Classification Technique for Face-Spoof Detection in Artificial Neural Networks using Concepts of machine Learning

by

Anika Anjum Una
17201139
Erina Haque
16141012
Nishat Sultana Ritu
17101203
Zarin Tasnim Haque
17101205
Rifat Shahran Opal
16201030

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

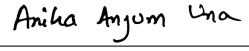
Department of Computer Science and Engineering
Brac University
June 2021

Declaration

It is hereby declared that

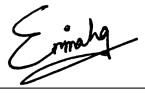
1. The thesis submitted is our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



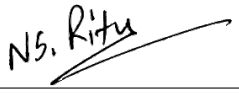
Anika Anjum Una

17201139



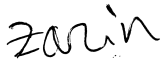
Erina Haque

16141012



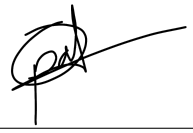
Nishat Sultana Ritu

17101203



Zarin Tasnim Haque

17101205



Rifat Shahran Opal

16201030

Approval

The thesis titled "Classification Technique for Face-Spoof Detection in Artificial Neural Networks using Concepts of Machine Learning" submitted by

1. Anika Anjum Una - 17201139
2. Erina Haque - 16141012
3. Nishat Sultana Ritu- 17101203
4. Zarin Tasnim Haque- 17101205
5. Rifat Shahran Opal- 16201030

Of spring, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on June 10 2021.

Examining Committee:

Moin Mostakim
Lecturer
Department Of Computer Science And Engineering
Brac University

Dr. Md. Golam Rabiul Alam
Associate Professor
Department Of Computer Science And Engineering
Brac University

Dr. Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

The thesis is carried out in complete compliance with research ethics, policies, regulations and codes set by BRAC University. We have used various information from different sources in order to pursue the research. To collect data, we read articles, journals from different websites, etc. The sources we have used here are interpreted in our own terms and are properly mentioned as a reference. We appreciate and give credit to every source that helped us to continue our work. Lastly, we declare that five authors of this paper hold liability if any violation of BRAC University standard is found.

Abstract

In biometric technology, face recognition techniques are considered the most significant research area. This technology is abundantly used in security services, smart cards, surveillance, social media, and ID verification. The number of countermeasures is gradually increasing, and many systems have been initiated to distinguish genuine access and fake attacks. In our paper, we propose a Convolutional Neural Network (CNN), which can obtain fine distinctions and abilities in a supervised manner. Deep convolutional neural networks have prompted a progression of breakthroughs for image classification. This paper introduces various architectures of CNN for detecting face spoofing using many convolutional layers. We have used VGG-16 under Convolutional Neural Networks (CNN) architecture in the proposed system for learning about the feature classification. Our proposed system has showcased an accuracy of 98% for Convolutional Neural Network (CNN), 63% for VGG16, and 50% for Support Vector Machine (SVM) respectively.

Keywords: Face Spoofing Techniques, Feature Classification, Machine Learning, Convolutional Neural Network (CNN), VGG-16.

Dedication

We would want to dedicate our thesis to our parents, who have helped us get this far by every means possible. Then, to our respected supervisor, Moin Mostakim sir; we could not have conducted our research without his instructions and guidelines. Lastly, with condolence, to all the people who had and still are suffering from loss of data stealing and identity thieving issues.

Acknowledgement

In the name of Allah, Most Gracious, Most Merciful, Who has given us the strength and perseverance, we are grateful to our family members, as well as our supervisor, Moin Mostakim sir, for his consistent guidance and aid in envisioning that this research is possible. He helped us whenever we needed any sort of assistance and guided us to improve ourselves.

For our parents' kind support and prayers, we are now on the verge of completing the final phase of our thesis. We also appreciate our fellow team members who held strong and united till the end to successfully complete the work.

Finally, we want to thank our institution and its administration for providing us with a platform from which we may go one step closer to achieving our main objectives.

Table of Contents

List of Figures

Chapter 1

Introduction

It is realized that the vast majority of the current face recognition frameworks are defenseless against spoofing attacks. A spoofing assault occurs when some- body attempts to pass a biometric face system by presenting a fake face in front of the camera. Face-spoofing location normally fills in as a preprocessing step of the face acknowledgment frameworks to decide whether the face picture is obtained from a genuine individual or a printed photograph (replay video). Along these lines, face-spoofing recognition is a twofold grouping issue. It is likewise called face-liveness discovery. The spoofing assault comprises the utilization of manufactured biometric attributes to acquire ill-conceived admittance to make sure about assets secured by a biometric verification framework. It is an immediate assault on the tangible contribution of a biometric framework, and the assailant doesn't require past information about the acknowledgment calculation. With the brilliant addition of video and picture databases, there is an unbelievable need for programmed comprehension and evaluation of data from the savvy framework as physically it is finding the opportunity to be surely distant.[1] Thus, the programmed face discovery framework assumes a significant job in face acknowledgment, outward appearance acknowledgment, head-present assessment, human-PC connection, and so forth.

1.1 Motivation

Face anti-spoofing is drawing increasing attention in academic and industrial fields as a security measure for face recognition systems. However, due to the diversity of Face spoofing types, including print-attacks, replay-attacks, mask attacks, Etc., it is still difficult to distinguish various fake faces. A spoofing assault happens when somebody attempts to pass a biometric face system by presenting a fake face in front of the camera. Face spoofing typically fills in as a preprocessing step of the face acknowledgment frameworks to decide whether the face picture is obtained from a genuine individual or a printed photograph (replay video). Along these lines, face spoofing recognition is a twofold grouping issue. It is likewise called face liveness discovery. The spoofing assault comprises manufactured biometric attributes to acquire ill-conceived admittance to make sure about assets secured by a biometric verification framework. It is an immediate assault on the tangible contribution of a biometric framework, and the assailant does not require past information about the acknowledgment calculation. With the brilliant addition of video and picture databases, there is an unbelievable need for programmed comprehension and evalu-

ation of data from the savvy framework as physically it is finding the opportunity to be undoubtedly distant. This way, the programmed face discovery framework assumes a significant job in face acknowledgment, outward appearance acknowledgment, head-present assessment, human-PC connection, and so forth. In this paper, we utilize a deep convolutional neural network to work against face spoofing. If we compare other manually built features like LBP, HOG, LBP-TOP, DoG, then we will find that features from CNN are more discriminative in nature. The popularity of face recognition systems also resulted in popularity in face spoofing attacks. Face spoofing detection is now mandatory in every face recognition system since technologies to clone human faces have gotten very advanced.

1.2 Objectives

In research, most contemporary facial recognition systems are known to be vulnerable to spoofing attacks. Face anti-spoofing is a safety feature for facial recognition systems that are gaining popularity in academia and industry. Face spoofing is often used as a preprocessing step in face recognition frameworks to determine if the face image is from a real person or a printed photograph. As a result, detecting face spoofing is a binary classification challenge.[2] Face liveness detection is another name for it, and it was created to combat various sorts of spoofing assaults. As a result, the spoofing attack uses falsified biometric features to obtain unauthorized access to resources protected by biometric authentication. It is a direct attack on the concrete contribution of a biometric framework, and the adversary does not need to know the recognition algorithm beforehand. With the exponential growth of video and picture data, there is an unimaginable need to automatically understand and assess information from intelligent systems, as doing so manually is becoming increasingly difficult.

The face is important in social interactions because it conveys an individual's identity and feelings. In comparison to robots, humans have a fantastic capacity to discriminate between distinct faces. As a result, the automatic face detection system is critical for face recognition, facial expression recognition, head-pose estimation, and human-computer interaction, among other things.[3] However, in this article, we will detail how to stave against face impersonation by the use of a deep convolutional neural network (CNN). We are tasked with accomplishing this as our first mission. Because CNN-learned features, rather than hand-crafted features, may better identify discriminatory characteristics in data-driven ways, this example might use CNN-learned features rather than hand-crafted features.[4]

The trials have proven more essential since they show that the more general qualities it acquires, the more versatile the tool may be in generating spoofing in many formats. Furthermore, we used the feature color and texture. Because the features indicated earlier might be regarded as changes in the face texture information or image quality, the facial appearance analysis-based approaches are sometimes referred to as texture or image quality analysis-based approaches. To conclude, the

researchers studied the use of pre-trained convolutional neural network techniques (CNN) for FR and classification accuracy by comparing the FR performance of the CNN with that of transference learning for extraction and classification using pre-trained CNN.

1.3 Methodology

Our first step was to collect the Real and Fake Face Detection dataset. We use TensorFlow and Keras; both Tensorflow and Keras is an open-source software library that delivers interfaces for ANN. This required basically online research work. We collected all datasets and real, fake images through online research and by ourselves and then tried to extract certain facial expression parameters using feature extraction and algorithms of machine learning. For this paper, we are using the Real and Fake Face Detection dataset. After unzipping the dataset, we will find two directories: real images and fake images. These images are then converted to a training dataset.

Firstly, we have to make all the images the same size. The images are reshaped into similar dimensions. Also, these images are removed from their colors. It is to be made sure that all images are converted to a size where the image quality is not compromised. We used CNN along with other algorithms, namely VGG 16. We then compare the results and accuracy rate of previous works with our work and try to achieve a better accuracy rate.

1.4 Thesis Overview

In this paper, we first attempt to analyze a thorough study of the related works of our thesis. The following section thus contains the literature review and a background on machine learning models and architecture we have used. The undermentioned literature review entails the research papers we found related to our topic.

In the next chapter, we have the Dataset Analysis section, which comprises the Dataset collection, labelling and refining sub-sections. A detailed description of the dataset we collected from Kaggle used in our thesis is made, which incorporates how the training and testing are performed upon the dataset .

The Model Specification section comes next where the paper discusses the specifications of the models applied in our research.

The subsequent chapter of Implementation describes the implementation procedure of our models.

Next, the section of Performance Analysis analyzes the results obtained and comments on the performance. Furthermore, it goes on to state the best model that we have received on our given datatset.

Finally, the Conclusion section concludes our thesis and proposes our suggested future work.

Chapter 2

Literature Review

2.1 Related Works

Many of the researchers have worked on CNN applications in a variety of ways. Many relevant studies on the issue of face recognition, face spoofing, feature extraction, and so on have been examined in many research articles. Face anti-spoofing technologies are used to distinguish between authentic and false faces. It's one of the most pressing concerns in today's biometric applications. We have covered some of the prior study papers that we have investigated in this section of our report. The following are some of the prior works:

The authors Asim, Ming, and Javed [5] introduced a revolutionary face anti-spoofing solution based on a deep CNN architecture. It uses the LBP-TOP descriptor to extract spatiotemporal facts about genuine access and imposter videos or image sequences. This method differs from the majority of current approaches in that it does not require preprocessing procedures such as face detection, face refinement, or rescaling. LBP-TOP is based on texture feature analysis. Fake faces, on the whole, have distinct textural traits than actual ones. In this study, they look at how time and space may be combined to create a dominating countermeasure for improving Local Binary Pattern (LBP) picture sequences. To identify edges, the first layer CNN employs raw pixels. The second layer detects higher-level characteristics such as face shapes by using edges to discern simple shapes and then forms to identify higher-level elements such as face shapes.

The last three Convolutional layers are chosen in this article since they contain the greatest information when compared to the first two levels. The suggested technique is evaluated using two datasets. On the REPLAY-ATTACK, the CASIA dataset demonstrated complete detection between actual access and imposter attacks and extremely competitive results. CASIA DB has a total of 50 topics (20 for training and 30 for the test set). Each resident has twelve sequences: three true access and nine imposter attacks. Images are collected in three quality levels for each resident. Three types of spoofing attacks are included in CASIA DB: warped picture assault, sliced photo attack, and video replay assault. The validation set, on the other hand, is provided by the REPLAY-ATTACK dataset and findings are expressed in terms of EER computed on the development set and HTER calculated on the test dataset. The error rate is measured per video, not per frame, and the final attack or genuine

video score is derived by averaging the scores of all frames chosen from the video. The final score video is then classed as a real or fake assault based on that.

