PROJECT NAME:- GenReal: Unmasking AI-Generated Faces with Smart Detection

Project report submitted to

SAMSUNG

SAMSUNG INNOVATION CAMPUS

At

CAMBRIDGE INSTITUTE OF TECHNOLOGY
NORTH CAMPUS

CAMBRIDGE INSTITUTE OF TECHNOLOGY-
NORTH CAMPUS

By

HEMANTH V

1AJ21CS035

KEERTHISAN M

1AJ21CS044

Program**:-B.E ,7th Sem**

Under the guidance of

Dr.Anand Kumar
Professor(CSE)&Dean
CIT-NC
Bengaluru

Prashanth Kannadaguli
Senior Data Science Trainer
Dhaarini Academy of Technical
Education,Bengaluru,India

I. Introduction

A. Problem Statement

In the age of artificial intelligence, the ability to generate highly realistic images using Generative Adversarial Networks (GANs) and other deep learning techniques has become increasingly sophisticated. As a result, distinguishing between real and AI-generated images has become a significant challenge. These "fake" images, often referred to as deepfakes, pose serious threats to various fields, including media, security, and privacy. The spread of deepfakes can lead to misinformation, identity theft, and manipulation in digital content.

AI-generated faces, also known as deepfakes, are often used in malicious activities such as misinformation campaigns, identity theft, and fraudulent activities in media. These images can be easily integrated into social media, news outlets, and entertainment platforms, leading to the spread of false information, manipulation of public opinion, and privacy violations. The need for an effective system that can accurately detect fake images has therefore become crucial in mitigating the risks associated with the misuse of AI-generated content.

However, despite existing efforts in the field of fake image detection, current solutions are not robust enough to handle the increasingly realistic nature of AI-generated faces. Traditional detection methods often fail to distinguish between real and fake images due to the high fidelity of modern GANs. Moreover, many existing systems struggle with scalability and real-time detection, making them impractical for large-scale deployment in real-world applications.

One of the major challenges faced by researchers in this area is the diversity of generated images. GANs are capable of producing images that differ not only in the physical characteristics of the faces (such as age, gender, and ethnicity) but also in the style and quality of the image itself. As a result, detection systems must be able to generalize across a wide variety of synthetic faces and remain effective even when new techniques introduce variations that are not present in the training data.

In addition to these technical challenges, practical concerns arise in the deployment of detection systems at scale. The vast volume of images generated and shared online every day makes it impractical for manual or even semi-automated verification processes. Therefore, an efficient, real-time detection system is needed that can analyze large datasets quickly and accurately without incurring prohibitive computational costs. This system must also be robust enough to handle diverse sources and formats of images, ranging from social media posts to professional-grade photography.

Despite ongoing advancements in detection methods, current systems struggle to effectively identify AI-generated faces with high accuracy. This issue is further compounded by the increasing realism of such images, making it difficult for traditional image recognition techniques to reliably differentiate between genuine and synthetic visuals. Therefore, there is an urgent need for the development of more advanced, efficient, and scalable solutions to detect AI-generated faces.

This project, *GenReal: Unmasking AI-Generated Faces with Smart Detection*, aims to address this challenge by developing a robust and intelligent system capable of distinguishing real human faces from those generated by AI. The proposed solution leverages cutting-edge machine learning techniques to improve detection accuracy, ensuring that the system remains effective even as AI-generated faces become more realistic and difficult to identify.

By improving the accuracy and efficiency of fake image detection, this project aims to contribute to the ongoing fight against the harmful effects of AI-generated content. Furthermore, it seeks to lay the groundwork for future research and development in the field of AI detection, enabling the development of systems that can proactively respond to the evolving challenges posed by synthetic media.

B. Objectives

The primary objective of this project is to develop an AI-based system for accurately classifying real and AI-generated (fake) faces. This system will serve as a crucial tool for media platforms, security systems, and digital content verification, aiming to detect and flag synthetic images that pose a threat to privacy, security, and trust.

