

Cairo University

Faculty of Engineering

Department of Computer Engineering

**Revelio**

We reveal secrets.



A Graduation Project Report Submitted to

**Faculty of Engineering, Cairo University**

in Partial Fulfillment of the requirements of the degree of

**Bachelor of Science in Computer Engineering**

**Presented by**

Ahmed Ayman Ahmed Ibrahim

**Supervised by**

Prof. Hoda Baraka

11 Jul. 2023

All rights reserved. This report may not be reproduced in whole or in part, by photocopying or other means, without the permission of the authors/department.

Table of Contents

[List of Figures 3](#_Toc139819453)

[List of Tables 4](#_Toc139819454)

[List of Abbreviations 6](#_Toc139819455)

[List of Symbols 6](#_Toc139819456)

[Contacts 7](#_Toc139819457)

[Chapter 1: Introduction 9](#_Toc139819458)

[1.1. Contribution 10](#_Toc139819459)

[1.2. Document Organization 11](#_Toc139819460)

[Chapter 2: Literature Survey 12](#_Toc139819461)

[2.1. Comparative Study of Previous Work 12](#_Toc139819462)

[2.2. Implemented Approach 12](#_Toc139819463)

[Chapter 3: System Design and Architecture 13](#_Toc139819464)

[3.1. Dynamic Texture Analysis 13](#_Toc139819465)

[3.1.1. Functional Description 13](#_Toc139819466)

[3.1.2. Modular Decomposition 13](#_Toc139819467)

[3.1.3. Design Constraints 26](#_Toc139819468)

[Chapter 4: System Testing and Verification 28](#_Toc139819469)

[Chapter 5: Conclusions 30](#_Toc139819470)

[References 31](#_Toc139819471)

# List of Figures

[Figure ‎1.1 Our whole system with a frame around the modules built by me 10](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819439)

[Figure ‎3.1 pre-processing pipeline 14](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819440)

[Figure ‎3.2 output image of LBP 15](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819441)

[Figure ‎3.3 three orthogonal planes [1] 16](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819442)

[Figure ‎3.4 3 × 3 neighborhood 16](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819443)

[Figure ‎3.5 α Directions 17](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819444)

[Figure ‎3.6 The output of the I function in four directions. 17](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819445)

[Figure ‎3.7 Temporal modes 17](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819446)

[Figure ‎3.8 Training pipeline 19](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819447)

[Figure ‎3.9 Testing Pipeline 19](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819448)

[Figure ‎3.10 Classification accuracy per manipulation technique 22](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819449)

[Figure ‎3.11 multi-class accuracy 24](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819450)

[Figure ‎3.12 Classification accuracy per manipulation technique in case of strong video compression 25](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819451)

[Figure ‎3.13 same frame with cf=23 & cf=40 26](file:///C:\Users\CMP\Downloads\Dynamic%20texture%20analysis%20report.docx#_Toc139819452)

# List of Tables

[Table 1: Comparing some SOFA Deep Learning-based methods on face manipulation detection methods. 12](#_Toc139819412)

[Table 2:Comparing some classical ML methods on face manipulation detection methods 12](#_Toc139819413)

[Table 3: LDP^2 (α) Algorithm 16](#_Toc139819414)

[Table 4 Training Machines 19](#_Toc139819415)

[Table 5 training time for each model 20](#_Toc139819416)

[Table 6 Kickoff model hyperparameters 21](#_Toc139819417)

[Table 7 Kickoff model accuracy 21](#_Toc139819418)

[Table 8 Classification accuracy on the single-manipulation scenario 22](#_Toc139819419)

[Table 9 The accuracy of multiple technique classification using binary classifiers. 23](#_Toc139819420)

[Table 10 multi-class accuracy 24](#_Toc139819421)

[Table 11 Classification accuracy on the single-manipulation scenario in case of strong video compression 26](#_Toc139819422)

[Table 12 The accuracy of multiple technique classification using binary classifiers in case of strong video compression. 26](#_Toc139819423)

[Table 13 Dynamic Texture Models accuracies 29](#_Toc139819424)

# List of Abbreviations

CDF Deepfake Classifier

CF Compression Factor

CF2 Face2Face Classifier

CFSW FaceSwap Classifier

CNT NeuralTextures Classifier

DF Deepfake

F2F Face2Face

FF++ FaceForensics++ dataset

FSW FaceSwap

LBP Local Binary Pattern

LDP Local Derivative Pattern

LDP2 The second-order directional

LDP-TOP Local Derivative Pattern on Three Orthogonal Planes

NT NeuralTextures

OR Original

ROI Region of interest

# List of Symbols

→       Direct Mode

←       Inverse Mode

↔       Bidirectional Mode

α        Angular Directions {0°, 45°, 90°, 135°}

F        Full Face Information

         Image Pixel Intensity

T        Top-Half of The Face in LDP

# Contacts

**Team Members**

|  |  |  |
| --- | --- | --- |
| **Name** | **Email** | **Phone Number** |
| Ahmed Ayman Ahmed | [ahmedayman1420@email.com](mailto:ahmedayman1420@email.com) | +201147771061 |
| Ammar Mohamed Sobhi | ammarmohamed13@gmail.com | +2 01116185647 |
| Mohamed Akram Abdelfattah | mohamed1999akram@gmail.com | +2 01032972946 |
| Omar Ahmed Mohamed | omar.ahmed314@hotmail.com | +2 01100086995 |

**Supervisor**

|  |  |  |
| --- | --- | --- |
| **Name** | **Email** | **Number** |
| Prof. Hoda Baraka | Hodabaraka60@gmail.com | +2 01005411083 |

*This page is left intentionally empty.*

# Introduction

Deepfakes, synthetic media generated using deep learning techniques, have gained significant attention in recent years due to their potential to deceive and manipulate visual content. As their sophistication increases, it becomes crucial to develop effective detection methods to combat the spread of misleading information and safeguard digital integrity. This introduction presents a comprehensive overview of a cutting-edge approach for deepfake detection using dynamic texture analysis.

Dynamic texture analysis is a powerful approach that focuses on temporal patterns and statistical properties of video sequences to uncover hidden inconsistencies within manipulated content. By leveraging the inherent dynamics present in genuine videos, this method aims to expose the subtle artifacts introduced during the creation of deepfakes.

The proposed approach consists of several key stages. First, an input video undergoes feature extraction, where spatiotemporal descriptors are computed to capture both appearance and motion characteristics. Next, a dynamic texture model is constructed by modeling the statistical properties of the extracted features. This model serves as a reference for differentiating between real and manipulated content.

## Contribution

During Revelio, my contribution was to the field of deepfake detection by developing a dynamic texture analysis model. Recognizing the growing concern regarding the potential misuse of deepfake technology, my aim was to enhance the accuracy and reliability of existing detection methods. To achieve this, I focused on leveraging the power of local derivative pattern, a technique that analyzes temporal changes in visual patterns.

