



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ  
UNIVERSITY OF PIRAEUS

DEPARTMENT OF DIGITAL SYSTEMS



NCSR DEMOKRITOS  
INSTITUTE OF INFORMATICS AND  
TELECOMMUNICATIONS

# Face anti-spoofing detection methods

by

Vasileios Papadopoulos

Submitted

in partial fulfilment of the requirements for the degree of

Master of Artificial Intelligence

at the

UNIVERSITY OF PIRAEUS

July 2022

University of Piraeus, NCSR “Demokritos”. All rights reserved.

Author . . . . .

Vasileios Papadopoulos  
II-MSc “Artificial Intelligence”  
July, 2022

Certified by. . . . .

Michael Filippakis  
Associate Professor, University of Piraeus  
Thesis Supervisor

Certified by. . . . .

Ilias G. Maglogiannis  
Head of the Department, University of Piraeus  
Member of Examination Committee

Certified by. . . . .

Maria Halkidi  
Associate Professor, University of Piraeus  
Member of Examination Committee

## **Acknowledgments**

I would like to thank my external (and first) supervisor, Dr. Giorgio Patrini, for providing me the opportunity to research in the field of media forensics, which at the start of my research project was unbeknownst to me. He has provided constant support and guidance throughout my internship period. My sincere gratitude also goes towards the team at Sensity B.V. for providing me with all the necessary resources and autonomy to smoothly conduct my research project. I would also like to thank Dr. Poulou Marilena for her assistance in supervising the analysis of the data, her contribution to the experimental part of the dissertation and her helpful comments in the research analysis. Finally, I would like to thank my family and friends for providing me moral support during the past year which has been overwhelming, to say the least. This material is based upon work supported by the «Sensity B.V» in The Netherlands. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the «Sensity B.V», the view of University of Piraeus and Inst. of Informatics and Telecom. of NCSR Demokritos.

# Face anti-spoofing detection methods

by

**Vasileios Papadopoulos**

Submitted to the II-MSc “Artificial Intelligence” on  
July, 2022,  
in partial fulfillment of the  
requirements for the MSc degree

## **Abstract**

We have entered an era of misinformation, fake news and impersonation fuelled by Artificial Intelligence (AI). Visual content have been jeopardized by malicious entities in an attempt to fool security systems by pretending someone else’s identity. Such entities, usually seek elevated access to critical infrastructures like online banking, government administration services and any KYC system. A Biometric’s system task, is to deploy security checks and measures to verify someones identity and authenticity(liveness). Given the societal impact of such commodification of impersonation, our research proposes learning-based methods with focus on learning to perform well on a significantly different target distributions a.k.a *Domain Adaptation* and *Domain Generalization*. In Deep Learning (DL), model performance depends heavily on the presence of high variation within training examples and usually, Machine Learning practitioners aim to collect data in such way that will guide model’s ability to *generalize*. However, even with enormous amount of data there is no guarantee that a model would perform equally well to unseen data in the same domain.

In our research, we explore and evaluate different deep and machine learning methods in an attempt to learn discriminative features that could generalize to several academic datasets and datasets *in-the-wild*. We define a classic binary classification problem which is used as a baseline model. We use Deep Convolutional Neural Networks(CNN) and two commonly used backbones, Resnet18 and EfficientNetb4. Then, we transform the task into a multi-task learning with auxiliary fully-connected heads and explore the impact in generalization and adaptation, and use different deep metric losses such *Center Loss* and *Triplet Loss* and *Gradients Orthogonality*. Finally, we investigate hand-crafted features such Histogram of Oriented Gradients(HOG) and Local Binary Patterns(LBP) and train classic machine learning classifier (eg. Support Vector Machine).

Thesis Supervisor: Michael Filippakis  
Title: Associate Professor @ University of Piraeus

# Contents

<b>Acknowledgments</b>	<b>3</b>
<b>1 Introduction</b>	<b>8</b>
1.1 Motivation . . . . .	8
1.2 Contributions . . . . .	9
<b>2 Liveness Detection</b>	<b>10</b>
2.1 Background . . . . .	10
2.2 Types of Presentation Attacks . . . . .	10
2.3 Liveness detection methods with generic devices . . . . .	11
<b>3 Overview of Liveness Detection methods using RGB images</b>	<b>13</b>
3.1 Taxonomy of Liveness Detection methods . . . . .	13
3.2 Liveness cue-based methods . . . . .	13
3.2.1 Motion-based methods . . . . .	13
3.3 Texture-based methods . . . . .	15
3.3.1 Static texture-based methods . . . . .	15
3.3.2 Dynamic texture-based methods . . . . .	16
3.4 3D reconstruction cue-based methods . . . . .	16
3.4.1 3D shape-based methods . . . . .	16
3.4.2 Depth map-based methods . . . . .	16
3.5 Deep learning-based methods . . . . .	17
3.5.1 Few-shot learning-based methods . . . . .	17
3.5.2 Domain adaptation-based methods . . . . .	17
<b>4 Existing Liveness Detection datasets</b>	<b>19</b>
4.1 Face anti-spoofing datasets . . . . .	19
4.2 Definitions . . . . .	19
4.3 Overview of academic and in-the-wild datasets . . . . .	20
4.4 Major limitations . . . . .	21
<b>5 Liveness Detection using hand-crafted features</b>	<b>22</b>
5.1 Local Binary Patterns (LBP) . . . . .	22
5.2 Histogram of Oriented Gradients (HOG) . . . . .	23
5.3 Principal Component Analysis (PCA) . . . . .	24
5.4 SVM classifier . . . . .	24

<b>6</b>	<b>Liveness Detection using Deep Learning</b>	<b>27</b>
6.1	<b>Neural Networks overview</b>	27
6.2	<b>Convolutional Neural Networks (CNN) overview</b>	27
6.2.1	ResNet-18	28
6.2.2	EfficientNet-B4	28
6.2.3	Normalization in Deep Neural Networks	29
6.2.4	Model Interpretability	30
6.3	Domain Generalization and Adaptation	31
6.4	<b>Liveness Detection as binary classification problem</b>	32
6.5	<b>Liveness Detection as multi-task problem</b>	33
6.5.1	Magnitude of Gradients	34
6.5.2	Direction of Gradients	34
6.6	<b>Deep metric learning networks for Liveness Detection</b>	35
6.6.1	Center loss	35
6.6.2	Triplet loss	36
<b>7</b>	<b>Experiments</b>	<b>38</b>
7.1	<b>Methodology</b>	38
7.1.1	Data pre-processing	38
7.1.2	Illumination Annotation	40
7.2	<b>Model Training Experiments</b>	41
7.3	<b>Hardware Specifications</b>	42
7.4	<b>Domain Adaptation and Generalization Results</b>	42
7.4.1	Dataset soup results	44
7.5	<b>Few-shots Learning for DeepFakes</b>	46
<b>8</b>	<b>Conclusion</b>	<b>50</b>
8.1	Future Work	50
	<b>Appendices</b>	<b>55</b>
.1	Data Augmentation	55
.2	Training Process	55
.3	Face cropping	55
.4	Dataset soup	55

# 1 Introduction

The remarkable success of face recognition systems has triggered the development of intelligent interactive applications in wide variety of sectors (eg. mobile-payments, airport check-in, government authentication services, etc). However, face anti-spoofing systems (FAS) are vulnerable to presentation attacks (PA) ranging from printed-photos, video-replays, 3D-masks and most recently by Deep fakes. Therefore, industry and academic community have been increasingly giving extensive attention for developing and securing face recognition systems.

Most traditional presentation attack detection (PAD) algorithms are based on hand-crafted features which require task-related prior knowledge to design. In addition, liveness-cues based methods have been developed to detect head movements(nodding), facial expressions(smiling), eye-blinking, gaze tracking and remote physiological signals. Physiological cues could be used to discriminate against presentation attacks however, they usually captured from long duration videos which make it inconvenient for practical deployment. Furthermore, an attacker could easily mimic them in video plays making them less reliable. Classical hand-crafted descriptors (eg. HOG, SIFT, LBP) are capable of extracting informative patterns for liveness from various color spaces. Hand-crafted descriptors have been researcher's sole focus up until recently, though with the advancements in deep learning and convolutional neural networks(CNN) they have been used in a more hybrid manner.

Most researchers formulate FAS as a binary classification problem and supervised by simple loss function (eg. binary cross entropy). An FAS differs from other face recognition tasks based on its self-evolving nature(eg. recurring attack-vs-defense paradigm which make it more challenging). For example, a binary computer vision task of Gender classification rely on appearance-based semantic cues while intrinsic features (eg. material, paper, screen etc) in FAS are content-independent (eg. not related to facial attributes).

## 1.1 Motivation

The significant rise of digitization yield to the development of deep learning in many different aspects. The advancements in face anti-spoofing systems have been exceptional though, the lack of models' generalization capabilities and the constant battle between *discrimination* and *generalization* is still an open problem. In addition, deep learning models explainability and interpretability is low-to-none making it difficult even for a human eye to correctly distinguish live and spoof images. Various works have been proposed in the literature with impressive results



## 1.2 Contributions

We explore the impact of multi task learning with averaging gradients and gradients orthogonality methods in domain generalization capabilities of the model. We also train kNN classifier with various shots of novel class and benchmark the behavior of different model configurations. The rest of the paper is organized as follow: In section 2 we will dive into the challenges a *Liveness Detector System* faces. In sections 3 and 4 a brief overview of previous methods as well a description of considered datasets will be given. In section 5 we will approach the problem using *classic* computer vision techniques with *feature extraction*. Sectors 6 presents a modern approach to the problem, utilizing deep learning while sector 7 shows experimental results.

## 2 Liveness Detection

### 2.1 Background

Biometric liveness detection refers to the use of computer vision technology and machine learning algorithms to detect the presence of a genuine/living user, rather than a representation (eg. photo, 3D mask, deepfakes). Presentation attack detection (PAD) systems can utilize active or passive detection methods. Typically, it is associated with facial recognition, but liveness could also be applied to voice recognition to distinguish present speakers from audio recordings or palm biometrics such as by detecting blood flow, and even iris recognition. In this work, we will mainly focus on *passive liveness* detection systems where the user is asked to present a single photo to a biometric system for authentication.

Face spoofing attacks on face recognition systems could be divided into two main categories: physical presentation and digital manipulation attacks. Liveness detection systems' main challenges are to ensure physical, digital and semantic integrity. Digital manipulation attacks could be easily generated by either adding, removing or replicating objects. Adding an object from a different image (splicing) or from a different location within the same image (copy-move) are some of the manipulation techniques. In addition, an object could be made to appear to be deleted by extending the background to cover it (inpainting). These manipulation examples where no sophisticated artificial intelligence tools are required are often called cheap-fakes. In this section, we will mainly focus on physical presentation attacks which intend to mislead FAS by presenting faces upon a physical medium in front of image sensors (eg. camera). For the rest of the paper, the terms Presentation Attack Detection and Face Anti-Spoofing will be used interchangeably.

### 2.2 Types of Presentation Attacks

Effectively, there are two types of Presentation Attacks (PAs). First, with the rise of internet and social media, there is an ever-increasing amount of shared photos and/or videos of people's faces. Impostors can easily collect and use them to fool a face authentication system. Such attacks are also called impersonation (spoofing) attacks. Second, another (less studied) type of Presentation Attack is called obfuscation attacks, where a person uses tricks to avoid being recognized by the system (but not necessarily by impersonating a legitimate user's identity).

Common presentation attacks, generally categorized as photo, video replay and 3D mask attacks (Figure 1). On the other hand obfuscation attacks rely on tricks to hide the user’s real identity, such as facial makeup, plastic surgery or face region occlusion. In this work, we focus on *spoof* (impersonation) attacks, where a malicious user might impose directly biometric data from a legitimate user or to create presentation attacks (eg. spoof/fakes) that will be used to mount an attack to face recognition system.

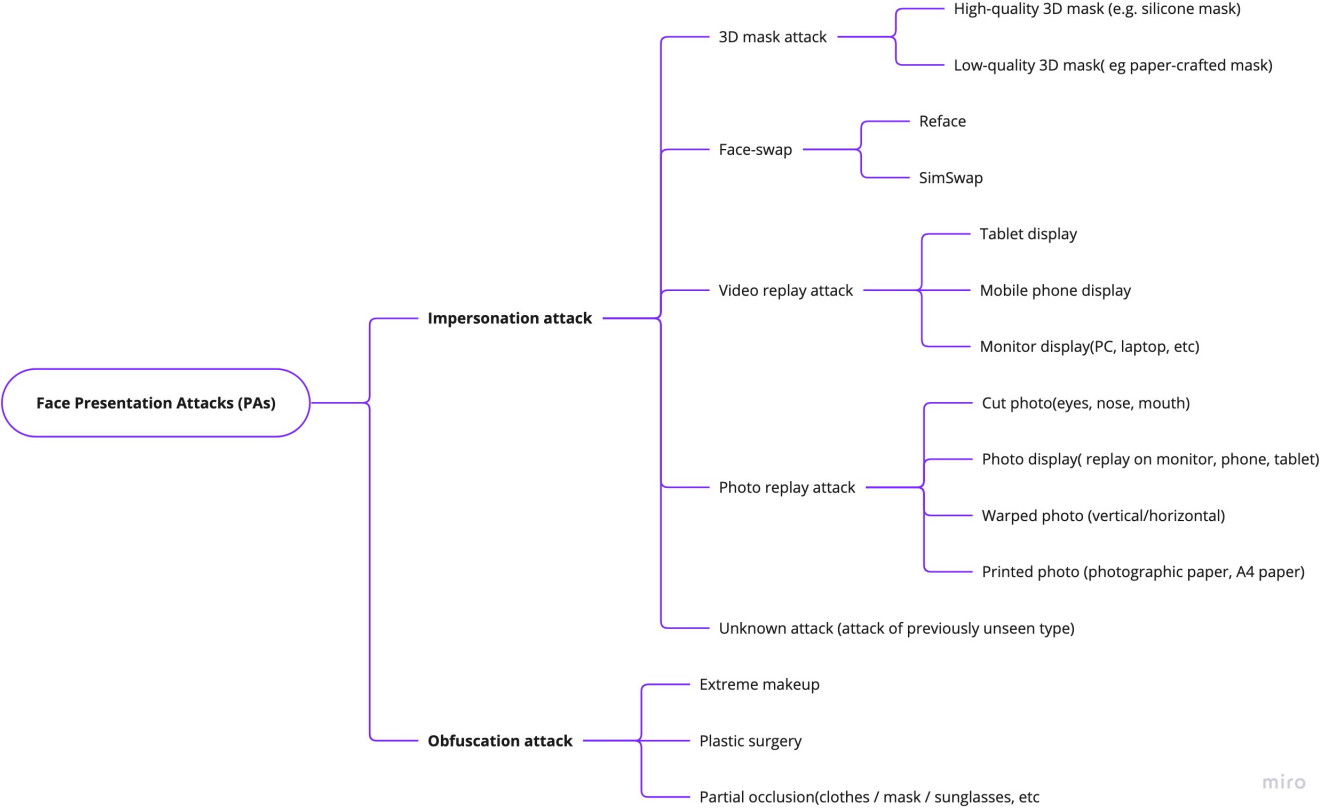


Figure 1: Knowledge map of Face Presentation Attacks

### 2.3 Liveness detection methods with generic devices

Generally, one can distinguish presentation attack detection systems based on specific hardware/sensors used in biometric device and RGB only approaches. The former includes specialized hardware such *thermal, infrared* sensors to facilitate detection. For example, 3D sensors can discriminate between legitimate and malicious 2d planar attempt by reconstructing depth-maps. Infrared sensors can easily detect video replays attacks since electronic devices (eg. monitors, phones) appear to be dark under infrared

illumination. Thermal sensors can give the temperature distribution of real/live faces. Specialized hardware detection usually is employed to specific scenarios where access is restricted (eg. protected premises). The major drawback of such devices is the lack of broadly availability to general public. This is the main reason our work focuses on generic devices based on simple RGB images.

