**REPUBLIC OF TURKEY**
**AKDENIZ UNIVERSITY**

# INVESTIGATION OF SHADE FEATURES UNDER IR ILLUMINATION FOR FACIAL BIOMETRIC AUTHENTICATION SYSTEMS

**Mehmet Ali Osman ATİK**

**INSTITUTE OF NATURAL AND APPLIED SCIENCES**

**DEPARTMENT OF COMPUTER ENGINEERING**

**MASTER THESIS**

**JUNE 2023**
**ANTALYA**

**REPUBLIC OF TURKEY**
**AKDENIZ UNIVERSITY**

# INVESTIGATION OF SHADE FEATURES UNDER IR ILLUMINATION FOR FACIAL BIOMETRIC AUTHENTICATION SYSTEMS

**Mehmet Ali Osman ATİK**

**INSTITUTE OF NATURAL AND APPLIED SCIENCES**

**DEPARTMENT OF COMPUTER ENGINEERING**

**MASTER THESIS**

**JUNE 2023**
**ANTALYA**

**REPUBLIC OF TURKEY**
**AKDENIZ UNIVERSITY**
**INSTITUTE OF NATURAL AND APPLIED SCIENCES**

# INVESTIGATION OF SHADE FEATURES UNDER IR ILLUMINATION FOR FACIAL BIOMETRIC AUTHENTICATION SYSTEMS

**Mehmet Ali Osman ATİK**

**DEPARTMENT OF COMPUTER ENGINEERING**

**MASTER THESIS**

**JUNE 2023**

**REPUBLIC OF TURKEY**
**AKDENIZ UNIVERSITY**
**INSTITUTE OF NATURAL AND APPLIED SCIENCES**

# INVESTIGATION OF SHADE FEATURES UNDER IR ILLUMINATION FOR FACIAL BIOMETRIC AUTHENTICATION SYSTEMS

**Mehmet Ali Osman ATİK**

**DEPARTMENT OF COMPUTER ENGINEERING**

**MASTER THESIS**

This thesis was accepted unanimously by the jury on 21/07/2023.

Asst. Prof. Dr. Mustafa Berkay YILMAZ (Supervisor)

Asst. Prof. Dr. Murat AK

Prof. Dr. Cafer Çalışkan

# ÖZET

## YÜZ BİYOMETRİK DOĞRULAMA SİSTEMLERİ İÇİN KIZILÖTESİ AYDINLATMA ALTINDA GÖLGE ÖZELLİKLERİNİN ARAŞTIRILMASI

**Mehmet Ali Osman ATİK**

**Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı**

**Danışman: Dr. Öğr. Üyesi Mustafa Berkay YILMAZ**

**Haziran 2023; 44 sayfa**

Baskı teknolojilerindeki ve yüksek çözünürlüklü ekranlardaki gelişmelerle birlikte, sunum saldırılarının mevcut yöntemlerle tespit edilmesi zorlaşmaktadır. Bu saldırılara karşı üç boyutlu yüz yapısı çıkarımı, canlılık tespitinde umut vadeden bir yöntem gibi görünmektedir. Bu tezde de üç boyutlu yüz yapısının gölgelenmesinden yararlanan basit bir yaklaşım önerilmiştir. Bu çalışma dahilinde geliştirilen bir Unity uygulaması olan unityStudio aracılığıyla üretilmiş sentetik bir veri setleri kullanılarak, yüzdeki aydınlanma yönünü tahmin edecek şekilde CNN modelleri eğitilmiştir. Ortam aydınlatmasını bastırmak ve baskın bir aydınlanma yönü oluşturmak için ikili bir IR LED düzeni kullanılmıştır. Bu tezde temel olarak, model yanıtı ile aydınlatma yönü bilgisi karşılaştırılarak, bir üç boyutlu yüz yapısının varlığının doğrulanabileceği düşüncesi savunulmuştur. Popüler CNN modelleri ve basit CNN uygulamaları üzerinde deneyler yapılarak başarılı CNN modelleri oluşturulmuş ve bu modellerin ve gerçek dünyadaki performansları test edilerek, gerekli eleştiriler sunulmuştur.



**ANAHTAR KELİMELER:** Canlılık Tespiti, CNN, Yüz Tanıma, Yapay Veri Kümesi Üretimi, Unity

**JÜRİ:** Dr. Öğr. Üyesi Mustafa Berkay YILMAZ

Asst. Prof. Dr. Murat AK

Prof. Dr. Cafer Çalışkan

## ABSTRACT

## INVESTIGATION OF SHADE FEATURES UNDER IR ILLUMINATION FOR FACIAL BIOMETRIC AUTHENTICATION SYSTEMS

**Mehmet Ali Osman ATİK**

**MSc Thesis in Computer Engineering**

**Supervisor: Asst. Prof. Dr. Mustafa Berkay YILMAZ**

**June 2023; 44 pages**

With the advancements in printing technologies and high-resolution screens, planar presentation attacks are becoming hard to detect with current methodologies. 3D facial structure extraction seems to be a promising way to go in Liveness detection. In this thesis, a simple approach has been proposed that benefits from the shading of 3D facial structure. CNN models were trained to estimate the lightning direction on the face by using a synthetic dataset generated for this purpose with a Unity application developed: unityStudio. A dual IR LED setup is used to suppress environmental lighting and to create a dominant lightning direction. The thesis is, by comparing the model response with prior knowledge of lightning direction, the presence of a 3D face structure will be able to be confirmed. Experiments were made on state-of-the-art CNN models and simple CNN implementations. Successful CNN models are trained on synthetic data and their real-world performances were criticized.



**KEYWORDS:** CNN, Facial Recognition, Liveness Detection, Synthetic Dataset Generation, Unity

**COMMITTEE:** Asst. Prof. Dr. Mustafa Berkay YILMAZ

Asst. Prof. Dr. Murat AK

Prof. Dr. Cafer Çalışkan

## ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my precious wife Elif, who supported me and my work throughout this process, and to my son Pamir Tuna, who has to make concessions on father-son game hours.

I would like to thank my supervisor Asst. Prof. Dr. Mustafa Berkay YILMAZ for his contribution to the emergence of this study by encouraging me to work in this field with his informative lectures and guidance through my process.

I also thank Prof. Dr. Melih GÜNAY for his academic advancement advice and endless support, and my all-lecture teachers for enlightening me with their knowledge.

## LIST OF CONTENTS

## TEXT OF OATH

I declare that this study *"Investigation of Shade Features Under IR Illumination for Facial Biometric Authentication Systems"*, which I present as my master thesis, is in accordance with the academic rules and ethical conduct. I also declare that I cited and referenced all material and results that are not original to this work.

23 / 06 / 2023

Mehmet Ali Osman ATİK

## LIST OF ABBREVIATIONS

2D : Two Dimensional
3D : Three Dimensional
AE : Auto Exposure
CM : Confusion Matrix
CNN : Convolutional Neural Network
CV : Computer Vision
DNN : Deep Neural Network
FC : Fully Connected
HP : Head Pitch Range
HR : Head Roll Range
HY : Head Yaw Range
IR : Infrared
LBP : Local Binary Patterns
LD : LED Distance
LI : LED Intensity
MLP : Multilayer Perceptron
NIR : Near Infrared
PAD : Presentation Attack Detection
ReLU : Rectified Linear Unit
ROC : Receiver Operator Curve
SS : Skin Smoothness
SVM : Support Vector Machine
UV : Ultraviolet

Decimal representation : 3.14

## LIST OF FIGURES

## LIST OF TABLES

# 1. INTRODUCTION

Biometric authentication mechanisms are widely used in today's digital security systems for ensuring privacy. Today's common biometric authentication methods can be shortlisted as signature recognition, fingerprint recognition, hand geometry recognition, face recognition, voice recognition, iris recognition, retinal recognition, DNA matching, and behavioral biometrics. Besides other biometric recognition mechanisms, facial recognition systems require almost no cooperation from the user and can even be done from a distance (Xin et al. 2017). Thus, facial recognition systems feel natural and highly accepted by society. With these features, facial recognition is one of the most studied biometric recognition mechanisms to detect the identity of users to grant secure access to confidential assets. Various facial recognition methods have been proposed and implemented successfully in the literature (Kortli et al. 2020).

As facial recognition systems became widespread, numerous spoofing attack methods began to be developed by malicious people. These attacks may target many steps in face recognition systems. Since attacking most of these steps requires technical knowledge of the system, a particular attack type is more popular than the rest: the attacks on the sensor itself, i.e., the camera, called "presentation attacks" in literature. These attack types can be summarized in three categories:

- Printed or digital photograph presentation attacks.
- Recorded video presentation attacks.
- 3D realistic mask presentation attacks.

Figure 1.1 shows an illustration of these attack types (Imageware Systems, 2020).

Fortunately, many Presentation Attack Detection (PAD) mechanisms are being developed by scientists as countermeasures for these attack types. Even better, studies are being made to ensure that the biometric samples are taken from a living subject, which is called liveness detection and considered a subset of PAD techniques. These mechanisms can be grouped under hardware-based and software-based methods (Akhtar et al. 2015; Ramachandra & Busch 2017; Xin 2017; Dahia et al. 2020). Details of these methods will be given in Section 2.


**Figure 1.1.** Common presentation attack types

In this thesis, a naive approach will be tested for determining the existence of the subject, which is also fusible with other techniques if needed. The idea is to indirectly detect the existence of a 3D face structure in front of the camera from the shade features created by a light source, a LED flash, placed close to the camera. This setup is chosen on purpose to simulate lightning from a LED flash placed on a mobile phone or similar handheld device. Considering the close distance of handheld devices an infrared (IR) LED flash illumination was introduced to comfort the users from the powerful lighting required to capture shadow features from the image in both daylight and night environment conditions. IR illumination can also reveal extra features which can be used by some of the software methods mentioned above (Kakadiaris 2005). A background survey of common techniques in literature and approaches similar to this will be given in Section 2. Unfortunately, there was no suitable dataset available for the specific purposes of this experiment, and creating a real dataset was not feasible so, the need for a synthetic dataset creation has been urged. Details of this dataset and creation method will be given in Section 3. As the next step, basic convolutional neural network (CNN) models and some existing models have been trained on the synthetic dataset to detect the direction of light sources placed close to the camera. Finally, using these trained models for estimating the light source position and comparing it with the prior knowledge of light source position, it will be possible to tell if there is a real 3D face structure present in front of the camera. Details of the network models will also be given in Section 3. Experiment results and model performances will be discussed in Section 4. The conclusion part and research opportunities will be given in Section 5.

## 2. LITERATURE REVIEW

As mentioned before, PAD techniques could be grouped under hardware-based and software-based methods.

Hardware-based methods benefit heavily from specialized equipment and a dedicated setup which usually require users to cooperate with the system. Some of the hardware-based equipment used in the PAD systems can be enlisted as near-infrared (NIR) cameras, thermal imaging cameras and multispectral imaging cameras, 3d reflectance scanners, and light field (plenoptic) cameras which require specialized software in addition to render final images (Ramachandra & Busch 2017).

Software-based methods can also be grouped into two categories: static methods and dynamic methods. Static methods inspect a single frame from a sequence of images to detect presentation attacks. These static methods include the extraction of micro textures, which are high-frequency features, using Local Binary Patterns (LBP) (Parveen et al 2015; Chen & Zhang 2018), Fourier spectrum analysis, which is another method for extracting the high-frequency features (Adamiak 2015; Akhtar et al. 2015) and diffusion speed measurement (Xin et al. 2017). On the other hand, dynamic methods inspect a sequence of consecutive images to detect presentation attacks. Dynamic methods include various eye blink detection methods, optical flow-based head, lip and eye movement detection, and Eulerian magnification methods which can be used to amplify and reveal the color changes in the face or micro head movements due to blood ventilation (Ramachandra & Busch 2017; Tsitiridis 2019). There are also challenge-response methods that require user cooperation scenarios which can be counted under dynamic methods.

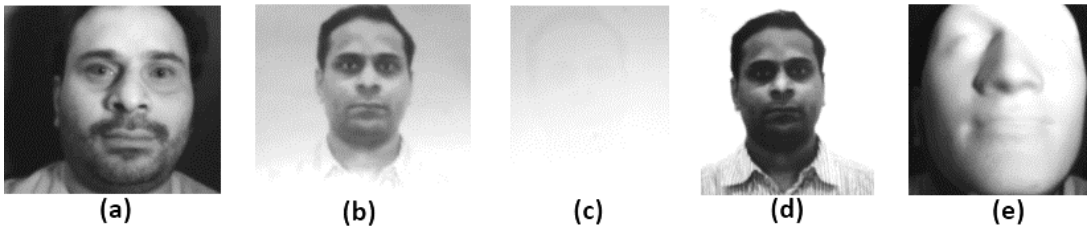



**Figure 2.1. a)** Real person; **b)** Laser printer; **c)** Inkjet printer; **d)** iPad; **e)** 3d mask

Using different spectrums of light can reveal precious features for facial recognition systems. IR is one of the most studied spectrums which can extract features such as blood veins under the face skin, and non-vivid colors of printed images. Figure 2.1. demonstrates the NIR light responses of a real face and some presentation attack methods (Ramachandra & Busch 2017). UV lightening is also considered in many studies which is also known to reveal extra biometric features but in the case of usage of sunscreen or makeup, systems using UV Spectrum were reported to present a poor performance (Samatzidis 2018). Figure 2.2. shows the effects of sunscreen applied on the face in the UV Spectrum.

**Figure 2.2.** Effects of sunscreen on the UV Spectrum

### 2.1. Related works

There are numerous studies made to reveal the 3D structure of the face for use in PAD and a few of them will be mentioned here that are somehow close to the method used in this thesis.

Wang et al. (2013) proposed the extraction of 3D face structure from a video sequence by detecting facial landmarks of different head positions and using a Support Vector Machine (SVM) classifier to make a separation between real and fake faces. They claim that their method achieves 100% accuracy on planar and warped presentation attack methods. Although it was not stated clearly in the paper, this method seems to be vulnerable against 3D realistic mask attacks.

In another study, instead of extracting 3D features Chan et al. (2018) uses two images of the subject, one with under flash illumination and one without, and by using LBP descriptors, differences between the two images have been calculated to classify the images. The paper claims a successful improvement in liveness detection. Although

flash illumination is involved in this study, it does not benefit from the 3D structure of the faces.

