

ENHANCED DEEP LEARNING ARCHITECTURES
FOR FACE LIVENESS DETECTION FOR STATIC AND
VIDEO SEQUENCES

Ranjana Koshy

Under the Supervision of Dr. Ausif Mahmood

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

AND ENGINEERING

THE SCHOOL OF ENGINEERING

UNIVERSITY OF BRIDGEPORT

CONNECTICUT

December, 2020


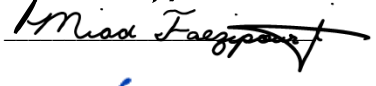
ENHANCED DEEP LEARNING ARCHITECTURES FOR FACE LIVENESS DETECTION FOR STATIC AND VIDEO SEQUENCES

Ranjana Koshy

Under the Supervision of Dr. Ausif Mahmood

Approvals

Committee Members

Name	Signature	Date
Dr. Ausif Mahmood		12-29-2020
Dr. Abdelshakour Abuzneid	AB ABuzneid	01/05/2021
Dr. Julius Dichter		1.5 2021
Dr. Miad Faezipour		1/05/2021
Dr. Syed Rizvi	Syed Rizvi	1/10/2021

Ph.D. Program Coordinator

Dr. Khaled M. Elleithy		1/23/2021
------------------------	--	-----------

Chairman, Computer Science and Engineering Department

Dr. Ausif Mahmood		12-29-2020
-------------------	--	------------

Dean, School of Engineering

Dr. Khaled M. Elleithy		1/23/2021
------------------------	--	-----------

ENHANCED DEEP LEARNING ARCHITECTURES FOR
FACE LIVENESS DETECTION FOR STATIC AND VIDEO
SEQUENCES

© Copyright by Ranjana Koshy 2020

ENHANCED DEEP LEARNING ARCHITECTURES FOR FACE LIVENESS DETECTION FOR STATIC AND VIDEO SEQUENCES

ABSTRACT

The major contribution of this research is the development of deep architectures for face liveness detection on a static image as well as video sequences that use a combination of texture analysis and deep Convolutional Neural Network (CNN) to classify the captured image or video as real or fake. Face recognition is a popular and efficient form of biometric authentication used in many software applications. One drawback of this technique is that, it is prone to face spoofing attacks, where an impostor can gain access to the system by presenting a photograph or recorded video of a valid user to the sensor. Thus, face liveness detection is a critical preprocessing step in face recognition authentication systems. The first part of our research was on face liveness detection on a static image, where we applied nonlinear diffusion based on an additive operator splitting scheme and a tri-diagonal matrix block-solver algorithm to the image, which enhances the edges and surface texture in the real image. The diffused image was then fed to a deep CNN to identify the complex and deep features for classification. We obtained high accuracy on the NUAA Photograph Impostor dataset

using one of our enhanced architectures. In the second part of our research, we developed an end-to-end real-time solution for face liveness detection on static images, where instead of using a separate preprocessing step for diffusing the images, we used a combined architecture where the diffusion process and CNN were implemented in a single step. This integrated approach gave promising results with two different architectures, on the Replay-Attack and Replay-Mobile datasets. We also developed a novel deep architecture for face liveness detection on video frames that uses the diffusion of images followed by a deep CNN and Long Short-Term Memory (LSTM) to classify the video sequence as real or fake. Performance evaluation of our architecture on the Replay-Attack and Replay-Mobile datasets gave very competitive results. We performed liveness detection on video sequences using diffusion and the Two-Stream Inflated 3D ConvNet (I3D) architecture, and our experiments on the Replay-Attack and Replay-Mobile datasets gave very good results.

ACKNOWLEDGEMENTS

My thanks are wholly devoted to God who has helped me all the way to complete this work successfully. I owe a debt of gratitude to my family for understanding and encouragement.

I am honored that my work has been supervised by Dr. Ausif Mahmood, and I would like to express my deepest appreciation for his guidance, tremendous help, and support.

I would like to thank my dissertation committee of Dr. Abdelshakour Abuzneid, Dr. Julius Dichter, Dr. Miad Faezipour, Dr. Syed Rizvi, and Dr. Khaled M. Elleithy for their time and valuable comments that helped me increase the quality of this dissertation.

TABLE OF CONTENTS

ABSTRACT.....	iv
ACKNOWLEDGEMENTS.....	vi
TABLE OF CONTENTS.....	vii
LIST OF TABLES.....	x
LIST OF FIGURES.....	xiv
CHAPTER 1: INTRODUCTION.....	1
1.1 Research Problem and Scope.....	1
1.2 Motivation behind the proposed research.....	1
1.3 Potential contributions of the proposed research.....	3
CHAPTER 2: LITERATURE SURVEY.....	6
2.1 Introduction.....	6
2.2 Static techniques proposed in the literature.....	8
2.3 Dynamic techniques proposed in the literature.....	19
2.4 Conclusion.....	25
CHAPTER 3: FACE LIVENESS DETECTION.....	28
3.1 Nonlinear Diffusion.....	28
3.2 Convolutional Neural Networks.....	30
3.3 Long Short-Term Memory (LSTM).....	31
CHAPTER 4: FACE LIVENESS DETECTION ON A SINGLE STATIC IMAGE.....	33
4.1 CNN-5.....	33
4.2 ResNet50.....	35

4.3 Inception v4	37
CHAPTER 5: END-TO-END REAL-TIME FACE LIVENESS DETECTION USING ANISOTROPIC DIFFUSION AND CONVOLUTIONAL NEURAL NETWORK	41
5.1 Specialized Convolutional Neural Network (SCNN)	42
5.2 Inception v4	43
CHAPTER 6: FACE LIVENESS DETECTION ON VIDEO SEQUENCES.....	45
6.1 CNN-LSTM	45
6.2 Two-Stream Inflated 3D ConvNet (I3D) architecture	47
CHAPTER 7: IMPLEMENTATION AND RESULTS.....	49
7.1 Face Liveness Detection on a static image	49
7.1.1 CNN-5	51
7.1.2 ResNet50.....	52
7.1.3 Inception v4	53
7.1.4 Comparison of the three architectures	55
7.1.5 Comparison with state-of-the-art methods.....	59
7.2 End-to-end real-time face liveness detection using anisotropic diffusion and convolutional neural network	60
7.2.1 Specialized Convolutional Neural Network (SCNN)	64
7.2.2 Inception v4	66
7.2.2.1 Results obtained with the Replay-Attack dataset.....	66
7.2.2.2 Results obtained with the Replay-Mobile dataset.....	68
7.2.3 Comparison with state-of-the-art methods.....	70
7.3 Face Liveness Detection on video sequences	72
7.3.1 CNN-LSTM	75
7.3.1.1 Results obtained with the Replay-Attack dataset.....	75
7.3.1.2 Results obtained with the Replay-Mobile dataset.....	79
7.3.1.3 Comparison of results with and without diffusion.....	82

7.3.1.4 Comparison with state-of-the-art methods.....	83
7.3.2 Two-Stream Inflated 3D ConvNet (I3D)	86
CHAPTER 8: CONCLUSION	89
REFERENCES	91

LIST OF TABLES

Table 2.1	Performance comparison on NUAA dataset [12]	15
Table 7.1	NUAA dataset	49
Table 7.2	Test results obtained with the CNN-5 architecture	51
Table 7.3	Test results obtained with the ResNet50 architecture	52
Table 7.4	Test results obtained with the Inception v4 architecture	53
Table 7.5	Evaluation with various numbers of Inception-A, Inception-B, Inception-C blocks (param. alpha=25)	53
Table 7.6	Evaluation using only Inception-stem, Inception-A blocks, Reduction-A block (param. alpha=25)	54
Table 7.7	Validation accuracy (%) obtained after each epoch (param. alpha=15)	55
Table 7.8	Comparison of the three architectures	56
Table 7.9	Accuracies obtained for various values of learning rates	57
Table 7.10	Accuracies obtained with each set of diffused images (the epochs are shown in parentheses)	57
Table 7.11	Comparison of our proposed methods with other state-of-the-art methods on the NUAA dataset	59
Table 7.12	Replay-Attack dataset	61

Table 7.13	Replay-Mobile dataset	62
Table 7.14	Highest validation accuracy (best model) obtained during validation, and the test results obtained by evaluating the best model on the test set, for the SCNN	65
Table 7.15	Best model obtained for each alpha while validating on the development set during training, with the Replay-Attack dataset, in Inception v4	67
Table 7.16	Test results obtained with the Replay-Attack dataset by evaluating the best model obtained for each alpha (in Table 7.15) on the test set, in Inception v4	67
Table 7.17	Best model obtained for each alpha while validating on the development set during training, with the Replay-Mobile dataset, in Inception v4	69
Table 7.18	Test results obtained with the Replay-Mobile dataset by evaluating the best model obtained for each alpha (in Table 7.17) on the test set, in Inception v4	69
Table 7.19	Comparison with state-of-the-art methods on the Replay-Attack dataset	70
Table 7.20	Comparison with state-of-the-art methods on the Replay-Mobile dataset	71
Table 7.21	Highest validation accuracy (best model) obtained for different values of learning rate by validating on diffused	76

validation images created with each alpha, during training, with Replay-Attack dataset, in CNN-LSTM (the best model for each alpha, and without diffusion, is highlighted in bold)

Table 7.22	Test accuracies and HTER obtained by evaluating the best model of each alpha (highlighted in Table 7.21) on the test set (the highest test accuracy and lowest HTER are indicated in bold)	76
Table 7.23	Highest validation accuracy (best model) obtained for different values of learning rate by validating on diffused test images created with each alpha, during training, with Replay-attack dataset, in CNN-LSTM (the best model for each alpha, and without diffusion, is highlighted in bold)	78
Table 7.24	HTER obtained by evaluating the best model of each alpha (in Table 7.23) on the test set (the overall lowest HTER is highlighted in bold)	78
Table 7.25	Highest validation accuracy (best model) obtained for different values of learning rate by validating on diffused validation images created with each alpha, during training, with Replay-Mobile dataset, in CNN-LSTM (the best model for each alpha, and without diffusion, is	79

	highlighted in bold)	
Table 7.26	Test accuracies and HTER obtained by evaluating the best model of each alpha (highlighted in Table 7.25) on the test set (the highest test accuracy and lowest HTER are indicated in bold)	79
Table 7.27	Highest validation accuracy (best model) obtained for different values of learning rate by validating on diffused test images created with each alpha, during training, with Replay-Mobile dataset, in CNN-LSTM (the best model for each alpha, and without diffusion, is highlighted in bold)	81
Table 7.28	HTER obtained by evaluating the best model of each alpha (in Table 7.27) on the test set (the overall lowest HTER is highlighted in bold)	81
Table 7.29	Comparison with state-of-the-art methods on the Replay-Attack dataset	84
Table 7.30	Comparison with state-of-the-art methods on the Replay-Mobile dataset	85
Table 7.31	Test accuracy and HTER obtained with the Replay-Attack dataset	87
Table 7.32	Test accuracy and HTER obtained with the Replay-Mobile dataset	87

LIST OF FIGURES

Figure 2.1	ELBP feature maps of a live and fake face [6]	9
Figure 2.2	Face liveness detection system using the DLTP feature descriptor [12]	13
Figure 2.3	The proposed approach by Maatta et al. [13]	16
Figure 2.4	Nonlinear diffusion followed by CNN applied to the NUAA dataset [18]	18
Figure 2.5	Outline of the proposed approach [25]	22
Figure 3.1	LSTM memory cell (adapted from [59])	32
Figure 4.1	Convolutional Neural Network (CNN-5) architecture	34
Figure 4.2	Basic residual block (adapted from [67])	36
Figure 4.3	ResNet50 architecture	36
Figure 4.4	Basic inception module (adapted from [72])	39
Figure 4.5	Inception v4 network (adapted from [71])	39
Figure 5.1	End-to-end Specialized Convolutional Neural Network (SCNN) architecture (alpha is learned by the network)	43
Figure 5.2	End-to-end architecture using the Inception v4 network	44
Figure 6.1	Convolutional Neural Network and Long Short-Term	46

	Memory (CNN-LSTM)	
Figure 6.2	Two-Stream Inflated 3D ConvNet (I3D) architecture (adapted from [77])	47
Figure 7.1	Samples from the NUAA database. The first row in (a) shows non-diffused real images in the database, and the first row in (b) shows non-diffused fake images. Each row below shows the corresponding diffused images created with value of param. alpha set to 15, 25, 50, 75, and 100 respectively	50
Figure 7.2	Plot showing epoch vs validation accuracy for Inception v4	55
Figure 7.3	Plot showing epochs vs accuracy for each architecture	56
Figure 7.4	Plot showing param. alpha vs accuracy for each architecture	58
Figure 7.5	Performance comparison on the NUAA dataset	60
Figure 7.6	Sample images from the datasets and their corresponding diffused versions. (a) Images from the Replay-Attack dataset. (b) Images from the Replay-Mobile dataset. The images in the first row of both (a) and (b) are real, and the images in the second row of both (a) and (b) are fake	63
Figure 7.7	Plot showing parameter alpha vs. test accuracy (Table 7.16)	67
Figure 7.8	Plot showing parameter alpha vs. HTER (Table 7.16)	68
Figure 7.9	Plot showing parameter alpha vs. test accuracy (Table 7.18)	69
Figure 7.10	Plot showing parameter alpha vs. HTER (Table 7.18)	70
Figure 7.11	Performance comparison (% Test accuracy) of the end-to-	71

	end networks on the Replay-Attack dataset (Table 7.19)	
Figure 7.12	Performance comparison (% HTER) of the end-to-end networks on the Replay-Attack dataset (Table 7.19)	71
Figure 7.13	Performance comparison (% Test accuracy) of the end-to-end networks on the Replay-Mobile dataset (Table 7.20)	72
Figure 7.14	Performance comparison (% HTER) of the end-to-end networks on the Replay-Mobile dataset (Table 7.20)	72
Figure 7.15	Sample images from the datasets and their corresponding diffused versions. (a) Replay-Attack dataset. (b) Replay-Mobile dataset. The images in the first two rows of both (a) and (b) are real, and the images in the third and fourth rows of both (a) and (b) are fake	74
Figure 7.16	Plot showing parameter alpha vs. test accuracy (Table 7.22)	76
Figure 7.17	Plot showing parameter alpha vs. HTER (Table 7.22)	77
Figure 7.18	Plot showing parameter alpha vs. HTER (Table 7.24)	78
Figure 7.19	Plot showing parameter alpha vs. test accuracy (Table 7.26)	80
Figure 7.20	Plot showing parameter alpha vs. HTER (Table 7.26)	80
Figure 7.21	Plot showing parameter alpha vs. HTER (Table 7.28)	82
Figure 7.22	Test accuracy (%) obtained with and without diffusion	82
Figure 7.23	HTER (%) obtained with and without diffusion	83
Figure 7.24	Performance comparison (HTER) on the Replay-Attack dataset (Table 7.29)	84

Figure 7.25	Performance comparison (Test Accuracy) on the Replay-Attack dataset (Table 7.29)	85
Figure 7.26	Performance comparison (HTER) on the Replay-Mobile dataset (Table 7.30)	86
Figure 7.27	Comparison of test accuracies obtained with I3D and CNN-LSTM	87
Figure 7.28	Comparison of HTER obtained with I3D and CNN-LSTM	88

CHAPTER 1: INTRODUCTION

1.1 Research Problem and Scope

Face recognition is a popular and efficient form of biometric authentication that is extensively used for identity management and secure access control in many web and mobile-related software applications. However, it has the key disadvantage of being easily spoofed, and the security system might not be able to distinguish between a real person and his/her photograph. An impostor can gain access to the system as a valid user by presenting a copy of the image, which may be a printed photograph or a displayed image on a smartphone or tablet, to the camera. This drawback of face recognition authentication makes it necessary to determine the liveness of the face before granting authentication. Therefore, prior to face recognition authentication, face liveness detection is important in order to detect whether the captured image is live or fake. To address the face spoofing attacks, researchers have proposed different methods for face liveness detection on static images and video sequences such as motion analysis, texture analysis, quality of captured images, etc. and recent research has focused on using deep CNN architectures for face liveness detection.

1.2 Motivation behind the proposed research

Due to the growing number of web and mobile related applications and their use in digital computing devices, ensuring secure access to these applications is essential. Compared to the usage of traditional credentials as an authentication mechanism, biometric authentication is a much more secure form of access control.

Traditional authentication methods like passwords and tokens only provide proof of knowledge and proof of ownership, whereas biometric authentication ensures that users are whom they claim to be. Passwords are prone to attacks such as brute-force attacks, dictionary attacks, and man-in-the-middle attack, and therefore do not provide a strong identity check. Hardware tokens can be stolen, and an attacker can try to gain access to the software that sends or receives a software token. A malicious user can authenticate and pose as the legitimate user. Therefore, passwords and tokens do not provide a strong identity check, whereas biometric authentication confirms a user's identity.

Biometric authentication determines the individual's identity based on biological characteristics that are unique to the individual, making it more secure than the traditional way of authentication. Face recognition, which is a popular and efficient form of biometric authentication, has the added advantage of being more convenient to deploy and also being a non-intrusive form of interaction, compared to other biometric authentication traits.

Web and mobile applications have gained widespread popularity, and therefore a security process such as biometric authentication is necessary for identification and access control. Accessing data and applications irrespective of location and time and at

low cost is an important benefit of mobile cloud computing [1]. Protecting these data and applications from unauthorized access is therefore a security issue in systems that require remote identity of subjects. Biometric authentication applied to cloud computing ensures that the rendered services are only accessible to authorized users, thus ensuring cyber security.

The technological evolution of biometric authentication has led to its deployment in diverse environments [2], and though biometric authentication methods are more secure than traditional methods, they are still prone to external attacks where illegitimate users can spoof the biometric system by presenting forged versions of the biometric trait to the sensor. Face recognition, which is a preferred form of biometric authentication due to its convenience and non-intrusive form of interaction, is therefore prone to spoofing attacks where an impostor can spoof the system by presenting a photograph or recorded video of the valid user to the sensor. Therefore, developing an anti-spoofing technique such as face liveness detection is essential to ensure secure access in face recognition authentication systems.

1.3 Potential contributions of the proposed research

We developed deep CNN architectures for liveness detection on static images to address face spoofing attacks in biometric authentication systems that use face recognition for authentication, and we achieved an effective solution to the face liveness detection problem. We first applied nonlinear diffusion based on an additive operator splitting scheme and a block-solver called tri-diagonal matrix algorithm, to the captured images. This produced diffused images, with edge information and surface texture of real

images more pronounced than fake ones. These diffused images were then fed to a CNN to extract the complex and deep features, for classification. We conducted numerous experiments with three different deep architectures, which included a 5-layer CNN, a residual network of 50 layers, and the inception network version 4. We evaluated the performance of each of these architectures on the NUAA dataset to determine an effective solution for liveness detection. We experimented with various values of the parameter alpha that defines smoothness of diffusion, and observed that for a lower value of this parameter, better accuracy is obtained since diffusing a captured image with higher values of this smoothness parameter blurs out important information from the image. Our implementation with the deep CNN architecture Inception v4, gave high accuracy with diffused images created with a smoothness parameter of 15, which has not been reported by any previous approach for face liveness detection on the NUAA dataset.

We also developed an end-to-end solution for face liveness detection in real-time on static images, by integrating the diffusion process as well as the face liveness classification using two deep CNN architectures, the Specialized Convolutional Neural Network (SCNN) and the Inception v4 into a single application. Rather than using a two-step process of first applying nonlinear diffusion, and then using a deep network for final liveness decision, the end-to-end architecture performs the diffusion and the liveness detection in a single step. Our experiments with the two deep architectures on the Replay-Attack and Replay-Mobile datasets gave promising results.

We developed a deep CNN-LSTM architecture for face liveness detection on video sequences, by using a combination of texture analysis and an architecture

comprising of a Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) to classify the video sequence as real or fake. As was done for static images, we first applied nonlinear diffusion based on an additive operator splitting scheme and a tri-diagonal matrix block-solver algorithm to the individual frames in the sequence, which enhances the edges and surface texture in the real image. We then fed the diffused image to a deep CNN to identify the complex and deep spatial features in the frames. This was then fed to the LSTM which detects temporal features in the sequence for sequence prediction and classification. Our experiments on the Replay-Attack dataset and Replay-Mobile dataset gave very competitive results. We further performed experiments using a combination of diffusion and the Two-Stream Inflated 3D ConvNet (I3D) architecture, for face liveness detection on video sequences, and our experiments on the Replay-Attack and Replay-Mobile datasets gave very good results.

CHAPTER 2: LITERATURE SURVEY

2.1 Introduction

Face recognition has become a popular and efficient means of biometric authentication due to advancements in the field of computer vision and image processing. However, with the widespread adoption of biometric technology, spoofing techniques have also increased, with the biometric system being forged to bypass the verification system [3]. Therefore, despite the advantage of face recognition as a convenient and non-intrusive form of access, it is still vulnerable to spoofing attacks where an illegitimate user can spoof the system by presenting a photograph or a recorded video of a valid user to the sensor. Hence, researchers have proposed various techniques to counteract spoofing attacks by first detecting the liveness of the face, as a precursor to face recognition. Therefore, only if the captured image is classified as a real face by the face liveness detection method, will a recognized face be granted authentication to the respective application.

There are two main categories of anti-spoofing methods, which are the hardware-based methods and software-based methods. Hardware-based methods require extra hardware to measure the required information, in addition to the camera of the face recognition system. The setup cost of the additional hardware is high, and moreover,

some methods require user cooperation such as speaking few words or rotating the head, which increases the detection time [4]. Also known as sensor-level methods, these techniques are integrated in the biometric sensor, and they generally measure the intrinsic properties, or the involuntary signals, or the responses to external stimuli of a living body [2]. Software based methods are more widely used, and liveness detection is performed based on the information contained in the captured images such as texture information, structure information, liveness sign and image quality, without using any additional hardware [4]. While hardware-based techniques rely on devices to detect a particular biometric trait, software-based techniques which are also known as feature-level methods extract the features of the biometric trait.

The software-based techniques are further categorized into static-based techniques and dynamic-based techniques. Static techniques are based on the analysis of a 2D static image, whereas dynamic techniques are based on the analysis of a sequence of input frames. A variety of software-based methods have been proposed to address face spoofing attacks to determine the liveness of a captured image, based on texture analysis, motion analysis, feature fusion, image quality, etc. Texture analysis, which is used by many static methods, is based on the fact that images from print have texture features that are different from real faces, and therefore analyzes the texture properties of the face image. Motion analysis aims at detecting the natural responses of the face, such as eye blinking, lip movement, and head rotation [5]. Feature fusion attempts to combine static and dynamic methods, by merging different texture features, or combining motion analysis with texture features [6]. Image quality-based approaches are based on the fact

that fake faces tend to be more seriously distorted by the imaging system, thereby yielding a lower quality image under the same capturing condition [5].

This dissertation focuses on the software-based static and dynamic techniques. The static technique analyzes a 2D facial image, and extracts important features that can be used to classify an image as real or fake. The dynamic technique analyzes a sequence of video frames containing 2D facial images, and extracts important features that can be used to classify the sequence as real or fake. Nonlinear diffusion is first applied to the captured images or video frames, and the diffused images are fed to CNN architectures, and further to an LSTM in the dynamic technique.

