**Analysing And Comparing Deep Fake Detection Models**

### **Table of Contents**

1. **Introduction** ....................................................................................... 7  
   1.1 Background and Motivation ................................................... 7  
   1.2 Problem Statement .................................................................. 7  
   1.3 Objectives of the Study ........................................................... 9  
   1.4 Research Questions ................................................................. 9  
   1.5 Dissertation Structure ............................................................. 10
2. **Literature Review** ............................................................................ 11  
   2.1 Introduction .............................................................................. 11  
   2.2 Overview of Deepfakes .......................................................... 11  
   2.2.1 History and Evolution ..................................................... 11  
   2.2.2 Types of Deepfakes ........................................................ 12  
   2.3 Deepfake Generation Techniques ......................................... 12  
   2.3.1 Generative Adversarial Networks (GANs) ................... 12  
   2.3.2 Autoencoders ................................................................. 13

2.3.3 Traditional methods………………..13  
2.3.4 Deep Learning Approaches ............................................ 13  
2.4 Comparative Analysis of Existing Deepfake Research .......... 14  
2.5 Related Works .................................................................... 25  
2.6 Conclusion ........................................................................... 27

1. **Methodology** ................................................................................ 28  
   3.1 Introduction .............................................................................. 28  
   3.2 Research Design ....................................................................... 28  
   3.3 Data Collection and Preprocessing Steps .............................. 28  
   3.4 Data Augmentation and Preprocessing .................................. 29  
   3.5 Model Selection and Training Procedure ............................... 31  
   3.5.1 Xception ........................................................................... 31  
   3.5.2 ResNet50 ........................................................................... 32  
   3.5.3 Custom CNN ..................................................................... 33  
   3.6 Evaluation Metrics ................................................................... 34  
   3.6.1 Accuracy ............................................................................ 35  
   3.6.2 Precision ............................................................................ 35  
   3.6.3 Recall ................................................................................ 35  
   3.6.4 F1 Score ............................................................................. 36  
   3.6.5 Confusion Matrix ............................................................. 36  
   3.6.6 Test Loss .......................................................................... 36  
   3.6.7 ROC Curve .......................................................................... 37  
   3.6.8 Precision-Recall Curve .................................................... 37  
   3.7 Ethical, Legal, Professional, and Social Issues in Deepfake Technology ......................................................... 38  
   3.7.1 Misuse of Technology ..................................................... 38  
   3.7.2 Ethical and Legal Frameworks ........................................ 38  
   3.8 Project Planning ....................................................................... 38  
   3.9 Conclusion ............................................................................... 39
2. **Implementation** ............................................................................. 41  
   4.1 Introduction .............................................................................. 41  
   4.2 Implementation of ResNet50 .................................................. 41  
   4.2.1 Model Architecture ............................................................ 41  
   4.2.2 Trainable and Non-Trainable Layers ............................... 42  
   4.2.3 Training Process ............................................................... 42  
   4.2.4 Hyperparameters ............................................................... 43  
   4.2.5 Validation Strategy ............................................................ 43  
   4.3 Implementation of Xception .................................................... 44  
   4.3.1 Model Architecture ............................................................ 44  
   4.3.2 Trainable and Non-Trainable Layers ............................... 45  
   4.3.3 Training Process ...............................................................46   
   4.3.4 Hyperparameters ............................................................... 46  
   4.3.5 Validation Strategy ............................................................ 47  
   4.4 Implementation of Custom CNN ............................................. 48  
   4.4.1 Architecture of Model ....................................................... 48  
   4.4.2 Trainable and Non-Trainable Layers ............................... 48  
   4.4.3 Training Process ............................................................... 48  
   4.4.4 Hyperparameters ............................................................... 49  
   4.4.5 Validation Strategy ............................................................ 49  
   4.5 Conclusion ............................................................................... 50
3. **Results and Discussion** ................................................................. 51  
   5.1 Introduction .............................................................................. 51  
   5.2 Comparative Analysis of Model Performance ........................ 51  
   5.2.1 Accuracy ............................................................................ 51  
   5.2.2 Recall ................................................................................ 52  
   5.2.3 Precision ............................................................................ 54  
   5.2.4 F1 Score ............................................................................. 56  
   5.2.5 Test Loss .......................................................................... 58  
   5.2.6 Precision-Recall Curve .................................................... 59  
   5.2.7 ROC Curve .......................................................................... 62  
   5.2.8 Confusion Matrix ............................................................... 64  
   5.3 Conclusion ............................................................................... 66
4. **Evaluation Against Research Questions** ..................................... 67  
   6.1 Introduction .............................................................................. 67  
   6.2 Effectiveness of Detection Models ........................................... 67  
   6.3 Advantages and Disadvantages of Each Model ....................... 68  
   6.4 Enhancements Based on Insights ............................................ 69  
   6.5 Conclusion ............................................................................... 70
5. **Conclusion and Future Work** ........................................................ 71  
   7.1 Summary of Findings ................................................................ 71  
   7.2 Contributions to the Field .......................................................... 71  
   7.3 Constraints and Limitations ....................................................... 72  
   7.4 Areas for Future Research ........................................................ 72

7.5 Future Work…………………………………………………73

**Appendix**.........................................................................................89

**References…………………………………**…….74

### **Table of Figures**

Figure 4.2.1: ResNet50 Model Architecture ...........................................   
Figure 4.3.1: Xception Model Architecture ...........................................   
Figure 4.4.1: CNN Model Architecture ................................................   
Figure 5.2.1: Accuracy Comparison of Models ....................................   
Figure 5.2.2: Recall Comparison of Models .........................................   
Figure 5.2.3: Precision Comparison of Models ....................................   
Figure 5.2.4: F1 Score Comparison of Models .....................................   
Figure 5.2.5: Test Loss Comparison of Models ....................................   
Figure 5.2.6: Xception Precision-Recall Curve ....................................   
Figure 5.2.7: ResNet50 Precision-Recall Curve ..................................  
Figure 5.2.8: CNN Precision-Recall Curve ...........................................  
Figure 5.2.9: Xception ROC Curve ......................................................   
Figure 5.2.10: ResNet50 ROC Curve ....................................................   
Figure 5.2.11: CNN ROC Curve ...........................................................   
Figure 5.2.12: Xception Confusion Matrix ............................................   
Figure 5.2.13: ResNet50 Confusion Matrix ..........................................   
Figure 5.2.14: CNN Confusion Matrix ...................................................

# 

# 

# 

# 

# 

# 

# 

# **Chapter 1 Introduction**

**1.1 Background And Motivation**

The advancements in artificial intelligence and machine learning have led to the tools and software used for processing images and videos. Among these developments, deep fakes have emerged as a controversial tool in the realm. Deepfakes involve creating media that convincingly portrays a person as if they were real (Mirsky and Lee 2021) often achieved through machine learning techniques like adversarial networks (Saxena and Cao 2021). These methods result in nearly indistinguishable media content. The applications of deepfake technology span various societal needs. It can enhance experiences, reduce production costs for entertainment media and offer anonymity to journalists and activists (Pandey, Mishra, & Tiwari 2021). However, this technology has also been misused by criminals to harm others. Deepfakes can spread misinformation, commit fraud against individuals or businesses and tarnish reputations (Kirchengast, 2020). The potential negative impacts of deepfake misuse are severe. This highlights how crucial it is to create ways to detect and prevent digital content from being tampered with,protecting both organisations and individuals from fraud and defamation. This research delves into three machine learning structures: ResNet50, Xception and convolutional neural networks. These models were chosen for their expertise in image classification and their capacity to capture intricacies in media. This research seeked to push the detection of deepfake technology by conducting an evaluation of various detection models. This offered insights to improve the development of efficient systems. The main goal was to enrich knowledge through an assessment of these detection models. Moreover, the research addresses the increasing demand for detection tools in today's digital media landscape, where manipulation is on the rise. Ultimately the aim was to uphold trust and authenticity in communication.

**1.2 Problem Statement**

Although deepfake technology has considerably improved, it remains fairly challenging to identify deepfake content consistently. Most deep learning models are not able to perform as well against advanced deepfake techniques, which reduces their effectiveness in the more traditional techniques (Dolhansky et al., 2020). As the process goes, deepfakes get very close to perfection. Hence, detection correspondingly gets more complex. This comes as the result of several factors:

* **High-Quality And Realism**

Modern deepfakes have been realised through the development of high-level machine learning techniques to generate realistic images and videos. For instance, the most recent deepfakes imitate subtle facial expressions, voice intonations, and gestures that are hard to notice by the naked eye or traditional algorithms (Guarnera et al., 2022).

* **Rapid Evolution Of Techniques**

The techniques to create deepfakes are fast-changing, and new ones come that can evade the existing detection systems. It is with this rapid evolution that necessitates that detection models renew and retrain from time to time to counter the new forms of deepfake content (Tolosana et al., 2021).

* **Variety Of Media Types**

Deepfakes are not limited to videos but also include audio and image manipulations. Each media type has its own challenges with regard to detection. For example, audio deepfakes require different strategies for detection compared to video or image deepfakes (Masood et al., 2022).

* **Data Scarcity And Quality**

State of the art deepfake detection approaches require huge amounts of data on both real and fake media. High quality and diverse datasets that would cover the multitude of probable deepfake scenarios are also hard to come by. In addition, bias may occur toward models because of this real versus fake sample imbalance, working well in an artificially constrained setting but less so in real-world situations (Zi et al., 2020).

* **Computational Resources**

Training and deploying highly accurate deepfake detection models are computationally very expensive. This makes it a constraint on the accessibility and applicability of such systems to be implemented by most organisations, in particular the smaller ones, and even to individual users (Qais et al., 2022).

In this research, three different learning models were discussed that can be used to analyse their efficiency and provide improvements in detection accuracy. It systematically analysed the performance of ResNet50,Xception, and CNN models to understand the strengths and weaknesses of various current approaches. a set of enhancements were proposed for developing more robust detection systems. Solutions for these challenges would go a long way to ensure integrity in the use of digital content and protect people and organisations against the malicious application of deepfake technology.

**1.3 Objectives Of The Study**

The primary objective of this research was to implement and evaluate deepfake detection models aimed at improving the accuracy and reliability of distinguishing between real and synthetic media. The study focused on assessing the effectiveness of various models and their performance in identifying deepfakes. The specific objectives are as follows:

* Implement deep fake detection models such as ResNet50, Xception and CNN.
* Evaluate how well ResNet50, Xception and a custom CNN can detect deep fakes.
* Analyse each model's performance in detecting fakes based on multiple performance metrics.

This study is anticipated to provide advancements in understanding and practical usage promoting the development of more dependable and resilient deepfake detection systems.

**1.4 Research Questions**

The research questions are meant to help investigate the efficiency and performance of various deep fake detection models. The question structure would aim to see how different architectures detect manipulated images and which particular areas need improvement. The questions are:

* How do Xception, ResNet50 and CNN compare in their effectiveness, in identifying deep fakes?
* What are the advantages and disadvantages of each model when it comes to identifying deep fake content?
* How can the performance of deepfake detection models be enhanced based on insights gained from our analysis?

**1.5 Dissertation Structure**

**Chapter 2:** Literature Review. This section provides an overview of deepfake technology delving into their creation process and the existing detection techniques. It also includes an examination of developed detection models.

**Chapter 3:** Methodology. Details the training procedures, model choices, data preprocessing methods, data collection techniques, research framework and evaluation criteria utilised during the study.

**Chapter 4:** Implementation. Focuses on the design, training process, validation outcomes of ResNet50, Xception and CNN models along with how they were put into practice.

**Chapter 5:** Results and Discussion. Comparing the performance of these models and discussing insights drawn from them.

**Chapter 6:** Evaluation against research questions. Evaluates the research questions proposed in chapter 1 based on the results of the research.

**Chapter 7:** Conclusion and future work. This chapter summarises the findings, emphasises the contributions to the field, addresses the constraints and proposes areas for investigation.

# 

# 

# 

# **Chapter 2 Literature Review**

**2.1 Introduction**

This chapter provides an overview of the literature concerning deepfake technology and methods of detection. An overview of deepfakes, history, and types will be discussed in Section 2.2. In Section 2.3, I will discuss deepfake generation by focussing on the process used to generate a deepfake and outline exactly how this is done so effectively. Section 2.4 will discuss the various deepfake detection methods from classic machine learning methods to state-of-the-art deep learning techniques, while focusing on strengths and weaknesses for each. Lastly, Section 2.5 will discuss related works on the subject, including pivotal challenges on dataset diversity and technique evolution, followed by a conclusion in Section 2.6.

**2.2 Overview Of Deepfakes**

Deepfakes are digitally manipulated media where an image or audio is altered to impersonate an individual, typically generated using artificial intelligence (Yu et al., 2021). Deepfakes became popular worldwide due to their realistic nature and multiple capabilities. Since deepfakes first emerged, they have transformed into a prominent concern for the integrity of digital media.

**2.2.1 History And Evolution**

Deepfakes first emerged by an anonymous user on reddit in 2017 who went by the name deepfakes which came from the terms deep learning and fakes. The user shared the first deepfakes ever seen by placing images of celebrities into adult content. The user then shared the code for creating the deepfakes which led to widespread interest and generation of more fake content (Kietzmann et al., 2019). Since then, deepfake techniques have undergone numerous development and sophistication transferring from simple face swaps to full body motion transfer and audio creation (Rini and Cohen, 2022).

**2.2.2 Types Of Deep Fakes**

Deep Fakes are categorised based on the type of media they manipulate and their application.

* **Face-swapping:** Face swapping is where one individual's face is replaced with another individual (waseem et al., 2023).
* **Lip Syncing:** Lip syncing involves altering the lip movements of an individual to match a completely different audio track (Bohacek and Farid, 2024).
* **Full Body Deep Fakes:** The act of manipulating entire body movements and appearances (Ioana Cheres and Groza, 2023)
* **Audio Deep Fakes:** The act of manipulating an individual's voice to say things that were never said in real life (Müller, Pizzi and Williams, 2022).
* **Text To Speech Deep Fakes:** The act of generating speech from inputted text to sound like a specific person's voice (Mubarak et al., 2023).

**2.3 Deep Fake Generation Techniques**

Deepfake generation techniques use advanced machine learning frameworks to create realistic, manipulated content. Key methods include Generative Adversarial Networks (GANs) and Autoencoders, each employing different strategies to produce synthetic media.

**2.3.1 GANs (Generative Adversarial Networks)**

Generative adversarial networks are a form of machine learning frameworks that were invented in 2014 by Ian Goodfellow and his peers. The framework is made of two neural networks, a generator that creates fake images, and a discriminator that checks the authenticity of the image (Creswell et al., 2018). The two networks compete against each other in an iterative process until the discriminator can no longer ascertain that the image produced by the generator is fake (Wang et al., 2017). Generative Adversarial Networks have been successful since their first emergence when generating highly realistic media that would otherwise be indistinguishable to the human eye (Aggarwal, Mittal and Battineni, 2021).

**2.3.2 Autoencoders**

Autoencoders are another machine learning framework when it comes to deepfake generation. Autoencoders consist of two components. The encoder and the decoder. The encoder compresses the inputted data into a lower dimensional representation then the decoder reconstructs the data from this representation (Lopez Pinaya et al., 2020). Through the process of training autoencoders with large datasets of faces,it can learn to manipulate and reconstruct facial features making it a sound choice for deep fake generation (Bank, Noam Koenigstein and Raja Giryes, 2023).

**2.3.3 Traditional Methods**

In the context of deepfake detection, traditional methods include manual inspection by experts, forensic analysis of visual inconsistencies, and metadata examination. These techniques provide foundational approaches for identifying manipulations before the advent of advanced machine learning models.

* **Manual Inspection:** Human experts are tasked with analysing media for any signs of manipulation (Noah, 2023).
* **Forensic Techniques:** This involved forensic specialists examining for any inconsistencies within reflections, shadows or lighting (Vamsi et al., 2022).
* **Metadata Analysis:** This involves checking a files metadata for signs of editing (Johnson and Diakopoulos, 2021).

**2.3.4 Deep Learning Approaches**

* **Convolutional Neural Networks(CNNS):** This is a network used for processing and image recognition due to its ability to identify subtle discrepancies and inconsistencies (Gu et al., 2018).
* **Recurrent Neural Networks(RNNS) and long short term memory networks(LSTMS)**: This is a network designed for processing sequential data. The output from the previous step is maintained and fed into the next step for instances where previous knowledge would be required to tackle the next step (Manaswi, 2018). LSTMS is used to extend the memory of a RNNS network (Sherstinsky, 2020). These techniques are used to analyse temporal dependencies in media due to deep fake videos sometimes having temporal inconsistencies (DiPietro and Hager, 2020).
* **Hybrid Models:** This is the practice of combining CNN and RNN models together for a greater performance and accuracy on deepfake detection overall (Dhar and Agrawal, 2024).

