Ministry of Higher
Education and Scientific Research
University of Diyala
College of science
Department of Computer Science

DeepFake Detection System

**A Thesis Submitted to the Department of Computer Science/ College of Science/ University of Diyala
In Partial Fulfilment of the Requirements for the Degree of MASTER in Computer Science**

By

Mohammed Akram Younus Al-Sa'ati

Supervised By

Asst. Prof. Dr .Taha Mohammad Hasan

2020AC **1442AH**

بسم الله الرحمن الرحيم

﴿ نَرْفَعُ دَرَجَاتٍ مَّن نَّشَاءُ وَفَوْقَ كُلِّ ذِي عِلْمٍ عَلِيمٌ ﴾

صدق الله العظيم

سورة يوسف

الاية (٧٦)

Acknowledgments

First of all, praise is to ***Allah*** *the lord of the whole creation, on all the blessing was the help in achieving this research to its end.*

I wish to express my thanks to my supervisors ***Dr. Taha Mohammad Hassan*** *for supervising this research and for the generosity, patience, and continuous guidance throughout the work. It has been my good fortune to have the advice and guidance from him. My thanks to the academic and administrative staff at the Department of the computer sciences\University of Diyala for their hospitality and generosity.*

I would like to express my gratitude to my ***wife*** *and* ***daughters*** *for their great support and continuous encouragement. Not to forget to mention the great generosity of* ***Mr. Khalil Ibrahim Abed Al-Qaisi****. This accomplishment would not have been possible without them.*

Dedication

To...

My father soul Prof. Dr.Akram Al-saati

My dear mother

My dear wife & my daughters

My Dear friend Mr.Khalil Al-Qaisi

All our distinguished teachers those who paved the way for our science and knowledge

Mohammed Akram Younus

(Supervisor's Certification)

We certify that this research entitled "DeepFake Detection System" was prepared by **Mohammed Akram Younus** under our supervision at the University of Diyala Faculty of Science Department of Computer Science, as partial fulfillment of the requirement needed to award the degree of Master of Science in Computer Science.

Signature: 

Name: Asst. Prof. Dr.Taha Mohammad Hasan

Date:

Approved by the University of Diyala Faculty of Science Department of Computer Science.

Signature: 

Name: Asst. Prof. Dr.Taha Mohammad Hasan

Date:

(Head of Computer Science Department)

(Linguistic Certification)

I certify that this research entitled “DeepFake Detection System” was prepared by Mohammed Akram Younus and was reviewed linguistically. Its language was amended to meet the style of the English language.

Signature:

Name:

Date:

Abstract

The easy and free entrance to large-scale public databases with the rapid quick progress of deep learning techniques and applying artificial intelligence (AI), especially the Generative Adversarial Networks (GAN), have driven to the production of very realistic fake videos content and guarantee an advanced level of realism with its implications towards corresponding society. This unlocks the door to a chain of sensational applications in different fields such as video games, film production, and advertising. On the other hand, it constitutes enormous security threats. Freely available software packages on the web allow any person, with minimum skills, to produce very realistic DeepFakes videos. This technology is used to blackmail and discredit people, manipulate the opinion of the public during elections, etc. There are no limits to the potential abuses of the human imagination. Subsequently, there is an imperious need for automated tools that have the capability of detecting such fake videos and averting the diffusion of dangerous fake multimedia content.

Previous works and researches in that field showed a lot of complexity with different accuracy. In this thesis, a new technique is described, proposing the use of the Wavelet Transform to determine the blur extent of the face region of interest (RIO) and the surrounding context by the use of edges type and make a comparison between them. This approach can successfully distinguish AI-generated counterfeit videos from genuine ones.

Based on the observations that the prevailing DeepFake set of rules can only generate images of constrained resolutions for the synthesized faces, which require additional distortion and obscuration to coordinate the valid appearances in the source video. Certainly, such changes will create unmistakable artifacts, the proposed method can effectively capture these artifacts by revealing the edge types and blurriness ratio. Most of the previous approaches need a large amount of DeepFake generated images and real portrayal to train the convolutional neural network (CNN). This technique does not need to bother with instances of

DeepFake produced depiction as negative preparation since it focuses on the artifacts as a distinctive feature in affine face warping to recognize if the videos are real or fake. Thus, economizes resource-demanding and time-consuming.

The used technique is more robust compared to others where such artifacts are general existed in DeepFake videos. The proposed method in this thesis was conducted using on the UADFV dataset and the result of the whole experiment result gave very good accuracy with great reliability, reached 100% on the mentioned dataset with a few nuances in each video test.

TABLE OF CONTENTS

List of Tables

List of Figures

List of Abbreviations

Abbreviations	Meaning
AI	Artificial Intelligence
ACC	Accuracy
API	Application Programming Interface
AUC	Area Under Curve
AUROC	Area Under the Receiver Operating Characteristic Curve
CNN	Convolutional Neural Network
CPU	Central Processing Unit
Dlib	Digital Library
DNN	Deep Neural Network
DWT	Discrete Wavelet Transform
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
IU	Image Understanding
LRCN	Long-term Recurrent Convolutional Network
LSTM	Long Short Term Memory
MCP	McCulloch and Pitts
MTCNN	Multi-Task Cascade Networking
PRNU	Photo Response Non-Uniformity
ResNet	Residential Network
ROI	Region of Interest
SVM	Support-Vector Machines
VGG	Visual Geometry Group
HOG	Histogram of Oriented Gradients

Chapter One

General Introduction

CHAPTER ONE
GENERAL INTRODUCTION

1.1 Introduction

In the last few years, the issue of fake news has turned into a threat to crowd discourse, human society, and democracy [1] [2]. Counterfeit news is a type of imaginative news-style to deceive the public with content that is fabricated [3] [4]. In 2020, 4.57 billion people were active internet users, encompassing 59 percent of the global population [5] [6], with a very large amount of images and videos are uploaded to the Internet each day. This includes millions of photos and over 400 hours of video content uploaded to social media and YouTube every minute. False news diffuses very fast especially via social media platforms, it can influence a very large number of internet users [7] where the studies showed that over 20% of the users follow the news through YouTube and Facebook [8].

The advanced technology of visual media advances has led to new facilities for processing and generating artificial videos. In particular, modern AI-based tools have been provided to create extremely hyper-realistic manipulated videos which are named DeepFakes, this new technology may comprise a serious threat to attack the general opinion on a certain event or the reputation of some individuals as almost with an average computer specification anyone can fabricate fake videos so easily that are virtually indistinguishable from actual real media [9]. Due to the rise in popularity of the misinformation the need to individuate this type of fake information becomes fundamental, a tool to confirm media news content genuinely, as new technologies permit persuasive manipulation of video [8]. Nowadays, the public lives in the era of what some have called a post-truth, it is characterized as information warfare running false news campaigns to manipulate public opinion driven by pernicious actors [10].

The proposed method shows a new forensic technique that can distinguish between fake and real video sequences with good accuracy. In this work the adoption of the inconsistency of the blur feature in the video frame to exploit the possible difference between ROI and the surrounded context. This clue is then used as a feature to be boosted by ResNet-50 classifiers. Preliminary results obtained on the DeepFake video dataset UADFV [11] [12] highlights very promising performances.

1.2 DeepFake videos

The term DeepFake is a mixture of Deep learning and Fake, DeepFakes are digitally manipulated hyper-realistic videos to depict individuals doing and saying things that have not occurred in reality. The production of DeepFakes depends mainly on artificial intelligence neural networks which are known as Generative Adversarial Networks (GANs) [13]. This neural network can learn in a comparable way to the brain of human beings. The more photos and videos of a person exposed to a neural network, the more accurately it can replicate the expressions of facial, and mannerisms of that person when producing a DeepFake video.

The most famous DeepFake applications and their highlights are introduced in Table (1.1). The procedure of DeepFake demand feeding two images (source and target) of different people into a Deep Learning [14] algorithm and train it to alternate their faces. It uses Artificial Intelligence and mapping of facial technology that swaps the face of a source person on a video into the face of the target person in the same video.

Table 1.1: Summary of well-known DeepFake applications.

Application	Key features
Faceswap	Shared parameters of the encoder are used. Uses two encoder-decoder pairs.
Faceswap-GAN	Perceptual misfortune using Visual Geometry Group (VGG) and adversarial disposed misfortune are added to the auto-encoder design.
DeepFaceLab	Bolster different face extraction modes, for example, Digital Library (Dlib), Multi-task Cascaded Convolutional Neural Networks (MTCNN), and Single Shot Scale-invariant Face Detector (S3FD). Grow the Faceswap model with new models.
DFaker	Implemented based on Keras library. DSSIM misfortune work is utilized to recreate the face.
DeepFake-TensorFlow	Same as DFaker but implemented depending on TensorFlow.

1.3 Related work

The detection of manipulated videos is much harder than fake image detection due to the solid corruption and degradation of the data of the frame after video compression [15]. A great challenge for methods designed to detect fake videos because of its temporal attributes that are differed among frames sets. Several related methods to the proposed work in this study have been reviewed in the literature.

- **P. Zhou et al. [16], 2017,** proposed a two-stream network for face tampering detection. They train GoogLeNet to detect tampering artifacts in a face classification stream and train a patch-based triplet network to leverage features capturing local noise residuals and camera characteristics as a second stream. Also, they used two different online face-swapping applications to create a new dataset that consists of 2010 tampered images, each of which contains a tampered face. They evaluated the proposed two-stream network on the newly collected dataset. Experimental results demonstrate the effectiveness of their method and it reaches 85.1% AUC. The Two-stream network is considered as a complex method and hard to train compared to the

results obtained. The proposed method was tested on the Celeb-DF and the results were too low, 53.8% AUC.

- **D. Afchar, et al. [15], 2018,** introduced a method to detect facial tampering in videos automatically and effectively at the mesoscopic level and focuses in particular on two recent techniques used to produce hyper-realistic faked videos, DeepFake and Face2Face. Due to the compression which severely degrades the data, traditional image forensics techniques are typically not well suited to videos. Thus, this work follows a deep learning approach at the mesoscopic level and presents two networks, both of which have a small number of layers to focus on image mesoscopic properties. A modified version of "Meso-4" is composed of a derivative of the "Inception module" proposed in [17], called "MesoInception-4". The solution they suggested was evaluated with a private dataset, achieving 98% ACC for the best accuracy results. The method was calibrated against unseen datasets in [18] and in some cases, as with "FaceForensics++", proved to be a robust approach, but its weakness was in finding the artifacts in some Deepfake videos, as seen in the results of the UADFV dataset which was 84.3% AUC.
- **M. Koopman, et al. [19], 2018,** proposed a method that uses the analysis of Photo Response Non-Uniformity (PRNU) to detect DeepFake video. The PRNU is characterized as an industrial facility deformity of light-sensitive sensors of advanced cameras known as noise patterns stemming. Every digital camera has its PRNU patterns and is considered as the digital image fingerprint [20] [21]. The exchanged faces are assumed to change the native PRNU patterns of video frames in the facial zone. The process starts by decomposing the videos into frames, then the facial regions are cropped. After that, it is then isolated sequentially into eight groups and an average of PRNU patterns is calculated for everyone. The results indicate that there is no correlation between the authenticity of the video and the variance in correlation scores. There does appear to be a correlation between the mean

correlation scores and the authenticity of the video, where on average original videos have higher mean normalized cross-correlation scores compared to the DeepFakes. Their proposed approach was evaluated over a private database created using 5 different mobile applications, achieving an average of 13.7% EER in manipulation detection which is considered a relatively high ratio compared to other methods.

- **M. Chang, et al. [22], 2018,** Proposed the Eye Blinking method to detect DeepFakes. Because of the way that an individual in DeepFakes has no regular eye blinking like that in non-manipulated videos. Normally images available on the internet do not show individuals with shut eyes, without having such images, DeepFake calculations cannot create scenes with faces that have normally blinking eyes. Distinguishing original from tampered videos, the author extracted the frames from the videos after that the eye areas are separated depending on six eye landmarks from face areas. Long-term recurrent convolutional network (LRCN) [23] is used on the cropped eye area sequences for the prediction of dynamic state. Based on CNN, the LRCN consists of a feature extractor, the arrangement learning relies upon Long Short Term Memory (LSTM), and the state forecast depends on the completely associated layer to gauge the probability of a shut-eye and open-eye state. Strong temporal dependencies were shown by the eye blinking, the LSTM implementation helps to capture these temporal patterns in a very effective way. The methodology was appraised on a lot of information that was gathered from the web comprising of 49 meetings and introduction recordings and their indistinguishable phony recordings delivered by the DeepFake algorithms. A promising result was gained from the proposed approach in detecting DeepFake videos, also, improvement can be performed by thinking about the dynamic pattern of blinking where exceedingly common blinking of the eyes might be considered as a sign of tampering. No long time

after this forensic technique was announced to the public, the upcoming age of synthesis strategies consolidated blinking into their frameworks.

- **Y. Li and S. J. Lyu [24], 2018,** proposed an approach that is based on the observations that the current DeepFake algorithm could only generate images of restricted resolutions, that need to be further warped to recreate the actual faces in the source video. These transforms leave distinctive artifacts in the resulting DeepFake videos, and they can be effectively captured by convolutional neural networks (CNNs). This approach was assessed on two DeepFake video datasets, called the DeepFakeTIMIT [25] and UADFV [12] [11]. The DeepFakeTIMIT dataset includes a set of 64×64 size low-resolution quality videos of and a second set of 128×128 high- resolution quality videos of with an around 10537 original images and 34,023 faked images obtained from 320 videos for each set of quality. The UADFV dataset consists of 49 genuine videos and 49 fake videos having around 32752 frames in total. Compared to previous methods that use a large amount of real and DeepFake images to train CNN classifier, this method doesn't need any negative training examples as it targets the objects in affine face warping as the distinctive feature for distinguishing real and fake images. This method was evaluated on the UADFV dataset on “VGG16” [26], ResNet50, ResNet101, and ResNet152 [27] models using the Area Under Curve (AUC) metric in two settings: image-based evaluation and video-based evaluation. For image-based evaluation, they process and send frames of all videos into the four networks respectively. The VGG16, ResNet50, ResNet101 and ResNet152 models achieve AUC performance 83.3%, 97.4%, 95.4%, 93.8%, respectively. ResNet networks have about 10% better performance compared to VGG16, due to the residual connections, which make the learning process more effective. Yet, ResNet50 has the best performance among the other ResNet networks. The video level performance of each type of CNN model. VGG16, ResNet50, ResNet101 and ResNet152 can achieve AUC performance 84.5%, 98.7%, 99.1%, 97.8%

respectively. In this video-based evaluation metric, the ResNet network still performs ∼15% better than VGG16. Yet, each ResNet model has a similar performance, as in the case of image-level classification.

- **X. Yang, et al. [11], 2019,** proposed a new way of revealing DeepFake videos created by Artificial Intelligence methods. This method is based on the observations that DeepFakes is created by splicing the synthesized face region into the original image, thereby introducing errors that can be revealed when the face images estimate 3D head poses. An SVM classifier is evaluated using a collection of real face images and DeepFakes using features based on this cue. The results, assessed using individual frames as an inspection unit with the output metric Area Under ROC (AUROC). The tests show the SVM classifier reaches an AUROC of 89.0% on the UADFV dataset. This indicates that the estimated difference between the head and the whole face from the central region is a good feature for identifying images generated by DeepFake. Besides, an estimation of the performance was carried out using individual videos as an analysis unit for the UADFV dataset. This is done by averaging the estimation of frame classification over the individual videos. They also conduct an ablation study comparing the performance of various types of features used in the SVM classifier.
- **E. Sabir et al. [28], 2019,** proposed a method that exploits the Spatio-temporal highlights to distinguish DeepFakes videos. Recurrent convolutional layers models are a class of profound learning models that have proved effective in exploiting temporal information from domain-wide image streams. Thus the best strategy for combining variations in these models with domain-specific face preprocessing techniques is refined through extensive experimentation to obtain state-of-the-art performance on benchmarks of publicly available video-based facial manipulation. Specifically, the attempt is to identify tampered faces in video sources achieving AUC results of 96.9% and 96.3% for the DeepFake and FaceSwap methods, respectively. Only the low-quality

videos were considered in the analysis. The test is also carried on the FaceForensics++ dataset [29], improving the previous state-of-the-art by up to an accuracy of 4.55% to reach 94.3% ACC.

- **H. Nguyen, et al. [30], 2019,** proposed detecting manipulated videos by the use of Capsule Networks. This type of network was introduced at the beginning to address the CNNs limitations when used in inverse graphics tasks. The evaluation of this method was on four datasets having a wide range of fake videos and images, which include the Ldiap Research Institute replay attack dataset [31]. The accuracy of face swapping detection at frame level on the DeepFake dataset was 95.93% and the accuracy of face swapping detection at video level on the DeepFake dataset was 99.23%.
- **O. de Lima et al. [32], 2020,** Showed that intra-frame inconsistency and temporal inconsistency among frames are found in DeepFake videos. A temporal-aware pipeline method was proposed which utilize CNN and Long Short Term Memory (LSTM) to spot DeepFake videos. Frame features are extracted by the CNN, after that it is passed into the LSTM to generate a descriptor of the temporal sequence. Afterward, for classifying manipulated videos from genuine videos, a fully-connected network is used based on the sequence descriptor. The system using the Celeb-DF dataset can accurately predict if the fragment being analyzed comes from a DeepFake video or not, this method outperformed state-of-the-art frame-based detection methods. This method tested some of the most popular networks that take advantage of temporal features. All the networks were trained on the Celeb-DF dataset starting from the pre-trained published weights. No layers were frozen for training. Each method was trained for over 25 epochs and the best ROCAUC scores were 74.87%, 99.43%, 97.59%, 99.30%, and 99.73 on Residential Communications Network, R2Plus1D, I3D, Mobile CubeSat Command and Control (MC3), and R3D respectively.

1.4 Statement of the problem

DeepFake technology can learn and utilize from the massive amounts of photos and videos found on the Internet to generate not only forged videos but in a very hyper-realistic video of individuals. Scampers may use these renderings to target different types of public figures and political leaders, such as the presidents of the leading countries, even executives of major celebrities. The consequence of this phenomenon would lead to distrust in what to hear or what to see. DeepFakes are a seemingly realistic video of an individual's generated using artificial intelligence that shows actions that never occurred in reality. These types of videos are becoming almost indistinguishable from the real videos.

Now, as we are living in the digital age, the ability of denial and fraud campaigns have advanced more than ever before, by coordinating online campaigns to spread artificial false, misleading, or malignant content. The habit of creating an alternative reality is a result of the incompetence of people to believe what they hear or see, people are more likely to pick out the reality that most mightily aligns with their thought. The substitutional reality may divide society by engaging in people's prejudice and by eliminating the common understanding of the truth.

1.5 The aim of the thises

This thesis aims to build a strong DeepFake detection system to distinguish between real and fake videos that have been generated by DeepFakes applications by using Discrete Wavelet Transform (DWT). Besides the use of DWT, a pre-trained ResNet-50 has been used to boost the capture of the artifacts to give more accurate results. Serious steps must be taken, and start developing tools for the detection of DeepFakes videos to limit the creation of this phenomenon

1.6 Thesis organization

The thesis is segmented into five chapters; a brief description of their contents is given below:

Chapter One: This chapter introduces an overview of the work and related works.

Chapter Two: This chapter introduced methods and descriptions for the theoretical background and techniques that are used in this thesis.

Chapter Three: This chapter describes the proposed systems with their design and implementation and the execution of the stages of the proposed system.

Chapter Four: This chapter presents the tests and the results of the proposed system.

Chapter Five: This chapter offers conclusions and systems for future work.

Chapter Two

Theoretical Background

CHAPTER TWO
THEORETICAL BACKGROUND

2.1 Introduction

The technological and scientific revolution has always created social and scientific transformation. The past few years displayed the progress in speed and scale at which such change is achieved. Artificial Intelligence (AI) in the 2010s, started to show its potential superb power. Driven at most by the emergence of deep learning and the use of neural networks to spot patterns in complex data. These improvements in technologies reach every area of daily life, such as education areas, health, production industries, and so on. Thus, recent advancements in computer and software technologies are the base for the society of tomorrow. The artificial intelligence is one of the hottest topics of Computer Vision, tremendous advances are achieved in self-driving cars, robotics as well as in various photo editing applications. Every day steady progress is being made in object detection. These days, GANs are also one of the things that researchers are concentrating on. Vision is revealing to us the technology future, it is scary what will be the end of its possibilities.

Python nowadays becomes the most used machine learning and artificial intelligence programming language. The reasons behind the popularity of the use of Python is its reliability and efficiency, prebuilt libraries, healthy and active and supportive community, and supports Big Data as well. Python has made its way into the most complex technologies such as Artificial Intelligence, Machine Learning, and Deep Learning. For these reasons and many other ones, this language has been used in this research thesis to achieve the best results possible.

2.2 History of DeepFake

Circa 1865, the first face-swapping attempt known was in one of the iconic portraits of U.S. President Abraham Lincoln, see Figure (2.1). The photo was a mix of Lincoln's head with the body of John Calhoun. The advance new techniques have radicalized visual-data manipulation. The access to computing infrastructure with the help of modern tools such as TensorFlow has endorsed this paradigm shift. Tampering images and videos has become an extremely accessible task and within the reach of any person with a computer with average specifications due to the help of auto-encoders. FakeApp and FaceApp are open-source software built upon this idea, have made it extremely easy for anyone to create DeepFakes video.

The DeepFakes first appear to the public was in 2017 when a Reddit website anonymous user has posted videos showing celebrities in disclose sexual situations that are fakely synthesized. These artificially manipulated video clips used the social media, image search engines, and TensorFlow, to replace the face of someone with another on the preexisting videos frame by frame.

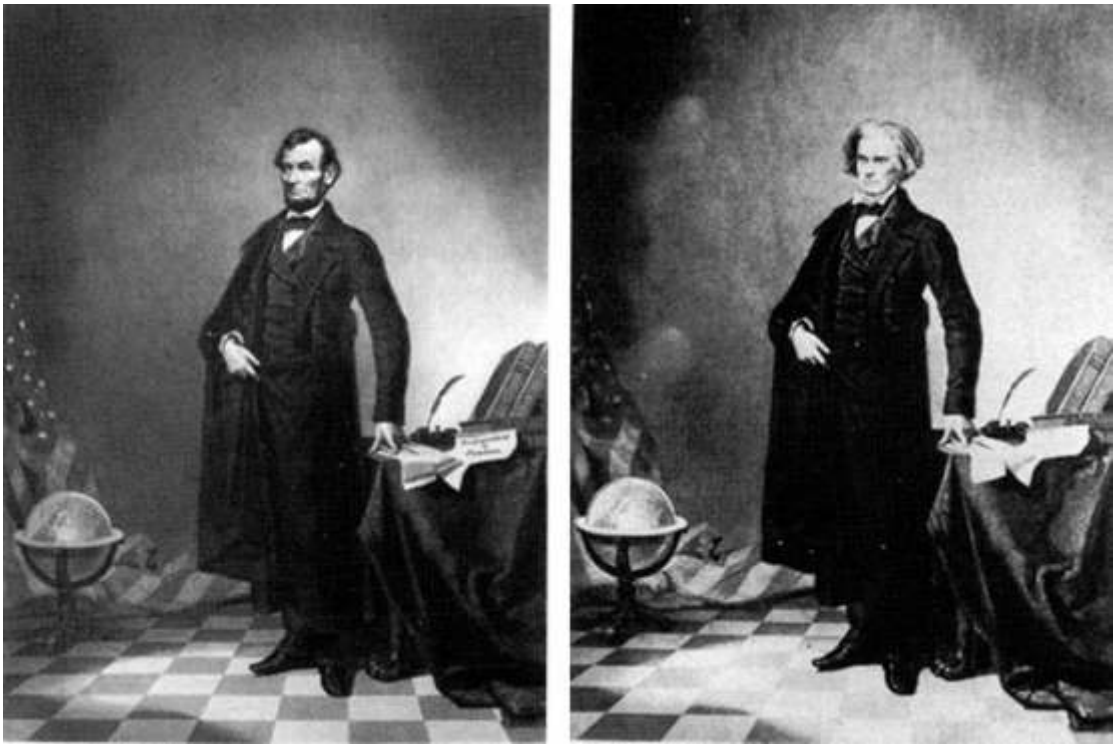


Figure 2.1: Lincoln-Calhoun Composite – Iconic Photo.

Lately, highly sophisticated artificial intelligence programs have been developed by researchers and scientists have resulted in a rising degree of accuracy when generating DeepFakes videos [33]. A simple application of this technology, such as free face-swapping applications, still poses a true potential national security threat as loge as they are driven by advanced algorithms and unsupervised learning. As the DeepFakes appearance was very realistic it was a target for the production of fake news, fake scandals, and malicious uses. Political tensions have already been created by these manipulated fake videos and they are being taken into account by governments.

2.3 The technological system of DeepFakes

The big question, “Would the phenomenon of DeepFakes has occurred and spread if the code of DeepFakes has not been shared?”, it appears that this commitment did not have a lot of effect on the scene. The opinion of the researchers says that by now, machine learning is an unavoidable development across many fields. This perspective bolstered by the history that showed how innovations have regularly been grown simultaneously by different gatherings or with assistance from various donors. The creators of DeepFake aren't as autonomous as the accounts would make them look except if the innovation being referred to be intensely subject to other technological advances. The DeepFakes method is very much the same case. The first code contribution was only 228 lines. An extra 170 lines of the valid DeepFake code were authorized from different developers [34], the following dependencies also required the [libraries codes] that must be downloaded exclusively to have the option to run the code:

1. Python – An interpreter software that is an open-source programming language.
2. OpenCV – Machine learning software library and an open-source computer vision library.
3. Tensorflow – (Originally coded by Google) an open-source machine learning framework library.
4. Keras— A high-level neural networks library for application programming interface (API).

These libraries are huge code subroutines, based on a lot of examination and code work. The Tensorflow library alone is made out of just about two million lines of code at the time of composing [34]. The code of the DeepFakes method requires a powerful computer processor unit (CPU) with a very powerful graphics processing unit (GPU). These hardware foundations and software libraries are fundamental for producing the DeepFake video.

Several machine learning technologies seen in the last few years have the ability to manipulate faces, the DeepFakes technology have public nature that makes it stand out in the scene. Not to forget the important implications are due to the difference between the development of software done secretly to produce software held in closed settings, and the development of the software was in cooperation with open-source practices.

2.4 Image Forensics

Digital images count as natural and active media for carrying valuable information over the globe. With the release of high-resolution digital cameras, PCs, and developed applications of photo editing, digital images forgery have become very popular [35]. Methods for detecting inauthentic facial images and videos has increased lately due to the massive fake images and videos published on the internet and precisely on social media and YouTube platforms. Those methods are generally based on finding inconsistencies in images, such as finding

artifacts [36], detecting distortions [37], and assessing the quality of an image [38]. In the context of synthetic images, however, the distortions and noise are not easy to detect due to the complexity and nonlinearity of the learning process [39]. Two different strategies exist to tackle this problem:

- Approaches of pure Deep Learning that work as a specific generative model detector [16][44][45].
- Approaches of semantic that evaluate the realism of the generated faces [22][25][40].

The first strategy investigates the noise and color distributions of specific networks [41][42], or train CNNs for synthetic images in a blindly way [15][43]. On the other hand, they are unqualified to be accepted as general mechanisms for synthetic portrait video detection, as they heavily depend on detecting artifacts ingrained to specific generative models.

However, semantic approaches, employ inconsistencies in the biological domain, such as attributes of facial [40], eye blink detection [22], and mouth movement inconsistency [25].

2.5 Computer Vision

It is the automated information extraction from images. The information of an image may be a camera position, object detection, 3D models, etc. It has been rapidly growing within the past few decades, generating tools that have the ability for the realization of visual information realization, exceptionally for scenery without accompanying administrative, structural, or descriptive text information. The internet has nowadays become a popular channel for graphical information transmission and exchanging. The indexing of huge collections of pictures by hand is an errand more than can be managed, reasonableness has started to direct and give some new procedures of scholarly access to computerized digital picture assortments [44]. The analysis of an image demands the incorporation of high-level concept creation with the interpretation and processing of inherent visual

features. The computer vision systems development has begun to interplay between machine and human image indexing techniques in the field of scholarly access to visual data. The community of image understanding (IU) suggests that the most productive approaches to IU cover learning and analysis of the information type being sought, the domain in which it will be used, and systematic testing to identify optimal methods.

The computer vision goal is to provide human-like computers with perception ability so that they can sense the surrounding environment, realize the sensed data, take suitable and appropriate actions, and learn from this experience to enhance future performance. The field of computer vision has gradually developed from the classical pattern application recognition and techniques of image processing to advanced image understanding applications, a vision of knowledge-based, systems that exhibit learning capability, and a vision of model-based. The capability to learn and to reason are the two senior capabilities of correlating with these types of systems. Recently, practical and theoretical advances are being achieved in the computer vision field and pattern recognition by new procedures and learning techniques, adaptation, and representation. It is likely reasonable to claim, in any case, that learning represents the following challenges frontier for computer vision.

2.6 Generative Adversarial Networks (GANs)

A machine learning model category, named Generative Adversarial Networks (GANs), which was developed in 2014 by Ian Goodfellow et al. [13]. Its concepts, two unsupervised neural networks challenge with each other. The two parts which form the GANs are the generative network, which responsible for generating candidates, and the discriminative network, which in turn evaluates them. The generative network part aims to produce data points that are comparable to the data points in the training set, and the discriminative network aims to judge these data points. The GAN's in total aims to raise the

discriminative network error rate to fool it with the produced data points that almost seem real.

Looking at the GANs architecture, see Figure (2.2), it is not hard to realize how GANs are particularly appropriate for generating DeepFakes. The Generative Adversarial Networks (GAN) model is composed of the input vector, generator, and discriminator. Among them, the generator and discriminator are implicit function expressions, usually implemented by deep neural networks. GAN can learn the generative model of any data distribution through adversarial methods with excellent performance. It has been widely applied to different areas since it was proposed. This model was optimized for generating new data with identical features to the training set model. In the images of faces DeepFake case, the GAN model is trained on the highly rich dataset which is optimized to produce pictures as close as possible to the original dataset.

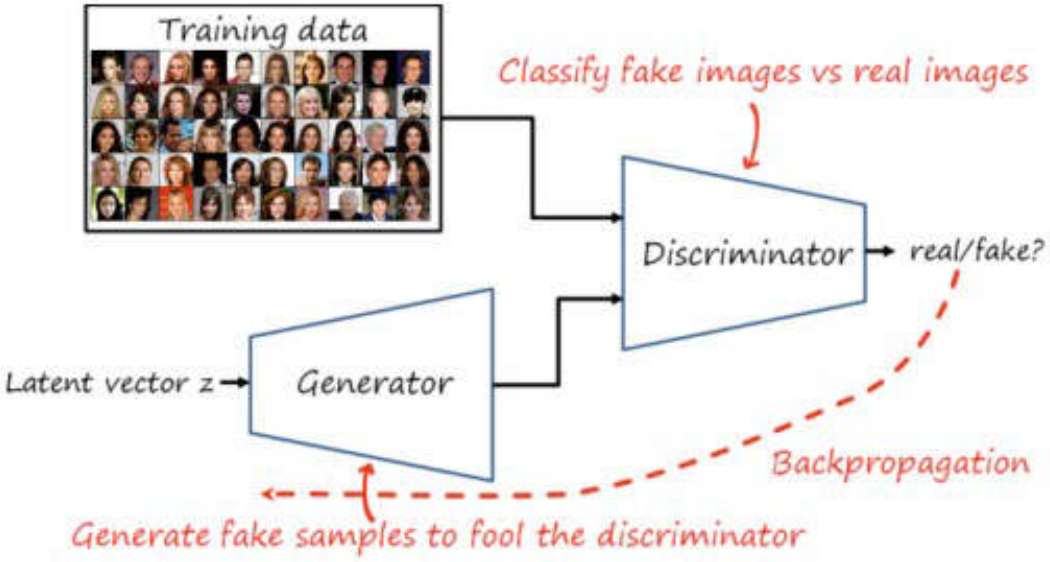


Figure 2.2: Block diagram of the Generative Adversarial Network.

Despite that, GANs are not that evil, it should be mentioned that beneficial usage occurs in other valid scopes, such as:

1. **Astronomy** - Sometimes astronomical photos need to be improved where random and systematic noise from the sky background is common, by employing GANs, features are recovered from artificially degraded photos and detailed features [45].
2. **Video game modding** - GANs have been utilized in games to up-scale low-resolution photos from old video games that retain original details levels while producing sharper textures, such as Twilight Princess, The Legend of Zelda, and Final Fantasy VII [46].
3. **Biological image synthesis -** It has been applied to the synthesis cell's fluorescence microscopy photos. Cells usually tend to have a simpler and more geometric global structure that simplifies the production of a photo. The correlation between the different fluorescent protein's spatial patterns reflects important biological functions, and synthesized images have to capture these relationships to be relevant for biological applications. The task at hand has been adapted by GANs to propose new models with casual dependencies between image channels that can generate multichannel images, which would be impossible to obtain experimentally [47].

2.7 Convolutional Neural Networks

The Convolutional Neural Networks (CNNs) is equivalent to the classic ANNs, Both are consist of neurons that self-optimize through learning. The same way the neuron receives input and performs an operation. The picture vectors go from the information crude to the last yield of the class score, the whole system communicates a single perceptive score work (weight). The misfortune capacities related to the classes are contained in the last layer, along these lines, standard deceives and tips produced for customary ANNs still apply.

The primary use of CNNs within images was in the field of pattern recognition, which is the only prominent difference between traditional ANNs and CNNs. This allows encoding image-specific features into the architecture, in this way the network much appropriate for image-focused tasks. ANN traditional forms have huge limitations, there is a big struggle with the complexity of the computational required to compute the data of the image. Benchmarking datasets of popular machine learning like the MNIST database of handwritten digits are appropriate for ANN of most forms.

2.8 Visual Artifacts

Often, when DeepFakes generate synthesized faces have a visual artifact [40] that comprises one or more of the following:

a) **Global Inconsistency** [46], the faces which are produced by DeepFakes are supposed to support the image interpolation, the different faces mixture is not always consistent, thus, that they lack global consistency. For example, images produced from DeepFakes usually have high variances in color between the right and left eye.