## 3 Overview of Liveness Detection methods using RGB images

### 3.1 Taxonomy of Liveness Detection methods

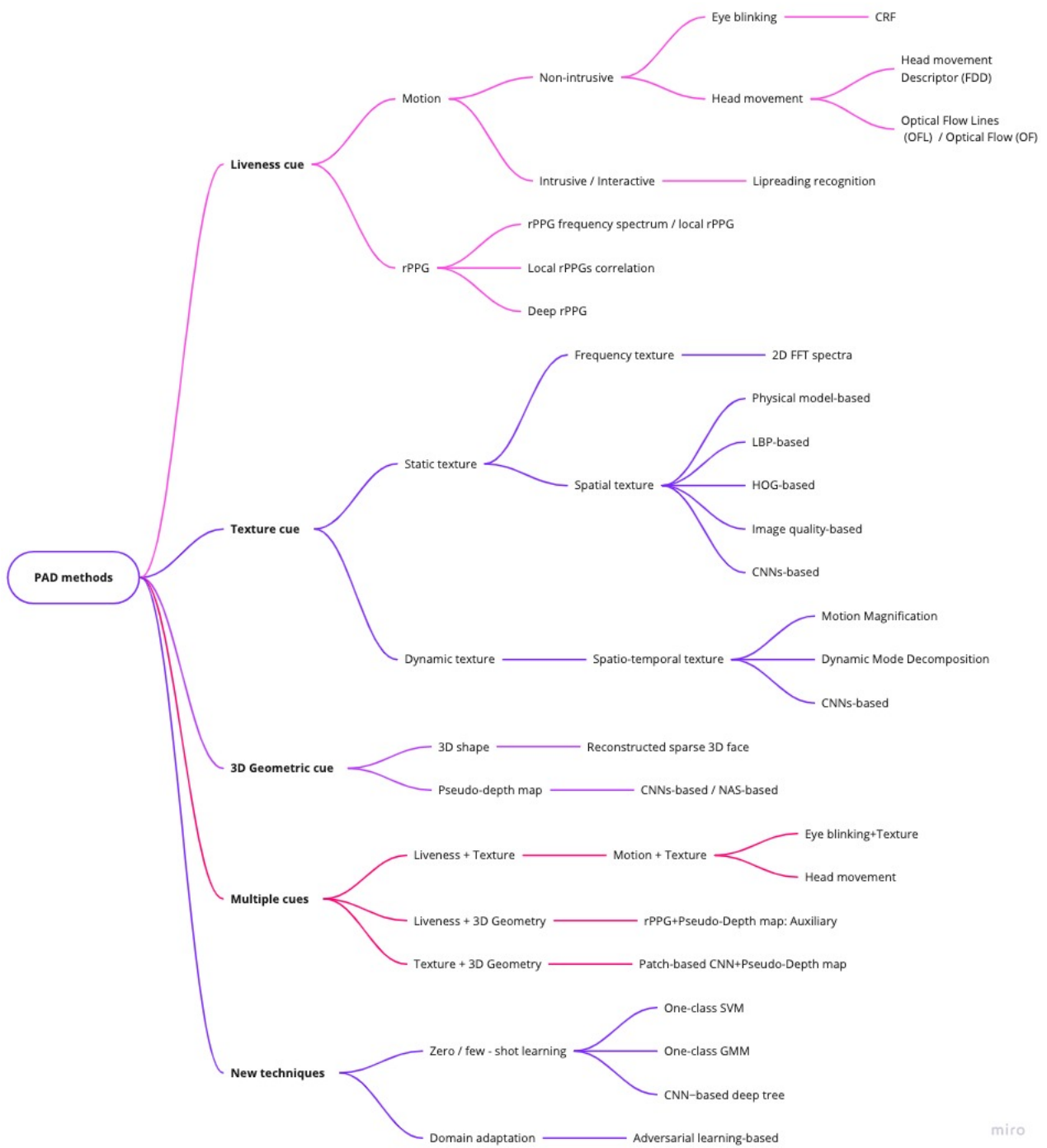
#### 3.2 Liveness cue-based methods

Liveness cue-based methods are the first attempt for detecting spoof faces. They aim to sense any facial motion of the presented subject such as mouth movement, facial expression and eye-blinking. Liveness cue-based methods can be categorized as motion-cue based methods and rPPG-based by detecting blood pressure changes utilizing special hardware.

##### 3.2.1 Motion-based methods

Motion based methods, can successfully detect static presentation attacks (eg. printed photo). However, they are generally vulnerable to video replays that possess liveness information of the subject such head movements, eye blinking and facial expressions. That is why the so called *Active Liveness Detection* was introduced. In such systems, the user is asked to perform a series of specific head movements such *tilting, rotation and mouth movement* in an interactive manner. Active Liveness is capable to detect video replays but they are intrusive to the end user and thus, to the attacker. Therefore, motion based methods are structure into intrusive and non-intrusive.

- **Non intrusive** methods seek for head movements cues and/or expression changes implicitly, without asking the user. Li et al. [36] proposed a methods to temporal changes due to subconscious head movements based on the energy changes of Frequency Dynamic Descriptor. These methods also called Passive Liveness Detection methods.
- **Intrusive** methods are usually based on a Challenge-Response mechanism that requires users to satisfy some requirements. There are based on some predetermined head/face movement (e.g. eyes blinking, head movement in a certain direction, adopting a given facial expression or uttering a certain sequence of word).



miro

Figure 2: Helicopter view of face presentation attack methods.

### 3.3 Texture-based methods

Texture feature-based techniques have been used extensively for face anti-spoofing systems. They are inherently non-intrusive and able to perceive with relative high accuracy photo-based and video-replay attacks. Texture-based methods rely on extracting discriminative features and train a machine learning classifier and usually is formulated as a binary classification problem. Texture cue-based methods could be further divided into static and dynamic texture-based methods. The former extracts spatial or frequential features from a single image while in contrast, dynamic texture-based methods explore temporal features extracted from video sequences. The next two sub-sections present the most prominent approaches from these two types.

#### 3.3.1 Static texture-based methods

Static texture-based classification methods, is a widely used technique to design presentation attacks detection systems. It mostly involves the extraction of carefully thought features and classic machine learning algorithms such SVM. The work on [36] analyzed the difference of light reflectivity between genuine and printed photos using the 2D Fourier spectra. It has shown that, live faces have much more high frequency components than the 2D spectrum of printed photos. This method works best with low resolution photos(124x84mm). [21] et al. used a physical model to analyze micro-textures with Bidirectional Reflectance Distribution Functions (BRDF). An SVM classifier was trained to discriminate between real and planar presentation attacks (eg. printed photos, video-replays). Another commonly used method for face-related tasks, is the Local Binary Pattern. [22] et al. proposed to apply three different LBPs on a normalized 64x64 image. First LPB is extracted from 8 pixel-cell with 2 pixel-radius, the second with 16 pixel-cell and 2 pixel-radius and third with 8 pixel-cell and 1 pixel-radius. The concatenation formed a 833-bin histogram and fed into a non-linear SVM classifier(RBF). More recently, deep learning-based methods are designed to learn texture features as opposed to hand-crafted features extraction. [3] et al. in 2014 was the first work to introduce CNNs for face anti-spoofing detection. They used AlexNet backbone architecture and replaced the last 1000-classes fully connected layer with a binary SVM. Unlikely to recent deep learning models, this approach was not an end-to-end framework though, it shown superiority compared to conventional hand-crafted features extraction methods. Finally, George et al. [4] in 2019 proposed Deep Pixel-wise Binary Supervision (DeepPixBiS) model based on DenseNet architecture. The work includes an additional pixel-wise binary cross-entropy loss  $L_2$  to binary cross-entropy loss.  $L_2$  is based on last feature-map where each pixel is annotated with 0 for live images and 1 from spoof. This way the network was forced to learn patch-wise features. DeepPixBiS have showed promising performance for both photo and video replay attacks.

### 3.3.2 Dynamic texture-based methods

## 3.4 3D reconstruction cue-based methods

3D geometric cue-based methods for presentation attack detection systems utilize 3D geometric features. A *live* face has a 3D structure which is characteristic and distinctive as opposed to a face projected into a 2D planar (eg. photo, a4). 3D face reconstruction from a 2D RGB image and estimation of depth map (i.e distance of each pixel from the camera) are two widely used methods. In the next two sections we will discuss the approaches based on these two cues.

### 3.4.1 3D shape-based methods

Wang *et al.* [1] trained an SVM classifier with input features which were extracted from the 3D reconstruction of 2D facial landmarks using multiple viewpoints. Those reconstructed structures are distinctively different between a live image and an image portrayed on 2D planar (eg. photo replay). A major drawback of this approach is that it requires multiple images from different angles (viewpoints) and could not be effectively trained with a single image. Furthermore, detecting facial landmarks is a challenge on its own and there are lots of inaccuracies.

### 3.4.2 Depth map-based methods

The significant progress in the field of computer vision made it possible to get good *depth estimation maps* using only RGB images, without the need of specialized hardware (eg 3D sensors). A depth map holds the estimated distances of each pixel from the capturing device. Depth maps reconstructed from RGB images are also called *Pseudo-Depth maps*. Atoum *et al.* [2] paper was the first work to propose discrimination methods of live and spoof images(eg. photo replays) based on depth-map. A live/actual face produces depth maps with varying height values while printed faces result on constant maps. The work proposed a CNN architecture with 11 fully connected layers to estimate the depth map of an image. Dataset annotation was performed using state of the art algorithm ([5], [6], [7]) for live images while 2D planar based presentation attacks set to zero. Then an SVM classifier trained with depth maps as input features. Wang *et al.* 2018, extended this approach to videos by proposing Face Anti-Spoofing Temporal-Depth networks (FAS-TD). FAS-TD networks are capable of capturing depth information and motion within a frame sequence(video). Optical Flow guided Feature Block (OFFB) and Convolution Gated Recurrent Units (ConvGRU) modules added to a depth-supervised neural network. FAS-TD can well capture short-term and long-term motion patterns of real faces and planar PAs. The proposed FAS-TD further improved the performance of the depth-map based PAD methods using a single frame as [2], [8]



and achieved state-of-the-art performances. Pseudo-map approach is very effective for detecting 2D planar presentation attacks though, in recent years more sophisticated attacks emerged (eg. 3D Masks) thus making them vulnerable to detect.

## 3.5 Deep learning-based methods

### 3.5.1 Few-shot learning-based methods

Due to recent advancements in state of the art PAD models, they have shown promising results in intra-dataset evaluation on existing publicly available datasets[x]. In practice, generalization capabilities still remain weak as it is very likely cross-datasets(not included in training process) to contain *unseen* or *under-represented* presentation attacks during training phase. Such scenarios make PAD still a challenging problem. Unlike to other computer vision tasks, (eg. face recognition) where collecting a large number of training examples is relatively an issue easy job, PAD datasets are much harder to find. Re-captured spoofing artifacts by a biometric system is rarely available on the internet. Therefore, research groups actively looking for a solution that leverages *zero-few show learning* to address previously unseen spoof types. Liu et al. [24] defined this problem as *Zero-Shot Face Anti-spoofing(ZSFA)*.

Arashloo et al. [25] look at this problem from the perspective of anomaly-detection problem. Real/Live faces represented the positive class and OneClassClassifier SVM was trained. Similarly, Nikisins et al. [?] used one class Gaussian Mixture Model(GMM) but contrary to [25] use a mixture of 3 available datasets, *replay-attack*[27], *replay-mobile*[27] and *MSU MFSD*[32]. The previous two methods, considered only live-class in one-class-classifier training. Though, spoof types(PA) could also contain valuable information to previously unseen attacks. The work in [24] proposed a Deep Tree CNN Network(DTN) where 13 known spoof-types analysed and clustered into 8 semantic subgroups using unsupervised tree learning. The resulted subgroups are used as 8 leaf nodes of DTN. Then, Tree Routing Unit (TRU) is learned to direct the known spoof-types to the appropriate tree leaf based on the features learned by the tree nodes. Convolutional Residual Unit (CRU). In each leaf node, a Supervised Feature Learning (SFL) module is attached and consists of a binary classifier and mask estimator. Unseen attacks can be discriminated based on estimated masked and the output of softmax classifier.

### 3.5.2 Domain adaptation-based methods

As mentioned previously, generalization ability is the greatest challenge in the field nowadays. Pereira et al. [33], intuitively collected and aggregated multiple available datasets. Then, first strategy was to train a CNN model with the aggregation of data. Secondly, was to use a score-fusion-based framework, in which each dataset was trained

separately and the sum of the normalized score of each model was used as the final prediction. However, even with the *dataset-soup* approach it is impossible to collect data from every possible device in all possible capture environments(eg. indoor, outdoor, illumination conditions). In contrast, even if the same semantic presentation attacks eg. printed-photo,video-replay can differ greatly between source and target domains, they are all based on the same physical material (eg. A4, screen). Thus, if there exists a shared feature space between source and target domains, then *domain-adaptation* can be applied.

Shao et al. [34], applied domain adaptation with adversarial learning to address generalization ability.  $N$  discriminators for  $N$  specific domains were trained, to help *generator's* feature space to learn generalized features for each domain. Triplet Loss was used to further discriminate features for both *intra-domain* and *inter-domain* (different datasets). This approach, achieves better results compared to other methods (eg. LBP-TOP [35] when increasing the number of source domains.

## 4 Existing Liveness Detection datasets

### 4.1 Face anti-spoofing datasets

Presentation attack detection datasets usually consists of two type of sets and they come either in videos or photos format. The first set, labelled *Live* contains documents of authentic faces of genuine users. The second set is labelled as *Spoof* and contains photos or videos of presentation attacks (spooftypes) (eg. 3D mask, printed photo, video replay).

A biometric system camera in any real-life application is generally the same device as end-user's uses to capture either a genuine or an impersonation face.



Figure 3: CelebASpoof dataset exploration.

### 4.2 Definitions

Face presentation attack detection (anti-spoofing) datasets, usually are structured as two different kinds of documents (files), in the form of videos or images(photos). The set of *live/genuine faces* that contains photos or videos of the genuine users and the set of presentation attack files (PA), containing photos or videos of printed photos, video replays, 3D mask, etc.

### 4.3 Overview of academic and in-the-wild datasets

In our research, we will mainly be focusing on 5 academic liveness detections datasets and two custom datasets which collected and annotated by developers in Sensity B.V. This work focuses on domain generalization thus, we chose CelebASpoof dataset as source domain. CelebASpoof consists of 625537 images from 10177 subjects where live images selected from *CelebA* [28] dataset while the authors collected and annotated the spoof counter parts. To assess generalization capabilities to target domains we use 4 additional datasets which contain a subset of 10 spoof types of source domain set. *Replay-Attack* [29] consists of 1300 video-clips of photos and video attack from 50 clients in different illumination conditions. Similarly, *Replay-Mobile* [27] consists of 1190 videos and photos from 40 clients. *SiW* [30] dataset is equipped with live and spoof videos from 165 subjects. For each subject, there are 8 live and up to 20 spoof videos, in total 4478 videos. All videos are in 30 fps, about 15 second length, and 1080P HD resolution. The live videos are collected in four sessions with variations of distance, pose, illumination and expression. The spoof videos are collected with several attacks such as printed paper and replay.