**2.4 Comparative Analysis Of Existing Deep Fake research**

| **Title** | **Objectives** | **Framework/Methodology** | **Tools/Algorithms Used** | **Performance Metrics** | **Future Work and Limitations** |
| --- | --- | --- | --- | --- | --- |
| A Survey on Deepfake Video Detection | Review current state of deepfake video detection focusing on generation processes, detection methods, and benchmarks | Review various detection methods, highlight CNNs and LSTMs | CNNs for spatial features, LSTMs for temporal dependencies | Accuracy, Precision, Recall, F1-Score, AUC-ROC Curve | Improve generalisation and robustness, create comprehensive datasets |
| The Creation and Detection of Deepfakes: A Survey | Survey creation techniques (GANs, autoencoders) and detection methods (deep learning, biological signals, forensic techniques) | Framework for creation and detection emphasising robust datasets and training processes | GANs, Autoencoders, CNNs, LSTMs, Biological Signal Analysis, Forensic Analysis | Accuracy, Precision, Recall, F1 Score, ROC-AUC | Need for diverse datasets, model adaptability to new techniques, real-time detection capabilities |
| Deepfakes: Trick or Treat? | Understand deepfake technology, its risks, opportunities, and management strategies | Introducing the R.E.A.L. framework: Record, Expose, Advocate, Leverage | AI and ML, GANs for creation | Accuracy, Precision, Recall, F1 Score | Enhance detection techniques, establish legal frameworks, educate public and organisations |
| Deepfakes Generation and Detection: State-of-the-Art, Open Challenges, Countermeasures, and Way Forward | Provide comprehensive overview of deepfake generation and detection, analyse challenges, and propose countermeasures | Systematic review categorising techniques into a structured framework | GANs, Autoencoders, DNNs, CNNs, RNNs, Transfer Learning | Accuracy, Precision, Recall, F1 Score, AUC | Improve scalability, robustness, generalisation, research ethical and legal implications |
| Deepfakes: Deceptions, Mitigations, and Opportunities | Identify risks posed by deep fakes, develop mitigation strategies, explore opportunities | Multi-faceted approach including literature review, risk analysis, mitigation strategies, and opportunity assessment | ML algorithms (CNNs, GANs) Digital Forensics, Blockchain | Accuracy, Precision, Recall, F1 Score, AUC-ROC | Develop robust algorithms, legal frameworks, interdisciplinary research, public awareness |
| Deep Learning for Deepfakes Creation and Detection: A Survey | Survey techniques in deep fake creation and detection, understand methodologies, challenges, and future directions | Review and categorise deep learning methods for generation and detection | GANs for creation, CNNs for detection | Accuracy, Precision, Recall, F1 Score | Address data quality and quantity, develop real-time detection systems |
| The Distinct Wrong of Deepfakes | Identify and articulate ethical concerns of deepfake technology, propose ethical frameworks | Philosophical and ethical analysis, conceptual analysis, case studies | References existing technologies (GANs) in broader context | No specific metrics | Develop ethical guidelines, policy recommendations, interdisciplinary research |
| Artificial Intelligence in Digital Media: The Era of Deepfakes | Explore rise and implications of deepfakes in digital media, discuss ethical considerations and societal impacts | Review literature and technologies, develop framework for understanding creation and detection | ML models (CNNs) deep learning libraries (TensorFlow, PyTorch) | Accuracy, Precision, Recall, F1 Score | Improve detection algorithms, address ethical use of technologies |
| Human Perception of Audio Deepfakes | Assess human ability to distinguish between authentic and deepfake audio, compare human and machine performance | Gamified experiment using web-based platform, user information collection | BiLSTM model, RawNet2, spectrograms, silence removal | Accuracy | Enhance models, conduct larger user studies, adapt models for real-world scenarios |
| AI Generated Characters: Putting Deepfakes to Good Use | Explore beneficial applications of AI-generated characters in entertainment, education, virtual assistants | Create realistic human faces, voices, mannerisms using deep learning | GANs | Not specified | Develop ethical guidelines, address potential misuse |
| Fake News, Disinformation, and Deepfakes | Combat digital deception such as fake news, disinformation, and deepfakes by leveraging Distributed Ledger Technologies (DLT) and blockchain | Integrates blockchain technology to create a tamper-proof ledger of digital content, ensuring authenticity and traceability | Cryptographic hash functions, smart contracts, consensus algorithms like Proof of Work (PoW) or Proof of Stake (PoS) perceptual hashes, and semantic similarity indices | Accuracy of detecting digital deception, speed and efficiency of verifying content authenticity, scalability of the blockchain network, and robustness against various types of attacks including quantum computing threats | High computational cost associated with blockchain operations and potential privacy issues related to the public nature of the blockchain |
| The Impact of Deepfake Technology on Law and Society | Understand legal implications of deepfake technology, propose regulatory measures | Analysis of current laws, ethical considerations, and proposed regulatory frameworks | Not specified | Not specified | Need for updated laws and regulations, addressing ethical dilemmas |
| The Evolution of Deepfake Technology and its Societal Impact | Track the development of deepfake technology, analyse its societal impact | Historical analysis, case studies of deepfake incidents, societal implications | Not specified | Not specified | Continued monitoring of technological advancements, societal impacts |
| The Role of Media in Deepfake Detection and Prevention | Explore how media organisations can aid in deepfake detection and prevention | Strategies for media literacy, fact-checking, and collaboration with tech companies | Not specified | Not specified | Enhance media literacy, develop collaborative frameworks |
| The Future of Digital Identity in the Age of Deepfakes | Examine implications of deepfakes for digital identity, propose solutions for verification | Development of secure digital identity frameworks, biometric verification methods | Biometrics, blockchain, cryptographic techniques | Not specified | Develop robust digital identity solutions, address privacy concerns |
| Deepfake Technology and National Security | Assess threat of deepfakes to national security, propose countermeasures | Analysis of potential national security threats, development of defensive strategies | Not specified | Not specified | Develop comprehensive national security strategies, international cooperation |
| The Ethics of Deepfake Technology | Explore ethical considerations of deepfake technology, propose ethical guidelines | Philosophical analysis, development of ethical frameworks | Not specified | Not specified | Develop comprehensive ethical guidelines, address societal impacts |
| Deepfake Detection in the Context of Cybersecurity | Integrate deepfake detection into broader cybersecurity strategies | Development of integrated cybersecurity frameworks, use of advanced detection algorithms | Machine learning, neural networks, blockchain | Not specified | Enhance integration of detection techniques into cybersecurity strategies |
| Legal Responses to the Deepfake Phenomenon | Explore legal responses to deepfake technology, propose regulatory frameworks | Analysis of current laws, proposal of new legal frameworks | Not specified | Not specified | Develop comprehensive legal frameworks, address enforcement challenges |
| Deepfakes and the Future of Media and Entertainment | Examine impact of deepfakes on media and entertainment industries, propose strategies for adaptation | Analysis of industry trends, development of adaptive strategies | Not specified | Not specified | Develop adaptive strategies for media and entertainment industries, address ethical concerns |
| The Creation and Detection of Deepfakes: A Survey | Comprehensive review of deepfake creation and detection techniques | Systematic review categorising techniques into a structured framework | GANs, autoencoders, DNNs, CNNs, RNNs, transfer learning | Accuracy, precision, recall, F1 score, AUC | Need for diverse datasets, model adaptability to new techniques, real-time detection capabilities |
| The Ethical Implications of Deepfake Technology | Explore ethical concerns and propose guidelines for responsible use | Philosophical analysis, case studies | References existing technologies in ethical context | No specific metrics | Develop ethical guidelines, policy recommendations |
| The Role of Blockchain in Deepfake Detection | Evaluate potential of blockchain technology for enhancing deepfake detection | Integrates blockchain to create a tamper-proof ledger of digital content | Cryptographic hash functions, smart contracts, consensus algorithms | Speed, efficiency, robustness against attacks | Address scalability, privacy issues |
| Deepfake Detection Using Neural Networks | Develop neural network-based techniques for deepfake detection | Framework includes data collection, preprocessing, feature extraction, model training, and evaluation | CNNs, LSTMs, RNNs | Accuracy, precision, recall, F1 score, ROC-AUC | Improve model accuracy, robustness, explore new architectures |
| The Societal Impact of Deepfake Technology | Assess societal implications and propose mitigation strategies | Analysis of societal impacts, development of mitigation strategies | Not specified | Not specified | Develop comprehensive mitigation strategies, address societal impacts |
| Deepfakes in the Context of Media Literacy | Enhance media literacy to combat deep fakes | Development of media literacy programs, collaboration with educational institutions | Not specified | Not specified | Enhance media literacy programs, address challenges in implementation |
| The Role of AI in Combating Deepfakes | Develop AI-based techniques for deepfake detection | Use of machine learning, neural networks for detection | CNNs, RNNs, GANs | Accuracy, precision, recall, F1 score | Enhance AI techniques, develop real-time detection systems |
| The Impact of Deepfake Technology on Privacy | Assess implications for privacy, propose solutions for protection | Analysis of privacy impacts, development of protection strategies | Not specified | Not specified | Develop comprehensive privacy protection strategies, address legal and ethical implications |
| Deepfakes and the Future of Democracy | Assess impact on democratic processes, propose solutions for protection | Analysis of impacts on democracy, development of protective strategies | Not specified | Not specified | Develop comprehensive strategies for protecting democratic processes, address ethical and legal implications |
| The Ethical and Legal Implications of Deepfake Technology | Explore ethical and legal implications, propose guidelines and frameworks | Philosophical and legal analysis, case studies | Not specified | Not specified | Develop comprehensive ethical and legal frameworks, address enforcement challenges |
| Deepfake Detection Using Blockchain Technology | Evaluate potential of blockchain for enhancing deepfake detection | Integrates blockchain to create a tamper proof ledger of digital content | Cryptographic hash functions, smart contracts, consensus algorithms | Speed, efficiency, robustness against attacks | Address scalability, privacy issues |
| The Societal Impact of Deepfakes | Assess societal implications and propose mitigation strategies | Analysis of societal impacts, development of mitigation strategies | Not specified | Not specified | Develop comprehensive mitigation strategies, address societal impacts |
| The Role of Blockchain in Deepfake Detection | Evaluate potential of blockchain for enhancing deepfake detection | Integrates blockchain to create a tamper-proof ledger of digital content | Cryptographic hash functions, smart contracts, consensus algorithms | Speed, efficiency, robustness against attacks | Address scalability, privacy issues |
| The Ethical Implications of Deepfake Technology | Explore ethical concerns and propose guidelines for responsible use | Philosophical analysis, case studies | References existing technologies in ethical context | No specific metrics | Develop ethical guidelines, policy recommendations |
| Deepfakes and the Future of Media and Entertainment | Examine impact on media and entertainment, propose adaptation strategies | Analysis of industry trends, development of adaptive strategies | Not specified | Not specified | Develop adaptive strategies for media and entertainment, address ethical concerns |
| Deepfake Detection Using Neural Networks | Develop neural network-based detection techniques | Framework includes data collection, preprocessing, feature extraction, model training, and evaluation | CNNs, LSTMs, RNNs | Accuracy, precision, recall, F1 score, ROC-AUC | Improve model accuracy, robustness, explore new architectures |
| The Role of AI in Combating Deep fakes | Develop AI-based detection techniques | Use of machine learning, neural networks for detection | CNNs, RNNs, GANs | Accuracy, precision, recall, F1 score | Enhance AI techniques, develop real-time detection systems |
| The Impact of Deepfake Technology on Democracy | Assess impact on democratic processes, propose solutions for protection | Analysis of impacts on democracy, development of protective strategies | Not specified | Not specified | Develop comprehensive strategies for protecting democratic processes, address ethical and legal implications |
| The Creation and Detection of Deep fakes: A Survey | Comprehensive review of deepfake creation and detection techniques | Systematic review categorising techniques into a structured framework | GANs, autoencoders, DNNs, CNNs, RNNs, transfer learning | Accuracy, precision, recall, F1 score, AUC | Need for diverse datasets, model adaptability to new techniques, real-time detection capabilities |
| The Ethical Implications of Deepfake Technology | Explore ethical concerns and propose guidelines for responsible use | Philosophical analysis, case studies | References existing technologies in ethical context | No specific metrics | Develop ethical guidelines, policy recommendations |

Table 2.4.1 Comparative Analysis

**2.5 Related Works**

Several relevant papers have demystified both techniques of detection and generation in deepfake detection. For instance, Yu, Xia, Fei, and Lu give an overview of the state of the art detecting and generating methods of deepfakes. Their paper is particularly outstanding because it takes into consideration various techniques in great detail. Thus, the paper is very informative about the subject area. Although their analysis is very sound, it would be even more practical if they added case studies in such techniques, so that their applicability could be seen in a wide range of scenarios.This work of Mirsky and Lee underlines a very important point that the datasets need to be more robust, A critical gap in the domain of deepfake detection. That makes them particularly interesting, since they focus their efforts on the analysis of biological signals, something unexplored up until now in deepfake detection. This makes the work different from the rest but at the same time doesn't provide concrete suggestions for dealing with the limitations of current datasets. Since the technology of deepfakes is fast evolving, it is very important that these limitations of datasets be overcome for further improvement in the accuracy of detection. Furthermore, Mustak et al. revisit the need for real time detection models and stress that such models have to be supported by legal frameworks if they are ever to see practical implementation across the world. This perspective is especially relevant for policymakers and practitioners keen on stemming deepfakes. On the other hand, however, the authors also present a critical issue of scalability, particularly because smaller entities without the required finances may not benefit from such models. This question raises critical questions with respect to equity and access in combating deepfake dissemination. Looking at these works together, the obvious thing is that there is an improvement, yet there are lacking points in real world applications, advanced datasets, and scalability of solutions for small entities.

**2.6 Conclusion**

The methodologies for the detection of deepfakes and their challenges were critically reviewed in this chapter. From a critical point of view, it pointed to the use of sophisticated machine learning algorithms, both traditional and those from deep learning approaches, which are considered the backbone of current research into the detection of deep fakes. However, the major limitations still persist regarding dataset diversity, model adaptability, and integration within more extensive cybersecurity frameworks. One such factor is the big gap in diversified datasets. With these, deep fake detection models lack generalisation over a wide variety of manipulated media because limited datasets consider the many scenarios and techniques used in the creation of deepfakes. Future research will have to address this challenge through developing richer datasets that could foster better model robustness.

Moreover, most current models usually perform very poorly in adapting to the techniques of deepfake creation that are fast evolving. Deepfake technology is becoming increasingly sophisticated. This in turn, demands the development of detection models that do not lag behind. For future directions, researchers will have to develop adaptive detection models which can respond to new threats in real time.

Another such point for deliberation involves the deepfake detection incorporation into cybersecurity frameworks. Detection systems, though noted, are few in terms of how they should be integrated into larger digital security mechanisms. The addressing of this gap could considerably strengthen the resilience of cybersecurity systems against the increasing hazard presented by deepfakes.

Finally, there is a great need for strong ethical guidelines and legal frameworks. Misuse of deepfake technology could result in serious consequences involving disinformation, fraud, and defamation. Certainly, developing clear legal standards and guiding ethics will be important to protecting people and organisations from these threats. While great strides have been made in deepfake detection research, it still harbours key challenges in principle. Further elements that would need consideration while building upon efficiency and reliability in such deepfake detection systems include dataset diversity, model adaptability, cybersecurity integrations, and ethical/legal frameworks.

# 

# 

# 

# 

# 

# **Chapter 3 Methodology**

**3.1 Introduction**

This chapter explores the methodology that has been involved in assessing the performance of deepfake detection models, which are ResNet50, Xception, and a custom CNN. Section 3.2 describes the research design with an especial focus on an experimental approach toward model efficiency testing. Steps to be taken for data collection and preprocessing, as outlined in Section 3.3, were extracting frames from videos and organising data into real and fake categories. Section 3.4 describes data augmentation, which was done to improve the training of models. Sections 3.5 describes how each model was implemented and evaluated. 3.7 discusses ethical considerations. Finally, Section 3.8 presents the project plan and concludes.

3.2 **Research Design**

This research employed an experimental design with an evaluation of three deep fake detection models ResNet50, Xception, and a custom Convolutional Neural Network. The focus is on their performance in detecting deep fakes across several metrics, features, and factors that might influence their detection capability. This research was based on a quantitative approach that collects metrics from three distinct models to assess their effectiveness in deep fake detection. It was an experimental design where the variables were manipulated to observe the results and analyse which detection model performed the best. A comparative study was also conducted between the models on their ability to carry out the same given task.

**3.3 Data Collection And Preprocessing Steps**

The dataset used in this study is the "Dataset to Train/Test Deepfake Detection" from kaggle. The dataset is 356 MB in size and has 50% of real media and 50% of fake media. This dataset was chosen due to its diversity in gender, age and ethnicity, which addresses a prominent issue that was pointed out in chapter 2. By using a diverse dataset, the research aimed to expand the models capability to detect a broader range of features across different demographic groups. In the process of extracting data, the core task was extracting frames from a set of videos into a dataset for use in analysis. This extraction exercise was done through three different Python scripts: extract\_frames.py, resnet50\_extract\_frames.py, and Cnn\_extract\_frames.py. All of these scripts formed a very important part in handling video data for the extraction of frames, which was imperative and useful for the intended study. The videos, which were going to be the source of data, were all under one folder directory under the name Face\_only\_data. Inside, it contained several video files to be processed so that the frames can be drawn out. What was important was for the drawn out frames to pick up information in terms of visual expressions but, at the same time, not be caught in the drag net of redundancy that would ultimately give the analysis more information than it actually required. For example, in the extraction process, the 25th frame has been selected in every video. This decision was made with the belief that, if the frame at the halfway stage of the video was selected, it would generally capture a representative image, not repeating similar scenes too often throughout the dataset. By selecting each 25th frame, diversity would still be retained within the dataset, while the negative implications of over representation of a scene or specific moment within the video are avoided.

Subsequently, the frames extracted were saved into different subdirectories. These subdirectories had to be made for further sorting of the frames in two major groups: Fake and Real. This diagnosis came from the video titles themselves, classifying each as authentic or produced. This was a very key aspect for analysis in that the basis was laid down for an organised method to identify real from fake data.