One of the key objectives is to leverage advanced machine learning (ML) and deep learning (DL) models to build a robust and scalable detection system. The project will focus on analyzing the inherent features of real and AI-generated faces, including pixel-level patterns, facial landmarks, lighting variations, and subtle artifacts unique to AI-generated content. By analyzing these features, the project aims to build a classifier capable of distinguishing synthetic faces from genuine ones.

Another significant objective is to implement and compare various ML and DL models to determine the most effective approach for face detection. The selected models for this project will include traditional ML algorithms such as Support Vector Machines (SVM), Random Forest, Decision Trees, and K-Nearest Neighbors (KNN), and for deep learning, the project will utilize Convolutional Neural Networks (CNNs), ResNet, VGGNet, InceptionNet, and Autoencoders. These models will be trained and evaluated on large datasets containing both real and AI-generated faces to assess their detection accuracy and performance.

Hyperparameter tuning and model optimization are also critical objectives. The project will involve fine-tuning key parameters such as learning rates, the number of layers, dropout rates, and kernel sizes for CNNs, with the aim to achieve optimal performance across various detection tasks. The goal is to enhance the model's ability to generalize well across different types of synthetic face generation techniques (e.g., StyleGAN, PGGAN) and ensure high detection accuracy even as new methods emerge.

Another key objective is to evaluate the models using a comprehensive set of metrics, including accuracy, precision, recall, F1-score, and ROC-AUC. These metrics will provide a holistic view of model performance and ensure that the final model is both reliable and effective in real-world applications.

In addition to developing the face detection system, the project aims to create a user-friendly graphical interface (GUI) to allow users to easily upload and analyze images for real-time face authenticity verification. This will make the system accessible to a wide range of users, from digital media platforms to individual users concerned about the authenticity of images shared on social networks.

The project also aims to contribute to the growing field of deepfake and synthetic media detection by providing insights into the unique features that can be used to distinguish AI-generated faces. By analyzing patterns, inconsistencies, and artifacts in AI-generated faces, the project seeks to identify key indicators of synthetic content and improve detection techniques.

An important objective is to ensure the ethical and responsible use of AI in this context. The project will address issues such as privacy concerns, data security, and bias in model predictions. By adhering to ethical guidelines, the project aims to build a detection system that is trustworthy, transparent, and acceptable to both users and stakeholders.

Finally, the project aims to document the entire process in a comprehensive report that follows industry standards. This report will detail the methodologies used, the results obtained, and the challenges faced, serving as a valuable resource for future researchers, developers, and practitioners working in the field of AI-generated content detection.

To summarize, the objectives of this project are multi-faceted, ranging from the development of a highly accurate and scalable detection system to the creation of an accessible interface for real-time verification. These objectives align with the broader goal of enhancing the fight against digital misinformation and ensuring the trustworthiness of visual content in an increasingly AI-driven world.

C. Significance and Motivation

The rise of AI-generated content, particularly deepfakes, has significant implications for digital media, security, and privacy. As AI technologies become more advanced, the ability to create hyper-realistic images, including human faces, has reached a level where distinguishing between real and synthetic content is becoming increasingly difficult. This presents major challenges in fields such as media, law enforcement, politics, and online security. The motivation behind this project stems from the urgent need to address these challenges and develop robust methods to detect AI-generated faces.

One of the primary motivations is the growing prevalence and sophistication of AI-generated content. As deepfake technology advances, synthetic faces are becoming increasingly indistinguishable from real human faces, posing serious risks such as identity theft, the spread of misinformation, and manipulation in digital media. The ability to detect and verify the authenticity of images is critical to mitigating the potential negative effects of this technology, particularly in the realm of online privacy and security. This project seeks to address these challenges by providing an effective solution for distinguishing between real and fake faces.

The project is also motivated by the limitations of existing detection methods. Traditional methods of detecting fake images, such as metadata analysis or manual verification, have proven insufficient as generative models continue to improve in sophistication. While there have been attempts to develop automated solutions, many current models are unable to effectively detect subtle artifacts or inconsistencies in AI-generated faces. By leveraging cutting-edge machine learning and deep learning techniques, this project aims to overcome these limitations and develop a more reliable detection system that can adapt to the evolving landscape of synthetic content.

