2024

Leveraging Machine Learning & Deep Learning Methodologies to Detect Deepfakes

Aniruddha Tiwari
Minnesota State University, Mankato

Leveraging Machine Learning & Deep Learning Methodologies to Detect Deepfakes

by

Aniruddha Tiwari

Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of

Science

In

Data Science

Minnesota State University,

Mankato, Minnesota

April 2024

04/02/2024

Leveraging Deep Learning Methodologies to Detect Deepfakes

Aniruddha Tiwari

This thesis has been examined and approved by the following members of the thesis committee.

Dr. Rushit Dave
Advisor

Dr. Rajeev Bukralia
Graduate Coordinator / Committee Member

Dr. Mansi Bhavsar,
Committee Member

Acknowledgments

Primarily, I wish to extend my deepest thanks to Minnesota State University, Mankato, a well-known institution whose influence transcends mere education. Its profound impact on my intellectual and personal growth cannot be overstated. Through its diverse academic offerings and nurturing environment, the university has not only equipped me with essential knowledge and skills but has also played a pivotal role in shaping the very essence of who I am today.

My Mentor and Professor Dr. Rushit Dave deserves special recognition for his invaluable contributions to my academic journey. His unwavering guidance, profound expertise, and tireless support have been instrumental in shaping my research endeavors. Professor Dave's commitment to bestow knowledge, his patience in answering my inquiries, and his constant motivation helped me to move forward with passion and determination.

Furthermore, I am deeply grateful for the guidance, feedback, and moral support I have obtained from my colleagues and acquaintances. Their presence has been a source of strength during challenging times and a cause for celebration during moments of achievement. Their diverse perspectives and shared experiences have enriched my understanding and fostered personal growth.

Lastly, I am deeply thankful to my family and friends for their love, understanding, and constant support during this journey. Their words of encouragement and sacrifices have been the primary motivation behind my pursuit of academic excellence, and I am eternally grateful for their constant belief in my abilities.

While this thesis represents an individual effort, it is undeniably a product of the cumulative support, insight, and motivation from many. To everyone who has played a part in this journey, whether large or small, I offer my sincerest gratitude. Your support has been invaluable, and I am truly humbled by the opportunity to express my thanks.

Glossary

- **Deepfake**: Morphed videos or images created by Artificial Intelligence
- **Fake Image Detection**: The process of identifying images that have been manipulated or altered in a deceptive manner to mislead viewers.
- **LSTM**: LSTM stands for Long Short-Term Memory which is a type of recurrent neural network that has an ability to remember patterns over long durations of time.
- **AUC:** A performance metric used to evaluate the effectiveness of a classification model, typically used with the ROC curve.
- **ROC:** A visual chart that plots the true positive rate versus the false positive rate for a binary classification system as its discrimination threshold changes.
- **GAN:** Generative Adversarial Networks (GANs) are a form of deep learning framework composed of two neural networks: the generator and the discriminator. The generator produces artificial data instances, like images, and the discriminator evaluates whether these samples are genuine or synthetic.
- **Autoencoders:** Type of neural network architecture used for unsupervised learning. They consist of an encoder network that compresses input data into a lower-dimensional representation, and a decoder network that reconstructs the original input data from this representation.
- **CNN:** Stands for Convolutional Neural Network, which is a deep learning architecture designed for processing structured grid data like images. It utilizes convolutional layers to automatically learn features from the input data.

Table of Contents

Table of Figures

Table of Tables

Leveraging Machine Learning & Deep Learning Methodologies to Detect Deepfakes

Aniruddha Tiwari

A Thesis in Partial Fulfillment of The Requirements for
The Degree of
Master of Science in Data Science

Minnesota State University,
Mankato, Minnesota
April 2024

Abstract

The rapid evolution of deep learning (DL) and machine learning (ML) techniques has facilitated the rise of highly convincing synthetic media, commonly referred to as deepfakes. These manipulative media artifacts, generated through advanced artificial intelligence algorithms, pose significant challenges in distinguishing them from authentic content. Given their potential to be disseminated widely across various online platforms, the imperative for robust detection methodologies becomes apparent. Accordingly, this study explores the efficacy of existing ML/DL-based approaches and aims to compare which type of methodology performs better in identifying deepfake content.

In response to the escalating threat posed by deepfakes, previous research efforts have focused on inventing detection models leveraging CNN architectures. However, despite promising results, many of these models exhibit limitations in reproducibility and practicality when confronted with real-world scenarios. To address these challenges, this study endeavors to develop a more generalized detection framework capable of discerning deepfake content across diverse datasets. By training simple yet effective ML and DL models on a curated Wilddeepfake dataset, this research assesses the viability of detecting authentic media from deepfake counterparts. Through comparative analysis and evaluation of model performance, this study aims to contribute to the advancement of reliable deepfake detection methodologies. The models used in this study have shown significant accuracies in classifying deepfake media.

Chapter 1

Introduction

In the past few years, people have faced emerging issues with AI-made face-swapped videos or images. Specifically, machine learning fabricates images/videos in such a way that it is usually difficult to tell them apart from the real ones, and for these reasons they are called deepfakes. Deepfakes are primarily synthetic media in which a person's face in the image or video has been replaced by the likeliness of some other person. For instance, consider a video in which a reporter is reading the news; now replace his face with Person B. Now, the result is that Person B is reading the news in the video. Various techniques have already been studied by researchers to distinguish between the fake and real images, which include machine learning techniques like Support vector machine (SVM) [1-3] and deep learning techniques like Convolutional Neural Networks (CNN) [4-6]. There are two most common ways of creating deepfakes, which are Generative adversarial networks (GANs) and Auto-encoders, which have the ability to misguide people by developing a high-dimensional pseudo image/video and making the audience believe that a well-known person is speaking/acting, whereas, in reality, he is not.

GANs are hard to train and difficult to use as a computational technique because they consist of a set of two neural networks, called generators and discriminators. This technique of deepfake generation believes that the generator should fool the discriminator; this makes the fakes more like real video and is significantly harder for the human eye to distinguish. Similarly, Autoencoders are another well-known deep learning technique; they have an encoder and a decoder function that carries some shared weights. Encoders and decoders can be utilized to take compressed details of an image

to outsmart the existing compressing standards [7]. There are three different ways of using face-swapping techniques, which are lip syncing (in which the lip region in the target image is replaced by another person's lip region and can make someone speak what they originally never spoke); face-swapping (in which the person's face in the target image is replaced by the person's image in source image) [8]; and the most dangerous one, head-puppetry (in which target person's face is animated through the person sitting in front of the camera). Figure. 1 is a very good example of head puppetry [9].

Figure 1. Example of head puppetry deepfake of President Barak Obama, from [9]

Deepfakes could be circulated over the globe with false news and people tend to believe and find it credible, which is an emerging threat to society. Detection of the deepfakes in the early years of the issue was possible even with the naked eye since you could see a color mismatch, the low resolution of synthesized faces, and sometimes temporal flickering [7], but since the current methodologies have transmogrified, detecting it with our eyes is not even possible, it is of that high quality [10, 11]. Deepfake has been a source of misinformation and malevolence since it was introduced, causing a threat to society and political agendas [12]. At the depth of deepfake is deception—which is knowingly, technically, or intentionally, causing someone to believe something which is not true, typically to address personal, political agendas[13].The repercussions of deception by deepfakes are far more perceptible compared to that of verbal deception, it

not only changes the oral content, but it also fabricates the visual properties of the videos/images on how the message was delivered [13].

Deepfakes produce tremendous threats for the future where fake news can be seen everywhere and can cause social impact, this unrestrained technological change deforms the truth, many times it could be intended for fun purposes, but often it is not. Therefore, I think there is a necessity for an effective, efficient detection techniques/algorithm which could stop the creation and usage of fake videos and evidently ML/DL techniques are good assets to recognize the human activity pattern to fight the threat [14-16]. In this study, I aim to present an overview of the latest advancements in deepfake detection models and address their limitations, along with outlining potential areas for future research.

The latter part of thesis is structured as follows: Chapter 2 will delve into a comprehensive literature review, encompassing past deepfake methodologies, our search methodology, and literature analysis; Chapter 3 will provide detailed insights into the WildDeepfake Dataset; Chapter 4 will elaborate on the methodology employed in my research, covering various aspects including feature extraction; Chapter 5 will present our findings; Chapter 6 will offer a thorough analysis; and finally, Chapter 7 will conclude with discussions on limitations and future directions. This will be followed by references and an Appendix containing code snippets for all models utilized in the study.

1.2 Detecting Deepfake's

Traditionally, deepfake detection methods involve combination of forensic analysis, complex algorithms and humanly inspection. State-of-the-art methods for deepfake detection often involve sophisticated neural network architectures, leveraging both temporal and spatial features extracted from the data. One prominent approach is to use Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) in combination with advanced training strategies and feature extraction techniques [15-17]. Deepfake detection algorithms may perform consistency checks across different modalities within the content, such as aligning facial movements with corresponding audio or verifying the coherence of body movements with the context of the scene [8]. Inconsistencies between different elements of the content can signal potential tampering. Advanced deep learning models, including CNNs and RNNs, can be trained to distinguish between real and fake media. These models learn to retrieve high-level features from the image data and classify whether it is genuine or synthetic[12]. Transfer learning, where pre-trained models are fine-tuned on deepfake detection tasks, has shown promising results in improving detection accuracy.

1.3 Research Imperative

Detection of the Deepfake's is very crucial for various reasons, thus the few are listed below:

Preservation of Trust in Media: In today's era, where information spreads rapidly through various digital platforms[18], maintaining trust in the authenticity of media content is paramount. Deepfakes, with their ability to convincingly alter audiovisual content, pose a significant threat to this trust. Without reliable methods to detect

deepfakes, individuals may become increasingly skeptical of the content they encounter, leading to a breakdown in trust in media sources and institutions.

Combatting Disinformation and Misinformation: Deepfakes have the potential to amplify the spread of disinformation and misinformation. They can be used to fabricate events, speeches, or statements, making it challenging for individuals to discern fact from fiction[18]. Detecting deepfake helps in identifying morphed medias, which results in spreading less false information in the world.

Protecting Individuals and Public Figures: Deepfakes can be used to create highly realistic forged videos or audio recordings of human beings, including public figures and celebrities, engaging in activities, or making statements they never did [9-10]. Such fabricated content can damage reputations, encourage harassment, or manipulate public opinion, and raise ethical concerns. Detecting deepfakes is crucial for protecting the integrity and privacy of individuals from such malicious attacks [8].

Preserving Democratic Processes: In democratic societies, the ability to make informed decisions based on accurate information is fundamental [8-10]. Deepfakes can be exploited to manipulate political discourse, influence elections, or undermine democratic processes [12]. By developing effective methods to detect deepfakes, we can mitigate the impact of such manipulative tactics and uphold the integrity of democratic institutions.

Advancing Technological Solutions: The growth of deepfakes underscores the need for ongoing research and development of advanced detection techniques. Investing in research to enhance the precision and efficiency of deepfake detection not only helps

address the current threat posed by deepfakes but also drives innovation in the broader field of computer vision, machine learning, and cybersecurity.

1.4 Objective

This study aims to evaluate the effectiveness of machine learning algorithms versus deep learning algorithms in the detection of deepfake videos, utilizing the Wild Deepfake dataset as a reference. The research endeavors to answer the following questions:

1. Machine learning algorithm demonstrates higher efficacy than Deep learning algorithm.
2. How does the inclusion of a previously unused dataset impact the performance of each algorithm?
3. What are the comparative strengths and weaknesses of machine learning and deep learning approaches in deepfake detection?

Research Question:

The central research question driving this investigation is: "What is the comparative effectiveness of machine learning and deep learning algorithms in detecting deepfake, considering the utilization of the Wild Deepfake dataset alongside previously unutilized datasets, and how can we develop an optimized deepfake detection model capable of maintaining both transferability across various datasets and achieving heightened accuracy?"

The objectives include identifying the best-performing machine learning and deep learning algorithms for deepfake detection, evaluating their performance with a

previously unused dataset, and conducting a comprehensive comparative analysis to elucidate their respective strengths and weaknesses.

Chapter 2

Literature Review

In recent years, the rise of AI-generated face-swapping videos and images, commonly known as deepfakes, has presented a significant societal challenge [5]. These manipulative media creations pose threats to privacy, integrity, and security [17-22]. Given their proliferation across social media platforms, it has become imperative to develop effective countermeasures against this phenomenon. Although researchers have long been working on deepfake detection, recent years have witnessed notable advancements in detection techniques. Detecting deepfakes entails a binary classification task [12][13], determining the authenticity of media content, thus requiring substantial datasets encompassing both genuine and synthetic videos for model training. Several publicly available datasets cater to this need, including Celeb-Deepfake (Celeb-DF) [7], FaceForensics++ [13], FFIW10K [13], Deepfake Detection Challenge (DFDC) [23], WildDeepfake [24], among others.

The purpose of this literature review is to explore the current body of research concerning the detection of deepfakes utilizing both machine learning and deep learning methodologies. This review will delve into the various techniques employed by researchers in this domain, along with an examination of the strengths and weaknesses associated with each approach. Through a comprehensive synthesis of these investigations, this review aims to offer insights into the prevailing landscape of deepfake detection research, while also highlighting potential avenues for future exploration and study.

2.1 Background

In order to grasp the essence of this thesis, it is crucial to dive into the principle of machine learning, a pivotal subfield within artificial intelligence. Machine learning entails the training of algorithms to formulate predictions by scrutinizing vast sets of data. This discipline encompasses various methodologies, including supervised, unsupervised, and reinforcement learning paradigms [18, 25, 26]. Supervised learning entails the training of algorithms on tagged data [19], where the model is familiarized with data points tagged with corresponding categories. Subsequently, it undergoes testing by anticipate labels or categories for unobserved data. Conversely, unsupervised learning involves training algorithms on unlabeled data, aimed at uncovering latent structures inherent in the data.

For the scope of this thesis, supervised learning is employed exclusively, wherein the deepfake images are categorized as either "real" or "fake." Deep learning, a branch of machine learning, utilizes neural networks to identify complex relationships between inputs and outputs. These networks consist of multiple layers of interconnected nodes that analyze data and produce predictions. Owing to the intricate network structure akin to the human brain, neural networks exhibit adaptability and capability in assimilating new information.

Deep learning proves particularly efficacious in the area of deepfake detection, owing to the added layers or nodes that amplify the complexity of the model. This inherent complexity mitigates the need for extensive human intervention [8, 27], thereby facilitating the analysis of more intricate input data. Notably, deep learning facilitates feature extraction in images with relatively less preprocessing overhead.

However, it is noteworthy that simpler machine learning techniques remain proficient in deepfake detection despite the persuasion of deep learning methodologies.

2.2 Search Strategy

To find the most relevant research papers for my study, I conducted a thorough search across multiple databases, primarily utilizing Google Scholar, Minnesota State Library resources and Papers with Code [28]. Papers With Code is an especially valuable resource as it not only hosts academic papers but also provides links to associated software and datasets. This comprehensive approach allowed me to identify papers that not only discussed deepfake detection methods but also utilized datasets pertinent to my research interests, such as the WildDeepfake [24] and Deepfake Detection Challenge (DFDC) [23] datasets.

In my search process, I employed a combination of keywords related to deepfake detection, machine learning, and face forgery to narrow down my results. By focusing on papers published after 2015, I ensured that my review encompassed the most recent advancements and findings in the field.

To guarantee the quality and relevance of the selected papers, I established inclusion criteria. Firstly, the paper had to propose a method for detecting deepfakes and utilize an explainable dataset with good amount of size like WildDeepfake, DFDC, or FaceForensics ++ dataset etc. Secondly, the papers were only included if they were published in peer-reviewed journals or conferences, guaranteeing a certain level of rigor and credibility. Lastly, the paper had to provide an assessment of the proposed deepfake detection method, enabling me to assess its effectiveness and performance.

Following the application of these inclusion criteria, I identified good amount of research papers that met all my requirements and were thus deemed suitable for inclusion in my literature review. These papers formed the basis of my study, providing valuable insights and perspectives on the current state of deepfake detection research.

2.3 Dataset

The search for relevant dataset began with a thorough exploration into the DeepFake Detection Challenge (DFDC) and the associated scholarly works [23]. The DFDC initiative, spearheaded by industry giants such as Facebook and Kaggle, aimed to combat the rising threat of deepfake technology. Central to this effort was the release of the DFDC dataset comprising over 128,000 videos, of which more than 104,000 were distinct deepfake creations. This dataset represented a significant leap in scale compared to existing resources, marking a pivotal milestone in the realm of deepfake detection.

