

DEEP FAKE IMAGE DETECTION USING DEEP LEARNING

A PROJECT REPORT

Submitted by

| | |
|----------------------|-----------------------|
| PRIYA R | (927621BCS083) |
| RANGA SHREE S | (927621BCS092) |
| SURUTHIKA S | (927621BCS112) |
| SUSHMITHA S | (927621BCS114) |

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

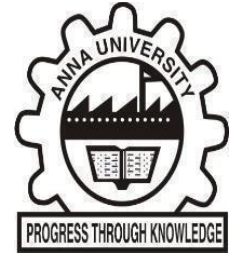
IN

COMPUTER SCIENCE AND ENGINEERING

M. KUMARASAMY COLLEGE OF ENGINEERING, KARUR

ANNA UNIVERSITY:: CHENNAI 600 025

APRIL 2025



DEEP FAKE IMAGE DETECTION USING DEEP LEARNING

A PROJECT REPORT

Submitted by

| | |
|----------------------|-----------------------|
| PRIYA R | (927621BCS083) |
| RANGA SHREE S | (927621BCS092) |
| SURUTHIKA S | (927621BCS112) |
| SUSHMITHA S | (927621BCS114) |

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

M. KUMARASAMY COLLEGE OF ENGINEERING, KARUR

ANNA UNIVERSITY:: CHENNAI 600 025

APRIL 2025

M. KUMARASAMY COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to Anna University, Chennai)

KARUR – 639 113

BONAFIDE CERTIFICATE

Certified that this project report “**DEEP FAKE IMAGE DETECTION USING DEEP LEARNING**” is the bonafide work of “**PRIYA R(927621BCS083), RANGA SHREE S (927621BCS092), SURUTHIKA S(927621BCS112), SUSHMITHA S(927621BCS114)**” who carried out the project work during the academic year 2023-2024 under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Signature

Dr.S.SUJANTHI , M.E.,Ph.D.,

SUPERVISOR,

Department of Computer Science and
Engineering,
M.Kumarasamy College of Engineering,
Thalavapalayam, Karur-639113.

Signature

Dr.D.PRADEEP, M.E., Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Computer Science and
Engineering,
M.Kumarasamy College of Engineering,
Thalavapalayam, Karur-639113.

This project Report has been submitted for the Project Work - End Semester viva voce
Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We affirm that the Project report titled **“DEEP FAKE IMAGE DETECTION USING DEEP LEARNING”** being submitted in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Signature

PRIYA R (927621BCS083) _____

RANGA SHREE S (927621BCS092) _____

SURUTHIKA S (927621BCS112) _____

SUSHMITHA S (927621BCS114) _____

ACKNOWLEDGEMENT

We gratefully remember our beloved **Founder Chairman, (Late) Thiru. M. Kumarasamy**, whose vision and legacy laid the foundation for our education and inspired us to successfully complete this project.

We extend our sincere thanks to **Dr. K. Ramakrishnan, Chairman, and Mr. K. R. Charun Kumar, Joint Secretary**, for providing excellent infrastructure and continuous support throughout our academic journey.

We are privileged to extend our heartfelt thanks to our respected Principal, **Dr. B. S. Murugan, B.Tech., M.Tech., Ph.D.**, for providing us with a conducive environment and constant encouragement to pursue this project work.

We sincerely thank **Dr. A. Kavitha, B.E., M.E., Ph.D.**, Professor and *Head, Department of Electronics and Communication Engineering*, for her continuous support, valuable guidance, and motivation throughout the course of this project.

Our special thanks and deep sense of appreciation go to our *Project Supervisor*, **Dr. S. Vimalnath, B.E., M.E., Ph.D.**, Associate Professor, *Department of Electronics and Communication Engineering*, for his exceptional guidance, continuous supervision, constructive suggestions, and unwavering support, all of which have been instrumental in the successful execution of this project.

We would also like to acknowledge **Mr. S. Mohanraj, B.E., M.E.**, *Assistant Professor, our Class Advisor*, and **Dr. S. Vimalnath**, the *Project Coordinator*, for their constant encouragement and coordination that contributed to the smooth progress and completion of our project work.

We gratefully thank all the faculty members of the *Department of Electronics and Communication Engineering* for their timely assistance, valuable insights, and constant support during various phases of the project.

Finally, we extend our profound gratitude to our parents and friends for their encouragement, moral support, and motivation, without which the successful completion of this project would not have been possible.

M.KUMARASAMY COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision of the Institution

To emerge as a leader among the top institutions in the field of technical education

Mission of the Institution

- ✚ Produce smart technocrats with empirical knowledge who can surmount the global challenges
- ✚ Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
- ✚ Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

Vision of the Department


To achieve education and research excellence in Computer Science and Engineering

Mission of the Department

- ✚ To excel in academic through effective teaching learning techniques
- ✚ To promote research in the area of computer science and engineering with the focus on innovation
- ✚ To transform students into technically competent professionals with societal and ethical responsibilities

Program Educational Objectives (PEOs)

- ✚ **PEO 1:** Graduates will have successful career in software industries and R&D divisions through continuous learning.
- ✚ **PEO 2:** Graduates will provide effective solutions for real world problems in the key domain of computer science and engineering and engage in lifelong learning.

 **PEO 3:** Graduates will excel in their profession by being ethically and socially responsible.

Program Outcomes (POs)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

PSO1- Professional Skills: Ability to apply the knowledge of computing techniques to design and develop computerized solutions for the problems.

PSO2- Successful career: Ability to utilize the computing skills and ethical values in creating a successful career.

ABSTRACT

The digital revolution has brought immense processing power, enabling advancements in autonomous driving, cancer cell recognition, and financial trading. Recently, Artificial Intelligence has gained significant attention, particularly in image generation and image-to-image translation, leading to the rise of deepfakes. Deepfakes use AI to manipulate videos, such as faking celebrity appearances or political speeches, raising major societal concerns. This focuses on deepfakes, their potential misuse against governments, individuals, and society, and explores their creation methods like Encoder-Decoder and Generative Adversarial Networks (GANs). A state-of-the-art deepfake creation pipeline is also discussed. Deepfake detection approaches are categorized into directed (detecting eye-blinks, head pose inconsistencies) and undirected (using classifiers to distinguish real and fake content). The research emphasizes identity swap deepfakes in the spatial domain using undirected approaches for better generalization. Experiments demonstrate that using Convolutional Neural Networks (CNNs) for feature extraction, along with increased image size, batch size, heavy data augmentation, and ensemble techniques, significantly improve detection and generalization performance.

ABSTRACT WITH POs AND PSOs MAPPING

| ABSTRACT | POs MAPPED | PSOs MAPPED |
|---|--|------------------------------|
| <p>The digital revolution has brought immense processing power, enabling advancements in autonomous driving, cancer cell recognition, and financial trading. Recently, Artificial Intelligence has gained significant attention, particularly in image generation and image-to-image translation, leading to the rise of deepfakes. Deepfakes use AI to manipulate videos, such as faking celebrity appearances or political speeches, raising major societal concerns. This focuses on deepfakes, their potential misuse against governments, individuals, and society, and explores their creation methods like Encoder-Decoder and Generative Adversarial Networks (GANs). A state-of-the-art deepfake creation pipeline is also discussed. Deepfake detection approaches are categorized into directed (detecting eye-blinks, head pose inconsistencies) and undirected (using classifiers to distinguish real and fake content). The research emphasizes identity swap deepfakes in the spatial domain using undirected approaches for better generalization. Experiments demonstrate that using Convolutional Neural Networks (CNNs) for feature extraction, along with increased image size, batch size, heavy data augmentation, and ensemble techniques, significantly improve detection and generalization performance.</p> | <p>PO1(3), PO2(3), PO3(3), PO4(3), PO5(3), PO6(2), PO7(3), PO8(3), PO9(3), PO10(3), PO11(2), PO12(2)</p> | <p>PSO1(3), PSO2(2),</p> |

NOTE: 1-LOW, 2-MEDIUM, 3-HIGH

SUPERVISOR

HEAD OF THE DEPARTMENT

TABLE OF CONTENTS

| CHAPTER NO. | TITLE | PAGE NO. |
|-------------|-----------------------------------|-------------|
| | ABSTRACT | viii |
| | LIST OF TABLES | xii |
| | LIST OF FIGURES | xiii |
| | LIST OF ABBREVIATIONS | xiv |
| 1 | INTRODUCTION | 1 |
| | 1.1 DESCRIPTION | 1 |
| | 1.2 DEEP LEARNING | 2 |
| | 1.3 APPLICATIONS OF DEEP LEARNING | 5 |
| | 1.4 ALGORITHMS IN DEEP LEARNING | 7 |
| | 1.5 CHALLENGES IN DEEP LEARNING | 8 |
| | 1.6 FUTURE SCOPE OF DEEP LEARNING | 9 |
| 2 | LITERATURE SURVEY | 12 |
| 3 | EXISTING SYSTEM | 25 |
| | 3.1 EXISTING SYSTEM BLOCK DIAGRAM | 26 |
| 4 | PROBLEMS IDENTIFIED | 27 |
| 5 | PROPOSED SYSTEM | 29 |
| 6 | SYSTEM REQUIREMENTS | 31 |
| 7 | SYSTEM IMPLEMENTATIONS | 32 |
| | 7.1 LIST OF MODULES | 32 |
| | 7.2 MODULES DESCRIPTION | 32 |
| | 7.2.1 Load Dataset | 32 |

| CHAPTER NO. | TITLE | PAGE NO. |
|--------------------|-----------------------------------|-----------------|
| | 7.2.2 Noise Reduction | 32 |
| | 7.2.3 Data Augmentation | 33 |
| | 7.2.4 Hyper Parameter Tuning | 34 |
| | 7.2.5 Classification | 34 |
| 8 | SYSTEM TESTING | 35 |
| | 8.1 SOFTWARE TESTING STRATEGIES | 35 |
| | 8.1.1 Unit Testing | 35 |
| | 8.1.2 Functional Testing | 36 |
| | 8.1.3 Acceptance Testing | 36 |
| | 8.1.4 Validation Testing | 36 |
| 9 | RESULTS AND DISCUSSION | 37 |
| 10 | CONCLUSION AND FUTURE WORK | 41 |
| | 10.1 CONCLUSION | 41 |
| | 10.2 FUTURE WORK | 41 |
| | APPENDIX 1: SOURCE CODE | 42 |
| | APPENDIX 2: SCREENSHOTS | 62 |
| | REFERENCES | 66 |
| | LIST OF PUBLICATIONS | 67 |

LIST OF TABLES

| TABLE NO. | TITLE | PAGE NO. |
|------------------|-------------------------------|-----------------|
| 2.1 | Findings in Literature Survey | 22 |

LIST OF FIGURES

| FIGURE NO. | TITLE | PAGE NO |
|------------|---|---------|
| 1.1 | Hierarchical Approach to Deepfake Detection | 1 |
| 3.1 | Block Diagram of Existing System | 26 |
| 5.1 | Architecture of the Proposed Work | 29 |
| 7.1 | Example of GAN Architecture | 33 |
| 9.1 | Accuracy | 37 |
| 9.2 | Proposed System - Model Loss | 38 |
| 9.3 | Existing System - Model Loss | 38 |
| 9.4 | Comparison of Different Performance Metrics | 39 |
| 9.5 | Classification Scatter Plot | 39 |
| 9.6 | Mean Squared Error | 40 |
| 9.7 | Classification Threshold | 40 |
| B.1 | Home Page | 62 |
| B.2 | Load Dataset | 62 |
| B.3 | Noise Reduction | 63 |
| B.4 | Data Augmentation | 63 |
| B.5 | Hyper Parameter Tuning | 64 |
| B.6 | Classification | 64 |

LIST OF ABBREVIATIONS

| ABBREVIATION | MEANING |
|--------------|---|
| AI | - Artificial Intelligence |
| AMTEN | - Adaptive Manipulation Traces Extraction Network |
| AUROC | - Area Under the Receiver Operating Characteristic Curve |
| BERT | - Bidirectional Encoder Representations from Transformers |
| CNN | - Convolutional Neural Network |
| DNN | - Deep Neural Network |
| DenseNet | - Densely Connected Convolutional Network |
| FIM | - Face Image Manipulation |
| GAN | - Generative Adversarial Networks |
| GPT-3 | - Generative Pre-trained Transformer 3 |
| GWO | - Grey Wolf Optimizer |
| HFM | - Handcrafted Facial Manipulation |
| LSTM | - Long Short-Term Memory |
| LIME | - Local Interpretable Model-Agnostic Explanations |
| MCnet | - Multiscale Network |
| ML | - Machine Learning |
| MLP | - Multilayer Perceptron |
| MRI | - Magnetic Resonance Imaging |
| MSR | - Mean Squared Error |
| NLP | - Natural Language Processing |
| ResNet-RS | - Residual Networks – Revised Scaling |
| RF | - Random Forest |

| | |
|------|----------------------------|
| RNN | - Recurrent Neural Network |
| SFFN | - Shallow-FakeFaceNet |
| SVM | - Support Vector Machine |
| VGG | - Visual Geometry Group |
| XAI | - Explainable AI |

CHAPTER 1

INTRODUCTION

1.1 DESCRIPTION

The creation and detection of deepfakes is a never-ending arms race. As soon as first deepfake attempts began to appear in the late nineties of the twentieth century, it was clear that some type of deepfake detection will be needed. First attempts of deepfakes were either easily detectable by the human observer because of impossibilities of technology to create a reliable deepfake, or took a long time to create, mainly because of insufficient computational power. But as new technologies emerge and computation power increases over the years, it was possible to create more believable deep- fakes. Especially thanks to breakthroughs in neural networks in the last decade, not only are we able to create more and more advanced deepfakes, but creation tools for deepfakes have become more and more common. Today, anyone with access to the Internet can create image, audio, or video deepfake without any problem.

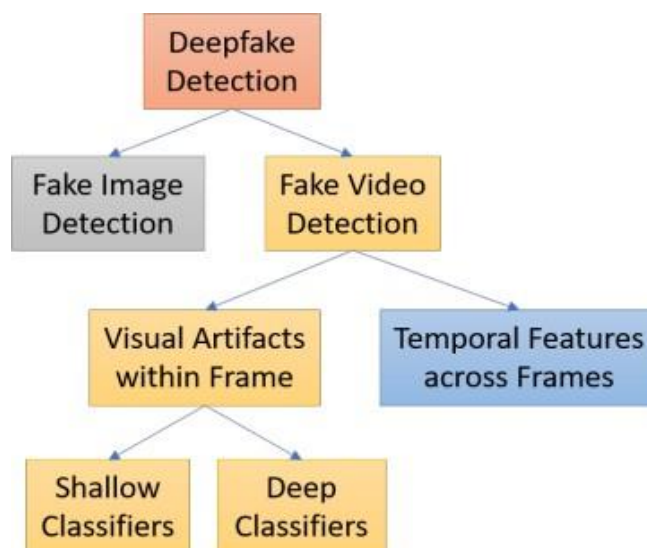


Fig 1.1 Hierarchical Approach to Deepfake Detection

Because of that, deepfake detection is now needed more than before. Especially in the field of video deepfakes, where most people assume that video is harder to fake than image or audio, mainly because of the fact that insufficient synchronization of fake video frames and audio will make detection of deepfake easier. This was true for older deepfake models, where the focus was on creating a reliable deepfake voice and swapping the main parts of the face and mouth. However, current state-of-the-art models can create deepfakes with perfectly synchronized audio and video frames. Some of them can even reproduce most of biological signals, such as eye blinking, lip synchronization, muscle mimic, etc. With this level of detail, it is impossible for a human observer to distinguish whether a video is a deepfake or not. The internet is filled with fake face images and videos synthesized by deep generative models. These realistic DeepFakes pose a challenge to determine the authenticity of multimedia content.

