



# Hochschule Darmstadt

Fachbereiche Mathematik und
Naturwissenschaften & Informatik

# Unknown Fingerprint Presentation Attack Detection Using Convolutional Autoencoders

Abschlussarbeit zur Erlangung des akademischen Grades

Master of Science (M. Sc.)
im Studiengang Data Science

vorgelegt von

**Marcel Grimmer**

| | | |
|---|---|---|
| Referent | : | Prof. Dr. Christoph Busch |
| Co-Referent | : | Jascha Kolberg |

| | | |
|---|---|---|
| Ausgabedatum | : | 07.11.2019 |
| Abgabedatum | : | 23.04.2020 |

# Eidesstattliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

*Weiterstadt, 17.04.2020*

Marcel Grimmer

# Abstract

The increasing popularity of biometric authentication systems simultaneously raises privacy and security issues. Among other vulnerabilities, presentation attacks (PAs) that are directed to the capture device pose a severe threat. In order to be prepared against such attacks, presentation attack detection methods are deployed. However, due to the variety of materials that can be used to fabricate a presentation attack instrument (PAI), the classification models must be designed to also protect against unknown presentation attacks.

The contribution of this thesis is the development of an unsupervised learning technique based on Convolutional Autoencoders and finger images stemming from two novel sensor technologies: *Laser Speckle Contrast Imaging* and *Multi-Spectrum Short-Wave Infrared*. On an experimental evaluation over a database of $19,598$ bona fide images and $4,226$ PAs, including 43 unique PAIs, an average detection equal error rate of 2.47% could be achieved.

# Contents

# 1. Introduction

According to the US Federal Trade Commission, more than 270,000 cases of credit card identity theft have been reported in 2019 with almost 165 million records containing personal data being exposed [15]. This indicates, that personal identification is crucial for many modern applications. However, using passwords and pins has a major drawback as they can be forgotten, lost, or stolen. To circumvent this problem, biometric authentication techniques focus on identification, based on unique biological characteristics. In this context, fingerprints have proven to be sufficiently unique, easy to capture, and less invasive than other biometric characteristics such as retina scans. Therefore, on 13th December 2004, the European Union issued a council regulation [20] for its member states which obligates them to store in every passport the fingerprint images of the document holder in order to ensure a strong biometric link between the biometric characteristic (the fingerprint) and the travel document. With a similar motivation recently the *Smart Border Project* of the European Commission [11] has been initiated to keep track of travellers from third countries, crossing the external borders of the Schengen area. This requires effective and efficient authentication systems that focus both on convenience and border security. As part of the Smart Border project, the *Entry/Exit System (EES)* [9] is planned to be a IT system to register travellers from third-countries, including the storage of fingerprints. Furthermore, since 2003, the *European Dactyloscopy (EURODAC)* system is used as an EU wide fingerprint database that assists with determining the member state responsible for examining an asylum application [10].

With regard to the wide range of applications, the growing demand raises the question of how vulnerable these systems are to external attacks. In this context, among numerous attack points, presentation attacks (PAs) constitute the greatest weakness since the attacker does not need any knowledge about the internal biometric system. To address that vulnerability, it is important to deploy algorithms which are able to distinguish between bona fide presentations and access attempts carried out by a remodelled or overlaid finger. The current state of research shows that there are many commercially available materials, such as Play-Doh or silicone, that can be used to execute a PA. Although many presentation attack instruments (PAIs) are already known, it turns out to be very challenging to learn a classification model that successfully detects the whole variety of fabricates. On the one hand, this is related to the high costs involved in fabricating an adequate number of PAIs. On the other hand, it is hard to protect the biometric system from PAs performed with previously unseen PAIs as most fingerprint presentation attack detection (PAD) approaches rely on classifying PAs on the basis of their similarity to those included in the training phase (*known PAD*). A common method to simulate the impact of unseen samples on the models performance is to include

a subset of PAIs only in the testing phase. Nevertheless, this leads to the problem of how to compose the subset in order to obtain results as close as possible to the real world scenario. Since these approaches still involve PAs within the training phase, they will be referred to as *semi-known PAD*.

Instead of learning about both the structure of bona fides as well as various PAs, *unknown* PAD (UPAD) methods are trained on bona fides solely. The main contribution of this thesis is to develop a UPAD method by implementing *Convolutional Autoencoders (Convolutional AEs)*. Convolutional AEs are learned to compress images while being able to reconstruct them without major loss of information. However, the reconstruction of the encodings is highly sensitive and fails for images that are not similar to those contained in the training set. This can be exploited to detect PAs by training a Convolutional AE only on bona fides. In the course of this work, three different model architectures are implemented and compared with each other. Additionally, a novel loss function has been elaborated in order to increase the robustness of the AE models. The final detection performance of the Convolutional AE model is then benchmarked against a semi-known PAD approach using pre-trained Convolutional Neural Networks, proposed by Gomez-Barrero et al. [26]. Furthermore, a set of One-Class Support Vector Machines (OC-SVMs) is trained to establish further unknown PAD baselines for the benchmark in this thesis. The experiments within this thesis are based on finger images that were captured during the BATL project [7] using a novel capturing device that was first introduced by Hussein et al. [31], including two novel sensor technologies: *Laser Speckle Contrast Imaging (LSCI)* and *Multi-Spectral Short-Wave Infrared (SWIR)*.

The thesis is organized as follows: Section 2 provides fundamental knowledge about fingerprint PAD, including ISO-standardized terminology. Next, section 3 presents the contributions of other authors dealing with related topics. Furthermore, section 4 introduces the methodology applied within this thesis, mainly focusing on techniques stemming from the deep learning domain. As a preparation for the following parts, section 5 describes the experimental framework which contains details about dataset, performance metrics, and implementation. Additionally, section 6 gives a brief overview of the proposed PAD methods. Section 7 then presents the experimental results of different Convolutional AE settings. Finally, in section 8 and 9 the elaborated AE model is benchmarked against other fingerprint PAD approaches and a final conclusion is drawn with suggestions for future works.

# 2. Fingerprint PAD Fundamentals

This section familiarizes the reader with the fundamental concepts of fingerprint PAD and introduces topic-related terms.

## 2.1. Standardized Terminology

To accelerate research in the field of biometric PAD and simultaneously increase the domain-related comparability of results, the technical committees from the International Organisation for Standardisation (ISO) and the International Electrotechnical Commission (IEC) established a standardized terminology and a testing framework. Since the contributions presented within this work are in compliance with the ISO suggestions, the most relevant domain-related terms are presented, extracted from the following documents.

- ISO/IEC 2382-37:2017 (*Information Technology - Vocabulary - Part 37: Biometrics)* gives a systematic description of the concepts in the field of biometrics [33]

- ISO/IEC 30107-1:2016 (*Information Technology — Biometric Presentation Attack Detection — Part 1: Framework* introduces terms and definitions that are useful in the specification, characterization and evaluation of presentation attack detection methods. [35]

- ISO/IEC 30107-3:2017 (*Information Technology — Biometric Presentation Attack Detection — Part 3: Testing and Reporting* provides principles and methods for performance assessment of presentation attack detection mechanisms. [36]

The following list of basic biometric terms has been extracted from ISO/IEC 2382-37:2017 without changes since they need to be formulated accurately in accordance to the standards.

- **biometric capture:** *obtain and record, in a retrievable form, signal(s) of biometric characteristic(s) directly from individual(s), or from representation(s) of biometric characteristic(s)*

- **biometric capture device:** *device that collects a signal from a biometric characteristic and converts it to a captured biometric sample. A biometric capture device can be any piece of hardware (and supporting software and firmware).A biometric capture device may comprise components such as an illumination source, one or more biometric sensors, etc.*

- **biometric capture process:** *series of actions undertaken to affect a biometric capture*
- **biometric capture subject:** *individual who is the subject of a biometric capture process*
- **biometric capture subsystem:** *biometric capture devices and any sub processes required to execute a biometric capture process*
- **biometric characteristic:** *biological and behavioural characteristic of an individual from which distinguishing, repeatable biometric features can be extracted for the purpose of biometric recognition*
- **biometric data:** *biometric sample or aggregation of biometric samples at any stage of processing, e.g. biometric reference, biometric probe, biometric feature or biometric property*
- **biometric data subject:** *individual whose individualised biometric data is within the biometric system*
- **biometric enrolment:** *act of creating and storing a biometric enrolment data record in accordance with an enrolment policy*
- **biometric enrolment data record:** *data record attributed to a biometric data subject, containing non-biometric data and associated with biometric reference dentifier(s)*
- **biometric identification:** *process of searching against a biometric enrolment database to find and return the biometric reference identifier(s) attributable to a single individual*
- **biometric presentation:** *interaction of the biometric capture subject and the biometric capture subsystem to obtain a signal from a biometric characteristic*
- **biometric recognition:** *automated recognition of individuals based on their biological and behavioural characteristics*
- **biometric sample:** *analogue or digital representation of biometric characteristics prior to biometric feature extraction*
- **biometric system:** *system for the purpose of the biometric recognition of individuals based on their behavioural and biological characteristics*

- **biometric verification:** *process of confirming a biometric claim through biometric comparison*
- **captured biometric sample:** *biometric sample resulting from a biometric capture process*
- **mode:** *combination of a biometric characteristic type, a sensor type and a processing method*
- **multi-modal:** *multiple in at least 2 out of 3 constituents of a mode in a single biometric system*

Furthermore, another set of standardizes terms stemming from ISO-IEC-30107:2016 [35] is defined as follows:

- **artefact:** *artificial object or representation presenting a copy of biometric characteristics or synthetic biometric patterns*
- **presentation attack (PA):** *presentation to the biometric data capture subsystem with the goal of interfering with the operation of the biometric system*
- **presentation attack detection (PAD):** *automated determination of a presentation attack*
- **presentation attack instrument (PAI):** *biometric characteristic or object used in a presentation attack*

Finally, the last list of relevant terms from ISO-IEC-30107-3:2017 [36] is presented:

- **bona fide presentation:** *interaction of the biometric capture subject and the biometric data capture subsystem in the fashion intended by the policy of the biometric system*
- **attack type:** *element and characteristic of a presentation attack, including PAI species, concealer or impostor attack, degree of supervision, and method of interaction with the capture device*
- **PAI species:** *class of presentation attack instruments created using a common production method and based on different biometric characteristics*
- **attack presentation classification error rate (APCER):** *proportion of attack presentations using the same PAI species incorrectly classified as bonafide presentations in a specific scenario*
- **bona fide presentation classification error rate (BPCER):** *proportion of bona fide presentations incorrectly classified as presentation attacks in a specific scenario*

## 2.2. Biometric Characteristics

The primary goal of biometric systems is to automatically recognize individuals based on one or more physiological and/or behavioural characteristics. However, according to ISO-IEC-24745:2011 [34] biometric characteristics need to fulfill specific properties in order to be able to reliably discriminate between individuals:

- **universality**: Every individual should have the characteristic.
- **uniqueness**: The characteristic should be distinguishable between individuals.
- **permanence**: The characteristics should not alter over time.
- **collectability**: The collection of the characteristic should be easy.
- **repeatability**: The characteristics should be sufficiently distinct and repeatable.

In theory, the above mentioned properties are sufficient to unambiguously recognize an individual. However, from the application point of view there are three additional properties which are considered as important:

- **performance**: Refers to the success rate in the recognition process.
- **acceptability**: Refers to the level of willingness of the subject using the biometric system.
- **spoof resistance**: Refers to the question of how difficult it is to replicate the biometric characteristic to bypass the biometric system

In the context of this thesis, only fingerprints will be taken into account which have proven to comply with most of the above mentioned criteria. From a history point of view, the modern usage of fingerprints began in the late 19th century [58] where authorities started to use body characteristics to recognize individuals. They figured out that fingerprints seem to differ from data subject to data subject and therefore concluded that it can be seen as an unique human trait. In 2004, a study from Han et al. [30] evaluated the uniqueness of fingerprints using a statistical analysis. In fact, they confirmed that fingerprints are sufficiently unique to distinguish one subject from another, even between identical twins. Nonetheless, the usage of fingerprints also has a drawback which is its vulnerability to presentation attacks. Today, imitating fingerprints can be done with low cost resources (e.g. Play-doh) and easy to follow online tutorials. Further, presenting a PAI to the capture device, an attacker does not need any background knowledge of how the internal biometric

comparison system is structured. Consequently, the demand for effective PAD methods is omnipresent. To this end, the next subsection introduces the reader further to the subject of PAs and explains where they take place within a biometric system.

## 2.3. Presentation Attack Detection

To have a better understanding of PAs, it is mandatory to understand how the underlying biometric system is structured. For this purpose, this subsection will summarize the most important ideas of the conceptual structure of a biometric system according to ISO/IEC 24745:2011 (Information Technology — Security Techniques — Biometric Information Protection) [34]. All operations that are included in a biometric system are depicted in figure 2.1.

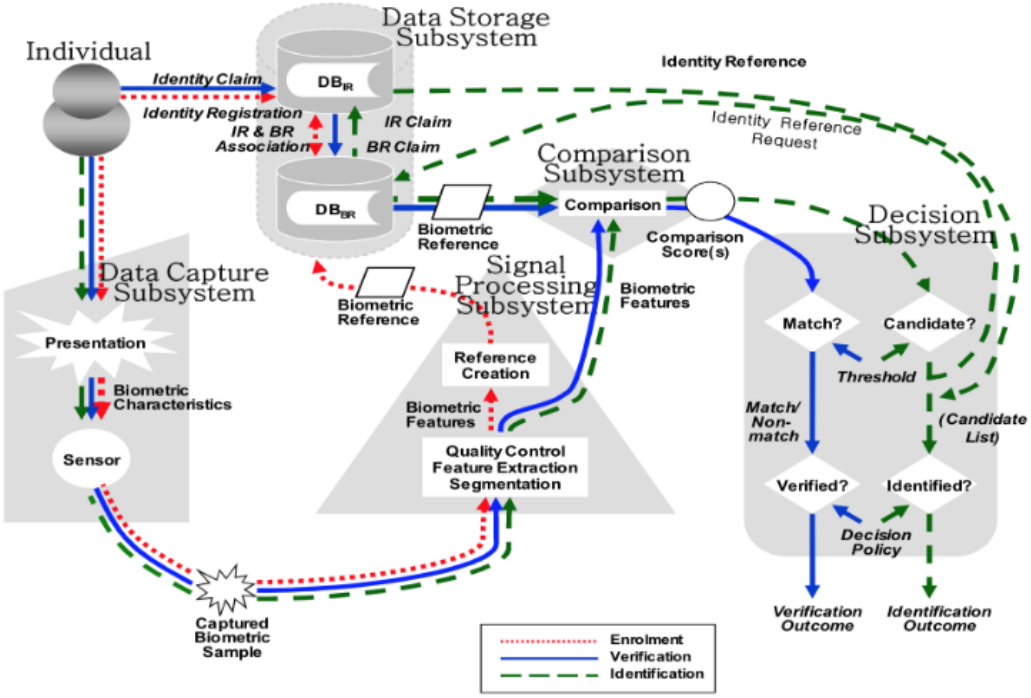


**Figure 2.1:** Conceptual structure of a biomteric system according to ISO/IEC 24745:2011 [34]

The gray areas indicate that the structure basically consists of the following parts:

- A **biometric data capture subsystem** that comprises biometric capture devices and/or sensors to capture signals from a biometric characteristic and turn them into biometric samples.

- A **signal processing subsystem** which extracts biometric features from a biometric sample to identify or verify them against biometric references stored in the data storage subsystem.
- A **data storage subsystem** that stores the biometric samples to link the enrolled biometric references to the identity reference.
- A **comparison subsystem** which ascertains the similarity between captured biomtric samples and stored biomteric references.
- A **decision subsystem** that determines whether the captured biometric sample and the biometric reference stem from the same biometric capture subject.

All of the above mentioned components in a biometric system can be target of an attack. Figure 2.2 shows all of the potential attack points as indicated by ISO/IEC 30107-1:2016 [35]. This thesis focuses on the detection of presentation attacks, highlighted by blue background color. The data capture subsystem is the most vulnerable component since the attacker does not need a deep understanding of the underlying biometric system to present a faked finger.

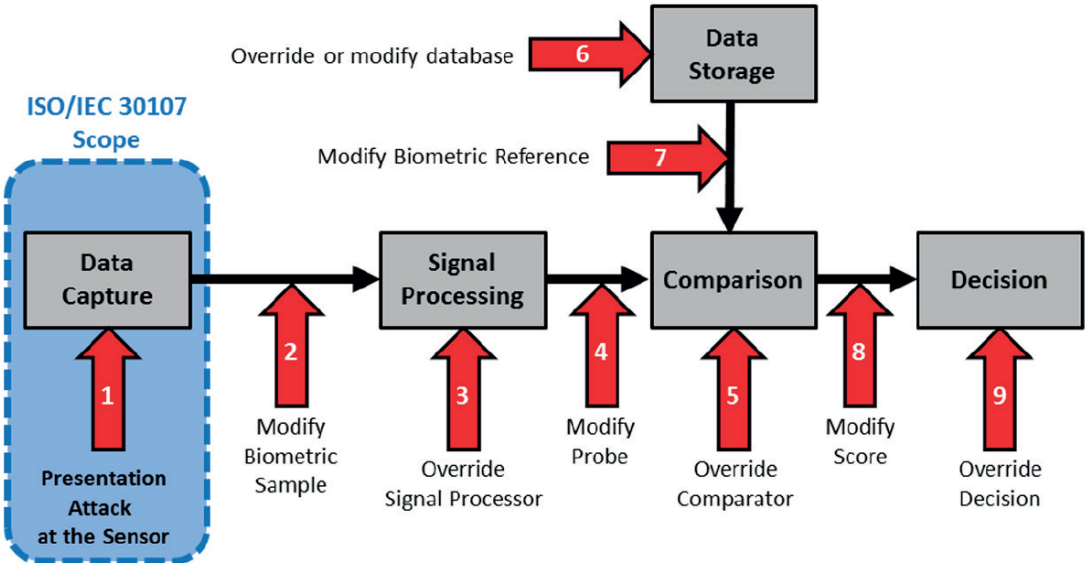


**Figure 2.2:** Generic attacks in a biometric system according to ISO/IEC 30107-1:2016 [35]

In general, a subversive biometric capture subject carrying out a PA can be classified as a biometric impostor or concealer. A biometric impostor aims to be verified as another subject whereas a biometric concealer intends to circumvent being recognized as he is aware that he is known to the system. This leads to the

challenge of designing both software and hardware based approaches to be able to detect impostor as well as concealer attacks. Detecting presentation attacks is challenging because of the wide range of distinct materials used to fabricate PAIs. A single PA that has been executed successfully can pose a serious security threat since it promotes the knowledge about weaknesses in the biometric capture subsystem. Fingerprint PA methods can be divided into *cooperative* casts on the one hand and *non-cooperative* casts on the other hand [3]. Cooperative casts refer to the fabrication of fake fingerprints or overlays with various materials like thermoplastic or silicon by exploiting the availability of the original bona fide. If the original fingerprint, in contrast to the cooperative casts, is not available, non-cooperative methods are used. They refer to the production of a PAI by indirectly obtaining the according fingerprint patterns. For example, the latent fingerprints individuals leave on suitable surfaces which can then be used to recreate the patterns of the fingerprint.

## 2.4. Soft- and Hardware-based PAD

To detect PAs on fingerprint biometric systems both software and hardware based mechanisms can be used. Software based methods tie in with the data generated by a sensor and are able to distinguish between bona fide and PA samples by processing the signal patterns contained in the input data. However, software based solutions are restricted to the sensors ability of capturing data that contains relevant patterns needed to detect PAs. On the other hand, software based approaches are appealing due to the low costs incurring during the deployment.

Software based approaches can be split into static and dynamic techniques where static methods are based on one single 2D scan and dynamic methods analyse a series of images. Basically the dynamic approach can be seen as an extension to the static one, taking into account the changes over time. Sweat pores [19] as well as the fingerprint ridges [70] are often used as characteristics in the field of static software based PAD as they can be easily extracted from high resolution images. The dynamic approaches are more complex because both hardware and software needs to be constructed in a way that enables capturing and processing multiple frames. For example, the analysis of multiple images over time can be useful for observing the distortion of the skin [4] that results by pressing the finger on the capturing device.

Hardware-based setups use information collected by additional sensors in order to detect particular properties of a living trait such as the blood pressure [45], heartbeat [6], or skin impedance [56]. Often, those approaches are more expensive compared to software-based solutions since they require additional hardware to be installed. To provide a more specific overview, section 3 summarizes both soft- and hardware based work that is related to this thesis.

# 3. Related Work

This section summarizes previous works related to the contributions of this thesis. The first subsection introduces a hardware-based approach that enables to capture fingerprints with two novel sensor technologies. Further, the effectiveness of using LSCI and SWIR data for fingerprint PAD will be validated by referencing promising research results. The second subsection is divided into two parts: First, different methods for detecting unknown and semi-known PAs will be presented. The second part introduces a general approach of how Convolutional AEs can be used for PAD.

## 3.1. Hardware-based Related Work

### 3.1.1. Capturing Device: LSCI & SWIR

One precondition to successfully detect PAs is to use hardware that is able to capture the structural differences between bona fides and PAs. In this context, the results presented within the thesis are based on a novel capture device, first introduced by Hussein et al. [31]. Among others, it includes two sensors: Multi-Spectral Short Wave Infrared (SWIR) and Laser Speckle Contrast Imaging (LSCI). Both sensing technologies operate in the SWIR spectrum between 900-1700 nm which turned out to be a huge advantage as it has a distinctive response to human skin and is independent of skin tone [12]. This has also been shown by Steiner et al. [68] who compared the remission intensities of six different skin tones and four common PAIs. They showed that different skin types react differently to the visible light spectrum, especially at wavelengths below 800 nm. They also found that the six skin tones were indistinguishable from the four presentation attack instruments when a visible light source below 700 nm was used. However, for wavelengths above 800 nm, they showed that the remission intensities of all skin tones behave similarly and on the other hand that they differ from those of the PAIs.

The new fingerprint capture device of Hussein et al. [31] has been constructed in a way that it is able to use Multi-Spectral SWIR and LSCI illumination while preserving backward compatibility with legacy systems. The capturing process works by the capture subject placing its finger on a fingerslot within the sensor. The fingeprint PAD camera is used to capture both SWIR and LSCI images. This involves switching between a 25 mm SWIR lens and a laser lens. The camera is a $64x64$ mm InGaAs area sensor with an almost linear response for wavelengths between 950 nm and 1650 nm. The field of view of both SWIR and LSCI lens corresponds to an area of $45x45$ mm. The interested reader is referred to Hussein et al. [31] for more detailed information on the design of this capture device.

### 3.1.2. Fingerprint PAD using LSCI & SWIR

Keilbach et al. [41] proposed a fingerprint PAD approach using LSCI data. To discriminate between bona fide and PA samples, they extracted handcrafted features such as *Local Binary Pattern [53] , Histograms of Oriented Gradients [14]* and *Binarized Statistical Image Features [39]* and applied SVMs for the classification part. The scores of the SVMs are then fused using majority voting. On a private dataset of 770 samples a BPCER of 0.21% vs. an APCER of 15.48% could be achieved. However, the authors reported that most of the misclassified PAIs are thin and transparent overlays such as dragon skin.

Another work on fingerprint PAD using SWIR data has been presented in [73] using Convolutional Neural Networks (CNN) for classification. In this context, the pretrained VGG-19 CNN model has been fine-tuned on a small private dataset including 12 different PAIs. The PA and bona fide samples could be separated without misclassifications.

In February 2019 Tolsana et al. [72] presented an analysis of data stemming from a new capture device, which was first introduced by Hussein et al. [31], enabling the acquisition of images within the short wave infrared (SWIR) spectrum. With their own database of 4700 bona fide samples and 35 different PAIs, Tolsana et al. introduced a method to detect PAs. On the one hand, they trained a residual CNN from scratch, on the other hand they fine-tuned a pretrained model (VGG-19). By fusing the results of the two models into one single prediction they were able to achieve a Detection Equal Error Rate (D-EER, i.e. APCER=BPCER) of 1.35%. To have an intuition of how the model behaves in regard to unknown presentation attacks, they used 5 different PAI species only for testing purposes.

