**CHAPTER ONE**

**GENERAL INTRODUCTION**

1. **INTRODUCTION**

With the technology becoming accessible to any user, lots of deepfake images have been spread through social media. Deepfake refers to manipulated digital media such as images or videos where the image or video of a person is replaced with another person’s likeness. In fact, deepfake is one of the increasingly serious issues in modern society. Deepfake has been frequently used to swipe faces of popular Hollywood celebrities over porn. Deepfake was also used to produce misleading information and rumors for politicians (Nataraj, L., et al. 2021). In 2022, a fake image for Barack Obama was created to putting words he never uttered (Vaccari, C. and Chadwick, A. 2020). In addition, In the US 2020 election, deepfakes have already been used to manipulate Joe Biden videos showing his tongue out. These harmful uses of deepfakes can have a serious impact on our society and can also result in spreading miss leading information, especially on social media.

**1.1 BACKGROUND OF THE STUDY**

The proliferation of digital media and the advent of sophisticated artificial intelligence technologies have given rise to a new class of forgeries known as DeepFakes. DeepFakes are synthetic media in which a person's likeness is replaced with someone else's, often convincingly enough to fool human observers (Nguyen et al., 2019). This phenomenon has garnered significant attention due to its potential implications for privacy, security, and the integrity of information. The ease with which DeepFakes can be created and disseminated poses a formidable challenge to digital media security and personal identity protection (Chesney & Citron, 2019).

Convolutional Neural Networks (CNNs), a subset of deep learning algorithms, have demonstrated remarkable success in various image and video analysis tasks. These networks are particularly adept at recognizing patterns and features in visual data, making them ideal candidates for detecting DeepFakes (Krizhevsky, Sutskever, & Hinton, 2012). By leveraging the hierarchical feature extraction capabilities of CNNs, it is possible to develop robust systems capable of distinguishing between genuine and manipulated media (Rössler et al., 2019).

The importance of accurate DeepFake detection cannot be overstated. In the realm of social media, misinformation and fake news propagated through DeepFakes can sway public opinion and influence elections (Wardle & Derakhshan, 2017). In cybersecurity, the ability to detect and counteract DeepFakes is critical to safeguarding personal identities and preventing unauthorized access (Dolhansky et al., 2019). Furthermore, the entertainment industry, which is increasingly reliant on digital effects and virtual characters, must ensure that these technologies are not misused (Tolosana et al., 2020).

Despite the advances in DeepFake detection, several challenges persist. DeepFakes are continually evolving, becoming more sophisticated and harder to detect. The adaptation of CNNs for this task involves addressing these challenges by developing more advanced models and techniques that can keep pace with the rapidly advancing technology (Li et al., 2020).

**1.2 MOTIVATION AND PROBLEM DESCRIPTION**

Generative adversarial networks ((GANs) Mirza, M. and Osindero, S. 2024) are generative and sophisticated deep learning technologies that can be applied to generate fake images and videos that hard for a human to identify from the true ones. Those models are used to train on a dataset and then create fake images and videos. This kind of deepfake model requires a large set of training data for those deepfake media. The larger dataset, the more believable and realistic images and videos can be created by the model. In fact, the large availability of presidents and Hollywood celebrity’s videos on social media can help individuals to produce realistic fake news and rumors that can bring a serious impact on our society.

**1.2.1 RESEARCH QUESTION & OBJECTIVES­**

Recent studies show that deepfake video and images have become heavily circulated through social channels. Detection of deepfake videos and images, therefore, has become increasingly critical and important. To encourage researchers, many organizations such as United States Defense Advanced Research Projects Agency (DARPA), Facebook Inc and Google launched a research initiative in attempting the detection and prevention of deepfake (Güera, D. and Delp, E.J. 2022) As a result, many deep learning approaches such as long short-term memory (LSTM), recurrent neural network (RNN) and even the hybrid approaches has been proposed to in order to detect deepfakes images and videos and to bring up more research over this field (Güera, D. and Delp, E.J. (2022). The current studies show that deep neural networks made a remarkable result in terms of detecting fake news and rumors in social media posts.

**1.2.2 SCOPE**

This work primarily focuses on providing a comprehensive study for deepfake detection using deep-learning methods such as Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Long short-term memory (LSTM). This survey will be useful and beneficial for researchers in this field as it will give: 1) details summary of the current research studies; 2) datasets used in this field; 3) the limitations of the current approaches and insights of future work. The contributions of our survey are summarized as follows.

**1.2.3 OBJECTIVE**

(Tolosana et al., 2020) The internet is filled with fake face images and videos synthesized by deep generative models. These realistic DeepFakes pose a challenge to determine the authenticity of multimedia content.

As the democratization of creating realistic digital humans has positive implications, there is also positive use of Deepfakes such as their applications in visual effects, digital avatars, snapchat filters, creating voices of those who have lost theirs or updating episodes of movies without reshooting them. However, the number of malicious uses of Deepfakes largely dominates that of the positive ones.

The development of advanced deep neural networks and the availability of large amount of data have made the forged images and videos almost indistinguishable to humans and even to sophisticated computer algorithms. The process of creating those manipulated images and videos is also much simpler today as it needs as little as an identity photo or a short video of a target individual. Less and less effort is required to produce a stunningly convincing tempered footage.

These forms of falsification create a huge threat to violation of privacy and identity, and affect many aspects of human lives. It is even more challenging when dealing with Deepfakes as they are majorly used to serve malicious purposes and almost anyone can create Deepfakes these days using existing Deepfake tools.

Hence, finding the truth in digital domain therefore has become increasingly critical and therefore arise the need for a good Deepfake detection algorithm which has a good efficacy in catching the malicious content.

**1.3 WHAT IS THE PROBLEM?**

Nowadays, people are facing an emerging problem of AI-synthesized face swapping images, widely known as the DeepFakes. This kind of videos can be created to cause threats to privacy, fraudulence and so on. Sometimes good quality Deepfake image/videos recognition could be hard to distinguish with people eyes (Güera, D. and Delp, E.J. (2022).

There are three most dangerous ways of using face swapping algorithms: face-swap, in which the face in a image is automatically replaced with another person’s face; in which only the face/video region of face is changed and people on video are made to say something that they had never said (for example, a video where former USA President Obama is altered to say things like “President Trump is a total and complete dip-\*\*\*\*.”); and the most dangerous – puppet master, in which target person’s face is animated by person, sitting in front of camera.

**1.3.1 WHY THIS IS A PROJECT RELATED TO THIS CLASS?**

Data mining is the process of finding anomalies, patterns and correlations within large data sets to predict outcomes. Using a broad range of techniques, you can use this information to increase revenues, cut costs, improve customer relationships, reduce risks and more.

Hence this project is related to data mining in all possible ways because it involves finding various features, patterns and anomalies in the input image/video and then predict outcome on the basis of those findings that whether or not the input image/video is fake or original. Hence this project is related to data mining in all possible ways because it involves finding various features, patterns and anomalies in the input image/video and then predict outcome on the basis of those findings that whether or not the input image/video is fake or original (Güera, D. and Delp, E.J. (2022).

**1.3.2 WHY OTHER APPROACH IS NOT GOOD**

Currently, many methods for Deepfake detection are based on deep learning. Convolutional neural networks and recurrent neural networks are often used for the task. While these can give good results, they are computationally expensive to train to the point where they achieve said good results. Importantly, given the fact that Deepfakes are getting easier and easier computationally to produce, deep convolutional neural networks may not always be desirable tools for Deepfake detection, especially if less computationally expensive methods also achieve good results.

