing the spoofing attempts would only decrease the amount of samples per class. Both the texture analysis method and the patch-based deep learning method were trained using the training data. For the patch-based deep learning method the evaluation data was used to select sensible hyperparameters and monitor training. Both methods were applied to the 5 datasets with different resolutions.

4.1 Texture Analysis

A texture analysis method was applied to the same data to be compared with the patch-based deep learning method. In order to create a decent baseline, while keeping the complexity down, the best method and colour space from the colour texture analysis paper [27] were used: the CoALBP method [26] performed on the HSV and YCbCr colour spaces. The algorithm was applied to every colour channel separately, resulting in 1024 features per colour space. The HSV and YCbCr colour spaces consist of three channels each, which amounts to 6144 features per sample. For every dataset a classifier was trained on the features of the training data and evaluated on the features of the validation data. The Support Vector Classification class [32] from scikit-learn [33] was used as classifier, using the the radial basis function kernel and probability estimation.

4.2 Patch-based deep learning

To train a deep network with limited data the patch-based approach [29] was used. In this approach a network is trained on small patches randomly extracted from the images. This significantly increased the amount of training samples, despite the limited number of facial crops. Moreover this allowed the method to use the original resolution, despite the different sizes of the facial crops. To evaluate an image the predictions of multiple patches were averaged.

4.2.1 Architecture

The network used was a ResNet-18 [17] created for CIFAR-10, which requires a 32 by 32 pixel input. To account for the lack of mean subtraction due to the patch-based approach a Batch Normalization layer was added to the start of the network [34]. This automatically performed the mean subtraction and variance division, while also augmenting the input. The network featured twenty-one convolutional layers and about three hundred thousand trainable parameters.

4.2.2 Patches

The patches were randomly extracted from samples selected by a weighted random sampler, ensuring an equal ratio of patches from genuine selfies and spoofing attempts. The size of the patches was 32 by 32 pixels, as required by the network. The only data augmentation applied was random horizontal flipping. Most other data augmentation techniques distort or resample the image, which is something we wanted to avoid.

4.2.3 Optimization

The initial learning rate and batch size were selected using manual search. This entailed trying a number of sensible combinations and aborting the ones where learning was unstable or too slow. The final selected initial learning rate was 1e-3 and the learning rate was reduced to 1e-4 after 75 epochs. The final selected batch size was 64 patches per batch. The other hyperparameters were adopted from the original ResNet paper [17], where the momentum was set to 0.9 and the weight decay was set to 1e-4. The network was trained using stochastic gradient descent minimizing the cross-entropy loss. The network was trained for 100 epochs, where each epoch contained 2^20 patches from the training set. After every epoch 2^15 patches from the validation set were used to validate the training.

4.2.4 Training

During training the cross-entropy loss of the training data was recorded. After every epoch the cross-entropy loss and the error were also calculated for the validation data. Figure 5 shows these metrics per epoch for the different datasets. The network trained on the "original size" dataset performed less than expected, with the worst training loss and validation accuracy. The network trained on the "64 by 64" dataset had the best training loss, but the validation loss shows it was likely overfitting. Surprisingly the validation error is only second to best, although less stable during training. The networks trained on the rest of the datasets show a remarkable trend: networks trained on datasets with smaller resolutions performed better on both the training data and the validation data.

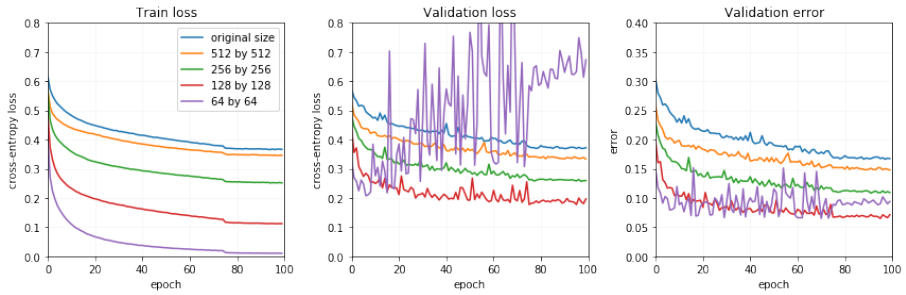

Figure 8: Cross-entropy loss of the training data and the validation data and the error at a threshold of 0.5 of the validation data for the network trained on the five different datasets.