A multi-model fingerprint PAD approach has been introduced by Gomez-Barrero et al. [27] who extract various features from both LSCI and SWIR images. SVMs are trained for each feature and the the scores are fused with a weighted sum to increase the robustness of the final classification. On their own dataset of 4700 samples they report an APCER of 6.6% vs. a BPCER of 0.2%.

A second multi-model approach of Gomez-Barrero et al. [26] is to apply two CNN models to the SWIR data. The first one is a fine-tuned version of the pretrained VGG-19 CNN model, while the second one is a shallow residual network trained from scratch. However, for the LSCI data, SVMs are applied to various handcrafted features, similar to Keilbach et al. [41]. Finally, the prediction scores of both datasets are fused which leads to an APCER of 3.15% vs. a BPCER of 0.12%.

## 3.2. Unknown Fingerprint PAD

Ding et al. [18] presented an approach to solve the problem of Fingerprint UPAD using a so called *one-class classification* paradigm. They introduced an ensemble of multiple one-class Support Vector Machine (OC-SVM) classifiers, each of which is trained on different feature sets (e.g. *Local Binary Pattern* [53]). The goal of every OC-SVM is to find the smallest possible hypersphere around the majority of training samples that relate to the class of bona fides. The boundaries of the hyperspheres are then refined using a small number of PA samples. To obtain a single classification, the estimates of all OC-SVMs are fused with an algorithm called *Least Square Estimation* proposed by Ding et al. [18]. The authors reported an APCER of 15.3% on the LivDet 2011 database [77].

Another way of solving the issue of UPAD was introduced by Rattani et al. [59]. They see a connection between the coarseness of the surface of fake fingerprint images and the consequent inability of models to identify new and unseen PAIs. They claim that the individual surface structures of the PAIs involved in the training process results in an overfitted model. To solve this problem they used both linear filter-based and non-linear threshold-based methods to eliminate the noise factors contained in the input images. After the preprocessing step, they exract a feature called *Local Binary Pattern* that they used as input for SVMs to obtain the classification result, similar to [18]. On the LivDet 2011 database they were able to achieve a D-EER of 10.2%.

Recently, Gonzàles-Soler et al. [28] introduced a new method using *dense Scale Invariant Feature Transformation* (dense-SIFT) [46] to extract local features from fingerprint images. In the second stage they apply three different feature encodings: *Bag of Words [13], Fisher Vectors [61] and Vector Locally Aggregated Descriptors [38].* For the final classification they utilize Support Vector Machines, one for each of the three encoded feature vectors. The authors highlight the ability of the proposed method of being able to generalize and define a common feature space between bona fide and PA samples. To test the performance on unknown samples, similar to [72], certain PAIs were excluded from training and used only for testing. In an evaluation of the LivDet 2015 databases [51], an average BPCER of 14.27% compared to an APCER of 1% could be measured.

## 3.3. PAD using Convolutional Autoencoders

Nikisins et al. [52] proposed a novel approach for face PAD using a combination of pretrained AEs and a simple Multi-Layer Perceptron (MLP) for the final classification. The multi-channel Autoencoders are used to extract features from multi-channel input data, which in this case is a stack of gray-scale, near infrared and Depth facial (BW-NIR-D Domain) images. It is worth mentioning that the

AE are trained only using bona fide samples, learning the appearance of real faces. Instead of collecting a lot of training data, they used transfer learning techniques, transferring the knowledge of facial images from RGB to BW-NIR-D Domain. Only the subsequently used MLP is trained using both bona fides and PAs to classify the input images. Based on a pretrained model, which was built on a database containing only RGB facial images (CelebA [47]), they fine-tuned their model on the Wide Multi-Channel Presentation Attack database (WMCA [24]), reporting a BPCER of 7.3% with a corresponding APCER of 1%.

To the best of my knowledge no previous works focused on using Convolutional Autoencoders for Fingerprint PAD based on LSCI or SWIR data.

# 4. Background

This section serves to build a fundamental understanding of the methods applied within the experimental part of this work. Nonetheless, since many deep learning techniques are very complex, some conceptual ideas will be discussed more detailed compared to others.

## 4.1. Fully-Connected Neural Networks

Fully-Connected Neural Networks (Fully-Connected NN) are the quintessential deep learning methods with a wide range of applications. All of the basic ideas presented in this subsection are extracted from Goodfellow et al. [29]. However, as a first step to understand the concept of a Fully-Connected NN, it is important to comprehend the basic notion of a perceptron, whose structure is depicted in figure 4.1.

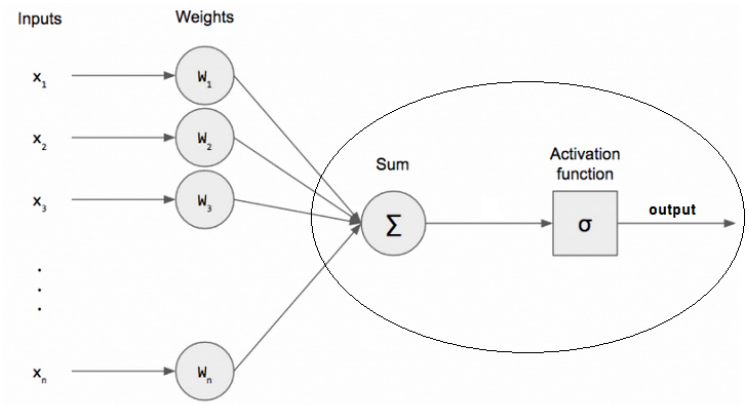


**Figure 4.1:** Structure of a Perceptron [17]

A perceptron can be imagined as a function that transforms an input vector x into a scalar value by calculating a linear combination between the vector elements $x_1, \ldots, x_N$ and a set of weights $w_1, \ldots, w_N$. Subsequently, the result of the linear combination is given to a non-linear activation function. The non-linearity criterion is important since otherwise, the network would only be able to learn linear patterns. Typically, a Fully-Connected NN can be described as a composition of multiple perceptrons that are interconnected and formed to layers, as shown in figure 4.2

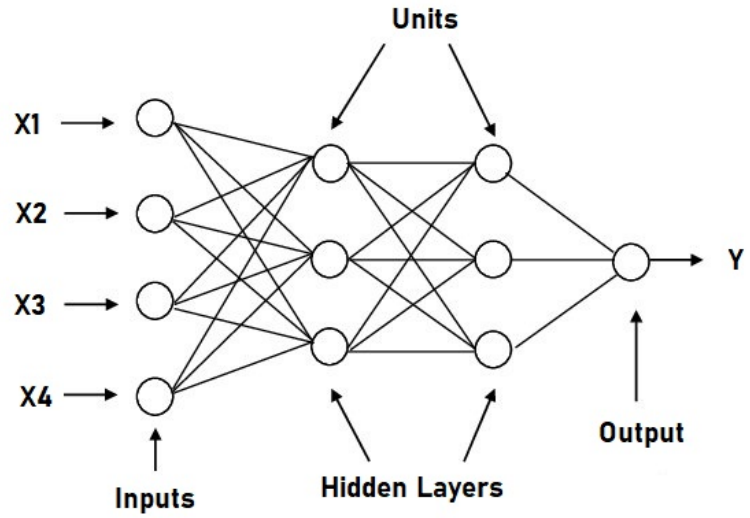


**Figure 4.2:** Example of a Fully-Connected NN [49]

The following components can be defined within a Fully-Connected NN:

- The *input layer* equals the input vector x
- The *output layer* corresponds to the last layer that is contained within the network
- All layers that are located between input and output are called *hidden layers*
- The nodes (i.e. perceptrons) within a layer are referred to as *units*

In figure 4.2, each of the units contained in layer l are connected to all of the units from the neighbored layers l-1 and l+1, which is the name given main characteristic of a Fully-Connected NN. However, the primary goal is to approximate some function $f$ by defining a mapping $y' = f'(x, \theta)$. Specifically, the parameters $\theta$ need to be learned in order to find the best function approximation. Translated to the above shown example, $\theta$ equals the weights $w_{ij}^l$ with $l$ denoting the layer, $i$

representing the unit within layer $l$, and $j$ defining the position of the weight within unit $i$. Yet, the challenge remains of how to find the optimal weight parameters. For this reason, the following three learning stages are briefly outlined.

The first phase is referred to as *feedforward*. Within this stage, the vector x is fed to the input layer and is then propagated through the entire network until it reaches the output layer, which produces the final result $y' = f(x, \theta)$ . Since the main goal of using Fully-Connected NNs is to approximate an unknown function $f$, the discrepancy between $y'$ and the given target values $y$ needs to be measured. This problem is addressed in the second stage, which involves calculating the value of a *loss function* $L(y, y')$. The primary goal of a Fully-Connected NN is to minimize the loss value by finding the optimal weight factors, which leads to the last stage, the *backpropagation and updating* phase. Backpropagation is a technique to propagate back the error from the loss function through the network. This can be done by calculating the gradients $\frac{\delta L(y,y')}{\delta w_{ij}^{l}}$ of the loss function in respect to all of the weights contained in the network. The gradients are then used to apply a *gradient descent* algorithm which updates the weights by changing them into the direction of the steepest descent of the partial derivatives. The updated weights are then used again for the feedforward phase, followed by the other steps. This leads to an iterative adjustment of the weights until a stop criterion is reached.

A major drawback of Fully-Connected NNs is the high number of parameters resulting from many units being connected. Especially for high dimensional inputs like images, this soon leads to the systems working on capacity. An alternative approach are *Convolutional Neural Networks* which are better suited to process multidimensional inputs.

## 4.2. Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a specialized kind of an Artificial Neural Network which can be assigned to the group of deep learning algorithms. CNNs were first mentioned in 1980 by Kunihiko Fukushima [23]. Research and development has since suffered from the high computational power required to use CNNs for both high-dimensional and large-scale data. This problem has been addressed in recent years by the continuous improvement of so-called graphical processing units (GPU). One reason why CNNs have become so popular, is because of their high efficiency compared to fully connected neural networks. This is due to the significantly lower amount of parameters involved in the training process. In this section, we will outline the major concepts of a CNN which are taken from Goodfellow et al. [29] and Aghadam et al. [2].

A typical layer of a CNN includes three different stages: The first stage uses several convolutions in parallel to produce a set of linear activations. In the second

stage, nonlinear functions are applied on the feature maps from the convolution operation, such as the *rectified linear activation function*. This enables the CNN to also understand nonlinear patterns from the input data. The **Pooling** stage is the last one involved in a convolution layer. Each of these components will be described in more detail in the following subsections.

### 4.2.1. Convolution Operation

Convolution is a linear operation that is based on two functions and allows to handle inputs of variable size. The first function $I(i, j)$ returns the value of the pixel stored at row $i$ and column $j$ of the input matrix. The second function $K(i, j)$ refers to the kernel and returns the weight at position $(i, j)$. The output that results by applying the convolution operation is called **feature map**. To have a common basis, convolution is mathematically defined as follows:

$$S(i,j) = (K \circ I)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{4.1}$$

In this context, m and n refer to the dimensions of the input image and $S$ is the final output at position $i$ and $j$. The application of the convolution operation is based on three important ideas: *sparse connectivity*, *parameter sharing*, and *equivalent representations*.

In a fully connected neural network, each output unit is connected directly to every input unit. In comparison to this, a convolution layer uses only sparse connections, since each output unit merely results from a subset of the input units. This leads to a lower amount of parameters, reducing memory requirements and improving the efficiency. This is due to the structural design of the kernel, whose dimension is smaller than the input image. Therefore, the smaller the kernel size, the sparser the interactions between input and output units. This results in a trade-off, because on the one hand the kernel must be large enough to identify local features. On the other hand, it negatively affects the performance if the size is chosen too large. By choosing a small kernel, the first layers in a CNN can only recognize simple structures and patterns in images (e.g. edges). The deeper the network is constructed, the more complex structures can be identified. That is because deeper layers indirectly interact with a larger portion of the input units. As figure 4.3 shows, the outputs of a convolution layer defines the input of the subsequent layer. The outputs of the layers can therefore be imagined as building blocks building on top of each other.

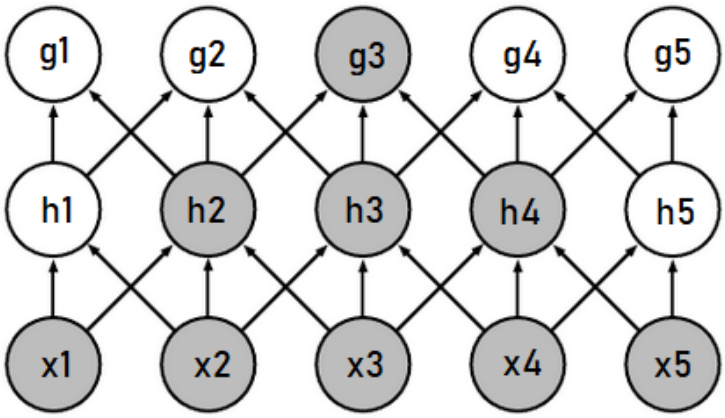


**Figure 4.3:** Visualization between the indirect dependencies between multiple layers in CNN [29] where $x_i$ denotes an element of the input vector $x$ and $h_i, g_i$ refer to the outputs of convolution operations in the first two layers

*Parameter sharing* is the next characteristic of a convolutional operation. The main idea behind parameter sharing is to learn one set of weights instead of separate sets for every location. The kernel matrix can be seen as a 2-dimensional window that slices through every part of the image, applying matrix multiplication. The reuse of the weights makes sense, because certain patterns in images often occur multiple times. This leads to filters that are optimized for the recognition of certain patterns (e.g. vertical edges). Sharing parameters has no positive effect on the runtime, because the number of operations remains the same. Nevertheless, the required storage capacity is reduced, because instead of $m \times n$, only $k \times k$ weights need to be stored, where k is the size of the kernel.

Another beneficial property of applying convolutional operations stems from the idea of sharing the same parameters and is referred to as *equivalence to translations*, which means that if the input changes, the output changes in the same way. With regard to images, a convolution creates 2-D feature maps that show where certain features appear in the input. If the position of an object in an image changes, it will move the same way in the output image. This phenomenon is related to the kernel, which slides through the input and ensures that a pattern is recognized regardless of its position. In case of multiple features that are learned with separate filters, a pooling unit can be even be invariant to other transformations such as rotations or distortions.

### 4.2.2. Pooling

The task of the pooling layer is to replace the output of the feature map at a certain position with a summary statistic of the nearby outputs. The most popular pooling functions being used in this context are called *max* and *average pooling*. The max

pooling function computes the maximum value that is in the filters receptive field, while the average function simply calculates the average. Figure 4.4 shows an example of how the summary metrics are extracted from the feature map.

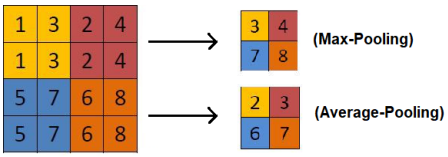


**Figure 4.4:** Example of max- and average-pooling operations [76]

One of the tasks of pooling is the dimension reduction of the feature map in order to eliminate less relevant information. On the other hand, the pooling allows the processing of images with variable size. Another advantage is the invariance to small translations. This means that by slightly translating an object in an image, the value of the pooled output will not change because the maximum value of that area remains the same.

## 4.3. Transfer Learning

Transfer learning refers to using a pre-trained model for another task with the same weights. Generally, two common approaches can be distinguished: First, the pre-trained model works as a feature extractor [54] by only using the model up to a specific layer. The extracted features can then be used as input for subsequent machine learning tasks. Second, the pre-trained model can be fine-tuned and tailored on a specific task [78] through adding further layers on top of it. As a result, the training process is more efficient because the weights have already been trained, preventing them from being randomly initialized. The main benefit of using transfer learning is that the pre-trained models are mostly based on an extensive database containing millions of data samples and therefore prevent the trained model to be biased towards properties of the training set.. Since the first layers in a CNN are optimized to extract low-level features such as edges, this can be exploited and applied to other problems.

## 4.4. Convolutional Autoencoder

A Convolutional Autoencoder (Convolutional AE) is a neural network optimized to copy its input data. The model consists of two components: the encoder $h = f(x)$ and the decoder $x' = g(h)$ function, both of which are implemented as a multi-layer CNN. This means that the Convolutional AE maps an input image $x$ to an output image $x'$. The output of the encoder function is a lower dimensional

latent representation of the original image. Out of this latent variable, the decoder function needs to reconstruct the original image. In order to force the model to learn correct parameters for decoding an encoded representation, the loss function needs to be minimized.

$$L\left(x, g\left(f\left(x\right)\right)\right) \tag{4.2}$$

This loss function penalizes $g(f(x))$ for being dissimilar from $x$. The choice of the loss function thus plays a decisive role in the application of Convolutional AEs. Three loss functions will be introduced in the next subsection to lay the foundation for the experimental part of this thesis. However, one important requirement that must be observed, is to design the architecture of a Convolutional AE in an undercompleted way. This refers to constrain $h$ to have a smaller dimension than the original input $x$. It forces the Convolutional AE to only extract most relevant features of the training data. Furthermore it prevents the model to be in danger of learning the identity function [66].

Once the model is trained, it is able to encode and reconstruct images that resemble the training data. In case of an input image that is dissimilar to the ones involved in training, reconstruction will fail. This leads to a high reconstruction error (see 4.2). The high input sensitivity of a Convolutional AE can be exploited to detect images that differ from the ones being used to train the model. That is why Convolutional AEs are very popular in the field of anomaly detection (e.g. [69], [40]). Transferred to the domain of fingerprint PAD, a Convolutional AE is only trained on bona fides. Later, the model can be used to detect unknown PAs by comparing the reconstruction error against a threshold.

### 4.4.1. Weighted Mean Squared Error

A Convolutional AE encodes the input and decodes it again so that the original input is reconstructed. This means that the loss function needs to be designed in a way that the reconstruction error is minimized. A common approach is to use the mean squared error (MSE) [5] which is defined as

$$L = \frac{1}{N}\sum_{j=1}^{N} mse^{j} \tag{4.3}$$

with

$$mse^{j} = \frac{1}{WHI} \cdot \sum_{w=1}^{W}\sum_{h=1}^{H}\sum_{i=1}^{I} e^{j}_{whi} \tag{4.4}$$

where $N, W, H, I$ denote the number of training samples, the width, height,

and number of input channels (e.g. three for RGB) of an input image x and $e^j_{whi} = (x^j_{whi} - x'^j_{whi})^2$ equals the pixel-wise squared error. Using the MSE as a loss function has the advantage that it is easy to understand and often preimplemented by many deep learning libraries such as Keras [42]. However, there is also a major drawback in case of random noise occurring in the data. Since the calculation of the MSE includes squaring the difference between every pixel of the input image, single outliers will have a huge impact on the reconstruction error. This will inevitably lead to an increased rate of bona fides erroneously classified as PAs. This lack of robustness against outliers is a well known challenge in the deep learning domain and is referred to as *robust estimation* [60]. The idea of increasing the robustness of an AE model for anomaly detection was picked up by [32] who introduced a weighted version of the MSE:

$$L = \frac{1}{N} \sum_{j=1}^{N} w^j \cdot mse^j \tag{4.5}$$

with $w^j$ defined as

$$w^j = \begin{cases} 1, & mse^j \leq C \\ 0, & mse^j > C \end{cases} . \tag{4.6}$$

where $C$ equals the $\alpha$-th quantile of $mse = [mse^1, \ldots, mse^N]$. The approach of Ishii et al. forsees to ignore training samples during the optimization process as soon as their measured MSE exceeds a defined threshold $C$. Translated to the problem of fingerprint PAD, that means that a certain percentage of bona fides will be ignored during the training phase. The authors state that their proposed loss function is useful to cope with unknown outliers within the training set since they will not distort the resulting model. Unknown outliers can occur, for example, if the data is not labelled. Therefore it will be difficult to split them from the inliers. However in case of this thesis, the training data is labelled and does not contain a single PA. Thus, the exclusion of bona fide samples exceeding $C$ could lead to an unwanted loss of information, since the ignored images still contain areas relevant for classification.

For that reason, the proposed loss function of Ishii et al. will be slightly adjusted within this work. The main idea is to integrate the weight factor such that it excludes image areas for which the target grey values can systematically not be reconstructed by the Convolutional AE. In other words, this means that the Convolutional AE is optimized to reconstruct the most meaningful areas of the images while ignoring random noise. The adjusted loss function is defined as follows.

$$L = \frac{1}{WHI} \cdot \sum_{j=1}^{N} \sum_{w=1}^{W} \sum_{h=1}^{H} \sum_{i=1}^{I} w_{whi}^{j} e_{whi}^{j} \tag{4.7}$$

with

$$w_{whi}^{j} = \begin{cases} 1, & e_{whi}^{j} \leq mse^{j} + C \cdot std^{j} \\ 0, & e_{whi}^{j} > mse^{j} + C \cdot std^{j} \end{cases} . \tag{4.8}$$

and

$$std^{j} = \sqrt{\frac{1}{WHI} \cdot \sum_{w=1}^{W} \sum_{h=1}^{H} \sum_{i=1}^{I} \left(e_{whi}^{j} - mse^{j}\right)^{2}} \tag{4.9}$$

Generally speaking, every pixel value will be compared with a threshold that is a linear combination of both mean and standard deviation of the squared error. Exceeding pixels will be ignored within the model optimization and during prediction. Thus, contrary to the MSE, it is assumed that this approach will prevent random noise from increasing the overall reconstruction error of bona fide samples. The remaining challenge however consists in finding the optimal constant value of $C$. By choosing a too low threshold the model might tend to over-generalize such that decisive patterns that are important for distinguishing between bona fides and PAs are not extracted anymore. On the other hand, if $C$ is too high, noisy data might be involved in both training and testing which leads to a less robust model and consequently to increased misclassification rates. This problem is related to the typical trade-off between bias and variance. To validate the assumptions made, section 7.2 compares the performance of a Convolutional AE model trained with the proposed weighted MSE against another one trained by using the usual MSE.

## 4.5. t-Distributed Stochastic Neighbor Embedding

In 2008, Van der Maat [48] presented **t-Distributed Stochastic Neighbor Embedding (t-SNE)** as a method to visualize high-dimensional data. T-SNE is a dimension reduction technique that converts a high-dimension data set $\mathcal{X} = x_1, x_2, \ldots, x_N$ into low-dimensional data $\mathcal{Y} = y_1, y_2, \ldots, y_N$. Usually, the high-dimensional data is reduced into the 2-dimensional space so that the samples can be visualized in a scatterplot. The main goal of reducing the dimension is to preserve as much of the relevant structure of the high-dimensional as possible in the low-dimensional space. Basically, t-SNE solves this problem by keeping the low-dimensional representations of very similar data points close together by using a nonlinear mapping algorithm. To this extend, t-SNE is able to capture local

structures of the high-dimensional data, as well as revealing global structures such as the presence of clusters.