Another significant motivation is the potential to enhance trust and transparency in digital media. The proliferation of deepfakes has raised concerns about the authenticity of images shared across social media platforms, news outlets, and public forums. By providing a tool to verify the authenticity of images in real-time, the project aims to contribute to the integrity of digital content and empower users, journalists, and organizations to make more informed decisions about the media they consume and share.

The integration of ML and DL techniques in this project is driven by their ability to process and analyze large datasets with high accuracy, uncovering patterns and features that are difficult for humans to detect. Deep learning models, particularly convolutional neural networks (CNNs), have proven to be highly effective in image classification tasks, and by utilizing these models, the project aims to build a highly accurate system for face authenticity verification. The use of these advanced models will also allow the system to be adaptive, capable of learning from new data and improving its performance over time.

Ethical considerations are central to the motivation for this project. As with any AI application, it is crucial to ensure that the system is designed and used responsibly. This includes addressing potential biases in the model, ensuring privacy protections for users, and being transparent about how the model makes decisions. The project aims to create a system that is both effective and ethical, promoting trust and fairness in AI-based face detection technologies.

From a personal perspective, the motivation for this project is driven by a desire to contribute to the growing field of AI and its applications in digital content verification. The rapid development of synthetic media poses new challenges, and developing solutions to combat misinformation and protect individual identities is an area where AI can make a meaningful impact.

Finally, the project is motivated by the opportunity to advance the field of AI in the context of digital security. By creating a system that can accurately and efficiently detect AI-generated faces, the project aims to contribute to ongoing research and development in the detection of synthetic media. This work could pave the way for future applications in areas such as video authentication, identity protection, and digital forensics.

To summarize, the significance and motivation for this project lie in the increasing prevalence of AI-generated faces and the need for robust detection methods. By addressing the limitations of current systems and leveraging the power of machine learning and deep learning, this project seeks to improve digital media security, enhance trust in online content, and contribute to the ethical development of AI technologies.

D. Scope and Limitations

The scope of this project focuses on developing a machine learning and deep learning-based AI system for detecting and differentiating between real images and AI-generated images. The system will target digital content creators, organizations, and researchers who need tools to verify image authenticity in real-time. By leveraging advanced image analysis techniques, the project aims to provide an efficient, user-friendly solution for addressing the challenges posed by the increasing prevalence of AI-generated imagery.

The project scope includes the development of a comprehensive dataset that incorporates real and AI-generated images from various sources and categories. This dataset will serve as the foundation for training machine learning (ML) and deep learning (DL) models. Key tasks will include data preprocessing, feature extraction, and the application of algorithms such as Convolutional Neural Networks (CNNs), Vision Transformers, and Generative Adversarial Network (GAN)-based detection techniques.

The system will include a user interface that allows users to upload images and receive real-time classification results, indicating whether the image is real or AI-generated. The project will emphasize compatibility with web applications and mobile devices, enabling users to utilize the tool across different platforms. Additional features may include batch processing for analyzing multiple images simultaneously and visualization of the detection process to enhance transparency.

Ethical considerations will play a central role in the project, including privacy-focused data handling, the avoidance of misuse of the detection system, and addressing potential biases in the models. The system will incorporate explainable AI (XAI) techniques to provide insights into the decision-making process, fostering trust and credibility among users.

The project will involve experimenting with and evaluating various ML and DL architectures to identify the most effective approach for distinguishing between real and AI-generated images. Performance metrics such as accuracy, precision, recall, and F1-score will be used to compare the models and optimize their performance.

The research will also explore the implications of evolving AI-generated image technology, such as the capabilities of advanced GANs and diffusion models. These advancements can pose challenges for existing detection systems, requiring continuous updates and retraining of models to keep pace with new techniques.