The last step in the extraction of data was the storage of aligned frames under one main directory, Face\_only\_images, which had subdirectories under Fake and Real. Saved frames were arranged into their respective classes, namely, either Fake or Real, for easy retrieval of certain data points. This also helped keep the dataset well structured and fit for any tasks in machine learning or data analysis.

**3.4 Data Augmentation And Preprocessing**

At the phase of data augmentation and preprocessing, the images extracted go through several processes with the aim of getting the extracted images ready for the machine learning models. It was in the stage of data augmentation that the dataset was diversified, and the models trained generalised to new, unseen data. Data preprocessing is made available through a set of Python scripts: data\_preprocessing.py, resnet50\_data\_preprocessing.py, and cnn\_data\_preprocessing.py, written separately to take care of a different part at the end of the preprocessing pipeline in a custom manner. There was an ImageDataGenerator function in the process with a set of options for augmenting and preprocessing images. More importantly, data augmentation was of essence since it did increase the size of the dataset although this is artificially caused by the creation of new images through various transformations hence helping to reduce overfitting and improve the robustness of the models. One of the basic augmentations that were used for augmenting the data in the project was rescaling, where the pixel values within the images are normalised by scaling them by a factor of 1./255. This was imperative in the sense that it very much helps the models converge faster during the training process through standard input data. This step changes the pixel values from the original range of 0 to 255 to 0 to 1.

Image horizontal and vertical flips were included in the dataset to account for face orientation and roll variations. These flips are new images flipping the original images, hence augmenting the dataset diversity by a factor of two. This had to be done so as to handle different face orientations, which makes the model robust to faces in any orientation. Further random augmentations applied were random rotations to a 20-degree maximum of only the frontal view. This again added more diversity to the dataset by changing the angle at which the faces had so far presented. Implementing a validation split ensures that models are not overfitting the training data and hence might generalise well. For that matter, 20% of the dataset was set aside for validation. This split then would allow the monitoring of how the model fared during training on unseen data, so that there would be an indication early enough as to how well it might perform in real world scenarios.

The dataset was partitioned between training and validation, and the images were then resized to work with the different models. This was performed to resize images to 224 by 224 pixels for the ResNet50 model, while it was resized to 256 by 256 pixels for the Xception and CNN models. It standardised the size of the input images and provides efficient training and inference in the architectures of the models.

During the training of the model, images were loaded in a batch of 32, a size that was equally efficient and memory friendly. During this classification task, the class mode was set in binary fashion, meaning that for every image, a label was assigned as either Fake or Real. The binary classification setup was very fundamental to the objective of this study, aimed at differentiating between real and manipulated facial images. In the training setup, the concept of the number of steps per epoch was central to setting up the training iteration. The number of steps per epoch was computed by taking the overall count of the training samples and dividing it by the batch size. Such a calculation was used to define how many batches of images would be treated within one epoch. Equally, the validation set size is the number of validation steps taken in order to ensure that, at every epoch, the model performance measurement is done in a consistent way.

**3.5 Model Selection And Training Procedure**

Three different detection models were evaluated for this proposal: Xception, ResNet50, and CNNs. Each model was selected based on its unique strengths and suitability for the task.

**3.5.1 Xception**

Xception was chosen due to the efficient utilisation of model parameters. Xception is a deep CNN-based architecture utilising depthwise separable convolutions, which separates the process of convolution into two steps: channel wise and spatial wise feature extraction (Sharma and Kumar, 2021). This would reduce computational complexity drastically and increase learning efficiency. Xception outperforms its predecessor, Inception-V3, in performance (Carnagie et al., 2022).The architecture of the Xception model involves a series of convolutional layers in extracting and processing features from images. For the task, the original top layers of the pre-trained Xception model were deleted to customise it into a network more appropriate for the binary classification of differentiating Fake from Real images. In their place, several custom layers were added. First was the GlobalAveragePooling2D layer, which reduced the dimensionality of the feature maps, though it still retained the most critical information. Then, a fully connected Dense layer with 1024 neurons and a ReLU activation function followed. This was to drive nonlinearity into the model. Hence, it will be in a position to learn complex patterns within the data. Finally, an output layer was added, which consisted of a Dense layer with the sigmoid activation function, thus completing the network architecture. This last layer would be essential for binary classification, as it returns a probability value of whether an image is classified as Fake or Real. This model was carefully designed during the training phase to avoid overfitting while making sure it attains high performance. It was trained on an Adam optimizer, which is extremely helpful in deep learning due to its adaptive learning rate features that aid in faster convergence. The learning rate used in this study was 0.001, which is a fairly standard value for the learning rate since it represents a good trade off between the learning pace and the stability of the training process. The loss function used here is binary cross-entropy, since this task is related to binary classification. This example trained the model on images provided by train\_generator. The model's performance was validated against another dataset provided by validation\_generator. The training was set to run for 10 epochs. This should be long enough to give the model sufficient time to learn underlying patterns in the data, but not so long that it will overfit. During this initial training phase, layers of the base Xception model were frozen, meaning that only the newly added custom layers were updated during training. This approach allowed the model to exploit features pre-trained on the Xception model while tuning the network for this task .Model development and training were possible due to the applications of tools like TensorFlow and Keras, providing an appropriate framework in which to implement deep learning models. High computational power in the execution of operations provided by TensorFlow was enhanced by the user friendly interface provided by Keras. This eased the process of model development, training, and evaluation in general through the workflow.

**3.5.2 ResNet50:** ResNet50 was chosen for its great handling of gradients, especially through the introduction of residual connections. This is a convolutional neural network performing image classification that rivals human accuracy (Koonce, 2021). ResNet50 has been very effective in identification tasks such as medical image analysis, object detection, and face recognition. It has five convolutional blocks that make it very accurate as implemented appropriately. It also decreases dimensionality through bottleneck layers, which can help reduce the vanishing gradient problem in ResNet50 (Wen, Li and Gao, 2019). This will improve the capabilities in image classification tasks. The architecture of the ResNet50 model was modified to be more appropriate to the requirements. The top layers were removed to give way to custom layers oriented towards binary classification. It is to have a GlobalAveragePooling2D layer whose role was to reduce the dimensionality of the feature maps outputted by the ResNet50 base without losing the most important information. The second Dense layer had 1024 neurons with a ReLU activation function to introduce some nonlinearity into the model. The ReLU activation function will play an important role in letting the model learn and represent complex patterns in the data.To prevent overfitting, a drop out layer with a rate of 0.5 was incorporated after the Dense layer. Finally, a Dense output layer with a sigmoid activation function was added. The output layer is designed to be specific for binary classification, returning a probability score of whether the image is Fake or Real. The training setup for this ResNet50 model was rather similar to the one used for the Xception model, so most of the parameters are consistent for different models used in this study. Adam optimizer was used for training with an adaptive learning rate that helps in efficient convergence. The learning rate was 0.00001, a balanced value that would trade off between the learning speed and stability. As binary classification problems make the binary cross-entropy loss fitting, it quantifies the difference between the predicted probabilities and real labels. In the course of training, the base layers of the ResNet50 model were frozen. This meant that only the newly added custom layers were trained, but the pre-trained layers of ResNet50 preserved the rich features learned from the huge ImageNet dataset. Freezing these layers helped leverage the powerful feature extraction capabilities of ResNet50 while fine-tuning the model for the task at hand of distinguishing between Fake and Real images. It is trained for 50 epochs, which is long enough to adapt to the new task but not so long that it starts to overfit.The tools used for developing and training the ResNet50 model were TensorFlow and Keras. The former offers a robust, yet flexible environment for the implementation of the Deep Learning model, while the latter provides a high level API that supports efficient model development, training, and evaluation. This ensures that the whole workflow remains clean throughout the project.

**3.5.3 CNN**

CNNs were considered as a result of their base role in computer vision. It excels in image segmentation, classification, and object detection (Lei, Pan and Huang, 2019). Convolutional and pooling layers are used in the CNN model for extracting local features and reducing spatial dimensions, which allows automatic hierarchical learning (Kaur et al., 2022). They yielded great success in facial recognition and are known for their ability to capture intricate patterns in visual data. This research proposed a custom-made CNN architecture with a series of convolutional and pooling layers to progressively extract and condense relevant features from the input images. The approach started with three convolutional layers as its principal feature extractors, all of which were configured with 32, 64, and 128 filters, respectively, using small kernel sizes to capture fine details in the images. During this process, max-pooling layers were applied after every convolutional layer. Through these max-pooling layers, drastic reduction in spatial dimensions of feature maps took place while retaining the most vital information. Again, this was meant to control the computational complexity of the model by avoiding overfitting. Following the convolutional and pooling layers, a flatten layer was used to transform the 2D feature maps into a 1D vector, making the data suitable for processing by fully connected layers. Then, it fed into a Dense layer of 512 neurons, following a ReLU activation function. The ReLU activation added non linearity to a network that would otherwise learn complex and abstract patterns from the input data. A Dropout layer with a rate of 0.5 is then applied to reduce overfitting by randomly setting a fraction of input units to zero during training. This was concluded with a final Dense layer, equipped with the sigmoid activation function, for the binary classification task at hand. It returned a probability value, allowing distinction between Fake and Real image classification. The model was compiled using the Adam optimizer, which is extremely popular in deep learning due to its adaptability of learning rate during training, which helps in faster convergence. The Learning rate was 0.001, a medium value that ensured both the learning rate and the stability of the training process. The binary cross entropy loss function was used, since the nature of the classification task is binary. This function is very effective in measuring the discrepancy between predicted probabilities and actual class labels. The accuracy metric was chosen for clear indications of the model's performance during training.Training was done using data generators, which allow turning up batches of images in training and validation, helping the model to experience a variety of data samples. The model was trained for 10 epochs, consistent with the time used in training other models used in this study. This period of training gave assurance to the CNN that it would learn the underlying patterns within this dataset without the risk of overfitting, hence guaranteeing good generalisation performance on new, unseen data. In the case of the model mentioned above, development and training were heavily based on TensorFlow and Keras. TensorFlow supplied the computational muscle needed to run the highly intensive operations involved in deep learning, whereas Keras created an easy to use front end interface for building and fine tuning the neural network. It was possible to implement the custom CNN efficiently from design to deployment with these tools.

**3.6 Evaluation Metrics**

The performance is evaluated in terms of a number of key metrics that together provide a full understanding of the quality of the model in differentiating between fake and real instances.

**3.6.1 Accuracy**

. Accuracy can be defined as the ratio of correctly classified instances, which means it includes both true positives and true negatives, against total instances in a dataset (Fisher, Lauria and Matheus, 2009) Mathematically, this is expressed using the following formula:

**(**Singh, 2023**)**

This formulation underscores accuracy in effect as a measure of how correctly the model identifies the target class and its absence. In effect, accuracy measures the ratio between the number of correct predictions made and the total of input across all classes.

**3.6.2 Precision**

Precision is defined to be the proportion of true positives against all the positive predictions made by the model (Barker et al., 2002) Another way of looking at it is the proportion of correct predictions in the total positive predictions yielded by the model. The mathematical formula for precision is as follows:

**​ (**Jason Brownlee, 2020**)**

​

This formula exemplifies that precision looks into the model's capability to ensure a very low level of false positives, which are when the model mispredicts the positive class.

**3.6.3 Recall**

Recall is defined as the proportion of actual positive instances of actual "Fake" videos that were correctly identified by the model (Lavie, Kenji Sagae and Jayaraman, 2004). Essentially, recall measures the model's ability to detect all relevant instances of the positive class; it ensures very few are misclassified. Mathematically, it is given by:

**(**Huilgol, 2020**)**

This formula focuses on the fact that recall is about minimising false negatives, in which the model fails to identify a positive instance.

**3.6.4 F1 Score**

F1 score is defined as the harmonic mean of precision and recall (Chicco and Jurman, 2020) This gives one overall metric, which puts these important facets of a model's performance together. By definition, the F1 score is expressed mathematically as:

(Sharma, 2023)

**3.6.5 Confusion Matrix**

Confusion matrix breaks down how well the model is performing. The outcome of classification in this table is categorised under four major quadrants: true positives, true negatives, false positives, and false negatives (Caelen, 2017). A confusion matrix thus provides a perfect insight into how well the model differentiates between classes, hence helping in an in-depth performance analysis.

The structure of the confusion matrix is as follows:

* **True Positives:** These are cases where the model correctly predicts the positive class (Schwenke and Schering, 2014) For instance, in tasks that aim at identifying fake pictures, the true positives would be the fake pictures the model has identified as fake.
* **True Negative**: These are the instances of the correct prediction of the negative class (Robinson, Keller and del Campo, 2022). For example, in fake image detection, the true negatives would be those real images that are correctly classified by the model as real.
* **False Positive:** Also known as Type-I errors, they occur when the model incorrectly predicts the positive class (Forstmeier, Wagenmakers and Parker, 2016) In other words, false positives are real images that are wrongly predicted to be fake by the model.
* **False Negatives:** This is a Type II error and occurs when the model misclassifies the negative class (Vadillo, Konstantinidis and Shanks, 2015) Therefore, in fake image detection, they would be those fake images which the model has misclassified as real.

**3.6.6 Test Loss**

It shows the value of the loss function, e.g. binary cross-entropy, for classification tasks, when the model is applied to the test dataset (Tate and Voss, 2024) This Holds the information about any presented model, never seen before in the training process, which serves as avital measure of how well the model generalises to never before seen new and other examples.

(Vishal Yathish, 2022)

The test loss is therefore relevant as it quantifies the potential of the model in how good this model is supposed to be at generalising well. , there are two main causes of high test loss:

**Overfitting:** This is an error in the model where it performs extraordinarily well on the training data and very poorly on test data (Hawkins, 2004) This occurs because the model is too complex and captures not only the underlying patterns but also the noise and specifics of the training data. The net effect is that it performs well on data used in training but at the cost of how it performs on new data.

**Underfitting:** When the model inadequately captures the underlying patterns in the training data, an under-fitted model shall actually be too simple to learn the complexities in the data (Bashir et al., 2020)Therefore, it will do poorly on both training and test datasets. Due to the model not having generalised properly, hence not learning the basic relations in data, we end up with a large test loss.

**3.6.7 ROC Curve**

ROC curve is a plot of the true positive rate, commonly referred to as recall, versus the false positive rate at different threshold levels(Kumar & Indrayan, 2011). It therefore gives an indication of the relative trade off between sensitivity and specificity. The AUC gives a single measure that refers to general performance. A higher AUC represents that the model is more effective in distinguishing between real and fake videos.

**3.6.8 Precision-Recall Curve**

The Precision-Recall Curve plots precision against recall at different thresholds, showing how well the model maintains the balance between identifying fake videos and avoiding false positives(Boyd et al., 2013) The area under this curve, PR AUC, gives insight into the model's performance in cases where a false positive is costlier than a false negative.

**3.7 Ethical, Legal, Professional, and Social Issues in Deep fake Technology**

Deepfake technology has raised numerous ethical, legal, professional, and social debates. These are core focus areas in the development and implementation of AI technologies, taking into consideration the misuse of the technology in many aspects.

**3.7.1 Misuse Of Technology**

One serious ethical issue involves the ways in which deepfake technology can be misused to spread misinformation, character assassination, or fraud (Freeman, 2007) This has raised an alarm among legal scholars and technologists alike, as it is now extremely easy to create such fake content. Several research papers call for the establishment of a regulatory framework that would avoid misuse. For that matter, ethical guidelines would be required to ensure responsible use; these should be coupled with public education on possible risks and measures that would ensure safety from deep fake harm.

**3.7.2 Ethical And Legal Frameworks**

Although many scholars have highlighted the ethical and legal ramifications of deepfake technology, surprisingly few concrete guidelines and regulatory schemes have been implemented. Calls for action emphasise that what is urgently required is detailed propositions that determine ethical boundaries and a legal framework for uses and development of deepfake technology (Jabr, 2021) Otherwise, misuse will still be one of the big issues.

**3.8 Project Planning**



Figure 3.8.1 Ghantt chart

Figure 3.8.1 depicts the Gantt chart employed in managing the research project. This chart indicates the scheduled timelines of tasks within the research, ensuring each phase was completed in its due course and in a timely manner. Some examples include data collection, preprocessing, model implementation, and evaluation. By allowing visualisation of these activities, the Gantt chart proved beneficial for tracking progress while focusing on deadlines so as to keep the project on track.

It also points out how several activities were divided and carried out in parallel, where possible, such as data preprocessing with model selection. Each of the models, including ResNet50, Xception, and CNN, had its own implementation and testing timeline, with distinct intervals for training, evaluation, and comparison. In such a way, this structured approach presupposed covering all aspects of research within the set timeframe and hence promoted efficiency, helping to meet project milestones as scheduled.

**3.9 Conclusion**

Chapter 3 presents a well structured methodology in conducting performance evaluation on deepfake detection models using ResNet50, Xception, and a custom CNN. The chapter opens with a clear experimental design that centres on the comparative study of such models with the express purpose of determining their various strengths and weaknesses. Variety was added to the dataset extracted from Kaggle, ensuring balance in both real and fake media, a common issue faced while working with deepfake detection datasets. The chapter also explains the preprocessing steps that were undertaken in extracting frames from videos, making sure that the dataset is diverse through augmentation techniques like horizontal and vertical flips, random rotations for model generalisation, scaling, and resizing of images to suit the input requirements of each model. Preprocessing included scaling and resizing images to fit the input requirements of each model, ensuring consistency in training and evaluation. Second, the training processes were well monitored, split between the training and validation sets to keep track of model performance without overfitting. Early stopping, dropout layers, and batch normalisation were some of the techniques used to balance the learning efficiency against robustness of the models. Specific hyperparameters tuned in each model include learning rates, batch sizes, and optimization algorithms, among others, like Adam. The metrics used for the research were listed along with the formulas. In general, the methodology in Chapter 3 sets the stage by providing an appropriate framework for the assessment of the current status of deep fake detection. Ethic and professional considerations were discussed along with presenting a gantt chart showing how the research progressed.