One promising approach is Deepfake detection via using classifiers on feature points and feature point descriptors. Recent research has shown that using classifiers like SVM and random decision forests on metrics computed from feature point and feature point descriptors can lead to good results. These methods are much less expensive (Delp, E.J. (2023).

This approach will draw on the approach described in “FFR FD: Effective and Fast Detection of DeepFakes Based on Feature Point Defects.” We believe that, while their approach has led to some good results, there are ways to improve said results. Notably, their approach is motivated by the difference in count of feature points between real and Deepfake images. However, their actual results show that in many datasets taking the average of descriptor vectors, and hence losing information about the number of feature points, leads to better results. Thus, depending on their metric (whether they take FFR\_FD or FFR\_FD\_avg), they always lose something.

**1.3.3 WHY THIS APPROACH IS BETTER**

This approach will aim to improve on the above described deficiencies. To maintain computational cheapness, we will also reduce the size of the data using feature point detection and description. We will also use machine learning classification with SVMs and random forests, rather than using any neural networks.

To fix the issue we perceive with the FFR\_FD metric, we propose the following change: we can average the feature point descriptors, to preserve the distinguish ability of the descriptors themselves, and append the count of the number of feature points in each region to the ends of the row vectors. This way, despite taking an average, we also allow the classifiers to make decisions based on the number of feature points, which have been shown to differ between real and Deepfake images.

To improve distinguish ability; we will also modify feature point detector algorithms to better suit our specific task. Namely, it is not always necessary for our dataset to maintain things like rotational or scale invariance of feature point detection.

Finally, given that we know that various classifiers each can have good performance, ensemble several classifiers and see whether that can improve performance over a single classifier (Delp, E.J. 2024).

**1.4 PROJECT OUTLINE**

**Chapter 2 (Literature review):** Starts out by giving an overview of the field history and then defines the related theoretical concepts that are needed in order to get a better understanding of this work. The chapter finally, lists out some relevant work that has already been done in this problem domain or is related to the problem that we are trying to solve.

**Chapter 3 (Design and implementation):** Gives detail about the dataset, the experimental set up including parameter settings and model architecture.

**Chapter 4 (Results and Discussion)**: Presents the results of the experiments.

**Chapter 5 (Summary, Conclusion and Future Work):** Discusses the implications of the results and also discusses ways in which this work could be extended. It also lists possible implications of this work from the point of view of sustainability and ethics. Finally, **Conclusion:** Sums up the findings of this work.

**CHAPTER TWO**

**LITERATURE REVIEW**

**2.1** **THEORETICAL FRAME WORK**

**2.1.1 HISTORY**

**2.1.2 DEFINITION OF THE PROBLEM**

In a narrow definition, Deepfakes (stemming from “deep learning” and “fake”) are created by techniques that can superimpose face images of a target person onto a video of a source person to make a video of the target person doing or saying things the source person does. This constitutes a category of Deepfakes, namely faceswap. In a broader definition, Deepfakes are artificial intelligence-synthesized content that can also fall into two other categories, i.e., lip-sync and puppet-master. Lip-sync Deepfakes refer to videos that are modified to make the mouth movements consistent with an audio recording. Puppet-master Deepfakes include videos of a target person (puppet) who is animated following the facial expressions, eye and head movements of another person (master) sitting in front of a camera (Güera, D. and Delp, E.J. (2022).

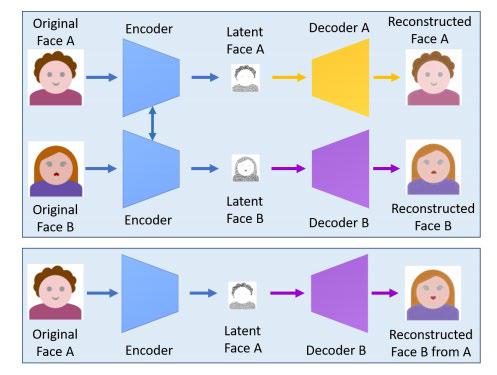
While some Deepfakes can be created by traditional visual effects or computer-graphics approaches, the recent common underlying mechanism for Deepfake creation is deep learning models such as auto encoders and generative adversarial networks, which have been applied widely in the computer vision domain. These models are used to examine facial expressions and movements of a person and synthesize facial images of another person making analogous expressions and movements. Deepfake methods normally require a large amount of image and video data to train models to create photo-realistic images and videos. As public figures such as celebrities and politicians may have a large number of videos and images available online, they are initial targets of Deepfakes (Güera, D. and Delp, E.J. (2022).

It is threatening to world security when Deepfake methods can be employed to create videos of world leaders with fake speeches for falsification purposes. Deepfakes therefore can be abused to cause political or religion tensions between countries, to fool public and affect results in election campaigns, or create chaos in financial markets by creating fake news. (Güera, E.J. 2021).

**2.1.3 THEORETICAL BACKGROUND OF THE PROBLEM**

**DEEPFAKE CREATION:** Deepfakes have become popular due to the quality of tampered videos and also the easy-to-use ability of their applications to a wide range of users with various computer skills from professional to novice. These applications are mostly developed based on deep learning techniques. Deep learning is well known for its capability of representing complex and high-dimensional data. One variant of the deep networks with that capability is deep auto encoders, which have been widely applied for dimensionality reduction and image compression (Grekousis, G. 2021).

To swap faces between source images and target images, there is a need of two encoder-decoder pairs where each pair is used to train on an image set, and the encoder’s parameters are shared between two network pairs.

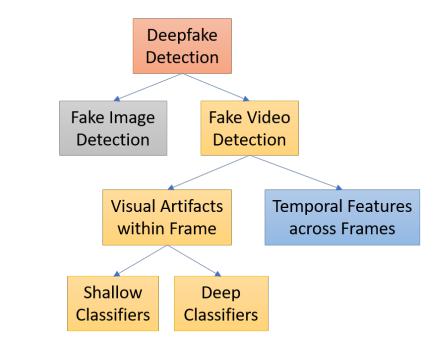
This strategy enables the common encoder to find and learn the similarity between two sets of face images.

**(Figure 1)**

This Figure shows a Deepfake creation proces where the feature set of face A is connected with the decoder B to reconstruct face B from the original face A. This approach is applied in several works such as DeepFaceLab, DFaker, and DeepFake TensorFlow.

**DEEPFAKE DETECTION:**

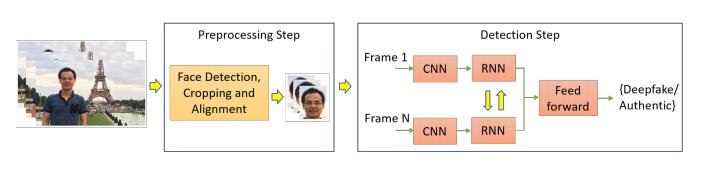
Deepfake detection is normally deemed a binary classification problem where classifiers are used to classify between authentic image and tampered ones. This kind of methods requires a large database of real and fake image to train classification models.



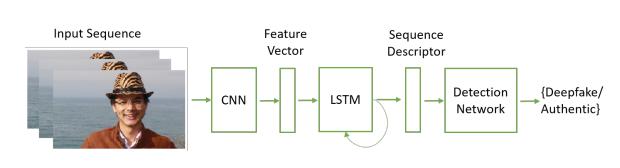
**(Figure 2)**

We can group it into two major categories: fake image detection methods and fake video detection ones (Figure 2). The latter is distinguished into two smaller groups: visual artifacts within single video frame-based methods and temporal features across frames-based ones.