T-SNE translates the euclidean distances between data points $x_i$ and $x_j$ in the high-dimensional space as a symmetric joint-probability distribution $\mathcal{P}$. Analogously, a joint-probability distribution $\mathcal{Q}$ is defined that describes the similarity of the data points $y_i$ and $y_j$ in the low-dimensional space. The main goal is to shape $\mathcal{Q}$, also *Embedding*, so that it behaves as similar as possible to $\mathcal{P}$. T-SNE addresses this problem by optimizing the positions of the low-dimensional data points by minimizing the cost function $KL$ given by the **Kullback-Leibler divergence** between $\mathcal{P}$ and $\mathcal{Q}$:

$$KL(\mathcal{P}||\mathcal{Q}) = \sum_{i=1}^{N}\sum_{j=1}^{N} p_{ij} \cdot \ln\left(\frac{p_{ij}}{q_{ij}}\right) \tag{4.10}$$

Given two data points from $x_i$ and $x_j$ from $\mathcal{X} = x_1, x_2, \ldots, x_N$, $p_{ij}$ models the probability that $x_i$ would choose $x_j$ as its neighbor. In this context, for each point a Gaussian kernel, $\mathcal{P}_\rangle$, is chosen. The variance of the Gaussian kernel $\sigma_i$ is dependent on the local density in the high-dimensional space. Then $p_{ij}$ is defined as follows:

$$p_{ij} = p_{ji} = \frac{p_{i|j} + p_{j|i}}{2N} \tag{4.11}$$

with

$$p_{j|i} = \frac{\exp\left(-\left(||x_i - x_j||^2\right) / \left(2\sigma_i^2\right)\right)}{\sum_{k\neq i}^{N} \exp\left(-\left(||x_i - x_k||^2\right) / \left(2\sigma_i^2\right)\right)} \tag{4.12}$$

where $p_{j|i}$ can be seen as the relative measure of similarity between $x_i$ and $x_j$, based on a local neighborhood centered around $x_i$. For the purpose of choosing $sigma_i$ for a given Gaussion kernel, the perplexity number $\theta$ is defined. This can be interpreted as a measure of the effective number of neighbors and be chosen individually. The value of $\sigma_i$ is chosen that for a fixed $\theta$ and each $i$ the following equation is satisfied:

$$\theta = 2^{-\sum_{j=1}^{N} p_{j|i} \cdot \log_2(p_{j|i})} \tag{4.13}$$

Instead of using a Gaussian distribution as in the case of $p_{j|i}$, the computation of the joint-probability distribution $\mathcal{Q}$ is based on a Student‘s t-Distribution. Given two lower-dimensional points $y_i$ and $y_j$, the similarity measure $q_{ij}$ is defined as follows:

$$q_{ij} = ((1 + ||y_i - y_j||^2) \cdot Z)^{-1} \tag{4.14}$$

where

$$Z = \sum_{k=1}^{N} \sum_{l \neq k}^{N} \left(1 + ||y_k - y_l||^2\right)^{-1} \tag{4.15}$$

The minimization of the $KL$-divergence from equation 4.10 is based on *Gradient Descent*, which gives an indication of the positional change of the low-dimensional points. The gradient of $KL$ is given by:

$$\frac{\delta KL}{\delta y_i} = 4 \left( \sum_{j \neq i}^{N} p_{ij} q_{ij} Z \left(y_i - y_j\right) - \sum_{j \neq i}^{N} q_{ij}^2 Z \left(y_i - y_j\right) \right) \tag{4.16}$$

The authors of [48] recommend to limit t-SNE on data sets not containing more than $10,000$ points as the computation and memory complexity increases quadratic with the number of data samples. In the course of this thesis, t-SNE has been applied to a maximum total number of 14,898 data samples without reaching system capacities.

## 4.6. One-Class Support Vector Machines

A One-Class Support Vector Machine (OC-SVM) defines a group of unsupervised learning algorithms that decide whether a new data point belongs to a class or not, i.e. being an outlier or not. In this context, the work of Schölkopf et al. [62] provides an algorithm that computes a binary function that captures regions in feature space where the probability density of the data lives. The decision function is defined as follows:

$$f(x) = sgn \left( \sum_{i=1}^{N} \alpha_i K(x, x_i) - \rho \right) \tag{4.17}$$

where $\alpha_i$ and $\rho$ are obtained by solving a minimization problem and characterize a hyperplane which has maximal distance from the origin in feature space, separating all data points from the origin (for further details see Schölkopf et al. [62]). As part of the decision function, a *radial basis function* kernel has been used, which is defined as:

$$K(x, x_i) = \exp \frac{||x - x_i||^2}{2\sigma^2} \tag{4.18}$$

where $\sigma$ automatically gets scaled within the scikit-learn machine learning library

[64], dependent on the variance of the input data. The RBF-Kernel value decreases with distance and ranges between zero and one, therefore it can be interpreted as a similarity measure between two data points [75]. Translated to equation 4.18, this means that the decision function $f(x)$ will be $+1$ for data points that are similar to the ones the OC-SVM was trained on. On the other hand, $f(x)$ outputs $-1$ for data points which are dissimilar to the training data, thus indicating an outlier.

# 5. Experimental Setup

This chapter provides a detailed overview of the experimental framework to prepare the reader for the next sections.

## 5.1. Dataset & Collection

In order to evaluate the suitability of LSCI and SWIR as a basis for unknown fingerprint PAD using Convolutional AEs, an extensive dataset is mandatory. Since most publicly accessible data sources for fingerprint images such as the LivDet dataset [55] do not include LSCI or SWIR data, an own capturing process has been initiated within the BATL project [7] using the capturing device described in section 3.1.1.

To make use of the observations of Steiner et al. [68], the SWIR images were taken within the following wavelengths: 1200 nm, 1300 nm, 1450 nm and 1550 nm. For each finger presentation, eight different images were captured, two per wavelength. The first one (SWIR image) with active SWIR illumination and the second (dark image) without any illumination. By subtracting the dark from the SWIR image, light sources not stemming from the illumination inside the capturing device can be removed.

For the LSCI sensor the laser module is used as the illumination source facing the finger slot. The scattering pattern appears when light from a coherent light source (i.e. laser) is reflected by a sufficiently rough surface. After the light gets reflected it interferes either constructively or destructively which leads to alternating dark and bright patterns. Since the laser light penetrates the skin to a certain depth, the speckle pattern can also detect temporal variations such as the blood flow in the finger [74]. To investigate this temporal effect, a video sequence of two seconds was recorded at a frame rate of 50 fps (i.e. 100 frames). Figure 5.1 shows sample images of a captured bona fide fingerprint using LSCI and SWIR sensors.

In several sessions a total of $19,598$ bona fide and $4,226$ PA samples were collected, including 43 unique PAIs. During the acquisition, the fingerprint images were labeled as either bona fide or PA in order to train binary classification algorithms. Additionally, the data has been cleaned by removing erroneous samples which were

caused through an improper placement of a subject's finger. Finally, the region of interest (ROI) has been extracted by cropping the image frames to a size of 100x300 pixels.

The rest of this section is dedicated to further inspect the PAIs involved to test the performance of an AE. Among others, *Ecoflex, Silicone, Gelatin*, and *Dragonskin* are the most represented materials used for fabrication. Since both structure and appearance within the PAIs varies a lot, they were assigned to specific classes. This is advantageous since the performance of an AE can be broken down to determine which type of PAI the model is most susceptible to. In this context, the following classes are defined:

- **Fake Fingers** refer to PAIs that reconstruct the whole finger
- **Overlay Opaque**.
- **Overlay Transparent**
- **Overlay Semi** refer to PAIs that can neither be classified as opaque nor transparent
- **Printout Transparent**
- **Printout Opaque**

A more detailed breakdown of the PAI species classes is given in appendix A. However, to provide an insight into the variety of different PAIs, figure 5.2 shows one randomly chosen example image of each class, both for LSCI and SWIR

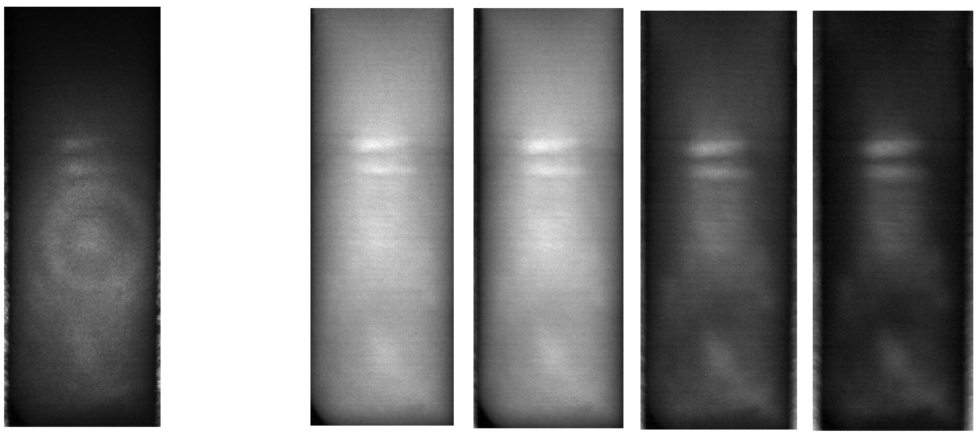

**Figure 5.1:** Examples LSCI and SWIR bona fide images - from left to right: LSCI, SWIR (1200nm, 1300nm, 1450nm, 1550nm)

data. Comparing the PA samples with figure 5.1 which shows bona fide examples, they can be clearly discriminated visually. This is an important characteristic as Convolutional Autoencoders are not able to detect PAs that look exactly the same as bona fides.

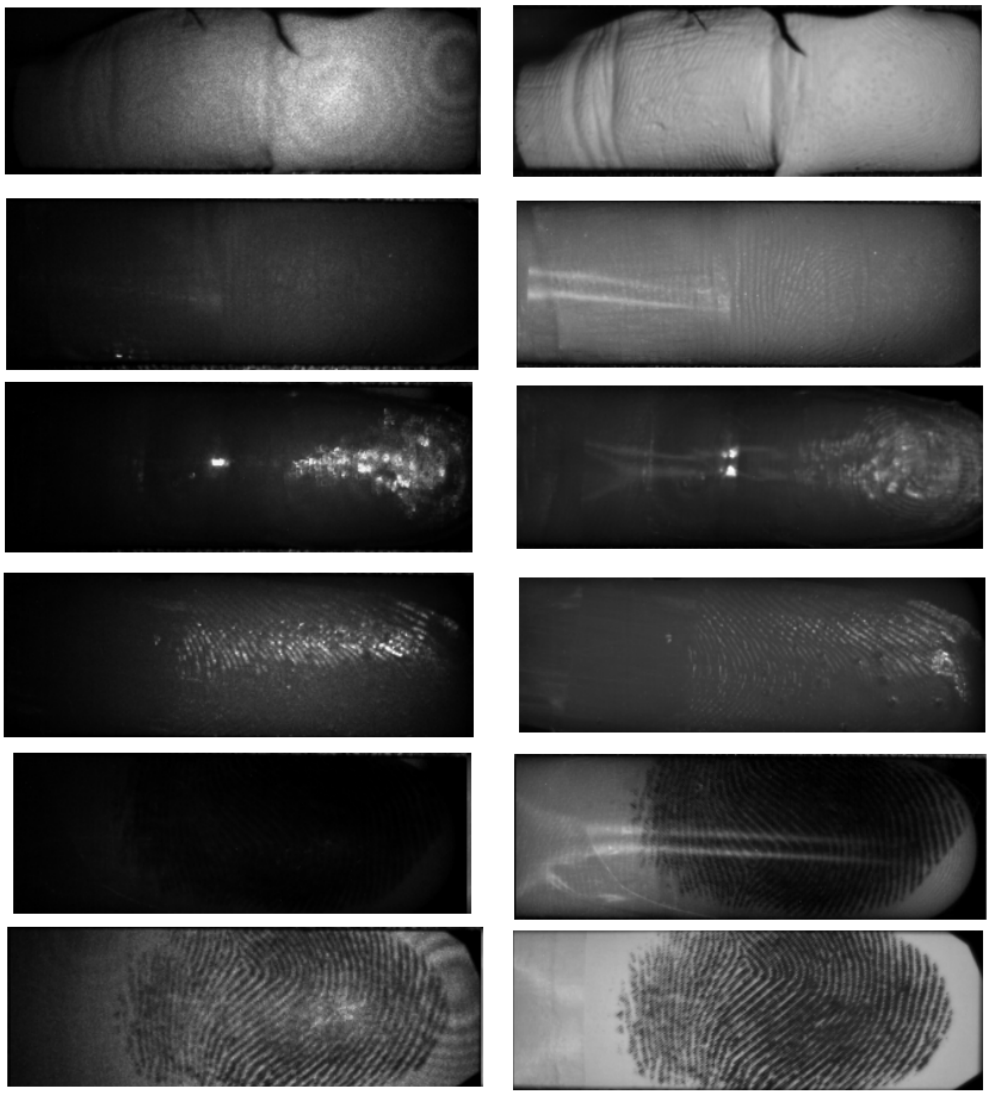

**Figure 5.2: Left:** LSCI image, **Right:** SWIR image captured with 1200nm, **Top to bottom:** Overlay Opaque (overlay ecoflex fleshtone), Fake Finger (dragonskin), Overlay Transparent (overlay knox gelatin), Overlay Semi (overlay wood glue), Printout Transparent, Printout Opaque (Printout Paper)

Since the number of samples belonging to the predefined PAI species classes is not equally distributed, figure 5.3 depicts the proportions of each class to the total amount of all PAs. Apparently, both opaqued overlays as well as fake fingers are

over-represented, containing approximately 80% of the PA samples. Contrarily, the printout classes are represented with only 2.6%. It is important to take this into consideration as even models achieving highly competitive performance numbers could potentially misclassify all samples stemming from an underrepresented class and therefore lead to a false security perception regarding the robustness of the biometric capture device. Thus, also the detection performance within the PAI species classes will be examined during the course of this thesis.

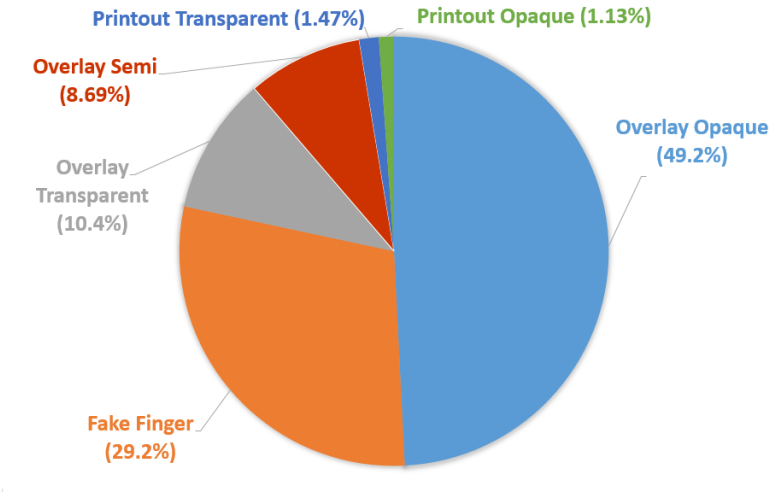


**Figure 5.3:** Visualization of the PAI species classes representations within the dataset

## 5.2. Evaluation

### 5.2.1. Metrics

In order to facilitate the comparison of the presented results, the reported evaluation metrics are in accordance with the *ISO/IEC - 30107-3* standard [36] and defined as follows:

- **Detection-Equal Error Rate (D-EER)** reports the misclassification rate when APCER equals BPCER
- **APCER20** reports the APCER for a fixed BPCER of 5%
- **BPCER20** reports the BPCER for a fixed APCER of 5%
- **APCER100** reports the APCER for a fixed BPCER of 1%
- **BPCER100** reports the BPCER for a fixed APCER of 1%

Generally, a low BPCER (e.g. as indicated with the fixed BPCER of 1%) represents a convenient PAD system because less bona fide fingerprints will accidentally be classified as attack presentations. On the other hand, the lower the APCER, the more PAs will be detected and therefore increase the security of the overall biometric system. The decision on which metric is most important therefore depends on the final application. To further facilitate the performance analysis between multiple models, *Detection Error Tradeoff (DET)* curves provide a graphical presentation of the trade-off error rates [50]. In case of fingerprint PAD, both APCER and BPCER are plotted against each other for multiple thresholds. In addition to the above listed metrics, the partially measured area under the DET-Curve (pAUC) has also been taken into consideration as part of the experimental evaluation. Since very high error rates are not of particular interest, the pAUC is calculated for both APCERs and BPCERs smaller than 20%, as depicted in figure 5.4.

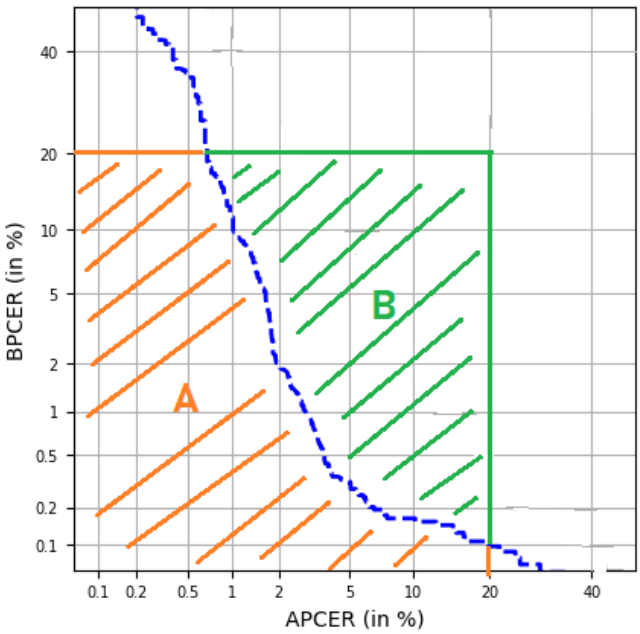


**Figure 5.4:** Graphical illustration of the area beneath (orange) and above (green) an example DET-curve (blue) with an according pAUC computed as $\frac{A}{A+B}$

### 5.2.2. K-Fold Cross Validation

Cross-Validation (CV) is a technique for evaluating the performance of a machine learning model on limited data sets. The motivation behind CV is to find out how

well the trained model predicts the classes of new and unseen samples that were not involved during the training process. On the one hand, it is desirable to include all available data samples to train the model. On the other hand, this means that the only possibility to validate the model is to use data that has already been seen by the model which often leads to overly optimistic results ([37], section 5.1). Generally speaking, it is mandatory to use different data partitions for the training and testing process. This is where the CV comes in: it involves partitioning the data into complementary subsets which are randomly chosen. The separation of the following three subsets has proven to be effective [8]:

- **Training subset.** The data samples used to train the model.
- **Validation subset.** The data samples used to provide an unbiased evaluation of the model fitted on the training subset while tuning the parameters of the classifier. The evaluation gets more biased the more information from the validation subset gets incorporated into the training process.
- **Test subset.** The data samples used to evaluate the performance of the final model. Neither model- nor hyperparameters get trained based on the test samples in order to provide an unbiased evaluation.

Splitting data into training, validation, and test subsets provides an intuitive technique to detect overfitting and evaluate the model in an unbiased manner. However, since not all of the data samples are included in the training process, this may result in important variability being lost. To anticipate this problem and to ensure that different samples are used for training and testing, a *k-fold Cross-Validation* will be performed throughout the experiments of this thesis. The idea of k-fold CV consists of randomly partitioning the dataset into k equal sized subsets ([22], section 7.10.1). The next step foresees to leave out the first of the k subsets as evaluation and the remaining $k-1$ subsets as training data. This process will then be repeated k times so that each subset has been used only once for validation. To obtain a single performance estimate, the k results can be averaged.

Since the computing time increases with increasing k, a value of $k = 3$ was chosen for the experiments. Choosing a low value for k still leads to a stable estimate of the true misclassification error as long as not too many classes are involved [44]. Figure 5.5 depicts the number of samples contained in training, test and validation on three different folds. The 30-20-50 split setting has been chosen after a comparison of alternative configurations (see appendix B). Since the fingerprints of some test subjects were captured multiple times, the data splits are based on the subject ID. This is important because data samples from the same subject, which are included in both training and test set, might lead to overly optimistic classification results. Since the Convolutional AEs are trained on bona fide samples

solely, the data split is applied to the total amount of bona fide fingerprint images. In addition to the total amount of test samples reported in figure 5.5, all of the 4,226 PA samples were added to the test set afterwards.

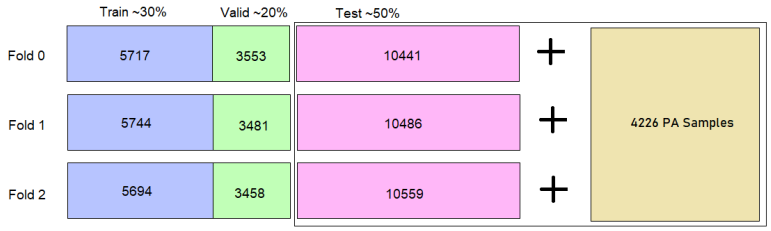


**Figure 5.5:** Graphical representations of the three folds used for 3-fold Cross Validation. After the data partitions were created, all PAs were added to the test set.

## 5.3. Implementational Details

### 5.3.1. Libraries

The presented AE architectures have been implemented with Keras, that is a python based deep learning library that facilitates the definition, training and evaluation of various deep learning model types [21]. Initially it has been developed for researchers to enable fast experimentation [43]. It provides a high-level API containing modular building blocks to be able to construct customized deep learning models. However, Keras does not include low-level operations such as tensor manipulations or differentiations [1]. For this reason, *Tensorflow* [71] is used as a low-level framework which provides all the necessary tensor operations, offering the perfect foundation for Keras. Furthermore, *Scikit-Learn* [63] is used as a python based machine learning library. It contains a pre-implemented function *OneClassSVM()* for both training and testing OC-SVMs, analogous to Schölkopf et al. [62]. All of the package versions used within this work are listed in appendix C.

### 5.3.2. Dynamic AE Model Creation

In order to be able to build a tailored AE model, it is essential to test different parameter settings. In this context, it is recommended to keep the model construction part as dynamic as possible. During implementation, the variable design of the number of layers turned out to be one of the biggest challenges. The problem and its solution is illustrated in figure 5.6. The arrow diagrams are simplified representations of two AE models, with the left version consisting of one downsampling and the right version consisting of two downsampling steps. The left version will not cause any problems because both width and height of the original image are even numbers. By adding an additional downsampling

operation the size of the latent space representation needs to be either rounded up or down. Since the reconstruction part involves two upsampling operations, the size is simply doubled twice to restore the original image. However, this would result in a reconstructed image with a shape of $32x32x1$ that differs from the original one. This leads to error occurrences in the calculation of the pixelwise mean squared error. The workaround solution includes a layerwise shape comparison. In case of a mismatch, the rightmost column and the last row are truncated to compensate the discrepancy.

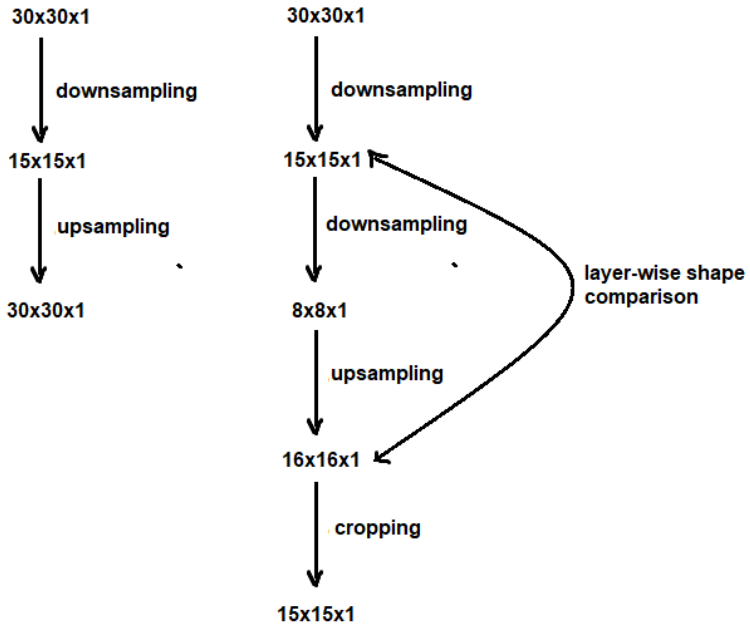


**Figure 5.6:** Illustration of the problem that arises with uneven shapes. **Left:** AE with a single downsampling operation, **Right:** AE with two downsampling oeprations

# 6. Proposed PAD methods

This section gives an overview of the experiments which are presented in section 7. Specifically, the first part introduces the reader to the baseline AE architectures that have been implemented within this thesis. Building on this, the second and third part discuss additional test settings. However, since the depictions within this section only serve as a first overview, various details, such as the filter sizes of the convolution operations, are not specified. A further description of the baseline architectures follows in section 7.1.1.

## 6.1. First Part: Baseline Models

The first part of the experiments introduces three baseline model architectures whose results are presented and benchmarked against each other to determine which one is best suited for fingerprint PAD on both LSCI and SWIR data. The structure of the architectures is depicted in figure 6.1. All of the baseline models are trained using MSE as a loss function and a single input channel, i.e. a single image. For the LSCI data, the 50th image out of a sequence of 100 images per captured finger has been chosen. The SWIR models are trained on a single image with a wavelength of 1550 nm. Looking at the architecture of the Dense-AE, a new operation (Flatten) is involved. Since a Fully-Connected NN demands for a vector input, the multi-dimensional output of the max pooling operation needs to be reshaped. Therefore, the *flatten* operation reshapes a multi-dimensional tensor to have a single dimension with the number of entries being equal to the total number of elements contained in the multi-dimensional tensor.