By utilizing dynamic texture analysis, I successfully designed a robust deepfake detection model capable of effectively distinguishing between authentic and manipulated videos. This approach enabled me to capture subtle variations in the motion dynamics of facial expressions, which are often altered in deepfake videos. Leveraging machine learning algorithms, I trained the model using a diverse dataset containing both real and synthetic videos.

To ensure the model's effectiveness, I conducted extensive experimentation and evaluation, carefully fine-tuning the parameters and optimizing its performance. The results were promising, as the dynamic texture analysis model demonstrated a high accuracy rate in identifying deepfake videos across various datasets and scenarios. It showcased an ability to detect sophisticated modifications, including those generated using state-of-the-art deepfake techniques.

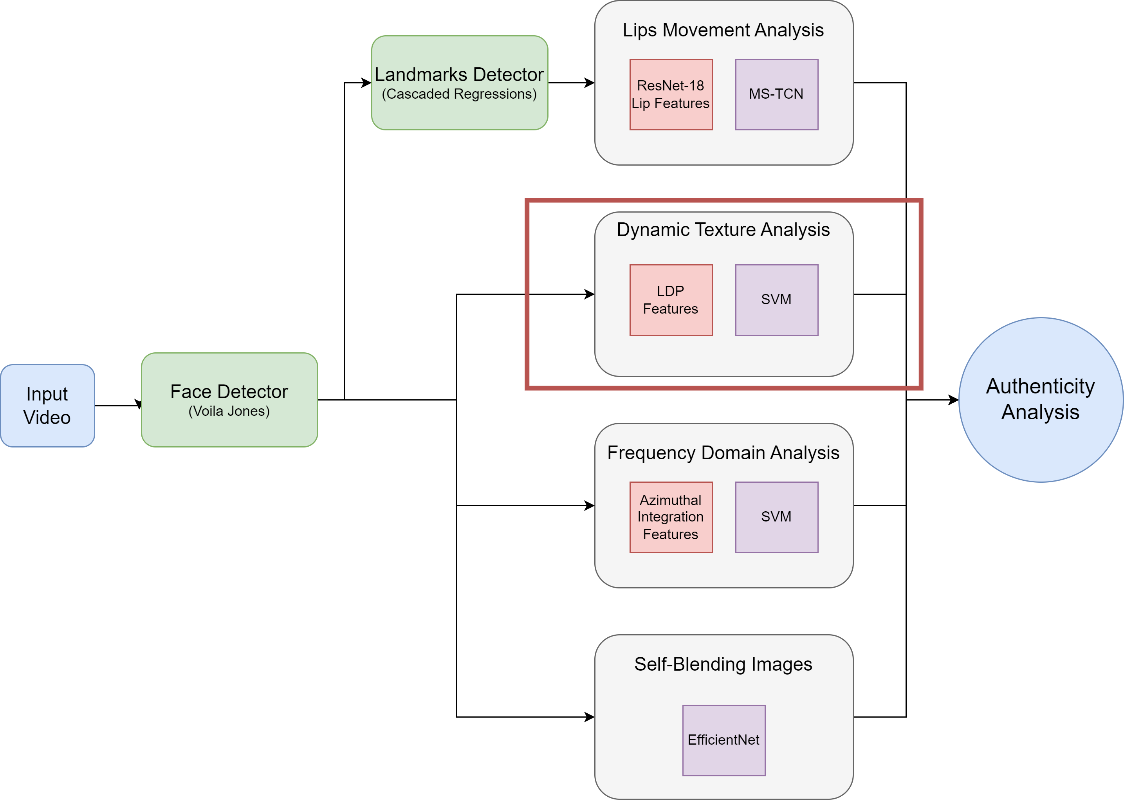


Figure ‎1.1 Our whole system with a frame around the modules built by me

## Document Organization

In Chapter 2 we do a literature survey about the face manipulation detection techniques and give some background information such as types of face manipulations used. In Chapter 3 we discuss the dynamic texture analysis architecture and pipeline and provide details sub-module. In Chapter 4 we show our system testing and verification.

# Literature Survey

## 2.1. Comparative Study of Previous Work

We will show some of the best approaches split into Deep Learning based methods and Classical based methods.

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Performance on FF++** |
| Xception​ | A method based on ImageNet model that is finetuned for binary classification. | AUC = 99.3% (HQ)  AUC = 99.8% (RAW) |
| CNN-aug | Generating synthesized fake dataset using different generator models and training on ProGAN model. | AUC = 99.1% (HQ)  AUC = 99.8% (RAW) |
| Patch-based​ | Train a CNN based on local patches rather the whole face. | AUC = 97.2% (HQ)  AUC = 99.9% (RAW) |
| SBI​ | Generating fake videos by inducing artifacts to real ones and train on EfficeintNet | AUC = 99.6% (RAW) |
| LipForensics | Using ResNet18 + MSTCN model pretrained on Lip reading | AUC = 99.7% (HQ)  AUC = 99.9% (RAW) |

Table 1: Comparing some SOFA Deep Learning-based methods on face manipulation detection methods.

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Performance on FF++** |
| Dynamic Texture Analysis [1] | Using Local Derivative Pattern features to extract temporal texture features and train on SVM | AUC = 95% |
| Frequency Domain Analysis [2] | Analyzing frequency domain of video frames using azimuthal integration features and train on SVM. | Accuracy = 90% |
| Matern et. al [3] | Extracting some visual features and train on MLP | AUC=78% |

Table 2:Comparing some classical ML methods on face manipulation detection methods

## 2.2. Implemented Approach

My module was dynamic Texture Analysis [1], which focuses on analyzing the frames texture of the video by extracting a temporal feature which is LDP, which gives the ability to analyze the video texture-wise.

# System Design and Architecture

## 3.1. Dynamic Texture Analysis

### 3.1.1. Functional Description

The module’s main function is to analyze the spatio-temporal texture dynamics of an input video to classify the real or manipulated sequences. This is achieved through using Local Derivative Pattern on Three Orthogonal Planes (LDP-TOP) that has been specifically designed for texture analysis in video sequences. LDP-TOP extends the original LBP technique by considering local derivatives on three orthogonal planes (horizontal, vertical, and diagonal) instead of just a single plane, enabling a more comprehensive characterization of the dynamic texture patterns. In this module, we targeted both spatial and time domain of the video sequence. This yields relatively large feature representations that can be learned through simpler classifiers, such as linear SVMs.

### 3.1.2. Modular Decomposition

#### 3.1.2.1 Pre-processing

First, video frames are extracted from each video of the dataset and do the following on each frame.