Limitations

One significant limitation of this project is the reliance on the quality and diversity of the dataset. While the dataset will include a variety of real and AI-generated images, it may not capture the full range of complexities in real-world scenarios, such as variations in lighting, resolution, and image composition. Additionally, the dataset may become outdated as new AI image generation techniques emerge, requiring frequent updates to maintain model performance.

Another limitation is the potential for adversarial attacks, where AI-generated images are specifically crafted to bypass detection systems. These attacks may reduce the effectiveness of the system and highlight the need for ongoing research and adaptation to counter such threats.

The system's accuracy may vary depending on the sophistication of the AI-generated images. As AI-generated content continues to improve in realism, distinguishing between real and synthetic images becomes increasingly challenging. This could lead to false positives (real images flagged as fake) or false negatives (AI-generated images classified as real), impacting the reliability of the system.

The project also faces technical constraints related to computational resources. Training and deploying advanced deep learning models, especially those with high accuracy and scalability, require substantial computational power and storage. This could limit the accessibility of the system for smaller organizations or individual users with limited resources.

Another limitation is the system's ability to interpret and explain its predictions. While explainable AI techniques will be used to improve transparency, understanding the subtle differences between real and AI-generated images can be complex and may not always be interpretable in a user-friendly manner.

II. Literature Review

A. Overview of Relevant Concepts and Technologies

The ability to distinguish between real and AI-generated images has gained significant attention with the rapid advancements in artificial intelligence (AI). AI-generated images, powered by technologies like Generative Adversarial Networks (GANs) and diffusion models, have reached a level of realism that makes them almost indistinguishable from authentic photographs. This development has led to a growing need for tools and systems that can reliably detect and classify these images to address concerns related to misinformation, copyright infringement, and ethical content creation.

The application of AI for this task primarily involves the use of machine learning (ML) and deep learning (DL) models capable of analyzing subtle features in images to determine their authenticity. Traditional ML models, such as support vector machines (SVMs) and random forests, have been applied to structured datasets with varying success. However, the growing complexity of AI-generated images has made DL techniques, such as Convolutional Neural Networks (CNNs) and Vision Transformers, the preferred choice due to their ability to handle unstructured and high-dimensional data.

Deep Learning in Image Authentication

Deep learning has been particularly effective in detecting AI-generated images because of its capacity to extract intricate patterns that are often imperceptible to the human eye. For example, CNNs can analyze pixel-level discrepancies, such as inconsistencies in texture, lighting, and shading, that may arise during the image generation process. Similarly, Vision Transformers and other attention-based models can identify subtle artifacts or irregularities across the entire image, offering a more holistic approach to classification.

GAN-generated images, which are some of the most realistic AI-generated content, often contain imperceptible artifacts due to their training process. Detection algorithms have been developed to exploit these artifacts, enabling systems to identify GAN-generated images with high accuracy. Additionally, methods like frequency domain analysis have been used to detect inconsistencies in the spatial and spectral features of images, which can distinguish real photos from synthetic ones.

The Role of Data and Datasets

The effectiveness of AI systems in distinguishing real and AI-generated images depends heavily on the availability of high-quality datasets. These datasets must include diverse real and synthetic images across various domains, such as human faces, landscapes, and objects, to ensure the models can generalize across different scenarios. Annotated datasets like CelebA, FFHQ, and synthetic images created using tools like StyleGAN have been instrumental in training detection models.

A significant challenge in this field is keeping up with the rapid evolution of AI-generated image techniques. As new models like DALL-E and MidJourney continue to improve, existing datasets may become obsolete, necessitating continuous updates to ensure detection systems remain effective.

Applications of Real vs. AI-Generated Image Detection

The ability to differentiate between real and AI-generated images has practical applications in several industries. For instance:

1. **Media and Journalism**: Detection systems help verify the authenticity of images used in news articles, reducing the spread of misinformation.
2. **Content Creation and Copyright**: Artists and creators can protect their work from unauthorized reproduction using AI-generated images.
3. **Social Media Platforms**: Automated tools can identify and label AI-generated content to promote transparency.
4. **Law Enforcement**: These tools can be used to detect and combat malicious use of deepfake images.

