Fingerprint Liveness Detection

Alice Yue, Anton Cherniavskyi, Guiliang Liu, Guo Liang Gan
Department of Computing Science
Simon Fraser University
{aya43, achernia, gla68, glgan} @ sfu.ca

Abstract

In this project, we try to develop robust machine learning models for fingerprint liveness detection system. We first extract features from fingerprint images and then develop machine learning models such as neural networks and SVM for classification of real and fake fingerprints. The feature extraction methods that we used were BSIF, WLD, LPQ, and CNN-RFW. When training our models, we implemented 10-fold cross-validation and some techniques to combat over-fitting such as dimensionality reduction of features and Gaussian noise layers. In order to have a larger training set, we also introduced new training data by augmenting available training images. We were able to achieve test accuracy ranging from 80% to 99% from different models, in which we outperform most of the contestants of the LivDet 2015 competition on Digital Persona data set.

1 Introduction

Fingerprint recognition is a popular way to perform biometric authentication. However, it is susceptible to spoof attacks [2, 9, 10]. The need for ways to combat spoof attacks has given rise to an area of research called liveness detection. As such, this paper explores solutions for this problem by utilizing machine learning techniques.

1.1 Motivation

Biometric authentication exploits a person's physiological trait, in this case the fingerprint, as a way to verify his/her identity. A spoof attack on a fingerprint recognition system is a process whereby a spoofer replicates the owner's fingerprint using a different material. [11].

Many countermeasures have been taken to deal with spoof attacks, on both the hardware and software level. The focus of software countermeasures are placed on the liveness detection problem: given a fingerprint image, determine whether it comes from a live user.

Liveness detection is a high stake security problem that has a need for highly accurate solutions. The difference between a live print and a spoof one is subtle that even humans have a hard time differentiating them. Hence, liveness detection relies heavily on texture-based pattern recognition.

1.2 Related Works

Due to its less intrusive and more user friendly nature, software based liveness detection has been a preferred method to counter fingerprint spoof attacks [17].

In the early 2000s, the predominant algorithm used was a series of feature extraction techniques after which the feature(s) is is used to train a support vector machine SVM [7]. The resulting classifier is then used to classify unseen fingerprint images, or a test set, as live or spoof. Popular feature extraction techniques include local binary patterns (LBP) [14], local phase quantization (LPQ) [15] and binary statistical image features (BSIF) [8].

More recently, researchers have started to make these classifiers using neural networks. [12] compared the use of convolution neural networks and BSIF as methods to extract features for SVMs, noting the potential effectiveness of convolution for fingerprint images. Meanwhile, [20] trained a neural network on small patches of a fingerprint image, and obtained a consensus score by combining the predictions from all patches of a single image. In the last year, [13] use a modified convolution deep network obtaining breakthrough results and winning the LivDet 2015 competition [16].

1.3 Dataset

We requested and worked with the LivDet 2015 [11] competition data set, the most recent benchmark data set. The data provided consists of both training and test data, each containing a set of live and a set of spoof fingerprint images scanned by different optical devices. Within the course period, we focused on the *Digital Persona* fingerprint image scans and there is a total of 2000 training images(1000 live and 1000 spoof) and 2500 test images(1000 live and 1500 spoof). Other optical devices include Green Bit, Biometrica, and Crossmatch. The materials used to create spoof fingerprints are ecoflex, gelatine, latex and woodglue. Also, the test set contains additional spoofing materials that are not present in the training set: liquid ecoflex and RTV (a silicone rubber).

2 Methods

The methodology we used in this project consists of two phases. The first phase is to extract features from images and the second phase is to train and test different machine learning models using these extracted features. We also spent some time trying Convolutional Neural Network models which take images as direct input. Below, we will first explain some of the methods we used for feature extraction and then we will explain the models that we tried.

2.1 Feature Extraction

The first step in our workflow is to extract a one dimensional feature vector from the training fingerprint images. Here, we utilized well known statistical feature extraction methodologies WLD (Section 2.1.1), LPQ [15], and BSIF [8]. Here we will only describe one of them which is WLD and we refer the readers to respective papers. We also tested both randomly weighted and trained convolution neural networks as alternative methods of feature extraction (see Section 2.1.2).