   In Face2Face [48], for example, the real-time capture of the face and reenactment of RGB Videos, illumination, and rendering estimations are modeled explicitly. In most illumination estimations of the deep-learning-based models often learned from the data implicitly. This results in imprecise incident illumination estimations. An inaccurate illumination estimation example in many DeepFakes is shading artifacts, where it can be spotted in one area of the nose which is unnaturally rendered dark; usually, observed on concavity or when multiple objects are located near each other. Moreover, often reflection details are missing in eyes produced through DeepFakes. The reflections of specular which noticeable in real images, usually unconvincingly produced in DeepFakes; missing reflections are usually simplified into a white blob, resulting in a dull eye appearance.

b) **Estimation Geometry Faulty** [46], as in the case of illumination, the estimations of the geometry are usually made to fit a morph-able model to images, artifacts that arise from imprecise estimations of geometry can be spotted as seen by the data from the Face2Face model. To be more precise, artifacts that are revealed as strong edges or high-contrast spots around the boundary of an overlaid face mask often appear on spots such as the face, eyebrows, and nose. Moreover, geometries like teeth are usually not modeled around; often teeth appear as a single white blob instead of as individual teeth.

2.9 ResNet-50 for classification the image

The "ResNet-50" is a model of Deep Learning that is pre-trained for image classification of the CNN, it is one of the classes of Deep Neural Networks [18] [49], usually used for analyzing visual imagery. The "ResNet-50" consists of fifty deep layers, it has been trained on a million photos of one thousand categories from the ImageNet database. This model has more than twenty-three million trainable parameters, which indicates how deep the model architecture, thus, makes it the best for recognition of an image. It is a very effective approach when compared to build a new one from scratch, otherwise, a great amount of data needed to be collected and trained. Many pre-trained deep models can be used like the "AlexNet", "GoogleNet", or "VGG19", but the "ResNet-50" is pointed for superb generalization performance with a minimum of errors rates on the process of recognition [50].

2.10 The architecture of ResNet-50

The name ResNet as it is well known comes from Residual Network and to be more specific it is a Residual Neural Network architecture. The connections of its identity characterize it. The connections of its identity take the input straightway to the last layer of each residual block, as illustrated in Figure (2.3).

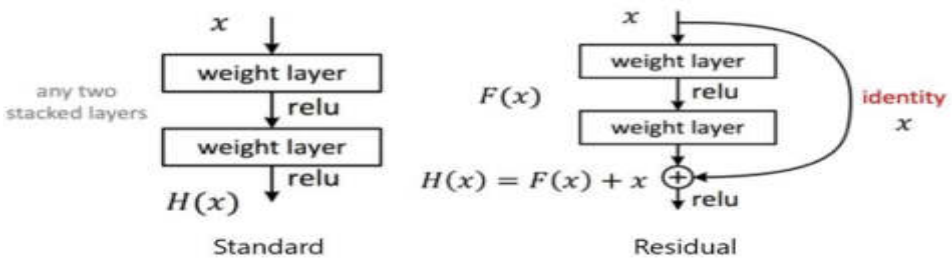


Figure 2.3: A simple ResNet-50 residual block compared to standard network. Where x is the image input to the residual block, $F(x)$ and $H(x)$ donate the image after Convolution.

Mathematically the researchers have proven that deeper neural networks have more power of representation [51]. The deep networks earn this power from hierarchically structuring shallower feature representations into deeper representations. In recognition of the face, for instance, the pixels form edges and edges form corners. Corners represent features of the facial like eyes, noses, and mouths. These facial features will form the face shape [52].

The ResNet-50 model is simply comprised of five stages each with a residual block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. The idea of residual blocks is very basic. In conventional neural systems, each layer takes passed into the following layer. In a system with residual blocks, each layer is passed into the following layer and directly into the layers around 2–3 jumps away, that is the thing that called identity connections.

As known, deep CNNs are not easy to train because of gradients vanishing in the long forward feed and backward propagate procedure. On the other hand, the residual neural network has parallel shortcut connections to the layers of normal convolutional. In a mathematic view, the layer of ResNet roughly calculated as:

$$H(x) = f(x) + id(x) = f(x) + x \quad (2.1)$$

Where the image input to the residual block is x, $F(x)$ is the image after convolution, $id(x)$ is the identity connection and $H(x)$ donate the image output of the residual block.

Parallel shortcuts work as highways and the gradients may easily flow back, in this way faster training is performed.

The ResNet has succeeded so bravely, and extra related work has been done. An improved version was proposed by the original authors of ResNet by adding up more direct identity connections to the network models [53].

2.11 Discrete Wavelets Transform

The Discrete Wavelet Transform (DWT) is a type of linear transformation that works on an integer power of two of the length of the data vector, the result will be a vector of the same length but numerically different. The DWT can detach data into components of different frequencies, then investigates each ingredient with a resolution that matches its scale. Figure (2.4), shows the DWT tree.

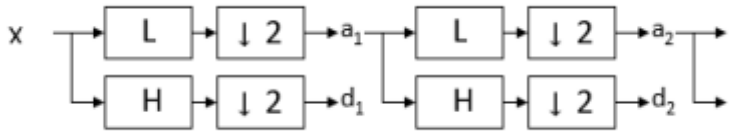


Figure 2.4: Discrete wavelets transform tree.

Where *L* and *H* denote low and high -pass filters, ↓ 2 indicate subsampling, a_1 the image after low pass filter, d_1 the image after high pass filter, a_2 the image after 2-level low pass filter, and d_2 the image after low pass filter plus high pass filter.

For the two-dimension image, the DWT algorithm is the same, firstly it takes all the rows of the image, and then all columns performed, as shown in Figure (2.5).

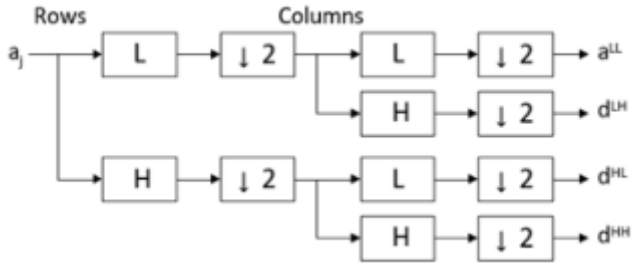


Figure 2.5: Two-dimensional pictures Wavelet decomposition.

Where *L* and *H* denote low and high -pass filters, ↓ 2 indicate subsampling, a^{LL} the image after 2-level low pass filter, d^{LH} the image after low pass filter plus high pass filter, d^{HL} the image after 2-level high pass plus low pass filter, and d^{HH} the image after 2-level high pass filter.

The DWT has a significant main feature which is a representation of the multiscale of the function. When utilizing the Wavelets, a function that is given may be analyzed at several levels of resolution. Also, the DWT can be orthogonal and invertible. It seems that Wavelets is efficient for the recorded textures analysis with various resolutions too.

2.11.1 Edge detection problems

Separating the scene into the background and object is an essential interpretation step in the image analysis. The human visual system carries out that process effortlessly. Several problems can be encountered when trying to imitate this action by designing computer vision algorithms. As a consequence of the existence of quantization, noise, and intensity changes in the authentic image edge detection may locate edges that do not exist. Farther more, it is probable to fail the detection of the existing edges completely. The degree of success in an edge detector relies on its ability to accurately mark the true edges.

The other issue of the problem is the localization of an edge faced within the detection of edges. The noise present in an image may lead to a shift in the detected edge position from its true position. In noisy images, the edge-detector capacity to determine an edge location as close as possible to its true position is a very serious factor in performance determination. Furthermore, the difficulty of the system of edge detection emerges from the fact that the intensity transitions sharpness indicate an edge is sharp due to their high-frequency components. As a consequence of that, any smoothing or linear filtering performed on these edges to suppress noise will also blur the significant transitions. Anyway, some sort of smoothing is necessary since the edge detection depends on the recognition of the image function, this amplifies all high-frequency components of the signal, including the noise components. The most widely smoothing filters used are the Low-pass filters. The smoothing amount applied depends on the smoothing operator size or scale. Generally, fine details of intensity changes are extracted

by the detector on a small scale, however, it tends to be more sentient to noise. Coarse details extracted of intensity changes on a larger scale, but several of the detected edges are more likely to have more localization errors [54]. Choosing an ideal single size of smoothing for all edges in a picture isn't a simple errand.

An alternative is offered which is multiscale edge detection by applying smoothing operators of different sizes to an image and extracting the edges at each scale, after that, combining the recovered edge information to form a more complete picture of the actual edge representation of the image. The technique's basic hypothesis is that different parts of an image have varying noisiness degrees and edges types; therefore, every image part needs to be differently smoothed. This technique, however, has associated problems such as how many filters should be used, how to determine the filters scales, and how to combine the responses from each filter to create a single edge map.

2.11.2 Using Haar Wavelet for edge detection

Since 1910, Haar functions are used, they were first introduced by Hungarian mathematician, Alfred Haar [55][56]. These days, there are many definitions of the Haar functions and several generalizations [57]. Furthermore, some modifications [58][59][60] were published and used too. Wavelets generally, with all modifications and generalizations, were prepared to adapt this notion to some practical applications [61]. The Haar functions have been used by the DWT in binary logic design, image coding, and edge extraction and it is one of today's most promising techniques.

Many sorts of edges in an image may be found such as “Dirac”, “Step”, and “Roof” edges [62][63]. In blurred images, some edges may lose their strength and some edges might disappear [4]. It has been seen that the strongest edge extent in blurred pictures is typically under 100 most of the time. Areas and quality of edges may effectively be investigated in the wavelet domain.

Detecting edges in a photo, first decomposed it into 4 subbands “LL”, “HL”, “LH”, and “HH” by stratifying one level of Discrete Wavelet Transform (DWT). The global properties of an image are found in the subband “LL”, where the “HL”, “LH”, and “HH” are horizontal, vertical, and diagonal detail subbands. Decomposition one level of Akiyo-video-sequence_Q320 [64] is shown in Figure (2.6). The “LL” subband at the upper left side, the “HL” subband at the upper right side, the “LH” subband found at the lower left side, and the “HH” subband at the lower right side.

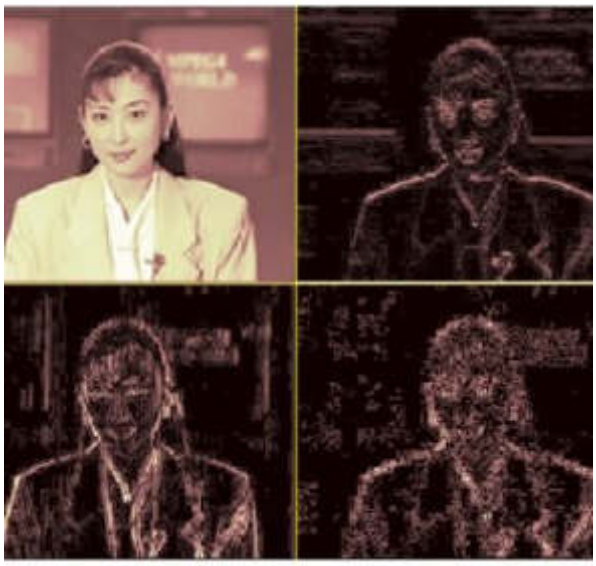


Figure 2.6: One-level-of-DWT-decomposition-for-Akiyo-video-sequence_Q320 [64].

The edges of an image are usually categorized into three types: the first one is the “Dirac-Structure” edge, secondly the “Step-Structure”, and the last one is the “Roof-Structure” [63][65][66]. A further classification of the Step-Structure is done, “Astep-Structure” and “Gstep-Structure” according to whether the intensity has gradual change or not. Different types of edges graphical descriptions are shown in Figure (2.7).

The sharpness of the “Gstep-Structure” and “Roof-Structure” edge is indicated by the variable α ($0< \alpha<\pi/2$), as much as the edge is sharp, α is large.

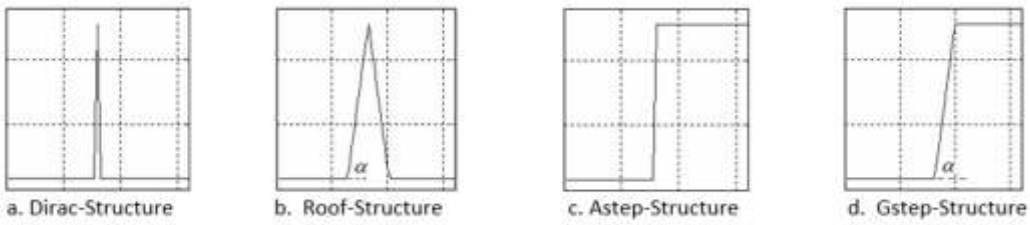


Figure 2.7: Graphic description of edges type [63].

The fundamental thought of the proposed scheme is as per the following: generally, most regular images contain all types of edges more or less, and nearly "Gstep-Structure" and "Roof-Structure" are sufficiently sharp. In case that blur is found, both "Dirac-Structure" and "Astep-Structure" will vanish. Besides, both "Gstep-Structure" and "Roof-Structure" are willing to lose their sharpness. The technique judges if a given picture is blurred or not as per on the off chance that it has "Dirac-Structure" or "Astep-Structure", and uses the level of "Gstep-Structure" and "Roof-Structure" which are increasingly likely to be found in a blurred image to decide the degree of the blur extent [67]. The scheme structure is shown in Figure (2.8).

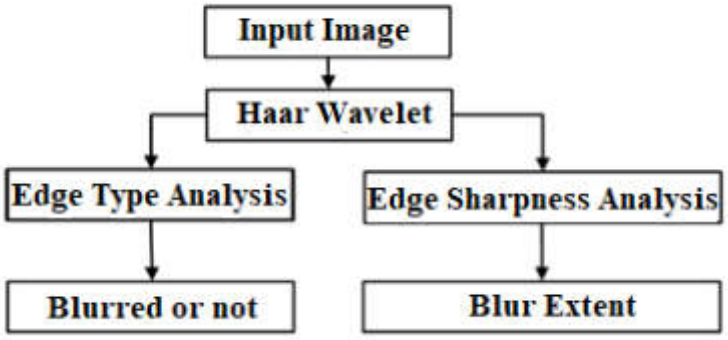


Figure 2.8: Blur detection Structure scheme.

2.11.3 HWT for edge detection Algorithm

Step1: Perform Harr wavelet transform to the original image and the decomposition level is 3. The result is a hierarchical pyramid-like structure.

Step2: Construct the edge map in each scale by applying the formula:

$$E\ map_i(K,l) = \sqrt{LH_i^2 + HL_i^2 + HH_i^2} \quad (where\ i = 1,2,3) \qquad (2.2)$$

Where E is the constructed edge map, (k,l) is the pixel coordinates, LH, HL, and HH are the sub-bands of the wavelet transform, i is iteration.

Step3: Partition the edge maps and find local maxima in each window. The window size in the highest scale is 2×2, the next coarser scale is 4×4, and the coarsest one is 8×8. The result is denoted as $E(\max)_i$ (*where i*= 1,2,3).

In this technique, $E(\max)_i$ states the edge intensity, the more intense the edge is, the larger $E(\max)_i$ is. For a specific threshold, if $E(\max)_i > threshold\ (k,l)$ is categorized as an edge point in the corresponding scale; otherwise, it is categorized as a point of non-edge.

The relative intensity in different scales of different types is summarized in Table (2.1).

Table 2.1: Effect of HWT on different types of edges

	Emax1	Emax2	Emax3
Dirac-Structure	Highest	Middle	Lowest
Astep-Structure	Highest	Middle	Lowest
Gstep-Structure	Lowest	Middle	Highest
Roof-Structure	Lowest	Middle	Highest
	Lowest	Highest	Middle

Another important property of Haar wavelet transform is its ability to recover the sharpness of the blurred edge: when observed on small scale, the blurred Roof-Structure and Gstep-Structure will become thinner and thus recover their sharpness.

The above two important properties of Haar wavelet transform lead to the following five rules [67]:

- **Rule 1:** If $E(\max)_1(k,l)$ > threshold or $E(\max)_2(k,l)$ > threshold or $E(\max)_3(k,l)$ > threshold (k,l) is an edge point.
- **Rule 2:** For any edge point(k,l), if $E(\max)_1 > E(\max)_2 > E(\max)_3$; (k,l) is Dirac-Structure or Astep-Structure.
- **Rule 3:** For any edge point(k,l), if $E(\max)_1 < E(\max)_2 < E(\max)_3$; (k,l) is Roof-Structure or Gstep-Structure.
- **Rule 4:** For any edge point(k,l), if if $E(\max)_2 > E(\max)_1$ and if $E(\max)_2 < E(\max)_3$; (k,l) is Roof-Structure.
- **Rule 5:** For any Gstep-Structure or Roof-Structure edge point(k,l), if $E(\max)_1(k,l)$ > threshold; (k,l) is more likely to be in a blurred image.

2.12 Face detection

Recently, technologies of Artificial Intelligence are developing actively, they offer massive feasibility for researchers in many fields. Object recognition, forecasting, analysis reaches high grades with the utilizing of machine learning and technologies of artificial intelligence. Computer vision turned into a very encouraging field of exploration. This territory of innovation is the most sought after in regular day to day existence. The most attractive one in computer vision issues that are by and large effectively investigated is face recognition. This improvement brought about the formation of a few kinds of libraries and APIs for face location detection.

Numerous arrangements are grown especially for a specific region, or qualified for solving and explaining several at once, set up in a particular programming language or with the help of all notable languages. Sometimes, among all the assortment of tools, it is not that easy to realize the best suited for solving the problem at hand. The most popular algorithm for face detection and finding facial expressions in images was the Dlib, which is based on the well-known “HOG - SVM pipeline” for object detection [68].

Originally, the primary author of the Dlib library is Davis King, it was developed in C++ with the support of machine learning applications [69]. The development philosophy core of the Dlib library is a devotion to conveying ability and usability. Therefore, all code in the library of Dlib is intended to be as compact as could be expected under the circumstances and comparably to not require a client to design or introduce anything.

Using the Dlib library, the bounding box can be gained by the face detector "get frontal face detector instruction". Dlib library is a perfect face detector and very accurate for extracting the points from frontal faces, see Figure (2.9 and 2.10).

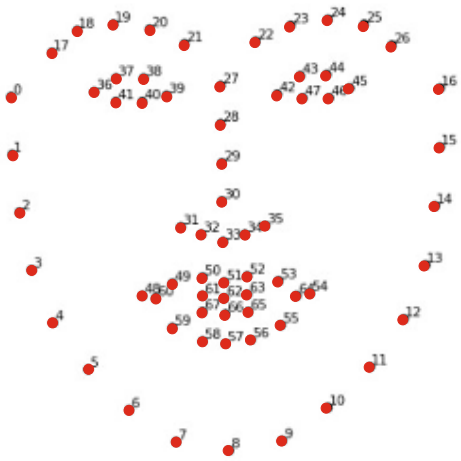


Figure 2.9: Reference points for recognizing the face zone.

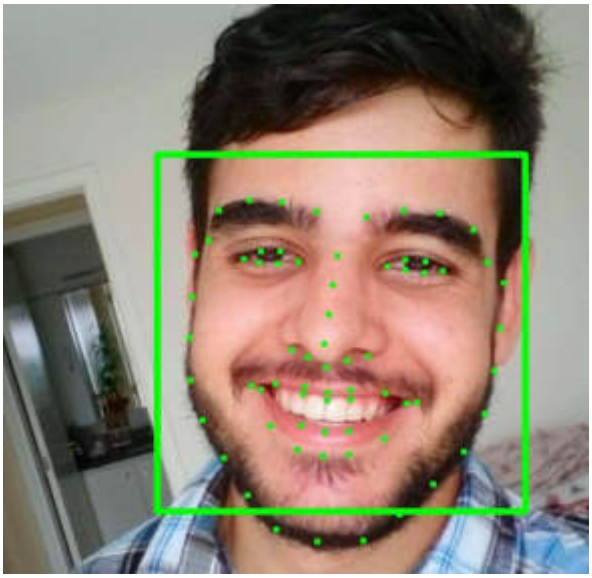


Figure 2.10: A simple processed by Dlib's shape predictor.

2.13 Histogram of Directional Gradients (HOG)

The histogram of directional gradients is a descriptor of singular points for object recognition, is used to detect faces in a photograph. HOG-descriptor is an image transformation into a multidimensional vector, which allows using the SVM classifier or a neural network. For the classification of HOG-descriptors usually used SVM. An example of the *HOG*-structure of the face is shown below in Figure (2.11).

The HOG implementation uses a sliding window classifier. Since most teaching algorithms with a teacher work in a feature space of a fixed dimension, the HOG feature vectors for different images must be of the same length, and therefore the images to be classified must be of the same size. Moreover, to ensure an acceptable quality of the solution to this problem, it is assumed that the image data contains objects of the same (similar) size, which are located in the same image area.

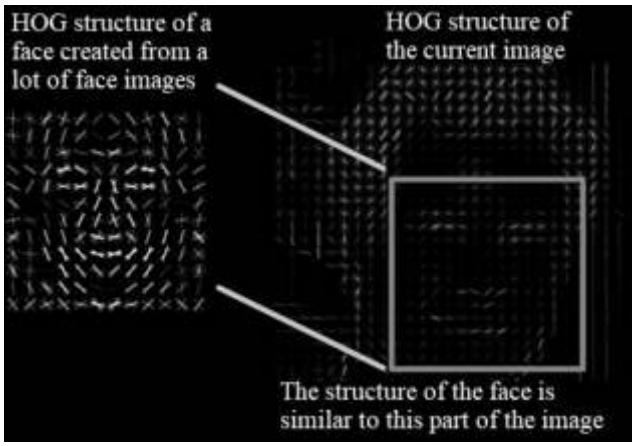


Figure 2.11: Example of the *HOG*-structure of the face

The HOG-descriptor from the source image is calculated as follows:

1- Color normalization and gamma correction are performed.

2- The gradient values are calculated vertically and horizontally.

3- The image is divided into a uniform grid of cells.

The basic unit of a HOG-descriptor is a block, a rectangular area of image pixels of a given size. A block consists of cells that are assigned a histogram of directions (inclination relative to the horizontal) of gradients. The HOG-descriptor is the vector of the components of the normalized histograms of cells from all areas of the block. As a rule, blocks overlap, that is, each cell is included in more than one final descriptor.

2.14 TensorFlow Hub & Transfer Learning

The TensorFlow library for many years has become the most ubiquitous open-source deep learning [70]. It does not only support high-performance computation through its cloud service it also has a well-established community for the library maintenance and update. The library of TensorFlow Hub is for reusable machine learning modules. The module which reusable machine learning module is a self-contained piece of a TensorFlow graph, it can be reused across different tasks in a process along with its weights and assets are known as transfer learning [71]. If it is compared to a neural network module with the normal training process, a transfer learning module is often trained with a smaller dataset, transfer learning has many other advantages such as increasing training speed and improved generalization.

If an image recognition module needs to be developed from scratch, usually, it requires hundreds or more of Graphics Processing Unit (GPU) hours [71]. The training dataset size can be reduced greatly by applying transfer learning on a trained module, a technique such that, can be used to solve the classification issues with comparatively small datasets. The approach of replacing transfer learning instead of the original layer and creates a new layer to classify new labels is shown in Figure (2.12).

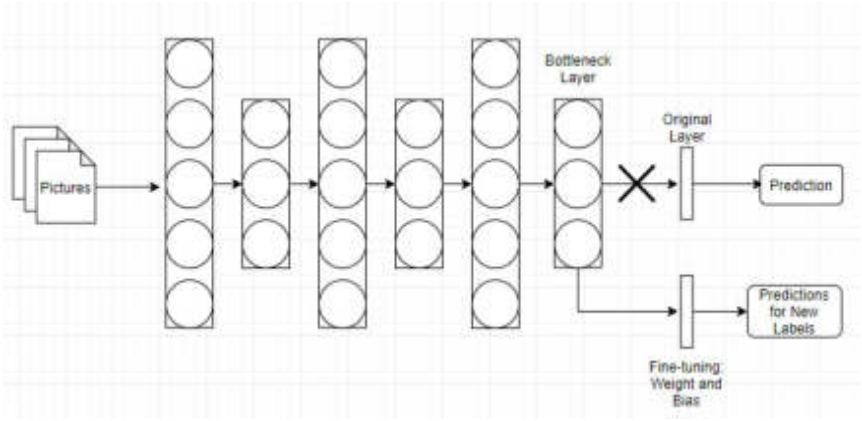


Figure 2.12: Transfer learning module. The original layer is replaced by a new layer which its weight and bias are fine-tuned for images classifying.

2.15 Results normalization

Often needed to compare sets of result scores gained from different scales. This needs the "elimination" of the measurement unit, this is called to normalize operation of the data. The normalization is divided into two main types, the first normalization type results from linear algebra, which handles and manages the data as a vector in a multidimensional space. Thus, to normalize such data is to transform the vector of data into a new data vector that its norm (length) is equal to 1. The second normalization type originates from the statistics, it eliminates the measurement unit by transforming the data into new scores with a standard deviation of one and a mean of zero. These transformed scores are called as Z-scores.

2.15.1 The norm of a vector

The *norm* of a vector in linear algebra measures the *length* which is egalitarian to the Euclidean distance of the endpoint of this vector to the vector space origin [72]. From the Pythagorean Theorem, this quantity is computed as the square root of the sum of the squared point of the vector. Also known as the Euclidean norm. This is a widely used norm in Machine learning which is used to calculate the root mean squared error.

$$\|x\|_2 = (\sum_i x_i^2)^{1/2} \quad (2.3)$$

So, for a vector *x,* the Norm would become:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \end{bmatrix}$$

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_i^2}$$

2.15.2 Normalizing with the norm

To normalize $\tilde{x}$, divide each element by $\|x\|$. The normalized vector denoted $\tilde{x}$, is equal to:

$$\tilde{x} = \frac{x_i}{\|x\|} \quad (2.4)$$

Chapter Three

The Proposed System

CHAPTER THREE
THE PROPOSED SYSTEM

3.1 Introduction

In this chapter, the implementation of the proposed DeepFake video detection system will be presented, an object-oriented feature of "Python version 3.6" was utilized all through the venture and endeavored to compose modular code that is easy to extend for future study. In the composition of the development classes, the base necessity requirement was used to make a set of classes with a typical interface from which it could separate a segment of the test framework. That would set the uniform interface highlights of the specific development classes, furthermore functions, and variables that would be shared by the child classes.

3.2 The proposed system

The proposed detection system distinguish out the fake faces in videos with good accuracy and reliable percentage. The Haar Wavelet has been used to determine the blur extent of the ROI and the surrounding area. A ResNet-50 neural network is added to robust the result. The proposed system consists of four fundamental stages: First, the video pre-processing; Second, the face detection stage; third, the Haar Wavelet for blur extent detection, and finally, the ResNet-50 for fake percentage estimation robustness. The main components of the system are shown in Figure (3.1).

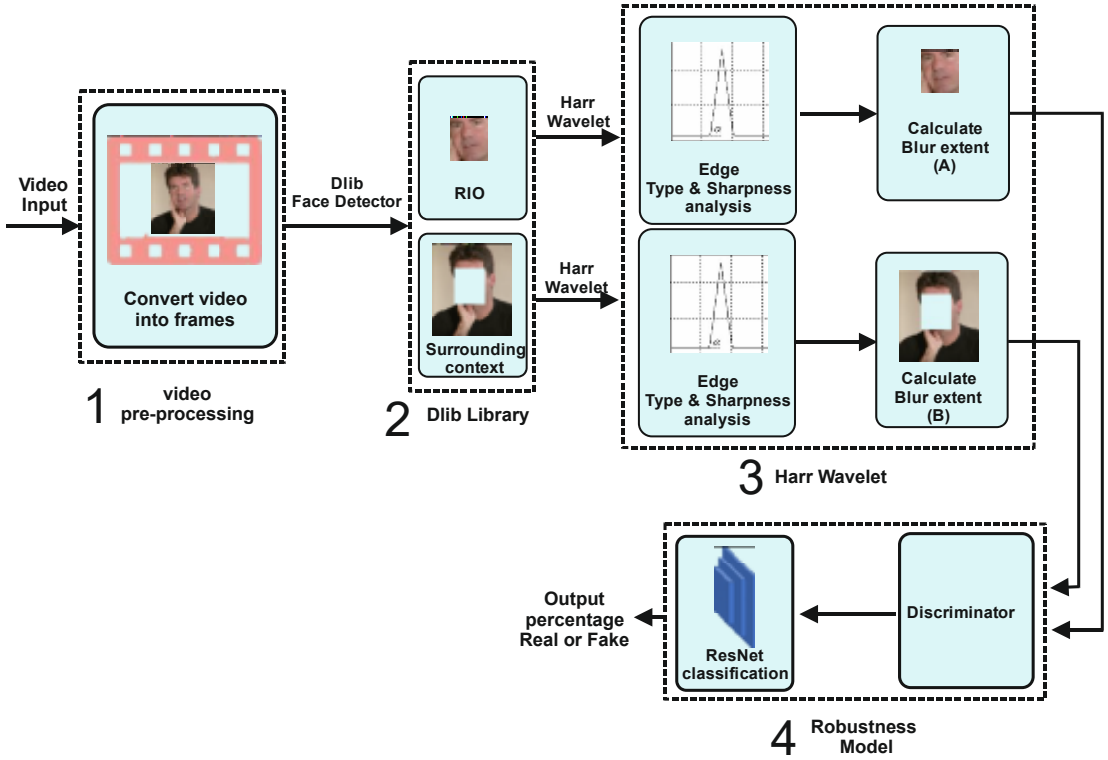


Figure 3.1: General block diagram illustrates the system workflow.

- **The video pre-processing stage:** In this stage the preparing images for further analysis by converting the video into image frames. Each frame is converted into a grayscale color, and de-noised.
- **The face detection stage**: Dlib library is used for face and facial landmark detection. This detector is based on the histogram of oriented gradients (HOG) and linear SVM as described in chapter 2.
- **Blur extent detection stage:** A blur detection scheme is been utilized, based on the edge type and sharpness analysis using the Haar Wavelet Transform. This scheme can determine the presence of blur in an image and also estimate its extent.
- **Synthesized Videos Detecting Stage:** ResNet-50 is used to detect synthesized videos by exploiting the face warping artifacts for fake percentage estimation robustness.

3.2.1 Video pre-processing stage

The pre-processing stage is the first step, where the video is converted into a stream of an array containing the image frames. This preparation is essential for the next steps, where the process will be on each frame of the video.

In this step colored (RGB) images frames of the video are converted into a gray level too. To reduce the time of processing, a grayscale image is used on the entire process instead of the color image. This is done because a three-channel image (color image) is time-consuming, so it was converted into a single channel (grayscale) image. For converting the RGB image frame, a function Cv2.Cvt.Color (imgin, imgt1, CV_BGR2GRAY) in OpenCV library is used.

Image de-noising is another step in the pre-processing stage, Image de-noising aims to subdue noise as much as could be while safeguarding and securing image features. The examination and analysis of multiresolution are carried on by the Wavelet Transform which is a strong tool to fulfill this purpose. In the domain of Orthonormal Wavelet, most of the data of the image are implicated in the largest coefficients of the Wavelet, whilst noise is uniformly diffused all over coefficients. Furthermore, the white Gaussian noise stays the same after the orthogonal transformation. One of the de-noising approaches, setting the value of the smallest coefficients to equal zero and then shrinking the rest over a certain value of threshold as shown in the algorithm (3.1) and (3.2).

Algorithm (3.1): Wavelet de-noising.
Input: Video frames **Output:** De-noised Video
Begin **Step1: Convert image to gray** \\(Preprocessing) gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) **Step2: Apply wavelet decomposition for 4 levels** coeffs = pywt.wavedec2(gray_image, 'haar', level=4)

Step3: Get the approximate Coefficients(LL) \\ detail coefficients for each level(LH,HL,HH)

```
[cA, (cH4, cV4, cD4), (cH3, cV3, cD3), (cH2, cV2, cD2), (cH1, cV1, cD1)] = coeffs
```

Step4: Apply SURE Shrink to HH coefficients for each level

```
cD4 = self.Sure_Shrink(cD4)
cD3 = self.Sure_Shrink(cD3)
cD2 = self.Sure_Shrink(cD2)
cD1 = self.Sure_Shrink(cD1)
```

Step5: Reconstruct image without noise

```
res = pywt.waverec2([cA, (cH4, cV4, cD4), (cH3, cV3, cD3), (cH2, cV2, cD2), (cH1, cV1, cD1)], 'haar')
```

Step6: Get and show the noise features of this frame

```
pure_noise = gray_image - res
cv2.imshow('image1',self.pure_noise)
cv2.waitKey(20)
```

Algorithm (3.2): SURE Shrink.

Input: Wavelet Coefficients

Output: SURE Shrinked coefficients

Begin

Step1: Set parameters

```
count = 0                    \\ variable for try lambda
minsure = 1e18               \\ variable to minimize the SURE equation
t = 0                        \\ the lambda that minimized SURE equation
size = coefficients.shape    \\ get the size of coefficients
```

Step2: Convert coeff from 2D to 1D \\ the absolute value in allcoeff

```
numofcoefficients = size[0] * size[1]
allcoeff = []
com_sum = 0
com_sum_coeff = []
for listcoeff in coefficients:
```

```
        for coeff in listcoeff:
            allcoeff.append(abs(coeff))
    allcoeff.sort()          // sort this list

Step3: Get the commulative sum of value^2 in allcoeff \\and put it in
                                                        com_sum_coeff
    for coeff in allcoeff:
        com_sum = com_sum + (abs(coeff) * abs(coeff))
        com_sum_coeff.append(com_sum)
    limit = math.sqrt(2 * math.log10(numofcoefficients))\\The limit for
                                                          trying lambda
    while (count <= limit):
        sum = 0
        ind = bisect.bisect_right(allcoeff, count) \\get the index of abs(coeff)
                                                    that is greater than count
        num_gt = len(com_sum_coeff) - ind  \\ number of abs(coeff)s that
                                                    are greater than count
        sum = sum + (num_gt * count * count) \\ apply the equation of SURE
                                                   Shrink
        if (ind > 0):
            sum = sum + com_sum_coeff[ind - 1]
        sure = numofcoefficients + sum - 2 * ind
        if sure < minsure:  # Minimization condition for SURE equation
            minsure = sure
            t = count
        count = count + .01  # add small value to try more lambdas
    coefficients = pywt.threshold(coefficients, t, 'soft') \\ apply soft
threshold
                                                        using lambda(t)
    return coefficients
```

3.2.2 Face Detection Stage

The Dlib library is used in this section. Dlib Computer Library is a cross-platform general-purpose library written in the C++ programming language. Its design is excessively influenced by ideas from a design by contract and component-based software engineering. It is a set of independent software components. It is open-source software released under a Boost Software License.