The creation procedure of the DFDC dataset involved collaboration with members who willingly permitted to have their images altered for research purposes. Advanced deepfake production techniques, including the Deepfake Autoencoder [5-8] method, Morphable-Mask/Nearest-neighbors face swap (MM/NN), Face Swapping GAN (FSGAN), StyleGAN, refinement, and audio swaps, were employed to produce realistic synthetic videos. Before insertion of the media in the dataset, these videos underwent preprocessing steps such as face tracking and alignment to ensure consistency and quality. Additionally, augmentation techniques such as face masks and Poisson blending were applied to enhance the realism of the deepfake videos, thereby posing a greater challenge for detection algorithms.

Beyond the sheer size and complexity of the dataset, the DFDC initiative also organized a Kaggle competition to foster innovation in deepfake detection [23]. With over 2,000 participants, the competition served as a catalyst for the development of novel detection methodologies. However, the evaluation process introduced a unique challenge; while models were initially assessed on a public test set during the competition, final evaluation occurred on a private test set, unbeknownst to participants. This discrepancy in evaluation datasets underscored the importance of robustness and generalization in deepfake detection models.

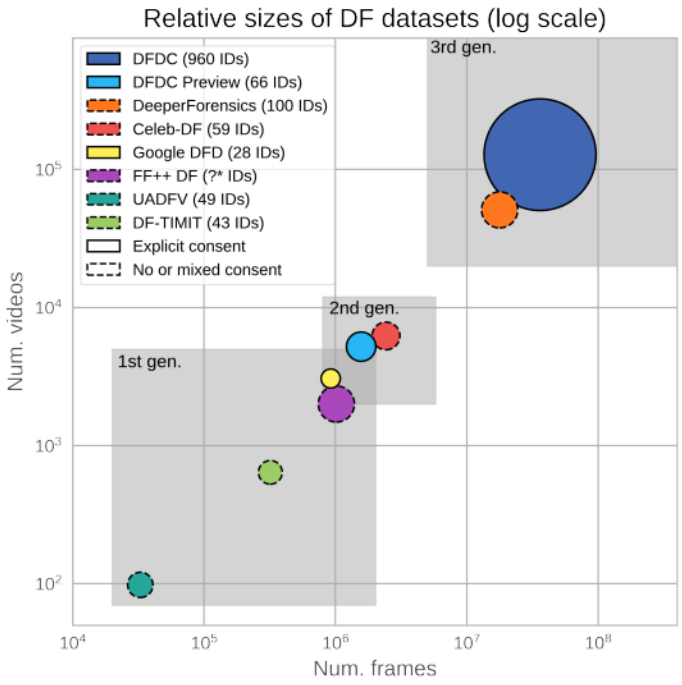

Figure 2.Relative size comparison of the current deepfake dataset from Extracted from [23]

In addition to the DFDC dataset, attention was also directed towards the WildDeepfake dataset [24], which offered a contrasting approach to data collection. Unlike the DFDC, which relied on controlled environments and known manipulation techniques, the WildDeepfake dataset was sourced entirely from the internet. This

dataset, comprising 7,314 face sequences from 707 videos [24], encompassed a diverse array of deepfake instances with varying qualities, angles, and creation methods. Despite its smaller scale compared to the DFDC, the WildDeepfake dataset posed unique challenges due to its heterogeneous nature and unknown origins of the deepfake content.

Researchers recognized the untapped potential of the WildDeepfake dataset and sought to explore its utility in deepfake detection research. However, the dataset's underutilization in prior studies highlighted the need for tailored detection methodologies capable of addressing its distinct characteristics. [24] for instance, proposed attention-based deepfake detection networks (ADDNets) designed to accommodate the temporal dynamics of deepfake sequences, thereby enhancing detection accuracy. Their findings underscored the value of developing specialized techniques to effectively address the obstacles posed by real-world deepfake scenarios.

Furthermore, the landscape of deepfake datasets expanded with the introduction of FaceForensics++ (FF++) and other similar resources. These datasets, comprising real videos morphed using popular deepfake techniques such as Deepfake, Face2Face, FaceSwap, and NeuralTextures, provided additional benchmarks for evaluating detection algorithms. However, researchers noted challenges related to generalization, as models trained on these datasets struggled to adapt to unseen deepfake variations due to the limited diversity of identities and manipulation techniques.

2.3 Previous Methods

Over the past few years, there has been a surge in fraudulent media in the world. This media which has brilliant deceptive ability is referred to as deepfakes, which is an alarming issue [29]. These manipulated media assets have the possibilty to undermine

trust, control public opinion, and facilitate various forms of fraud and misinformation. Recognizing the gravity of this threat, researchers and technologists have intensified efforts to develop effective methods for detecting and mitigating the impact of deepfakes.

Detecting deepfake is fundamentally a binary classification problem: distinguishing between authentic and manipulated media. To train accurate detection models, a diverse dataset encompassing both genuine and fake media is indispensable. Such datasets provide the foundation for machine learning algorithms to discern patterns and characteristics unique to deepfakes [2,5]. However, creating such datasets is challenging due to the lack of authentic deepfake-free media and the ethical considerations surrounding the use of manipulated content.

Despite the importance of simple machine learning techniques in deepfake detection, the prevailing trend in research has leaned towards more sophisticated approaches, particularly those leveraging deep learning and neural networks [5,6]. These advanced methods offer superior performance in discerning subtle manipulations and complex patterns characteristic of deepfakes [16]. However, their complexity often necessitates significant computational resources and expertise, limiting their accessibility and applicability in certain contexts.

In navigating the landscape of deepfake detection research, a comprehensive review conducted by Juefei-Xu et al. [30] sheds light on the evolving strategies and methodologies employed in both generating and detecting deepfakes. By analyzing a wide array of papers and studies, the authors elucidate the dynamic interplay between adversarial actors creating deepfakes and researchers developing countermeasures. This

iterative process drives continual advancements in both deepfake generation techniques and detection methods, underscoring the need for ongoing vigilance and innovation in addressing this evolving threat landscape.

While deepfake detection remains a challenging and multifaceted endeavor, recent progress in research offers promising insights and approaches to tackle this issue [5]. By leveraging interdisciplinary expertise, large-scale datasets [23, 24], and innovative algorithmic techniques, researchers are gradually enhancing the robustness and efficacy of deepfake detection systems. However, continued collaboration and vigilance across academia, industry, and policymakers are essential to stay ahead of emerging threats posed by deepfake technology.

2.3.1 Strategies for Detecting Deepfakes

In the portion of the study dedicated to detecting deepfakes, Juefei-Xu et al.[30] provide a comprehensive overview of three primary methodologies used for this purpose: biology-based detection, frequency-based, and spatial-based [30].

Spatial-based detection methods focus on examining visual deviations within the spatial domain of media to distinguish between genuine and fake media. This approach encompasses techniques such as image forensics-based detection, which scrutinizes pixel-level variations using various forensic tools, and deep neural network (DNN) based detection [30, 31], that leverages DNN models to retrieve spatial features from visuals.

Frequency-based detection methods, on the other hand, identify discrepancies in fabricated images that may not be readily apparent in the spatial area [32]. These

discrepancies originate from shortcomings inherent in GANs, a popular technique used for creating deepfakes. By analyzing frequency patterns, these methods can uncover subtle artifacts indicative of manipulation.

Biological-based detection approaches exploit environmental signs present in authentic media to differentiate them from fakes. This consists of detecting inconsistencies in visual and auditory cues, such as discrepancies in lip-syncing or facial expressions, which may be absent or unnatural in synthesized media. Additionally, biological signals like motion, facial emotions, for instance even heart rate can serve as indicators of authenticity.

While spatial and frequency-based detection methods are effective at identifying overt visual artifacts in deepfake media, they often struggle with generalization to unknown techniques and susceptibility to adversarial attacks[30, 32]. This means that they may be easily fooled by minor alterations to the input data, compromising their reliability in present scenarios where such attacks are prevalent.

Moreover, [30] suggests that as deepfake technology continues to evolve, deepfakes may become indistinguishable from real media, delivering spatial and frequency-based detection less effective. In such scenarios, biological-based signals may emerge as a more vigorous solution for identifying deepfakes. However, as deepfake generation techniques advance to replicate natural signals more accurately, the efficacy of biological-based detection may diminish.

Additionally, the authors of [30] acknowledge the existence of alternative detection approaches that do not neatly fit into the aforementioned classes, although these

approaches are less commonly utilized. Notably, convolutional neural networks (CNNs) [22, 29, 30] emerge as a popular choice for deepfake detection, particularly in image analysis, due to their effectiveness in extracting features from visual data. In contrast, linear machine learning methods are seldom utilized for deepfake detection tasks. Figure 3 provides a summary of deepfake detection methods. Spatial methods are shown in blue, frequency methods in green, biological methods in yellow, and other methods in red.

Type ID	Name
Type I1	Image Forensics based Detection
Type I2	DNN-based Detection
Type I3	Obvious Artifacts Clues
Type I4	Detection and Localization
Type I5	Facial Image Preprocessing
Type II1	GAN-based Artifacts
Type II2	Frequency Domain
Type III1	Visual-audio Inconsistency
Type III2	Visual Inconsistency
Type III3	Biological Signal in Video
Type IV1	Others

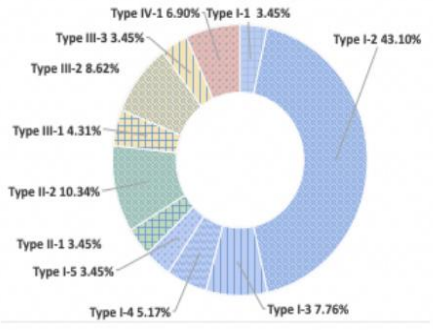

Figure 3. Various types of Deepfake detection categorize extracted from [30]

2.3.2 Model Evaluation

Different methods and models in machine learning need to be assessed using various evaluation metrics to understand how well they perform. Since there isn't a one-size-fits-all approach for evaluating models, researchers must consider a range of metrics. Frequently employed measures in this area encompass precision, accuracy, F1 score, recall, confusion matrices, ROC curves, and the AUC (Area Under the Curve). [33, 34].

Accuracy is a straightforward measure that tells us the ratio of correctly organized events out of all events. Whereas Precision and recall focuses on the efficacy of the model, specifically on positive instances. Precision calculates the fraction of truly

ranked positive events out of all instances classified as positive, while recall measures the shares of correctly grouped positive events out of all actual positive events [27]. The F1 score combines precision and recall into a single value, providing an overall assessment of the model's performance.

To gain a better understanding of these metrics, researchers often use confusion matrices. These matrices provide a comprehensive breakdown of the model's effectiveness by showing the counts of true positives, true negatives, false positives, and false negatives [27, 33, 35]. True positives represent the correctly classified positive instances, while false positives indicate false instances that were falsely grouped as positive. Similarly, true negatives and false negatives symbolize the same values but for negative instances.

Additionally, ROC curves offer a visual representation of the model's ability to discriminate between true and false events. By plotting the true positive rate against the false positive rate, researchers can visually distinguish the performance of several machine learning models. The area under the ROC curve (AUC) quantifies the overall performance of the model, with a perfect model achieving an AUC of 100%.

By utilizing these performance metrics and evaluation methods, researchers can effectively assess the performance of machine learning models in identifying deepfakes. This facilitates informed decision-making regarding the selection of models for future research and application in real-world scenarios.

2.3.3 Prior Models

In exploring more sophisticated methods beyond the simpler approaches discussed in this study, CNNs emerge as a notable alternative. CNNs are a type of deep neural network precisely designed for analyzing visual data, making them well-suited for tasks involving image processing and recognition. Bonettini et al. [36] suggested a novel approach for classifying manipulated faces in videos by harnessing the power of CNNs, particularly through the use of ensemble learning [36].

Ensemble learning involves merging several individual models to make predictions collectively, often resulting in improved performance compared to using a single model alone. The authors of [36] utilized an ensemble of different CNN-trained models to improve the accuracy and reliability of their detection method. To kickstart their approach, they employ the EfficientNet family of models, renowned for their exceptional accuracy and computational efficiency relative to other CNN architectures. EfficientNetB4, a specific model from the EfficientNet family, is chosen as the foundation due to its favorable balance of low computational requirements, fast processing speed, and high accuracy. However, the authors [36] introduced two key modifications to further enhance the performance of EfficientNetB4 in detecting manipulated faces within videos [36].

The first modification involves incorporating an attention mechanism into the model architecture. This attention mechanism facilitates the model not only to focus on the appropriate regions within the input videos but also provides insights into which parts of the images the network deems most informative for making accurate predictions. A

visual depiction of this modified architecture is illustrated in Figure 4. which can be seen on the next page.

Additionally, the authors [36] adopted a Siamese training strategy as the second modification. This training strategy involves training the model by comparing pairs of examples, enabling it to learn more effectively from the data by discerning differences between various instances.

The proposed method by the authors of [36] is rigorously evaluated using two widely recognized datasets: Face Forensics (FF++) and DeepFake Detection Challenge (DFDC). Impressively, the model achieves a high AUC of 94.44% on the FF++ dataset and 87.82% on the DFDC dataset, indicating its strong performance in detecting manipulated content across different datasets.

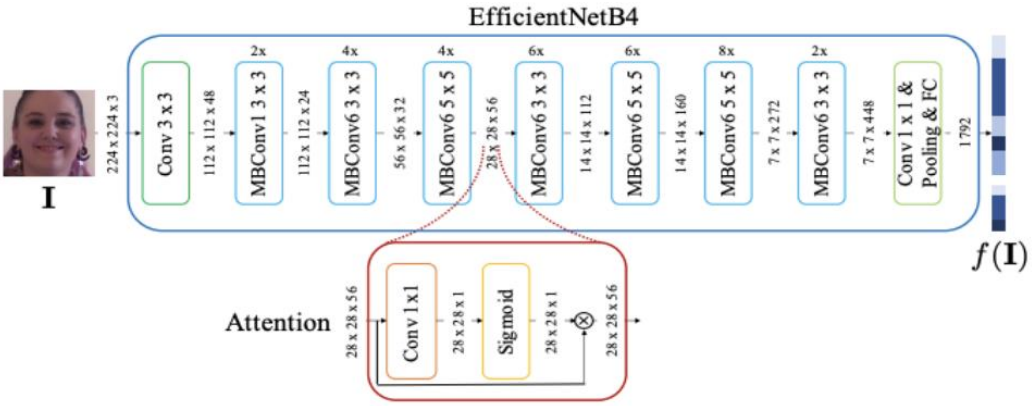

Figure 4. EfficientNetB4Att Architecture extracted from [36]

Barni et al. [37] present a technique for detecting images generated by Generative Adversarial Networks (GANs). They introduced two deep learning algorithms: the CoNet and the Cross-CoNet. The CoNet examines the conjunction matrices of the RGB channels within a picture to discern between genuine and GAN developed images. In

contrast, the Cross-CoNet [37] enhances the CoNet by considering the interrelations among the color bands, enabling it to separately calculate color and spatial co-occurrences for each band.

Aforementioned models exhibit strong performance on the StyleGAN2 dataset [37], achieving a performance accuracy of 98.15% for the CoNet and 99.70% for the Cross-CoNet. Notably, the Cross-CoNet demonstrates heightened robustness against post-processing, indicating its capacity to maintain effectiveness even when input data undergoes alterations. In the context of machine learning, "robustness" denotes a model's capacity to sustain performance despite changes in input data, suggesting enhanced utility in practical applications. The authors [37] recommended future investigations exploring deliberate attacks to test the model's resilience, as well as its performance when exposed to unfamiliar datasets.

A variety of methodologies have been explored in the quest to identify deepfake content, each targeting specific aspects such as facial inconsistencies, temporal irregularities, audio-visual disparities, and spatial anomalies within videos [23-26]. In a study aimed at discerning manipulated photos generated through face-swapping technology, researchers introduce the DenseNet169 model, leveraging facial-warping artifacts as discernible indicators [7]. To establish a robust detection framework, negative data samples comprised unaltered images of individuals augmented with various noise types including Gaussian blur, Exponential blur, and Rayleigh blur. However, due to the absence of Gaussian noise during testing after applying affine transformation to the source image, researchers explored alternative variants to ensure comprehensive detection coverage.