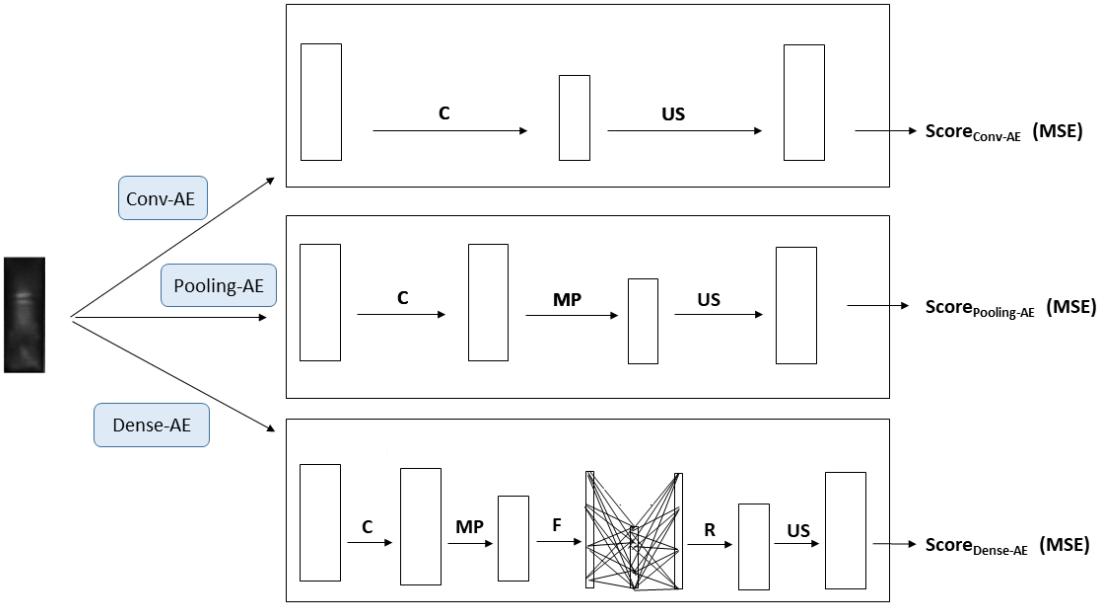


**Figure 6.1:** The first part of the experiments includes three baseline architectures, defined as *Conv-*, *Pooling-*, and *Dense-AE*. Operations: C = Convolution, US = Upsampling, MP = Max Pooling, F = Flatten, R = Reshape

## 6.2. Second Part: Weighted MSE

Section 4.4 introduced a weighted MSE as an alternative loss function for Convolutional AEs. Therefore, the second part of the experiments examines the impact of replacing the MSE loss function with the proposed weighted MSE, as depicted

in figure 6.2. To accomplish this, the architecture of the best performing baseline model will be retrained with exactly the same parameters but with the loss function replaced by the weighted MSE. Additionally, since the weighted MSE entails a further hyperparameter $C$, the effects of different parameter choices are also examined.

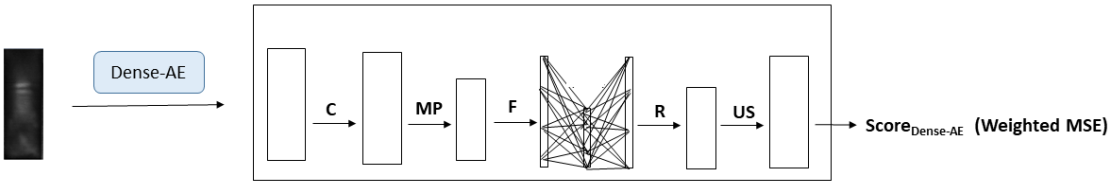


**Figure 6.2:** The second part of the experiments evaluates the effect of using a Dense-AE with a *Weighted MSE* instead of the usual variant

## 6.3. Third Part: Multi-dimensional Inputs

Finally, the third part adopts the same model configurations as previously but with multi-dimensional input channels. The addressed question here is whether the supplementary dimensions contain additional information that are of value for the distinction between bona fides and PAs. In case of LSCI, a 3-dimensional input tensor is used, composed of the first, middle, and last image that were extracted from the sequence of 100 images. Since the acquisition of the SWIR images included four different wavelengths, all of them were concatenated to a 4-dimensional input tensor to exploit the whole spectrum of information, as shown in figure 6.3.

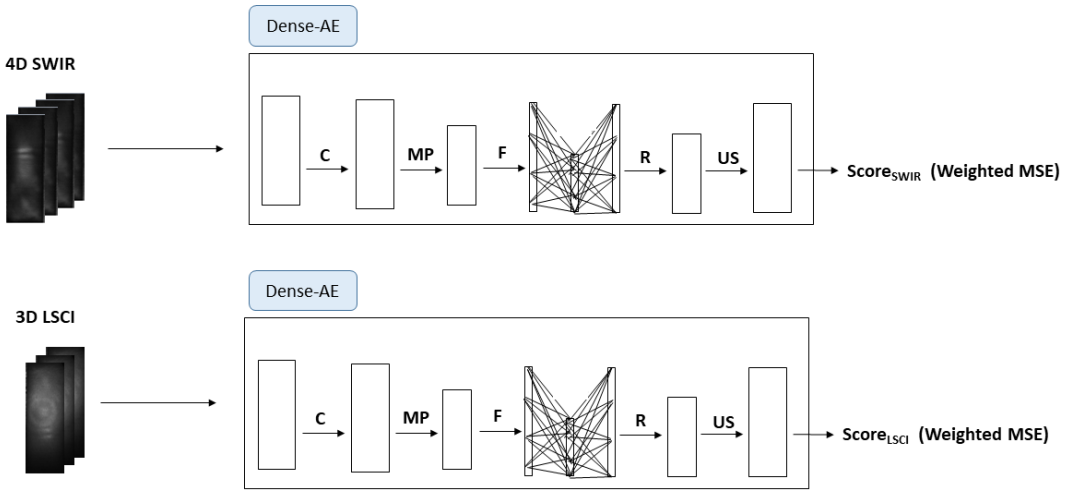


**Figure 6.3:** The third test setting includes multi-dimensional inputs for a Dense-AE on both LSCI (3D) and SWIR (4D) data.

# 7. Experimental Results

The following subsections are structured according to section 6 and present the results of all experiments performed during the course of this thesis

## 7.1. Baseline Model Architectures

### 7.1.1. Model Descriptions

This subsection introduces the three baseline AE model architectures which were implemented and tested as part of the thesis. The basic structure corresponds to the one described in section 4.4. It consists of two different parts: the encoding and decoding stage. Figure 7.1 allows to compare the layers contained in each model. To make it easier to refer to the three architecture types, they are defined as **Conv-AE**, **Pooling-AE**, and **Dense-AE** (left to right in figure 7.1). The names refer to the type of layers which were successively added to the architecture. So the Conv-AE includes only convolutions, while the Pooling-AE additionally consists of max pooling operations. Finally, the Dense-AE model also comprises dense layers, which correspond to the node structure in an ordinary Fully-Connected NN introduced in section 4.1.

First, a more detailed comparison between the structure of the Conv- and Pooling-AE model is been discussed: The only difference between them is how dimension reduction works. The idea behind the first model structure is to halve the dimension of the input tensor by using convolution operations with a striding number of two, which refers to the number of pixels a convolutional filter moves. Within the Pooling-AE, the convolution operation uses a stride value of one so that the dimension is preserved. The reduction of the dimension takes place in the second layer, the maximum pooling layer, which is described in section 4.2.2. By choosing a kernel size of two, the dimension is cut in half. The decoding stages of both architectures are identical and consist of an upsampling and convolutional operation. The upsampling layer simply repeats the rows and columns such that the dimension is doubled and again matches the dimension of the original input tensor. The consistency of the input and output dimension is mandatory because of the pixel-based mean squared error calculation involved in the optimization process.

Next, the architectural differences between the Pooling- and Dense-AE model are analyzed. The first two layers are identical and reduce the dimension by applying a maximum pooling operation. Unlike the Pooling-AE, the Dense-AE model uses a Fully-Connected NN between the convolutional encoding and decoding stage. Since Fully-Connected NNs require an one-dimensional tensor input, it is mandatory to flatten the multi-channel tensor output of the first maximum pooling layer. The

structure of the Fully-Connected NN complies with the basic structure of an AE. It consists of an encoding and decoding phase, whereby the dimension is reduced in the encoding and then rebuilt in the decoding part. Afterwards, both model architectures use the same technique to reconstruct the original input image.

The motivation to develop and test the performance of these models stems from related works that either use the Conv- or Dense-AE model structure. Specifically, the inspiration is grounded on scientific findings of Springenberg et al. [67] who claim that the max pooling operation can simply be replaced by a convolutional layer with an increased stride without significant loss in accuracy. This view is contrary to that of Goodfellow et al. [29] who state that the max pooling operation leads to an invariance of translations in smaller regions. The experimental results show which architecture is better suited for the task of fingerprint PAD using LSCI and SWIR data. Finally, the construction of the Dense-AE can be seen as an extension of the Pooling-AE. As mentioned by Ke et al. [40] the task of the Fully-Connected NN is to further process the feature maps of the previous layers by combining them to find interdependent patterns. The following subsections present the experimental results of all baseline model architectures.

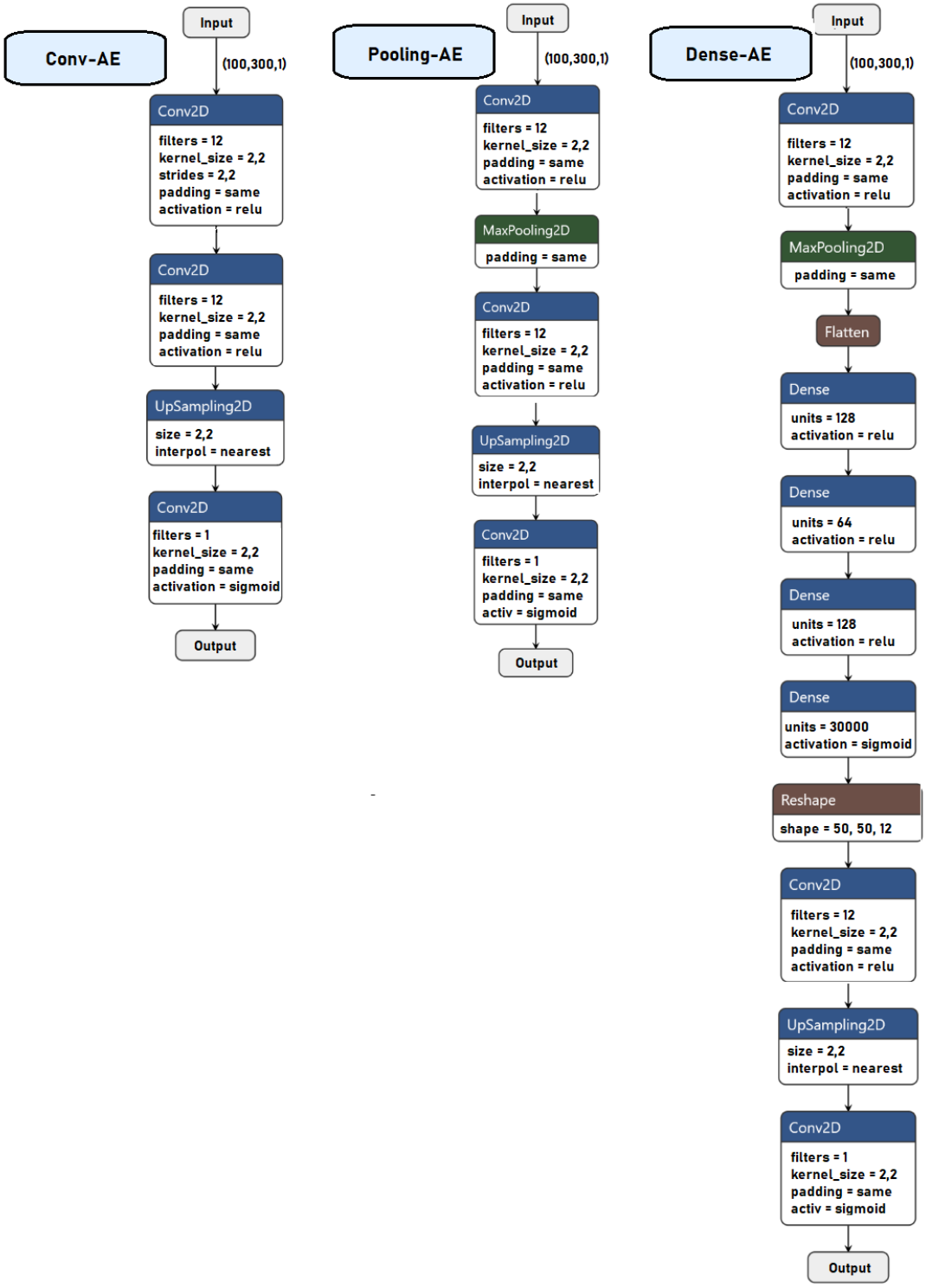


**Figure 7.1: Left:** Conv-AE including only convolutional operations, **Middle:** Pooling-AE using both convolutional and max pooling operations, **Right:** Dense-AE including convolutional and max pooling operations in addition to a Fully-Connected NN

### 7.1.2. Conv-AE Model

First, the results of the Conv-AE are presented. Table 7.1 lists the performance measures introduced in section 5.2.1 for each partition used as part of the Cross-Validation. The bold numbers represent the mean value of all three partitions. In addition to the performance measurements of the LSCI and SWIR data set, the results of a score fusion is given, using a simple weighted average. The question of how the scores are weighted is addressed in figure 7.2, which depicts multiple DET-curves each of which representing a different weight setting. The comparison of several DET curves can be interpreted subjectively, especially when the curves intersect. In this case, it often depends on whether the model should rather focus on the safety or the convenience of the biometric system. However within this thesis, the decision of which model performs best relies on the *partially measured area under curve (pAUC)* which has been introduced in section 5.2.1. This leads to a more general evaluation of the models, as they are not specifically meant to be optimized for safety or convenience purposes.

The first question to discuss is which sensor technology better suits to distinguish between bona fide and PA samples. When comparing the average measurements in

| Metric | Partition | LSCI (%) | SWIR (%) | Fused (%) |
|---|---|---|---|---|
| D-EER | 0 | 20.71 | 14.46 | 14.46 |
| | 1 | 20.93 | 12.27 | 12.27 |
| | 2 | 21.11 | 10.82 | 10.82 |
| | **Avg** | **20.92** | **12.52** | **12.52** |
| BPCER20 | 0 | 99.97 | 47.64 | 47.64 |
| | 1 | 99.99 | 52.20 | 52.20 |
| | 2 | 99.99 | 19.32 | 19.32 |
| | **Avg** | **99.98** | **39.72** | **39.72** |
| APCER20 | 0 | 28.38 | 19.62 | 19.62 |
| | 1 | 28.74 | 15.30 | 15.30 |
| | 2 | 28.65 | 18.88 | 18.88 |
| | **Avg** | **28.56** | **17.93** | **17.93** |
| BPCER100 | 0 | 100.00 | 76.34 | 76.34 |
| | 1 | 100.00 | 83.00 | 83.00 |
| | 2 | 100.00 | 67.76 | 67.76 |
| | **Avg** | **100.00** | **75.70** | **75.70** |
| APCER100 | 0 | 45.32 | 24.26 | 24.26 |
| | 1 | 45.22 | 21.20 | 21.20 |
| | 2 | 44.48 | 28.28 | 28.28 |
| | **Avg** | **45.01** | **24.58** | **24.58** |

**Table 7.1:** Cross Validation results of a Conv-AE on LSCI and SWIR data. Additionally, a score fusion is given whose weights have been chosen according to table 7.2

table 7.1, it becomes clear that the SWIR model is the superior one. All performance indicators have a lower value, which leads to the conclusion that the SWIR data set is more suitable as a basis for the Conv-AE. Another question to be addressed is whether the LSCI and SWIR data complement each other. As mentioned before, the optimal weights have to be found in order to fuse the two scores of the models. Figure 7.2 shows seven different DET-curves, each representing different score weights. Only the first fold will be taken into account to find the optimal weight setting since the measurements across the three partitions are similar.

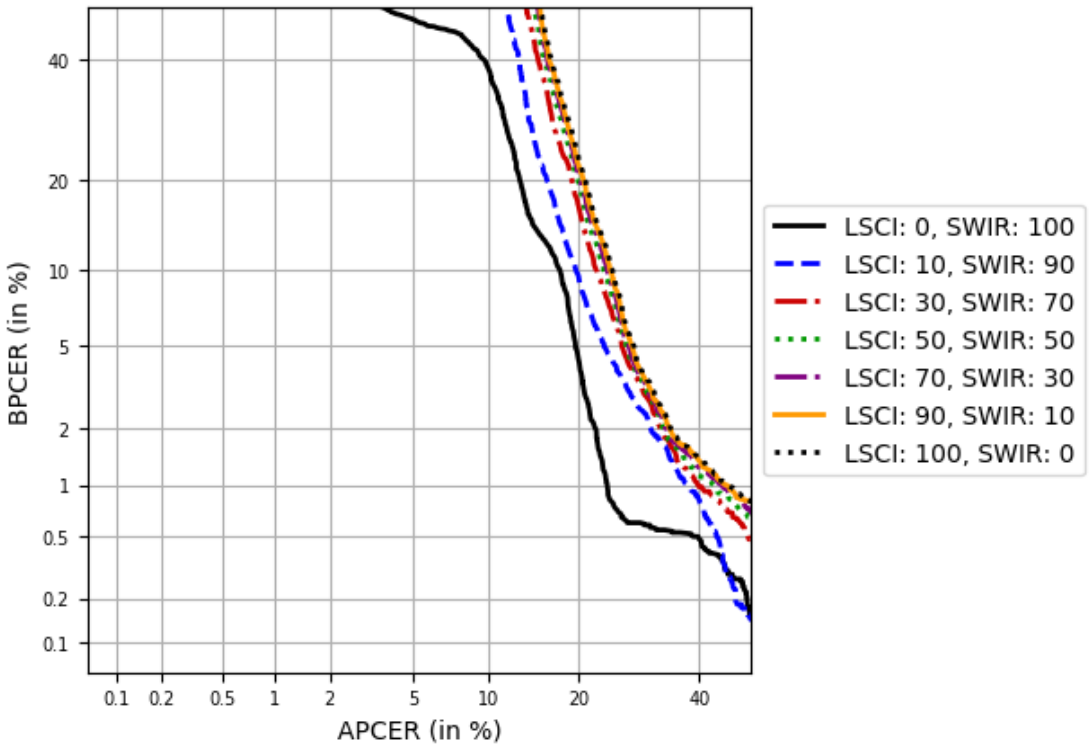


**Figure 7.2:** DET-Curves after fusing the scores of Conv-AE Models, which are trained on both LSCI and SWIR data

Figure 7.2 shows that the straight black curve outperforms all other weighted fusions. Therefore, the best setting is a weight of zero for the the scores of the LSCI model. Furthermore, table 7.2 presents the respective pAUC values and indicates that an increasing weight for the LSCI model, the pAUC value also increases. Hence, the overall performance decreases and we can conclude that the LSCI and SWIR models are not complementary, as the best results were obtained using the SWIR model alone.

| LSCI weight (%) | SWIR weight (%) | pAUC (%) |
|---|---|---|
| 0.0 | 100.0 | **84.94** |
| 10.0 | 90.0 | 94.02 |
| 30.0 | 70.0 | 99.57 |
| 50.0 | 50.0 | 99.94 |
| 70.0 | 30.0 | 100.00 |
| 90.0 | 10.0 | 100.000 |
| 100.0 | 0.0 | 100.00 |

**Table 7.2:** pAUC values measured in a range between 0-20% of the Conv-AE for multiple score weights

### 7.1.3. Pooling-AE

This subsection is structured exactly like the previous one: First, table 7.3 lists the performance metrics for each of the three data folds. Next, table 7.4 contains the pAUC values of the DET-curves shown in figure 7.3, which are used to determine the optimal score weights of the score fusion between LSCI and SWIR models. A comparison of the averaged performance metrics indicates once again that the SWIR model performs better than its counterpart. The results are therefore similar to the previous evaluation of the Conv-AE model. Looking at the performance of the fused model, it indicates that no complementary effect could be measured since the numbers are identical to the ones from the single SWIR model. Supplementary, figure 7.3 depicts the DET curves corresponding to the different weight settings.

| Metric | Partition | LSCI (%) | SWIR (%) | Fused (%) |
|---|---|---|---|---|
| D-EER | 0 | 20.10 | 11.01 | 11.01 |
| | 1 | 19.17 | 10.78 | 10.78 |
| | 2 | 19.54 | 10.66 | 10.66 |
| | **Avg** | **19.60** | **10.82** | **10.82** |
| BPCER20 | 0 | 99.99 | 40.32 | 40.32 |
| | 1 | 97.38 | 42.66 | 42.66 |
| | 2 | 96.90 | 22.48 | 22.48 |
| | **Avg** | **98.09** | **35.15** | **35.15** |
| APCER20 | 0 | 27.22 | 14.05 | 14.05 |
| | 1 | 28.16 | 13.55 | 13.55 |
| | 2 | 28.37 | 16.96 | 16.96 |
| | **Avg** | **27.92** | **14.85** | **14.85** |
| BPCER100 | 0 | 100.00 | 67.43 | 67.43 |
| | 1 | 100.00 | 69.36 | 69.36 |
| | 2 | 100.00 | 54.49 | 54.49 |
| | **Avg** | **100.00** | **63.76** | **63.76** |
| APCER100 | 0 | 42.86 | 22.92 | 22.92 |
| | 1 | 51.03 | 21.22 | 21.22 |
| | 2 | 46.92 | 26.02 | 26.02 |
| | **Avg** | **46.94** | **23.39** | **23.39** |

**Table 7.3:** Cross Validation results of a Pooling-AE on LSCI and SWIR data. Additionally, a score fusion is given whose weights have been chosen according to table 7.4

The straight black curve, similar to the last observations, stands out clearly from the others. This shows once again that the inclusion of the LSCI model leads to worse results. This observation can be confirmed by a pAUC value of 61.44% according to Table 7.4. It also shows that the higher the LSCI model is weighted, the higher the pAUC values, indicating a gradual deterioration in performance. Summarizing the results of the Pooling-AE model, it should be mentioned that the models are still far from providing a practical use for a biometric system. An D-EER of about 10% means that every tenth PA as well as every tenth bona fide sample has been misclassified. This can neither be considered particularly safe nor particularly convenient. In the next subsection, therefore, the results of the third type of AE are presented.