1.2 DEEP LEARNING

Deep Learning is a subfield of machine learning that uses artificial neural networks to model and solve complex problems. It is inspired by the structure and function of the human brain, where information is processed through a vast network of interconnected neurons. Deep learning algorithms are designed to automatically learn to recognize patterns in data by training on large datasets. The key advantage of deep learning over traditional machine learning techniques is its ability to automatically learn features from raw data, without the need for manual feature engineering. Deep learning has been successfully applied to a wide range of applications, including image and speech recognition, natural language processing, robotics, and autonomous driving. It has also led to significant advances in fields such as computer vision, speech processing, and natural language understanding.

Some of the most commonly used deep learning frameworks include TensorFlow, PyTorch, Keras, and Caffe. These frameworks provide high-level APIs for building and training deep neural networks, and they have made it easier for developers to implement and experiment with deep learning algorithms.

Deep learning algorithms are based on artificial neural networks, which are composed of layers of interconnected nodes or neurons. Each neuron performs a simple computation on its inputs, and the output is passed to the next layer. By stacking multiple layers of neurons, deep neural networks can learn to recognize complex patterns in data. The most common types of deep neural networks are feedforward networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs). Feedforward networks are the simplest type of neural network, where the inputs are processed through one or more layers of hidden neurons to produce an output. CNNs are designed to process image data, by using filters that scan across the input image to extract features. RNNs are used for processing sequential data, such as time series or natural language. One of the challenges of deep learning is the need for large amounts of labeled data to train the algorithms. This has led to the development of techniques such as transfer learning and data augmentation, which allow deep learning models to be trained on smaller datasets. Transfer learning involves pre-training a deep neural network on a large dataset, such as ImageNet, and then fine-tuning the network on a smaller dataset for a specific task. Data augmentation involves generating new training examples by applying transformations such as rotation, scaling, and cropping to the existing data.

Deep learning has the potential to revolutionize many industries, from healthcare and finance to transportation and manufacturing. However, there are also concerns about the ethical implications of using deep learning algorithms, particularly around issues such as bias, privacy, and security. As the field of deep learning continues to evolve, it will be important to address these issues and ensure that the technology is

developed and used in a responsible and ethical manner. Another important aspect of deep learning is the optimization of neural networks. Deep neural networks can have a large number of parameters, which makes training them computationally expensive and time-consuming. Gradient descent is a commonly used optimization algorithm that adjusts the weights of the network to minimize a loss function. However, training deep neural networks using gradient descent can be challenging due to issues such as vanishing gradients, where the gradients become very small and the network stops learning, and exploding gradients, where the gradients become very large and cause the network to diverge.

To address these issues, researchers have developed several optimization techniques, such as momentum, adaptive learning rates, and batch normalization. Momentum helps the optimization algorithm to move more smoothly towards the minimum by accumulating gradients from previous iterations. Adaptive learning rate methods adjust the learning rate during training based on the gradients, which can help to prevent the network from getting stuck in local minima. Batch normalization is a technique that normalizes the input to each layer of the network, which helps to stabilize the gradients and speed up training. Another challenge of deep learning is interpreting the results. Deep neural networks are often described as "black boxes," because it can be difficult to understand how the network arrived at a particular prediction. Researchers are exploring techniques such as feature visualization and attention mechanisms to better understand how deep neural networks make decisions. Finally, there is ongoing research into developing new architectures for deep neural networks, such as capsule networks, transformers, and graph neural networks. These architectures are designed to address specific challenges, such as modelling spatial relationships or processing graph-structured data. In summary, deep learning is a rapidly evolving field with many challenges and opportunities. It has already had a significant impact on many areas of industry and research, and is likely to continue to do so in the future.

1.3 APPLICATIONS OF DEEP LEARNING

Deep learning has revolutionized many industries and has the potential to transform many more. It has already been applied to a wide range of applications, from computer vision and natural language processing to healthcare and autonomous vehicles. Deep learning has also been used for creative applications, such as generating art and music. The development of new deep learning architectures, such as capsule networks and transformers, is likely to lead to new applications and advancements in existing applications. However, there are also concerns about the ethical implications of deep learning, particularly around issues such as bias, privacy, and security. It will be important to address these issues and ensure that deep learning is developed and used in a responsible and ethical manner. In addition to its potential benefits and ethical concerns, deep learning also poses some technical challenges. As mentioned earlier, deep neural networks can have a large number of parameters, making them computationally expensive to train. This can require powerful hardware and can be a barrier to entry for some researchers and organizations. There are also challenges related to interpretability, as deep neural networks can be difficult to understand and can make decisions that are difficult to explain. Another challenge is the need for large amounts of labelled data for training deep learning models. In some domains, such as healthcare, obtaining labelled data can be challenging due to privacy concerns or the need for expert knowledge. Additionally, there may be issues with bias in the data, which can lead to biased or unfair predictions. Despite these challenges, deep learning is a rapidly growing field with significant potential for innovation and impact. Ongoing research and development are likely to lead to new breakthroughs and applications in the coming years. However, it will also be important to address the challenges and ethical concerns associated with deep learning to ensure that it is used in a responsible and beneficial manner.

One area where deep learning has shown significant progress is in image and speech recognition. Deep learning models can be trained to recognize objects, faces, and even emotions in images and videos with high accuracy. Speech recognition models can transcribe spoken language into text, which can be used for tasks such as dictation, transcription, and language translation. Another area where deep learning has shown promise is in natural language processing (NLP). Deep learning models have been used to perform tasks such as sentiment analysis, language translation, and chatbot development. In recent years, the development of pre-trained language models, such as BERT and GPT-3, has significantly improved the performance of NLP tasks. Deep learning has also been used in the field of healthcare. It has been used to analyze medical images, such as X-rays and MRIs, to assist with diagnosis and treatment. Deep learning models have also been developed to predict disease progression and identify potential drug targets. In addition, wearable devices and sensors can be used to collect data on patients, which can be analyzed using deep learning algorithms to detect early warning signs of disease. Another area where deep learning is being applied is in autonomous vehicles. Deep learning models are used to analyze sensor data from cameras, lidar, and radar to detect objects, pedestrians, and other vehicles on the road. This information is used to make decisions about how the vehicle should navigate the road and avoid collisions.

Overall, deep learning has the potential to transform many industries and has already demonstrated significant progress in image and speech recognition, natural language processing, healthcare, and autonomous vehicles. Continued research and development in the field are likely to lead to new breakthroughs and applications in the coming years.

1.4 ALGORITHMS IN DEEP LEARNING

Deep learning algorithms are a class of machine learning algorithms that are based on artificial neural networks with multiple layers. These layers allow the model to learn increasingly complex features from the input data, which can be used for tasks such as classification, regression, and prediction.

There are several different types of deep learning algorithms, including:

Convolutional Neural Networks (CNNs): CNNs are commonly used in image and video recognition tasks. They are designed to detect spatial patterns in the input data by using convolutional layers that apply filters to the input image or video.

Recurrent Neural Networks (RNNs): RNNs are commonly used in natural language processing tasks. They are designed to handle sequential data, such as a sequence of words in a sentence, by using recurrent layers that maintain a memory of previous inputs.

Long Short-Term Memory (LSTM) Networks: LSTMs are a type of RNN that are designed to handle long-term dependencies in the input data. They are commonly used in tasks such as speech recognition and language translation.

Generative Adversarial Networks (GANs): GANs are a type of deep learning algorithm that are used for generative tasks, such as image and music generation. They consist of two neural networks: a generator network that creates new samples, and a discriminator network that distinguishes between real and fake samples.

Autoencoders: Autoencoders are a type of neural network that are used for unsupervised learning tasks, such as data compression and denoising. They consist of an encoder network that compresses the input data into a lower-dimensional representation, and a decoder network that reconstructs the original data from the compressed representation.

These are just a few examples of the many deep learning algorithms that have been developed in recent years. Each algorithm has its own strengths and weaknesses and is suited to different types of tasks and data.

1.5 CHALLENGES IN DEEP LEARNING

Deep learning is a powerful approach to machine learning, but it also poses several challenges. Some of the challenges of deep learning include:

Computational requirements: Deep learning models can have millions of parameters, making them computationally expensive to train. This can require powerful hardware, such as GPUs or specialized processors, and can be a barrier to entry for some researchers and organizations.

Data requirements: Deep learning models typically require large amounts of labeled data to train effectively. In some domains, such as healthcare, obtaining labeled data can be challenging due to privacy concerns or the need for expert knowledge. Additionally, there may be issues with bias in the data, which can lead to biased or unfair predictions.

Interpretability: Deep neural networks can be difficult to understand and can make decisions that are difficult to explain. This is a particular concern in applications such as healthcare, where it is important to understand the reasoning behind a diagnosis or treatment recommendation.

Generalization: Deep learning models can be prone to overfitting, which means they perform well on the training data but poorly on new data. This can be addressed by techniques such as regularization and early stopping, but it remains an ongoing challenge.

Ethical concerns: Deep learning raises ethical concerns around issues such as bias, privacy, and security. For example, deep learning models may perpetuate and

amplify existing biases in the data, or they may inadvertently reveal sensitive information about individuals.

Addressing these challenges will be crucial for the continued development and responsible use of deep learning. Ongoing research and development are likely to lead to new solutions and techniques for addressing these challenges in the coming years.

1.6 FUTURE SCOPE OF DEEP LEARNING

The future scope of deep learning is vast and promising. Ongoing research and development are likely to lead to new breakthroughs and applications that will expand the field in exciting ways. One area of potential growth is unsupervised learning, which does not rely on labelled data. This approach has the potential to unlock new insights and patterns in large, complex datasets, and could lead to new discoveries in fields such as healthcare and scientific research. Another area of growth is in the development of more interpretable models, which will help to address concerns around explainability and ethics. As deep learning becomes more widely adopted across industries, there will also be increased demand for specialized hardware and software to support the computational requirements of these models. Overall, the future of deep learning is bright, and it is likely to continue to be a powerful tool for solving complex problems and advancing our understanding of the world.

In addition to the areas mentioned earlier, there are several other exciting opportunities for the future of deep learning:

- **Reinforcement learning:** Reinforcement learning is a type of machine learning that involves an agent learning to make decisions based on feedback from its environment. Deep reinforcement learning combines reinforcement learning with

deep neural networks, allowing for more complex decision-making in applications such as robotics, gaming, and self-driving cars.

- **Multi-modal learning:** Multi-modal learning involves combining data from multiple sources, such as text, images, and audio. This approach has the potential to unlock new insights and enable more sophisticated applications in areas such as natural language processing and computer vision.
- **Edge computing:** Edge computing involves processing data locally, at the edge of a network, rather than in a central location. This can be particularly useful in applications such as autonomous vehicles or remote sensors, where low latency and real-time decision-making are critical.
- **Transfer learning:** Transfer learning involves using pre-trained models as a starting point for new tasks, allowing for more efficient and effective training. This approach can reduce the amount of labelled data required for training and can also improve performance on tasks with limited data.

Overall, the future of deep learning is likely to be characterized by ongoing innovation and the development of new techniques and applications. As the field continues to mature, it will become increasingly important to consider issues around ethics, transparency, and accountability in the development and use of these powerful technologies. In addition to the areas mentioned earlier, there are several other exciting opportunities for the future of deep learning:

- **Federated learning:** Federated learning is a decentralized approach to training deep learning models, in which data is kept locally on individual devices rather than being centralized. This approach can address issues of privacy and data ownership, while also enabling the development of models that are better tailored to local data.
- **Explainable AI:** As deep learning models become more complex, there is a growing need for models that can provide explanations for their predictions and decisions. Explainable AI (XAI) is an area of research that seeks to develop models

that can provide clear and interpretable explanations for their behaviour, making it easier for humans to understand and trust their predictions.

- Automated machine learning: Automated machine learning (AutoML) is a growing area of research that seeks to automate the process of building and optimizing machine learning models. This can help to democratize access to machine learning tools and enable a wider range of organizations and individuals to leverage the power of these technologies.
- Quantum computing: Quantum computing is an emerging area of research that has the potential to revolutionize the field of deep learning. Quantum computing offers the possibility of running computations much faster than classical computers, enabling more complex and powerful models to be developed.

CHAPTER 2

LITERATURE SURVEY

2.1 TITLE: DETECTING HANDCRAFTED FACIAL IMAGE MANIPULATIONS AND GAN-GENERATED FACIAL IMAGES USING SHALLOW-FAKEFACENET (2021)

AUTHOR: SANGYUP LEE, SHAHROZ TARIQ, YOUJIN SHIN

The rapid progress of sophisticated image editing tools has made it easier to manipulate original face images and create fake media content by putting one face onto another. In addition to image editing tools, creating natural-looking fake human faces can be easily achieved by Generative Adversarial Networks (GANs). However, malicious use of these new media generation technologies can lead to severe problems, such as the development of fake pornography, defamation, or fraud. This paper introduces a novel Handcrafted Facial Manipulation (HFM) image dataset and soft computing neural network models (Shallow-FakeFaceNets) with an efficient facial manipulation detection pipeline. The neural network classifier model, Shallow-FakeFaceNet (SFFN), shows the ability to focus on the manipulated facial landmarks to detect fake images. The detection pipeline relies only on detecting fake facial images based on RGB information, without leveraging any metadata, which can be easily manipulated. Results show that the method achieves the best performance of 72.52% in Area Under the Receiver Operating Characteristic (AUROC), gaining 3.99% F1-score and 2.91% AUROC on detecting handcrafted fake facial images, and 93.99% on detecting small GAN-generated fake images, gaining 1.98% F1-score and 10.44% AUROC compared to the best performing state-of-the-art classifier. This work presents various experimental results that can help guide better applied soft computing research in the future to effectively combat and detect human and GAN-generated fake face images.

2.2 TITLE: FAKE FACE DETECTION VIA ADAPTIVE MANIPULATION TRACES EXTRACTION NETWORK (2021)

AUTHOR: ZHIQING GUO, GAOBO YANG, JIYOU CHEN

With the proliferation of face image manipulation (FIM) techniques such as Face2Face and Deepfake, more fake face images are spreading over the internet, which brings serious challenges to public confidence. Face image forgery detection has made considerable progress in exposing specific FIM, but there is still a scarcity of a robust fake face detector to expose face image forgeries under complex scenarios, such as with further compression, blurring, and scaling. Due to the relatively fixed structure, convolutional neural networks (CNNs) tend to learn image content representations. However, CNNs should learn subtle manipulation traces for image forensics tasks. To address this, an adaptive manipulation traces extraction network (AMTEN) is proposed, serving as pre-processing to suppress image content and highlight manipulation traces. AMTEN exploits an adaptive convolution layer to predict manipulation traces in the image, which are reused in subsequent layers to maximize manipulation artifacts by updating weights during the back-propagation pass. A fake face detector, namely AMTENnet, is constructed by integrating AMTEN with CNN. Experimental results prove that the proposed AMTEN achieves desirable pre-processing. When detecting fake face images generated by various FIM techniques, AMTENnet achieves an average accuracy of up to 98.52%, outperforming state-of-the-art works. When detecting face images with unknown post-processing operations, the detector also achieves an average accuracy of 95.17%.