The experimental process involved face extraction using the dlib package, followed by random affine transformations and resizing, supplemented by the addition of random blur effects before final resizing to produce authentic images. Subsequent evaluation of the detection model was conducted using the Celeb-DF dataset [7, 38], a repository of 1000 high-definition videos featuring synthesized segments devoid of original facial artifacts, thereby posing a substantial trail for accurate detection [7].

In a complementary endeavor focusing on both the generation and identification of deepfakes, researchers delve into the utilization of autoencoders and Generative Adversarial Networks (GANs) [39]. For deepfake generation, two autoencoders are employed: the first captures the features of the root image, while the second learns the features of the goal image. Subsequently, the goal image is recreated using the decoder of the source image, resulting in media charged with attributes of the source image. To refine the quality of generated deepfakes, authors propose the utilization of DFDNet [39], an image enhancement method.

For detection purposes, researchers advocate for the adoption of MesoNet, a deep neural network designed to scrutinize fine-grained details within compressed frames [39]. MesoNet comprises four layers of successive Convolutions and pooling, followed by a dense hidden layer network. Training the model on a dataset containing over 5000 images grouped as real or fake, similar to previous studies, yielded a detection confidence interval of 80%. This comprehensive approach demonstrates significant advancements in both the creation and detection of deepfake content.

The investigation conducted by [28] has brought to light a notable observation regarding deepfake videos: many lack the natural occurrence of eye blinking. However, recognizing

the adversarial tactics employed to counter detection methods, where adversaries manipulate the eye blinking pattern, presents a fresh challenge. In response, the authors of [11] introduce DeepVision, an algorithm aimed at detecting deepfakes by scrutinizing significant deviations in eye blinking patterns. Eye blinking, a spontaneous action influenced by various factors such as age, gender, and overall physical condition, offers

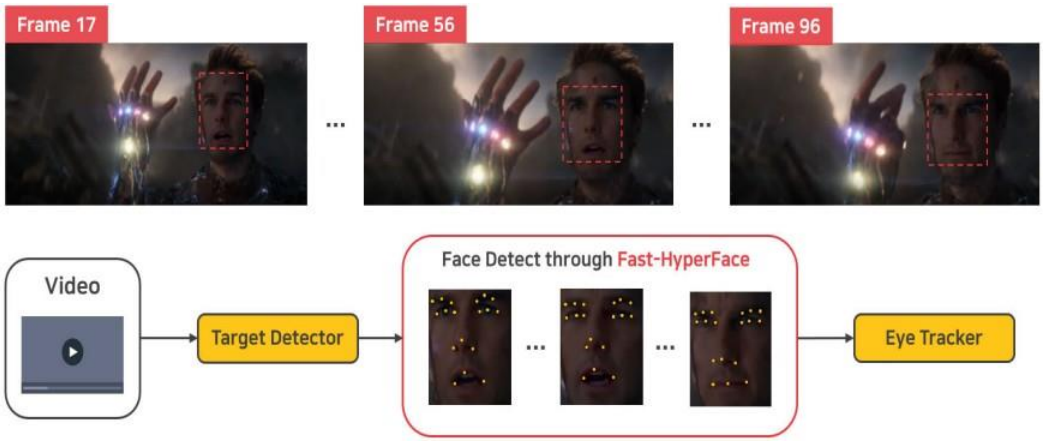

Figure 5. Visualization of DeepVision's eye tracker provides insight into its ability to measure eye blinking periods using the Eye-Aspect-Ratio algorithm [11]

a unique avenue for classification due to its iterative and unconscious nature. Leveraging this, DeepVision adopts an algorithm that meticulously examines behavioral and cognitive indicators affecting eye blinking patterns, providing a novel approach to verifying media integrity as illustrated in Figure 5.

Additionally, [40] underscores the limitations structured in conventional feature extraction methods, which often yield single artifacts, thereby impeding model performance. To address this challenge, authors [40] supported integrating an attention mechanism into the detection model to obtain both global and local facial features, thereby enhancing accuracy. This integration is used in models such as XceptionNet, ResNet-18, and ResNet-50, with the ReLU activation function replaced by Swish

activation function for improved performance with deep neural networks [30]. Experimental validation utilizing the FaceForensics++ dataset demonstrates significant classification results, with the proposed architecture based on the ResNet50 model showcased in Figure 6.

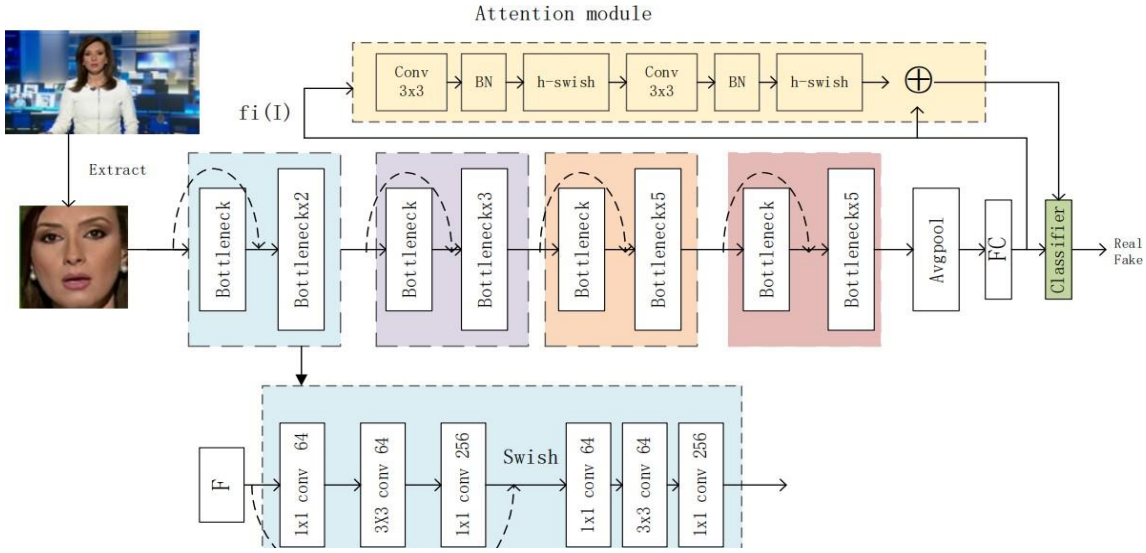

Figure 6. ResNet-50 model enhanced with an attention mechanism [29].

MesoNet, a model extensively employed in various studies [41] [42], is designed to delve into the mesoscopic features of images by utilizing a streamlined architecture with fewer layers. However, its initial design proved inadequate for processing compressed media, as it led to the degradation of background noise.

Another noteworthy advancement comes from [43], who introduced a novel hybrid architecture combining CNN and VGG16 for deepfake classification. This innovative approach outperformed existing techniques, with their Deepfake predictor (DFP) achieving remarkable accuracy and precision rates of 94%.

Additionally, [43] proposed a comprehensive method that integrates two distinct approaches to differentiate between authentic and fake videos. By leveraging ResNet-50

and a super-resolution algorithm [43], they enhanced the accuracy of identifying deepfakes, particularly in low-resolution videos. The resultant model exhibited a high accuracy rate of 94.4% on the UADFV dataset [43].

Moreover, [24] addressed fundamental challenges inherent in deepfake detection by proposing a DNN-based framework. Their approach aimed to tackle issues related to effective detection, applicability to compressed videos, and model complexity. Leveraging the FaceForensics++ dataset [24] with varying compression levels, they developed a

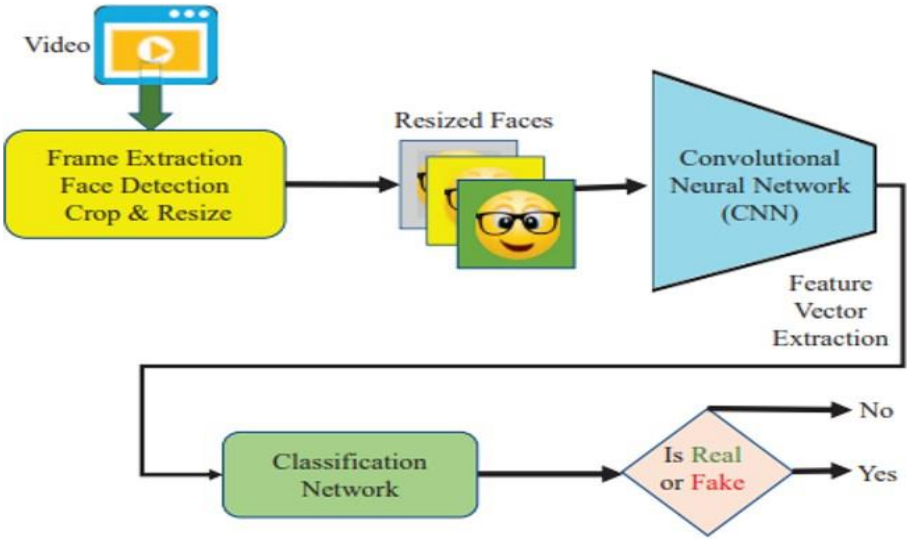

Figure 7. Representation of the proposed architecture [24].

sophisticated algorithm comprising two key modules: a CNN for retrieving frame features and a classifier network for detecting fake videos. After thorough experimentation with different CNN modules, including ResNet50, InceptionV3, and XceptionNet, authors of [24] opted for XceptionNet due to its superior accuracy. This comprehensive approach not only enhances deepfake detection but also lays the groundwork for addressing critical challenges in the field.

Having explored CNN-based studies extensively, we now turn our attention to LSTM-based investigations, which offer a promising avenue for advancing deepfake detection methodologies. LSTM architectures have shown considerable efficacy in capturing temporal dependencies and sequential patterns inherent in deepfake videos, thus presenting a compelling alternative to CNN-based approaches.

Researchers at [44] have directed their efforts towards the detection of video deepfakes, employing sophisticated DNN such as LSTM and InceptionResNetV2, a fusion of ResNet and InceptionNet with 164 layers designed for object detection and feature extraction from images. LSTM architectures address issues like the vanishing gradient problem in RNN and are specifically engineered to secure long-term dependencies in input data, processing them sequentially [44]. Leveraging a pre-trained InceptionResNetV2 CNN for feature extraction helps alleviate training and size constraints, streamlining the model's development. Subsequently, a 2048 LSTM layer is employed to analyze video manipulation across different temporal intervals, comparing frames to discern alterations and classify them accordingly. Furthermore, the proposed model of [44] exhibits the ability to identify manipulated segments within videos.

Conversely, the study by [45] delved into the exploration of intra-frame modeling and inter-frame features to authenticate videos. Utilizing an optical flow-based feature retrieving approach, the authors fed time dependent features into a model which combines CNN and RNN architectures, achieving an impressive performance accuracy of 92% with the FaceForensics++ dataset [24]. Detailed insight into the proposed workflow is depicted in Figure 8.

I have observed that several investigations focus on either time-related details or visual signs, yet scholars in [30] have unearthed a method to differentiate between fake and authentic videos based on the pattern of human eye blinks, like what the developers of DeepVision have done. To discern between open and closed eyes, the researchers suggest employing a sophisticated neural network model blending CNN with RNN, known as Long-Term Recurrent Network. The suggested architecture of the LRCN model is illustrated in Figure 9.

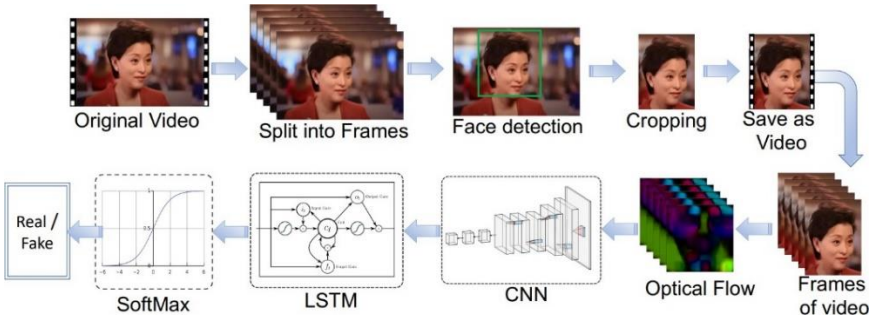

Figure 8. Workflow to detect deepfakes [37]

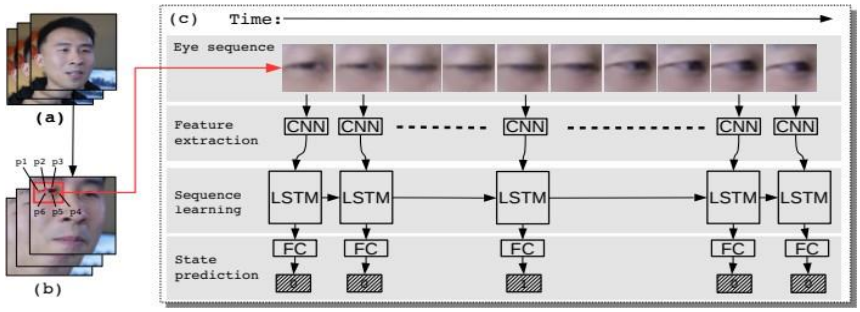

Figure 9. LRCN method[28]

Meanwhile, the researchers in [46] propose a structured approach. Initially, authors utilize a facial recognition network to identify the subject, followed by CNN to extract key features, employing Gaussian blur and noise to filter out high-frequency disturbances. Subsequently, the features are transmitted to the LSTM layer, where a sequence of

temporal attributes are constructed, analyzing tweaked facial aspects across frames. Additionally, the authors incorporate Recycle-GAN [30] into their method. The advantage of incorporating Recycle-GAN lies in its ability to iterate through evaluations, refining its parameters based on feedback, thereby improving its accuracy and adaptability.

To summarize, prior methodologies have shown notable advancements in enhancing existing models and exploring novel avenues for identifying deepfakes. The literature examined in this study underscores the diverse array of strategies available for deepfake detection, encompassing advanced techniques like CNNs and LSTMs. Moreover, researchers have searched into various datasets and training methodologies, yielding enhanced model performance and increased adaptability to different forms of manipulation. However, the same problems continue to exist. The ever-evolving nature of deepfake methods necessitates continual adjustments and enhancements to detection approaches. Additionally, the absence of standardized datasets and evaluation criteria complicates result comparison across studies.

To conclude, deepfake detection remains an active research domain, demanding ongoing exploration and experimentation. While this review highlights promising detection methods, there remains room for enhancing model accuracy and applicability. Additionally, it is important to understand comprehensive detection methods. Therefore, rather than developing a new sophisticated model, this research aims to gain deeper insights into an obscure dataset using fundamental machine learning and deep learning techniques.

2.3.4 Literature Analysis

The presented Table 1 on page number 32 showcases a detailed overview of different investigations centered on identifying deepfake content. Every record in the table shows a distinct approach, explaining the methods utilized, the impact of the study on the domain, and the outcomes attained. Significantly, the 'Result' column predominantly relies on the accuracy metric as a standard for assessing effectiveness. This decision is supported by the characteristics of the datasets employed, which are imbalanced and make metrics such as precision and recall less dependable.

The studies enumerated utilize a spectrum of methodologies, ranging from advanced deep learning architectures such as VGG16 and CNNs to traditional machine learning classifiers like Gaussian Naive Bayes, Support Vector Classifier (SVC) for SVM and Random Forest Classifier which can be used for Random Forest model. This diversity underscores the dynamic landscape of research in this domain, reflecting varied perspectives and approaches aimed at classifying fraudulent media. The primary way that we can evaluate the models is by observing the Accuracy of the model. Various studies showed that they could get accuracy higher than 80%. By which we can interpret that current methods are performing pretty well in classifying fraudulent images from the deepfake content.

In our research, I have found that authors are using various advanced algorithms to spot deepfake videos. These include CNN, DNN, RNN, and LSTM [20 – 40]. Sometimes, CNN is mixed with other methods to tweak certain settings and identify fake videos better. Some researchers are focused on spotting visual clues, while others look at changes in human behavior or characteristics that only humans have. For example, some models are

really good at spotting changes in how often people blink their eyes [11]. These methods were tested on diverse datasets like FaceForensics++ [15], Celeb-DF [7], UADFV[47], WildDeepfakes [24], and DFDC [21]. The UADFV dataset has around 34.6K images, split evenly between fake and real ones. These datasets were made using various techniques like GANs, Autoencoders, or other ways to alter images[47].