# **Chapter 4 Implementation**

**4.1 Introduction**

The detailed implementation of the three deepfake detection models, ResNet50, Xception, and CNN, is presented in the following sections. Section 4.2 overviews the structure and training process of ResNet50, whereas Section 4.3 presents the implementation details of Xception. Finally, Section 4.4 discusses the custom-designed CNN, while pointing out the distinctive features with respect to design and training.

**4.2 Implementation Of ResNet50**

1. **Model Architecture**

The architecture is imported from tensorflow.keras.applications with pre-trained weights from ImageNet. In order to make the model appropriate for binary classification, the top fully connected layers are removed from the ImageNet classification task by setting the argument include\_top to False. A GlobalAveragePooling2D layer follows the ResNet50 base model to reduce the dimensionality by averaging each feature map, converting the multi-dimensional data into a one-dimensional format. This helps in good processing and enhances model performance. A Dense layer with 1024 neurons and ReLU as its activation function introduces non-linearity, enabling the model to learn complex patterns for distinguishing between real and fake images. Thereafter, a Dropout layer of rate 0.5 is introduced to avoid overfitting. Finally, the output layer consists of a single neuron with a sigmoid activation function, which is suitable for binary classification; output values fall between 0 and 1, representing the probability of an image being real or fake.

1. **Trainable And Non Trainable Layers**

In this work, all layers within the ResNet50 base model are non-trainable, with their weights frozen to those pre-trained on ImageNet. The feature extraction capability of ResNet50 remains intact. However, the trainable custom layers added atop of this base model allow their weights to be adjusted during training. This combination would ensure the model takes advantage of pre-trained feature extraction while fine-tuning for the specific task of deep fake detection.

1. **Training Process**

Data augmentation is done to make the training data more diverse and to enhance the model's generalising capability to new data. The images are rescaled by 1/255, which brings the pixel values within the range of 0 to 1. Random horizontal and vertical flips and 20-degree rotations are done to make the model invariant to these transformations. A 20% validation split is used to split the data for model validation to ensure overfitting is monitored.

1. **Hyperparameters**

The optimizer used for training is Adam, with an initial learning rate of 0.00001. This value of the learning rate provides a compromise between rapid convergence and overshooting around the optimum. A batch size of 32 involves processing 32 images through the model before weights updates. The number of epochs used for training is 50, which represents one complete pass through an entire training dataset.

1. **Validation Strategy**

A validation strategy is implemented to monitor the performance of the model in generalising well after every epoch. The validation set comprises 20% of the entire dataset, and its performance is checked throughout training. Overfitting is monitored to ensure that the model performs well on unseen data. Steps per epoch are estimated as the total number of training samples divided by batch size. Similarly, for the validation set, pre-computed validation steps are performed to keep the same evaluation protocol throughout the training.

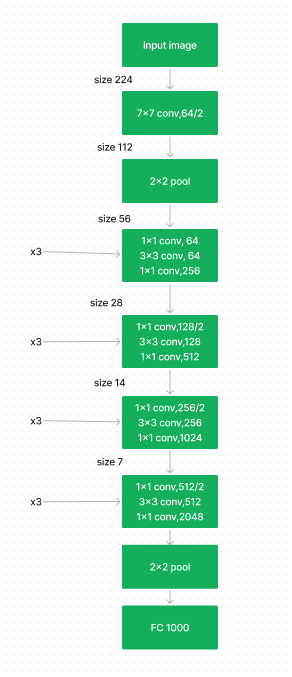


Figure 4.2.1 ResNet50 Model Architecture

**4.3 Implementation Of Xception**

1. **Model Architecture**

The Xception model has been initialised with pre trained weights from ImageNet via tensorflow.keras.applications. In addition, setting the argument include\_top to False excludes the top layers of the Xception model, which were used for classification on ImageNet. This allows one to add the custom layers on top for deep fake detection. A GlobalAveragePooling2D comes after the base model in order to reduce data dimensionality without the loss of critical information. Next, add a Dense layer with 1024 neurons and the ReLU activation function for introducing non-linearity into the model and helping it learn complex patterns. Finally, add the output layer for binary classification between real and fake images, comprising one neuron with the sigmoid activation function.

1. **Trainable And Non-Trainable Layers**

This implementation freezes all layers of the Xception base model, just as done with ResNet50. The pre trained weights are kept intact to preserve general features learned from ImageNet. Only custom layers are set as trainable so that the model gets tuned for deepfake detection.

1. **Training Process**

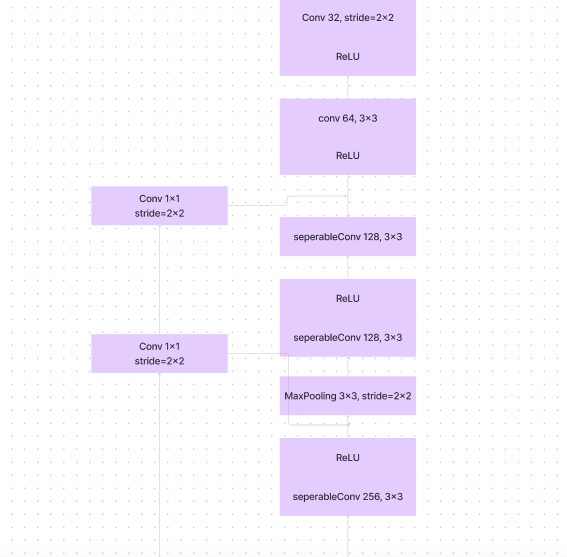
Data augmentation increases the diversity of the training data and helps the network generalise more effectively. All images are scaled to values between 0 and 1 using a factor of 1./255. Also, data augmentation consists of random horizontal and vertical flips and 20-degree rotations add variability to make the model resilient against different orientations. The actual dataset is then divided into training and validation subsets based on an 80-20 split.

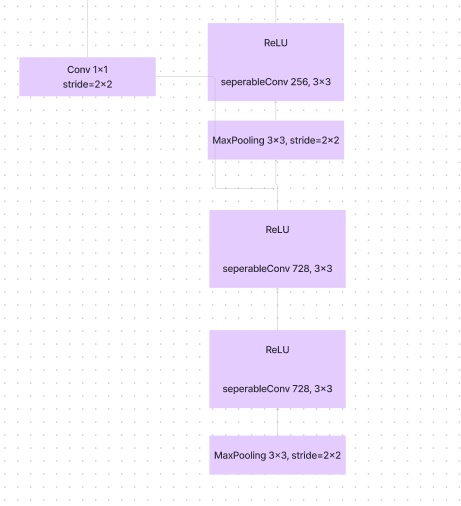
1. **Hyperparameters**

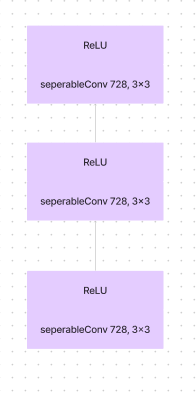
The model uses the Adam optimizer, with a learning rate of 0.001. The batch size is 32, so the model updates its weights after every 32 images it goes through. The model is run for 10 epochs, where one epoch is equal to one complete pass over the training dataset.

1. **Validation Strategy**

This model uses a validation set at the end of each epoch, using 20% of the data. That estimates steps per epoch and validation steps by dividing a total number of samples by batch size, to monitor generalisation capability in order not to overfit.







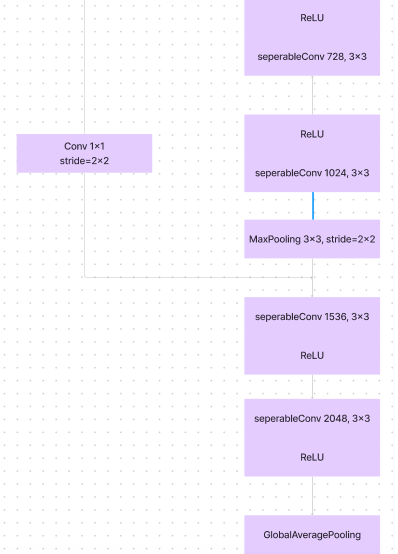


figure *4.3.1* Xception model architecture

**4.4 Implementation Of CNN Model**

1. **Architecture Of Model**

The proposed CNN architecture for deepfake detection is a configuration comprising multiple convolutional layers that successively extract the features of the input images. The input image size is taken as (256, 256, 3). This starts with a Conv2D layer, which is configured with 32 filters, a kernel size of (3, 3), and ReLU for activation. A MaxPooling2D layer can be used to reduce the spatial dimensions, having a pool size of (2, 2). This is furthered with increased filter sizes of 64 and 128 to capture features of higher complexities. The extracted features are then flattened by the Flatten layer to achieve one-dimensional vectors from two-dimensional feature maps. Next, a Dense layer of 512 neurons with ReLU for the introduction of nonlinearity and a Dropout layer with a rate of 0.5 is added to the network to avoid overfitting. The output layer is one neuron that carries out binary classification with the help of the sigmoid activation function.

1. **Trainable And Non-Trainable Layers**

Unlike ResNet50 and Xception, this model does not depend on pre-trained layers. Hence, all layers are trainable from scratch, which allows them to learn specific patterns and features in this particular deep fake detection problem.

1. **Training Process**

Data augmentation also includes rescaling with 1./255, random flips, and 20-degree rotations. It is divided as well-80% to training and 20% to validation-so that it will generalise well.

1. **Hyperparameters**

It implements the Adam optimizer with a learning rate of 0.001. Batch size is kept at 32, and the number of epochs for training are set to 10. The loss function used will be binary cross entropy.

1. **Validation Strategy**

Similar to the other models, a 20% validation set is used to validate the model at each epoch. The number of validation steps is calculated by dividing the number of samples in the validation set by batch size to keep it in sync with the training process.

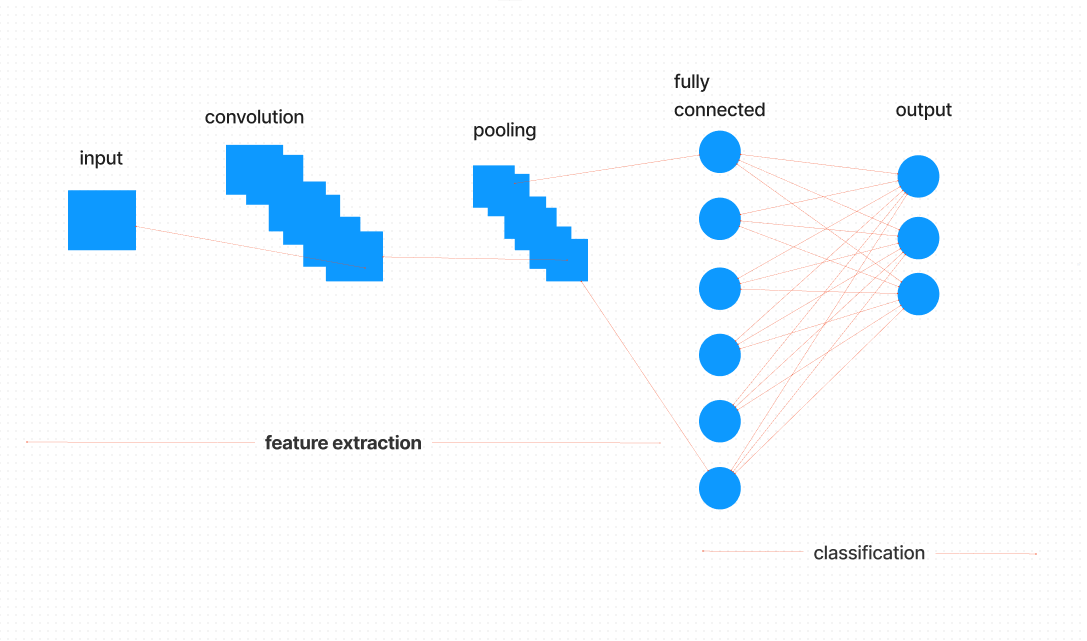


Figure4.4.1CNN model architecture

**4.5 Conclusion**

Implementation of deepfake detection models has shed light on their performances. Each model has its own strengths, though a few challenges emerged, more particularly with respect to accuracy and generalisation. The comparison of the results based on these models has been done in the next chapter, showing the real performances of each.

# 

# 

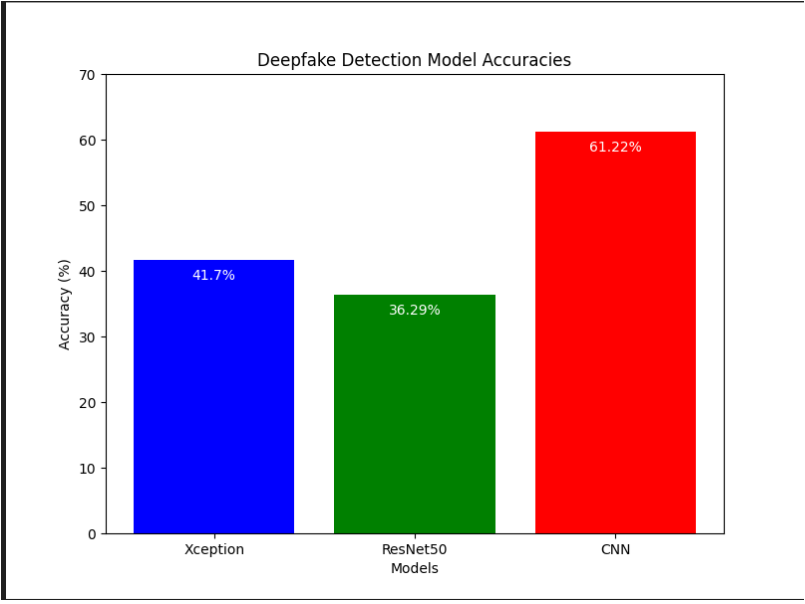
# **Chapter 5: Results and Discussion**

**5.1 Introduction**

This chapter presents the results obtained from the deepfake detection models implemented in the previous chapter. An In depth analysis and comparison of the results is done in Section 5.2 will each metric that was listed in chapter 3.

**5.2 Comparative Analysis of Model Performance**

**5.2.1 Accuracy**

****Figure 5.2.1 accuracy comparison

The CNN model gave the highest accuracy among the models with a score of 61.22%. Although the performance is higher than random guessing, which would be at around 50% in a balanced binary classification task, it still suggests that the model struggles with the deep fake detection task. This low accuracy would therefore insinuate that the CNN model fails in the capture of complicated patterns, which are necessary in the effective discrimination between real and fake images. This further pinpoints the fact that though the CNN is extracting features, it is not sufficient. In order for the model to improve in performance, certain tuning may be required, more data, or even replacing the architecture as a whole with one better suited for the task. In contrast, the Xception model had an accuracy of only 41.27%, which is far poorer than that achieved by CNN and very close to random guesses. This hints that the dataset was really hard on the Xception model, casting serious doubts over its adequacy for deep fake detection in its current form. This low accuracy denotes that Xception is not appropriately grasping the distinguishing features between real and fake images, due to insufficient training, poor preprocessing of the data, or perhaps an architecture not in line with the nature of the data. This performance points to a need for more extensive training, better data preprocessing, or reconsideration of the architecture of the model to do better. The ResNet50 model had the lowest accuracy score among the three. It attained only 36.29% accuracy, which is even below the level of random guessing.This poor performance could have resulted when ResNet50 was either overfitting, underfitting, or simply failing to generalise from the training data onto the test data. The low accuracy implies that ResNet50 may not be suitable for the task at hand without radical changes. The issues can be due to poor training data, incorrect selection of hyperparameters, or the inability of the model to learn significant features from the dataset. These results indicate that the ResNet50 model used requires heavy modifications, or the complete replacement of the model for one better suited to the deep fake detection challenge. While the CNN model has shown some promise, the Xception and ResNet50 models did not return good performance. Further work is needed to enhance the efficiency of these models. This could involve changing the architectures, cleaning up the dataset, or applying other metrics that might be necessary for better understanding.

**5.2.2 Recall**

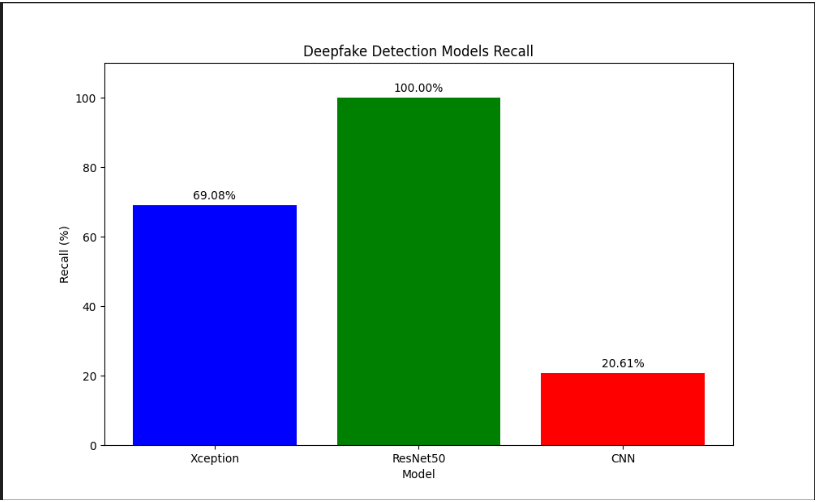
****

Figure 5.2.2 recall comparison

The Xception model obtains a recall of 69.08%, which indicates that 69% of the media have been correctly identified by this model as being deepfakes. This would suggest that the model performs reasonably well in identifying deepfakes but does not identify approximately 31% of actual deepfakes. That is a promising result, but it underlines some room for improvement. This is especially relevant in applications where a single failure in deepfake detection could have disastrous consequences, such as fraud prevention systems or media verification systems. The relatively high recall speaks well of the Xception model for its powerful feature extraction capability, but of course, improvements in training, such as augmentation or dealing better with complex variations of deepfakes, are further refinements that could improve the detection rate. The recall of 100% from the ResNet50 model implies that all deepfake media were identified correctly in the dataset. From a recall perspective, this is an ideal situation, as no deepfakes were missed by the model. This Suggests considerable value in the case of deepfake detection, where the false negatives or the number of missed deepfakes are to be brought down to zero at all costs. However, perfect recall generally comes at the cost of precision. For this reason, along with recall, it is useful to investigate the precision of the model to ensure balanced performance on both metrics. However, the recall score for the ResNet50 model suggests that it is somewhat robust and suitable for deepfake detection in contexts where capturing every instance of deepfake content is crucial.