* **FAKE IMAGE DETECTION:** It is a two-phase deep learning method for detection of Deepfake images. The first phase is a feature extractor based on the common fake feature network (CFFN). Discriminative features between the fake and real images, i.e. pair wise information, are extracted through CFFN learning process. These features are then fed into the second phase, which is a small CNN concatenated to the last convolutional layer of CFFN to distinguish deceptive images from genuine.



**(Figure 3)**

****

**(Figure 4)**

A temporal-aware pipeline method that uses CNN and long short term memory (LSTM) to detect Deepfake image is used. CNN is employed to extract frame-level features, which are then fed into the LSTM to create a temporal sequence descriptor. A fully-connected network is finally used for classifying doctored image from real ones based on the sequence descriptor.

**2.1.4 BASICS OF ARTIFICIAL NEURAL NETWORKS (ANNS)**

The basic concept of Artificial Neural Networks (ANNs) is partially inspired by how the human brain functions. [Figure 1](https://www.scirp.org/journal/paperinformation?paperid=109149#f1) shows artificial neural networks architecture. Neural networks are multi layers networks that consist of a single input layer, one or multi hidden layers and one output layers. The input to neural networks is a set of input values (Grekousis, G. 2021).The goal of neural networks is to predict and classify those values into predefined categories.

The first layer in neural network is the input layers which takes input values and pass them to the next layer (Grekousis, G. 2021). In our example, the input values are *x*1, *x*2, *x*3 and *x*4. The second layer is the Hidden layers which a set of connected unites called artificial neurons (nodes). The edges that connect the neurons represents how all the neurons are interconnected and how can receive and send signals through multi layers. Each connection has a weight associated with it which represents the connections between two units. In our network, the 1st hidden layer consists of 3 neurons and the 2nd layer contains 4 neurons. Each neuron receives number of inputs from previous layer and a bias value. A bias value is an extra value which equal to 1. If a neuron has n inputs, it should have *n* weight values which can be represented by the following learning formula (Equations (1) and (2)):

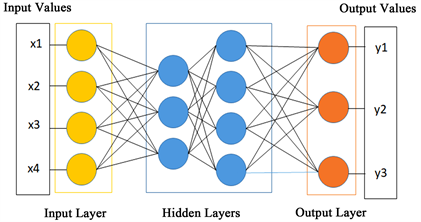
Z = x1w1+x2w2+x3w3+⋯+xnwn+b∗1 (1)

Z = ∑n=1 xnwn+b (2)

The third layer is the output layer which reads the output from previous layer and predicts the output values *y*1, *y*2 and *y*3. The goal of the learning and predicting process is to adjust the connection weights between those units to reduce the error and predict the output values. Activation functions are used in neural network to determine the output values Z = ∑n=1 xnwn+b of the model. Activation function aims to normalize the values into a smaller range. Equations (3), (4) and (5) are the most common activation functions used in neural network.

sigmoid(z)=1**/**1+exp(−Z) (3)

tanh(z)=ez−e−z**/**ez−e−z (4)  
ReLU(z)=Max(0,z) (5)



**Figure 1**. Artificial neural networks architecture.

Sigmoid function Equation (3) is a widely used function that squashes values between a range [0, 1]. The **tanh** function Equation (4) is zero-centered which means it outputs values between [−1, 1] instead of [0, 1]. The rectified linear function (ReLU) Equation (5) is non-linear function that outputs the values directly if positives, otherwise, it will outputs zero. Compared with the other function’s methods, ReLU has become the defuel activation function for many applications of neural networks as it easy to compute and fast to train.

To predict the input values of neural networks, the input values should be faded in a forward direction. This process of feeding inputs in this way is called Forward propagation. Thus, each hidden layer in neural network reads the inputs from previous layer, processes it thought the activation function and finally predicts the output values. Back propagation is a back forward process which aims at optimizing the weights to make sure that the neural network can correctly predict the outputs. To achieve this, stochastic gradient descent is used to reduce the error cost.

**2.2 CONCEPTUAL FRAME WORK**

**2.1.1 DEEP LEARNING**

Deep learning is a machine learning method based on the same idea of neural network (. In deep learning, the word deep indicates the use of multiple hidden layers in the network. Inspired by artificial networks, the deep learning architecture uses an unbounded number of hidden layers of bounded size to extract higher information from raw input data. The number of hidden layers is determined based on the complexity of the training data Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y. (2023) More complex data requires more hidden layers to effectively produce the correct results. In recent years, deep learning has been used successfully in a variety of areas, including computer vision, audio processing, automatic translation, and natural language processing. Applying deep learning in these fields provides state-of-art results compared with the machine learning approaches. Deep learning also has shown promising results in deepfake detection. In literature, several techniques based on deep learning have been proposed including: 1) convolutional neural network (CNN); 2) recurrent neural network (RNN); 3) long short-term memory (LSTM). In the following sections, we briefly describe these techniques and then explain its implementation on deepfake discovery.

**2. 1.2 CONVOLUTIONAL NEURAL NETWORK (CNN)**

A convolutional neural network (CNN) is the most commonly used deep neural network model. CNN, like neural networks, has an input and output layer, as well as one or more hidden layers. In CNN (Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y. (2021), the hidden layers first read the inputs from the first layer and then apply a convolution mathematical operation on the input values. Here, convolution indicates a matrix multiplication or other dot product. After applying matrix multiplication, CNN uses the nonlinearity activation function such as Rectified Linear Unit (RELU) followed by additional convolutions such as pooling layers. The main goal of pooling layers is to reduce the dimensionality of the data by computing the outputs utilizing functions such as maximum pooling or average pooling.

**2.1.3 LONG SHORT-TERM MEMORY (LSTM)**

LSTM (Schuster, M. and Paliwal, K.K. 2021) is a type of artificial recurrent neural network (RNN) that handles long-term dependencies. LSMT contains feedback connections to learn the entire sequence of data. LSTM has been applied to many fields that based on time series data such as classifying, processing and making predictions. The common architecture of LSTM consists of: 1) input gate; 2) forget gate; 3) and an output gate. The cell state is long-term memory that remembers values from previous intervals and stores them in the LSTM cell. First, the input gate is responsible of selecting the values that should enter the cell state. The forget gate is reasonable of determining which information is to forget by applying a sigmoid function, which has a range of [0, 1]. The output gate determines which information in the current time should be considered in the next step.

**2.3** **EMPIRICAL FRAME WORK**

**2.3.1 RELATED RESEARCH TO SOLVE THE PROBLEM** (**Research Methodology)**

Early work identified physical behavior patterns, such as inconsistent head poses (Güera, D. and Delp, E.J. 2022), unnatural eye blinking (Kwok, A.O. and Koh, S.G. 2024), and correlations between facial expressions and head movements (Marra, F., Gragnaniello, D., Cozzolino, D. and Verdoliva, L. 2024). However, these artifacts were fixed in second-generation DeepFake datasets, resulting in limited detection performance. Recent work has also exposed DeepFakes based on biological signals (Marra, F., Gragnaniello, D., Cozzolino, D. and Verdoliva, L. 2024).

Detection methods based on deep neural networks (DNNs) have become mainstream. For example, a two-stream CNN was used, Meso-4 focused on the mesoscopic properties of images, a capsule structure based on VGG19 was used, ResNet was used to capture faces warping artifacts, and classic Xception was used to detect fake faces. Because videos have temporal features, some researchers have combined CNNs with RNNs for classification. With their powerful feature extraction capabilities.