Di Martino (2021) also states that 2D features are easier to exploit and proposes a 3D shading-based facial feature extraction which uses a generalized Lambertian model for avoiding the hardware complexities and computing power of actually reconstructing the face geometry. Two images are captured for this purpose and one of them is captured under an ambient flash coming from relatively high angles i.e., 15 to 60 degrees, and trained SVM and Deep Neural Network (DNN) models are used to detect whether this pair was a genuine or spoofing example.

The study of Zhang et al. (2016) is partially similar to this one. In the proposed method, firstly lightening direction of the environment is estimated from a primarily taken image. Then a new lightning was introduced from the opposite direction possible and the expected illumination on the face was reconstructed from a 3D morphable model. Finally, the combined result is compared with the secondary image taken. Since the illumination of planer attacks will not be changed these spoofing methods are detected but a 3D mask still fools this technique.

### 2.2. Convolutional Neural Networks

CNNs' are a highly used neural network structure that is giving successful results in many Computer Vision (CV) studies such as image classification. A CNN consists of three main layers:

- Convolutional layer: this layer extracts the features from the input image for all channels by scanning the image with a convolution kernel which is usually a 3*3 filter. After each convolution layer, an activation layer is applied to introduce a non-linearity to the network. This activation function can be thought of as a measure of the importance of that feature for the model. The function can be a smooth non-linear function such as $tanh$ or $sigmoid$ function but in CNN architectures usually, a $ReLU$ function is used which is a linear function.

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.1}$$

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

$$\text{relu}(x) = \max(0, x) \tag{2.3}$$

- Pooling layer: this layer is used for reducing the dimensionality of features generated by the previous convolutional layer. It can be thought of as a kind of subsampling with a pre-defined size filter. Two methods are common for this layer. The first one is Max pooling which takes the max value in the filtered feature map. The other one is Average pooling which calculates the average of the filtered feature area as output.
- Fully Connected (FC) layer: this layer is built just like a Multilayer Perceptron (MLP) where every neuron from the input layer is connected to every other neuron in the next layer. The classification process is done in these layers by training the network.

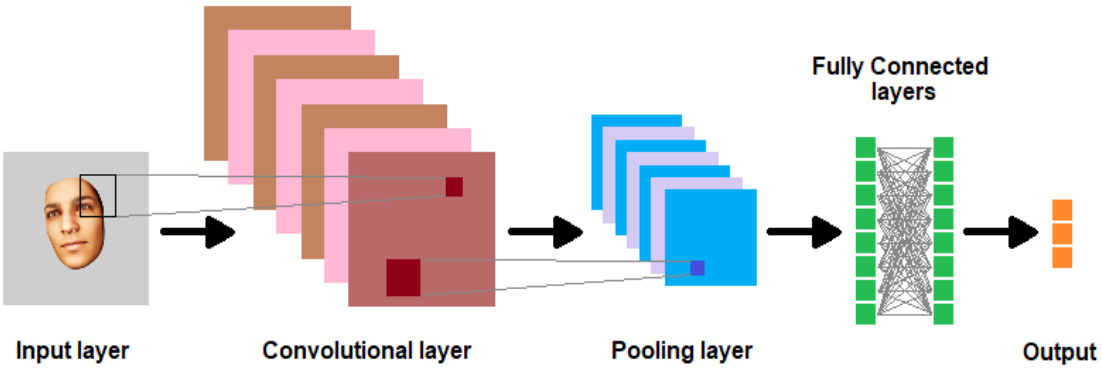



**Figure 2.3.** Basic CNN architecture

In addition to these three main layers usually, a Dropout layer is applied after the FC layer to randomly drop some neurons from contributing to the model to prevent overfitting the training data and make a good generalization. On the output layer, the model gives the probabilities for each class. Figure 2.3. show the basic architecture of a CNN.

In more complex models, convolution and pooling layers can be concatenated, hidden layers can be added to the FC layer, and FC + Dropout layers can be concatenated for the specific needs of the model. Besides the network architecture, there are some hyperparameters used to optimize the model in various aspects:

- Image size: large images may provide more details but also require more computational power and time for training.
- The number of categories: this indicates the number of classes that the model should classify, training data should also contain samples from all classes.
- Epochs: each epoch consists of a forward pass through the network and a backpropagation phase where the weight is updated for all samples in the dataset. Increasing epochs may improve the model performance but too many epochs may also lead to overfitting.
- Batch size: this defines how many samples are processed in one epoch. By using larger batch sizes model can be trained faster but requires more memory space. On the other hand, small batch sizes provide better updates for each sample.
- Learning rate: this is a multiplier for the weight updates. Small values result in slow but accurate convergence. Big values may result faster but also may miss the optimal solution.
- Dropout rate: this determines the fraction of input units to be dropped.
- Convolution filters: The number of convolution filters used. More filters may extract more features with the cost of computational power and time.
- Convolution kernel size: Dimensions of the filters. Large filters are used for extracting bigger features, smaller filters are used for extracting local features.
- Convolution strides: this is the step size of the convolution filter. Using bigger strides reduces the size of output with the cost of some information loss.
- Pool size: this is the window size of the down-sampling filter used for dimension reduction. Higher pool sizes will cause smaller outputs.
- Pool strides: this is the step size for the down-sampling filter.
- Dense Units: This determines the number of neurons in the FC layer.
- Activation function: this function determines the response of each neuron. The choice of this function may highly impact the performance of the trained model.

Some of the popular CNN architectures for image classification used in this thesis for comparison purposes are AlexNet (Krizhevsky et. Al 2017), DenseNet (Huang et al. 2017), MobileNet (Howard et al. 2017), ResNet (He et al. 2016), VGGNet (Simonyan & Zisserman 2014) and Xception (Chollet 2017). A comparison of these architectures with default hyperparameters setup is given in Table 2.1.

**Table 2.1.** Feature comparison of introduced CNN architectures

| Parameters | AlexNet | DenseNet | MobileNet | ResNet | VGGNet | Xception |
|---|---|---|---|---|---|---|
| Image Input Size | 227x227 | 227x227 | 224x224 | 224x224 | 224x224 | 299x299 |
| Conv Layers | 5 | Varies | Varies | Varies | 16-19 | Varies |
| Kernel Sizes | 11x11, 5x5 | 3x3 | 3x3 | 7x7, 3x3 | 3x3 | 3x3 |
| Conv Filters | 96, 256, 384, 384, 256 | Varies | Varies | Varies | Varies | Varies |
| Conv Strides | 4, 1, 1, 1, 1 | 1 | 1 | 2, 1 | 1 | 1 |
| Pool Sizes | 3x3 | 2x2 | 2x2 | 2x2 | 2x2 | 2x2 |
| Pool Strides | 2, 2, 2, 2, 2 | 2 | 2 | 2 | 2 | 2 |
| Dense Units | 4096 | Varies | 1024 | 1000 | 4096 | 2048 |
| Activation Function | ReLU | ReLU | ReLU6 | ReLU | ReLU | ReLU |
| Dropout Rate | 0.5 | 0.2-0.5 | 0.25 | 0.0-0.5 | 0.0-0.5 | 0.0-0.5 |
| Batch Size | 128 | 32-128 | 32-128 | 32-256 | 32-128 | 32-128 |
| Learning Rate | 0.01 | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 |
| Epochs | 90 | 10 | 10 | 50 | 10-100 | 10-100 |
| Maximum Categories | 1000 | Varies | Varies | Varies | Varies | Varies |

### 2.3. 3D face models and datasets

Unfortunately, there are not too many 3D faces or 3D head datasets available freely for academic use. Most known ones can be listed as:

- Basel Face Model (Paysan et al. 2009) is a face model generated from 200 individuals. It is a morphable model and creates a basis for face generation. There are newer versions of this model presented over the years.
- FaceWarehouse (Cao et al. 2013) is a 3D facial expression dataset captured from 150 people of various ages and ethnicity.
- FLAME (Li et. Al 2017) is another head model that can be used to generate face models from 2D images. Flame is generated from 3800 4D scans of human expressions and a total of 33000 scans.
- FaceScape (Yang et al. 2020) is a recent dataset containing 18760 hi-detailed 3D face models generated from 938 individuals from ages 16 to 70 and each presenting 20 different facial expressions. This one also has a model for predicting a face model from 2d images.
- Deep3DFaceReconstruction (Deng et al. 2019) is a pre-trained model for 3D face reconstruction from single 2D images. This model uses Basel Face Model and FaceWarehouse expressions basis for generating 3D face models. It is also trainable for custom models.

Fortunately, there are tons of image datasets available for researchers freely. In this work, the Academic Dataset by Generated Photos, which is another synthetic dataset for faces, is selected. The dataset consists of 10000 face images from a variety of ethnicities, ages, and genders. Each image comes with a metafile containing facial feature landmarks (mouth, right eyebrow, left eyebrow, right eye, left eye, nose, jaw) and facial attributes (head pose, gender, makeup, emotion, facial hair, hair “hair color, hair length, bald”, occlusion, ethnicity, eye color, smile, age).

# 3. MATERIAL AND METHOD

Briefly, the Deep3DFaceReconstruction pipeline was selected for generating 3D head models of the images from the Academic Dataset by Generated Photos. Then, MeshLab filters are used to extract the texture from the pipeline result and to optimize the generated model. Finally, unityStudio, a Unity app developed within this thesis for generating datasets, is used to generate datasets for training various CNN models. All the work in this thesis is done on a desktop computer with the specifications below:

- Processor : Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz
- Ram : Kingston 16 GB DDR3 1866 MHz Dual
- GPU : NVIDIA GeForce GTX 1060 6GB
- Motherboard : ASUSTeK COMPUTER INC. Z87-A
- OS disk : Corsair Neutron XTI SSD 240 GB
- Data disk : Seagate Barracuda Pro 10 TB
- OS : Microsoft Windows 10 Pro 64 Bit Build 19045

## 3.1. Synthetic Dataset Generation

### 3.1.1. Image qualification and preprocessing

Some selection and restriction parameters were defined to fine-select the images that will be used in model creation:

- Age: a minimum age of 15 is selected to eliminate baby and child faces.
- Head pitch: a range between -12 to 18 degrees selected for preserving the frontal presentation of faces.
- Head roll: a maximum of 10 degrees of head rolling on both sides is allowed for easier face detection.
- Head yaw: head rotations are restricted to a maximum of 18 degrees to ensure both sides of the face are visible for generating accurate shadows from the facing light sources.
- Image bug: this represents the bug probability in the synthetic dataset, a threshold of 0.2 was sufficient for most of the cases.

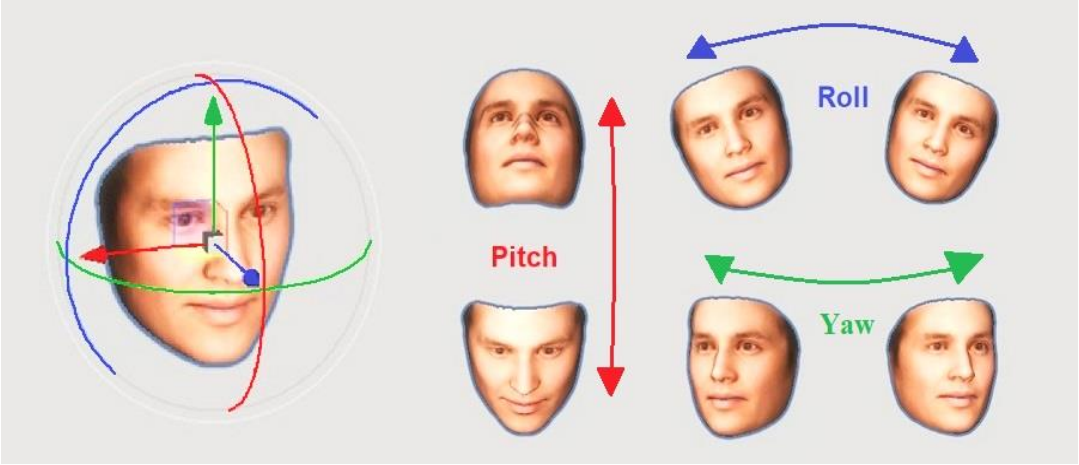



**Figure 3.1.** Head orientations

Figure 3.1 shows the head rotation axes that are restricted in the final selected image sets. Some faces were occluded by glasses that were not wanted for texture generation. Some images and corresponding metadata were not matching in facial landmarks. Thus, visual confirmation of each image was necessary. Apart from this, the Deep3DFaceReconstruction pipeline required specific 5-point landmark features to be supplied for each image. For this purpose, a Python script, *check_landmarks.py* was written (see Figure A1). With the aid of a script, 10 different subsets were visually selected from the dataset each containing 512 images and corresponding 5-point metadata that were extracted and recalculated from original meta files. Figure 3.2. represents the gender and head pose distributions of selected images dataset.

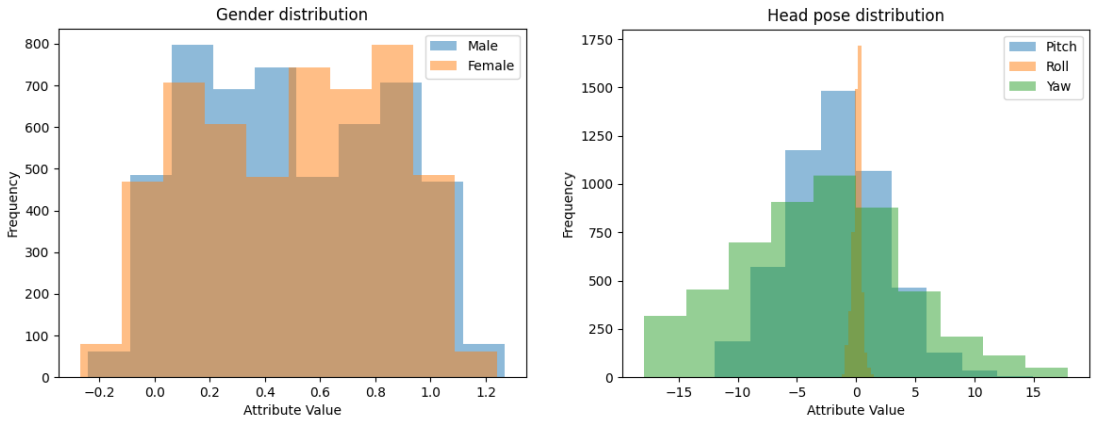



**Figure 3.2.** Gender and head orientation distributions of the image dataset

### 3.1.2. 3D face model generation with Deep3DFaceReconstruction

As mentioned above, Deep3DFaceReconstruction is used for constructing 3D face objects from the processed images and extracting 5-point facial landmarks. However, it was mentioned that a better PyTorch implementation is available, which is easier to use and has a much better performance, building that implementation failed because of inconsistent dependencies. So, the earlier version of the pipeline implementation is used in this thesis. The pipeline requires the use of Basel Face Model 2009 (BFM09) and Expression Basis from Facewarehouse which are both free to use for academic purposes but require a request from an academic domain. Supplied demo code was modified for batch processing of image datasets prepared earlier and corresponding 3D models in .obj format were created in a new folder. Although generated models were in .obj format they were not directly usable in Unity, because there was no texture file generated but colored vertices were produced instead by the pipeline. Thus, a conversion was needed.