This library is utilized to detect faces and their landmarks in the image frame, if there is none it will skip the frame, if more than one face is found, all of them will be tested in the next stage, algorithm (3.2). If one of the faces in the frame is fake, the whole frame counted as fake.

Algorithm (3.2): Face detection ROI.

Input: Image frame
Output: Array of faces in image frames (ROIs)

```
Begin
Step1: Loop for the length of the frames in the video
    for i in range(1, len(self.Video)):

Step2: Check how many faces in an image // If one of the faces in the
       frame is fake, the whole frame counted as fake.
    try:
        im = Video[22]
        face_info=lib.align(im[:,:,(2,1,0)],front_face_detector,
        lmark_predictor)
        if len(face_info) == 0:            // No face has been found
           logging.warning('No faces are detected.')
        else:
           logging.info('[1] faces are detected.'.format(len(face_info)))

Step3: Extract ROI and facial landmark // Cut the face region
    for _, point in face_info:
        rois = []
        roi, _ = lib.cut_head([im], point)

Step4: Append ROI to the ROIs array
        rois.append(roi) // Add the image to rois array
```

When selecting the context around the ROI, a mask is used to hide the ROI as illustrated in the algorithm (3.3).

The benefit of using this algorithm is to get accurate readings of the blur extent of the surrounding area without the interference of the blur extent of the ROI value.

Algorithm (3.3): Masking face area.

Input: Image frame
Output: The frame image with a mask on ROI

Begin

Step1: The procedure for face detection \\Find face in the image (frame)

```
def get_face_loc(im, face_detector, scale=0):
    faces = face_detector(np.uint8(im), scale)
    face_list = []
```

Step2: Check if faces exist and find the location

```
    if faces is not None or len(faces) > 0: // if face is found append it
      for i, d in enumerate(faces):
        try:
            face_list.append([d.left(), d.top(), d.right(), d.bottom()])
        except:
            face_list.append([d.rect.left(),d.rect.top(),d.rect.right(),
        d.rect.bottom()])
```

Step3: Return the face location points

```
    return face_list
```

Step4: Add mask // add mask to the face area

```
def get_all_face_mask(shape, face_list):
    mask = np.zeros(shape)
    for _, points in face_list:
    mask += np.int32(get_face_mask(shape[:2], points))
    mask = np.uint8(mask > 0)
    return mask
```

3.2.3 Blur Extent Detection Stage

One of the best techniques for the signals analyzing used today is the HWT. For the edge detection of an image, the Haar Wavelet Transform gives a facility to choose the size of the picture subtleties that will be detected. Wavelets Transform can isolate the lower frequencies from higher frequencies effectively, which is prime significant for edge location. Furthermore, the wavelet scale sets the size of the identified edges. In the Discrete Wavelet Transform, numerous signals go through a wavelet filter for the decision of the scale. For a 2-Dimension image, wavelet examination is completed as far as vertical and horizontal function, and the edges are distinguished independently. In this work, HWT is used for blur extent detection as illustrated in the algorithm (3.4):

Algorithm (3.4): Wavelet for blur detection.

Input: Image frame
Output: Blur percentage and its extent

Begin

Step1: Get the size of the image

```
M, N = img.shape
```

Step2: Crop input image to be 3 divisible by 2

```
img = img[0:int(M/16)*16, 0:int(N/16)*16]\\ The new image size will be
                                          divisible by 8,4, and 2.
```

Step3: Compute Haar wavelet of the input image

```
LL1,(LH1,HL1,HH1)= pywt.dwt2(img, 'haar') \\ Application of 2D haar to
                                             LL1
LL2,(LH2,HL2,HH2)= pywt.dwt2(LL1, 'haar') \\ Another application of
                                             2D haar to LL2
LL3,(LH3,HL3,HH3)= pywt.dwt2(LL2, 'haar') \\ Another application of
                                             2D haar to LL3
```

Step4: Construct the edge map in each scale

```
E1 = np.sqrt(np.power(LH1, 2)+np.power(HL1, 2)+np.power(HH1, 2))
```

```
E2 = np.sqrt(np.power(LH2, 2)+np.power(HL2, 2)+np.power(HH2, 2))
E3 = np.sqrt(np.power(LH3, 2)+np.power(HL3, 2)+np.power(HH3, 2))
M1, N1 = E1.shape
```

Step5: Sliding window size level 1 \\ widow size 8X8

```
sizeM1 = 8
sizeN1  = 8
```

Step6: Sliding windows size level 2 \\ widow size 4X4

```
sizeM2 = int(sizeM1/2)
sizeN2 = int(sizeN1/2)
```

Step7: Sliding windows size level 3 \\ widow size 2X2

```
sizeM3 = int(sizeM2/2)
sizeN3 = int(sizeN2/2)
```

Step8: Number of edge maps, related to sliding windows size

```
N_iter = int((M1/sizeM1)*(N1/sizeN1))
Emax1 = np.zeros((N_iter))
Emax2 = np.zeros((N_iter))
Emax3 = np.zeros((N_iter))
count = 0
```

Step 9: Sliding windows index of level 1 \\initialize widow 1 starting point

```
x1 = 0 \\ For window 1 horizontal stride
y1 = 0 \\ For window 1 vertical stride
```

Step10: Sliding windows index of level 2\\initialize widow 2 starting point

```
x2 = 0 \\ For window 2 horizontal stride
y2 = 0 \\ For window 2 vertical stride
```

Step 11: Sliding windows index of level 3\\ initialize widow 3 starting point

```
X3 = 0 \\ For window 3 horizontal stride
Y3 = 0 \\ For window 3 vertical stride
```

Step12: Sliding windows limit on the horizontal dimension

```
Y_limit = N1-sizeN1      \\ Indicates the limits of the sliding window
```

```
while count < N_iter: \\Get the maximum value of slicing windows over
                          edge maps in each level
  Emax1[count] = np.max(E1[x1:x1+sizeM1,y1:y1+sizeN1])
  Emax2[count] = np.max(E2[x2:x2+sizeM2,y2:y2+sizeN2])
  Emax3[count] = np.max(E3[x3:x3+sizeM3,y3:y3+sizeN3])

  if y1 == Y_limit: \\ If sliding windows ends horizontal direction move
                       along vertical direction and resets horizontal
                       direction
    x1 = x1 + sizeM1
    y1 = 0
    x2 = x2 + sizeM2
    y2 = 0
    x3 = x3 + sizeM3
    y3 = 0
    count += 1

Step13: windows moves along horizontal dimension \\windows moves
                                                   along horizontal
                                                   dimension
  else:
    y1 = y1 + sizeN1
    y2 = y2 + sizeN2
    y3 = y3 + sizeN3
    count += 1

Step14: Get edge point \\ Edge points which is greater than the threshold
 EdgePoint1 = Emax1 > threshold;
 EdgePoint2 = Emax2 > threshold;
 EdgePoint3 = Emax3 > threshold;

 \\ Rule 1 Edge Points
 EdgePoint = EdgePoint1 + EdgePoint2 + EdgePoint3
 n_edges = EdgePoint.shape[0]

 \\ Rule 2 Dirak-Structure or Astep-Structure
 DAstructure = (Emax1[EdgePoint] > Emax2[EdgePoint]) *
```

```
(Emax2[EdgePoint] > Emax3[EdgePoint]);

\\ Rule 3 Roof-Structure or Gstep-Structure
RGstructure = np.zeros((n_edges))
for i in range(n_edges):
    if EdgePoint[22] == 1:
        if Emax1[22] < Emax2[22] and Emax2[22] < Emax3[22]:
            RGstructure[22] = 1

\\ Rule 4 Roof-Structure
RSstructure = np.zeros((n_edges))
for i in range(n_edges):
    if EdgePoint[22] == 1:
        if Emax2[22] > Emax1[22] and Emax2[22] > Emax3[22]:
            RSstructure[22] = 1

\\ Rule 5 Edge more likely to be in a blurred image
BlurC = np.zeros((n_edges));
for i in range(n_edges):
    if RGstructure[22] == 1 or RSstructure[22] == 1:
        if Emax1[22] < threshold:
            BlurC[22] = 1

Step15: Calculate the percentage
  Per = np.sum(DAstructure)/np.sum(EdgePoint)

Step16: If No RSstructure
  if np.sum(RSstructure) == 0:
    BlurExtent = 100
  else:
    BlurExtent = np.sum(BlurC)/np.sum(RSstructure)

Step17: Return the value of the percentage and blur extent
    return Per, BlurExtent
```

This approach functions based on thresholding the pixels within a specified window. The approach uses the average value of pixels within a window as the

threshold and it is likened to an averaging filter. Depending on the presence, absence, and the value of the pixels within that window for specifying the threshold. The detector of an edge has restrictions in detecting edges in noisy and faint images. Significative edges are probably to be detected in an unnoisy image while edges could be neglected in a noisy image, hence, the approach cannot adequately analyze noisy images. Most of the images have some degree of noise; therefore, a reliable edge detection technique must be sensitive to edges and insensitive to noise.

3.2.4 Robustness Stage

Residual Network 50 is a Convolutional Neural Network that is 50 layers deep. The network input image size is 224×224. ResNet-50, see Figure (3.2 and 3.3), is an efficient framework in training a deeper neural network. Like any deep learning framework, the layers are nothing but nonlinear processing units for feature extraction and transformation. They also include hidden layers of an artificial neural network and sets of propositional formulas.

In general, in a Deep Convolutional Neural Network, a few layers are stacked and are prepared for the task needing to be done. The system of the network learns several (significant/mid/low) level features toward the end of its layers. In Residual Learning, rather than attempting to get learned with certain features, it attempts to become familiar with certain residuals. Residual can be just comprehended as the deduction of features gained from the input of that layer. Using the shortcut connection Residual Network does this (connecting the input directly from the n^{th} layer to the $(n+x)^{th}$ layer). The training of such a form of networks has been proved easier than training simple deep convolutional neural networks, and also the problem of degrading accuracy is resolved.

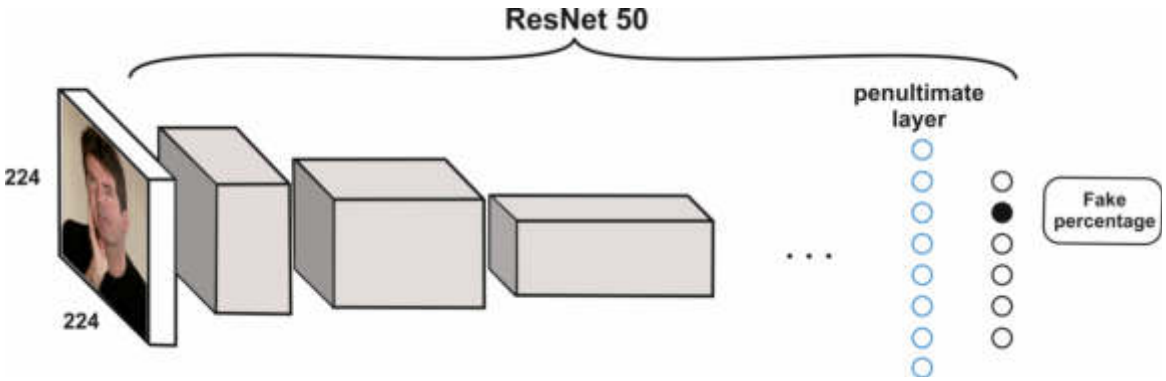


Figure 3.2: ResNet-50 architecture.

Figure 3.3: A block diagram representation of pre-trained Resnet-50 architecture.

3.2.4.A ResNet-50 configuration

The Initialization of the pre-trained ResNet-50 which has been used in this model is shown in table (3.1).

Table 3.1: ResNet-50 configuration.

BASE_NETWORK :	ResNet50
PRETRAINED_MODELS :	True
IMG_SIZE :	[224, 224, 3]
NUM_CLASSES :	2
CLASS_WEIGHTS :	1
TRAIN :	{'LEARNING_RATE': 0.001, 'DECAY_RATE': 0.95, 'NUM_EPOCH': 20

The term "Learning_Rate" is a hyper-boundary used to control the difference in the model because of the evaluated blunder each time the model

updates its weights. Choosing the best learning rate is a real big challenge, if the value that is set as learning rate is very small, it might bring about a long training process that could get to a deadlock state, while a picking value that is set as learning rate is very big may result in an unsettled training process.

The "Decay_Rate" learning value is used as a technique for modern neural networks training. It begins with a large value of Learning_rate and then decays it multiple times.

The term "Epoch" is a measure of the number of times that all of the training vectors are used once to update the weights. For batch training, all of the training samples pass through the learning algorithm simultaneously in one epoch before the weights are updated.

3.2.4.B Using the ResNet-50 for Detecting Artifacts

The ResNet-50 is used to exploit the face warping artifacts, the resolution inconsistency in fake faces after ensuing compression venture to create the final image or video frames in the videos of DeepFake. Consequently, a Convolutional Neural Network (CNN) is proposed to distinguish the existence of such artifacts from the identified face area and its encompassing zones. The used ResNet-50 is a pre-trained neural network, it has been trained on images of faces gathered from the web (about 24,442 JPEG as positive examples of face images). The used technique is to detect the artifacts of the affine face warping steps in the DeepFake creation pipeline, the negative samples generation process is simplified by simulating the affine face warping step straightly as illustrated in Algorithm (3.5).

The accompanying strides to create negative samples to prepare the CNN model is as follow:

- By the use of the Dlib software package, the faces zone is detected and extracted from the whole images.
- The next step would be aligning the faces into various scales and pick one random scale, by the use of Gaussian blur with kernel size (5×5) it is

smoothed. This procedure aims to produce more resolution cases of the affine warped faces, in this way it can simulate different kinds of resolution inconsistency.

- The outcome of the previous step, the face which has been smoothed will undergo an affine warp to the sizes of the genuine faces to simulate the DeeFake creation pipeline's artifacts.

Algorithm (3.5): Generate negative examples.

Input: Image frame
Output: Put a mask on ROI

Begin

Step1: Get image size

```
h, w = imgs[0].shape[:2]
x1, y1 = np.min(point, axis=0)
x2, y2 = np.max(point, axis=0)
```

Step2: Change the dominations \\create minimum bounding for face area

```
delta_x = (x2 - x1) / 8
delta_y = (y2 - y1) / 5
x1_ = np.int(np.maximum(0, x1 - delta_x))
x2_ = np.int(np.minimum(w-1, x2 + delta_x))
y1_ = np.int(np.maximum(0, y1 - delta_y))
y2_ = np.int(np.minimum(h-1, y2 + delta_y))
```

Step3: Apply changes on the image \\ change the size of bounding box

```
imgs_new = []
    for i, im in enumerate(imgs):
        im = im[y1_:y2_, x1_:x2_, :]
        imgs_new.append(im)
        locs = [x1_, y1_, x2_, y2_]
```

3.2.5 Detecting Fake Video

The algorithm (3.6) illustrate the procedure of detecting fake and real frames of the target video. Where the probability of being fake is calculated

according to the norm of the results of all frames using formula (2.3) and (2.4). The threshold (0.5) is not a fixed value.

Algorithm (3.6): Detect video if fake or real

Input: Fake probability
Output: Detect video if fake or real

```
Begin
Step1: Detect fake frames
    if prob<0.5: \\ Fake case
        draw_face_rect(im, point,0,255,0)
        draw_face_landmarks(im, point)
        print ("posibillty of being Fake : ",prob,"     Real\n")
        xmin, ymin = np.min(point[:, 0]), np.min(point[:, 1])
        font = cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(im, str(prob) ,(xmin,ymin-8), font, .5,(0,255,0) ,1,
        cv2.LINE_AA)
        cv2.putText(im, str(i) ,(10,20), font, .5,(0,255,0),1,cv2.LINE_AA)
        cv2.putText(im, "/"  ,(38,20), font, .5,(0,255,0),1,cv2.LINE_AA)
        cv2.putText(im, str(len(self.Video)) ,(45,20), font, .5,(0,255,0),1,
        cv2.LINE_AA)

Step2: Detect real frames
    elif prob>0.5:\\ Real case
        draw_face_rect(im, point,0,0,255)
        draw_face_landmarks(im, point)
        print ("probability : ",prob,"     Forged\n")
        xmin, ymin = np.min(point[:, 0]), np.min(point[:, 1])
        font = cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(im, str(prob) ,(xmin,ymin-8), font, .5,(0,0,255),1,
        cv2.LINE_AA)
        cv2.putText(im, str(i) ,(10,20), font, .5,(0,255,0),1,cv2.LINE_AA)
        cv2.putText(im, "/"  ,(38,20), font, .5,(0,255,0),1,cv2.LINE_AA)
        cv2.putText(im, str(len(self.Video)) ,(45,20), font, .5,(0,255,0),1,
        cv2.LINE_AA)
```

Chapter Four

The Experimental Test and Results

CHAPTER FOUR
THE EXPERIMENTAL TEST AND RESULTS

4.1 Introduction

The performance of the proposed method in this thesis is evaluated by conducting a set of experiments done on the UADFV Dataset [12]. These experiments and their results are described in full detail in this chapter. The experiment is conducted using a computer running Windows10 operating system with an Intel® Core™ i7-7700HQ processor with a memory of 16GB. The computer also contains an Nvidia GTX1080Ti with 8GB of memory that is used to accelerate the neural networks' computations. The method is implemented and evaluated using the Python programming language version 3.6.

4.2 UADFV dataset

DeepFake video detection using the Haar Wavelet Transform system was tested on the UADFV dataset which is available publicly. The UADFV dataset is made up of 98 videos, 49 counterfeit videos, and 49 original videos and these videos contain around 32,752 frame sequences, each frame in the video will be extracted as a single image at a frame rate of 30 frames per second, next, the frames will be passed into the facial recognition Dlib module for facial extraction. In the dataset, The length of each video ranges between 2 up to 44 seconds, the average of all videos length is around 11.26 seconds.

4.3 Experimentation results

As illustrated below, the experimental results will be classified into two major types depending on the detection of blur extent by wavelet transformation. In the beginning, all the frames image data will go through the preprocessing stages, the image is classified into two main categories, images that labeled as real (original) and images that are labeled as fake (counterfeit). The frames

images of each video are classified as real or fake according to the percentage of being fake, less the 0.5 is classified as real images and above 0.5 is classified as fake images, each one will be labeled with its result and the norm is calculated for the whole video (as described in 2.13). The difference between exceedingly of these frame images is very difficult to be distinguished by the naked eye due to the hidden artifacts and low resolution on detecting the forgery videos.

In the carried experiment on the UADFV dataset, the proposed detection system has achieved a high detection rate ratio (100%) to localize and detection of DeepFake videos. The DeepFake detection possibility ratio is defined as the number of the mean average of all frames possibility detection of being fake in each video.

The results of each video bellow are represented with the following parameters:

- The name of the video which is under test: the video files of the UADFV are numbered from ‘0000’ to ‘0048’. The name of the dataset videos is described as the file name with ‘_fake’ extension which refers to fake video. The real video file names are without the ‘_fake’ extension (E.g. 0000.mp4 is a real video and 0000_fake.mp4 is a fake video)
- The ‘Number of REAL frames chains’ and the ‘Number of FAKE frames chains’, both show the number of contiguous frames of the real and fake detected frames, only longer than 30 continuous frames are counted, which represent 1 second of a video scene. In some cases the chain is cut by one or two frames, this case is taken under consideration.
- The ‘Max length of REAL chain’ AND ‘Max length of FAKE chain’, both show the longest chain in the two cases real and fake cases.
- The ‘Total number of REAL frames’ and ‘Total number of FAKE frames’, both show the total frame count for each video at the rate of 30 frames per second.

- Possibility of being Fake, The possibility of being fake is calculated by taking the mean average of all the possibilities of the total frames. The higher possibility percentage refers to the high possibility of being fake and vice versa, the smaller percentage of possibility refers to real.

  The threshold which distinguishes between real and fake is 0.5, any value under 0.5 is considered as real and any value above 0.5 is considered as fake. This is not a fixed threshold, it can be changed due to the circumstances.

- The three graphs for each video are described as follows:
  - The first graph represents the relation between the blur extent of the real and fake frames as shown in Figure (4.1). It is very noticeable that in real cases the amplitude of both real and fake have the same pattern. The only difference in the amplitude value, but the case of fake videos the pattern is never the same. This illustrates that in real video cases the region of the face and surrounding context almost have the same blur function.

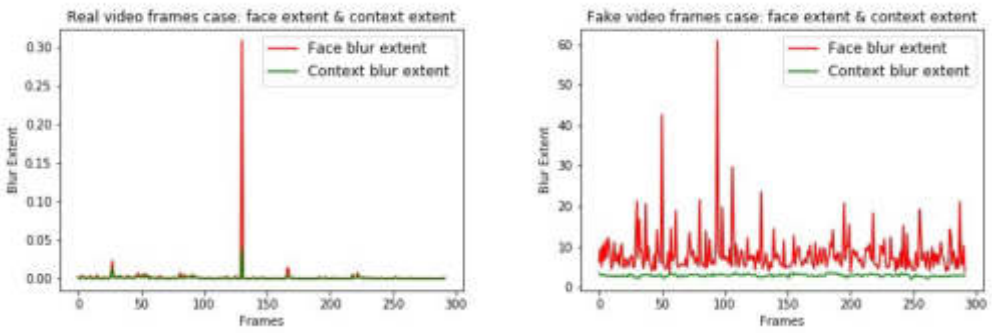


Figure 4.1: The difference between the real and fake patterns of blur extent in real and fake cases. On the left, the results of the real video 0000.mp4. On the right, the fake video 0000_fake.mp4.

  - The second graph represents the possibility of all frames.
  - The third graph represents the difference between the bure extent between the ROI and context region. It is obvious that in real video cases it is relatively small.

4.4 Results Evaluation

The test and result of the whole experiment done on the UADFV dataset, each test was carried on the real video and the fake version of the same video. The experiment result gave good and accurate results with a few nuances.

The experimental results are classified into two main groups depending on some criteria as described in each group:

Group (1): This group of results holds the real-video and fake-video data who have a significant low possibility of being a fake-video for real video test and high possibility of being a fake-video for fake ones. The table (4.1) shows the videos which were less than 0.1 possibilities of being a fake-video and larger than 0.8 for fake-videos. The details of the parameters of each test in both cases (real-video shaded with light green color and fake version of the same video shaded in light red color). In each testing case, the count of real and fake frames, the max length of the chain count, and the total number of frames are shown. The table also shows the percentage of probability of being fake for the two versions of the tested video.

Table 4.1: The parameters result in real and fake videos of the group (1).

Video name	Real/fake	Real frames			Fake frames			Probability of being Fake
		Number of REAL frames chains	Max length of REAL chains	Total number of REAL frames	Number of FAKE frames chains	Max length of FAKE chains	Total number of FAKE frames	
0000	Real	1	292	292	0	0	0	0.0014450
	Fake	0	1	2	1	290	290	0.9907342
0002	Real	1	244	244	0	0	0	0.0008504
	Fake	0	0	0	1	244	244	0.9960736
0003	Real	1	148	148	0	0	0	0.0040497
	Fake	0	0	0	1	148	148	0.9956517
0008	Real	1	259	259	0	0	0	0.0019239
	Fake	0	0	0	1	259	259	0.9964447
0009	Real	1	295	295	0	0	0	0.0078645

	Fake	0	1	2	1	293	293	0.9942743
0010	Real	1	454	454	0	1	1	0.0348293
	Fake	5	26	105	1	350	350	0.8955917
0011	Real	1	415	415	0	1	1	0.0214686
	Fake	0	0	0	1	416	416	0.9985050
0012	Real	1	204	204	0	2	6	0.0348293
	Fake	3	25	44	1	166	166	0.9924196
0013	Real	1	374	374	0	0	0	0.0092202
	Fake	0	0	0	1	374	374	0.9993946
0014	Real	1	204	204	0	0	0	0.0008383
	Fake	0	2	4	1	200	200	0.9980878
0015	Real	1	217	217	0	0	0	0.0007775
	Fake	0	2	3	1	214	214	0.9948997
0016	Real	1	129	129	0	0	0	0.0079271
	Fake	0	0	0	1	129	129	0.9998679
0017	Real	1	113	113	0	0	0	0.0073886
	Fake	0	0	0	1	113	113	0.9934983
0020	Real	1	111	111	0	0	0	0.0012944
	Fake	0	0	0	1	111	111	0.9955046
0021	Real	1	146	146	0	0	0	0.0009763
	Fake	0	0	0	1	146	146	0.9899419
0022	Real	1	345	345	0	0	0	0.0092905
	Fake	0	1	3	1	342	342	0.9896996
0023	Real	1	227	227	0	2	2	0.0579688
	Fake	0	0	0	1	229	229	0.9981300
0024	Real	1	226	226	0	0	0	0.0073902
	Fake	0	0	0	1	226	226	0.9994646
0025	Real	0	150	150	0	0	0	0.0092650
	Fake	0	1	2	0	148	148	0.9848901
0026	Real	1	236	236	0	0	0	0.0500271
	Fake	0	0	0	1	236	236	0.9948576
0027	Real	1	193	193	0	0	0	0.0061967
	Fake	0	2	9	1	184	184	0.9800793
0028	Real	1	128	128	0	2	2	0.0689215
	Fake	2	24	39	1	89	89	0.9313839
0030	Real	1	285	285	0	0	0	0.0006197

	Fake	0	0	0	1	285	285	0.9995751
0031	Real	1	167	167	0	0	0	0.0085000
	Fake	0	0	0	1	167	167	0.9947833
0035	Real	1	134	134	0	0	0	0.0025192
	Fake	3	31	56	3	38	78	0.8907156
0041	Real	1	1025	1025	0	0	0	0.0020856
	Fake	1	787	787	14	29	238	0.6366360
0043	Real	5	89	265	0	3	6	0.0703729
	Fake	5	223	599	15	62	337	0.6686435
0044	Real	1	1143	1143	0	1	1	0.0169384
	Fake	6	343	676	8	199	468	0.7862259
0045	Real	1	890	890	0	5	7	0.0794575
	Fake	6	69	168	3	484	727	0.9746366
0046	Real	1	685	685	0	0	0	0.0189505
	Fake	3	187	314	7	152	372	0.8705340
0047	Real	1	795	795	0	0	0	0.0059579
	Fake	8	24	126	1	669	669	0.9712068
0048	Real	1	906	906	0	1	1	0.0045685
	Fake	2	66	299	9	145	608	0.6431676

Group (2): This second group of results also holds the real-video and fake-video data who have a relatively low possibility of being a fake-video which was larger than (0.1) but still less than (0.5) and it is classified as real. Table (4.2) shows the details of the parameters of each test in both cases (real-video shaded with light green color and fake version of the same video shaded in light red color). The table illustrates the details of the parameters of the percentage of the probability of being fake for the two versions of the tested video. It is observed that the results gave accurate classification due to the larger numbers of the real frames count compared to the number of fake count of fake frames in testing real-videos case. There were no problems in classifying the fake-videos.

Table 4.2: The parameters result in real and fake videos of the group (2).

Video name	Real/fake	Real frames			Fake frames			Probability of being Fake
		Number of REAL frames chains	Max length of REAL chains	Total number of REAL frames	Number of FAKE frames chains	Max length of FAKE chains	Total number of FAKE frames	
0001	Real	2	107	176	1	15	16	0.6358514
	Fake	0	0	0	1	192	192	0.9932180
0004	Real	1	109	109	1	3	7	0.3922831
	Fake	0	0	0	1	116	116	0.9995335
0005	Real	2	149	169	1	13	18	0.4290066
	Fake	0	1	1	1	182	182	0.9790777
0006	Real	1	133	133	1	6	6	0.1586502
	Fake	0	1	1	1	138	138	0.9988409
0007	Real	1	126	126	2	20	34	0.3396877
	Fake	0	0	0	1	160	160	0.9988966
0018	Real	4	60	197	4	114	190	0.4399268
	Fake	0	0	0	1	387	387	0.9982541
0019	Real	1	160	160	1	12	24	0.4710102
	Fake	0	0	0	1	184	148	0.9958181
0029	Real	1	155	155	0	1	1	0.2109814
	Fake	1	8	13	1	143	143	0.9763031
0032	Real	1	186	168	0	2	5	0.1686206
	Fake	0	2	2	1	189	189	0.9927478
0033	Real	2	36	48	0	3	5	0.2391251
	Fake	0	1	1	2	40	35	0.9987076
0034	Real	1	255	255	1	10	17	0.4651044
	Fake	2	170	191	3	43	69	0.9376071
0036	Real	1	109	109	1	6	17	0.4072702
	Fake	1	68	68	2	45	58	0.7316802
0037	Real	1	103	103	0	3	6	0.1861137
	Fake	0	0	0	1	109	109	0.9972766
0038	Real	1	101	101	1	16	22	0.3061335
	Fake	1	6	6	1	117	117	0.9999005
0039	Real	1	111	111	1	6	6	0.2687746
	Fake	0	0	0	1	117	117	0.9999034
0040	Real	1	1206	1206	1	12	46	0.2221643
	Fake	8	15	133	1	1091	1091	0.9738446
0042	Real	1	1299	1299	0	4	23	0.1663141
	Fake	12	51	303	1	1019	1019	0.9699268

The Figures (4.2 to 4.50) illustrate the relationship between the blur extent for both cases (Real-version and Fake-version) of all UADFV dataset files.

Looking at the results shown in tables (4.1 and 4.2), there were some cases like in 0036_Fake.mp4, 0041_Fake.mp4, 0043_Fake.mp4, and 0044_Fake.mp4 the reading of the total number of real frames was bigger than the reading of the total number of fake frames, thus the classification of the video points the video is fake due to many considerations such as the big difference between the blur of RIO region and the surrounding context and the mismatch of the pattern between the blur of RIO region and the surrounding context, see Figure (4.38, 4.43, 4.45 and 4.46).