2.2 Static techniques proposed in the literature

Many methods have been proposed to address face spoofing attacks to determine the liveness of a captured image, by analyzing the 2D static image. Static techniques are a non-intrusive form of interaction which is convenient to users. Liu et al. [6] presented the extraction of an Enhanced Local Binary Pattern (ELBP) of a face map that served as the classification features. To better capture the difference between live and fake faces, they derived the enhanced facial representation using the ELBP operators. The ELBP codes were calculated based on the circular neighborhood and by applying bilinear interpolation at nonlinear pixel coordinates. As shown in Figure 2.1, the client ELBP feature map contains more details than the impostor feature map.

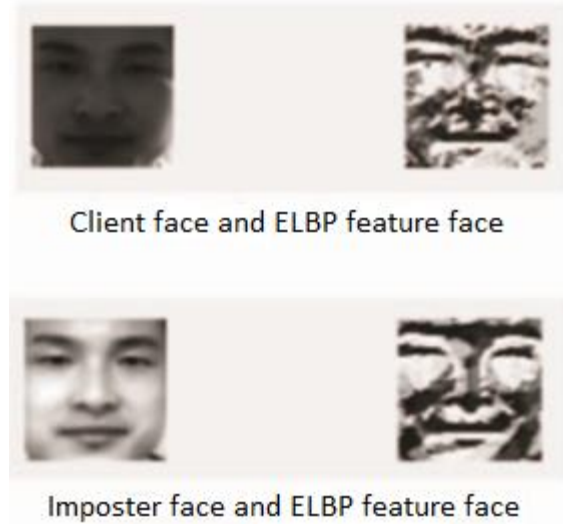


Figure 2.1. ELBP feature maps of a live and fake face [6].

These features, when fed to a Support Vector Machine (SVM) classifier, identifies whether the face map is real or fake. They achieved over 95% correct recognition rate on the NUAA dataset. Das et al. [7] described an approach based on frequency and texture analyses for differentiating between live and fake faces. The frequency analysis was performed by selecting four random images from an input image sequence, and then constructing a subset. The images were then transformed into the frequency domain using 2D discrete Fourier transform and the frequency descriptor was calculated to determine the temporal changes in the face. The texture analysis was done using Local Binary Patterns (LBP) which is a grayscale invariant texture measure. The LBP assigns a code for each pixel and its neighbors, by considering the relative intensity, as shown in equation 2.1,

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) 2^p, \quad s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (2.1)$$

where P corresponds to the number of neighboring pixels, R is the radius of the

corresponding circle, g_p corresponds to the grayscale value of the p equally spaced pixels on the circle of radius R , g_c corresponds to the grayscale value of the center pixel, and $s(x)$ denotes the threshold function of x . The resulting feature vector was fed to an SVM classifier with a radial basis function kernel for classification.

In the proposed method by Kim et al. [5], the key idea is that the difference in surface properties between live and fake faces can be efficiently estimated by using diffusion speed. The illumination energies on a live face tend to move faster because of their non-uniformity, whereas those on a 2D surface like a fake face are evenly distributed, and hence diffuse slowly. Therefore, the diffusion speed provides useful information that can be used to discriminate between live and fake faces. They computed the diffusion speed by utilizing the total variation flow and extracted anti-spoofing features based on the local patterns of diffusion speeds. These features were then fed to a linear SVM classifier to determine the liveness of the facial image. They first conducted nonlinear diffusion based on the Additive Operator Splitting (AOS) scheme. The diffusion speed at each pixel position (x, y) , which represents the amount of difference on the log space between the diffused image and the original image, is given by

$$s(x, y) = |\log(u^0(x, y) + 1) - \log(u^L(x, y) + 1)| \quad (2.2)$$

where L denotes the total number of iterations. The local speed patterns for efficiently capturing even small differences between live and fake faces is defined as,

$$f_{LSP}(x, y) = \sum_{1 \leq i \leq n} 2^{i-1} LSP^i(x, y), \quad (2.3)$$

$$LSP^i(x, y) = \begin{cases} 1, & \text{if } s(x, y) > s(x_i, y_i) \\ 0, & \text{otherwise} \end{cases}, \quad (2.4)$$

where n is the number of sampling pixels in the neighborhood of 3×3 pixels, (x_i, y_i) denotes the position of the neighborhood pixels centered at (x, y) . Based on the $f_{LSP}(x, y)$ values of an image block, the histogram features were built, and by concatenating the LSP histograms of each block, the face image was represented as a single vector which was then fed into the linear SVM classifier for classification. They reported a detection accuracy of 98.45% on the NUAA dataset.

Yeh et al. [8] proposed an algorithm that relies on the property of digital focus with various depths of field (DOFs). Preprocessing was done by analyzing the nose and bottom right part of the cheek. The level of blurriness was shown to be different in live and fake images due to the effect of the depth of field. The camera was set to focus on the nose, and for a live image, the captured nose will be sharp and clear, and the bottom right part of the cheek will be blurred due to the effect of the DOF. For a 2D printed photo, there will not be any difference between the nose and cheek parts because there is no DOF effect. Classification was then done using the k-nearest-neighbor algorithm.

A method that uses a flash against 2D spoofing attack was described by Chan et al. [4]. In this method, where both software and hardware techniques were used, two images per person were captured, one using flash and the other without using flash. Therefore, in addition to software techniques which consider textural and structure information, an additional device, flash, was used to enhance the performance. The textural information from the face was measured using a descriptor based on uniform LBP, and three other descriptors were used to capture the structural information of the face using the standard deviation and mean of the grayscale difference between the two

captured images of a person. Classification was based on the difference between the images with and without flash measured by the four descriptors.

Luan et al. [9] proposed a method where three types of features were extracted, which are specular reflection ratio, hue channel distribution features, and blurriness. The specular reflection indicates the geometry of objects, and the geometry of printed photos being flat, the specular reflection will be distributed on a plane. On the other hand, the real faces being typically 3D, the specular reflection will give rise to an unsmooth and uneven distribution. The recaptured imaging of physical media can bring equipment related hue differences, and therefore the hue channel distribution features provide discrimination between live and fake faces. Additionally, since recapturing of images causes blurriness, the ratio of feature changes of gray level co-occurrence matrix was used as blurriness features for discriminating between live and fake images. Classification of the images was done based on these three features using an SVM.

Tan et al. [10] presented a method where the anti-spoofing task was formulated as a binary classification problem, by considering the Lambertian reflectance to discriminate between live and fake faces. They analyzed the 2D Fourier spectra by Difference of Gaussian (DoG) filtering to remove noise and extended the sparse logistic regression classifier nonlinearly and spatially for classification. The DoG-based method is based on the idea that the fake image passes through the camera system twice, which makes it more distorted and of lower quality than a real image under the same imaging conditions. They used two Gaussian filters with different standard deviations, where the filter with lower value of standard deviation was used for keeping sufficient detail without noise,

and the filter with higher value of standard deviation was used to filter out misleading low spatial frequency information. The authors reported promising results on the NUAA dataset.

In [11], Zhang et al. used multiple Difference of Gaussian (DoG) filters to extract the high frequency information from the face image. By properly setting the Gaussian σ , the low frequency information and noise were removed. Four DoG filters were used with values of (0.5, 1), (1, 1.5), (1.5, 2), and (1, 2) for the inner Gaussian variance and outer Gaussian variance (σ_1, σ_2). The concatenated filtered images were then fed to an SVM classifier for classification.

The authors of [12] introduced a texture descriptor known as the Dynamic Local Ternary Pattern (DLTP), where the textural properties of facial skin were explored using a dynamic threshold setting, and the support vector machine with a linear kernel was used for classification. The block diagram of their recommended face liveness detection system using the DLTP feature descriptor is shown in Figure 2.2 below.

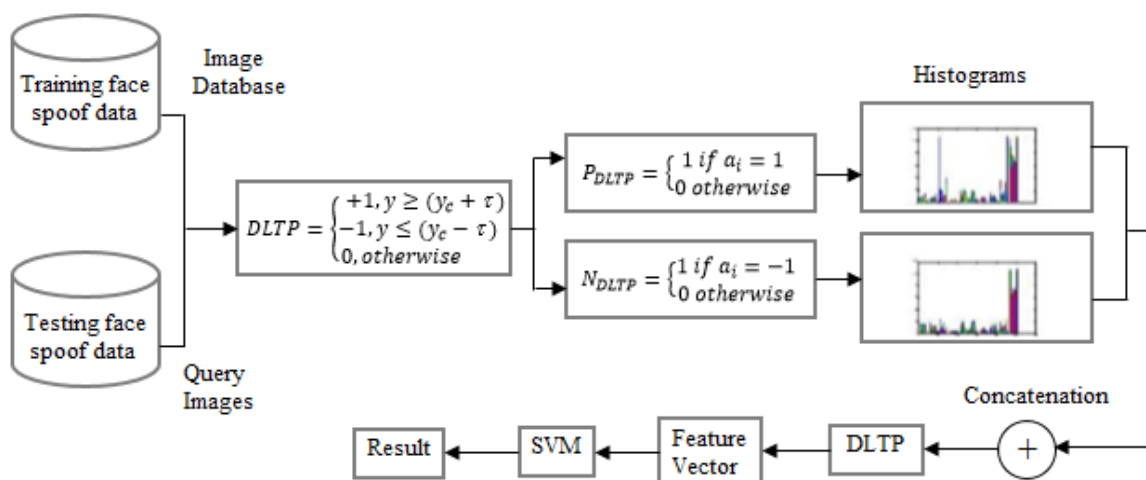


Figure 2.2. Face liveness detection system using the DLTP feature descriptor (adapted from [12]).

The authors adopted Weber's law for tuning the threshold value dynamically for every image pattern in the local ternary pattern, where the threshold equation for ternary quantization in LTP is given by equation 2.5.

$$a_i = \begin{cases} 1 & \text{if } p_i > c + \tau \\ 0 & \text{if } c - \tau \leq p_i \leq c + \tau \\ -1 & \text{if } p_i < c - \tau, \end{cases} \quad (2.5)$$

where c is the intensity value of the center pixel, p_i ($i = 0, 1, \dots, p-1$) is the intensity value of the neighborhood pixels, and τ is the threshold value. In DLTP, an equation for threshold was introduced as in equation 2.6, instead of choosing a fixed value of threshold as in LTP.

$$\frac{|p_i - c|}{c} = \tau \quad (2.6)$$

Equation 2.6 was used to generate the DLTP code, in which the threshold value is dynamically generated and assigned the code of 1, 0, and -1 within the zone of \pm threshold value around the central pixel value. The dynamic setting of different threshold values for every patch of the image is automatically supported by the value of τ . The threshold value (τ) obtained from equation 2.6 above was used to generate DLTP code as shown in equation 2.7 below.

$$DLTP(a_i) = \begin{cases} 1 & \text{if } p_i > c + \tau \\ -1 & \text{if } p_i \leq c - \tau \\ 0 & \text{otherwise} \end{cases} \quad (2.7)$$

As shown in Figure 2.2, the ternary code (+1, 0, -1) was generated in DLTP and was further divided into its corresponding positive and negative parts. These were treated

as two separate binary patterns, the histograms were computed separately, and the results were then combined at the end of computation. The support vector machine with linear kernel was used for classifying the live and fake images. Performance evaluation of the proposed method on the NUAA dataset, Replay-Attack dataset, and CASIA dataset showed that, due to the dynamic values of threshold in DLTP as opposed to fixed values in LTP, the DLTP outperforms LTP giving higher accuracy and lower error rates. Table 2.1 shows the performance comparison of DLTP with other approaches based on texture analysis.

Table 2.1. Performance comparison on NUAA dataset [12].

Techniques	HTER (%)
Local Binary Pattern Variance (LBPV)	11.97 and 13.05
Local Binary Pattern (LBP)	18.32, 19.03 and 13.17
Local Ternary Pattern (LTP)	7.4
Dynamic Local Ternary Pattern (DLTP)	3.5

Maatta et al. [13] analyzed the texture of facial images using Multiscale Local Binary Patterns (MLBP). Their proposed method adopted the local binary patterns for describing not only the micro-textures, but also the spatial information. The micro-texture patterns were then encoded into an enhanced feature histogram, which was fed to an SVM classifier. The authors suggest that micro-texture details needed for discriminating between live and fake faces can best be detected using a combination of different LBP operators. Therefore, they derived an enhanced facial representation using multi-scale LBP operators.

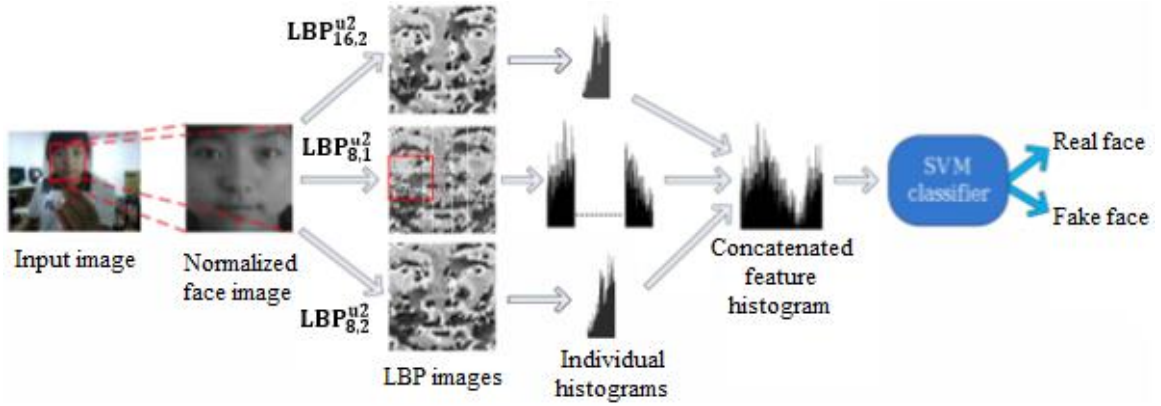


Figure 2.3. The proposed approach by Maatta et al. [13].

As depicted in Figure 2.3, the LBP operator, $LBP_{8,1}^{u2}$, was applied to the normalized face image and the resulting image was divided into 3x3 overlapping regions. The local histograms from each region were computed and aggregated into a single histogram. Two other histograms from the whole face image were then computed using $LBP_{8,2}^{u2}$ and $LBP_{16,2}^{u2}$ operators, and these two histograms were added to the previously computed histogram. Finally, a nonlinear SVM classifier with radial basis function kernel was used for classifying the image as real or fake. The authors reported a classification accuracy of 98% on the NUAA dataset.

Chingovska et al. [14] addressed the problem of detecting face spoofing attacks using texture features based on Local Binary Patterns (LBP), and their variations as well. For the basic LBP method, they calculated the LBP histogram in two different ways. In the first method, they calculated the LBP features for all pixels in the image and distributed them in one histogram, and in the second method, they divided the image into 3x3 blocks, calculated the LBP histogram for each block, and concatenated these histograms to form the final feature vector. Their experiments with variations of LBP

include transitional LBP, direction-coded LBP, and modified LBP. With the transitional LBP operator, the binary patterns were formed by comparing two consecutive neighboring pixels of the central pixel circularly in the clockwise direction. For the directional LBP, the intensity variation along the four base directions through the central pixel was encoded in two bits. For the modified LBP operator, the values of the neighboring pixels were compared with the average of the intensity values in a 3x3 neighborhood. They also examined two different classifiers, which included the Linear Discriminant Analysis (LDA) and the Support Vector Machine (SVM) with radial kernel basis function. Their experiments on the Replay-Attack dataset gave lowest HTER for the modified LBP followed by the $LBP_{3 \times 3}^{u2}$. Gagnaniello et al. [15] proposed a domain-aware CNN architecture by adding appropriate regularization terms to the loss function, with which they obtained satisfactory results on the Replay-Attack dataset.

Recent work in face liveness detection has been based on the use of deep CNN architectures [16]-[17] as these provide better liveness detection accuracy than the previously mentioned approaches. The work proposed in [16] focused on training deep CNNs for liveness detection by employing data randomization techniques similar to bootstrapping. Here, the training data was continuously randomized before applying to the CNN, in a form of single small mini-batches. Therefore, rather than randomly arranging the training set once, they continuously picked random mini-batches from the whole dataset at each training epoch, thereby reducing the training time. Their proposed approach gave satisfactory results in intra-database and cross-database face liveness detection tests.

The research proposed in [17] utilized a combination of diffusion of the input face followed by only a three-layer CNN architecture, to detect face spoofing attacks that utilize a single frame of sequenced frames. They applied an AOS-based schema with a large time step size to generate the speed-diffused image. The large time step parameter helps in extracting the sharp edges and texture features in the input image, and the CNN extracts the local and complex features from the diffused image. Experiments performed by using a single frame of each video from the Replay-Attack dataset which consists of 1200 short videos of real-access and spoofing attacks, gave an HTER of 10%. The same CNN architecture as mentioned in [17] when applied to the NUAA dataset gave an accuracy of 99% [18]. Their proposed CNN architecture extracts not only the depth information but also the texture surface of the face, as shown in Figure 2.4 below. The output of the first convolutional layer shows that the real face has more edges and distinct corners around the eyes, nose, lips, and cheek region, whereas the fake face has fewer edges.

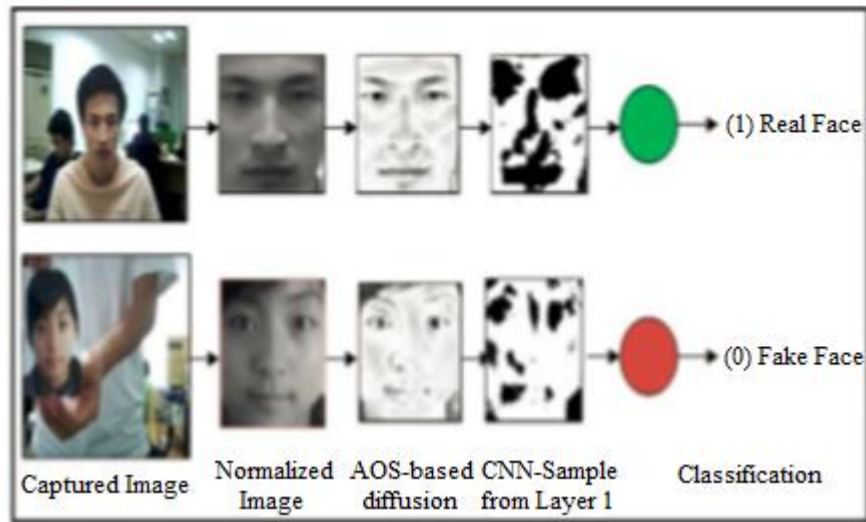


Figure 2.4. Nonlinear diffusion followed by CNN applied to the NUAA dataset [18].

2.3 Dynamic techniques proposed in the literature

In addition to static techniques, many dynamic techniques for face liveness detection have been proposed. However, since the dynamic approach is based on analyzing the temporal and spatial features of a sequence of input frames as opposed to a single image in the static approach, it is computationally expensive [18]. Some methods involve user cooperation in following certain instructions, which is inconvenient for the user.

In [19], Kim et al. proposed a method by using the focus function of the camera, for face liveness detection. Their approach made use of the variation of pixel values by focusing between two images sequentially taken in different focuses. The camera can be controlled to take pictures focused on facial features such as nose and ears. For a real face, the focused region will be clear and the rest blurred due to depth information, whereas there will be little difference between images taken in different focuses from a printed copy of a face. As the first step, two sequential pictures were taken, where in one the camera was focused on the nose which is nearest to the lens, and in the other, the camera was focused on the ear which is farthest from the lens. The Sum Modified Laplacian (SML) was utilized as the focus value measurement. After obtaining SMLs of each image, the sum of the difference in SML in each column of both images was computed. For a real face, this sum showed similar patterns consistently, whereas for a fake face, it did not. The differences in these patterns between real and fake faces were used as features for liveness detection.

In [20], the authors proposed three liveness clues, with two in temporal domain, and one in spatial domain, which were then fused to give the final decision. The first of these clues is the non-rigid motion clue, since real faces can exhibit non-rigid facial motions compared to fake faces. The second is the face-background consistency, which is based on the fact that, since fake faces always reside on certain displaying medium, fake facial motion is consistent with the background motion. The third is the banding effect which exists in only fake images due to the degradation of quality in reproduction.

Wang et al. [21] proposed a face liveness detection approach where the sparse structure information in 3D space was analyzed. Facial landmarks were detected from the given face video, and key frames were selected. Then, the sparse 3D facial structure was recovered from these selected key frames. After recovering the sparse 3D structure, the structures were aligned and the structure features were extracted for classification. Following alignment, the 3D coordinates of sparse structure were concatenated to form a feature vector. An SVM classifier was then trained based on the features to distinguish between the real and fake faces.

Pereira et al. [22] presented a counter measure against face spoofing attacks based on the Local Binary Patterns from Three Orthogonal Planes (LBP-TOP) operator combining both space and time information into a single multiresolution texture descriptor. The concept of Volume Local Binary Patterns (VLBP) extends LBP to image sequences, exploring both space and time information. VLBP considers the frame sequence as a parallel sequence, in order to capture inter-frame patterns in textures. The

VLBP output is defined as in equation 2.8, by considering a 3x3 kernel and thresholding the surroundings of each pixel with the central pixel of the frame sequence.

$$VLBP_{L,P,R} = \sum_{q=0}^{3P+1} f(i_c - i_q) 2^q, \quad f(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \quad (2.8)$$

where L is the number of predecessor and successor frames, P is the number of neighbors of i_c that corresponds to the gray intensity of the evaluated pixel, i_q corresponds to the gray intensity of a specific neighbor of i_c , and R is the radius of the neighborhood. However, the histogram of this descriptor contains 2^{3P+1} elements, and the number of bins in such a histogram is not computationally tractable. Therefore, the LBP-TOP operator, which is a simplification of the VLBP operator, was used. In LBP-TOP, instead of considering the frame sequence as three parallel planes, three orthogonal planes intersecting the center of a pixel in the XY, XT, and YT directions were considered, where T is the time axis. Each frame of the original frame sequence was grayscaled and passed through a face detector, and then geometrically normalized to 64x64 pixels. The LBP operators were then calculated for each plane (XY, XT, and YT), and the histograms were computed and concatenated. Classification was done using both LDA and SVM with radial basis function kernel. Better results were obtained with the SVM classifier, and performance evaluation on the Replay-Attack dataset gave satisfactory results with an HTER of 7.60%.

Bharadwaj et al. [23] presented an approach for spoofing detection in face videos using motion magnification. They used LBP and a motion estimation approach using Histogram of Oriented Optical Flow (HOOOF) descriptor for feature extraction, and LDA

and nearest neighbor for classification. Tang et al. [24] proposed a liveness detection protocol called face flashing that flashes randomly generated colors and verifies the reflected light. It is a challenge-response protocol, where the challenge is a picture displayed on a screen and light is emitted by the screen onto the subject's face. The response is the light that is reflected immediately by the subject's face. This challenge-response was repeated many times, so that sufficiently enough responses could be collected to ensure security. Their proposed method proved effective, and gave high accuracy in various environments.

Anjos et al. [25] proposed a technique that is based on foreground and background motion correlation using optical flow. As shown in Figure 2.5, the video was first converted to grayscale before being fed to the face detector, and eventually to the optical flow estimator unit. The grayscaled video, the face detector output and the estimated flow were fed to the counter-measure that outputs scores on which the sequence was finally evaluated.