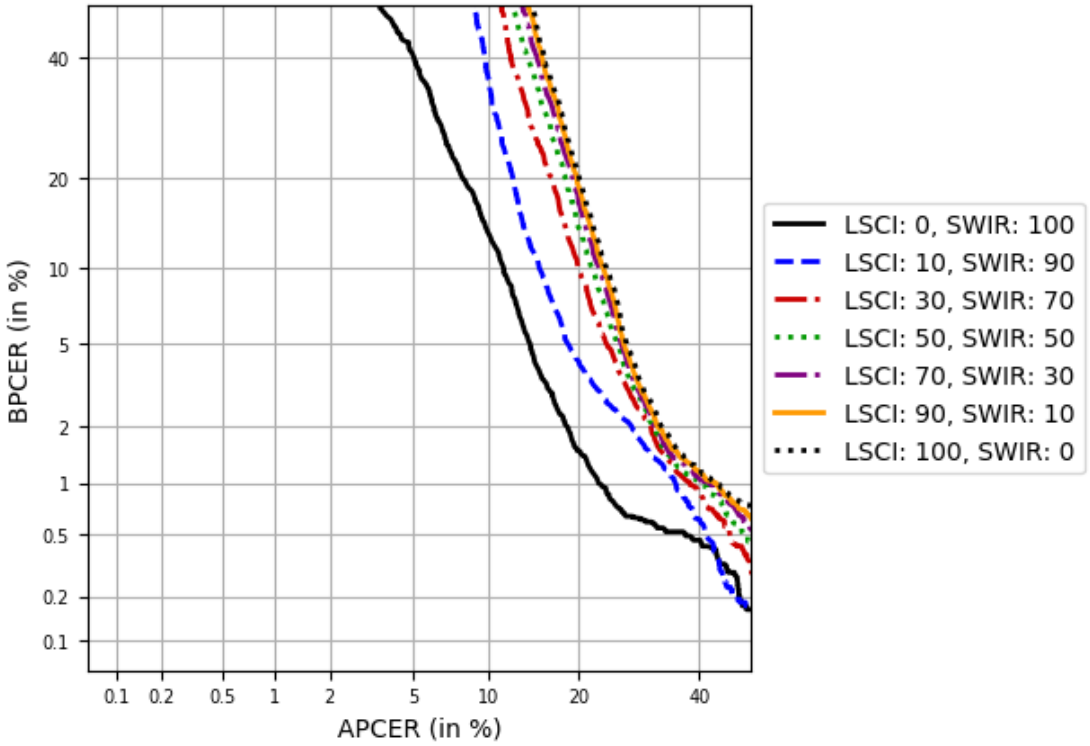


**Figure 7.3:** DET-Curves after fusing the scores of Pooling-AE Models, which are trained on both LSCI and SWIR data

| LSCI weight (%) | SWIR weight (%) | pAUC (%) |
|---|---|---|
| 0.0 | 100.0 | **61.44** |
| 10.0 | 90.0 | 79.15 |
| 30.0 | 70.0 | 95.20 |
| 50.0 | 50.0 | 98.68 |
| 70.0 | 30.0 | 99.72 |
| 90.0 | 10.0 | 99.93 |
| 100.0 | 0.0 | 100.00 |

**Table 7.4:** pAUC values measured in a range between 0-20% of the Pooling-AE for multiple score weights

### 7.1.4. Dense-AE

The evaluation of the results of the Dense-AE model is again structured in the same way as the previous subsections. Table 7.5 shows the performance metrics across all three data folds for the LSCI, SWIR, and fused models. In the last two subsections, the models trained on the SWIR dataset have outperformed the LSCI models without exceptions. This time, however, it seems to be exactly the opposite.

Except for the APCER100, all of the averaged measurements indicate that the LSCI model performs slightly better. This is a surprising result which proves that the choice of the model architecture has a significant impact on the final result. Additionally, looking at the fused scores, it can be observed that the combination of LSCI and SWIR models leads to a significant improvement. Comparing the average D-EER of the LSCI models with the ones measured by fusing the scores, they differ by over two percentage points which corresponds to percentage change of minus 35.5%.

| Metric | Partition | LSCI (%) | SWIR (%) | Fused (%) |
|---|---|---|---|---|
| D-EER | 0 | 6.59 | 6.01 | 4.01 |
| | 1 | 7.01 | 6.68 | 4.01 |
| | 2 | 6.67 | 8.46 | 5.05 |
| | **Avg** | **6.76** | **6.85** | **4.36** |
| BPCER20 | 0 | 8.56 | 9.29 | 3.20 |
| | 1 | 10.66 | 12.23 | 2.63 |
| | 2 | 11.83 | 26.88 | 5.33 |
| | **Avg** | **10.35** | **16.13** | **3.72** |
| APCER20 | 0 | 8.75 | 6.55 | 3.55 |
| | 1 | 9.01 | 7.58 | 3.50 |
| | 2 | 7.86 | 11.64 | 5.05 |
| | **Avg** | **8.54** | **8.59** | **4.03** |
| BPCER100 | 0 | 31.64 | 46.01 | 18.86 |
| | 1 | 29.47 | 46.45 | 27.43 |
| | 2 | 47.73 | 61.66 | 49.07 |
| | **Avg** | **36.28** | **51.37** | **31.79** |
| APCER100 | 0 | 33.97 | 13.14 | 20.90 |
| | 1 | 26.85 | 14.31 | 15.81 |
| | 2 | 29.41 | 20.12 | 20.51 |
| | **Avg** | **30.08** | **15.86** | **19.07** |

**Table 7.5:** Cross Validation results of a Dense-AE on LSCI and SWIR data. Additionally, a score fusion is given whose weights have been chosen according to table 7.6

The next step consists of analyzing the performance of the fused models using the DET-curves in Figure 7.4. Compared to the last two subsections, it is more difficult to see which DET curve performs best, since most curves are intersecting. Using the pAUC values according to table 7.5 as a decision criterion, the setting corresponding to the blue curve stands out with the lowest value. In this scenario, the LSCI scores are weighted with 10% while the SWIR scores are weighted with 90%. However, a closer look at the lines reveals that the black and blue curves intersect. This phenomenon can be confirmed by looking at Table 7.5. While the SWIR model has a APCER100 value of 15.86%, the fused model amounts

to a higher rate of 19.07%. This observations behaves the same across all three partitions. It can therefore be assumed that in an application that relies on the security of the biometric system, it is more valuable to stick to the SWIR model. Finally, after the results of the three baseline architectures have been presented, the next subsection deals with the comparison between them to decide which one is best suited for UPAD.

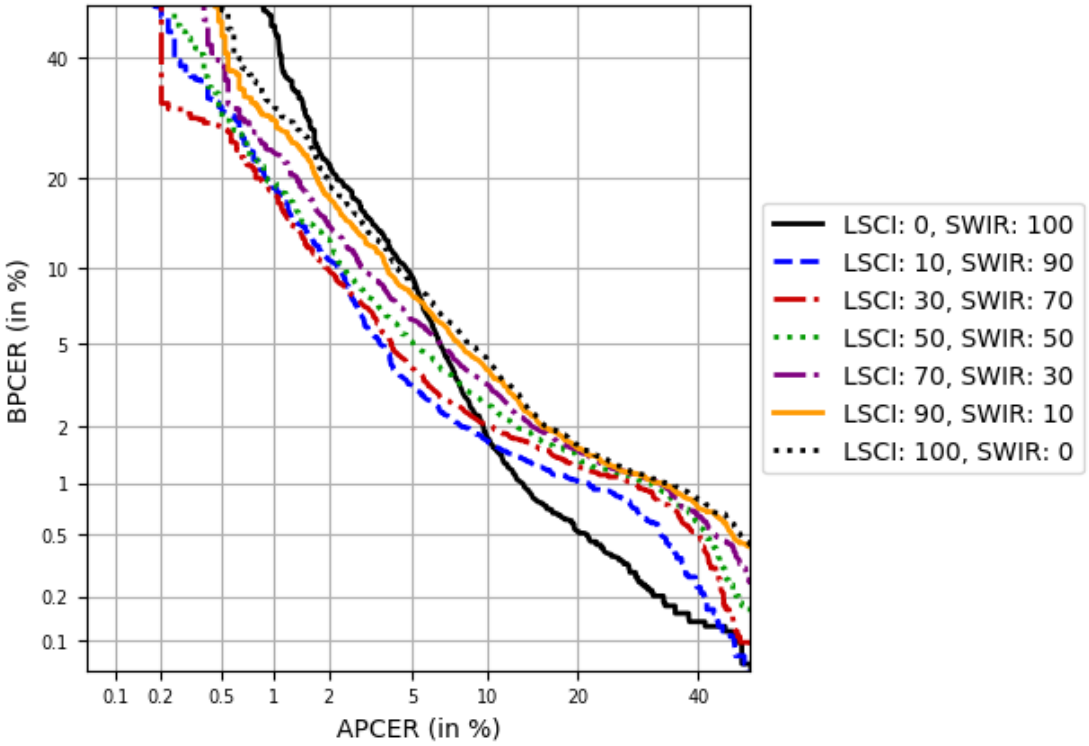


**Figure 7.4:** DET-Curves after fusing the scores of Dense-AE Models, which are trained on both LSCI and SWIR data

| LSCI weight (%) | SWIR weight (%) | pAUC (%) |
|---|---|---|
| 0.0 | 100.0 | 28.16 |
| 10.0 | 90.0 | **18.79** |
| 30.0 | 70.0 | 20.18 |
| 50.0 | 50.0 | 23.51 |
| 70.0 | 30.0 | 27.24 |
| 90.0 | 10.0 | 31.27 |
| 100.0 | 0.0 | 33.46 |

**Table 7.6:** pAUC values measured in a range between 0-20% of the Dense-AE for multiple score weights

### 7.1.5. Cross-Model Comparison

The last subsections were meant to see how the different AE architectures perform on both LSCI and SWIR data. However, the question of which architecture generally outperforms the others remains still open. To address this, figure 7.5 shows three different DET-curves each of which representing the best setting of the three model architectures. The decision of which model to choose for the comparison is based on the pAUC values given in table 7.2,7.4 and 7.6.

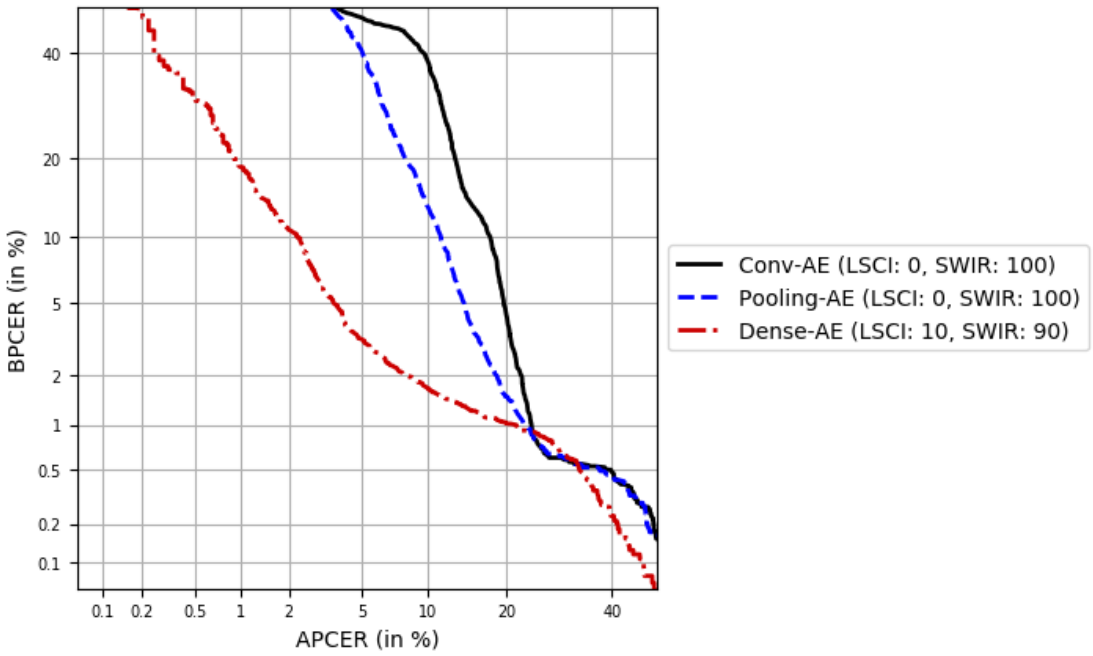


**Figure 7.5:** DET-Curves for comparing the fused performance between Conv-, Pooling-, and Dense-AE. Score weights have been chosen according to pAUC (see tables 7.2, 7.4, 7.6

At first sight, it is clear to see that the red curve, belonging to the Dense-AE, clearly stands out. Accordingly, the pAUC with a value of 18.79% is also significantly lower compared to the other values of 84.94% (Conv-AE) and 61.44% (Pooling-AE), respectively. It can therefore be concluded that the Dense-AE architecture is the most appropriate to distinguish between PAs and bona fides. This also confirms the statement of Ke et al. [40] who emphasize the importance of using a Fully-Connected NN within the autoencoder architecture. Their main argument is that after reducing the dimension with either convolutional or max pooling operations, the resulting local features still need to be combined with each other. So without the use of a Fully-Connected NN, the architecture is simply not able to capture the dependencies between the extracted local features, resulting in

a loss of information.

Another interesting observation is the discrepancy between the curves of the Pooling- and Conv-AE model. The Pooling-AE model seems to perform much better although the only difference between the two models is the dimension reduction technique. This is contrary to Steinberger et al. [67], who stated that the pooling operation can simply be replaced by increasing the stride of the convolutional layers without sacrificing accuracy. This reinforces the thesis of Goodfellow et al. [29], who considers the pooling operation important since it leads to a translation invariance of the detected local features in smaller regions.

After evaluating the three baseline architectures, the conclusion is that the Dense-AE model best suites on the LSCI and SWIR dataset. It should also be mentioned that the collection of both LSCI and SWIR data is useful as the combination of the scores leads to a significant performance boost. Yet, the potential of using Convolutional AEs as a method for UPAD is not exhausted since the next section introduces the results obtained by using the proposed weighted MSE outlined in section 4.4.

## 7.2. Evaluation of weighted MSE

After introducing the results of the baseline AE models this section expands upon the idea of using the Dense-AE since it generated the most accurate predictions. However, in section 4.4, another loss function has been introduced which will be referred to as *weighted MSE*. Inspired by the work of Ishii et al. [32], the basic concept is to make the Dense-AE model more robust by ignoring image areas that the AE fails to reconstruct, thus increasing the reconstruction error. The usage of the proposed weighted MSE integrates an additional hyperparameter $C$ which is part of the calculation of the threshold to decide whether a pixel value will be ignored or included in the training phase. According to equation 4.8, choosing a value of $C = 0.8$ indicates a threshold configuration of $mse^j + 0.8 \cdot std^j$ where $j$ denotes the $j$-th sample within the batch set. Figure 7.6 shows two graphs, each containing DET-Curves for different hyperparameter settings of $C$.

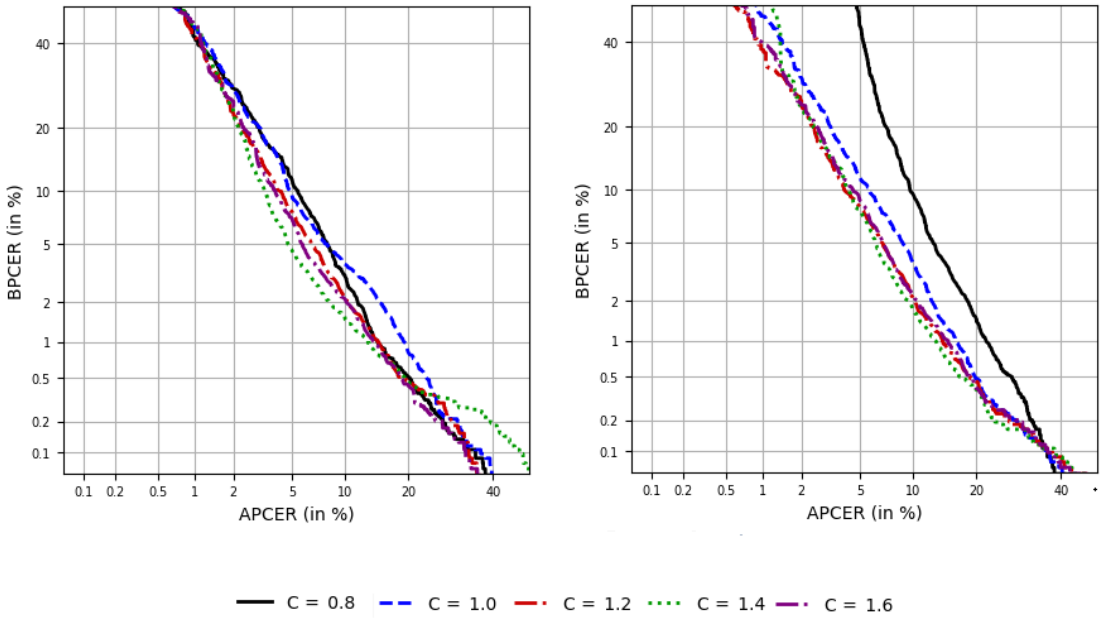


**Figure 7.6:** DET-Curves of a Dense-AE, trained with a weighted MSE and different settings of hyperparameter $C$. **Left:** LSCI **Right:** SWIR

The first observation to record is the impact of $C$ on the model performances. Looking at the graph on the right, the black curve clearly stands out from the other ones. It represents a Dense-AE which is trained with a configuration of $C = 0.8$ which is also the lowest value tested. However, since the model performs worst, this indicates that too much data has been ignored during the training stage. Consequently, the model over-generalizes the data such that it is not able to distinguish between bona fides and PAs. This can also be confirmed by the highest measured pAUC which is given in table 7.9. Looking at these values, it is interesting to see that they decrease as the value of $C$ increases. Only the last setting of $C = 1.6$ indicates a turnaround point where the performance seems to get worse again.

The results of the cross validation with $C = 1.4$ are given in table 7.8. It includes

| C | LSCI (%) | SWIR (%) |
|---|---|---|
| 0.8 | 33.36 | 55.87 |
| 1.0 | 34.78 | 35.32 |
| 1.2 | 27.84 | 27.88 |
| 1.4 | **23.07** | **27.00** |
| 1.6 | 26.43 | 29.03 |

**Table 7.7:** pAUC values measured in a range between 0-20% of different hyperparameter settings of $C$

Dense-AE models which are trained on three data folds. The comparison of the averaged metrics imply that the performance between the different datasets is quite similar. Looking at the fused score results, the LSCI and SWIR models complement each other as it significantly improved the numbers. Figure 7.7 shows the different DET-curves each of which representing different weights for the score fusion.

| Metric | Partition | LSCI (%) | SWIR (%) | Fused (%) |
|---|---|---|---|---|
| D-EER | 0 | 4.80 | 6.05 | 3.12 |
| | 1 | 7.05 | 6.00 | 4.20 |
| | 2 | 7.13 | 6.09 | 3.61 |
| | **Avg** | **6.33** | **6.05** | **3.64** |
| BPCER20 | 0 | 4.51 | 8.03 | 1.87 |
| | 1 | 11.46 | 8.72 | 3.36 |
| | 2 | 12.38 | 9.01 | 2.02 |
| | **Avg** | **9.45** | **8.59** | **2.42** |
| APCER20 | 0 | 4.68 | 6.73 | 2.34 |
| | 1 | 9.01 | 6.48 | 3.85 |
| | 2 | 8.50 | 6.52 | 3.20 |
| | **Avg** | **7.40** | **6.58** | **3.13** |
| BPCER100 | 0 | 45.19 | 38.04 | 14.95 |
| | 1 | 45.73 | 37.93 | 15.71 |
| | 2 | 39.04 | 46.47 | 24.43 |
| | **Avg** | **43.32** | **40.81** | **18.36** |
| APCER100 | 0 | 13.14 | 14.17 | 7.14 |
| | 1 | 20.60 | 13.90 | 11.55 |
| | 2 | 15.57 | 13.71 | 7.49 |
| | **Avg** | **16.44** | **13.93** | **8.73** |

**Table 7.8:** Cross Validation results of a Dense-AE that is trained with a weighted MSE on both LSCI and SWIR data. Additionally, a score fusion is given whose weights have been chosen according to table 7.9

At first glance, the blue curve stands out from the other ones. This indicates that the optimal weight factors provide for weighting the LSCI scores with 10% and the SWIR scores with 90% which is the same conclusion as the Dense-AE presented in the previous section. This first visual analysis of the DET curves can finally be validated with table 7.9 which depicts the pAUC values. The lowest measurement is printed bold and belongs to the 10-90 setting, reaffirming the initial assumption.

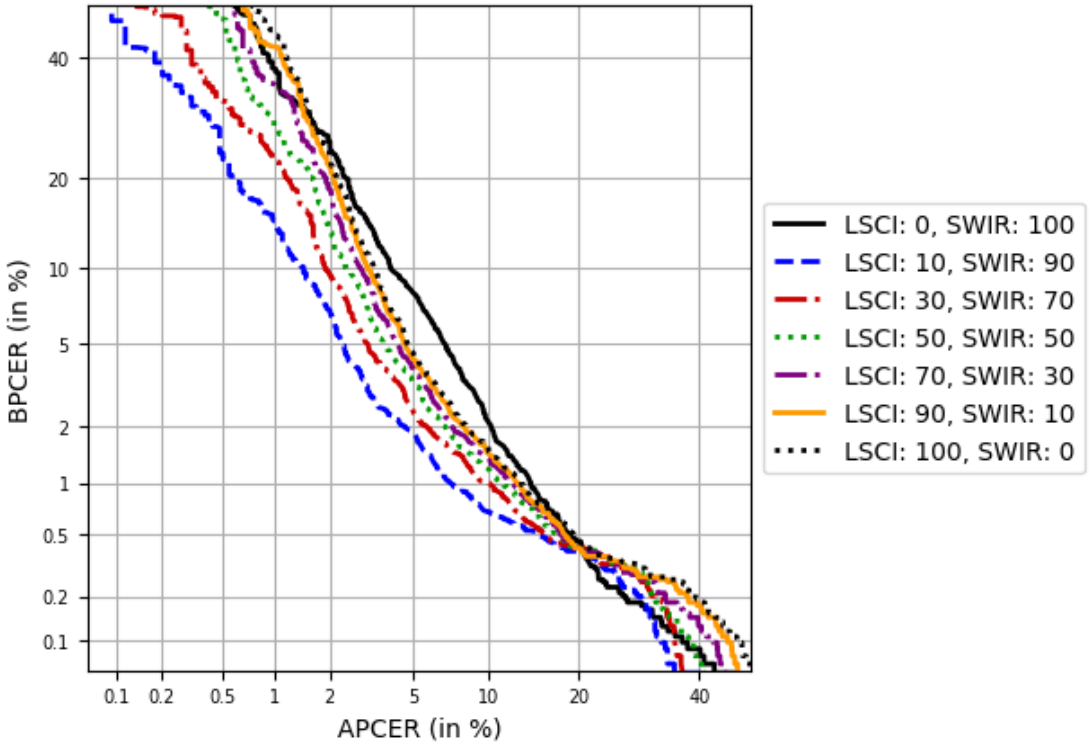


**Figure 7.7:** DET-Curves after fusing the scores of Dense-AE models, which are trained with a Weighted MSE on both LSCI and SWIR data

| LSCI weight (%) | SWIR weight (%) | pAUC (%) |
|---|---|---|
| 0.0 | 100.0 | 27.88 |
| 10.0 | 90.0 | **12.01** |
| 30.0 | 70.0 | 15.77 |
| 50.0 | 50.0 | 18.56 |
| 70.0 | 30.0 | 20.64 |
| 90.0 | 10.0 | 22.22 |
| 100.0 | 0.0 | 23.07 |

**Table 7.9:** pAUC values, measured in a range between 0-20% of the Dense-AE, trained with a weighted MSE, for multiple score weights

Now that the results of the models based on the weighted MSE loss function have been presented, the next step consists of comparing the proposed against the original MSE approach.

### 7.2.1. MSE vs. Weighted MSE

Next, the comparison between the weighted and usual MSE will be addressed. It is assumed that the model based on the weighted MSE outperforms its counterpart since it ignores noisy areas during the training. For this purpose, figure 7.8 depicts two DET curves. The black one represents the Dense-AE model trained with the usual MSE while the blue one stands for the weighted MSE method.