* Face detection: after extracting video frames, dlip library is used to detect the region of interest (ROI). Then we resized the face to (128, 128) and converted to grayscale. After project integration, we used our face detection model instead of python dlip.
* Temporal partition: it is a crucial step in video analysis and processing that involves dividing a video into smaller segments or temporal windows based on time intervals. Once the video frames are converted to grayscale, overlapping temporal windows are isolated. These windows are defined by a duration parameter "d," representing the length of each window in seconds.
* Area selection: At this stage, we introduce the capability to select a specific area of the face for feature analysis. This allows us to determine the importance and relevance of different facial regions in the chosen feature representation. The objective is to understand which regions of the face contribute most significantly to the analysis. In our tests, we consider three distinct cases, denoted by uppercase letters: the top-half (T), the bottom-half (B), or the full-face information (F). These cases refer to the specific region or subset of the face that will be used for subsequent analysis. For the "top half" case, only the upper portion of the face is selected. This includes the forehead and upper part of the eyes. In the "bottom-half" case, only the lower portion of the face is considered for feature analysis. This involves the nose, mouth, and chin. Finally, the "full face information" case encompasses the entire face region. All facial features and regions are taken into account for feature analysis. By testing and comparing these different cases, we gain insights into the relevance and contribution of specific facial regions to the chosen feature representation.

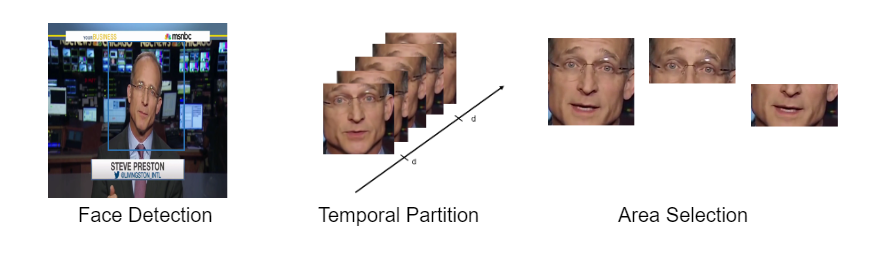


Figure ‎3.1 pre-processing pipeline

The output of the preprocessing stage for each video is a 4D array with a size of (H, W, K, N). Here, (H, W) represents the height and width of each frame (it could be top-half, bottom-half or full face), which in this case is (128, 128) due to frame resizing. K is the result of multiplying the video's frame rate by the time window (d), which gives the total number of frames within that time window. N is calculated by dividing the total video length by the time window (d), representing the number of video partitions.

#### 3.1.2.2 Feature Extraction

##### 3.1.2.2.1 Local Binary Pattern

Local Binary Pattern (LBP) is a widely used and effective texture descriptor in image processing and computer vision. It is particularly useful for tasks such as texture classification, face recognition, and object detection. LBP encodes the local structure and texture information of an image by comparing the pixel values of a central pixel with its neighboring pixels. The LBP algorithm works as follows: for each pixel in an image, a binary code is assigned based on the relative intensity values of the neighboring pixels. Starting from the central pixel, the neighboring pixels are compared to the central pixel, and if the neighbor's intensity is greater than or equal to the central pixel's intensity, a value of 1 is assigned; otherwise, a value of 0 is assigned. This process is performed for each pixel, resulting in a binary pattern for the entire image.

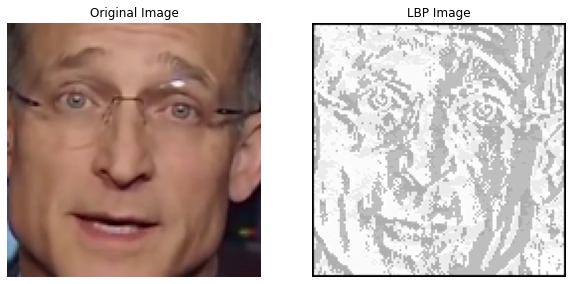


Figure ‎3.2 output image of LBP

##### 3.1.2.2.2 Local Derivative Pattern

We aim to exploit both spatial and temporal domains in the analysis of video sequences. To this purpose, we considered the Local Derivative Pattern features (LDP), a feature extraction technique widely used in image processing and computer vision for tasks such as texture analysis, face recognition, and object detection. The LDP, a generalization of the widely used Local Binary Pattern (LBP), is a pointwise operator applied to 2D arrays of pixels that encodes diverse local spatial relationships.

As suggested in [4], we considered the second-order directional LDPs with direction α, where α ∈ {0°, 45°, 90°, 135°}. The second-order directional LDPs are an extension of the traditional Local Derivative Pattern (LDP) method that incorporates second-order derivative information and directional filtering for capturing local contrast and texture information in images. The second-order directional LDPs consider both gradient magnitude and gradient orientation in the analysis of image patches. The gradient magnitude measures the rate of change of intensity, while the gradient orientation indicates the direction of the intensity variation. These measures are calculated based on the second-order derivatives.

Given a 2D array of pixels, the (α) at the location(h, w) is an 8-bit vector as shown in Table , the (α) are extracted for every pixel of the 2D array and their -bin histogram is computed; this is replicated for the four different directions {0°, 45°, 90°, 135°}, and the histograms are concatenated, so the output feature vector length from 2D array will be .

To exploit both spatial and temporal domains in the analysis of video sequences [1], proposes to extend the computation of LDP histograms to 3D arrays. This is done by sequentially considering the three central 2D arrays along each dimension that intersect orthogonally as shown in Figure ‎3.3 three orthogonal planes and again concatenating the obtained histograms, yielding the so-called LDP-TOP features. The output feature vector length form LDP-TOP will be , where 4 are the directions and 3 are the orthogonal planes.

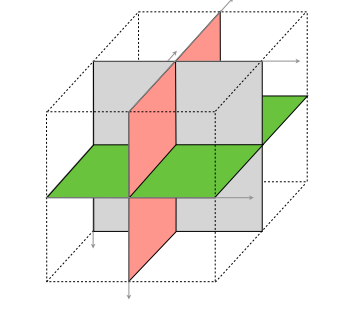


Figure ‎3.3 three orthogonal planes [1]

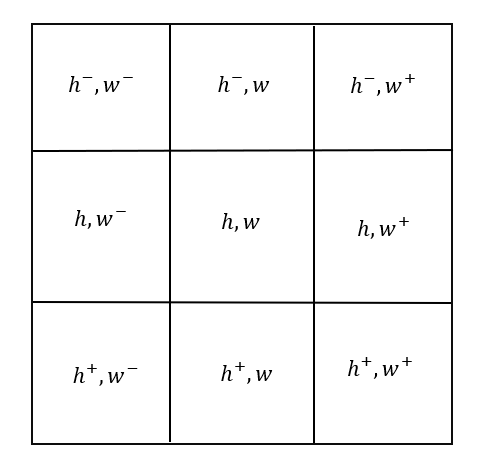


Figure ‎3.4 3 × 3 neighborhood

Algorithm:

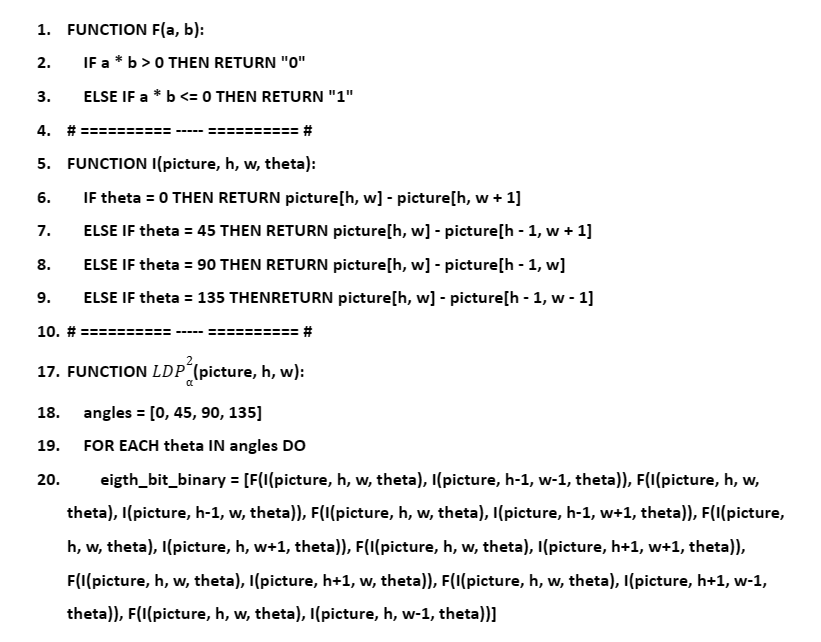
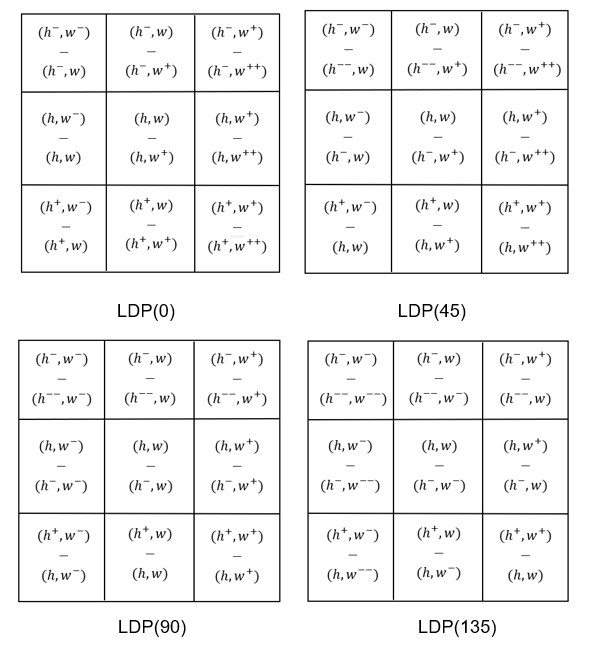


Table 3: LDP^2 (α) Algorithm



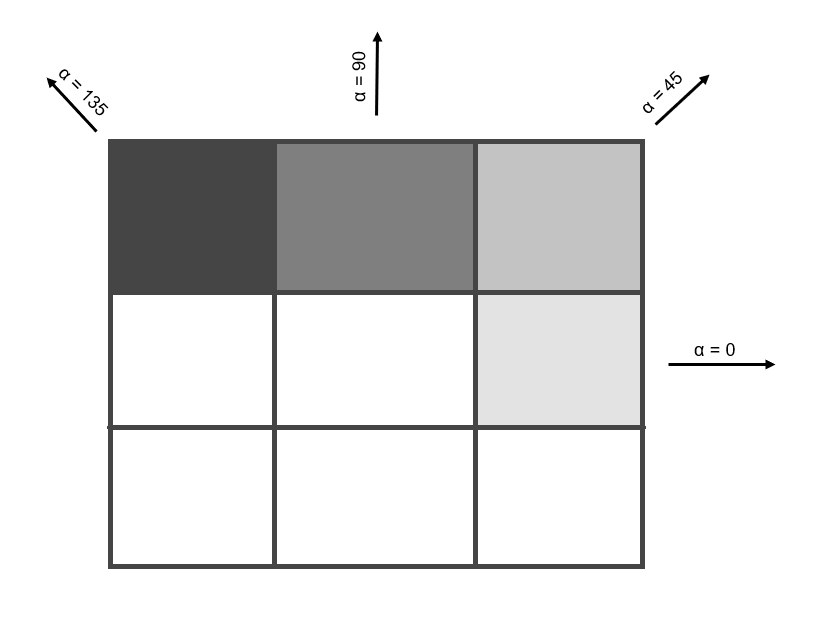


Figure ‎3.5 α Directions

Figure ‎3.6 The output of the I function in four directions.

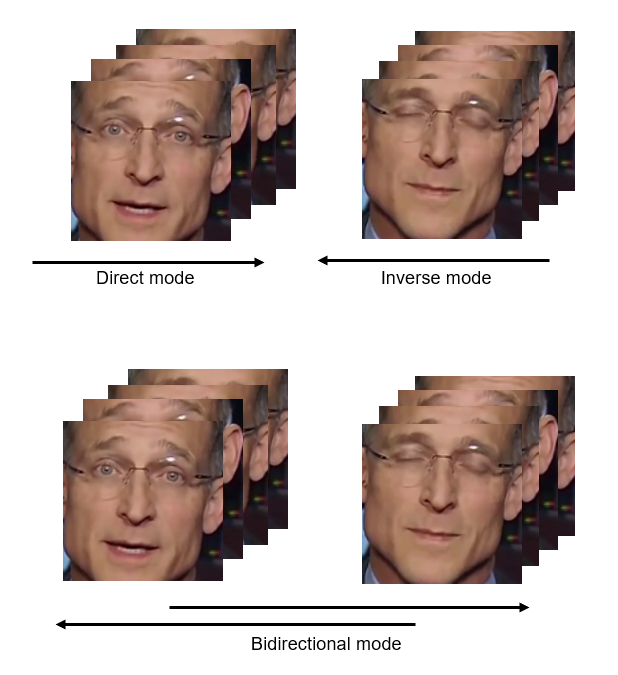
In order to explore potential peculiarities in the way the temporal information is captured by LDPs, paper proposed to run the feature extraction in three different temporal modes:

Figure ‎3.7 Temporal modes

* Direct mode (→): S is processed forward along the temporal direction.
* Inverse mode (←): S is processed backward along the temporal direction starting from the last frame and ending with the first frame.
* Bidirectional mode (↔): S is processed in both directions and histograms are concatenated, so the output feature vector length will be doubled.