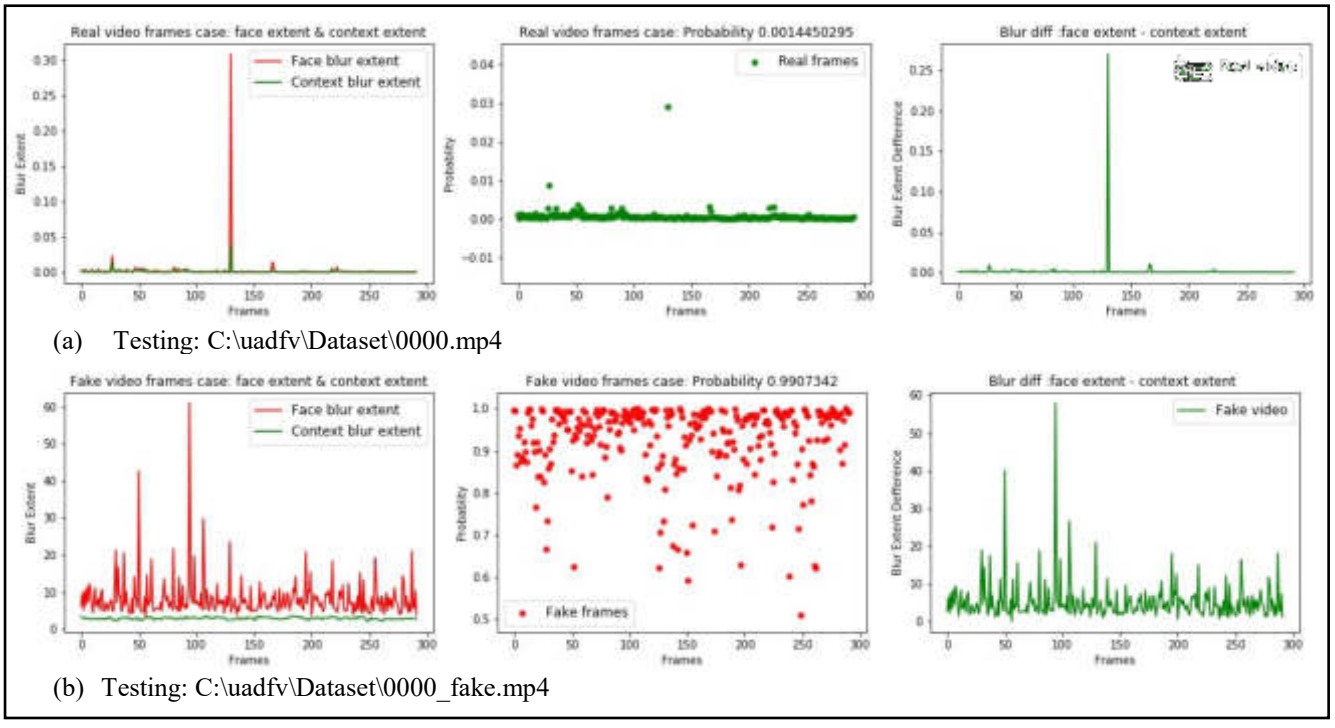
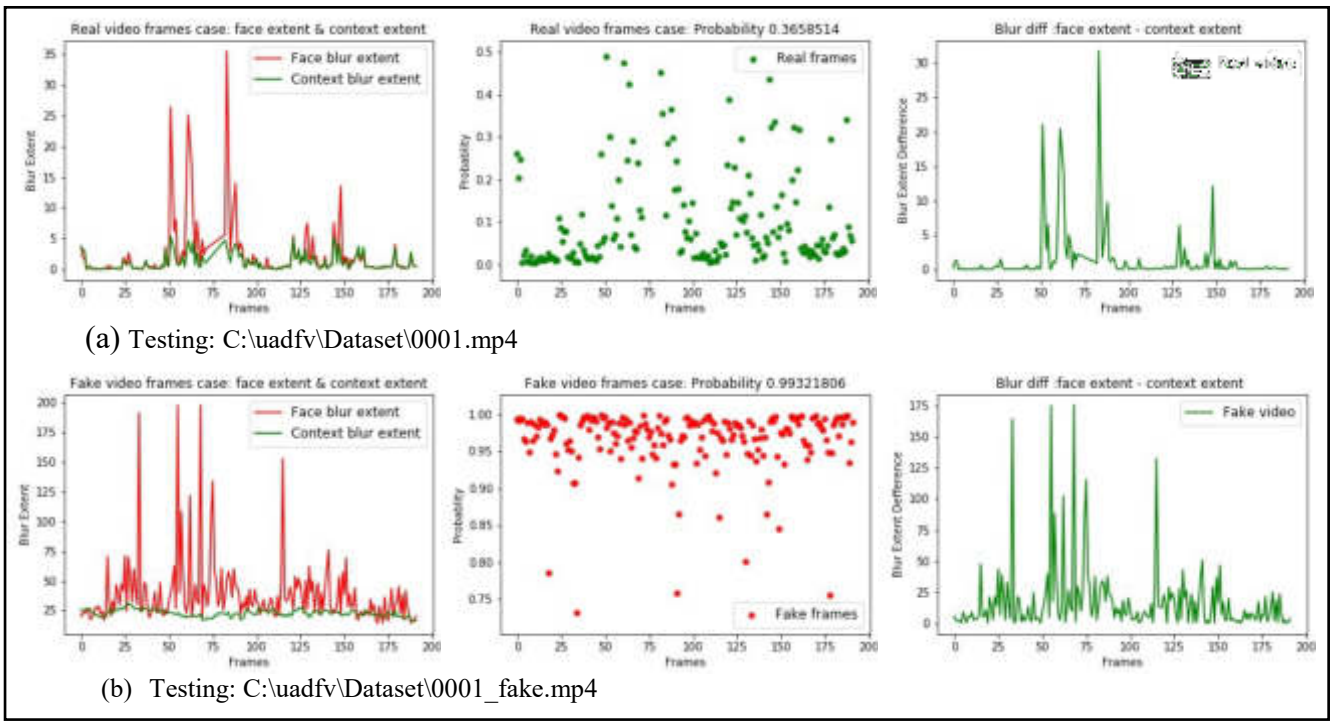


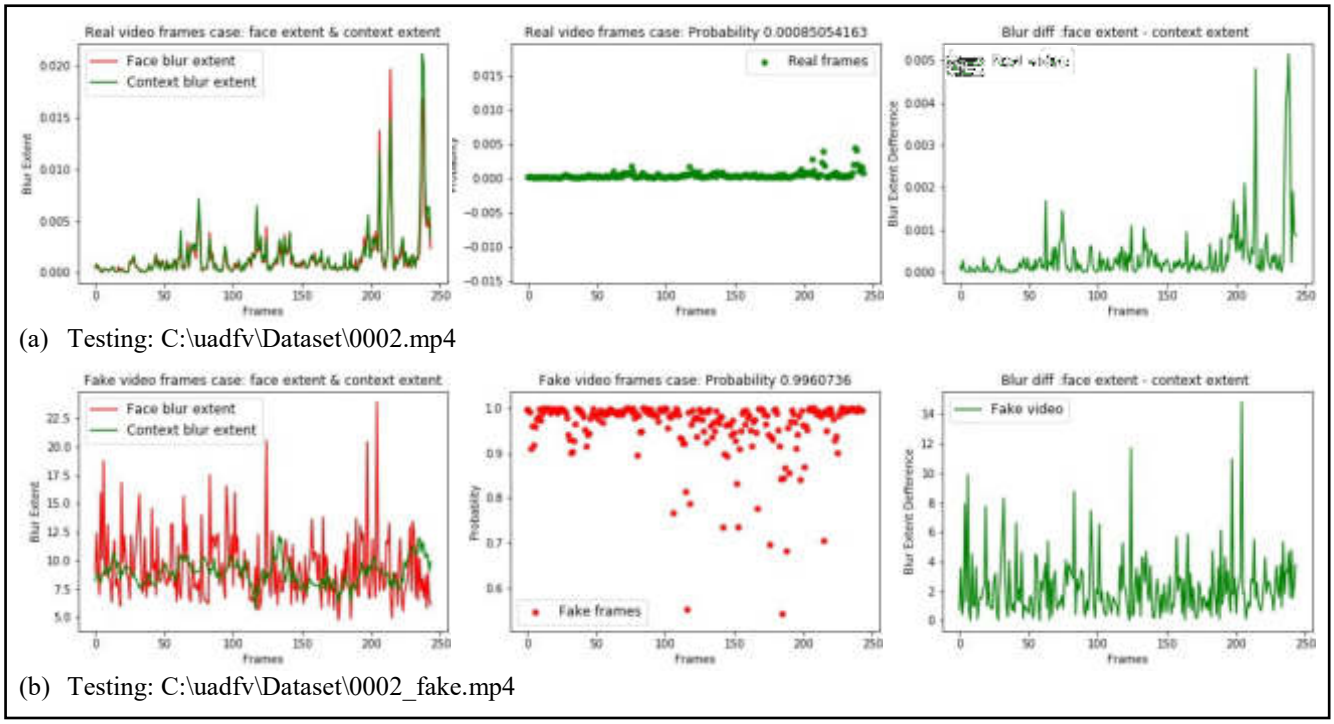
Figure (4.2): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0000.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0001.mp4

(b) Testing: C:\uadfv\Dataset\0001_fake.mp4

Figure (4.3): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0001.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0002.mp4

(b) Testing: C:\uadfv\Dataset\0002_fake.mp4

Figure (4.4): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0002.mp4". (a) Real video-version, and (b) Fake video-version.

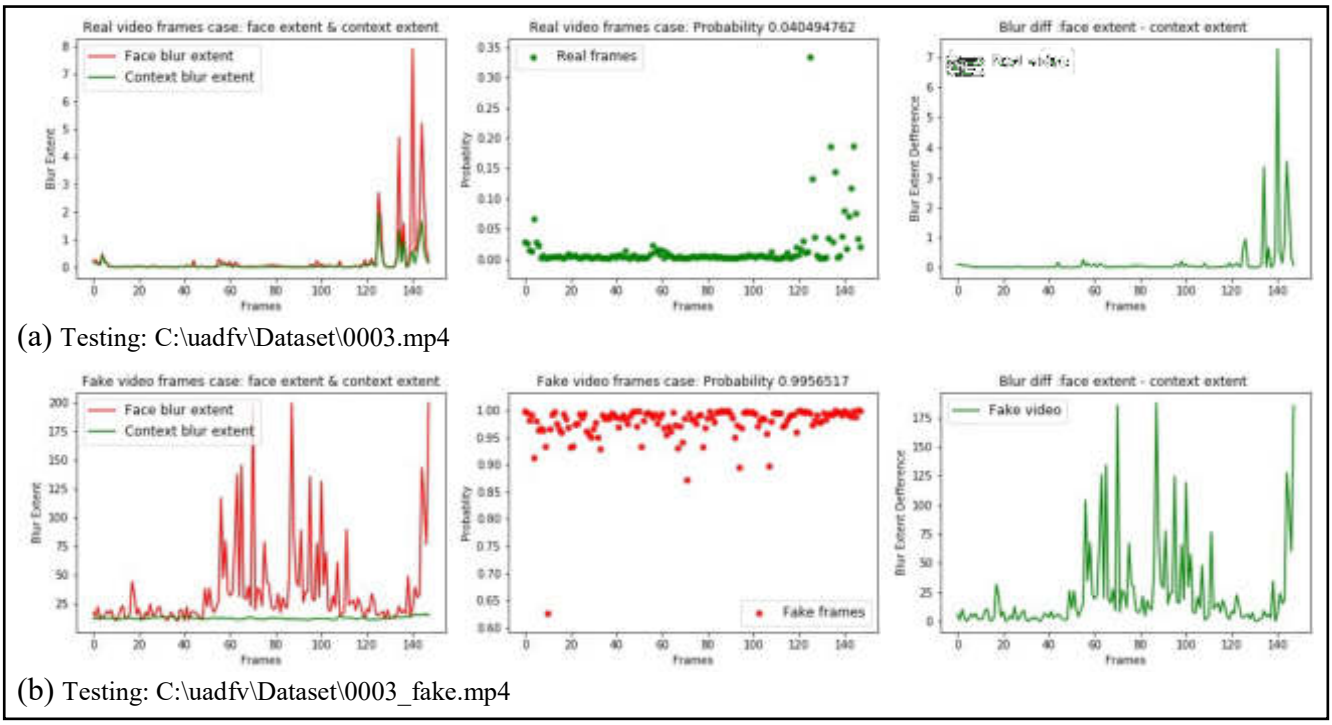


Figure (4.5): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0003.mp4". (a) Real video-version, and (b) Fake video-version.

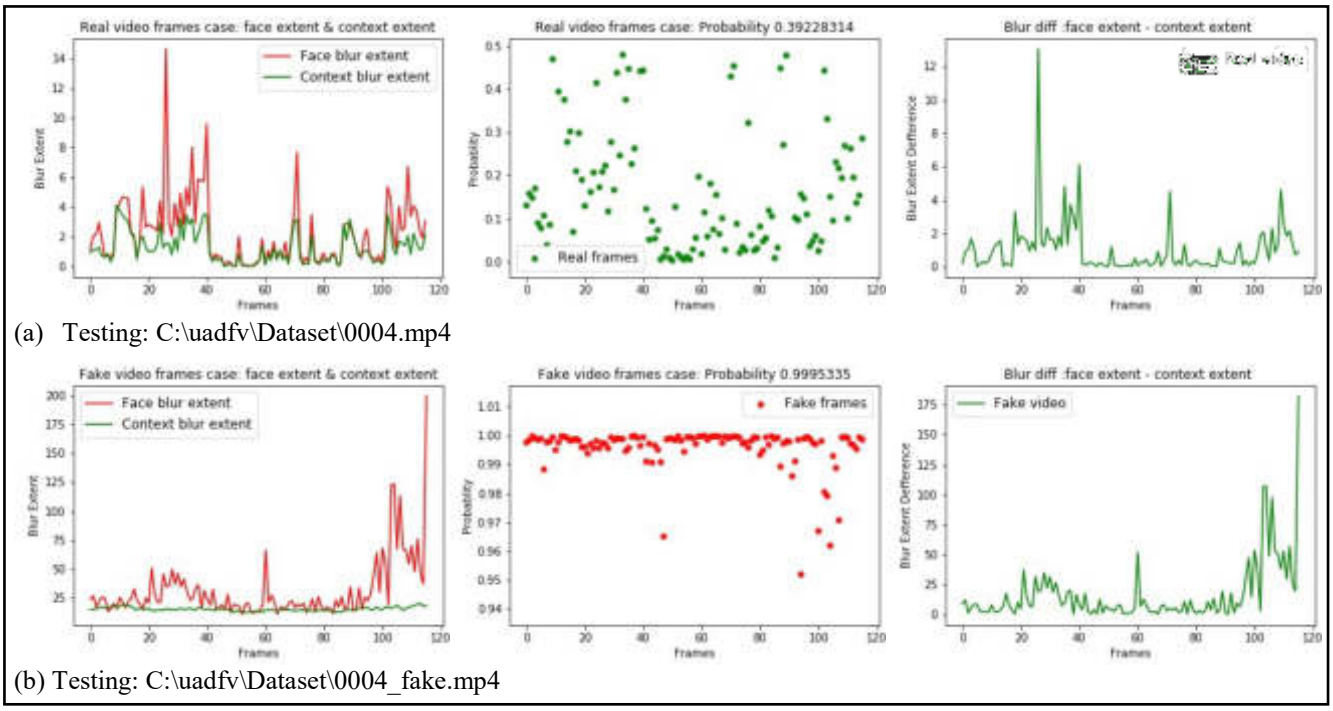
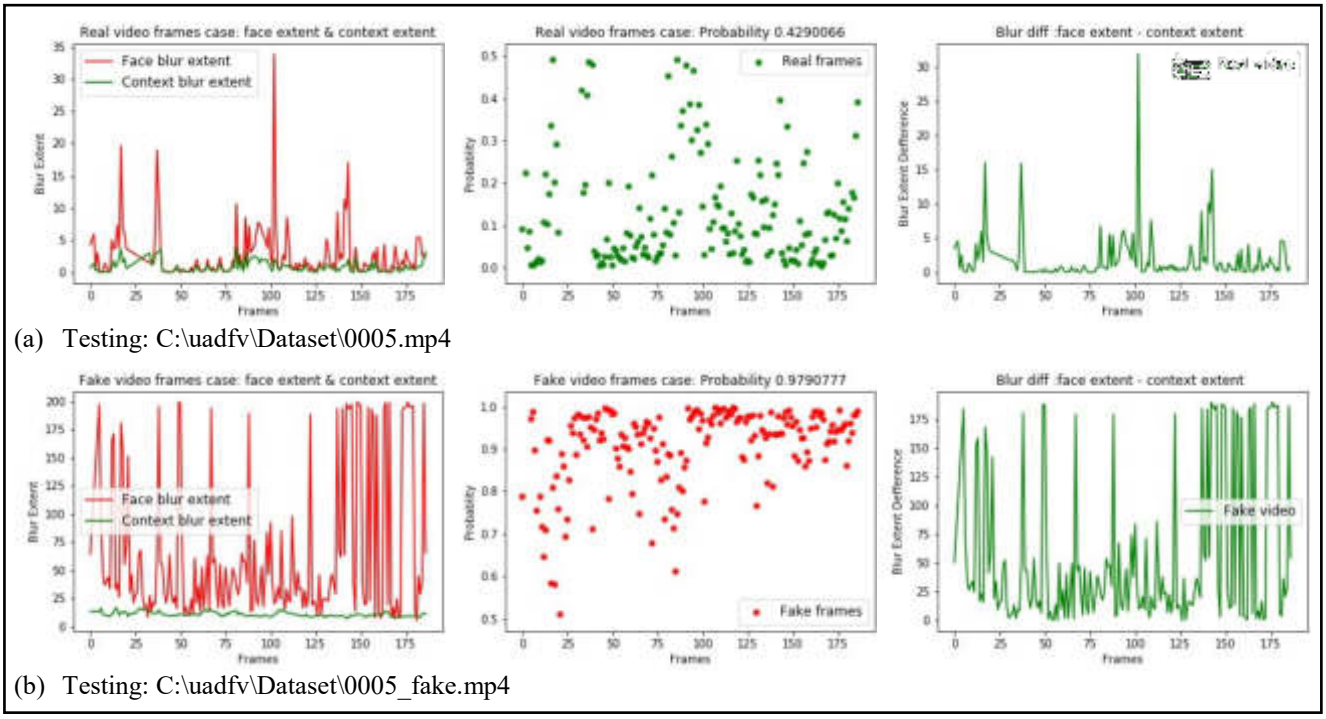


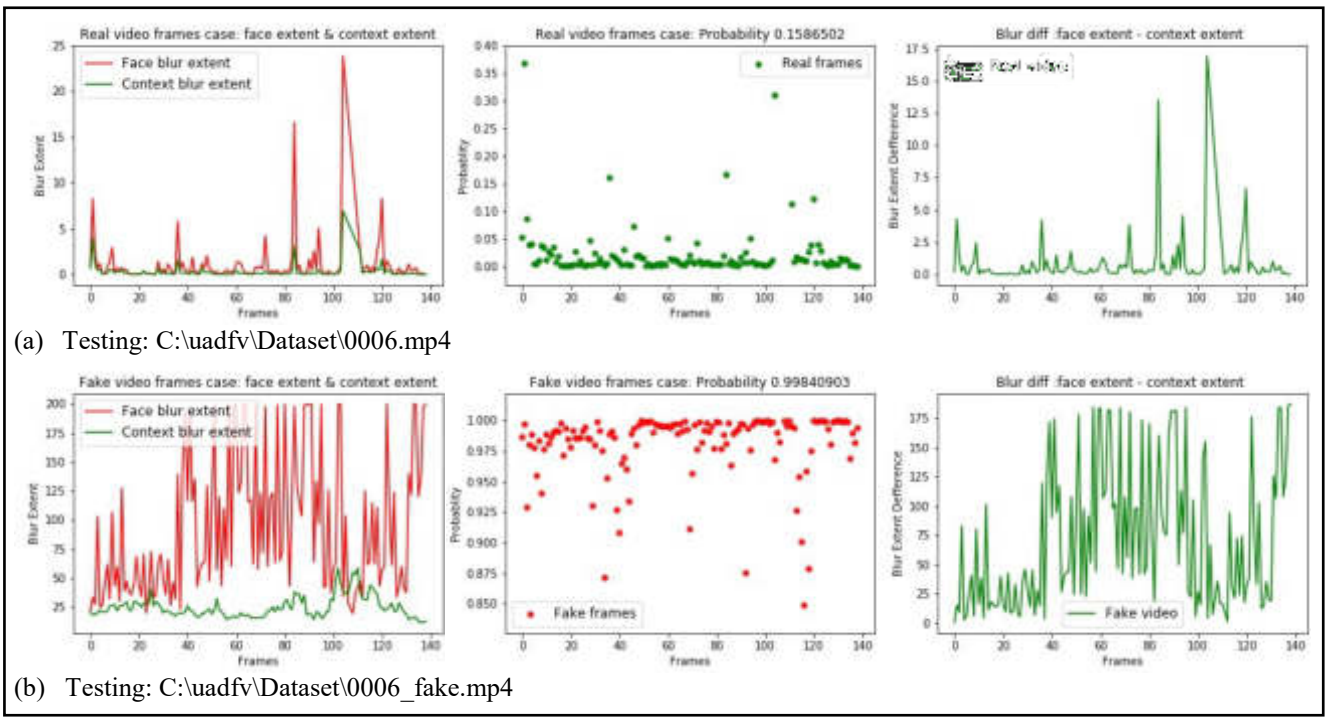
Figure (4.6): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0004.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0005.mp4

(b) Testing: C:\uadfv\Dataset\0005_fake.mp4

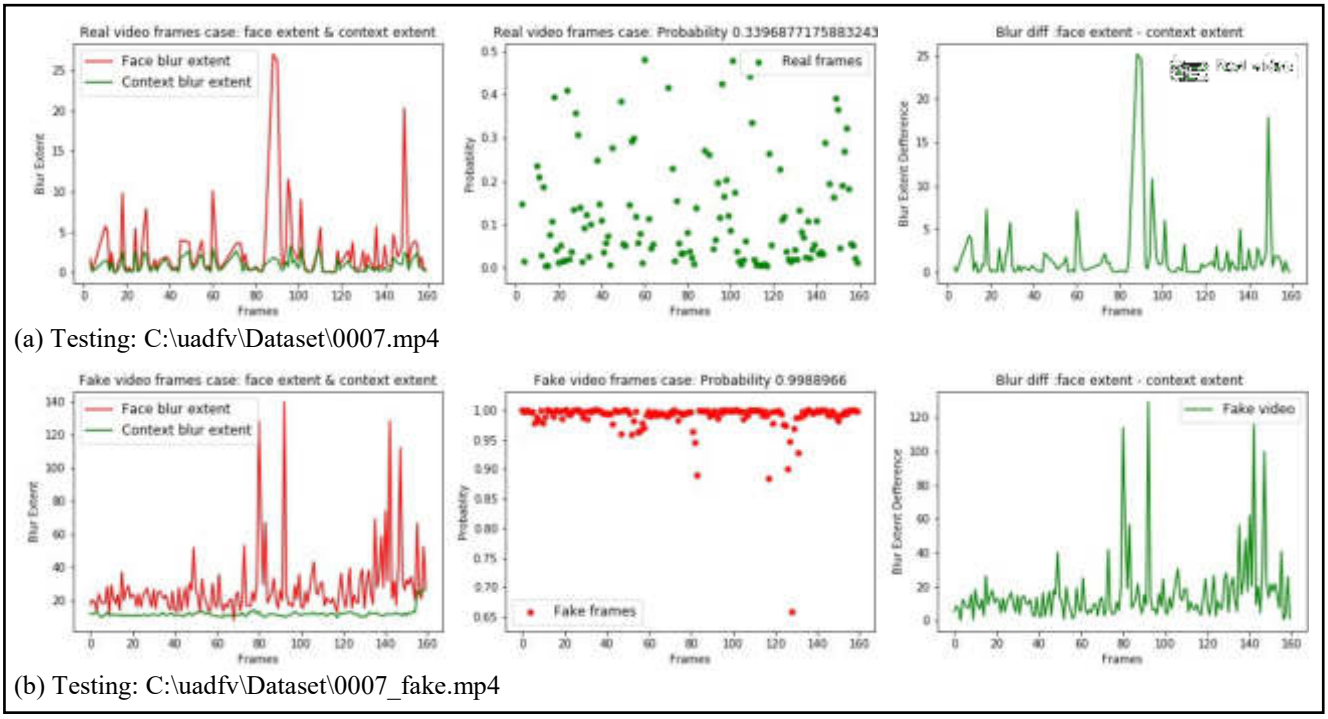
Figure (4.7): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0005.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0006.mp4

(b) Testing: C:\uadfv\Dataset\0006_fake.mp4

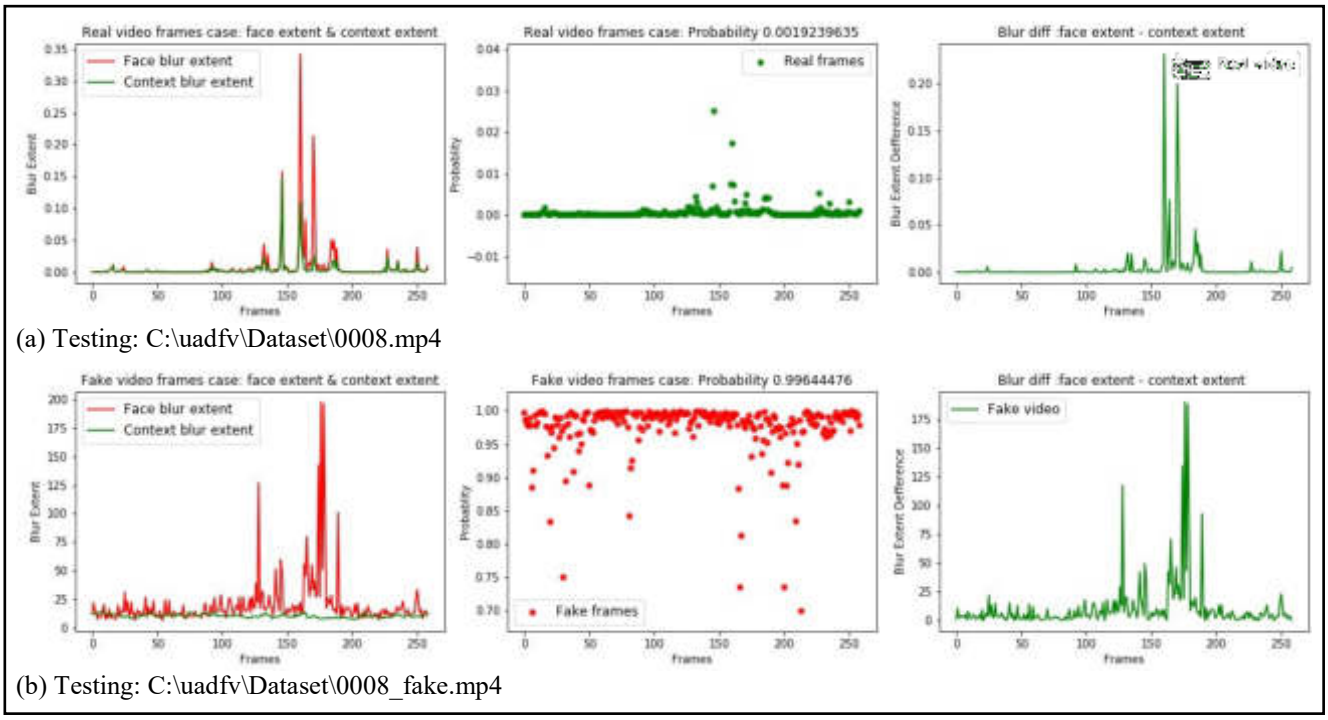
Figure (4.8): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0006.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0007.mp4

(b) Testing: C:\uadfv\Dataset\0007_fake.mp4

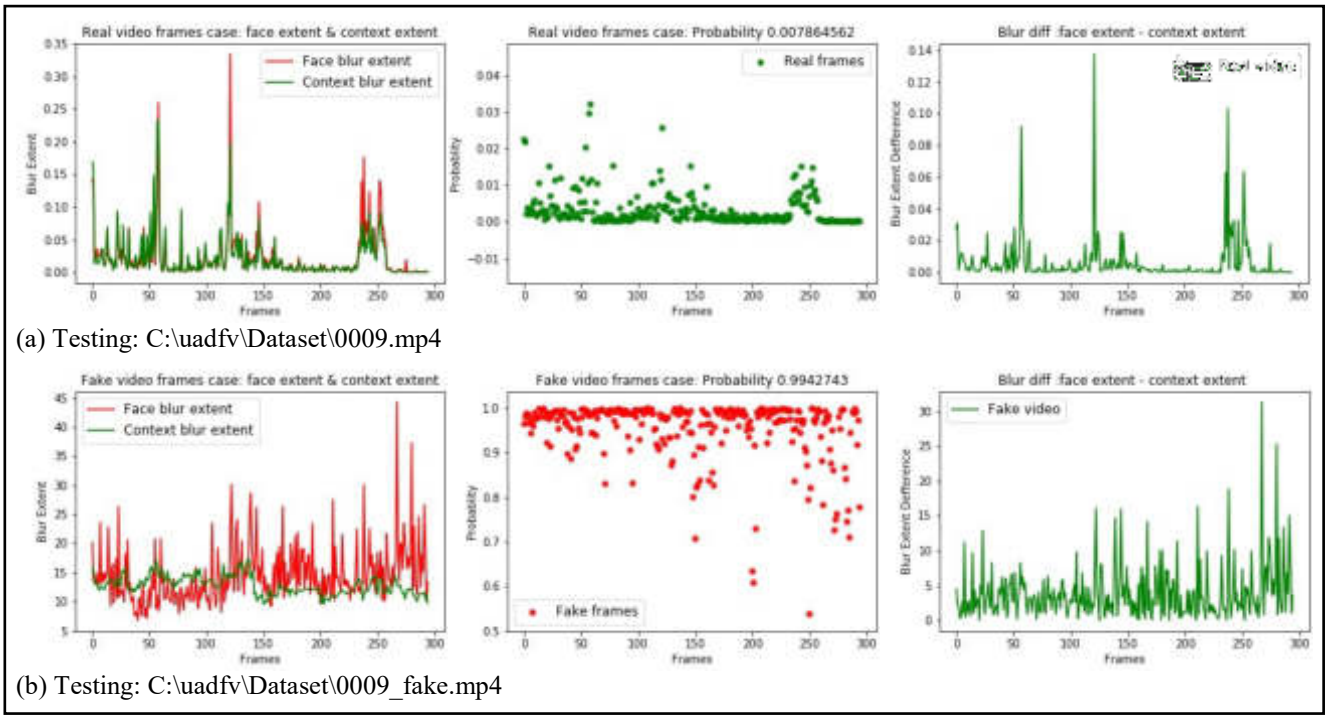
Figure (4.9): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0007.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0008.mp4

(b) Testing: C:\uadfv\Dataset\0008_fake.mp4

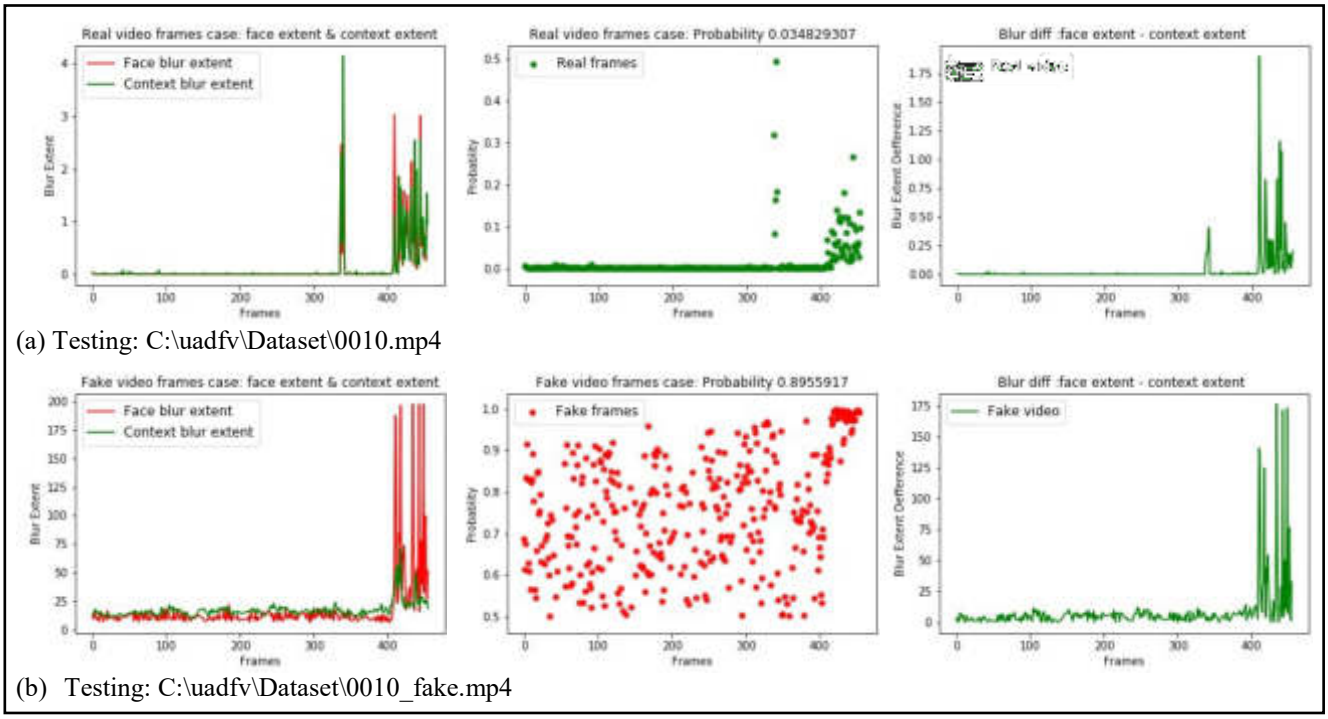
Figure (4.10): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0008.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0009.mp4

(b) Testing: C:\uadfv\Dataset\0009_fake.mp4

Figure (4.11): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0009.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0010.mp4

(b) Testing: C:\uadfv\Dataset\0010_fake.mp4

Figure (4.12): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0010.mp4". (a) Real video-version, and (b) Fake video-version.

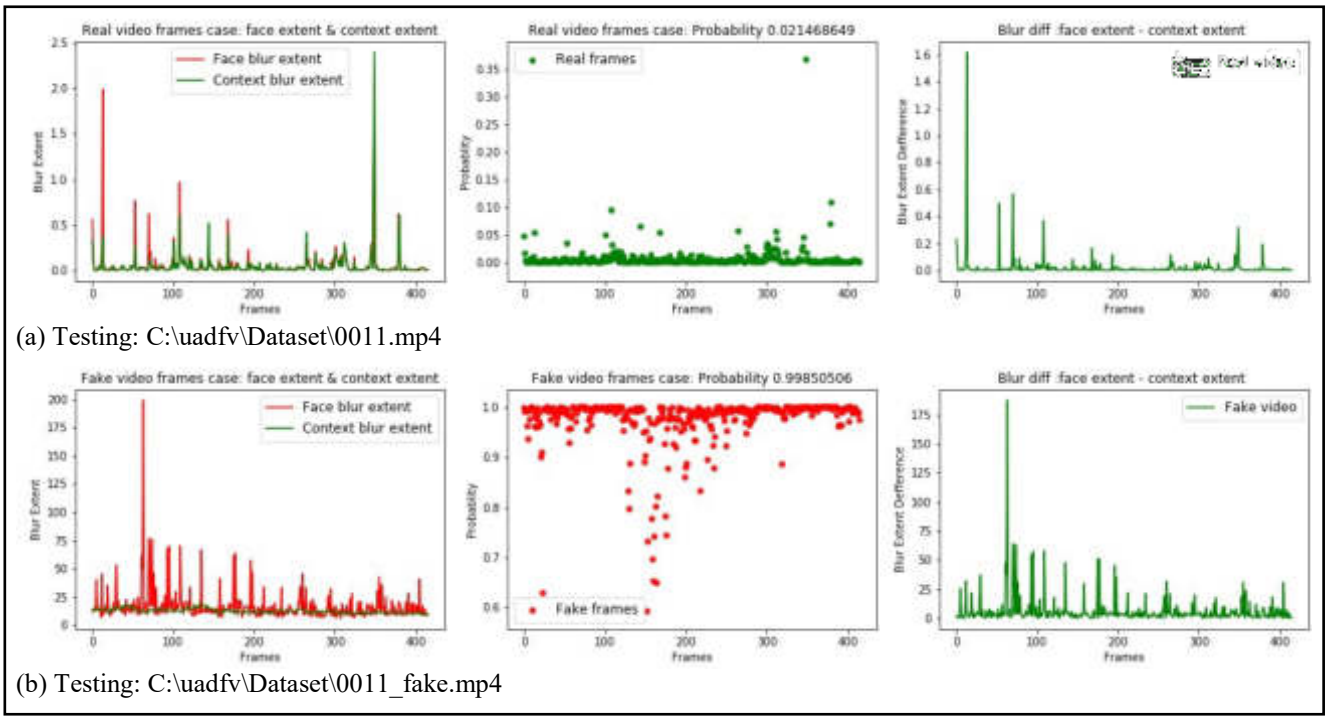


Figure (4.13): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0011.mp4". (a) Real video-version, and (b) Fake video-version.