Dataset	Ethnicity/Gender	Spoof types	Subjects Images Videos
Replay-Attack	Caucasian 76% Asian 22% African 2%	Printed photos Photo display Video replay	50/200/1000
Replay-Mobile	Unknown	Video replay	50/-/1190
SiW	Caucasian 76% Asian 22% African 2%	Printed photos 4x video replays	165/1320/3300
Eyeblink8	Female 50% Male 50%	Live only 3D Mask Phone Pad PC	4/-/8
CelebASpoof	Female 52% Male 48%	Region Mask Upper Body Mask Face Mask A4 Poster Photo	10177/625537/-
Dev-set-v1	Male 100%	Video replay	5/2462/60
Dev-set-v2	Female 65% Male 35%	Printed photos Photo display Video replay	127/124/-

Table 1: High level comparison of existing academic datasets of face anti-spoofing. Last two rows are collected datasets.

Finally, we include *Eyeblink8* [31] which consists of only 8 videos with persons acting normally in front of the camera. Figures 19 and 20 show examples of face crops and crop sizes distribution for CelebASpoof, Eyeblink8, Replay-Mobile, Replay-Attack and SiW datasets. *Devs-set-v1* and *Devs-set-v2* are custom datasets. V1 consists of photo and video replays from 4 persons under different illumination conditions (sometimes extreme cases) with total 2502 frames. V2 is collected from newspapers, magazines and people from Youtube and contains spoof types such, printed-photo, screen-photo replay (mobile, tablet, screen). It consists of 128 images. An overview of datasets and be found on Table 1.

#### 4.4 Major limitations

Given the acquisition challenges for Liveness detection systems mentioned above, the existing task-related datasets are (relatively to other computer vision problems) still limited not only in terms of volume, but also in terms of diversity regarding the types of presentation attacks and 2D planar surface used to project genuine faces, acquisition devices used to capture real faces, etc. In particular, as of today, there is still no public large-scale datasets for Presentation Attack Detection in the wild, whereas there are several for other fields (eg. face recognition). This hampers the development of PAD systems which are still well below the requirements imposed by most real world applications. It partly explains why other face related problems have had rapid success. The biggest challenge lies in generalization ability of trained models. The latter, there is a consensus among researchers and teams that it could only be achieved with large diversity datasets, including hand-crafted features learning as well as deep learning approaches. An ideal dataset is considered a dataset **in-the-wild** this enormous amount of diversity.

## 5 Liveness Detection using hand-crafted features

### 5.1 Local Binary Patterns (LBP)

Local Binary Pattern is a type of descriptor used in computer vision. It was proposed in 1994 by T. Ojala et al. [10] and used in texture classification tasks. It was mainly designed for gray-scale but it can be also extended to RGB images. The local binary pattern operator transforms an image into an array or image of integer labels describing small-scale (local) textures. Their statistical analysis or the labels directly can be fed as input features to train a machine learning model. It is assumed that a texture has two complementary local aspects, a pattern and its strength. The operator works in sliding windows (blocks) where the middle pixel is used as a threshold to change the value of neighboring pixels.

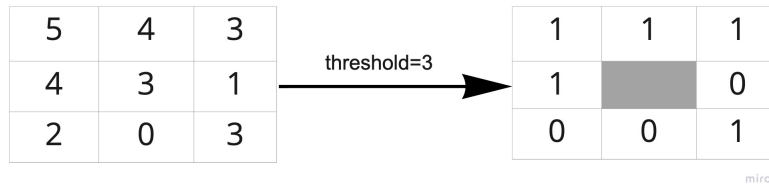


Figure 4: LBP thresholding operation

All pixels in each block:

- Thresholded by its center pixel value and then converted to decimal number.
- All transformed neighboring pixels are summed up to obtain a label for the center pixel.

Formally it is defined as:

$$LBP_{R,P} = \sum_{p=0}^{P-1} s(g_p - g_c) 2^p \quad (1)$$

where  $s$  is:

$$s(x) = \begin{cases} 0, & x < 0 \\ 1, & otherwise \end{cases} \quad (2)$$

and  $g_p$  denotes neighboring pixels in each block with center pixel value of  $g_c$ .  $p$  in the number of sampling points (eg.  $p = 0, 1, \dots, 7$  for a  $3 \times 3$  cell size.  $R$  is the radius of operation.

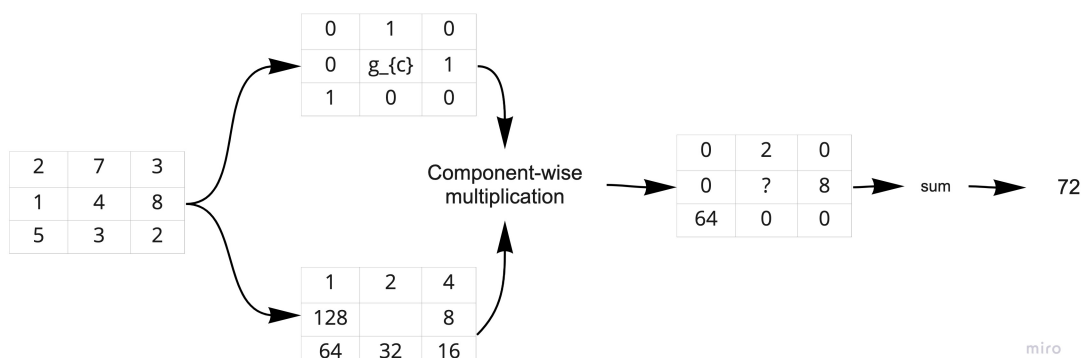


Figure 5: Example of LBP operation.

A local binary pattern is called *uniform* if its uniformity measure is at most 2. Uniformity measure  $U$  (*pattern* is the number of bitwise transitions from 0 to 1 and vice versa. Most of local patterns in natural images are uniform. Ojala et al. [1] noticed that in facial images, uLBP accounts for 90. of patterns with block size 8 and radius 1 and 85. with block size 8 and radius 2. In addition to LBP, Local Phase Quantization (LPQ) has been proposed. A common issue in computer vision tasks is the degradation of image quality cause of blurriness. Blur can be caused by misfocused optics or slight motion of camera while capturing. LPQ, quantize local phase information of Fourier transformations. LPB is image rotation invariant while LPQ is also blur invariant.

## 5.2 Histogram of Oriented Gradients (HOG)

Histogram of Oriented Gradients is a powerful feature descriptor for object detection. It was proposed in 1986 by McConnell of Wayland Research Inc. HOG computes pixel-wise gradients and orientations and stores them in a histogram. The method simplifies the representation of the image and discards any non crucial information. In other words, it minimizes the noise. As a pre-processing step, all images are resized to the same shape and *HOG cell size* is defined. Usually the cell is either  $8 \times 8$  or  $16 \times 16$  depending on input image size. The image then is split into the various cells in vertical and horizontal directions. For each cell the gradient is calculated in each axis  $(x,y)$ ,  $g_x$ ,  $g_y$

### 5.3 Principal Component Analysis (PCA)

Principal Component Analysis is a classic dimensionality reduction technique which is used to get deep inside of data in fields of probabilities and statistics. It is able to uncover lower dimensional patterns of large data of certain statistical distribution and helps train more robust models. PCA represent the statistical variations in the data based on hierarchical coordinate system. The directions in the coordinate system capture the maximum amount of variance in data. The goal is to find the most dominant combinations of features that describe as much of the data as possible. PCA finds the best fitting line by maximizing the sum of squared distances from the projected points to the origin.

### 5.4 SVM classifier

Support Vector Machines (SVM), is a classification algorithm where is it’s goal is to separate points of two or multiple classes using a line. Typically, in multi dimension planes SVM finds a *hyperplane* where points are separable. SVMs are *maximum margin classification algorithms* and use Hinge loss.

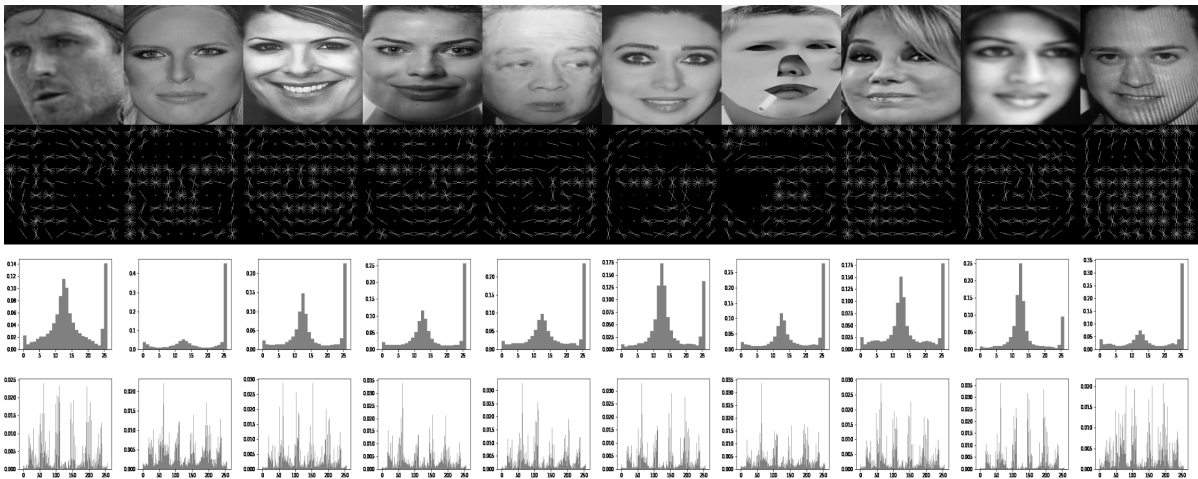


Figure 6: Example of extracted features. First row is the original image, second row HOG descriptor and rows 3 and 4 LBP and LPQ respectively.

Our first approach to discriminate live and presentation attacks is to extract HOG, LPB, LPQ features from a subset of source domain train-dataset and feed into an SVM classifier. We use 5K live images and 500 for each of the 10 spoof types (total 10K) as training set. For the test set we use 1K and 100 images respectively (total 2K). We train SVM with each feature descriptor separately using a linear kernel. In addition, we concatenate all descriptors and perform dimensionality reduction with PCA preserving



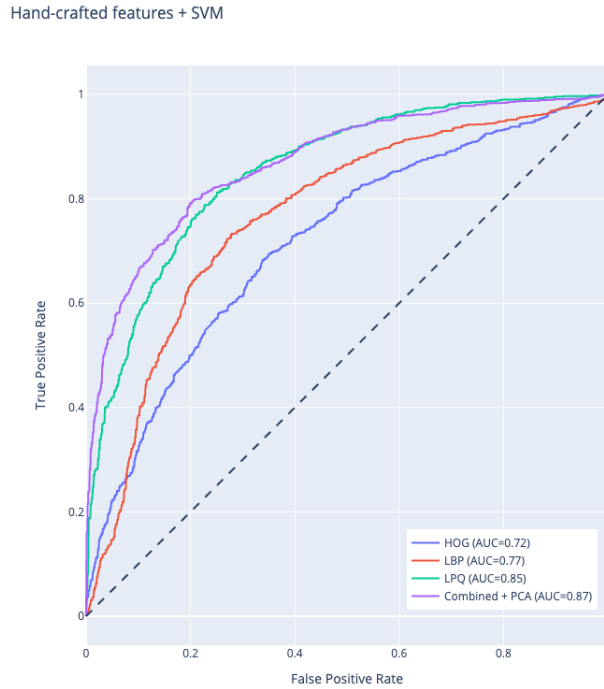


Figure 7: Receiver Operating Characteristic curves from hand-crafted features.

95% of original variation and train again. We perform hyper-parameter tuning for parameter  $\mathbf{C}$  which instructs the optimizer of the trade off of correctly classified training examples against maximization of the decision function’s margin. Each feature descriptor is adjusted to 512 dimensions. LPQ gives the best equal error rate (EER) in single feature extraction experiment (22.4%) while the best overall EER is achieved with features concatenation + PCA (20.5%) Table 2 summarizes the results. It is worth mentioning that even a 32GB of ram is insufficient to accommodate large datasets, thus we restricted learning process in terms of samples. Also, given the inherit limitations of SVM on handling large data, we do not assess the performance of the model in target domains since there is a considerable gap on test set evaluation of source domain.

Descriptor	Threshold	EER(%)	TPR(%)	AUC
HOG	0.42	33.3%	66.7(%)	0.718
LBP	0.39	27.5%	72.5%	0.768
LPQ	0.40	22.4%	77.6%	0.855
Combined + PCA	0.32	20.5%	79.5%	0.87

Table 2: Classification results with hand-crafted features and SVM Linear kernel.

In the next section we will present modern methods for Liveness Detection. Specif-

ically, we will examine multi-task learning and different loss functions to assess the generalization capabilities of the models.

## 6 Liveness Detection using Deep Learning

### 6.1 Neural Networks overview

An Artificial Neural Networks (ANN) can be seen as an algorithmic approach to learning.

**Definition:** An *artificial neuron* with *weights*  $w_1, \dots, w_n \in R$ , *bias*  $b \in R$  and *activation function*  $\sigma$  : is defined as the function  $f: R^n \rightarrow R$  given by

$$f(x_1, \dots, x_n) = \sigma \left( \sum x_i w_i + b_i \right) \quad (3)$$

Concatenating multiple artificial neurons leads to *composition of affine linear maps and activation functions*. Common Non-linear Activation functions are Sigmoid, ReLU, Heaviside, tanh. A large collection of hierarchical structured artificial neurons form the *Deep Neural Networks*. DNN are function approximation models and based on *Universal Approximation Theorem* (Cybenko 1989, Hornik 1991) they can approximate any *continuous* function with arbitrary accuracy and complexity.

### 6.2 Convolutional Neural Networks (CNN) overview

Convolutional neural network (CNN) is a category of deep neural networks which has found great success in computer vision applications (Krizhevsky et al., 2017). Since the early works by Fukushima (1988) and LeCun et al. (1989), CNNs are now a cutting edge area of research within deep learning. Besides the back propagation-based learning approach, CNNs draw important principles from the field of neuroscience. For example, the feature maps of a CNN can be associated with the spatial maps of the visual cortex V1 of the human brain (Goodfellow et al., 2016).

We provide a brief overview of the fundamental components which contribute towards the basic network architecture used for image classification

- *Convolutional Layer:* This is the crux of a CNN. Every neuron in a convolutional layer performs a linear convolution operation over the input image  $x$  using a convolutional 2-dimensional kernel/filter  $h$ , i.e.  $y = x \square h$ . If the input image has multiple channels, we perform a channel-wise convolution operation.
- *Pooling Layer:* This layer is added in succession of a convolutional layer and is used to downsample the size of convolutional feature map outputs. This is achieved via a sliding window kernel where we apply the pooling operation, most

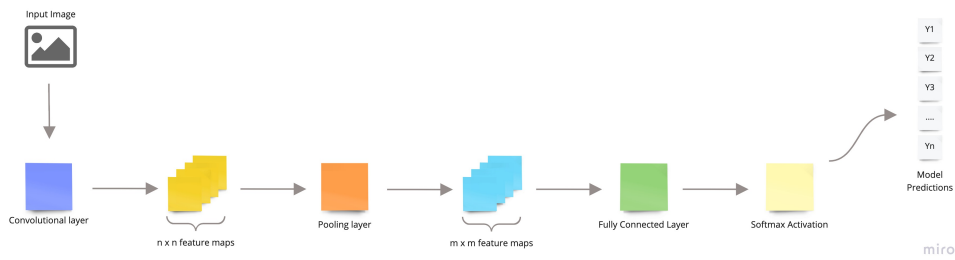


Figure 8: Vanilla Convolutional Neural Network (CNN) architecture with convolution, pooling and fully connected layers. Model predictions are presented after passing fully connected layer output through a softmax activation function.

commonly MEAN and MAX. This operation is performed to avoid overfitting the model over the training data.