2.1.1 Weber Local Descriptor *(WLD)*

Weber Local Image Descriptor (WLD) consists of two components: differential excitation (ε) and orientation (Θ). For a target pixel (x_j) in a fingerprint image, we compute the colour intensity difference between x_j and its neighbors $\varepsilon(x_j)$:

$$\varepsilon(x_j) = arctan[\sum_{i=0}^{n} \frac{intensity(x_i) - intensity(x_j)}{intensity(x_j)}] \quad (1)$$

x_0	x_1	x_2
x_3	x_4	x_5
x_6	x_7	x_8

Figure 1: WLD target pixel x_4 and positions of its neighbors

$\arctan$ is a normalization factor to keep $\varepsilon(x_j)$ in a finite range. n is the number of pixels neighbouring x_j such that x_i is x_j's neighbour pixel.For our experiments, we used a 3×3 neighborhood, which is typically capable of extracting effective features for liveness detection [6]. WLD is inspired by Weber's Law [1], which claims that the just noticeable difference is a constant propositional of the original stimulus.

Next, we calculate the gradient orientation Θ. Using the example in Fig[1], the gradient orientation of x_j is defined by:

$$\Theta(x_4) = \arctan(\frac{intensity(x_5) - intensity(x_1)}{intensity(x_7) - intensity(x_3)}) \tag{2}$$

We then collect the differential excitation (ε) and orientation (Θ) for each pixel and reduce them into a single vector. In our case, we extracted 120 bins for differential excitation and 8 bins for orientation, as recommended by [6].

2.1.2 Convolutional Neural Networks using Random Filter Weights *(CNN-RFW)*

CNN is another method of extracting a one dimensional feature from an image. [4] describes a family of CNNs, whose architecture is designed to generate "biologically-inspired" feature vectors of images. The particular CNN model we used is made up of a single image input layer, several interweaving convolution (containing linear filters) and spatial pooling layers, which are then flattened and normalized to become the input image's feature vector. See Table 1 and Figure 3 for more details.

Table 1: Structure of our CNN-RFW model

Block	**Operation**	**Parameters**
Input	Grayscale	-
	Normalize	-
1..3	Linear Filtering	64 filters, 3x3
		128 filters, 5x5
		256 filters, 5x5
	ReLU Activation	min=0, no saturation
	Spatial Pooling	order=1, 7x7
		order=1, 5x5
		order=10, 7x7
	Normalize	-

2.2 Dimensionality Reduction and Principal Component Analysis *(PCA)*

Dimensionality reduction is a technique to combat over-fitting. Dimensionality reduction is a process of obtaining a smaller set of features from the original feature set, which can be used to recover most of the variability in the data. [5] One of the most popular dimensionality reduction techniques is PCA. PCA derives a set of linear approximations from high-dimensional data by returning a lower dimensional *linear* manifold. This manifold represents the maximum variability from a higher dimensional manifold [5].

2.3 Image Augmentation

Image augmentation creates a larger variety and amount of training data which helps in refining our classifiers and again, fight over-fitting. We augmented the images via a two step process, as described in [12]: (1) perform horizontal flip and (2) crop five smaller overlapping images from both the original image and its flipped copy. These five images come seperately from the four corners and from the centre of the image (see Figure 2), resulting in a total of ten new images for every original sample. We chose this strategy because the ideal preprocessing stage should include cropping and eliminate image rotation, thus canceling some transformations of the traditional augmentation techniques.