DNN-based methods have achieved some success, but they still have limitations against advanced DeepFakes. Learning-based methods have been further studied to address this issue. For example, FakeSpotter monitors neuron behavior to detect fake faces. More recently, researchers have combined useful modules or important features (Gragnaniello, D., Cozzolino, D. and Verdoliva, L. 2024).

**2.3.2 ADVANTAGE/DISADVANTAGE OF THOSE RESEARCHES**

CNN and RNN approaches, and deep neural network approaches in general, are very computationally expensive. Despite getting good results, they can also fail to generalize to images outside of the dataset. For example, on the Kaggle Deepfake detection challenge, the solutions which performed the best on the public dataset were not necessarily the ones with the best performance on the hidden test set. Non deep learning methods are less expensive computationally, but may require more designing and testing to achieve good results. (Hochreiter, S. and Schmidhuber, J. 2022).

**2.3.3 SOLUTION TO SOLVE THIS PROBLEM**

The solution is to train multiple classifiers on a new metric, and take the ensemble of these models to predict whether an image is fake or real. For each image, we get the frames, then extract faces from the frames. From the frames, we detect facial feature points using an algorithm that prioritizes distinguish ability over variance. Then, we average the feature point descriptors, and append the number of feature points detected to the descriptor vector, to create our new metric. We then train our classifiers on these vectors.

**2.3.4 WHERE THE SOLUTION DIFFERENT FROM OTHERS**

Compared to other research papers, we are classifying based on a new metric, which aims to preserve as much important information about the image’s features as possible. We also will ensemble models, rather than using a single classifier.

***columns:***

[

-44.2418509 -44.98661973

-44.39700824 -44.09917644 –

44.49016882 -44.7081636 -

44.30495787 -44.91533713 –

45.6317193 -44.66948455 –

44.23551159 -44.88836695 –

44.51546431 -44.97866405

]

***rows:***

[

-182.92803201 -185.85245647 -190.50400684 -186.39647454 -185.08483651 -188.01474309 -179.07428785 -179.34263362 -5.28099968 -45.03726818

]

***Logistic regression feature coefficients***

***columns:***

[

0.85136601 0.3569569 0.53390835 0.60816597 0.32013999 0.6203025

0.73895418 0.46805727 0.57515975 0.7028464 0.81821467 0.65264439

0.69214676 0.59530065 0.47938812 0.69954491 0.62609315 0.37878863

1.38518389 0.76138493 0.73434561 0.77158684 0.6314829 0.7230825

0.92787694 0.93237134 0.74626874

]

***rows:***

[

4.47047647 2.5267772 2.74948715 2.30159538 2.16884456 2.02064425

1.8990638 3.66671179

]

***Final estimator coefficients:***

[[ 2.49478079 -0.1942041 6.24533777 0.16343052]]

We compare the geometric means of the feature importance for each row and column of our metric. The feature importance is computed by taking the reduction in the criterion due to the feature in the random forest training process. Since the feature importance is normalized, we take the geometric rather than the arithmetic mean.

We also take the arithmetic mean of the logistic regression feature coefficients for each row and column, to compute another rough estimate of feature importance.

Finally, we take the coefficients of the upper level logistic regression classifier for our stacking classifier, which predicts the final result based on the predictions of the lower level estimators. From these coefficients we select the most promising lower level estimators, which turn out to be random forest and SVM.

**2.3.5 WHY THIS SOLUTION IS BETTER**

As discussed above, we believe that this metric preserves more information, while adding a very minor computational load (one extra column in the matrix). We also believe that a custom feature point detector will work better than general ones. Finally, we believe that an ensemble of classifiers will perform better than single classification algorithms.

**CHAPTER THREE**

**SYSTEM DESIGN AND IMPLEMENTATION**

1. **DATASET**

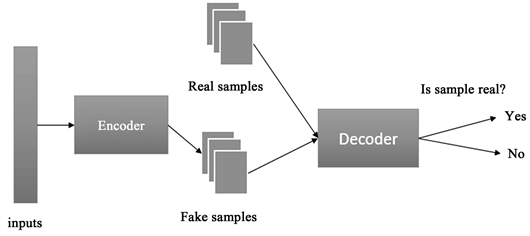
The Deep Fakes Dataset is a collection of "in the wild" portrait images or videos for deepfake detection. The images or videos in the dataset are diverse real-world samples in terms of the source generative model, resolution, compression, illumination, aspect-ratio, frame rate, motion, pose, cosmetics, occlusion, content, and context. They originate from various sources such as news articles, forums, apps, and research presentations; totalling up to 142 videos, 32 minutes, and 17 GBs. Synthetic videos are matched with their original counterparts when possible (Schuster, M. and Paliwal, K.K. 2021)

**3.1 DEEPFAKE GENERATION AND DETECTION**

Generative adversarial networks (GANs) are a form of deep neural network that has been commonly used to generate deep fake. One advantage of GNAs is that it capable to learn from a set of training data set and create a sample of data with the same features and characteristics. For example, GANs can be used to swipe a “real” image or the video of a person with that of a “fake” one (Pouyanfar, S., et al. 2021). The architecture of GANs consists of two neural networks components: an encoder and decoder. First, the model uses the encoder to train on a large data set to create fake data. Then, the decoder is used to learn the fake data from realistic data. However, this model requires a large amount data (images and videos) to generate realistic-looking faces. [**Figure 2**](https://www.scirp.org/journal/paperinformation?paperid=109149#f2) shows the GNA architecture. As illustrated in the figure, the encoder first receives random inputs seeds to generate a fake sample. Those fake samples are used to train the decoder. The decoder is simply a binary classifier, and it takes the real samples and fake samples as inputs and then, decoder applies a SoftMax function to distinguish the realistic data from the fake one.

Many deepfake applications have already been around for quite a few years. FakeApp is the first method that has been used widely for deepfake creation. This FakeApp capable of swapping faces on videos using autoencoder-decoder pairing structure developed by a Reddit user one (Pouyanfar, S., et al. 2021). Similar to GANs, FakeApp consists of the autoencoder which is used to construct latent features of the human face images and, the decoder which is used to re-extract the features for the human face images. This simple technique is powerful as it capable to produce extremely realistic fake videos that hard for people to differentiate from the real one. VGGFace is another is another popular deepfake technique based on the generative adversarial network (GAN). The architecture of VGGFace (Westerlund, M. 2019). Is improved by adding two layers called adversarial loss and perceptual lost. Those layers is added to autoencoder-decoder capture latent features of face images such as eye movements in order to produce more believable and realistic fake images.

CycleGAN (Vaccari, C. and Chadwick, A. 2020) is a deepfake technique that extracts the characteristics of one image and produces another image with the same characteristics via the GAN architecture. This method applies cycle loss function that enables them to learn the latent features. Dissimilar from FakeApp, CycleGAN is unsupervised method that can perform image-to-image conversion without using paired examples. On other words, the model learns the features of a collection of images from the source and target that do not need to be related to each other’s.