4.2.5 Evaluation

The networks were trained with single patches as input to predict whether the patch is from a genuine sample or spoofing attempt. To evaluate a full facial crop multiple patches were uniformly sampled from the image. The predictions of the patches were averaged before applying the softmax function. For every image a uniform sampling of 1024 patches (32 horizontal and 32 vertical) was

used in this project. For every dataset the facial crops from the validation data were used to evaluate the network trained on patches from the corresponding training data.

5 Results

The texture analysis method and patch-based deep learning method were compared based on their performance on the validation data. The effect of resizing the facial crops was also evaluated on the validation data using the five datasets. The best method and dataset combination was selected and the validation data was then used to set a good threshold, where less than one percent of the spoofing attempts were classified as genuine selfies. Finally the testing data was used to evaluate the performance of the selected method, dataset and threshold.

5.1 Validation data

To compare the performance the Receiver Operating Characteristic (ROC) curves are plotted. These curves show the relation between the True Positive Rate and the False Positive Rate, which can be interpreted as the ratio of genuine selfies correctly accepted and the ratio of spoofing attempts falsely accepted, respectively. The ROC curve is a great way to measure performance on the validation set, as the best threshold is still unknown and the ROC curve is a threshold independent metric. To quantify the ROC curve two metrics were extracted from it. The first metric is the Area Under the ROC curve (AUROC), which measures the area under the ROC curve. The larger the area is, the better the performance. The second metric is the True Positive Rate at a 0.01 False Positive Rate, which corresponds to the rate with which the model can automatically pass genuine samples while only passing one percent of the spoofing attempts as well.

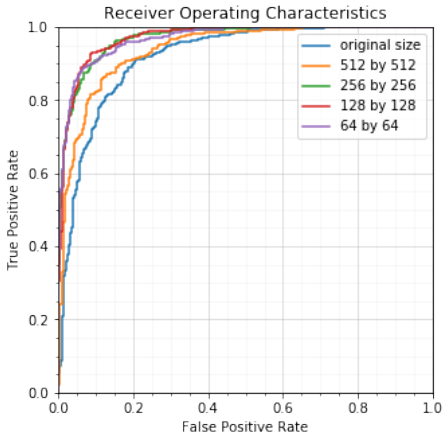

Figure 9: Receiver Operating Characteristics for the texture analysis method trained and evaluated on the five different datasets.

		original size	512 by 512	256 by 256	128 by 128	64 by 64
Area Under the ROC curve		92.43 %	93.64 %	96.81 %	97.12 %	96.34 %
FPR = 0.0088	TPR	27.20 %	37.20 %	58.40 %	60.80 %	57.20 %

Table 3: Area Under the Receiver Operating Characteristic curve and True Positive Rate at a False Positive Rate close to 1 percent for the texture analysis method trained and evaluated on the five different datasets.

Figure 9 shows the ROC curves from the texture analysis method, while table 3 shows the metrics extracted from these ROC curves. The method trained and evaluated on the "original size" dataset performed worst according to both metrics, while the method performed best on the datasets resized to the three smallest resolutions. This contradicts our assumption it is best to use the original resolution and avoid resizing the image. The method trained and evaluated on the "128 by 128"

dataset performed best, with an AUROC of 97.12 %. When allowing less than one percent of the spoofing attempts to be falsely accepted it can automatically pass 60.80 % of the genuine selfies.

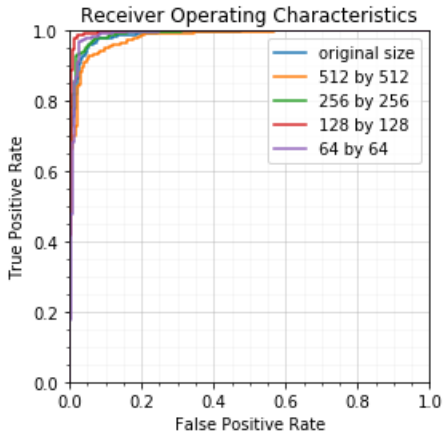

Figure 10: Receiver Operating Characteristics for the patch-based deep learning method trained and evaluated on the five different datasets.

		original size	512 by 512	256 by 256	128 by 128	64 by 64
Area Under the ROC curve		98.89 %	98.14 %	99.21 %	99.68 %	98.93 %
FPR = 0.0088	TPR	70.44 %	62.00 %	81.78 %	92.44 %	69.78 %

Table 4: Area Under the Receiver Operating Characteristic curve and True Positive Rate at a False Positive Rate close to 1 percent for the patch-based deep learning method trained and evaluated on the five different datasets.