### 3.1.3. Model preprocessing with PyMeshLab

MeshLab (*Maggiordomo et. Al 2021*) has powerful tools for 3D processing. As the next step three MeshLab filters needed to be applied consecutively for converting the colored vertices to defragmented textures in .png format. Additionally, a custom export operation was necessary. All required steps were:

1. File > Import mesh…
2. Filters > Texture > Parametrization: Trivial Per-Triangle
3. Filters > Texture > Transfer: Vertex Attributes to Texture
4. Filters > Texture > Texture Map Defragmentation
5. File > Export mesh…

Then, the optimized mesh and generated texture were ready to be used. Since there were 5120 models that needed to be processed the same way, a batch processing code, *meshlab_batch_process.py*, has been coded by using *PyMeshLab* (Muntoni & Cignoni 2021) which is a Python interface for MeshLab. The related code section for the required steps for transformation is given below.

```
try:
    ms = pymeshlab.MeshSet()
    ms.load_new_mesh(modelPath)
    ms.compute_texcoord_parametrization_triangle_trivial_per_wedge()
    ms.transfer_attributes_to_texture_per_vertex(textname=model[:-4])
    ms.apply_texmap_defragmentation()
    ms.save_current_mesh(savePath, save_textures=True, texture_quality=-1)
```

In the texture map defragmentation step the prefix *"texdefrag_"* and the suffix *"_optimized_texture_0"* are added to the model file name, and generated texture is saved with the .png extension. Processing each 3D model set of 512 models generated from matching image set has taken about 7 hours and 30 minutes (see Figure A2).

### 3.1.4. Dataset generation with unityStudio

A simulation of a photo studio designed in Unity in order to render custom illuminated faces. Figure 3.2 shows an illustration of the designed scene where LD denotes the distance between the camera center and the center of LED flashes, and CD denotes the distance of the camera to the center of the face model.

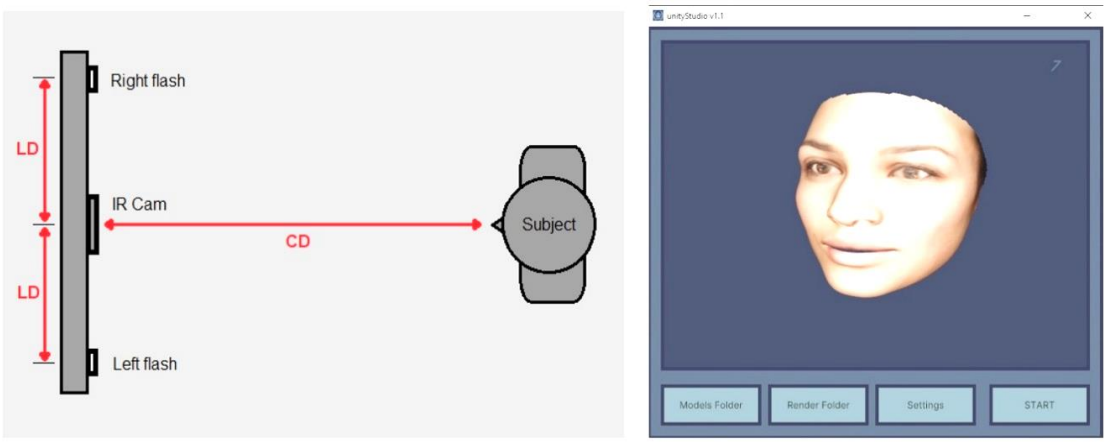



**Figure 3.3 a)** Scene design; **b)** unityStudio rendering faces

Since multiple variations of face model distances, LED flash distances, LED intensities, and head orientations would be needed to generate a sparse dataset, a helper application was developed for generating a custom dataset from given models: unityStudio (Atik 2023). Graphical user interfaces of the application are presented in Figure A3. Usage of the application is straightforward:

1. Select the model's folder containing 3D models and matching texture files via the "Models Folder" button. The application assumes that the matching texture file names contain the prefix and the suffix mentioned above and have the .png file format. i.e., *texdefrag_3dModelFileName_optimized_texture_0.png*
2. Optionally, select a destination folder for the render outputs. If not, the application creates a default output folder in the same directory as the selected models' folder by adding a "_Renders" suffix to the model folder's name.
3. Use the "Settings" button to change the default settings of the application for generating custom datasets. Table 3.1 shows the options on the settings page, practical ranges, and default values.
4. Hit the "START" button to engage the batch render process.

unityStudio will generate three face images for every model in the selected folder with the default or custom settings:

- An image illuminated with the left flash and added the suffix "_Left"
- An image illuminated with the right flash and added the suffix "_Right"
- An image illuminated with both flashes and added the suffix "_Dual"

Generated images will be saved in .png format with a clean *alpha channel*. Thus, a custom background can be added to simulate different environments (See Figure A.4).

There was already a randomness coming from the image dataset that is defined in the first qualification phase. This could be removed by using the orientation values provided in the accompanying metadata files, but it is kept to preserve the originality of the dataset. Either way, the application allows defining a custom randomization on top of it which might become handy if different kinds of datasets need to be created.

Auto exposure mode makes the camera adapt the lighting in the scene and creates a visible face unless LED intensities are set to extreme values out of the

reasonable range. This can be turned off to simulate weak lightening or bright lightening which can be used to generate high contrast between the shaded and enlightened regions of the face models.

**Table 3.1.** unityStudio settings

| Features | | Ranges | Default | Type |
|---|---|---|---|---|
| Camera Distance | (CD) | 20 – 100 | 30 | cm |
| LED Distance | (LD) | 20 – 100 | 30 | mm |
| LED Intensity | (LI) | 10 – 1000 | 50 | lux |
| Head Pitch Range | (HP) | 0 – 20 | 10 | degree |
| Head Yaw Range | (HY) | 0 – 20 | 10 | degree |
| Head Roll Range | (HR) | 0 – 20 | 10 | degree |
| Skin Smoothness | (SS) | 0 – 1 | 0.2 | float |
| Auto Exposure | (AE) | On - Off | On | bool |

### 3.1.5. Generated datasets

There are two different datasets generated with the unityStudio: the *mixedSet* and the *brightSet.*

The *mixedSet* has been generated with the auto-exposure mode set to *On* for all samples to simulate the standard behavior of commercial cameras. Also, head orientations are kept in a narrower range. Table 3.2. shows the preferred settings to generate the *mixedSet* dataset.

The *brightSet* has been generated with the auto-exposure mode set to *Off* for all samples to simulate a powerful flash causing high contrast between lightened and shaded areas of the face. Also, the head orientation range was expanded for some sets to enhance shade features. Table 3.3. shows the modified settings to generate the *brightSet* dataset. Figure 3.4. presents the effect of the auto-exposure feature on the generated images from the same model.

**Table 3.2.** unityStudio settings of generated *mixedSet*

| Set | CD | LD | LI | HP | HY | HR | SS | AE |
|---|---|---|---|---|---|---|---|---|
| 512 A | 30 | 30 | 30 | 0 | 0 | 0 | 0.2 | On |
| 512 B | 30 | 30 | 50 | 5 | 5 | 5 | 0.2 | On |
| 512 C | 30 | 30 | 100 | 5 | 5 | 5 | 0.2 | On |
| 512 D | 30 | 50 | 30 | 0 | 0 | 0 | 0.2 | On |
| 512 E | 30 | 50 | 50 | 5 | 5 | 5 | 0.2 | On |
| 512 F | 30 | 50 | 100 | 5 | 5 | 5 | 0.2 | On |
| 512 G | 30 | 75 | 30 | 0 | 0 | 0 | 0.2 | On |
| 512 H | 30 | 75 | 50 | 5 | 5 | 5 | 0.2 | On |
| 512 I | 30 | 75 | 100 | 5 | 5 | 5 | 0.2 | On |
| 512 K | 30 | 100 | 50 | 0 | 0 | 0 | 0.2 | On |

**Table 3.3.** unityStudio settings of generated *brightSet*

| Set | CD | LD | LI | HP | HY | HR | SS | AE |
|---|---|---|---|---|---|---|---|---|
| 512 A | 30 | 30 | 300 | 0 | 0 | 0 | 0.2 | Off |
| 512 B | 30 | 30 | 500 | 5 | 5 | 5 | 0.2 | Off |
| 512 C | 30 | 40 | 1000 | 10 | 10 | 10 | 0.2 | Off |
| 512 D | 30 | 50 | 300 | 0 | 0 | 0 | 0.2 | Off |
| 512 E | 30 | 50 | 500 | 5 | 5 | 5 | 0.2 | Off |
| 512 F | 30 | 60 | 1000 | 10 | 10 | 10 | 0.2 | Off |
| 512 G | 30 | 70 | 300 | 0 | 0 | 0 | 0.2 | Off |
| 512 H | 30 | 80 | 500 | 5 | 5 | 5 | 0.2 | Off |
| 512 I | 30 | 90 | 1000 | 10 | 10 | 10 | 0.2 | Off |
| 512 K | 30 | 100 | 500 | 0 | 0 | 0 | 0.2 | Off |

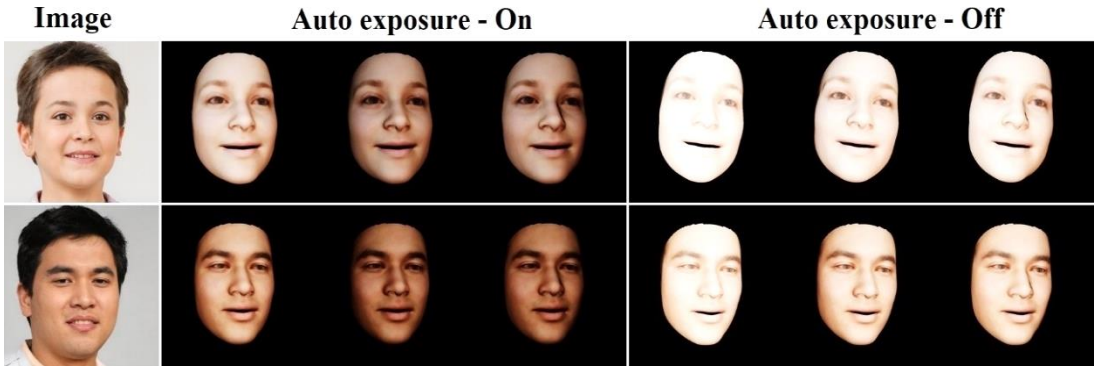



**Figure 3.4.** Auto exposure samples

Each dataset consists of 15360 images of three illumination classes. A total of 30720 images were generated in two datasets. The test sets and the training sets are divided roughly preserving the same ratio. The exact numbers are given in Table 3.4.

**Table 3.4.** Training set and test set splits of datasets

| Dataset | Training set | Percent | Test set | Percent | Total |
|---|---|---|---|---|---|
| mixedSet | 12600 | 82 | 2760 | 18 | 15360 |
| brightSet | 12288 | 80 | 3071 | 20 | 15360 |

### 3.2. CNN Models

In this thesis, popular image recognition models AlexNet, DenseNet, MobileNet, ResNet, VGGNet, and Xception were chosen to make comparisons. Additionally, some basic CNN implementations are introduced into the comparisons. The Keras library implementations are used to train the CNN models with the generated datasets. Since AlexNet implementation did not exist in the Keras library, a manual implementation is realized. Details of these models are presented in Table B1.

Besides the popular models, custom CNN models were trained with the general architecture given in Figure 3.5. The shared parameters of the custom models are given in Table 3.5. Model-specific parameters are given in Table 3.6.

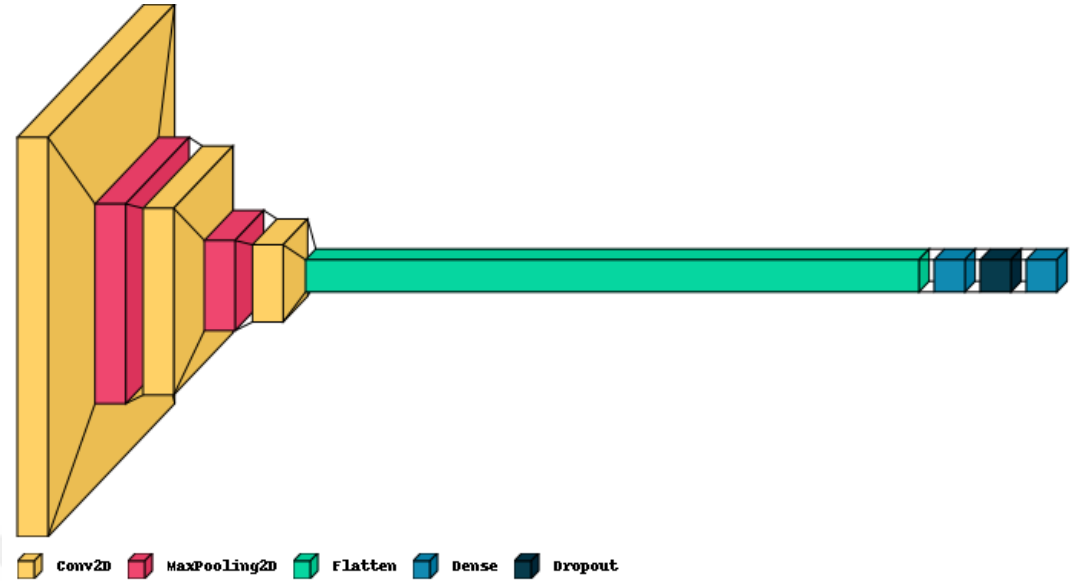



**Figure 3.5.** Custom CNN model visualization

**Table 3.5.** Shared parameters of CNN models

| Parameters | Custom CNN Models |
|---|---|
| Convolution Layers | 3 |
| Kernel Sizes | (3, 3), (3, 3), (3, 3) |
| Pool Sizes | 2x2 |
| Pool Strides | 2 |
| Dense Units | 64 |
| Activation Function | ReLU |
| Dropout Rate | 0.2 |
| Batch Size | 32 |
| Learning Rate | 0.001 |
| Epochs | 10 |
| Categories | 3 |