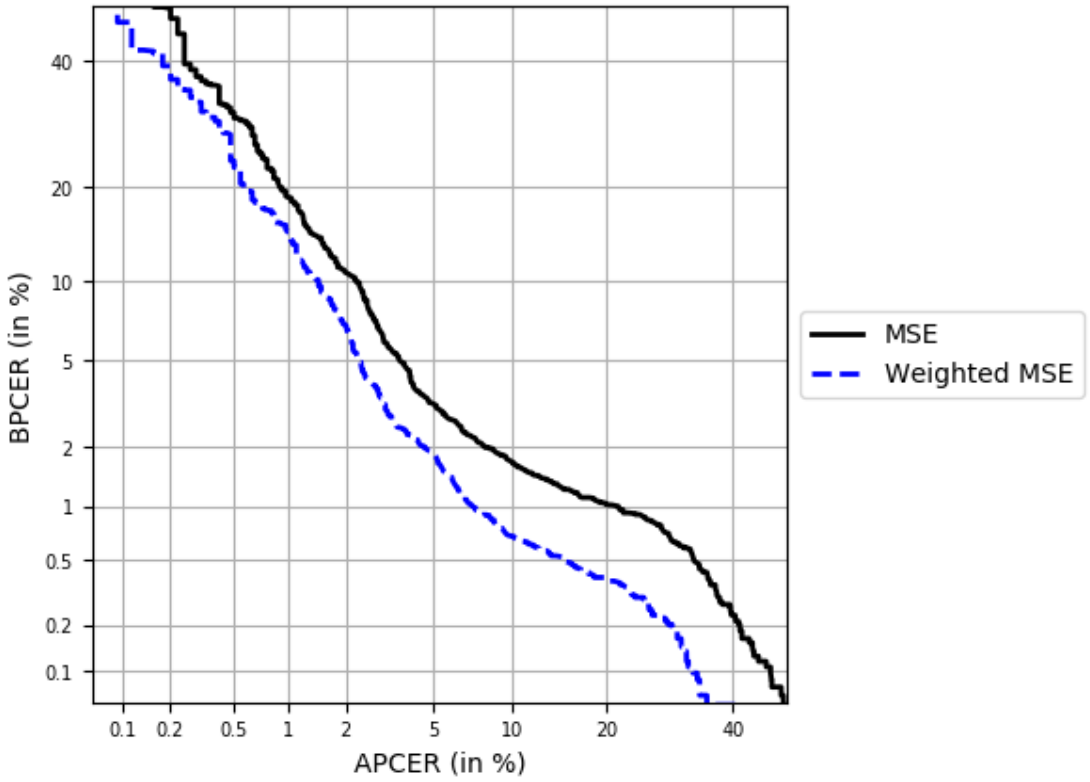


**Figure 7.8:** DET-Curves of the fused scores of two Dense-AE models, the first being trained with a weighted MSE and the second with a usual MSE

The comparison of the two curves shows that the blue curve runs consistently below the black one. Thus, indicating that using the proposed weighted MSE achieves a significant performance boost since the two curves have no intersection points. To compare the impact of the two loss functions further, figure 7.9 depicts the reconstruction errors of all images contained in the test set. Besides to the loss function, the boxplots are also divided into bona fide and PA samples.

To recall the assumption of using the proposed weighted MSE, the expectation is that the AE model is more robust against outlier areas within bona fide training data. Thus, the reconstruction error of bona fide samples containing noisy areas

is reduced and therefore leads to a lower misclassification rate. With that in mind, it is interesting to observe that the interquartile range within the bona fide class changes by replacing the usual MSE by the weighted variant. Especially the topmost point of the upper whisker decreases significantly. This means that bona fides that were previously difficult to reconstruct can now be better rebuild.

But not only the results of the bona fides, but also those of the PAs should be analyzed. In order to find an optimal threshold to distinguish between the two classes, it is mandatory to check whether the reconstruction errors also change. There would be no benefit if the loss values of the PAs decrease the same amount as the ones of the bona fides. For this purpose, the boxplots of the PAs only visualize the lower whiskers in order to be able to compare them to the bona fides on the same scale. The general loss values are naturally larger as they were not included in the training process. Also the interquartile ranges are longer since a variety of different PAIs are analyzed as a single class. In this context, it is negligible that the upper parts of the boxplots are missing, as they represent poorly reconstructed PAs that can easily be detected. However, the most interesting observation is that the overlapping interval between the upper whisker of the bona fides and the lower one of the PAs gets narrower after replacing the loss function. This inevitably means that the misclassification rate decreases and therefore the initial assumption of producing a more robust model is confirmed.

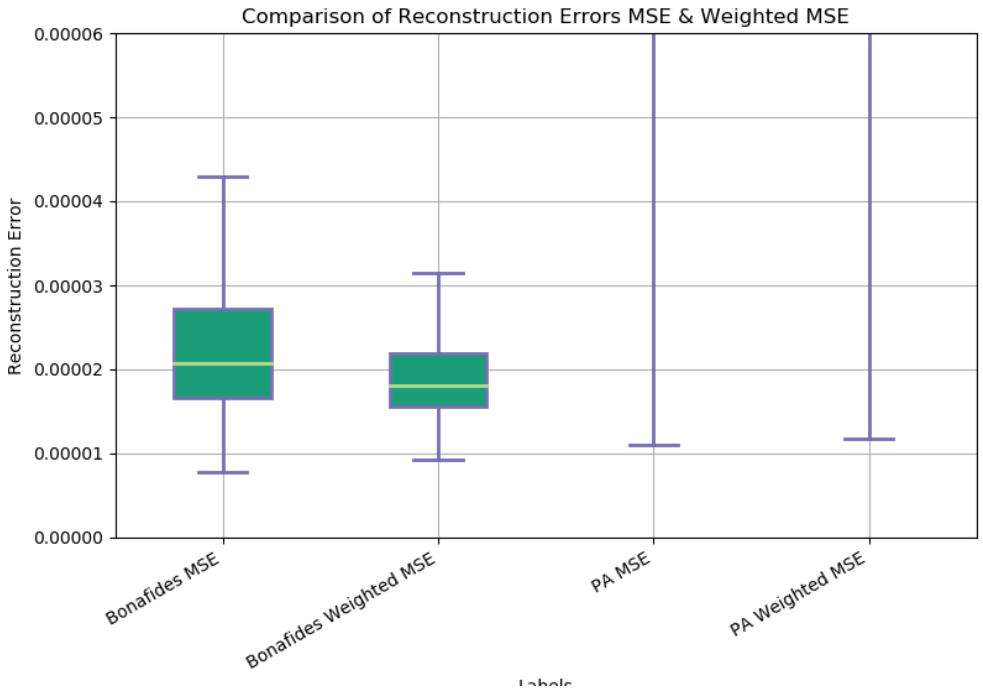


**Figure 7.9:** Visualizes the change in reconstruction errors by replacing the MSE with a Weighted MSE

To establish a deeper understanding of how the AE processes bona fide samples, figure 7.10 shows four reconstructed fingerprint images in the SWIR domain. The two images on the left are fingerprints based on the Dense-AE model trained with the weighted MSE while the right ones are based on the usual MSE. The topmost row shows the bona fide samples within the test set that resulted in the highest reconstruction error. The middle row, on the other hand, depicts the reconstructed images of the Dense-AE model. Finally, the bottom row visualizes the pixel-wise absolute difference between the original and the reconstructed image as a heatmap. This makes it easier to comprehend which areas of the image fail to be reconstructed.

The first conspicuous phenomenon to remark are the bright spots in the original images on the right. This indicates that the fingers did not cover the whole slot during the capturing process. In regards to the reconstruction capability of the AE, the light source is a disturbing factor which leads to an increased loss value. A different scenario can be observed on the left side. Contrary to the usual MSE, the weighted variant excludes outlier areas from both training and testing. Consequently, instead of the light affected samples where only a small part is responsible for the high loss values, the AE now fails to predict images where the high reconstruction errors are spread all over the image.

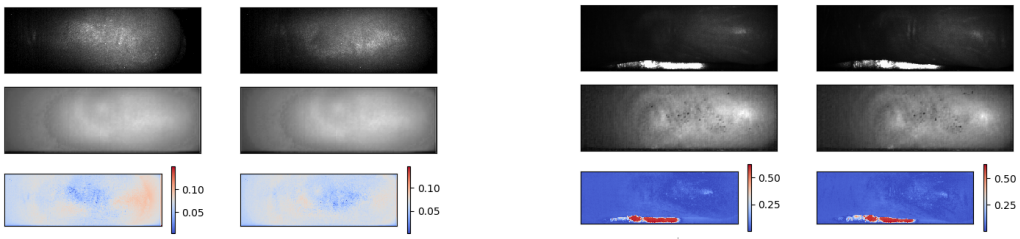


**Figure 7.10:** Visualizes the reconstructed LSCI bona fide images (middle) with the highest reconstruction errors in addition to the original images (top) and the pixel-wise absolute difference (bottom). **Left:** Weighted MSE, **Right:** MSE

Similarly to the analysis of the LSCI images, figure 7.11 shows bona fides within the SWIR domain that ended up with the highest loss values during the test set evaluation. In contrast to figure 7.10, the same type of images caused high reconstruction errors. Once again, some fingers of the biometric capture subjects did not cover the whole capturing slot, resulting in bright spots. The camera automatically focuses on the light, making the rest of the image appear very dark. Apparently, with the chosen hyperparameter setting, the weighted MSE is not able to filter out all of the bright areas which leaves potential for further optimizing $C$. Nevertheless, it is noticeable that the reconstructed images on the right are more effected from the disturbing light as the model tries to reconstruct it. On the left side however, the reconstructed images resemble more the successfully captured bona fides. This indicates that using the weighted MSE as a loss function prevents the bright spots from distorting the optimization process, therefore producing a more robust model. Since only the bona fide images with the highest reconstruction errors are included in this subsection, the interested reader is referred to appendix D where the bona fide samples with the lowest reconstruction errors are presented.

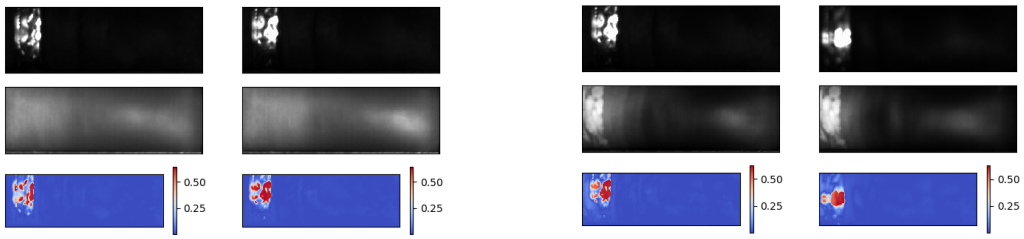


**Figure 7.11:** Visualizes the reconstructed SWIR bona fide images (middle) with the highest reconstruction errors in addition to the original images (top) and the pixel-wise absolute difference (bottom). **Left:** Weighted MSE, **Right:** MSE

Now, that the positive effect of the weighted MSE has been validated, the next section is dedicated to the question of how increasing the input channels effects the final results.

## 7.3. Evaluation of Multi-dimensional Inputs

Part of this section is to further improve the classification results by training the Dense-AE model with multi-dimensional input tensors. The assumption is that bundling several fingerprint images includes more relevant information for the final classification. In case of the LSCI dataset that contains a sequence of 100 images for each finger sample, there might be temporal information included, such as the blood flow. Therefore the first, middle, and last fingerprint images have been extracted to form a 3-dimensional input tensor. In regard to the SWIR dataset, four different wavelengths (1200nm, 1300nm, 1450nm, and 1550nm) were captured. In contrast to the last sections where only a single wavelength was used, all images are now combined into a 4-dimensional input tensor. This exploits the idea of involving complementary information throughout the whole spectrum of wavelengths. For this purpose, analogous to the last sections, figure 7.12 demonstrates how the choice of the hyperparameter $C$ effects the predictions.

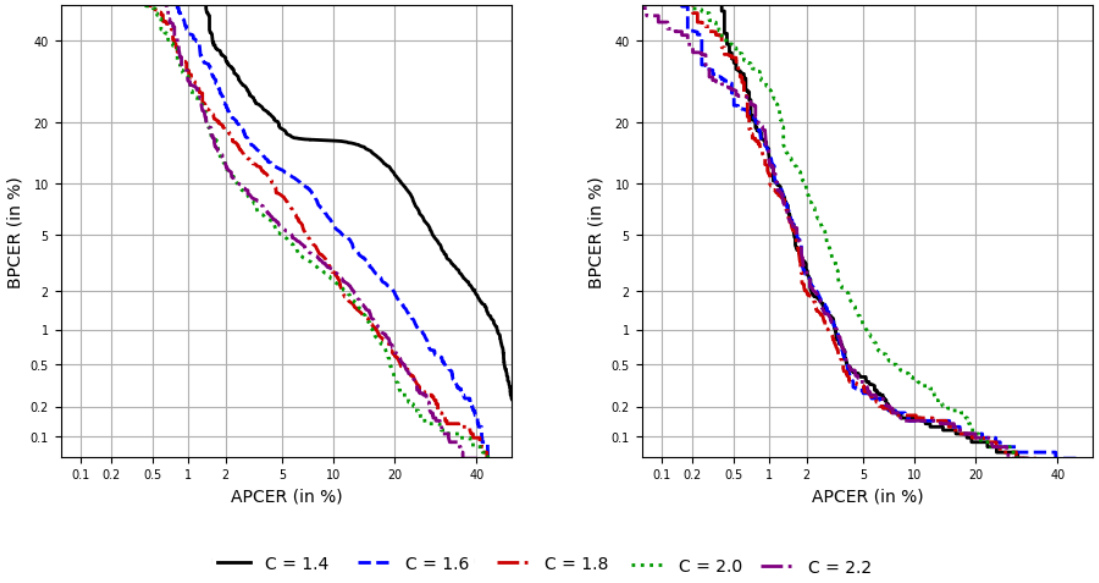


**Figure 7.12:** DET-Curves of Dense-AE, trained with a weighted MSE and multiple input dimensions, including different settings of hyperparameter $C$. **Left:** LSCI **Right:** SWIR

Comparing the curves between the two datasets, it can be observed that the volatility remarkably differs. While the curves on the left (LSCI) tend to be varying more, the ones on the right (SWIR) are more robust. Considering the pAUC values of the LSCI models in table 7.10, the lower $C$ gets chosen, the more the model tends to overgeneralize the data, expressing in a poorer performance.

| $C$ | **LSCI (%)** | **SWIR (%)** |
|---|---|---|
| 1.4 | 82.65 | 7.86 |
| 1.6 | 41.10 | 7.90 |
| 1.8 | 28.82 | **7.30** |
| 2.0 | **22.45** | 8.94 |
| 2.2 | 24.37 | 7.95 |

**Table 7.10:** pAUC values measured in a range between 0-20% of different hyperparameter settings of $C$

After optimizing $C$, figure 7.13 shows the comparison between the single input based models of the previous section and the ones trained with multiple dimensions. At this point, it is beneficial to decide as early as possible whether increasing the input dimension improves the results since the training process requires more time and resources.

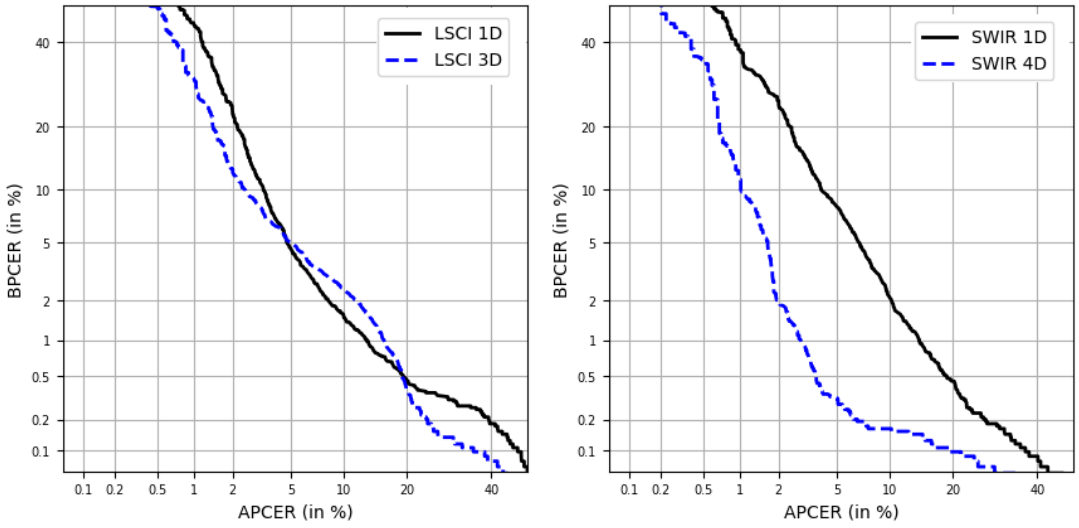


**Figure 7.13:** Comparison of the performance between Dense-AE models, trained on single vs. multiple input dimensions. **Left:** LSCI, **Right:** SWIR

Using multiple dimensions, as described above, has two opposite effects. On the left, the curves intersect two times and it is not clear which one outperforms its counterpart. Also the measured pAUC values of 23.07% and 22.45% reflect that adding two additional input dimensions has little effect. This is most likely due to the high similarity between the 100 images. However, directing the focus on the right to the SWIR results, the additional information from the other wavelengths significantly improve the predictions. The blue curve consistently runs below the black one. Furthermore, the measured pAUC improved from 12.01% to a value of 7.30%. Therefore it can be concluded that acquiring fingerprints in the SWIR

domain on several wavelengths leads to a complementary effect. To validate this observation and to prevent it from being caused by the random data split, table 7.11 summarizes several metrics on three data folds.

| Metric | Partition | LSCI (%) | SWIR (%) | Fused (%) |
|---|---|---|---|---|
| D-EER | 0 | 4.94 | 2.00 | 2.00 |
| | 1 | 5.25 | 2.29 | 2.29 |
| | 2 | 6.23 | 3.02 | 3.02 |
| | **Avg** | **5.47** | **2.47** | **2.47** |
| BPCER20 | 0 | 4.93 | 0.31 | 0.31 |
| | 1 | 5.25 | 0.28 | 0.28 |
| | 2 | 7.98 | 1.13 | 1.13 |
| | **Avg** | **6.05** | **0.57** | **0.57** |
| APCER20 | 0 | 4.87 | 1.66 | 1.66 |
| | 1 | 5.30 | 1.63 | 1.63 |
| | 2 | 6.84 | 2.54 | 2.54 |
| | **Avg** | **5.67** | **1.94** | **1.94** |
| BPCER100 | 0 | 30.73 | 11.74 | 11.74 |
| | 1 | 28.86 | 12.39 | 12.39 |
| | 2 | 36.15 | 23.79 | 23.79 |
| | **Avg** | **31.91** | **15.97** | **15.97** |
| APCER100 | 0 | 15.92 | 2.89 | 2.89 |
| | 1 | 12.24 | 3.04 | 3.04 |
| | 2 | 14.57 | 5.19 | 5.19 |
| | **Avg** | **14.24** | **3.71** | **3.71** |

**Table 7.11:** Cross Validation results of a Dense-AE that is trained with multiple input dimensions on both LSCI and SWIR data. Additionally, a score fusion is given whose weights have been chosen according to table E.1

Fortunately, the numbers obtained between the three partitions are all in the same order of magnitude. Apparently, the measurements of the fused scores are identical to those of the SWIR model. That is because increasing the weight factors on the LSCI scores consistently led to a performance decline. Therefore it can be concluded that both SWIR and LSCI model do not complement each other in this scenario. The interested reader is referred to appendix E where the DET-Curves, each representing different weight combinations, can be found. As the SWIR results look very promising, a further analysis of the PAI species classes introduced in section 5.1 is presented in the next subsection.

### 7.3.1. PAI Species Class Analysis

So far, the previous sections focused on reporting overall performance metrics. However, as discussed earlier, it is also required to establish an understanding of which PAI species leads to the highest number of misclassified PAs. In this context, figure 7.14 depicts seven boxplots, each representing the reconstruction errors of a different PAI species class.

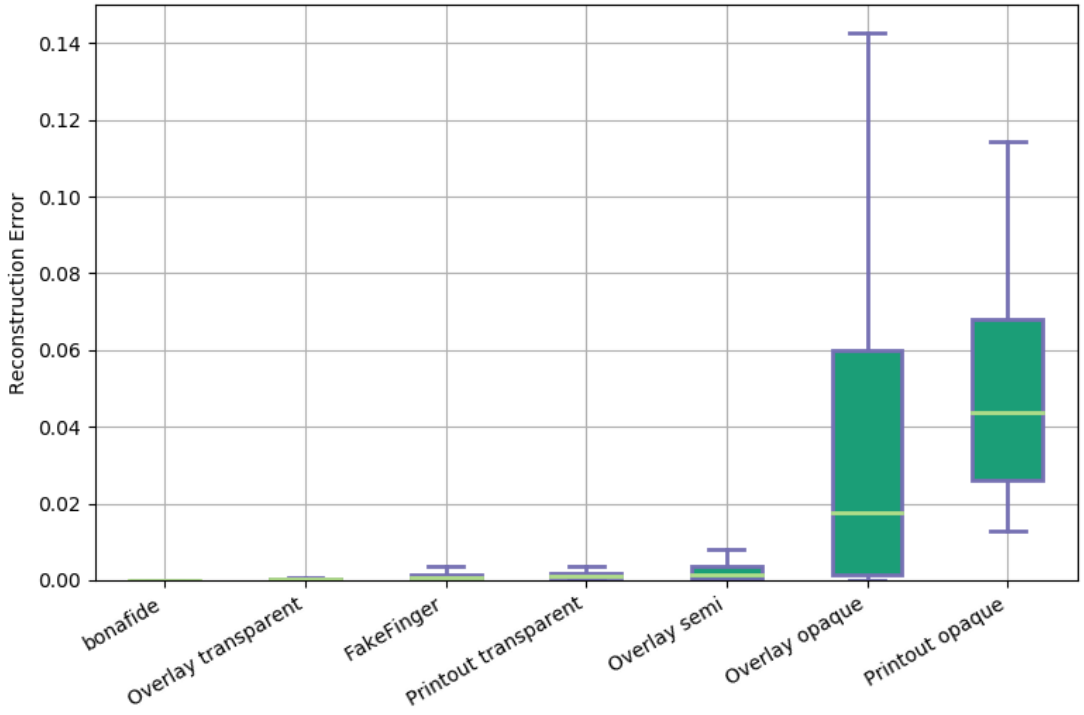


**Figure 7.14:** Overview of the reconstruction errors within the PAI species classes, obtained by a Dense-AE model, trained on SWIR data with multiple input dimensions and a weighted MSE (ascending by median)

Since the boxplots are sorted by the magnitude of its median, the first observation worth mentioning is that the bona fide fingerprint images have the lowest reconstruction error in average. This speaks for the models capability of learning about the structure of real fingerprints. On the other hand, looking at the PAI species classes with the highest medians, both opaque overlays as well as opaque printouts are located at the top. Especially, the reconstruction errors of the samples stemming from the opaque overlay class have a wide range, indicating that the images are heterogeneous. Yet, it is difficult to estimate which PAI species classes are most responsible for the classification errors. For that reason, figure 7.15 depicts the same setting on a more appropriate scale that enables a deeper analysis of

whisker overlaps.

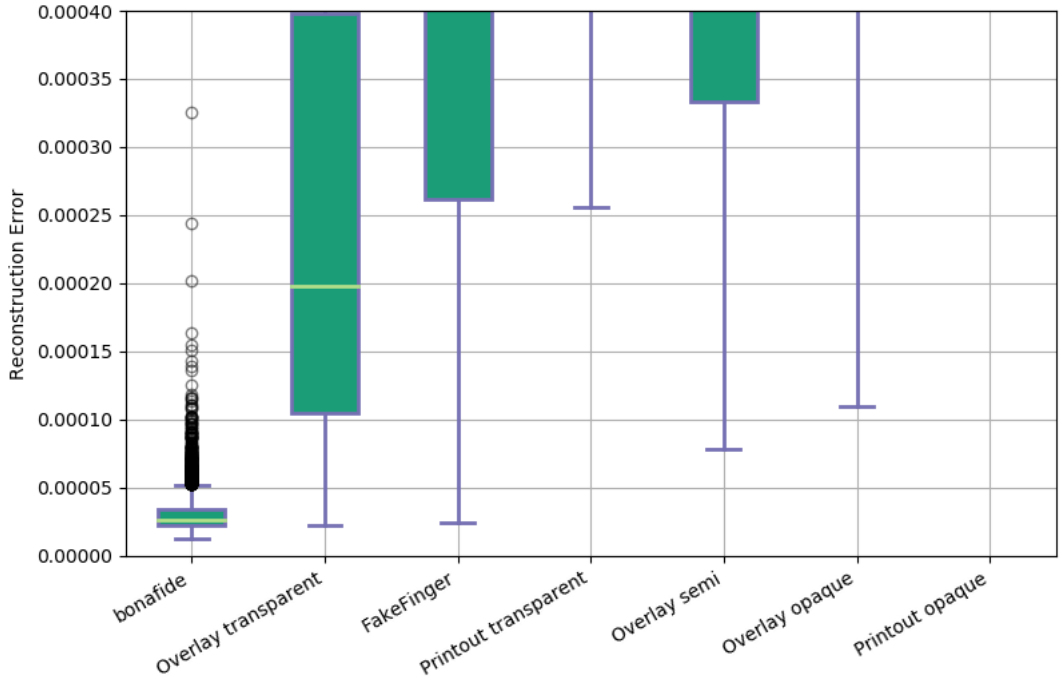


**Figure 7.15:** Overview of the reconstruction errors within the PAI species classes, obtained by a Dense-AE model, trained on SWIR data with multiple input dimensions and a weighted MSE (ascending by median)

Unlike before, the boxplots are supplemented by outlier values which allow to better estimate how they influence the position of the optimal threshold. On this scale however, it is easy to recognize that the most sophisticated PAs are either transparent overlays or fake fingers since the whiskers of their boxplots have the largest overlaps with the bona fide one. In contrast, the PAs belonging to the printout classes are perfectly separable from the bona fides as even the lowest reconstruction error is far greater then most of the bona fide outliers. The boxplots whiskers of both semi and opaque overlays do not overlap with the bona fides. Nevertheless, choosing the threshold too low will inevitably lead to several bona fide outliers being misclassified. Therefore, for biometric systems with convenience in mind, it might be beneficial to choose a higher threshold. To have a rough idea of how the Dense-AE model internally structures samples from different PAI species classes, figure 7.16 shows a t-SNE scatterplot where every data point represents one test sample. As depicted in figure 7.1, the latent representation of an image within the Dense-AE architecture is a 64-dimensional vector. Consequently, using the Dense-AE model to encode the 14780 images within the test set leads

accordingly to 14780 64-dimensional vectors. The latent representations dimensions can then be reduced into the two-dimensional space using t-SNE. This allows to draw conclusions about how the Dense-AE structures samples stemming from different PAI species classes.