In another paper [6] written by Boulkenafet, Komulainen and Hadid provided an innovative and attractive methodology for detecting face spoofing using color texture analysis in another study. This research investigates how effectively several color image representations (RGB, HSV, YCbCr) can be utilized to describe the intrinsic color texture differences of real and false faces, as well as if they give complementary representations. Texture descriptions were created with gray-scale pictures in mind. However, by integrating the information retrieved from multiple color channels, it may be used to color pictures. The color texture of facial photographs is investigated using five descriptors in this article. Three recent face anti-spoofing databases are used to evaluate the performance of the proposed anti-spoofing technique: CASIA Face Anti-Spoofing Database (CASIA FASD), Replay-Attack Database, and MSU Mobile Face Spoof Database (MSU MFSD). These three datasets contain recordings of actual client visits as well as various spoofing attack attempts collected with various images quality devices such as mobile phones, webcams, and digital system cameras. Initially, the performance of color texture features will be compared to their gray-scale counterparts, and then complementary facial color texture representations will be combined to form the final face description used in the anti-spoofing method, and its performance will be compared to state-of-the-art algorithms. Finally, undertake cross-database tests to test the proposed approach's generalization capabilities. The encouraging findings of the cross-database examination show that, compared to gray-scale analogs, facial color texture representation is more reliable under unknown settings.

Authors Mo, Chen and Luo in paper [7] suggested a CNN-based approach for detecting faked facial photos. It is created using the most up-to-date approach and includes comprehensive trial findings that show the suggested approach can efficiently distinguish between fake and real facial photographs with good visual quality. GANs (generative adversarial networks) are a popular generative model that learns the distribution from large amounts of data and generates a new sample. A GAN usually consists of two parts: a generator and a discriminator. The discriminator determines if the input data is genuine or fake, while the generator understands how to make fake data that is indistinguishable from actual data. The RGB color picture with the size M M 3 is used as the model input. First, use a high pass filter to convert the input pictures into residuals and then feed the residuals into three-layer groups. A convolutional layer (3x3 sizes, 1x1 stride) equipped with LReLu and a max-pooling layer (2x2 sizes, 2x2 strides) are included in each group. The first convolutional layer's output feature map number is 32, whereas the following convolutional layers' output feature map numbers are double the corresponding input feature map numbers. The last group's output feature maps are then combined and fed into two fully connected layers. Both are equipped with LReLu and have 1024 and 512 units, respectively. Finally, the output probability is calculated using the Softmax layer. Finally, while current GAN-based approaches may produce realistic-looking faces (or other visual objects and sceneries), certain evident statistical aberrations are necessarily created and can serve as shreds of proof for fraudulent ones, according to experimental data.

In a paper [8] written by Benlamoudi, Samai, Ouafi, Bekhouche, Taleb-Ahmed and Hadid suggested an anti-spoofing approach for discriminating between 'live' and 'fake' faces in their article. The Fisher Score was utilized to minimize the histogram by focusing on the LBP overlapping process. The LBP is an image fixer that transforms a picture into a matrix or a more detailed picture. A 3x3 pixel picture block was used in the original LBP. To produce a label for the center pixel, the block's mean pixel value is thresholded, then multiplied by powers of two, and finally summed. Because the neighborhood is 8 pixels wide, there are a total of 28 = 256 labels available. Because the neighborhood is made up of 8 pixels, there are a total of 28 = 256 labels that differ in terms of the relative gray levels of the center and its surroundings. One of the most prominent approaches for picking characteristics is the Fisher Score. Fisher scoring is based on the notion of selecting each attribute individually depending on its score under the Fisher criterion. In comparison to much previous work, the methodology examined on NUAA Photograph Imposter and CASIA Face Anti-Spoofing Databases, which contain several genuine and phony faces, demonstrated encouraging results. They utilized the Viola-Jones method to detect all of the components of the face photos for the experiment and the Active Shape Model with Stasm to detect the eyes.

All cropped faces are reduced to 64x64 pixels in size. After that, they partition the facial picture into three 3×3 overlapping sections and apply the LBP operator to each of them. Each region's 243 bin histograms are produced and merged to form a single 2187 bin histogram. The Fisher score is then used to minimize the number of histogram bins. Finally, they utilize an SVM nonlinear classifier with a radial basis function kernel to identify whether or not the input image is a living face. A collection of positive (true faces) and negative (false faces) samples from the data set are used to train the SVM classifier initially. They obtained an EER of 1% using the NUAA database, which is the best result among the other research. They found good results for poor and normal quality (EER = 7) in the CASIA Database (2% and 8.8%, respectively). Finally, in the case of the textural algorithm, the general test of their work is superior to the others.

In a paper [9] written by Pitaloka, Wulandari, Basaruddin, and Liliana refined the CNN algorithm to recognize 6 fundamental emotions and compared several pre-processing approaches to identify effects in order to demonstrate their performance. CNN is the place to be. The studies were conducted utilizing data from JAFFE, CK +, and MUG, which meant that each participant was given instructions from an expert to display six different emotions. CK + 5 is a popular tool for detecting emotions and analyzing facial expressions. Has 123 participants, Japanese Female Facial (deleted) (JAFFE) has 213 photos exhibiting 6 fundamental emotions of 10 Japanese women, and the MUG record contains 86 Caucasian individuals Crop, Resize, Noise Addiction, and Normalizations. The picture from the preprocessing stage is fed to the first convolution layer with a core size of 5 x 5 to recognize facial emotions using CNN. The rectified line unit (ReLu) is used to activate all layers, with the goal of preserving the qualities of each Output. The CNN model is trained using the RMSProp software. AdaGrad altered it to enhance speed, particularly for non-convex configurations, and to modify the gradient to exponential. The error function is based on cross-entropy. On an NVIDIA GPU version 375.74 from

Nvidia-375, CPU type 16xi7-5960X, 65 GB RAM, and Ubuntu 16.04 as the operating system, the preprocessing and CNN stage were completed using Python libraries such as OpenCV and TensorFlow. The accuracy of the preprocessing stages b + f was the greatest, at 93.14 percent. In terms of accuracy, the Global Contrast Normalizing (GCN) step outperforms previous normalization procedures, although it falls short of the ROI. Finally, when compared to another preprocessing phase and raw data, we discovered that face recognition as a single preprocessing phase obtained a substantial result with 86.08 percent accuracy. Combining these strategies, on the other hand, can raise CNN accuracy to 97.06 percent.

From another research article, we get to know that the authors Shih and Liu [10] developed a new approach of face identification by merging DFA, face class modeling, and SVM, according to another study. By collecting the input picture, its 1-D Haar Wavelet interpretation, and its frequency estimations, selective feature analysis generates a feature vector. After that, the face class modeling calculates the Document of the face class and develops a distribution-based metric for categorizing faces and non-faces. Finally, SVM distinguishes the structures in input data into the face or non-face class using the distribution-based metric. Experiments involving photographs from the MIT–CMU test sets were used to assess the effectiveness of their novel face recognition technology. The distribution-based measure classifies the input picture forms into different kinds: face (patterns that are near to the face), faceless (patterns that are not close to the face), and faceless (patterns that are not near to the face) (patterns outside the face class) and uncertain class (patterns that is neither near nor distant from the face). Eventually, an SVM detects faces in the undetermined class and sends their layouts to the DFA-SVM decision rule, which categorizes them whether as face or non-face. The SVM is a particular manifestation of statistical learning theory. It refers to a technique called structural risk reduction, which reduces the functional impact in terms of both scientific risk and probability value. The DFA-SVM approach uses training samples from the FERET database, which includes 600 frontal face photos. Images from multiple DFA-SVM algorithms are included in the MIT-CMU test kits, which provide the test data. The DFA-SVM approach, which trains on a basic collection of pictures yet operates on many more complicated pictures, demonstrated high convergence speed in the experiments. Facial recognition photographs for training an SVM with a linear regression of the second degree. Both face and non-facial pictures are standardized to a spatial resolution of 16 16 pixels. Our DFA-SVM approach yields a correct image stabilization rate of 98.2 percent with two error rates using CMU test kits.

The authors' Neves, Tolosana, Vera-Rodriguez, Lopes and Proença [11] in their paper introduced a new technique. It is based on the GAN fingerprint removal facial identification system. In their research paper, they considered four different public databases. They have to use CASIA-WebFace and VGGFace2 for detecting a real face. CASIA-WebFace consists of 494,414 images from 10,575 actors and actresses of IMDb. Those images have a random variety of pose and facial expression and it also contains different resolutions. From 9,131 different data, VggFace2 consists 3,31 million images. Each data has a mean of 363 pictures. Those pictures have a lot of variations in pose, age, illumination, ethnicity and profession. In this paper, they have used TPDNE and 100K-Faces for synthetic images. They have collected

unique pictures from the website2. 100K-Faces have consisted of 100,000 synthetic face images using StyleGAN. From a variety of 69 models, StyleGAN was a trained network using around 29,000 photos. They have used the PyTorch framework for all the testing combinations with an NVIDIA Titan X GPU. They divided every database into two different datasets. The first one is used for the development and training of the systems (70%). The second one for the final evaluation (30%). They also divided the development dataset into two separate subsets. They are training (75%) and validation (25%). In the experimental framework, the same number of real and synthetic pictures has been considered. For the development part, multiple users are considered for biasness. Suppose when the picture quality is decreased by 4/7. They have shown that these Outputs contain poor generalization capacity of state-of-the-art face manipulation identification systems to unknown conditions.

In a research paper by J. Yang et al., [12], a supervised approach is taken to learn features applying deep convolutional neural network (CNN)—a high discriminative potential– that results in a more than 70% decrease of Half Total Error Rate (HTER) being achieved. For data preparation, this novel method uses a common Viola-Jones for face detection in OpenCV. A set of local binary features are extracted in this step which is used for learning linear regression in each cascade, obtaining refined face location based on face landmarks. Furthermore, the authors take the background into account for classification. They mentioned in their paper that studying the background allows CNN to learn discriminative methods more effectively than hand-crafted feature extraction methods. In this CNN architecture, the input images from the datasets used are prepared with five scales; images contain more background with the increase in scale. The data is augmented temporally besides spatial augmentation. Due to more informative data, better performance was achieved in most of the scenarios with multiple frames. As they mentioned in their paper, when the CNN is fed with multiple frames, the CNN can learn both spatial and temporal features.

Experiments are carried out on REPLAY-ATTACK and CASIA datasets. In the CASIA dataset, the spoofing images are implemented using attacks of three categories, i.e., warped photo of the person, cut photo and electronic screen attacks. In REPLAY-ATTACK, three spoofing types are used, namely, print attack, digital photo attack, and video attack.

For performance comparison, Half Total Error Rate (HTER) is measured. Intra-test and Inter-test are conducted on each dataset to evaluate the generalization ability of the method. Moreover, both datasets are combined to evaluate the performance further. Drastic improvements are found in both the dataset results. In the experiments, it is well-observed that background areas significantly improve anti-spoofing models' competency to generalize. On the CASIA dataset, the best scale was found to be 3, while 5 was the best scale for the REPLAY-ATTACK dataset. The input data had caused such a difference, as was stated. The authors formed an argument stating that face-spoofing is not a facial classification concern; rather, on regions where the fakeness of the face can be identified. While the inter-test protocol did not show satisfactory results, HTERs lower than 5% were achieved in the

intra-test and combined protocols on two of the datasets, proving to be remarkable improvements in this area of face-spoof classification.