**Table 3.6.** Specific parameters of CNN models

| Model | Image Size | Kernel Sizes | Conv Filters |
|---|---|---|---|
| CNN64_333 | 64x64 | (3, 3), (3, 3), (3, 3) | 32, 32, 32 |
| CNN128_333 | 128x128 | (3, 3), (3, 3), (3, 3 | 32, 32, 32 |
| CNN256_333 | 256x256 | (3, 3), (3, 3), (3, 3) | 32, 32, 32 |
| CNN128_753 | 128x128 | (7, 7), (5, 5), (3, 3) | 32, 32, 32 |
| CNN128_531 | 128x128 | (5, 5), (3, 3), (1, 1) | 32, 32, 32 |
| CNN128_531_v2 | 128x128 | (5, 5), (3, 3), (1, 1) | 128, 64, 32 |
| CNN128_531_v3 | 128x128 | (5, 5), (3, 3), (1, 1) | 64, 32, 16 |

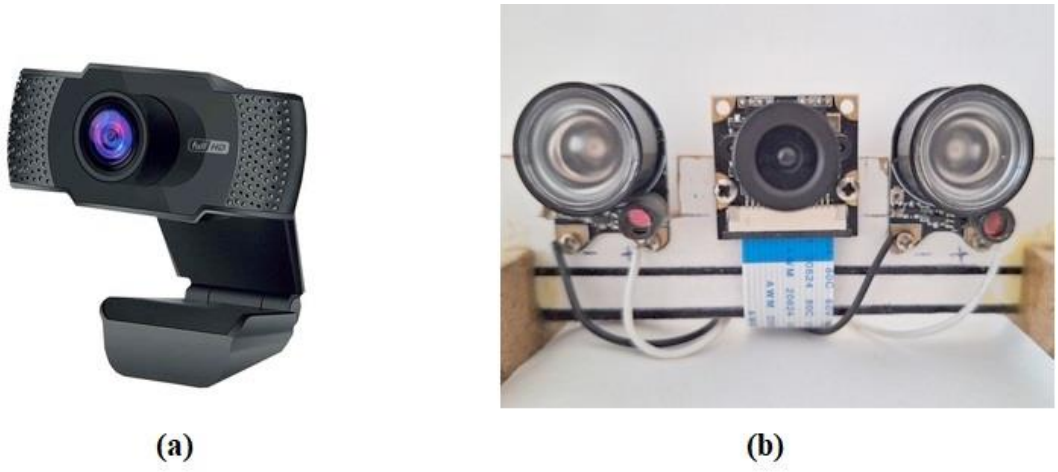
Training complex CNN models requires long computation times. Training times for all models are given in Table 3.7. for reference. Thus, initially, all models have been trained on *mixedSet* data for comparison. Then the best-performing models were trained on *brightSet* data in the second experiment. Performance metrics of the models were calculated using the confusion matrix results, such as accuracy, precision, recall, and F1 scores.

**Table 3.7.** *mixedSet* models training times

| Model | Training time | Model | Training time |
|---|---|---|---|
| AlexNet | 1h 5m 45s | CNN64_333 | 0h 2m 32s |
| DenseNet | 7h 39m 53s | CNN128_333 | 0h 9m 43s |
| MobileNetV2 | 2h 9m 47s | CNN256_333 | 0h 40m 41s |
| ResNet50 | 6h 13m 5s | CNN128_753 | 0h 16m 42s |
| ResNet50V2 | 5h 8m 52s | CNN128_531 | 0h 10m 3s |
| VGG16 | 12h 15m 18s | CNN128_531_v2 | 0h 36m 28s |
| Xception | 11h 29m 30s | CNN128_531_v3 | 0h 17m 12s |

### 3.3. Camera Setups

Two setups were prepared for measuring the real-time performances of the final models. A USB Full-HD Webcam mounted to the desktop computer has been used for testing the *mixedSet* models. Two headlight units were used along with the webcam to simulate flashes. For *brightSet* models, an IR camera and two IR flashes attached to a Raspberry 3B computer were prepared. IR LED distances to the camera center are fixed at 30mm. A helper program has been written to control the LEDs through two relays. Detailed setup images are given in Figure C.1. and Figure C.2. Webcam and IR camera images are represented in Figure 3.6. below. Since the used 1W IR LEDs were producing a max of 70 lux illumination, IR camera tests are made in a dark environment to simulate a powerful IR LED effect. Real-time camera performances of models are tested from the captured video sequences on a single subject. 1000 frames were captured for each illumination class, Left, Right, and Dual with both cameras i.e., 3000 images from the Webcam and 3000 images from the IR camera. These two Real-time datasets were used to measure the performance of selected CNN models. Sample images captured by Webcam and IR camera capture images are given in Figure E.1. and Figure E.2.



(a) (b)

**Figure 3.6. a)** Full-HD Webcam; **b)** IR Camera and IR LEDs

# 4. RESULTS AND DISCUSSION

Generating the dataset was a crucial part of this thesis. With the help of the unityStudio application, custom datasets were able to be produced fast and easily. The measured average processing speed was 0.725 models per second on the desktop computer with the given specifications. Thus, generating each dataset took roughly 1 hour.

## 4.1. CNN Results

Performance metrics of all CNN models were calculated from the confusion matrix generated from training time and from the test set separately. Used metrics are accuracy, precision, recall, and F1 score which are well-known metrics for measuring classification performances.

Accuracy measures the general correctness of the predictions made by the model. Since the datasets are well balanced this accuracy metric highly represents the performance of the models. Precision can be described as the ability to identify the positive samples correctly while avoiding false positives. The higher the precision, the fewer the false positives. Recall is a measure of identifying the positive samples throughout the samples that are known to be true already. The higher the recall, the fewer the false negatives. The F1 score is calculated as the harmonic mean of precision and recall and is useful when the training set is imbalanced. As the models are predicting three classes; precision, recall, and F1 scores are calculated for each class separately. To make the comparisons on the same table, arithmetic means of these metrics are used. Performance metrics of popular CNN models and custom CNN models are given in Table 4.1. and Table 4.2. respectively. All, values are calculated with a precision of the $8^{th}$ decimal but in tables, accuracies were rounded to the $4^{th}$ decimal, and the other metrics were rounded to $3^{rd}$ decimal precision to fit all values i.e., training data accuracy of the VGG16 model was 0.99999206 but rounded to 1.0000.

For the popular models, the DenseNet model was unsuccessful in predicting all three classes. The model predicted all Dual class members as in Left class or Right class. By being worse than the DenseNet model, the MobileNet model failed to predict both Left and Right classes and predicted all test samples as Dual classes.

On the other hand, the ResNet models and the VGGNet model performed outstanding performance on successfully classifying all three classes. AlexNet and Xception models also did a great classification performance.

**Table 4.1.** Performance metrics of popular CNN models

| Model | mixedSet Training Data | | | | mixedSet Test Data | | | |
|---|---|---|---|---|---|---|---|---|
| | Acc. | Prec. | Recall | F1 | Acc. | Prec. | Recall | F1 |
| AlexNet | 0.9250 | 0.931 | 0.926 | 0.925 | 0.9217 | 0.922 | 0.921 | 0.921 |
| DenseNet | 0.6421 | 0.761 | 0.652 | 0.517 | 0.6576 | 0.658 | 0.526 | 0.526 |
| MobileNet | 0.3389 | 0.113 | 0.333 | 0.169 | 0.3333 | 0.333 | 0.167 | 0.167 |
| ResNet50 | 0.9980 | 0.998 | 0.998 | 0.998 | 0.9982 | 0.998 | 0.998 | 0.998 |
| ResNet50V2 | 0.9986 | 0.988 | 0.986 | 0.987 | 0.9996 | 0.992 | 0.992 | 0.992 |
| VGG16 | 1.0000 | 0.999 | 1.000 | 1.000 | 0.9989 | 0.999 | 0.999 | 0.999 |
| Xception | 0.9750 | 0.998 | 0.975 | 0.976 | 0.9793 | 0.979 | 0.979 | 0.979 |

**Table 4.2.** Performance metrics of custom CNN models

| Model | mixedSet Training Data | | | | mixedSet Test Data | | | |
|---|---|---|---|---|---|---|---|---|
| | Acc. | Prec. | Recall | F1 | Acc. | Prec. | Recall | F1 |
| CNN64_333 | 0.9905 | 0.991 | 0.990 | 0.991 | 0.9942 | 0.994 | 0.994 | 0.994 |
| CNN128_333 | 0.9972 | 0.997 | 1.000 | 0.998 | 0.9975 | 0.998 | 0.997 | 0.997 |
| CNN256_333 | 0.9948 | 1.000 | 0.999 | 0.995 | 0.9953 | 0.995 | 0.995 | 0.995 |
| CNN128_753 | 0.9967 | 0.997 | 0.999 | 0.998 | 0.9634 | 0.963 | 0.963 | 0.963 |
| CNN128_531 | 0.9944 | 0.997 | 1.000 | 0.995 | 0.9975 | 0.997 | 0.997 | 0.997 |
| CNN128_531_v2 | 0.9926 | 0.995 | 0.993 | 0.997 | 0.9938 | 0.994 | 0.994 | 0.994 |
| CNN128_531_v3 | 0.9941 | 0.994 | 0.995 | 0.994 | 0.9938 | 0.998 | 0.994 | 0.994 |

As a matter of fact, simple custom CNN models have performed surprisingly well on this particular classification task by reaching scores close to most advanced models. This shows that even basic CNN models are powerful enough for classification tasks. Appendix D gives the complete results for training loss graphics, test data Confusion Matrix (CM), and Receiver Operator Curves (ROC) of all models.

ResNet50V2, VGG16, CNN128_333, and CNN128_531 were selected for training on *brightSet* data. Training times and performance metrics are given in Table 4.3. and Table 4.4. The best performer of the *mixedSet* model VGG16 gave a very poor performance on *brightSet*. On the other hand, simple CNN models were observed to have a very good performance again both on training and the test data.

**Table 4.3.** *brightSet* models training times

| Model | Training time | Model | Training time |
|---|---|---|---|
| ResNet50V2 | 5h 7m 28s | CNN128_333 | 0h 9m 57s |
| VGG16 | 11h 48m 1s | CNN128_531 | 0h 16m 11s |

**Table 4.4.** Performance metrics of CNN models on *brightSet*

| Model | brightSet training data | | | | brightSet test data | | | |
|---|---|---|---|---|---|---|---|---|
| | Acc. | Prec. | Recall | F1 | Acc. | Prec. | Recall | F1 |
| ResNet50V2 | 0.8653 | 0.903 | 0.867 | 0.868 | 0.8515 | 0.897 | 0.852 | 0.853 |
| VGG16 | 0.3275 | 0.109 | 0.333 | 0.164 | 0.3334 | 0.111 | 0.333 | 0.167 |
| CNN128_333 | 0.978 | 0.979 | 0.978 | 0.978 | 0.9782 | 0.979 | 0.978 | 0.978 |
| CNN128_531 | 0.9752 | 0.976 | 0.975 | 0.975 | 0.9792 | 0.979 | 0.979 | 0.979 |

### 4.2. Camera Experiment Results

The real-world performances of all models were drastically low. Because of failure in the training session, the VGG16 model was not included in the camera experiments. Table 4.5. shows the performances of models.

**Table 4.5.** Performance metrics of CNN models on camera captures

| Model | Webcam Capture | | | | IR Camera Capture | | | |
|---|---|---|---|---|---|---|---|---|
| | Acc. | Prec. | Recall | F1 | Acc. | Prec. | Recall | F1 |
| ResNet50V2 | 0.2487 | 0.148 | 0.248 | 0.175 | 0.3410 | 0.445 | 0.341 | 0.517 |
| CNN128_333 | 0.4537 | 0.533 | 0.454 | 0.403 | 0.3307 | 0.206 | 0.331 | 0.167 |
| CNN128_531 | 0.5623 | 0.691 | 0.562 | 0.518 | 0.3187 | 0.313 | 0.319 | 0.288 |

This time ResNet50V2 model made the worst predictions by labeling almost all samples in the test set as Left. The custom models were also having a hard time classifying the test set. Reducing the output classes to Left and Right might have better performance since it might be easier to separate in the absence of the Dual class. A final experiment was done by training the CNN128_531 model on both training sets to classify only the Left and the Right directions. The model trained with the mixedSet was labeled as mixedSet_CNN128_531_LR, and the model trained on brightSet was labeled as brightSet_CNN128_531_LR. The models were tested with all test sets once again. The results of the models are given in Table 4.6. and Table 4.7.

**Table 4.6.** Performance metrics of the brightSet_CNN128_531_LR model

| brightSet_CNN128_531_LR | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| brightSet test data | 0.9927 | 0.9927 | 0.9926 | 0.9931 |
| webcamCapture | 0.8055 | 0.8246 | 0.8055 | 0.8025 |
| streamCapture | 0.5435 | 0.5435 | 0.5442 | 0.5429 |

**Table 4.7.** Performance metrics of the mixedSet_CNN128_531_LR model

| mixedSet_CNN128_531_LR | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| mixedSet test data | 0.999 | 0.9989 | 0.9989 | 0.9989 |
| webcamCapture | 0.874 | 0.8982 | 0.8735 | 0.8714 |
| streamCapture | 0.522 | 0.5439 | 0.5215 | 0.45125 |

With this final experiment, models with acceptable accuracy were observed on one of the datasets webcamCapture. Yet, the IR webcam responses of the models were still low. ROC curves and confusion matrices related to the test phases of the models are given in Appendix F.

## 5. CONCLUSION

In this work, a synthetic dataset generation method was introduced, a synthetic dataset generator application was developed, and the generated datasets have been used which are not restricted only to the concept of this study, but usable in many other studies as well.

Also, some of the state-of-the-art CNN models and some basic CNN implementations were compared by means of their performance on this particular classification task: detecting the light source direction from the facial images. It has been observed that it was possible to detect the direction of the light source close to the camera from direct inputs. This basic distinction can be used to understand the 3D facial structure's existence in front of the camera to eliminate planar-type presentation attacks. Furthermore, the method is enhanced by the use of IR illumination and IR camera which are known to reveal extra features that can be additionally used for both facial recognition and liveness detection. The choice of IR illumination also has the advantage that it can be used in both day and night applications, and the use of powerful flashes does not cause discomfort as it is barely visible to human eyes.