Contrasting that, the custom CNN model has the lowest recall score, with only 20.61% detection of deepfake videos. This means that it misses almost 80% of the actual deepfake media. This again suggests that the model generalises poorly to recognize deep features in videos. As poor as this recall is, it signals a serious limitation in the architecture of the model due to either inefficient feature extraction or not enough depth/sufficient training. This very low recall of the custom CNN indicates that the current model is unsuitable for reliable deepfake detection, at least in some applications in which overlooking deepfakes may have grave ethical, legal, or security repercussions.

**5.2.3 Precision**

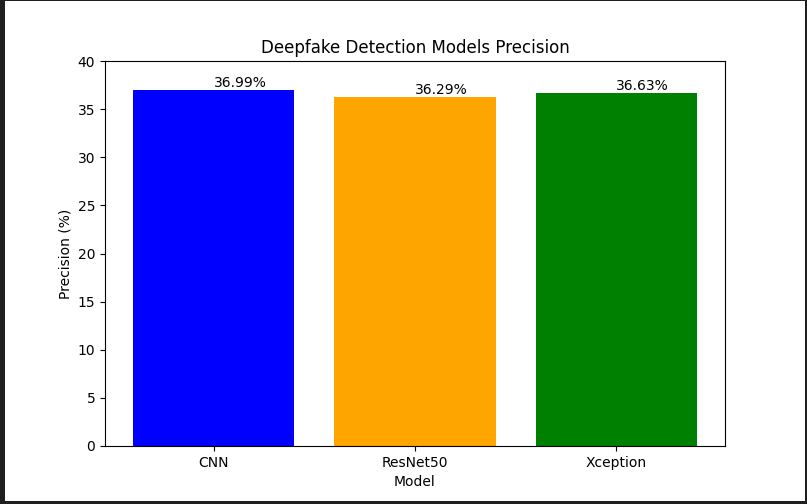
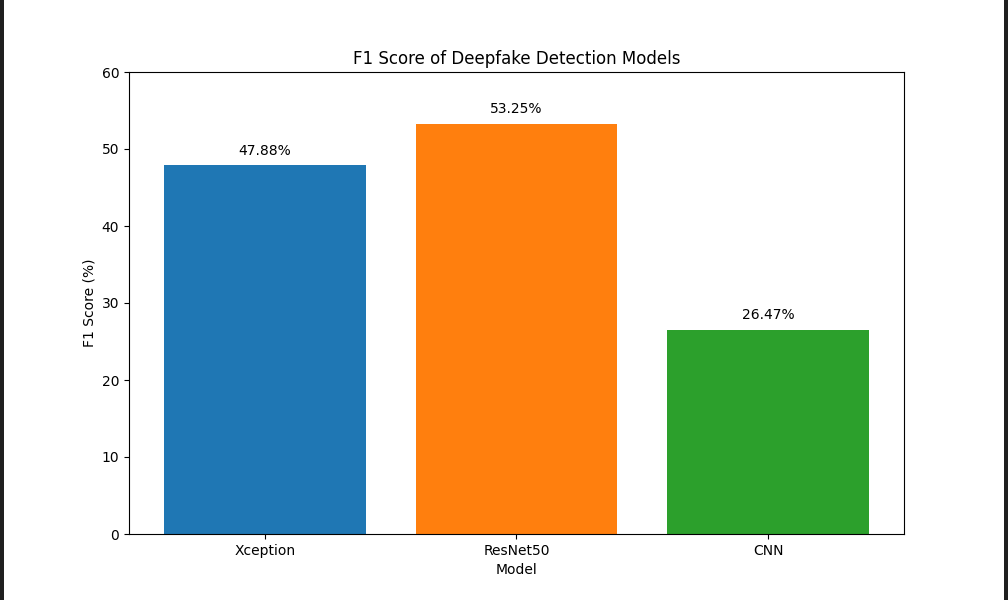
****

Figure 5.2.3 precision comparison

The precision values for the three models are quite close to each other. That means there is no significant superiority of one model over another in terms of precision. Such similarity in performances could suggest that the models struggle similarly in differentiating between deepfake and real media. However, their very low precision of about 36% shows that all models flag a large selection of real media as deepfakes. This means that a reasonable ratio of positive predictions are actually false positives or real media flagged as deepfakes. The closeness of the precision scores across models would also suggest that these models are making similar kinds of misclassifications.

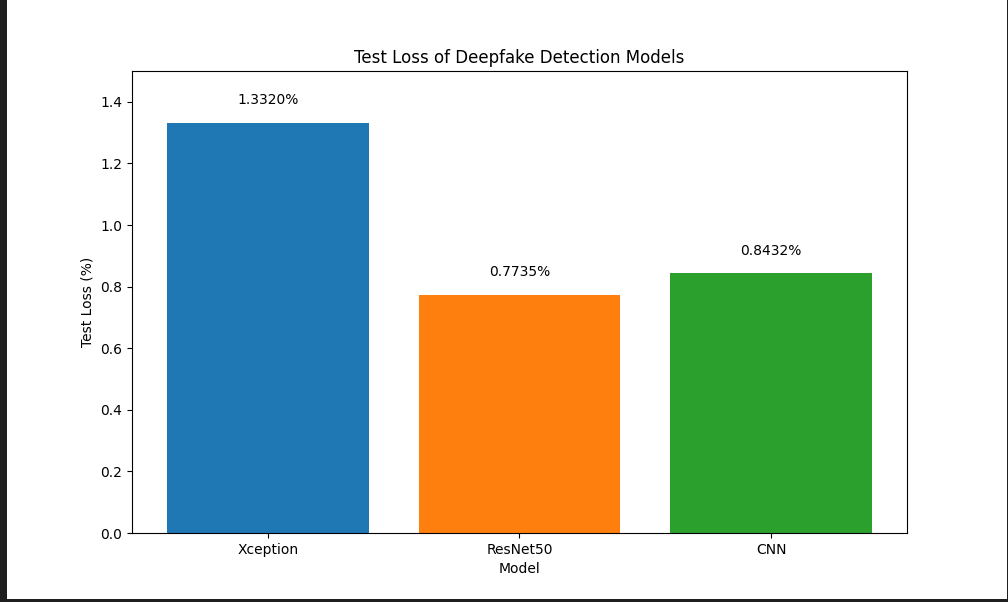
**5.2.4 F1 Score**

****

**Figure 5.2.4 f1 score comparison**

The best performance, considering the F1 score, was achieved by ResNet50 with 53.25%, followed by Xception with 47.88%, whereas the custom CNN had 26.47%. While ResNet50 and Xception performed better than the custom CNN, these results are still way below optimal. An F1 score of around 50% means the models struggle to consistently identify deepfakes without losing either precision or recall. With a score of 53.25%, ResNet50 has achieved a relatively successful detection of deepfakes but has misclassified quite a substantial number of real videos as fakes or missed part of the deepfakes. While ResNet50's deep structure allows it to catch complex features, the fact that it scores 53.25% means half of its predictions went wrong with either false positives or false negatives. Although the Xception model was fine tuned for image classification tasks, it produces an F1 score of only 47.88%, which underlines further the difficulty inherent in the deepfake detection task. Nearly half of its predictions were leading to misclassifications; therefore, the model fails to generalise well on this dataset. While the architecture uses depthwise separable convolutions to enhance feature extraction, features alone are not sufficient to push its performance in an acceptable range for real world deployments in high stake deepfake detection scenarios. The performance of the custom CNN is far worse, for it has an F1 score of 26.47%, indicating how it struggles to find a balance between precision and recall. The low score shows that this model can't capture the required features that will enable it to differentiate between real and fake media; hence, the high number of incorrect classifications.

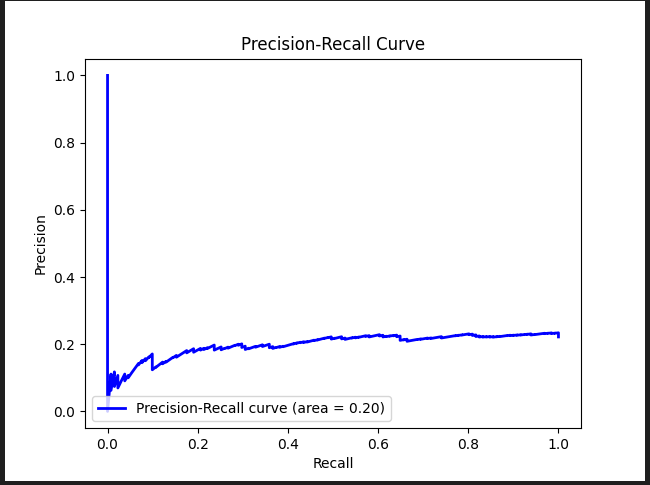
**5.2.5 Test Loss**

****

**Figure 5.2.5 test loss comparison**

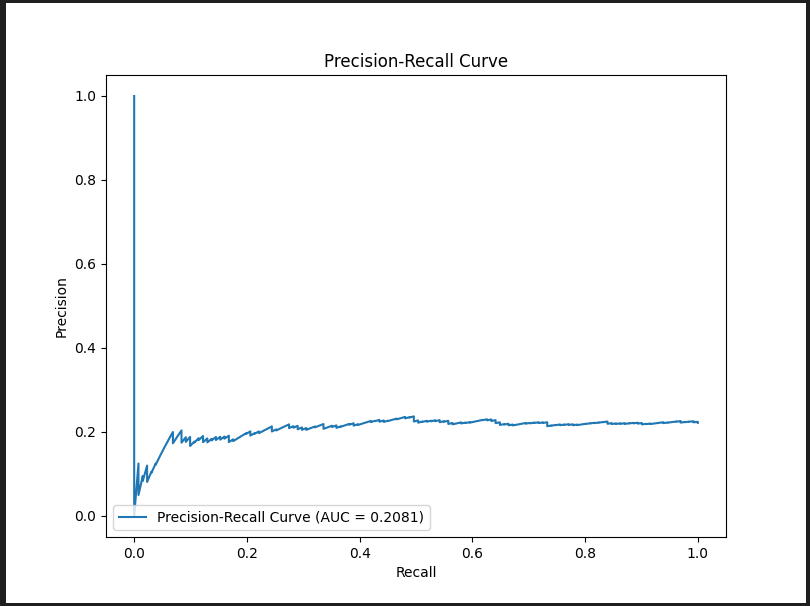
As shown, the highest test loss of 1.3320% was by the Xception model, which suggests that the model struggled more than the others in distinguishing real and deepfake media. This could be a result of the Xception models complexity. On the other hand, ResNet50 recorded the lowest test loss at 0.7735%, proving better generalisation than Xception. This agrees with the design of residual connections to maintain the flow of gradients and reduce the vanishing gradient problem. The loss for the custom CNN model is 0.8432%, higher than ResNet50 but still better than Xception. Considering this is a more simplistic model, this result against Xception should be noted at great value. In other words, even less complex architectures are competitive in deepfake detection with proper design and tuning. Concretely, ResNet50 has the best performance in terms of test loss, while Xception is underwhelmingly bad considering it's really high model complexity. The custom CNN sits somewhere in the middle, which might just make it worth being a more efficient variant with further refinement.

**5.2.6 Precision Recall Curve**



**Figure 5.2.6 Xception precision recall curve**

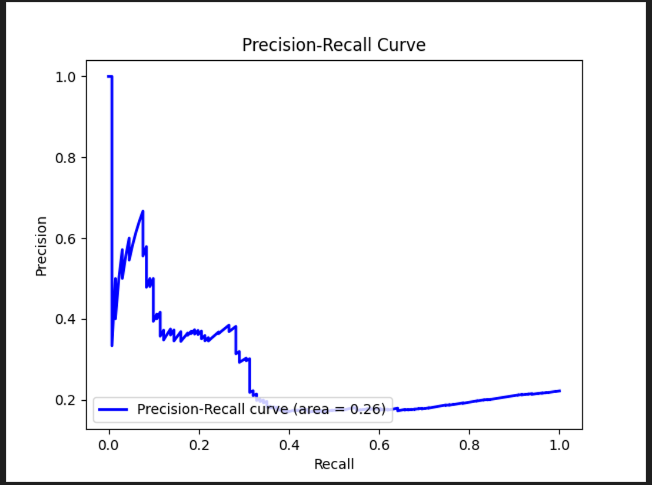
The PR AUC is 0.20, thus indicating poor model performance. The precision goes down as recall increases. Indicating that the model can never achieve a balance in identifying true positives correctly, deepfakes, while avoiding false positives ultimately failing to provide decent precision and recall simultaneously. The figure shows high precision near zero then decreases rapidly and then flat lines at about 0.2, reflecting that without misclassifying some real media, the model cannot predict deepfakes accurately.



**Figure 5.2.7 ResNet50 precision recall curve**

The figure shows that the ResNet50 model has a AUC curve of 0.2081,similar poor

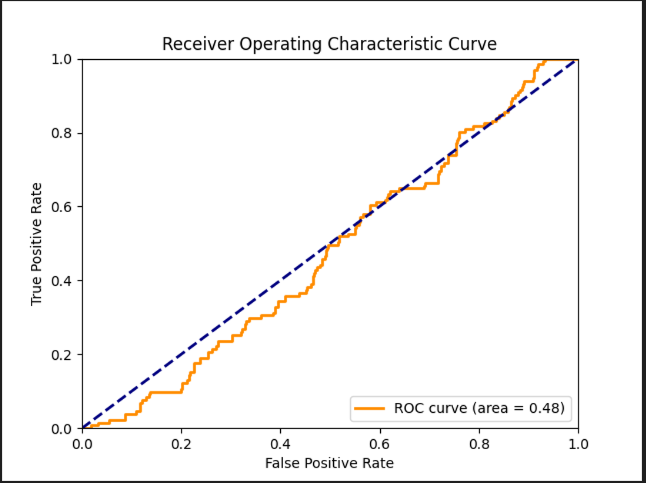
performance as with the Xception model. The curve is initially high with Precision for low recalls but quickly drops and stabilises at lower precisions for most of the recall range. This points to difficulties of the model in striking a balance between precision and recall, very likely flagging many real media as deepfakes and missing out on quite a number of real deepfakes-the case of low recall.The nature of this curve shows that while the model increases recall, precision decreases significantly. This means that it cannot correctly classify a large proportion of deepfakes while misclassifying authentic content.



**Figure 5.2.8 CNN precision recall curve**

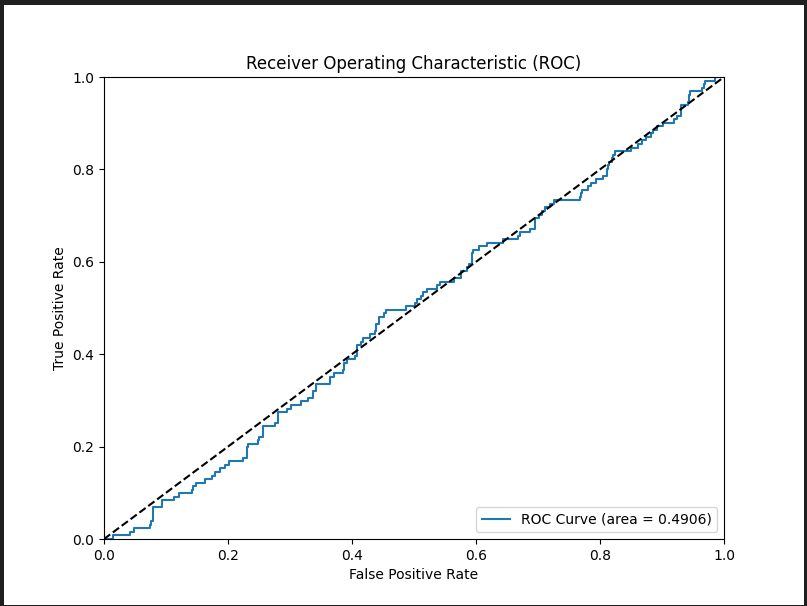
The AUC of the CNN model is 0.26, which is really indicative of its poor balance between precision and recall from the image shown of the precision-recall curve. While it outperformed both the ResNet50 and Xception models by a margin, at 0.2081 and 0.20, respectively, this is still indicative of suboptimal performances for the models in detecting deepfakes. The low precision of most of the recall values indicates that while the model tried to capture more true positives, it struggled to sustain actual correct predictions. Indeed, it suggests that even though the CNN slightly outperformed the other models, all models handled the dataset poorly and needed further refinement or investigation of more advanced architectures or techniques.