- **Fully Connected Layer:** This layer acts as the classifier head of the network architecture and combines the feature maps from the convolutional and pooling layers. The 2-dimensional feature maps are converted to 1-dimensional vector and passed through a fully connected layer to get final output classes.

### 6.2.1 ResNet-18

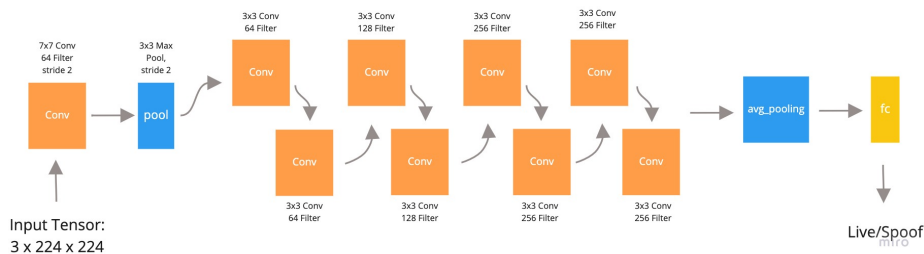


Figure 9: Resnet18 Architecture.

### 6.2.2 EfficientNet-B4

The Efficient-B4 architecture (Figure 8) consists of a two 2D-convolution (conv) layers with kernel sizes  $3 \times 3$  and  $1 \times 1$ , respectively, and stride  $2 \times 2$  and  $1 \times 1$ , respectively. Inputs are zero-padded before passing through the convolution layers. Both convolution layers are followed by a Batch Normalization (bn) layer.

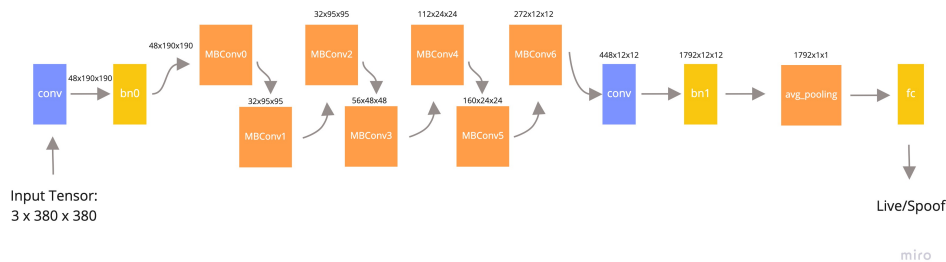


Figure 10: EffNet-B4 Architecture. conv: 2D-Convolution Layer, bno, bn1: Batch Normalization Layers, MBConv: Mobile Inverted Residual Bottleneck Block, avg pooling: Adaptive 2D Average Pooling Layer, and fc: Fully Connected Layer.

The first Batch Normalization (bno) layer is followed by 7 Mobile Inverted Residual Bottleneck (MBConv) blocks. The MBConv blocks are dependent on the width and depth coefficients of the B4 variant. The second Batch Normalization (bn1) layer is followed by a 2D Adaptive Average Pooling (avg pooling) layer and a Fully Connected (fc) layer to get the final classification outputs. We define the input dimensions as: batch size  $\times$  channels  $\times$  height  $\times$  width.

### 6.2.3 Normalization in Deep Neural Networks

When training deep neural networks, it is a common practice to compute a gradient descent step over mini-batches of training data. Neural networks trained over mini-batches result in faster convergence of network weights when compared to neural networks trained over the entire training data per gradient descent step. But this approach also introduces a phenomenon known as internal covariate shift. Internal covariate shift slows down network training because of the varying data distribution of mini-batches. This increase in training time is due to smaller learning rates which are required to reach convergence. The issue has since been tackled by an optimisation trick proposed by Ioffe and Szegedy (2015) called Batch Normalization. Batch normalization reduces the internal covariate shift of a deep neural network by normalising the input tensors before passing it through a layer in the network architecture. This drastically reduces training time through faster convergence speed. Since the first CNN architecture to include batch normalization, GoogLeNet (Szegedy et al., 2015), it is now an integral component of modern CNN architectures for a myriad of computer vision applications. Huang et al. (2019) propose a new strategy for normalization called Iterative Normalization which achieves better optimisation and performance over batch normalization by whitening the feature space of an input tensor via Newton’s iterations. Whitening is a linear transformation operation which normalises and decorrelates the vector space of an input data. We discuss this work because it is a crucial component of our research

in interpretable lip-sync deepfake detection.

#### 6.2.4 Model Interpretability

The purpose of explainable AI differs between different stakeholder groups. For a developer stakeholder group, consisting of data scientists and AI researchers, model performance metrics are sufficient to understand and, to a certain extent, interpret a deep neural network's behaviour. These metrics consist of, but not limited to, model accuracy, type I and II errors. The same is not true for other stakeholder groups such as regulatory bodies, executive members and end users (Barredo Arrieta et al., 2020 [15]). In the domain of deepfake detection, we can consider two main stakeholder groups - end users and regulatory bodies. The end user group mainly consist of humans who use a deepfake detection tool on videos they provide. On the other hand, the regulatory bodies group consist of humans who are in-charge of setting regulations and policies regarding the use of AI. We also include the judiciary bodies in this group as they are responsible to enforce legal actions against any misuse of the deepfake technology. For both groups, it is important that the deepfake detection tool is capable of convincing a non-expert human about its predictions and provide sufficient supplementary information to justify its decision. Within explainable AI, model interpretability is an area of research which aims to make neural network models interpretable to all stakeholder groups. Doshi-Velez and Kim [18] provide a detailed overview of the importance of model interpretability, which ranges from better understanding of a deep neural network's behaviour to the ethics of and fairness in its end-user applications. Gilpin et al. [16] and Mittelstadt et al. [17] distinguish model interpretability into two categories: inside explanations which describe the internal model behaviour, and outside explanations which provide explanations to justify a neural network model's decisions. Gilpin et al. (2018) provide an overview of available model interpretability methods within the aforementioned model interpretability categories:

- *Outside explanations:* with linear proxy models and decision trees, summarising model decisions, and salience mapping input data to final prediction output.
- *Inside explanations:* with network representations via layer and neuron attribution; interpretable neural network architectures, consisting of attention models or disentangled representations, which can generate explanations without needing an external model interpretability method.

The developer stakeholder group also uses internal explanations to debug their model training experiments (Adebayo et al. [19]). In the context of model training, debugging is the process of identifying components in a training experiment which negatively impacts model performance. To this extent, our research aimed to explore influence functions (Koh and Liang, 2017) for improving our baseline models. But, the work by

Basu et al. [20] demonstrate that using influence functions to debug deep neural network models is not effective as they are unstable and unreliable with deeper network architectures. Due to this, our research explores other outside explanation methods to debug and improve the detector model performance. Zeiler and Fergus (2014) [23] propose the Occlusion algorithm, which is a perturbation-based input attribution method to visualise region-wise importance of an input image contributing towards the model prediction output.

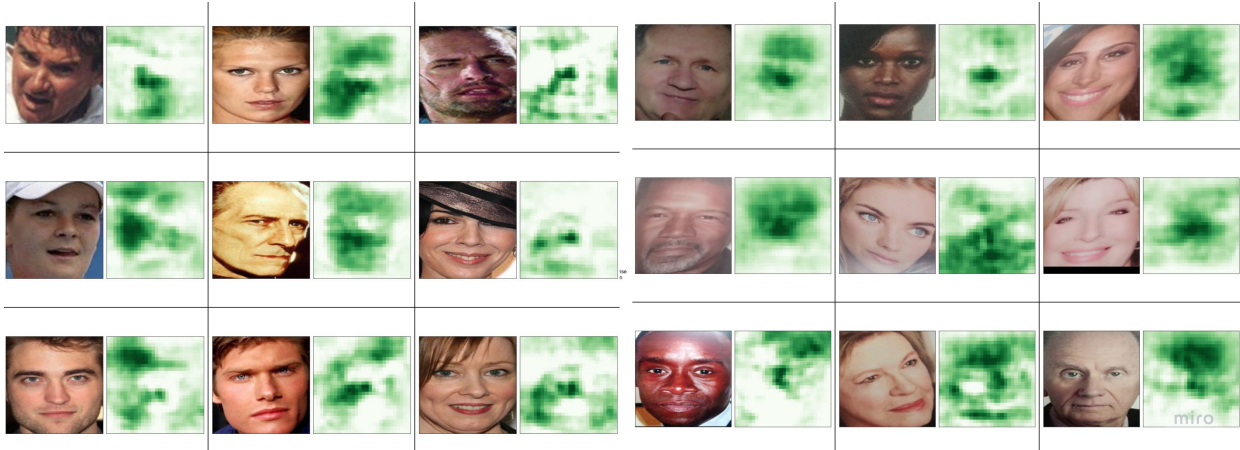


Figure 11: Attribution example for live class (first 3 columns) and A4 spoof (last 3 columns).

### 6.3 Domain Generalization and Adaptation

Deep learning is essential an optimization process which goal is to find a mapping function that matches features  $x$  to labels  $y$ . We can formulate the optimization problem as: given the training set with  $m, x, y$  we write:

$$set_{train} = \{(x_i^{train}, y_i^{train})\}_{i=1}^m \rightarrow Q_{x, y} \quad (4)$$

where  $m$  is the number of samples of the training set. Features and labels coming from a certain distribution denoted by  $Q_{x, y}$ .  $Q_{x, y}$  is a join distribution between features and labels. For the actual training process, we define a loss function and minimize the average loss error in the training set.

$$min_{\theta} \left( \frac{1}{m} \sum_{i=1}^m L(f_{\theta}(x_i^{train}), y_i^{train}) \right) \Rightarrow \theta^* \quad (5)$$

where  $\theta^*$  are the parameters of the optimum function  $f_{\theta^*}$  and  $L$  is the loss function. Similarly, we define the test set as:

$$set_{test} = \{(x_i^{test}, y_i^{test})\}_{i=1}^n \rightarrow Q_{x,y} \quad (6)$$

where  $n$  is the number of test examples. We evaluate over the test set and compute the test error as:

$$error_{test} = \frac{1}{n} \sum_{i=1}^n L(f_{\theta^*}(x_i^{test}), y_i^{test}) \quad (7)$$

Finally, the whole goal of the optimization process is to find a mapping function that minimizes the error on test set. It has been shown that even for over-parameterized models, the gap between train and test errors can be small given a sufficient number of samples.

The above optimization process makes a key assumption. It defines that both train and test sets distributions are the same, as it can be seen from equations (4) and (6). This assumption fails short in almost any real life application as frequently real world examples are different. In the general case, the training distribution is usually referred as *source domain* and denoted with  $Q_{x,y}$  while the test distribution as *target domain* and denoted with  $P_{x,y}$

$$Q_{x,y} \neq P_{x,y} \quad (8)$$

For the liveness detection problem, it is very common the target domain to differ significantly, for example different illumination conditions, wide variety of samples acquisition devices(camera sensors) or large variety of different materials a face might be displayed on (eg. monitors, phones, papers). All these variables constitute the gap between domains. Domain generalization and adaptation is still an open problem. In our work, we focus on ideas of multi-task learning, deep metrics losses and few-shot-knn techniques to assess the best strategy to train a robust model.

## 6.4 Liveness Detection as binary classification problem

Deep learning is a subset of Machine Learning (ML). ML is a computer program and is said to learn from experience  $E$  with respect to some tasks  $T$  with performance evaluation metric  $P$ , if its performance measure can improve with  $E$  on  $T$ . Binary classification task is the task of classifying samples of a certain distribution into two classes. In liveness detection the goal is to assign live or spoof label for each sample in training set. In binary classification tasks the objective is to minimize, usually a single loss function (eg. Binary Cross Entropy BCE or Cross Entropy CE) and map input features  $X$  into two labels  $Y$ . CelebASpoof dataset contains images of 11 classes, where class 0 is the genuine/live label and labels 1-10 correspond to 10 different spoof types. Naturally, liveness detection systems mainly concern the binary distinction of labels, as identifying



accurately the type of potential attack plays second role to accurately predict between live and spoof class. In this experiment we treat the 10 separate spoof classes as one, denoted by label 1. Such transformation of the dataset yields to heavily imbalanced distribution between the two classes. In such cases the model will learn to predict spoof class as during back propagation step the dominant class in mini-batch will heavily dictate the direction of gradients. In the next section we will address class imbalance.

## 6.5 Liveness Detection as multi-task problem

Multi-task Learning is an approach to introduce inductive bias contained in parallel learned tasks' training signals. Inductive bias transfer, helps improving the generalization capabilities of the main learned task as what is learned for each task can be used to help other tasks. It acts as regularization technique as it learns multiple tasks simultaneously by exploiting both *task-generic* and *task-specific* information. Multi-task networks typically consist of multiple fully connected heads, each of which has its own loss function [9]. They share model's backbone weights but, in essence, they compete for networks' limited resources. Although auxiliary tasks share weights they can drastically help learning the original task of the model and improve upon the main binary task. Multi-task learning have been successfully used in many applications, from speech recognition, computer vision to natural language processing. The total loss function comprised from multiple individual losses from each independent task/head.

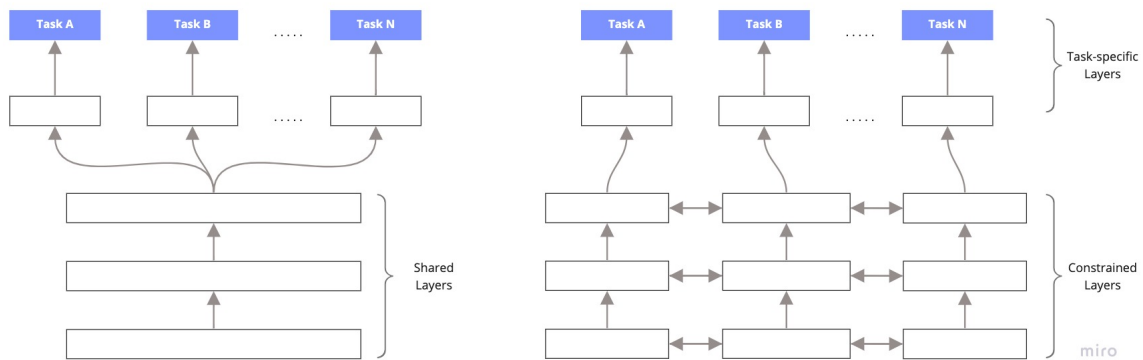


Figure 12: **Hard** (left) and **Soft** (right) parameter sharing for multi-task learning in deep neural networks.

Typically, *multi-task learning* is split into two categories. Hard and Soft parameter sharing.

**Hard sharing**, all hidden layers are shared from all tasks/heads, a practice that helps to avoid *overfitting*. Intuitively, the more tasks the model tries to learn the more it has

to find a representation capable to capture information for all of them.

**Soft sharing**, each auxiliary task operates on its own model and parameters. Then, the distance of parameters among each tasks are regularized using  $L_2$  of *trace* norm.