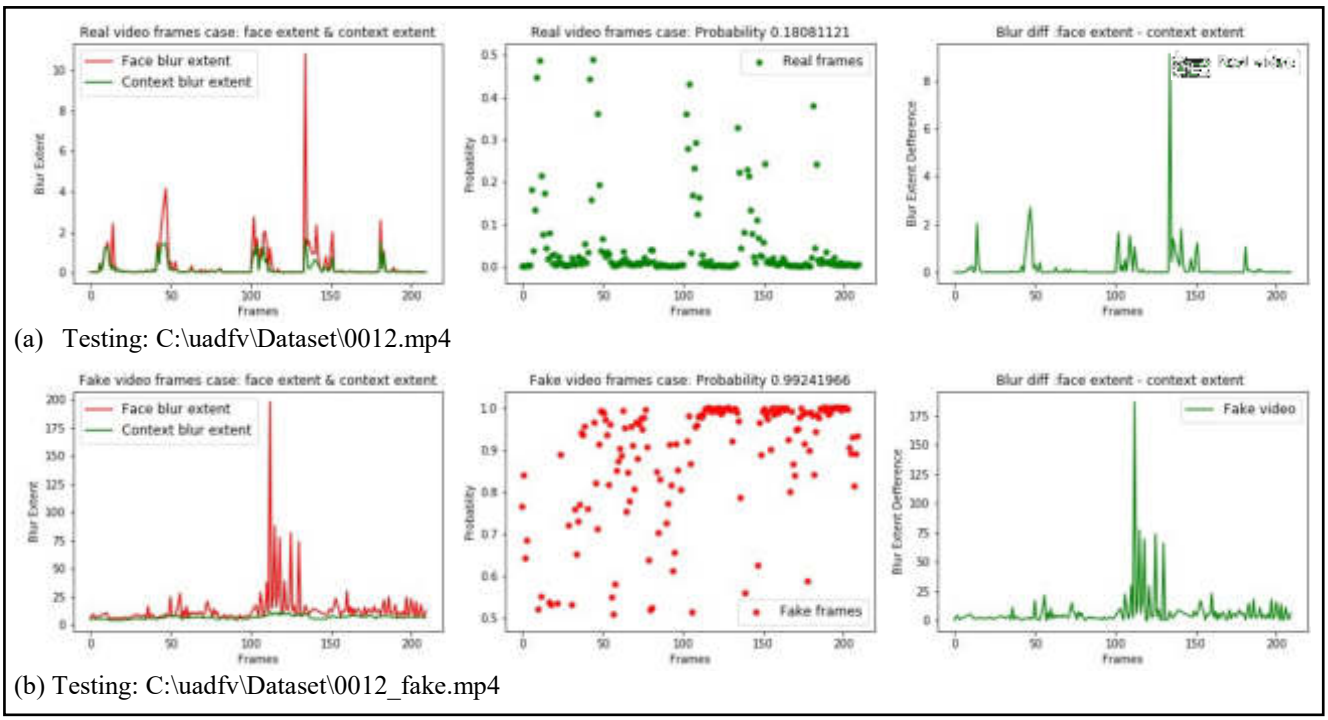


Figure (4.14): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0012.mp4". (a) Real video-version, and (b) Fake video-version.

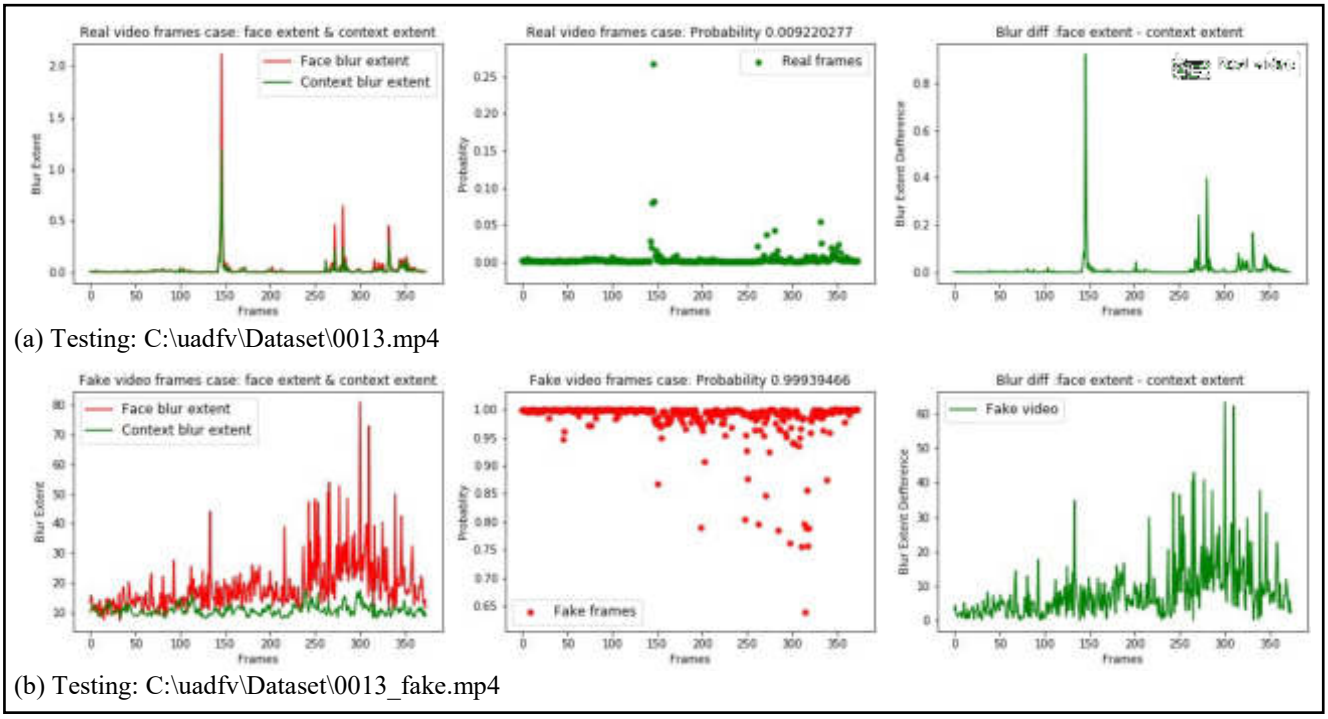


Figure (4.15): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0013.mp4". (a) Real video-version, and (b) Fake video-version.

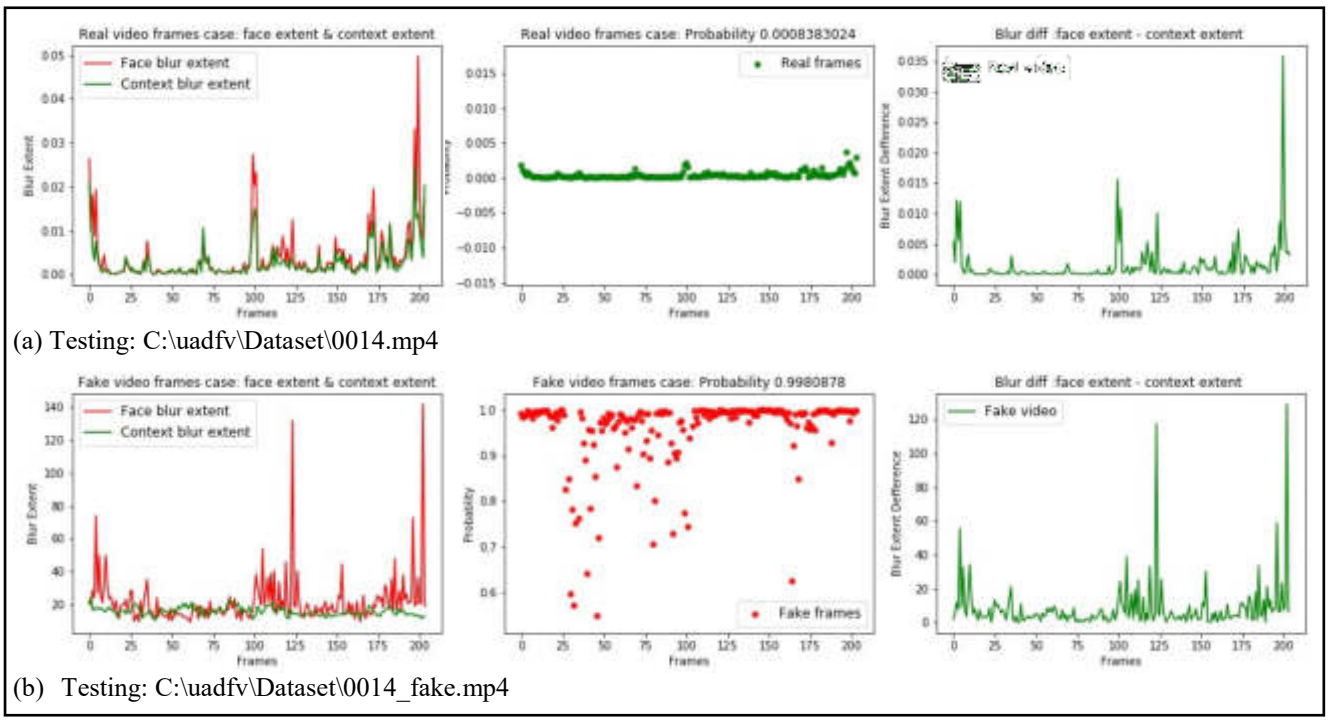
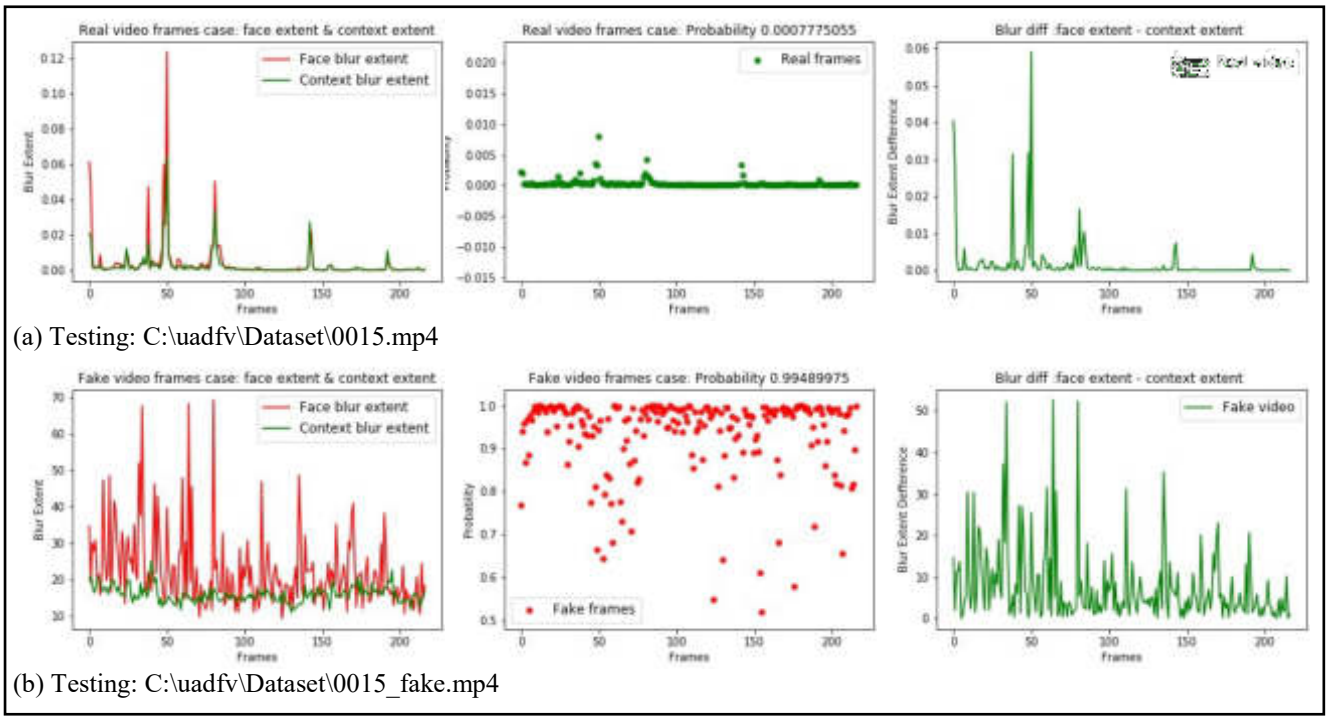


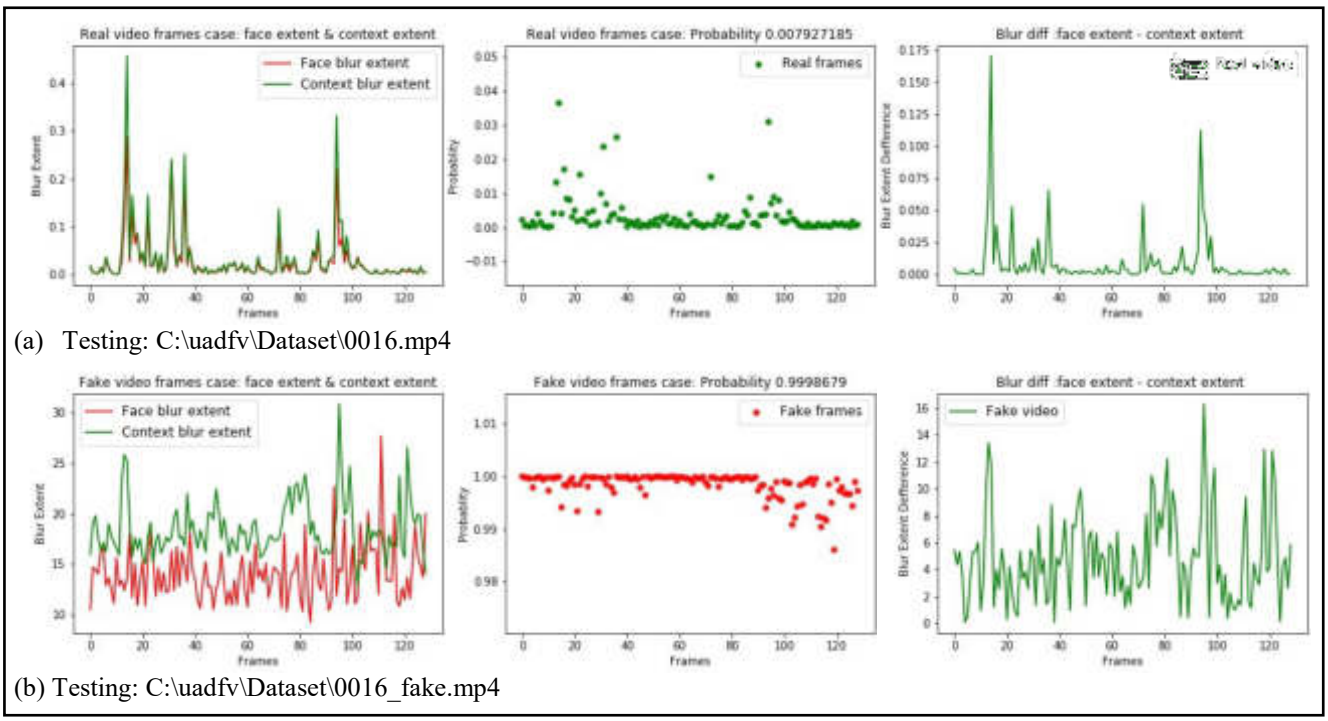
Figure (4.16): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0014.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0015.mp4

(b) Testing: C:\uadfv\Dataset\0015_fake.mp4

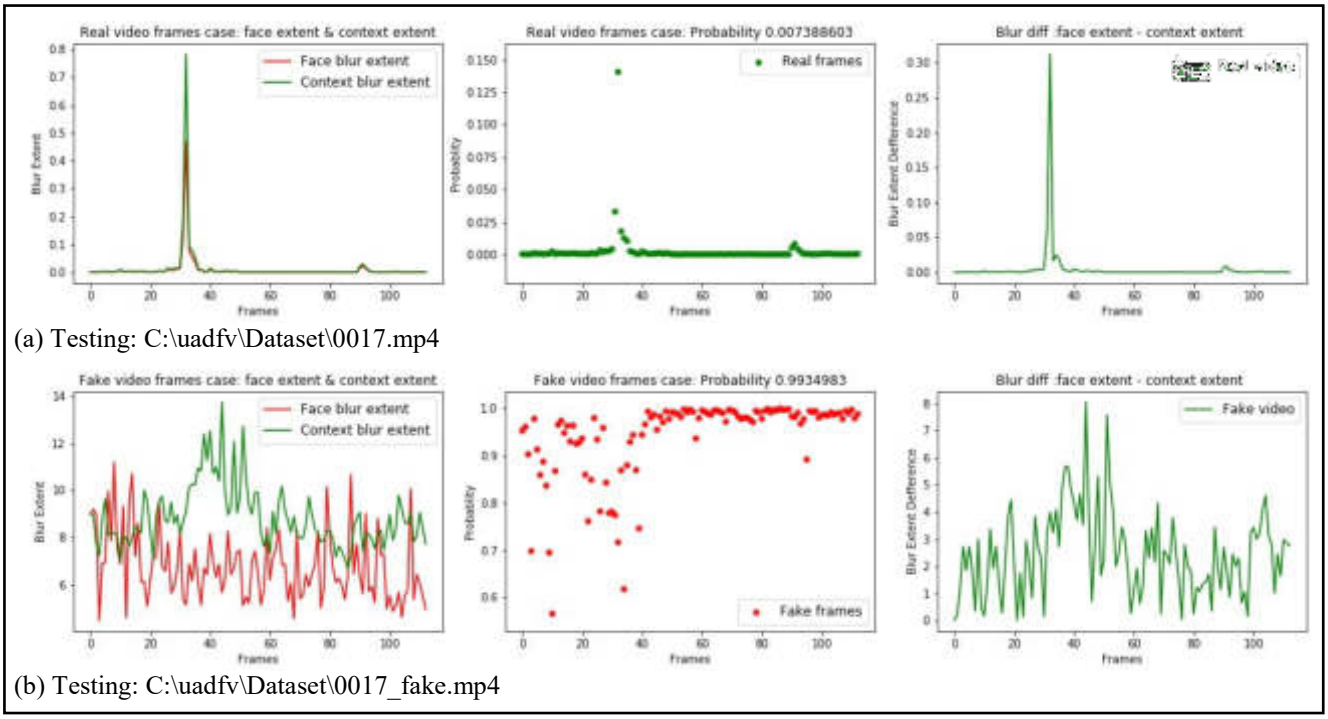
Figure (4.17): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0015.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0016.mp4

(b) Testing: C:\uadfv\Dataset\0016_fake.mp4

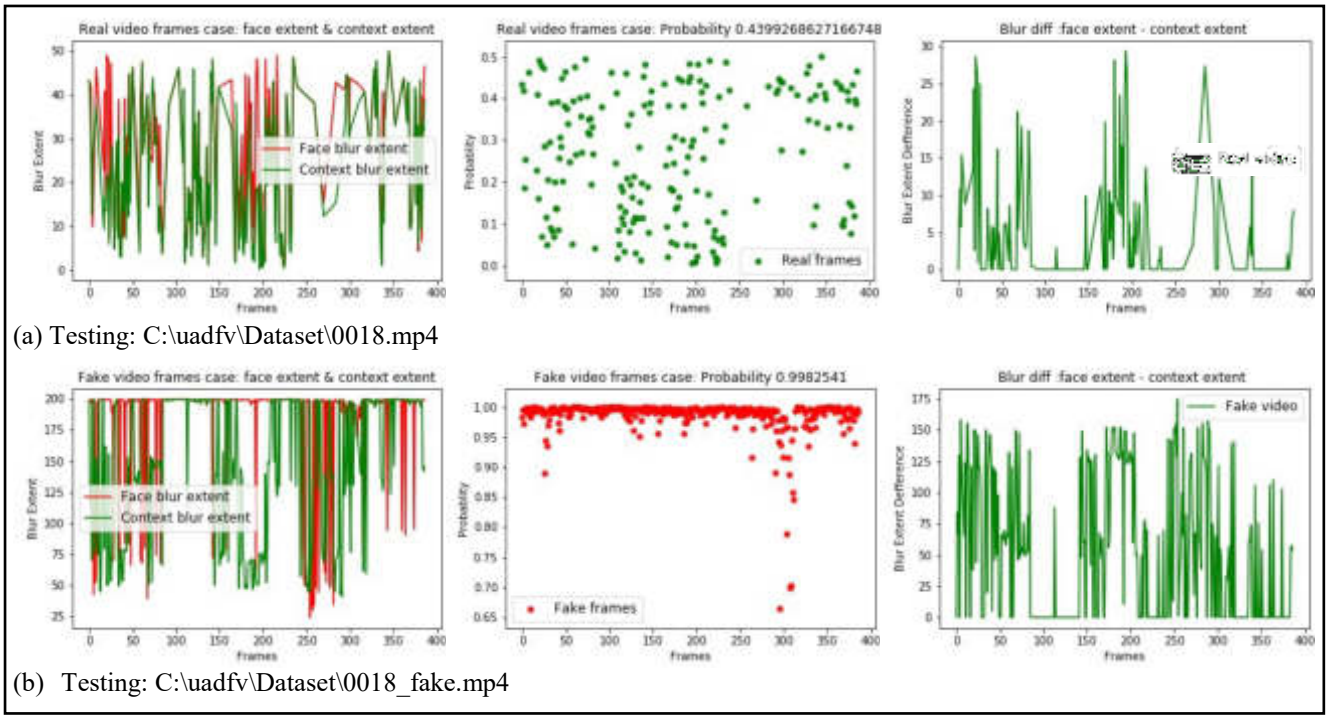
Figure (4.18): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0016.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0017.mp4

(b) Testing: C:\uadfv\Dataset\0017_fake.mp4

Figure (4.19): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0017.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0018.mp4

(b) Testing: C:\uadfv\Dataset\0018_fake.mp4

Figure (4.20): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0018.mp4". (a) Real video-version, and (b) Fake video-version.

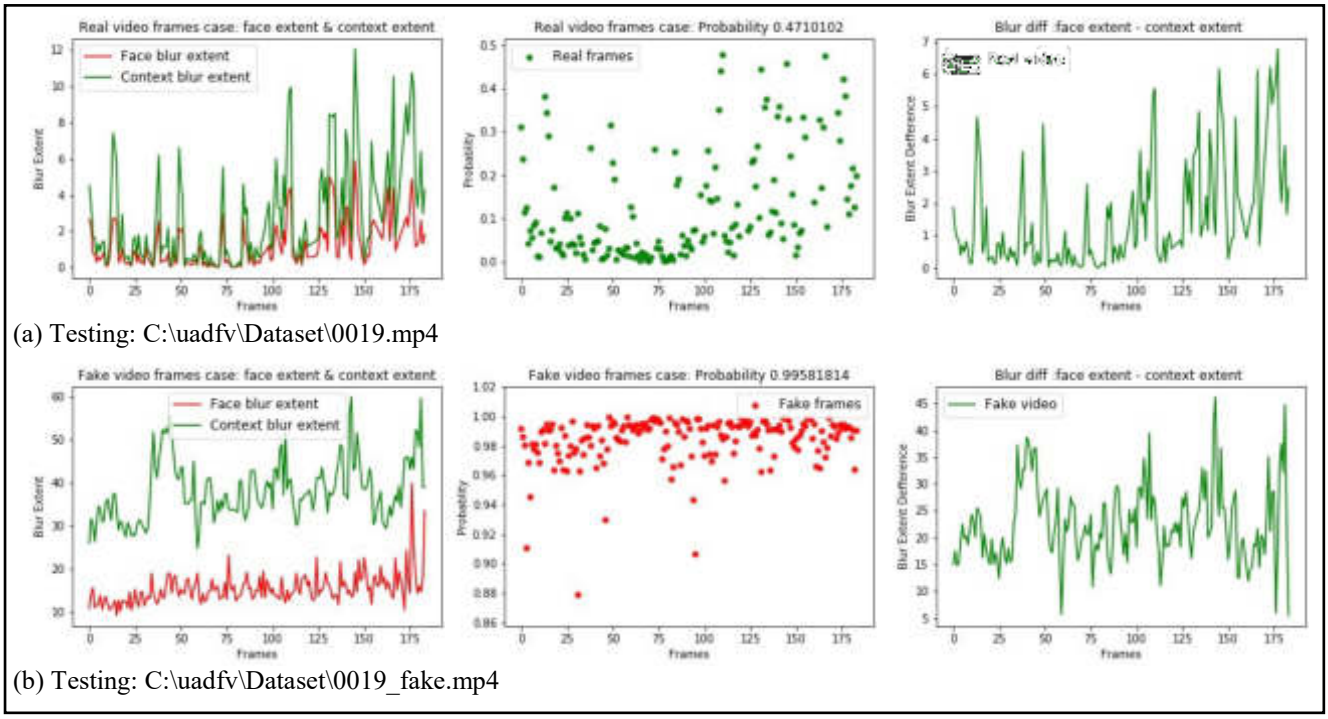


Figure (4.21): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0019.mp4". (a) Real video-version, and (b) Fake video-version.

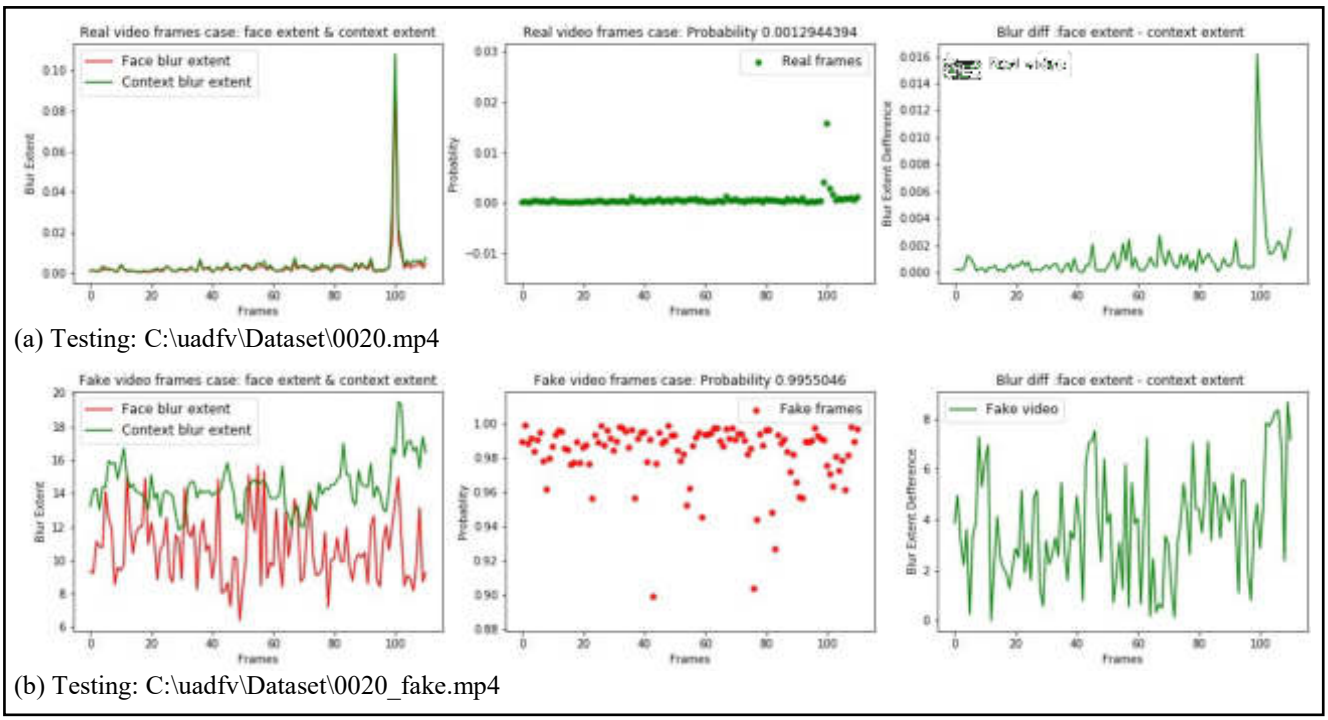


Figure (4.22): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0020.mp4". (a) Real video-version, and (b) Fake video-version.

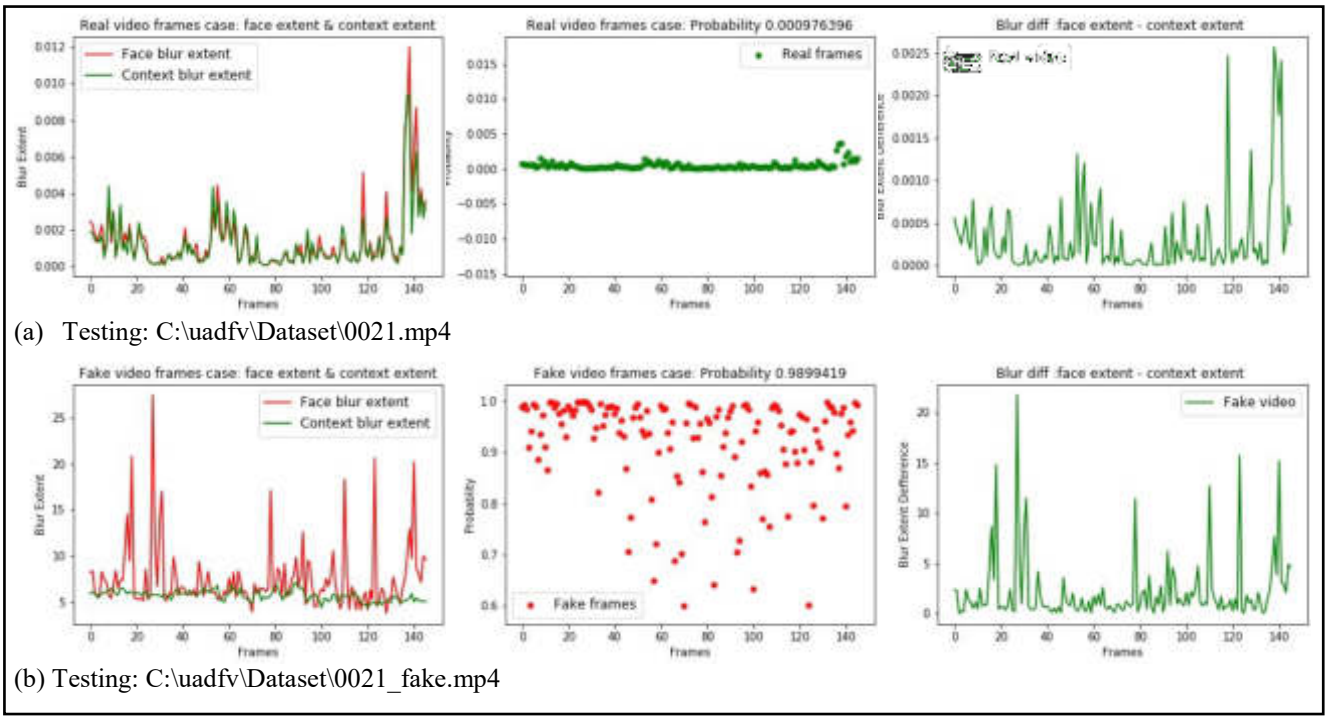


Figure (4.23): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0021.mp4". (a) Real video-version, and (b) Fake video-version.

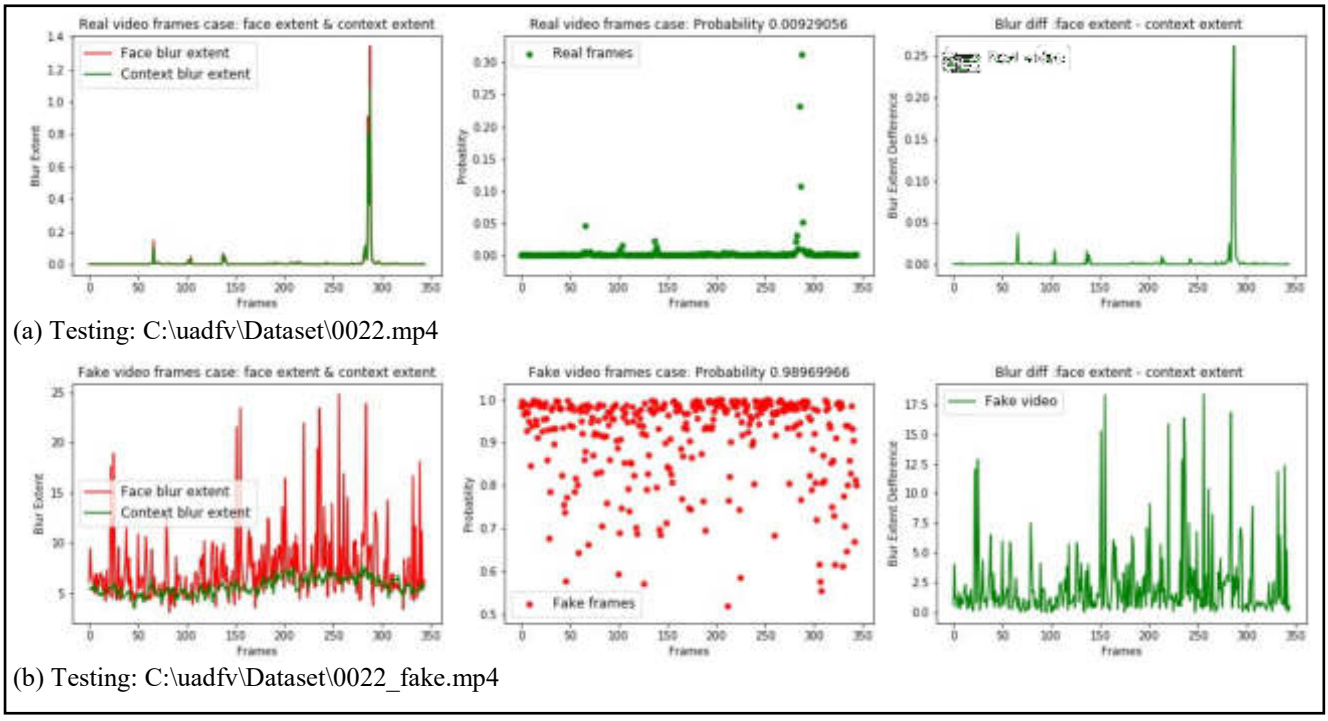


Figure (4.24): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0022.mp4". (a) Real video-version, and (b) Fake video-version.