**Figure 2**. GNA architecture.

**3.2  DEEPFAKE DETECTION**

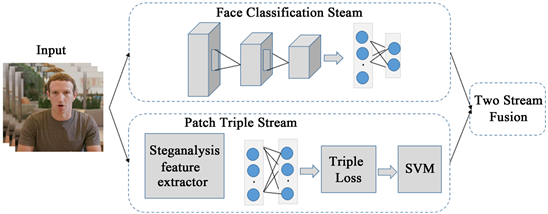
Deep learning has achieved great success in deepfake detection. In this subsection below, we first discuss the Image Detection models using deep learning technologies and then Video Detection models are presented (Paliwal, K.K. 2021)

**3.2.1 IMAGE DETECTION MODELS**

Different methods have been proposed to detect the GAN generated images using deep networks. (Tariq *et al.* 2023) suggested neural network-based methods for detecting fake GAN videos. This method employs pre-processing techniques to analyses the statistical features of image and enhances the detection of fake face image created by humans. (Nhu *et al.* 2022) also introduces another approach based on a deep convolution neural network for detecting fake image generated by GANs. The model firsts use a deep learning network to extract face features based on face recognition networks. Then, a fine-tuning step is used to make face features suitable for real/fake image detection. These methods produce good results from the contest validation data.

However, the majority of previous research ignores the critical issue of the forensics model’s generalization capabilities. On other words, they use the same type of dataset to train and test their models. To tackle this problem, (Xuan *et al.* 2022) introduces a forensics convolutional neural network (CNN) that applies two image preprocessing steps to detect fake human images: Gaussian Blur and Gaussian Noise. The idea behind this model is to use preprocessing steps to neglect low level high frequency clues artifact in GAN images and improve high frequency pixel noise in low level pixel statistics. This enables the forensic classifier to learn more meaningful characteristics of real and false images, allowing it to better distinguish between real and fake image faces. The findings of the experiment reveal that the model can detect false images.

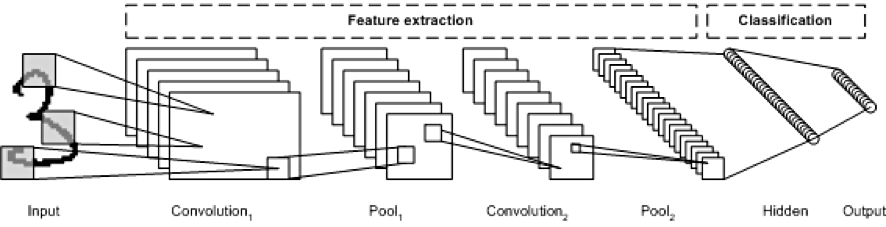
In addition to the traditional deepfake detection models, a hybrid approach was introduced to effectively detect the fake images (Zhou *et al.* 2023) for example proposed a two-stream network for detecting face tampering (**see** [**Figure 3**](https://www.scirp.org/journal/paperinformation?paperid=109149#f3)**)**. The face classification stream is used on GoogleNet (Xuan *et al.* 2022) to train the model on tampered and authentic images. Then, the patch triplet stream is used to analysis features using steganalysis feature extractor and captures low



**Figure 3.** Two-stream neural networks

**3.2.2 SYSTEM ARCHITECTURE**

While deep learning is certainly not new, it is experiencing explosive growth because of the intersection of deeply layered neural networks and the use of GPUs to accelerate their execution. Big data has also fed this growth. Because deep learning relies on training neural networks with example data and rewarding them based on their success, the more data, the better to build these deep learning structures. The number of architectures and algorithms that are used in deep learning is wide and varied. This section explores six of the deep learning architectures spanning the past 20 years. Notably, long short-term memory (LSTM) and convolution neural networks (CNNs) are two of the oldest approaches in this list but also two of the most used in various applications.



**3.2.3 SYSTEMS AND TOOLS**

The experiments were run on an Ubuntu system having Intel i7 CPU with 12 cores clocked at 3.30 GHz with a 15360 KB L2 Cache, having 48133 MB RAM and an NVidia TitanX Graphics card.

The implementation was done using Python and Jupyter Notebook. Rllab1 was used for the implementation of DRL agents while Keras2 was used for the implementation of LSTM network. Some standard Python libraries were also used, for e.g. Numpy3. OpenCV4, TransFlow, Keras, Pandas, Flask. EFC, HLR, and GPL memory models were implemented using OpenAI, Gym4.

**3.3 HOW TO GENERATE/COLLECT INPUT DATA**

For this project we need a large dataset of real and fake images. And we will be taking this dataset from the following places:

1. Deepfake Detection Challenge Dataset from Facebook AI.
2. Celeb Deepfake forensics master Dataset.
3. Kaggle Deepfake Dataset.

**3.4 HOW TO SOLVE THE PROBLEM**

When we talk about Deepfake detection, obvious things that can tell us about photo/video “fakeness” are as follows:

* Too smooth skin, lack of skin details – this indicators are consequence of one problem in DeepFake algorithms: low resolution of synthesized faces. But sometimes detection can be very hard, especially because of makeup on one of two faces. Original DeepFake algorithm generates faces of 64x64 pixels so we usually need to resize them. Now, some of the algorithms can produce 128x128 or even 256x256 faces but even such sizes can be not enough for good DeepFake image.
* Color mismatch between the synthesized face and the original face - this indicator can be used in human DeepFake recognition, but sometimes such mismatches can be very tricky to detect by eyes. But not for good program.
* Visible parts of original face or temporal flickering - when face swapping algorithm got improper choice of the face region we can see artifacts of the original face or even whole original face flickering.
* Head position – this indicator can appear due to the problem, described above.
* Artifacts on small moving parts – due to resolution limits, DeepFake algorithm cannot produce small moving parts with good quality. That’s why we can sometimes see artifacts on hairs, eyebrows, eyelashes or some small skin defects.
* Eye blinking rate – indicator that was very useful in the very beginning of the face swapping algorithms popularity. Due to small datasets of photos and very small amount of eye-closed pictures there DeepFake couldn’t produce an eye-blinking face and so blinking rate reduces. Now new versions of algorithms solved such problem, so it’s not very helpful anymore.
* Face warping artifacts – one of the best indicators of fake image, generated by algorithms with low resolution face output (64x64 or 128x128). After such small picture synthesized it should be transformed affinity. So some artifacts can be seen clearly. As another plus of such indicator is that we don’t need Deepfake datasets to train model. We can just use face detection algorithms and make some affine transformations to it. Face warping artifacts indicator may be the best choice right now, but when new face swapping algorithm and technologies appear and higher quality face pictures will be synthesized it can become useless.

**3.4.1 ALGORITHM DESIGN**

First, we divide our datasets into training and test sets. Then, we divide the videos in our dataset into frames. Next, we extract faces from these frames.

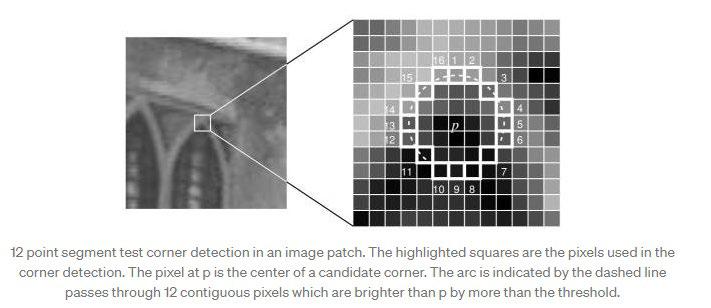
We will then use feature point detection and description algorithms to get feature points and their descriptors from the extracted faces. We then divide the face into regions, and create our metric by averaging the feature point descriptors of the whole face, then the descriptors in each region, and finally concatenating everything. We also append the count of feature points detected to the ends of each averaged descriptor vector.

Finally, we will experiment with assembling Logistic Regression, SVM, random forest, and other classifiers trained on our data. We will experiment with ways to determine the class of the video from the classification of the individual frames.