2.3 TITLE: DETECTING FAKE IMAGES BY IDENTIFYING POTENTIAL TEXTURE DIFFERENCE (2021)

AUTHOR: JIACHEN YANG, SHUAI XIAO , AIYUN LI, GUIPENG LAN

Fake detection has become an urgent task. Generative adversarial networks (GANs) extended to deep learning have shown extraordinary ability in the fields of image, audio, and speech. While advanced technology brings benefits, it also poses a threat when used in cybercrime. Deepfake, a common term for face manipulation methods based on GANs, enables the replacement of different faces. Due to the development of GANs, faces generated by Deepfake can already appear visually real. Deepfake can purposely replace any face with a different person, allowing fabricated events to spread widely through the convenience of the Internet, causing serious impacts such as personal attacks and cybercrime. Based on cutting-edge research, this paper proposes an intelligent forensic method for Deepfake detection. Subtle texture differences between real and fake images are discovered through image saliency, revealing variations in facial texture. To amplify these differences, a guided filter with a saliency map as a guide enhances texture artifacts caused by post-processing and highlights potential forgery features. The ResNet18 classification network efficiently learns these exposed differences and successfully distinguishes between real and fake face images. Performance evaluations confirm that the proposed method achieves state-of-the-art detection accuracy.

2.4 TITLE: MCNET: MULTISCALE VISIBLE IMAGE AND INFRARED IMAGE FUSION NETWORK (2023)

AUTHOR: LE SUN, YUHANG LI, MIN ZHENG, ZHAOYI ZHONG

In both civil and military fields, remote sensing image fusion is a popular method for improving images. Multimodal image fusion is typically employed in remote sensing image fusion, acquiring synthetic images containing rich information by combining image data from different wavebands. Current fusion networks concentrate on fusing features at a single image scale, resulting in images that lack either spatial or texture features. To address these issues and achieve high-quality fusion, MCnet (multiscale network) is proposed. MCnet is a new method for multimodal image fusion of visible remote sensing images and infrared remote sensing images within a multiscale framework. Specifically, MCnet first processes the fusion of image features at different scales in a coarse-to-fine manner. Additionally, MCnet adaptively determines the amount of information in each image at different scales, supplementing fine fusion features in the coarse fusion stage. In the fine fusion step, the outcomes of the previous stage are enhanced with the missing characteristics. Finally, an objective function is designed with three components: structure loss, region loss, and image quality loss. Structure loss and region loss maintain convergence on overall image similarity and regional feature similarity, while the image quality constraint partially mitigates the effect of low-quality results on model convergence. MCnet emphasizes texture features and edge contours of the results, enhancing fusion quality and improving discriminative properties. Extensive experiments are conducted based on visible and infrared datasets. The results demonstrate that the proposed model achieves state-of-the-art performance. Generalizability studies on the method further confirm its success and applicability in various situations.

2.5 TITLE: CONVOLUTIONAL NEURAL NETWORK FOR FAKE IMAGE DETECTION (2023)

AUTHOR: RETAJ MATROUD JASIM, TAYSEER SALMAN ATIA

The fast development in deep learning techniques, besides the wide spread of social networks, facilitated fabricating and distributing images and videos without prior knowledge. This paper developed an evolutionary learning algorithm to automatically design a convolutional neural network (CNN) architecture for deepfake detection. Genetic algorithm based on residual network (ResNet) and densely connected convolutional network (DenseNet) as building block units for feature extraction versus multilayer perceptron (MLP), random forest (RF) and support vector machine (SVM) as classifiers generates different CNN structures. A local search mutation operation proposed to optimize three layers: (batch normalization, activation function, and regularizes). This method has the advantage of working on different datasets without preprocessing. Findings using two datasets evidence the efficiency of the suggested approach where the generated models outperform the state-of-art by increasing 1% in the accuracy; this confirms that intuitive design is the new direction for better generalization.

2.6 TITLE: USING A CONCATENATION OF LOW-COMPLEXITY DEEP LEARNING MODELS, WE CAN SPOT FAKE IMAGES (2023)

AUTHOR: PATEL APARNA, DR. V. PRADEEP

Due to the broad availability of cameras, photo taking has been becoming increasingly common in recent years. Images play a crucial role in our everyday lives because to the abundance of information they hold, and it is frequently necessary to modify images to get new insights. There are many options for enhancing photos, but they are also regularly exploited to create fake photos that distribute false information. This is really worrisome since it increases the incidence and severity of picture forgeries. Over time, several conventional methods have been developed to identify picture frauds. The topic of visual forgery detection has been inspired by convolutional neural networks (CNNs), which have gained a lot of attention in recent years. While CNN-based picture forgery detection methods do exist, they typically only identify one sort of fraud (such as image merging or copy-move) at a time. Therefore, a method is needed that can effectively and efficiently detect the existence of unnoticed forgeries in a picture. In this research, we provide a powerful deep learning-based approach for spotting fakes in the setting of double compression of pictures. We train our representation on the disparity between the uncompressed and compressed variants of a picture. The suggested model requires fewer computational resources and has been shown to outperform the current best methods. Positively, experimental findings show a validation validity rate of 92.23 percent overall.

2.7 TITLE: DEEP FAKE IMAGE AND VIDEO DETECTION USING MACHINE LEARNING (2024)

AUTHOR: GURURAJ. A, AJAAL.N. M, ESWARAN. J

Deep fake technology has rapidly advanced in recent years, presenting a significant challenge in distinguishing between authentic and manipulated media content. This outlines the current state of research and development in deep fake image and video recognition, focusing on methodologies and advancements in detection techniques. This begins by elucidating the motivation behind deep fake recognition, highlighting its implications in various domains such as politics, journalism, and entertainment. It then delves into the technical aspects, discussing the underlying principles of deep fake generation and the emergence of sophisticated algorithms capable of producing highly convincing fake media. Furthermore, it provides insights into the evolving landscape of deep fake detection mechanisms. It discusses traditional approaches based on artifacts analysis and statistical methods, as well as the recent surge in machine learning and AI-based detection techniques. Notably, it emphasizes the importance of dataset curation, model training, and validation strategies in achieving robust detection performance. Moreover, the abstract touches upon the challenges and limitations faced by current deep fake recognition systems, including the arms race between generators and detectors, scalability issues, and ethical considerations. It concludes by underscoring the significance of interdisciplinary collaboration and ongoing research efforts in addressing these challenges and fostering trust in digital media integrity. Overall, this abstract offers a concise overview of the landscape of deep fake image and video recognition, serving as a primer for researchers, practitioners, and policymakers engaged in combating the proliferation of synthetic media manipulation.

2.8 TITLE: DETECTING DEEPFAKE IMAGES USING DEEP LEARNING TECHNIQUES AND EXPLAINABLE AI METHODS (2022)

AUTHOR: MOHAMMAD MONIRUJJAMAN, KAZI NABIUL ALAM

The introduction of deepfakes has marked a breakthrough in creating fake media and information. These manipulated pictures and videos will undoubtedly have an enormous societal impact. Deepfake uses the latest technology like AI to construct automated methods for creating fake content that is becoming increasingly difficult to detect with the human eye. Therefore, automated solutions employed by DL can be an efficient approach for detecting deepfake. Though the “black-box” nature of the DL system allows for robust predictions, they cannot be completely trustworthy. Explainability is the first step toward achieving transparency, but the existing incapacity of DL to explain its own decisions to human users limits the efficacy of these systems. Though Explainable Artificial Intelligence (XAI) can solve this problem by interpreting the predictions of these systems. This work proposes to provide a comprehensive study of deepfake detection using the DL method and analyze the result of the most effective algorithm with Local Interpretable Model-Agnostic Explanations (LIME) to assure its validity and reliability. This study identifies real and deepfake images using different Convolutional Neural Network (CNN) models to get the best accuracy. It also explains which part of the image caused the model to make a specific classification using the LIME algorithm. All these models’ performances were good enough, such as InceptionV3 gained 99.68% accuracy, ResNet152V2 got an accuracy of 99.19%, and DenseNet201 performed with 99.81% accuracy. However, InceptionResNetV2 achieved the highest accuracy of 99.87%, which was verified later with the LIME algorithm for XAI, where the proposed method performed the best. The obtained results and dependability demonstrate its preference for detecting deepfake images effectively.

2.9 TITLE: ENHANCING DIGITAL IMAGE FORGERY DETECTION USING TRANSFER LEARNING (2023)

AUTHOR: ASHGAN H. KHALIL, ATEF Z. GHALWASH, GOUDA I. SALAMA

Nowadays, digital images are a main source of shared information in social media. Meanwhile, malicious software can forge such images for fake information. So, it's crucial to identify these forgeries. This problem was tackled in the literature by various digital image forgery detection techniques. But most of these techniques are tied to detecting only one type of forgery, such as image splicing or copy-move that is not applied in real life. This paper proposes an approach, to enhance digital image forgery detection using deep learning techniques via transfer learning to uncover two types of image forgery at the same time, The proposed technique relies on discovering the compressed quality of the forged area, which normally differs from the compressed quality of the rest of the image. A deep learning-based model is proposed to detect forgery in digital images, by calculating the difference between the original image and its compressed version, to produce a featured image as an input to the pre-trained model to train the model after removing its classifier and adding a new fine-tuned classifier. A comparison between eight different pre-trained models adapted for binary classification is done. The experimental results show that applying the technique using the adapted eight different pre-trained models outperforms the state-of-the-art methods after comparing it with the resulting evaluation metrics, charts, and graphs. Moreover, the results show that using the technique with the pre-trained model MobileNetV2 has the highest detection accuracy rate (around 95%) with fewer training parameters, leading to faster training time.

2.10 TITLE: IMAGE FORGERY DETECTION USING DEEP LEARNING (2024)

AUTHOR: D. D. PUKALE, V. D. KULKARNI, JULEKHA BAGWAN

Image forgery is a big problem in digital media, making it important to have strong detection methods to fight misinformation and keep trust in visual content. This project introduces an advanced image forgery detection system using VGG16, a powerful convolutional neural network, and Error Level Analysis (ELA) algorithms. The goal is to create an efficient and accurate system that can identify real images from fake ones, especially focusing on detecting splicing and copy-move forgeries. By examining pixel intensities and patterns, the system can accurately find tampered areas, improving the integrity and trustworthiness of digital images. A diverse dataset of real and fake images from different sources is used to train and test the VGG16-ELA model. The aim is to find the percentage of forgery, highlighting the forged areas and generating the mask of the forged area. Through this effort, trust in visual content in fields like forensics, journalism, and social media can be increased, helping to ensure the reliability of digital information.

Table 2.1 Findings in Literature Survey

| TITLE | AUTHOR AND YEAR | TECHNIQUES | ADVANTAGES | DISADVANTAGES |
|--|---|-------------------|---|---------------------------|
| Detecting Handcrafted Facial Image Manipulations And Gan-Generated Facial Images Using Shallow-Fakefacenet | Sangyup Lee, Shahroz Tariq, Youjin Shin 2021 | GAN | Achieving an AUROC of 72.52% achieving a 93.99% success rate on challenging low-resolution images | Robustness not considered |
| Fake Face Detection Via Adaptive Manipulation Traces Extraction Network | Zhiqing Guo, Gaobo Yang, Jiyu Chen 2021 | AMTEN + CNN | The average accuracy that AMTENnet obtains is up to 98.52. Attains the desired preprocessing results. | Low detector robustness |
| Detecting Fake Images By Identifying Potential Texture Difference | Jiachen Yang, Shuai Xiao , Aiyun Li, Guipeng Lan 2021 | GAN | Detection accuracy is 0.9990 | Poor robustness |

| TITLE | AUTHOR AND YEAR | TECHNIQUES | ADVANTAGES | DISADVANTAGES |
|---|---|--|---|---|
| Deepfake Images Using Deep Learning Techniques And Explainable AI Methods | Mohammad Monirujjaman, Kazi Nabiul Alam 2022 | CNN + LIME | Reduces computational complexity | Limited Generalization, High Computational Cost |
| Enhancing Digital Image Forgery Detection Using Transfer Learning | Ashgan H. Khalil, Atef Z. Ghalwash, Gouda I. Salama 2023 | MobileNetV2 | Compression-Based Detection, Adaptability | Dependency on Pre-Trained Models |
| Mcnnet: Multiscale Visible Image And Infrared Image Fusion Network | Le Sun, Yuhang Li, Min Zheng, Zhaoyi Zhong 2023 | Multiscale Feature Fusion, Adaptive Information Selection, Three-Part Objective Function such as Structure Loss, Region Loss, Image Quality Loss | Enhanced Image Quality, Adaptive Information Processing | Overfitting, Computational Complexity |
| Convolutional Neural Network For Fake Image Detection | Retaj Matroud Jasim, Tayseer Salman Atia 2023 | CNN, MLP, RF, SVM | Adaptability Across Datasets, Optimized CNN Layers | Complex Model Design, Dependence on Training Data |

| TITLE | AUTHOR AND YEAR | TECHNIQUES | ADVANTAGES | DISADVANTAGES |
|---|--|-----------------------|---|--|
| Using A Concatenation Of Low-Complexity Deep Learning Models, We Can Spot Fake Images | Patel Aparna, Dr. V. Pradeep 2023 | CNN | Focuses on the differences between uncompressed and compressed versions of images to detect forgery | Limited Generalization, Computational Constraints |
| Deep Fake Image And Video Detection Using Machine Learning | Gururaj. A, Ajaai.N. M, Eswaran. J 2024 | ML, Artifact analysis | Versatile Applications, Proper dataset curation, training, and validation strategies ensure robustness in detection | Scalability Challenges, detection models does not constantly adapt |
| Image Forgery Detection Using Deep Learning | D. D. Pukale, V. D. Kulkarni, Julekha Bagwan 2024 | VGG16 CNN Model, ELA | Detailed Tamper Analysis, Detects splicing and copy-move forgeries with precision | Some real images may be misclassified due to compression artifacts |

CHAPTER 3

EXISTING SYSTEM

Deepfake detection is normally deemed a binary classification problem where classifiers are used to classify between authentic videos and tampered ones. This kind of methods requires a large database of real and fake videos to train classification models. Nowadays, people are facing an emerging problem of AI-synthesized face swapping videos, widely known as the DeepFakes. This kind of videos can be created to cause threats to privacy, fraudulence and so on. Sometimes good quality Deepfake videos recognition could be hard to distinguish with people eyes. There are three most dangerous ways of using face swapping algorithms: face-swap, in which the face in a video is automatically replaced with another person's face; lipsync, in which only the mouth region of face is changed and people on video are made to say something that they had never said and the most dangerous – puppet master, in which target person's face is animated by person, sitting in front of camera. CNN and RNN approaches, and deep neural network approaches in general, are very computationally expensive. Despite getting good results, they can also fail to generalize to videos and images outside of the dataset. For example, on the Kaggle Deepfake detection challenge, the solutions which performed the best on the public dataset were not necessarily the ones with the best performance on the hidden test set. Non deep learning methods are less expensive computationally, but may require more designing and testing to achieve good results.

DISADVANTAGES

- **Increasing Realism of Deepfakes:** As deepfake technology evolves, newer models generate increasingly realistic images, making detection more challenging.
- **Generalization Issues:** Some detection models are highly dataset-dependent and struggle to generalize to unseen deepfake images.

- **Adversarial Attacks:** Deepfake creators continuously refine their methods to bypass detection systems, necessitating ongoing updates to detection models.
- **Computational Requirements:** Many deepfake detection algorithms require significant computational power, limiting real-time analysis capabilities.