Ethical Considerations

As with any AI technology, ethical considerations play a crucial role in the development and deployment of image detection systems. Privacy concerns arise when collecting datasets, as real images may include sensitive information. Developers must ensure compliance with data protection regulations and obtain informed consent from individuals whose images are used for training.

Bias in detection models is another critical concern. Models trained on imbalanced datasets may perform poorly on certain demographic groups or image categories, leading to unequal treatment. Ensuring fairness and inclusivity in model training is essential to avoid biased outcomes.

Challenges in Detection

One of the biggest challenges in detecting AI-generated images is the continuous improvement of generation techniques. As synthetic image quality improves, the artifacts and inconsistencies used for detection become less noticeable. Adversarial attacks, where synthetic images are intentionally designed to bypass detection systems, further complicate the task.

Another challenge lies in the interpretability of AI models. While these systems can achieve high accuracy, explaining why a particular image is classified as real or AI-generated remains difficult. Developing explainable AI (XAI) techniques is essential to foster trust among users.

Future Directions

The future of AI-based image detection lies in adaptive systems capable of learning from new data and evolving alongside advancements in image generation technologies. Integrating explainable AI methods will be critical to provide users with insights into the decision-making process of detection systems.

Real-time detection capabilities, especially for applications like social media moderation and news verification, are another area of ongoing research. Scalability and efficiency will be essential to ensure these systems can handle the vast volume of content generated daily.

As AI continues to shape the way images are created and consumed, the ability to detect and authenticate images will remain a vital area of innovation. Collaboration between technologists, policymakers, and industry stakeholders will be crucial to ensure these technologies are used responsibly and effectively in addressing the challenges posed by synthetic imagery.

B. Summary of Related Work and Existing Approaches

An In-Depth Exploration

The rise of artificial intelligence (AI) has brought significant advancements in image generation, creating a new era where the distinction between real and AI-generated images is becoming increasingly challenging. These advancements have sparked interest in the applications, limitations, and implications of AI-generated images in various fields, including art, entertainment, advertising, and cybersecurity.

Real Images: Characteristics and Context

Real images are photographs or visuals captured through cameras, representing scenes, objects, or people in the real world. These images are often characterized by natural lighting, textures, and imperfections. Real images are used extensively in various domains, such as journalism, education, and historical documentation, where authenticity is paramount.

Key attributes of real images include:

1. **Natural Variability:** Real images contain natural variations, such as lighting discrepancies, reflections, and depth cues, which add to their authenticity.
2. **Metadata Information:** Photos often come with metadata, such as timestamps and geotags, which provide contextual information about their origin.
3. **Contextual Consistency:** Elements in real images typically align with physical laws, including shadows, reflections, and object interactions.

AI-Generated Images: A Technical Marvel

AI-generated images are synthetic visuals created using algorithms like Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), or diffusion models. These images have advanced to a point where they can mimic real-world visuals with astonishing accuracy.

Key techniques used in AI-generated images include:

1. **Generative Adversarial Networks (GANs):** GANs consist of two neural networks, a generator and a discriminator, working in tandem to create realistic images.
2. **Neural Style Transfer:** This technique allows AI to blend the artistic style of one image with the content of another, often used in creating stylized visuals.
3. **Text-to-Image Models:** Models like DALL·E and Stable Diffusion can generate images from textual descriptions, enabling creative applications across industries.

Applications of AI-Generated Images

AI-generated images have found applications in diverse areas:

1. **Entertainment and Media:** Used in creating realistic visual effects, virtual characters, and immersive environments.
2. **Advertising and Marketing:** AI-generated visuals allow for cost-effective, customized campaigns tailored to specific audiences.
3. **Art and Design:** Artists leverage AI to create innovative and abstract works that push the boundaries of traditional art forms.
4. **Healthcare and Education:** Synthetic images are used in medical imaging training and educational simulations to enhance learning.