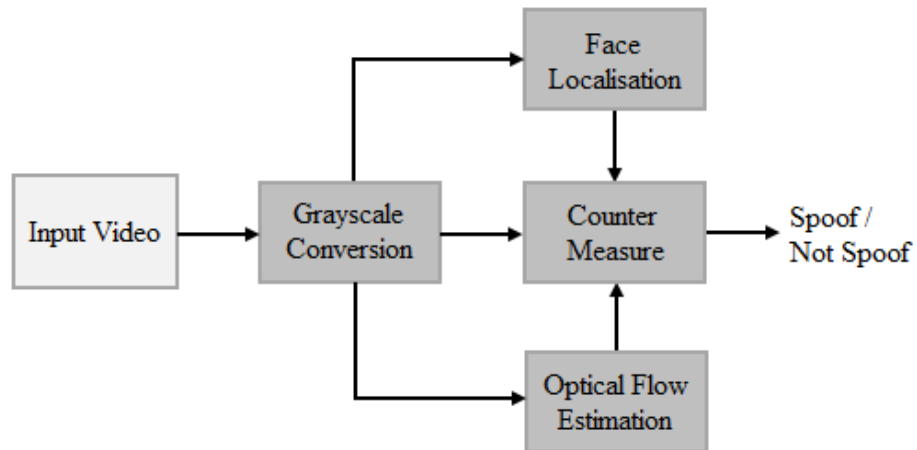


Figure 2.5. Outline of the proposed approach (adapted from [25]).

The proposed algorithm tried to detect motion correlations between the head of the user who is trying to authenticate and the background of the scene. It used fine-grained motion direction for deriving the correlation between these two regions. The direction of objects in the scene was estimated using optical flow techniques, which provides precise information of motion parameters between the regions of interest. The extracted features were then fed to a binary classifier to classify the sequence as real or fake.

Wen et al. [26] proposed a detection algorithm based on Image Distortion Analysis (IDA). They extracted four different features namely specular reflection, chromatic moment, blurriness, and color diversity to form the feature vector, which was then fed to an SVM classifier. Yeh et al. [27] proposed an approach against face spoofing attacks based on perpetual image quality assessment with multi-scale analysis. They used a combination of an image quality evaluator and a quality assessment model for selecting effective pixels to create the image quality features for liveness detection. Pan et al. [28] presented a time-based presentation attack detection algorithm for capturing the texture changes in a frame sequence. They used a Motion History Image (MHI) descriptor to get the primary features, and used LBP and a pre-trained CNN to get the secondary feature vectors, which were then fed to a classifier network. In the work proposed in [29], LBP-TOP is cascaded with a CNN to extract spatio-temporal features from video sequences, followed by SVM with RBF kernel for classification.

Xu et al. [30] proved that a deep architecture combining LSTM with CNN can be used for face anti-spoofing in videos. Local and dense features were extracted by the

CNN, while the LSTM captured the temporal relationships in the input sequences. The authors of [31] also proposed a joint CNN-LSTM network for face anti-spoofing in video sequences, by focusing on the motion cues across video frames. They used the Eulerian motion magnification as preprocessing to enhance the facial expressions of individuals. Then the CNN was used for extracting the high discriminative features of video frames, and the LSTM was used to capture the temporal dynamics in the videos.

Costa-Pazo et al. [32] introduced the Replay-Mobile database, and they describe two face presentation attack detection methods that were applied to the database. In one method, Image Quality Measures (IQM) were used as features, and in the other, a texture-based approach using Gabor-jets were used. Classification was done using the support vector machine with a radial basis function kernel. In the work proposed in [33], hand-crafted features and deep features were extracted by using a combination of Local Binary Patterns (LBP) and a pre-trained CNN model based on the VGG-16 network architecture for the liveness detection on single frames of the Replay-Mobile dataset.

Nikisins et al. [34] proposed an anomaly detection, or a one-class classifier-based face presentation attack detection system having better generalization properties against unseen types of attacks. They used an aggregated database consisting of three publicly available datasets, which includes the Replay-Attack dataset and Replay-Mobile dataset, for their experiments. Their system consisted of a preprocessor, feature extractor, and one-class classifier. They also evaluated and reported the results of some of the successful, previously published face liveness detection systems on the Replay-Attack and Replay-Mobile datasets, such as the LBP-based system in [14], the IQM-based

system where feature vector of a frame is a concatenation of quality measures introduced in [26] and [35], and the motion-based approach in [36]. Evaluation using the LBP-based system and IQM-based system were done on the frame-level, whereas the motion-based approach was done on video sequences.

Fatemifar et al. [37] adopted the anomaly detection approach where the detector is trained on genuine accesses only, using one-class classifiers built using representations obtained from deep pre-trained CNN models. Each frame in a video clip was photometrically normalized based on the retina method for reducing the impact of various lighting conditions before they were fed to pre-trained networks. They used different CNN architectures and anomaly detectors of which the class-specific Mahalanobis distance with GoogleNet features achieved better performance on the Replay-Mobile dataset. Arashloo [38] presented a one-class novelty detection approach based on kernel regression using the Replay-Mobile dataset in which only bona fide samples were used in the training process, as in [37]. A projection function defined in terms of kernel regression maps bona fide samples onto a compact cluster of target samples, and provides the best separability of normal samples from outliers with classification based on the Fisher criterion. Other mechanisms, which include a multiple kernel fusion approach, sparse regularization, and client-specific and probabilistic modeling were also incorporated to improve performance.

2.4 Conclusion

Face liveness detection is a necessary step in biometric authentication systems that use face recognition as the means of authentication. There are several methods that

have been proposed in the literature for face liveness detection. We provided a comprehensive review of these face liveness detection techniques, which can be classified into static techniques and dynamic techniques. The static techniques analyze a 2D static image for detecting spoofing attacks that present a printed photograph or an image displayed on a smartphone or tablet to the camera, while dynamic techniques analyze a sequence of input frames for detecting spoofing attacks that present a recorded video to the sensor.

Static techniques generally use texture analysis, frequency analysis, and image quality analysis in detecting the liveness of the captured image. In many of the static techniques, texture analysis is a preferred method, which is generally based on the Local Binary Patterns (LBP) or variations of it, such as ELBP, DLTP, MLBP, etc. These techniques have given promising results, however, advancements in printing technology can make some of the texture analysis methods less reliable. Other techniques include analysis of the 2D Fourier spectra using DoG filtering, using deep CNNs with data randomization, combination of diffusion and a CNN, etc. Compared to the above-mentioned static methods, our proposed method achieved higher accuracy on the NUAA dataset, using the Inception network version 4 architecture. In addition to the texture analysis, we also used a deep CNN architecture to identify the complex and deep features for classification. None of the existing works have employed deep architectures such as the ResNet50 and the Inception v4 that we employed in our research. The end-to-end solution we proposed has the advantage of detecting the liveness of a static image in real-time without having to go through a preprocessing step.

Dynamic techniques analyze the temporal and spatial features of a sequence of input frames. Some of these techniques include texture analysis using a variation of the local binary pattern such as VLBP, while other techniques rely on the 3D structure information, detection of motion over the frames in the sequence, etc. Compared to some of the methods described above that made use of hand-crafted feature extraction, we used a CNN that does the feature extraction by itself, thus eliminating hand-engineered feature extraction. The work proposed in [31] used 13 convolutional layers of the VGG-16, while we used a 4-layer CNN as the front-end of our architecture with which we got competitive results. The work presented in [37] and [38] used pre-trained models such as GoogleNet and ResNet50, while we designed a CNN-LSTM architecture with which we obtained better results on the Replay-Mobile dataset. The enhancement in our work is to apply nonlinear diffusion to the frames in the sequence to obtain the sharp edges and preserve the boundary locations, and then feed the diffused frames to the CNN-LSTM, and the Two-Stream Inflated 3D ConvNet (I3D).

CHAPTER 3: FACE LIVENESS DETECTION

In the case of face liveness detection of a captured static image, we first apply nonlinear diffusion to the image, and then feed the diffused image to CNN architectures. In the case of face liveness detection of video sequences, we first apply nonlinear diffusion to the frames in the sequence, and then feed the diffused frames to a CNN architecture for capturing the spatial features. The CNN output is then fed to an LSTM for capturing the temporal dynamics in the sequence. We also feed the diffused frames to a Two-Stream Inflated 3D ConvNet architecture.

3.1 Nonlinear Diffusion

In linear diffusion, the input image is smoothened at a constant rate in all directions to remove noise. Therefore, the smoothing process does not consider information regarding important image features such as edges [39]. The solution of the linear diffusion equation is given by:

$$\partial I / \partial t = \text{div} (d \nabla I), \quad (3.1)$$

where I is the image, d is the scalar diffusivity, and div is the divergence operator. This is somewhat equivalent to convolving the image with a Gaussian kernel, and hence linear diffusion can be regarded as a low pass filtering process.

Nonlinear diffusion is a denoising technique which denoises the image by preserving the edges. The edge-preserving capability of nonlinear diffusion makes it a powerful denoising technique, as the information contained in high spatial frequency components is preserved [40]. The denoised image is a solution of the diffusion equation with a diffusion coefficient that varies spatially. Nonlinear diffusion applied to face liveness detection helps in distinguishing a fake image from a real image, as the edges obtained from a fake image will be faded, while those obtained from a real image will remain clear. Anisotropic diffusion, which is nonlinear diffusion based on a partial differential equation, prevents the blurring and localization issues associated with linear diffusion, and focuses on reducing the image noise without reducing significant parts of the image content such as edges. It improves the scale-space technique, enhances the boundaries, and preserves the edges [41]. The diffusion coefficient is locally adapted and is chosen as a function of the image gradient, which varies with both the edge location and its orientation in order to preserve the edges. The nonlinear diffusion process is defined by the equation.

$$\partial I / \partial t = \operatorname{div} (g(|\nabla I|) \nabla I), \quad (3.2)$$

where ∇I is the gradient, and the diffusivity g is a function of the gradient ∇I .

During the diffusion process, the nonlinear diffusion filter detects the edges and preserves the locations using explicit schemes. The Additive Operator Splitting (AOS) scheme is a semi-implicit scheme which addresses the problem of regularization associated with anisotropic diffusion [42]. This scheme is stable for all time steps, and ensures that all co-ordinate axes are treated equally, as defined by equation (3.3) [43].

AOS enables fast diffusion, resulting in smoothing of the edges in fake images while the edges in real images will be preserved. Image information inside objects will be blended, while image information along edges will be left intact. The iterative solution in AOS is given in equation (3.3).

$$(I_k)^{t+1} = \sum_{l=1}^m (mI - \tau m^2 A_l)^{-1} I_k^t, \quad (3.3)$$

where I_k is the diffused image, m is the number of dimensions, k represents the channel, I is the identity matrix, A_l is the diffusion, and τ is the time steps (referred to as param. alpha in our implementation). In the two-dimensional case, $m = 2$, and the equation then becomes:

$$(I_k)^{t+1} = (2I - 4\tau A_1)^{-1} I_k^t + (2I - 4\tau A_2)^{-1} I_k^t, \quad (3.4)$$

where A_1 and A_2 denote the diffusion in the horizontal and vertical directions. The equation is split into two parts in the operator splitting scheme. The solution to each is computed separately and results are then combined.

The block-solver Tri-Diagonal Matrix Algorithm (TDMA) is a simplified form of Gaussian elimination, useful in solving tri-diagonal systems of equations. The AOS scheme, together with TDMA can, therefore, be used to efficiently solve the nonlinear, scalar-valued diffusion equation [43]. We implement the AOS scheme in the first layer of our implementations.

3.2 Convolutional Neural Networks

The Convolutional Neural Network (CNN), introduced by Lecun et al. [44], combines three architectural concepts of local receptive fields, shared weights, and spatial or temporal subsampling in order to ensure some degree of shift, scale, and distortion invariance. They eliminate the need for hand-crafted feature extraction and extract local features by restricting the receptive fields of hidden units to be local. CNNs have proved successful in many machine learning tasks such as handwriting recognition [44], natural language processing [45], text classification [46], image classification [44], face recognition [47], face detection [48], object detection [49], video classification [50], object tracking [51], super resolution [52], human pose estimation [53], and so forth.

The CNN architecture takes an image as input, performs the required number of convolutions and subsampling, and then feeds the outputs obtained to the required number of fully connected layers.

3.3 Long Short-Term Memory (LSTM)

The LSTM was introduced by Hochreiter et al. [54], and the constant error backpropagation within memory cells in the LSTM results in its ability to bridge very large time lags. LSTMs have been used successfully in video captioning [55], video object detection [56], intrusion detection in computer networks [57], hand gesture recognition [58], speech recognition, language modeling, sentiment analysis, text prediction, etc. The LSTM model resembles a standard Recurrent Neural Network (RNN), with the difference being that each node in the hidden layer is replaced by a memory cell (Figure 3.1). The self-connected recurrent edge of fixed weight one in each node ensures that the gradient can pass across many time steps without vanishing or

exploding [59]. Each cell consists of multiplicative units called gates (input, output, forget) that provide continuous analogues of write, read, and reset operations [60]. The multiplicative gate units learn to open and close access to the constant error flow through internal states of the cells [54].

LSTMs in combination with CNNs have proven successful in person identification using lip texture analysis [61], 3D gait recognition [62], image-to-video person re-identification [63], action recognition in video sequences [64], text classification [65], identification of pedestrian attributes in video sequences [66], etc.

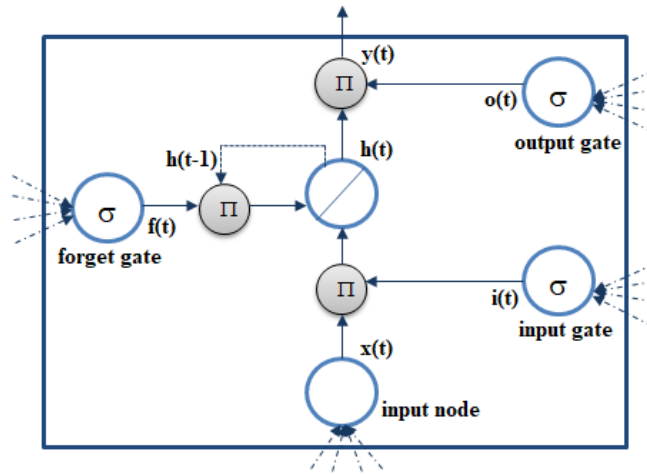


Figure 3.1. LSTM memory cell (adapted from [59]).

CHAPTER 4: FACE LIVENESS DETECTION ON A SINGLE STATIC IMAGE

For face liveness detection on a single image, we first applied nonlinear diffusion based on the Additive Operator Splitting (AOS) scheme and the block-solver Tri-Diagonal Matrix Algorithm (TDMA) to the captured image, in order to enhance the edges and preserve the boundary locations of the image. These diffused input images were then fed to the CNN architecture to extract the complex and deep features, and finally classify the image as real or fake. We used three different CNN architectures, and performed a comparative evaluation on their performance, thus gaining insight into why a particular architecture is better suited for face liveness detection. The CNN architectures we used in our experiments are described below:

4.1 CNN-5

This architecture consists of a total of five layers, which includes two convolutional layers, one subsampling layer, and two fully connected layers. The inputs to the network are the 64×64 size nonlinear diffused counterparts of the captured original images. The first layer is a convolutional layer C1 that consists of 12 feature maps, each of size 56×56 , where each unit in a feature map is the result of the convolution of the local receptive field of the input image with a 9×9 kernel. The kernel

used in the convolution is the set of connection weights used by the units in the feature map. Each unit in a feature map shares the same set of weights and bias, so they detect the same feature at all possible locations in the input. Other feature maps in the layer use different sets of weights and biases, extracting different local features [44]. The next layer is also a convolutional layer, C2, consisting of 18 feature maps, each of size 50×50 , obtained by the convolution of the feature maps in C1 with 7×7 kernels. C2 is followed by a subsampling layer S2 of 18 feature maps, each of size 25×25 , obtained by applying average pooling of size (2, 2) to the corresponding C2 layer feature maps, thereby reducing the resolution of the feature maps in the C2 layer by half. The next layer is a fully connected hidden layer of 50 neurons, followed by a fully connected output layer of 2 neurons, and dropouts of probability 0.25 and 0.4 are applied to the pooling layer and hidden layer respectively. The complete architecture is illustrated in Figure 4.1.

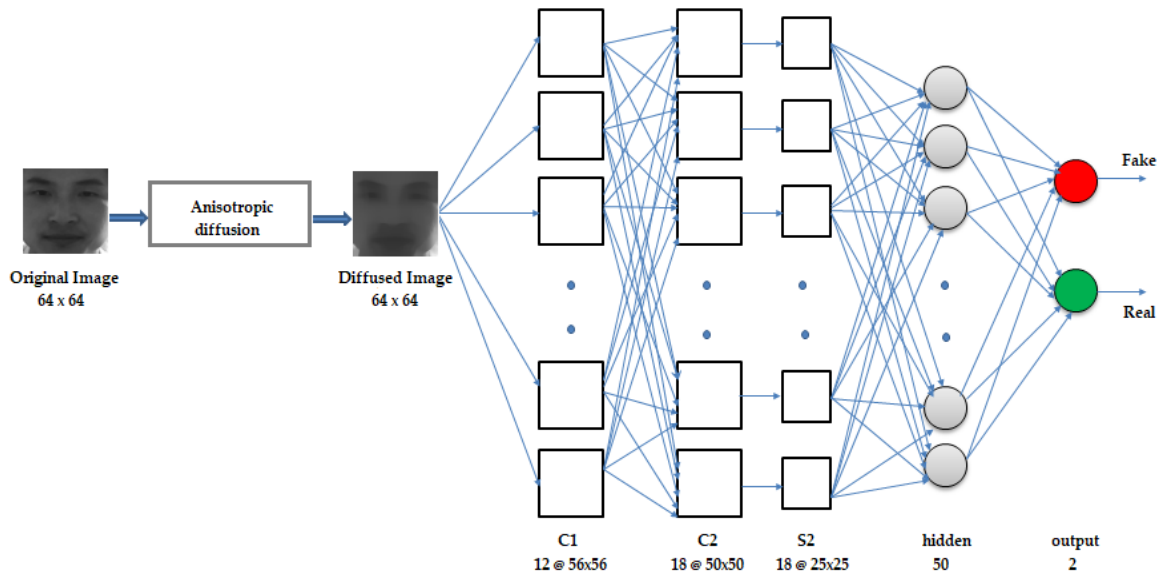


Figure 4.1. Convolutional Neural Network (CNN-5) architecture.

To introduce nonlinearity into the model, the Rectified Linear Unit (ReLU) activation function is applied to the outputs of C1, C2, and the hidden layer of 50 neurons, restricting the outputs of these layers to be 0 or x , where x is the output of the neuron before the activation function is applied. The sigmoid activation function is applied to the output layer, giving an output in the range of 0 to 1. The network was trained by backpropagation using the Adam optimization algorithm, with mean squared error as the loss function. The diffusion block was implemented via direct implementation of the diffusion equations.

4.2 ResNet50

As the depth of a deep network increases, the accuracy becomes saturated and the network degrades rapidly. The deep residual learning framework improves the degradation problem and eases the training of the network [67]. In a residual network, there are shortcut connections which skip one or more layers. To the standard CNN, skip connections are added that bypass a few convolution layers. Each bypass produces a residual block where the convolution layers predict a residual that is added to the block's input. Therefore, these shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers. This gives rise to a shallower architecture, and the entire network can still be trained end-to-end. The identity connections add neither extra parameters nor computational complexity to the network. Residual networks have been used successfully in age and gender estimation [68], for hyperspectral image classification [69], and other classification tasks. Even though very

deep residual networks (152 layers) have been used, in this work, we only used 50 layers, as our focus was on a comparative evaluation of architectures.

The ResNet50 is a residual network of 50 layers, which consists of identity residual blocks and convolutional residual blocks stacked together. The shortcut connections allow the gradient to be backpropagated to earlier layers, preventing the vanishing gradient problem associated with very deep networks. The central idea of residual networks (ResNets) is to learn the additive residual function, using an identity mapping realized through identity skip connections [70]. The shortcut connections parallel to the main path of convolutional layers allow the gradient to backpropagate through them, resulting in faster training. Figure 4.2 shows the basic residual block, and Figure 4.3 shows the ResNet50 architecture used in our research.

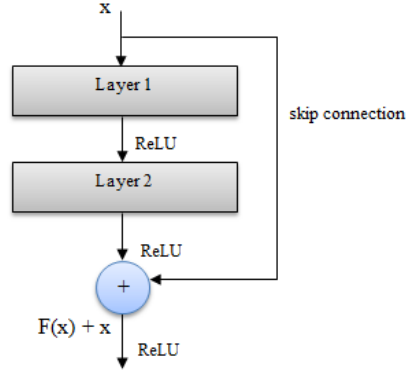


Figure 4.2. Basic residual block (adapted from [67]).

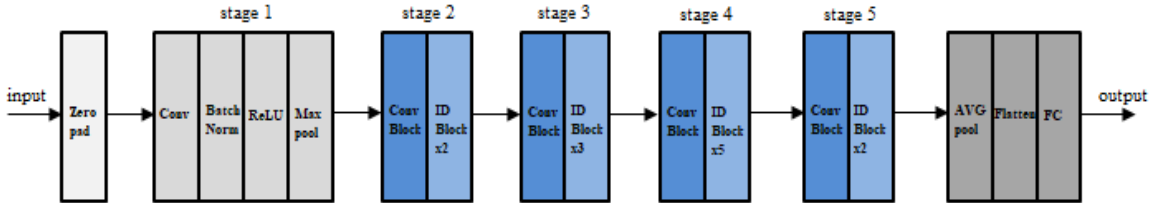


Figure 4.3. ResNet50 architecture.

The architecture consists of identity blocks and convolutional blocks stacked together in stages 2–5. The identity residual block is used when the input and output dimensions are same before the addition, whereas the convolutional residual block is used when the input and output dimensions are different before the addition.

The inputs to the network are the 64×64 nonlinear diffused images. The network was trained using the Adam optimization algorithm, with categorical cross-entropy as the loss function and softmax as the classifier. Cross-entropy measures the performance of a classification model where the output is a probability distribution, giving a value between 0 and 1. Since our targets are in categorical format with two classes (fake, real), we used the categorical cross-entropy as the loss function.

4.3 Inception v4

Inception architectures have shown to achieve very good performance at relatively low computational cost [71]. The inception network’s architecture provides improved utilization of the computing resources inside the network through a design that increases the depth and width of the network while keeping the computational budget constant. The network consists of inception modules stacked upon each other, with occasional pooling layers to reduce the resolution of the grid. Information is processed at various scales by applying filters of different sizes to the same layer, which are then aggregated so that the following stage can abstract features from different scales simultaneously [72].

The basic inception module (Figure 4.4) used filter sizes 1×1 , 3×3 , and 5×5 , the convolution outputs and a max pooling output of which were concatenated into a

single output vector forming the input to the next stage. Instead of using a filter of an appropriate size at a level, filters of multiple sizes were used, making the network wider than deeper, so that features at different scales could be recognized. The resulting feature maps were then concatenated before going to the next layer. This was further refined by incorporating dimensionality reductions by applying 1×1 convolutions before the 3×3 and 5×5 convolutions. Auxiliary classifiers were also included to prevent the vanishing gradient problem.

Later versions incorporated factorization of filters into smaller convolutions by replacing the 5×5 filters with two 3×3 filters [73]. This reduced the computational cost, resulting in a reduced number of parameters and faster training. Additionally, the $n \times n$ convolutions were replaced by a $1 \times n$ convolution followed by an $n \times 1$ convolution, which further increased the computational cost savings. Also, filter banks were expanded to avoid representational bottleneck. Batch normalization was applied to the fully connected layer of auxiliary classifiers for additional regularization, and label smoothing was added as regularization to the loss formula to make the model more adaptable and prevent overfitting. Deep CNN with inception modules has been used for person re-identification [74], inception network version 3 has been used for Next Generation Sequencing (NGS) – based pathogen detection [75], and a residual inception network has been used in multimedia classification [76].