#### 3.1.2.3 Training

After getting the feature vector from LDP-TOP, it's time for training. The feature vectors computed from each temporal sequence, so LDP-TOP may generate four or five feature vectors from a single video based on its length.

The training process involves a set of real and manipulated videos, that we indicate as TRr (labeled as 0) and TRm (labeled as 1), respectively and all of them are used as inputs for training the classifier. The dataset comes with a standard split of videos for training and testing. In order to enable a fair comparison with other recently proposed approaches, we also considered the same training and testing set, yielding the sets TRD with |TRD| = 860 and TSD with |TSD| = 140, where D ∈ {OR, DF, F2F, FSW, NT}. Different subsets will be combined according to the experimental scenario considered.

#### 3.1.2.3.1 Training Model

The suggested classifier in the paper is Support Vector Machines (SVM). Support Vector Machines (SVM) are often used in deepfake detection problems because of their ability to handle high-dimensional data and their effectiveness in separating classes. Here are a few reasons why SVMs are commonly applied in deepfake detection:

* Non-linear classification: SVMs can efficiently classify data that is not easily separable by a linear decision boundary. Deepfake detection involves complex and non-linear patterns, and SVMs can capture these patterns effectively.
* Robustness against overfitting: SVMs have a regularizing parameter that helps prevent overfitting, which is crucial when dealing with limited training data in deepfake detection.
* Ability to handle high-dimensional feature spaces: Deepfake detection often involves analyzing large amounts of image or video data. SVMs can handle high-dimensional feature spaces and are not adversely affected by the "curse of dimensionality" that can impact other algorithms.
* Margin-based learning: SVMs aim to find a decision boundary that maximizes the margin between classes, leading to improved generalization performance. This provides robustness against noise and can help distinguish between real and fake samples in deepfake detection.
* Support for both binary and multiclass classification: Deepfake detection can involve classifying samples into multiple categories. SVMs can be extended to handle multiclass classification problems using techniques such as one-vs-one or one-vs-rest.

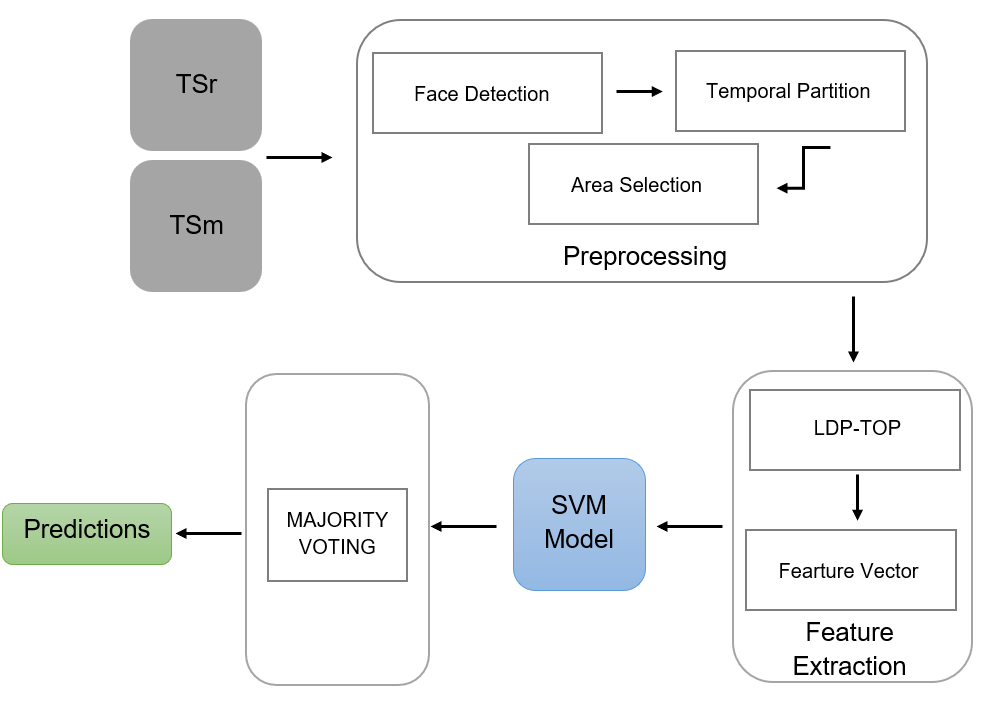
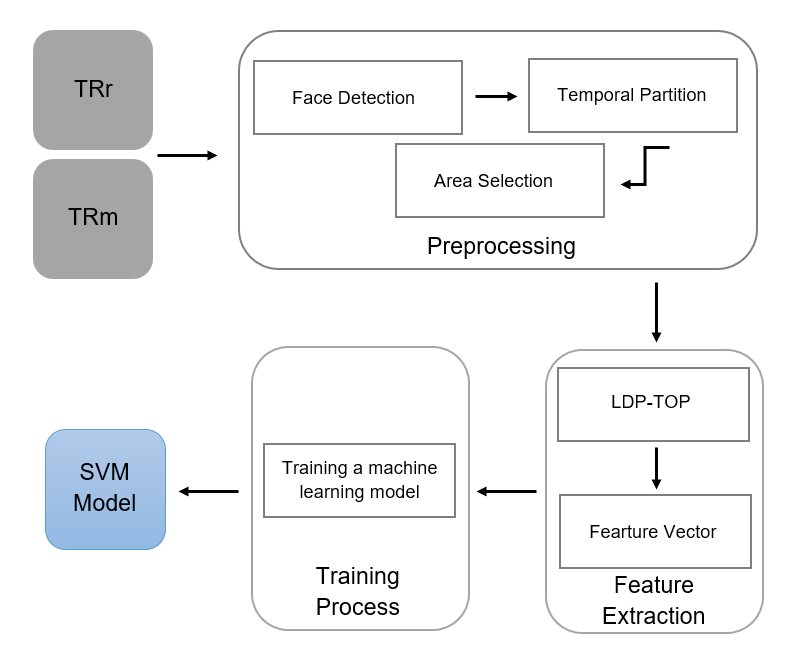


Figure ‎3.8 Training pipeline

Figure ‎3.9 Testing Pipeline

#### 3.1.2.3.2 Training Machines

Due to the enormous dataset and the large number of models necessary to cover the analysis in the paper as shown in Table 5 training time for each model, training all these models on a single machine was impractical. Fortunately, several cloud providers offer online model training services. For our project, we chose Azure as our cloud provider. Azure provides a specialized package for students known as the Azure for Students program. This program grants students’ access to a wide range of Azure services and resources. We utilized this program to train our models and obtain the necessary analysis. Training machines are found in

Table ‎3.2 Training Machines.

|  |  |  |
| --- | --- | --- |
| Name | Specifications | Cost |
| PC1 | 4 cores, 24GB RAM, 256GB storage | - |
| Standard\_D4\_v3 | 4 cores, 16GB RAM, 100GB storage | 0.19/hr$ |
| Standard\_E4ds\_v4 | 4 cores, 32GB RAM, 150GB storage | 0.29/hr$ |
| Standard\_E8s\_v3 | 8 cores, 64GB RAM, 128GB storage | 0.50/hr$ |
| Standard\_NC6 | 6 cores, 56GB RAM, 380GB storage | 0.90/hr$ |
| Standard\_E16s\_v3 | 16 cores, 128GB RAM, 256GB storage | 1.01/hr$ |

Table 4 Training Machines

#### 3.1.2.3.3 Training Time

We listed the training time of each model to demonstrate the significant amount of time taken to obtain the required analysis.