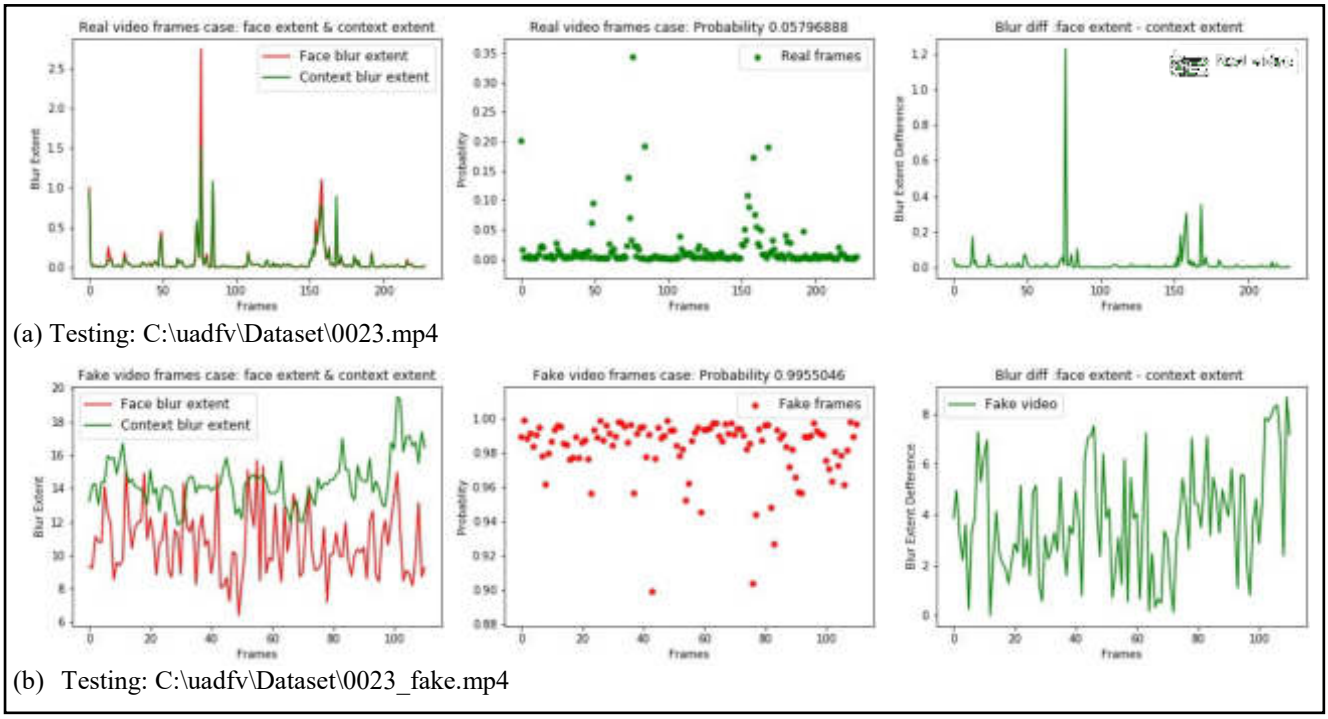


Figure (4.25): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0023.mp4". (a) Real video-version, and (b) Fake video-version.

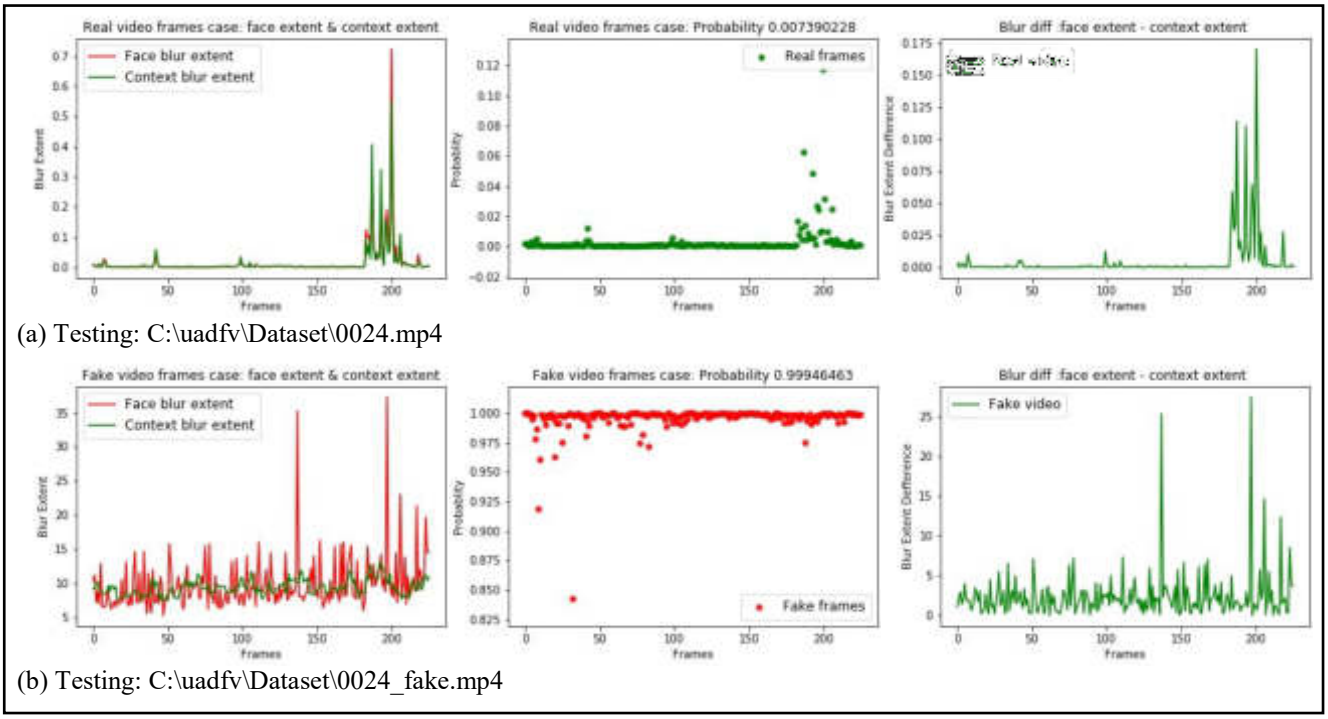
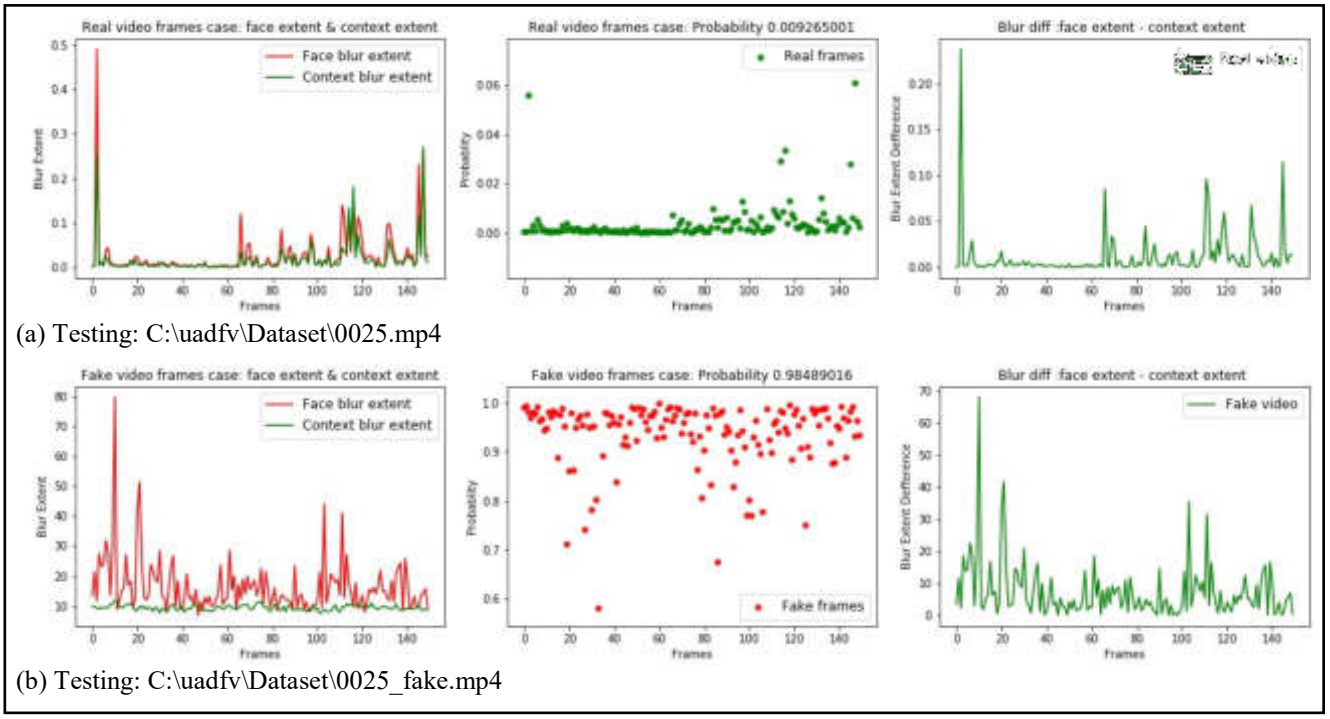


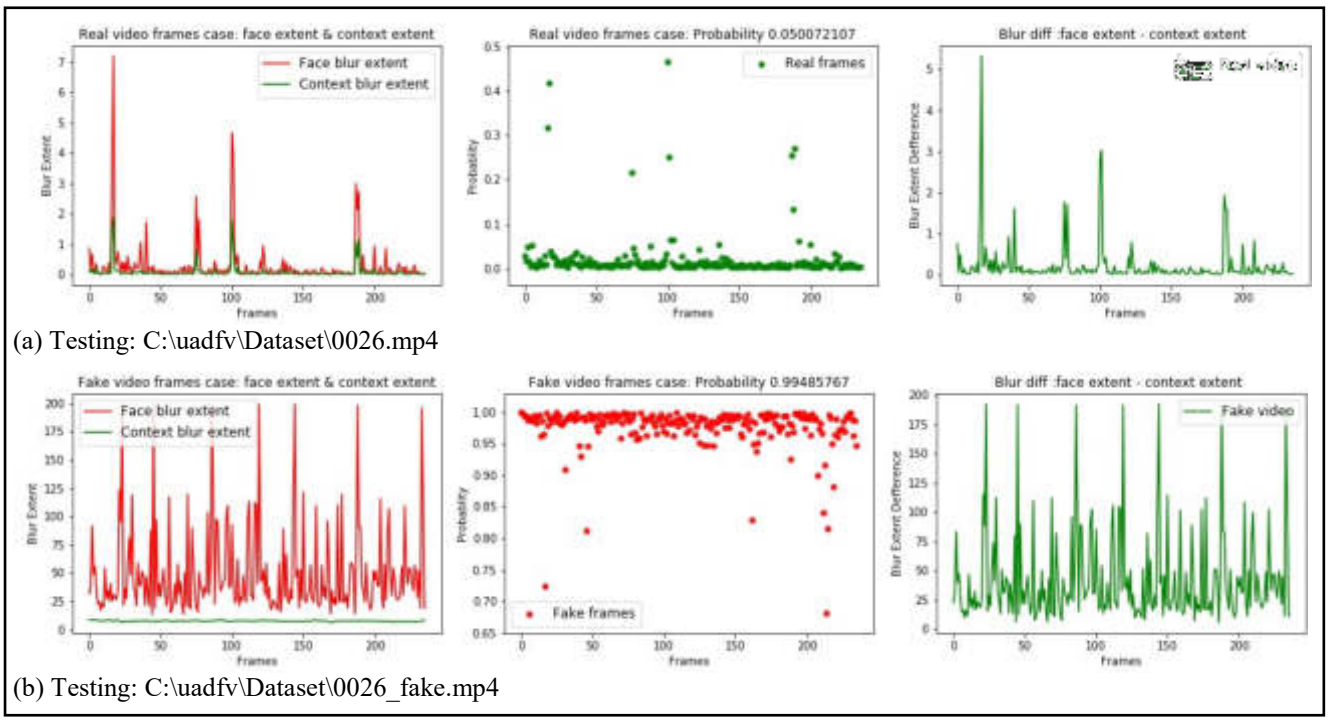
Figure (4.26): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0024.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0025.mp4

(b) Testing: C:\uadfv\Dataset\0025_fake.mp4

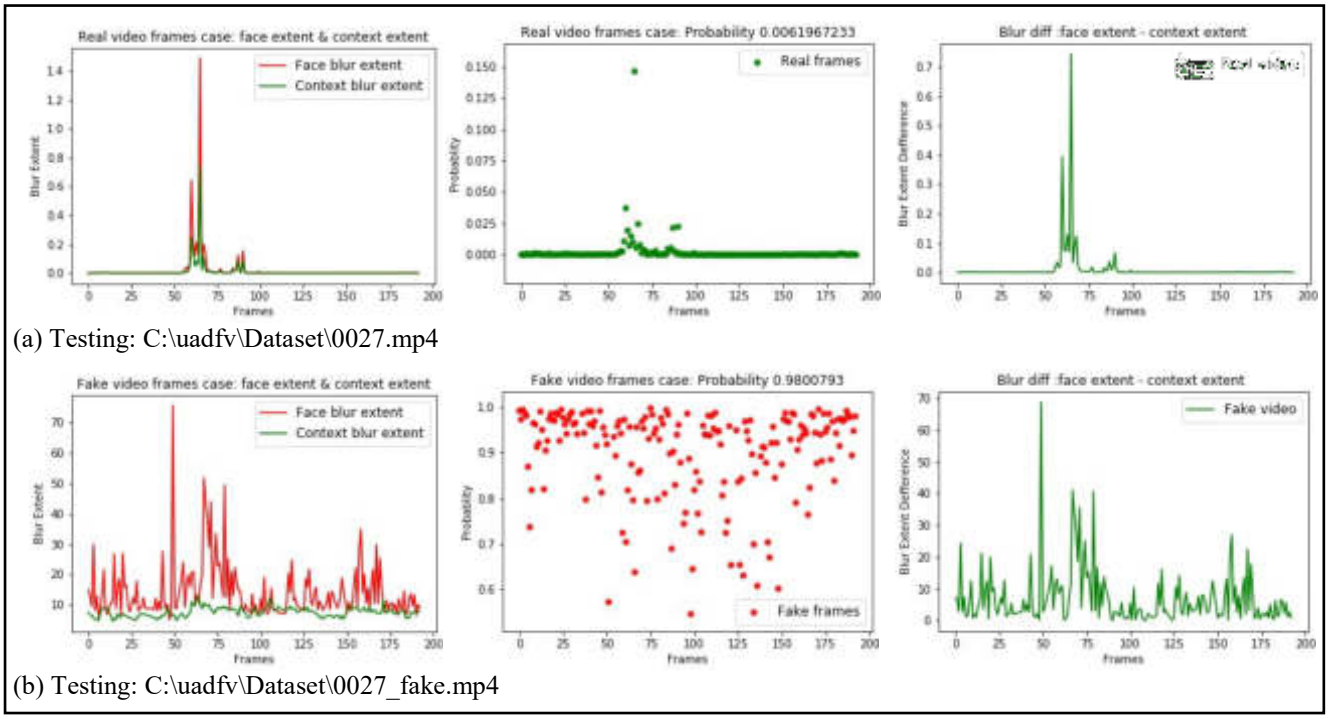
Figure (4.27): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0025.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0026.mp4

(b) Testing: C:\uadfv\Dataset\0026_fake.mp4

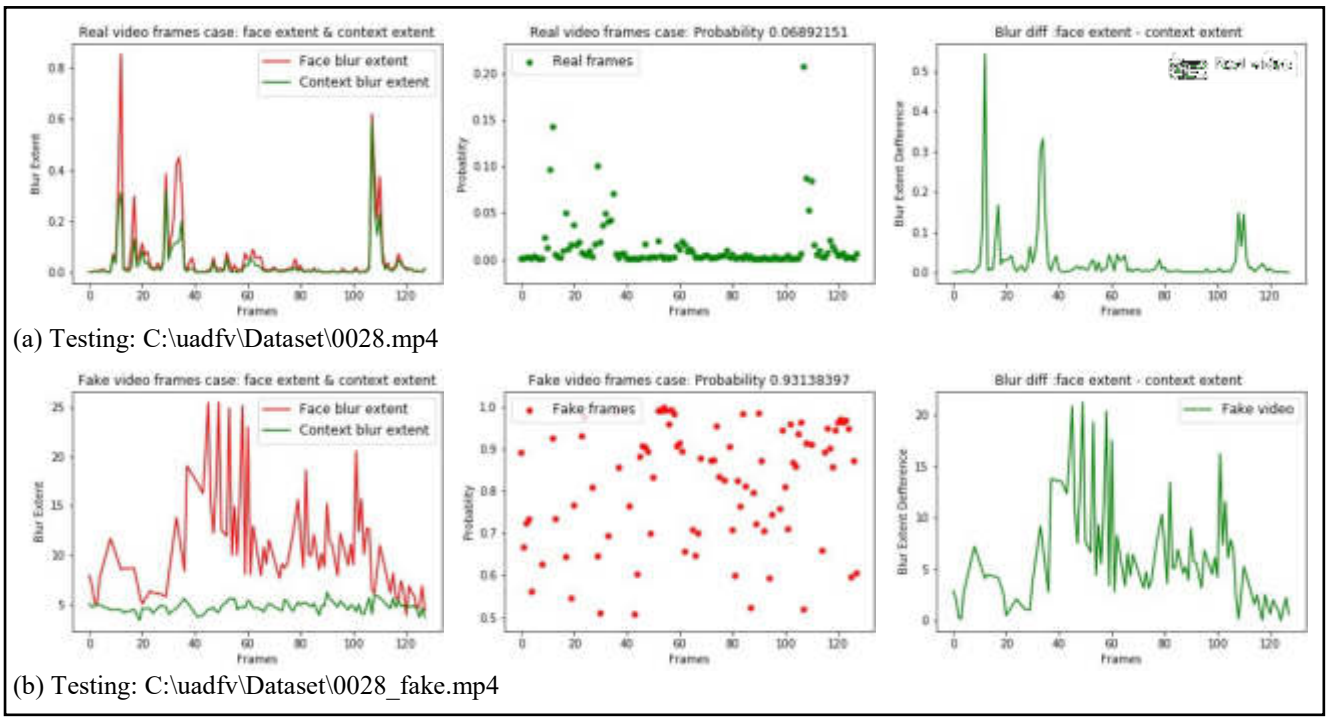
Figure (4.28): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0026.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0027.mp4

(b) Testing: C:\uadfv\Dataset\0027_fake.mp4

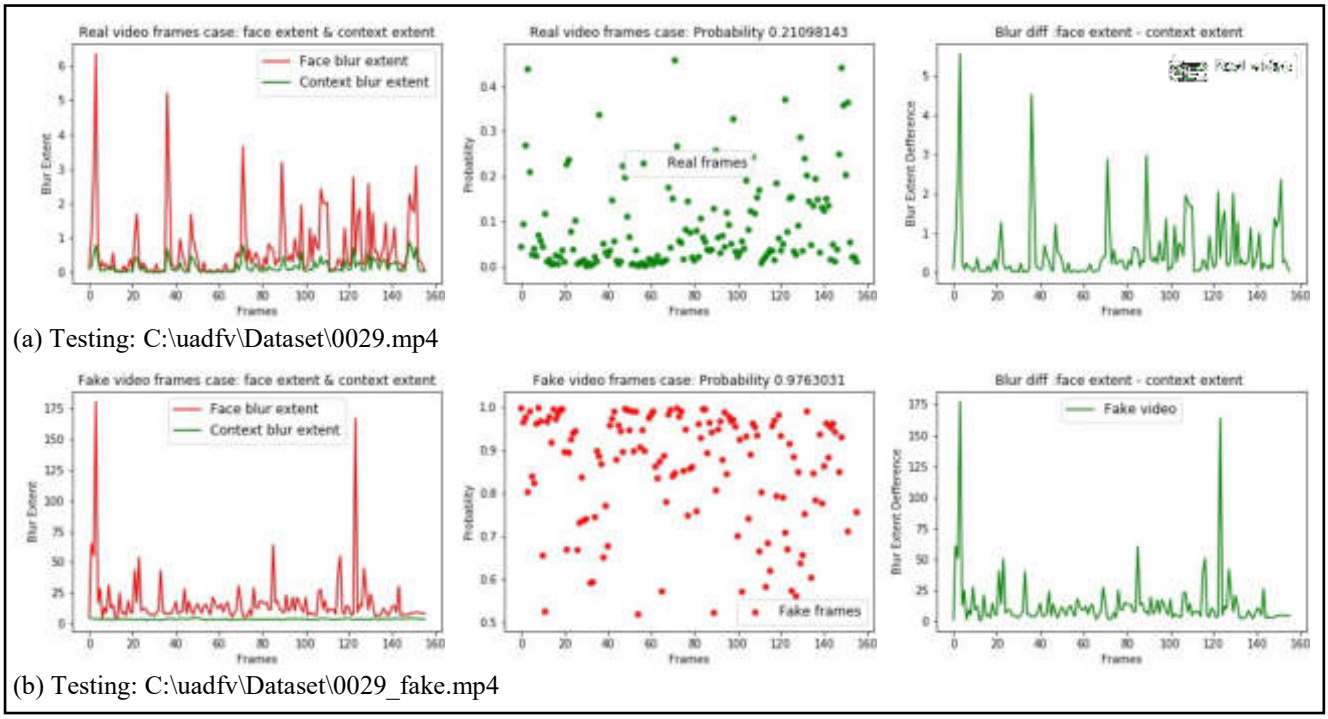
Figure (4.29): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0027.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0028.mp4

(b) Testing: C:\uadfv\Dataset\0028_fake.mp4

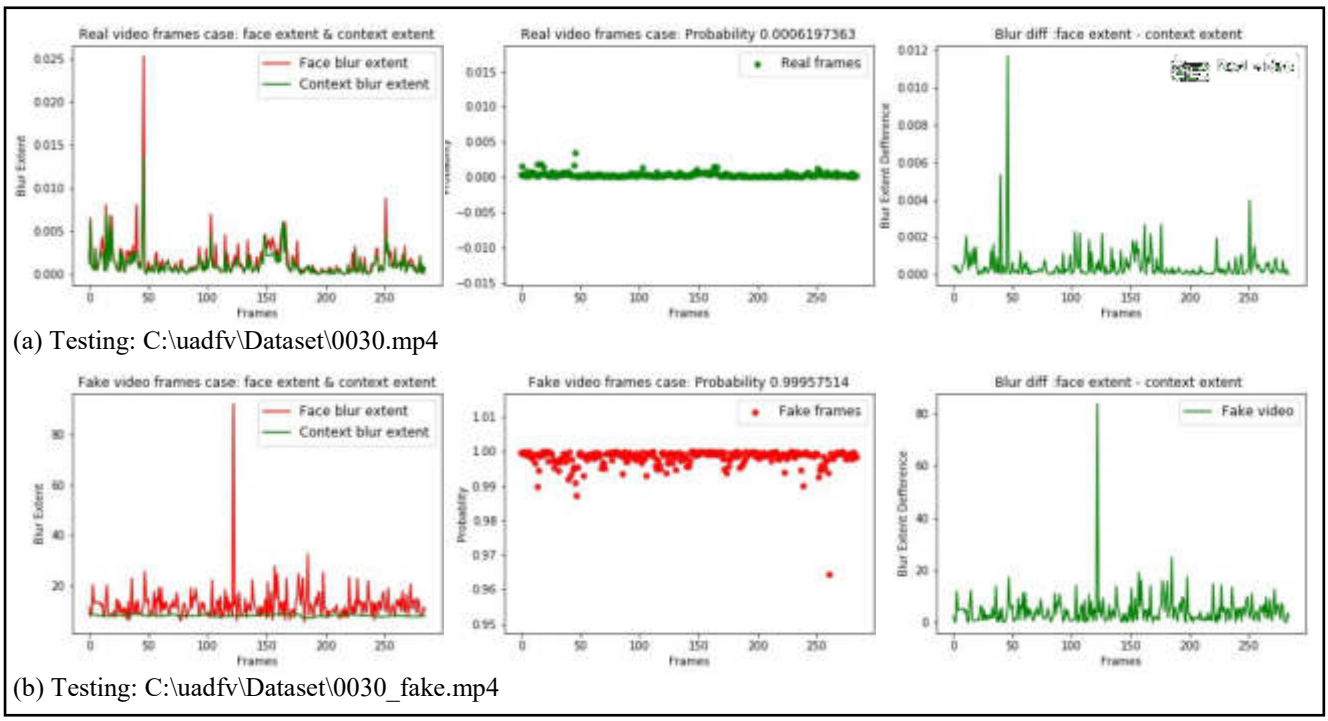
Figure (4.30): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0028.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0029.mp4

(b) Testing: C:\uadfv\Dataset\0029_fake.mp4

Figure (4.31): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0029.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0030.mp4

(b) Testing: C:\uadfv\Dataset\0030_fake.mp4

Figure (4.32): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0030.mp4". (a) Real video-version, and (b) Fake video-version.

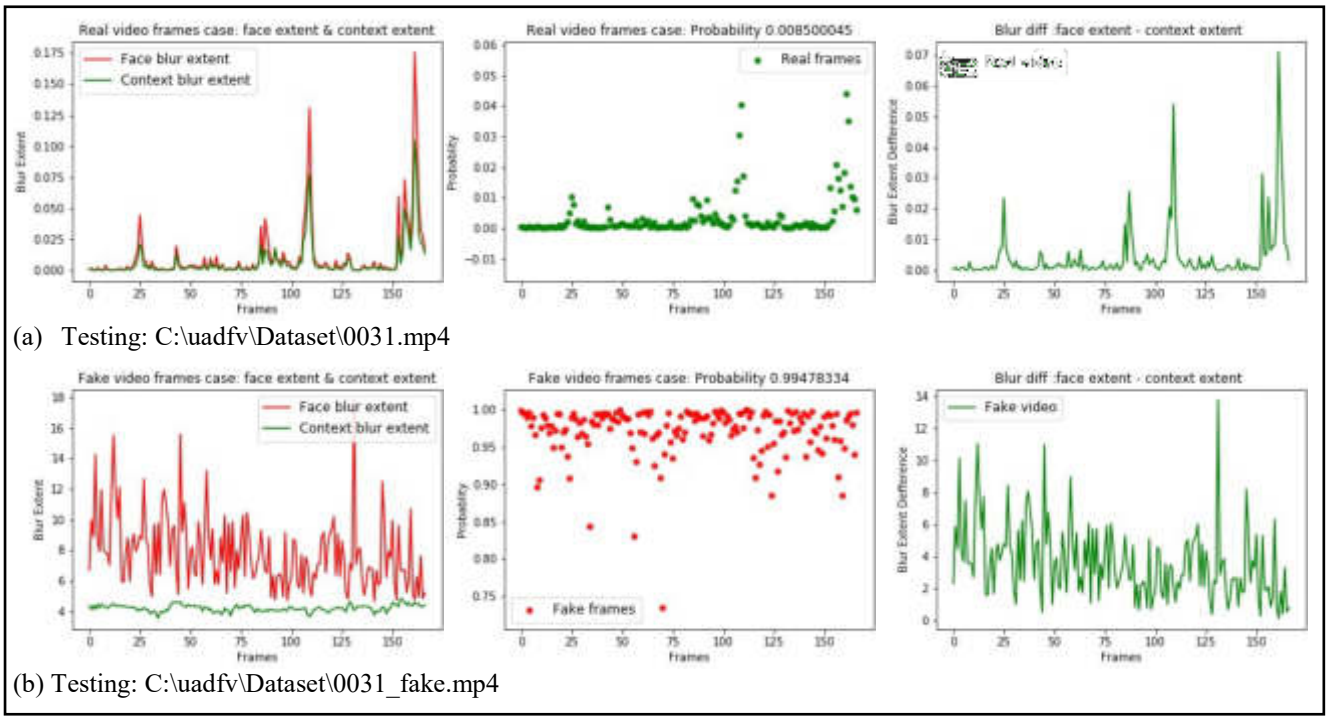


Figure (4.33): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0031.mp4". (a) Real video-version, and (b) Fake video-version.

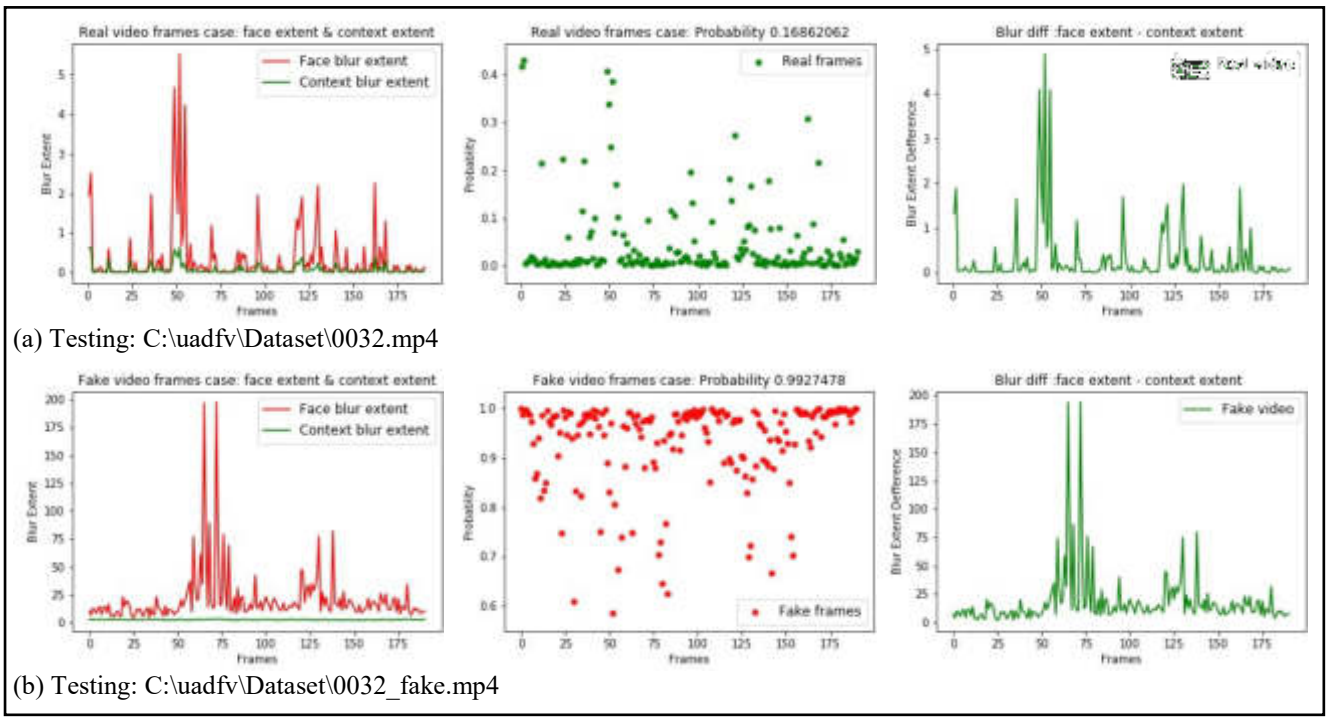


Figure (4.34): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0032.mp4". (a) Real video-version, and (b) Fake video-version.

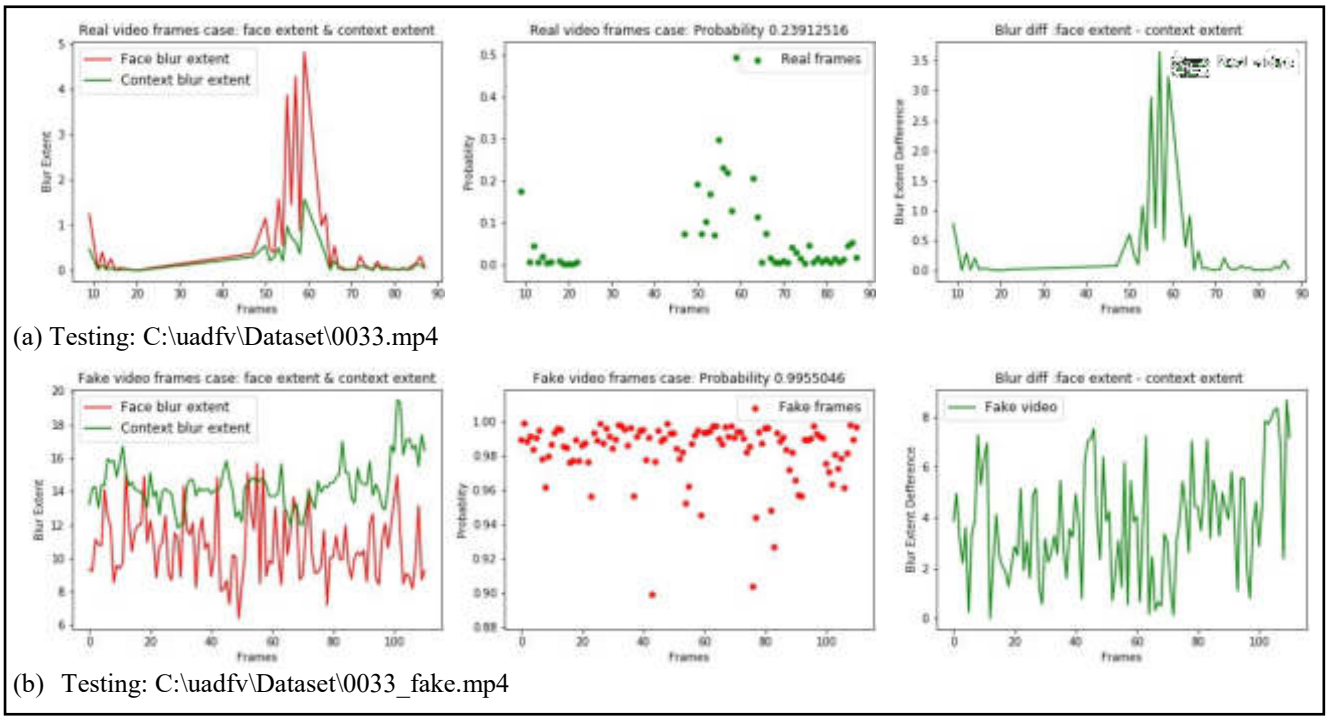


Figure (4.35): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0033.mp4". (a) Real video-version, and (b) Fake video-version.