**5.2.7 Receiving Characteristic Curve**



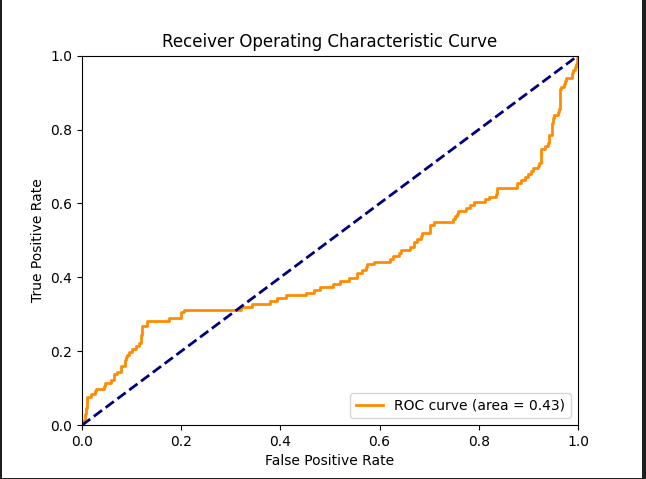
**Figure 5.2.9 Xception roc curve**

An AUC value of 0.48 suggests that this model is only performing a little better than random chance. The AUC of 0.5 represents random guessing, and here, since the AUC is lower than 0.5, it means the Xception model fails to differentiate between true positives and false positives effectively. This poor result agrees with the earlier observation from the precision-recall curve with an AUC of 0.20, which showed the difficulty faced by the Xception model in detecting deepfakes effectively in this dataset.



**Figure 5.2.10 ResNet50 roc curve**

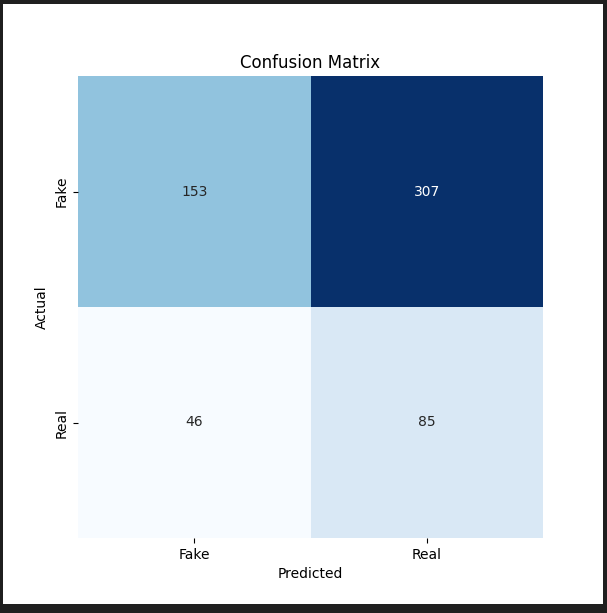
with an AUC of 0.4906, a value extremely close to 0.5. That means the model performance is basically almost as good as random guessing. Much like the Xception model, ResNet50 does a poor job in distinguishing true positives from the false ones since its entire curve remains very near the diagonal.This low AUC underlines how the ResNet50 model fails to achieve high predictive power on this dataset. It agrees with the majority of the metrics used, which also reported poor performance.



**Figure 5.2.11 CNN roc curve**

This figure represents the ROC curve of the CNN model, with an AUC of 0.43. That is to say, the CNN model is capable of distinguishing true positives from false positives in performance even lower than random guessing. In the same way as ResNet50 and Xception models, this model also has an AUC much less than 0.5, showing poor performance in classifying data, reflecting the poor performance of the CNN model on deepfake detection. While CNN was slightly better than ResNet50 and Xception in terms of precision-recall, the low AUC value obtained in this ROC analysis highlights that this model is still performing poorly overall.

**5.2.8 Confusion Matrix**



**Figure 5.2.12 Xception confusion matrix**

**True Positives (TP)**: 153 (Fake predicted as Fake)

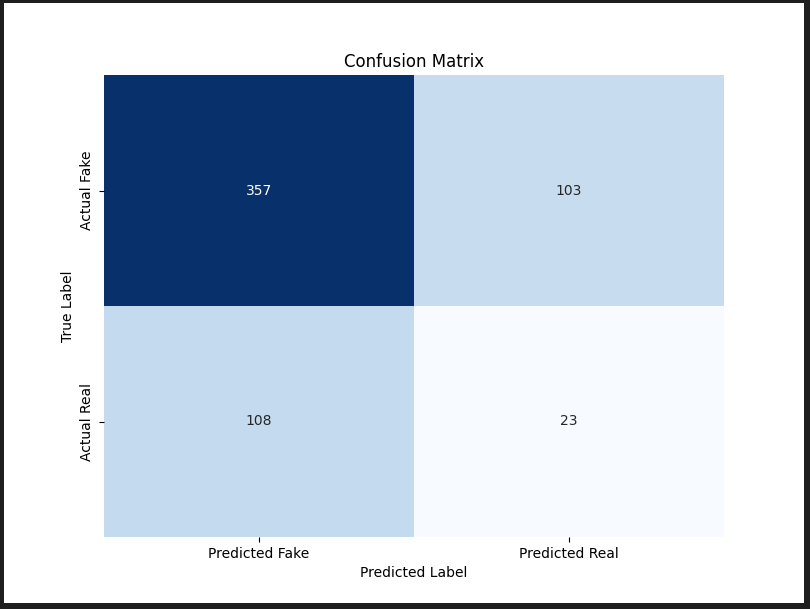
**True Negatives (TN)**: 85 (Real predicted as Real)

**False Positives (FP)**: 307 (Fake predicted as Real)

**False Negatives (FN)**: 46 (Real predicted as Fake)

**False Positives (307) vs. False Negatives (46)**

The Xception model shows a high number of false positives, meaning it classifies many fake videos as real. This is a big concern with regard to deep fake detection, since it may allow a good quantity of deep fakes to bypass this without raising any alarms. Similarly, low false negatives hint at the fact that when it actually detects something as being fake, most of the time, it is correct. However, its general dependability is curtailed by the high rate of false positives.



**Figure 5.2.13 ResNet50 confusion matrix**

**True Positives (TP)**: 357 (Fake predicted as Fake)

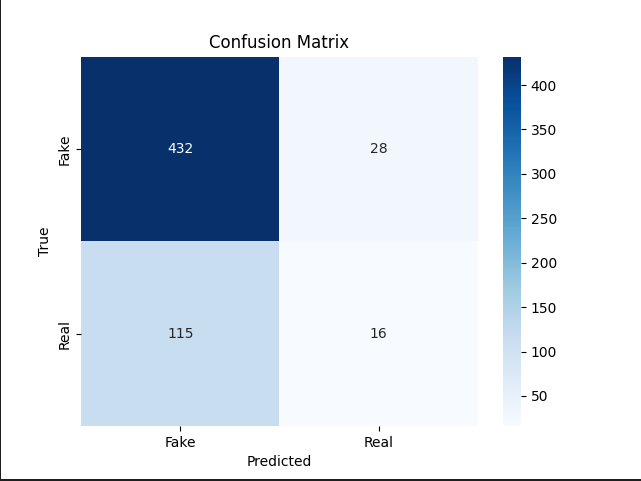
**True Negatives (TN)**: 23 (Real predicted as Real)

**False Positives (FP)**: 103 (Fake predicted as Real)

**False Negatives (FN)**: 108 (Real predicted as Fake)

**False Positives: 103 vs. False Negatives: 108**

The matrix shows that, although the ResNet50 model is reasonably balanced in terms of error distribution between false positives and false negatives, slightly more weight goes towards false negatives. That means this model will misclassify real videos as fake almost as often as it misclassified fake videos as real. Consequently, due to this balance, the model does an average job and does not particularly excel in minimising one type of error over another.



**Figure 5.2.14 CNN confusion matrix**

**True Positives (TP)**: 432 (Fake predicted as Fake)

**True Negatives (TN)**: 16 (Real predicted as Real)

**False Positives (FP)**: 28 (Fake predicted as Real)

**False Negatives (FN)**: 115 (Real predicted as Fake)

**False Positives: 28 vs. False Negatives: 115**

CNN has a low false positive rate which is quite good since most fake videos are identified as such, but it does have a relatively high false negative rate. This indicates that it often misclassified real videos as fake. While CNN is very cautious in terms of not letting deep fakes slip through, doing so at a low FP comes with the cost of potentially lessening the validity of real content.

**5.3 conclusion**

This chapter compared the performance of three models for the detection of deepfakes ResNet50, Xception, and CNN. None of the models showed desirable performance overall across all metrics. rather, they each showed specific strengths and weaknesses. CNN had the highest accuracy with low recall, whereas ResNet50 achieved perfect recall but at the expense of a high false-positive rate. Xception balanced recall with feature extraction but struggled with high false positives. It is indicated by these findings that further real-world applicability of deepfake detection models would be achieved with more advanced architecture, larger datasets, and fine-tuning of hyperparameters.

.

# 

# 

# 

# 

# 

# **Chapter 6: Evaluation against research questions**

**6.1 Introduction**

This chapter would answer the research questions developed in Chapter 1 with the view of the results of the implementation in Chapter 5. In Sect. 6.2, the effectiveness of the models are compared. Section 6.3 goes over the advantages and disadvantages of each model. Section 6.4 outlines enhancements based on insights gained from the analysis. This chapter will conclude with a summary of critical evaluations and future directions in Section 6.5.

**6.2 Effectiveness of Detection models in identifying deep fakes.**

**How do Xception, ResNet50, and CNN compare in their effectiveness in identifying deep fakes?**

The evaluation of the deepfake detection models revealed significant limitations in their effectiveness, with none of the models achieving optimal results across the key metrics. While each model had its own strengths, their overall performance was far from ideal, indicating that more work is needed to improve the detection of deepfakes. It is further supported by the fact that the Xception model is known for feature extraction from images quite efficiently and resulted in an accuracy as abysmal as 41.27%, which does not classify between the real and the fake media effectively. Although this is an advanced architecture with support for depthwise separable convolutions, low accuracy itself shows its inability to catch up with the depth of manipulations brought about by deepfakes. However, Xception showed some strength in recall at 69.08%, where it actually marked a good number of deepfakes correctly. All this came with a great number of false positives, meaning it marked plenty of real images as fake. This is an indication that this imbalance between precision and recall makes the Xception model less ideal in media verification systems or fraud detection, where accuracy is of major importance. The ResNet50 model achieves the best recall score of 100%, but overall is performing worst among the three models. Accuracy was lowest of the three models at 36.29%, barely better than random guessing. While catching literally every deepfake sounded pretty promising for ResNet50, it turns out this model has a high rate of false positives, which brings down its utility in real world applications. In this case, perfect recall for the model means that it flagged all deepfakes but misclassified many real images, making it not practical for use where both precision and recall are of equal importance. That means that ResNet50 has a strong drawback concerning the trade off of deepfakes identification for minimising false positives, which means that it identifies deepfakes but at the cost of excessive misclassifications.

The CNN model was far superior to the other two, with an accuracy score of 61.22%. Whereas this would point to a better performance of CNN in telling the difference between real and fake media, its recall rate was very low, standing at 20.61%, missing a great portion of deepfakes. This is an important weakness because with such low recall, it implies that CNN has poorly generalised the features distinguishing deepfakes from natural videos therefore, it misses many. The improvement in accuracy, compared to Xception and ResNet50, is promising however, the model can't catch up with the large number of deepfakes, which makes it unsuitable to work in high stakes environments where missing a deepfake will come with serious consequences. On the whole, the performance for these models remained far from ideal, where each metric served to indicate key flaws. None of them could provide such a balanced performance in terms of accuracy, precision, and recall, which is much desired for effective detection. The poor results clearly indicate the complexity of the deepfake detection tasks and further point toward the need for more developed approaches with improved datasets and architectural refinement to derive models that could accurately tell a real from a fake.

**6.3 Advantages and Disadvantages of Detection models.**

**What are the advantages and disadvantages of each model when it comes to identifying deep fake content?**

Each of the detection models has a certain set of distinct advantages and disadvantages. The Xception model has good recall and is very efficient in identifying most deepfakes due to an efficiently designed architecture for feature extraction. It, therefore, works effectively with media that has subtle manipulations. However, it carries the disadvantages of relatively poor accuracy and a high rate of false positives,misclassifying real content as fake. ResNet50, having the highest recall, effectively catches almost all deepfakes. As a very robust architecture, it fits well for detailed feature recognition tasks. However, this model struggles with very poor accuracy and a high number of false positives. Hence, it is less practical to be applied in real life for cases where precision is essential. While the architecture is simpler,CNN manages to attain the highest accuracy among the three, arguably giving a much better balance in differentiating real from fake contents. Despite this,the low recall means it misses most deepfakes, which again is not as reliable as capturing most, if not all, deepfakes in situations that demand it.

**6.4 How can the performance of deepfake detection models be enhanced based on insights gained from our analysis?**

The analysis of the deepfake detection models highlights key areas for improvement to enhance their overall performance and reliability. With the poor metric scores overall across the models,it suggests the models require further development. One area that could be improved would relate to the enlargement of datasets. More variation and size would add a lot to the generalising capability of the models while considering the training datasets. The models are not able to properly detect more complex deepfakes,this could have been due to the dataset that was used at the time of training. While deepfakes are constantly evolving, this would increase diversity and representativeness in the dataset of real and manipulated media for the models to learn from. Therefore, they need to be trained on a wide variety of deepfake techniques and real media variations so that they can reliably find complex manipulations in reality. Other important reasons for the enhancement in performance could be the fine tuning of the architecture of these models. While current architectures are competent, they do not seem to be optimised for the complexity associated with deepfake detection. Increasing the capacity of these architectures by adding more layers, or using hybrid model approaches could further enhance their potential in the efficient extraction and processing of relevant features. Fine-tuning such architectures through focus limitation to particular specific challenges presented by deepfakes for example, subtle facial manipulations or audio-visual inconsistencies,could lead to an enhancement of general deepfake detection accuracy.lastly, having regular model updates can help the exciting models detect evolving deepfakes by being up to date with evolving techniques.

**6.5 Conclusion**

In summary, this chapter critically evaluated the effectiveness of the three models. The relative analysis has shown that while each model was performing well on certain aspects, none of them gave a balanced or optimal performance across the key metrics of accuracy, precision, and recall. The ResNet50 model performed very poorly, especially in terms of its accuracy, despite high recall, which limits its practical application. While the Xception model featured an effective efficiency in feature extraction, high rates of false positives were identified with it. Though CNN provided better accuracy, it couldn't generalise well to detect high percentages of deepfakes which was evident from its low recall rate. From the results of the evaluation against research questions, it could be deduced that there is plenty of room for improvement in detection models, Thus, the results of this work point out the difficulties in developing reliable deepfake detection models and call for further sophistication in architecture and techniques with respect to real world applications.

# 

# 

# **Chapter 7: Conclusion And Future Work**

**7.1 Introduction**

This section will draw upon and summarise the findings, the contribution to knowledge in the areas relevant to deep fake detection, discussion on limitations of the research, and recommendations for further research. It has been done to reflect upon research outputs, to highlight valuable additions done with this work, and also to suggest areas where further information should be gathered in order to enhance the understanding and capabilities of deepfake detection.

**7.2 Summary Of Key Findings**

The custom CNN returned the best performance of 61.22%, but it did very poorly in recall, at 20.61%, missing many actual deepfakes. This poor recall reflects the model's inability to generalise effectively to unseen deepfake data, limiting its applicability to real word detection tasks. ResNet50, while reaching perfection in recall of 100%, obtained an unacceptably low accuracy of 36.29%, demonstrating a very high level of false positives. This suggests that although the model identified all deepfakes, the model still misclassified many real images as fakes. This finding presents a trade off within detection models suggesting that high recall comes at the expense of precision, particularly when using image classification architectures like ResNet50​. The Xception model was more balanced, with a recall of 69.08%, but its overall accuracy remained low at 41.27%. Xception's depthwise separable convolutions provided effective feature extraction. However, the model still struggled with high false positives. This agrees with the observations in the literature that while advanced convolutional architectures like Xception can do very well in feature extraction, they usually require bigger and more diverse data to attain improvement in accuracy. Most models require high quality and diverse datasets to perform optimally. One of the biggest challenges within deepfake detection was the lack of complete datasets that captured wide varieties of deepfake detection techniques. Without diversity in the data, models are prone to overfitting and poor generalisation, as evidenced in this research. This limitation has been echoed by Guarnera et al. (2022) and Tolosana et al. (2021) in the literature review, who emphasise the need for continuous dataset expansion to keep pace with evolving deepfake techniques​. Another finding is the rapid advancement of deepfake generation techniques, which constantly outpace detection methods as evidenced in chapter 2. This "cat-and-mouse" situation, noted by Kirchengast (2020) and Mirsky & Lee (2021), requires regular updating and retraining of the detecting models so that the models can remain effective. Traditional detection models, such as those based on manual inspection or leveraging forensic techniques, are increasingly ineffective against sophisticated deepfake content. It is clear from the literature that the approach to detection is shifting towards deep learning models. This approach however, has been shown to have its limitations as presented in this research.

**7.3 Research Contributions**

This work offers several contributions to studies on deepfake detection. The research compares the performance of three different deep learning architectures, gives an overview of their strengths and weaknesses when working on the deep fake detection task, and points out the relevance of dataset diversity and preprocessing techniques to increase model performance as well as more powerful computational resources for handling the deep learning model demands.Moreover, this work underlined that deep fake detection models still have room for optimization, and architectures used so far needed adaptation or replacement with more suitable ones for better performance.