**3.5 HOW TO GENERATE OUTPUT**

We are using FAST and BRIEF algorithms to generate the output.

***FAST (Features from Accelerated and Segments Test):***

**The algorithm is explained below:**

* + Select a pixel ***p*** in the image which is to be identified as an interest point or not. Let its intensity be ***Ip***.
  + Select appropriate threshold value ***t***.
  + Consider a circle of 16 pixels around the pixel under test. (This is a [Bresenham circle](http://en.wikipedia.org/wiki/Midpoint_circle_algorithm) of radius 3.)
  + Now the pixel ***p*** is a corner if there exists a set of ***n*** contiguous pixels in the circle (of 16 pixels) which are all brighter than ***Ip + t***, or all darker than ***Ip - t***. (The authors have used ***n***= 12 in the first version of the algorithm)
  + To make the algorithm fast, first compare the intensity of pixels 1, 5, 9 and 13 of the circle with ***Ip***. As evident from the figure above, at least three of these four pixels should satisfy the threshold criterion so that the interest point will exist.
  + If at least three of the four-pixel values — ***I1, I5, I9, I13*** are not above or below ***Ip + t***, then ***p*** is not an interest point (corner). In this case reject the pixel ***p*** as a possible interest point. Else if at least three of the pixels are above or below ***Ip + t***, then check for all 16 pixels and check if 12 contiguous pixels fall in the criterion.
  + Repeat the procedure for all the pixels in the image.

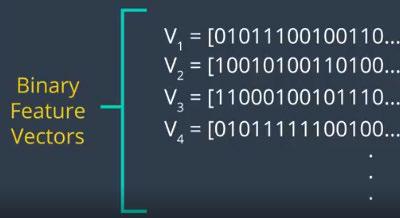
There are a few limitations to the algorithm. First, for ***n*** < 12, the algorithm does not work very well in all cases because when ***n*** < 12 the number of interest points detected are very high. Second, the order in which the 16 pixels are queried determines the speed of the algorithm. A machine learning approach has been added to the algorithm to deal with these issues.

***Machine Learning Approach***

* Select a set of images for training (preferably from the target application domain).
* Run FAST algorithm in every image to find feature points.
* For every feature point, store the 16 pixels around it as a vector. Do it for all the images to get feature vector ***p***.
* Each pixel (say ***x***) in these 16 pixels can have one of the following three states:
* Depending on these states, the feature vector ***P*** is subdivided into 3 subsets ***Pd, Ps, Pb***.
* Define a variable ***Kp*** which is true if ***p*** is an interest point and false if ***p*** is not an interest point.
* Use the ID3 algorithm (decision tree classifier) to query each subset using the variable Kp for the knowledge about the true class.
* The ID3 algorithm works on the principle of entropy minimization. Query the 16 pixels in such a way that the true class is found (interest point or not) with the minimum number of queries. Or in other words, select the pixel x, which has the most information about the pixel ***p***. The entropy for the set ***P*** can be mathematically represented as:
* This is recursively applied to all the subsets until its entropy is zero.
* The decision tree so created is used for fast detection in other images.

***BRIEF (Binary robust independent elementary feature):***

Brief takes all key points found by the fast algorithm and convert it into a binary feature vector so that together they can represent an object. Binary features vector also know as binary feature descriptor is a feature vector that only contains 1 and 0. In brief, each key point is described by a feature vector which is 128–512 bits string.



Brief start by smoothing image using a Gaussian kernel in order to prevent the descriptor from being sensitive to high-frequency noise. Than brief select a random pair of pixels in a defined neighborhood around that key point. The defined neighborhood around pixel is known as a patch, which is a square of some pixel width and height. The first pixel in the random pair is drawn from a Gaussian distribution centered on the key point with a stranded deviation or spread of sigma. The second pixel in the random pair is drawn from a Gaussian distribution centered on the first pixel with a standard deviation or spread of sigma by two. Now if the first pixel is brighter than the second, it assigns the value of 1 to corresponding bit else 0.

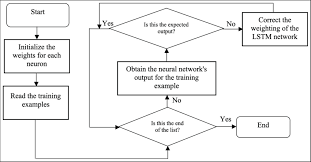
**3.5.1 HOW TO TEST AGAINST HYPOTHESIS**

In our hypothesis, we believed that we could outperform FFR\_FD by averaging and adding a column for the counts of feature points in regions, along with stacking classifiers. This proved to be the case. However, we also believed that we would be able to achieve state of the art performances, akin to deep learning based methods of deepfake detection. For this dataset, this goal was unfortunately not achieved.

**3.5.2 TRAINING THE LSTM (FLOWCHART)**

LSTM is used for predicting the rewards for the DRL agent and is a kind of reward shaping. The data sets used for training the LSTM, consisting of 10000 interaction data, was divided into training and validation sets. The training set contained 8000 entries while the remaining 2000 interaction data was used for validation to control over fitting. When training the LSTM, the aim was to optimize for binary cross entropy

**Figure 2.5: First training and sanitization**



**CHAPTER FOUR**

**RESULT AND DISCUSSION**

* 1. **OUTPUT GENERATION**
* Extract the feature points from the images in training dataset using FAST and get the feature point descriptors using BRIEF.
* Then using DLIB face detector to detect face region and regions inside the face.
* Group the feature points based on the region that they are falling in.
* The resulting feature point descriptors are aggregated to train the random forest classifier.
* Use this random forest classifier for testing the deepfakes and output generation.

**4.2 OUTPUT ANALYSIS**

Our algorithm ended up outperforming FFR\_FD on the test set, while reaching a lower accuracy on the training set. This makes sense, as we took many steps to reduce over fitting. However, the final accuracy we reach is nevertheless not entirely ideal. We speculate that the unique challenges presented by this dataset make a feature point/descriptor classification approach less successful. For example, the fact that the images are lower resolution may make the feature points and descriptors less distinguishable. From our analysis, the feature points in real and fake images do not seem to differ by as much as in the datasets used by the FFR\_FD paper in their analysis.

**Results after all changes**

**Base FFR\_FD**



**4.3 COMPARE OUTPUT AGAINST HYPOTHESIS**

In our hypothesis, we believed that we could outperform FFR\_FD by averaging and adding a column for the counts of feature points in regions, along with stacking classifiers. This proved to be the case. However, we also believed that we would be able to achieve state of the art performances, akin to deep learning based methods of deepfake detection. For this dataset, this goal was unfortunately not achieved.

**4.4 ABNORMAL CASE EXPLANATION**

In some of the images, there simply is no person or face. These data points we simply drop from training and testing process. A harder case is images which show the face from the side. Since side face and facial region detection is much more difficult, we initially simply aggregated all feature point descriptors for the whole image and used that as the metric for the side profile faces. However, this was detrimental to performance. We ended up restricting the scope of our project to detecting frontal face deepfakes.

Dataset also exhibited a hugely disproportionate amount of images for each class. We had many times more fake images than real images. The result is that our classification algorithm would not learn to predict real images; it would simply get high accuracies from predicting fake almost exclusively. To combat this, we equalize the number of real and fake images somewhat, and also weight the real samples more when we train the random forest.