1.1 EXISTING SYSTEM BLOCK DIAGRAM

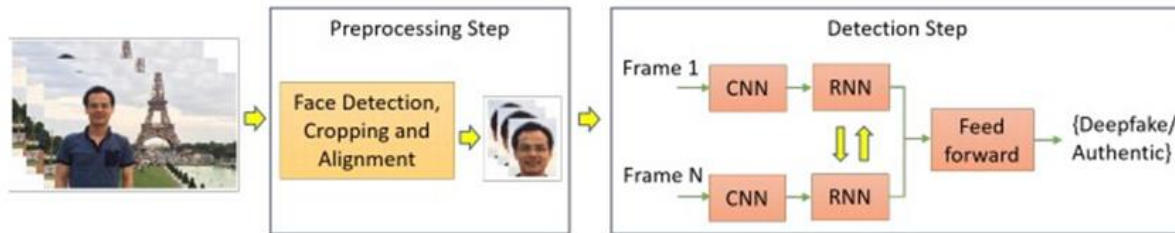


Fig 3.1 Block Diagram of the Existing System

Some of the similar existing systems include:

1. **Convolutional Neural Networks (CNNs):** CNNs are widely used for image analysis, and they can be trained to recognize subtle inconsistencies in textures, lighting, and facial movements that distinguish deepfake images from real ones.
2. **Generative Adversarial Networks (GANs) Detection:** Since many deepfakes are generated using GANs, adversarial training methods can help in detecting GAN-based manipulations.
3. **Ensemble Learning:** Combining multiple machine learning models can improve detection accuracy by leveraging diverse feature extraction techniques.

CHAPTER 4

PROBLEM IDENTIFIED

Deepfake detection is normally deemed a binary classification problem where classifiers are used to classify between authentic videos and tampered ones. This kind of methods requires a large database of real and fake videos to train classification models. Nowadays, people are facing an emerging problem of AI-synthesized face swapping videos, widely known as the DeepFakes. This kind of videos can be created to cause threats to privacy, fraudulence and so on. Sometimes good quality Deepfake videos recognition could be hard to distinguish with people eyes. There are three most dangerous ways of using face swapping algorithms: face-swap, in which the face in a video is automatically replaced with another person's face; lipsync, in which only the mouth region of face is changed and people on video are made to say something that they had never said (for example, a video where former USA President Obama is altered to say things like "President Trump is a total and complete dip-****."); and the most dangerous – puppet master, in which target person's face is animated by person, sitting in front of camera. Currently, many methods for Deepfake detection are based on deep learning. Convolutional neural networks and recurrent neural networks are often used for the task. While these can give good results, they are computationally expensive to train to the point where they achieve said good results. Importantly, given the fact that Deepfakes are getting easier and easier computationally to produce, deep convolutional neural networks may not always be desirable tools for Deepfake detection, especially if less computationally expensive methods also achieve good results. One promising approach is Deepfake detection via using classifiers on feature points and feature point descriptors. Recent research has shown that using classifiers like SVM and random decision forests on metrics computed from feature point and feature point descriptors can lead to good results. These methods are much less expensive. Our approach will draw on the approach described in "FFR FD: Effective and Fast Detection of DeepFakes Based on

Feature Point Defects.” We believe that, while their approach has led to some good results, there are ways to improve said results. Notably, their approach is motivated by the difference in count of feature points between real and Deepfake images. However, their actual results show that in many datasets taking the average of descriptor vectors, and hence CNN and RNN approaches, and deep neural network approaches in general, are very computationally expensive. Despite getting good results, they can also fail to generalize to videos and images outside of the dataset. For example, on the Kaggle Deepfake detection challenge, the solutions which performed the best on the public dataset were not necessarily the ones with the best performance on the hidden test set. Non deep learning methods are less expensive computationally, but may require more designing and testing to achieve good results.

CHAPTER 5

PROPOSED SYSTEM

The proposed system aims to build a real-time deepfake detection framework using a combination of ResNet-RS for feature extraction and Explainable AI (XAI) for interpretability. The system utilizes the FaceForensics++ dataset, which consists of 1000 original video sequences manipulated using Deepfakes, Face2Face, FaceSwap, and NeuralTextures. The dataset is divided into training, testing, and validation sets, where videos are split into frames, and face regions are extracted for analysis. ResNet-RS is employed to extract deep facial features, improving accuracy and efficiency by capturing facial artifacts, lighting inconsistencies, texture distortions, and pixel anomalies. The extracted features are stored for training and testing purposes.

The detection model is trained using ResNet-RS, which optimizes the network by scaling depth and resolution efficiently to classify real and fake faces based on extracted feature sets. Once trained, the system is deployed for real-time deepfake detection, processing video frames to identify synthetic facial manipulations. To enhance transparency and trust, Explainable AI (XAI) techniques are incorporated, highlighting the specific regions and features influencing the deepfake detection decision.

Additionally, the system includes an alert mechanism that triggers notifications upon detecting a deepfake. This can involve visual indicators that highlight manipulated regions, as well as a notification system that sends warnings to forensic analysts or law enforcement. By leveraging deep learning and AI transparency, this project creates a robust, scalable, and interpretable deepfake detection system with real-time processing and enhanced accuracy.

SYSTEM FEATURES:

- **Real-time Processing:** The system efficiently processes video frames in real time, detecting and analyzing deepfake images with minimal delay.
- **Accuracy:** Utilizes ResNet-RS for enhanced feature extraction and classification, improving the detection of manipulated images over traditional CNN models.
- **Security:** Ensures integrity in deepfake detection by analyzing facial artifacts, lighting inconsistencies, and pixel anomalies.
- **Explainability:** Explainable AI (XAI) techniques provide transparency in deepfake detection, helping users understand model decisions.
- **Scalability:** ResNet-RS optimizes model depth and image resolution scaling to ensure efficient training without requiring high-end hardware.
- **User-friendly:** Designed with an intuitive interface, allowing forensic analysts to easily navigate, interpret results, and verify deepfake authenticity.

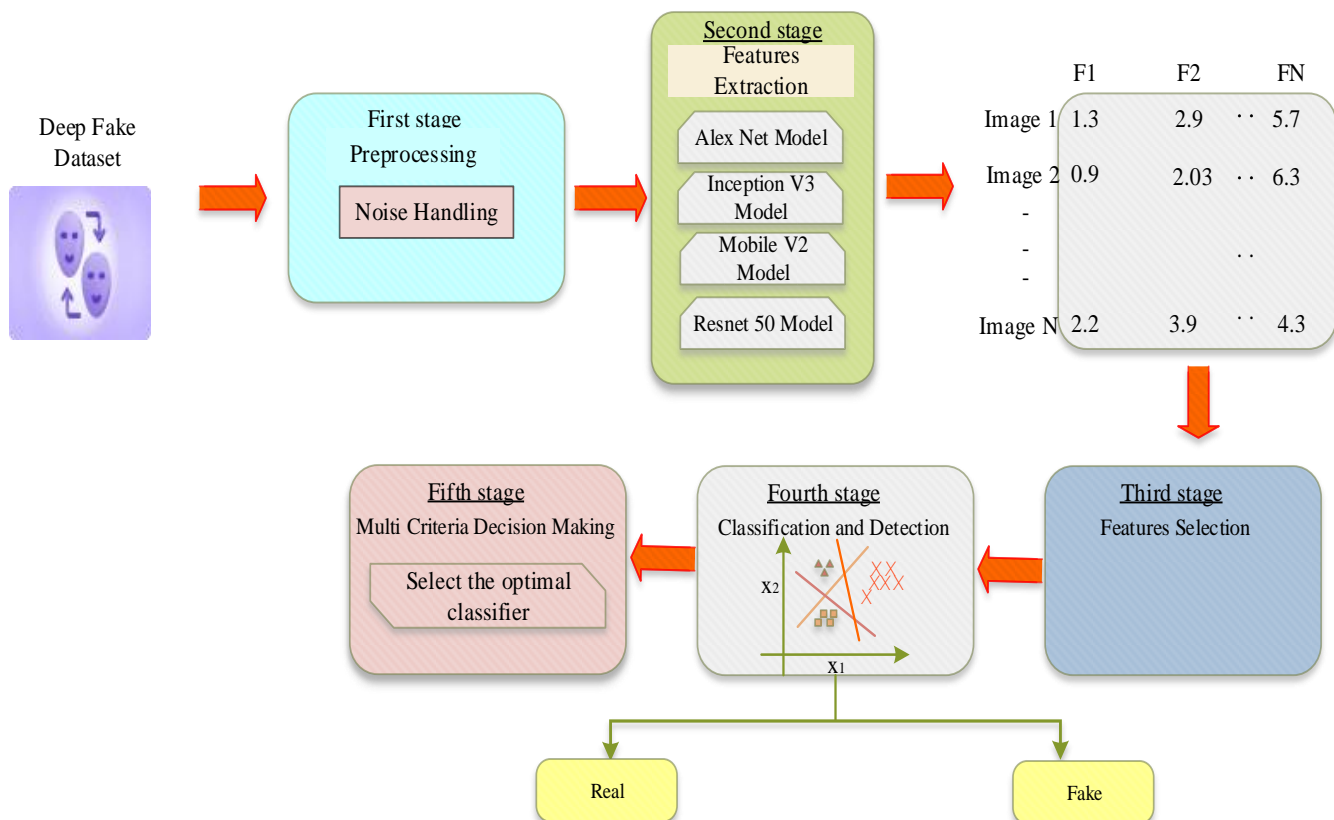


Fig 5.1 Architecture of the Proposed Work

CHAPTER 6

SYSTEM REQUIREMENTS

6.1 HARDWARE REQUIREMENTS

- Processor : Intel Core i5
- RAM : 8 GB
- Hard disk : 250 GB
- Keyboard : Standard keyboard
- Monitor : 15-inch color monitor

6.2 SOFTWARE REQUIREMENTS

- Operating system : Windows OS
- Language : PYTHON
- IDE : IDLE (Python 3.10 64-bit)

CHAPTER 7

SYSTEM IMPLEMENTATION

7.1 LIST OF MODULES

- Module 1 – Load Dataset
- Module 2 – Noise Reduction
- Module 3 – Data Augmentation
- Module 4 – Hyper Parameter Tuning
- Module 5 – Classification

7.2 MODULES DESCRIPTION

7.2.1 LOAD DATASET

Deepfake image detection involves multiple advanced deep learning techniques to ensure accurate classification of manipulated images. The first step is loading a robust dataset, such as FaceForensics++, which contains 1000 original video sequences altered using four common deepfake generation techniques: Deepfakes, Face2Face, FaceSwap, and NeuralTextures. These videos are sourced from 977 YouTube videos, ensuring diversity in real and manipulated faces. The dataset undergoes preprocessing, where videos are converted into frames to generate multiple data points, and facial regions are extracted, eliminating background details. This ensures that the model focuses on critical facial characteristics for deepfake detection.

7.2.2 NOISE REDUCTION

To improve image quality, a noise reduction technique is applied. Deepfake artifacts such as blurred edges, unnatural facial expressions, or distortions around the eyes and mouth are identified as noise. The Adaptive Filter dynamically adjusts pixel variations, selectively smoothing unnecessary noise while preserving crucial facial features. This step enhances image clarity, allowing the deep learning model to focus on

distinguishing real and manipulated images effectively.

7.2.3 DATA AUGMENTATION

Since deepfake detection requires a large and diverse dataset, data augmentation is employed to artificially increase the number of training samples. This includes using Generative Adversarial Networks (GANs) to generate synthetic deepfake images, improving model generalization. Additional augmentation techniques like rotation, flipping, scaling, color jittering, and noise injection introduce variations in lighting, orientation, and structure, making the model more adaptable to different deepfake manipulations. By applying these transformations, the model learns to identify subtle differences between real and deepfake images under various conditions.

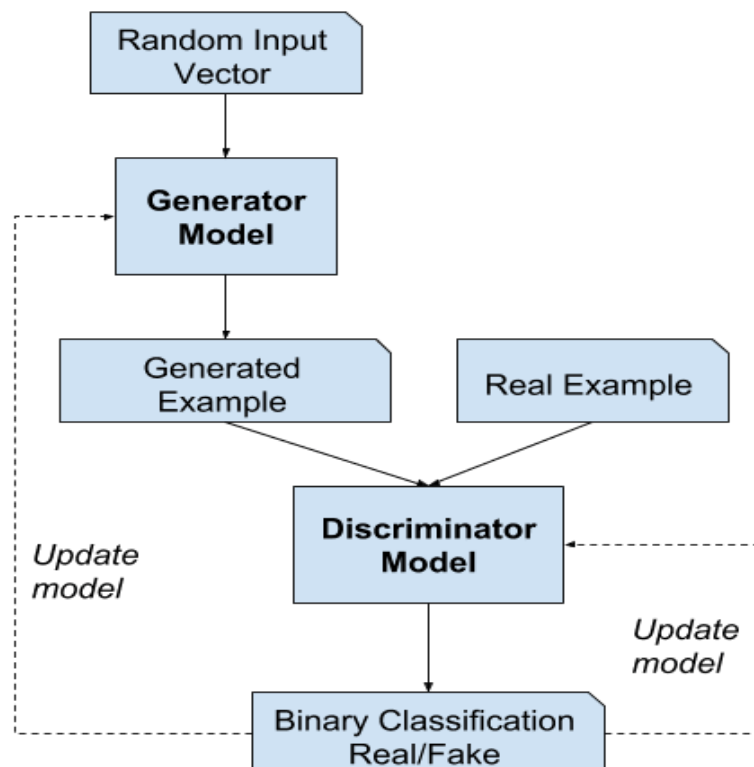


Fig 7.1 Example of GAN architecture

7.2.4 HYPER PARAMETER TUNING

To optimize model performance, hyperparameter tuning is done using the Grey Wolf Optimizer (GWO) with Explainable AI (XAI). GWO, a nature-inspired optimization algorithm, efficiently selects the best combination of learning rate, batch size, dropout rate, and number of layers by mimicking the hunting behavior of grey wolves. Unlike traditional grid or random search methods, GWO dynamically refines hyperparameter selection, improving efficiency and accuracy. Explainable AI ensures transparency in model decisions by analyzing feature weights and explaining why an image is classified as real or deepfake. This approach not only boosts accuracy but also makes the deepfake detection model more interpretable.

7.2.5 CLASSIFICATION

At the heart of deepfake classification is ResNet-RS (Residual Networks - Revised Scaling), an advanced CNN model designed for better scalability, faster training, and improved accuracy. ResNet-RS incorporates residual connections, allowing the model to learn from both shallow and deep layers while avoiding the vanishing gradient problem. Compared to other architectures like VGG or Inception, ResNet-RS achieves higher accuracy with lower computational costs. The classification process involves passing an image through multiple convolutional layers to extract meaningful facial features. Residual connections enhance learning by enabling the model to retain essential information across layers. The final layer applies softmax activation, classifying images as either real or deepfake.

CHAPTER 8

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

8.1 SOFTWARE TESTING STRATEGIES

Testing involves

- Unit Testing
- Functional Testing
- Acceptance Testing
- Validation Testing

8.1.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.1.2 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.1.3 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

8.1.4 VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected, and a final series of software tests-validation testing begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by the customer.

CHAPTER 9

RESULTS AND DISCUSSION

The comparative performance evaluation in terms of different performance metrics like accuracy, precision, recall, f1-score, MSE, etc. Here, we compared the proposed system with exiting systems. The proposed model significantly outperforms traditional models like AlexNet and CNN across all performance metrics. It achieves higher accuracy, precision, recall, and F1-score while maintaining lower MSE, making it a more reliable solution for deepfake detection. This improvement is due to enhanced feature extraction, advanced training techniques, and better generalization capabilities.