The datasets I talked about earlier helped me compare different models, but the results could vary based on which dataset is used. Each model is good at spotting certain types of fake media, like those made with GANs or Autoencoders. So, one model might not work well with all datasets. Some studies focused on analyzing time-based features and did well, while fewer looked at human behavior. The DeepVision model did great, with an accuracy of about 87.5%. Results from these studies show that deep learning models like CNN, MesoNet, InceptionNet, and XceptionNet are effective. Overall, these models have really good model performance, showing improvement in spotting deepfakes. However, the CNN based models perform best only for the dataset which was used to train the model. If for instance we change the dataset, the model's accuracy would drop significantly. So, it's valuable to use the appropriate model for the appropriate dataset to get accurate results.

Table 1. Literature Analysis

Title	Methods	Results	Dataset	Contribution
Methods of deepfake detection based on machine learning [7]	DenseNet169 + Rayleigh Blur	Accuracy: 60%	Celeb-DF	Found the indicators that can distinguish whether face manipulation is applied on any media.
DeepVision: Deepfakes detection using eye blinking pattern [11]	DeepVision with integrity verification	Accuracy: 87.5%	Eye Blinking Prediction Dataset from Kaggle.	Invented new model that was able to set apart fake videos which has inconsistent eye blinking with significant accuracy.
Deepfakes creation and detection using deep learning [40]	MesoNet with DFDNet image enhancer	Accuracy: 80%	Online dataset containing 5000 images.	Application of image enhancer over the images increased the classification accuracy of the MesoNet.
Combining deep learning and super-resolution algorithms for deepfake detection [35]	ResNet-50 with super resolution pre-processing.	Accuracy: 94.4%	UADFV	Model with super resolution had good accuracy. However, could not perform better on the head pose estimator.
A novel machine learning based method for deepfake video detection in social media [48]	ResNet-50, InceptionV3, XceptionNet	Accuracy: 88%, 86%, 96%	FaceForensics++	Trained the model with immediate compression, which resulted in significant increase in accuracy.
Deepfake Detection using InceptionResnetV2 and LSTM [36]	InceptionResNetV2 with LSTM	20 Epochs: Accuracy 84.75% 40 Epoch: Accuracy91.48%	Celeb-DF	Basis of their model is to look for the left-over traces which are not visible by naked eye and due to which they acquired significant accuracy.
A hybrid CNN-LSTM model for video deepfake detection by leveraging optical flow features [49]	CNN-Optical Flow-LSTM	Accuracy: 91.21%	FaceForensics++	Integrated Optical flow features in the CNN model to gain higher accuracy.
In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking [50]	CNN-VGG and LRCN	Accuracy: 96.2%	CEW Dataset	Introduced the LRCN (Long-Term Convolutional Network) to capture the eye blinking pattern.

Chapter 3

Dataset

For this thesis work, I carefully selected a dataset named WildDeepfake due to its unique characteristics and challenges it poses for deepfake detection [24]. Unlike many other datasets used in previous studies, WildDeepfake offers a diverse range of authentic samples, encompassing a wide variety of quality/pixel levels, angles, scenes, colored backgrounds, various lighting conditions, resolutions, compression rates, and different deepfake methods. This dataset consists of 7,314 face sequences which were extracted from 707 videos, with 3,805 images being grouped as real and 3,509 being grouped as fake. These images were taken from variety of unknown internet locations, with absolutely no information regarding their creation methods.

While the WildDeepfake dataset might not be as extensive as some other popular datasets like DFDC, its practical examples make it invaluable for my project. The diversity within the dataset presents significant challenges for deepfake detection, as it requires models to be robust across various scenarios. Testing these challenges using a variety of models is crucial to ensure the value and reliability of the detection methods. Figure 10. Shows an example of the images that are available in WildDeepfake dataset.

Figure 8. Sample image from the WildDeepfake Dataset [24]

To process the images within the WildDeepfake dataset, a detailed approach was adopted by [24]. Authors utilized the Multi-Task Cascaded Convolutional Neural Networks (MTCNN) face detector to identify facial regions in each video frame and employed an ImageNet-pre-trained MobileNetV2 network to retrieve facial attributes. Subsequently, facial landmark removal by the dlib landmark detector aligned all faces in an order to create uniformly sized images. These images captured a wide range of subjects, demographics, lighting conditions, angles, positions, and more, making them highly representative of real-world scenarios.

However, obtaining access to the WildDeepfake dataset was not easy at all since. It is not publicly available, and access had to be obtained directly from the creators. The dataset were shared through a Google Drive, but organizing the dataset proved to be a time-consuming task. The data were stored in .tar files, which needed to be downloaded and extracted, and they were spread across multiple nested folders. Careful sorting and organization were required to create a coherent structure for further analysis.

After organizing the images, data cleaning was performed to guarantee the integrity and accuracy of the dataset. Image files were carefully inspected to eliminate any mixed values or redundant images. Once the data was cleaned and organized, efforts were made to streamline the data processing pipeline. Custom functions were developed to open images, assign labels, and extract features, preparing the dataset for model training and evaluation as seen in Table 2.

Table 2. Custom functions developed to assign the label, extract features and data organization.

Function Name	Description
copy_tar_gz_files	Iterate through the main folder and subfolder and copies all the “tar.gz” files to destination folder
extract_tar_files_recursive	Extract the “tar.gz” files from all the folder to the extract folder
copy_png_files	Iteratively looks for “.png” files inside the extract folder and copies to the final folder
get_images_and_labels	Function to extract the feature from the image and assign them a label.
extract_features	Function to extract the feature which initializes what type of features needs to be extracted.

The code, accessible in Appendix A.1 to extract the data to be ready for the assigning label and extracting the features, operated as follows:

1) The code has the functionality for managing files within specified directories.
2) It facilitates the identification and copying of `.tar.gz` files to a designated destination directory.
3) Furthermore, it enables the extraction of contents from `.tar.gz` files, with subsequent deletion of the original compressed file.
4) Additionally, the code offers support for locating and transferring `.png` files to the destination directory.
5) Execution of the code initiates the process of copying PNG files from the specified source directory to the designated destination directory.

The code, accessible in Appendix A.2, works the following way:

1. Initialize seven new containers for the six characteristics and one label.

2. For each subdirectory within the designated data folder:

   i. Navigate to the subdirectory and locate files with the extension ".png".
   ii. Open and interpret the image file.
   iii. Derive all attributes utilizing the feature extraction process.
   iv. Assign each attribute to its respective container.
   v. Append the provided label (e.g., fake/real) to each picture.

3. Provide the compiled records of all characteristics and tags.

Additionally, a secondary scan of the dataset was conducted to eliminate any anomalies or inaccurate data points. During the data preprocessing phase, six distinct attributes which are entropy, wrapped, noise, blur, keypoints, and blobs were extracted and stored in separate containers after processing each file.

The feature extraction process that can be seen in Appendix A.2, calls the function “extract_features”, which uses an image data as an input and performs the feature extraction. After converting the input image to grayscale to accommodate certain feature requirements, a series of manipulations are applied to extract each feature individually. Subsequently, a feature dictionary is established to meticulously store each feature along with its corresponding value. This comprehensive dictionary is then passed back, enabling seamless integration into subsequent functions. In the next chapter I will be explaining the specifics of all the six features mentioned before in detail.

As mentioned above, the custom function for opening a folder and extracting the contents inside the folders was done by giving a directory path to the dataset folder, which post iterative execution assigns all the fake images as "fake”. This process was then

replicated for authentic data. Subsequently, a structured dataframe like a spreadsheet was generated to combine both fabricated and real datasets alongside their respective features. Post image processing, the resultant dataframe consisted of 84,980 distinct rows of image data, ensuring the requisite format for training, and evaluating machine learning models. An illustrative representation of the dataframe is showcased in Table 3.

Table 3. Illustrating features with a Data Frame [24]

Entropy	Wrapped	Noise	Blur	Keypoints	Blobs	Label
6.9667	6.20356	0.072730	49.8958	0	2	real
7.6364	6.22125	0.077251	151.995305	5	0	fake

Chapter 4

Methodology

This portion demonstrates the approach opted for constructing machine learning and deep learning models. Following an outline of the computational resources required for this project, the initial segment clarifies the feature retrieval procedure used to generate a CSV table from the data. However, the latter part examines each of the selected models and identifies the features deemed most efficacious for model training. Additionally, I opted to employ two machine learning models and one deep learning-based model to ascertain the optimal performer on the WildDeepfake dataset.

4.1 Computational and Software Used

The computational resources utilized for this study comprised lab machines at Minnesota State University, Mankato, equipped with an Intel Core i7-7700K CPU and 32GB of RAM. These machines also featured NVIDIA GeForce GTX 980 GPUs with 4GB of memory. For software utilized, Jupyter Notebook with Python version 3.9 served as the primary development environment along with Visual Studio as per the execution needs. All requisite libraries were included, with particular emphasis on scikit-learn for model creation. Notably, the scikit-learn library facilitated the execution of various machine learning algorithms, including RandomForestClassifier and svm.SVC [51] by the sklearn.

Additionally, for feature extraction from images, deep learning libraries such as TensorFlow and Keras were employed, with the VGG16 [44] model serving as a pre-trained neural network for extracting high-level features from images. These libraries

provided robust tools for building and training neural networks [44], enabling the extraction of informative features essential for model training and evaluation.

4.2 Feature Extraction

The process of attribute selection involved experimentation and analysis. Upon examining existing studies, identifying optimal features and extraction methods proved challenging. Prior models often depend on sophisticated pre-existing techniques like ResNet, MesoNet, and Xception models for image and feature extraction [52]. These methodologies leverage deep learning algorithms to isolate facial data frames from longer videos and extract features. However, as the dataset for this project consisted solely of facial images, such advanced technology seems to be irrelevant.

Despite possessing previous knowledge of machine learning model setup, a key challenge emerged: the extraction of features from the dataset. To enable machine learning models to operate effectively, image data must be configured into tabular format. Thus, the objective was to compile a table of actionable attributes from the images, facilitating pattern identification and accurate predictions by the models. Furthermore, all features utilized were numerical, encompassing both continuous and discrete values.

Initially, the selection and extraction of initial features were undertaken from a subset of images for model training and recognition purposes. A decision tree model was established to validate the model development process and image classification capability of the machine because it is straightforward and easy to understand. It breaks down the decision-making process into simple, understandable steps, mimicking how humans make decisions. This makes it suitable for tasks like image classification, where we want to

understand how certain features lead to specific outcomes. Decision tree model Serving as a template, this model ensured that upon feature extraction, they could seamlessly integrate into a model, guaranteeing proper formatting and accurate outcomes. However, challenges emerged concerning data format discrepancies. For instance, the Histogram of Oriented Gradients, considered potentially advantageous for object detection within images, was unable to be effectively utilized for training basic models in this study. Despite its extraction from all images, it outputs a lengthy dimensional array, which posed an interpretation challenge for decision tree models that rely on individual features.

To find valuable features and secure proper formatting, a method was employed where individual images were chosen and validated with prospective features available in the scikit-image library. This particular library was selected because of its familiarity with previous classes and its well-documented resources. By exploring different options, studying examples, and analyzing the code, promising attributes were found. Initially, ten numerical components were selected based on their probable significance. The accuracy of the resulting model showed significant improvement compared to previous tests that only considered color data, indicating positive progress in the project.

To refine the selected features, I tested the models by deleting one attribute at a time and examining the accuracy. This method is favored in identifying which features improved accuracy and which did not. Initially chosen feature like "Local Maxima" didn't impact accuracy. After multiple rounds of selection and hit and trial, I decided on six features which are: blobs, key points, phase unwrapping, blur, noise, and Entropy. Table 4 on page number 41 shows the change in accuracy when each feature was removed.

Bigger numbers indicate more valuable features. Blur was consistently the most valuable, followed by entropy. Noise and phase unwrapping had the smallest impact on accuracy when removed. Despite this, every feature helped detect deepfakes. We are going to talk more in depth about each of the selected features.

Table 4. The decrease in accuracy post removing the features from the model.

Feature	Random Forest	SVM
Entropy	1.64	3.46
Blobs	0.42	3.65
Blur	1.73	5.22
Keypoints	0.23	1.44
Phase Unwrapping	0.16	0.82
Noise	0.01	3.78

4.2.1 Final Selected Features

4.2.1.1 Entropy

Entropy is a measure of complexity in an image, allowing us to understand how varied the gray-level distributions are within it. This complexity can capture subtle differences in shades of gray Figure 11, which are crucial for detecting anomalies or patterns in the

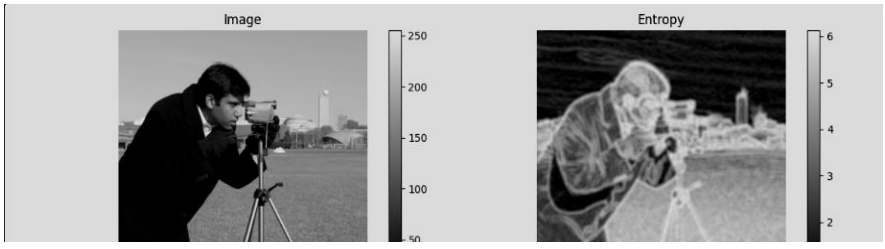

Figure 9. Visualization of an Entropy from [53]

image. To compute entropy, I used the Shannon entropy formula, which calculates the entropy value based on the probabilities of different pixel values in the image.

Essentially, it sums up the product of each pixel's probability and the logarithm of that probability. This process results in a continuous float value representing the entropy of the image, providing valuable information for machine learning models to interpret and analyze images effectively [53]. Where p(K) represents the probability of pixels.

$$S = -\sum(p(k) * \log(p(k)))$$

4.2.1.2 Noise

The "noise" feature originated from a technique called "non-local means denoising for reatining textures," as described in scikit-image [53]. In Figure 12, the image demonstrates the effect of de-noising using the non-local means filter. This algorithm replaces a pixel's value with an average derived from other nearby pixels, effectively restoring textures. However, instead of removing noise from the image, I measured the standard deviation of the noise using the function named estimate_sigma [53, 54]. The assumption was that highly fluctuating noise values might indicate fake images, while more consistent values could signify real images.

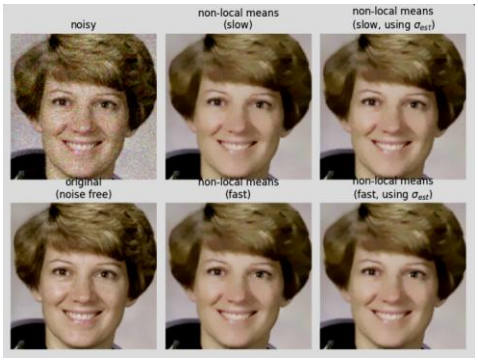

Figure 10. From the Scikit image library [53], visualization of Noise and denoising.

4.2.1.3 Phase Unwrapping

Phase unwrapping helps reveal the true signal hidden within an image that can only be seen within a limited range. This uncovers important numerical data from the image, crucial for the models to understand. The unwrapping, depicted in Figure 13, produces a lengthy list of values from the image in the form of an array. To make this data useful for the machine learning models to interpret, I determined its range by subtracting the minimum from the maximum value. Which gave a continuous decimal number, that can be utilized by the model to interpret the wrapping value. The goal was for this range to offer insight into the image's characteristics, potentially varying between real and fake images [53].

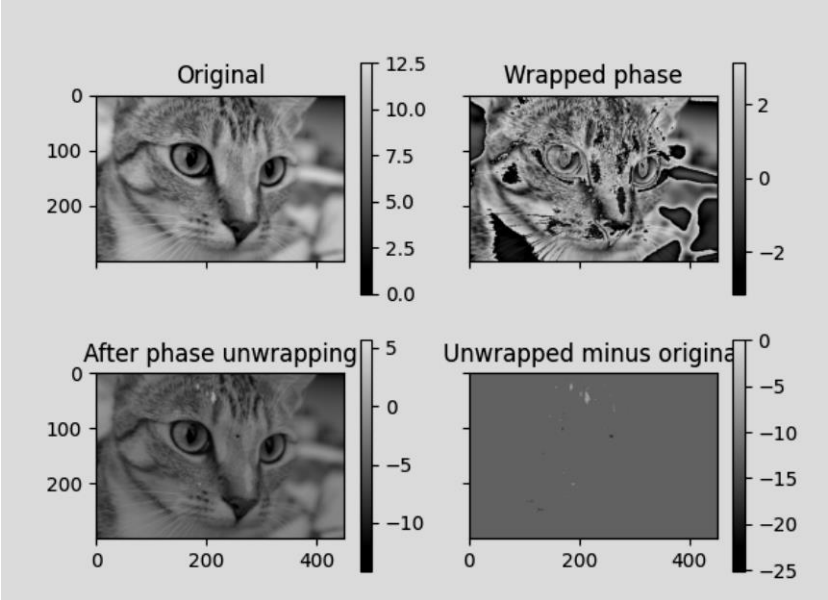

Figure 11. Visualization of Phase unwrapping from [53]

4.2.1.4 Keypoints

The keypoints feature relies on a special detector called CENSURE, known for its ability to detect details in images regardless of their size. According to the scikit-image library,

this detector performs better than others when it comes to tasks like aligning images and estimating movement [53, 54]. When applied to an image, the CENSURE detector identifies specific points of interest, called keypoints. From these keypoints, I extracted a value representing their quantity, which the models could use to learn patterns. This value was a whole number, useful for training the ML models. The expectation was that the count of keypoints might vary between fake and real images. keypoints are represented as red dots in Figure 14 [53].