The training time of models varies depending on several factors, including the complexity of the model, the size of the dataset used for training and the computational resources available.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Kernal | Dataset Version | Training Data (videos) | Training Time in minutes |
| Kickoff | RBF | Cf23 | 500 | 5172 |
| Deepfakes | RBF | Cf23 | 1720 | 3.36 |
| Deepfakes | Linear | Cf23 | 1720 | 120.45 |
| Face2Face | RBF | Cf23 | 1720 | 100.23 |
| Face2Face | Linear | Cf23 | 1720 | 4156.1 |
| FaceSwap | RBF | Cf23 | 1720 | 4.13 |
| FaceSwap | Linear | Cf23 | 1720 | 201.06 |
| NeuralTextures | RBF | Cf23 | 1720 | 12.33 |
| NeuralTextures | Linear | Cf23 | 1720 | 6935.11 |
| Multi-Classifier | RBF | Cf23 | 4300 | 123.05 |
| Multi-Classifier | Linear | Cf23 | 4300 | 11213.51 |
| Deepfakes | RBF | Cf40 | 1720 | 100 |
| Deepfakes | Linear | Cf40 | 1720 | 10864 |
| Face2Face | RBF | Cf40 | 1720 | 180 |
| FaceSwap | RBF | Cf40 | 1720 | 7 |
| FaceSwap | Linear | Cf40 | 1720 | 7132 |

Table 5 training time for each model

#### 3.1.2.4 Experimental results

The next sections present the experimental tests conducted in order to validate the proposed method in practical scenarios.

##### 3.1.2.4.1 CF23 Experimental results

In this section, models trained on the version of the dataset subject to a light compression (H.264 with constant rate quantization parameter equal to 23). We have tested the feature representation and classification framework in several experimental scenarios, which are described in detail in the next subsections.

##### 3.1.2.4.1.1 Kickoff model

The main purpose of this model is to validate the feature representation and classification framework. This model is exceptional as it has been trained on 500 real videos and 500 manipulated videos (125 videos from each technique). Additionally, we used a list of hyperparameters (as shown in Table 6 Kickoff model hyperparameters) in this model to determine the parameters that yield the highest accuracy. We tested the performance of our model on an equal number of training data, consisting of 500 real videos and 500 manipulated videos (125 videos from each technique). The resulting accuracy is not bad but not good, as shown in Table 7 Kickoff model accuracy.

The kickoff model approves the following:

* We can rely on LDP-TOP as a feature representation.
* The best hyperparameters are (C=0.1, kernel=’rbf', gamma=0.1).
* Face2Face and NeuralTextures are more complex techniques than Deepfakes and FaceSwap.

|  |  |  |
| --- | --- | --- |
| **C** | **Kernal** | **Gamma** |
| **0.1** | Linear | 1 |
| 1 | **Rbf** | **0.1** |
| 5 | Poly | 0.01 |
| 10 | Sigmoid | 0.001 |
| 20 |  | |
| 30 |
| 40 |
| 50 |
| 60 |
| 70 |
| 100 |
| 200 |

Table 6 Kickoff model hyperparameters

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm Version | Real | Manipulated | Cross-Dataset |
| (F, →) | 85.54% | 73.68% | 79.87% |

Table 7 Kickoff model accuracy

**3.1.2.4.1.2 Single-technique scenario**

In this section, we evaluated the performance of our approach in distinguishing between original videos and videos that have undergone specific manipulation techniques. The primary objective was to showcase the capabilities of each classifier when tested with its corresponding dataset. As a result, we obtained an SVM classifier for each manipulation technique, namely CDF, CF2F, CFSW, and CNT. The videos in these sets were inputted into the training pipeline described in Figure ‎3.8 Training pipeline.

During this phase, we present the results obtained by utilizing the full-face facial area (F) and the forward temporal mode (→). Additionally, we trained the classifier using two different kernels, namely Linear and RBF. This resulted in a total of 8 classifiers, two for each manipulation technique, allowing us to observe their variations and interactions. Results are depicted as bar plots in Figure ‎3.10 Classification accuracy per manipulation technique in terms of accuracy and full numerical results are reported in Table 8 Classification accuracy on the single-manipulation scenario.

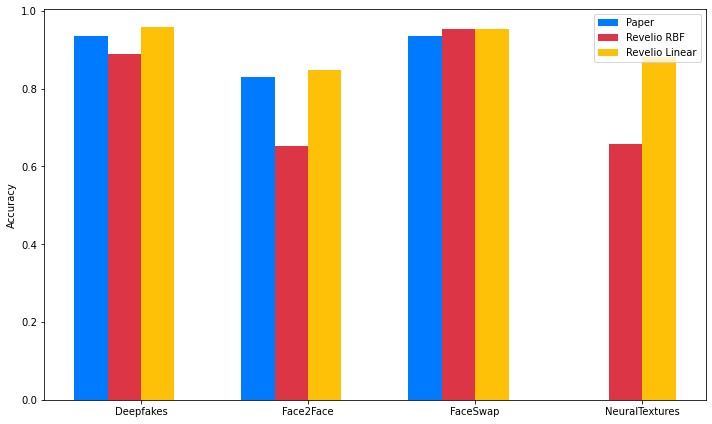


Figure ‎3.10 Classification accuracy per manipulation technique

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Deepfakes | Face2Face | FaceSwap | NeuralTextures | Cross-Dataset |
| Paper | 93.57% | 82.86% | 93.57% | Not Covered | 90.00% |
| Revelio RBF | 88.92% | 65.35% | 93.35% | 65.71% | 78.3325% |
| Revelio Linear | 95.71% | 84.64% | 95.35% | 88.21% | 90.97% |

Table 8 Classification accuracy on the single-manipulation scenario

As indicated in Table 8 Classification accuracy on the single-manipulation scenario, the accuracies of Face2Face and NeuralTextures are lower than those of Deepfakes and FaceSwap. This implies that Face2Face and NeuralTextures typically result in more challenging manipulations to detect.