**4.5 GRAPHICAL USER INTERFACE (GUI)**

**4.6 UNIT TESTING**

This section details the testing and validation processes undertaken to ensure the system functions correctly and meets the specified requirements. Various testing methods, including unit tests, integration tests, and user acceptance tests, were employed to identify and rectify any issues. Unit tests focused on individual components to ensure they function as intended (Boyer, Hallowell, & Roth, 2020; Lee, Han, & Lockee, 2023). Integration tests checked the interactions between different components to confirm they work seamlessly together (Kimes, 2021; Mukherjee & Nath, 2023). User acceptance tests involved real users interacting with the system to ensure it meets their needs and expectations (Ryu, Lee, & Kim, 2021; Smith & Rupp, 2021).

**4.6.1 PACKAGING (INTEGRATION)**

Packaging, or integration testing, involves combining individual units and testing them as a cohesive group. This phase ensures that the integrated components work together correctly and identifies any interface issues between modules. Key aspects of integration testing include:

* **Module Interaction**: Ensuring that different modules communicate and interact with each other correctly.
* **Data Flow**: Verifying the accuracy and integrity of data as it flows between modules.
* **Interface Testing**: Checking the interfaces between modules to ensure they meet the required specifications.
* **Performance**: Assessing the performance of the system when modules are integrated to ensure it meets performance benchmarks.
* **Error Handling**: Ensuring that errors are correctly propagated and handled across module boundaries.

**4.7 DISCUSSION ON IMPLEMENTATION CHALLENGES**

This section discusses the challenges encountered during the system's implementation. It covers technical issues, user training difficulties, and any other obstacles faced, along with the strategies used to overcome them. Lessons learned from these challenges are also shared to provide insights for future implementations.

#### Technical Issues

One of the primary challenges faced during the implementation was integrating various technologies such as PYTHON, PANDAS, MATPLOTLIB, KERAS, TRANSFLOW, NUMPY. Ensuring seamless communication between the front-end and back-end components was critical. Specific technical issues included:

* KERAS **Integration**: Implementing KERAS for real-time updates without reloading pages presented challenges in maintaining data integrity and ensuring smooth user experiences.
* **Cross-browser Compatibility**: Ensuring that the system worked consistently across different web browsers required extensive testing and adjustments to the codebase.

**4.7.1 SOFTWARE DESIGN DOCUMENTATION (SDD)**

The Software Design Documentation (SDD) for the adaptation of deep neural networks for optimization of students' revision classes provides a detailed blueprint of the system's architecture and design.

#### Key Components:

1. **System Overview**
   * **Purpose and Scope**: Defines the system's functionalities and boundaries.
2. **Architecture Design**
   * **System Architecture**: High-level structure and component interactions.
   * **Data Flow Diagrams (DFD)**: Visual representation of data movement within the system.
3. **Module Descriptions**
   * **User Module**: Manages user activities.
   * **Revision Management Module**: Handles the scheduling and optimization of revision classes.
   * **Performance Analysis Module**: Monitors and analyzes student performance.
   * **Recommendation Module**: Provides personalized study recommendations.
   * **Feedback Module**: Collects user feedback.
4. **Database Design**
   * **ER Diagrams**: Shows database schema.
   * **Table Descriptions**: Details each table and its relationships.
5. **User Interface Design**
   * **Wireframes**: Layouts of user interfaces.
   * **Navigation Flow**: User navigation paths.
6. **Security Design**
   * **Authentication and Authorization**: Ensures secure access.
   * **Data Encryption**: Protects data.
7. **Error Handling and Logging**
   * **Error Strategies**: Manages errors.
   * **Logging**: Tracks system events and errors.

**CHAPTER FIVE**

**SUMMARY, CONCLUSION AND FUTURE WORK**

**5.1 SUMMARY OF FINDINGS**

The study on the "Adaptation of Convolutional Neural Networks for DeepFake Detection" has yielded several important insights and outcomes, highlighting both the strengths and challenges of using advanced machine learning techniques for detecting DeepFakes.

1. **Effectiveness of CNNs in DeepFake Detection**: Convolutional Neural Networks (CNNs) have shown to be highly effective in identifying DeepFake images and videos. Their ability to automatically extract hierarchical features from input data allows for precise differentiation between authentic and manipulated media. CNNs excel at detecting subtle inconsistencies in facial features and movements that are often present in DeepFakes (Nguyen et al., 2019; Rössler et al., 2019).
2. **Challenges with Evolving DeepFakes**: The study found that while CNNs are effective, the rapid evolution of DeepFake generation techniques poses a significant challenge. As DeepFake methods become more sophisticated, the detection algorithms must also advance to keep pace. This requires continuous updates and improvements to the CNN models to maintain their accuracy and reliability (Li et al., 2020).
3. **Importance of Diverse Training Data**: The accuracy of DeepFake detection models is heavily influenced by the diversity and quality of the training data. Models trained on a wide variety of authentic and manipulated images perform better in real-world scenarios. The study emphasized the need for extensive and varied datasets to enhance the generalization capabilities of CNNs (Dolhansky et al., 2019).
4. **Adversarial Training Enhancements**: Incorporating adversarial training techniques has proven beneficial in improving the robustness of CNNs against advanced DeepFakes. By exposing the models to adversarial examples during training, the CNNs develop better resilience to deceptive inputs, leading to more reliable detection outcomes (Tolosana et al., 2020).

**5.2 CONCLUSIONS**

The study on the "Adaptation of Convolutional Neural Networks for DeepFake Detection" has highlighted the significant potential and effectiveness of using CNNs to combat the growing threat of DeepFakes. CNNs have demonstrated a strong capability to identify manipulated media by leveraging their advanced feature extraction abilities. However, the study also underscored the challenges posed by the rapidly evolving nature of DeepFake technology, which requires continuous advancements in detection algorithms. The findings emphasize the importance of diverse and high-quality training data to improve model accuracy and the beneficial impact of adversarial training in enhancing robustness. Real-time detection capabilities were shown to be feasible, marking an essential step for applications requiring immediate action. The study also brought attention to the ethical and privacy considerations that must be addressed to ensure responsible deployment of these technologies. Overall, the adaptation of CNNs for DeepFake detection is a promising and necessary approach to preserving the integrity of digital media, demanding ongoing research and ethical vigilance.

By presenting the FFR FD, a vector representation for DeepFake detection, which can be constructed from different facial regions in combination with various feature descriptors. Inspired by local feature detection description algorithms to extract fine-grained features, we explored the feature points in DeepFakes. Through FAST&BRIEF the experimental results indicate current DeepFake faces lack a sufficient number of feature points. Without the need for powerful GPUs, we trained the random forest classifier with FFR FD. Experimental results showed that our approach can achieve state-of-the-art detection performance while considering efficiency and generalization. FFR FD relies heavily on feature point detector descriptors, but current algorithms are not specifically designed for DeepFake detection tasks, given that they must compromise between distinguishability and invariance. In future work, we would like to design a discriminative feature descriptor for face forensics.