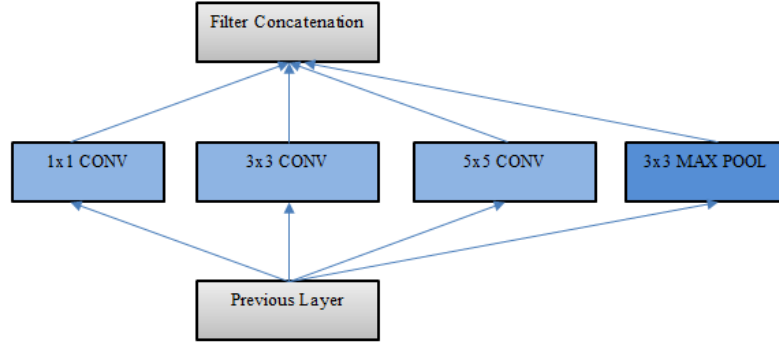


Figure 4.4. Basic inception module (adapted from [72]).

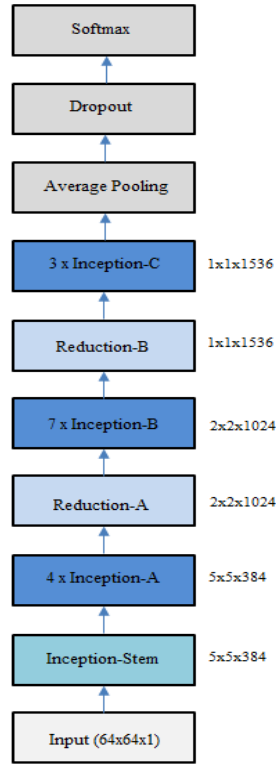


Figure 4.5. Inception v4 network (adapted from [71]).

In our experiments, we used the Inception v4 architecture, which is a more uniform simplified structure compared with the earlier versions. This model consists of three different inception blocks where each is used repeatedly a certain number of times,

and two reduction blocks for changing the width and height of the grid. The inputs to the network are the 64×64 nonlinear diffused images. The complete network (Figure 4.5) consists of an inception stem, 4 x inception A blocks, 1 x reduction A block, 7 x inception B blocks, 1 x reduction B block, and 3 x inception C blocks, followed by average pooling, dropout, and softmax layers. The network was trained using the Adam optimization algorithm, with categorical cross-entropy as the loss function and softmax as the classifier.

CHAPTER 5: END-TO-END REAL-TIME FACE LIVENESS DETECTION USING ANISOTROPIC DIFFUSION AND CONVOLUTIONAL NEURAL NETWORK

For the end-to-end real-time solution for liveness detection on static images, instead of using a preprocessing step (via Matlab code) for diffusing the images and feeding them to the deep CNN network as described in Chapter 4, we developed an end-to-end framework with diffusion as well as deep CNNs. We used a combined architecture where the diffusion process and deep CNN are implemented in a single step. We used two different methods for the end-to-end solution. In the first method, we used an alpha trainable network that computes the smoothness of diffusion parameter (α), and the diffused image was then created using this computed α . This diffused image was then fed to a pre-trained three-layer CNN model (CNN layers with Batch Normalization) that gave 97.50% accuracy on the Replay-Attack dataset, and 99.62% accuracy on the Replay-Mobile dataset, respectively. In the second method, we fixed a value for the smoothness of diffusion parameter (α) using which we created the diffused image, and then fed the diffused image to an Inception v4 network.

In either case, we fed the original captured images to the framework, where the first layer computed the nonlinear diffusion based on the Additive Operator Splitting

(AOS) scheme and the efficient block-solver, Tri-Diagonal Matrix Algorithm (TDMA). This enhances the edges and preserves the boundary locations of the real image. The diffused input image was then fed to the deep CNN architecture or the Inception v4 network to extract the complex and deep features, and to classify the image as real or fake. Our integrated implementation results in real-time detection of liveness.

5.1 Specialized Convolutional Neural Network (SCNN)

In this architecture, the original image is first fed to an alpha network to compute the value of the smoothness of diffusion parameter (alpha). This is a neural network comprising of a hidden layer of 15 neurons followed by a dense layer of one neuron, which outputs the alpha. The Rectified Linear Unit (ReLU) activation function is applied to the neurons in these layers. The SCNN model consists of three convolutional layers C1, C2, and C3 with 16, 32, and 64 feature maps respectively, where kernel sizes of 15×15 , 7×7 , and 5×5 are used in the convolutions. Each convolution is followed by batch normalization, and max pooling is applied to the C1 and C2 layers after batch normalization for reducing the resolution. The higher filter size in the C1 layer is important to extract the diffusion enhanced features for liveness detection. The C3 layer batch normalization is followed by a dense layer of 64 neurons, and a dense output layer of one neuron. The ReLU activation function is applied to the convolution layers and the hidden layer, and the sigmoid activation function is applied to the output layer. The SCNN was trained using the binary cross-entropy loss function, and the Adam optimizer with an initial learning rate set to 0.001.

Figure 5.1 below shows the proposed architecture. The nonlinear diffusion code implemented in TensorFlow converts the original image to diffused form with the parameter α determined from the alpha network (bottom left in Figure 5.1). During backpropagation, the weights in the alpha network will be updated, and the updated weights are used in the computation of the output of the single neuron in the dense layer (parameter α) on the next forward pass.

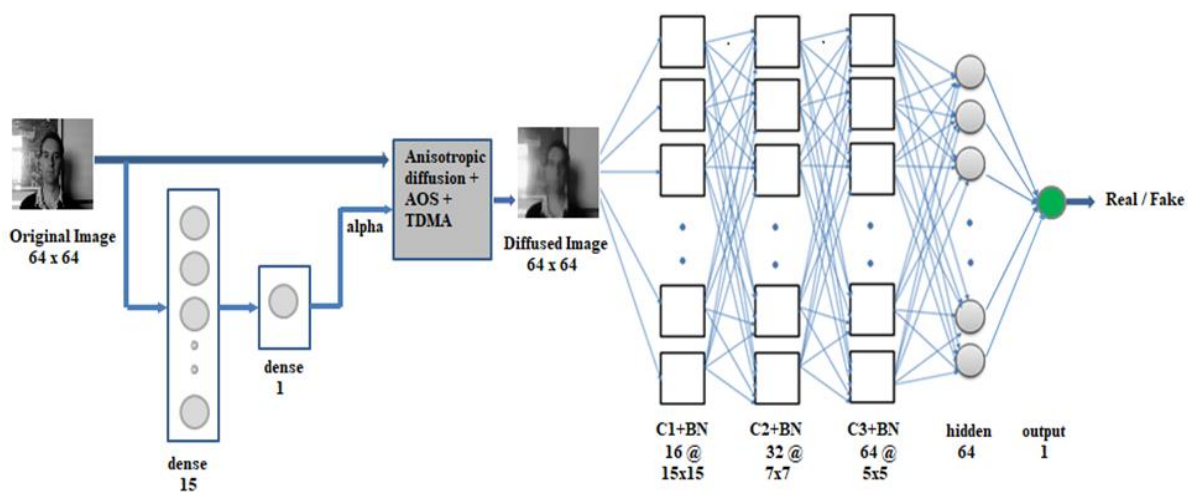


Figure 5.1. End-to-end Specialized Convolutional Neural Network (SCNN) architecture (α is learned by the network).

5.2 Inception v4

In this framework, the CNN part in Figure 5.1 is replaced with the Inception v4 network. However, instead of using an alpha network that computes the smoothness of diffusion (α), we fix the value of α in order to improve the real-time performance. The first stage is the nonlinear diffusion stage, whose input is the original 64×64 input image captured through the webcam, which is followed by the Inception network v4 architecture described in section 4.3. The complete architecture is illustrated in Figure

5.2. The network was trained by using the Adam optimization algorithm. Since our targets are in a categorical format with two classes of fake and real, we used categorical cross-entropy as the loss function. The diffusion block was implemented via direct implementation of the diffusion equations.



Figure 5.2. End-to-end architecture using the Inception v4 network.

CHAPTER 6: FACE LIVENESS DETECTION ON VIDEO SEQUENCES

For the liveness detection on video sequences, we first applied the nonlinear diffusion based on the AOS scheme and TDMA to the individual frames of the video sequence, as done for the static images. This enhances the edges and preserves the boundary locations of the real image. These diffused input images were then fed to two different architectures.

6.1 CNN-LSTM

This architecture consists of a CNN as the front-end, followed by an LSTM. The diffused images were fed one-by-one to the CNN, which extracts the complex and deep features. The output of the CNN was then fed to the LSTM, which detects the temporal information in the sequence. The input gate of the LSTM cell indicates how much of the new information must be stored in the cell state, the forget gate indicates how much of the internal state can be removed, and the output gate indicates how much of the cell state can be sent as output to the next time-step. Finally, the output dense neural network layer classifies the sequence as real or fake.

The CNN part in our CNN-LSTM network has convolutional layers C1 and C2, one subsampling layer, and a fully connected layer of 50 neurons (adapted from the

CNN-5 in section 4.1). This is followed by the LSTM layer, which consists of 60 cells, and a feedforward output layer of two neurons. The sigmoid activation function is applied to the output layer, giving an output in the range 0 to 1. Nonlinear diffusion is first applied to the frames in each input sequence, and the diffused frames are fed to the CNN.

The CNN captures the spatial information in the sequence by extracting the complex and deep discriminative features, and the hidden layer of CNN produces an output of 50 features per frame. The input to the LSTM layer is three-dimensional, where the three dimensions are samples, time steps, and features (i.e., batch size, 20, 50), where 20 is the number of frames (time steps) per sequence. The LSTM layer captures the long-term temporal dependencies across frames in the sequence, and the 60 features obtained from the LSTM layer are fed to the output layer, which then classifies the 20-frame sequence as real or fake. The complete architecture is illustrated in Figure 6.1 below.

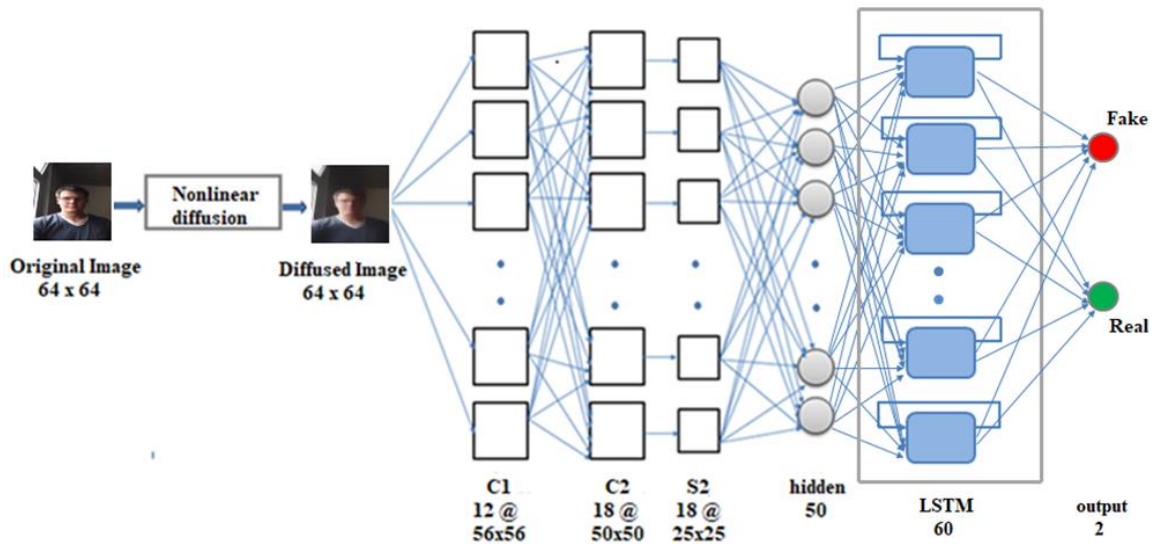


Figure 6.1. Convolutional Neural Network and Long Short-Term Memory (CNN-LSTM).

6.2 Two-Stream Inflated 3D ConvNet (I3D) architecture

This architecture [77] consists of two 3D ConvNets in which the filters and pooling kernels are inflated into 3D by adding an additional temporal dimension, which leads to very deep naturally spatio-temporal classifiers. The CNN used is the ImageNet pre-trained Inception v1 network with batch normalization, and a two-stream configuration is adopted. Nonlinear diffusion is first applied to the frames in each input sequence, and the diffused frames are fed to the network.

The inputs to one 3D ConvNet are the diffused RGB frames, and the inputs to the other 3D ConvNet are the corresponding optical flow frames. While 3D ConvNets may directly learn about the temporal information from an RGB stream, their performance is greatly enhanced with the addition of an optical flow stream. The figure below (Figure 6.2) shows the Two-Stream Inflated 3D ConvNet (I3D) architecture.

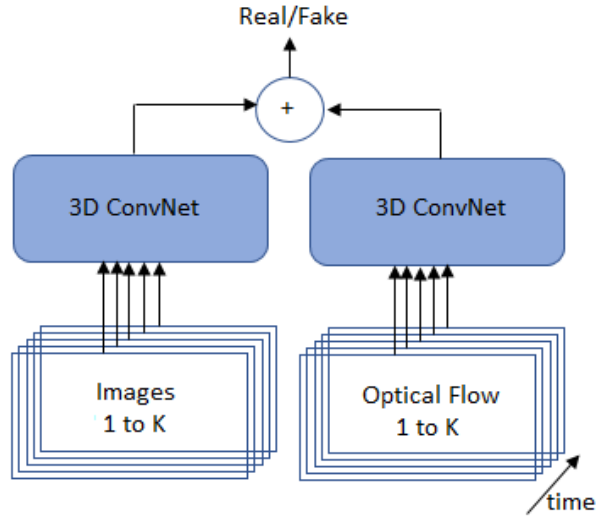


Figure 6.2. Two-Stream Inflated 3D ConvNet (I3D) architecture (adapted from [77]).

As shown in figure, one 3D ConvNet is trained on diffused RGB inputs, while the other is trained on optical flow inputs which carry optimized smooth flow information. The two networks are trained separately and their predictions are averaged at test time.

CHAPTER 7: IMPLEMENTATION AND RESULTS

7.1 Face Liveness Detection on a static image

For face liveness detection on a single static image using the three architectures CNN-5, ResNet50, and Inception v4, we used NUAA as the dataset. The NUAA Photograph Impostor database [10] consists of images of 15 subjects and is publicly available. It contains both live and photographed faces of the subjects, captured using a webcam in three sessions, with each session at a different place with different illumination conditions. There is a total of 12,614 images in the dataset, with 3491 training images and 9123 test images, as shown in Table 7.1. Each image is 64×64 in size and in grayscaled representation. There is no overlap between the training set and test set images. The training set images were taken during sessions 1 and 2, and the test images were taken during session 3. The training set contains images of subjects 1–9, whereas the test set contains images of subjects 1–15.

Table 7.1. NUAA dataset.

	Training Images	Test Images	Total
Real (Client)	1743	3362	5105
Fake (Impostor)	1748	5761	7509
Total	3491	9123	12614

For all the images in the training set and test set of the NUAA dataset, we created the corresponding nonlinear diffused images using the MATLAB implementation code by the author of [43]. We created five sets of diffused images for our experiments, with different values (15, 25, 50, 75, and 100) of the parameter param. alpha that defines the smoothness of diffusion. Figure 7.1 shows some samples from the NUAA dataset, and the corresponding diffused images.

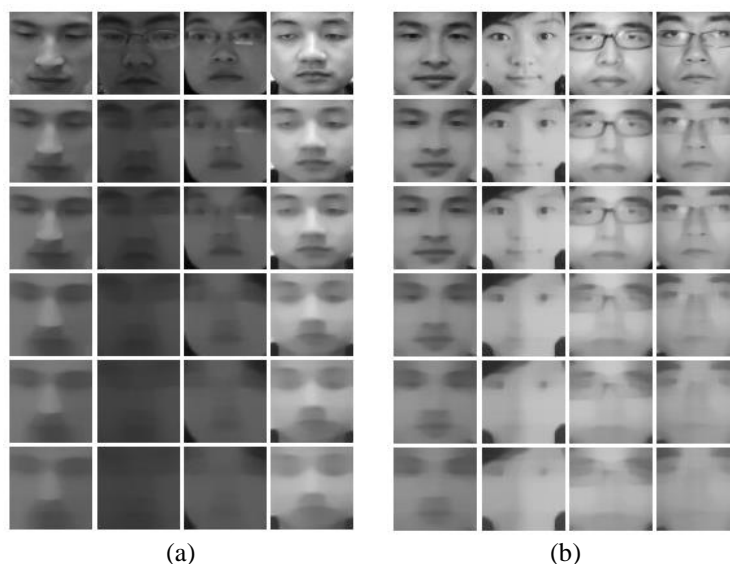


Figure 7.1. Samples from the NUAA database. The first row in (a) shows non-diffused real images in the database, and the first row in (b) shows non-diffused fake images. Each row below shows the corresponding diffused images created with value of param. alpha set to 15, 25, 50, 75, and 100 respectively.

Most of our experiments were implemented on a 32GB RAM PC. The code was written in Python, using the Keras library with TensorFlow backend, for all the three architectures. We tested the CNN architectures with each set of diffused images, i.e. with diffused images created by setting the smoothness of diffusion parameter param. alpha to 15, 25, 50, 75, 100. We conducted numerous experiments with the three CNN

architectures on the NUAA dataset. We performed several tests by changing the hyper-parameters during the learning phase for improving the test accuracy. We compared the three architectures in terms of performance and, further compared their performance with other existing liveness detection methods. For each of the CNN architectures we used, we computed the test accuracy after training for various epochs.

7.1.1 CNN-5

With the CNN-5 architecture, using the diffused images created by setting the value of the parameter that defines smoothness of diffusion (param. alpha) to 25, we obtained a test accuracy of 95.99% after training for 30 epochs using the Adam optimizer and mean-squared-error loss function. The learning rate was set to 0.001, which is the default of the Adam optimizer, and the batch size was set to 32. The activation functions used were ReLU for the convolutional layers and hidden layer, and sigmoid for the output layer. Average pooling was applied to the C2 layer for down-sampling. Table 7.2 shows the test accuracies obtained with CNN-5 after training for various epochs.

Table 7.2. Test results obtained with the CNN-5 architecture.

Epochs	10	15	20	25	30	35	40
% Test Accuracy	62.30	69.04	65.77	79.14	95.99	81.15	76.47

We obtained a slightly less accuracy of 94.78% when max pooling was used after the C2 layer, categorical cross-entropy as the loss function, and softmax as the classifier. This accuracy was obtained after training for 35 epochs, with Adadelata as optimizer and learning rate of 0.1 which is its default, and with batch size set to 32.

7.1.2 ResNet50

With the ResNet50 architecture, various combinations of kernel initialization, loss functions, activation functions, and classifiers, were experimented. With the diffused images created by setting the smoothness of diffusion (param. alpha) to 25, we obtained a test accuracy of 95.85% after training for 20 epochs, with the Adam optimizer, categorical cross-entropy loss function, and softmax classifier. The learning rate was kept at the default of Adam optimizer, which is 0.001, and batch size was set to 32. The table below (Table 7.3) shows the accuracies obtained with ResNet50 after training for various epochs.

Table 7.3. Test results obtained with the ResNet50 architecture.

Epochs	10	15	20	25	30	35	40
% Test Accuracy	86.75	73.21	95.85	76.25	82.96	75.32	88.61

We also tested for liveness detection by reducing the number of feature maps in the convolutions by a quarter. In addition to the Adam optimizer, we also experimented with Adadelta, RMSprop, SGD optimizers, but the best results were obtained with Adam optimizer. We also experimented with two different kernel initializers, the glorot-uniform() initializer and the random-uniform() initializer. It was observed that, on average, glorot-uniform() initializer gives better results than random-uniform() initializer. We further observed that the network with the original number of feature maps in the convolutions gives better results than the network with reduced number of feature maps.

7.1.3 Inception v4

With the Inception v4 architecture, we achieved test accuracy of 100% that has not been reported by any previous approach for face liveness detection on the NUAA dataset. This high accuracy was obtained with diffused images created by setting the smoothness of diffusion parameter (param. alpha) to 15, after training for various epochs ranging from 10 to 40, using the Adam optimizer, categorical cross-entropy loss function, and softmax classifier. The learning rate used was the default of Adam optimizer, which is 0.001, and batch size was set to 32. The table below (Table 7.4) shows the summary of the results obtained with Inception v4 after training for various epochs.

Table 7.4. Test results obtained with the Inception v4 architecture.

Epochs	10	15	20	25	30	35	40
% Test Accuracy	100	100	100	100	100	100	100

We initially used diffused images created by setting the smoothness of diffusion (param. alpha) to 25, and experimented with different combinations of the number of inception A blocks, inception B blocks, and inception C blocks, with batch size set to 32. It was observed that the default of 4 inception A blocks, 7 inception B blocks, 3 inception C blocks gives better results. The results obtained are summarized in Table 7.5.

Table 7.5. Evaluation with various numbers of Inception-A, Inception-B, Inception-C blocks (param. alpha=25).

Number of blocks used	Epochs	Test Accuracy
4A, 7B, 3C	10	95.04%
3A, 5B, 2C	10	74.29%
7A, 9B, 5C	10	86.53%

We also conducted experiments keeping only parts of the architecture. We experimented with using only the inception stem, the inception A blocks, and reduction A block, eliminating blocks B and C. We further tried using only the inception stem followed by blocks A and B, leaving out block C, and also inception stem followed by blocks A and C, leaving out block B. The results in these cases were not as accurate. By using only the inception stem, inception A blocks, and reduction A block, the test accuracies obtained with various numbers of inception A blocks are summarized in Table 7.6 below.

Table 7.6. Evaluation using only Inception-stem, Inception-A blocks, Reduction-A block (param. $\alpha=25$).

Number of Inception A blocks	Epochs	Test Accuracy
4	10	89.07%
5	10	91.17%
6	10	90.91%
7	10	91.59%

With the diffused images created by setting the smoothness parameter (param. α) to 15, we also performed experiments by using a sample of the training images as validation dataset. Out of the 3491 training images, we used 3000 images for training, and 491 images for validation. After each epoch during the training process, the model was evaluated on the validation set, and this gives an estimate of the skill of the model. Table 7.7 shows the validation accuracy after each epoch, when the inception v4 network was trained for 25 epochs, and Figure 7.2 shows the corresponding plot of epoch versus validation accuracy.

Table 7.7. Validation accuracy (%) obtained after each epoch (param. alpha=15).

Epoch	Accuracy	Epoch	Accuracy	Epoch	Accuracy
1	85.13	11	99.39	21	99.19
2	97.76	12	99.19	22	98.17
3	97.15	13	98.98	23	98.37
4	96.74	14	99.19	24	99.59
5	99.19	15	98.98	25	99.19
6	99.19	16	99.59		
7	98.57	17	99.39		
8	99.19	18	98.57		
9	99.19	19	98.98		
10	98.57	20	98.78		

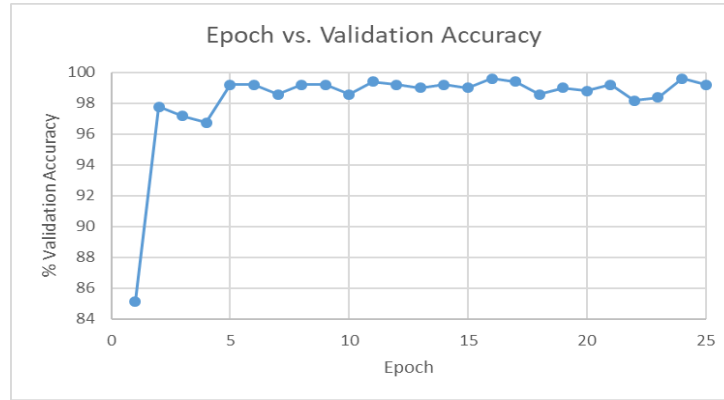


Figure 7.2. Plot showing epoch vs validation accuracy for Inception v4.