Figure 10 shows the ROC curves from the patch-based deep learning method, while table 4 shows the metrics extracted from these ROC curves. It is clear that this method outperformed the texture analysis method, as it performed better on both metrics regardless of the dataset. The method performed best when applied to the "128 by 128" dataset as well, with an AUROC of 99.68 %. When allowing less than one percent of the spoofing attempts to be falsely accepted it can automatically pass 92.44 % of the genuine selfies.

5.2 Testing data

Based on the results on the validation data the patch-based deep learning method trained and evaluated on the "128 by 128" dataset was selected. The threshold was set to accept less than one percent of the spoofing attempts in the validation data. The testing data was resized to 128 by 128 pixels to match the dataset. Table 5 shows the performance on the testing data. Because the testing data is supposed to represent the real distribution (except for the ratio between genuine selfies and spoofing attempts) it also includes samples where the face detection failed. Whenever a sample scored below the threshold or a face detection failed it was classified as "Warning", only when the method was sure the sample is genuine it was classified as "Passed". When applied to the testing data the method was able to automatically accept 90.51 % of genuine selfies, while only falsely passing 1.63 % of the spoofing attempts. Most of the falsely passed spoofing attempts were screen attacks, while the screen attacks were overrepresented in the testing set, which can explain the slightly higher than expected false positive rate.

	Passed	Warning
Genuine selfies	3357	352 (76 failed)
Spoofing attempts	21	1270 (75 failed)

Table 5: Performance of the patch-based deep learning model using patches from facial crops resized to 128 by 128 pixels on the testing data.

6 Conclusion

6.1 Resolution

The results for both methods show a similar trend, where resizing all facial crops to a smaller resolution performed better than resizing to a larger resolution such as 512 by 512 pixels or not resizing at all. Resizing all facial crops to 128 by 128 pixels performed best for both methods. However, in the case of the texture analysis method resizing to 64 by 64 pixels or 256 by 256 pixels performed similarly, the difference might just be due to chance. In the case of the patch-based deep learning method resizing to 64 by 64 pixels suffered from overfitting, it might have performed better with more data. Nonetheless, it seems resizing all facial crops to 128 by 128 pixels is a good practice. Our initial assumption it is best to use the original resolution and avoid resizing the image appears unjustified.

6.2 Method

The results also show the patch-based deep learning method clearly outperformed the texture analysis method, regardless of the size of the facial crops. Deep neural networks usually require large amounts of data, the ResNet-18 was trained with over one million images in the original paper [17]. Using the patch-based approach we were able to train the same network with less than 32,000 images, of which only 2,845 were spoofing attempts. When allowing less than one percent of the spoofing attempts to be falsely accepted the patch-based deep learning method was able to achieve a 92.44 % automation rate, versus the 60.80 % automation rate of the texture analysis method, which is an increase of over fifty percent.

6.3 Automation

The patch-based network in combination with resizing all facial crops to "128 by 128" and selecting a threshold on the validation data performed very well on the testing data as well. The testing data was selected from a different time period to prevent overlap with the training data. It also included all and only spoofing attempts from that time period to keep the same distribution between different spoofing attempts. The results show it has learned very well to automatically accept genuine selfies, without accepting too many spoofing attempts. Figure 11 shows the heatmaps of this model on the facial crops extracted from the examples in figure 6. A heatmap is the result of the network on 1024 uniformly sampled patches, projected back to the location of the patches. The yellow pixels indicate the network predicted the patch at that location was extracted from a spoofing attempts. The figure shows a clear difference between the original image and the spoofing attempts.