It has been observed that it was not crucial to deploy complex CNN models for every scenario, but simpler CNN models were also powerful enough for classification. Simpler models are beneficial by means of training time and generated model sizes. For reference, all the custom models generated were less than 20 megabytes in comparison to 276 megabytes of Resnet models and 1500 megabytes of the VggNet model. Small model sizes are also beneficial for runtime speed and hardware requirements of the system.

Also, the synthetic data generated was not representing the real use cases well enough, even the best-scoring models presented a poor performance in the camera experiments. This may be because of the weak lighting setup of camera capture experiments that did not match the powerful lighting in training datasets. The creation of synthetic data should be fine-tuned to match the real-world use cases. But it is not easy to find IR-illuminated facial textures for generating synthetic data. Perhaps this might be another area of research interest.

As a final notice, even though models' scores on the camera-captured images were low, it has been observed that on the live demonstrations of cameras, models were performant in discriminating the left side illumination from the light side illumination. Especially with the models trained for two classes, Left and Right, the performance of the models was boosted.

A few ideas will be introduced as future study subjects within the concept of using illumination direction for spoof detection:

- Some image-filtering methods may be introduced to emphasize shade features i.e., a dual flash may be used to generate a base image, and right and left illuminated images may be subtracted from the base image to enhance shade features.
- The idea of narrow-band imagery and matching illumination can be carried to any desired range on the spectrum in theory. Furthermore, each direction can be illuminated by non-overlapping bands and both shading patterns can be extracted from a single image. This will surely require dedicated hardware.
- An idea as a countermeasure for video replay attacks is to use short sequences of random direction illuminations i.e., an 8-bit pattern: left - right - right - left - right - right - left - left which is hard to reproduce by the attackers but can be easily decoded by the system.
- Similarly, lighting duration and luminosity differences may be introduced as further enhancements.

## 6. REFERENCES

Adamiak, K., Żurek, D., & Ślot, K. (2015). Liveness detection in remote biometrics based on gaze direction estimation. *Proceedings Of The 2015 Federated Conference On Computer Science And Information Systems*. doi: 10.15439/2015f307

Akhtar, Z., Micheloni, C., & Foresti, G. (2015). Biometric Liveness Detection: Challenges and Research Opportunities. *IEEE Security & Privacy*, *13*(5), 63-72. doi: 10.1109/msp.2015.116

Atik, M. A. O. (2023). unityStudio: Unity-based photo studio simulation with dual side flash for custom dataset generation. https://github.com/aliosmanatik/unityStudio

Cao, C., Weng, Y., Zhou, S., Tong, Y., & Zhou, K. (2013). Facewarehouse: A 3d facial expression database for visual computing. IEEE Transactions on Visualization and Computer Graphics, 20(3), 413-425.

Chan, P., Liu, W., Chen, D., Yeung, D., Zhang, F., Wang, X., & Hsu, C. (2018). Face Liveness Detection Using a Flash Against 2D Spoofing Attack. *IEEE Transactions On Information Forensics And Security*, *13*(2), 521-534. doi: 10.1109/tifs.2017.2758748

Chen, Y., & Zhang, W. (2018). Iris Liveness Detection: A Survey. *2018 IEEE Fourth International Conference On Multimedia Big Data (Bigmm)*. doi: 10.1109/bigmm.2018.8499061

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).

Dahia, G., Jesus, L., & Pamplona Segundo, M. (2020). Continuous authentication using biometrics: An advanced review. *Wires Data Mining And Knowledge Discovery*, *10*(4). doi: 10.1002/widm.1365

Deng, Y., Yang, J., Xu, S., Chen, D., Jia, Y., & Tong, X. (2019). Accurate 3D face reconstruction with weakly-supervised learning: From single image to image set.

2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). doi:10.1109/cvprw.2019.00038

Di Martino, J., Qiu, Q., & Sapiro, G. (2021). Rethinking Shape From Shading for Spoofing Detection. *IEEE Transactions On Image Processing*, *30*, 1086-1099. doi: 10.1109/tip.2020.3042082

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

Imageware Systems (2020). https://www.iwsinc.com/products/biointellic-intelligent-antispoofing-system/

Kakadiaris, I., Passalis, G., Theoharis, T., Toderici, G., Konstantinidis, I., & Murtuza, N. (2005). Multimodal Face Recognition: Combination of Geometry with Physiological Information. *2005 IEEE Computer Society Conference On Computer Vision And Pattern Recognition (CVPR'05)*. doi: 10.1109/cvpr.2005.241

Kortli, Y., Jridi, M., Falou, A. A., & Atri, M. (2020). Face Recognition Systems: A Survey. *Sensors (Basel, Switzerland)*, *20*(2). https://doi.org/10.3390/s20020342

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional Neural Networks. Communications of the ACM, 60(6), 84–90. doi:10.1145/3065386

Li, T., Bolkart, T., Black, M. J., Li, H., & Romero, J. (2017). Learning a model of facial shape and expression from 4D scans. ACM Transactions on Graphics, 36(6), 1–17. doi:10.1145/3130800.3130813

Maggiordomo, A., Cignoni, P., & Tarini, M. (2021). Texture defragmentation for photo-reconstructed 3D models. Computer Graphics Forum, 40(2), 65–78. doi:10.1111/cgf.142615

Muntoni , A., & Cignoni, P. (2021, January). pymeshlab. Retrieved from https://doi.org/10.5281/zenodo.7852056

Ramachandra, R., & Busch, C. (2017). Presentation Attack Detection Methods for Face Recognition Systems. *ACM Computing Surveys*, *50*(1), 1-37. doi: 10.1145/3038924

Parveen, S., Ahmad, S., Hanafi, M., & Adnan, W. (2015). Face anti-spoofing methods. *CURRENT SCIENCE*, *108*(8), 1491 - 1500.

Paysan, P., Knothe, R., Amberg, B., Romdhani, S., & Vetter, T. (2009). A 3D face model for pose and illumination invariant face recognition. 2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance. doi:10.1109/avss.2009.58

Samatzidis, T., Siegmund, D., Goedde, M., Damer, N., Braun, A., & Kuijper, A. (2018). The Dark Side of the Face: Exploring the Ultraviolet Spectrum for Face Biometrics. *2018 International Conference On Biometrics (ICB)*. doi: 10.1109/icb2018.2018.00036

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Tsitiridis, A., Conde, C., Gomez Ayllon, B., & Cabello, E. (2019). Bio-Inspired Presentation Attack Detection for Face Biometrics. *Frontiers In Computational Neuroscience*, *13*. doi: 10.3389/fncom.2019.00034

Wang, T., Yang, J., Lei, Z., Liao, S., & Li, S. (2013). Face liveness detection using 3D structure recovered from a single camera. *2013 International Conference On Biometrics (ICB)*. doi: 10.1109/icb.2013.6612957

Xin, Y., Liu, Y., Liu, Z., Zhu, X., Kong, L., & Wei, D. et al. (2017). A survey of liveness detection methods for face biometric systems. *Sensor Review*, *37*(3), 346- 356. doi: 10.1108/sr-08-2015-0136

Zhang, X., Hu, X., Ma, M., Chen, C., & Peng, S. (2016). Face spoofing detection based on 3D lighting environment analysis of image pair. 2016 23Rd International Conference On Pattern Recognition (ICPR). doi: 10.1109/icpr.2016.7900093

## APPENDIX A

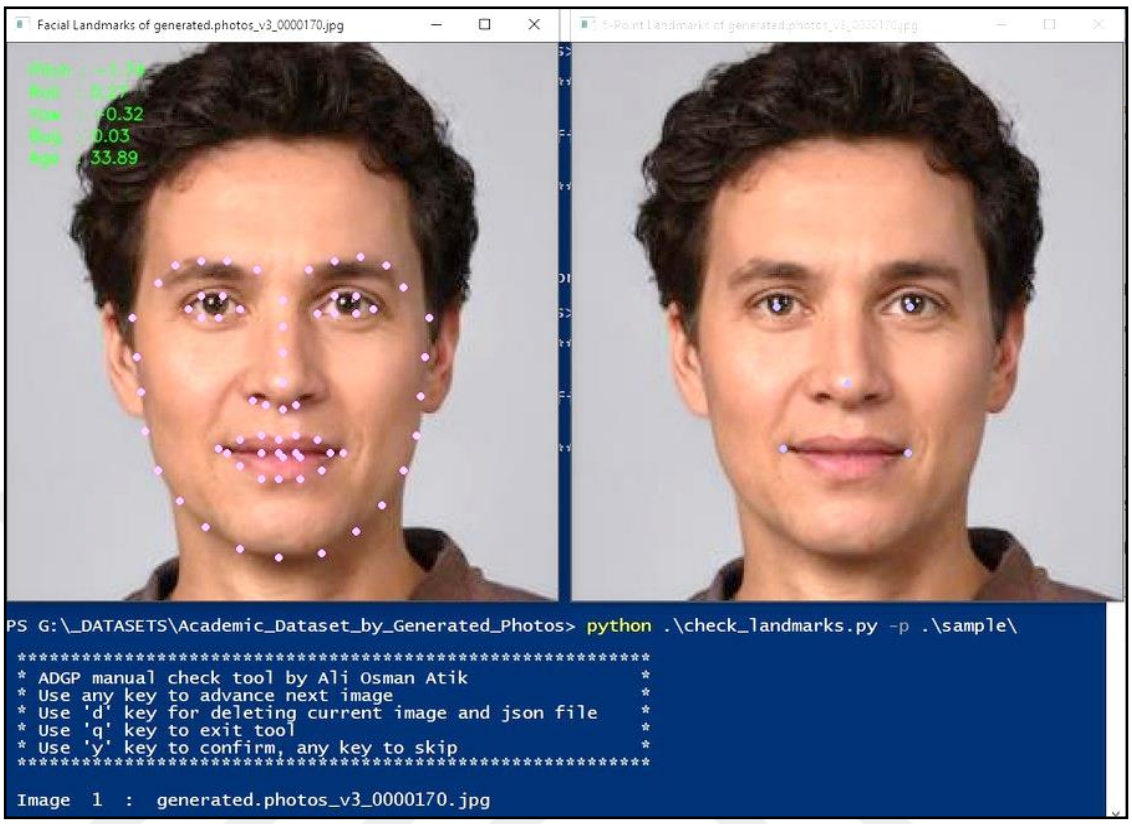



**Figure A.1.** Screen capture - *check_landmarks.py*

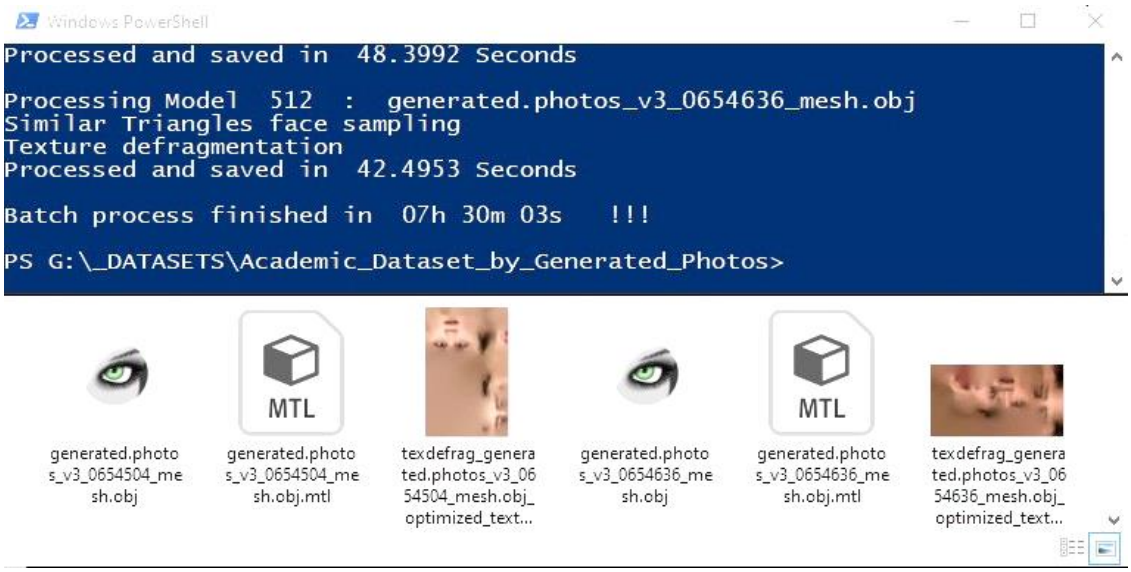



**Figure A.2.** Screen capture - *meshlab_batch_process.py*


**Figure A.3.** Screen captures - *unityStudio*

**Figure A.4.** Generated faces placed on a custom background

## APPENDIX B

**Table B.1.** Details of selected CNN models

| Models | Image Input Size | Conv. Layers | Kernel Sizes | Convolution Filters |
|---|---|---|---|---|
| AlexNet | 227x227 | 5 | (11, 11), (5, 5), (3, 3) | 96, 256, 384, 384, 256 |
| DenseNet | 227x227 | 120 | (7, 7), (3, 3), (1, 1) | 64, (128, 32)x6, 128, (128, 32)x12, 256, (128, 32)x24, 512, (128, 32)x16 |
| MobileNetV2 | 224x224 | 52 | (3x3) | 32, 32, 16, 96, 96, 24 (144, 144, 24), (144. 144, 32), (192, 192, 32)x2, (192, 192, 64), (384, 384, 64)x3, (384, 384, 96), (576, 576, 96)x2, (576, 576, 160), (960, 960, 160)x2, (960, 960, 320, 1280) |
| ResNet50 | 224x224 | 53 | (7, 7), (3, 3), (1, 1) | 64, 64, 64, 256,256, (64, 64, 256)x2, 128, 128, 512, 512, (128, 128, 512)x3, 256, 256, 1024, 1024, (256, 256, 1024)x5, 512, 512, 2048, 2048, (512, 512, 2048)x2 |
| ResNet50V2 | 224x224 | 53 | (7, 7), (3, 3), (1, 1) | 64, 64, 64, 256,256, (64, 64, 256)x2, 128, 128, 512, 512, (128, 128, 512)x3, 256, 256, 1024, 1024, (256, 256, 1024)x5, 512, 512, 2048, 2048, (512, 512, 2048)x2 |
| VGG16 | 224x224 | 13 | (3x3) | 64, 64, 128, 128, 256, 256, 256, 512, 512, 512, 512, 512, 512 |
| Xception | 299x299 | 6 | (3, 3), (1, 1) | 32, 64, 128, 256, 728, 1024 |