7.1.4 Comparison of the three architectures

An overall comparison of the three architectures showed that the Inception v4 had superior performance for face liveness detection compared with ResNet50 and CNN-5. The accuracies of ResNet50 and CNN-5 differed only by a slight amount, and though their accuracies were not as high as the Inception v4 network, they still provided competitive results, showing that they too can be used for face liveness detection of anisotropic diffused captured images. Table 7.8 shows the summary of the obtained results.

Table 7.8. Comparison of the three architectures.

Architecture	Epochs	Optimizer	Loss function	Test Accuracy
CNN-5	30	Adam	mean-squared-error	95.99%
ResNet50	20	Adam	categorical-crossentropy	95.85%
Inception v4	10	Adam	categorical-crossentropy	100%

Figure 7.3 shows the plot of epochs versus accuracy for the CNN architectures, using the results in Tables 7.2, 7.3, 7.4.

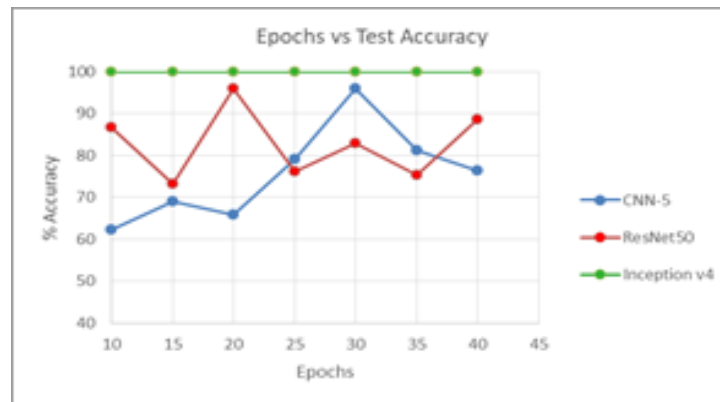


Figure 7.3. Plot showing epochs vs accuracy for each architecture.

We further observed from our experiments using the three networks that the best optimizer was the Adam optimizer. We tried various values of learning rates for this optimizer to see if increasing or decreasing the learning rate from the default value of 0.001 would improve the test accuracy, but it was observed that the default learning rate gave higher accuracy for ResNet50 and CNN-5. The Inception v4 still had 100% accuracy despite changing the learning rate. Table 7.9 summarizes the observations made.

Table 7.9. Accuracies obtained for various values of learning rates.

Architecture	Learning rate	Epochs	Accuracy
CNN-5	0.001	30	95.99%
	0.002	30	88.97%
	0.005	30	63.14%
	0.0008	30	72.83%
ResNet50	0.001	20	95.85%
	0.002	20	62.74%
	0.005	20	64.80%
	0.0008	20	66.76%
Inception v4	0.001	10	100%
	0.002	10	100%
	0.005	10	100%
	0.0008	10	100%

We tested each of the CNN architectures with each of the five sets of diffused images that we created with various values of the parameter alpha. We determined that the level of smoothness of the diffused images played a significant role in determining the liveness of the captured images. Table 7.10 summarizes the results obtained with each architecture using each set of diffused images. The entries in parentheses indicate the number of epochs we trained the network to achieve that accuracy. Figure 7.4 shows the corresponding plot of param. alpha of the diffused images versus the test accuracies obtained with each architecture.

Table 7.10. Accuracies obtained with each set of diffused images (the epochs are shown in parentheses).

param. alpha	CNN-5	ResNet50	Inception v4
15	88.69% (25)	78.76% (40)	100% (10)
25	95.99% (30)	95.85% (20)	95.04% (10)
50	79.41% (20)	70.21% (30)	91.76% (5)
75	76.35% (20)	78.77% (30)	87.56% (1)
100	75.77% (30)	74.47% (30)	86.04% (5)

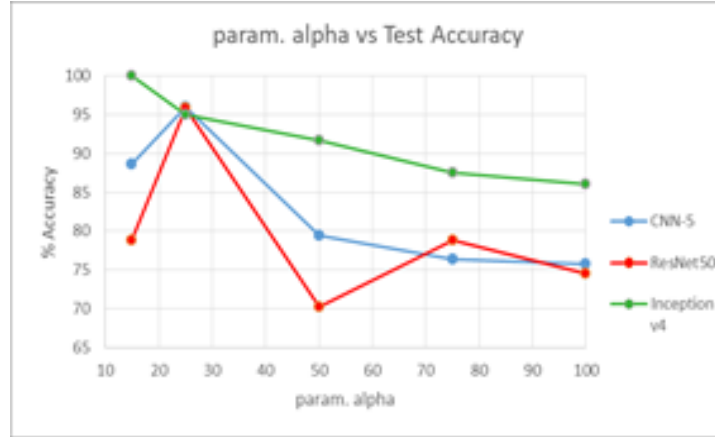


Figure 7.4. Plot showing param. alpha vs accuracy for each architecture.

As we have mentioned in our experimental results, with a lower value of the smoothness of diffusion parameter (param. alpha = 15), Inception v4 had superior performance compared with CNN-5 and ResNet50. Since filters of multiple sizes operate at a level in an inception network, features at different scales can be recognized. Therefore, even with a smoothness parameter of 15, Inception v4 recognized the important features that highlighted the edges and boundary locations to differentiate between a live and fake image. With a higher value of the smoothness parameter (param. alpha = 25), CNN-5 and ResNet50 performed slightly better than the Inception v4 network. However, for still higher values of param. alpha (50, 75, and 100), Inception v4 outperformed CNN-5 and ResNet50.

In terms of computation speed, ResNet50 was faster than Inceptionv4 due to the skip connections that allow propagation of the gradients through the shortcut paths by skipping a few convolutional layers. Inception v4 took approximately 20 minutes to achieve 100% accuracy in 15 epochs, whereas ResNet50 took 20 minutes to achieve 95.85% accuracy in 20 epochs. CNN-5 took approximately 5 minutes to achieve the

95.99% accuracy after 30 epochs. This CNN, being only five layers deep and with the added advantage of a reduced number of parameters due to the shared weights concept in CNNs, achieved faster training.

7.1.5 Comparison with state-of-the-art methods

We compared the performance of our approach with other state-of-the-art approaches on the NUAA dataset, such as methods that make use of enhanced local binary patterns [6], dynamic local ternary patterns [12], difference of Gaussian filtered images [10], local speed patterns [5], multiscale local binary patterns [13], and a deep CNN proposed in [18]. Compared with these methods, our Inception v4 model achieved 100% accuracy on the test set, making it highly suitable for the classification of a two-dimensional image as being real or fake.

Table 7.11. Comparison of our proposed methods with other state-of-the-art methods on the NUAA dataset.

Method	Test accuracy
ELBP [6]	95.1%
DLTP [12]	94.5%
DoG [10]	87.5%
LSP [5]	98.5%
MLBP [13]	98%
CNN [18]	99%
CNN-5 (proposed method)	95.99%
ResNet50 (proposed method)	95.85%
Inception v4 (proposed method)	100%

The Inception v4 network outperformed other state-of-the-art methods, proving that it is indeed a successful method for use in face recognition applications that use liveness detection as a precursor. The ResNet50 and CNN-5 networks did not achieve accuracies as high as the Inception v4 network, but these architectures still gave

competitive results when compared to the rest of the methods. Table 7.11 and Figure 7.5 confirm these findings.

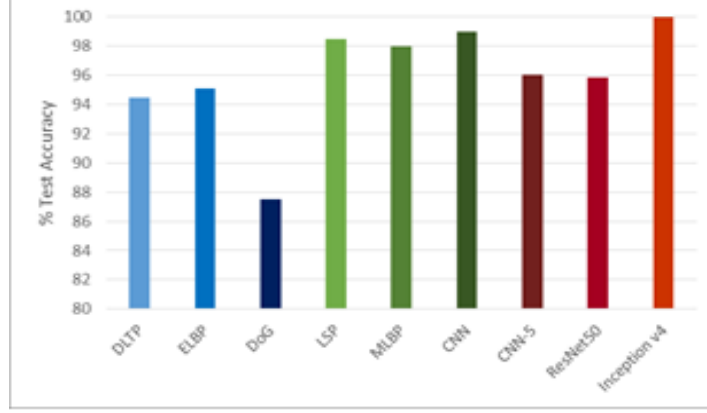


Figure 7.5. Performance comparison on the NUAA dataset.

7.2 End-to-end real-time face liveness detection using anisotropic diffusion and convolutional neural network

For the end-to-end real-time face liveness detection on a static image, we used the Replay-Attack dataset and the Replay-Mobile dataset. We describe the experimental results, and the hyperparameter settings used in the experiments. We also do a performance comparison with other existing liveness detection methods on static images.

The Replay-Attack database [14] is a 2D face spoofing attack database that consists of 1200 short video recordings of real-access and attack attempts of 50 different subjects. The frames in the video clip have a resolution of 320×240 pixels. As shown in Table 7.12, the data is divided into three sub-groups comprising of training, development, and test sets. Clients that appear in one dataset do not appear in any other dataset. Both real and attack videos were taken under two different lighting conditions, controlled and

adverse. There are 4 real and 20 attack videos per subject. The attack videos include four mobile attacks using an iPhone screen, four screen attacks using an iPad screen, and two hard-copy print attacks with each captured in two different modes, hand-based attacks, and fixed-support attacks. The training set consists of 15 subjects and 360 video clips, the development set consists of 15 subjects and 360 video clips, and the testing set consists of 20 subjects and 480 video clips.

Table 7.12. Replay-Attack dataset.

	Number of Subjects	Real-access Videos	Attack Videos	Total
Training set	15	60	300	360
Development set	15	60	300	360
Testing set	20	80	400	480
Total	50	200	1000	1200

The Replay-Mobile dataset [32] consists of 1030 video clips of photo and video attacks of 40 clients. The videos were recorded under different lighting conditions with an iPad Mini2 (running iOS) and an LG-G4 smartphone (running Android), and the frames in the video clips are of a 720×1080 resolution. The videos are grouped into training set, development set, and test set. These sets are disjoint, and therefore clients in one set do not appear in the other sets. The real-access videos were taken under five different lighting conditions (controlled, adverse, direct, lateral, diffuse). In order to produce the attacks, high-resolution photos and videos from each client were used under similar conditions as in their authentication sessions (light on and light off).

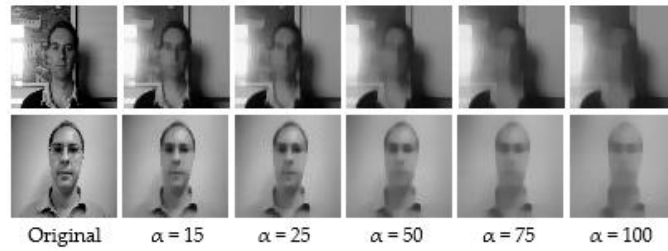
For the real-access, each client recorded 10 videos, with two videos in each of the five lighting conditions. Two kinds of attacks were performed, which include matte-screen attacks and print attacks. Each client recorded 16 attack videos, which include four mobile attacks and four tablet attacks using a mattescreen. Two print attacks each were captured by fixed-support and hand-held smartphone. Two print attacks each were captured by fixed-support and hand-held tablet. There are 12 subjects in the training set with 120 real-access and 192 attack videos, 16 subjects in the development set with 160 real-access and 256 attack videos, and 12 subjects in the test set with 110 real-access videos (since one subject was not available) and 192 attack videos (Table 7.13).

Table 7.13. Replay-Mobile dataset.

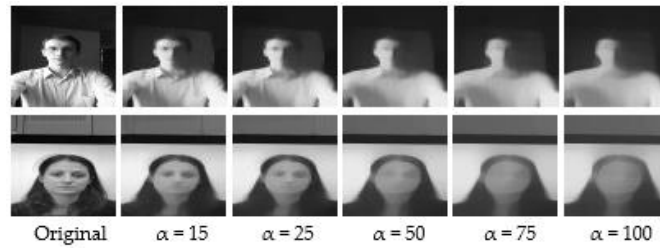
	Number of Subjects	Real-access Videos	Attack Videos	Total
Training set	12	120	192	312
Development set	16	160	256	416
Testing set	12	110	192	302
Total	40	390	640	1030

Our experiments were implemented on an Intel Xeon E3-1271 @3.60GHz with 32GB RAM PC. For diffusing the captured image in the end-to-end real-time liveness detection of a static image, we implemented the diffusion equations, as described in section 3.1 to integrate with the TensorFlow. The SCNN and Inception v4 code were also written using TensorFlow. The complete code implements the framework by doing the implementation and classification in a single step.

For our experiments using the end-to-end solution, the original images of the Replay-Attack dataset and Replay-Mobile dataset read from folders were grayscaled and then fed to the end-to-end architectures. Figure 7.6 shows some samples from the Replay-Attack dataset and the Replay-Mobile dataset, and the corresponding diffused images. The first row in both (a) and (b) shows real images from the Replay-Attack and Replay-Mobile datasets respectively, and the second row in (a) and (b) shows fake images. The first column in (a) and (b) are original non-diffused images, and the remaining columns are their diffused versions created with parameter alpha set to 15, 25, 50, 75, and 100 respectively. It can be observed that a value below 50 produces better diffused images with highlighted edges and enhanced boundaries in the real image, whereas a high value of 75/100 for the smoothness parameter alpha blurs out important edges from the images.



(a)



(b)

Figure 7.6. Sample images from the datasets and their corresponding diffused versions. (a) Images from the Replay-Attack dataset. (b) Images from the Replay-Mobile dataset. The images in the first row of both (a) and (b) are real, and the images in the second row of both (a) and (b) are fake.

We conducted numerous experiments individually on the grayscaled images of the Replay-Attack dataset and the Replay-Mobile dataset. We tuned the hyperparameters by validating on the validation (development) set during training, and the best model obtained was evaluated on the test set. We computed the test accuracy, and the Half Total Error Rate (HTER), which is defined as:

$$\text{HTER} = (\text{FAR} + \text{FRR})/2,$$

where FAR is the False Acceptance Rate, and FRR is the False Rejection Rate.

$\text{FAR} = \text{False Acceptance} / \text{Number of Impostor images}$

$\text{FRR} = \text{False Rejection} / \text{Number of Real images}$

We used 20 frames of each video clip in the training, development, and testing sets, which were resized to size 64×64 . Thus, a total of 7200 training images, 7200 development set (validation) images, and 9600 testing images were used from the Replay-Attack dataset, and 6240 training images, 8320 development set images, and 6040 test set images were used from the Replay-Mobile dataset.

7.2.1 Specialized Convolutional Neural Network (SCNN)

Using the Specialized Convolutional Neural Network (SCNN), our experiments on the Replay-Attack dataset gave test accuracy of 96.03% and HTER of 7.53%, and gave 96.21% test accuracy and 4.96% HTER on the Replay-Mobile dataset. We first trained the three-layer CNN on diffused training images created with an alpha parameter of 15 for 100 epochs by validating on the diffused validation images created with the same alpha, for tuning the hyperparameters. We experimented with various learning

rates, and the best model obtained gave an accuracy of 97.50% at a learning rate of 0.005 for the Replay-Attack dataset, and 99.62% accuracy at a learning rate of 0.0005 for the Replay-Mobile dataset. We used these pre-trained models in the CNN part of the end-to-end SCNN architecture.

We trained the SCNN network on the original training images for 30 epochs while validating on the original images of the validation set. We used the Adam optimizer, binary cross-entropy loss function, batch size of 32, and sigmoid activation function in the output layer. The training was done by setting the learning rate of the optimizer to its default value (0.001), and also using the learning rates which gave the best model of the pre-trained CNN-3, which were 0.005 for the Replay-Attack dataset, and 0.0005 for the Replay-Mobile dataset. The smoothness of diffusion (alpha) was learned by the network each time. The original image fed to the network was diffused with this learned alpha, and the diffused image was then fed to the pre-trained 3-layer CNN. The weights of the better model were saved during the 30 epochs of training. Following training, the best model weights obtained during validation were loaded, and the network model was compiled and evaluated on the test set.

Table 7.14. Highest validation accuracy (best model) obtained during validation, and the test results obtained by evaluating the best model on the test set, for the SCNN.

	Replay-Attack	Replay-Mobile
Best model (%)	95.04	98.56
Test Accuracy (%)	96.03	96.21
HTER (%)	7.53	4.96

The experimental results are summarized in Table 7.14. The real-time evaluation of a test image on a trained SCNN network takes approximately 0.021s for the Replay-Attack dataset, and 0.016s for the Replay-Mobile dataset.

7.2.2 Inception v4

For the end-to-end solution using Inception v4 network, we trained the network on the original images of the training set for 30 epochs using a fixed value for the smoothness of diffusion parameter (α), while validating on the original images of the validation (development) set for tuning the hyperparameters and selecting the best model for evaluation. We repeated the training process for various values of α and learning rates of the optimizer used. The best model obtained was then evaluated on the test set. We used the Adam optimizer, categorical cross-entropy loss function, and softmax classifier in the last stage with a batch size set to 32. The tables and plots below summarize the results obtained.

7.2.2.1 Results obtained with the Replay-Attack dataset

Table 7.15 shows the best model (highest validation accuracy) results obtained for various values of α with the Replay-Attack dataset when validating on the validation set during training for 30 epochs, for tuning the hyperparameters. The overall best model obtained was for α of 15, at the default learning rate of 0.001. The results of evaluation of the best models for each α (Table 7.15), on the test set of the Replay-Attack dataset, are shown in Table 7.16. It can be observed that the best results obtained are test accuracy of 94.77%, and HTER of 13.54% for α of 15. The plots of α vs.

test accuracy and alpha vs. HTER, according to results in Table 7.16, are shown in Figures 7.7 and 7.8.

Table 7.15. Best model obtained for each alpha while validating on the development set during training, with the Replay-Attack dataset, in Inception v4.

Alpha	15	25	50	75	100
Val. Accuracy (%)	94.81	94.25	93.40	93.44	93.31

Table 7.16. Test results obtained with the Replay-Attack dataset by evaluating the best model obtained for each alpha (in Table 7.15) on the test set, in Inception v4.

Alpha	15	25	50	75	100
Test Accuracy (%)	94.77	94.18	93.54	91.94	93.35
HTER (%)	13.54	15.01	16.25	17.31	16.01

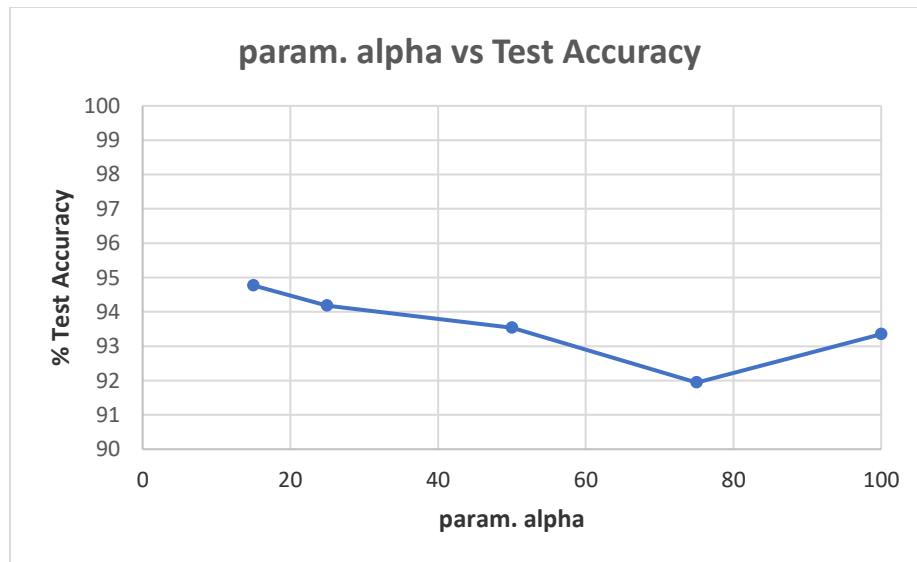


Figure 7.7. Plot showing parameter alpha vs. test accuracy (Table 7.16).

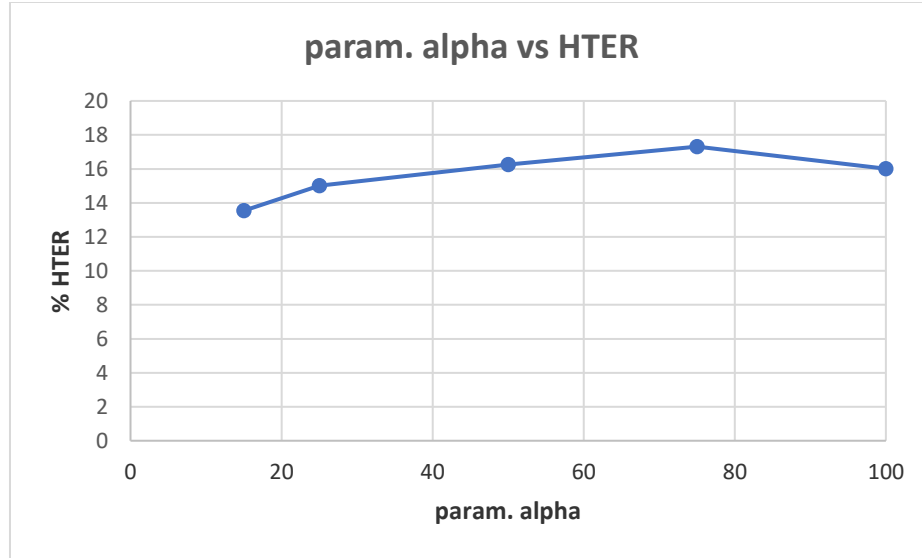


Figure 7.8. Plot showing parameter alpha vs. HTER (Table 7.16).

On a trained Inception v4 network, real-time evaluation of a test image takes approximately 0.022s for the Replay-Attack dataset.

7.2.2.2 Results obtained with the Replay-Mobile dataset

Table 7.17 shows the best model (highest validation accuracy) results obtained for various values of alpha with the Replay-Mobile dataset when validating on the validation set during training for 30 epochs, for tuning the hyperparameters. The overall best model obtained was for alpha of 15, at the default learning rate of 0.001. The results of evaluation of the best models for each alpha (Table 7.17) on the test set of the Replay-Mobile dataset are shown in Table 7.18. The best results obtained are test accuracy of 95.53%, and HTER of 5.94% for alpha of 15. The plots of alpha vs. test accuracy and alpha vs. HTER, according to results in Table 7.18, are shown in Figures 7.9 and 7.10.

Table 7.17. Best model obtained for each alpha while validating on the development set during training, with the Replay-Mobile dataset, in Inception v4.

Alpha	15	25	50	75	100
Val. Accuracy (%)	98.20	96.98	96.49	95.43	94.66

Table 7.18. Test results obtained with the Replay-Mobile dataset by evaluating the best model obtained for each alpha (in Table 7.17) on the test set, in Inception v4.