### 6.5.1 Magnitude of Gradients

As mentioned above, in MTL the goal is to find the set of parameters  $\theta$  that optimizes each task’s objective. Usually, hard-parameter sharing is applied where the same embeddings learn generic representations. Obviously, in MTL when tasks are completely unrelated to each other, the optimization process would not converge into something useful. In liveness detection problems, the entire set of classes(fine-grained classes) (eg. 3D mask, replay-attacks, printed photo, etc). transformed into a coarse classes map that describes the same problem from higher abstraction. For example, a multi-class problem with a live class and 3 spoof type classes is converted into a binary classification task and used as additional head in neural network architecture. Thus, the total loss function is the summation of each individual loss.

$$L_{total} = \sum_{i=1}^T w_i l_i \quad (9)$$

where  $i$  iterates though task-heads,  $l_i$  is the computed loss of task  $i$  and  $w_i$  its assigned contribution/weight to total loss signal.  $w$  is a hyper-parameter and typically is hard to tune. However, the objective landscape in respect to parameters  $\theta$  for each task could differ drastically. A certain configuration set for  $\theta$ , that optimizes one task might result in higher loss in other task. This results to *conflicted gradients* as shown in figure [13]. In such scenarios, practitioners assign weights ( $\alpha$ ) to each loss signal to control or prioritize certain task. The final direction of gradient  $d$  is given by:

$$d = \alpha_1 * g_1 + \alpha_2 * g_2 \quad (10)$$

There are few approaches to tackle the problem of conflicted gradients such Multi-Gradient Descent Algorithm or Conflict-Averse Gradient Descent but in this work we will investigate orthogonal loss from Guiqing He et al. [13].

### 6.5.2 Direction of Gradients

In an attempt to improve the classification performance of the multi-task network, we will use orthogonality loss. If two feature vectors are completely independent, they carry the most information and can be used as feature selectors. Such vectors are said to be *orthogonal* to each other. Guiqing He et al. [13] proposed a novel orthogonal loss function that could be applied to multi-task networks.

$$L(x) = \frac{1}{N} \min_{f_1, f_2, \dots, f_k} \sum Tr[f_s(x) f_g^T(x)] \quad (11)$$

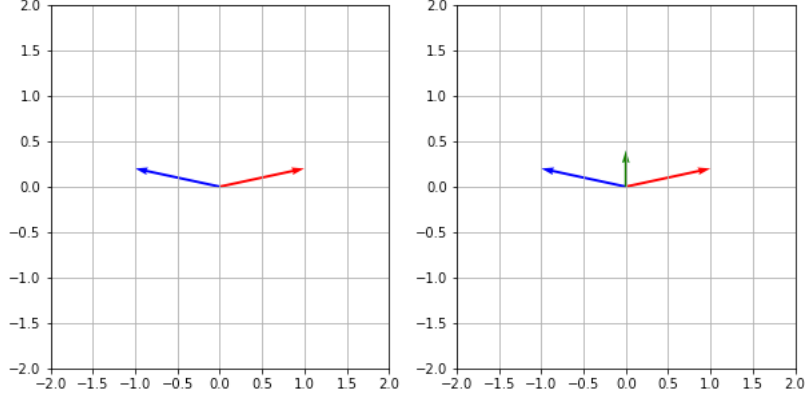


Figure 13: Example of **conflicted** gradients.

where  $f_g(x)$  represents the coarse classifier embeddings (features) of N images while  $f_s(x)$  represents the fine classifier embeddings for N images. The term  $f_s(x)f_g^T(x)$  represents the sum of dots products of fine and coarse classifier features with batch size N. The mathematical operator  $Tr$  represents the *trace* of a given squared matrix and it is defined as the summation of the elements in the main diagonal.  $\alpha$  is a hyper parameter and its magnitude controls the influence of orthogonality loss into the entire network parameters during backpropagation.

## 6.6 Deep metric learning networks for Liveness Detection

### 6.6.1 Center loss

A deep neural network performs a classification task by learning to discriminate the feature space and creating decision boundaries. State of the art deep neural network models achieve this objective via softmax activation function, which acts as a supervision signal. Wen et al. (2016) [11] proposed Center Loss to enhance a model's ability to accurately discriminate the latent feature space for face recognition task. Center loss tries to minimize intra-class variations while keeping feature embeddings of different classes as separable as possible. It acts as an additional supervision signal which discriminates the model's latent feature space by assigning it class centers,  $c_{y_i}$ . Formally, Center Loss can be defined as:

$$L_c = \frac{1}{2} \sum_{i=1}^m \|x_i - c_{y_i}\|_2^2 \quad (12)$$

where  $x_i$  denotes input vector with  $y_i$  as the target class in training dataset.

The formulation describes the intra-class variations of deep features of the entire training dataset.  $c_{y_i}$  denotes the  $y_i$ th class center of deep features. As authors described

in [11] such formulation is inefficient as it requires the entire training set in order to update the centers.

If we only use the softmax loss as supervision signal, the resulting deeply learned features would contain large intra-class variations. On the other hand, if we only supervise CNNs by the center loss, the deeply learned features and centers will degrade to zeros (At this point, the center loss is very small). Simply using either of them could not achieve discriminative feature learning. So it is necessary to combine them to jointly supervise the CNNs, as confirmed by our experiments.

### 6.6.2 Triplet loss

A Triplet Network is closely related to Siamese Networks [14] but instead of pairs, is comprised of 3 similar feed-forward networks with shared backbones(parameters sharing). In computer vision a triplet network is fed with 3 images and returns  $L_2$  distances of embeddings representation of *positive* and *negative* images from the *anchor* sample. Formally is denoted as:

$$L(a, p, n) = \max(0, D(a, p) - D(a, n) + margin) \quad (13)$$

where  $D(x, y)$  is the distance between the learned features representation of  $x$  and  $y$ . Usually,  $L_2$  distance is used as distance metric but alternately cosine similarity can also be used. The objective of this function is to keep the distance between the anchor and positive smaller than the distance between the anchor and negative. Triplet loss function is suitable for problems with high number of classes (eg. Face recognition systems) but, generally speaking, it is hard to train and *sampling* techniques need to be applied. Elad Hoffer and Nir Ailon [12] details 3 inherently strategies.

- **Easy triplets:** Triplets with 0 loss,  $d(a, p) + margin < d(a, n)$ .
- **Hard triplets:** Negative images is closer to anchor than positive,  $d(a, n) < d(a, p)$ .
- **Semi-hard triplets:** triplets where the negative is not closer to the anchor than the positive, but which still have positive loss,  $d(a, p) < d(a, n) < d(a, p) + margin$ .

In our research, we implemented semi-hard mining and mining as pre-processing step. Before training, we prepare the data is such way that for each subject we construct triplets following:  $live_i, positive_i, negative_i$ .



Figure 14: Example of triplet. Anchor, Positive and Negative image.

## 7 Experiments

### 7.1 Methodology

As mentioned in previous chapters, our research explores learning-based methods to a) detect presentation attacks coming from the same distribution, b) evaluate different loss functions and assess the performance in multiple target domains (source dataset bias) and c) generate a new deepfake presentation attack type and detect using few-shots learning approach. Given the data-driven nature of such learning-based methods, we need a high-quality, high variation liveness detection dataset. At the time of our literature review of the current SOTA datasets, we identified that there was no open-sourced rich (in terms of variations) dataset (eg. in-the-wild), thus we used CelebASpoof[x] which is a large dataset that covers most of known presentation attacks. In addition, we collected and annotated two, relatively small, *Devs* datasets to further assess generalization to more realistic target domains. Lastly, we generate face-swapped counterparts from test-set of source domain dataset (CelebASpoof) using SimSwap model [38] with original weights.

#### 7.1.1 Data pre-processing

For video-datasets, we pre-process all the videos by extracting frames at equal intervals per video and saving metadata information such as frame rate, frame count and video resolution. The extracted video frames are compiled together to create an image dataset. We then perform face detection on the extracted frames and to dataset comprised by images using the MTCNN model [37] and save the detected face locations. The video metadata and face locations saved in individual JSON data formats. Figure [] shows cropped faces of source domain train split CelebASpoof dataset. Face detector, is sensitive to noise and tends to give a relative high number of false predictions. Images showing a single person, could have predicted with multiple bounding boxes thus it becomes difficult to ensure that only face-crops will be fed into the network for training. We assess the *Noise vs Rejection* trade-off by searching within a cut-off range and choose the lower bound (minimum crop size). In cut-off point, bounding box size expressed in pixels (width x height) which lower than threshold it gets rejected. The goal is to maximize the number of images with single number of predicted bounding box.

Another pre-preprocessing step is preparing crops to meet the input size of each architecture. Instead of resizing, we use tiling in order to avoid interpolation that affect artifacts created during image capturing. Such artifacts like, Moire Pattern in screen-replay attacks are important cues for better discrimination.

Finally, we create data selector files which split the image dataset into train, validation and test sets with a split ratio of 0.7 : 0.1 : 0.2. During training, on all images in mini-batches, we perform multiple *on-the-fly* data augmentations after resizing step



Figure 15: Example of false predicted bounding boxes.

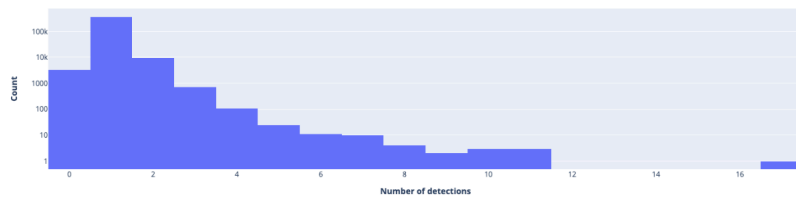


Figure 16: Histogram of bounding boxes @ cut-off point 2600 pixels

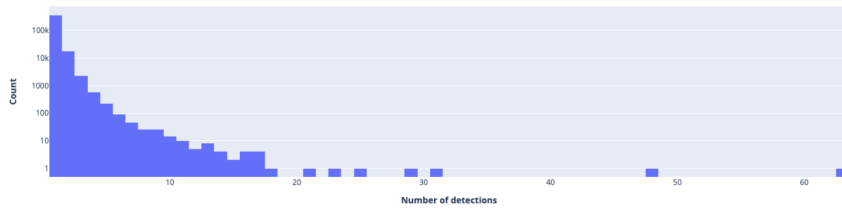


Figure 17: Histogram of bounding boxes predicted by MTCCN

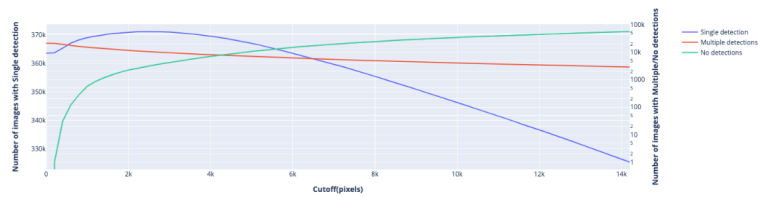


Figure 18: Cut-off exploration

to increase the robustness and performance of the trained detector model (refer to Appendix A.1 for more details).

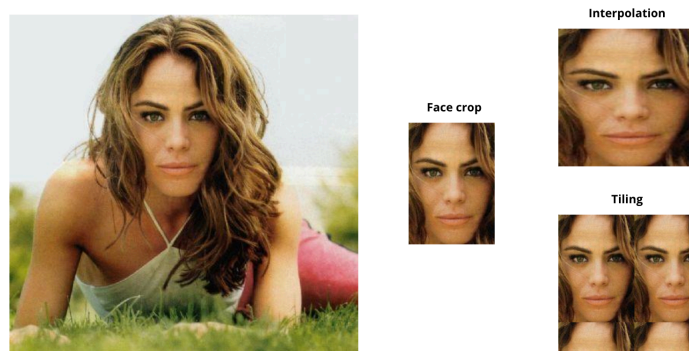


Figure 19: Example of crop tiling as opposed to resizing.



Figure 20: **Top:** Tight face crop, scale factor 1.0x. **Bottom:** scale factor 1.3x.

### 7.1.2 Illumination Annotation

Having established the importance of multi-task learning and the extra information smaller tasks could give to help the main binary task, we further update the model by including an additional fully connected head. The head named  $Illumination_{fc}$  has input size of 512 dimension and output of 5. It is worth mentioning, that for this training task on Resnet18 was considered.

For this purpose, we will annotate the entire train set considering **only** spoof type labels. The goal is to assign each image sample to one of the 5 illumination classes. The annotation process is the following:

- **Live** images(class=0) are annotated also with label 0 on  $Illumination_{fc}$ .
- **Spoof** images(classes=1 – 10) are annotated according to 4 bins.

To extract the illumination information from each image we convert RGB images into LAB color space. LAB was originally introduced by Richard Hunter. It is similar



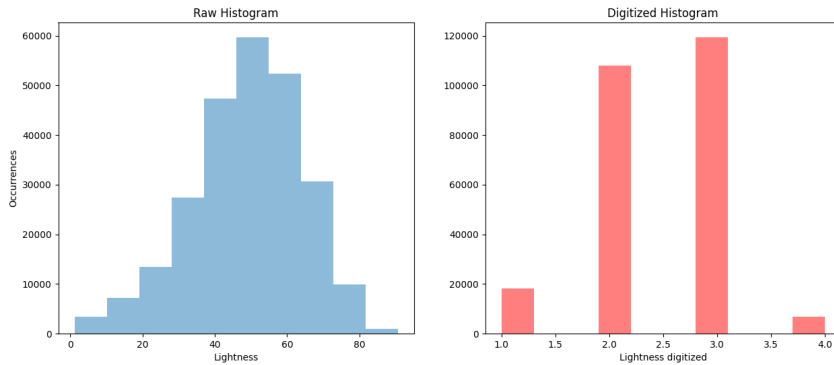


Figure 21: Raw histogram (left) and digitized with 4 bins (right) for spoof type class only. Sampled 5000 samples per type with total of 50000 images.

to geographic coordinates (longitude, latitude and altitude) and it is used to space uniformly perceived color differences. It comprises from three elements. A) **L** stands for *illumination*, B) **a** is about *red/green* values and C) **b** stands for *blue/yellow* values. For our purpose, we are interested only on *L* term. In order to extract this information, we perform face detection with MTCCN and convert them to gray-scale. We use PILLOW library and calculate the average illumination for each face. The resulted values range from 0-255. Figure 13 shows a histogram of illumination for 50000 training images. It is worth mentioning that the number of bins is essentially a *hyper-parameter* and empirically we found that 5 bins give the best results.

## 7.2 Model Training Experiments

We select Resnet18 and EffNet-B4 architectures to train baseline single task liveness detector models. We initialise all models with pre-trained weights on ImageNet dataset. The default setup of the model training experiments uses the AdamW optimizer with  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ , learning rate  $lr = 0.0001$  and batch size of 32. We add an early stopping condition with patience = 5 for all model training experiments. At the end of the training experiment, we select best model checkpoint based on the validation loss function output, computed at the end of each validation epoch. We use the best model checkpoints for evaluating model performance on all testsets (source and target domain).

To create the training and validation mini-batches, an input image is cropped using its face location coordinates. Then, it is resized to an image resolution of 224 x 224 for Resnet18 and 380 x 380 for EffNet-B4 architecture. As the input for all training experiments are cropped faces, all samples with no detected faces are filtered out. Finally, we get 533,899 image samples for training and 133,477 image samples for validation. After

resizing the training and validation mini-batches, they are normalised and converted to tensors. To handle class imbalance, we perform *online* class weights calculation and feed them into loss. Furthermore, we convert binary task ( $Binary_{fc}$ ) into a multi-task learning and train multiple models using the same backbones (Resnet18 and EffNet-B4). In more details, we first add extra fully connected head  $AttackType_{fc}$ , then a third referred as  $Illumination_{fc}$  in sequential manner. Also, we train with a) *center loss* on binary case, b) *orthogonal loss* on two heads case and c) *triplet loss* on headless models. A summary of all experiments is shown in table 3.