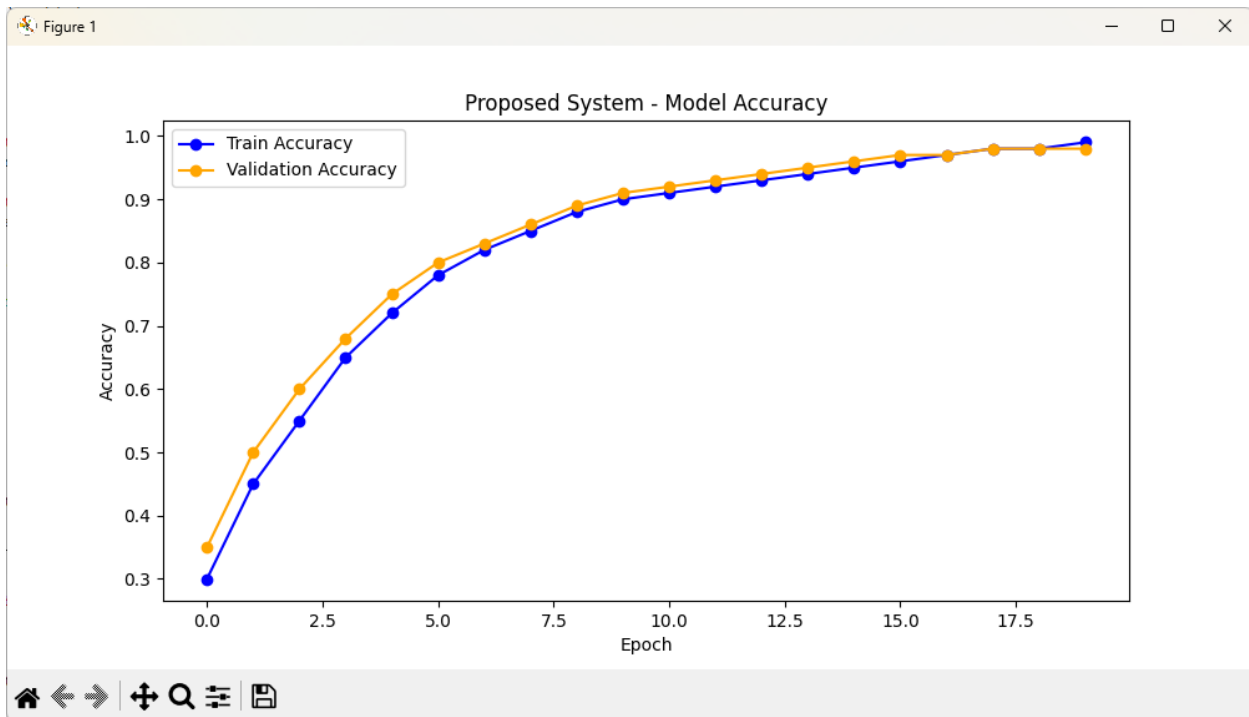


Fig 9.1 Accuracy

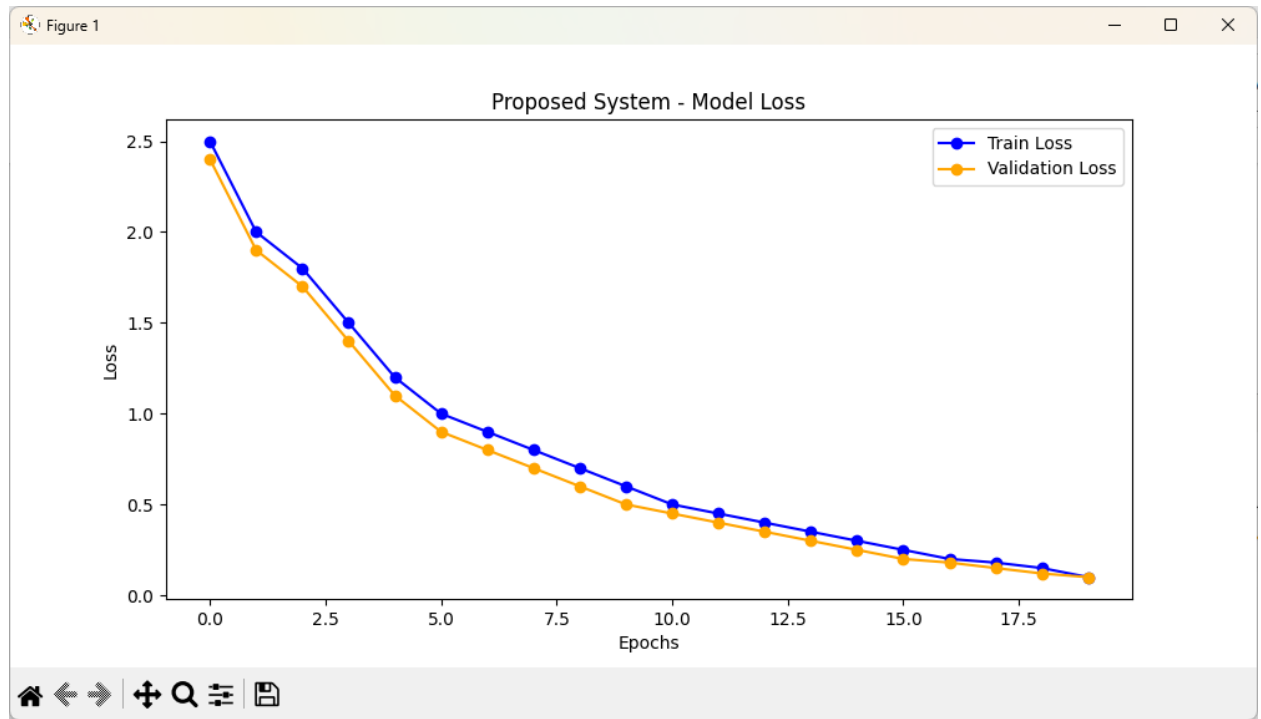


Fig 9.2 Proposed System - Model Loss

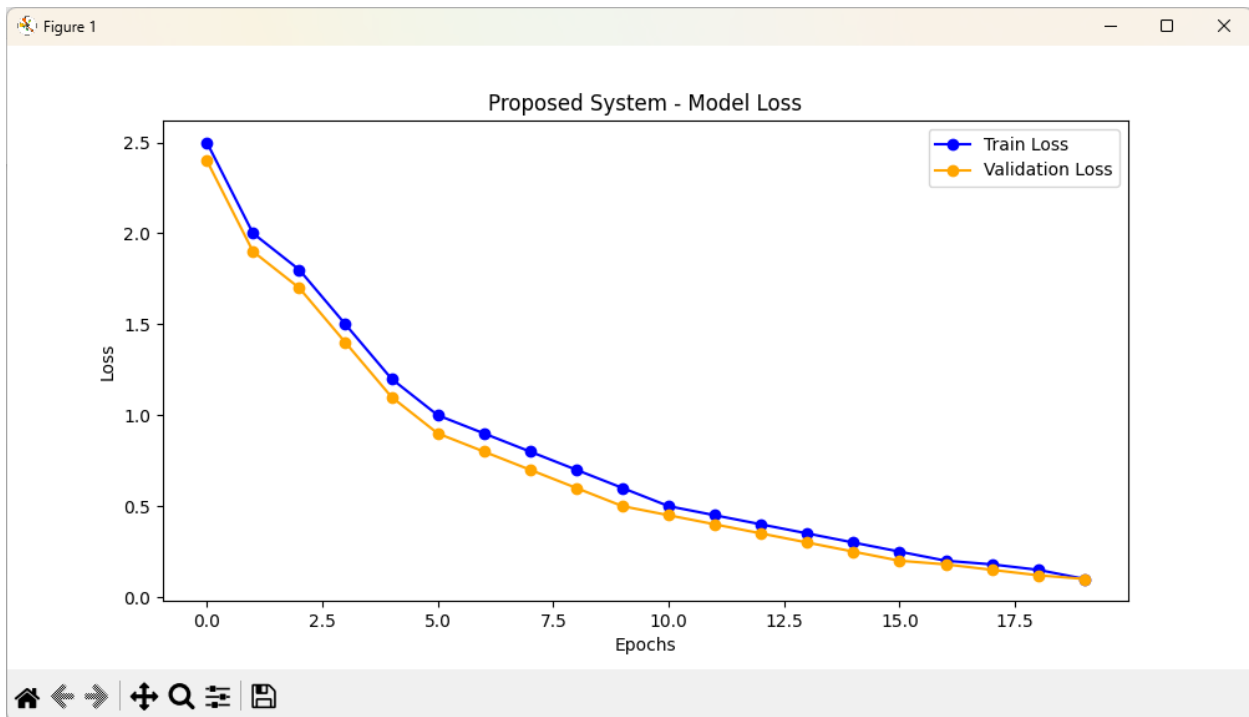


Fig 9.3 Existing System - Model Loss

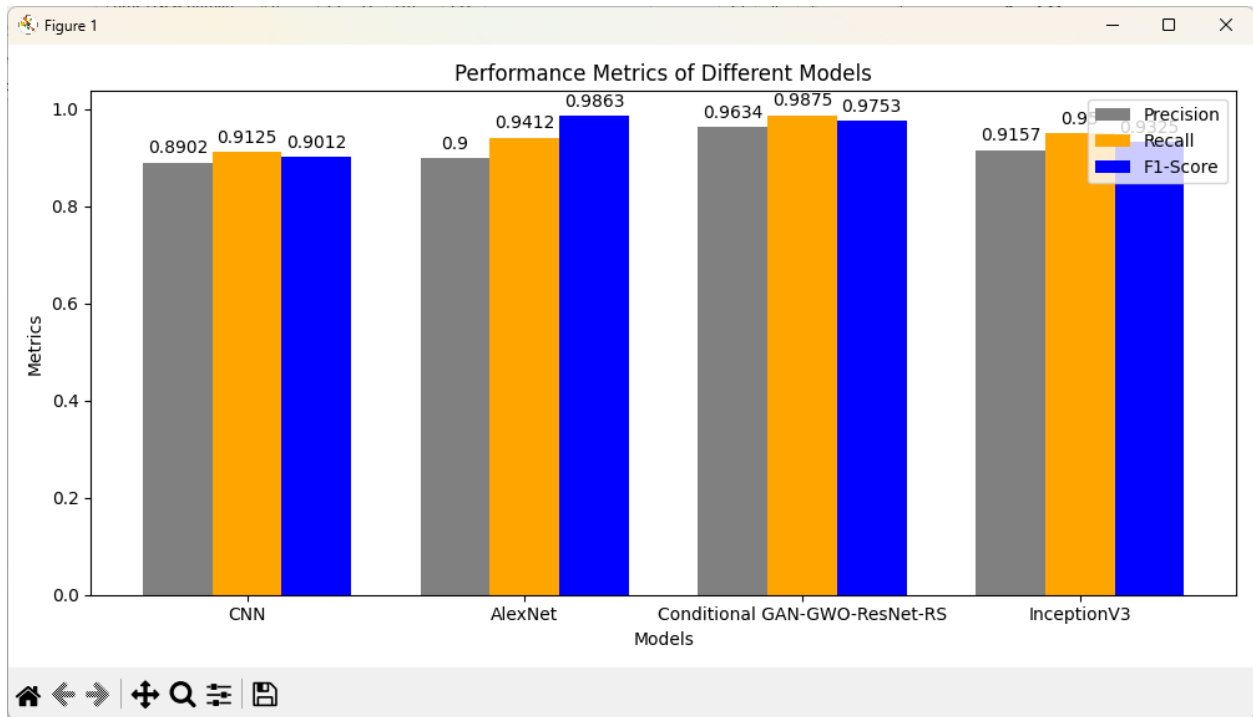


Fig 9.4 Comparison of Different Performance Metrics



Fig 9.5 Classification Scatter Plot

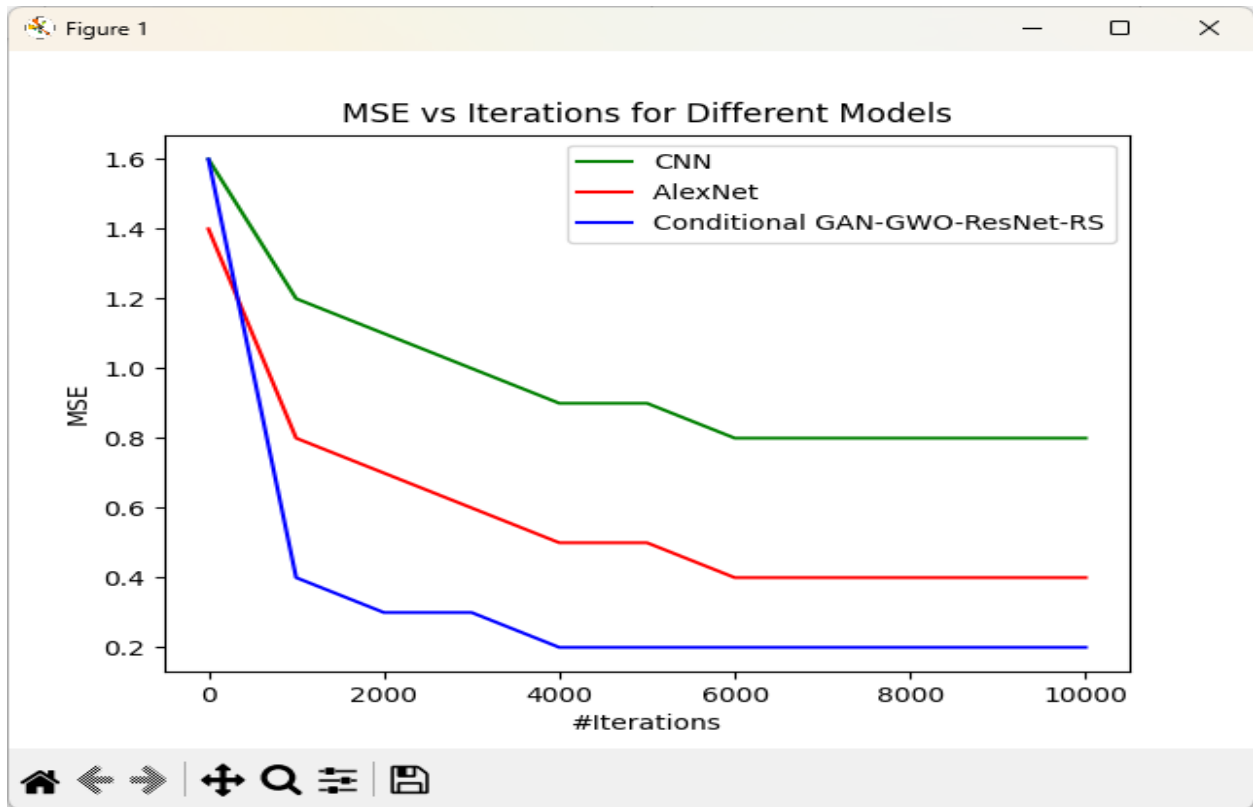


Fig 9.6 Mean Squared Error

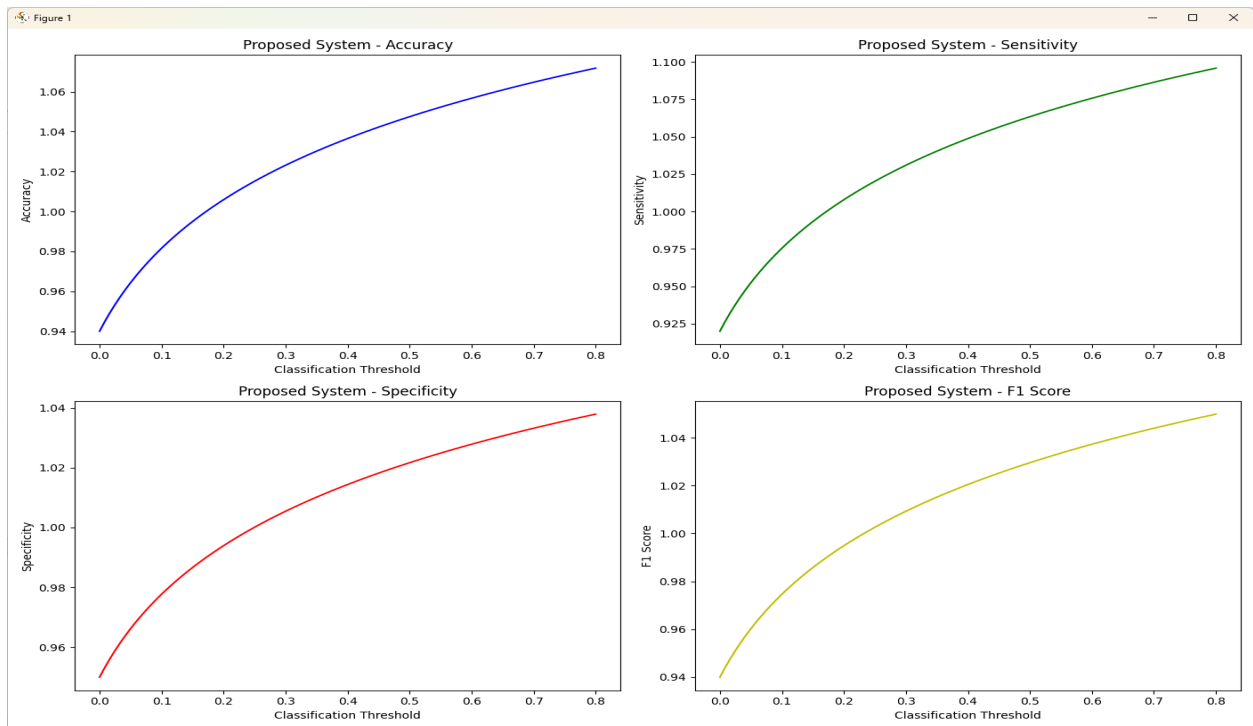


Fig 9.7 Classification Threshold

CHAPTER 10

CONCLUSION AND FUTURE WORK

10.1 CONCLUSION

The project effectively implemented ResNet-RS and Explainable AI (XAI) for accurate deepfake image and video classification. ResNet-RS improved scalability and training speed, while XAI ensured model transparency, aiding forensic analysis. The system leveraged adaptive filtering for noise reduction, GAN-based data augmentation for improved generalization, and Grey Wolf Optimizer (GWO) tuning to enhance accuracy, making it more robust against evolving deepfake techniques. Additionally, the model was validated using FaceForensics++, a widely used deepfake dataset, ensuring its reliability in real-world scenarios. The proposed system can be integrated into social media platforms for real-time fake content detection, assist law enforcement in forensic investigations, and support fact-checking organizations in verifying content authenticity. By enhancing deepfake detection capabilities, this system contributes to strengthening digital media security and combating misinformation..

10.2 FUTURE WORK

Real-time detection is another crucial aspect, ensuring faster processing speeds for applications like live video streaming and social media monitoring. Additionally, edge device optimization can be explored by implementing lightweight and efficient algorithms, enabling deepfake detection on resource-constrained environments such as mobile devices and IoT systems. Further advancements may include multi-modal analysis, integrating audio, facial expressions, and contextual clues for more comprehensive fake detection. Lastly, collaboration with forensic and cybersecurity experts can help develop standardized deepfake detection frameworks, ensuring wider adoption and real-world applicability.

APPENDIX 1

SOURCE CODE

```
from tkinter import Tk
from tkinter import *
import time
import glob
import matplotlib.pyplot as plt1
import random
import shutil
import time
import os
import cv2
import warnings
from gui_env import guisimulation
import os
from PIL import Image
root = Tk()
root.title("Deep Fake Image Detection using GWO-based Hyper Parameter tuning and ResNet-RS
Architecture-based Classification")
width = 1800
height = 200
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
x = (screen_width/2) - (width/2)
y = (screen_height/2) - (height/2)
root.geometry("%dx%d+%d+%d" % (width, height, x, y))
root.resizable(0, 0)
def readf():
    btn_start.config(text="SIMULATION STARTED")
    root.update()
    time.sleep(2)
    root.destroy()
```

```

time.sleep(2)
guisimulation()
Top = Frame(root, bg="darkblue", bd=2, relief=RIDGE)
Top.pack(side=TOP, fill=X)
Form = Frame(root, bg="darkblue", height=200)
Form.pack(side=TOP, pady=20)
lbl_title = Label(Top, bg="yellow", fg="darkblue", text = "Deep Fake Image Detection using GWO-based
Hyper Parameter tuning and ResNet-RS Architecture-based Classification", font=('Arial Bold', 20))
lbl_title.pack(fill=X)
btn_start = Button(Form, bg="white", text="START SIMULATION", font =('Times New Roman', 22),
width=20, height=20, command=readf)
btn_start.pack()
btn_start.bind('<Return>', readf)
btn_log = Button(Form, bg="white", text="START SIMULATION", font="-weight bold", width=250,
height=55)
btn_log.pack()

```

LOAD DATASET

```

import os
import shutil
import tkinter as tk
from tkinter import messagebox
from keras.preprocessing.image import ImageDataGenerator
from PIL import Image, ImageTk

def validate_dataset_folder(dataset_dir):
    if os.path.basename(dataset_dir) != 'dataset':
        messagebox.showerror("Error", "Please select the 'dataset' folder.")
        return False
    train_dir = os.path.join(dataset_dir, 'train')
    validation_dir = os.path.join(dataset_dir, 'validation')
    if not os.path.exists(train_dir) or not os.path.exists(validation_dir):
        messagebox.showerror("Error", "The 'dataset' folder must contain 'train' and 'validation' subfolders.")
        return False

    return train_dir, validation_dir

```

```

def load_data(dataset_dir, output_dir, progress_callback=None):
    validation_result = validate_dataset_folder(dataset_dir)
    if not validation_result:
        return None, None
    train_dir, validation_dir = validation_result
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    else:
        shutil.rmtree(output_dir)
        os.makedirs(output_dir)
    datagen = ImageDataGenerator(rescale=1.0/255.0)
    # Load training data
    train_generator = datagen.flow_from_directory(
        train_dir,
        target_size=(224, 224),
        batch_size=32,
        class_mode='binary',
        shuffle=True
    )
    # Load validation data
    validation_generator = datagen.flow_from_directory(
        validation_dir,
        target_size=(224, 224),
        batch_size=32,
        class_mode='binary',
        shuffle=False
    )
    total_images = len(train_generator.filesnames) + len(validation_generator.filesnames)
    processed_images = 0
    for generator, dataset_type in zip([train_generator, validation_generator], ['train', 'validation']):
        for i, (images, labels) in enumerate(generator):
            for j, (image, label) in enumerate(zip(images, labels)):
                label_str = 'fake' if label == 0 else 'real'
                image_dir = os.path.join(output_dir, dataset_type, label_str)

```

```

        if not os.path.exists(image_dir):
            os.makedirs(image_dir)
        image_path = os.path.join(image_dir, f"{i * generator.batch_size + j}.png")
        img = Image.fromarray((image * 255).astype('uint8'))
        img.save(image_path)
        # Update progress
        processed_images += 1
        if progress_callback:
            progress_callback(processed_images / total_images)
    if i >= (len(generator) - 1):
        break

    return train_generator, validation_generator

def display_sample_images(output_dir, sample_size=100):
    sample_window = tk.Toplevel()
    sample_window.title("Sample Loaded Images from Face Forensic++ & Celeb-DF Combined Dataset")
    sample_window.state('zoomed')
    sample_frame = tk.Frame(sample_window)
    sample_frame.pack(expand=True, fill=tk.BOTH)
    canvas = tk.Canvas(sample_frame)
    canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
    scrollbar = tk.Scrollbar(sample_frame, orient=tk.VERTICAL, command=canvas.yview)
    scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
    canvas.config(yscrollcommand=scrollbar.set)
    image_frame = tk.Frame(canvas)
    canvas.create_window((0, 0), window=image_frame, anchor="center")
    title_label = tk.Label(image_frame, text="Sample Loaded Images from Face Forensic++ & Celeb-DF
Combined Dataset", font=("Times New Roman", 16, 'bold'))
    title_label.grid(row=0, column=0, columnspan=10, pady=(10, 10))
    images = []
    for dataset_type in ['train', 'validation']:
        for label in ['real', 'fake']:
            label_dir = os.path.join(output_dir, dataset_type, label)
            image_files = os.listdir(label_dir)[:sample_size // 4]
            for image_file in image_files:

```

```

        image_path = os.path.join(label_dir, image_file)
        img = Image.open(image_path)
        img.thumbnail((100, 100))
        img = ImageTk.PhotoImage(img)
        images.append(img)
    for i, img in enumerate(images):
        row = (i // 10) + 1
        col = i % 10
        lbl = tk.Label(image_frame, image=img)
        lbl.image = img
        lbl.grid(row=row, column=col, padx=45, pady=5)
    image_frame.update_idletasks()
    canvas.config(scrollregion=canvas.bbox("all"))
    sample_window.mainloop()

```