**Figure 7.16:** Low-dimensional representations of the latent vectors of a Dense-AE, trained on SWIR data with multiple input dimension and a weighted MSE using t-SNE

Apparently, the largest clusters are formed by the bona fides. However, also samples that were assigned as opaque overlays form clusters that are spread throughout the plot. This might be a sign that the class can be further splitted into subclasses

which is also indicated by the high range of reconstruction errors measured within the boxplot chart. It is conceivable that next to the material, another dimension, such as the brightness, could be considered to define more detailed classes. The more specific the classes are formulated, the more comprehensive will be the understanding of how the Dense-AE model reacts to different PA types. Also of interest are the bona fide samples that fall into another cluster as they might be considered as outliers and therefore should be excluded from both test and training set. Unfortunately, a deeper analysis of the received clusters is outside of the scope of this thesis, leaving potential for future works.

As a first conclusion, it can be summarized that the best performing Dense-AE model is trained on the SWIR dataset, including multiple input dimensions and a weighted MSE loss function. The following section uses these findings to benchmark them against comparable fingerprint PAD techniques.

# 8. Benchmarking

In order to compare the performance of the evaluated methodology of this thesis, a few other fingerprint PAD approaches will be presented as a benchmark for the proposed Dense-AE. The following overview in advance will briefly outline how the following subsection are structured.

- Within section 8.1, the performance of the proposed Dense-AE model is benchmarked against a CNN model introduced by Gomez-Barrero et al. [25]. Since the AE was only trained on bona fide samples, whereas the training of CNNs includes both bona fide and PAs, a *leave-one-out* technique was used to increase comparability. For this purpose, several CNN models were trained, excluding different PAI groups and including them only in the test set.

- Section 8.2 compares the performance of the proposed Dense-AE model with various OC-SVM approaches which are also trained on bona fides solely. Since OC-SVMs require vector inputs instead of tensors with multiple channels, the first OC-SVM has been trained on the latent representations as a substitute of the decoder part of the Dense-AE. Additionally, two pre-trained CNNs (VGG19 [65], VGGFace [57]) were used to extract feature vectors from the original fingerprint images, which were then taken as an input for two separately trained OC-SVMs.

## 8.1. Dense-AE vs. CNN

As mentioned in the introductory part of this section, the proposed Dense-AE model is benchmarked against the CNN approach of Gomez-Barrero et al. [25]. Specifically, the fine-tuned VGGFace model produced the most promising results and is therefore used as a benchmark. Since both the work of Gomez-Barrero et al. as well as this thesis originated as part of the *Biometrics and Internet-Security Research Group*, both of the models were trained and tested on exactly the same data, which increases the validity of the comparison. However, since the CNN reference model needs to be trained on both bona fides and PAs, it is unclear how it classifies unseen attacks that were not part of the training set. For this reason, in order to establish a more appropriate benchmark, several CNN models were trained, each excluding a different PAI group from the training phase. These left-out groups are constructed of the PAI species classes introduced in section 5.1 and composed as follows:

- **Fake fingers:** Includes only Fake fingers
- **Fake fingers + Opaque:** Includes Fake fingers, Opaque Overlays and Opaque Printouts
- **Overlay + Printout:** Inlcudes Fake fingers, Opaque Overlays, Semi Overlays, Transparent Overlays, Opaque Printouts and transparent Printouts
- **Overlay Semi:** Includes only Semi Overlays
- **Semi + Transparent:** Includes Semi Overlays, Transparent Overlays and Transparent Printouts
- **Overlay Opaque:** Includes Opaque Overlays and Opaque Printouts
- **Transparent:** Includes Transparent Overlays and Transparent Printouts

Given the above defined groups, figure 8.1 shows the first part of the DET-curves corresponding to the left-out groups.

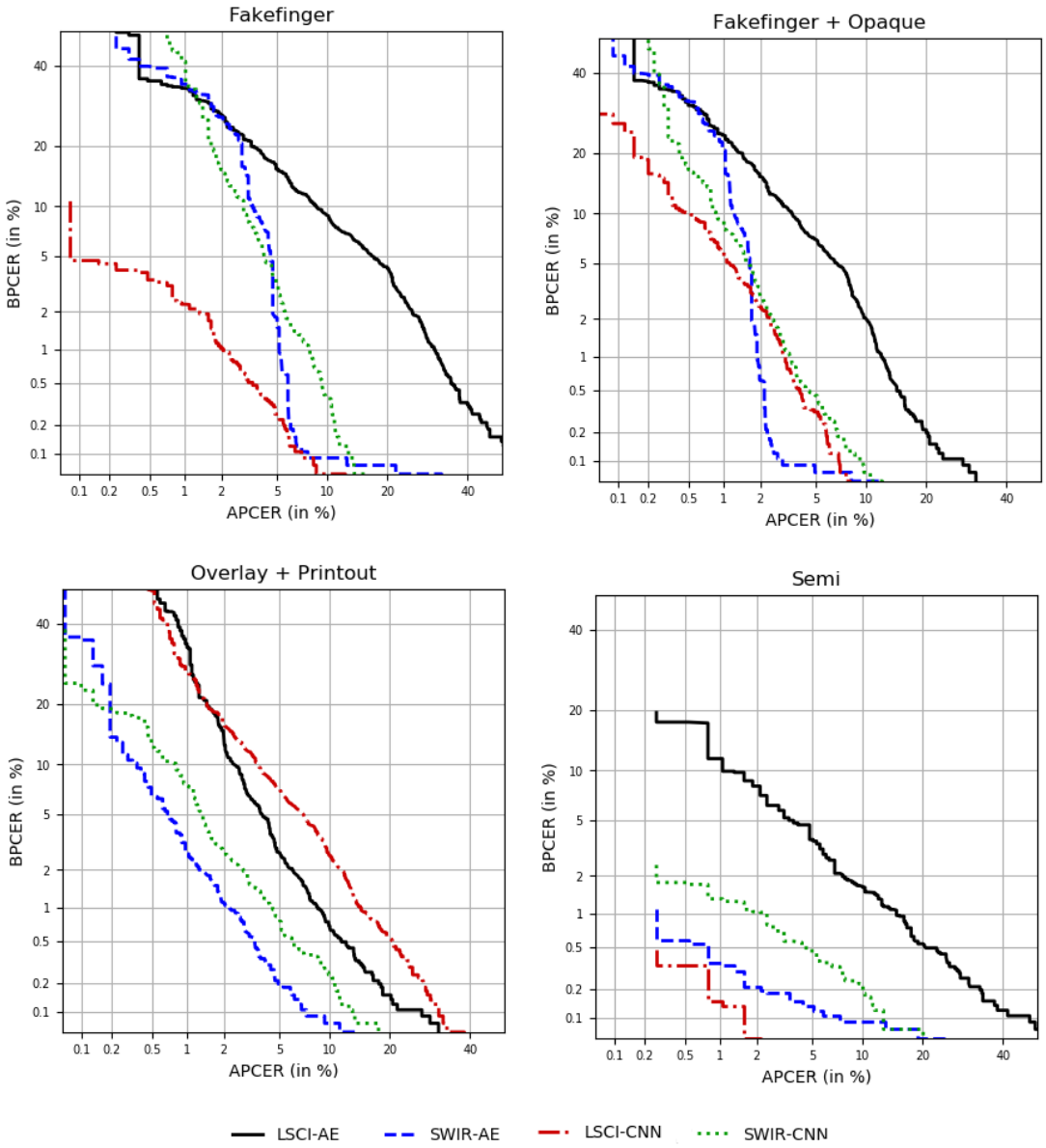


**Figure 8.1:** Comparison of the Performance between the best performing AE models from section 7 with the CNN approach described above within different *leave-one-out* groups

The first visual analysis of the AE curves proves that the LSCI model generally perform worse than its counterparts. However, the SWIR model looks more promising as it shows competitive results compared to the CNN approaches. Looking at the *Overlay + Printout* chart, it clearly outperforms the CNN models on both datasets. However, the comparison gets more difficult in the *Fakefinger + Opaque* scenario where multiple curves intersect. Therefore, it is up to the performance

metrics given in table 8.1 and 8.2 to decide which model is superior. Nevertheless, this decision should also be attuned to the final applications purpose. Within the *Fakefinger* such as the *Semi* group, the CNN model trained on the LSCI dataset works best. Finally, figure 8.2 depicts the DET-curves of the remaining PAI left-out groups.

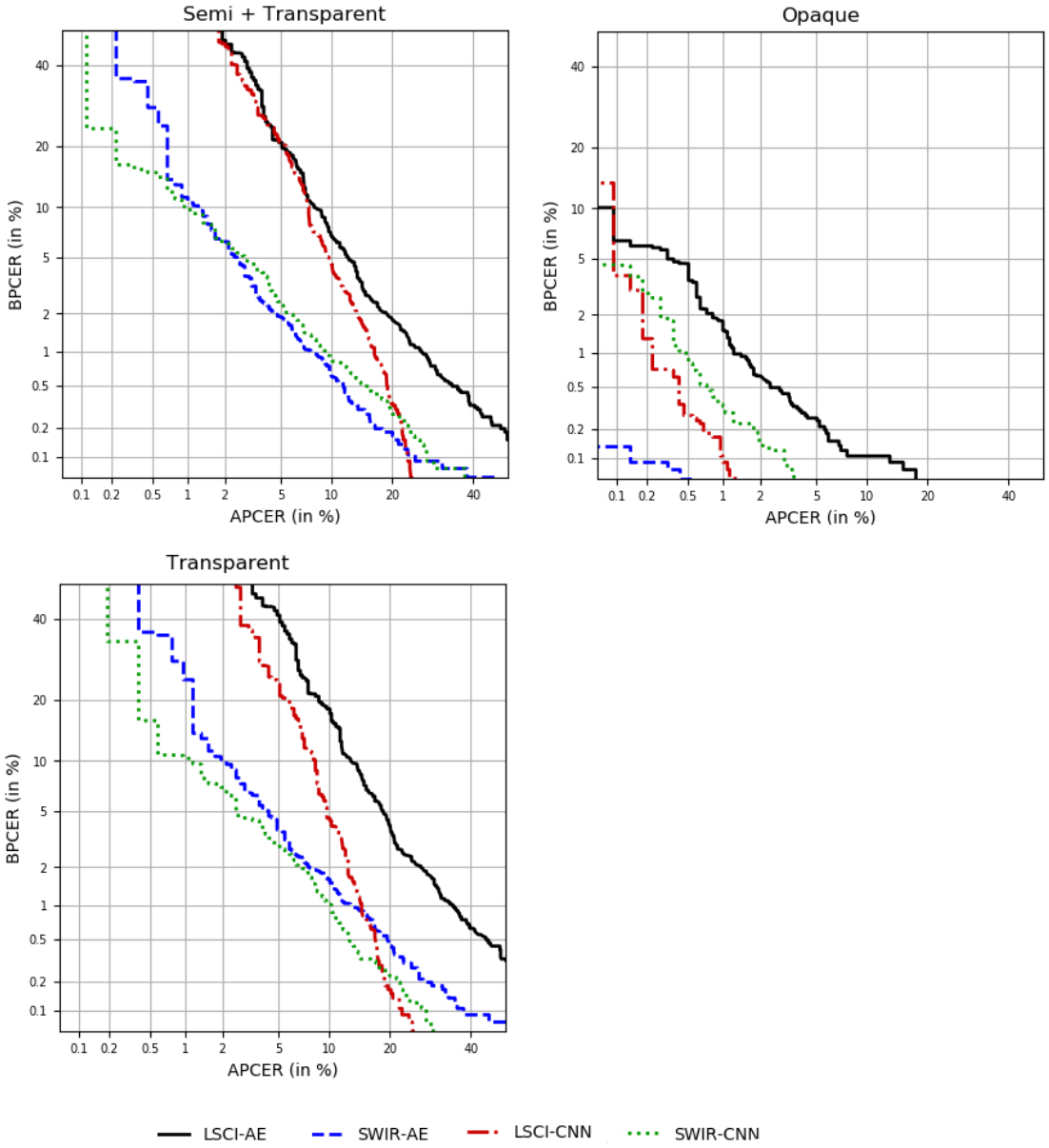


**Figure 8.2:** Comparison of the Performance between the best performing AE models from section 7 with the CNN approach described above within different *leave-one-out* groups

Summarizing the observations of the three remaining PAI groups, the impression is confirmed that the AE model which is trained on the LSCI data performs worst. Of particular interest is the AE model trained on SWIR data as it outperforms all of its counterparts within the *Opaque* group. This means that especially for opaque overlay PAs, which account for more than half of the test set data, the Dense-AE model seems to be the best choice. Last but not least, the SWIR AE and CNN models perform similarly within the *Semi + Transparent* and *Transparent* group.

Table 8.1 and 8.2 compare the above depicted DET-curves based on the pAUC on the one hand and the D-EER on the other. The lowest number in a row is marked in bold, indicating that the according model type is best suited to detect this kind of unseen PAs. As already derived in the above visual analysis, the SWIR AE model is of particular interest, as it is most competitive. And indeed, in three out of seven groups it outperforms the other models in terms of the measured pAUC and even in four regarding the D-EER. This proves that the transition from semi to unknown fingerprint PAD does not necessarily lead to a performance decline. It is noticeable that the worst results among all models were obtained when only transparent overlays were included in the test set. This indicates that these types of PAs resemble most to real fingerprint images and thus represent the most sophisticated PAs.

| **PAI species Class** | **S-AE** | **S-CNN** | **L-AE** | **L-CNN** |
|---|---|---|---|---|
| Fake Finger | 19.1 | 17.51 | 53.63 | **2.18** |
| Fake Finger + Opaque | 7.17 | 6.73 | 24.22 | **4.73** |
| Overlay + Printout | **3.46** | 6.55 | 17.03 | 26.59 |
| Semi | 1.81 | 1.86 | 16.08 | **0.18** |
| Semi + Transparent | **11.04** | 12.22 | 49.77 | 42.87 |
| Opaque | **0.06** | 0.58 | 1.98 | 0.56 |
| Transparent | 18.34 | **12.57** | 71.41 | 44.48 |

**Table 8.1:** Measured pAUC values of the DET-Curves from figure 8.2 and 8.1

| **PAI species Class** | **S-AE** | **S-CNN** | **L-AE** | **L-CNN** |
|---|---|---|---|---|
| Fake Finger | 4.59 | 4.45 | 9.41 | **1.68** |
| Fake Finger + Opaque | **1.81** | 2.75 | 5.86 | 2.27 |
| Overlay + Printout | **1.63** | 2.35 | 4.27 | 5.88 |
| Semi | 0.52 | 1.34 | 4.93 | **0.37** |
| Semi + Transparent | **3.21** | 3.87 | 8.88 | 7.78 |
| Opaque | **0.17** | 0.65 | 1.22 | 0.45 |
| Transparent | 4.49 | **3.95** | 11.89 | 8.58 |

**Table 8.2:** Measured D-EERs of the DET-Curves from figure 8.2 and 8.1

In conlcusion, it can be stated that particularly the Dense-AE trained on the SWIR dataset performs highly competitive compared to the CNN models. In addition to a comparable performance, the use of AEs for the leave-one-out test scenario also had an efficiency advantage since only a single model needed to be trained. On the other hand, seven different CNN models were trained in order to exclude each group.

## 8.2. Dense-AE vs. One-Class SVMs

However, since the comparison between a model trained on PAs with a model that was only trained on bona fides still is a bit unbalanced, this section provides a comparison against another unknown fingerprint PAD approach. More specifically, various OC-SVMs were trained using a RBF-Kernel. Since OC-SVMs are trained on one-dimensional vectors, the following three feature extraction techniques were chosen to provide a compatible input.

- The first OC-SVM is trained on the latent representations of the the Dense-AE model. As the t-SNE plot 7.16 shows, we can observe that the distribution is well structured in the 64-dimensional space and might therefore be a suitable input. Generally thinking, the OC-SVM then replaces the decoder phase of the Dense-AE model.

- The second OC-SVM is trained on feature vectors which were extracted with the *VGG-19* pre-trained CNN [65] to exploit transfer learning. The CNN was trained on *ImageNet* [16] which is a large-scale hierarchical image database containing 3.2 million annotated images.

- As an alternative pre-trained CNN, VGGFace [57] is also used as a feature extractor. This model was trained on a large-scale dataset containing 2.6 million facial images of 2.6k people. Since the training process of the VGGFace model was exposed to faces, it might be extracting more relevant skin-features the OC-SVM could benefit from.

Both the DET-Curves of the Dense-AE such as the OC-SVMs are depicted in figure 8.3.

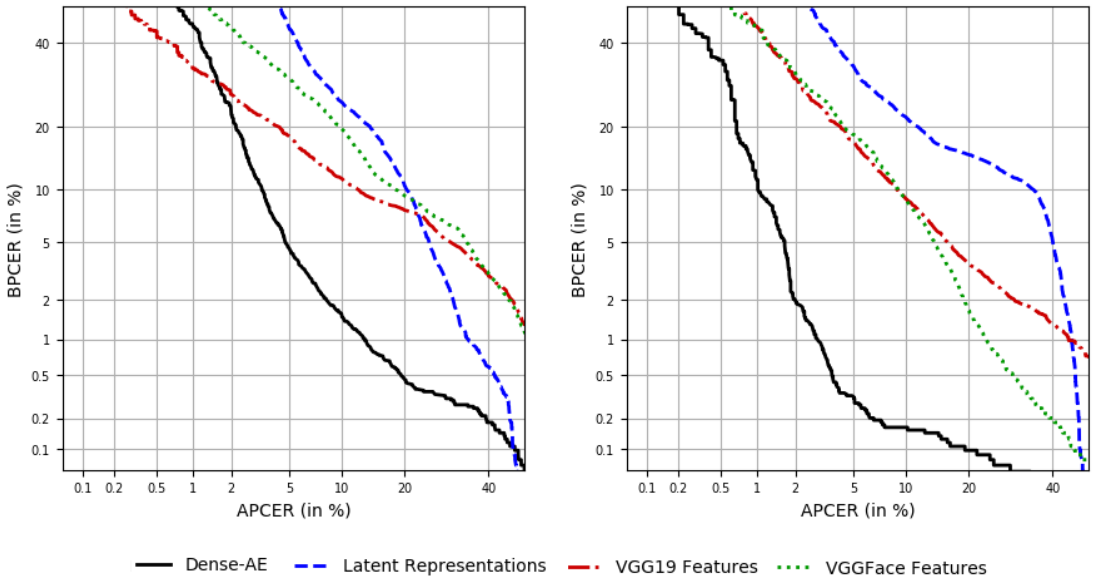


**Figure 8.3:** Comparison of the Performance between the best performing AE models from section 7 with the OC-SVM approaches described above. **Left:** LSCI, **Right:** SWIR

The black curves clearly stand out on both sides which means that the Dense-AE outperformed all of the OC-SVM models. This can also be validated by table 8.3 and 8.4 which show that the pAUC as well as the D-EER of the Dense-AE are lowest. Notably, both OC-SVMs that are trained on the VGG19 and VGGFace features perform quite similar. The worst results however were measured in case of the OC-SVM that was trained on the latent representations, reporting D-EERs higher than 15% on both datasets.

| **Model** | **SWIR** | **LSCI** |
|---|---|---|
| Dense-AE | **7.30** | **23.07** |
| Latent OC-SVM | 92.98 | 92.62 |
| VGG19 OC-SVM | 54.51 | 64.96 |
| VGGFace OC-SVM | 53.01 | 82.36 |

**Table 8.3:** Measured pAUC values of the DET-Curves from figure 8.3

| Model | SWIR | LSCI |
|---|---|---|
| Dense-AE | **2.00** | **4.80** |
| Latent OC-SVM | 16.17 | 16.21 |
| VGG19 OC-SVM | 9.49 | 10.78 |
| VGGFace OC-SVM | 9.52 | 13.62 |

**Table 8.4:** Measured D-EERs values of the DET-Curves from figure 8.3

Finally, it can be concluded that the proposed Dense-AE model consistently outperformed the OC-SVMs which again validates the effectiveness of using Convolutional AEs for detecting unknown fingerprint PAs. However, future works may focus on establishing better benchmarks by extracting more relevant features from the fingerprint images.

# 9. Conclusion

This thesis aims to evaluate whether Convolutional AEs are well suited as a methodology for unknown fingerprint PAD, especially in regard to LSCI and SWIR data. To address this question, three AE architectures have been implemented and compared with each other. Based on the results of the cross validation, it can be concluded that the Dense-AE significantly outperforms both other model architectures. An average D-EER of 4.36% has been measured compared to values of 10.82% (Pooling-AE) and 12.52% (Conv-AE).

Furthermore, a weighted MSE has been introduced as a novel loss function to increase the robustness of a Convolutional AE. This is accomplished through excluding image areas that the AE fails to reconstruct during the optimization phase. Hence, the AE focuses on the most relevant areas while disturbing noise is ignored. The positive effect of the weighted MSE could be demonstrated, as the replacement of the loss function reduces the average D-EER of the Dense-AE from 4.36% to 3.64% and the according pAUC values from 18.79% to 12.01%.

Finally, the classification results could be further improved by using multiple input dimensions. In case of the LSCI data, a 3-dimensional input has been tested to exploit the temporal effect that results from capturing images in a sequence. However, despite an average improvement of the D-EER from to 6.33% to 5.47%, the partially measured AUC improved only slightly from 23.07% to 22.45%. Contrarily, a clearer result could be achieved in case of the SWIR data, where the inclusion of four different wavelengths reduced the average D-EER from 6.05% to 2.47% while at the same time the pAUC decreased from 27.00% to 7.30%. To investigate if the LSCI and SWIR models complement each other, their scores have been fused with a weighted average. However, a complementary effect could not be confirmed as the best classification results were measured using the SWIR model alone.

The last part of the work compares the results of the best performing LSCI and SWIR AE models to alternative fingerprint PAD approaches, trained and tested on the same data partitions. First, the Dense-AE has been compared to a fine-tuned VGGFace CNN introduced by Gomez-Barrero et al. [26]. Since the CNN is trained on both bona fides and PAs, seven different models were trained, each excluding a different group of PAs from the training process. The analysis of the performances indicate that the Dense-AE trained on SWIR data achieves highly competitive results. Considering the D-EERs, it outperforms the CNN models in four out of seven PA groups.