Figure 12. Representation of Keypoints in red dots by [53].

4.2.1.5 Blur

One of the important features is the estimated strength of the blur, In Figure 15 on the next page, is an example of a blurred image. To quantify the blur, a filter was applied to the image, and the average blur intensity across the entire image was computed. This calculation resulted in a continuous value, which was included as one of the features for training the future machine learning models. The expectation was that fake images might exhibit higher blur values compared to real ones.

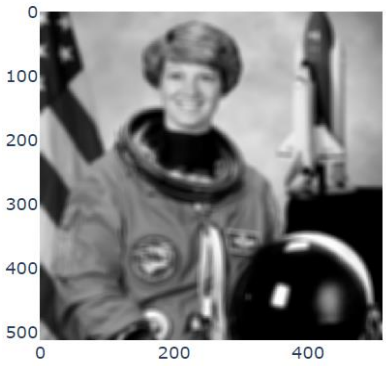

Figure 13. Sample of the strength of blur from [53]

4.2.1.6 Blobs

The final feature that has been selected was the blobs. The method employs the Difference of Gaussian (DoG) technique to identify blobs, which are areas of either radiant spots on darker backgrounds or dark patches on glowing backgrounds in an image. In Figure 16, you can see blobs detected in a picture of stars in space. A straightforward count of these blobs was obtained from images, all standardized to the same size. This count was then incorporated into the models with the expectation that altered pictures might exhibit different numbers of blobs compared to authentic ones.

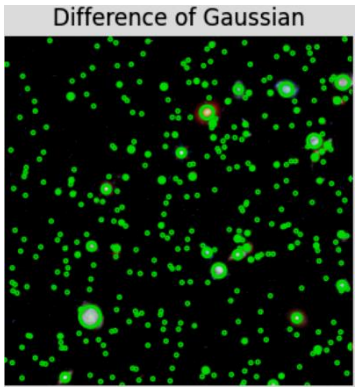

Figure 14. Example of difference of Gaussian method used in Blobs [53].

.

Table 5. Extracted features from the WildDeepfake dataset.

Extracted Feature
Entropy
Blobs
Blur
Keypoints
Phase Unwrapping
Noise

4.3 Model Selection

Post the extraction of features from the image dataset, six selected features as shown in Table 5. served as the basis for training the ML models. The selected models included the Random Forest and SVM. These choices were made because of their robust capabilities, widespread adoption, simplicity, and existing literature on deepfake classification. Furthermore, these models were anticipated to get diverse outcomes. Notably, each of these approaches operates on the principles of supervised learning, wherein models are trained using labeled data (i.e., "fake" and "real") to anticipate outcomes. We will discuss about the model more in the following subsections.

4.3.1 Random Forest

The Random Forest algorithm, employed for classification, stands out as a robust ensemble learning technique that harnesses the collective power of various decision trees. Every decision tree within the ensemble is trained on a distinct subset of the dataset [34], ensuring an equitable representation of data points. This process promotes diversity in the trees' perspectives, enhancing the model's ability to capture complex patterns and relationships present in the data. By randomly selecting observations and features for each

tree, Random Forest mitigates the chance of overfitting and fosters a more generalized understanding of the underlying data distribution.

Figure 17 provides a visual representation of the Random Forest architecture, showcasing its inherent flexibility in handling datasets with diverse features and complexities. Through the collaborative efforts of individual decision trees, the algorithm aggregates their outputs to arrive at a final classification decision [34]. This ensemble approach not only improves the robustness and reliability of predictions but also enhances the model's resilience to noise and outliers present in the data.

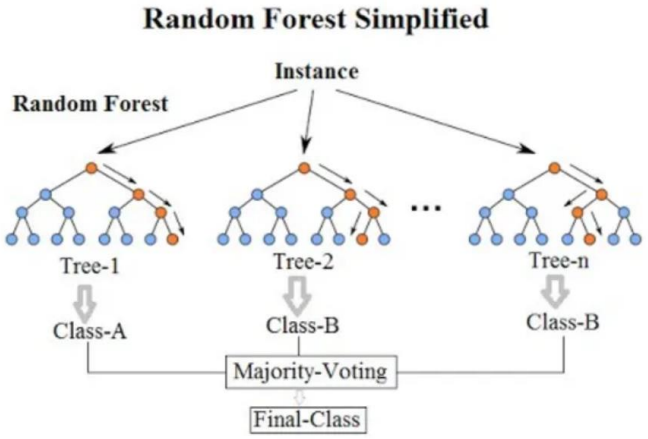

Figure 15. Random Forest Model extracted from [55].

In the context of deepfake detection, Random Forest emerges as an optimal choice due to its capacity to effectively handle the intricacies and nuances associated with distinguishing between authentic and manipulated media [34, 55]. By leveraging 250 decision trees, carefully selected based on their collective performance across various iterations, the model achieves heightened accuracy, as verified by the results exhibited in Table 6.

Table 6 Accuracy of Random Forest with multiple estimators

Count of Estimators	Accuracy
10	0.9854
50	0.9878
100	**0.9884**
150	0.9879
200	0.9878
250	0.9881

4.3.2 Support Vector Machine (SVM)

Support Vector Machine is a smart computer tool used for categorizing images into different groups. It does this by creating a boundary, which could be a line or a more complex shape, to distinguish between various sets of image points as effectively as it can. [53]. Imagine having different colored balls scattered on a table, and you want to draw lines between them so that each color is as far apart as possible. That's like what SVM does in higher-dimensional spaces [53].

The goal of SVM is to identify the best possible line or surface (called a hyperplane) that separates the different groups while enhancing the space, or margin, between them. This maximization of space makes the model more robust and better at generalizing to new, unobserved data. Figure 18 helps in understanding the 2-dimensional classification task.

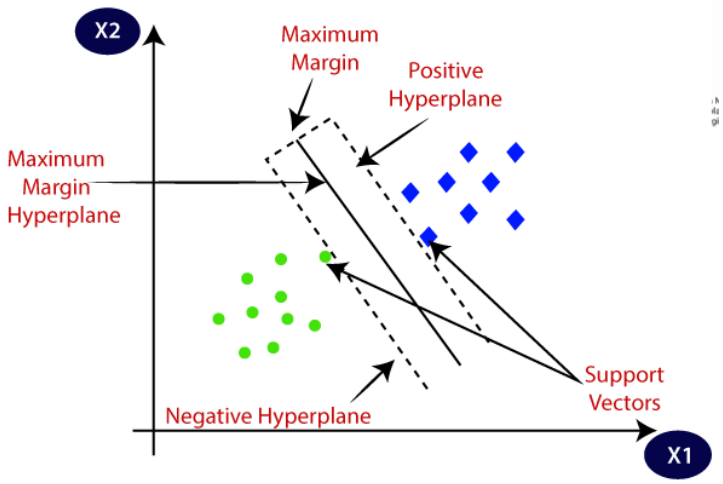

Figure 16. Visual representation of SVM model image extracted from [52]

For my thesis purpose, I chose SVM because it is known for handling complex, high-dimensional data well [53]. I used a specific setting called the radial basis function (RBF) kernel, which is a mathematical formula that helps SVM create the best possible separation between different data points. I found that this setting worked best for our purposes after experimenting with different options.

The initial step involves preparing my data for analysis and training my model to recognize patterns effectively. This process begins by segmenting the data into two primary components: X and Y variables. The X variable encompasses all the features present in our dataset, while the Y variable symbolizes the labels denoting whether the data corresponds to real or fake images.

Following the data organization, I proceed to divide the data into training and testing subsets. Through experimentation, I determined that allocating 80% of the data for training and reserving the remaining 20% for testing yielded optimal results. This split ratio was selected after exploring various combinations, including 70/30, 75/25, and

90/10, with the 80/20 split consistently outperforming the others. Moreover, to ensure fairness and reliability, the data split is randomized for each run of the program. Once the model is trained on the training dataset, its performance is evaluated using key metrics such as accuracy, precision, recall, and the F1 score. These metrics provide insights into the model's ability to correctly identify images as real or fake and highlight the efficacy of different features in contributing to its performance. Based on these evaluations, adjustments are made to refine the model, focusing on features such as the number of estimators and the choice of kernel.

Overall, this detailed methodology ensures a systematic and thorough exploration of the dataset and the model's performance.

4.3.3 MesoNet

MesoNet is a DL based approach which is designed to efficiently identify face tampering in the media. It especially targets the detection of hyper realistic forged medias. Traditional image forensics techniques often falter with videos due to compression that significantly weakens the data [33]. MesoNet proposes a solution through a deep learning model that focuses on the meso-level properties of images, boasting a high classification rate.

MesoNet comprises two networks: Meso-4 and MesoInception-4, both characterized by a low number of layers aimed at analyzing mesoscopic features of images for forgery detection [33].

Meso-4 starts with four layers of convolutions and pooling followed by a dense network with one hidden layer. The convolutional layers use ReLU activation functions and batch normalization to regularize their output and prevent the vanishing gradient effect, while

the fully connected layers employ dropout to enhance robustness [33]. It has 27,977 trainable parameters. Figure 19 on page number 52 shows the network architecture of the model. Where the layers and the parameters are in the boxes whereas the output sizes are next to the arrow sign.

MesoInception-4 modifies the initial structure by incorporating a version of the inception module at the beginning, swapping the first two convolutional layers. This increases the function space in which the model is maximized, using dilated convolutions to handle multi-scale information. This architecture has 28,615 trainable parameters. Figure 20 on page number 53 shows the network architecture of the MesoInception-4 model.

The process involves identifying fabricated data through deep learning networks that analyze the mesoscopic level of image details. Given that microscopic analyses (e.g., image noise) are ineffective in the compressed video context and higher-level semantic analyses are easily fooled, especially with human faces, MesoNet employs an intermediate approach. The training involves successive batches of images undergoing slight transformations to improve generalization and robustness, using the ADAM optimizer for weight optimization.

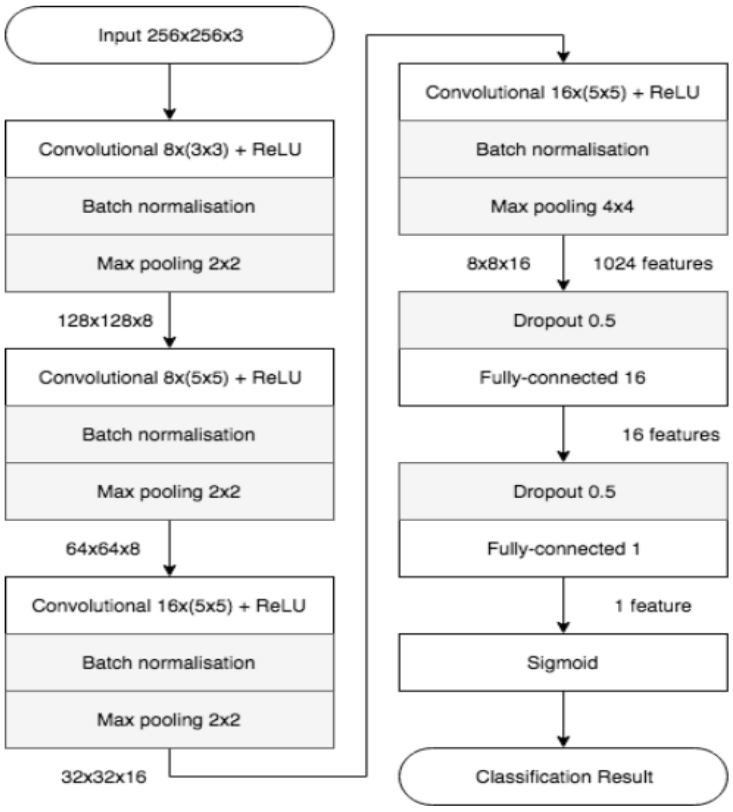

Figure 17. The network architecture of Meso-4 [33]

For classification, MesoNet utilizes a relatively simple network structure that has demonstrated surprisingly effective results, attributing its success to focusing on mesoscopic properties rather than attempting to analyze either very high-level semantic details or low-level noise patterns. Image aggregation techniques are also explored to enhance detection rates further by leveraging multiple frames from the same video, effectively increasing the overall detection accuracy.

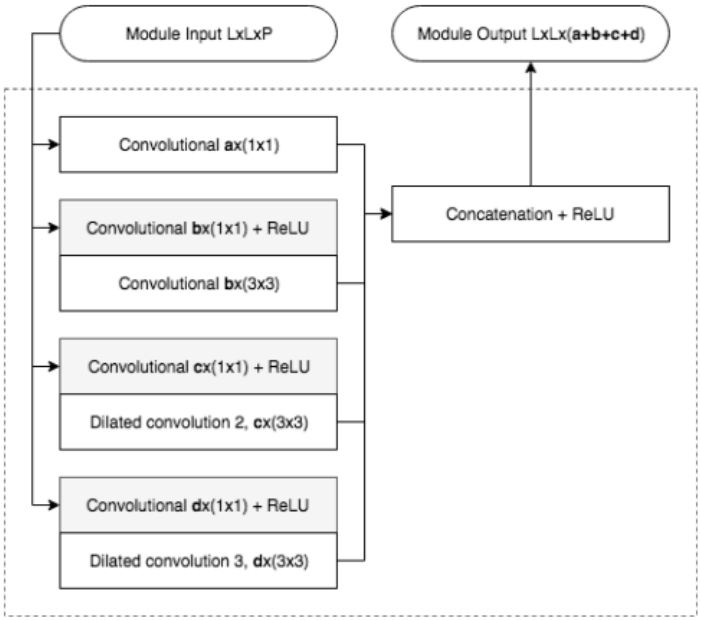

Figure 18. Network architecture utilizing inception modules in MesoInception-4 [33]

In my process, we identify 'X' as the collection of input data that I feed into my system and 'Y' as the corresponding set of true values I aim to predict. The relationship between 'X' and 'Y' is assessed by the function 'f', which acts as a predictor within our chosen algorithm, making educated guesses on the nature of the inputs. I have trained my networks using a library named Keras, renowned for its efficacy in implementing neural networks with Python. The networks learned through repeated trials with small groups of images, subtly tweaking their internal settings to improve their predictive accuracy using a method known as ADAM optimization [33]. I started with a base learning rate of 0.0001 incrementally decreasing after every 1000 iteration until down to 0.000001. Which helps in generalization and robustness. Moreover, to verify that my MesoNet doesn't just memorize the images but understands the underlying patterns, for that reason some random manipulations were made to the image data like zoom, and shift in brightness.

This strategy strengthens the MesoNet model, helping it to perform well not just with the images it has seen but also with new ones it hasn't encountered before.

Chapter 5

Results

In this part of the discussion, I explored the results from three different approaches based on ML and DL that I have put to the test for classifying deepfakes. The main aim I am addressing with this work is to craft and refine these approaches to recognize these sophisticated fake media effectively. I gathered more than 7,000 images, gave them a digital makeover to highlight the telling signs I am after, and then put my three approaches to work: the random forest algorithm, the support vector machine, and MesoNet. Each mentioned model was trained on 80% of the dataset and then tested on the remaining 20% to see how well they could identify the fakes.

I did not just overlook the model's performance ; but I have measured their success with a bunch of different benchmarks: how often they're on the mark (accuracy), how much they can be trusted (precision), how good they are at catching the fakes (recall), and their overall detective prowess (F1 score). To understand this measurement closely, I used some handy visual helpers—a confusion matrix and a curve that shows their trade-off between identifying all the fakes without raising the concerns.

To draw this curve for each method, I used a smart piece of code from a library called sklearn.metrics. This code will need two pieces of information: whether an image is real or fake and how suspicious the tool is about each image. Then, I told it which images were real. This gave me a bunch of points showing how often my models might make mistakes or catch fake images as I change how careful they are. These points started from being very careful, where they would rather miss a fake than accuse something

innocent, to being very cautious, where model would rather accuse everything just to be safe. Going through these steps helps me see how good my methods are at different levels of caution.