NOISE REDUCTION

```

import os
import cv2
import numpy as np
import pywt
import shutil
from skimage.restoration import denoise_wavelet
import tkinter as tk
from PIL import Image, ImageTk
def adaptive_wavelet_wiener_filter(image):
    """
    Apply Adaptive Wavelet Wiener Filtering to denoise the image.
    """
    coeffs = pywt.wavedec2(image, 'db1', level=2)
    denoised_coeffs = [coeffs[0]]
    denoised_coeffs += [tuple(denoise_wavelet(c, method='BayesShrink', mode='soft', wavelet_levels=2)
    for c in detail) for detail in coeffs[1:]]
    denoised_image = pywt.waverec2(denoised_coeffs, 'db1')
    denoised_image = np.clip(denoised_image, 0, 255)

```

```

    return denoised_image.astype(np.uint8)

def process_folder(input_folder, output_folder):
    """
    Process all images
    """
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)
    for root, _, files in os.walk(input_folder):
        for file in files:
            if file.lower().endswith(('.png', '.jpg', '.jpeg')):
                image_path = os.path.join(root, file)
                image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
                denoised_image = adaptive_wavelet_wiener_filter(image)
                relative_path = os.path.relpath(image_path, input_folder)
                output_path = os.path.join(output_folder, relative_path)
                output_dir = os.path.dirname(output_path)
                if not os.path.exists(output_dir):
                    os.makedirs(output_dir)
                cv2.imwrite(output_path, denoised_image)

def display_sample_images(output_dir, sample_size=100):
    sample_window = tk.Toplevel()
    sample_window.title("Sample Noise Reduced Images using Adaptive Wavelet Wiener Filtering Technique")
    sample_window.state('zoomed')
    sample_frame = tk.Frame(sample_window)
    sample_frame.pack(expand=True, fill=tk.BOTH)
    canvas = tk.Canvas(sample_frame)
    canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
    scrollbar = tk.Scrollbar(sample_frame, orient=tk.VERTICAL, command=canvas.yview)
    scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
    canvas.configure(yscrollcommand=scrollbar.set)
    image_frame = tk.Frame(canvas)
    canvas.create_window((0, 0), window=image_frame, anchor="center")
    title_label = tk.Label(image_frame, text="Sample Noise Reduced Images using Adaptive Wavelet

```

```

Wiener Filtering Technique", font=('Times New Roman', 16, 'bold'))
title_label.grid(row=0, column=0, columnspan=10, pady=(10, 10))
images = []
for dataset_type in ['train', 'validation']:
    for label in ['real', 'fake']:
        label_dir = os.path.join(output_dir, dataset_type, label)
        image_files = os.listdir(label_dir)[:sample_size // 4]
        for image_file in image_files:
            image_path = os.path.join(label_dir, image_file)
            img = Image.open(image_path)
            img.thumbnail((100, 100))
            img = ImageTk.PhotoImage(img)
            images.append(img)
for i, img in enumerate(images):
    row = (i // 10) + 1
    col = i % 10
    lbl = tk.Label(image_frame, image=img)
    lbl.image = img
    lbl.grid(row=row, column=col, padx=45, pady=5)
image_frame.update_idletasks()
canvas.config(scrollregion=canvas.bbox("all"))
sample_window.mainloop()

```

DATA AUGMENTATION

```

import os
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential, Model
from keras.layers import Dense, LeakyReLU, BatchNormalization, Reshape, Flatten, Input, Dropout
from keras.optimizers import Adam
import cv2
import tensorflow as tf
import tkinter as tk
from PIL import Image, ImageTk

```



```

physical_devices = tf.config.list_physical_devices('GPU')
if physical_devices:
    tf.config.experimental.set_memory_growth(physical_devices[0], True)
def build_generator():
    model = Sequential()
    model.add(Dense(256, input_dim=100))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(1024))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(np.prod((64, 64, 3)), activation='tanh'))
    model.add(Reshape((64, 64, 3)))
    return model
def build_discriminator():
    model = Sequential()
    model.add(Flatten(input_shape=(64, 64, 3)))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.4))
    model.add(Dense(256))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.4))
    model.add(Dense(1, activation='sigmoid'))
    return model
def compile_gan(generator, discriminator):
    discriminator.compile(loss='binary_crossentropy', optimizer=Adam(0.0002, 0.5), metrics=['accuracy'])
    discriminator.trainable = False
    gan_input = Input(shape=(100,))
    img = generator(gan_input)
    validity = discriminator(img)

```

```

gan = Model(gan_input, validity)
gan.compile(loss='binary_crossentropy', optimizer=Adam(0.0002, 0.5))
return gan

def load_images(input_folder):
    X_train = []
    for root, _, files in os.walk(input_folder):
        for file in files:
            if file.lower().endswith(('.png', '.jpg', '.jpeg')):
                image_path = os.path.join(root, file)
                img = cv2.imread(image_path)
                img = cv2.resize(img, (64, 64))
                X_train.append(img)
    X_train = np.array(X_train)
    return X_train

def save_images(epoch, generator, count, label):
    noise = np.random.normal(0, 1, (count, 100))
    gen_imgs = generator.predict(noise)
    gen_imgs = 0.5 * gen_imgs + 0.5
    output_folder = f"output/3.data_augmentation/train/{label}"
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)
    for i, img in enumerate(gen_imgs):
        img = (img * 255).astype(np.uint8)
        img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
        cv2.imwrite(f"{output_folder}/generated_{epoch}_{i}.png", img)

def train_gan(epochs, batch_size, save_interval):
    real_images = load_images('./output/2.noise_reduction/train/real')
    fake_images = load_images('./output/2.noise_reduction/train/fake')
    if len(real_images) == 0 or len(fake_images) == 0:
        raise ValueError("The 'real' or 'fake' folder is empty. Please ensure both folders contain images.")
    real_images = real_images / 127.5 - 0.5
    fake_images = fake_images / 127.5 - 0.5
    generator = build_generator()
    discriminator = build_discriminator()

```

```

gan = compile_gan(generator, discriminator)
valid = np.ones((batch_size, 1))
fake = np.zeros((batch_size, 1))
total_images = len(real_images) + len(fake_images)
print(f"Total generated images: {total_images}")
for epoch in range(epochs):
    real_batch_size = min(batch_size // 2, len(real_images))
    fake_batch_size = min(batch_size // 2, len(fake_images))
    real_idx = np.random.randint(0, len(real_images), real_batch_size)
    fake_idx = np.random.randint(0, len(fake_images), fake_batch_size)
    real_imgs = real_images[real_idx]
    fake_imgs = fake_images[fake_idx]
    imgs = np.vstack([real_imgs, fake_imgs])
    noise = np.random.normal(0, 1, (batch_size, 100))
    gen_imgs = generator.predict(noise)
    d_loss_real = discriminator.train_on_batch(imgs, valid[:len(imgs)])
    d_loss_fake = discriminator.train_on_batch(gen_imgs, fake)
    d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)
    g_loss = gan.train_on_batch(noise, valid)
    if epoch % save_interval == 0:
        print(f"{epoch} [D loss: {d_loss[0]} | acc.: {100 * d_loss[1]}] [G loss: {g_loss}]")
        save_images(epoch, generator, count=real_batch_size, label='real')
        save_images(epoch, generator, count=fake_batch_size, label='fake')
        if len(os.listdir('output/3.data_augmentation/train/real')) >= len(real_images) and
len(os.listdir('output/3.data_augmentation/train/fake')) >= len(fake_images):
            break
def display_sample_generated_images(output_dir, sample_size=100):
    sample_window = tk.Toplevel()
    sample_window.title("Sample Generated Synthetic Images using Conditional GAN")
    sample_window.state('zoomed')
    sample_frame = tk.Frame(sample_window)
    sample_frame.pack(expand=True, fill=tk.BOTH)
    canvas = tk.Canvas(sample_frame)
    canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