Alpha	15	25	50	75	100
Test Accuracy (%)	95.53	93.29	91.09	91.26	92.55
HTER (%)	5.94	7.90	9.07	10.91	9.69

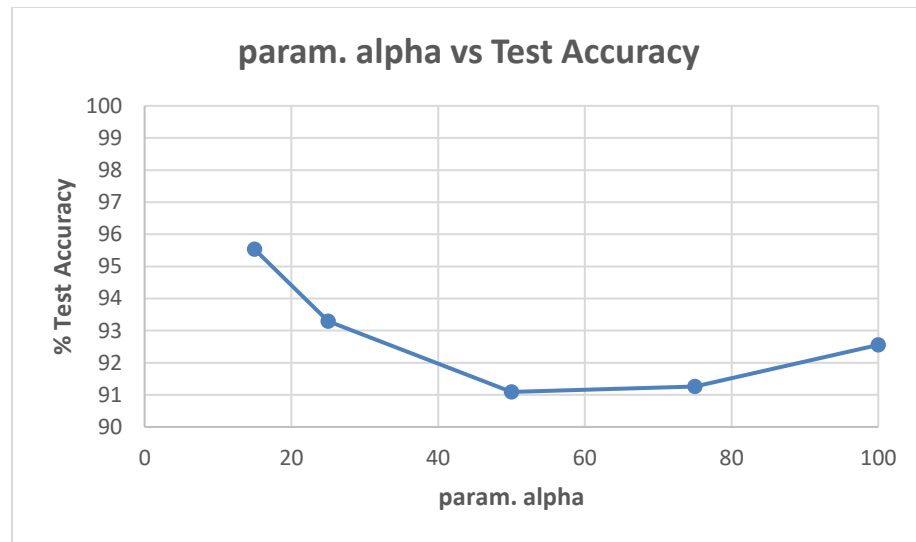


Figure 7.9. Plot showing parameter alpha vs. test accuracy (Table 7.18).

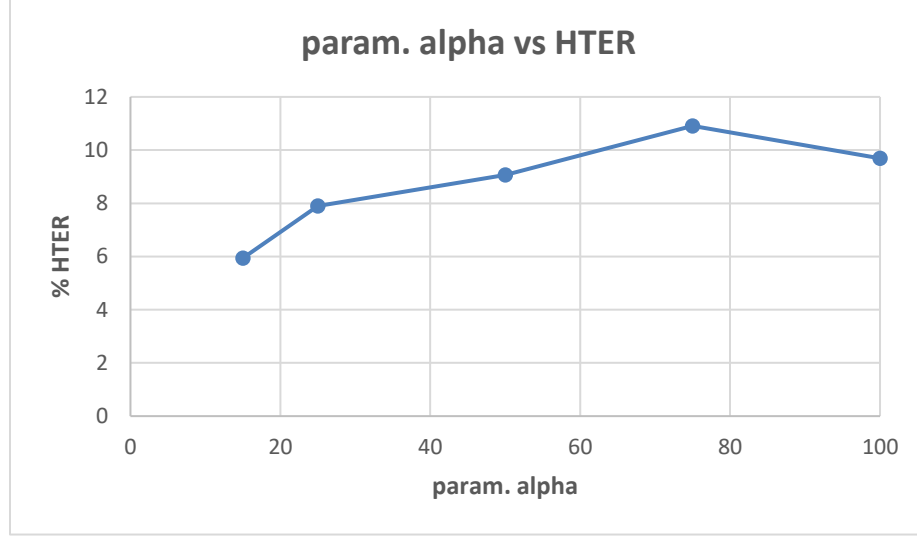


Figure 7.10. Plot showing parameter alpha vs. HTER (Table 7.18).

On a trained Inception v4 network, real-time evaluation of a test image takes approximately 0.019s for the Replay-Mobile dataset.

7.2.3 Comparison with state-of-the-art methods

The performance of our end-to-end approaches using the SCNN and Inception v4 were compared with other proposed methods on the Replay-Attack dataset and Replay-Mobile dataset, as shown in Tables 7.19 and 7.20, and Figures 7.11–7.14.

Table 7.19. Comparison with state-of-the-art methods on the Replay-Attack dataset.

Method	Test Accuracy	HTER
DLTP [12]		4.8%
Diffusion speed [5]		12.50%
Diffusion-CNN [18]		10%
LiveNet [16]		5.74%
CNN [15]	97.83%	
CNN-LBP [33]	75.25%	
LBP [34]		15.6%
IQM [34]		4.6%
SCNN (proposed method)	96.03%	7.53%
Inception v4 (proposed method)	94.77%	13.54%

Table 7.20. Comparison with state-of-the-art methods on the Replay-Mobile dataset.

Method	Test Accuracy	HTER
CNN-LBP [33]	90.52%	
LBP [34]		17.2%
IQM [34]		4.1%
SCNN (proposed method)	96.21%	4.96%
Inception v4 (proposed method)	95.53%	5.94%

In Tables 7.19 and 7.20, some of the entries are blank because the test accuracy and HTER were not reported by the authors.

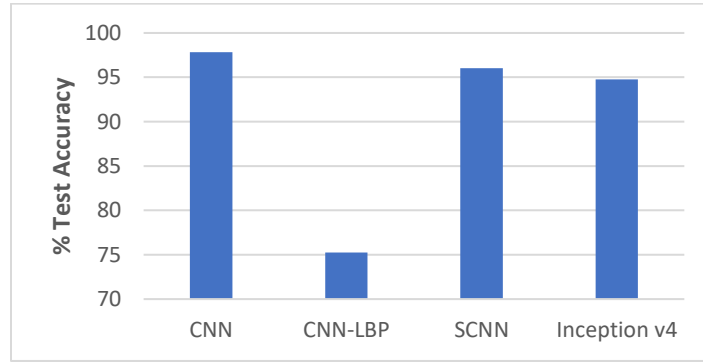


Figure 7.11. Performance comparison (% Test accuracy) of the end-to-end networks on the Replay-Attack dataset (Table 7.19).

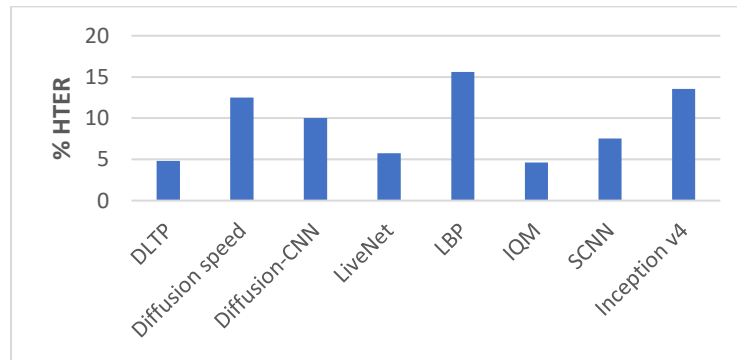


Figure 7.12. Performance comparison (% HTER) of the end-to-end networks on the Replay-Attack dataset (Table 7.19).

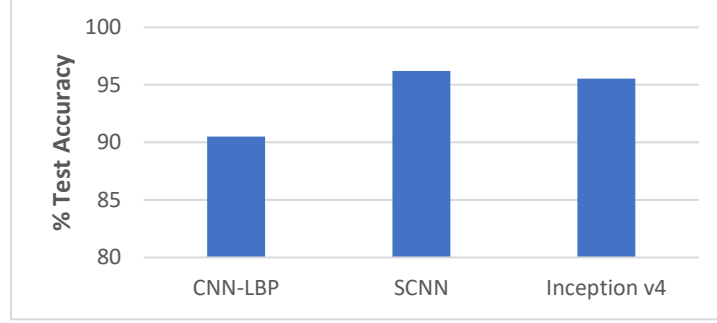


Figure 7.13. Performance comparison (% Test accuracy) of the end-to-end networks on the Replay-Mobile dataset (Table 7.20).

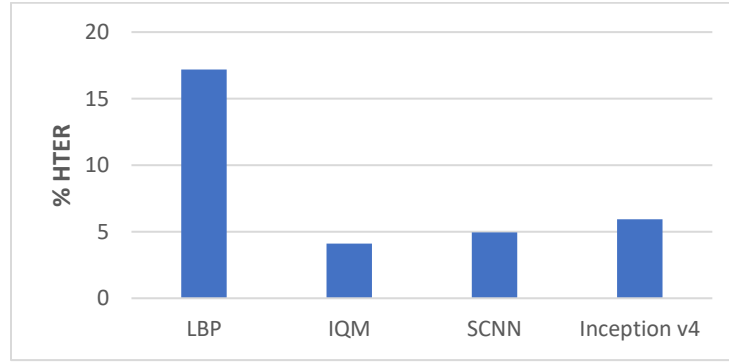


Figure 7.14. Performance comparison (% HTER) of the end-to-end networks on the Replay-Mobile dataset (Table 7.20).

Unlike the other methods, our proposed method is an end-to-end solution capable of liveness detection in real-time. Even though the accuracy and HTER are slightly below the best reported model, the tradeoff of real-time performance is an important goal. Therefore, these architectures are capable of providing an end-to-end solution with the advantage of being real-time for use in face recognition applications.

7.3 Face Liveness Detection on video sequences

For face liveness detection on video sequences using the CNN-LSTM architecture and the I3D architecture, we used the Replay-Attack dataset and Replay-Mobile dataset. We present the experimental results, and the hyperparameter settings used in the experiments. We further do a performance comparison with other existing liveness detection methods on video sequences that use these two datasets.

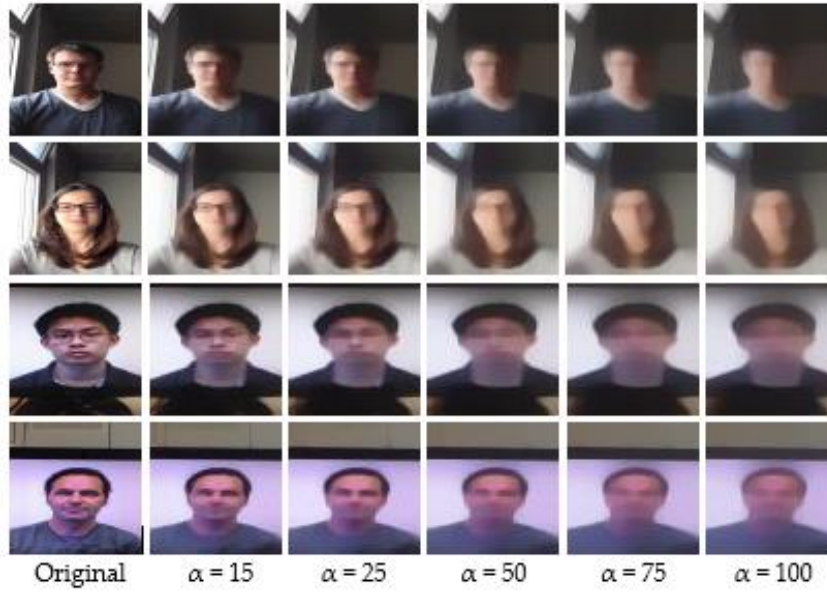
As was done for the end-to-end solution, our experiments were implemented on an Intel Xeon E3-1271 @3.60GHz with 32GB RAM PC. The diffusion equations were implemented using the Matlab code (version 9.8, R2020a) by the author of [43], and the implementations of the CNN-LSTM and I3D were carried out in Python using Keras library with TensorFlow backend.

We used 20 frames of each video clip in the training, development, and test sets. We resized the frames to a size of 64×64 , and created five sets of diffused images for our experiments with different values (15, 25, 50, 75, and 100) of the parameter α that defines the smoothness of diffusion. Figure 7.15 below shows some samples from the Replay-Attack and Replay-Mobile datasets, and their corresponding diffused versions. The first two rows in both (a) and (b) show real images from the Replay-Attack and Replay-Mobile datasets respectively, and the second and third rows in (a) and (b) show fake images in the datasets. The images in the first column in (a) and (b) are original non-diffused images, and the images in the remaining columns are their diffused versions created with the parameter α set to 15, 25, 50, 75, and 100, respectively. We tested our proposed framework with each of these sets of diffused images. We conducted numerous experiments on the Replay-Attack dataset and Replay-Mobile dataset by

changing the hyperparameters during the learning phase.



(a)



(b)

Figure 7.15. Sample images from the datasets and their corresponding diffused versions. (a) Replay-Attack dataset. (b) Replay-Mobile dataset. The images in the first two rows of both (a) and (b) are real, and the images in the third and fourth rows of both (a) and (b) are fake.

7.3.1 CNN-LSTM

We computed the test accuracy and HTER of the CNN-LSTM architecture after training for various cases using diffused images created with param. alpha set to 15, 25, 50, 75, 100. We used the Adam optimizer, mean-squared-error loss function, and batch size was set to 32. The activation functions used were ReLU for the convolutional layers and hidden layer, and sigmoid for the output layer. The number of neurons in the hidden layer of the CNN was set to 50, and the number of cells in the LSTM was set to 60. We trained the CNN-LSTM network for 100 epochs on the training set, while validating on the validation (development) set for tuning the hyperparameters. During each epoch of training, if a better validation accuracy was obtained, the model was saved. We repeated this process for various values of learning rates. At the end of training, the saved model (i.e., the one that gave the highest validation accuracy) was loaded, and then evaluated on the test set for the test accuracy and HTER. We obtained 98.71% test accuracy and 2.77% HTER on the Replay-Attack dataset, and 95.41% test accuracy and 5.28% HTER on the Replay-Mobile dataset. The tables and plots below summarize our results.

7.3.1.1 Results obtained with the Replay-Attack dataset

Table 7.21 shows the best model (highest validation accuracy) results obtained with the Replay-Attack dataset when validating on the validation set during training for 100 epochs, for tuning the hyperparameters. The results of evaluation of the best model for each alpha in Table 7.21, on the test set of the Replay-Attack dataset, are shown in Table 7.22. It can be observed that the best results obtained are test accuracy of 98.71% and HTER of 2.77% with diffused images created with alpha of 15.

Table 7.21. Highest validation accuracy (best model) obtained for different values of learning rate by validating on diffused validation images created with each alpha, during training, with Replay-Attack dataset, in CNN-LSTM (the best model for each alpha, and without diffusion, is highlighted in bold).

Alpha	Learning rate				
	0.001	0.002	0.005	0.0008	0.0005
15	98.22	98.90	95.24	97.46	97.18
25	99.13	95.56	94.74	98.89	97.78
50	96.00	98.58	96.71	98.07	97.51
75	97.56	96.11	96.53	96.17	97.50
100	96.94	95.04	95.53	98.61	98.33
w/o diffusion	97.28	95.46	96.67	97.94	97.54

Table 7.22. Test accuracies and HTER obtained by evaluating the best model of each alpha (highlighted in Table 7.21) on the test set (the highest test accuracy and lowest HTER are indicated in bold).

Alpha	15	25	50	75	100	Without diffusion
Test Accuracy (%)	98.71	97.35	97.65	95.54	96.77	96.27
HTER (%)	2.77	5.09	4.41	11.29	6.21	8.31

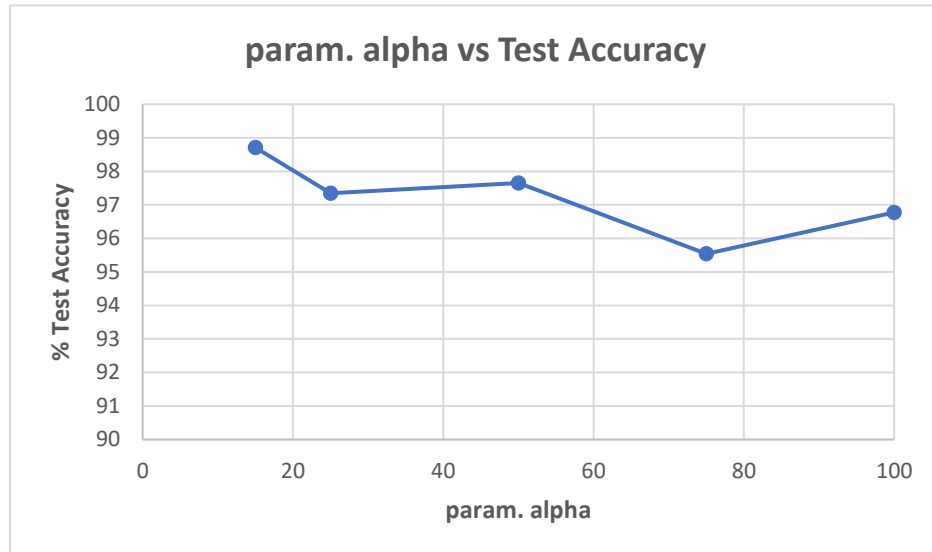


Figure 7.16. Plot showing parameter alpha vs. test accuracy (Table 7.22).

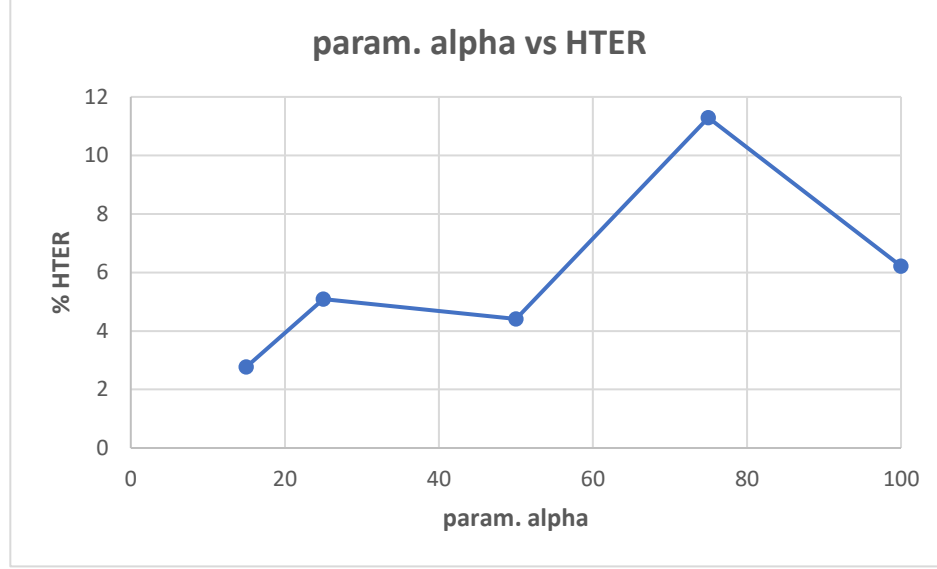


Figure 7.17. Plot showing parameter alpha vs. HTER (Table 7.22).

The plots of alpha vs. test accuracy and alpha vs. HTER, according to the results in Table 7.22, are shown in Figures 7.16 and 7.17.

We performed experiments without diffusion, i.e., by feeding the original non-diffused images directly to the network. The best validation accuracy obtained while validating on the validation set during training was 97.94%, as shown in Table 7.21. This model, when loaded and evaluated on the non-diffused original images of the test set, gave 96.27% test accuracy and 8.31% HTER (Table 7.22). Therefore, by applying diffusion, there is a significant improvement in accuracy and HTER by 2.44% and 5.54%, respectively.

We also computed test results by training the network on the train set, and validating on the testing set instead of the validation set. Tables 7.23, 7.24, and Figure 7.18 show the results obtained.

Table 7.23. Highest validation accuracy (best model) obtained for different values of learning rate by validating on diffused test images created with each alpha, during training, with Replay-Attack dataset, in CNN-LSTM (the best model for each alpha, and without diffusion, is highlighted in bold).

Alpha	Learning rate				
	0.001	0.002	0.005	0.0008	0.0005
15	97.92	97.05	97.47	96.56	97.25
25	98.40	97.34	92.73	98.78	98.65
50	96.65	95.50	89.49	97.09	99.17
75	94.82	96.07	94.30	96.75	95.76
100	96.67	96.18	97.33	96.68	98.51
w/o diffusion	98.42	94.36	83.33	98.75	97.39

Table 7.24. HTER obtained by evaluating the best model of each alpha (in Table 7.23) on the test set (the overall lowest HTER is highlighted in bold).

Alpha	15	25	50	75	100	Without Diffusion
HTER (%)	6.25	1.73	2.0	6.97	2.97	3.75

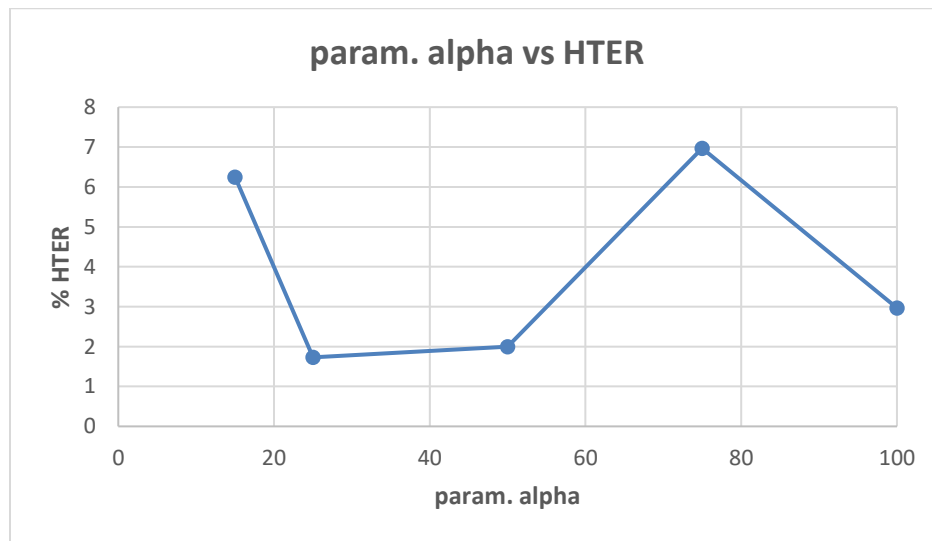


Figure 7.18. Plot showing parameter alpha vs. HTER (Table 7.24).

7.3.1.2 Results obtained with the Replay-Mobile dataset

Table 7.25 shows the best model (highest validation accuracy) results obtained with the Replay-Mobile dataset when validating on the validation set during training for 100 epochs, for tuning the hyperparameters. The results of evaluation of the best model for each alpha (Table 7.25) on the test set of the Replay-Mobile dataset, are shown in Table 7.26, and the corresponding plots are shown in Figures 7.19 and 7.20. The best results obtained are test accuracy of 95.41%, and HTER of 5.28%, for alpha of 100 and 75.

Table 7.25. Highest validation accuracy (best model) obtained for different values of learning rate by validating on diffused validation images created with each alpha, during training, with Replay-Mobile dataset, in CNN-LSTM (the best model for each alpha, and without diffusion, is highlighted in bold).

Alpha	Learning rate				
	0.001	0.002	0.005	0.0008	0.0005
15	97.81	95.89	96.83	96.57	97.73
25	98.26	95.91	97.00	97.34	97.40
50	97.40	95.04	93.31	97.27	98.08
75	98.59	96.39	97.31	98.03	97.34
100	98.91	96.71	95.11	97.70	97.01
w/o diffusion	98.97	97.08	88.95	97.88	97.42

Table 7.26. Test accuracies and HTER obtained by evaluating the best model of each alpha (highlighted in Table 7.25) on the test set (the highest test accuracy and lowest HTER are indicated in bold).