The random forest model performed impressively well, achieving an accuracy of 98.86%, precision of 98.89%, recall of 98.66%, and F1 score of 97.85%. Figure 21 displays the confusion matrix, while Figure 21 showcases the ROC curve for the random forest model's outcomes. In the confusion matrix depicted in Figure 22 on page number 57, there are 8241 true positives and 8559 true negatives, showing the random forest's correct predictions. However, there were 92 false positives, suggesting that the model sometimes incorrectly predicted label 1. In contrast, there were 104 false negatives, indicating instances where the model missed predicting label 1 when it should have.

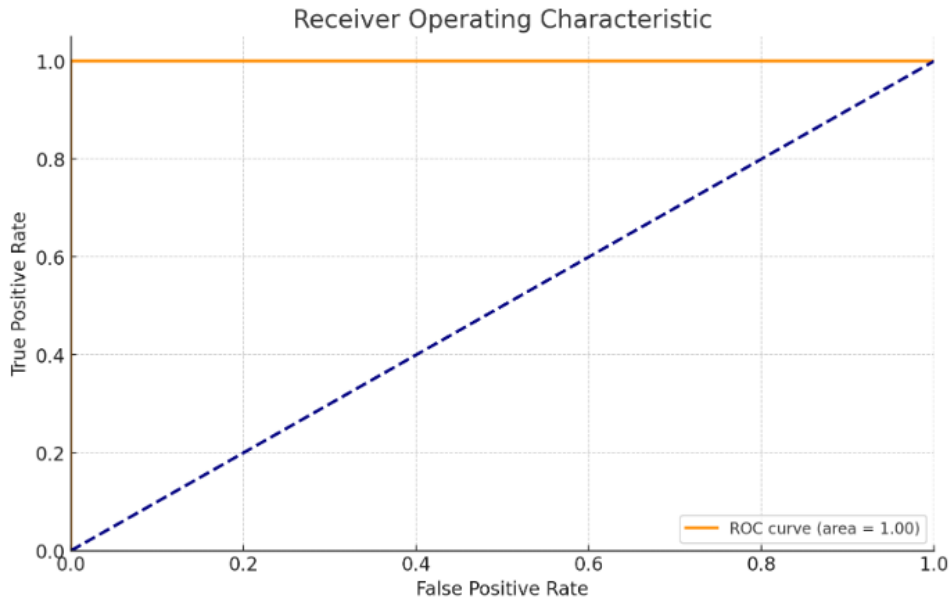

Figure 21. ROC curve for the Random Forest Shows the perfect classifier.

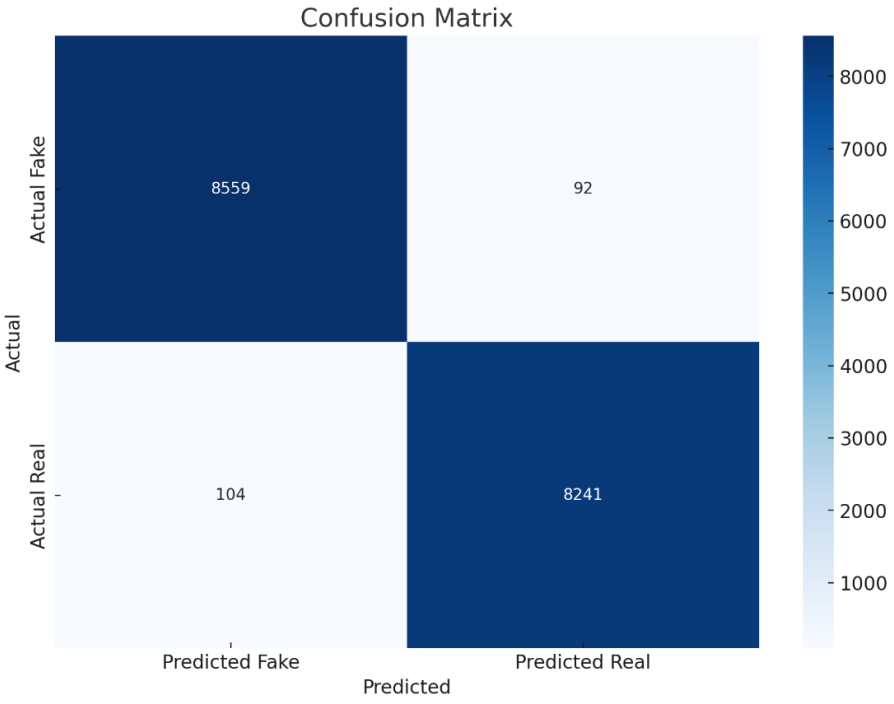

Figure 22. Confusion Matrix for Random Forest.

The SVM model did well, with an accuracy of 84.49%, precision of 84.10%, recall of 84.43%, and F1 score of 84.273%. Figure 23 on page number 58 shows the confusion matrix, and Figure 24 on page number 58 displays the ROC curve for the SVM model's performance. The confusion matrix shown in Figure 23 displays 7043 true positives and 7325 true negatives. However, the model came across a count of 1326 false positives, which indicates over-prediction of label 1 in some cases. Contrast to that the false negatives we comparatively to 1302.

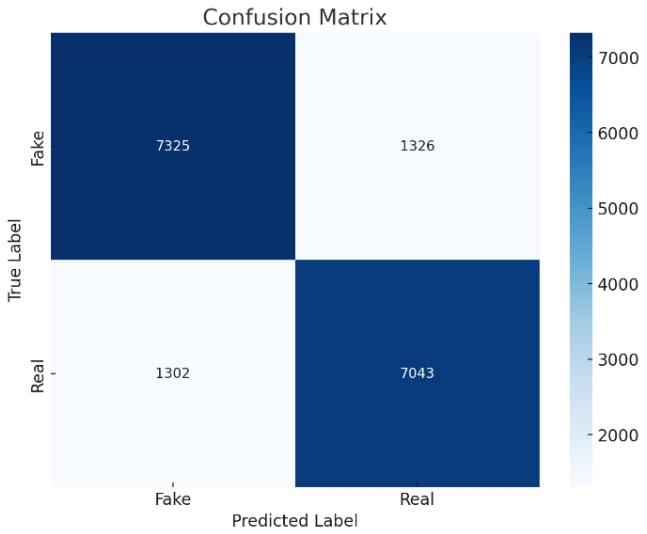

Figure 19. Confusion matrix for SVM model.

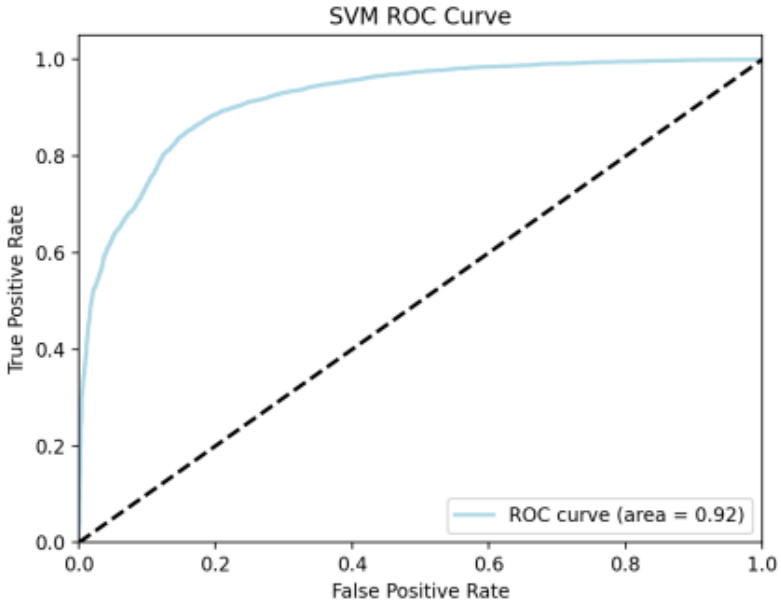

Figure 20. ROC curve for the SVM model.

MesoNet model, which was based on deep learning did pretty well, In the evaluation of MesoNet to correctly identify the fraudulent media in the WildDeepfake dataset [24], the

Meso-4 network achieved an average accuracy of 89.1%, while the MesoInception-4 network reached an accuracy of 91.7%. Meso-4 has an AUC score of 0.946 which recommends that model performed reasonably good in classifying the deepfakes. Nonetheless, the MesoInception-4 has an AUC score of 0.968.which interpreted that the model performed even better to Meso-4 in the identification of the fraudulent media. Thus, higher the AUC score value implies that the MesoInception-4 is more effective and efficient in predicting the deepfakes contrary to Meso-4.

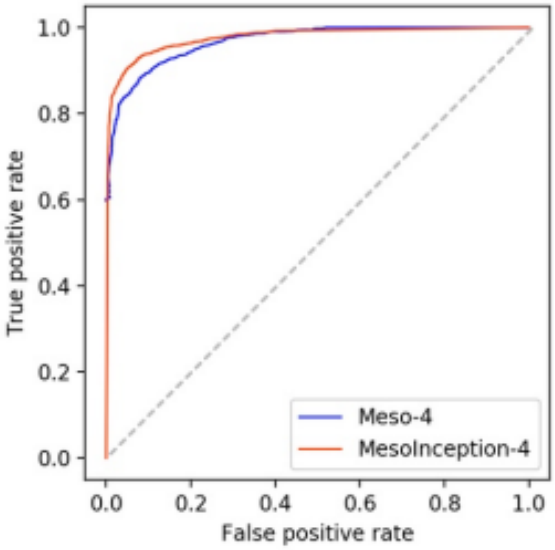

Figure 21. Roc curve for the MesoNet model.

Chapter 6

Analysis

Looking closely at the results in chapter 5, it's clear that the models excelled in identifying deepfakes. All three models which includes SVM, Random Forest, and MesoNet achieved high accuracy rates, even when compared to datasets that were considered easier to analyze. Notably, they outperformed previous state-of-the-art models designed for this explicit dataset, as seen in Tables 6 and Table 7 on page number 61.

When I examined the performance of the Random Forest and SVM models, the random forest model emerged as the top performer across various metrics like accuracy, precision, recall, and F1 score. Its strength lies in its ability to handle complex datasets with numerous features and intricate relationships between them. Whereas, the SVM model showcased lower performance, partly due to its sensitivity to hyperparameters and susceptibility to overfitting when dealing with large feature sets.

It's essential to consider that each model has its strengths and weaknesses. While random forest excels with complex data, Similarly, SVM can be effective with small, linearly separable datasets but may struggle with larger and more intricate data.

Table 7. Results of all the Approaches used in this study.

Dataset	Approaches	Accuracy
WildDeepfake	R*andom Forest*	98.86%
WildDeepfake	Meso-4	89.1 %
WildDeepfake	MesoInception-4	91.1 %

WildDeepfake	SVM	84.52%

Table 8. Comparative analysis of various models on WildDeepfake Dataset.

Title	Dataset	Method	Accuracy
FedForgery: Generalized Face Forgery Detection with Residual Federated Learning [56]	WildDeepfake	FedForgery	68.03 %
Improved Xception with Dual Attention Mechanism and Feature Fusion for Face Forgery Detection [57]	WildDeepfake	Dual Attention Mech Xception	83.32%
Spatiotemporal Inconsistency Learning for DeepFake Video Detection [30]	WildDeepfake	STIL	84.12%
Exploiting Fine-grained Face Forgery Clues via Progressive Enhancement Learning [58]	WildDeepfake	Progressive Enhancement Learning	84.14%
Fighting Deepfake by Exposing the Convolutional Traces on Images [59]	WildDeepfake	RECCE	83.25%
Identity Mappings in Deep Residual Networks [60]	WildDeepfake	ResNetV2-50	63.99%
Xception: Deep Learning with Depthwise Separable Convolutions [61]	WildDeepfake	XceptionNet	69.25%
Methods of deepfake detection based on machine learning [7]	Celeb-DF	DenseNet169 + Rayleigh Blur	60 %
DeepVision: Deepfakes detection using eye blinking pattern [11]	Eye Blinking Prediction Dataset from Kaggle	DeepVision with integrity verification	87.5%

Deepfakes creation and detection using deep learning [40]	Online dataset containing 5000 images.	MesoNet with DFDNet image enhancer	80%

Overall, these findings undermine the significant impact of model selection on classification performance. The random forest model's proficiency in handling critical data and feature interactions positions it as a strong contender for various categorization tasks. Nonetheless, it is very crucial to realize the relative strength of respective methods and use them with appropriate dataset. Moreover, the performance of the MesoNet deep learning model stands out significantly, surpassing others trained on different datasets. When comparing all three models, random forest demonstrated superior accuracy, followed by MesoNet and then SVM as shown in the Table 7 on page number 60.

6.1 Contribution

During the early stages of my research, I explored deeply into the domain of deepfakes. I thoroughly analyzed various deep learning and machine learning models used in different fields, putting them through their paces on specific datasets to gauge their effectiveness. This involved a meticulous examination and gathering of valuable insights from relevant research papers. To better understand the complex inner workings of these models, I methodically broke down their components. The findings of this extensive investigation resulted in a research paper titled "Leveraging Deep Learning Approaches for Deepfake Detection," which was published in the esteemed Archives of the 7th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence in 2023 by the

Association of Computing Machinery (ACM). You can find this paper on Google Scholar and the ACM library.

The prime goal of this thesis study is to figure out if we could nail down those sneaky morphed videos, aka deepfakes, with accuracy. I was curious about how good machine learning and deep learning algorithms are at telling them apart from the real ones. So, I dove in to see just how well they could do it. In the beginning, there were quite a few challenges in figuring out how to manage the data. Specifically, I needed to extract it properly from the original folder, which was organized in a rather complex way. To tackle this extraction hurdle, I crafted a code. This code smoothly handles various data extraction tasks using different functions, as detailed in Table 2 on page 34. Each function mentioned in the table takes data from one folder and moves it to another. In simpler terms, the output of one function becomes the input for the next, resulting in a neatly organized dataset ready for the feature extraction process.

By reading various literature and understanding the pre-existing models, I chose to work with SVM, random forest and MesoNet because SVM and Random Forest are favored for their robustness in handling high-dimensional data, flexibility in feature representation, and ability to prevent overfitting, crucial for distinguishing between genuine and deepfake images. MesoNet, purposefully designed for deepfake detection, leverages its specialized CNN architecture to efficiently capture subtle manipulation artifacts, ensuring high detection accuracy. Moreover, MesoNet strikes a balance between computational efficiency and performance, making it suitable for real-time or large-scale deepfake detection tasks. By integrating SVM, Random Forest, and MesoNet into my

project framework, I have harness their complementary strengths to enhance the overall effectiveness and robustness of my deepfake detection system across diverse scenarios and variations, thus contributing significantly to the field of deepfake detection and mitigation efforts. Each algorithm underwent a bit of fine-tuning regarding how it perceives images. For the machine learning models, I made tailored adjustments to handle arrays of features during both training and testing phases. Meanwhile, MesoNet stands out with its capability to extract features directly from images, streamlining the process without the need for extra preprocessing steps. However, we did encounter some minor challenges, particularly regarding image input and output sizes, which needed tweaking to better suit the unique characteristics of the WildDeepfake dataset.

Chapter 7

Conclusion

As deepfake technology becomes more accessible, detecting fake videos is becoming increasingly challenging. This is concerning because spotting deepfakes is crucial for keeping information reliable in today's digital world. With deepfake technology getting more advanced, it's easier to manipulate images and videos, spreading false info. This can damage trust in real news sources and even cause problems in areas like politics and security. Since we rely so much on social media, we can expect to see more deepfakes, making detecting them even more important.

In this research work, I have looked at different ways to find deepfakes using machine learning and deep learning methods. I discovered that techniques involving machine learning algorithms tend to be better at telling real from fake media. However, deep learning algorithm is significantly closer. By improving how I detect deepfakes, I can make sure the information I see is accurate and trustworthy, protecting against harmful uses of deepfake.

Both machine learning and deep learning models have done well in spotting deepfakes, showing they are reliable approaches for detection purposes. However, it is machine learning based random forest model outsmarted the other two models used in this study. But it's essential to recognize their limitations and keep looking for ways to make them better.