In another paper, S. B. Rajeswaran et al. [13] proposed a face-spoofing method using Convolutional Neural Network, based on color features extracted by L*a*b* color space model. Color features such as luminous intensity and other channel chroma components are used to distinguish between real and spoof faces. L*a*b* color space has three parameters- L*, a* and b* for luminosity, green-red component and blue-yellow color component, respectively. There are two stages in this proposed system – Face detection in an image and Face Verification of real or spoof face in an image. The first mentioned step is implemented using two methods:

1. Viola-Jones algorithm by using Haar-Cascade features and AdaBoost techniques 2. Histogram of Oriented Gradients (HOG)

HOG is observed to produce greater accuracy in face detection and is thus used in the spoof-detection system. Features of local texture and distortion in the image are found. Subsequently, the extracted features are trained for Face Verification using VGG7 CNN architecture in two phases: Train-Phase and Test-Phase. The model is trained with real and spoof face and the weight file is used to classify the image in the test phase. VGG7 CNN architecture consists of 5 convolutional layers, 3 max-pooling layers after the convolutional layers; they are followed by two fully connected dense layers and the softmax classifier is used to classify the images. Adam is used for the optimizer and the loss function sparse categorical entropy is used. The experiment is carried out in real-time using a hardware system and an external camera; the OpenCV package is used to support Tensorflow, Torch and Caffe. The VGG7 CNN architecture, used in classifying input images, failed to predict the faces properly under certain improper light conditions, while it produced precise results for face-spoofing detection under trained constant surrounding light.

Authors Almabdy, S., and Elrefaei, L., [14] in a paper proposed a CNN based approach for face recognition and investigated the resulting face Recognition performance through CNN using two approaches, the first one being, application of Pre-trained CNN AlexNet and Resnet-50 with SVM and the second approach being, extraction of features and classification through applying transfer learning on AlexNet Model. Pre-processing was performed, followed by face recognition, and subsequently, face classification. Tests on various datasets such as FEI face dataset, ORL dataset, LFW dataset and Youtube dataset were carried out. The different methods using Pre-trained models and SVM were compared and analyzed, henceforth. Finally, accuracy ranging from 94%-100% was obtained on the given datasets. The suggestion of introducing more datasets was made for future developmental work in this field so as to train the mentioned CNN models.

Studies on cues by R. Tronci et al. [15] for photo-attack detection in another paper have given insight and perspective into the real-life problem of spoofing attacks. According to the paper, a face-spoof attack can be generated in three ways: generating a photograph, reproducing a video, presenting a 3D reproduction of a

face, all of a valid user. It is stated that the problem of photo-attack detection can take place in two complementary directions: static and video analysis, where static analysis is based on a photo-attack causing a certain loss of information and peculiar noise; video analysis is to detect facial physiological clues like blinks, changes in facial expressions and mouth movements. In this work, the problem of 2D face-spoof detection is regarded by the authors as a fusion of multiple clues that result from examining static and video studies in the respective scenes. This procedure entails three steps, which are: static feature extraction, video-based feature extraction and score level fusion. For static analysis, visual features, for instance, color and edge directivity descriptor, Gabor texture, RGB, are used. For video analysis, clues such as blink and motion of the foreground mask are taken into account. Using a weighted sum, fusion is next performed at the score level. The static analysis implemented in this paper exploits SVM classifiers and different visual features as mentioned above and, through a dynamic score combination methodology, yields excellent performance.

Another paper by authors L. Souza et al. [16] presents an extensive analysis of face-spoof detection research works published in recent times. The analysis is done based on fundamental parts, i.e., classifiers and descriptors. Descriptors were classified into motion, color, texture, shape, reflectance and frequency, while classifiers were categorized into discriminant, distance metric, regression and heuristic. Taking into consideration of the most significant public datasets in the field, namely CASIA, MSU-MFSD, Replay-Attack, Print-Attack, etc., the survey brings forth a comparative performance analysis for a greater understanding of advancement and future aims in the field. The authors, in their paper, as a result of their survey, mentioned the following face-spoofing attacks possible: flat printed photo, eye-cut photo attack, warped photo attack, video playback attack and mask attack.

From the aforementioned descriptors, texture features are the most commonly used evidence for spoof detection, assuming that printed spoofs contain texture patterns that are absent in real faces. In this case, LBP is observed as the very first choice, which is a gray-scale, illumination invariant, texture coding technique that compares and labels every pixel relative to its neighbor, resulting in a binary number through concatenation. The texture is then described by a histogram of the final computed labels generated. Different configurations such as LBPV, multi-scale LBP, Muvis, etcetera have also been explored. Motion is the second crucial descriptor used in mainly two ways for this purpose. One way is the detection and description of intra-face variations such as head rotation, eye blinking and facial expressions. An alternative path to use motion is through evaluating how consistently a user interacts in his given external surroundings, wherein motion correlation that lies between the face of the person and its background are evaluated.

Classifiers, such as discriminant techniques, are most widely used, which distinguishes different classes by minimizing intra-class and/or maximizing inter-class variations. SVM is the most common classification technique, which often presents superior results to the rest. Performance analysis is done based on two types of errors: number of false acceptance (NFA) and number of false rejection (NFR).

Comparing these error rate pairs and showing ROC curve based on their probabilities, it is observed that spoofing attack continues to be a security challenge and despite the robust methods obtained, they do not favor breakthroughs in the field since they tend to follow the same recipe. There is seen to be a considerable gap from research work to real work applications in an efficient way. It is suggested that further work is focused on more difficult data-sets and unbiased evaluation methods.

Chapter 3

Dataset Analysis

In this chapter, we have provided details of the data collection process, data labeling and refining stage, dataset validation and the test data.

3.1 Data Collection

It is no surprise to us that cybercrime is spiraling in this generation. Companies are paying loads to machine learning engineers only to explore biometric facial recognition for their security. However, even the best facial recognition techniques have their flaws. Photos of general people are fed into recognition software for spoofing.

As we are focused on classifying the techniques of face spoof, it is important that a correct dataset is chosen. In this paper "Real and Fake Face Detection" dataset is used, which was provided to Kaggle by the Dept. of Computer Science of Yonsei University. It was created taking into consideration the dangers that come with a fake identity. The dataset consists of high-quality photoshopped images of people's faces. The pictures are blended with different faces where the features are manipulated extensively.

The images are altered by experts only because the creators kept in mind the training of the classifier for these images. Inside the dataset, two directories can be found: (i) training real (1081 files), (ii) training fake (960) files.

3.2 Data Labeling and Refining

The collection of these images is categorized in two folds: real and fake. The fake images are changed by experts, who successfully merged different features into one. The real images are taken as candid so that our models can determine results on a higher level.

3.2.1 Training Dataset

We used around 1634 images to fit the model. This is almost 80% of the whole dataset, including both real and fake images. We fed the images into our models, and our models were executed on them.

3.2.2 Validation Dataset

Of these 1634 training images, 20% of the data was used to provide an unbiased evaluation of the training dataset. The evaluation gets more biased as configurations of the models are integrated with the validation dataset.

Figure 3.1: Fake Images

Figure 3.2: Real Images

3.3 Testing Dataset

We used 20% of the dataset for testing. This test dataset tells us on which standard the evaluation of the models is. After training of the dataset is done, the test dataset is worked upon by the models. Different models of CNN provide different levels of accuracy. It is to be noted that no testing dataset was used in the training dataset, and no training dataset was used in the testing dataset.

Chapter 4

Model Specification

In this section, we will look at the different convolutional neural network based models and algorithm of SVM that we have used to determine the results.

4.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is one of the practiced forms of Artificial Neural Network. Initially, it was used to recognize intricate image patterns with accurate and precise architecture. CNN is vastly used in neural networks for image classification and recognition. Basically, CNN will take an input of the image and look through its pixels (h*w*d)[17][18].

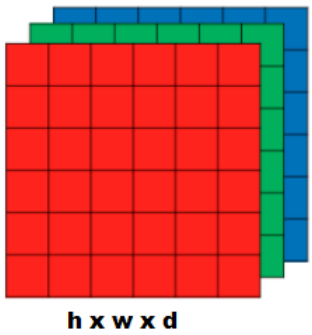


Figure 4.1: An Image Matrix of dimension

For testing and training, images are passed through stacks of convolution layers with kernels as filters. Layers include pooling padding, moving through fully connected layers and application of the soft-max function.

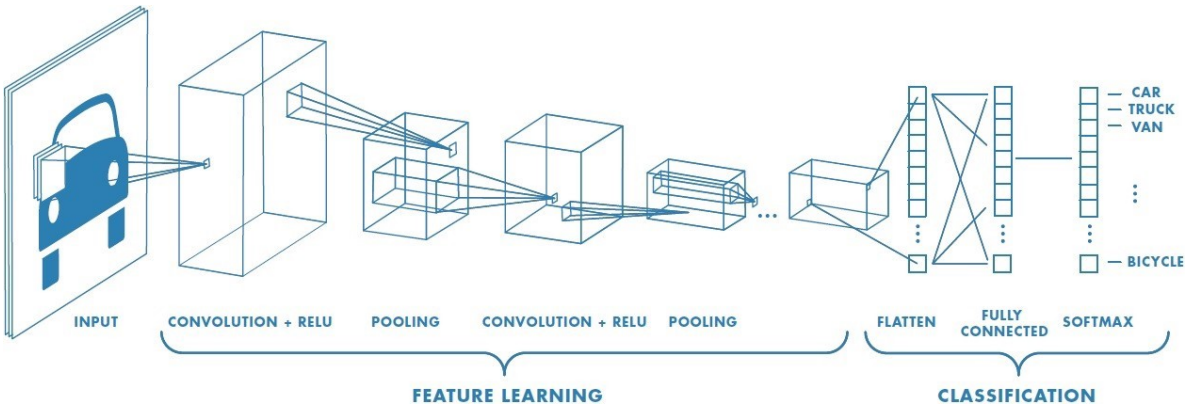


Figure 4.2: Neural network with many convolutional layers

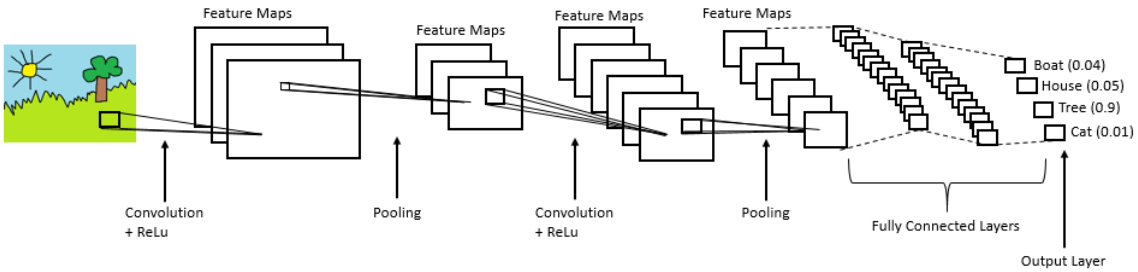


Figure 4.3: Complete CNN architecture

4.1.1 Layers of CNN

- **Convolution Layer**

The first layer of CNN is called convolution. It is used to extract the input image. The convolution layer gains the knowledge of input the image features in order to protect the image resolution. It is done by using small squares of the image data. This is fully a mathematical operation that uses two inputs as an image matrix and one kernel.[18]

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**

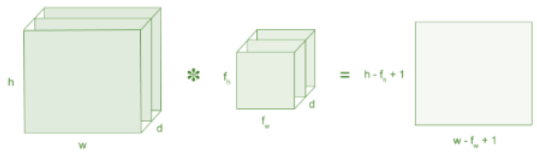


Figure 4.4: Image matrix multiplies kernel or filter matrix

Convolution performs different functions using different filters or kernels. Consider figure 4.5 illustrated on the following page as an example:

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Figure 4.5: Few common filters

- **Strides**

In CNN, the number of pixels that move over the input matrix is called the stride. The movement of filters over pixels depends on the stride number.

- **Padding**

When filters do not fit the image properly, we used padding. It is done by padding the image with zeros, which is called zero-padding, or by removing the image segments which did not fit into the kernels, known as valid-padding.

- **Non Linearity (ReLU)**

To operate the non-linear function, CNN used a Rectified Linear Unit. The output comes out as f(x) = max(0,x). This operation is essential because it provides non-linearity to the image input.

- **Pooling Layer**

When the image input is too large, the padding layer subtracts the parameter numbers. CNN uses spatial pooling to minimize the measurements of the image, but this also reserves the principal information. Max-pooling, Average pooling and Sum pooling are the three types of spatial pooling.