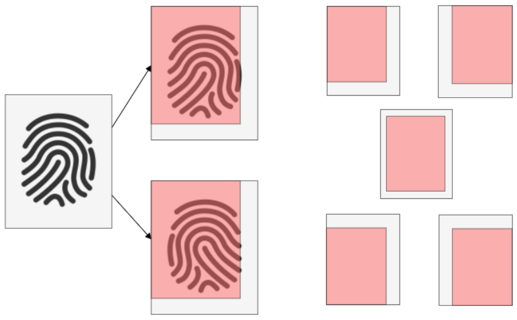
Figure 2: Image augmentation scheme

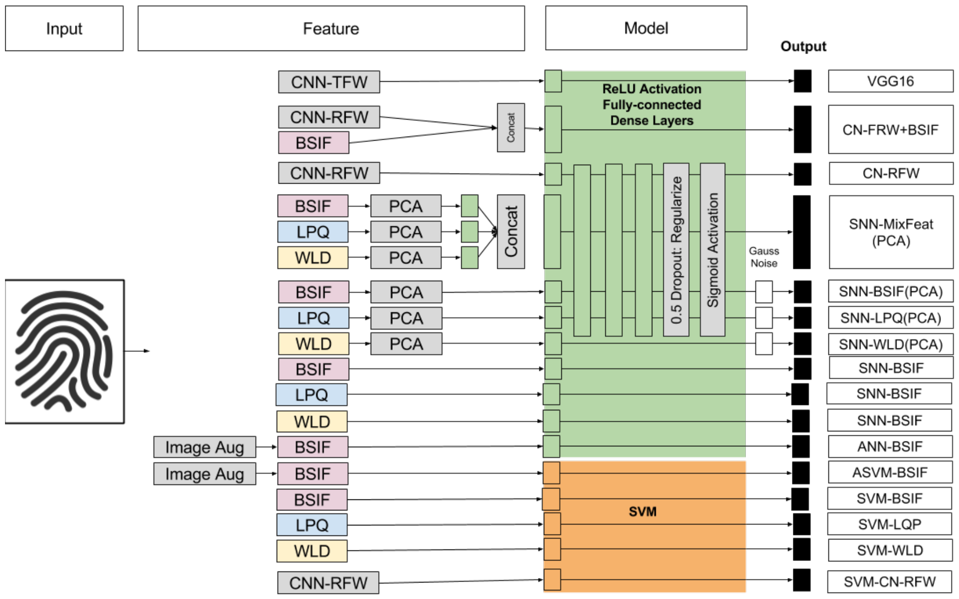

Figure 3: Models Tested

2.4 Description of the Models

Here we present the descriptions of the models we tried.

2.4.1 Support Vector Machines (SVM)

- **SVM with BSIF/LPQ/WLD features**: BSIF, LPQ and WLD features were generated separately for the whole dataset (training and test images). For each set of features, the corresponding SVM models were trained and tested independently. Parameters were selected by cross validation and average validation accuracy for different parameters are shown in Table 2 under Training Process section.
- **SVM with CNN-RFW features**: SVM classifiers were trained on CNN-RFW features.

2.4.2 Neural Networks with Extracted Features

- **Simple Neural Network using BSIF/WLD/LPQ features (SNN-BSIF/WLD/LPQ and SNN-MixFeat)**: Apply dimensionality reduction(PCA) to features extracted by different local image descriptors. SNN-BSIF/WLD/LPQ models are multi-layer neural network that take the reduced features as input. The SNN-MixFeat model merges 3 different models SNN-BSIF, SNN-WLD and SNN-LPQ.

- **Neural Network with CNN-RFW features**: NN classifiers were trained on features extracted via Convolutional neural network. Figure 3 shows the structure of our NN classifier.
- **Neural Network with CNN-RFW and BSIF features**: Concatenated two feature sets via Merge layer to see how NN classifier may take advantage of using multiple feature sets simultaneously.

2.4.3 Convolutional Neural Networks (CNN)

CNNs are different from traditional neural networks as they take into account the spatial structure of images. The CNN we have used are comprised of a sequence of convolutional and pooling layers. Please refer to [13], [18] for more details.

- **VGG16** [18]: We trained the full VGG16 convolution model modifying the output dense layer to suit our binary classification.
- **Inception v3** [19]: We used pre-trained model with a custom classifier block on top: a fully connected layer with ReLU activation, dropout layer with standard rate 0.5, and a single output with sigmoid activation.

2.5 Training and Cross Validation

Most of our implementations are in Python and MATLAB.

2.5.1 SVM

We conducted stratified 10-folds cross validation to find the best SVM model and the best model parameters. Each folds contain the same amount of live and spoof data. We tested different ranges of penalty parameter ν and kernel coefficient γ for Nu-SVM. We chose parameter values providing the highest average validation accuracy over the ten runs. Table 2 shows CV results for the *Digital Persona* dataset.

Table 2: Average accuracy of SVM model with BSIF features during cross-validation

		γ						
		1e-09	1e-07	1e-05	0.001	0.1	10	1e+03
ν	0.3	75.82	79.43	**80.60**	62.18	63.60	60.92	60.67
	0.5	75.50	79.90	79.57	62.64	63.25	60.54	60.79
	0.7	73.80	77.40	76.32	62.01	63.21	60.24	60.15

2.5.2 Neural Networks

All of our neural networks implementations are based on Keras library [3] which works on top of Tensorflow. Same as SVM, we did stratified 10-folds cross validation for our NN models. We used binary cross entropy as a loss function.

$$E = \sum_n t_n \log y_n + (1 - t_n) \log(1 - y_n) \tag{3}$$

Some of our models use AdaDelta [21] and some use Stochastic Gradient Descent optimizers. AdaDelta provides faster convergence as compared to SGD. The validation accuracy for some models are in Table 3. Also, PCA helps combat over-fitting issue and the difference in validation loss can be seen in Figure 4 and Figure 5.