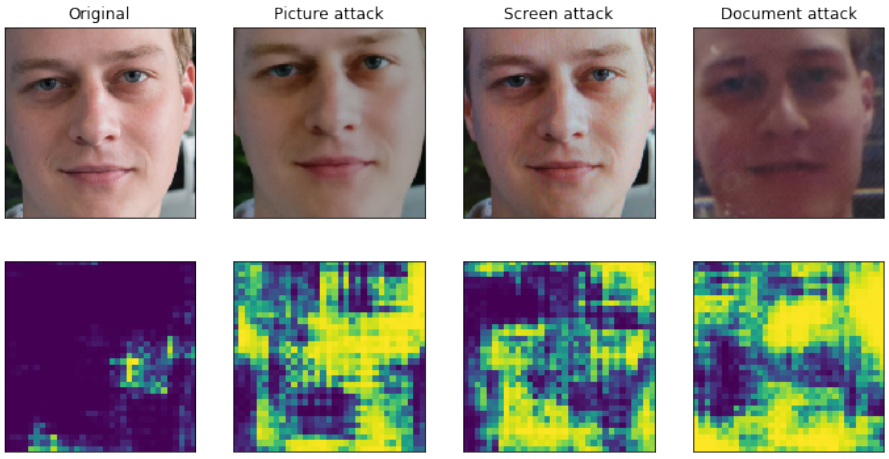

Figure 11: Heatmaps of the patch-based deep learning model trained on the "128 by 128" dataset, evaluated on four facial crops resized to 128 by 128 pixels. A yellow pixel predicts a patch from a spoofing attempts, while a blue pixel predicts a patch from a genuine selfie.

7 External dataset

The patch-based deep learning method was also applied to a public face spoofing dataset, to compare it with other state-of-the-art methods. Some minor modifications were necessary to achieve good performance on this dataset, because of its evaluation protocols.

7.1 OULU-NPU dataset

The OULU-NPU dataset [35] is a face spoofing detection dataset designed to simulate real-world variations in attack devices, recording mobile phones and environments. There are two types of spoofing attempts: picture attack and device attack (which is the same as a screen attack). The dataset is designed with four different evaluation protocols in mind. In every protocol the data is divided based on one or all of the variations. For example in protocol 1 a method is trained using videos recorded in environments 1 and 2, but tested using videos recorded in environment 3. In protocol 2 a method is trained using spoofing attempts created by printer 1 and display device 1, but tested using spoofing attempts created by printer 2 and display device 2. In protocol 3 a method is trained using five out of six recording mobile phones, but tested using the other in a leave-one-out fashion. Protocol 4 combines all three protocols, so all videos used for testing have different attack devices, recording mobile phone and environment than the videos used for training. Regardless of the protocol a method is always trained and tested using different people. Figure 12 shows the effect of the three variations on a single person. Table 6 shows the different protocols in more detail.