<b>Model</b>	<b>Task</b>	<b>Heads</b>	<b>Loss</b>
Resnet18 Binary Task	binary	$Binary_{fc}$	1x cross-entropy
Resnet18 MTL Spoof	multi-task	$Binary_{fc}$ $AttackType_{fc}$	2x cross-entropy
Resnet18 MTL	multi-task	$Binary_{fc}$ $AttackType_{fc}$ $Illumination_{fc}$	3x cross-entropy
Resnet18 Center Loss	binary	$Binary_{fc}$	cross-entropy + center loss
Resnet18 Orthogonal Loss	multi-task	$Binary_{fc}$ $AttackType_{fc}$	cross-entropy + orthogonal loss
Resnet18 Triplet Loss	embeddings	headless	triplet-loss
EffNet-B4 Binary Task	binary	$Binary_{fc}$	1x cross-entropy
EffNet-B4 MTL Spoof	multi-task	$Binary_{fc}$ $AttackType_{fc}$	2x cross-entropy

Table 3: Overview of training experiments.

### 7.3 Hardware Specifications

We use a local GPU workstation machine with an 8-core Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz microprocessor chip, 32 GB RAM and TITAN Xp graphics card with 12GB VRAM. The machine runs on Ubuntu 18.04.5 LTS operating system, CUDA version 10.0 and NVIDIA driver version 450.119.03. For all the training experiments, we create a virtual python == 3.7.4 environment with pytorch == 1.8.0.

### 7.4 Domain Adaptation and Generalization Results

In this section we will train various model configurations and loss functions (Table 3) in order to assess single-dataset generalization capabilities. As mentioned above, CelebASpoof is used for training and it comes from a source distribution denoted as  $Q(x, y)$ . All various target domains contain presentation attacks which were already seen during training but coming from different distributions. We define 4 evaluation metrics which will be used extensively throughout experiments.

- **False Positive Rate:** is the probability a given image to incorrectly classified into spoof/presentation attack class
- **Area Under the Curve:** is the entire two dimension area under ROC curve. AUC aggregates the performance across all possible classification thresholds.
- **Equal Error Rate:** EER is given by the point in classification thresholds space where both False Negative Rate(FNR) and False Positive Rate(FPR) are equal and minimized. It is computed from Detection Error Tradeoff (DET) curve.
- **Half Total Error Rate:** HTER is the average error rate at a given classification threshold.  $HTER = \frac{FNR+FPR}{2}$

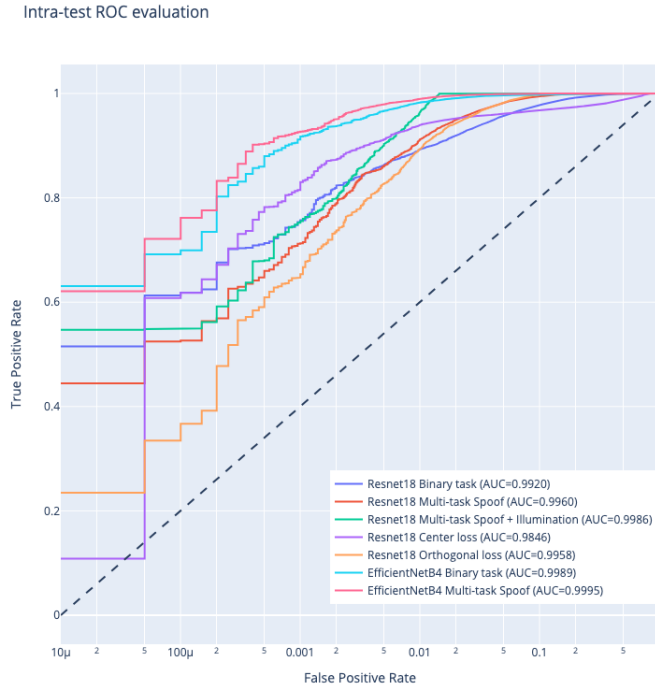


Figure 22: Receiver Operating Characteristic curves for all models in **intra-test** setting.

We prefix with **intra-test** all metrics regarding evaluations of test-set of source domain and **cross-test** for evaluations of target domains. Figure 14 and table 4 presents results of all training experiments of source domain. We see that ResNet18 with Center loss and Orthogonal loss experiments are very close to binary only model in terms of

<b>Model</b>	<b>EER</b>	<b>AUC</b>
Resnet18 Binary Task	0.046	0.992
Resnet18 MTL Spoof	0.032	0.996
Resnet18 MTL	<b>0.013</b>	<b>0.9986</b>
Resnet18 Center Loss	0.041	0.9846
Resnet18 Orthogonal Loss	0.033	0.9958
EffNet-B4 Binary Task	0.014	0.9989
EffNet-B4 MTL Spoof	<b>0.01</b>	<b>0.9995</b>

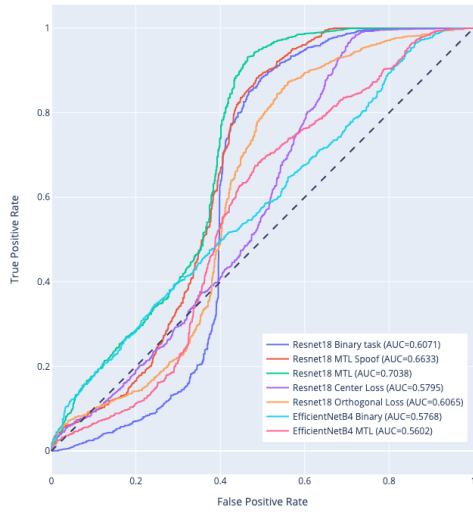
Table 4: Source domain test-set evaluation metrics, **intra-test**

EER and AUC though, Resnet18 Multi-Task with 3 heads (binary, spoof type and illumination) achieves the best EER (0.013) and AUC (0.9986). In addition, the configuration of ResNet18 with two heads outperforms the classic binary classification task. Similarly, when we compare intra-test performance of EfficientNetB4 for binary and multi-task cases, we notice an improved performance in later configuration of the experiments. For train and validation sets please refer to appendix A.2 and Figure 18. As we have seen in chapter 6.5.1 and equation (9) when the objective function comprises from multiple loss functions (multi-task learning) then, inevitable we introduce another hyper-parameter  $w_i$  where  $i$  denotes  $i$ -th task (head) and it is the strength of loss signal of to overall gradients. Different weights will result to different total loss. In our experiments we found that for the Resnet18 and 3 heads configuration (Resnet18 MTL), the best set of loss weights was 1.0 for binary head, 0.5 spoof type head and 0.01 for illumination head. However for Resnet18 2 heads (binary + spoof type) the best set was 0.5 respectively. For the orthogonal loss training we used balanced weights for each loss signal. This is up for future exploration. Table 5 shows detailed results of target domains.

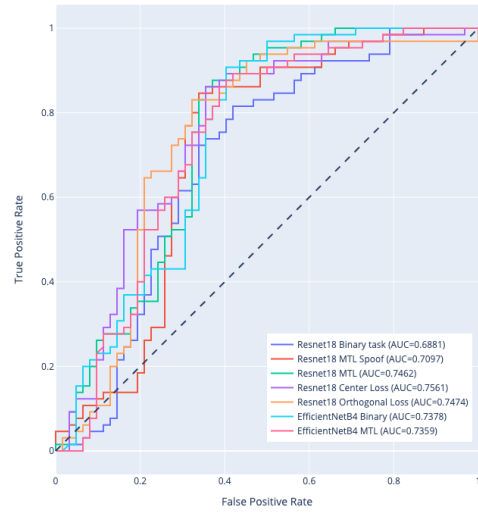
#### 7.4.1 Dataset soup results

A natural question which arises, since all datasets are available during training. What is the performance of bundling/merging and training all datasets? As expected, Resnet18 MTL gives good results with combined EER of 10%. For more information please refer to appendixes Figures 22,23,24,24.

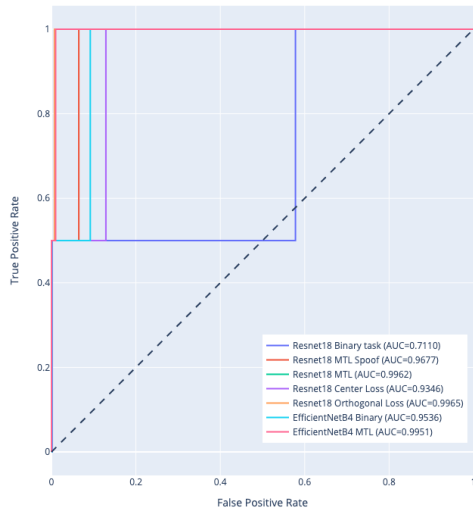
ROC: Dev Dataset



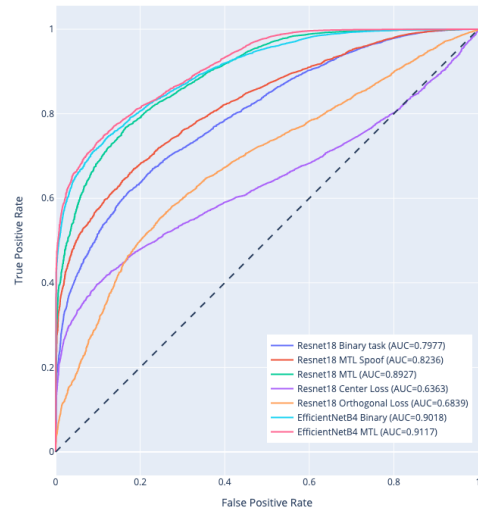
ROC: Dev Dataset HQ



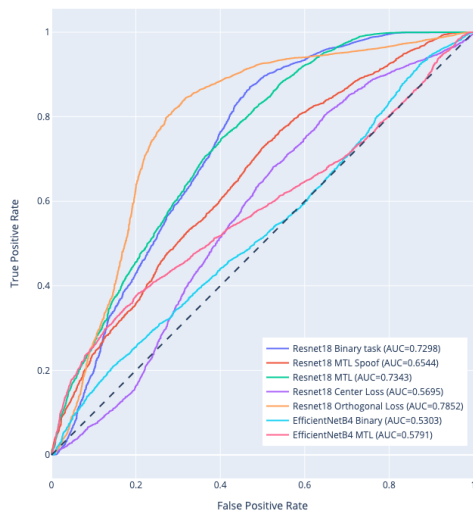
ROC: Eyeblink8



ROC: Replay Attack



ROC: Replay Mobile



ROC: SIW

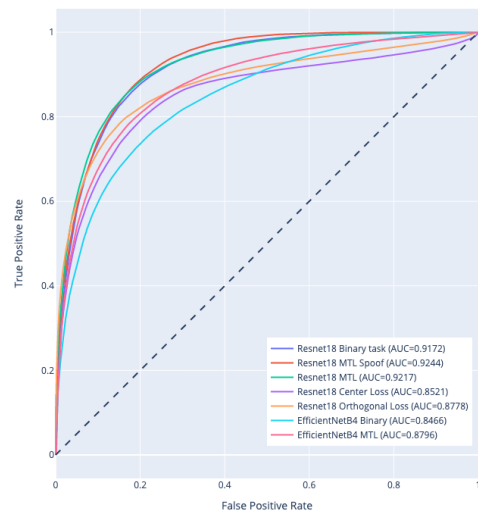


Figure 23: Receiver Operating Characteristic curves for all models in **cross-test** setting.

## 7.5 Few-shots Learning for DeepFakes

So far we have seen the impact of different training configurations by introducing multiple heads into binary task and assess different loss functions. We address the problem of liveness detection by considering classic spoof types (eg. printed photo, video replays) but with the evolvement of deep neural networks new more sophisticated spoof attempts arise. An attacker can use synthetic face generation or face-swap to fool a face recognition system. To this end, we created a synthetic dataset by performing face-swap using state of the model SimSwap [38]. SimSwap can realize arbitrary face swapping on both images and videos with just a single trained model. We used the test split of CelebASpoof dataset and generated 10K face-swapped images based on same gender. We excluded all images not labelled as adults and used 5K for training and 5K for testing. Is it worth mentioning that we didn't use SimSwap directly from the original repository, but instead, we developed a Deepfake Offensive Toolkit (DOT) which uses SimSwap as backend. More information of DOT can be found here [39].

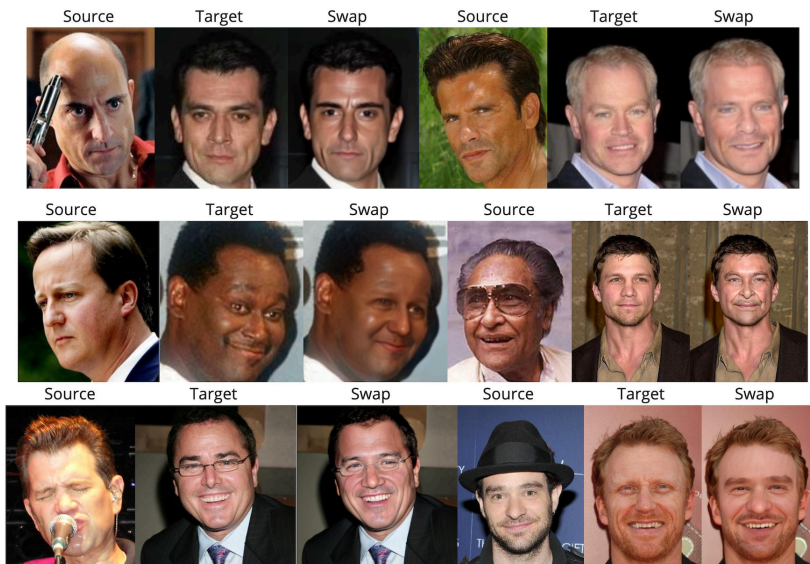


Figure 24: Face-swap examples using SimSwap model.

We used all trained models presented in section 7.4, performed forward pass and stored the embeddings of the last layer before classification.