3 Experimental Results

Table 4 and Table 5 shows the performance metrics of our SVM and Neural Network models. Apart from test accuracy, we introduced 2 different metrics namely AUC and ACE. AUC of SVM measures

Table 3: Average validation accuracy of SNN models with dimensionality reduction

	SNN-BSIF	SNN-LPQ	SNN-WLD	SNN-MixFeat
Average Validation Accuracy	99.54	100	99.55	95.82

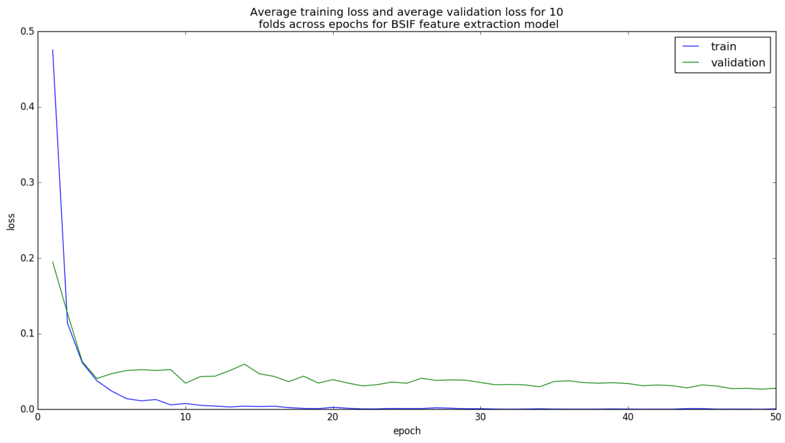

Figure 4: Average Training and Validation Loss without PCA

the probability that it ranks a randomly chosen live example higher than a randomly chosen fake example. ACE measures the average of fake positive rate and fake negative rate i.e. the average of the rate of misclassified live examples and the rate of misclassified spoof examples.

Table 4: SVM test results for Digital Persona data set

Method	Accuracy	AUC	ACE
BSIF + SVM	87	0.94	12.8
LPQ + SVM	89.4	0.96	10.5
WLD + SVM	84.55	0.93	15.45
Augment+BSIF+SVM	84.86	-	13.82
CNN-RFW + SVM	81.16	-	17.95

Table 5: Neural networks test results for Digital Persona data set

Method	Acc.	ACE
SNN-BSIF(no PCA)	85.24	14.28
SNN-BSIF(PCA)	98.28	1.45
SNN-LPQ(no PCA)	94.32	4.92
SNN-LPQ(PCA)	99.2	1.17
SNN-WLD(no PCA)	90.08	8.62
SNN-WLD(PCA)	98.3	1.33
Mix-BSIF-CNN-RFW + NN	83.24	17.15
SNN-MixFeat(no PCA)	86.56	12.45
SNN-MixFeat(PCA)	97.44	2.03
CNN-RFW+NN	81.88	18.16
Inception v3	66.61	28.93
VGG16	85.7	20.06

4 Conclusion

Overall, most of our models performed well, obtaining over 80% and at most 99% accurate. Using Graduate Student Descent(GSD) in experimenting different models, BSIF, WLD and LPQ features

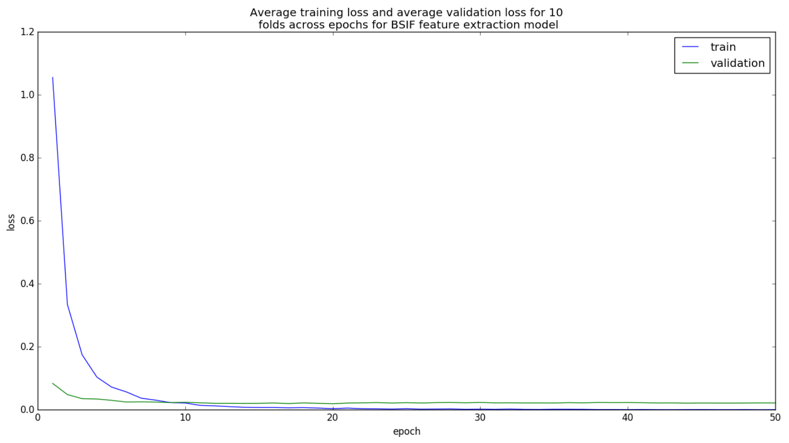

Figure 5: Average Training and Validation Loss with PCA

suit both SVM and NN classifiers better in comparison to CNN-RFW features, as shown in Table 5. Furthermore, data augmentation, dimensionality reduction method and Gaussian Noise layers in neural networks also help in attaining higher validation and test accuracy as it combats over-fitting.