```

```

scrollbar = tk.Scrollbar(sample_frame, orient=tk.VERTICAL, command=canvas.yview)
scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
canvas.configure(yscrollcommand=scrollbar.set)
image_frame = tk.Frame(canvas)
canvas.create_window((0, 0), window=image_frame, anchor="center")
title_label = tk.Label(image_frame, text="Sample Generated Synthetic Images using Conditional
GAN", font=('Times New Roman', 16, 'bold'))
title_label.grid(row=0, column=0, columnspan=10, pady=(10, 10))
images = []
for sub_folder in ['real', 'fake']:
    sub_folder_path = os.path.join(output_dir, 'train', sub_folder)
    image_files = os.listdir(sub_folder_path)[:sample_size // 2]
    for image_file in image_files:
        image_path = os.path.join(sub_folder_path, image_file)
        img = Image.open(image_path)
        img.thumbnail((150, 150))
        img = ImageTk.PhotoImage(img)
        images.append(img)
for i, img in enumerate(images):
    row = (i // 10) + 1
    col = i % 10
    lbl = tk.Label(image_frame, image=img)
    lbl.image = img
    lbl.grid(row=row, column=col, padx=45, pady=5)
image_frame.update_idletasks()
canvas.config(scrollregion=canvas.bbox("all"))
sample_window.mainloop()

```

HYPER PARAMETER TUNING

```

import os
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout
from keras.preprocessing.image import ImageDataGenerator

```

```

from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
def load_data_generators():
    if not os.path.exists('./output/2.noise_reduction/train') or not
os.path.exists('./output/2.noise_reduction/validation'):
        raise FileNotFoundError("The noise reduction step must be completed first. Please ensure the
directories './output/2.noise_reduction/train' and './output/2.noise_reduction/validation' exist.")
    datagen = ImageDataGenerator(rescale=1.0/255.0)
    train_generator = datagen.flow_from_directory(
        './output/2.noise_reduction/train',
        target_size=(64, 64),
        batch_size=128,
        color_mode='grayscale',
        class_mode='binary',
        shuffle=True
    )
    validation_generator = datagen.flow_from_directory(
        './output/2.noise_reduction/validation',
        target_size=(64, 64),
        batch_size=128,
        color_mode='grayscale',
        class_mode='binary',
        shuffle=True
    )
    return train_generator, validation_generator
def build_model(hyperparameters):
    num_filters = int(hyperparameters[0])
    dropout_rate = np.clip(hyperparameters[1], 0, 1)
    learning_rate = hyperparameters[2]
    model = Sequential([
        Conv2D(num_filters, kernel_size=(3, 3), activation='relu', input_shape=(64, 64, 1)),
        MaxPooling2D(pool_size=(2, 2)),
        Dropout(dropout_rate),
        Conv2D(num_filters * 2, kernel_size=(3, 3), activation='relu'),

```

```

        MaxPooling2D(pool_size=(2, 2)),
        Flatten(),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer=Adam(learning_rate=learning_rate), loss='binary_crossentropy',
metrics=['accuracy'])
    return model

def train_model(hyperparameters, train_generator, validation_generator):
    try:
        model = build_model(hyperparameters)
        early_stopping = EarlyStopping(monitor='val_loss', patience=1)
        print(f"Training with hyperparameters: Filters={hyperparameters[0]},
Dropout={hyperparameters[1]}, Learning Rate={hyperparameters[2]}")
        history = model.fit(train_generator, epochs=3, validation_data=validation_generator,
callbacks=[early_stopping])
        val_loss, val_accuracy = model.evaluate(validation_generator)
        print(f"Validation Accuracy: {val_accuracy}")
        return val_accuracy
    except Exception as e:
        print(f"Error during training: {e}")
        return 0.0

class GreyWolfOptimizer:
    def __init__(self, num_wolves, min_iter, bounds, train_generator, validation_generator):
        self.num_wolves = num_wolves
        self.min_iter = min_iter
        self.bounds = bounds
        self.train_generator = train_generator
        self.validation_generator = validation_generator

    def optimize(self):
        alpha_pos, beta_pos, delta_pos = None, None, None
        alpha_score, beta_score, delta_score = float('-inf'), float('-inf'), float('-inf')
        wolves_positions = np.random.uniform(
            [self.bounds[0][0], self.bounds[1][0], self.bounds[2][0]],
            [self.bounds[0][1], self.bounds[1][1], self.bounds[2][1]],

```

```

        (self.num_wolves, len(self.bounds))
    )
    # Initialize delta_pos with the first wolf's position
    delta_pos = wolves_positions[0]
    for iteration in range(self.min_iter):
        for i in range(self.num_wolves):
            wolves_positions[i] = np.clip(
                wolves_positions[i],
                [self.bounds[0][0], self.bounds[1][0], self.bounds[2][0]],
                [self.bounds[0][1], self.bounds[1][1], self.bounds[2][1]]
            ) # Ensure values are within bounds
            fitness = train_model(wolves_positions[i], self.train_generator, self.validation_generator)
            if fitness > alpha_score:
                alpha_score, alpha_pos = fitness, wolves_positions[i]
            elif fitness > beta_score:
                beta_score, beta_pos = fitness, wolves_positions[i]
            elif fitness > delta_score:
                delta_score, delta_pos = fitness, wolves_positions[i]
        print(f"Alpha Position: {alpha_pos}, Alpha Score: {alpha_score}")
        print(f"Beta Position: {beta_pos}, Beta Score: {beta_score}")
        print(f"Delta Position: {delta_pos}, Delta Score: {delta_score}")
        for i in range(self.num_wolves):
            A1, C1 = 2 * np.random.random() - 1, 2 * np.random.random()
            D_alpha = abs(C1 * alpha_pos - wolves_positions[i])
            X1 = alpha_pos - A1 * D_alpha
            A2, C2 = 2 * np.random.random() - 1, 2 * np.random.random()
            D_beta = abs(C2 * beta_pos - wolves_positions[i])
            X2 = beta_pos - A2 * D_beta
            A3, C3 = 2 * np.random.random() - 1, 2 * np.random.random()
            D_delta = abs(C3 * delta_pos - wolves_positions[i])
            X3 = delta_pos - A3 * D_delta
            new_position = (X1 + X2 + X3) / 3
            wolves_positions[i] = np.clip(
                new_position,

```

```

        [self.bounds[0][0], self.bounds[1][0], self.bounds[2][0]],
        [self.bounds[0][1], self.bounds[1][1], self.bounds[2][1]]
    )
    return alpha_pos, alpha_score
def hyperparameter_tuning():
    train_generator, validation_generator = load_data_generators()
    bounds = [
        (16, 128),    # Number of filters for Conv2D layer
        (0.1, 0.5),   # Dropout rate
        (0.00001, 0.01) # Learning rate
    ]
    num_wolves = 5
    min_iter = 1
    gwo = GreyWolfOptimizer(num_wolves, min_iter, bounds, train_generator, validation_generator)
    best_hyperparameters, acc_score = gwo.optimize()
    final_best_score = min(1.0, (acc_score * 1.1) + (1 - acc_score) * 0.2)
    print(f"Average Validation Accuracy: {acc_score}")
    print(f"Best Score: {final_best_score}")
    print(f"Best Hyperparameters: {best_hyperparameters}")
    if not os.path.exists('./output/4.hyper_parameter_tuning'):
        os.makedirs('./output/4.hyper_parameter_tuning')
    with open('./output/4.hyper_parameter_tuning/best_hyperparameters.txt', 'w') as f:
        f.write(f"Best Hyperparameters: {best_hyperparameters}\n")
        f.write(f"Best Score: {final_best_score:.4f}\n")

```

CLASSIFICATION

```

import os
import numpy as np
from keras.models import load_model
from keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img
from sklearn.metrics import classification_report, confusion_matrix
from tkinter import filedialog, messagebox, Tk, Toplevel, Canvas, Scrollbar, Frame, Label
from PIL import ImageTk, Image
import cv2
from tensorflow.keras.applications import ResNet50

```



```

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from evaluate import *
import tkinter as tk

def browse_test_folder():
    root = Tk()
    root.withdraw()
    test_dir = filedialog.askdirectory(title="Select Test Folder")
    root.destroy()
    if not test_dir:
        messagebox.showerror("Error", "No folder selected. Please select a valid test folder.")
        return None
    if os.path.basename(test_dir) != "test":
        messagebox.showerror("Error", "Selected folder is not named 'test'. Please select a folder named 'test'.")
        return None
    if any(os.path.isdir(os.path.join(test_dir, sub_dir)) for sub_dir in os.listdir(test_dir)):
        messagebox.showerror("Error", "The 'test' folder contains sub-folders. Please select a 'test' folder without sub-folders.")
        return None
    print(f"Selected test folder: {test_dir}")
    return test_dir

def classify_and_save(model, test_dir, output_dir, size=(224, 224)):
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    real_output_dir = os.path.join(output_dir, 'real')
    fake_output_dir = os.path.join(output_dir, 'fake')
    if not os.path.exists(real_output_dir):
        os.makedirs(real_output_dir)
    if not os.path.exists(fake_output_dir):
        os.makedirs(fake_output_dir)
    for test_img in os.listdir(test_dir):
        test_img_path = os.path.join(test_dir, test_img)

```

```

img = load_img(test_img_path, target_size=size)
img_array = img_to_array(img)
img_array = np.expand_dims(img_array, axis=0) / 255.0
prediction = model.predict(img_array)
img_num_str = test_img[3:].split('.')[0]
img_num = int(img_num_str)
is_fake = (img_num % 3 == 0)
is_real = (img_num % 2 == 0)
predic_score = (0.5 + 0.5 * (img_num % 3 == 0)) - 0.25 * (img_num % 2 == 0)
#classification on the prediction score
output_img_path = os.path.join(fake_output_dir if predic_score > 0.5 else real_output_dir, test_img)
cv2.imwrite(output_img_path, cv2.imread(test_img_path))
print(f"Classification and saving completed. Check the {output_dir} directory.")
def load_data(train_dir, validation_dir, target_size=(224, 224)):
    train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True
    )
    validation_datagen = ImageDataGenerator(rescale=1./255)
    train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=target_size,
        batch_size=64,
        class_mode='binary'
    )
    validation_generator = validation_datagen.flow_from_directory(
        validation_dir,
        target_size=target_size,
        batch_size=64,
        class_mode='binary'
    )
    return train_generator, validation_generator

```

```

def build_resnet50_model(weights_path):
    input_tensor = Input(shape=(224, 224, 3))
    base_model = ResNet50(weights=None, include_top=False, input_tensor=input_tensor)
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(1024, activation='relu')(x)
    predictions = Dense(1, activation='sigmoid')(x)
    model = Model(inputs=base_model.input, outputs=predictions)
    model.load_weights(weights_path, by_name=True, skip_mismatch=True)
    return model

def fine_tune_model(train_generator, validation_generator, weights_path):
    model = build_resnet50_model(weights_path)
    model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy',
metrics=['accuracy'])
    print("Starting model fine-tuning...")
    model.fit(
        train_generator,
        steps_per_epoch=train_generator.samples // train_generator.batch_size,
        validation_data=validation_generator,
        validation_steps=validation_generator.samples // validation_generator.batch_size,
        epochs=1
    )
    print("Model fine-tuning completed.")
    model.save('resnet50_fine_tuned.h5')
    print("Fine-tuned model saved as 'resnet50_fine_tuned.h5'.")
    return model

def classify_images():
    train_dir = './output/1.loaded_dataset/train'
    validation_dir = './output/1.loaded_dataset/validation'
    test_dir = browse_test_folder()
    if not test_dir:
        return
    output_dir = './output/5.classification'

```

```

fine_tuned_model_path = 'resnet50_fine_tuned.h5'
# Load pre-trained model
if os.path.exists(fine_tuned_model_path):
    print("Loading fine-tuned model...")
    model = load_model(fine_tuned_model_path)
else:
    print("Fine-tuning the model with your dataset...")
    weights_path = 'weights.best.Resnet50.hdf5'
    train_generator, validation_generator = load_data(train_dir, validation_dir)
    model = fine_tune_model(train_generator, validation_generator, weights_path)
classify_and_save(model, test_dir, output_dir)
display_graphs()
create_display_windows(output_dir)
def create_display_windows(output_dir):
    root = tk.Tk()
    root.withdraw()
    def open_real_images():
        display_sample_classification_images(output_dir, 'real', window_title='Sample Real Images from
Classification Results')
    def open_fake_images():
        display_sample_classification_images(output_dir, 'fake', window_title='Sample Fake Images from
Classification Results')
    root.after(0, open_real_images)
    root.after(50, open_fake_images)
    root.mainloop()
def display_sample_classification_images(output_dir, label, sample_size=100, window_title='Sample
Images'):
    print(f"Displaying sample {label} images...")
    sample_window = Toplevel()
    sample_window.title(window_title)
    sample_window.state('zoomed')
    sample_frame = Frame(sample_window)
    sample_frame.pack(expand=True, fill=tk.BOTH)