## APPENDIX C

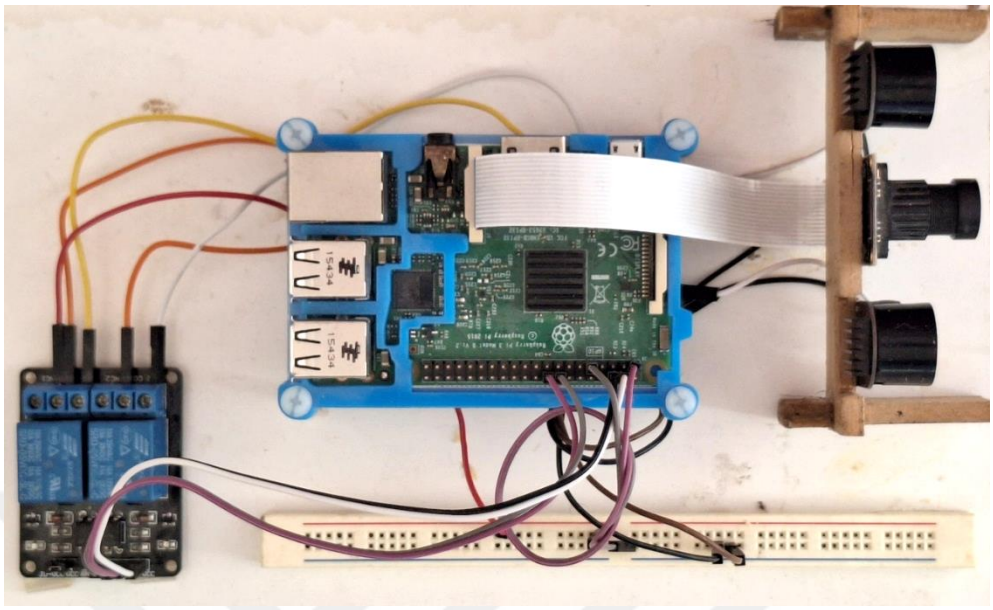


**Figure C.1**. IR camera setup on Raspberry Pi 3B

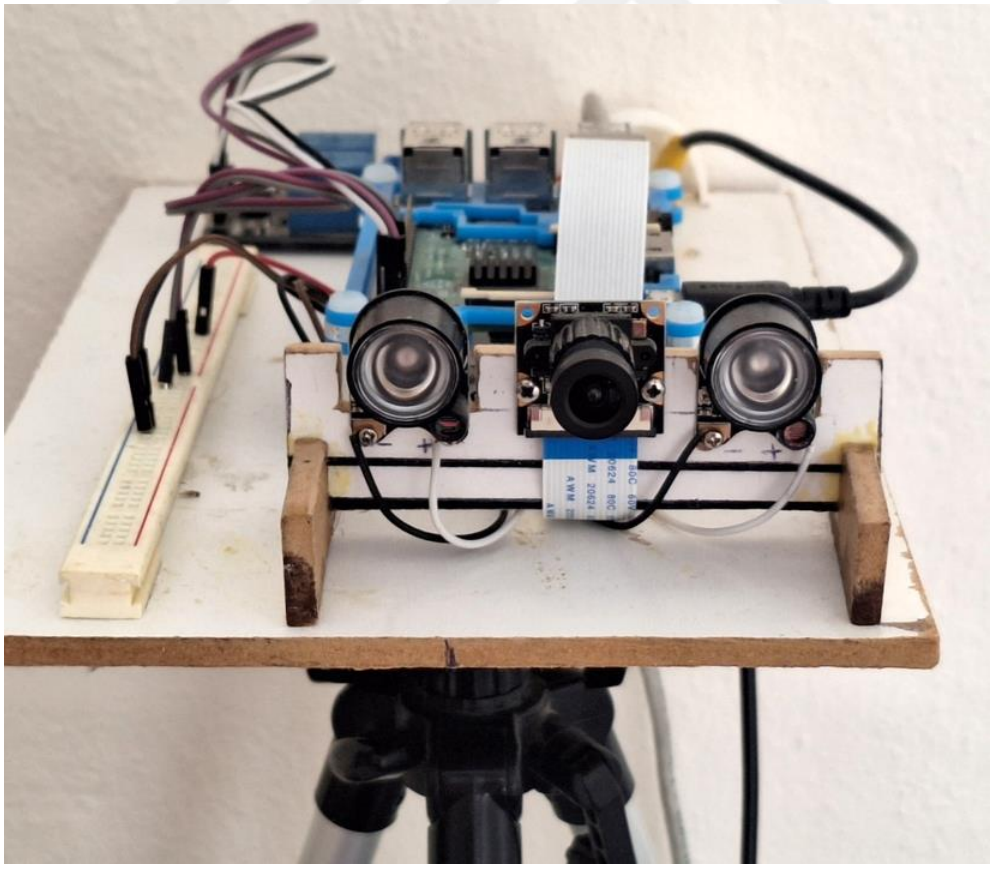


**Figure C.2**. IR camera setup on a tripod

## APPENDIX D

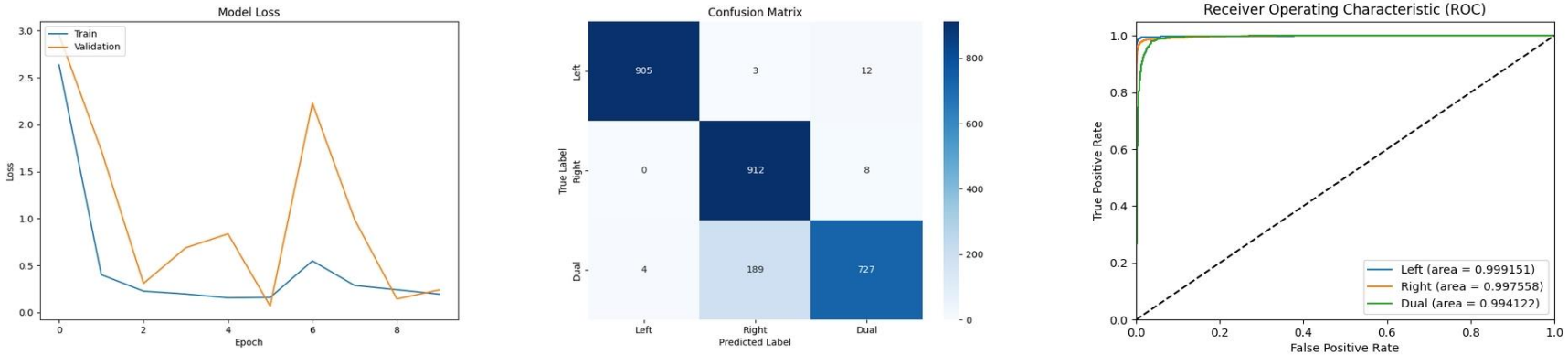



**Figure D.1.** Training loss, confusion matrix, and ROC curve for AlexNet



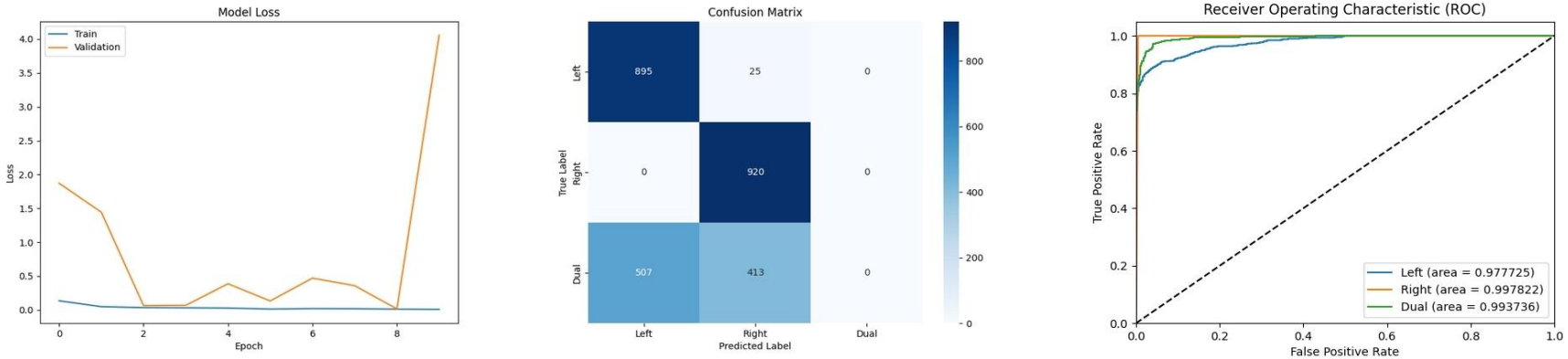



**Figure D.2.** Training loss, confusion matrix, and ROC curve for DenseNet

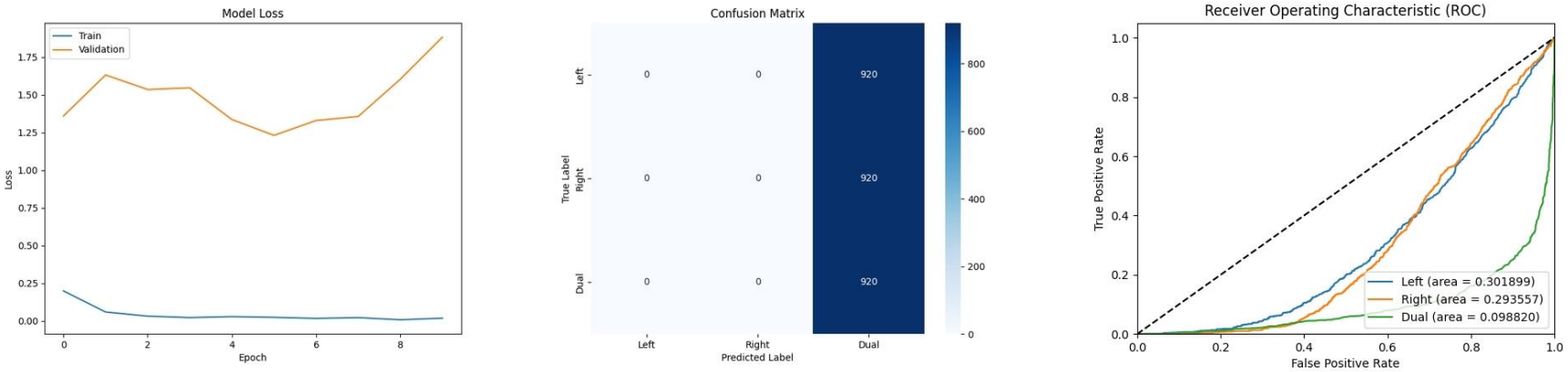



**Figure 5.3.** Training loss, confusion matrix, and ROC curve for MobileNet V2



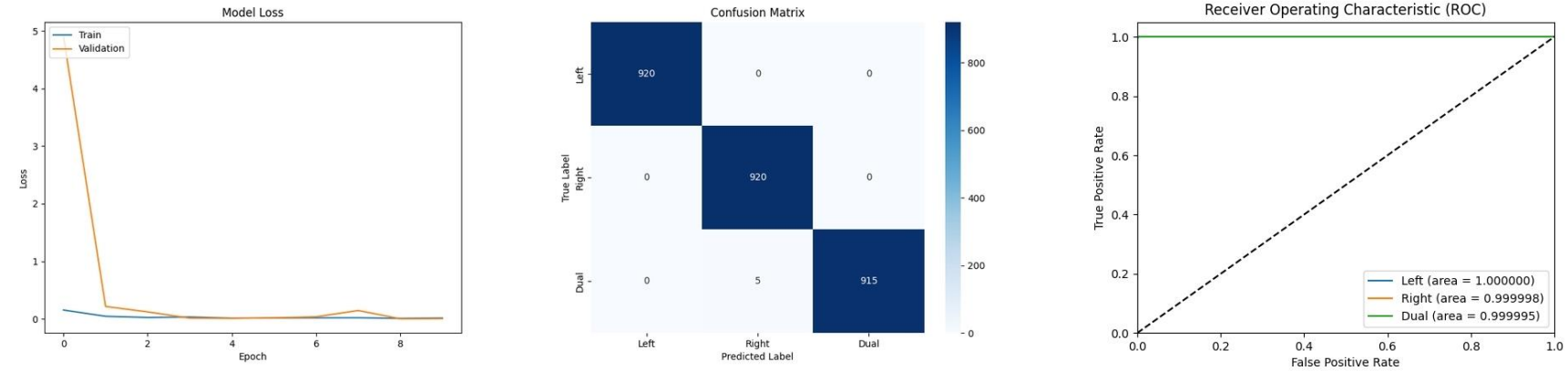



**Figure 5.4.** Training loss, confusion matrix, and ROC curve for ResNet 50

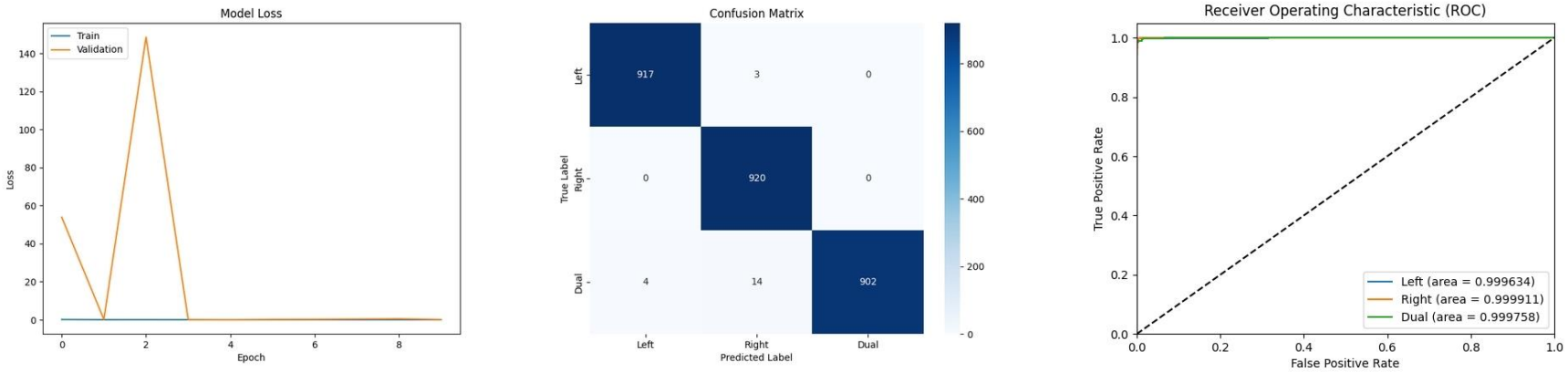



**Figure 5.5.** Training loss, confusion matrix, and ROC curve for ResNet 50 V2



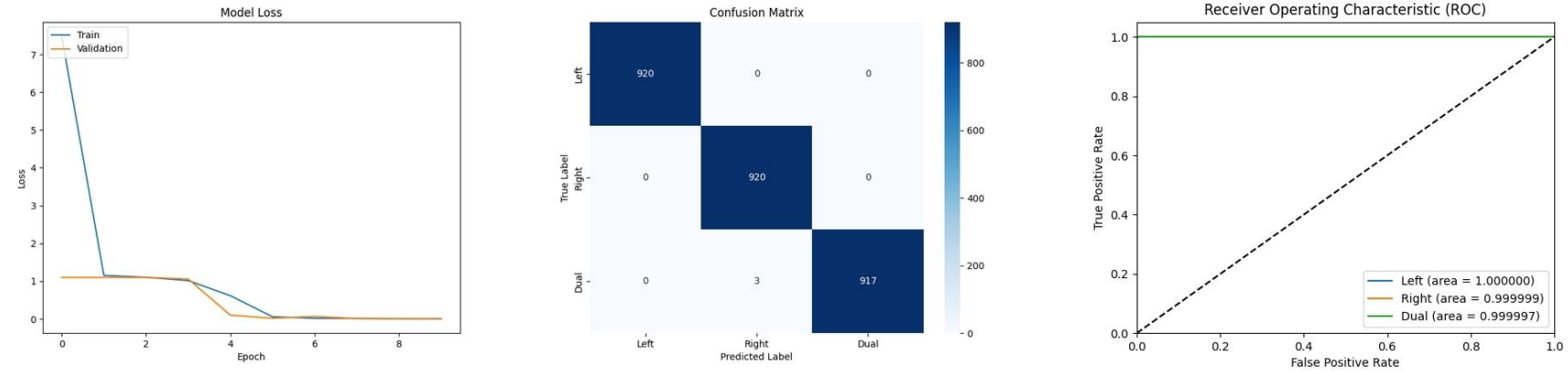



**Figure 5.6.** Training loss, confusion matrix, and ROC curve for VGG 16

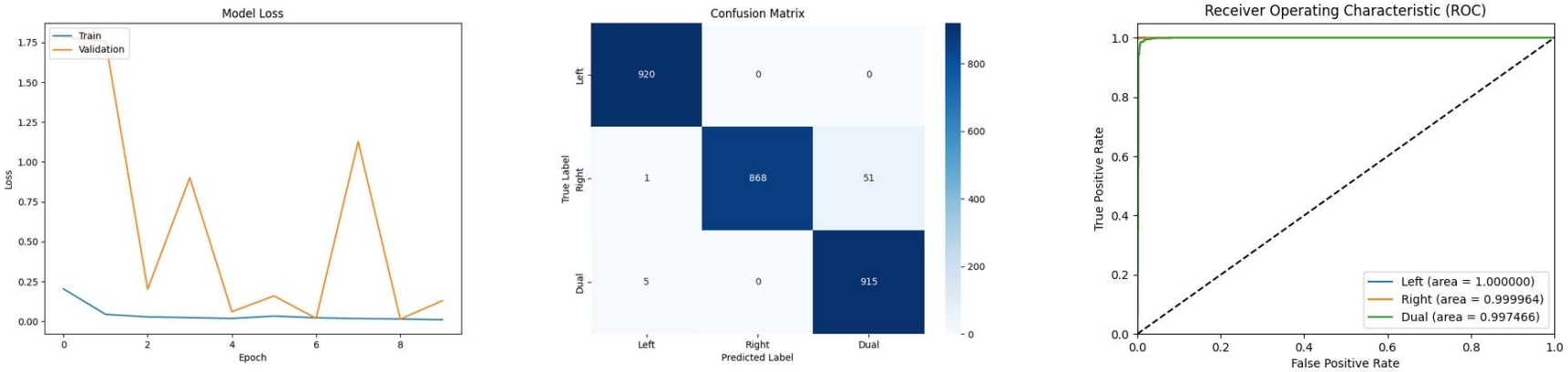



**Figure 5.7.** Training loss, confusion matrix, and ROC curve for



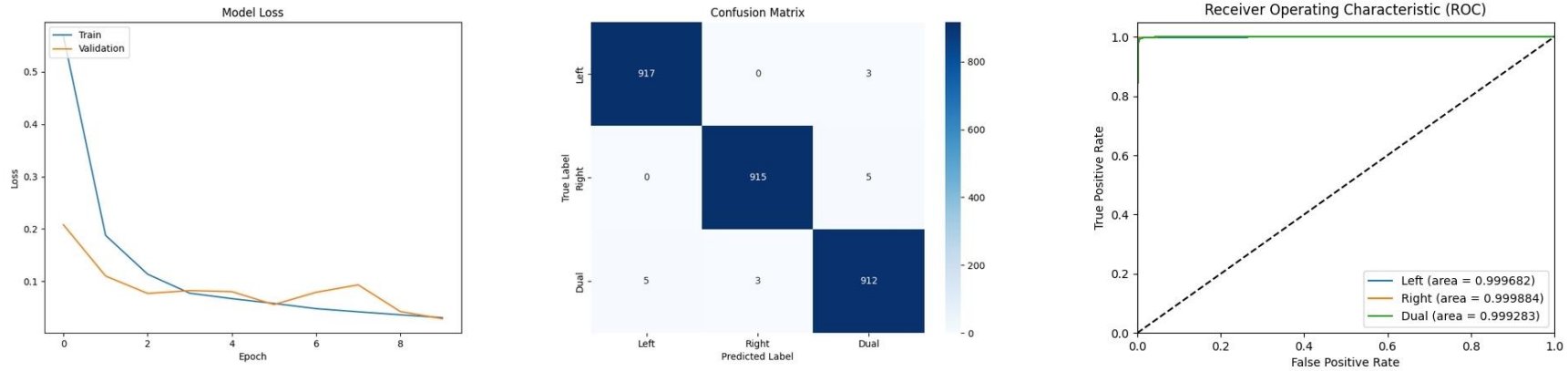