7.1 Limitations

In cutting edge era, spotting fake videos and images, known as deepfakes, is a big challenge. The methods I have for telling if a video is fake often rely on how it was made. For example, one model might work well at finding fake videos made with a certain technique but struggle with others. Most of the time, I try to spot fake videos by looking at facial features and how things move in the video. But for really good fake videos, these methods don't always work as well. Sometimes, the models can't detect fake videos if the face is straight towards the camera because models can't pick up on subtle changes in color or quality.

Researchers say that these models have some big limitations. They struggle to handle new ways of making deepfakes, defend against attacks, and explain why they think a video is fake. Plus, models don't always work well with real-world deepfakes, especially if they are made in a way that the model hasn't seen before.

One concern with this study is that it only used WildDeepfake dataset. Besides the dataset tried to cover a wide range of fake videos, using more datasets would help check if the models can spot deepfakes in different situations. Also, the study only looked at a few machine learning and deep learning methods.

Even though this study has some limitations, machine learning methods have revealed significant results in identifying deepfakes. Compared to deep learning methods, these basic machine learning models are simpler to understand and don't need as much computer power. By knowing what these models can and can't do and by trying out new

methods, I can keep improving at finding deepfakes and stopping them from causing harm.

7.2 Future Work

Moving forward, my aim is to broaden our research scope to encompass both image and video datasets created using various methods, such as GANs or Autoencoders. By doing so, I can ensure that our models are not limited to specific deepfake generation techniques. Additionally, future investigations should delve into the cost-effectiveness of building these models and explore optimal approaches for detecting inconsistencies across the entire face, regardless of specific facial regions. However, such endeavors will require extensive knowledge and in-depth study to progress further.

While my current work has shown promise in identifying deepfakes within a high-quality dataset, there remains a vast landscape to explore in this field. To move the field forward, we need to tackle current problems and look into what we can study next.

One crucial domain to focus on is applying machine learning and deep learning methods to diverse datasets. For instance, the DFDC dataset offers a wealth of opportunities due to its extensive range of visual actors, picture qualities, and creation methods. Conducting cross-dataset evaluations will enhance the robustness and generalization of our models, ultimately resulting in more effective detection of real deepfakes.

Moreover, investing time and effort into feature extraction is paramount. By leveraging the rich array of features available in libraries like scikit-image, I can better

interpret their value for detecting deepfake. Investigating various spatial, frequency, and biological features will equip our models to detect a wide range of deepfakes effectively.

Staying abreast of research efforts is crucial as the proliferation of fake content continues unabated on the internet. With the advancement of AI-generated media, discerning truth from fiction online may become increasingly challenging. Through ongoing research endeavors, we aim to stay at the forefront of AI development and uphold the integrity of digital content.

To summarize, it's vital to recognize the ongoing battle between deepfake generation and detection methods. While deepfakes are transforming increasingly convincing, relying solely on complex detection methods may not suffice. Instead, prioritizing a deeper understanding of models and utilizing simpler tactics for detection is key. While our research marks a step in this direction, further efforts are needed to combat deepfakes and fake content effectively.

References

[1] Harsh Agarwal. 2021. Deepfake Detection Using SVM. 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC) (2021). DOI:https://doi.org/10.1109/icesc51422.2021.9532627

[2] Jacob Mallet, Laura Pryor, Rushit Dave, Naeem Seliya, Mounika Vanamala, and Evelyn Sowells-Boone. 2022. Hold on and swipe: A touch-movement based continuous authentication schema based on machine learning. 2022 Asia Conference on Algorithms, Computing and Machine Learning (CACML) (2022). DOI:http://dx.doi.org/10.1109/cacml55074.2022.00081

[3] Faten F. Kharbat, Tarik Elamsy, Ahmed Mahmoud, and Rami Abdullah. 2019. Image Feature Detectors for Deepfake Video Detection. 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA) (2019). DOI:https://doi.org/10.1109/aiccsa47632.2019.9035360

[4] J C Dheeraj, Krutant Nandakumar, A V Aditya, B S Chethan, and G C R Kartheek. 2021. Detecting Deepfakes Using Deep Learning. 2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT) (2021). DOI:https://doi.org/10.1109/rteict52294.2021.9573740

[5] Nyle Siddiqui, Rushit Dave, Mounika Vanamala, and Naeem Seliya. 2022. Machine and deep learning applications to mouse dynamics for Continuous User Authentication. Machine Learning and Knowledge Extraction 4, 2 (2022), 502–518. DOI:http://dx.doi.org/10.3390/make4020023

[6] Zach DeRidder, Nyle Siddiqui, Thomas Reither, Rushit Dave, Brendan Pelto, Mounika Vanamala and Naeem Seliya, “Continuous User Authentication Using Multi-Finger Mobile Touch Dynamics Based on Machine Learning”, 9th International Conference on Soft Computing & Machine

Intelligence, IEEE 2022

[7] Artem A. Maksutov, Viacheslav O. Morozov, Aleksander A. Lavrenov, and Alexander S. Smirnov. 2020. Methods of Deepfake Detection Based on Machine Learning. 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) (2020). DOI:https://doi.org/10.1109/eiconrus49466.2020.9039057

[8] Neeraj Guhagarkar, Sanjana Dessai, Swanand Vaishyampayam and Ashwini save. 2021. Deepfakes detection techniques: A review. 2021 VIVA-Tech IJRI(2021).

[9] Jihyeon Kang, Sang-Keun Ji, Sangyeong Lee, Daehee Jang, and Jong-Uk Hou. 2022. Detection Enhancement for Various Deepfake Types Based on Residual Noise and Manipulation Traces. IEEE Access 10, (2022), 69031-69040. DOI:https://doi.org/10.1109/access.2022.3185121

[10] Deng Pan, Lixian Sun, Rui Wang, Xingjian Zhang, and Richard O. Sinnott. 2020. Deepfake Detection through Deep Learning. 2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT) (2020). DOI:https://doi.org/10.1109/bdcat50828.2020.00001

[11] Tackhyun Jung, Sangwon Kim, and Keecheon Kim. 2020. DeepVision: Deepfakes Detection Using Human Eye Blinking Pattern. IEEE Access 8, (2020), 83144-83154. DOI:https://doi.org/10.1109/access.2020.2988660

[12] Sreeraj Ramachandran, Aakash Varma Nadimpalli, and Ajita Rattani. 2021. An Experimental Evaluation on Deepfake Detection using Deep Face Recognition. 2021 International Carnahan Conference on Security Technology (ICCST) (2021). DOI:https://doi.org/10.1109/iccst49569.2021.9717407

[13] Jeffrey T. Hancock and Jeremy N. Bailenson. 2021. The Social Impact of Deepfakes. Cyberpsychology, Behavior, and Social Networking 24, 3 (2021), 149-152.

DOI:https://doi.org/10.1089/cyber.2021.29208.jth

[14] Wei Tee, Rushit Dave, Naeem Seliya, Mounika Vanamala, " A close look into Human Activity Recognition models using Deep Learning, 2nd International Conference on Information Technology and Cloud Computing IEEE 2022.

[15] Wei Tee, Rushit Dave, Naeem Seliya, Mounika Vanamala "Human Activity Recognition using Mobile Sensors with a focus on Machine Learning" Asia Conference on Algorithms, Computing & Machine Learning, IEEE, 2022.

[16] Nyle Siddiqui, Rushit Dave, Mounika Vanamala, and Naeem Seliya. 2022. Machine and deep learning applications to mouse dynamics for Continuous User Authentication. Machine Learning and Knowledge Extraction 4, 2 (2022), 502–518. DOI:http://dx.doi.org/10.3390/make4020023

[17] David Guera and Edward J. Delp. 2018. Deepfake Video Detection Using Recurrent Neural Networks. 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS) (2018). DOI:https://doi.org/10.1109/avss.2018.8639163

[18] Yuezun Li, Ming-Ching Chang, and Siwei Lyu. 2018. In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking. 2018 IEEE International Workshop on Information Forensics and Security (WIFS) (2018). DOI:https://doi.org/10.1109/wifs.2018.8630787

[19] Mounika Vanamala, Xiaohong Yuan, and Kanishka Bandaru. 2019. Analyzing CVE Database Using Unsupervised Topic Modelling. 2019 International Conference on Computational Science and Computational Intelligence (CSCI) (2019). DOI:https://doi.org/10.1109/csci49370.2019.00019

[20] Mounika Vanamala, Xiaohong Yuan, and Kaushik Roy. 2020. Topic Modeling And Classification Of Common Vulnerabilities And Exposures Database. 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD) (2020). DOI:https://doi.org/10.1109/icabcd49160.2020.9183814

[21] Mounika Vanamala, Xiaohong Yuan, Walter Smith and Joi Bennett. 2022. Interactive Visualization Dashboard for Common Attack Pattern Enumeration Classification. ICSEA 2022 : The Seventeenth International Conference on Software Engineering Advances (ICSEA)(2022).

[22] Aakash Varma Nadimpalli and Ajita Rattani. 2022. On improving cross-dataset generalization of deepfake detectors. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)(2022). DOI:https://doi.org/10.48550/arXiv.2204.04285

[23] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang and Cristian Canton Ferrer. 2020. The deepfake detection challenge (dfdc) dataset. arXiv:2006.07397v4. Retrieved from https://arxiv.org/abs/2006.07397

[24] Bojia Zi, Minghao Chang, Jingjing Chen, Xingjun Ma, and Yu-Gang Jiang. 2020. WildDeepfake. Proceedings of the 28th ACM International Conference on Multimedia (2020). DOI:https://doi.org/10.1145/3394171.3413769

[25] Brown, Sara. "Machine Learning, Explained." MIT Sloan, 21 Apr. 2021, mitsloan.mit.edu/ ideas-made-to-matter/machine-learning-explained.

[26] "Machine Learning: A Quick Introduction and Five Core Steps." Centric Consulting, 8 Dec. 2022, https://centricconsulting.com/blog/machine-learning-a-quick-introduction -and-five-core-steps/.

[27] Deep Learning vs. Machine Learning: Beginner's Guide." Coursera, https://www.coursera.org/articles/ai-vs-deep-learning-vs-machine-learning-beginners-guid

[28] Papers with Code, https://paperswithcode.com/.

[29] Aya Ismail, Marwa Elpeltagy, Mervat S. Zaki, and Kamal Eldahshan. 2021. A New Deep Learning-Based Methodology for Video Deepfake Detection Using XGBoost. Sensors 21, 16 (2021), 5413. DOI:https://doi.org/10.3390/s21165413

[30] Juefei-Xu, Felix, et al. "Countering Malicious Deepfakes: Survey, Battleground, and Horizon." International Journal of Computer Vision, 23 Mar. 2022, https://arxiv.org/abs/2103.00218v3.

[31] Oscar de Lima, Sean Franklin, Shreshtha Basu, Blake Karwoski and Annet George.2020. Deepfake Detection using Spatiotemporal Convolutional Networks. arXiv:2006.14749. Retrieved from https://doi.org/10.48550/arXiv.2006.14749

[32] Gu, Zhihao, et al. "Spatiotemporal Inconsistency Learning for Deepfake Video Detection." Proceedings of the 29th ACM International Conference on Multimedia (MM '21), 11 Oct. 2021, https://arxiv.org/abs/2109.01860v3.

[33] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. 2018. MesoNet: a Compact Facial Video Forgery Detection Network. 2018 IEEE International Workshop on Information Forensics and Security (WIFS) (2018). DOI:https://doi.org/10.1109/wifs.2018.8630761

[34] "What Is a Random Forest?" TIBCO Software, https://www.tibco.com/reference-center/ what-is-a-random-forest.

[35] Nikita S. Ivanov, Anton V. Arzhskov, and Vitaliy G. Ivanenko. 2020. Combining Deep Learning and Super-Resolution Algorithms for Deep Fake Detection. 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) (2020). DOI:https://doi.org/10.1109/eiconrus49466.2020.9039498

[36] Bonettini, Nicolò, et al. "Video Face Manipulation Detection through Ensemble of CNNS." 2020 25th International Conference on Pattern Recognition (ICPR), 16 Apr. 2020, https://arxiv.org/abs/2004.07676.

[37] Barni, Mauro, et al. "CNN Detection of Gan-Generated Face Images Based on Cross-Band Co-Occurrences Analysis." IEEE Workshop on Information Forensics and Security, 2 Oct. 2020, https://arxiv.org/abs/2007.12909.

[38] Li, Yuezun, et al. "Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics." 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 27 Sept. 2019, https://arxiv.org/abs/1909.12962.

[39] Sonya J. Burroughs, Balakrishna Gokaraju, Kaushik Roy, and Luu Khoa. 2020. DeepFakes Detection in Videos using Feature Engineering Techniques in Deep Learning Convolution Neural Network Frameworks. 2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR) (2020). DOI:https://doi.org/10.1109/aipr50011.2020.9425347

[40] Hady A. Khalil and Shady A. Maged. 2021. Deepfakes Creation and Detection Using Deep Learning. 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC) (2021). DOI:https://doi.org/10.1109/miucc52538.2021.9447642

[41] Prajit Ramachandran, Barret Zoph, Quoc V. Le.2017. Searching for Activation Function. arXiv:1710.05941. Retrieved from https://doi.org/10.48550/arXiv.1710.05941

[42] Bahar Uddin Mahmud, Afsana Sharmin. 2021. Deep Insiths of Deepfake Technology: A Review. arXiv:2105.00192. Retrieved from https://doi.org/10.48550/arXiv.2105.00192

[43] Ali Raza, Kashif Munir, and Mubarak Almutairi. 2022. A Novel Deep Learning Approach for Deepfake Image Detection. Applied Sciences 12, 19 (2022), 9820. DOI:https://doi.org/10.3390/app12199820

[44] Priti Yadav, Ishani Jaiswal, Jaiprakash Marvari, Vibhash Choudhary and Gargi Khanna.2021. International Conference on Emerging Technologies:AI.IO T, and CPS for Science Technology Applications (2021)

[45] Pallabi Saikia, Dhwani Dholaria, Priyanka Yadav, Vaidehi Patel, and Mohendra Roy. 2022. A Hybrid CNN-LSTM model for Video Deepfake Detection by Leveraging Optical Flow Features. 2022 International Joint Conference on Neural Networks (IJCNN) (2022). DOI:https://doi.org/10.1109/ijcnn55064.2022.9892905

[46] Vedant Jolly, Mayur Telrandhe, Aditya Kasat, Atharva Shitole, and Kiran Gawande. 2022. CNN based Deep Learning model for Deepfake Detection. 2022 2nd Asian Conference on Innovation in Technology (ASIANCON) (2022). DOI:https://doi.org/10.1109/asiancon55314.2022.9908862

[47] Junke Wang, Zuxuan Wu, Wenhao Ouyang, Xintong Han, Jingjing Chen, Yu-Gang Jiang, and Ser-Nam Li. 2022. M2TR: Multi-modal Multi-scale Transformers for Deepfake Detection. Proceedings of the 2022 International Conference on Multimedia Retrieval (2022). DOI:https://doi.org/10.1145/3512527.3531415

[48] Alakananda Mitra, Saraju P. Mohanty, Peter Corcoran, and Elias Kougianos. 2020. A Novel Machine Learning based Method for Deepfake Video Detection in Social Media. 2020 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS) (2020). DOI:https://doi.org/10.1109/ises50453.2020.00031

[49] Pallabi Saikia, Dhwani Dholaria, Priyanka Yadav, Vaidehi Patel, and Mohendra Roy. 2022. A Hybrid CNN-LSTM model for Video Deepfake Detection by Leveraging Optical Flow Features. 2022 International Joint Conference on Neural Networks (IJCNN) (2022). DOI:https://doi.org/10.1109/ijcnn55064.2022.9892905Deep

[50] Yuezun Li, Ming-Ching Chang, and Siwei Lyu. 2018. In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking. 2018 IEEE International Workshop on Information Forensics and Security (WIFS) (2018). DOI:https://doi.org/10.1109/wifs.2018.8630787

[51] "Scikit-Learn Machine Learning in Python." Scikit-Learn, https://scikit-learn.org/stable/index.html

[52] Jiaying Wang, Yongfeng Qi, Jinlin Hu, and Jihong Hu. 2022. Face forgery detection with a fused attention mechanism. 2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA) (2022). DOI:https://doi.org/10.1109/cvidliccea56201.2022.9824499

[53] "Scikit-Image's Image Processing in Python." Scikit-Image, https://scikit-image.org/.

[54] "Scikit-Learn Machine Learning in Python." Scikit-Learn, https://scikit-learn.org/stable/index.html

[55] https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d

[56] Liu, Decheng, et al. “Fedforgery: Generalized Face Forgery Detection with Residual Federated Learning.” Conference on Artificial Intelligence, 18 Oct. 2022, https://arxiv.org/abs/2210.09563.