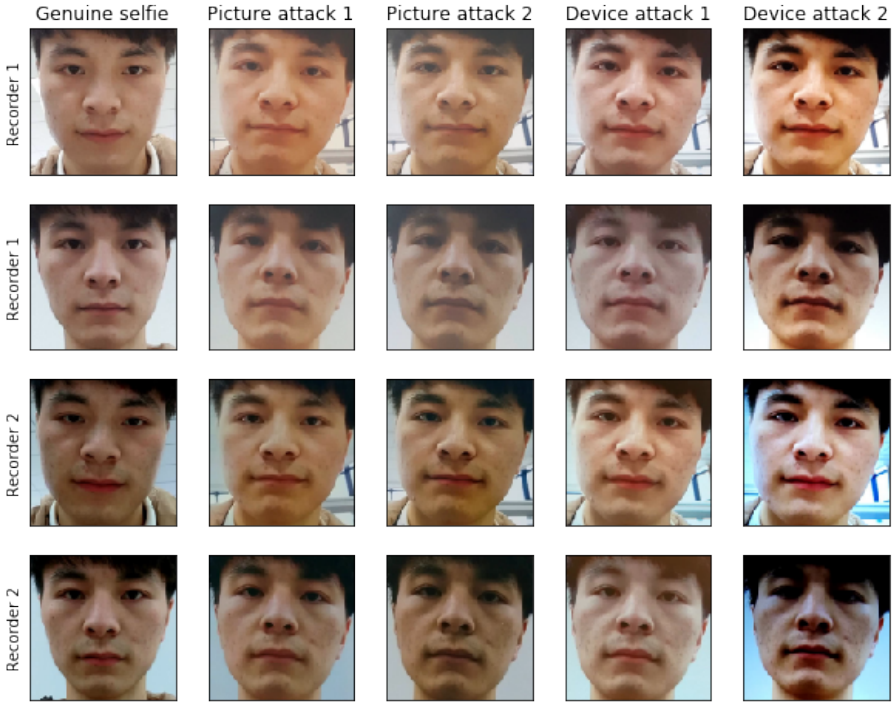

Figure 12: Variations in attack devices, recording mobile phone and environments for a single user in the OULU-NPU dataset. Row 1 and 3 are from environment 1, while row 2 and 4 are from environment 2.

Protocol	Subset	Session	Phones	Users	Attacks created using	# real videos	# attack videos	# all videos
Protocol I	Train	Session 1,2	6 Phones	1-20	Printer 1,2; Display 1,2	240	960	1200
	Dev	Session 1,2	6 Phones	21-35	Printer 1,2; Display 1,2	180	720	900
	Test	Session 3	6 Phones	36-55	Printer 1,2; Display 1,2	120	480	600
Protocol II	Train	Session 1,2,3	6 Phones	1-20	Printer 1; Display 1	360	720	1080
	Dev	Session 1,2,3	6 Phones	21-35	Printer 1; Display 1	270	540	810
	Test	Session 1,2,3	6 Phones	36-55	Printer 2; Display 2	360	720	1080
Protocol III	Train	Session 1,2,3	5 Phones	1-20	Printer 1,2; Display 1,2	300	1200	1500
	Dev	Session 1,2,3	5 Phones	21-35	Printer 1,2; Display 1,2	225	900	1125
	Test	Session 1,2,3	1 Phone	36-55	Printer 1,2; Display 1,2	60	240	300
Protocol VI	Train	Session 1,2	5 Phones	1-20	Printer 1; Display 1	200	400	600
	Dev	Session 1,2	5 Phones	21-35	Printer 1; Display 1	150	300	450
	Test	Session 3	1 Phone	36-55	Printer 2; Display 2	20	40	60

Table 6: Detailed information about the protocols from the OULU-NPU dataset (table from [35]).

7.2 Method

The patch-based deep learning method was applied to the OULU-NPU dataset. Due to storage and time limitations only the first frame with a detected face was used per video. The faces were detected, cropped and resized to 128 by 128 pixels. The patch-based deep learning method required one important modification: every patch was standardized to zero mean and unit variance. Without this, the method would perform very well on the training and validation data, but very poorly on the testing data. It seems standardizing every patch separately helped overcome the difference in distribution between the training and validation data on the one hand and the testing data on the other, caused by the protocols. Another small modification was the amount of samples per epoch: instead of 2^20 patches per epoch only 2^18 patches per epoch were used. The rest of the hyperparameters remained the same (see section 4.2).

7.3 Evaluation

The validation data (Dev in table 6) was used to set the best threshold at the Equal Error Rate (EER). The EER is the error rate (or proportion of misclassifications) at the threshold where the error rate within the genuine selfies and the error rate within the spoofing attempts is equal. The Average Classification Error Rate (ACER) was calculated on the testing data (Test in table 6) given the threshold. The calculation is a bit complicated. First the Bona fide Presentation Classification Error Rate (BPCER) is calculated, which is simply the error rate within the genuine selfies. Second the Attack Presentation Classification Error Rate (APCER), which is the error rate within either the picture attacks or the device attacks, whichever is larger. Finally the ACER is simply the average between the BPCER and the APCER.

7.4 Results

Table 7 reports both the EER and the ACER for a number of methods. The CNN-RNN method is the method from the paper "Learning Deep Models for Face Anti-Spoofing: Binary or Auxiliary Supervision" [36]. To the best of our knowledge it has the best performance on the OULU-NPU dataset, however it was released near the end of this thesis. The GRADIANT method performed the best on protocol 2, 3 and 4 in the IJCB 2017 competition. Unfortunately it is not clear what kind of method was used. Patch-based DL refers to the patch-based deep learning method. In contrast to the CNN-RNN and GRADIANT methods the patch-based deep learning method only used one frame per video.