The extracted features are fed into kNN classifier for training. We performed grid search to find the best number of neighbors. In order to evaluate the ability of the embeddings of each model to adapt to unseen classes, we adapted few-shots learning techniques(FSL). FSL is needed due to rareness of samples which in our context due to new spoof type attacks. It aims to reduce data gathering process and associated computational in case of re-training costs. We formulated our experiments is such a way that for

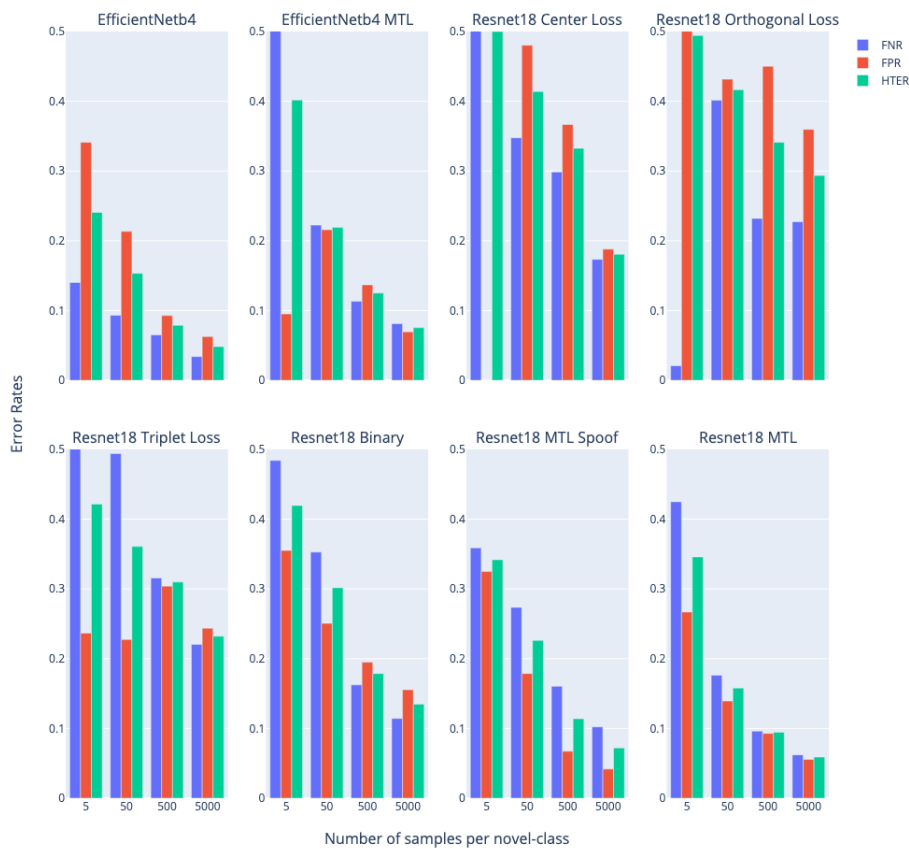


Figure 25: Error rates for different k-shots.

each model we train 4 different kNN classifiers based on number of the newly available spoof type/class samples. We start with 5 samples (k-shots) and increase by factor of 10 (5,50,500,5000). We evaluate the different experiments with 3 metrics, FNR, FPR and HTER. Figure 16. shows that all model-embeddings + kNN, can significantly improve error rates as the number of samples of the novel class increases. Though, some models such Resnet18 Orthogonal Loss seems to have learnt more discriminative features of source domain and it fails to adapt as quickly. On the other hand, EfficientNetB4 MTL and especially Resnet18 MTL are able to reach approx. 10% error rate with relatively few shots (k=500). Tables 5 and 6 present errors rates for all experiments.

Model	Target Domain	AUC	EER
Resnet18 Binary Task	Devs-set-v1	0.6071	0.395
	Devs-set-v2	0.6881	0.354
	Eyeblink8	0.711	0.539
	Replay-Attack	0.7977	0.289
	Replay-Mobile	0.7298	0.341
	SiW	0.9172	0.162
Resnet18 MTL Spoof	Devs-set-v1	0.663	0.387
	Devs-set-v2	0.709	0.33
	Eyeblink8	0.9677	0.032
	Replay-Attack	0.8236	0.266
	Replay-Mobile	0.6544	0.399
	SiW	0.9244	0.158
Resnet18 MTL	Devs-set-v1	0.73	<b>0.385</b>
	Devs-set-v2	0.7462	0.284
	Eyeblink8	0.9962	0.004
	Replay-Attack	0.8927	<b>0.24</b>
	Replay-Mobile	0.7343	0.337
	SiW	0.9217	<b>0.157</b>
Resnet18 Center Loss	Devs-set-v1	0.5795	0.483
	Devs-set-v2	0.7561	0.292
	Eyeblink8	0.9346	0.064
	Replay-Attack	0.6363	0.4
	Replay-Mobile	0.7852	0.438
	Replay-SiW	0.8521	0.24
Resnet18 Orthogonal Loss	Devs-set-v1	0.606	0.413
	Devs-set-v2	0.7474	<b>0.276</b>
	Eyeblink8	0.9965	<b>0.004</b>
	Replay-Attack	0.6839	0.354
	Replay-Mobile	0.7852	<b>0.246</b>
	SiW	0.8778	0.187
EffNet-B4 Binary Task	Devs-set-v1	0.5768	0.461
	Devs-set-v2	0.7378	0.354
	Eyeblink8	0.9536	0.046
	Replay-Attack	0.9018	0.197
	Replay-Mobile	0.5303	0.492
	SiW	0.8466	0.233
EffNet-B4 MTL Spoof	Devs-set-v1	0.5602	<b>0.422</b>
	Devs-set-v2	0.7359	<b>0.323</b>
	Eyeblink8	0.9951	<b>0.005</b>
	Replay-Attack	0.9117	<b>0.19</b>
	Replay-Mobile	0.5791	<b>0.448</b>
	SiW	0.8796	<b>0.196</b>

Table 5: Target domain test-set evaluation metrics, **cross-test**. Best scores in bold grouped by model architecture.



<b>Models</b>	<b>k-Shots</b>	<b>FNR</b>	<b>FPR</b>	<b>HTER</b>
EfficientNetb4 Binary	5	0.140	0.341	0.24
	50	0.09	0.213	0.153
	500	0.06	0.092	0.078
	5000	0.03	0.062	0.048
Resnet18 Center Loss	5	1.0	0.0	0.5
	50	0.34	0.48	0.41
	500	0.29	0.36	0.33
	5000	0.17	0.18	0.18
Resnet18 Orthogonal Loss	5	0.02	0.96	0.49
	50	0.4	0.43	0.41
	500	0.23	0.45	0.34
	5000	0.22	0.35	0.29
Resnet18 Triplet Loss	5	0.02	0.96	0.49
	50	0.4	0.43	0.41
	500	0.23	0.45	0.34
	5000	0.22	0.35	0.29

Table 6: Error rates for binary and metric losses k-shot CNN+KNN.

<b>Models</b>	<b>k-Shots</b>	<b>FNR</b>	<b>FPR</b>	<b>HTER</b>
Resnet18 Binary	5	0.48	0.35	0.41
	50	0.35	0.24	0.30
	500	0.16	0.19	0.17
	5000	0.11	0.15	0.13
Resnet18 MTL Spooof	5	0.35	0.32	0.34
	50	0.27	0.17	0.22
	500	0.16	0.06	0.11
	5000	0.1	0.04	0.07
Resnet18 MTL	5	0.42	0.26	0.34
	50	0.17	0.13	0.15
	500	0.09	0.09	0.09
	5000	0.06	0.05	0.05
EfficientNetb4 MTL	5	0.7	0.095	0.4
	50	0.22	0.0215	0.21
	500	0.113	0.136	0.12
	5000	0.081	0.069	0.075

Table 7: Error rates for multi-task learning k-shot CNN+KNN.

## 8 Conclusion

The primary goal of our research is to provide groundwork in domain generalization and adaptation in Liveness Detection problem. We introduced a third classification head for custom annotated image illumination labels and trained Resnet18 in a multi-task configuration. We established a baseline (binary task) performance with  $EEER = 4.6\%$  and improved to  $EEER = 1.3\%$  with multiple fully connected heads. We have shown the importance of multi-task learning and to how exploit source domain dataset knowledge map to extract better performance in both intra and cross evaluation results. We have explored multiple loss functions (eg. triplet loss, center loss, orthogonality loss) in an attempt to improve up the binary classification task. Furthermore, we introduced a new synthetic face-swapped dataset with SimSwap [38] and applied few shots learning with kNN. We concluded in both scenarios (generalization and adaptation) multi-task learning is superior to other approaches.

### 8.1 Future Work

The gained knowledge from our research into liveness detection has been very illuminating, especially in the area of domain generalization and adaptation. We've seen the importance of multi-task learning with average gradients as well as orthogonal gradients. For the former, would be interested to test other backbones like Xception, VGG and vision transformers(ViT). In our work we've restricted orthogonality only to 2 heads and mixing illumination head would be an interesting experiment too. In addition, to further assess generalization capabilities, as a future work Generative models attract some attention.

## References

- [1] Tao Wang, Jianwei Yang, Zhen Lei, Shengcai Liao, and Stan Z Li. *Face liveness detection using 3d structure recovered from a single camera*. In 2013 international conference on biometrics (ICB), pages 1–6. IEEE, 2013.
- [2] Yousef Atoum, Yaojie Liu, Amin Jourabloo, and Xiaoming Liu. *Face anti-spoofing using patch and depth-based cnns*. In 2017 IEEE International Joint Conference on Biometrics (IJCB), pages 319–328, 2017.
- [3] Jianwei Yang, Zhen Lei, and Stan Z Li. *Learn convolutional neural network for face anti-spoofing*. arXiv preprint arXiv:1408.5601, 2014.
- [4] Anjith George and Sébastien Marcel. *Deep pixel-wise binary supervision for face presentation attack detection*. In 2019 International Conference on Biometrics (ICB), pages 1–8. IEEE, 2019.
- [5] Amin Jourabloo and Xiaoming Liu (2017). *Pose-invariant face alignment via cnn-based dense 3d model fitting*. International Journal of Computer Vision, 124(2):187–203, 2017.
- [6] Volker Blanz and Thomas Vetter (2003). *Face recognition based on fitting a 3d morphable model*. *IEEE Transactions on pattern analysis and machine intelligence* 25(9):1063–1074, 2003.
- [7] Takashi Matsumoto (1991). *Graphics system shadow generation using a depth buffer* 1991. US Patent 5,043,922.
- [8] Yaojie Liu, Amin Jourabloo, and X. Liu (2018). *Learning deep models for face anti-spoofing: Binary or auxiliary supervision*. IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 389–398, 2018.
- [9] Ruder, S. (2017) *An Overview of Multi-Task Learning in Deep Neural Networks*. arXiv: 1706.05098
- [10] Ojala, T., Pietikainen, M. and Maenpaa, T. (2002) *Multi Resolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24, 971-987. <http://dx.doi.org/10.1109/TPAMI.2002.1017623>
- [11] Wen, Y., Zhang, K., Li, Z., Qiao, Y. (2016). *A Discriminative Feature Learning Approach for Deep Face Recognition*. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9911. Springer, Cham.

- [12] Hoffer, E., Ailon, N. (2015). *Deep Metric Learning Using Triplet Network*. In: Feragen, A., Pelillo, M., Loog, M. (eds) *Similarity-Based Pattern Recognition. SIMBAD 2015*. Lecture Notes in Computer Science(), vol 9370. Springer, Cham.
- [13] G. He, Y. Huo, M. He, H. Zhang and J. Fan (2020) *A Novel Orthogonality Loss for Deep Hierarchical Multi-Task Learning*, in IEEE Access, vol. 8, pp. 67735-67744, 2020.
- [14] Koch, Gregory, Zemel, Richard and Salakhutdinov, Ruslan. (2015) *Siamese Neural Networks for One-shot Image Recognition*. ICML deep learning workshop.
- [15] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bernetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. *Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai*. *Information Fusion*, 58:82–115, 2020.
- [16] Leilani H. Gilpin, Cecilia Testart, Nathaniel Fruchter, and Julius Adebayo. *Explaining explanations to society*, 2019.
- [17] Brent Mittelstadt, Chris Russell, and Sandra Wachter. *Explaining explanations in ai*. In *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT\* '19*, page 279–288, New York, NY, USA, 2019.
- [18] Finale Doshi-Velez and Been Kim. *Towards a rigorous science of interpretable machine learning*, 2017.
- [19] Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. *Debugging tests for model explanations*. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 700–712. Curran Associates, Inc., 2020.
- [20] Samyadeep Basu, Phillip Pope, and Soheil Feizi. *Influence functions in deep learning are fragile*. *CoRR*, abs/2006.14651, 2020.
- [21] Jiamin Bai, Tian-Tsong Ng, Xinting Gao, and Yun-Qing Shi. *Is physics-based liveness detection truly possible with a single image?* In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 3425–3428. IEEE, 2010.
- [22] Jukka Määttä, Abdenour Hadid, and Matti Pietikäinen. *Face spoofing detection from single images using micro-texture analysis*. In *Biometrics (IJCB)*, 2011 international joint conference on, pages 1–7. IEEE, 2011.

- [23] Matthew D. Zeiler and Rob Fergus. *Visualizing and understanding convolutional net-works*. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, page 818-833
- [24] Yaojie Liu, Joel Stehouwer, Amin Jourabloo, and Xiaoming Liu. *Deep tree learning for zero-shot face anti-spoofing*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4680–4689, 2019.
- [25] Shervin Rahimzadeh Arashloo, Josef Kittler, and William Christmas. *An anomaly detection approach to face spoofing detection: A new formulation and evaluation protocol*. *IEEE Access*, 5:13868–13882, 2017.
- [26] Olegs Nikisins, Amir Mohammadi, André Anjos, and Sébastien Marcel. *On effectiveness of anomaly detection approaches against unseen presentation attacks in face anti-spoofing*. In 2018 International Conference on Biometrics (ICB), pages 75–81. IEEE, 2018.
- [27] Artur Costa-Pazo, Sushil Bhattacharjee, Esteban Vazquez-Fernandez, and Sébastien Marcel. *The replay-mobile face presentation-attack database*. In 2016 International Conference of the Biometrics Special Interest Group (BIOSIG), pages 1–7. IEEE, 2016.
- [28] <https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>
- [29] <https://www.idiap.ch/en/dataset/replayattack>
- [30] <http://cvlab.cse.msu.edu/siw-spoof-in-the-wild-database.html>
- [31] <https://www.blinkingmatters.com/research>
- [32] Di Wen, Hu Han, and Anil K Jain. *Face spoof detection with image distortion analysis*. *IEEE Transactions on Information Forensics and Security*, 10(4):746–761, 2015.
- [33] Tiago de Freitas Pereira, André Anjos, José Mario De Martino, and Sébastien Marcel. *Can face antispoofing countermeasures work in a real world scenario?* In 2013 international conference on biometrics (ICB), pages 1–8. IEEE, 2013.
- [34] R. Shao, X. Lan, J. Li, and P. Yuen. *Multi-adversarial discriminative deep domain generalization for face presentation attack detection*. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 10015–10023, 2019
- [35] Tiago de Freitas Pereira and Anjos et al. *Lbp-top based countermeasure against face spoofing attacks*. In ACCV, pages 121–132. Springer, 2012.

- [36] Jiangwei Li, Yunhong Wang, Tieniu Tan, and Anil K Jain. *Live face detection based on the analysis of fourier spectra*. In Biometric technology for human identification, volume 5404, pages 296–303. International Society for Optics and Photonics, 2004.
- [37] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. *Joint face detection and alignment using multitask cascaded convolutional networks*. IEEE Signal Processing Letters, 23(10):1499–1503, 10 2016. ISSN 1558-2361. doi: 10.1109/LSP.2016.2603342.
- [38] Renwang Chen, Xuanhong Chen, Bingbing Ni, Yanhao Ge. *SimSwap: An Efficient Framework For High Fidelity Face Swapping*. Proceedings of the 28th ACM International Conference on Multimedia. 2020
- [39] <https://github.com/sensity-ai/dot> *Deepfake Offensive Toolkit*

# Appendices

## Appendix A

### .1 Data Augmentation

For all model training experiments, we resize the training and validation samples based on the network architecture. Thus, we resize samples to  $224 \times 224$  for Resnet18 architecture and  $380 \times 380$  for EffNet-B4. We normalise the samples on all RGB colour channels with mean = [0.485,0.456,0.406] and standard deviation std = [0.229, 0.224, 0.225]. Table A.2 describes the multiple image augmentations with their respective selection probabilities p.