Alpha	15	25	50	75	100	Without diffusion
Test Accuracy (%)	91.87	92.76	94.29	94.39	95.41	95.20
HTER (%)	8.33	7.90	6.41	5.28	5.55	5.91

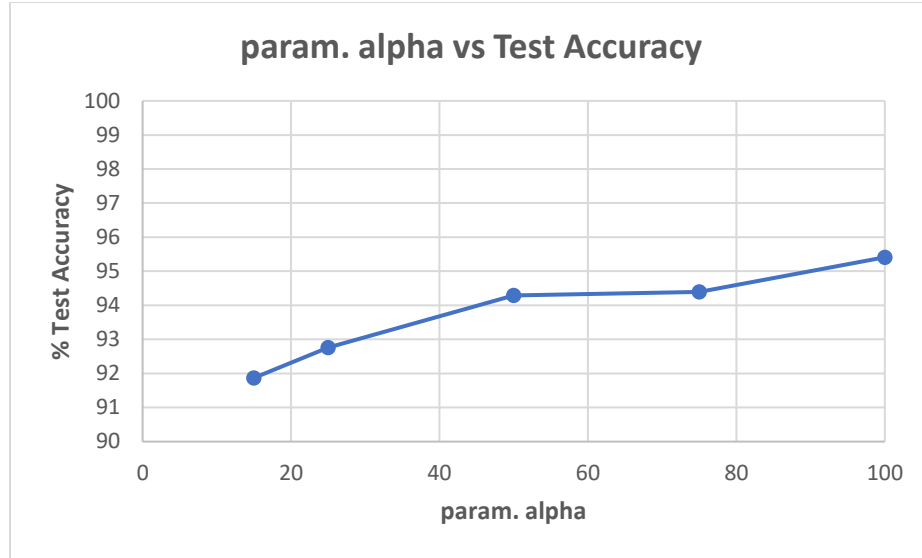


Figure 7.19. Plot showing parameter alpha vs. test accuracy (Table 7.26).

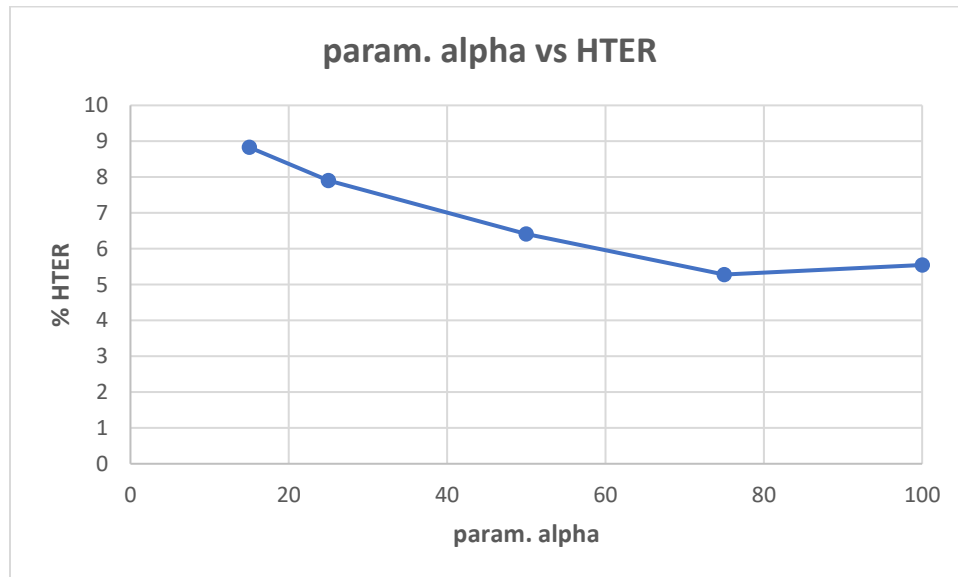


Figure 7.20. Plot showing parameter alpha vs. HTER (Table 7.26).

In the experiments performed without diffusion, i.e., by feeding the original non-diffused images directly to the network, the best validation accuracy obtained while validating on the validation set during training was 98.97%, as shown in Table 7.25. This

model, when loaded and evaluated on the original non-diffused test set images, gave 95.20% test accuracy and 5.91% HTER (Table 7.26). In this case, by applying diffusion, though there is no significant improvement as obtained with the Replay-Attack dataset, there is still a slight improvement in accuracy and HTER.

We also computed test results by training the network on the train set, and validating on the testing set. Tables 7.27, 7.28, and Figure 7.21 show the results obtained.

Table 7.27. Highest validation accuracy (best model) obtained for different values of learning rate by validating on diffused test images created with each alpha, during training, with Replay-Mobile dataset, in CNN-LSTM (the best model for each alpha, and without diffusion, is highlighted in bold).

Alpha	Learning rate				
	0.001	0.002	0.005	0.0008	0.0005
15	93.69	97.07	94.40	94.50	95.50
25	93.69	95.28	92.33	95.33	94.69
50	93.16	94.67	95.43	93.71	92.67
75	95.28	92.67	90.23	96.01	94.88
100	93.71	94.95	90.78	94.74	94.17
w/o diffusion	93.26	94.04	92.38	92.76	95.53

Table 7.28. HTER obtained by evaluating the best model of each alpha (in Table 7.27) on the test set (the overall lowest HTER is highlighted in bold).

Alpha	15	25	50	75	100	Without Diffusion
HTER (%)	3.47	5.83	4.97	4.68	5.77	5.06

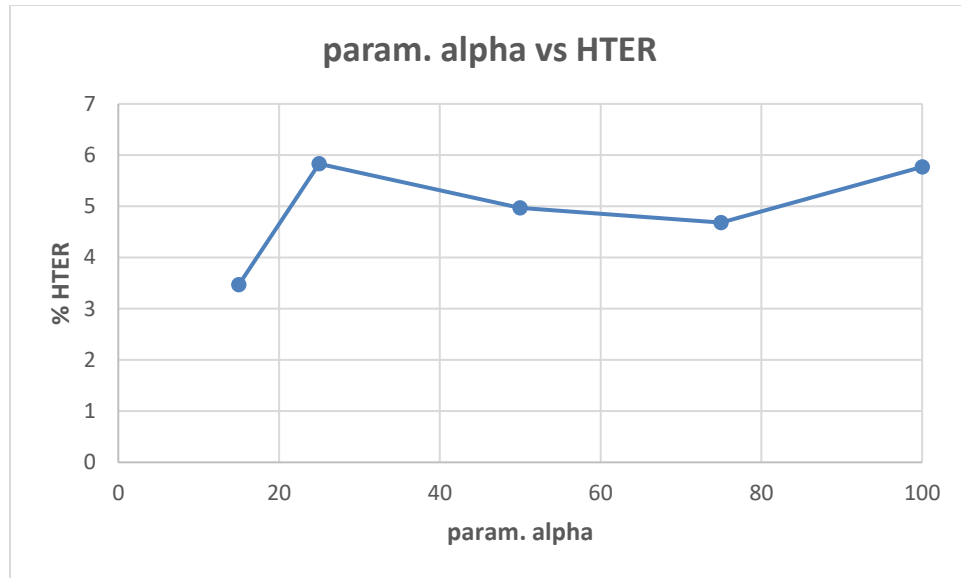


Figure 7.21. Plot showing parameter alpha vs. HTER (Table 7.28).

7.3.1.3 Comparison of results with and without diffusion

The charts below (Figures 7.22 and 7.23) show the best results with diffusion and the results without diffusion for both the Replay-Attack dataset (Table 7.22) and the Replay-Mobile dataset (Table 7.26).

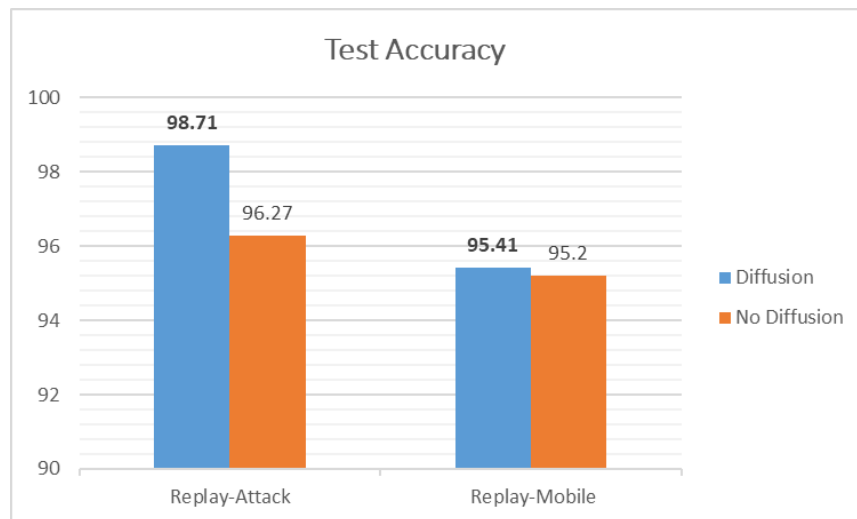


Figure 7.22. Test accuracy (%) obtained with and without diffusion.

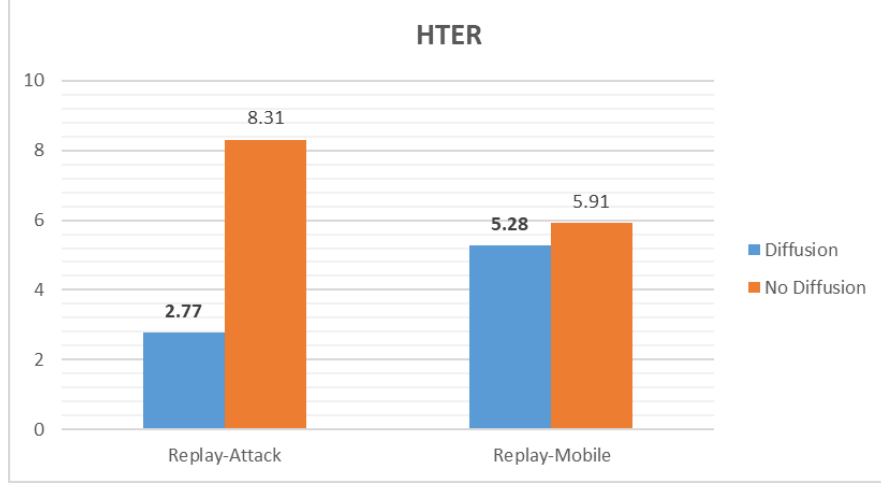


Figure 7.23. HTER (%) obtained with and without diffusion.

The training of the CNN-LSTM framework is very fast, as it takes only about 15 min for the Replay-Attack dataset, and 14 min for the Replay-Mobile dataset. Evaluation takes about 5s for the Replay-Attack dataset, and 4.1s for the Replay-Mobile dataset.

7.3.1.4 Comparison with state-of-the-art methods

The performance of our proposed approach was compared with state-of-the-art methods for liveness detection on the Replay-Attack dataset, as shown in Table 7.29, and Figures 7.24 and 7.25. In [22], an HTER of 7.60% was achieved, and in [23], they achieved HTER of 6.62% using LBP and SVM, and HTER of 1.25% using HOOF and LDA. In [27], 5.38% HTER was achieved using the multi-scale analysis, and in [28], the MHI-LBP gave HTER of 3.9% and MHI-CNN gave HTER of 4.5%. The CNN LBP-TOP method proposed in [29] gave HTER of 4.7%. In the work proposed in [31] that makes use of motion cues for face anti-spoofing, 100% and 96.47% accuracy were achieved when tested separately on Replay-Attack (controlled) and Replay-Attack (adverse) test

sets. The work proposed in [34] reported HTER of 13.2%. We achieved 98.71% accuracy and 2.77% HTER when we tested our proposed framework on the entire testing set of the Replay-Attack database.

Table 7.29. Comparison with state-of-the-art methods on the Replay-Attack dataset.

Method	Test Accuracy	HTER
LBP-TOP [22]		7.60%
LBP and SVM [23]		6.62%
HOOF and LDA [23]		1.25%
Multi-scale analysis [27]		5.38%
MHI-LBP [28]		3.9%
MHI-CNN [28]		4.5%
CNN LBP-TOP [29]		4.7%
CNN-LSTM (C) [31]	100%	
CNN-LSTM (A) [31]	96.47%	
Motion-based approach [34]		13.2%
CNN-LSTM (proposed method)	98.71%	2.77%

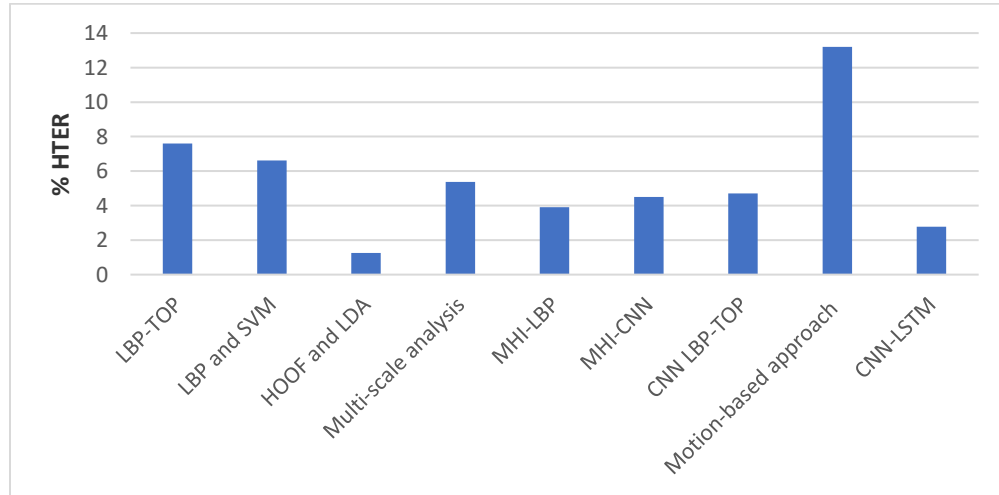


Figure 7.24. Performance comparison (HTER) on the Replay-Attack dataset (Table 7.29).

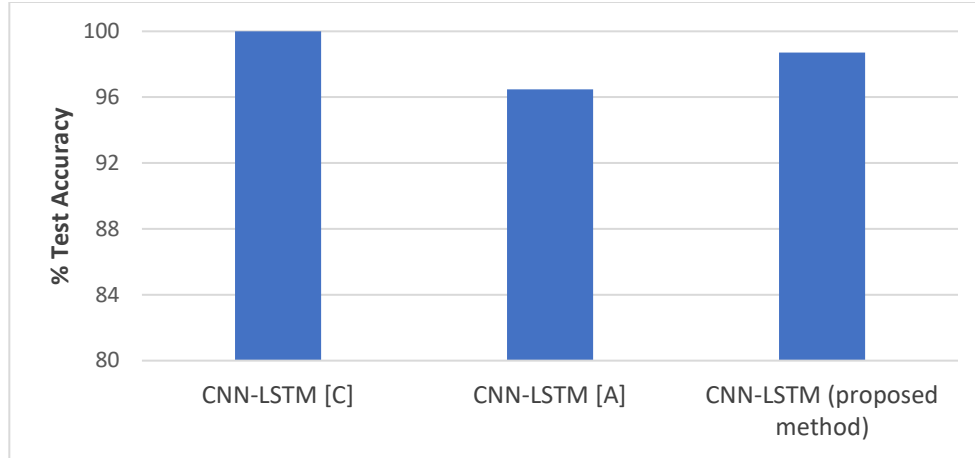


Figure 7.25. Performance comparison (Test Accuracy) on the Replay-Attack dataset (Table 7.29).

Table 7.30 and Figure 7.26 below shows the comparison of our method with state-of-the-art methods on the Replay-Mobile dataset. In [32], HTER of 7.80% was achieved for IQM, and HTER of 9.13% was achieved for Gabor-jets. In the work proposed in [34], HTER of 10.4% was achieved. The anomaly detection approach proposed in [37] gave HTER of 13.70%, and the one-class multiple kernel fusion regression approach proposed in [38] gave 13.64% HTER.

Table 7.30. Comparison with state-of-the-art methods on the Replay-Mobile dataset.

Method	Test Accuracy	HTER
IQM [32]		7.80%
Gabor-jets [32]		9.13%
Motion-based approach [34]		10.4%
Anomaly detection [37]		13.70%
Kernel fusion regression [38]		13.64%
CNN-LSTM (proposed method)	95.41%	5.28%

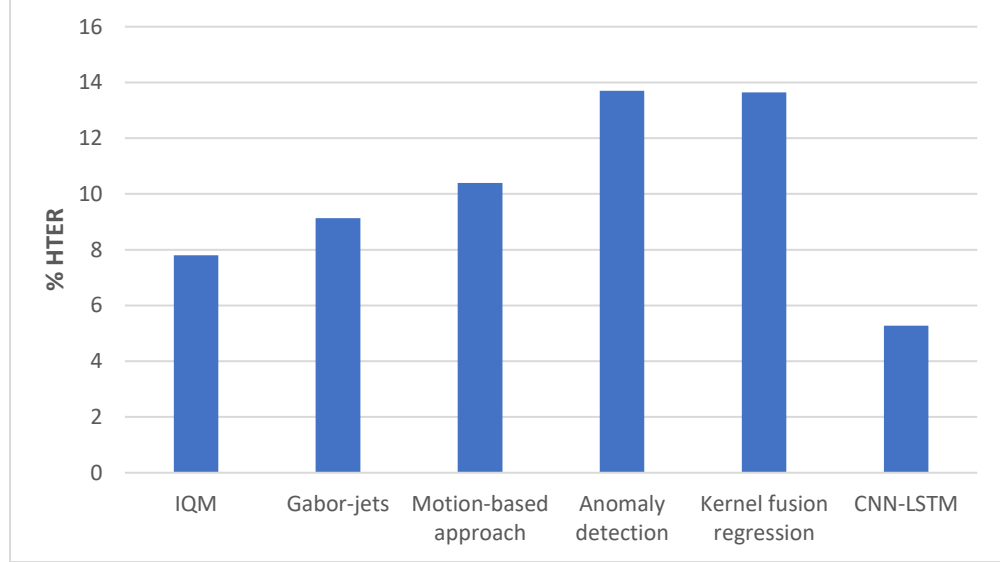


Figure 7.26. Performance comparison (HTER) on the Replay-Mobile dataset (Table 7.30).

In tables 7.29 and 7.30, some of the entries are blank because the test accuracy and HTER were not reported by the authors. As shown in Tables 7.29 and 7.30, our architecture gave very competitive results when compared to other state-of-the-art methods in the literature for the Replay-Attack dataset, and gave the lowest HTER when compared to state-of-the-art methods for the Replay-Mobile dataset. This proves that it is an efficient solution for use in face recognition applications that require liveness detection on video frames for anti-spoofing.

7.3.2 Two-Stream Inflated 3D ConvNet (I3D)

We experimented with the I3D architecture [77] on the Replay-Attack dataset and Replay-Mobile dataset. As was done for face liveness detection using the CNN-LSTM architecture, we performed experiments on the five sets of diffused images that were created with various values of the parameter alpha. We present the experimental results in tables 7.31 and 7.32.

Table 7.31. Test accuracy and HTER obtained with the Replay-Attack dataset.

Alpha	15	25	50	75	100
Test Accuracy (%)	100	99.17	99.79	98.75	98.12
HTER (%)	0.12	0.99	1.88	2.12	11.75

Table 7.32. Test accuracy and HTER obtained with the Replay-Mobile dataset.

Alpha	15	25	50	75	100
Test Accuracy (%)	99.34	97.35	99.67	99.67	99.67
HTER (%)	0.91	5.00	2.34	1.04	0.45

We obtained 100% accuracy and 0.12% HTER with the Replay-Attack dataset, and 99.67% accuracy and 0.45% HTER with the Replay-Mobile dataset.

The graphs below shows the performance comparison of the I3D architecture with the CNN-LSTM.

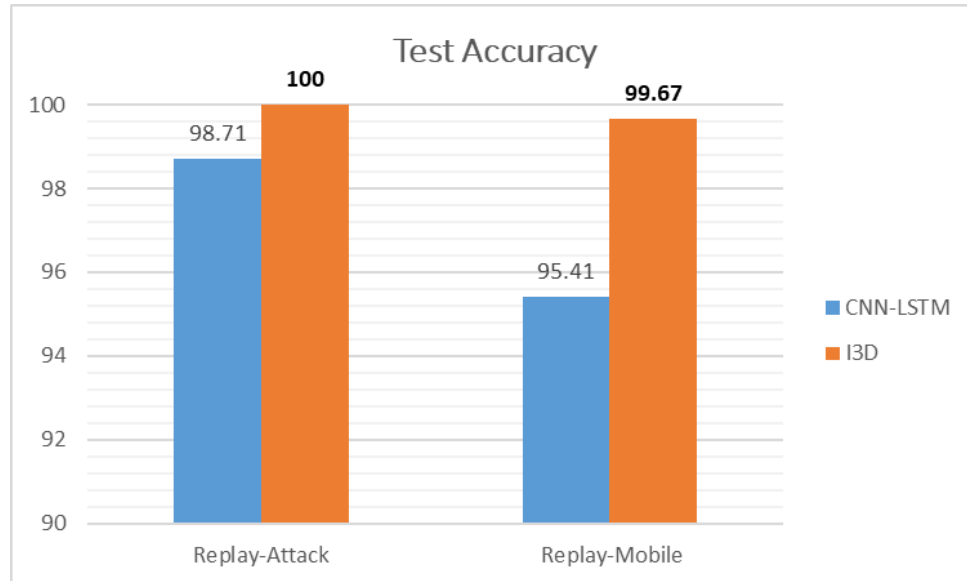


Figure 7.27. Comparison of test accuracies obtained with I3D and CNN-LSTM.

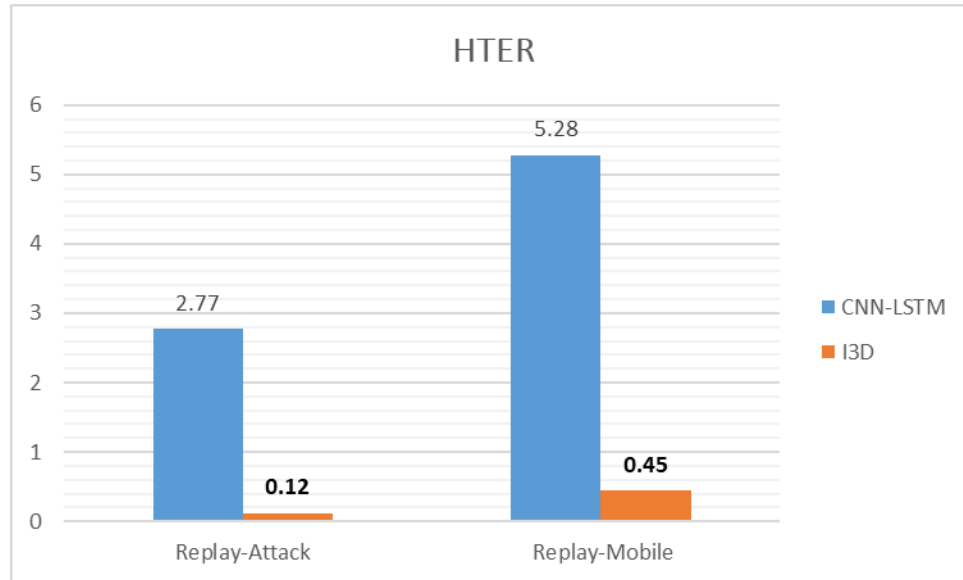


Figure 7.28. Comparison of HTER obtained with I3D and CNN-LSTM.