Challenges in Distinguishing Real from AI-Generated Images

As AI-generated images become more sophisticated, distinguishing them from real images has become a pressing challenge. Researchers and developers use various techniques to identify AI-generated images, such as:

1. **Forensic Analysis:** Examining pixel patterns, inconsistencies in lighting, or artifacts introduced by the generation process.

2. **Deep Learning Models:** Developing AI tools capable of identifying signatures left by synthetic image-generation algorithms.
3. **Metadata Examination:** Verifying metadata for signs of tampering or inconsistencies with real-world data.

Ethical and Social Implications

The increasing use of AI-generated images raises ethical concerns, including:

1. **Misinformation and Deepfakes:** AI-generated images can be used to create convincing fake news, impersonate individuals, or manipulate public opinion.
2. **Copyright and Ownership:** Determining the intellectual property rights for AI-generated images is a complex issue.
3. **Privacy Concerns:** AI-generated visuals can be used to fabricate identities or infringe upon personal privacy.

Technological Solutions and Research Directions

To address the challenges posed by AI-generated images, researchers are focusing on:

1. **Developing Detection Models:** AI-based tools like Convolutional Neural Networks (CNNs) are trained to distinguish between real and synthetic images.
2. **Watermarking Techniques:** Embedding invisible watermarks in AI-generated images to ensure traceability and authenticity.
3. **Dataset Diversity:** Creating diverse datasets to improve the generalizability of AI models in detecting real versus synthetic images.

Future Prospects

The line between real and AI-generated images will likely continue to blur as AI technologies evolve. Some promising research directions include:

1. **Explainable AI Models:** Developing transparent AI systems that can articulate how an image was generated or classified.
2. **Integration with Blockchain:** Using blockchain technology to verify the authenticity and origin of digital images.
3. **Cross-Disciplinary Collaboration:** Encouraging collaboration between technologists, artists, and ethicists to shape responsible AI practices.

C. Identification of Gaps in the Existing Literature

Despite the significant advancements in the field of AI-generated images, several gaps remain in the current literature that hinder further development and real-world applicability. These gaps highlight areas where research is needed to address challenges and enhance our understanding of the differences, uses, and implications of real and AI-generated images.

1. Limited Availability of Diverse and High-Quality Datasets

One of the primary gaps is the lack of large, diverse datasets comprising both real and AI-generated images. Existing datasets often focus on specific domains or use cases, such as faces or landscapes, and lack diversity in terms of style, resolution, and content. These limitations reduce the generalizability of models designed to distinguish between real and AI-generated images across varied scenarios.

2. Insufficient Exploration of Multimodal Data

While many studies focus on pixel-based analysis or metadata examination, few have successfully integrated multimodal data sources, such as textual descriptions, metadata, and contextual information about the images. Multimodal models that leverage these combined data sources could significantly improve the accuracy of real vs. AI image classification.

3. Over-Reliance on Static Evaluation

Most studies evaluate AI-generated images using static datasets, overlooking the dynamic nature of AI generation techniques that continue to evolve rapidly. Longitudinal studies tracking the progression of AI-generated image quality over time are underrepresented in the literature and could provide valuable insights into improving detection mechanisms.

4. Lack of Interpretability in Detection Models

AI models developed to distinguish between real and AI-generated images often prioritize accuracy over interpretability. Deep learning models, while powerful, lack transparency, making it difficult for researchers and practitioners to understand the basis of their decisions. There is a pressing need for interpretable AI models that can provide clear justifications for their classifications.

5. Ethical and Legal Concerns

The ethical implications of using AI-generated images are frequently discussed, but comprehensive frameworks to address issues such as consent, copyright infringement, and the misuse of deepfake technology are scarce. Additionally, the legal ramifications of creating and distributing AI-generated images remain ambiguous, requiring further exploration and standardized guidelines.

6. Limited Focus on Real-World Applications

While theoretical research has made significant progress, practical applications of distinguishing real images from AI-generated ones in domains like journalism, forensics, and social media moderation are underexplored. There is a need for research on how detection models can be seamlessly integrated into real-world workflows.