**Figure 5.8.** Training loss, confusion matrix, and ROC curve for CNN_64_333

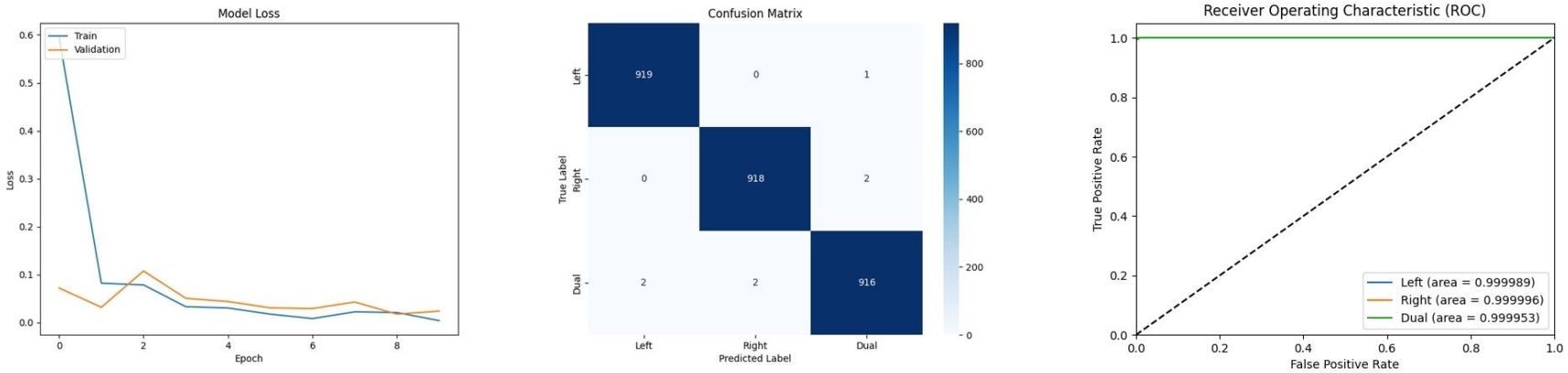



**Figure 5.9.** Training loss, confusion matrix, and ROC curve for CNN_128_333



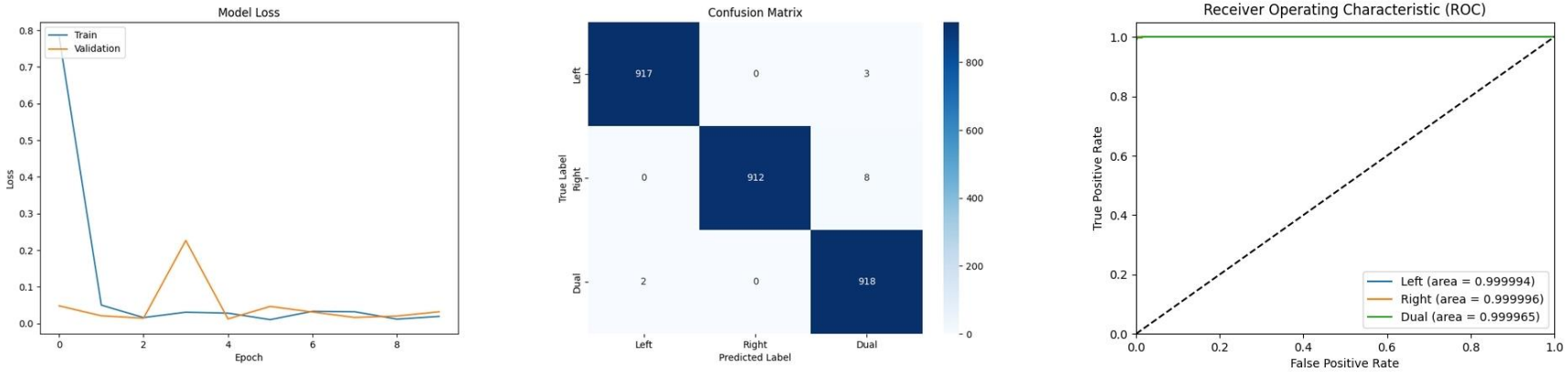



**Figure 5.10.** Training loss, confusion matrix, and ROC curve for CNN_256_333

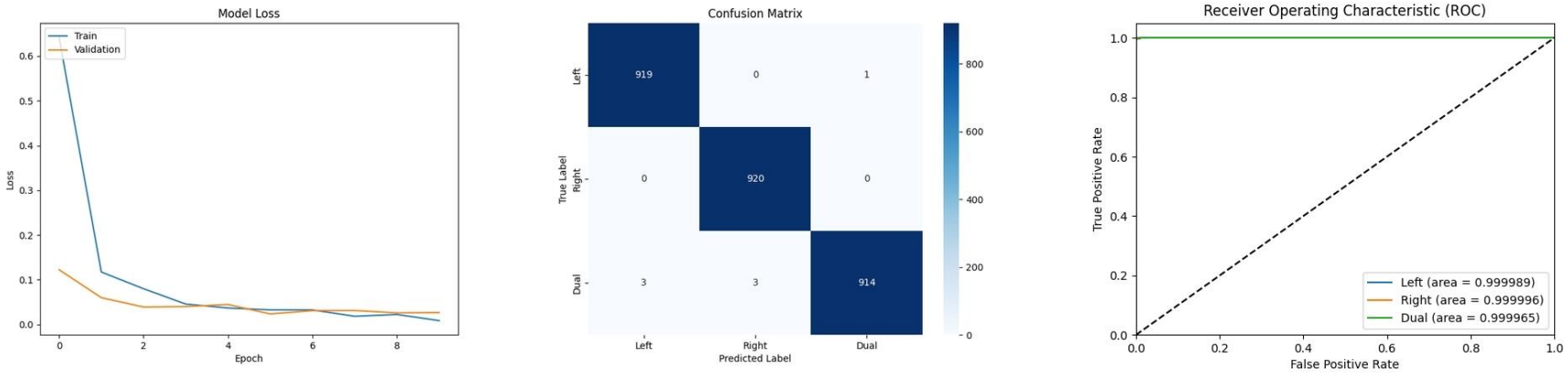



**Figure 5.11**. Training loss, confusion matrix, and ROC curve for CNN_128_531



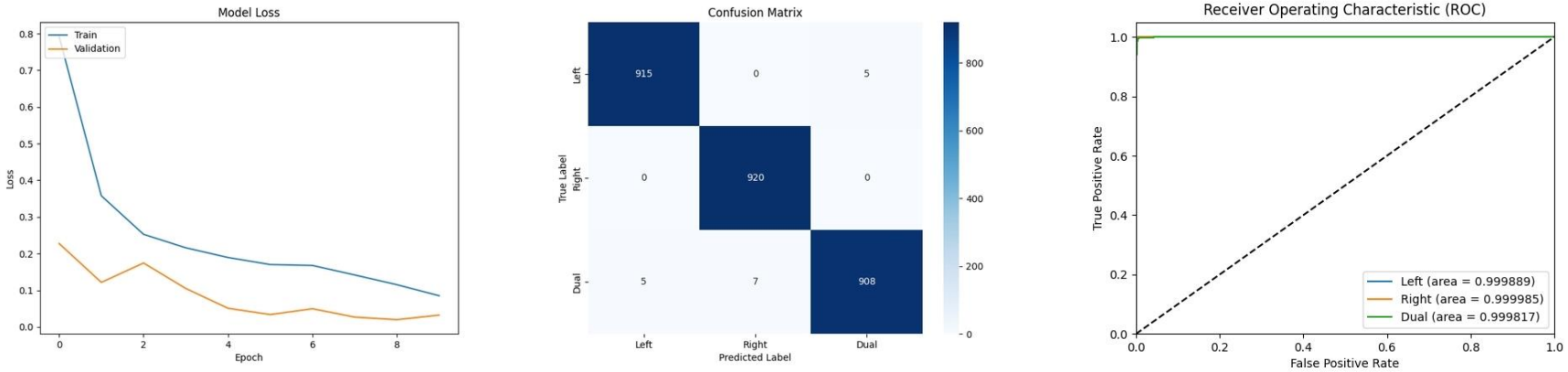



**Figure 5.12.** Training loss, confusion matrix, and ROC curve for CNN_128_531_V2

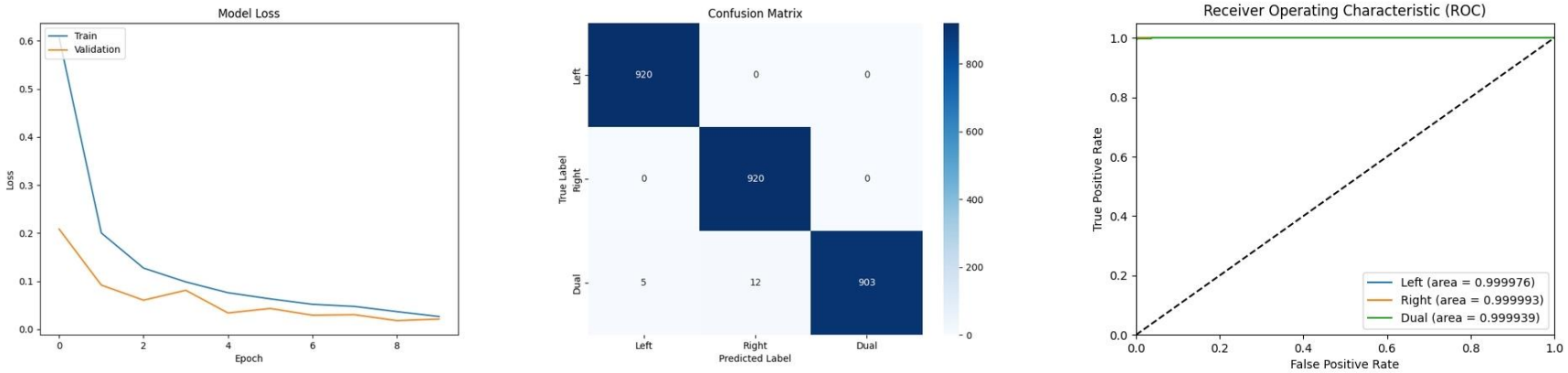



**Figure 5.13.** Training loss, confusion matrix, and ROC curve for CNN_128_531_V3



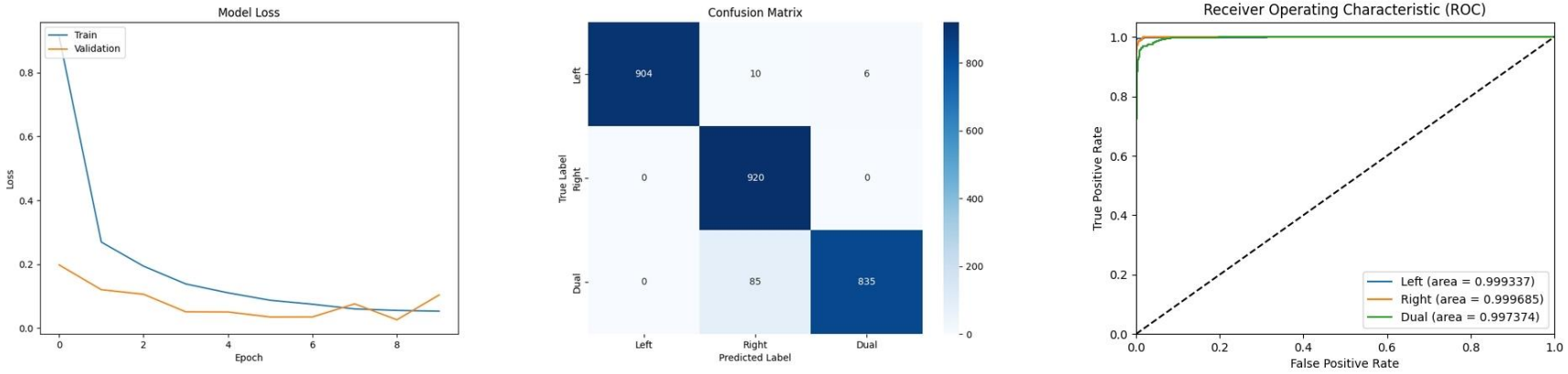



**Figure 5.14.** Training loss, confusion matrix, and ROC curve for CNN_128_753

**APPENDIX E**

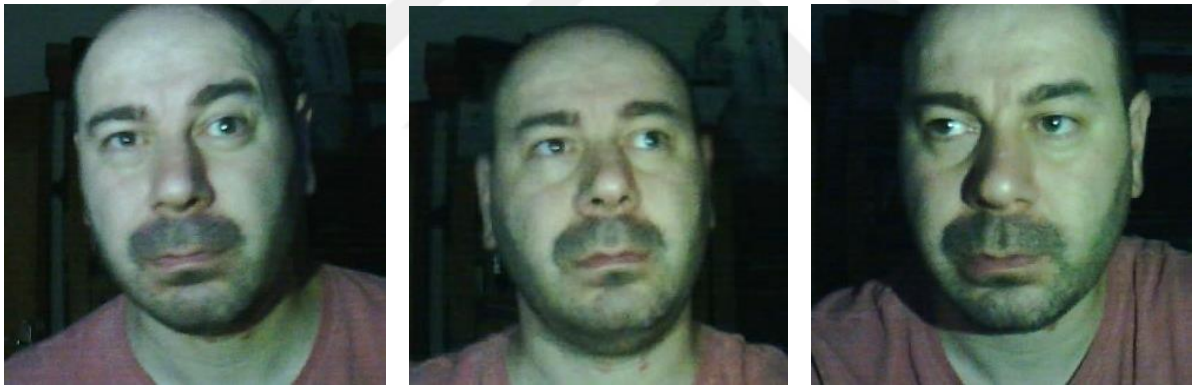


**Figure E.1.** Webcam capture samples of Dual, Left, and Right illuminated face



**Figure E.2.** IR camera capture samples of Dual, Left, and Right illuminated face

## APPENDIX F

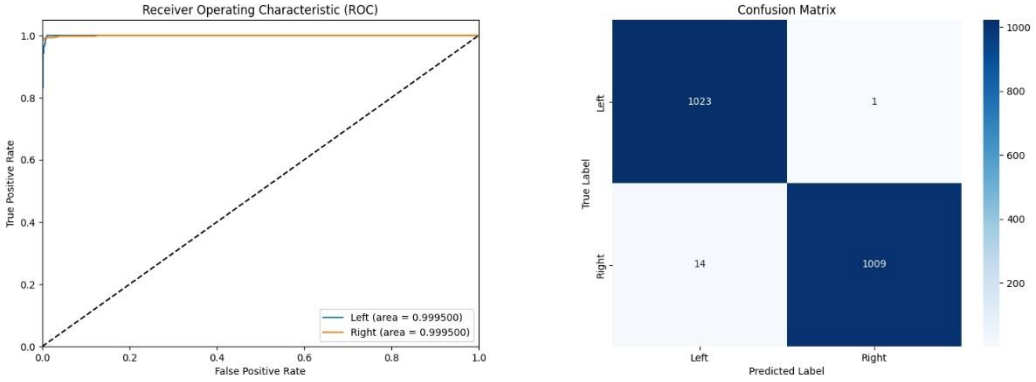