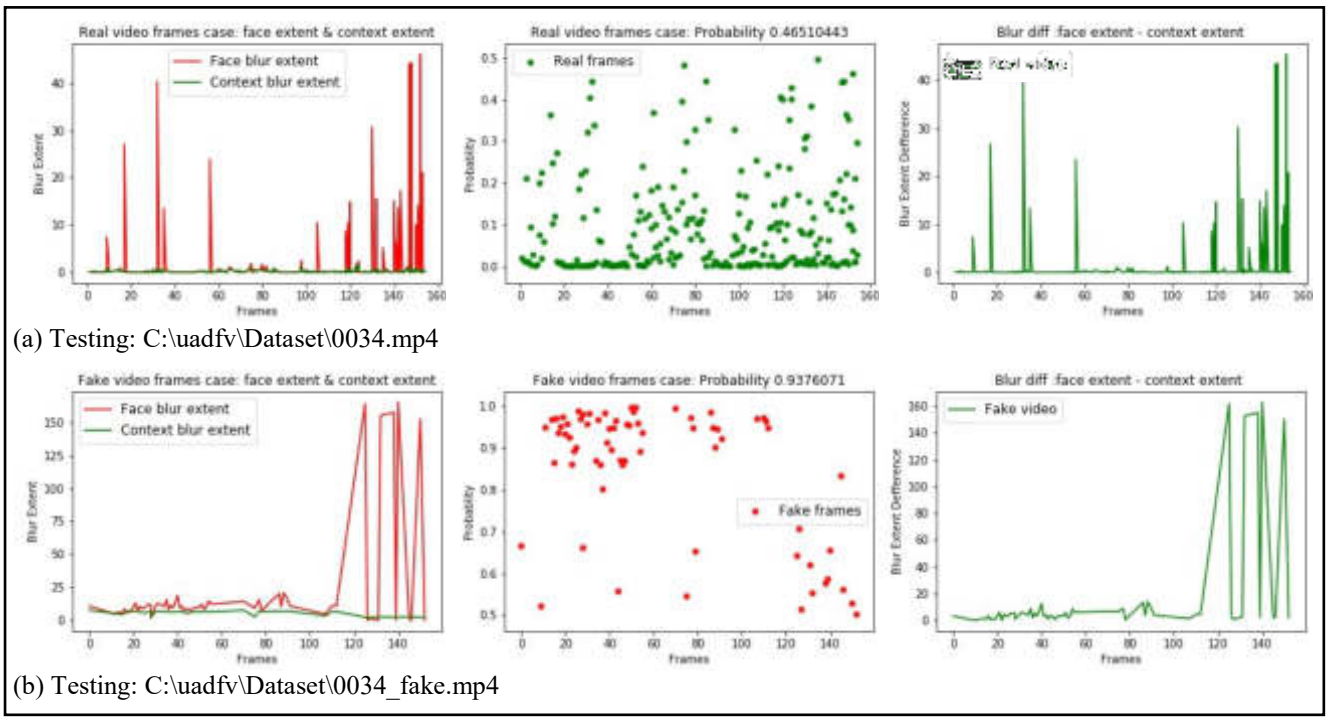


Figure (4.36): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0034.mp4". (a) Real video-version, and (b) Fake video-version.

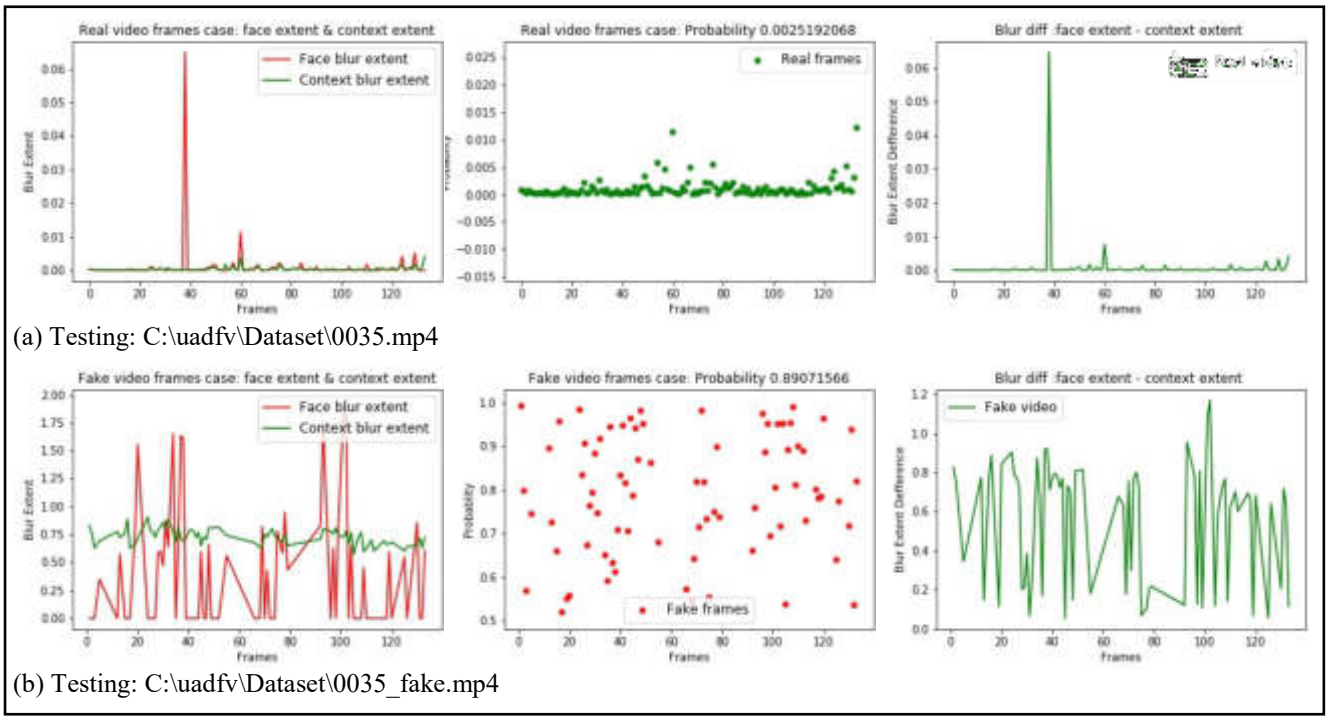


Figure (4.37): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0035.mp4". (a) Real video-version, and (b) Fake video-version.

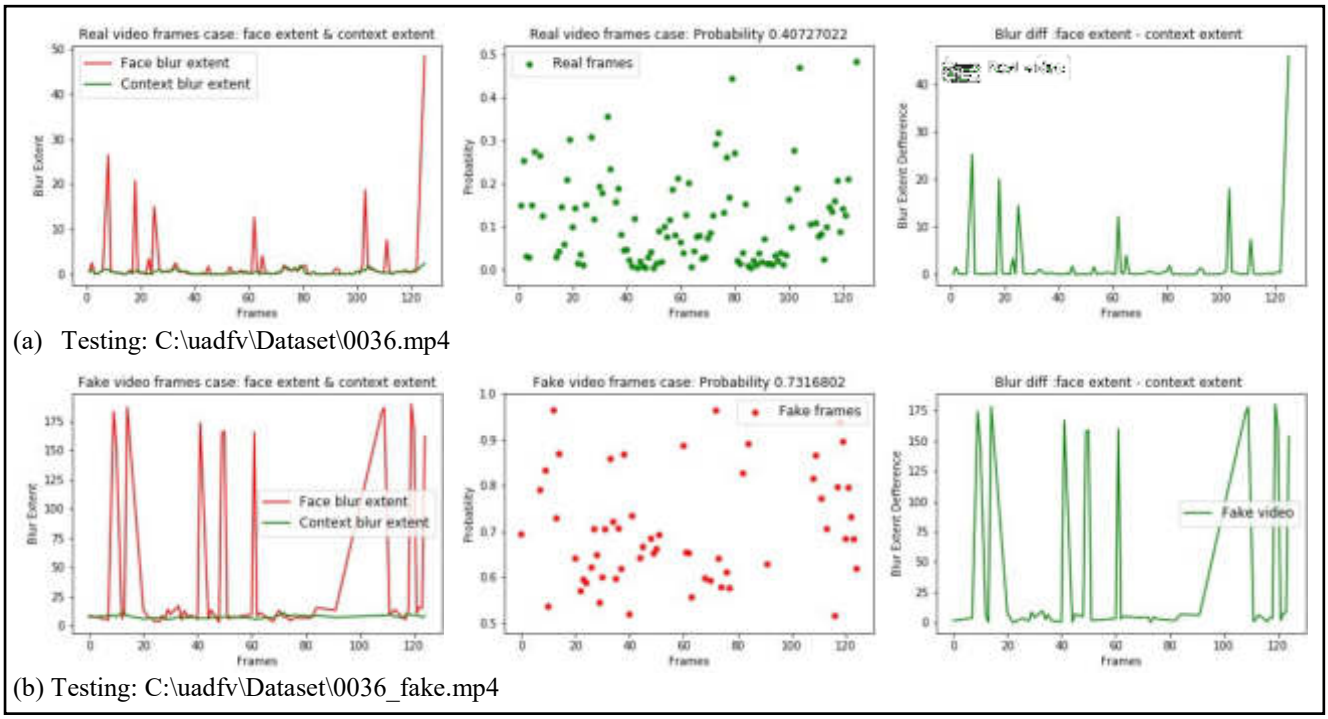


Figure (4.38): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0036.mp4". (a) Real video-version, and (b) Fake video-version.

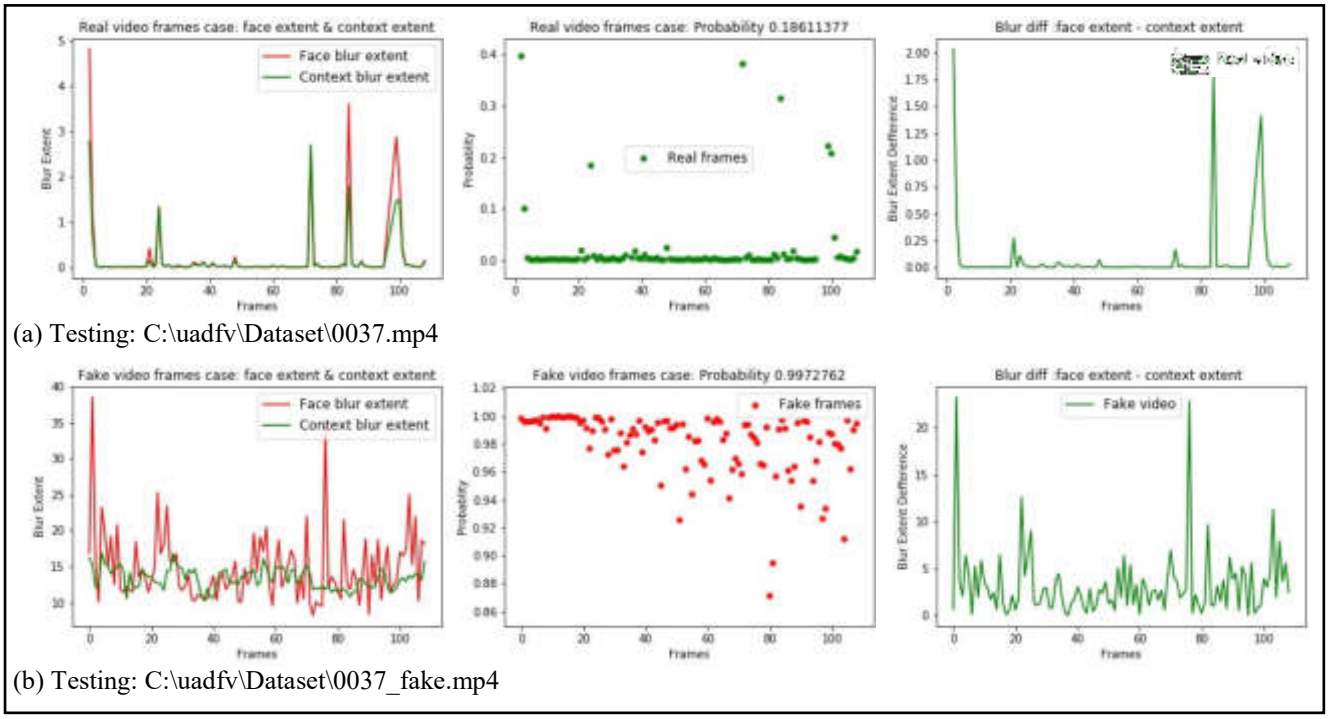


Figure (4.39): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0037.mp4". (a) Real video-version, and (b) Fake video-version.

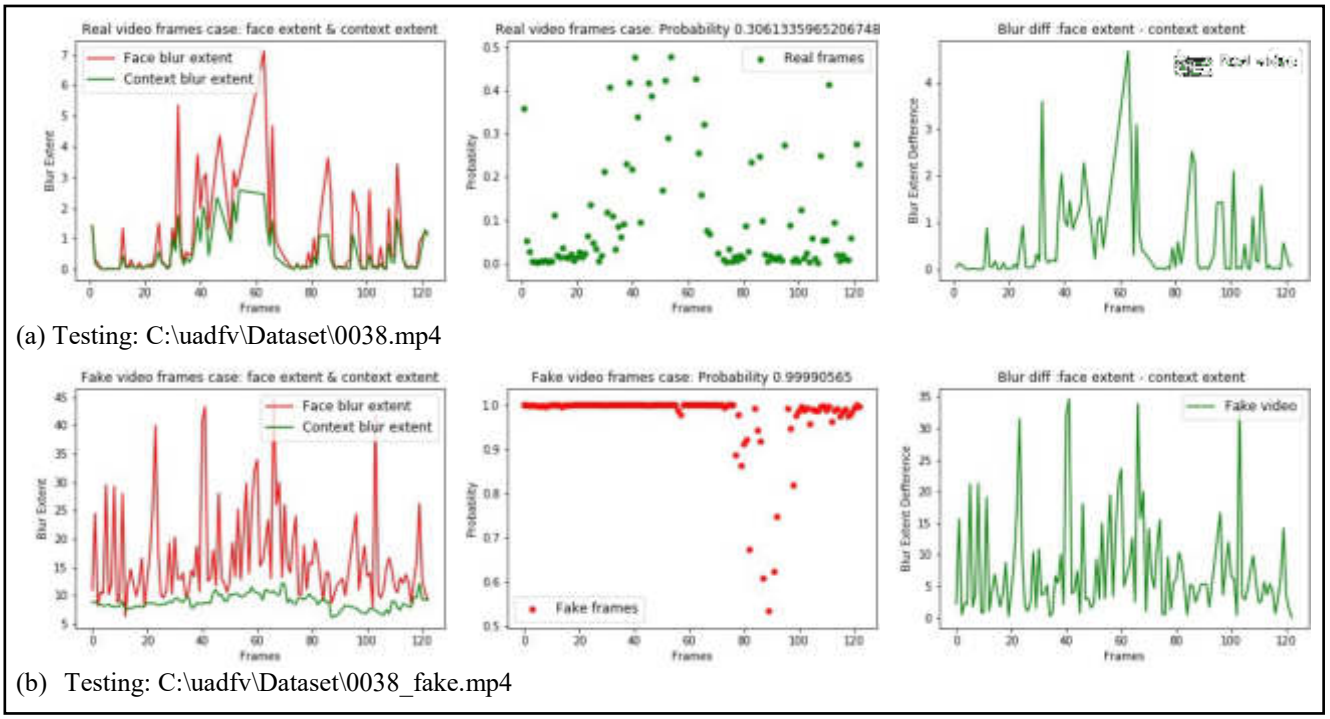
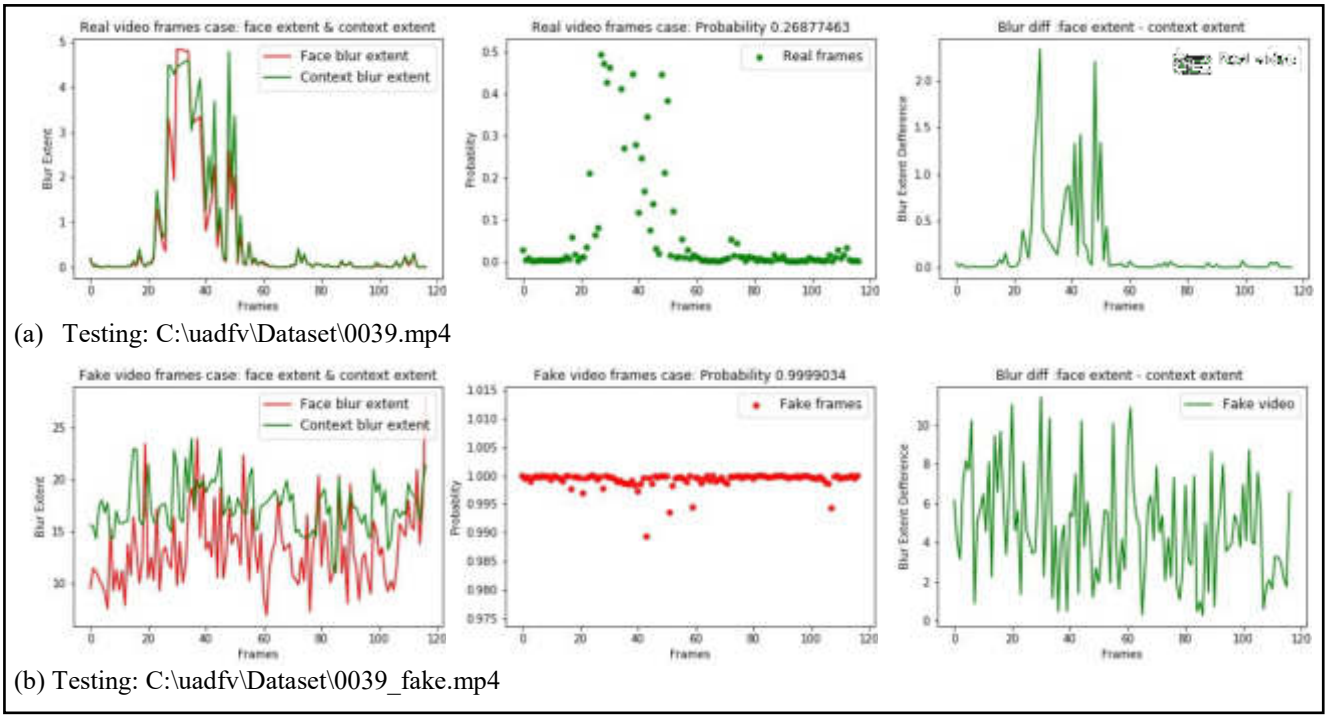


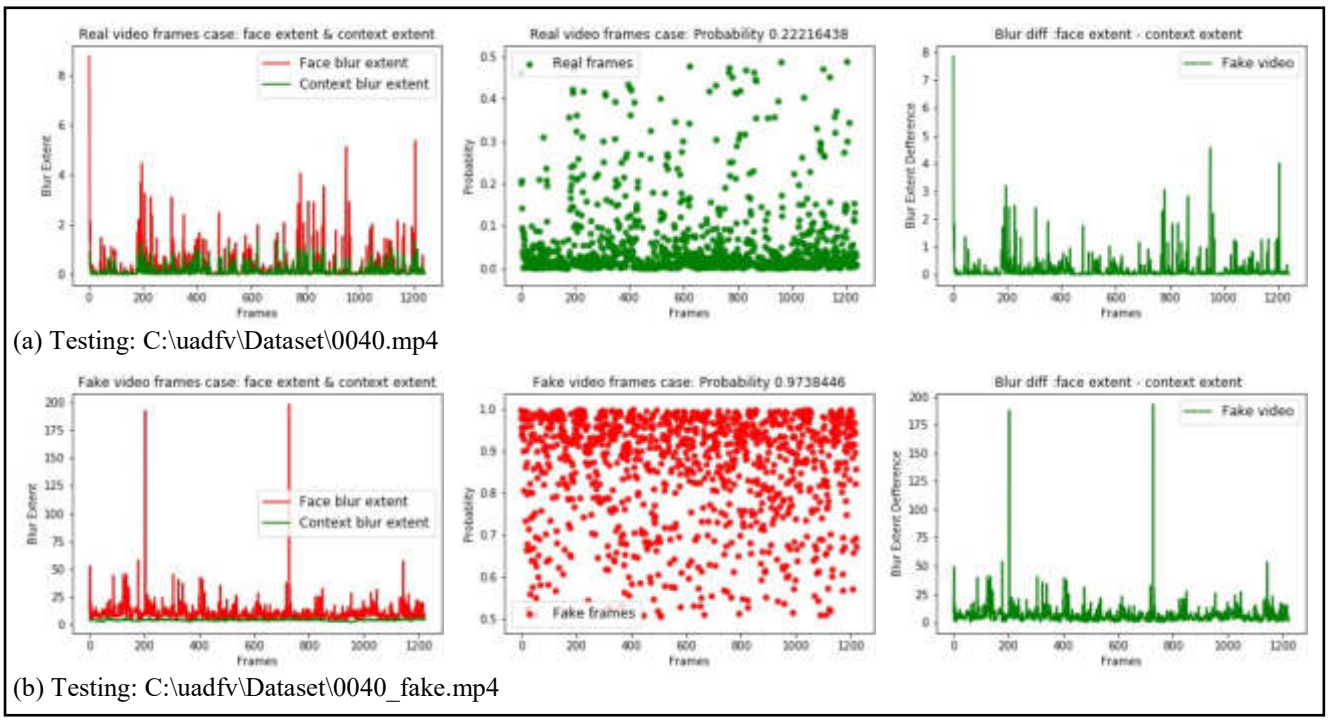
Figure (4.40): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0038.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0039.mp4

(b) Testing: C:\uadfv\Dataset\0039_fake.mp4

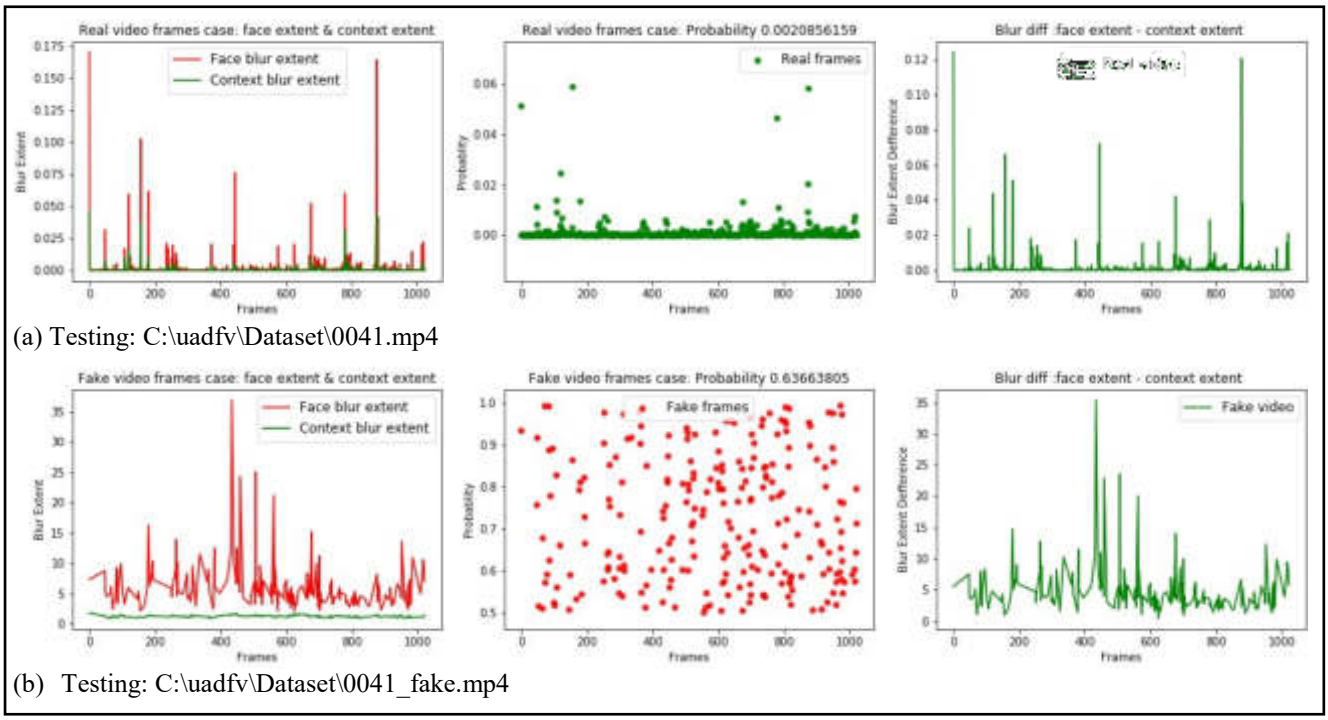
Figure (4.41): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0039.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0040.mp4

(b) Testing: C:\uadfv\Dataset\0040_fake.mp4

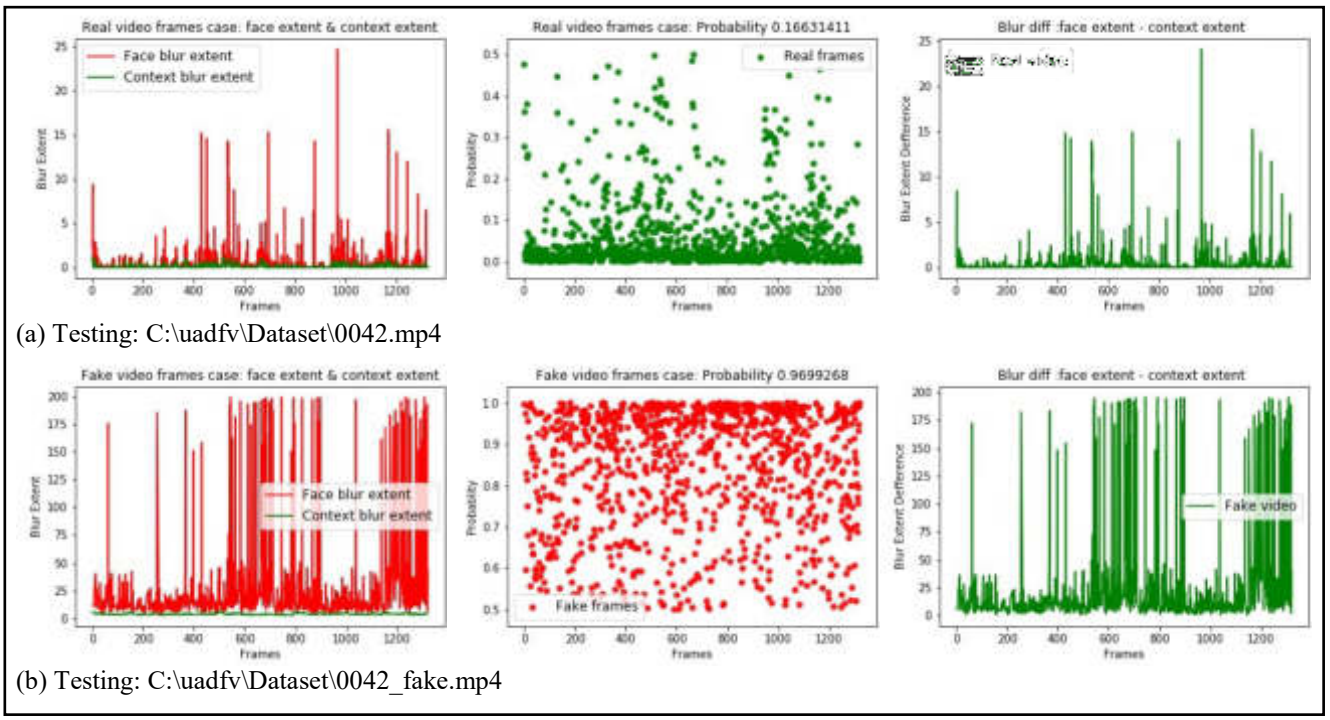
Figure (4.42): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0040.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0041.mp4

(b) Testing: C:\uadfv\Dataset\0041_fake.mp4

Figure (4.43): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0041.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0042.mp4

(b) Testing: C:\uadfv\Dataset\0042_fake.mp4

Figure (4.44): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0042.mp4". (a) Real video-version, and (b) Fake video-version.

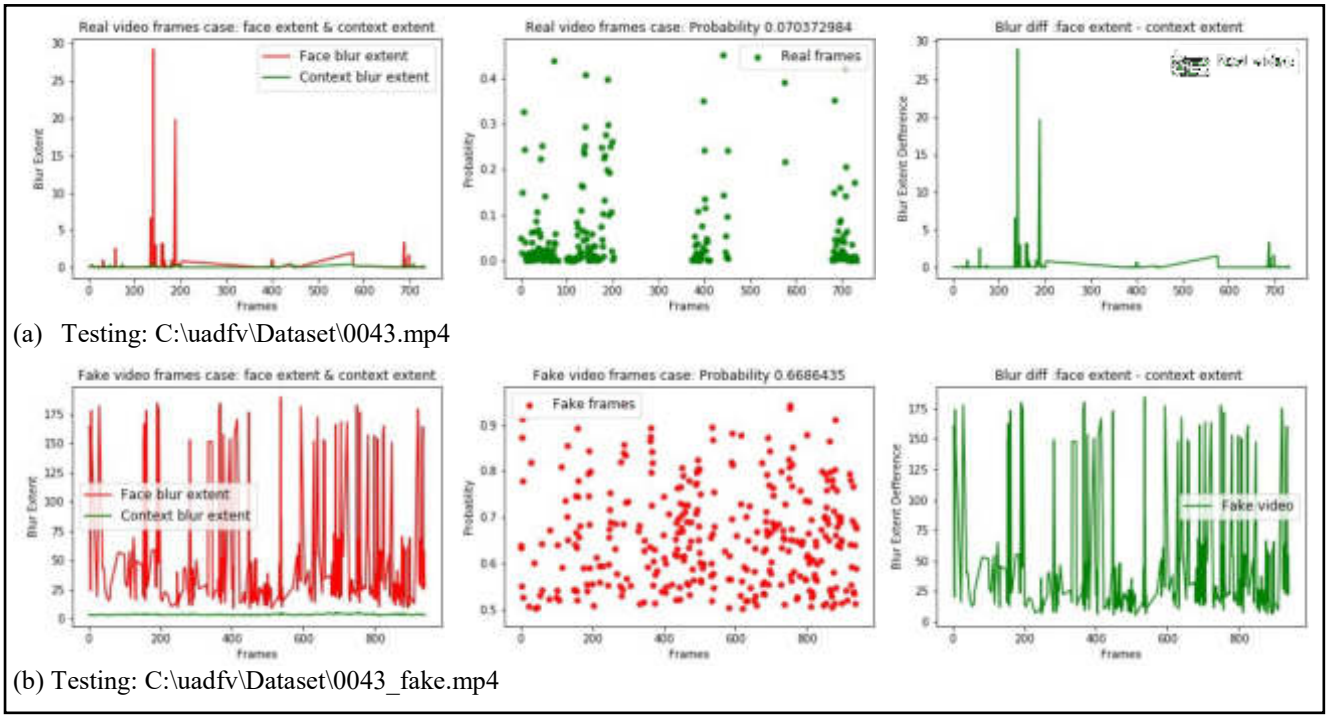


Figure (4.45): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0043.mp4". (a) Real video-version, and (b) Fake video-version.