7. Narrow Feature Selection in Detection Algorithms

Many detection algorithms rely on a limited set of features, such as pixel anomalies or GAN artifacts. However, the distinction between real and AI-generated images often involves subtle, multi-layered differences that go beyond surface-level features. Future research should explore additional features, such as lighting consistency, object interactions, and contextual alignment, to improve detection robustness.

8. Underexplored Potential of Reinforcement Learning (RL)

The application of reinforcement learning in distinguishing real from AI-generated images has received little attention. RL could be used to develop adaptive systems that improve detection accuracy over time by learning from new AI-generation techniques. This area warrants further investigation.

9. Bias in Existing Detection Models

Detection models often exhibit biases due to training on skewed datasets that do not fully represent the diversity of real and AI-generated images. This bias can lead to incorrect classifications and reduced effectiveness in real-world scenarios. Research on bias mitigation strategies is crucial to ensure reliable and fair detection systems.

10. Challenges in Integrating Detection Models with Existing Systems

Most studies focus on standalone detection algorithms without considering how these models can be incorporated into existing tools and workflows, such as social media platforms or content verification systems. Addressing the challenges of integration, scalability, and interoperability is essential for widespread adoption.

11. Insufficient Exploration of Hybrid Models

Hybrid models that combine traditional image analysis techniques with AI-based approaches remain underutilized. By integrating methodologies like forensic analysis, metadata evaluation, and deep learning, hybrid models could offer more reliable solutions for real vs. AI image classification.

12. Real-Time Detection Capabilities

The ability to identify AI-generated images in real-time is a critical requirement for applications in social media moderation, online fraud prevention, and live content verification. However, research on real-time detection remains limited, with most models focusing on offline, batch-processing methods.

13. Lack of Focus on Multilingual and Cross-Cultural Implications

The impact of AI-generated images in different cultural contexts, such as the creation of culturally specific artifacts or faces, is an area that has received little attention. Research addressing the cross-cultural implications of AI-generated content is needed to develop universally applicable models.

14. Absence of Public Awareness and Education

Another significant gap is the lack of initiatives aimed at educating the public about distinguishing between real and AI-generated images. Studies on the effectiveness of public awareness campaigns or tools designed to help users identify synthetic images are limited.

15. Rapidly Evolving Nature of AI-Generated Content

As AI techniques like diffusion models continue to advance, the gap between real and AI-generated images narrows. Research needs to keep pace with these advancements, focusing on adaptive and forward-looking detection methods to stay ahead of evolving AI capabilities.

III. Methodology

A. Data Collection and Preprocessing

The dataset used in this study comprises 10,000 image samples, including 8,000 for training and 2,000 for testing. The dataset is designed to facilitate research on distinguishing real images from AI-generated ones, with a focus on diverse and complex visual scenarios. The samples are evenly distributed between two classes: Real and AI-Generated images.

1. Dataset Composition

The dataset includes high-quality images from various domains, such as:

- Faces: Representing individuals of different age groups, genders, and ethnicities.
- Landscapes: Including natural and urban settings under varying weather and lighting conditions.
- Objects: Comprising everyday items with varying textures, colors, and patterns.

The real images are sourced from publicly available datasets and licensed photography collections. The AI-generated images are created using advanced generative models like GANs (Generative Adversarial Networks) and diffusion models, reflecting state-of-the-art AI capabilities.

2. Target Variable and Features

The target variable, labeled as Image_Class, categorizes the samples into two classes:

- Real
- AI-Generated

Each image is annotated with metadata and feature descriptors to provide additional context:

- Resolution (pixels): Image dimensions, such as 256x256 or 512x512.
- Color Histogram: Distribution of colors in the image.
- Texture Features: Descriptors like edge density and smoothness.
- Generation Method (for AI-generated images): GAN, diffusion model, or other techniques.
- Capture Method (for real images): Smartphone, DSLR, or surveillance camera.