<b>Augmentation</b>	<b>p</b>
Horizontal Flip	0.5
Motion Blur	0.2
Gaussian Noise	0.2
Random Brightness Contrast (Brightness Limit = 0.4, Contrast Limit = 0.4)	0.2
Hue Saturation Value (Hue Shift Limit = 5, Saturation Shift Limit = 5)	0.1
Rotate	0.2
Label Smoothing	0.01

Table 8: *on-the-fly* image augmentations with selection probabilities p for model training experiments.

### .2 Training Process

All mentioned previously, all training experiments are performed on single dataset(CelebASpoof) which considered the source domain.

### .3 Face cropping

### .4 Dataset soup



Figure 26: **Augmentations** with the following order Original - HorizontalFlip - MotionBlur - GaussNoise - RandomBrightnessContrast - HueSaturationValue - Rotate



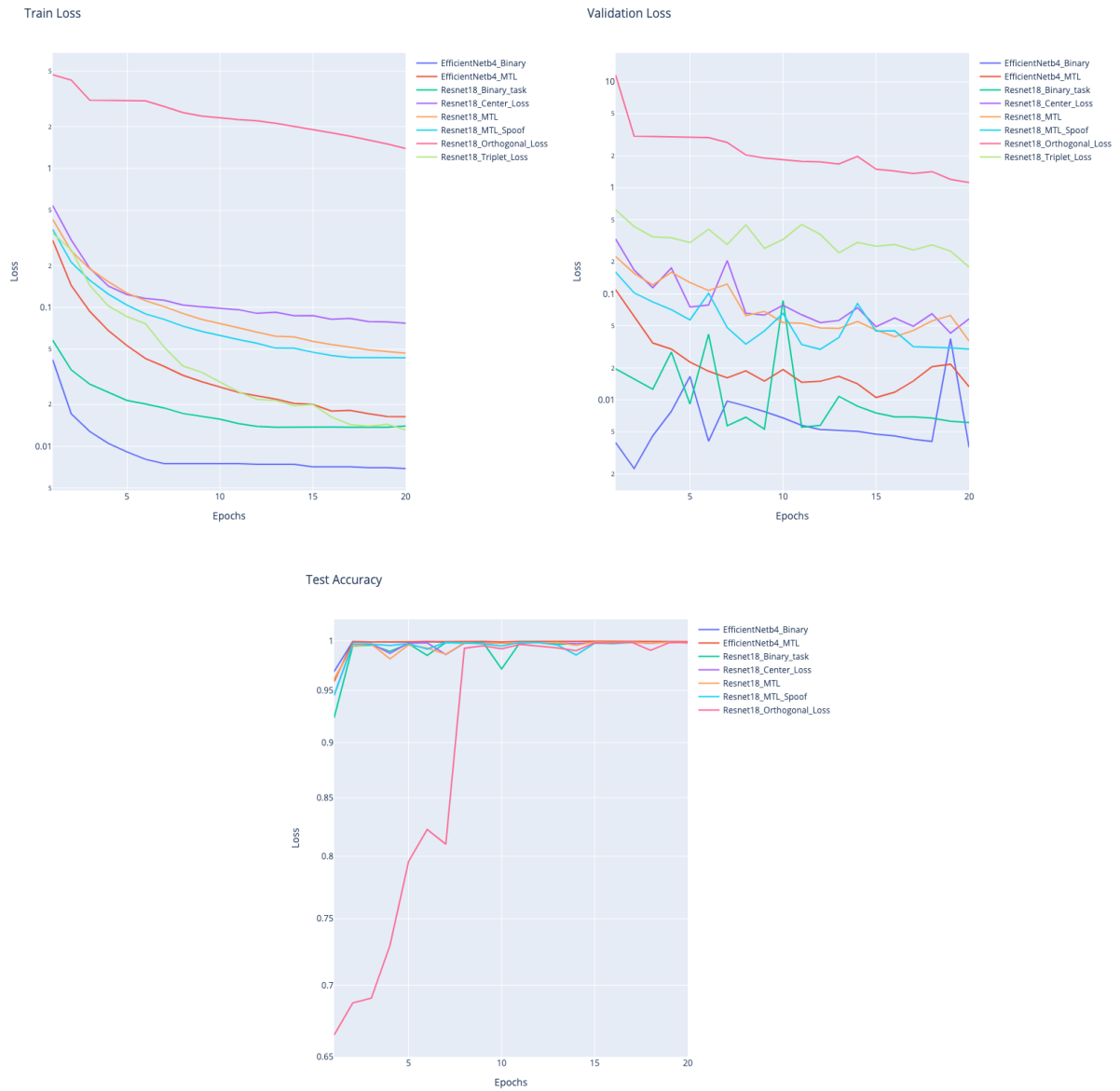


Figure 27: Losses and accuracy on train and validation sets.

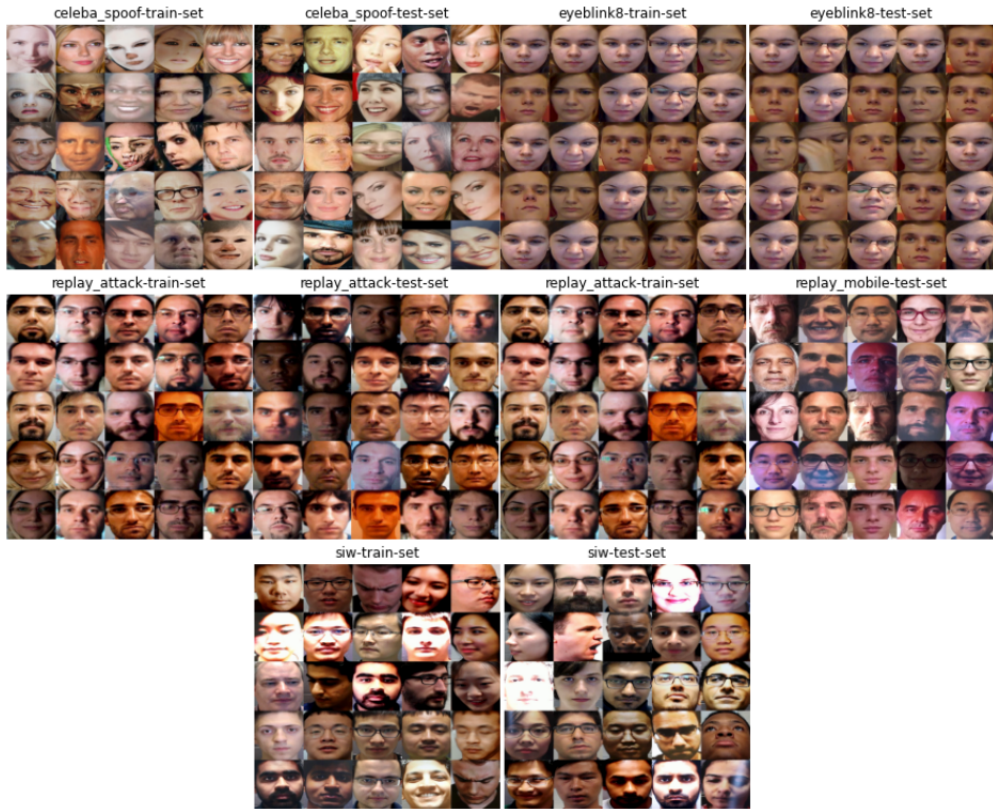


Figure 28: Datasets crop examples

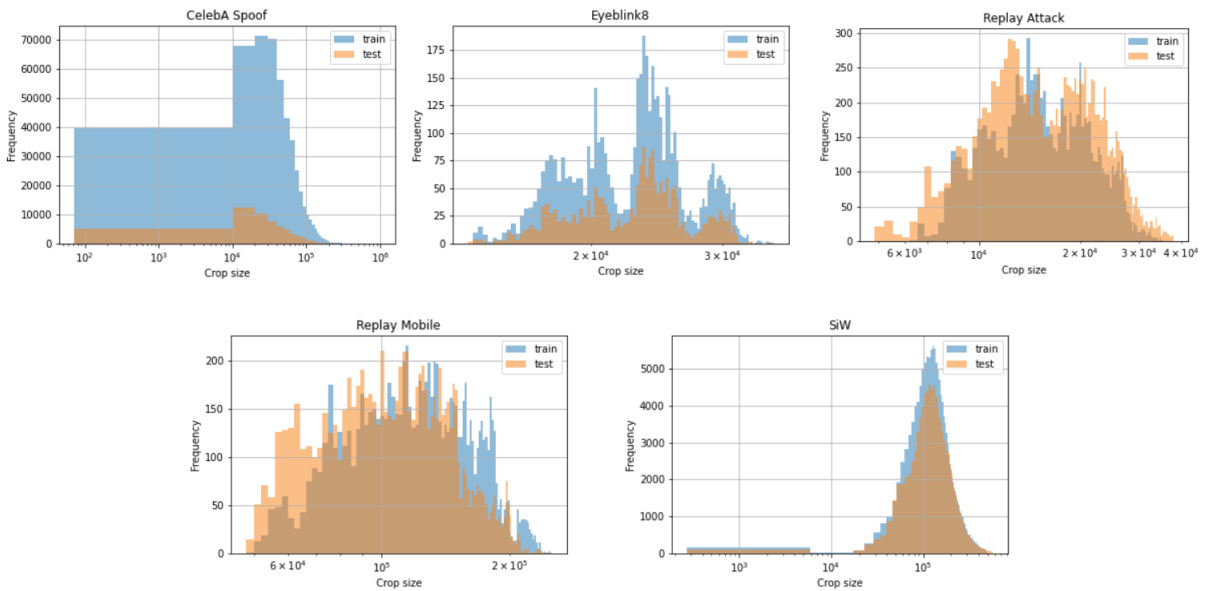












Figure 29: Crops distribution for train and test splits for all datasets.

runs.summary["inspection"]

	image	target	prediction	confidence	loss	classified	dataset
134		1	0	0.7386	0.9599	fn	/data-0/siw_processed/
135		1	0	0.9594	1.255	fn	/data/CelebA_Spoof_/CelebA_Spoof/
136		1	0	0.9979	1.31	fn	/data-0/siw_processed/
137		1	0	0.7016	0.9149	fn	/data/CelebA_Spoof_/CelebA_Spoof/
138		1	0	0.978	1.281	fn	/data/CelebA_Spoof_/CelebA_Spoof/
139		1	0	0.9137	1.19	fn	/data-0/replay-attack_processed/
140		1	0	0.583	0.7796	fn	/data/CelebA_Spoof_/CelebA_Spoof/
141		1	0	0.8925	1.161	fn	/data/CelebA_Spoof_/CelebA_Spoof/
142		1	0	0.8841	1.149	fn	/data/CelebA_Spoof_/CelebA_Spoof/
143		1	0	0.6873	0.8979	fn	/data-0/replay-attack_processed/

← < 134 - 143 of 200 > →

Figure 30: Evaluation inspection table.

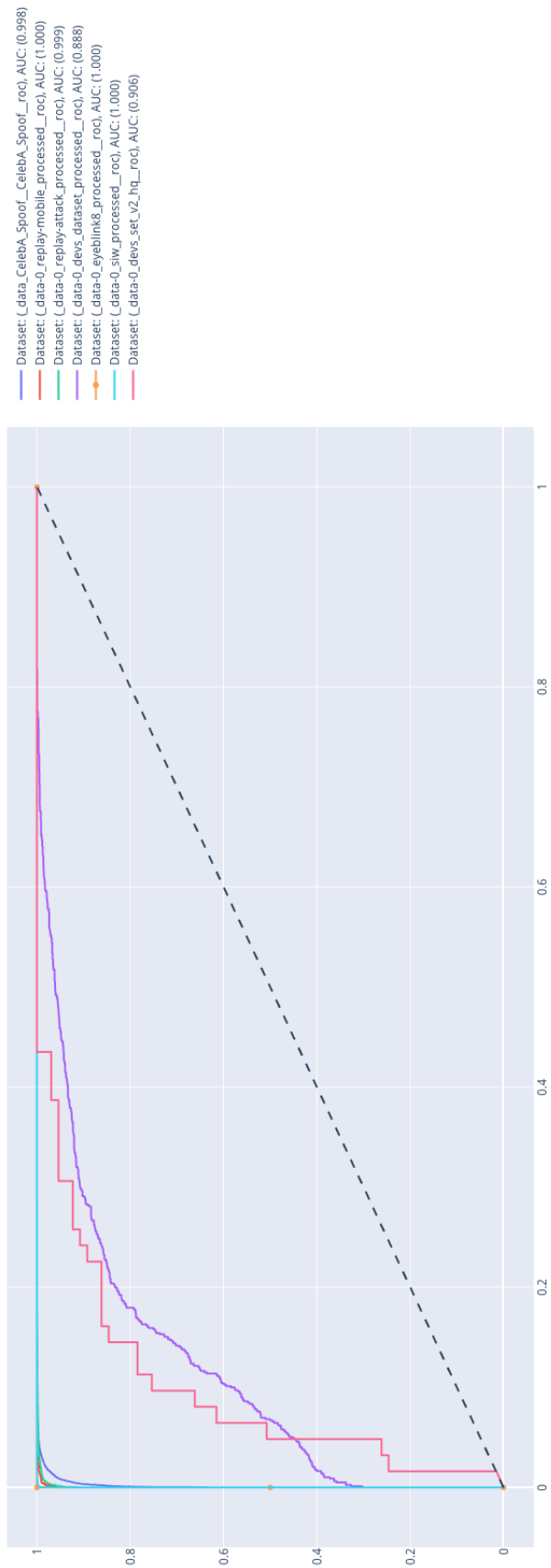


Figure 31: Receiver operating characteristic curve for datasets soup with Resnet18 MTL.

```
runs.summary["rates"]
```

	dataset_name	fnr	tnr	fpr	tpr	accuracy	f1
1	/data-0/eyeblink8_processed/	0.5	1	0	0.5	0.9995	0.6667
2	/data-0/devs_set_v2_hq/	0.3692	0.9194	0.08065	0.6308	0.7717	0.7387
3	/data/CelebA_Spoof_/CelebA_Spoc	0.07349	0.9946	0.005421	0.9265	0.9467	0.9607
4	/data-0/devs_dataset_processed/	0.3212	0.8658	0.1342	0.6788	0.7388	0.7793
5	/data-0/replay-mobile_processed/	0.01454	0.996	0.004027	0.9855	0.9893	0.9915
6	/data-0/replay-attack_processed/	0.04117	0.998	0.001974	0.9588	0.9684	0.9787
7	/data-0/siw_processed/	0.002236	0.9977	0.002291	0.9978	0.9977	0.998
8	combined	0.03587	0.9952	0.004841	0.9641	0.9768	0.9791

← < 1 - 8 of 8 > →

Figure 32: False Positive, False Negative, True Positive and True Negative rates, datasets soup with Resnet18 MTL.

runs.summary["eer\_report"]

	dataset_name	eer	eer_threshold	optimal_threshold	tpr
1	_data_CelebA_Spoof__CelebA_Spoof_	0.02011	0.01827	0.01186	0.9838
2	_data-0_replay-mobile_processed_	0.009045	0.215	0.3182	0.99
3	_data-0_replay-attack_processed_	0.01166	0.03188	0.04203	0.9866
4	_data-0_devs_dataset_processed_	0.1866	0.07454	0.03033	0.8409
5	_data-0_eyblink8_processed_	0	0.4337	0.4337	1
6	_data-0_siiv_processed_	0.002261	0.5076	0.5546	0.9976
7	_data-0_devs_set_v2_hq_	0.1576	0.05279	0.06301	0.8462
8	combined	0.01079	0.3102	0.3393	0.9905

← < 1 - 8 of 8 > →

Figure 33: Equal Error Rates for datasets soup with Resnet18 MTL.