Protocol	Method	EER (%)	ACER (%)
1	CNN-RNN	-	1.6
	GRADIANT	1.1	6.9
	Patch-based DL	0.6 ± 0.2	4.5 ± 0.7
2	CNN-RNN	-	2.7
	GRADIANT	0.9	2.5
	Patch-based DL	0.6 ± 0.2	5.0 ± 1.0
3	CNN-RNN	-	2.9 ± 1.5
	GRADIANT	0.9 ± 0.4	3.8 ± 2.4
	Patch-based DL	0.7 ± 0.2	8.5 ± 6.5
4	CNN-RNN	-	9.5 ± 6.0
	GRADIANT	1.1 ± 0.3	10.0 ± 5.0
	Patch-based DL	0.4 ± 0.3	11.7 ± 7.3

Table 7: The EER and ACER reported for the patch-based deep learning method and two other methods. The full results of the patch-based deep learning method can be found in the Appendix.

The results in table 7 show the performance of the patch-based deep learning method is quite competitive with the CNN-RNN and the GRADIANT methods, even though it only used a single frame per video for training and evaluation. In protocol 1 it even outperformed the GRADIANT method with a lower ACER, however the CNN-RNN performed best in most protocols with the lowest ACER. Note that the patch-based method outperformed the GRADIANT method on the validation set in all protocols with the lowest EER. It seems it had more trouble with the different variations in the testing data than the other methods. In protocol 3 and 4 the methods were trained and tested six times, so the averages and standard deviations were reported. However the patch-based deep learning method is stochastic, so it was trained and tested on the same data six times even in protocol 1 and 2. The results show the standard deviation is significantly larger in protocol 3 and 4 than in protocol 1 and 2. This indicates the effect of using a different recording mobile phone as testing data is larger than just expected due to the stochasticity. The patch-based deep learning method only used 600 samples as training data in protocol 4, but still performed well, due to the patch-based approach.

7.5 Conclusion

It was difficult to compare the patch-based deep learning method on a public face spoofing dataset, as most datasets use videos instead of still pictures. However even when only using a single frame from the video the patch-based method was still quite competitive. The difference between the EER and ACER does show the importance of using training data with the same variations and distribution as the real data.

8 Discussion

8.1 Improvements

The patch-based approach performed well on the internal data. However there are still some improvements possible to the current model. The threshold for the patch-based method was set using the validation data to accept less than one percent of the spoofing attempts as genuine selfies. However on the testing data 1.63 % of the spoofing attempts were falsely accepted. This is likely caused by the different distributions of spoofing attempts between the validation data and the test data. The validation data contained a similar amount of picture, screen and document attacks, while the testing data contained mostly screen attacks. It would be best to set a new threshold based on the testing data, as it has more samples and the same distribution of spoofing attempts as the actual data. Furthermore the dlib face detector was used to detect faces because of its ease of use, but there are better options. Two percent of the genuines sample in the testing data were not evaluated because the dlib face detector failed. An easy way to increase the automation rate would be to replace the face detector with a better one.

8.2 Future work

The patch-based method is a good solution to the problem of face spoofing. It clearly outperformed the texture analysis method on the internal dataset. It was also very competitive with the state-of-the-art on the OULU-NPU dataset, while only using single frames instead of the full video is a big disadvantage. However it is very reliant on the types of spoofing attempts in the training data. In this thesis only picture, screen and document attacks were used to train the models. In order to account for novel methods of spoofing, such as 3-dimensional spoofing attacks, the method should be retrained once new spoofing attacks start to emerge. Another option not addressed in this thesis is using a one-class classifier or anomaly detection, where a method is trained with genuine selfies only. Over time more data will become available at Onfido. This opens up more possibilities for future work. Training a separate network for every type of spoofing makes it easier to adapt to new types of spoofing without retraining everything. Separate networks or a multi-class network also give more informative results and might even perform better. Once enough data is available the patch-based approach can be dropped in favor of the traditional holistic approach, where the full facial crop is used as input to a network. This allows the network to combine information from multiple locations in the face, which will return better results. As shown in this thesis resizing the facial crops is not detrimental to the performance, so with enough data there is no need for the patch-based approach.