Additionally, the accuracy of Revelio RBF is lower than that of both the paper and Revelio Linear, likely due to using the default value of gamma. I believe that performing a grid search to obtain the best hyperparameters would increase the accuracies of Revelio RBF. The primary reason we did not employ grid search is the extended training time required by the models. As indicated in Table 5 training time for each model, certain models require significant time for training on the given data, leaving insufficient time for grid search.

In summary, the best results in terms of accuracy are achieved with Revelio Linear, which demonstrates an average accuracy of 90.97% across datasets.

**3.1.2.4.1.3 Multiple-technique scenario**

We now turn our attention to the scenario where manipulation techniques are combined. Specifically, we consider the more realistic case where the test video can be either real or manipulated using any technique from the dataset.

**3.1.2.4.1.3.1 Combining binary classifiers.**

The first method to achieve multiple-technique classification involves combining the outcomes of classifiers trained on individual manipulation techniques. This approach also enables us to estimate the specific manipulation technique used in the event of a positive detection.

More specifically, we propose assigning each test video a label ∈ {0, 1} by combining the outputs of the binary classifiers CDF, CF2F, CFSW, and CNT. This results in four predicted labels: p(DF), p(F2F), p(FSW), and p(CNT). A video is classified as manipulated if one of the four detectors returns the label 1. Additionally, in the case of p = 1, the maximum value among the scores is selected as an indicator of the specific manipulation technique used to create the video.

The accuracy of multiple technique classification using binary classifiers can be observed in Table 9 The accuracy of multiple technique classification using binary . Additionally, we have reported the false positive rate, which represents the fraction of original videos erroneously classified as manipulated, and the false negative rate, reflecting the fraction of manipulated videos erroneously classified as original in the table.

|  |  |  |
| --- | --- | --- |
| False Positive Rate | False Negative Rate | Accuracy |
| 27.1% | 4.46% | 91% |

Table 9 The accuracy of multiple technique classification using binary classifiers.

**3.1.2.4.1.3.2 multi-class classifier**

Instead of combining binary classifiers to achieve multiple-technique classification, we will train a single classifier on the entire dataset, which includes real videos and videos with various manipulation techniques. This approach allows us to obtain a multi-class classifier capable of performing multiple-technique classification.

The paper states, 'We have observed that training a single binary classifier on the entire dataset leads to poor results. This can possibly be attributed to the linearity of the classifier used, which appears to be inadequate for accurately separating the two classes through a hyperplane in the feature space. Instead of emphasizing the requirement for a single classifier to achieve perfect separation, we propose combining the outcomes of classifiers trained in individual manipulation techniques.'

The results obtained from the multi-class classifier do not align with those presented in the paper. We achieved an accuracy of approximately 90% using the linear version of the multi-class classifier. Thus, indicating that the classifier is capable of identifying the hyperplane that effectively separates the different manipulation techniques. Results are depicted as bar plots in Figure ‎3.11 multi-class accuracy in terms of accuracy and full numerical results are reported in Table 10 multi-class accuracy.

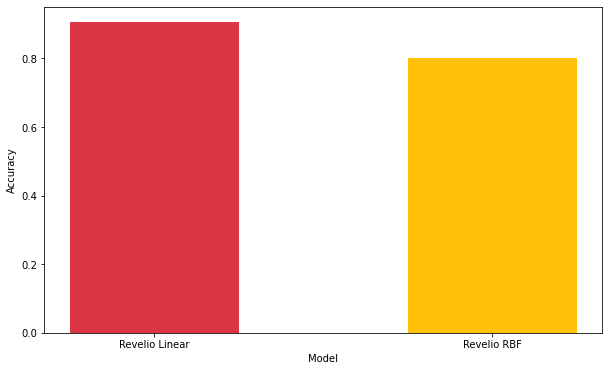


Figure ‎3.11 multi-class accuracy

|  |  |
| --- | --- |
| Model | Accuracy |
| Revelio Linear | 90.57% |
| Revelio RBF | 80.14% |

Table 10 multi-class accuracy

##### 3.1.2.4.2 CF40 Experimental results

In this section, models trained on the version of the dataset subject to a high compression with cf = 40. Compressing videos can potentially affect the performance of deepfake detection algorithms.

The artifacts and distortions caused by video compression may impact the performance of deepfake detection algorithms. Deepfake detection methods often rely on analyzing patterns, inconsistencies, or artifacts that are indicative of manipulation. If the compression process alters or removes these patterns or introduces additional artifacts, it can make the detection task more challenging or less accurate.

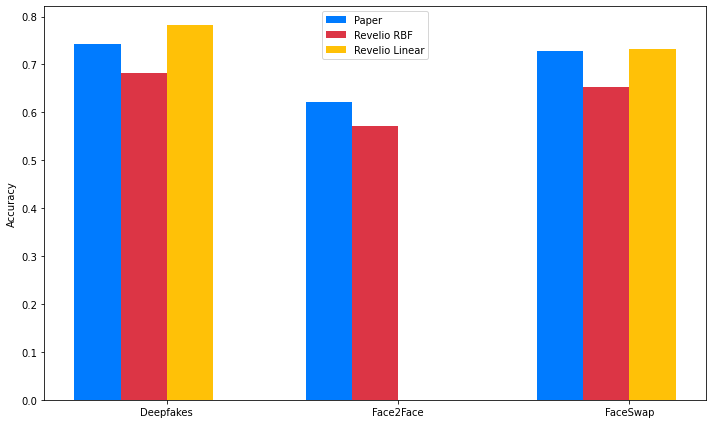
To get the analyses of single technique scenario, we trained the classifiers on dataset with cf = 40 using two different kernels, namely Linear and RBF. This resulted in 5 classifiers, two for each manipulation technique. Results are depicted as bar plots in Figure ‎3.12 Classification accuracy per manipulation technique in case of strong video compression in terms of accuracy and full numerical results are reported in Table 11 Classification accuracy on the single-manipulation scenario in case of strong video compression.

Figure ‎3.12 Classification accuracy per manipulation technique in case of strong video compression

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Deepfakes | Face2Face | FaceSwap | Cross-Dataset |
| Paper | 74,29% | 62,14% | 72,86% | 69,76% |
| Revelio RBF | 68.21% | 57.14% | 65.35% | 63.56% |
| Revelio Linear | 78.21% | Not Covered | 73.21% | 75.71% |

Table 11 Classification accuracy on the single-manipulation scenario in case of strong video compression

To get the analysis of multiple technique classification, we combined the output of binary classifiers that trained on dataset with cf = 40. The accuracy of multiple technique classification using binary classifiers can be observed in Table 12 The accuracy of multiple technique classification using binary classifiers in case of strong video .

|  |  |  |
| --- | --- | --- |
| False Positive Rate | False Negative Rate | Accuracy |
| 34.57% | 15.35% | 75.23% |

Table 12 The accuracy of multiple technique classification using binary classifiers in case of strong video compression.