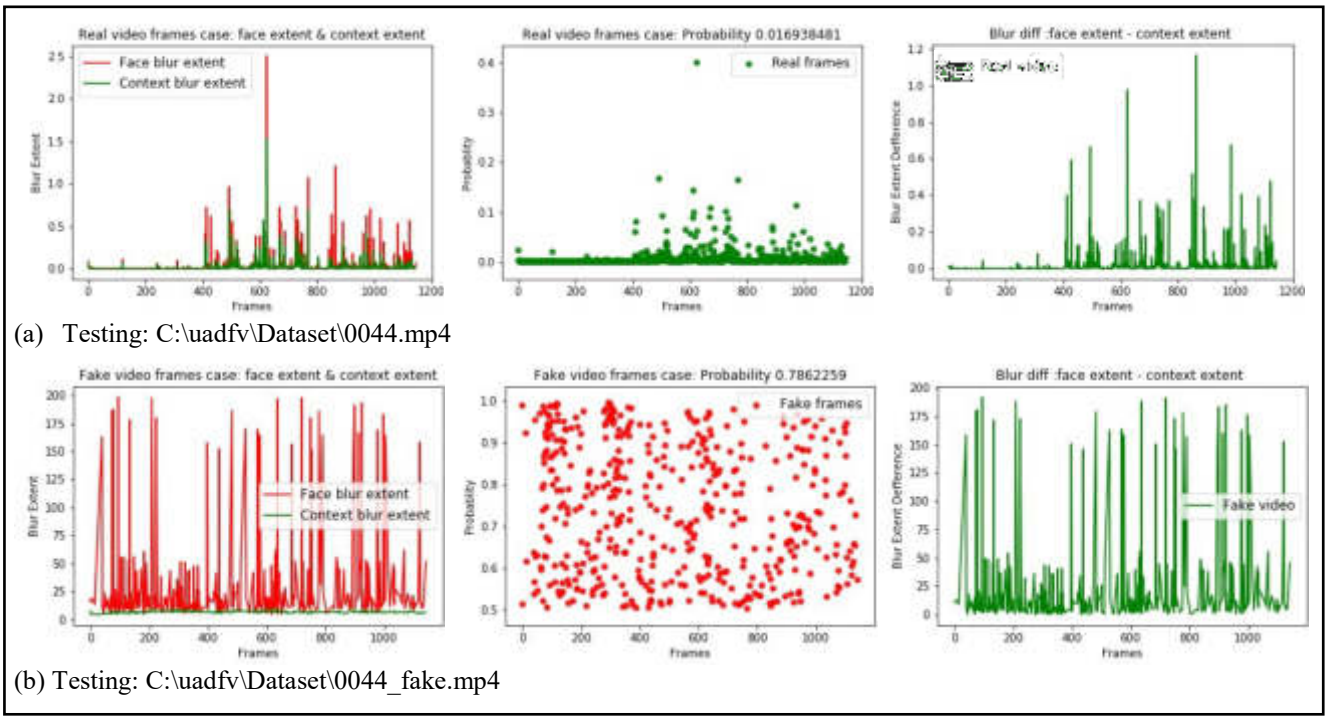
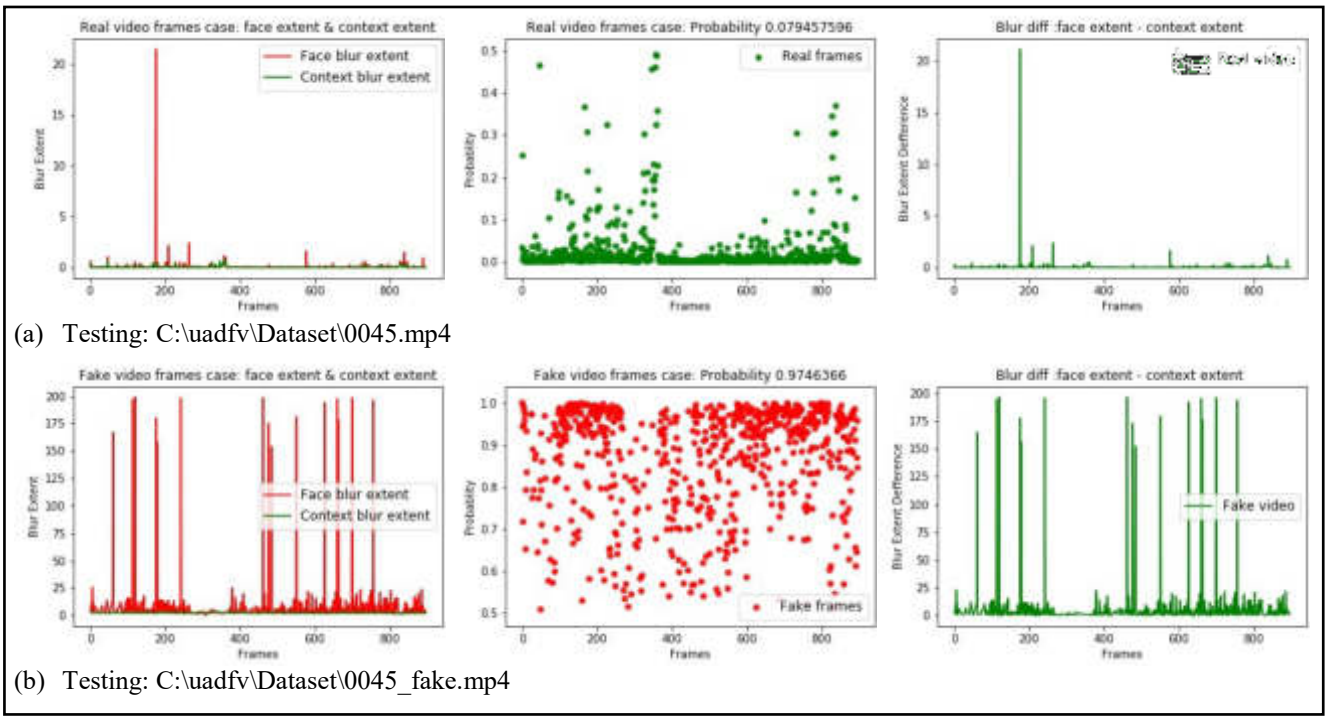


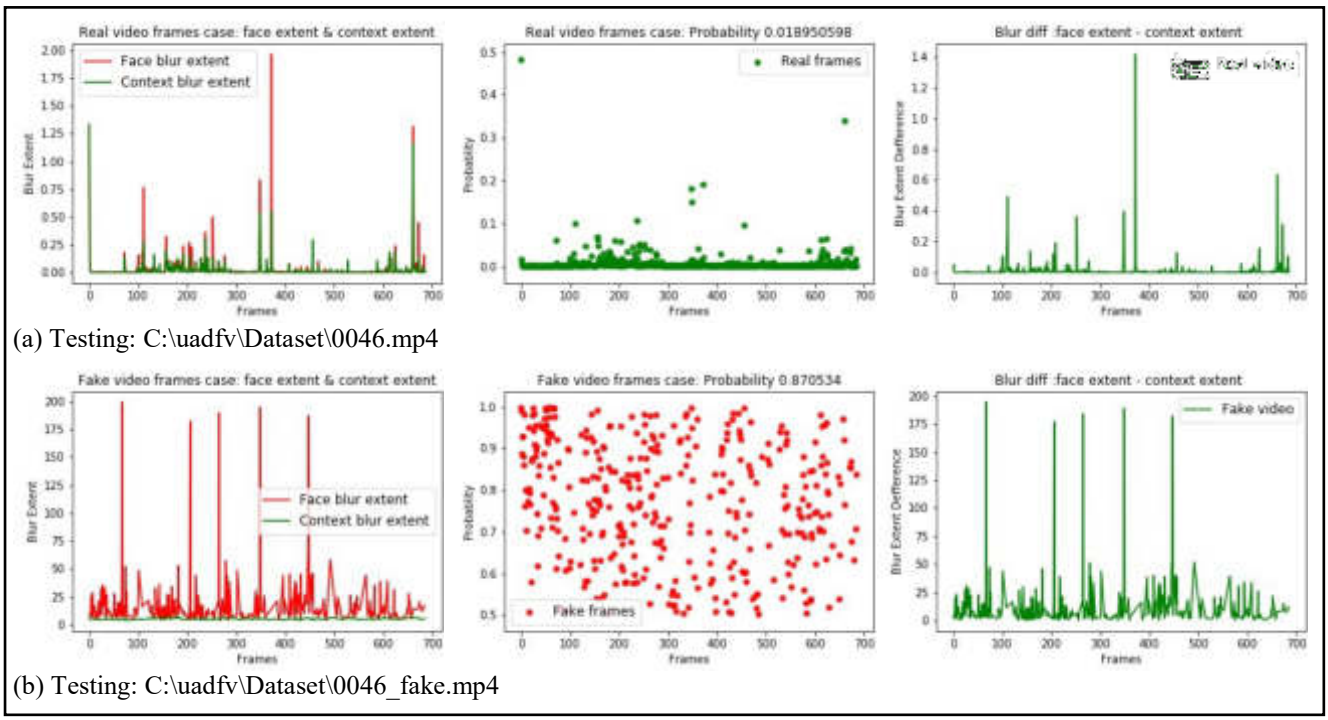
Figure (4.46): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0044.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0045.mp4

(b) Testing: C:\uadfv\Dataset\0045_fake.mp4

Figure (4.47): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0045.mp4". (a) Real video-version, and (b) Fake video-version.



(a) Testing: C:\uadfv\Dataset\0046.mp4

(b) Testing: C:\uadfv\Dataset\0046_fake.mp4

Figure (4.48): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0046.mp4". (a) Real video-version, and (b) Fake video-version.

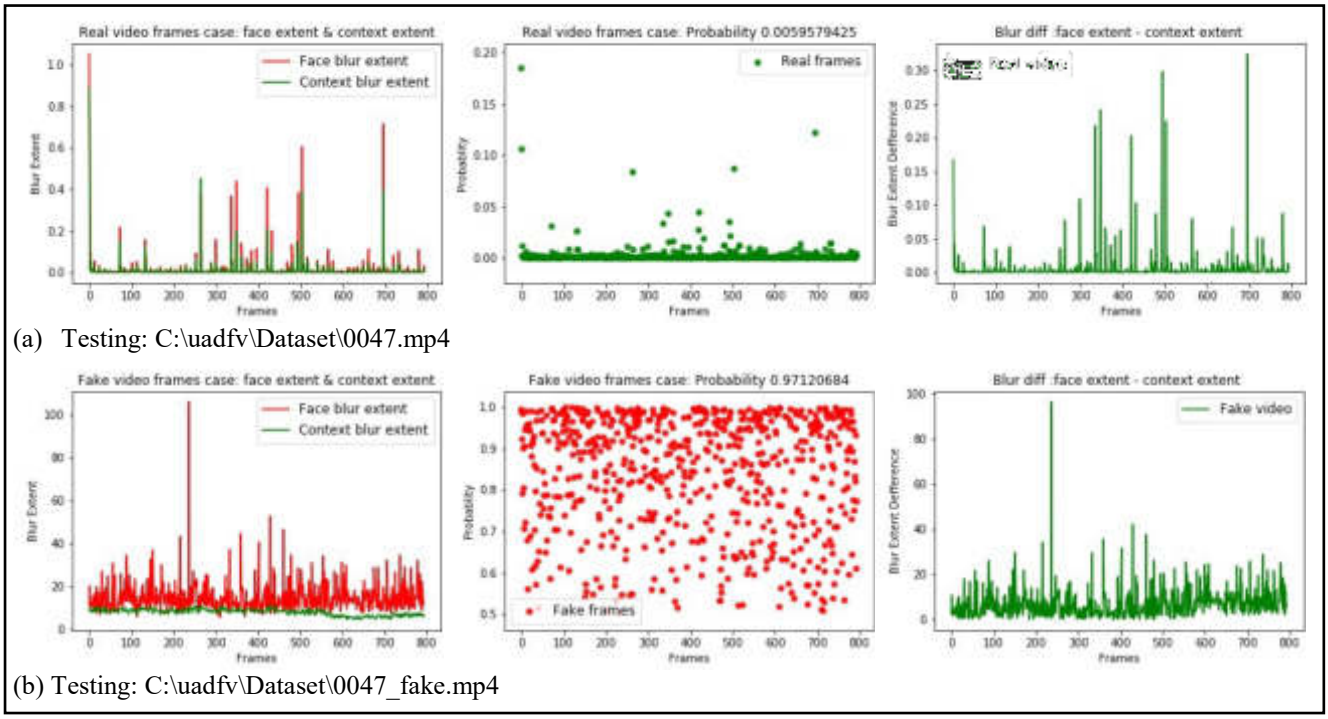


Figure (4.49): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0047.mp4". (a) Real video-version, and (b) Fake video-version.

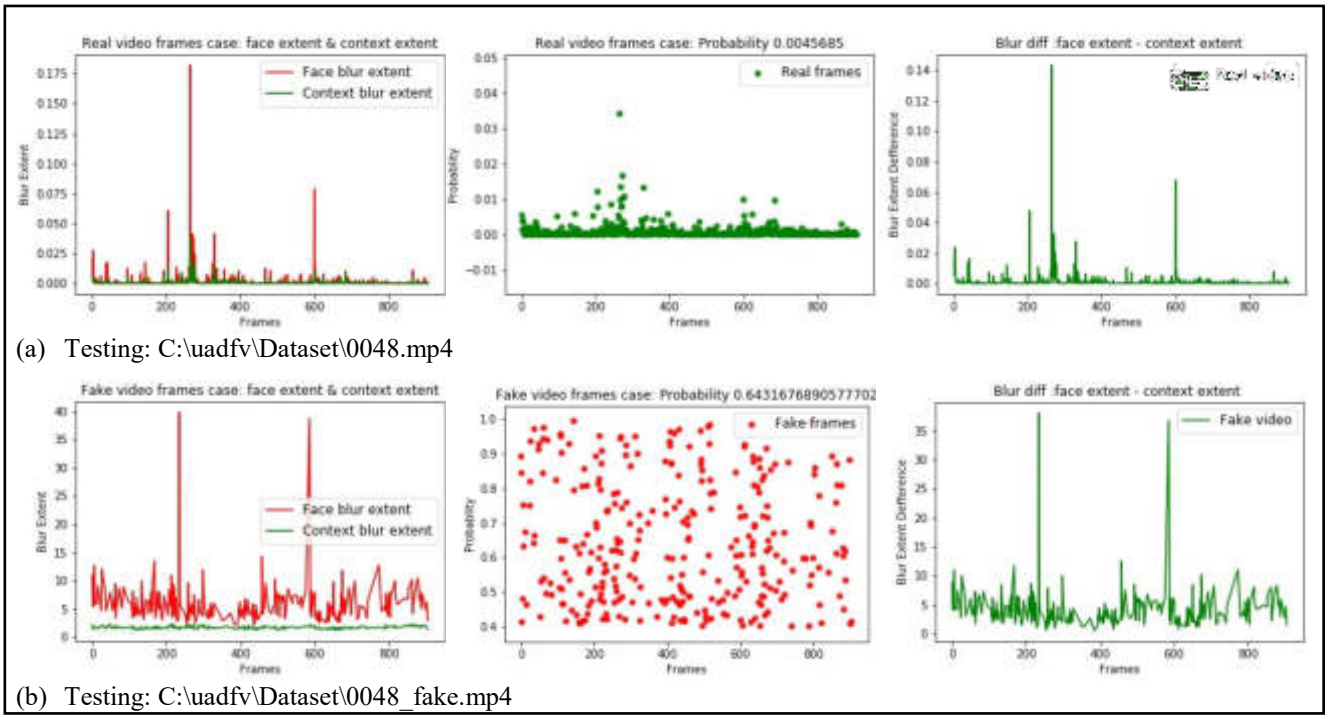


Figure (4.50): The face blur extent & context blur extent, probability of being fake, and blur differences for video "0048.mp4". (a) Real video-version, and (b) Fake video-version.

4.5 Test sample

A test sample is shown in Figure 4.51, the video is one of the "UADFV" datasets, with a length of 292 frames (approximately 10 sec) which prove the strength of the proposed method in the detection of fake synthesized face. The results show the difference between the sharpness edges of (ROI) area by the edge sharpness value (the large value the more sharpness the edges are). A comparison of the methods that have been using the UADFV dataset is shown in table (4.3).

Table 4.3: A comparison of DeepFake methods that are tested with the UADFV dataset.

Methods	Study	Classifier	Performance	Year
Two-stream NN	Zhou et ta. [16]	CNN+SVM	85.1%	2017
Meso-4	Afchar et al. [15]	CNN	84.3%	2018
Head Pose	Yang et al. [11]	SVM	89.0%	2019
Face Warping Features	Li et al. [24]	CNN	97.7%	2019
Proposed Model		CNN+DWT	100%	2020

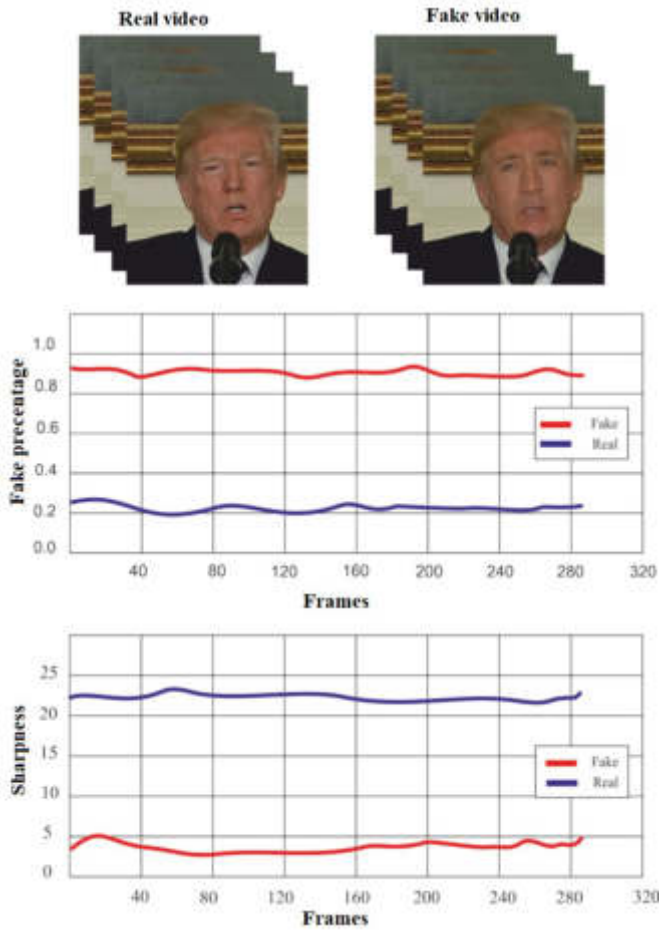


Figure 4.51: Sample test of the proposed method on one of the "UADFV" DeepFake dataset.

Chapter Five

Conclusions and Future Works

CHAPTER FIVE
CONCLUSIONS AND FUTURE WORKS

5.1 Conclusion

This thesis proposes a new approach to detect the forged videos generated by GANs called DeepFake videos. This detection process depends mainly on the imitations of current DeepFake software which generates limited low-resolution synthesized faces, thus causes distinctive artifacts, where a blur function is used to diminish the presence of these artifacts.

Several conclusions have been stimulated through this work and the obtained test results for the proposed system. Some of these conclusions are summarized in the following remarks:

1. This method differs from previously explored methods because the networks make decisions by incorporating artificial intelligence with the DWT.
2. An exhaustive analysis of the DeepFakes detection evolution, focusing on facial regions and fake detection performance.
3. The Wavelet transform is a powerful tool to fulfill this purpose by detecting the blur by the edges type.
4. The testing results from experimentation on the “UADFV” Dataset indicate that this method is very effective with high accuracy with significantly less time compared to other methods.
5. Although the proposed algorithm can detect fake videos with a satisfactory success rate. However, the proposed algorithm still has some limitations (e.g. a bit of accuracy is influenced by the change of illumination, video stream resolution, and noise ratio in the background).

5.2 Future Work

It is observable that combat and the challenge between those who employ advanced machine learning to produce DeepFakes videos with people who make a great effort to detect DeepFakes videos are growing and increasing. Due to the increase of DeepFakes quality and its performance, the methods of detection need to be amended and improve accordingly. This work will hope to be continued in the future in several directions as explained in the following suggestions:

1- The revelation and inspiration should concentrate on what has broken by artificial intelligence can be fixed by artificial intelligence as well.

2- The methods of detection are still in their early stages and various methods have been suggested and evaluated but using different datasets. It would be a good idea to create a growing dataset and updated benchmark of DeepFakes detection methods to validate the continuous development. By doing that the training process of detection models will be facilitated, especially those which require a major training set.

3- Furthermore, most of the current detection methods depend on the drawbacks of the DeepFake production pipelines, i.e. finding weaknesses of the processes to attack them. This type of knowledge and information is not always easily obtainable in adversarial environments where attackers usually try not to reveal such creation technologies of DeepFake video. That represents detection method development real challenge and research in the future should concentrate on introducing more robust, generalizable, and scalable methods.

4- Moreover, research should be directed into distribution platforms such as social media to increase the effectiveness in dealing with the diffuse impact of DeepFakes by using integrated detection methods. Using effective detection methods such as filtering or screening technique mechanism can be performed on these platforms to facilitate DeepFakes detection.

5- The new blockchain technology has been introduced effectively in many fields, but very few studies so far handle the DeepFake detection problems based on this technology. Where it can produce a unique unchangeable blockchain of metadata, it is a great solution tool for digital provenance. The research direction of blockchain technology integration is still far from being a mature solution.

References

References

[1] L. Borges, B. Martins, P. J. J. o. D. Calado, and I. Quality, "Combining similarity features and deep representation learning for stance detection in the context of checking fake news," vol. 11, no. 3, pp. 1-26, 2019.

[2] J. Farkas and J. Schou, *Post-truth, fake news and democracy: Mapping the politics of falsehood*. Routledge, 2019.

[3] S. M. Jang and J. K. J. C. i. H. B. Kim, "Third-person effects of fake news: Fake news regulation and media literacy interventions," vol. 80, pp. 295-302, 2018.

[4] M. Aldwairi and A. J. P. C. S. Alwahedi, "Detecting fake news in social media networks," vol. 141, pp. 215-222, 2018.

[5] S. J. K. P. L. Kemp, We Are Social Ltd. and H. Inc, "Digital 2019: Essential insights into how people around the world use the internet, mobile devices, social media, and E-commerce," 2019.

[6] J. Clement, " Worldwide digital population as of January 2020," *https://www.statista.com/statistics/617136/digital-population-worldwide/,* 2020.

[7] Á. Figueira and L. J. P. C. S. Oliveira, "The current state of fake news: challenges and opportunities," vol. 121, pp. 817-825, 2017.

[8] K. E. J. L. H. T. N. Anderson, "Getting acquainted with social networks and apps: combating fake news on social media," 2018.

[9] J. J. T. J. Fletcher, "Deepfakes, Artificial Intelligence, and Some Kind of Dystopia: The New Faces of Online Post-Fact Performance," vol. 70, no. 4, pp. 455-471, 2018.

[10] A. Qayyum, J. Qadir, M. U. Janjua, and F. J. I. P. Sher, "Using Blockchain to Rein in the New Post-Truth World and Check the Spread of Fake News," vol. 21, no. 4, pp. 16-24, 2019.

[11] X. Yang, Y. Li, and S. Lyu, "Exposing deep fakes using inconsistent head pose," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2019, pp. 8261-8265: IEEE.

[12] Y. Li, X. Yang, P. Sun, H. Qi, and S. J. a. p. a. Lyu, "Celeb-df: A new dataset for deepfake forensics," 2019.

[13] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672-2680.

[14] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. J. N. N. Maida, "Deep learning in spiking neural networks," vol. 111, pp. 47-63, 2019.

[15] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "Mesonet: a compact facial video forgery detection network," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2018, pp. 1-7: IEEE.

[16] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, "Two-stream neural networks for tampered face detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1831-1839: IEEE.

[17] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "Faceforensics++: Learning to detect manipulated facial images," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1-11.

[18] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Icdar*, 2003, vol. 3, no. 2003.

[19] M. Koopman, A. M. Rodriguez, and Z. Geradts, "Detection of deepfake video manipulation," in *Conference: IMVIP*, 2018.

[20] J. Lukas, J. Fridrich, M. J. I. T. o. I. F. Goljan, and Security, "Digital camera identification from sensor pattern noise," vol. 1, no. 2, pp. 205-214, 2006.

[21] K. Rosenfeld and H. T. Sencar, "A study of the robustness of PRNU-based camera identification," in *Media Forensics and Security*, 2009, vol. 7254, p. 72540M: International Society for Optics and Photonics.

[22] Y. Li, M.-C. Chang, and S. Lyu, "In ictu oculi: Exposing ai created fake videos by detecting eye blinking," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, Hong Kong, 2018, pp. 1-7: IEEE.

[23] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625-2634.

[24] Y. Li and S. J. a. p. a. Lyu, "Exposing deepfake videos by detecting face warping artifacts," 2018.

[25] P. Korshunov and S. Marcel, "Speaker inconsistency detection in tampered video," in *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 2375-2379: IEEE.

[26] K. Simonyan and A. J. a. p. a. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.

[28] E. Sabir, J. Cheng, A. Jaiswal, W. AbdAlmageed, I. Masi, and P. J. I. Natarajan, "Recurrent convolutional strategies for face manipulation detection in videos," vol. 3, no. 1, 2019.

[29] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. J. a. p. a. Nießner, "Faceforensics: A large-scale video dataset for forgery detection in human faces," 2018.

[30] H. H. Nguyen, J. Yamagishi, and I. Echizen, "Capsule-forensics: Using capsule networks to detect forged images and videos," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2019, pp. 2307-2311: IEEE.

[31] I. Chingovska, A. Anjos, and S. Marcel, "On the effectiveness of local binary patterns in face anti-spoofing," in *2012 BIOSIG-proceedings of the international conference of biometrics special interest group (BIOSIG)*, 2012, pp. 1-7: IEEE.

[32] O. de Lima, S. Franklin, S. Basu, B. Karwoski, and A. J. a. p. a. George, "Deepfake Detection using Spatiotemporal Convolutional Networks," 2020.

[33] S. Dack, "Deep Fakes, Fake News, and What Comes Next," ed: Retrieved from The Henry M. Jackson School of International Studies: https …, 2019.

[34] T. D. Fikse, "Imagining deceptive deepfakes: an ethnographic exploration of fake videos," University of Oslo, ESST – Society, Science, and Technology in Europe, 2018.

[35] M. H. Khudhur, J. Waleed, H. Hatem, A. M. Abduldaim, and D. A. Abdullah, "An Efficient and Fast Digital Image Copy-Move Forensic Technique," in *2018 2nd International Conference for Engineering, Technology and Sciences of Al-Kitab (ICETS)*, 2018, pp. 78-82: IEEE.

[36] M. Barni *et al.*, "Aligned and non-aligned double JPEG detection using convolutional neural networks," vol. 49, pp. 153-163, 2017.

[37] Z. Boulkenafet, J. Komulainen, A. J. I. T. o. I. F. Hadid, and Security, "Face spoofing detection using color texture analysis," vol. 11, no. 8, pp. 1818-1830, 2016.

[38] J. Galbally and S. Marcel, "Face anti-spoofing based on general image quality assessment," in *2014 22nd International Conference on Pattern Recognition*, 2014, pp. 1173-1178: IEEE.

[39] S. Vishwanath Peri and A. Dhall, "DisguiseNet: a contrastive approach for disguised face verification in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 25-31.

[40] F. Matern, C. Riess, and M. Stamminger, "Exploiting visual artifacts to expose deepfakes and face manipulations," in *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, 2019, pp. 83-92: IEEE.

[41] H. Li, B. Li, S. Tan, and J. J. a. p. a. Huang, "Detection of deep network generated images using disparities in color components," 2018.

[42] A. Roy, D. Bhalang Tariang, R. Subhra Chakraborty, and R. Naskar, "Discrete cosine transform residual feature-based filtering forgery and splicing detection in JPEG images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1552-1560.

[43] S. Tariq, S. Lee, H. Kim, Y. Shin, and S. S. Woo, "Detecting both machine and human-created fake face images in the wild," in *Proceedings of the 2nd International Workshop on Multimedia Privacy and Security*, 2018, pp. 81-87.

[44] N. Sebe, I. Cohen, A. Garg, and T. S. Huang, *Machine learning in computer vision*. Springer Science & Business Media, 2005.

[45] K. Schawinski, C. Zhang, H. Zhang, L. Fowler, and G. K. J. M. N. o. t. R. A. S. L. Santhanam, "Generative adversarial networks recover features in astrophysical images of galaxies beyond the deconvolution limit," vol. 467, no. 1, pp. L110-L114, 2017.

[46] C. Zeng and R. Olivera-Cintrón, "Preparing for the World of a" Perfect" Deepfake," 2019.

[47] A. Osokin, A. Chessel, R. E. Carazo Salas, and F. Vaggi, "GANs for biological-image synthesis," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2233-2242.

[48] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of RGB videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2387-2395.

[49] M. A. Nielsen, *Neural networks and deep learning*. Determination press San Francisco, CA, 2015.

[50] N. Sharma, V. Jain, and A. J. P. C. S. Mishra, "An analysis of convolutional neural networks for image classification," vol. 132, pp. 377-384, 2018.

[51] M. Telgarsky, "benefits of depth in neural networks," in *Conference on Learning Theory*, 2016, pp. 1517-1539.

[52] X. Shen, W. Chen, G. Zhao, and P. J. F. i. P. Hu, "Recognizing Micro-expression: An Interdisciplinary Perspective," vol. 10, p. 1318, 2019.

[53] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European conference on computer vision*, 2016, pp. 646-661: Springer.

[54] M. J. I. T. o. S. Basu, Man, and P. C. Cybernetics, "Gaussian-based edge-detection methods-a survey," vol. 32, no. 3, pp. 252-260, 2002.

[55] A. J. M. A. Haar, "Zur theorie der orthogonalen funktionensysteme," vol. 71, no. 1, pp. 38-53, 1911.

[56] R. S. Stanković, B. J. J. C. Falkowski, and E. Engineering, "The Haar wavelet transform: it's status and achievements," vol. 29, no. 1, pp. 25-44, 2003.

[57] L. Zeng, C. P. Jansen, S. Marsch, M. Unser, and P. R. J. I. t. o. m. i. Hunziker, "Four-dimensional wavelet compression of arbitrarily sized echocardiographic data," vol. 21, no. 9, pp. 1179-1187, 2002.

[58] R. L. Claypoole, G. Davis, W. Sweldens, and R. G. Baraniuk, "Adaptive wavelet transforms for image coding," in *Asilomar Conference on Signals, Systems, and Computers*, 1997.
[59] A. Muñoz, R. Ertlé, and M. J. S. p. Unser, "Continuous wavelet transform with arbitrary scales and O (N) complexity," vol. 82, no. 5, pp. 749-757, 2002.
[60] P. Porwik and L. Agnieszka, "The new graphic description of the Haar wavelet transform," in *International Conference on Computational Science*, 2004, pp. 1-8: Springer.
[61] I. Drori, D. J. I. t. o. v. Lischinski, and c. graphics, "Fast multiresolution image operations in the wavelet domain," vol. 9, no. 3, pp. 395-411, 2003.
[62] R. Ansari, H. Yadav, and A. J. I. J. o. C. A. Jain, "A Survey on Blurred Images with Restoration and Transformation Techniques," vol. 68, no. 22, pp. 29-33, 2013.
[63] Y. Y. Tang, L. Yang, J. J. I. T. o. S. Liu, Man, and P. B. Cybernetics, "Characterization of Dirac-structure edges with wavelet transform," vol. 30, no. 1, pp. 93-109, 2000.
[64] M. Masoumi and S. J. I. J. o. C. A. Amiri, "A high capacity digital watermarking scheme for copyright protection of video data based on YCbCr color channels invariant to geometric and non-geometric attacks," vol. 51, no. 13, 2012.
[65] Y. Y. Tang, L. Yang, and L. Feng, "Characterization and detection of edges by Lipschitz exponents and MASW wavelet transform," in *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No. 98EX170)*, 1998, vol. 2, pp. 1572-1574: IEEE.
[66] Y. Tang and L. Feng, "Multi-resolution adaptive wavelet edge detection," in *The 2th International Conference on Multimodal Interface*, 1999.
[67] H. Tong, M. Li, H. Zhang, and C. Zhang, "Blur detection for digital images using wavelet transform," in *2004 IEEE international conference on multimedia and expo (ICME)(IEEE Cat. No. 04TH8763)*, 2004, vol. 1, pp. 17-20: IEEE.
[68] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, 2005, vol. 1, pp. 886-893: IEEE.
[69] Dlib. (2015). *Dlib C++ Library*. Available: http://dlib.net/intro.html
[70] J. Hale, "Deep learning framework power scores 2018–towards data science," ed, 2018.
[71] Z. Zhang, "Detect forgery video by performing transfer learning on Deep Neural Network," 2019.
[72] H. Abdi and L. J. J. E. o. r. d. Williams, "Normalizing data," vol. 1, 2010.

المُستخلص

إن الدخول السهل والمجاني إلى قواعد بيانات عامة واسعة النطاق ، والتقدم السريع لتقنيات التعلم العميق ، وتطبيقات الذكاء الاصطناعي ، وخاصة شبكات الخصومة التوليدية ، ادى إلى إنتاج مقاطع فيديو مزيفة وبمستوى متقدم الواقعية مولداً آثاراً سلبية تجاه المجتمع. أصبحت الحدود بين الوسائط المرئية الحقيقية والاصطناعية (المزيفة) ضعيفة للغاية ومن الصعب التمييز بينهما. فتح هذا المجال الباب أمام سلسلة من التطبيقات المثيرة في مجالات مختلفة مثل ألعاب الفيديو وإنتاج الأفلام والإعلانات ، ومن ناحية أخرى فإنه يشكل تهديدات هائلة للأمن.

تسمح حزم البرامج المتاحة مجانًا على الويب لأي شخص مع الحد الأدنى من المهارات بإنتاج مقاطع فيديو مزيفة وواقعية للغاية ، وهذا ما تم تسميته "DeepFakes". تستخدم هذه التكنولوجيا في بعض الاحيان لابتزاز الناس أو تشويه سمعتهم ، والتلاعب برأي الجمهور أثناء الانتخابات حيث لا يوجد حدود للانتهاكات المحتملة من قبل الخيال البشري ، مما ولدى حاجة ماسة إلى أدوات آلية لديها القدرة على اكتشاف وتجنب انتشار محتوى الوسائط المتعددة المزيفة والخطيرة.

في هذا العمل ، نصف تقنية جديدة يستخدم فيها التحويل المويج لتحديد مدى التمويه لمنطقة الوجه البشري والسياق المحيط به باستخدام نوع الحواف في الصورة. يمكن لهذا النهج أن يميز بنجاح مقاطع الفيديو المزيفة التي تم إنشاؤها بواسطة الذكاء الاصطناعي عن مقاطع الفيديو الأصلية.

استنادًا إلى الملاحظات التي تفيد بأن مجموعة البرمجيات "DeepFake" السائدة يمكنها فقط إنشاء صور للوجوه المركبة بدقة مقيدة ومحددة، والتي تتطلب اضافة تمويه وإخفاء المعالم الدقيقة للوجة في الفيديو المصدر. بالتأكيد ، ستخلق مثل هذه التغييرات قطعاً تمييزية "Artifacts" لا لبس فيها ، ويمكن للطريقة المقترحة التقاط هذه القطع التمييزية بفعالية من خلال الكشف عن أنواع الحواف. تحتاج معظم الأساليب السابقة إلى قدر كبير من الصور التي تم إنشاؤها بواسطة "DeepFake" والتصوير الحقيقي لتدريب مصنف "CNN". حيث لا تحتاج التقنية المستخدمة في الطريقة المقترحة إلى عناء مع حالات تصوير DeepFake المنتج كتجهيز سلبي للتدريب لأنه يركز على القطع التمييزية كميزة في تزييف الوجه لمعرفة ما إذا كانت مقاطع الفيديو حقيقية أم مزيفة.

مع استخدام الأنظمة العصبية التلافيفية (CNNs) لالتقاط القطع التمييزية ، تم تعزيز الطريقة بالاستعانة بشبكات العصبية المدربة مسبقا لإعطاء نتائج أكثر دقة. مزايا الطريقة المقترحة هي:

(1) يمكن رصد القطع التمييزية المذكورة مباشرة باستخدام عمليات معالجة بسيطة على الصورة مما يوفر الموارد "Computing Resources" والوقت ، بينما الطرق السابقة كانت بحاجة لتدريب نموذج DeepFake لتوليد أمثلة سلبية مما يتطلب موارد اضافية واستهلاكاً للوقت.

(2) تعتبر التقنية المستخدمة أكثر قوة مقارنة بالآخرين حيث هناك الكثير من القطع التمييزية بشكل عام في اكثرية مقاطع فيديو الـ DeepFake.

(3) الطريقة المقترحة في هذه الأطروحة تم تجربتها باستخدام مجموعة بيانات UADFV ونتيجة التجربة بأكملها أعطت دقة جيدة للغاية مع موثوقية كبيرة ، وصلت إلى 100٪ مع بعض الفروق الدقيقة في كل اختبار فيديو.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة ديالى
كلية العلوم
قسم علوم الحاسوب

نظام اكتشاف التزيف العميق

رسالة

مقدمة الى قسم علوم الحاسوب – كلية العلوم – جامعة ديالى كجزء من متطلبات نيل درجة الماجستير في إختصاص علوم الحاسوب

من قبل

محمد اكرم يونس ذنون الساعاتي

بإشراف

أ.م.د. طه محمد حسن

1442هـ

2020م