- **Fully Connected Layer**

The image input matrix is first converted to a vector and inserted into a fully FC layer. The image below shows the matrix being flattened into a vector and the fully connected layer turning the vector into a model.[19] [20]

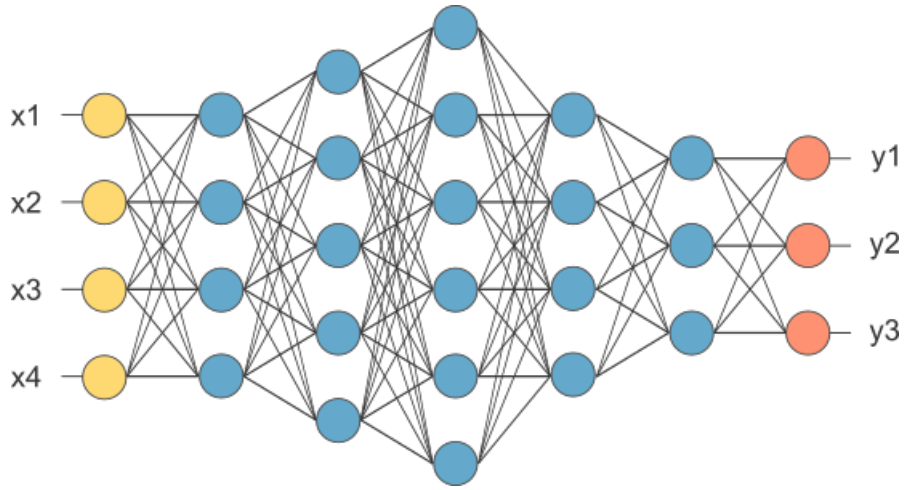


Figure 4.6: Convolutional Nodes

In order to implement the CNN model, we have imported and installed all the necessary packages in python. These include NumPy, sklearn, TensorFlow and matplotlib. Next, we needed to load the data. The dataset is loaded from the data directory and then the features are selected by dividing the dataset to dependent (target) variable, which is represented as y and independent (feature) variables, represented by X'. Furthermore, the images are all converted to gray-scale and then resized at width = 150 and height = 150. Our dependent variable comprises two classes, real and spoof.

Then feature vectors are created for each class and saved.[19] After loading the data, we needed to divide the dataset into the features and corresponding labels. Then it is divided into training and test sets using sklearn and subsequently, we performed feature scaling.[21] The model is then formulated, compiled and fit-trained and is next ready for the testing phase, where it consequently acquires an accuracy of 98%.

CNN Procedure:

An overall summary of the layers of our CNN model for detecting face-spoof are given below:

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 256)	2560
activation (Activation)	(None, 48, 48, 256)	0
max_pooling2d (MaxPooling2D)	(None, 24, 24, 256)	0
conv2d_1 (Conv2D)	(None, 22, 22, 256)	590080
activation_1 (Activation)	(None, 22, 22, 256)	0
max_pooling2d_1 (MaxPooling2	(None, 11, 11, 256)	0
conv2d_2 (Conv2D)	(None, 9, 9, 256)	590080
activation_2 (Activation)	(None, 9, 9, 256)	0
max_pooling2d_2 (MaxPooling2	(None, 4, 4, 256)	0
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 1)	4097
activation_3 (Activation)	(None, 1)	0
dense_1 (Dense)	(None, 1)	2
activation_4 (Activation)	(None, 1)	0

Figure 4.7: Summary of CNN Model

4.2 VGG16

VGG16 is another adaptation of the VGG model with 16 convolutional layers. It works in 3*3 convolutions with many filters. It provides the best extraction of features from images and is vastly preferred now. Because of having 138 million parameters, it is a challenge to handle. But, it can be carried out using transfer learning where the model is previously trained on a dataset and for the best accuracy these parameters are improved. The values of updated parameters can be used .[22]

VGG16 Layers
Convolution using 64 filters
Convolution using 64 filters + Max pooling
Convolution using 128 filters
Convolution using 128 filters + Max pooling
Convolution using 256 filters
Convolution using 256 filters
Convolution using 256 filters + Max pooling
Convolution using 512 filters
Convolution using 512 filters
Convolution using 512 filters+Max pooling
Convolution using 512 filters
Convolution using 512 filters
Convolution using 512 filters+Max pooling
Fully connected with 4096 nodes
Fully connected with 4096 nodes
Output layer with Softmax activation wit 1000 nodes.

Figure 4.8: VGG16 Layers

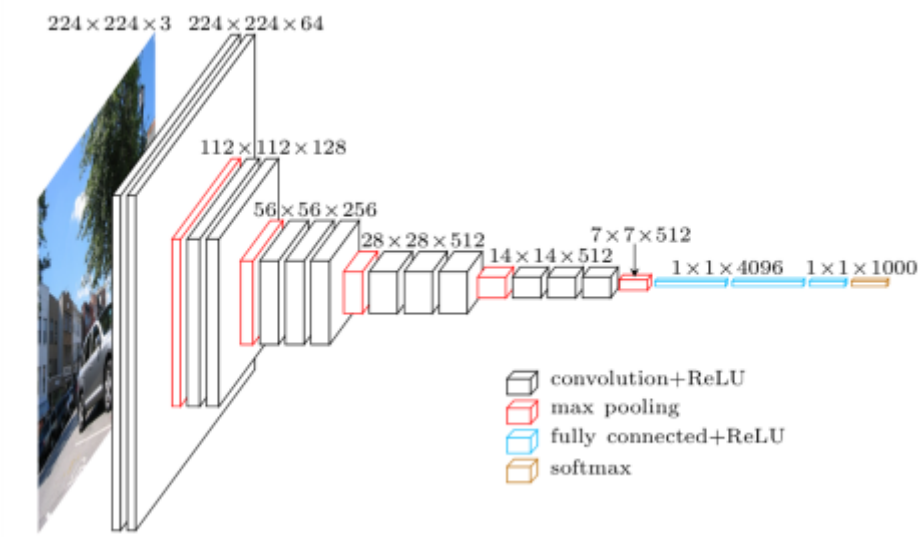


Figure 4.9: VGG16 Overview

An RGB Image of fixed size (224*224) is inputted to layer 1 then runs through stacks of convolutional layers. Filters are used using small sensory receptors of 3*3 to seize the whole image fully. One of the configurations performs linear transformation utilizing 1*1 convolution filters. Pixel of 1 is fixed for convolution stride. The spatial padding is also used here to preserve the resolution after convolution. Five max-pooling layers implement spatial pooling. Lastly, the 2×2 pixel window is run through max-pooling with a stride of 2.

Three Fully-Connected layers pass through many convolutional layers where the first two Fully-Connected layers consist of 4096 channels each, and the third one executes ILSVRC classification in 1000-way, and each class contains 1000 channels. The Soft-max layer is the final layer. The configurations of the last three layers are in the same network. The layers are rectified with a Rectified Linear unit for non-linearity.[23] [24]

In a recent paper of 2018, VGG16 was implemented over a dataset provided by the large company State farm, which consisted of 2D dashboard images. There are more than 20,000 training sets and approximately 80,000 testing sets. The training was done over Google Cloud Platform with a 32 batch size. Data was augmented on both models. For both, the models' accuracy was 75%-77%.[24]

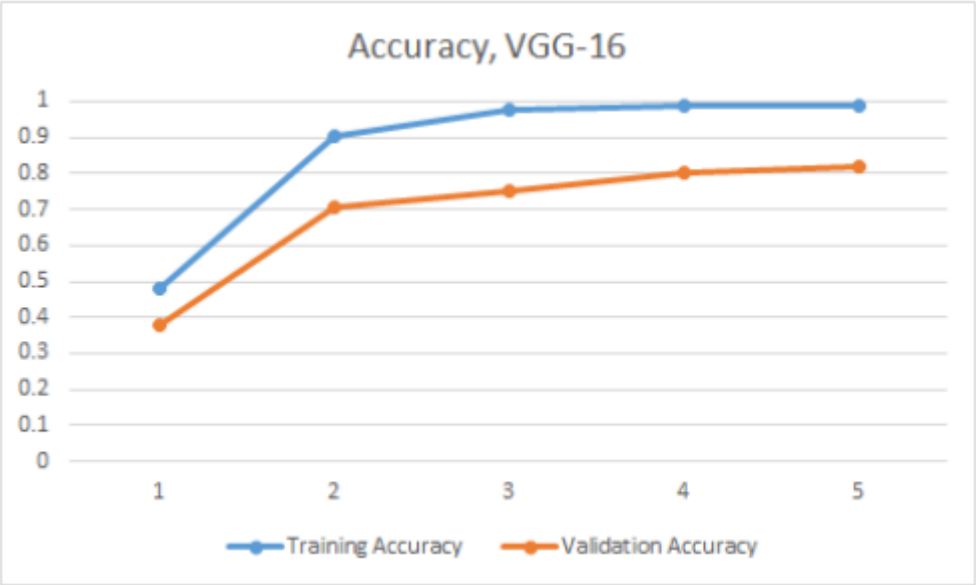


Figure 4.10: Training and validation accuracy of VGG16

4.3 SVM Classifier)

Support Vector Machines (SVM) is a supervised machine learning algorithm commonly used to solve problems that involve classification. It tries to find a fine line of segregation in the dataset and tries to divide the data along a plane known as the hyperplane. The hyperplane divides the points on an x dimensional space, where x is the number of features or categories. This helps to categorize the data points in an organized manner. There can be many possible hyperplanes to split the data, but we need to find the one that created the greatest distance between the data points. The distance between the nearest two sample points is known as the margin, and the points are the support vector.

After using the SVM classifier, we got an accuracy of 50% for the verification of the input data. The hypothesis function h is determined as:

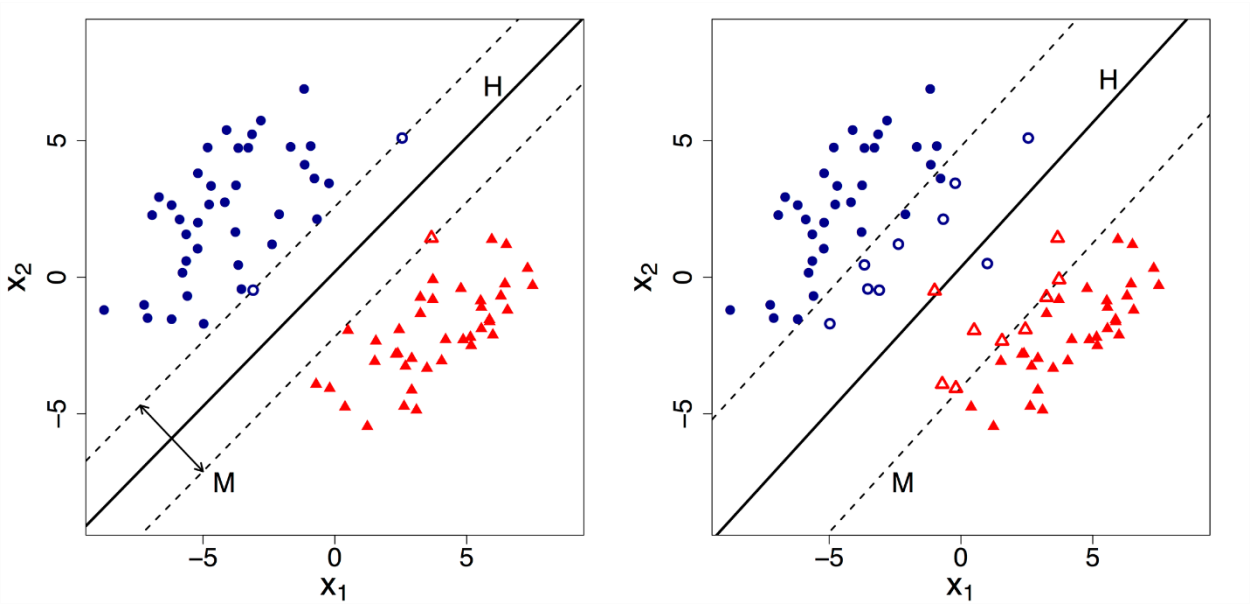


Figure 4.11: Support Vectors

The following hypothesis function h is defined as:

$$h(x_i) = \begin{cases} +1 & if\ w \cdot x + b \geq 0 \\ -1 & if\ w \cdot x + b < 0 \end{cases}$$

Here, the point above the hyperplane or on the hyperplane will be categorized as class +1, and the point below the hyperplane will be categorized as class -1.