We were unable to get the accuracy of Inception V3 with weights trained on ImageNet on par with other models. Our hypothesis is that it may not have captured the distinctive properties of live and fake fingerprints such that it is generalizable to unseen data (i.e. its network structure may be too complex for our task).

Throughout this project, we recommend testing more neural network structures on the liveness detection problem based on features extracted from local image descriptors such as BSIF and LPQ. Furthermore, we notice that there is a significant over-fitting issue for this problem and we recommend using combinations of techniques such as image augmentation, dimensionality reduction, Gaussian noise layers and Dropout rates to finetune machine learning models.

5 Contributions

Alice Yue: Alice trained the full VGG16 model, wrote initial models for concatenated feature models, contributed to the report and the poster.

Anton Cherniavskyi: Anton extracted CNN-RFW features, trained CNN-RFW + SVM/NN, mixed inputs CNN-RFW/BSIF + NN and Inception V3), tested data augmentation, contributed to the report and the poster.

Guiliang Liu: Guiliang extracted BSIF, LPQ and WLD features, applied lib-SVM to train the model and test the classifying accuracy, trained Inception on the images, contributed to the report and poster.

Guo Liang Gan: Guo Liang worked on the SNN models, with and without PCA. Besides, he is also in charge of the reports and poster design.

References

[1] S Annadurai. *Fundamentals of digital image processing*. Pearson Education India, 2007.

[2] Battista Biggio, Zahid Akhtar, Giorgio Fumera, Gian Luca Marcialis, and Fabio Roli. Security evaluation of biometric authentication systems under real spoofing attacks. *IET biometrics*, 1(1):11–24, 2012.

[3] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[4] David Cox and Nicolas Pinto. Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In *Automatic Face and Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference*. IEEE, 2011.

[5] Ali Ghodsi. Dimensionality reduction a short tutorial. *Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada*, 37:38, 2006.

[6] Diego Gragnaniello, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva. Fingerprint liveness detection based on weber local image descriptor. In *Biometric Measurements and Systems for Security and Medical Applications (BIOMS), 2013 IEEE Workshop on*, pages 46–50. IEEE, 2013.

[7] Qiangui Huang, Sheng Chang, Chun Liu, Binbin Niu, Meng Tang, and Zhe Zhou. An evaluation of fake fingerprint databases utilizing svm classification. *Pattern Recognition Letters*, 60:1–7, 2015.

[8] Juho Kannala and Esa Rahtu. Bsif: Binarized statistical image features. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 1363–1366. IEEE, 2012.

[9] Tsutomu Matsumoto, Hiroyuki Matsumoto, Koji Yamada, and Satoshi Hoshino. Impact of artificial gummy fingers on fingerprint systems. In *Electronic Imaging 2002*, pages 275–289. International Society for Optics and Photonics, 2002.

[10] Helen Meyer. Six biometric devices point the finger at security. *Computers & Security*, 17(5):410–411, 1998.

[11] Valerio Mura, Luca Ghiani, Gian Luca Marcialis, Fabio Roli, David A Yambay, and Stephanie A Schuckers. Livdet 2015 fingerprint liveness detection competition 2015. In *Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on*, pages 1–6. IEEE, 2015.

[12] Rodrigo Frassetto Nogueira, Roberto de Alencar Lotufo, and Rubens Campos Machado. Evaluating software-based fingerprint liveness detection using convolutional networks and local binary patterns. In *Biometric Measurements and Systems for Security and Medical Applications (BIOMS) Proceedings, 2014 IEEE Workshop on*, pages 22–29. IEEE, 2014.

[13] Rodrigo Frassetto Nogueira, Roberto de Alencar Lotufo, and Rubens Campos Machado. Fingerprint liveness detection using convolutional neural networks. *IEEE Transactions on Information Forensics and Security*, 11(6):1206–1213, 2016.

[14] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

[15] 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.

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[17] Stephanie AC Schuckers. Spoofing and anti-spoofing measures. *Information Security technical report*, 7(4):56–62, 2002.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[19] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015.

[20] Chenggang Wang, Ke Li, Zhihong Wu, and Qijun Zhao. A dcnn based fingerprint liveness detection algorithm with voting strategy. In *Chinese Conference on Biometric Recognition*, pages 241–249. Springer, 2015.

[21] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.