[57] Lin, Hao, et al. “Improved Xception with Dual Attention Mechanism and Feature Fusion for Face Forgery Detection.” 2022 4th International Conference on Data Intelligence and Security (ICDIS), 29 Sept. 2021, https://arxiv.org/abs/2109.14136v1.

[58] Gu, Qiqi, et al. “Exploiting Fine-Grained Face Forgery Clues via Progressive Enhancement Learning.” Proceedings of the AAAI Conference on Artificial Intelligence, 28 Dec. 2021, https://arxiv.org/abs/2112.13977.

[59] Guarnera, Luca, et al. “Fighting Deepfake by Exposing the Convolutional Traces on Images.” IEEE Access, 7 Aug. 2020, https://arxiv.org/abs/2008.04095.

[60] He, Kaiming, et al. “Identity Mappings in Deep Residual Networks.” NASA/ADS, ui.adsabs.harvard.edu/abs/2016arXiv160305027H/abstract.

[61] Chollet, François. “Xception: Deep Learning with Depthwise Separable Convolutions.” CVPR, 4 Apr. 2017, arxiv.org/abs/1610.02357

Appendix A

Code for Data Organization/Processing

The code in the appendix organizes the data, post which is ready for the feature extraction phase.

A.1 Data Extraction

```
# Following piece of code read a folder for the tar file, extracts the

# Folder with its material and deletes the all the “.tar.gz” file

import os

import tarfile

import shutil


def copy_tar_gz_files(main_folder, destination_folder):

    # Check if the main folder exists

    if not os.path.exists(main_folder):

        print (f"Main folder '{main_folder}' does not exist.")

        return


    # Create destination folder if it doesn't exist

    os.makedirs(destination_folder, exist_ok=True)


    # Iterate through each item in the main folder

    for item in os.listdir(main_folder):

        item_path = os.path.join(main_folder, item)

        # Check if the item is a directory (subfolder)

        if os.path.isdir(item_path):

            # Recursively call the function to copy .tar.gz files in subfolders

            copy_tar_gz_files(item_path, destination_folder)

        elif item.endswith('.tar.gz'):
```

```
            # Copy the .tar.gz file to the destination folder
            shutil.copy(item_path, destination_folder)
            print(f"Copied '{item}' to '{destination_folder}'")

def extract_tar_files_recursive(destination_folder, extract_folder):
    # Check if the main folder exists
    if not os.path.exists(destination_folder):
        print(f"Main folder '{main_folder}' does not exist.")
        return

    # Iterate through each item in the main folder
    for item in os.listdir(destination_folder):
        item_path = os.path.join(destination_folder, item)
        # Check if the item is a directory

        if item.endswith('.tar.gz'):
            # Extract the .tar.gz file
            print(f"\nExtracting files from '{item}':")
            os.makedirs(extract_folder, exist_ok=True)
            with tarfile.open(item_path) as tar:
                tar.extractall(path=extract_folder)
                print(f"Extracted files to '{extract_folder}'")

        if item.endswith('.tar.gz'):
            try:
                os.remove(item_path)
                print("Deleted file : {item_path}")
            except OSError  as e:
                print("Error occured deleting the file:", e)
```

```
def copy_png_files(main_folder, destination_folder):
    # Check if the main folder exists
    if not os.path.exists(main_folder):
        print(f"Main folder '{main_folder}' does not exist.")
        return

    # Create the destination folder (where you want the extracted folder to be copied) if it doesn't exist
    os.makedirs(destination_folder, exist_ok=True)

    # Iterate through each item in the main folder
    for item in os.listdir(main_folder):
        item_path = os.path.join(main_folder, item)
        # Check if the item is a directory (subfolder)
        if os.path.isdir(item_path):
            # Recursively call the function to copy .png files in subfolders
            copy_png_files(item_path, destination_folder)
        elif item.endswith('.png'):
            # Copy the .tar.gz file to the destination folder
            shutil.copy(item_path, destination_folder)
            print(f"Copied '{item}' to '{destination_folder}'")

if __name__ == "__main__":
    main_folder = "C:/Users/******** /DeepfakeResearch/faketrainfolder/"
    destination_folder = "C:/Users//******** /DeepfakeResearch/faketrainpictures/"
    #extract_folder = destination_folder
    # Call the function to extract .tar.gz files recursively
    #copy_tar_gz_files(main_folder, destination_folder)
    #extract_tar_files_recursive(destination_folder,extract_folder)
    copy_png_files(main_folder, destination_folder)
```

A.2 Data Organization

Following function accepts a folder path containing images as an input, utilizes existing functions to extract features from each image, and compiles these features into separate lists. Additionally, it receives the corresponding tags for each image. Finally, it returns an array containing the feature data of all the data.

```
import os
import io


def get_images_and_labels(folder, label):
  calc_entropy = []
  calc_wrapped = []
  calc_noise = []
  calc_blur = []
  calc_keypoints = []
  calc_blobs = []
  assign_labels = []

  for subfolder in os.listdir(folder):
    subfolder_path = os.path.join(folder, subfolder)
    if not os.path.isdir(subfolder_path):
      continue

    for filename in os.listdir(subfolder_path):
      if not filename.endswith(".png"):
        continue

      try:
        img = io.imread(os.path.join(subfolder_path, filename))
        features = extract_features(img)
```

```
            assign _entropy.append(features['entropy'])
            assign _wrapped.append(features['wrapped'])
            assign _noise.append(features['noise'])
            assign _blur.append(features['blur'])
           assign _keypoints.append(features['keypoints'])
            assign _blobs.append(features['blobs'])
            assign_labels.append(label)
        except Exception as e:
            print(f"Error while processing image {filename}: {e}")

  return assign _entropy, assign _wrapped, assign _noise, assign _blur, assign _keypoints, assign
_blobs, assign_labels
```

The code below invokes the previously mentioned function to process both fake and real images. Subsequently, it compiles the extracted features into a dataframe.

```
# Assign the data
fake_entropy, fake_wrapped, fake_noise, fake_blur, fake_keypoints,
fake_blobs, fake_labels = get_images_and_labels(r" C:/Users/********
/DeepfakeResearch/faketrainfolder/", "fake")


real_entropy, real_wrapped, real_noise, real_blur, real_keypoints,
real_blobs, real_labels = get_images_and_labels(r"" C:/Users/********
/DeepfakeResearch/realtrainfolder/", "real")


data = pd.DataFrame({
  "Final_Entropy": fake_test_entropy + real_test_entropy,
  " Final_Wrapped": fake_test_wrapped + real_test_wrapped,
  " Final_Noise": fake_test_noise + real_test_noise,
  " Final_Blur": fake_test_blur + real_test_blur,
  " Final_Keypoints": fake_test_keypoints + real_test_keypoints,
  " Final_Blobs": fake_test_blobs + real_test_blobs,
  " Final_Label": fake_test_labels + real_test_labels
})
```

```
# saving the data to a csv formatted file

data.to_csv(data.csv")

print(all_data.head())
```

A.3 Feature Extraction

The following function accepts in an image, extracts all the features, and adds the data to a dictionary:

```
def extract_features(imgs):

gray_img = color.rgb2gray(img)


# Entropy Feature Extraction

entropy = skimage.measure.shannon_entropy(img)


# Wrapped Feature Extraction image_wrapped = np.angle(np.exp(1j * img)) max_val =
np.max(image_wrapped)

min_val = np.min(image_wrapped) wrapped_range = max_val - min_val


# Noise Feature Extraction astro = img_as_float(img) astro = astro[30:180, 150:300] sigma = 0.08

noisy = random_noise(astro, var=sigma ** 2)

sigma_est = np.mean(estimate_sigma(noisy, multichannel=True))


# Blur Feature Extraction

blurred_images = [ndi.uniform_filter(img, size=k) for k in range(2, 32, 2)] img_stack =
np.stack(blurred_images)


# Keypoints Feature Extraction detector = CENSURE() detector.detect(gray_img)


# Blob Dog Feature Extraction

blobs_dog = blob_dog(gray_img, max_sigma=1, threshold=.1)
```

```
features = { ' assign_entropy': entropy,

' assign_wrapped': wrapped_range , ‘assign_noise': sigma_est,

' assign_blur': np.mean(img_stack), ' assign_keypoints': len(detector.keypoints), ' assign_blobs': len(blobs_dog)

}

return features
```

Appendix B

Code for Model Training and Testing

For the training and testing purposes of the model, the WildDeepfake dataset was randomly split between training and testing set with a ratio of 80:20. Where 80% of the data is for training and 20% data is for testing purpose. Python is used for developing the model as for making classification.

B.1 Random Forest Model

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns


# Accessing the CSV file
data_RF = pd.read_csv("data.csv")


# Separating features and labels
X = data_RF [["assign_Entropy", " assign_Wrapped", " assign_Noise", " assign_Blur", " assign_Keypoints", " assign_Blobs"]]
y = data_RF ["assign_Label"]


# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)


# Standardizing the features
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Training the Random Forest classifier
forest_model = RandomForestClassifier(n_estimators=250, random_state=1)
forest_model.fit(X_train, y_train)

# Making predictions on the test set
forest_y_pred = forest_model.predict(X_test)

# Evaluating performance
forest_accuracy = accuracy_score(y_test, forest_y_pred)
forest_precision = precision_score(y_test, forest_y_pred, pos_label='real')
forest_recall = recall_score(y_test, forest_y_pred, pos_label='real')
forest_f1 = f1_score(y_test, forest_y_pred, pos_label='real')

print("Random Forest Accuracy:", forest_accuracy)
print("Random Forest Precision:", forest_precision)
print("Random Forest Recall:", forest_recall)
print("Random Forest F1 Score:", forest_f1)

# Forming the confusion matrix
forest_cm = confusion_matrix(y_test, forest_y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(forest_cm, annot=True, cmap="Blues", fmt='g', xticklabels=['fake', 'real'], yticklabels=['fake', 'real'])
plt.title("Random Forest Confusion Matrix")
plt.ylabel("True label")
plt.xlabel("Predicted label")
plt.show()

# Plotting the ROC curve
forest_probs = forest_model.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, forest_probs, pos_label='real')
```

```
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, color='lightblue', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='black', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest ROC Curve')
plt.legend(loc="lower right")
plt.show()
```

B.2 SVM Model

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
data_SVM = pd.read_csv("data.csv")

# Separate features and labels
X = data_SVMEntropy", "Wrapped", "Noise", "Blur", "Keypoints", "Blobs"]]
y = data_SVM["Label"]

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize the SVM classifier
svm_model = SVC(kernel='rbf', random_state=1)

# Train the SVM model
svm_model.fit(X_train, y_train)

# Make predictions on the test set
svm_y_pred = svm_model.predict(X_test)

# Evaluate the performance of the model
svm_accuracy = accuracy_score(y_test, svm_y_pred)
svm_precision = precision_score(y_test, svm_y_pred, pos_label='real')
svm_recall = recall_score(y_test, svm_y_pred, pos_label='real')
svm_f1 = f1_score(y_test, svm_y_pred, pos_label='real')

print("SVM Accuracy:", svm_accuracy)
print("SVM Precision:", svm_precision)
print("SVM Recall:", svm_recall)
print("SVM F1 Score:", svm_f1)

# Generate the confusion matrix
svm_cm = confusion_matrix(y_test, svm_y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(svm_cm, annot=True, cmap="Blues", fmt='g', xticklabels=['fake', 'real'], yticklabels=['fake', 'real'])
plt.title("SVM Confusion Matrix")
```

```
plt.ylabel("True label")
plt.xlabel("Predicted label")
plt.show()

# Plot the ROC curve
svm_probs = svm_model.decision_function(X_test)
fpr, tpr, thresholds = roc_curve(y_test, svm_probs, pos_label='real')
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, color='lightblue', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='black', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('SVM ROC Curve')
plt.legend(loc="lower right")
plt.show()
```

B.3 MesoNet

```
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Model as KerasModel
from tensorflow.keras.layers import Input, Dense, Flatten, Conv2D, MaxPooling2D, BatchNormalization, Dropout, Reshape, Concatenate, LeakyReLU
from tensorflow.keras.optimizers import Adam

# Define constants
IMGWIDTH = 256
```

```
class Classifier:
    def __init__(self):
        self.model = 0

    def predict(self, x):
        if x.size == 0:
            return []
        return self.model.predict(x)

    def fit(self, x, y):
        return self.model.train_on_batch(x, y)

    def get_accuracy(self, x, y):
        return self.model.test_on_batch(x, y)

    def load(self, path):
        self.model.load_weights(path)

class Meso4(Classifier):
    def __init__(self, learning_rate=0.001):
        self.model = self.init_model()
        optimizer = Adam(learning_rate=learning_rate)
        self.model.compile(optimizer=optimizer, loss='mean_squared_error', metrics=['accuracy'])

    def init_model(self):
        x = Input(shape=(IMGWIDTH, IMGWIDTH, 3))

        x1 = Conv2D(8, (3, 3), padding='same', activation='relu')(x)
        x1 = BatchNormalization()(x1)
        x1 = MaxPooling2D(pool_size=(2, 2), padding='same')(x1)

        x2 = Conv2D(8, (5, 5), padding='same', activation='relu')(x1)
```

```
        x2 = BatchNormalization()(x2)
        x2 = MaxPooling2D(pool_size=(2, 2), padding='same')(x2)

        x3 = Conv2D(16, (5, 5), padding='same', activation='relu')(x2)
        x3 = BatchNormalization()(x3)
        x3 = MaxPooling2D(pool_size=(2, 2), padding='same')(x3)

        x4 = Conv2D(16, (5, 5), padding='same', activation='relu')(x3)
        x4 = BatchNormalization()(x4)
        x4 = MaxPooling2D(pool_size=(4, 4), padding='same')(x4)

        y = Flatten()(x4)
        y = Dropout(0.5)(y)
        y = Dense(16)(y)
        y = LeakyReLU(alpha=0.1)(y)
        y = Dropout(0.5)(y)
        y = Dense(1, activation='sigmoid')(y)

        return KerasModel(inputs=x, outputs=y)

class MesoInception4(Classifier):
    def __init__(self, learning_rate=0.001):
        self.model = self.init_model()
        optimizer = Adam(learning_rate=learning_rate)
        self.model.compile(optimizer=optimizer, loss='mean_squared_error', metrics=['accuracy'])

    def InceptionLayer(self, a, b, c, d):
        def func(x):
            x1 = Conv2D(a, (1, 1), padding='same', activation='relu')(x)

            x2 = Conv2D(b, (1, 1), padding='same', activation='relu')(x)
            x2 = Conv2D(b, (3, 3), padding='same', activation='relu')(x2)
```

```
            x3 = Conv2D(c, (1, 1), padding='same', activation='relu')(x)
            x3 = Conv2D(c, (3, 3), dilation_rate=2, strides=1, padding='same', activation='relu')(x3)

            x4 = Conv2D(d, (1, 1), padding='same', activation='relu')(x)
            x4 = Conv2D(d, (3, 3), dilation_rate=3, strides=1, padding='same', activation='relu')(x4)

            y = Concatenate
(axis=-1)([x1, x2, x3, x4])

            return y
        return func

    def init_model(self):
        x = Input(shape=(IMGWIDTH, IMGWIDTH, 3))

        x1 = self.InceptionLayer(1, 4, 4, 2)(x)
        x1 = BatchNormalization()(x1)
        x1 = MaxPooling2D(pool_size=(2, 2), padding='same')(x1)

        x2 = self.InceptionLayer(2, 4, 4, 2)(x1)
        x2 = BatchNormalization()(x2)
        x2 = MaxPooling2D(pool_size=(2, 2), padding='same')(x2)

        x3 = Conv2D(16, (5, 5), padding='same', activation='relu')(x2)
        x3 = BatchNormalization()(x3)
        x3 = MaxPooling2D(pool_size=(2, 2), padding='same')(x3)

        x4 = Conv2D(16, (5, 5), padding='same', activation='relu')(x3)
        x4 = BatchNormalization()(x4)
        x4 = MaxPooling2D(pool_size=(4, 4), padding='same')(x4)

        y = Flatten()(x4)
```

```
    y = Dropout(0.5)(y)
    y = Dense(16)(y)
    y = LeakyReLU(alpha=0.1)(y)
    y = Dropout(0.5)(y)
    y = Dense(1, activation='sigmoid')(y)

    return KerasModel(inputs=x, outputs=y)
```