References

[1] Yan Li, Ke Xu, Qiang Yan, Yingjiu Li, and Robert H Deng. Understanding osn-based facial disclosure against face authentication systems. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 413–424. ACM, 2014.

[2] Javier Galbally, Sébastien Marcel, and Julian Fierrez. Biometric antispoofing methods: A survey in face recognition. *IEEE Access*, 2:1530–1552, 2014.

[3] Onfido | identity verification. https://www.onfido.com/.

[4] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[6] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

[7] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[8] DO Hebb. The organization of behavior. 1949. *New York Wiely*, 2002.

[9] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

[10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[11] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[12] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[13] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[16] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[20] Apple. Face id security, 2017.

[21] Ctirad Sousedik and Christoph Busch. Presentation attack detection methods for fingerprint recognition systems: a survey. *Iet Biometrics*, 3(4):219–233, 2014.

[22] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041, 2006.

[23] Ville Ojansivu and Janne Heikkilä. Blur insensitive texture classification using local phase quantization. In *International conference on image and signal processing*, pages 236–243. Springer, 2008.

[24] Vladimir Vapnik. *Statistical learning theory. 1998*. Wiley, New York, 1998.

[25] Jukka Määttä, Abdenour Hadid, and Matti Pietikäinen. Face spoofing detection from single images using micro-texture analysis. In *Biometrics (IJCB), 2011 international joint conference on*, pages 1–7. IEEE, 2011.

[26] Ryusuke Nosaka, Yasuhiro Ohkawa, and Kazuhiro Fukui. Feature extraction based on co-occurrence of adjacent local binary patterns. In *Pacific-Rim Symposium on Image and Video Technology*, pages 82–91. Springer, 2011.

[27] Zinelabidine Boulkenafet, Jukka Komulainen, and Abdenour Hadid. Face spoofing detection using colour texture analysis. *IEEE Transactions on Information Forensics and Security*, 11(8):1818–1830, 2016.

[28] Jianwei Yang, Zhen Lei, and Stan Z Li. Learn convolutional neural network for face anti-spoofing. *arXiv preprint arXiv:1408.5601*, 2014.

[29] Yousef Atoum, Yaojie Liu, Amin Jourabloo, and Xiaoming Liu. Face anti-spoofing using patch and depth-based cnns. In *Biometrics (IJCB), 2017 IEEE International Joint Conference on*, pages 319–328. IEEE, 2017.

[30] dlib c++ library. http://dlib.net/.

[31] Davis E King. Max-margin object detection. *arXiv preprint arXiv:1502.00046*, 2015.

[32] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[33] scikit-learn: machine learning in python. http://www.scikit-learn.org/.

[34] Marcel Simon, Erik Rodner, and Joachim Denzler. Imagenet pre-trained models with batch normalization. *arXiv preprint arXiv:1612.01452*, 2016.

[35] Zinelabinde Boulkenafet, Jukka Komulainen, Lei Li, Xiaoyi Feng, and Abdenour Hadid. Oulu-npu: A mobile face presentation attack database with real-world variations. In *Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on*, pages 612–618. IEEE, 2017.

[36] Yaojie Liu, Amin Jourabloo, and Xiaoming Liu. Learning deep models for face anti-spoofing: Binary or auxiliary supervision. *arXiv preprint arXiv:1803.11097*, 2018.

9 Appendix

Protocol	Dev	Test				
		Video	Print	Overall		
	EER (%)	APCER (%)	APCER (%)	APCER (%)	BPCER (%)	ACER (%)
1	0.65 ± 0.18	0.62 ± 0.32	5.49 ± 1.06	5.49 ± 1.06	3.61 ± 2.13	4.55 ± 0.69
2	0.59 ± 0.21	6.20 ± 2.83	4.72 ± 1.70	6.20 ± 2.83	3.75 ± 1.15	5.00 ± 1.04
3	0.71 ± 0.20	2.08 ± 2.14	6.67 ± 8.59	6.67 ± 8.59	10.00 ± 12.02	8.54 ± 6.48
4	0.42 ± 0.27	4.17 ± 6.07	9.17 ± 5.34	9.17 ± 5.34	13.33 ± 13.12	11.67 ± 7.31

Table 8: The full results of the patch-based deep learning method on the OULU-NPU dataset.