The second comparison includes two OC-SVMs that are trained on the latent representations of the Dense AEs (LSCI + SWIR). Additionally, four OC-SVMs were trained on image features extracted with pre-trained CNNs (VGG-19, VGGFace). The measured D-EERs indicate that none of the OC-SVMs produced competitive results, as the best OC-SVM (VGG-19) achieved a pAUC of 53.01% with an according D-EER of 9.52%.

All in all, the objective of this work has been achieved, as the proposed Dense-AE has proven to be highly competitive with other approaches. These findings could be of particular interest in a scenario where the production of a wide range of PAIs is too cost-intensive. Future works might expand on the idea of developing techniques for detecting unknown PAs as the ensemble of multiple approaches could further improve the classification results.

# Appendix

## A. Breakdown of PAI Species

The following tables depict all of the PAIs that are contained within the PAI species classes:

| PAI Species Class | PAI Species Names | Count |
|---|---|---|
| Overlay Opaque | overlay ecoflex fleshtone | 699 |
| | overlay conductive silicone black | 432 |
| | overlay ecoflex factor2 tan | 336 |
| | overlay ballistic gelatin fleshtone | 194 |
| | overlay PDMS fleshtone | 122 |
| | overlay conductive silicone | 100 |
| | overlay silicone solutions | 98 |
| | overlay urethane with gold | 72 |
| | overlay dental material | 51 |
| | overlay dragonskin opaque | 17 |
| | bandage plaster (no fp) | 14 |
| | **Total** | **2135** |
| Fake Finger | dragonskin | 426 |
| | ecoflex | 147 |
| | monster latex | 78 |
| | wax | 74 |
| | ecoflex with graphite | 72 |
| | latex with gold | 69 |
| | ecoflex with nanotips white | 54 |
| | playdoh orange | 53 |
| | 3D printed finger | 48 |
| | dental material | 33 |
| | dragonskin with nanotips white | 27 |
| | silly putty original | 25 |
| | dragonskin with barepaint | 24 |
| | playdoh white | 24 |
| | 3D printed finger with Ag | 24 |
| | playdoh yellow | 24 |
| | ecoflex with barepaint | 18 |
| | silly putty metallic | 15 |
| | silly putty glow-in-the-dark | 15 |
| | playdoh black | 15 |
| | **Total** | **1265** |

**Table A.1:** Shows the specific PAIs and the according species classes - Part 1

| PAI Species Class | PAI Species Names | Count |
|---|---|---|
| Overlay Transparent | overlay two part silicone | 157 |
| | overlay knox gelatin | 107 |
| | overlay dragonskin clear | 106 |
| | overlay monster latex | 34 |
| | overlay school glue white | 25 |
| | overlay wax | 18 |
| | overlay school glue clear | 2 |
| | **Total** | **449** |
| Overlay Semi | overlay conductive silicone yellow | 160 |
| | overlay school glue | 76 |
| | overlay wood glue | 70 |
| | overlay dragonskin semi | 47 |
| | overlay ecoflex | 24 |
| | **Total** | **377** |
| Printout transparent | printout transparent | **64** |
| Printout opaque | printout paper | **49** |

**Table A.2:** Shows the specific PAIs and the according species classes - Part 2

## B. Evaluation of different Partition Sizes

In order to decide how to split into train, valid, and test sets, different partition sizes have been tested on both LSCI and SWIR data. Figure B.1 shows the according DET-curves. Since the curves of the 30-20 and 50-30 splits perform similar in both cases, decision has been made to use 30% of the bona fides as training and 20% of them as validation data. In this setting, the training process involves enough bona fide samples to learn relevant patterns, while at the same time the test set contains enough data for a profound evaluation.

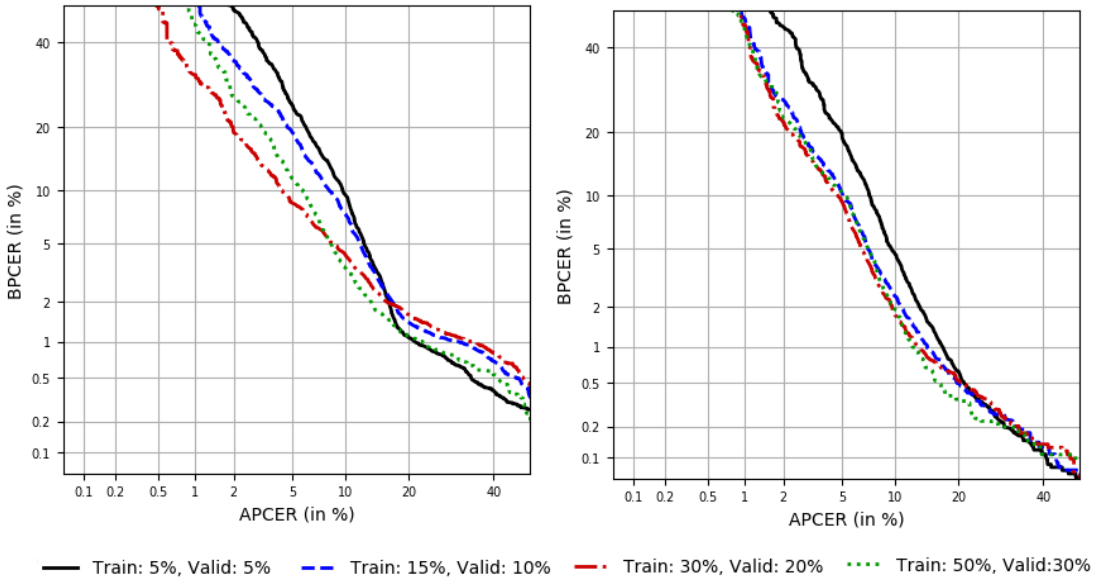


**Figure B.1:** Evaluation of different data partition settings based on a Dense-AE model. **Left:** LSCI, **Right:** SWIR

# C. Packages and Versions

The following list specifies all packages that have been used for the experiments within this thesis:

```
absl-py==0.8.1
astor==0.7.1
batl-utils-modules==0.1
certifi==2019.11.28
cycler==0.10.0
gast==0.3.2
grpcio==1.23.0
h5py==2.10.0
HDAFingerPAD==0.1
joblib==0.14.0
Keras==2.3.1
Keras-Applications==1.0.8
Keras-Preprocessing==1.1.0
keras-vggface==0.6
kiwisolver==1.1.0
Mako==1.1.0
Markdown==3.1.1
MarkupSafe==1.1.1
matplotlib==3.1.1
mkl-fft==1.0.15
mkl-random==1.1.0
mkl-service==2.3.0
mock==3.0.5
numexpr==2.7.0
numpy==1.17.4
olefile==0.46
pandas==0.25.3
patsy==0.5.1
Pillow==6.2.1
protobuf==3.11.1
pygpu==0.7.6
pyparsing==2.4.5
python-dateutil==2.8.1
pytz==2019.3
PyYAML==5.2
scikit-learn==0.21.3
scipy==1.3.1
```

```
seaborn==0.9.0
six==1.13.0
statsmodels==0.10.1
tables==3.4.4
tensorboard==1.13.1
tensorflow==1.13.1
tensorflow-estimator==1.13.0
termcolor==1.1.0
Theano==1.0.4
tikzplotlib==0.8.7
tornado==6.0.3
Werkzeug==0.16.0
```

# D. Visualization of lowest Reconstruction Error

The following two figures D.1 and D.2 visualize those bona fide images with the lowest reconstruction errors. Analogous to section 7.2.1, the first row visualizes the original image while the second row depicts the reconstructed one. Additionally, the third row shows the pixel-wise absolute difference as a heat-map.

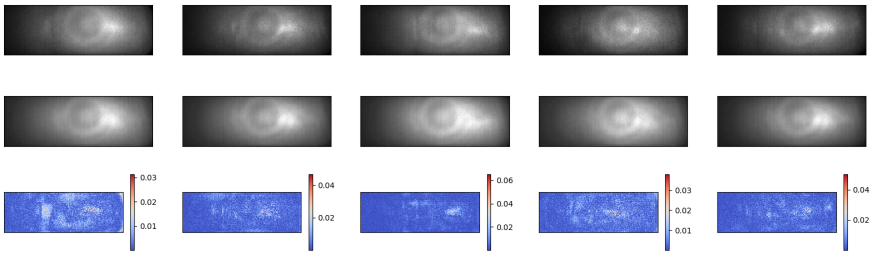

**Figure D.1:** Visualizes the reconstructed LSCI images (middle) with the lowest reconstruction errors in addition to the original images (top) and the pixel-wise absolute difference (bottom). **Left:** Weighted MSE, **Right:** MSE

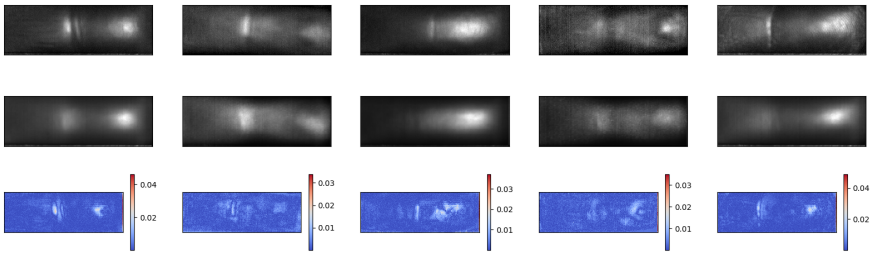

**Figure D.2:** Visualizes the reconstructed SWIR images (middle) with the lowest reconstruction errors in addition to the original images (top) and the pixel-wise absolute difference (bottom). **Left:** Weighted MSE, **Right:** MSE

## E. Score-Fusion: SWIR (4D) vs. LSCI (3D)

Figure E.1 shows DET-curves, each representing a different weight setting for the score fusion between LSCI and SWIR model. The first scores stem from a Dense-AE model that is trained on SWIR data, including four wavelengths (1200 nm, 1300 nm, 1450 nm, 1550 nm). These scores are fused with those of a Dense-AE that is trained on LSCI data with three input dimensions (first, middle, and last). The according pAUC values are depicted in table E.1.

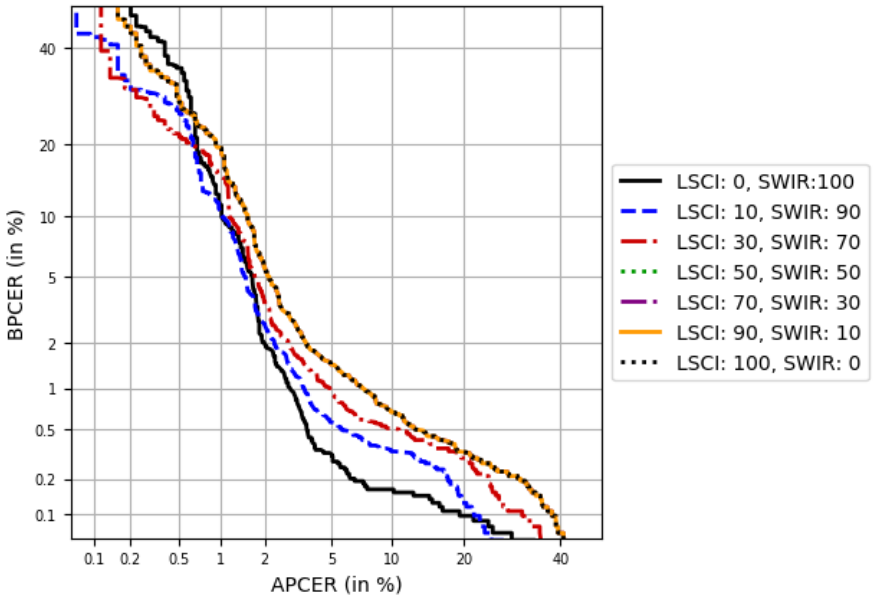


**Figure E.1:** DET-Curves after fusing the scores of Dense-AE Models, which are trained with the weighted MSE and multiple input dimensions on both LSCI and SWIR dataset

| LSCI weight (%) | SWIR weight (%) | pAUC (%) |
|---|---|---|
| 0.0 | 100.0 | **7.30** |
| 10.0 | 90.0 | 7.84 |
| 30.0 | 70.0 | 8.36 |
| 50.0 | 50.0 | 11.97 |
| 70.0 | 30.0 | 15.11 |
| 90.0 | 10.0 | 19.80 |
| 100.0 | 0.0 | 23.07 |

**Table E.1:** pAUC values, measured in a range between 0-20% of the Dense-AE, trained with multi-dimensional inputs, for multiple score weights

# List of Figures

# List of Tables

# References


[1] M. Abadi et al. "Tensorflow: A system for large-scale machine learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation OSDI 16)*. 2016, pp. 265–283.

[2] H. H. Aghdam and E. J. Heravi. *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classifcation*. Springer, 2017.

[3] Z. Akhtar, C. Micheloni, and G. L. Foresti. "Biometric liveness detection: Challenges and research opportunities". In: *IEEE Security & Privacy* 13.5 (2015), pp. 63–72.

[4] A. Antonelli et al. "Fake finger detection by skin distortion analysis". In: *IEEE Transactions on Information Forensics and Security* 1.3 (2006), pp. 360–373.

[5] J. O. Berger. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

[6] L. Biel et al. "ECG analysis: a new approach in human identification". In: *IEEE Transactions on Instrumentation and Measurement* 50.3 (2001), pp. 808–812.

[7] da/sec Biometrics and I. S. R. Group. *BATL: Biometric Authentication with a Timeless Learner*. `https://dasec.h-da.de/projects/batl/`. Accessed: 2020-04-06.

[8] J. Brownlee. *What is the Difference Between Test and Validation Datasets?* `https://machinelearningmastery.com/difference-test-validation-datasets`. Accessed: 2020-02-09.

[9] E. Comission. *Entry/Exit System*. `https://ec.europa.eu/home-affairs/what-we-do/policies/borders-and-visas/smart-borders/ees_en`. Accessed: 2020-04-19.

[10] E. Comission. *Identification of applicants (EURODAC)*. `https://ec.europa.eu/home-affairs/what-we-do/policies/asylum/identification-of-applicants_en`. Accessed: 2020-04-19.

[11] E. Comission. *Smart Borders*. `https://ec.europa.eu/home-affairs/what-we-do/policies/borders-and-visas/smart-borders_en`. Accessed: 2019-10-22.

[12] C. C. Cooksey, B. K. Tsai, and D. W. Allen. "A collection and statistical analysis of skin reflectance signatures for inherent variability over the 250 nm to 2500 nm spectral range". In: *Active and passive signatures V*. Vol. 9082. International Society for Optics and Photonics. 2014, p. 908206.

[13] G. Csurka et al. "Visual categorization with bags of keypoints". In: *Workshop on statistical learning in computer vision, ECCV*. Vol. 1. 1-22. Prague. 2004, pp. 1–2.

[14] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 886–893.

[15] L. Daly. *Identity Theft and Credit Card Fraud Statistics for 2020*. `https://www.fool.com/the-ascent/research/identity-theft-credit-card-fraud-statistics/`. Accessed: 2020-04-19.

[16] J. Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[17] M. Desphande. *Perceptrons: The First Neural Networks*. `https://python-machinelearning.pro/perceptrons-the-first-neural-networks`. Accessed: 2020-04-14.

[18] Y. Ding and A. Ross. "An Ensemble of One-Class SVMs for Fingerprint Spoof Detection Across Different Fabrication Materials". In: 2016.

[19] M. Espinoza and C. Champod. "Using the number of pores on fingerprint images to detect spoofing attacks". In: *2011 International Conference on Hand-Based Biometrics*. IEEE. 2011, pp. 1–5.

[20] European Parliament. *Council Regulation (EC) No 2252/2004 of 13 December 2004 on standards for security features and biometrics in passports and travel documents issued by Member States*. 2004.

[21] C. François. *Deep learning with Python*. Manning Publications Company, 2017.

[22] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

[23] K. Fukushima. "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position". In: 1980.

[24] A. George et al. "Biometric Face Presentation Attack Detection with Multi-Channel Convolutional Neural Network". In: *IEEE Transactions on Information Forensics and Security* (2019).

[25] M. Gomez-Barrero and C. Busch. "Multi-Spectral Convolutional Neural Networks for Biometric Presentation Attack Detection". In: *Proc. Norwegian Information Security Conference (NISK)*. 2019.

[26] M. Gomez-Barrero, J. Kolberg, and C. Busch. "Multi-Modal Fingerprint Presentation Attack Detection: Analysing the Surface and the Inside". In: *2019 International Conference on Biometrics (ICB)*. IEEE. 2019, pp. 1–8.

[27] M. Gomez-Barrero, J. Kolberg, and C. Busch. "Towards multi-modal finger presentation attack detection". In: *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE. 2018, pp. 547–552.

[28] L. J. González-Soler et al. "Fingerprint Presentation Attack Detection Based on Local Features Encoding for Unknown Attacks". In: 2019.

[29] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[30] Y. Han et al. "A study on evaluating the uniqueness of fingerprints using statistical analysis". In: *International Conference on Information Security and Cryptology*. Springer. 2004, pp. 467–477.

[31] M. Hussein et al. "Fingerprint Presentation Attack Detection Using A Novel Multi-Spectral Capture Device and Patch-Based Convolutional Neural Networks". In: *2018 IEEE Int. Workshop on Information Forensics and Security (WIFS)*. 2018.

[32] Y. Ishii and M. Takanashi. "Low-cost Unsupervised Outlier Detection by Autoencoders with Robust Estimation". In: *Journal of Information Processing* 27 (2019), pp. 335–339.

[33] ISO/IEC JTC1 SC37 Biometrics. *ISO/IEC 2382-37. Information Technology - Vocabulary - Part 37: Biometrics*. 2017.

[34] ISO/IEC JTC1 SC37 Biometrics. *ISO/IEC 24745:2011. Information Technology - Security techniques - Biometric Information protection*. 2011.

[35] ISO/IEC JTC1 SC37 Biometrics. *ISO/IEC 30107-1. Information Technology - Biometric presentation attack detection - Part 1: Framework*. 2016.

[36] ISO/IEC JTC1 SC37 Biometrics. *ISO/IEC 30107-3. Information Technology - Biometric presentation attack detection - Part 3: Testing and Reporting*. 2017.

[37] G. James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[38] H. Jegou et al. "Aggregating local image descriptors into compact codes". In: *IEEE transactions on pattern analysis and machine intelligence* 34.9 (2011), pp. 1704–1716.

[39] J. Kannala and E. Rahtu. "Bsif: Binarized statistical image features". In: *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*. IEEE. 2012, pp. 1363–1366.

[40] M. Ke, C. Lin, and Q. Huang. "Anomaly detection of Logo images in the mobile phone using convolutional autoencoder". In: *2017 4th International Conference on Systems and Informatics (ICSAI)*. IEEE. 2017, pp. 1163–1168.

[41] P. Keilbach et al. "Fingerprint presentation attack detection using laser speckle contrast imaging". In: *2018 International Conference of the Biometrics Special Interest Group (BIOSIG)*. IEEE. 2018, pp. 1–6.

[42] *Keras - Usage of loss functions*. `https://keras.io/losses/`. Accessed: 2020-03-21.

[43] *Keras Documentation*. `https://keras.io/`. Accessed: 2020-02-07.

[44] R. Kohavi. "A study of cross-validation and bootstrap for accuracy estimation and model selection". In: *Ijcai*. Vol. 14. 2. Montreal, Canada. 1995, pp. 1137–1145.

[45] P. D. Lapsley et al. *Anti-fraud biometric scanner that accurately detects blood flow*. US Patent 5,737,439. 1998.

[46] T. Lindeberg. "Scale invariant feature transform". In: (2012).

[47] Z. Liu et al. "Large-scale celebfaces attributes (celeba) dataset". In: *Retrieved August* 15 (2018), p. 2018.

[48] L. van der Maaten and G. Hinton. "Visualizing Data using t-SNE". In: 2008.

[49] D. B. Marques et al. "Recent developments on statistical and neural network tools focusing on biodiesel quality". In: *International Journal of Computer Science and Application* 3.3 (2014), pp. 97–110.

[50] A. Martin et al. *The DET curve in assessment of detection task performance*. Tech. rep. National Inst of Standards and Technology Gaithersburg MD, 1997.

[51] V. Mura et al. "LivDet 2015 Fingerprint Liveness Detection Competition 2015". In: *7th Intl. Conf. on Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE*. IEEE. 2015, pp. 1–6.

[52] O. Nikisins, A. George, and S. Marcel. "Domain Adaptation in Multi-Channel Autoencoder based Features for Robust Face Anti-Spoofing". In: 2019.

[53] T. Ojala, M. Pietikäinen, and D. Harwood. "A comparative study of texture measures with classification based on featured distributions". In: *Pattern recognition* 29.1 (1996), pp. 51–59.

[54] M. Oquab et al. “Learning and transferring mid-level image representations using convolutional neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1717–1724.

[55] G. Orrù et al. “Livdet in action-fingerprint liveness detection competition 2019”. In: *arXiv preprint arXiv:1905.00639* (2019).

[56] D. W. Osten et al. *Biometric, personal authentication system*. US Patent 5,719,950. 1998.

[57] O. M. Parkhi, A. Vedaldi, and A. Zisserman. “Deep Face Recognition”. In: *British Machine Vision Association* (2015).

[58] N. Ratha and R. Bolle. *Automatic fingerprint recognition systems*. Springer Science & Business Media, 2003.

[59] A. Rattani and A. Ross. “Minimizing the Impact of Spoof Fabrication Material on Fingerprint Liveness Detector”. In: 2014.

[60] P. J. Rousseeuw and M. Hubert. “Robust statistics for outlier detection”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1.1 (2011), pp. 73–79.

[61] J. Sánchez et al. “Image classification with the fisher vector: Theory and practice”. In: *International journal of computer vision* 105.3 (2013), pp. 222–245.

[62] B. Schölkopf et al. “Support vector method for novelty detection”. In: *Advances in neural information processing systems*. 2000, pp. 582–588.

[63] *Scikit-Learn - Machine Learning in Python*. `https://scikit-learn.org/stable/`. Accessed: 2020-04-17.

[64] *Scikit-Learn OneClass SVM Documentation*. `https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html`. Accessed: 2020-04-14.

[65] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-scale Image Recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[66] S. Skansi. *Autoencoders. In: Introduction to Deep Learning. Undergraduate Topics in Computer Science*. Springer, Cham, 2018.

[67] J. T. Springenberg et al. “Striving for simplicity: The all convolutional net”. In: *arXiv preprint arXiv:1412.6806* (2014).

[68] H. Steiner et al. “Design of an active multispectral SWIR camera system for skin detection and face verification”. In: *Journal of Sensors* 2016 (2016).

[69] L. Strothmann, U. Rascher, and R. Roscher. "Detection of Anomalous Grapevine Berries Using All-Convolutional Autoencoders". In: *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2019, pp. 3701–3704.

[70] B. Tan and S. C. Schuckers. "New approach for liveness detection in fingerprint scanners based on valley noise analysis". In: *Journal of Electronic Imaging* 17.1 (2008), p. 011009.

[71] *Tensorflow*. `https://www.tensorflow.org/`. Accessed: 2020-04-17.

[72] R. Tolosana et al. "Biometric Presentation Attack Detection: Beyond the Visible Spectrum". In: IEEE. 2019.

[73] R. Tolosana et al. "Towards fingerprint presentation attack detection based on convolutional neural networks and short wave infrared imaging". In: *2018 International Conference of the Biometrics Special Interest Group (BIOSIG)*. IEEE. 2018, pp. 1–5.

[74] P. G. Vaz et al. "Laser speckle imaging to monitor microvascular blood flow: a review". In: *IEEE reviews in biomedical engineering* 9 (2016), pp. 106–120.

[75] J.-P. Vert, K. Tsuda, and B. Schölkopf. "A primer on kernel methods". In: *Kernel methods in computational biology* 47 (2004), pp. 35–70.

[76] P. Yadav. *Deeper Understanding of Neural Networks - Part 1: CNNs*. `https://towardsdatascience.com/a-deeper-understanding-of-nnets-part-1-cnns-263a6e3ac61`. Accessed: 2020-04-15.

[77] D. Yambay et al. "LivDet 2011 Fingerprint Liveness Detection Competition 2011". In: *5th IAPR Int. Conf. on Biometrics (ICB), 2012*. IEEE. 2012, pp. 208–215.

[78] J. Yosinski et al. "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.