**7.4 Limitations of the Study**

Limitations that might have affected the results of this study include the following. First, due to limited computational power, extensive hyperparameter tuning and training the models on more considerable datasets was not possible. Second, the dataset used was comparatively small, it therefore may not have provided enough variance for the models to learn robust features, leading to poor generalisation.The study was further limited in scope to the performance assessment of three deep learning models, whereby no investigation as to whether hybrid models or other architectures could do better was pursued in this deep fake detection paper. Limited by this scope of study, the integration of such deepfake detection into wider cybersecurity frameworks was not further explored.

**7.5 Future Work**

Future research for deepfake detection should focus on several key areas for advancing the field. This would involve expanding datasets to wide varieties of deepfake techniques and scenarios for improving model generalisation. Model optimization could include further explorations on hybrid models and transformers as alternative architectures that have resulted in good performances on similar tasks. This will be enabled by the use of higher performing GPU servers and other advanced computational resources. Hence, larger data sets can be handled and models trained more effectively with extensive hyperparameter optimization. Additionally, investigations should be oriented to explain how deep fake detection contributes to the development of more effective cybersecurity strategies in an effort to enhance general digital security schemes. The final point relates to the need to develop ethical guidelines and a legal framework that can help control deepfake technology misuse and make the regulations clear about the complete usage across various applications.

# 

# 

# Appendix

**Appendix A data\_preprocessing.py**

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define data generators with data augmentation for training and validation

datagen = ImageDataGenerator(

rescale=1./255,

horizontal\_flip=True,

vertical\_flip=True,

rotation\_range=20,

validation\_split=0.2

)

# Setup train and validation generators

train\_generator = datagen.flow\_from\_directory(

'dataset\_train\_model', # Path to directory with fake and real subdirectories

target\_size=(256, 256),

batch\_size=32,

class\_mode='binary',

subset='training'

)

validation\_generator = datagen.flow\_from\_directory(

'dataset\_train\_model', # Path to the directory with fake and real subdirectories

target\_size=(256, 256),

batch\_size=32,

class\_mode='binary',

subset='validation'

)

**Appendix B train\_model.py**

from tensorflow.keras.applications import Xception

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D

from tensorflow.keras.optimizers import Adam

from data\_preprocessing import train\_generator, validation\_generator

# Load the Xception model pre-trained on ImageNet

base\_model = Xception(weights='imagenet', include\_top=False, input\_shape=(256, 256, 3))

# Add custom layers on top of the base model

x = base\_model.output

x = GlobalAveragePooling2D()(x)

x = Dense(1024, activation='relu')(x)

predictions = Dense(1, activation='sigmoid')(x)

# Combine the base model with the custom layers

model = Model(inputs=base\_model.input, outputs=predictions)

# Freeze the base model layers

for layer in base\_model.layers:

layer.trainable = False

# Compile the model

model.compile(optimizer=Adam(learning\_rate=0.001), loss='binary\_crossentropy', metrics=['accuracy'])

# Train the model

model.fit(

train\_generator,

epochs=10,

validation\_data=validation\_generator,

steps\_per\_epoch=train\_generator.samples // 32,

validation\_steps=validation\_generator.samples // 32

)

# Save the trained model

model.save('xception\_deepfake\_detection\_model.h5')

**Appendix C extract\_frames.py**

import cv2

import os

video\_directory = 'Face\_only\_data'

image\_directory = 'Face\_only\_images'

# Create directories if they don't exist

os.makedirs(image\_directory, exist\_ok=True)

# Iterate through all video files in video\_directory

for video\_file in os.listdir(video\_directory):

video\_file\_path = os.path.join(video\_directory, video\_file)

if os.path.isfile(video\_file\_path): # Check if it's a file (not a directory)

# Determine class based on filename ('Real' or 'Fake')

if 'realface' in video\_file.lower():

cls = 'Real'

else:

cls = 'Fake'

image\_path = os.path.join(image\_directory, cls.lower()) # Save frames under lowercase folder names

os.makedirs(image\_path, exist\_ok=True)

# Capture frames from video

cap = cv2.VideoCapture(video\_file\_path)

frame\_count = 0

while True:

ret, frame = cap.read()

if not ret:

break

# Optionally, process frame (resize, etc.) before saving

frame\_count += 1

if frame\_count % 25 == 0: # Example: Extract every 25th frame

frame\_filename = f'{os.path.splitext(video\_file)[0]}\_frame{frame\_count}.jpg' # Using video filename without extension

frame\_filepath = os.path.join(image\_path, frame\_filename)

cv2.imwrite(frame\_filepath, frame)

cap.release()

print("Frame extraction completed.")

**Appendix D Evaluate\_model.py**

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import precision\_score, recall\_score, f1\_score, roc\_curve, auc, precision\_recall\_curve, confusion\_matrix

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Define function to calculate precision, recall, and F1 score

def calculate\_metrics(y\_true, y\_pred):

precision = precision\_score(y\_true, y\_pred)

recall = recall\_score(y\_true, y\_pred)

f1 = f1\_score(y\_true, y\_pred)

return precision, recall, f1

# Define data generator for test set

datagen = ImageDataGenerator(rescale=1./255)

# Setup test generator

test\_generator = datagen.flow\_from\_directory(

'Face\_only\_images', # Directory containing 'real' and 'fake' subdirectories of extracted frames

target\_size=(256, 256),

batch\_size=32,

class\_mode='binary',

shuffle=False # Important to keep the order of predictions the same as the true labels

)

# Load the trained model

model = load\_model('xception\_deepfake\_detection\_model.h5')

# Predict probabilities on the test data

y\_probs = model.predict(test\_generator).flatten() # Flatten to a 1D array

# Extract true labels from the generator

y\_true = test\_generator.classes

# Calculate True Positives, True Negatives, False Positives, and False Negatives

threshold = 0.5

y\_pred = (y\_probs >= threshold).astype(int)

TP = np.sum((y\_true == 1) & (y\_pred == 1))

TN = np.sum((y\_true == 0) & (y\_pred == 0))

FP = np.sum((y\_true == 0) & (y\_pred == 1))

FN = np.sum((y\_true == 1) & (y\_pred == 0))

print(f'True Positives: {TP}')

print(f'True Negatives: {TN}')

print(f'False Positives: {FP}')

print(f'False Negatives: {FN}')

# Evaluate the model on the test data

loss, accuracy = model.evaluate(test\_generator)

print(f'Test Loss: {loss:.4f}')

print(f'Test Accuracy: {accuracy \* 100:.2f}%')

# Calculate precision, recall, and F1 score using the function

precision, recall, f1 = calculate\_metrics(y\_true, y\_pred)

print(f'Precision: {precision \* 100:.2f}%')

print(f'Recall: {recall \* 100:.2f}%')

print(f'F1 Score: {f1 \* 100:.2f}%')

# Plot Confusion Matrix

cm = confusion\_matrix(y\_true, y\_pred)

plt.figure(figsize=(6, 6))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False, xticklabels=['Fake', 'Real'], yticklabels=['Fake', 'Real'])

plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.savefig('confusion\_matrix.png') # Save as PNG

plt.close()

# Plot Distribution of Predictions

plt.figure(figsize=(6, 6))

plt.bar(['True Positives', 'True Negatives', 'False Positives', 'False Negatives'], [TP, TN, FP, FN], color=['green', 'blue', 'red', 'orange'])

plt.title('Distribution of Predictions')

plt.ylabel('Count')

plt.savefig('distribution\_of\_predictions.png') # Save as PNG

plt.close()

# Calculate ROC Curve

fpr, tpr, roc\_thresholds = roc\_curve(y\_true, y\_probs)

roc\_auc = auc(fpr, tpr)

# Plot ROC Curve

plt.figure()

plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.0])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic Curve')

plt.legend(loc="lower right")

plt.savefig('roc\_curve.png') # Save as PNG

plt.close()

# Calculate Precision-Recall Curve

precision\_values, recall\_values, pr\_thresholds = precision\_recall\_curve(y\_true, y\_probs)

pr\_auc = auc(recall\_values, precision\_values)

# Plot Precision-Recall Curve

plt.figure()

plt.plot(recall\_values, precision\_values, color='b', lw=2, label=f'Precision-Recall curve (area = {pr\_auc:.2f})')

plt.xlabel('Recall')

plt.ylabel('Precision')

plt.title('Precision-Recall Curve')

plt.legend(loc="lower left")

plt.savefig('precision\_recall\_curve.png') # Save as PNG

plt.close()

# Optionally, print TPR and FPR at different thresholds

for i, threshold in enumerate(roc\_thresholds):

print(f'Threshold: {threshold:.2f}, FPR: {fpr[i]:.2f}, TPR: {tpr[i]:.2f}')

**Appendix E resnet50\_extract\_frames.py**

import cv2

import os

video\_directory = 'Face\_only\_data'

image\_directory = 'Face\_only\_images'

# Create directories if they don't exist

os.makedirs(image\_directory, exist\_ok=True)

# Iterate through all video files in video\_directory

for video\_file in os.listdir(video\_directory):

video\_file\_path = os.path.join(video\_directory, video\_file)

if os.path.isfile(video\_file\_path): # Check if it's a file (not a directory)

# Determine class based on filename ('Real' or 'Fake')

if 'realface' in video\_file.lower():

cls = 'Real'

else:

cls = 'Fake'

image\_path = os.path.join(image\_directory, cls.lower()) # Save frames under lowercase folder names

os.makedirs(image\_path, exist\_ok=True)

# Capture frames from video

cap = cv2.VideoCapture(video\_file\_path)

frame\_count = 0

while True:

ret, frame = cap.read()

if not ret:

break

# Optionally, process frame (resize, etc.) before saving

frame\_count += 1

if frame\_count % 25 == 0: # Example: Extract every 25th frame

frame\_filename = f'{os.path.splitext(video\_file)[0]}\_frame{frame\_count}.jpg' # Using video filename without extension

frame\_filepath = os.path.join(image\_path, frame\_filename)

cv2.imwrite(frame\_filepath, frame)

cap.release()

print("Frame extraction completed.")

**Appendix F resnet50\_data\_preprocessing.py**

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define data generators with data augmentation for training and validation

datagen = ImageDataGenerator(

rescale=1./255,

horizontal\_flip=True,

vertical\_flip=True,

rotation\_range=20,

validation\_split=0.2

)

# Setup train and validation generators

train\_generator = datagen.flow\_from\_directory(

'dataset\_train\_model', # Path to the directory with 'fake' and 'real' subdirectories

target\_size=(224, 224),

batch\_size=32,

class\_mode='binary',

subset='training'

)

validation\_generator = datagen.flow\_from\_directory(

'dataset\_train\_model', # Path to the directory with 'fake' and 'real' subdirectories

target\_size=(224, 224),

batch\_size=32,

class\_mode='binary',

subset='validation'

)

**Appendix G Resnet50\_train\_model.py**

import os

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.models import Model

from tensorflow.keras.applications import ResNet50

from tensorflow.keras.regularizers import l2 # Import l2 regularizer

from resnet50\_data\_preprocessing import train\_generator, validation\_generator

from sklearn.utils import class\_weight

import numpy as np

# Load ResNet50 base model pre-trained on ImageNet

base\_model = ResNet50(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

# Unfreeze the last few layers of the ResNet50 model for fine-tuning

for layer in base\_model.layers[-30:]: # Unfreeze the last 30 layers

layer.trainable = True

# Add custom layers on top of the base model

x = base\_model.output

x = GlobalAveragePooling2D()(x)

x = Dense(1024, activation='relu', kernel\_regularizer=l2(0.001))(x) # Use l2 regularizer

x = Dropout(0.5)(x) # Add dropout to prevent overfitting

predictions = Dense(1, activation='sigmoid')(x)

# Combine the base model with the custom layers

model = Model(inputs=base\_model.input, outputs=predictions)

# Compile the model with a lower learning rate

model.compile(optimizer=Adam(learning\_rate=0.00001), loss='binary\_crossentropy', metrics=['accuracy'])

# Calculate class weights to handle class imbalance

class\_weights = class\_weight.compute\_class\_weight(

'balanced',

classes=np.unique(train\_generator.classes),

y=train\_generator.classes

)

class\_weights\_dict = dict(enumerate(class\_weights))

# Train the model

model.fit(

train\_generator,

epochs=50, # Increase the number of epochs to allow more training

validation\_data=validation\_generator,

steps\_per\_epoch=train\_generator.samples // 32,

validation\_steps=validation\_generator.samples // 32,

class\_weight=class\_weights\_dict # Apply class weights

)

# Save the trained model

model.save('resnet50\_deepfake\_detection\_model.h5')

**Appendix H resnet50\_evaluate\_model.py**

import seaborn as sns

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import confusion\_matrix, precision\_score, recall\_score, f1\_score, roc\_curve, auc, precision\_recall\_curve

import matplotlib.pyplot as plt

import numpy as np

# Define function to calculate precision, recall, and F1 score

def calculate\_metrics(y\_true, y\_pred):

precision = precision\_score(y\_true, y\_pred)

recall = recall\_score(y\_true, y\_pred)

f1 = f1\_score(y\_true, y\_pred)

return precision, recall, f1

# Define data generator for test set

datagen = ImageDataGenerator(rescale=1./255)

# Setup test generator

test\_generator = datagen.flow\_from\_directory(

'Face\_only\_images', # Directory containing 'real' and 'fake' subdirectories of extracted frames

target\_size=(224, 224),

batch\_size=32,

class\_mode='binary',

shuffle=False # Ensure data is not shuffled for consistent evaluation

)

# Load the trained model

model = load\_model('resnet50\_deepfake\_detection\_model.h5')

# Predict probabilities on the test data

y\_pred\_prob = model.predict(test\_generator)

# Convert predicted probabilities to binary predictions

y\_pred = y\_pred\_prob.round()

# Extract true labels from the generator

y\_true = test\_generator.classes

# Calculate the confusion matrix

cm = confusion\_matrix(y\_true, y\_pred)

# Plot the confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='g', cmap='Blues', cbar=False,

xticklabels=['Predicted Fake', 'Predicted Real'],

yticklabels=['Actual Fake', 'Actual Real'])

plt.title('Confusion Matrix')

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.savefig('confusion\_matrix.png') # Save the confusion matrix as a PNG file

plt.show()

# Evaluate the model on the test data

loss, accuracy = model.evaluate(test\_generator)

print(f'Test Loss: {loss:.4f}')

print(f'Test Accuracy: {accuracy \* 100:.2f}%')

# Calculate precision, recall, and F1 score using the function

precision, recall, f1 = calculate\_metrics(y\_true, y\_pred)

# Print precision, recall, and F1 score

print(f'Precision: {precision \* 100:.2f}%')

print(f'Recall: {recall \* 100:.2f}%')

print(f'F1 Score: {f1 \* 100:.2f}%')

**Appendix I cnn\_data\_preprocessing.py**

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define data generators with data augmentation for training and validation

datagen = ImageDataGenerator(

rescale=1./255,

horizontal\_flip=True,

vertical\_flip=True,

rotation\_range=20,

validation\_split=0.2

)

# Setup train generator

train\_generator = datagen.flow\_from\_directory(

'dataset\_train\_model', # Path to the directory with 'fake' and 'real' subdirectories

target\_size=(256, 256),

batch\_size=32,

class\_mode='binary',

subset='training'

)

# Setup validation generator

validation\_generator = datagen.flow\_from\_directory(

'dataset\_train\_model', # Path to the directory with 'fake' and 'real' subdirectories

target\_size=(256, 256),

batch\_size=32,

class\_mode='binary',

subset='validation'

)

**Appendix J cnn\_train\_model.py**

import os

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.models import Sequential

from tensorflow.keras.optimizers import Adam

from cnn\_data\_preprocessing import train\_generator, validation\_generator

# Define the CNN model

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(256, 256, 3)),

MaxPooling2D(pool\_size=(2, 2)),

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D(pool\_size=(2, 2)),

Conv2D(128, (3, 3), activation='relu'),

MaxPooling2D(pool\_size=(2, 2)),

Flatten(),

Dense(512, activation='relu'),

Dropout(0.5),

Dense(1, activation='sigmoid')

])

# Compile the model

model.compile(optimizer=Adam(learning\_rate=0.001), loss='binary\_crossentropy', metrics=['accuracy'])

# Train the model

model.fit(

train\_generator,

epochs=10,

validation\_data=validation\_generator,

steps\_per\_epoch=train\_generator.samples // 32,

validation\_steps=validation\_generator.samples // 32

)

# Save the trained model

model.save('cnn\_deepfake\_detection\_model.h5')

**Appendix K Cnn\_extract\_frames.py**

image\_directory = 'Face\_only\_images'

# Create directories if they don't exist

os.makedirs(image\_directory, exist\_ok=True)

# Iterate through all video files in video\_directory

for video\_file in os.listdir(video\_directory):

video\_file\_path = os.path.join(video\_directory, video\_file)

if os.path.isfile(video\_file\_path): # Check if it's a file (not a directory)

# Determine class based on filename ('Real' or 'Fake')

if 'realface' in video\_file.lower():

cls = 'Real'

else:

cls = 'Fake'

image\_path = os.path.join(image\_directory, cls.lower()) # Save frames under lowercase folder names

os.makedirs(image\_path, exist\_ok=True)

# Capture frames from video

cap = cv2.VideoCapture(video\_file\_path)

frame\_count = 0

while True:

ret, frame = cap.read()

if not ret:

break

# Optionally, process frame (resize, etc.) before saving

frame\_count += 1

if frame\_count % 25 == 0: # Example: Extract every 25th frame

frame\_filename = f'{os.path.splitext(video\_file)[0]}\_frame{frame\_count}.jpg' # Using video filename without extension

frame\_filepath = os.path.join(image\_path, frame\_filename)

cv2.imwrite(frame\_filepath, frame)

cap.release()

print("Frame extraction completed.")