Chapter 5

Implementation

5.1 Overview

The implementation is performed with deep learning including high-level API Keras that is sitting on top of TensorFlow, thus simplifying it. Now, both TensorFlow and Keras are open-source software libraries that deliver interfaces for Artificial Neural Networks (ANN).

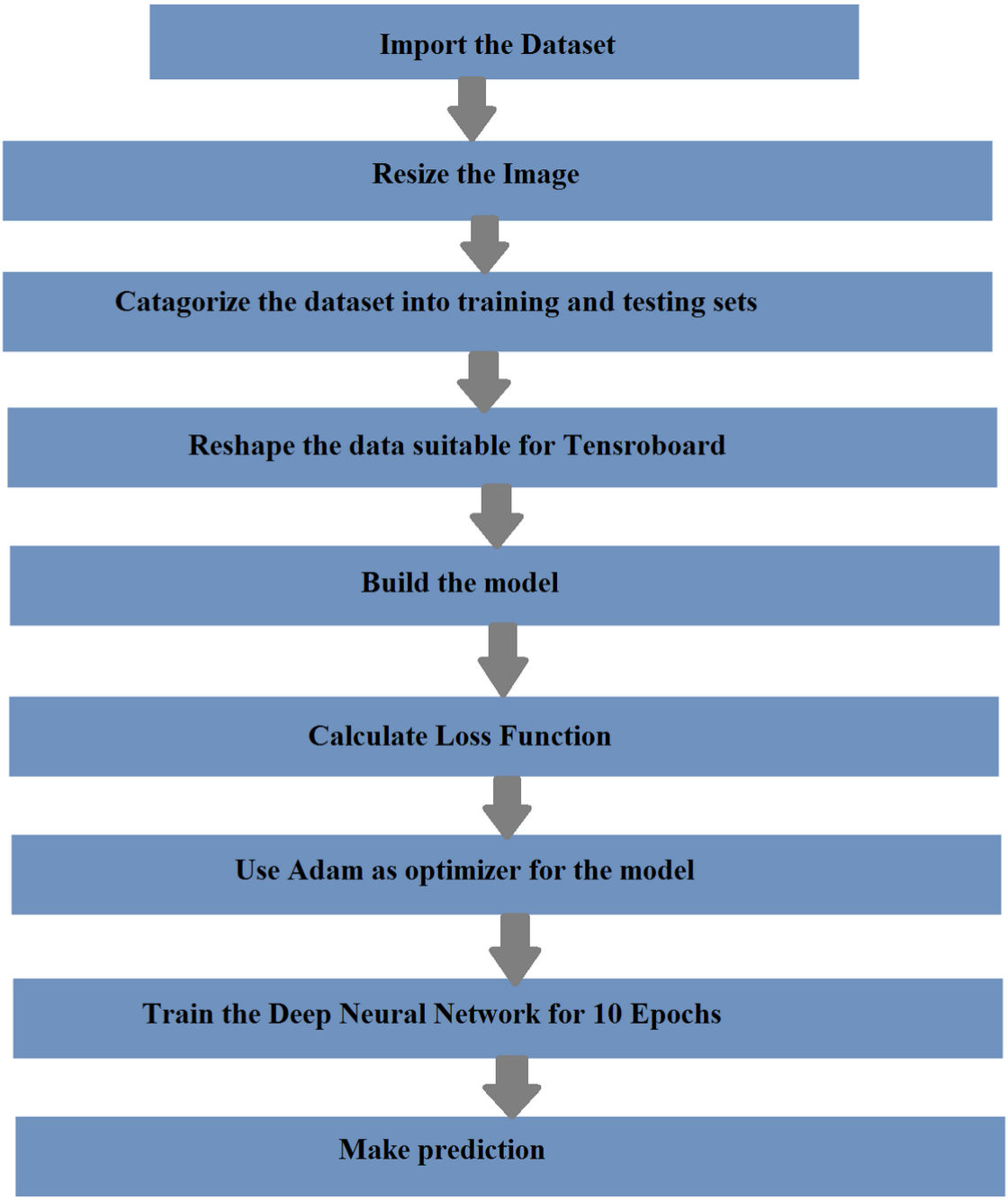


Figure 5.1: Implementation Flowchart

To begin with, the TensorFlow version has been updated to 1.10.0. The model is built sequentially. This means that a direct order has to be followed without going backward. The input image needs to be flattened since neural network layers are flat. Keras has a built-in Flatten layer. This flattened image is the input layer. This layer will go through hidden layers under a neural network layer called a dense layer.

This is also known as a fully connected layer; wherein each node is connected to the previous and succeeding layer. The dense layer has 128 units within. The activation operation for this layer is the Rectified Linear Unit (ReLU). Multiple dense layers can be added as well.

The final layer has 10 nodes where per node provides a single number prediction. In this case, our activation function is a soft-max function since we are really actually looking for something more like a probability distribution which, the possible prediction options that we are passing features through are. Following this, the model is compiled, and the model is ready to be trained .[25] [26]

For the calculation of the loss metric, Adam optimizer, a default in Keras, is used. Loss metric is the calculation of any sort of error, as the neural network only aims to reduce loss. Then the output from the layers is fit into the model. The loss metric decreases thus improving the accuracy of the model.

5.2 Loading the data

For this paper, we are using the Real and Fake Face Detection dataset. After unzipping the dataset, we find two directories: real images and fake images. These images are then converted to a training dataset. The first task here is to make all the images the same size. The images are reshaped into similar dimensions. Also, these images are removed from their colors. It is to be made sure that all images are converted to a size where the image quality is not compromised.

Before training the dataset, some images are kept aside for testing. This can be done manually on the computer. The test sets are kept in another folder. The training dataset is created and printed in python. It is important to make sure the dataset is balanced. Here, balanced means there should be the same number of samples in each class, and if the dataset is not balanced, we can either trim the samples of the larger set to match the smaller set or class weights can also be passed to the model. Balancing is necessary because we do not want the model to learn to predict a single class right away and then get stuck when newer classes are introduced. Additionally, the dataset must be shuffled, or else the classifier will learn to predict patterns.

5.3 Implementation with CNN

The convolutional neural network now has the maximum popularity in the field of face detection

The basic CNN follows this structure: Convolution -¿ Pooling -¿ Convolution -¿ Pooling -¿ Fully Connected Layer -¿ Output The convolution process collects atomic data and creates a feature map based on that raw data. Pooling is a top-down example, also called "max-pooling," in which we select an area and then use the maximum value of that area which gives a new value for the entire region. The fully connected layer has all its nodes "fully connected" like typical neural networks. However, convolutional layers are not always fully connected. Therefore, each level of convolution and pooling is also a hidden layer inside CNN. After that, we have a fully linked layer and then an output layer.[24] [22]

5.4 Analyzing Model with Tensorboard

Tensorboard is an application that provides an outlook of the models. We need to import Tensorboard into our model. The tensorboard callbacks object from the trained dataset. This callback is passed through a fit method. The callbacks come out as a list and other callbacks can be passed through the list as well. The model is defined using CNN and it goes under all its structures. After the code is run, the output is saved in a new directory. Using a tensorboard, the results can be visualized. This is the result we have found without any activation added to the dense layer:

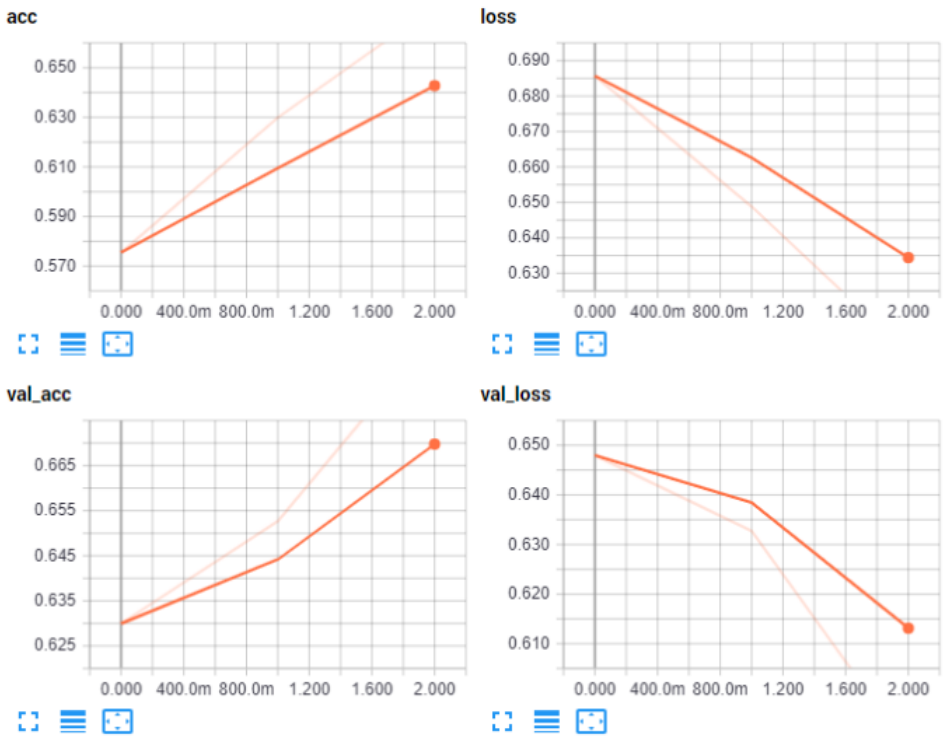


Figure 5.2: The dense layer without activation added

The result we found after adding an activation function to the dense layer is given below:

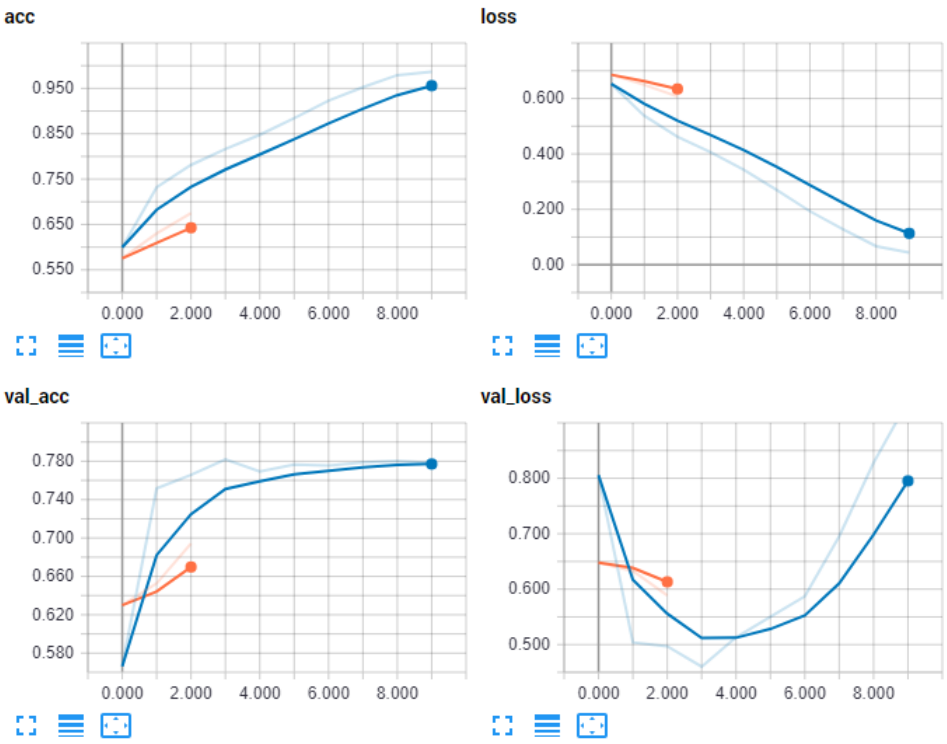


Figure 5.3: The dense layer after activation added

Optimizing Models with TensorBoard:

Here, we modified the layers in the architecture. At first we used layers from figure 5.4 to get a decent result. We made sure to revise the older parameters while changing the newer ones. None of the rounds in our optimization are similar. Initially, the models are given different weights. This had an impact on the model, particularly on the number of epochs.

```
1-conv-32-nodes-0-dense-1534801383
2-conv-32-nodes-0-dense-1534801383
3-conv-32-nodes-0-dense-1534801383
1-conv-64-nodes-0-dense-1534801383
2-conv-64-nodes-0-dense-1534801383
3-conv-64-nodes-0-dense-1534801383
1-conv-128-nodes-0-dense-1534801383
2-conv-128-nodes-0-dense-1534801383
3-conv-128-nodes-0-dense-1534801383
1-conv-32-nodes-1-dense-1534801383
2-conv-32-nodes-1-dense-1534801383
3-conv-32-nodes-1-dense-1534801383
1-conv-64-nodes-1-dense-1534801383
2-conv-64-nodes-1-dense-1534801383
3-conv-64-nodes-1-dense-1534801383
1-conv-128-nodes-1-dense-1534801383
2-conv-128-nodes-1-dense-1534801383
3-conv-128-nodes-1-dense-1534801383
1-conv-32-nodes-2-dense-1534801383
2-conv-32-nodes-2-dense-1534801383
3-conv-32-nodes-2-dense-1534801383
1-conv-64-nodes-2-dense-1534801383
2-conv-64-nodes-2-dense-1534801383
3-conv-64-nodes-2-dense-1534801383
1-conv-128-nodes-2-dense-1534801383
2-conv-128-nodes-2-dense-1534801383
3-conv-128-nodes-2-dense-1534801383
```

Figure 5.4: Modified Layers

Figure 5.5 below shows the result after a combination of layers from figure 5.4 and activation in the dense layer. We will analyze the performance of the best CNN model obtained thereby, in our oncoming chapter.

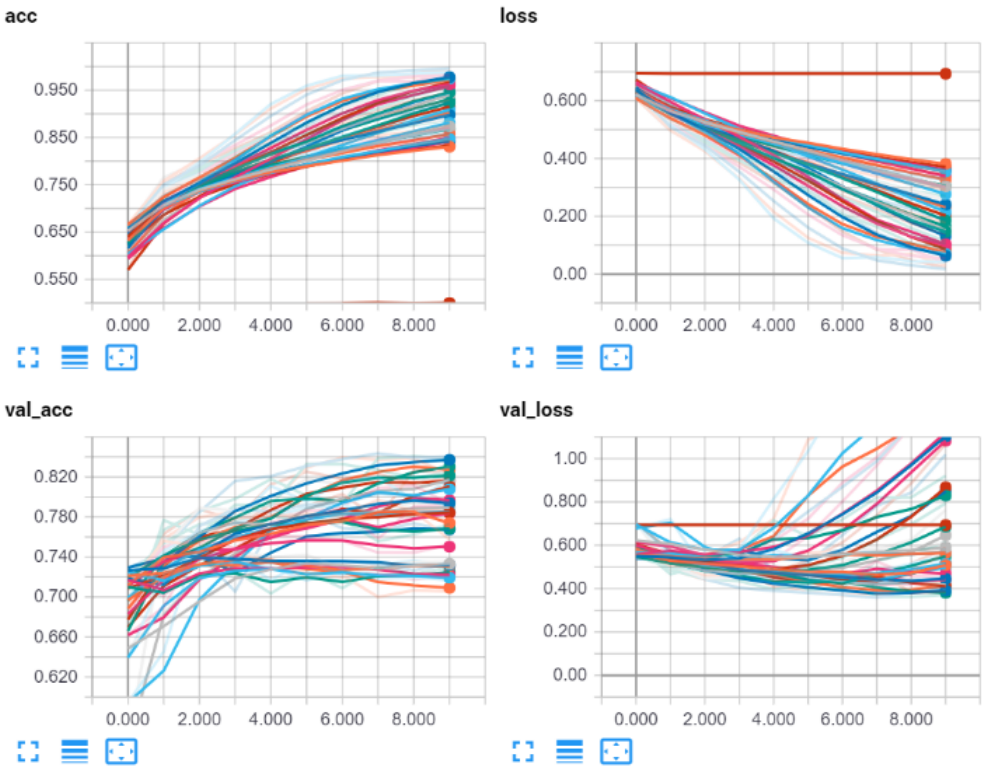


Figure 5.5: Combinations of different layers of CNN Models

Chapter 6

Performance Analysis

As we have already specified the models and architecture, we effectively ran our dataset on, we now analyze the performance of each of them. We shall observe how the performance between each one varies significantly and note the model which consequently shows the greatest performance amongst all.

6.1 CNN- Model 256x3

From our research, we have come to a conclusion that an increase in the convolutional layers has resulted in the best performance, as has been acquired. Hence, we have applied 3 convolutional layers; correspondingly, we have applied 3 activation and pooling layers, and finally, 2 dense layers in the creation of the model CNN and found an accuracy of 98.9%. We observed that in comparison to the CNN architecture of VGG 16, this CNN model took less time to execute.

Figure 6.1 shows the gradual rise in accuracy and fall in loss over every epoch of the training of the model on the given dataset.

```
Epoch 1/10
126/126 [==============================] - 167s 1s/step - loss: 0.6948 - a
ccuracy: 0.5128 - val_loss: 0.6507 - val_accuracy: 0.6438
Epoch 2/10
126/126 [==============================] - 179s 1s/step - loss: 0.6621 - a
ccuracy: 0.5983 - val_loss: 0.6103 - val_accuracy: 0.6746
Epoch 3/10
126/126 [==============================] - 174s 1s/step - loss: 0.6052 - a
ccuracy: 0.6779 - val_loss: 0.6054 - val_accuracy: 0.6734
Epoch 4/10
126/126 [==============================] - 188s 1s/step - loss: 0.5564 - a
ccuracy: 0.7040 - val_loss: 0.5144 - val_accuracy: 0.7478
Epoch 5/10
126/126 [==============================] - 183s 1s/step - loss: 0.4194 - a
ccuracy: 0.8088 - val_loss: 0.4805 - val_accuracy: 0.7920
Epoch 6/10
126/126 [==============================] - 234s 2s/step - loss: 0.3107 - a
ccuracy: 0.8783 - val_loss: 0.4023 - val_accuracy: 0.8234
Epoch 7/10
126/126 [==============================] - 350s 3s/step - loss: 0.2008 - a
ccuracy: 0.9263 - val_loss: 0.2723 - val_accuracy: 0.9175
Epoch 8/10
126/126 [==============================] - 3838s 31s/step - loss: 0.0927 -
accuracy: 0.9783 - val_loss: 0.2094 - val_accuracy: 0.9320
Epoch 9/10
126/126 [==============================] - 173s 1s/step - loss: 0.0534 - a
ccuracy: 0.9880 - val_loss: 0.1975 - val_accuracy: 0.9477
Epoch 10/10
126/126 [==============================] - 178s 1s/step - loss: 0.0333 - a
ccuracy: 0.9933 - val_loss: 0.1673 - val_accuracy: 0.9564
```


Figure 6.1: Training accuracy and loss in every Epoch

The following figure shows the accuracy and loss we acquired upon testing on the given dataset. While this model reached a high accuracy of 98.9%, we acquired an average validation loss, i.e., 54%.

```
print(val_loss)
print(val_acc)
```

```
36/36 [==============================] - 10s 279ms/step - loss: 0.0544 - accuracy: 0.9895
0.054440610110759735
0.9895470142364502
```

Figure 6.2: Validation Accuracy and Loss

In the figures below, the blue curves show for test/validation over each epoch and the red curves show for trains in every epoch.

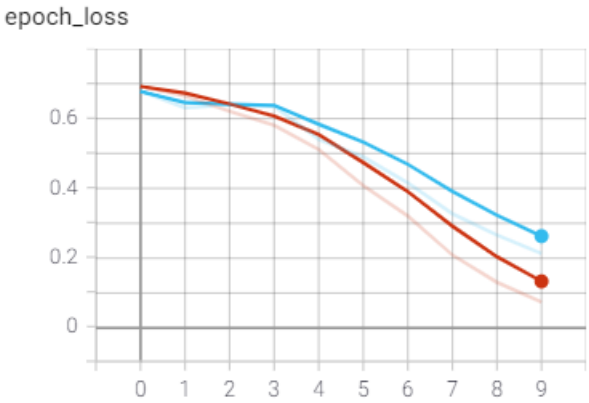


Figure 6.3: The internal Loss in every Epochs

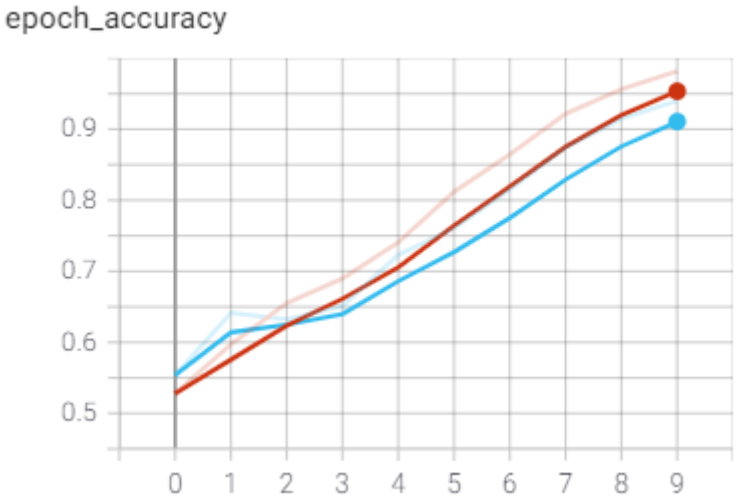


Figure 6.4: The internal Accuracy in every Epochs

From the above-mentioned figures, we can see the decrease of loss and increase of accuracy with every Epoch. The testing accuracy (blue) is slightly less than training accuracy (red) as epochs increase. This means there might be slight over-fitting. We kept 10 iterations while fitting the model. We have shown the internal values here. The performance hereof CNN gave us our best results. This is because CNN can learn appropriate features from an image at different levels, which is quite similar to the human brain.

Moreover, the CNN model shares weights. CNN takes less memory. The model requires less preprocessing. These are the reasons behind the efficient performance shown by CNN.