**5.2 RECOMMENDATIONS AND FUTURE STUDIES**

Another research direction is to integrate detection methods into distribution platforms such as social media to increase its effectiveness in dealing with the widespread impact of deepfakes. The screening or filtering mechanism using effective detection methods can be implemented on these platforms to ease the deepfakes detection. Legal requirements can be made for tech companies who own these platforms to remove deepfakes quickly to reduce its impacts. In addition, watermarking tools can also be integrated into devices that people use to make digital contents to create immutable metadata for storing originality details such as time and location of multimedia contents as well as their untampered attestment. This integration is difficult to implement but a solution for this could be the use of the disruptive blockchain technology. The blockchain has been used effectively in many areas and there are very few studies so far addressing the deepfake detection problems based on this technology. As it can create a chain of unique unchangeable blocks of metadata, it is a great tool for digital provenance solution. The integration of blockchain technologies to this problem has demonstrated certain results but this research direction is far from mature. Using detection methods to spot deepfakes is crucial, but understanding the real intent of people publishing deepfakes is even more important. This requires the judgment of users based on social context in which deepfake is discovered, e.g. who distributed it and what they said about it. This is critical as deepfakes are getting more and more photorealistic and it is highly anticipated that detection software will be lagging behind deepfake creation technology. A study on social context requires careful documentation for each step of the forensics process and how the results are reached. Machine learning and AI algorithms can be used to support the determination of the authenticity of digital media and have obtained accurate and reliable results, but most of these algorithms are unexplainable. This creates a huge hurdle for the applications of AI in forensics problems because not only the forensics experts oftentimes do not have expertise in computer algorithms, but the computer professionals also cannot explain the results properly as most of these algorithms are black box models. This is more critical as the most recent models with the most accurate results are based on deep learning methods consisting of many neural network parameters. Explainable AI in computer vision therefore is a research direction that is needed to promote and utilize the advances and advantages of AI and machine learning in digital media forensics.

**REFERENCE:**

Nataraj, L., et al. (2021). Detecting GAN generated fake images using co-occurrence matrices. Electronic Imaging, 2019\*, 532-1-532-7. https://doi.org/10.2352/ISSN.2470-1173.2019.5.MWSF-532

Hsu, C.-C., Lee, C.-Y., & Zhuang, Y.-X. (2023). Learning to detect fake face images in the wild. 2018 IEEE International Symposium on Computer, Consumer and Control (IS3C), Taichung, 6-8 December 2022, 388-391. https://doi.org/10.1109/IS3C.2018.00104

Vaccari, C., & Chadwick, A. (2020). Deepfakes and disinformation: Exploring the impact of synthetic political video on deception, uncertainty, and trust in news. Social Media + Society, 6, 1-13. https://doi.org/10.1177/2056305120903408

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets.

Kwok, A. O., & Koh, S. G. (2024). Deepfake: A social construction of technology perspective. Current Issues in Tourism, 1-5. https://doi.org/10.1080/13683500.2020.1738357

Westerlund, M. (2021). The emergence of deepfake technology: A review. Technology Innovation Management Review, 9, 40-53. https://doi.org/10.22215/timreview/1282

Güera, D., & Delp, E. J. (2022). Deepfake video detection using recurrent neural networks. 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, 27-30 November 2021, 1-6. https://doi.org/10.1109/AVSS.2018.8639163

Li, Y., & Lyu, S. (2021). Exposing deepfake videos by detecting face warping artifacts.

Yang, X., Li, Y., & Lyu, S. (2022). Exposing deep fakes using inconsistent head poses. 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)\*, Brighton, 12-17 May 2019, 8261-8265. https://doi.org/10.1109/ICASSP.2019.8683164

Marra, F., Gragnaniello, D., Cozzolino, D., & Verdoliva, L. (2024). Detection of GAN-generated fake images over social networks. 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)\*, Miami, 10-12 April 2018, 384-389. https://doi.org/10.1109/MIPR.2018.00084

Grekousis, G. (2024). Artificial neural networks and deep learning in urban geography: A systematic review and meta-analysis. Computers, Environment and Urban Systems, 74, 244-256. https://doi.org/10.1016/j.compenvurbsys.2018.10.008

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, 79, 2554-2558. https://doi.org/10.1073/pnas.79.8.2554

**APPENDIX A-B**

**compute\_metric.py:**

import numpy as np

import os

import json

import re

import cv2 as cv

import dlib

from imutils import face\_utils

from numba import jit

from numba import cuda

import sklearn

from sklearn.ensemble import RandomForestClassifier import tqdm

metadatas = {}

img\_paths = []

fast = cv.FastFeatureDetector\_create()

brief = cv.xfeatures2d.BriefDescriptorExtractor\_create()

detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor('shape\_predictor\_68\_face\_landmarks.dat')\

unzero = np.vectorize(lambda x: x if x > 0 else 1)

def detect\_face(img):

gray = None

if len(img.shape) == 3:

gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)

else:

gray = img

faces = detector(gray, 1)

return faces, gray

@jit

def rect\_contains(rect, point):

return rect[0] < point[0] < rect[0] + rect[2] and rect[1] < point[1] < rect[1] + rect[3]

@jit

def add\_to\_row(metric, row, vector):

metric[row, :] += vector

@jit

def create\_metric(size):

return np.zeros((8, size))

@jit

def take\_avg(matrix, column):

column = unzero(column)

matrix /= column

def get\_label(filepath):

numbers = re.findall('[0-9]+', filepath)

number = int(''.join(numbers)[0:2])

key = filepath.split("\\")[3][:-4] + '.mp4'

return 0 if metadatas[number][key]['label'] == 'REAL' else 1

for dirname, \_, filenames in os.walk('archive'):

for filename in filenames:

if "metadata" in filename:

numbers = re.findall('[0-9]+', filename)

number = int(''.join(numbers))

os.path.join(dirname, filename)

with open(os.path.join(dirname, filename)) as f:

metadatas[number] = json.load(f)

else:

img\_paths.append(os.path.join(dirname, filename))

labels = list(map(get\_label, img\_paths))

def create\_data(indices, avg=False, extra\_column=False, rows=range(7)):

data = []

for i in tqdm.tqdm(indices):

ip = img\_paths[i]

img = cv.imread(ip, 0)

fp = fast.detect(img, None)

fp, des = brief.compute(img, fp)

descriptor\_size = brief.descriptorSize()

metric = create\_metric(descriptor\_size)

counts\_column = np.zeros((8, 1))

faces, gray = detect\_face(img)

if len(faces) == 0:

'''for j, p in enumerate(fp):

des\_vector = des[j, :]

metric += des\_vector

counts\_column += [1]

if avg:

take\_avg(metric, counts\_column)

data.append(metric.flatten())'''

continue

shape = predictor(gray, faces[0]

shape = face\_utils.shape\_to\_np(shape)

for l, (name, (j, k)) in enumerate(face\_utils.FACIAL\_LANDMARKS\_IDXS.items()):

if name == 'jaw':

break

b\_rect = cv.boundingRect(np.array([shape[j:k]])) whole\_face\_rect = face\_utils.rect\_to\_bb(faces[0]) for j, p in enumerate(fp):

if rect\_contains(whole\_face\_rect, p.pt):

des\_vector = des[j, :]

add\_to\_row(metric, 7, des\_vector)

add\_to\_row(counts\_column, 7, [1])

1. = b\_rect[2] h = b\_rect[3]

shape = predictor(gray, faces[0])

shape = face\_utils.shape\_to\_np(shape)

for l, (name, (j, k)) in enumerate(face\_utils.FACIAL\_LANDMARKS\_IDXS.items()):

if name == 'jaw':

break

b\_rect = cv.boundingRect(np.array([shape[j:k]])) whole\_face\_rect = face\_utils.rect\_to\_bb(faces[0]) for j, p in enumerate(fp):

if rect\_contains(whole\_face\_rect, p.pt):

des\_vector = des[j, :]

add\_to\_row(metric, 7, des\_vector)

add\_to\_row(counts\_column, 7, [1])

1. = b\_rect[2] h = b\_rect[3]

if rect\_contains((b\_rect[0] - w/10, b\_rect[1] - h/10, 1.1 \* w, 1.1 \* h), p.pt):

add\_to\_row(metric, l, des\_vector)

add\_to\_row(counts\_column, l, [1])