CHAPTER 8: CONCLUSION

We have developed face liveness detection architectures for static as well as video sequences. For static images, we first applied nonlinear diffusion based on an additive operator splitting scheme and a block-solver tri-diagonal matrix algorithm, to the captured images. This generates diffused images, with edge information and surface texture of real images more pronounced than fake ones. These diffused images were then fed to a CNN to extract the complex and deep features, for classifying the images as real or fake. Our implementation with the deep CNN architecture, Inception v4, on the NUAA dataset gave 100% accuracy, showing that it can efficiently classify a two-dimensional diffused image as real or fake. An analysis of the various test results obtained showed that the smoothness of the diffusion is an important factor in determining the liveliness of the captured image. We determined that better results are obtained with lower values of the smoothness parameter since higher values of this parameter blur out important information from the image. For static images, we also developed an end-to-end real-time solution to the face liveness detection problem. Previous best approaches relied on a separate preprocessing step for creating diffused images whereas our work has integrated the diffusion process as well as the face liveness classification into a single TensorFlow application. We used a Specialized CNN (SCNN) and the Inception v4 network in conjunction with the anisotropic diffusion for liveness classification. Our proposed

framework for both the end-to-end solutions produced promising results on the Replay-Attack and Replay-Mobile datasets, compared favourably to other state-of-the-art methods in the literature, and it has the added advantage of accomplishing face liveness detection in real-time.

We also proposed a solution for face liveness detection on video sequences using a combination of diffusion, CNN, and LSTM. We first applied nonlinear diffusion to each frame in the sequence, which makes the edge information and surface texture of a real image more pronounced than that of a fake image. The CNN extracts the complex and deep spatial features of each frame, and the LSTM captures the temporal dynamics in the sequence in order to classify the video sequence as real or fake. Our architecture produced competitive results compared to other state-of-the-art methods in the literature. Our experiments with the Replay-Attack dataset produced 98.71% test accuracy and an HTER of 2.77%, and our experiments on the Replay-Mobile dataset gave test accuracy of 95.41% and HTER of 5.28%, proving that it is a successful method for liveness detection of sequences. Our experiments using the Two-Stream Inflated 3D ConvNet (I3D) architecture for liveness detection on video sequences gave 100% test accuracy and 0.12% HTER on the Replay-Attack dataset, and 99.67% test accuracy and 0.45% HTER on the Replay-Mobile dataset.

REFERENCES

- [1] I. A. Rassan and H. AlShaher, “Securing Mobile Cloud Computing using Biometric Authentication (SMCBA),” in *2014 Int. Conf. Comput. Sci. and Comput. Intell.*, Las Vegas, NV, USA, Mar. 10-13, 2014, vol. 1, pp. 157-161, doi: 10.1109/CSCI.2014.33.
- [2] J. Galbally, S. Marcel, and J. Fierrez, “Biometric antispoofing methods: A survey in face recognition, *IEEE Access*, vol. 2, pp. 1530-1552, 2014, doi: 10.1109/ACCESS.2014.2381273.
- [3] C. Lai and C. Tai, “A smart spoofing face detector by display features analysis,” *Sensors*, vol. 16, no. 7, pp. 1136, July 2016, doi: 10.3390/s16071136. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4970178/>. [Accessed: May 16, 2019].
- [4] P. P. K. Chan et al., “Face liveness detection using a flash against 2D spoofing attack,” *IEEE Trans. Inf. Forensics and Secur.*, vol. 13, no. 2, pp. 521–534, Feb. 2018, doi: 10.1109/TIFS.2017.2758748.
- [5] W. Kim, S. Suh, and J. Han, “Face liveness detection from a single image via diffusion speed model,” *IEEE Trans. Image Process.*, vol. 24, no. 8, pp. 2456–2465, Aug. 2015, doi: 10.1109/TIP.2015.2422574.

- [6] X. Liu, R. Lu, and W. Liu, "Face liveness detection based on enhanced local binary patterns," in *2017 Chinese Automat. Congress (CAC)*, Jinan, China, Oct. 20-22, 2017, pp. 6301–6305, doi: 10.1109/CAC.2017.8243913.
- [7] D. Das and S. Chakraborty, "Face liveness detection based on frequency and micro-texture analysis," in *Int. Conf. Advances in Eng. & Technol. Res. (ICAETR-2014)*, Unnao, India, Aug. 1-2, 2014, pp. 1–4, doi: 10.1109/ICAETR.2014.7012923.
- [8] C. Yeh and H. Chang, "Face liveness detection with feature discrimination between sharpness and blurriness," in *15th IAPR Int. Conf. Mach. Vision Appl. (MVA)*, Nagoya, Japan, May 8-12, 2017, pp. 398–401, doi: 10.23919/MVA.2017.7986885.
- [9] X. Luan, H. Wang, W. Ou, and L. Liu, "Face liveness detection with recaptured feature extraction," in *Int. Conf. Secur., Pattern Anal., and Cybern. (SPAC)*, Shenzhen, China, Dec. 15-17, 2017, pp. 429–432, doi: 10.1109/SPAC.2017.8304317.
- [10] X. Tan, Y. Li, J. Liu, and L. Jiang, "Face liveness detection from a single image with sparse low rank bilinear discriminative model," in *Proc. 11th Eur. Conf. Comput. Vision (ECCV 2010)*, Heraklion, Crete, Greece, Sept. 5-11, 2010, pp. 504–517.
- [11] Z. Zhang, J. Yan, S. Liu, Z. Lei, D. Yi, and S. Z. Li, "A face antispoofing database with diverse attacks," in *2012 5th IAPR Int. Conf. Biometrics (ICB)*, New Delhi, India, Mar. 29 – Apr. 1, 2012, pp. 26-31, doi: 10.1109/ICB.2012.6199754.
- [12] S. Parveen, S.M.S. Ahmad, N. H. Abbas, W. A. W. Adnan, M. Hanafi, and N. Naeem, "Face liveness detection using Dynamic Local Ternary Pattern (DLTP)," *Computers*, vol. 5, no. 2, 10, May 2016, doi: 10.3390/computers5020010.

- [13] J. Maatta, A. Hajid, and M. Pietikainen. “Face spoofing detection from single images using micro-texture analysis,” in 2011 *Int. Joint Conf. Biometrics (IJCB)*, Washington, DC, USA, Oct. 11-13, 2011, pp. 1–7, doi: 10.1109/IJCB.2011.6117510.
- [14] I. Chingovska, A. Anjos, and S. Marcel, “On the effectiveness of local binary patterns in face anti-spoofing,” in 2012 *BIOSIG – Proc. of the Int. Conf. of Biometrics Special Interest Group (BIOSIG)*, Darmstadt, Germany, Sept. 6-7, 2012, pp. 1-7.
- [15] D. Gragnaniello, C. Sansone, G. Poggi, and L. Verdoliva, “Biometric spoofing detection by a domain-aware convolutional neural network,” in 2016 *12th Int. Conf. on Signal-Image Technology & Internet-Based Systems (SITIS)*, Naples, Italy, Nov. 28 – Dec. 1, 2016, pp. 193–198, doi:10.1109/SITIS.2016.38.
- [16] Y. A. U. Rehman, L. M. Po, and M. Liu, “LiveNet: Improving features generalization for face liveness detection using convolution neural networks,” *Expert Syst. Appl.*, vol. 108, pp. 159–169, Oct. 2018, doi: 10.1016/j.eswa.2018.05.004.
- [17] A. Alotaibi and A. Mahmood, “Enhancing computer vision to detect face spoofing attack utilizing a single frame from a replay video attack using deep learning,” in 2016 *Int. Conf. Optoelectronics and Image Process. (ICOIP)*, Warsaw, Poland, June 10-12, 2016, pp. 1–5, doi: 10.1109/OPTIP.2016.7528488.
- [18] A. Alotaibi and A. Mahmood, “Deep face liveness detection based on nonlinear diffusion using convolutional neural network,” *SIViP*, vol. 11, no. 4, pp. 713–720, May 2017, doi: 10.1007/s11760-016-1014-2.
- [19] S. Kim, S. Yu, K. Kim, Y. Ban, and S. Lee, “Face liveness detection using variable focusing,” in 2013 *Int. Conf. Biometrics (ICB)*, Madrid, Spain, June 4-7, 2013, pp. 1-6, doi: 10.1109/ICB.2013.6613002.

- [20] J. Yan, Z. Zhang, Z. Lei, D. Yi, and S. Z. Li, "Face liveness detection by exploring multiple scenic clues," in *2012 12th Int. Conf. Control Automat. Robot. & Vision (ICARCV)*, Guangzhou, China, Dec. 5-7, 2012, pp. 188-193, doi: 10.1109/ICARV.2012.6485156.
- [21] T. Wang, J. Yang, Z. Lei, S. Liao, and S. Z. Li, "Face liveness detection using 3D structure recovered from a single camera," in *2013 Int. Conf. Biometrics (ICB)*, Madrid, Spain, June 4-7, 2013, pp. 1-6, doi: 10.1109/ICB.2013.6612957.
- [22] T. de Freitas Pereira, A. Anjos, J. Mario de Martino, and S. Marcel, "LBP – TOP based countermeasure against face spoofing attacks," in *ACCV'12: Proc. 11th Int. Conf. Comput. Vision – Volume Part I*, Daejeon Korea, Nov. 5-6, 2012, pp. 121-132.
- [23] S. Bhardwaj, T. I. Dhamecha, M. Vatsa, and R. Singh, "Computationally efficient face spoofing detection with motion magnification," in *2013 IEEE Conf. Comput. Vision and Pattern Recognit. Workshops*, Portland, OR, USA, June 23-28, 2013, pp. 105–110, doi: 10.1109/CVPRW.2013.23.
- [24] D. Tang, Z. Zhou, Y. Zhang, and K. Zhang, "Face Flashing: a secure liveness detection protocol based on light reflections," Aug. 2018. [Online]. Available: <https://arxiv.org/pdf/1801.01949.pdf>. [Accessed: May 28, 2019].
- [25] A. Anjos, M. M. Chakka, and S. Marcel, "Motion-based counter-measures to photo attacks in face recognition," *IET Biometrics*, vol. 3, no. 3, pp. 147-158, Sept. 2014, doi: 10.1049/iet-bmt.2012.0071.
- [26] D. Wen, H. Han, and A. K. Jain, "Face spoof detection with image distortion analysis," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 4, pp. 746-761, April 2015, doi: 10.1109/TIFS.2015.2400395.

- [27] C. Yeh and H. Chang, "Face liveness detection based on perpetual image quality assessment features with multi-scale analysis," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Tahoe, NV, USA, Mar. 12-15, 2018, pp. 49-56, doi: 10.1109/WACV.2018.00012.
- [28] S. Pan and F. Deravi, "Spatio-temporal texture features for presentation attack detection in biometric systems," in *2019 8th International Conference on Emerging Security Technologies (EST)*, Colchester, United Kingdom, July 22-24, 2019, pp. 1-6, doi: 10.1109/EST.2019.8806220.
- [29] 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)*, Chengdu, China, June 2-4, 2017, pp. 234-238, doi: 10.1109/ICIVC.2017.7984552.
- [30] Z. Xu, S. Li, and W. Deng, "Learning temporal features using LSTM-CNN architecture for face anti-spoofing," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Kuala Lumpur, Malaysia, Nov. 3-6, 2015, pp. 141-145, doi: 10.1109/ACPR.2015.7486482.
- [31] X. Tu, H. Zhang, M. Zie, Y. Luo, Y., Zhang, and Z. Ma. "Enhance the motion cues for face anti-spoofing using CNN-LSTM architecture," Jan. 2019. [Online]. Available: <https://arxiv.org/pdf/1901.05635.pdf>. [Accessed: July 16, 2019].
- [32] A. Costa-Pazo, S. Bhattacharjee, E. Vasquez-Fernandez, and S. Marcel, "The REPLAY-MOBILE face presentation-attack database," in *2016 Int. Conf. of the Biometrics Special Interest Group (BIOSIG)*, Darmstadt, Germany, Sept. 21-23, 2016, pp. 1-7, doi: 10.1109/BIOSIG.2016.7736936.

- [33] P. K. Das, B. Hu, C. Liu, K. Cui, P. Ranjan, and G. Xiong, "A new approach for face anti-spoofing using handcrafted and deep network features," in *2019 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, Zhengzhou, China, Nov. 6–8, 2019; pp. 33–38, doi:10.1109/SOLI48380.2019.8955089.
- [34] O. Nikisins, A. Mohammadi, A. Anjos, and S. Marcel, "On effectiveness of anomaly detection approaches against unseen presentation attacks in face anti-spoofing," in *2018 Int. Conf. on Biometrics (ICB)*, Gold Coast, QLD, Australia, Feb. 20–23, 2018, pp. 75–81, doi:10.1109/ICB2018.2018.00022.
- [35] J. Galbally, S. Marcel, and J. Fierrez, "Image quality assessment for fake biometric detection: Application to iris, fingerprint, and face recognition," *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 710–724, Feb. 2014, doi: 10.1109/TIP.2013.2292332.
- [36] A. Anjos and S. Marcel, "Counter-measures to photo attacks in face recognition: A public database and a baseline," in *2011 Int. Joint Conf. Biometrics (IJCB)*, Washington, DC, USA, Oct. 11–13, 2011, pp.1–7, doi: 10.1109/IJCB.2011.6117503.
- [37] S. Fatemifar, S. R. Arashloo, M. Awais, and J. Kittler, "Spoofing attack detection by anomaly detection," in *2019 IEEE Int. Conf. Acoustics, Speech and Signal Process. (ICASSP)*, Brighton, UK, May 12–17, 2019, pp. 8464–8468, doi:10.1109/ICASSP.2019.8682253.

- [38] S. R. Arashloo, “Unseen face presentation attack detection using class-specific sparse one-class multiple kernel fusion regression,” Dec. 2019. [Online]. Available: <https://arxiv.org/pdf/1912.13276.pdf>. [Accessed: August 29, 2020].
- [39] E. Erdem, “Linear diffusion,” 2012. [Online]. Available: <https://web.cs.hacettepe.edu.tr/~erkut/bil717.s12/w03-lineardif.pdf>. [Accessed: Sept. 19, 2019].
- [40] M. R. Hajiaboli, M. O. Ahmad, and C. Wang, “An edge-adapting Laplacian kernel for nonlinear diffusion filters,” *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1561–1572, April 2012, doi: 10.1109/TIP.2011.2172803.
- [41] P. Perona and J. Malik, “Scale-space and edge detection using anisotropic diffusion,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, July 1990, doi: 10.1109/34.56205.
- [42] J. Weickert, B. M. T. H. Romeny, and M. A. Viergever, “Efficient and reliable schemes for nonlinear diffusion filtering,” *IEEE Trans. Image Process.*, vol. 7, no. 3, pp. 398–410, Mar. 1998, doi: 10.1109/83.661190.
- [43] J. Ralli. “PDE based image diffusion and AOS,” 2014. [Online]. Available: http://www.jarnoralli.fi/joomla/images/pdf/diffusion_and_aos.pdf. [Accessed: Aug. 28, 2017].
- [44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.
- [45] P. Li, J. Li, and G. Wang, “Application of convolution neural network in natural language processing,” in *2018 15th Int. Comput. Conf. Wavelet Act. Media Technol.*

- and Inf. Process. (ICCWAMTIP)*, Chengdu, China, Dec. 14-16, 2018, pp. 120–122, doi: 10.1109/ICCWAMTIP.2018.8632576.
- [46] Y. Hu, Y. Yi, T. Yang, and Q. Pan, “Short text classification with a convolutional neural networks based method,” in *2018 15th Int. Conf. Control, Automat., Robot., and Vision (ICARCV)*, Singapore, Singapore, Nov. 18-21, 2018, pp. 1432–1435, doi: 10.1109/ICARCV.2018.8581332.
- [47] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, “Face recognition: A convolutional neural-network approach,” *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 98–113, Jan. 1997, doi: 10.1109/72.554195.
- [48] C. Garcia and M. Delakis, “Convolutional face finder: A neural architecture for fast and robust face detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1408–1423, Nov. 2004, doi: 10.1109/TPAMI.2004.97.
- [49] H. Zhu, X. Chen, W. Dai, K. Fu, Q. Ye, and J. Jiao, “Orientation robust object detection in aerial images using deep convolutional neural network,” in *2015 IEEE Int. Conf. Image Process. (ICIP)*, Quebec City, QC, Canada, Sept. 27-30, 2015, pp. 3735–3739, doi: 10.1109/ICIP.2015.7351502.
- [50] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *2014 IEEE Conf. Comput. Vision and Pattern Recognit.*, Columbus, OH, USA, June 23-28, 2014, pp. 1725–1732, doi: 10.1109/CVPR.2014.223.
- [51] Y. Qi, Y. Wang, and Y. Liu, “Object tracking based on deep CNN feature and color feature,” in *2018 14th IEEE Int. Conf. Signal Process. (ICSP)*, Beijing, China, Aug. 12-16, 2018, pp. 469–473, doi:10.1109/ICSP.2018.8652470.

- [52] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016, doi: 10.1109/TPAMI.2015.2439281.
- [53] H. Vu, E. Cheng, R. Wilkinson, and M. Lech, “On the use of convolutional neural networks for graphical model-based human pose estimation,” in *2017 Int. Conf. Recent Adv. in Signal Process., Telecommun. & Comput. (SigTelCom)*, Da Nang, Vietnam, Jan. 9-11, 2017, pp. 88–93, doi: 10.1109/SIGTELCOM.2017.7849801.
- [54] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735-1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
- [55] L. Gao, Z. Guo, H. Zhang, X. Xu, and H. T. Shen, “Video captioning with attention-based LSTM and semantic consistency,” *IEEE Trans. Multimedia*, vol. 19, no. 9, pp. 2045-2055, Sept. 2017, doi: 10.1109/TMM.2017.2729019.
- [56] Y. Lu, C. Lu, and C. Tang, “Online video object detection using association LSTM,” in *2017 IEEE Int. Conf. Comput. Vision (ICCV)*, Venice, Italy, Oct. 22-29, 2017, pp. 2363-2371, doi: 10.1109/ICCV.2017.257.
- [57] A. H. Mirza and S. Cosan, “Computer network intrusion detection using sequential LSTM neural networks autoencoders,” in *2018 26th Signal Process. and Commun. Appl. Conf. (SIU)*, Izmir, Turkey, May 2-5, 2018, pp. 1-4, doi: 10.1109/SIU.2018.8404689.
- [58] C. R. Naguri and R. C. Bunescu, “Recognition of dynamic hand gestures from 3D motion data using LSTM and CNN architectures,” in *2017 16th IEEE Int. Conf. Mach. Learn. and Appl. (ICMLA)*, Dec. 18-21, 2017, Cancun, Mexico, pp. 1130-1133, doi: 10.1109/ICMLA.2017.00013.

- [59] Z. C. Lipton, J. Burkowitz, and C. Elkan. “A critical review of recurrent neural networks for sequence learning,” Oct. 2015. [Online]. Available: <https://arxiv.org/pdf/1506.00019.pdf>. [Accessed: July 5, 2019].
- [60] W. D. Mulder, S. Bethard, and M. Moens, “A survey on the application of recurrent neural networks to statistical language modeling,” *Comput. Speech Lang.*, vol. 30, no. 1, pp. 61-98, Mar. 2015, doi: 10.1016/j.csl.2014.09.005.
- [61] Z. Lu, X. Wu, and R. He, “Person identification from lip texture analysis,” in *2016 IEEE Int. Conf. Digit. Signal Process. (DSP)*, Beijing, China, Oct. 16-18, 2016, pp. 472-476, doi: 10.1109/ICDSP.2016.7868602.
- [62] Y. Liu, X. Jiang, T. Sun, K. Xu, “3D gait recognition based on a CNN-LSTM network with the fusion of SkeGEI and DA features,” in *2019 16th IEEE Int. Conf. Adv. Video and Signal Based Surveillance (AVSS)*, Taipei, Taiwan, 18-21 Sept. 2019, pp. 1-8, doi:10.1109/AVSS.2019.8909881.
- [63] D. Zhang, W. Wu, H. Cheng, R. Zhang, Z. Dong, and Z. Cai, “Image-to-video person re-identification with temporally memorized similarity learning,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 10, pp. 2622-2632, Oct. 2018, doi: 10.1109/TCSVT.2017.2723429.
- [64] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, “Action recognition in video sequences using deep bi-directional LSTM with CNN features,” *IEEE Access*, vol. 6, pp. 1155-1166, 2017, doi: 10.1109/ACCESS.2017.2778011.
- [65] C. Li, G. Zhan, and Z. Li, “News text classification based on improved Bi-LSTM-CNN,” in *2018 9th Int. Conf. Inf. Technol. in Medicine and Educ. (ITME)*, Hangzhou, China, Oct. 19-21, 2018, pp. 890-893, doi: 10.1109/ITME.2018.00199.

- [66] J. Xu and H. Yang, "Identification of pedestrian attributes based on video sequence," in *2018 IEEE Int. Conf. Adv. Manuf. (ICAM)*, Yunlin, Taiwan, Nov. 16-18, 2018, pp.467-470, doi: 10.1109/AMCON.2018.8614752.
- [67] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conf. Comput. Vision and Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, June 27-30, 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
- [68] S. H. Lee, S. Hosseini, H. J. Kwon, J. Moon, H. I. Koo, and N. I. Cho, "Age and gender estimation using deep residual neural network," in *2018 Int. Workshop on Adv. Image Technol. (IWAIT)*, Chiang Mai, Thailand, Jan. 7-9, 2018, pp. 1–3, doi: 10.1109/IWAIT.2018.8369763.
- [69] Z. Zhong, J. Li, L. Ma, H. Jiang, and H. Zhao, "Deep residual networks for hyperspectral image classification," in *2017 IEEE Int. Geosci. and Remote Sens. Symp. (IGARSS)*, Fort Worth, TX, USA, July 23-28, 2017, pp. 1824–1827, doi: 10.1109/IGARSS.2017.8127330.
- [70] K. He, X. Zhang, S. Ren, and J. Sun. "Identity mappings in deep residual networks," July 2016. [Online]. Available: <https://arxiv.org/pdf/1603.05027.pdf>. [Accessed: Aug. 5, 2018].
- [71] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," Aug. 2016. [Online]. Available: <https://arxiv.org/pdf/1602.07261.pdf>. [Accessed: Sept. 17, 2018].
- [72] C. Szegedy et al., "Going deeper with convolutions," in *2015 IEEE Conf. Comput. Vision and Pattern Recognit. (CVPR)*, Boston, MA, USA, June 7-12, 2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.

- [73] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and C. Wojna, “Rethinking the inception architecture for computer vision,” in *2016 IEEE Conf. Comput. Vision and Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, June 27-30, 2016, pp. 2818–2826, doi: 10.1109/CVPR.2016.308.
- [74] Y. Zhang, W. Wang, and J. Wang, “Deep discriminative network with inception module for person re-identification,” in *2017 IEEE Visual Commun. and Image Process. (VCIP)*, St. Petersburg, FL, USA, Dec. 10-13, 2017, pp. 1–4, doi: 10.1109/VCIP.2017.8305028.
- [75] R. Sinha and J. Clarke, “When technology meets technology: Retrained ‘Inception V3’ classifier for NGS based pathogen detection,” in *2017 IEEE Int. Conf. Bioinf. and Biomedicine (BIBM)*, Kansas City, MO, USA, Nov. 13-16, 2017, pp. 1–5, doi: 10.1109/BIBM.2017.8217942.
- [76] S. Pouyanfar, S. Chen, and M. Shyu, “An efficient deep residual-inception network for multimedia classification,” in *2017 IEEE Int. Conf. Multimedia and Expo (ICME)*, Hong Kong, China, July 10-14, 2017, pp. 373–378, doi: 10.1109/ICME.2017.8019447.
- [77] J. Carreira and A. Zisserman, “Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset,” Feb. 2018. [Online]. Available: <https://arxiv.org/pdf/1705.07750.pdf>. [Accessed: July 30, 2020].