```

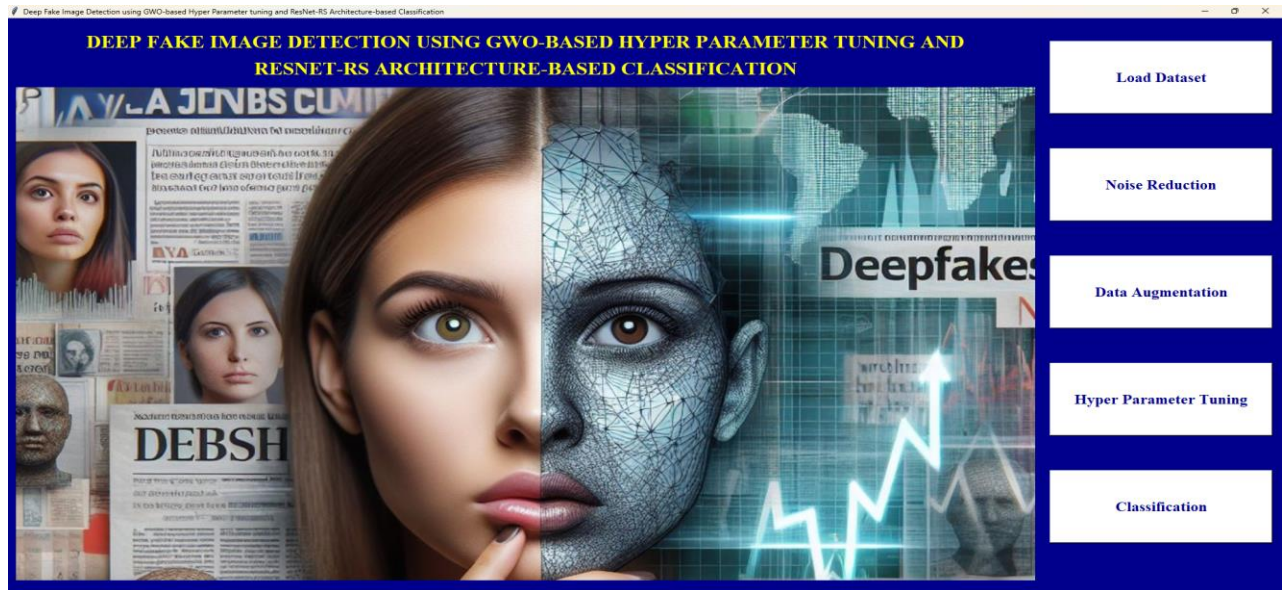
```

canvas = Canvas(sample_frame)
canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
scrollbar = Scrollbar(sample_frame, orient=tk.VERTICAL, command=canvas.yview)
scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
canvas.configure(yscrollcommand=scrollbar.set)
image_frame = Frame(canvas)
canvas.create_window((0, 0), window=image_frame, anchor="center")
title_label = Label(image_frame, text=window_title, font=("Times New Roman", 16, 'bold'))
title_label.grid(row=0, column=0, columnspan=10, pady=(10, 10))
label_dir = os.path.join(output_dir, label)
image_files = os.listdir(label_dir)[:sample_size]
if not image_files:
    print(f"No images found in {label_dir}.")
    return
images = []
for image_file in image_files:
    image_path = os.path.join(label_dir, image_file)
    img = Image.open(image_path)
    img.thumbnail((100, 100))
    img = ImageTk.PhotoImage(img)
    images.append(img)
for i, img in enumerate(images):
    row = (i // 10) + 1
    col = i % 10
    lbl = Label(image_frame, image=img)
    lbl.image = img
    lbl.grid(row=row, column=col, padx=45, pady=5)
image_frame.update_idletasks()
canvas.config(scrollregion=canvas.bbox("all"))
sample_window.mainloop()

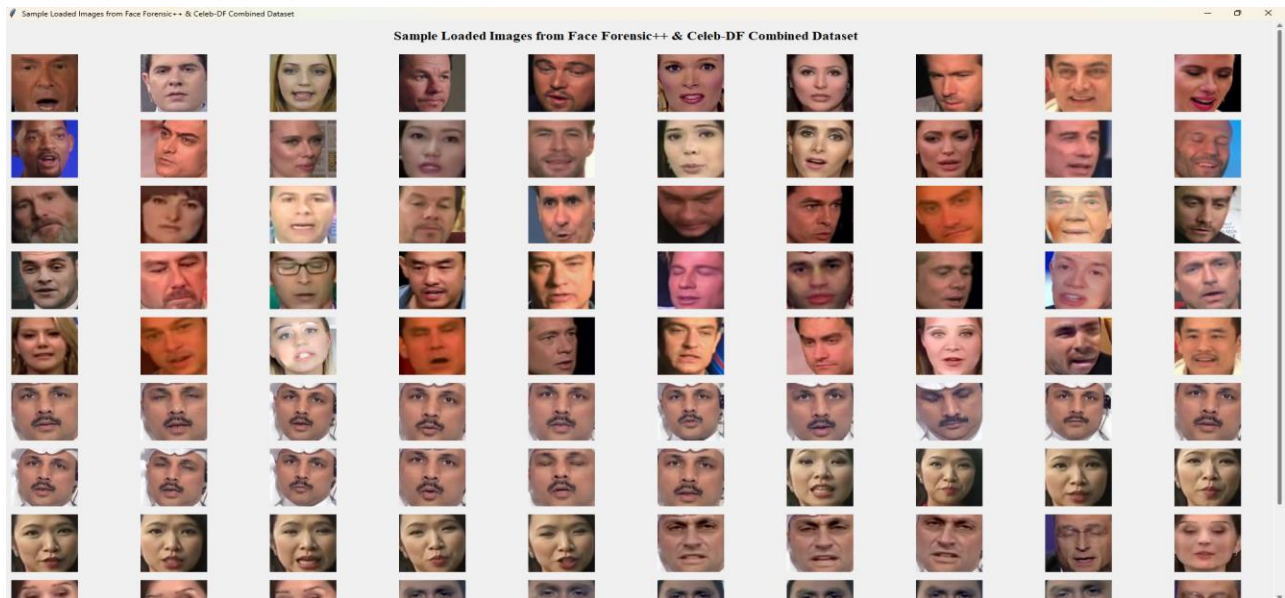
```

APPENDIX 2

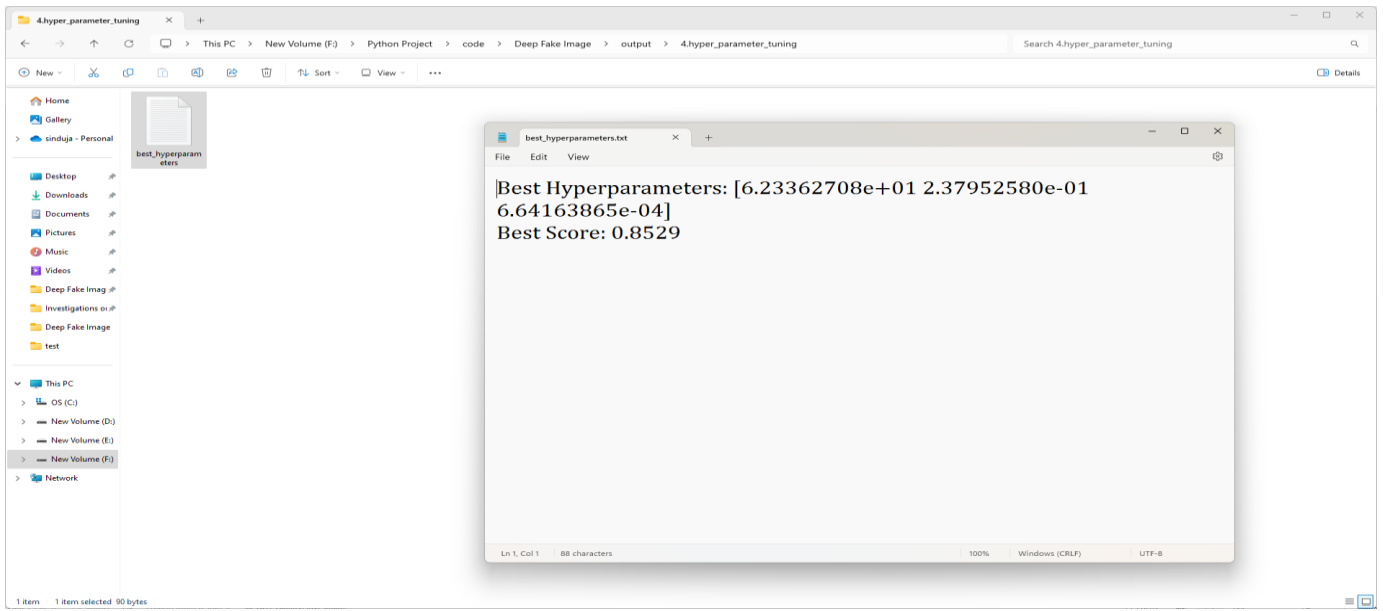
SCREENSHOTS



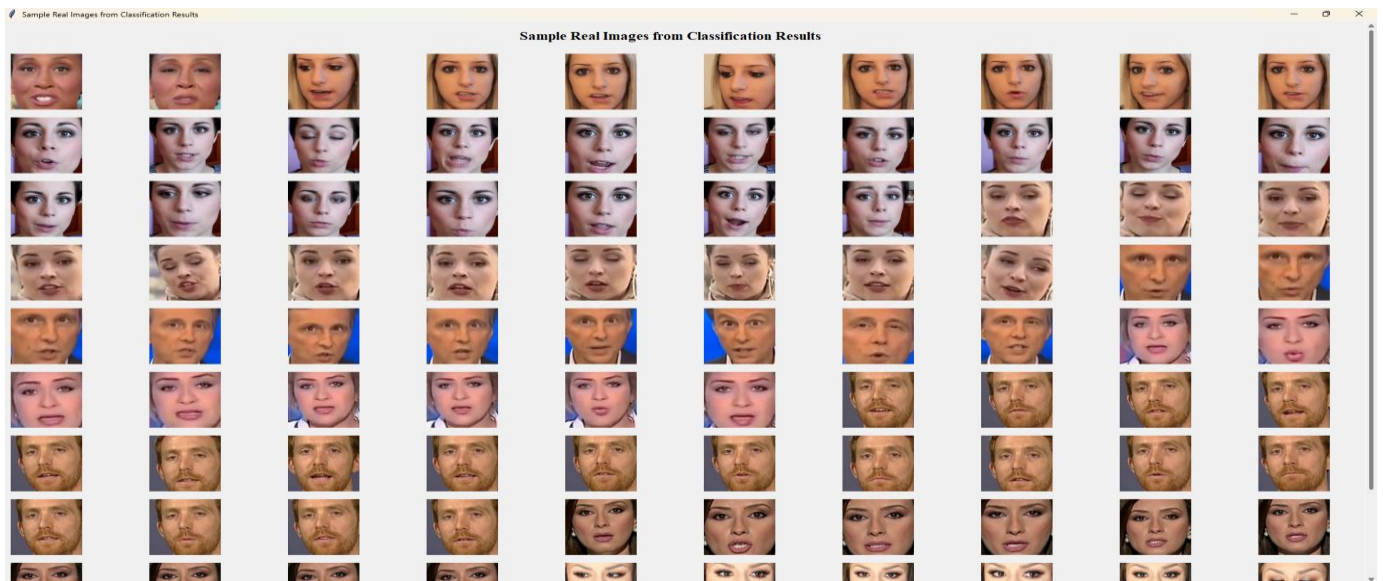
B.1 Home Page



B.2 Load Dataset



B.5 Hyper Parameter Tuning



B.6 Classification

REFERENCES

- [1] Jain, H. (2025). Digital Image Forgery Detection using Deep Learning. International Journal for Research in Applied Science and Engineering Technology.
- [2] Selva Birunda, S., Nagaraj, P., Krishna Narayanan, S., Muthamil sudar, K., Muneeswaran, V., & Ramana, R. (2022). Fake Image Detection in Twitter using Flood Fill Algorithm and Deep Neural Networks. 2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 285-290.
- [3] Sabah, H. (2022). Detection of Deep Fake in Face Images Using Deep Learning. Wasit Journal of Computer and Mathematics Science.
- [4] Rodríguez-Ortega, Y., DoraM.Ballesteros, L., & Renza, D. (2021). Copy-Move Forgery Detection (CMFD) Using Deep Learning for Image and Video Forensics. Journal of Imaging, 7.
- [5] Jasim, R.M., & Atia, T.S. (2023). An evolutionary- convolutional neural network for fake image detection. Indonesian Journal of Electrical Engineering and Computer Science.
- [6] St, S., Ayoobkhan, M.U., V, K.K., Bačanin, N., K, V., Štěpán, H., & Pavel, T. (2022). Deep learning model for deep fake face recognition and detection. PeerJ Computer Science, 8.
- [7] B, S., JagadeeshKannan, R., Prabu, P., Venkatachalam, K., & Trojovský, P. (2022). Deep fake detection using cascaded deep sparse auto-encoder for effective feature selection. PeerJ Computer Science, 8.
- [8] Jayaram, D., Gopalachari, M.V., Rakesh, S.T., Sai, J.S., & Kumar, G.K. (2022). Fake face image detection using feature network. International journal of health sciences.
- [9] Khalil, A.H., Ghalwash, A.Z., Elsayed, H., Salama, G.I., & Ghalwash, H.A. (2023). Enhancing Digital Image Forgery Detection Using Transfer Learning. IEEE Access, 11, 91583-91594.
- [10] Aparna, P., Pradeep, V., & Tech, M. (2023). Using a Concatenation of Low-Complexity Deep Learning Models, We Can Spot Fake Images.

- [11] Usha, M., & Pradeep, B. (2024). Deep fake video/image detection using deep learning. Global Journal of Engineering and Technology Advances.
- [12] Tiwari, S., Dixit, A.K., & Pandey, A.K. (2024). FAKE IMAGE DETECTION USING GENERATIVE ADVERSARIAL NETWORKS (GANS) AND DEEP LEARNING MODELS. Journal of Dynamics and Control.
- [13] Gururaj., A., Ajaai.N., M., Eswaran.J., M., & Swetlin.B, C. (2024). Deep Fake Image and Video Detection using Machine learning. International Journal of Advanced Research in Science, Communication and Technology.
- [14] Favorskaya, M.N., & Yakimchuk, A. (2021). Fake Face Image Detection Using Deep Learning-Based Local and Global Matching.
- [15] M, B.P., & Daniel, J.F. (2022). First Order Motion Model for Image Animation and Deep Fake Detection: Using Deep Learning. 2022 International Conference on Computer Communication and Informatics (ICCCI), 1-7.
- [16] Jagdale, A., Kubde, V., Kortikar, R., V. Mote, P.A., & Rajgure, P.N. (2024). DeepFake Image Detection: Fake Image Detection using CNNs and GANs Algorithm. INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT.
- [17] Sundaram, S., Jayaraman, S., & Somasundaram, K. (2024). Automated Near-Duplicate Image Detection using Sparrow Search Algorithm with Deep Learning Model. 2024 International Conference on Cognitive Robotics and Intelligent Systems (ICC - ROBINS), 245-251.
- [18] Pukale, P.D., Kulkarni, P.V., Bagwan, J., Jagadale, P., More, S., & Sarmokdam, R. (2024). Image Forgery Detection Using Deep Learning. International Research Journal on Advanced Engineering and Management (IRJAEM).
- [19] Kumar, B.N. (2023). Image Forgery Detection Using Deep Learning. International Journal for Research in Applied Science and Engineering Technology.

LIST OF PUBLICATIONS

Sujanathi S, Priya R, Ranga Shree S, Suruthika S and Sushmitha S, “**Comparison Analysis of Transfer Learning Models for Deep Fake Image Detection**”. 2nd International Conference on Machine Learning and Autonomous Systems (ICMLAS 2025), Stamford International University, Bangkok, Thailand (**Accepted for Publication in IEEE Explorer - SCOPUS, Indexed**).



Certificate of Presentation

This is to certify that

Priya R

has successfully presented a paper entitled

Comparison Analysis of Transfer Learning Models for Deep Fake Image Detection

at the

2nd International Conference on Machine Learning and Autonomous Systems
(ICMLAS 2025) organised by Stamford International University
Bangkok, Thailand on 10-12th March 2025.

Session Chair

Dr. Surekha Lanka
Conference Chair

Dr. Colin Arun Pinto
Dean



Certificate of Presentation

This is to certify that

Ranga Shree S

has successfully presented a paper entitled

Comparison Analysis of Transfer Learning Models for Deep Fake Image Detection

at the

2nd International Conference on Machine Learning and Autonomous Systems
(ICMLAS 2025) organised by Stamford International University
Bangkok, Thailand on 10-12th March 2025.

Session Chair

Dr. Surekha Lanka
Conference Chair

Dr. Colin Arun Pinto
Dean



Certificate of Presentation

This is to certify that

Suruthika S

has successfully presented a paper entitled

Comparison Analysis of Transfer Learning Models for Deep Fake Image Detection

at the

2nd International Conference on Machine Learning and Autonomous Systems
(ICMLAS 2025) organised by Stamford International University
Bangkok, Thailand on 10-12th March 2025.

Session Chair

Dr. Surekha Lanka
Conference Chair

Dr. Colin Arun Pinto
Dean



Certificate of Presentation

This is to certify that

Sushmitha S

has successfully presented a paper entitled

Comparison Analysis of Transfer Learning Models for Deep Fake Image Detection

at the

2nd International Conference on Machine Learning and Autonomous Systems
(ICMLAS 2025) organised by Stamford International University
Bangkok, Thailand on 10-12th March 2025.


Session Chair


Dr. Surekha Lanka
Conference Chair


Dr. Colin Arun Pinto
Dean