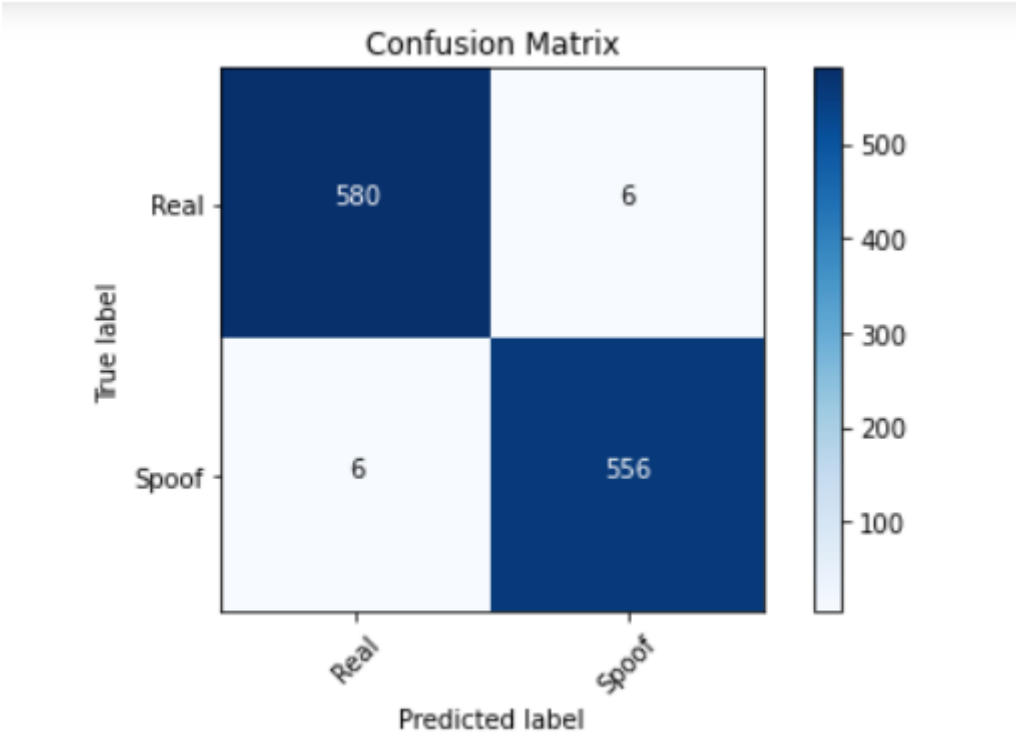


Figure 6.5: Confusion Matrix of CNN Model 256x3

A brief description of the confusion matrix is provided below:

- Correct Real Faces Predictions: 580
- Incorrect Real Faces Predictions: 6
- Correct Fake Faces Predictions: 556
- Incorrect Fake Faces Predictions: 6

6.2 VGG16

VGG16 has a 16-layer combination consisting of convolution and fully connected layers. We have trained our dataset and used it to create the VGG model. We have trained on 1468 samples and validated 164 samples. Here we got the accuracy of training 68.39% and accuracy of validation 63.41%. We chose this model in our research because it has considered being one of the finest vision model architectures, outperforming every other architecture by miles. The internal accuracy and loss graph are shown in figure 6.6 and figure 6.7.

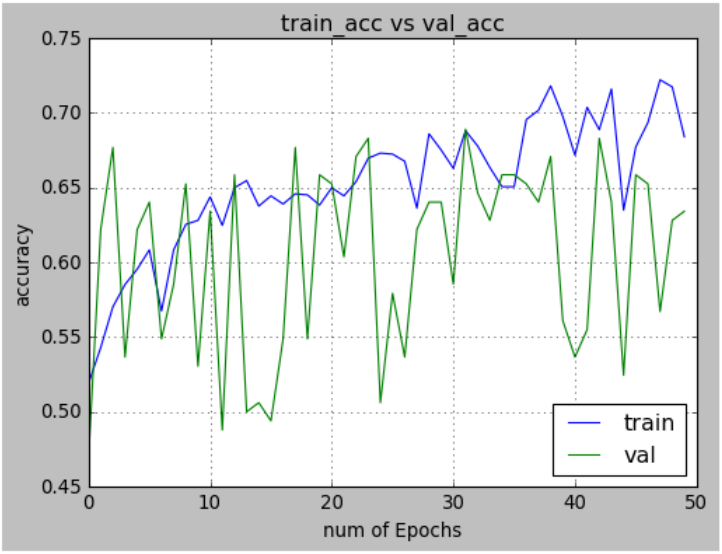


Figure 6.6: The internal Accuracy in every Epochs

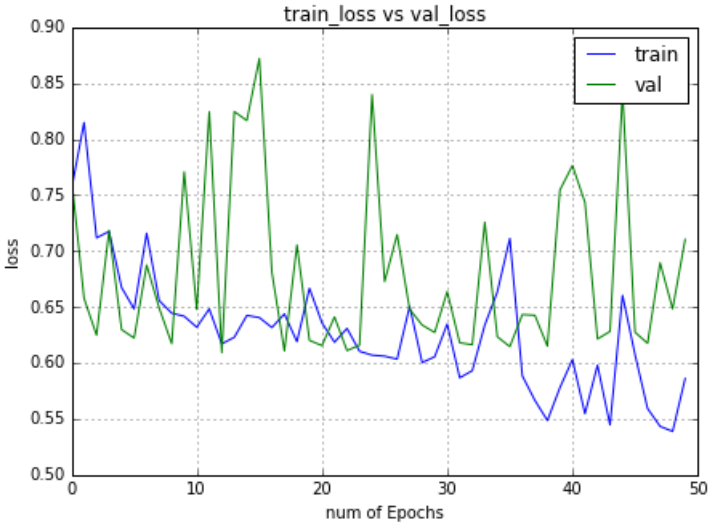


Figure 6.7: The internal loss in every Epochs

Here we got a training loss of 58.61% and a validation loss of 71.04%. The increase of training and testing accuracy and decrease of training and testing loss is noticed in the above figures with the increased epochs. The distance between training and the testing graph is too small. So we can say that the model we created was ideal. VGG16 has a good representation of the functions. It is an efficient architecture for benchmarking, taking a specific task into consideration.

Confusion Matrix: Actual Test Data- Real Faces[209], Fake Faces[200] Predicted Data- Real Faces[263], Fake Faces[146]

Figure 6.8 shows the Confusion matrix, without normalization

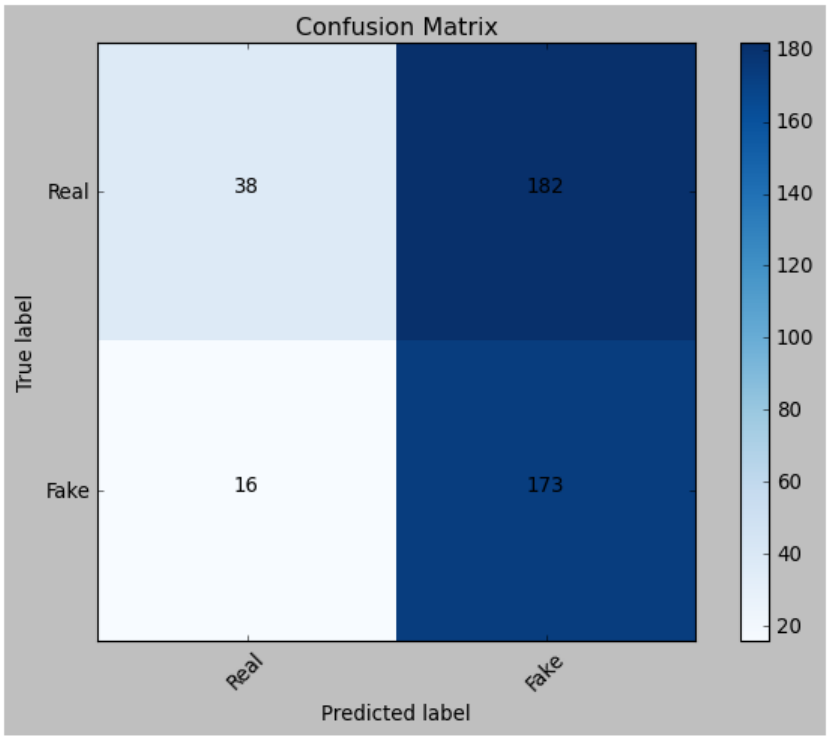


Figure 6.8: Confusion Matrix Predicted Table

A brief description of the confusion matrix is provided below:

- Correct Real Faces Predictions: 38
- Incorrect Real Faces Predictions: 182
- Correct Fake Faces Predictions: 173
- Incorrect Fake Faces Predictions: 16

6.2.1 Prediction vs. Actual

The figures below show whether the actual input matches the predicted result. Here, we have taken a fake face as an input and tested to see whether the actual input matches the predicted result. From Figures 6.9 and 6.10, we can observe that the actual input matches the predicted.

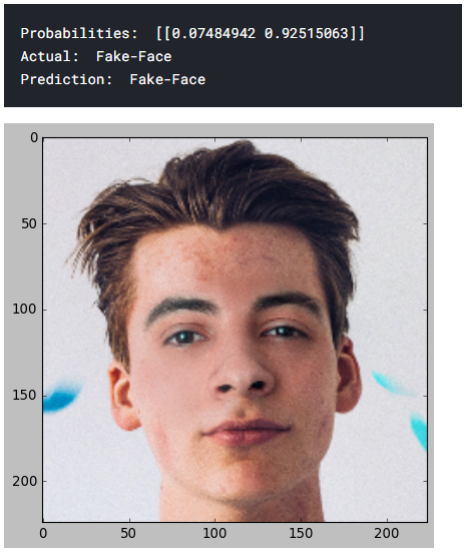


Figure 6.9: Actual vs Prediction Result

Figure 6.10: Actual vs Prediction Result

Here, we have taken real face as an input and tested to see whether the input and output match or not. From Figures 6.11 and 6.12, we can see that the predicted result matches the input image. Hence, the prediction was right.

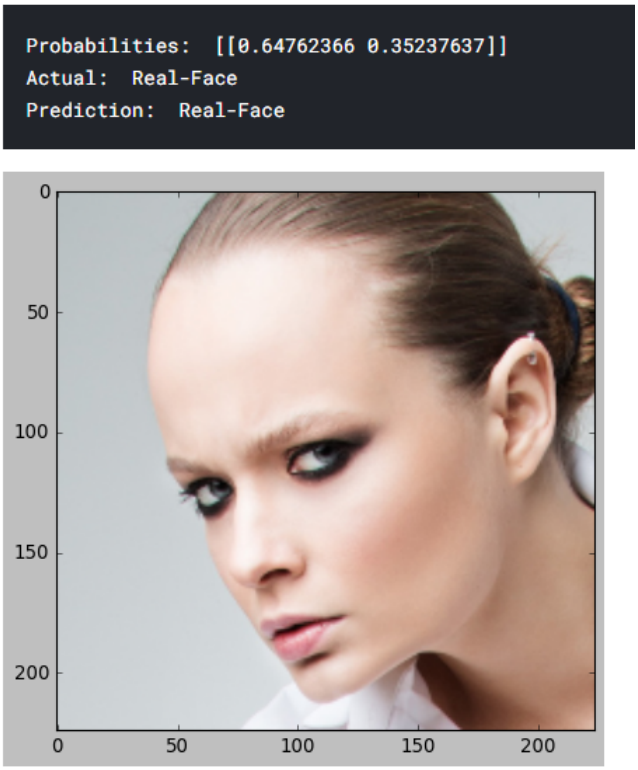


Figure 6.11: Actual vs Prediction Result

```
Probabilities:  [[0.7394622 0.2605378]]
Actual:  Real-Face
Prediction:  Real-Face
```

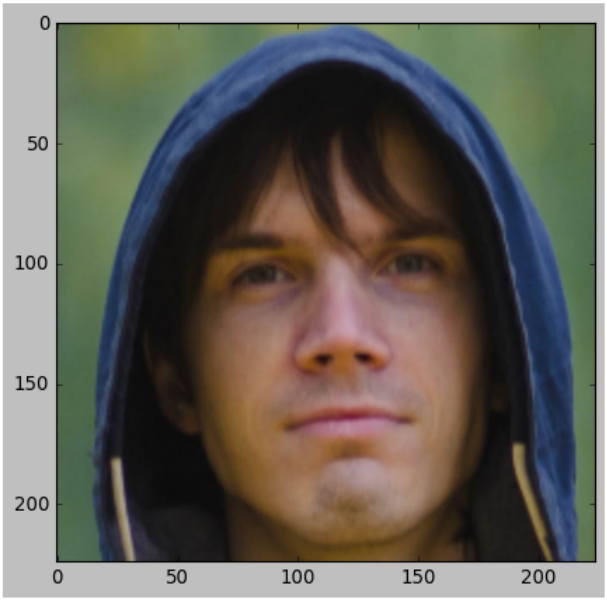


Figure 6.12: Actual vs Prediction Result

Here, we have taken a real face as input and test to see whether the predicted result matches the actual input. From Figures 6.13 and 6.14, we can observe that the actual input does not match the predicted result. Hence, the prediction was wrong.