**Appendix L cnn\_evaluate\_model.py**

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import precision\_score, recall\_score, f1\_score, roc\_curve, auc, precision\_recall\_curve, confusion\_matrix

# Define function to calculate precision, recall, and F1 score

def calculate\_metrics(y\_true, y\_pred):

precision = precision\_score(y\_true, y\_pred)

recall = recall\_score(y\_true, y\_pred)

f1 = f1\_score(y\_true, y\_pred)

return precision, recall, f1

# Define data generator for test set

datagen = ImageDataGenerator(rescale=1./255)

# Setup test generator

test\_generator = datagen.flow\_from\_directory(

'Face\_only\_images', # Directory containing 'real' and 'fake' subdirectories of extracted frames

target\_size=(256, 256),

batch\_size=32,

class\_mode='binary',

shuffle=False # Important to keep the order of predictions the same as the true labels

)

# Load the trained model

model = load\_model('cnn\_deepfake\_detection\_model.h5')

# Predict probabilities on the test data

y\_probs = model.predict(test\_generator).flatten() # Flatten to a 1D array

# Extract true labels from the generator

y\_true = test\_generator.classes

# Calculate True Positives, True Negatives, False Positives, and False Negatives

threshold = 0.5

y\_pred = (y\_probs >= threshold).astype(int)

TP = np.sum((y\_true == 1) & (y\_pred == 1))

TN = np.sum((y\_true == 0) & (y\_pred == 0))

FP = np.sum((y\_true == 0) & (y\_pred == 1))

FN = np.sum((y\_true == 1) & (y\_pred == 0))

print(f'True Positives: {TP}')

print(f'True Negatives: {TN}')

print(f'False Positives: {FP}')

print(f'False Negatives: {FN}')

# Evaluate the model on the test data

loss, accuracy = model.evaluate(test\_generator)

print(f'Test Loss: {loss:.4f}')

print(f'Test Accuracy: {accuracy \* 100:.2f}%')

# Calculate precision, recall, and F1 score using the function

precision, recall, f1 = calculate\_metrics(y\_true, y\_pred)

print(f'Precision: {precision \* 100:.2f}%')

print(f'Recall: {recall \* 100:.2f}%')

print(f'F1 Score: {f1 \* 100:.2f}%')

# Calculate ROC Curve

fpr, tpr, roc\_thresholds = roc\_curve(y\_true, y\_probs)

roc\_auc = auc(fpr, tpr)

# Plot ROC Curve and save as PNG

plt.figure()

plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.0])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic Curve')

plt.legend(loc="lower right")

plt.savefig('roc\_curve.png')

plt.close()

# Calculate Precision-Recall Curve

precision\_values, recall\_values, pr\_thresholds = precision\_recall\_curve(y\_true, y\_probs)

pr\_auc = auc(recall\_values, precision\_values)

# Plot Precision-Recall Curve and save as PNG

plt.figure()

plt.plot(recall\_values, precision\_values, color='b', lw=2, label=f'Precision-Recall curve (area = {pr\_auc:.2f})')

plt.xlabel('Recall')

plt.ylabel('Precision')

plt.title('Precision-Recall Curve')

plt.legend(loc="lower left")

plt.savefig('precision\_recall\_curve.png')

plt.close()

# Compute confusion matrix

cm = confusion\_matrix(y\_true, y\_pred)

# Plot confusion matrix and save as PNG

plt.figure()

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Fake', 'Real'], yticklabels=['Fake', 'Real'])

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.savefig('confusion\_matrix.png')

plt.close()

# Distribution of Predictions

plt.figure()

plt.hist([y\_true, y\_pred], bins=2, label=['True Labels', 'Predicted Labels'], alpha=0.7)

plt.xlabel('Class')

plt.ylabel('Count')

plt.title('Distribution of Predictions')

plt.legend()

plt.savefig('distribution\_of\_predictions.png')

plt.close()

print("Evaluation and plotting completed.")

# 

# 

# 

# 

# 

# 

# 

# References

# 

Aggarwal, A., Mittal, M., & Battineni, G. (2021). Generative adversarial network: An overview of theory and applications. *International Journal of Information Management Data Insights*, *1*(1), 100004. https://doi.org/10.1016/j.jjimei.2020.100004

Bank, D., Noam Koenigstein, & Raja Giryes. (2023). Autoencoders. *Springer EBooks*, 353–374. https://doi.org/10.1007/978-3-031-24628-9\_16

Barker, J. R., Bollman, M., Ringold, P. L., Sackinger, J., & Cline, S. P. (2002). Evaluation of Metric Precision for a Riparian Forest Survey. *Environmental Monitoring and Assessment*, *75*(1), 57–78. https://doi.org/10.1023/a:1014259902449

Bashir, D., Montañez, G. D., Sehra, S., Segura, P. S., & Lauw, J. (2020). An Information-Theoretic Perspective on Overfitting and Underfitting. *AI 2020: Advances in Artificial Intelligence*, 347–358. https://doi.org/10.1007/978-3-030-64984-5\_27

Bohacek, M., & Farid, H. (2024). *Lost in Translation: Lip-Sync Deepfake Detection from Audio-Video Mismatch*. Openaccess.thecvf.com. https://openaccess.thecvf.com/content/CVPR2024W/WMF/html/Bohacek\_Lost\_in\_Translation\_Lip-Sync\_Deepfake\_Detection\_from\_Audio-Video\_Mismatch\_CVPRW\_2024\_paper.html

Boyd, K., Eng, K. H., & Page, C. D. (2013). Area under the Precision-Recall Curve: Point Estimates and Confidence Intervals. *Advanced Information Systems Engineering*, 451–466. https://doi.org/10.1007/978-3-642-40994-3\_29

Caelen, O. (2017). A Bayesian interpretation of the confusion matrix. *Annals of Mathematics and Artificial Intelligence*, *81*(3-4), 429–450. https://doi.org/10.1007/s10472-017-9564-8

Carnagie, J. O., Prabowo, A. R., Budiana, E. P., & Singgih, I. K. (2022). Essential Oil Plants Image Classification Using Xception Model. *Procedia Computer Science*, *204*, 395–402. https://doi.org/10.1016/j.procs.2022.08.048

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1). https://doi.org/10.1186/s12864-019-6413-7

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018, January 13). *Generative Adversarial Networks: An Overview*. IEEE Xplore; IEEE. https://ieeexplore.ieee.org/abstract/document/8253599/authors#authors

Dhar, A., & Agrawal, E. (2024). Detecting AI-Generated Deep Fakes Using ResNext CNN and LSTM-Based RNN: A Robust Approach for Real-Time Video Manipulation Detection. *Lecture Notes in Networks and Systems*, 543–554. https://doi.org/10.1007/978-981-97-0641-9\_37

DiPietro, R., & Hager, G. D. (2020, January 1). *Chapter 21 - Deep learning: RNNs and LSTM* (S. K. Zhou, D. Rueckert, & G. Fichtinger, Eds.). ScienceDirect; Academic Press. https://www.sciencedirect.com/science/article/abs/pii/B9780128161760000260

Dolhansky, B., Bitton, J., Pflaum, B., Lu, J., Howes, R., Wang, M., & Ferrer, C. C. (2020). The DeepFake Detection Challenge Dataset. *ArXiv:2006.07397 [Cs]*. https://arxiv.org/abs/2006.07397

Fisher, C. W., Lauria, E. J. M., & Matheus, C. C. (2009). An Accuracy Metric. *Journal of Data and Information Quality*, *1*(3), 1–21. https://doi.org/10.1145/1659225.1659229

Forstmeier, W., Wagenmakers, E.-J., & Parker, T. H. (2016). Detecting and avoiding likely false-positive findings - a practical guide. *Biological Reviews*, *92*(4), 1941–1968. https://doi.org/10.1111/brv.12315

Freeman, J. M. (2007). Beware: The misuse of technology and the law of unintended consequences. *Neurotherapeutics*, *4*(3), 549–554. https://doi.org/10.1016/j.nurt.2007.04.003

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, *77*, 354–377. https://doi.org/10.1016/j.patcog.2017.10.013

Guarnera, L., Giudice, O., Guarnera, F., Ortis, A., Puglisi, G., Paratore, A., Bui, L. M. Q., Fontani, M., Coccomini, D. A., Caldelli, R., Falchi, F., Gennaro, C., Messina, N., Amato, G., Perelli, G., Concas, S., Cuccu, C., Orrù, G., Marcialis, G. L., & Battiato, S. (2022). The Face Deepfake Detection Challenge. *Journal of Imaging*, *8*(10), 263. https://doi.org/10.3390/jimaging8100263

Hawkins, D. M. (2004). The Problem of Overfitting. *Journal of Chemical Information and Computer Sciences*, *44*(1), 1–12. https://doi.org/10.1021/ci0342472

Huilgol, P. (2020, September 3). *Precision vs Recall | Precision and Recall Machine Learning*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/

Ioana Cheres, & Groza, A. (2023). The profile: unleashing your deepfake self. *Multimedia Tools and Applications*, *82*(20), 31839–31854. https://doi.org/10.1007/s11042-023-14568-x

Jabr, F. (2021). John A. Long - Publications List. *Publicationslist.org*, *14*(6).

Jason Brownlee. (2020, January 2). *How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification*. Machine Learning Mastery. https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/

Johnson, D. G., & Diakopoulos, N. (2021). What to do about deepfakes. *Communications of the ACM*, *64*(3), 33–35. https://doi.org/10.1145/3447255

Kaur, G., Sinha, R., Tiwari, P. K., Yadav, S. K., Pandey, P., Raj, R., Vashisth, A., & Rakhra, M. (2022). Face mask recognition system using CNN model. *Neuroscience Informatics*, *2*(3), 100035. https://doi.org/10.1016/j.neuri.2021.100035

Kietzmann, J., Lee, L. W., McCarthy, I. P., & Kietzmann, T. C. (2019). Deepfakes: Trick or treat? *Business Horizons*, *63*(2), 135–146. https://doi.org/10.1016/j.bushor.2019.11.006

Kirchengast, T. (2020). Deepfakes and image manipulation: criminalisation and control. *Information & Communications Technology Law*, *29*(3), 308–323. https://doi.org/10.1080/13600834.2020.1794615

Koonce, B. (2021). ResNet 50. *Apress EBooks*, 63–72. https://doi.org/10.1007/978-1-4842-6168-2\_6

Kumar, R., & Indrayan, A. (2011). Receiver operating characteristic (ROC) curve for medical researchers. *Indian Pediatrics*, *48*(4), 277–287. https://doi.org/10.1007/s13312-011-0055-4

Lavie, A., Kenji Sagae, & Jayaraman, S. (2004). The Significance of Recall in Automatic Metrics for MT Evaluation. *Lecture Notes in Computer Science*, *3265*, 134–143. https://doi.org/10.1007/978-3-540-30194-3\_16

Lei, X., Pan, H., & Huang, X. (2019). A Dilated CNN Model for Image Classification. *IEEE Access*, *7*, 124087–124095. https://doi.org/10.1109/access.2019.2927169

Lopez Pinaya, W. H., Vieira, S., Garcia-Dias, R., & Mechelli, A. (2020, January 1). *Chapter 11 - Autoencoders* (A. Mechelli & S. Vieira, Eds.). ScienceDirect; Academic Press. https://www.sciencedirect.com/science/article/abs/pii/B9780128157398000110

Manaswi, N. K. (2018). RNN and LSTM. *Deep Learning with Applications Using Python*, 115–126. https://doi.org/10.1007/978-1-4842-3516-4\_9

Masood, M., Nawaz, M., Malik, K. M., Javed, A., Irtaza, A., & Malik, H. (2022). Deepfakes Generation and Detection: State-of-the-art, Open Challenges, Countermeasures, and Way Forward. *Applied Intelligence*, *53*(4). https://doi.org/10.1007/s10489-022-03766-z

Mirsky, Y., & Lee, W. (2021). The Creation and Detection of Deepfakes. *ACM Computing Surveys*, *54*(1), 1–41. https://doi.org/10.1145/3425780

Mubarak, R., Alsboui, T., Alshaikh, O., Inuwa-Dutse, I., Khan, S., & Parkinson, S. (2023, December 18). *A Survey on the Detection and Impacts of Deepfakes in Visual, Audio, and Textual Formats | IEEE Journals & Magazine | IEEE Xplore*. Ieeexplore.ieee.org. https://ieeexplore.ieee.org/abstract/document/10365143

Müller, N. M., Pizzi, K., & Williams, J. (2022). Human Perception of Audio Deepfakes. *Proceedings of the 1st International Workshop on Deepfake Detection for Audio Multimedia*, 85–91. https://doi.org/10.1145/3552466.3556531

Noah. (2023, October 24). *How to Detect Deepfake Images: Unmasking the Digital Deception*. Medium. https://medium.com/@cadiuxe/how-to-detect-deepfake-images-unmasking-the-digital-deception-d167899f85bc

Pandey, C. K., Mishra, V. K., & Tiwari, N. K. (2021). Deepfakes: When to Use It. *2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART)*. https://doi.org/10.1109/smart52563.2021.9676297

Qais, A., Rastogi, A., Saxena, A., Rana, A., & Sinha, D. (2022, November 12). *Deepfake Audio Detection with Neural Networks Using Audio Features*. IEEE Xplore; IEEE. https://ieeexplore.ieee.org/abstract/document/9862519/authors#authors

rini, regina, & cohen, leah. (2022, October 22). *Redirecting...* Heinonline.org. https://heinonline.org/HOL/Page?collection=journals&handle=hein.journals/jetshy22&id=151&men\_tab=srchresults

Robinson, A., Keller, L. R., & del Campo, C. (2022). Building insights on true positives vs. false positives: Bayes’ rule. *Decision Sciences Journal of Innovative Education*. https://doi.org/10.1111/dsji.12265

Saxena, D., & Cao, J. (2021). Generative Adversarial Networks (GANs). *ACM Computing Surveys*, *54*(3), 1–42. https://doi.org/10.1145/3446374

Schwenke, C., & Schering, A. (2014). True Positives, True Negatives, False Positives, False Negatives. *Wiley StatsRef: Statistics Reference Online*. https://doi.org/10.1002/9781118445112.stat06783

Sharma, N. (2023, June 6). *Understanding and Applying F1 Score: A Deep Dive with Hands-On Coding*. Arize AI. https://arize.com/blog-course/f1-score/

Sharma, S., & Kumar, S. (2021). The Xception model: A potential feature extractor in breast cancer histology images classification. *ICT Express*, *8*(1). https://doi.org/10.1016/j.icte.2021.11.010

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, 132306. https://doi.org/10.1016/j.physd.2019.132306

Singh, K. (2023, October 12). *Decoding the Confusion Matrix: A Comprehensive Guide to Classification Model Evaluation*. Dzone.com; DZone. https://dzone.com/articles/decoding-the-confusion-matrix-a-comprehensive-guid

Tate, C., & Voss, C. (2024). Combining Evaluation Metrics via Loss Functions. *ACL Anthology*, 242–250. https://aclanthology.org/2006.amta-papers.27/

Tolosana, R., Romero-Tapiador, S., Fierrez, J., & Vera-Rodriguez, R. (2021). DeepFakes Evolution: Analysis of Facial Regions and Fake Detection Performance. *Pattern Recognition. ICPR International Workshops and Challenges*, 442–456. https://doi.org/10.1007/978-3-030-68821-9\_38

Vadillo, M. A., Konstantinidis, E., & Shanks, D. R. (2015). Underpowered samples, false negatives, and unconscious learning. *Psychonomic Bulletin & Review*, *23*(1), 87–102. https://doi.org/10.3758/s13423-015-0892-6

Vamsi, V. V. V. N. S., Shet, S. S., Reddy, S. S. M., Rose, S. S., Shetty, S. R., Sathvika, S., S, S. M., & Shankar, S. P. (2022). Deepfake Detection in Digital Media Forensics. *Global Transitions Proceedings*, *3*(1). https://doi.org/10.1016/j.gltp.2022.04.017

Vishal Yathish. (2022, August 4). *Loss Functions and Their Use In Neural Networks - Towards Data Science*. Medium; Towards Data Science. https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9#:~:text=A%20loss%20function%20is%20a%20function#:~:text=A%20loss%20function%20is%20a%20function

Wang, K., Gou, C., Duan, Y., Lin, Y., Zheng, X., & Wang, F.-Y. (2017). Generative adversarial networks: introduction and outlook. *IEEE/CAA Journal of Automatica Sinica*, *4*(4), 588–598. https://doi.org/10.1109/JAS.2017.7510583

waseem, saima, syed abu bakar, syed abdul rahman, ahmed, bilal ashfaq, & omar, zaid. (2023, May 12). *DeepFake: on Face and expression Swap: a Review*. IEEE Xplore; IEEE. https://ieeexplore.ieee.org/document/10285057

Wen, L., Li, X., & Gao, L. (2019). A transfer convolutional neural network for fault diagnosis based on ResNet-50. *Neural Computing and Applications*, *32*(10), 6111–6124. https://doi.org/10.1007/s00521-019-04097-w

Yu, P., Xia, Z., Fei, J., & Lu, Y. (2021). A Survey on Deepfake Video Detection. *IET Biometrics*, *10*(6). https://doi.org/10.1049/bme2.12031

Zi, B., Chang, M., Chen, J., Ma, X., & Jiang, Y.-G. (2020). WildDeepfake. *Proceedings of the 28th ACM International Conference on Multimedia*. https://doi.org/10.1145/3394171.3413769