**Figure F.1.** ROC and CM of brightSet_CNN128_531_LR on the test set

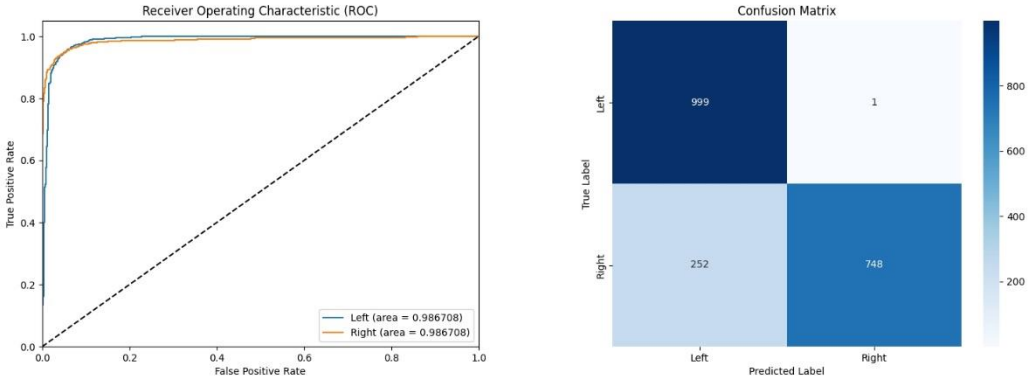



**Figure F.2.** ROC and CM of brightSet_CNN128_531_LR on the webcam set

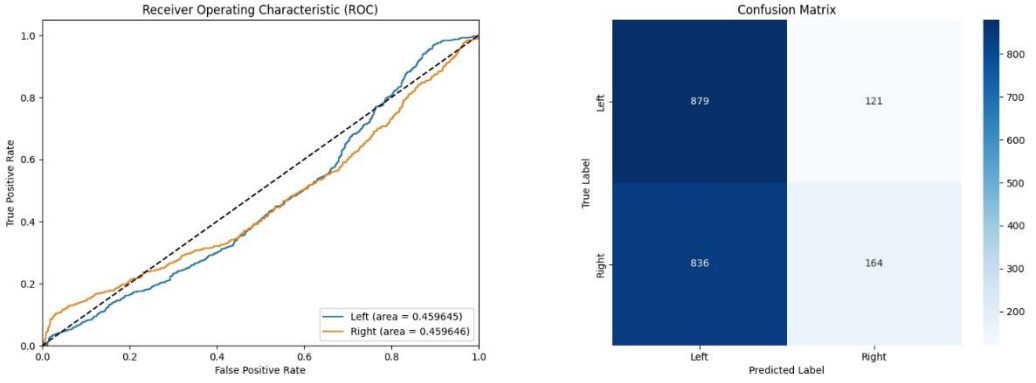



**Figure F.3.** ROC and CM of brightSet_CNN128_531_LR on the IR camera set

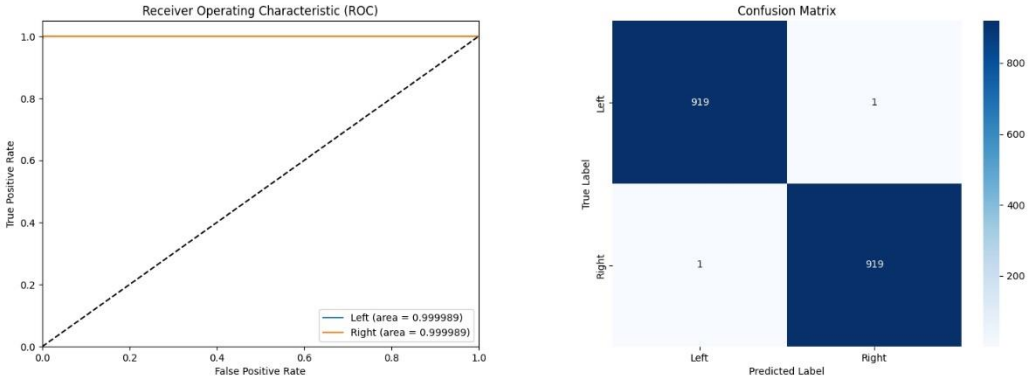



**Figure F.4.** ROC and CM of mixedSet_CNN128_531_LR on the test set

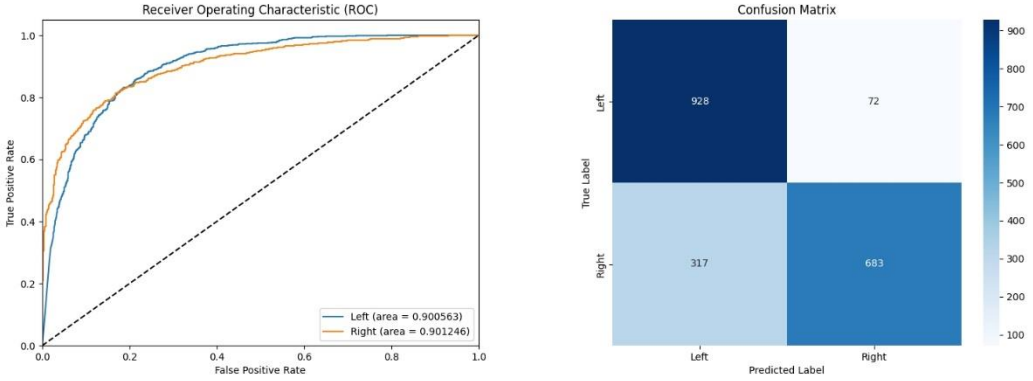



**Figure F.5.** ROC and CM of mixedSet_CNN128_531_LR on the webcam set

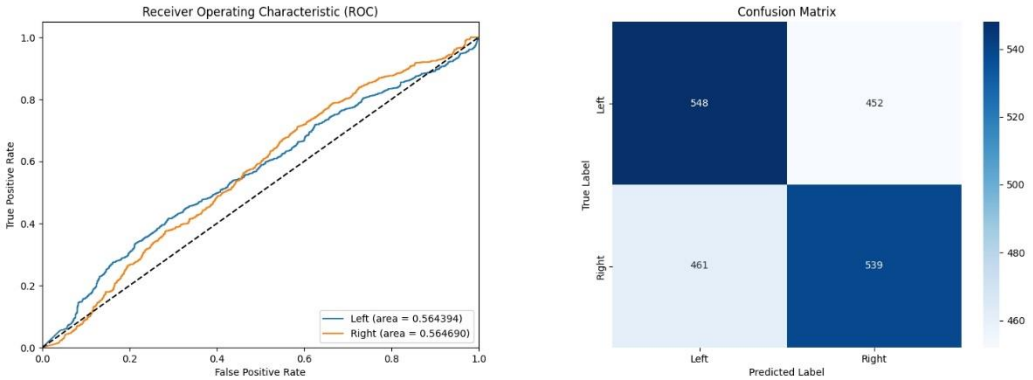



**Figure F.6.** ROC and CM of mixedSet_CNN128_531_LR on the IR camera set

# CURRICULUM VITAE

**Mehmet Ali Osman Atik**

## EDUCATION

| | |
|---|---|
| Master of Science<br>2020 - 2023 | Akdeniz University<br>Institute of Natural and Applied Sciences,<br>Computer Engineering, Antalya, Turkey |
| Bachelor of Science<br>2014 - 2019 | İstanbul Technical University<br>Faculty of Computer and Informatics Engineering,<br>Computer Engineering, İstanbul, Turkey |

## WORK EXPERIENCE

| | |
|---|---|
| | |

## PUBLICATIONS