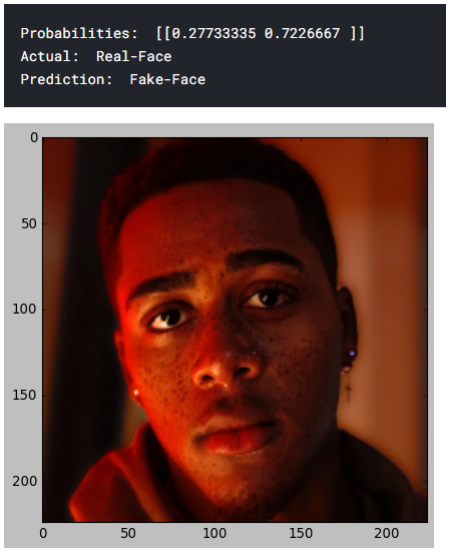


Figure 6.13: Actual vs Prediction Result

Figure 6.14: Actual vs Prediction Result

6.3 SVM Classifier

SVM Classifier showed comparatively poor performance than the rest. We chose this algorithm since it is known to work well with a clear margin of separation. The accuracy that we obtained from the classifier was 50%. Even though it was memory-efficient in our case, it required a long time for training. In the figures mentioned below, we can see that the decrease of validation loss over every epoch is evident, while the validation accuracy remains consistent over all the epochs.

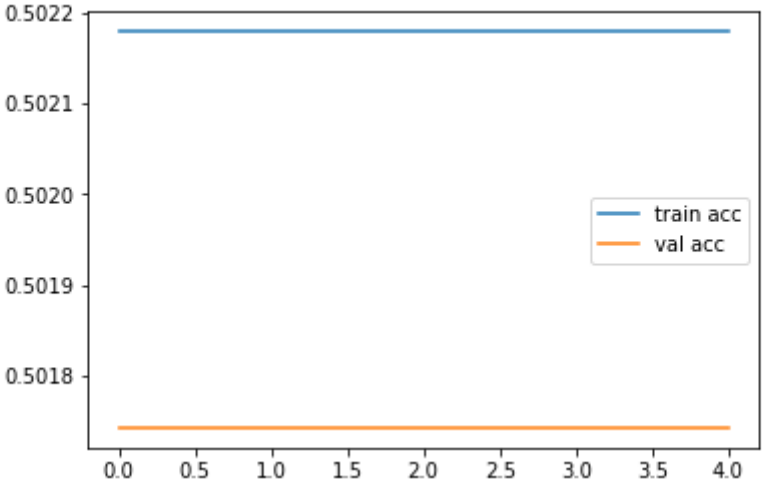


Figure 6.15: The internal accuracy in every Epoch

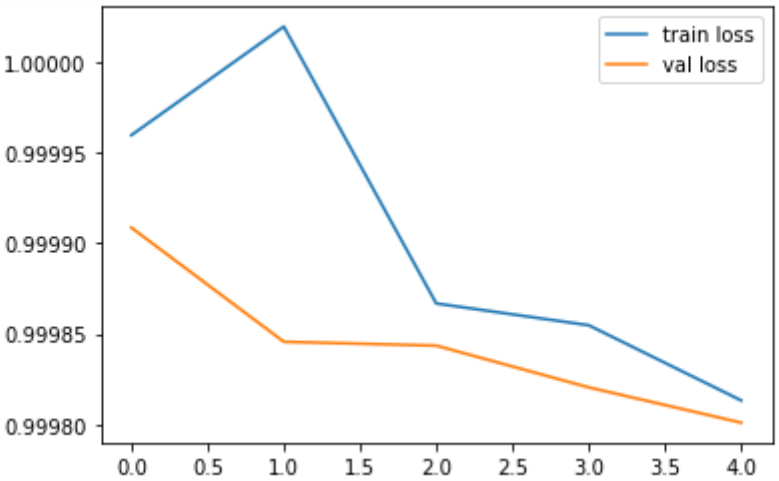


Figure 6.16: The internal loss in every Epoch

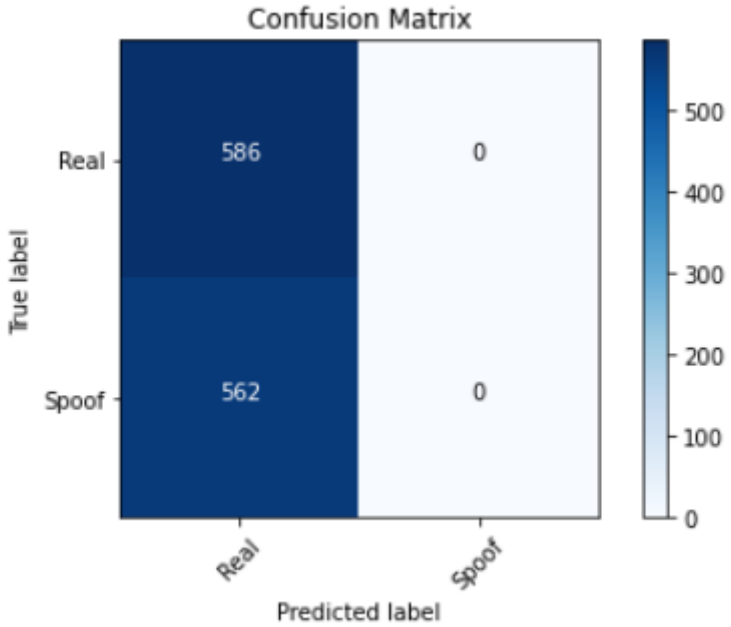


Figure 6.17: Confusion Matrix of SVM Classifier

A brief description of the confusion matrix is provided below: T

Chapter 7

Conclusion

7.1 Future Perspectives

In our research, we would like to incorporate future works that can help to address challenges that need to be resolved, as well as overcome limitations encountered thus far. We aspire to explore CNN architectures further, thereby including ResNet, VGG19 and Inception, alongside which we will implement widely-used algorithms such as: Logistic Regression, k-Nearest Neighbors, Decision Trees, etc. We intend to create a more realistic as well as difficult dataset that covers greater variations of face attack-types and lighting conditions so as to train our future models. Furthermore, our goal shall be to use additional evaluation metrics, namely Half Total Error Rate (HTER) and Area under Curve (AUC) for a broader analysis of performance achieved.

7.2 Conclusion

Face-Spoof Detection has been a very challenging endeavor in the past few years. Spoofing attacks persist in being a security challenge for face biometric systems, and there has been much effort in the field to find robust methods. Despite all the popularity it has gained in recent years, this task is yet to be fully controlled. We believe that residual training of deep neural networks has a much bigger prospect for face recognition tasks. However, in this paper, we have proposed to use CNN for image recognition because of its precise result. Upon using the CNN model, we have tried different data improvement techniques. Moreover, we have used VGG-16 in the proposed system for learning about feature classification. Since CNN works using layers embedded within by connecting with neurons, adding layers upon layers for this model has been crucial to analyze the performance accurately. As has been previously mentioned, the data set used for this model comprises both real and heavily edited forged facial images. Face-spoofing has been given great importance in this model and based on this data set, and this model can be said to be successful in terms of facial verification.

Reference

[1] A. Benlamoudi, D. Samai, A. Ouafi, S. E. Bekhouche, A. Taleb-Ahmed, and A. Hadid, "Face spoofing detection using local binary patterns and fisher score," in *2015 3rd International Conference on Control, Engineering Information Technology (CEIT)*, 2015, pp. 1–5. DOI: 10.1109/CEIT.2015.7233145.

[2] Boulkenafet, Zinelabidine, Komulainen, Jukka, Hadid, and Abdenour, "Face spoofing detection using colour texture analysis," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1818–1830, 2016. DOI: 10.1109/TIFS.2016.2555286.

[3] M. Asim, Z. Ming, and M. Y. Javed, "Cnn based spatio-temporal feature extraction for face anti-spoofing," *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, pp. 234–238, 2017.

[4] H. Mo, B. Chen, and W. Luo, "Fake faces identification via convolutional neural network," *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, 2018.

[5] M. Asim, Z. Ming, and M. Y. Javed, "Cnn based spatio-temporal feature extraction for face anti-spoofing," in *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, IEEE, 2017, pp. 234–238.

[6] Z. Boulkenafet, J. Komulainen, and A. Hadid, "Face spoofing detection using colour texture analysis," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1818–1830, 2016.

[7] H. Mo, B. Chen, and W. Luo, "Fake faces identification via convolutional neural network," in *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, 2018, pp. 43–47.

[8] A. Benlamoudi, D. Samai, A. Ouafi, S. E. Bekhouche, A. Taleb-Ahmed, and A. Hadid, "Face spoofing detection using local binary patterns and fisher score," in *2015 3rd International Conference on Control, Engineering & Information Technology (CEIT)*, IEEE, 2015, pp. 1–5.

[9] D. A. Pitaloka, A. Wulandari, T. Basaruddin, and D. Y. Liliana, "Enhancing cnn with preprocessing stage in automatic emotion recognition," *Procedia computer science*, vol. 116, pp. 523–529, 2017.

[10] S. Almabdy and L. Elrefaei, "Deep convolutional neural network-based approaches for face recognition," *Applied Sciences*, vol. 9, p. 4397, Oct. 2019. DOI: 10.3390/app9204397.

[11] J. C. Neves, R. Tolosana, R. Vera-Rodriguez, V. Lopes, H. Proença, and J. Fierrez, "Ganprintr: Improved fakes and evaluation of the state of the art in face manipulation detection," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 1038–1048, 2020.

[12] J. Yang, Z. Lei, and S. Li, "Learn convolutional neural network for face anti-spoofing," *ArXiv*, vol. abs/1408.5601, 2014.

[13] Rajeswaran, Shatish, and K. V, "Face-spoof detection system using convolutional neural network," 2019.

[14] Almabdy, Soad, Elrefaei, and Lamiaa, "Deep convolutional neural network-based approaches for face recognition," *Applied Sciences*, vol. 9, no. 20, 2019, ISSN: 2076-3417. DOI: 10.3390/app9204397. [Online]. Available: https://www.mdpi.com/2076-3417/9/20/4397.

[15] R. Tronci, D. Muntoni, G. Fadda, M. Pili, N. Sirena, G. Murgia, M. Ristori, S. Ricerche, and F. Roli, "Fusion of multiple clues for photo-attack detection in face recognition systems.," in *IJCB*, IEEE Computer Society, 2011, pp. 1–6, ISBN: 978-1-4577-1358-3. [Online]. Available: http://dblp.uni-trier.de/db/conf/icb/ijcb2011.html#TronciMFPSMRRR11.

[16] L. Souza, L. Oliveira, M. Pamplona, and J. Papa, "How far did we get in face spoofing detection?" *Engineering Applications of Artificial Intelligence*, vol. 72, pp. 368–381, Jun. 2018, ISSN: 0952-1976. DOI: 10.1016/j.engappai.2018.04.013. [Online]. Available: http://dx.doi.org/10.1016/j.engappai.2018.04.013.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[18] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks.," in *NIPS*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 2377–2385. [Online]. Available: http://dblp.uni-trier.de/db/conf/nips/nips2015.html#SrivastavaGS15.

[19] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger, *Deep networks with stochastic depth*, cite arxiv:1603.09382Comment: first two authors contributed equally, 2016. [Online]. Available: http://arxiv.org/abs/1603.09382.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "The journal of machine learning research," vol. 15, pp. 1943–1955, 2014.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 1943–1955, 2016. [Online]. Available: http://dblp.uni-trier.de/db/journals/pami/pami38.html#ZhangZHS16.

[23] R. B. Hadiprakoso, H. Setiawan, and Girinoto, "Face anti-spoofing using cnn classifier face liveness detection," in *2020 3rd International Conference on Information and Communications Technology (ICOIACT)*, 2020, pp. 143–147. DOI: 10.1109/ICOIACT50329.2020.9331977.

[24] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size.," in *ACPR*, IEEE, 2015, pp. 730–734, ISBN: 978-1-4799-6100-9. [Online]. Available: http://dblp.uni-trier.de/db/conf/acpr/acpr2015.html#LiuD15.

[25] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, cite arxiv:1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, *Going deeper with convolutions*, cite arxiv:1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842.