### 3.1.3. Design Constraints

One of the primary constraints for deepfake detection models is video compression, which can potentially impact the performance of deepfake detection algorithms in the following ways:

* Loss of visual quality: Video compression algorithms aim to reduce the file size by removing or reducing redundant information in the video frames. As a result, the compressed video may exhibit lower quality, introducing compression artifacts and distortions. These artifacts can obscure or alter subtle manipulation indicators, making it more challenging for deepfake detection models to identify the presence of manipulation.
* Increased noise and artifacts: Compression techniques can introduce additional noise and video artifacts, such as blocking artifacts, banding, or color shifts. These artifacts can potentially interfere with the detection model's ability to accurately analyze and classify video frames, reducing the effectiveness of the detection process.

Figure ‎3.13 same frame with cf=23 & cf=40

As seen in section 3.1.2.4.2, when using a dataset with cf=40 instead of cf=23, the accuracy of the single technique scenario cross-dataset decreased from 90.97% to 75.71%, and the accuracy of the multiple technique classification decreased from 91% to 75.23%. Thus, the effect of video compression on system accuracy is evident.

The second constraint of deepfake detection is poor generalization across different manipulation techniques. Deepfake detection models need to be capable of detecting various types of deepfake manipulations, but they are usually only able to detect the manipulation techniques they were trained on. If we test the model with a technique that it was not trained on, I believe the model's accuracy will not be as good as its accuracy on trained techniques.

# System Testing and Verification

#### 4.1. Dynamic Texture Analysis Testing

Unit testing for a deepfake detection model plays a crucial role in ensuring its robustness and reliability. Deepfake detection models are designed to identify manipulated or synthesized media content, which requires a high degree of accuracy. Unit testing helps verify the functionality of individual components or units within the detection model.

During unit testing, different aspects of the model are thoroughly examined. This includes testing the integrity of the data preprocessing pipeline, evaluating the performance of feature extraction algorithms, and validating the effectiveness of the classification models. Each component is tested in isolation to ensure that it functions as expected and returns correct results.

#### 4.1.1. Feature Extraction Testing

During this stage, we tested the different components of feature extraction algorithm to ensure they produce the expected output.

Tested components:

* Function (F): one of the components of the feature extraction algorithm, used to analyze how patterns change in an image. We tested this component by inputting test cases and verifying if the output matches the expected results.
* Function (I): an important component of the feature extraction algorithm is used to measure the change in intensity between two adjacent pixels in specific directions (0, 45, 90, 135 degrees) to analyze how patterns change in an image. We tested the I function by inputting test cases and verifying if the output matches the expected results.
* Function (ldp\_pixel): component of the feature extraction algorithm is used to obtain the eight binary bits of a specific pixel. LDP\_pixel works by assigning binary values (0 or 1) to each neighboring pixel based on its intensity. By encoding these binary values in a clockwise or counterclockwise order, a binary pattern is generated for the center pixel. This process is repeated for every pixel in the image, resulting in a new image where each pixel holds its corresponding LDP value. We tested the ldp\_pixel function by inputting test images and verifying the output patterns.
* Function (LDP\_TOP): The main component of the feature extraction algorithm is used to calculate LDP on the orthogonal planes. It takes frames as input and outputs the feature vector, which is used to train the SVM model. We tested LDP\_TOP function by checking the length of feature vector = 3072.

#### 4.1.2. Model Testing

Model testing in deepfake detection is a critical phase to evaluate the performance and effectiveness of the developed detection models. Testing involves assessing the model's ability to accurately classify media content as either genuine or manipulated. This process helps validate the model's generalization capabilities and its performance on unseen data.

During model testing, a diverse set of test data is used to assess the model's robustness and resilience against various types of deepfake techniques. The test data can include manipulated videos or original videos that were not used during the model's training phase. This ensures that the model is evaluated on real-world scenarios and can detect new or unknown deepfake methods.

Different evaluation metrics are employed to measure the model's performance during testing, but we will show the accuracy of each model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Kernal | Dataset Version | Training Data (videos) | Accuracy |
| Kickoff | RBF | Cf23 | 500 | 79.87% |
| Deepfakes | RBF | Cf23 | 1720 | 88.92% |
| Deepfakes | Linear | Cf23 | 1720 | 95.71% |
| Face2Face | RBF | Cf23 | 1720 | 65.35% |
| Face2Face | Linear | Cf23 | 1720 | 84.64% |
| FaceSwap | RBF | Cf23 | 1720 | 93.35% |
| FaceSwap | Linear | Cf23 | 1720 | 95.35% |
| NeuralTextures | RBF | Cf23 | 1720 | 65.71% |
| NeuralTextures | Linear | Cf23 | 1720 | 88.21% |
| Multi-Classifier | RBF | Cf23 | 4300 | 80.14% |
| Multi-Classifier | Linear | Cf23 | 4300 | 90.57% |
| Deepfakes | RBF | Cf40 | 1720 | 68.21% |
| Deepfakes | Linear | Cf40 | 1720 | 78.21% |
| Face2Face | RBF | Cf40 | 1720 | 57.14% |
| FaceSwap | RBF | Cf40 | 1720 | 63.56% |
| FaceSwap | Linear | Cf40 | 1720 | 75.71% |

Table 13 Dynamic Texture Models accuracies

# Conclusions

Revelio is a face manipulation detection system used to detect various face manipulation methods. We have managed to produce a very strong system that is able to detect face manipulation with very high accuracy. Throughout this book we explained the complete process of building the system and how it was built, we conclude that the face manipulation detection is still not an easy problem due to many circumstances such as the realistic fake videos produced that make it too hard to detect, and the ability to produce realistic fake videos will keep improving, so we have to keep up with the advancing technologies which is a big challenge and the problem will still be under the research.

# References

|  |  |
| --- | --- |
| [1] | M. C. P. a. G. B. Bonomi, "Dynamic texture analysis for detecting fake faces in video sequences.," *Journal of Visual Communication and Image Representation 79.,* 2021. |
| [2] | R. M. K. a. J. K. Durall, "Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions.," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.,* 2020. |
| [3] | F. C. R. a. M. S. Matern, "Exploiting visual artifacts to expose deepfakes and face manipulations.," *IEEE Winter Applications of Computer Vision Workshops (WACVW).,* 2019. |
| [4] | Baochang Zhang, Yongsheng Gao, Sanqiang Zhao, and Jianzhuang Liu., "Local derivative pattern versus local binary pattern: Face recognition," *IEEE,* 2010. |