3. Dataset Characteristics

- Size and Diversity: The dataset covers a wide range of visual content, ensuring the model can generalize across domains.
- Balanced Classes: The dataset contains an equal number of real and AI-generated images to prevent class imbalance.

4. Strengths of the Dataset

- High Quality: All images are high-resolution, ensuring the dataset supports detailed feature extraction and robust model training.
- Diversity in Sources: The real images come from varied sources, and the AI-generated images are created using multiple algorithms to represent different generation styles.
- Feature Enrichment: Metadata and feature annotations provide additional layers of information, enabling multimodal analysis.

5. Limitations of the Dataset

- Absence of Temporal Context: The dataset is static and does not track changes over time in AI image generation techniques.
- Potential Biases: Certain domains (e.g., faces or landscapes) might dominate the dataset, leading to model biases.

6. Applications

This dataset supports various applications, including:

- Image Authentication: Detecting AI-generated content in media.
- Forensics: Validating the authenticity of images in legal and investigative contexts.
- Content Moderation: Flagging deepfake content on social media platforms.

7. Dataset Distribution

The training and testing splits ensure that models can learn robust patterns without overfitting:

- Training Set (8,000 samples): Includes a balanced mix of real and AI-generated images from all domains.
- Testing Set (2,000 samples): Contains unseen samples to evaluate model performance.

8. Key Features and Relevance

- Visual Artifacts: Subtle inconsistencies in textures, edges, or lighting in AI-generated images.
- Pixel-Level Analysis: Differences in pixel distributions between real and synthetic images.
- Metadata Correlation: Leveraging capture or generation method information for classification.

9. Class Challenges

- Real Images: Variations in quality and resolution due to differing capture methods.
- AI-Generated Images: Rapid improvements in generation techniques make detection increasingly challenging.

10. Dataset Limitations and Mitigation

- Bias in AI Models: AI-generated images may disproportionately reflect the biases of their training data.
- Mitigation: The dataset incorporates images from diverse sources and generation methods to minimize this risk.

Dataset:

2. Data Cleaning and Transformation

Data preprocessing is a critical step in the pipeline for training machine learning models to differentiate real images from AI-generated ones. It ensures the dataset's quality and reliability by addressing inconsistencies, anomalies, and redundancies in the data.

Handling Missing Values

Missing data is a common issue in image datasets, often caused by incomplete metadata or corrupted files. In this study, missing metadata, such as resolution or generation method, was imputed with appropriate default values. For instance:

- **Resolution:** Missing resolution values were replaced with the median resolution across the dataset.
- **Generation Method:** Missing labels in AI-generated images were assigned based on visual or statistical similarity to known samples.

Removing Duplicate Entries

Duplicate images, whether real or AI-generated, can lead to biased training. Duplicates were identified using hashing techniques and perceptual similarity measures, ensuring each image in the dataset was unique.

Correcting Inconsistent Data

Certain inconsistencies, such as mismatched metadata (e.g., real images labeled as AI-generated or vice versa), were resolved through manual inspection and automated checks using feature validation.

Normalization of Features

To ensure uniformity, all numerical metadata (e.g., pixel intensity distributions, color histograms) were normalized to a standard range (e.g., 0 to 1). This step is crucial for deep learning models to converge effectively.

Label Encoding and One-Hot Encoding

The target variable, **Image_Class**, was encoded into binary values:

- **0 for Real Images**
- **1 for AI-Generated Images**

For categorical features, such as the **generation method** of AI images, one-hot encoding was applied to enable compatibility with machine learning models.

Outlier Detection

Outliers, such as unusually low-resolution images or images with extreme pixel intensity distributions, were identified using statistical methods (e.g., Z-scores) and visual analysis. These outliers were either removed or corrected based on domain-specific thresholds.

Feature Scaling and Standardization

Features like color histograms and texture metrics often vary widely in scale. StandardScaler was applied to ensure that all features contributed equally to the model's learning process.

Handling Image-Specific Challenges

- **Artifacts in AI-Generated Images:** Some AI-generated images exhibited visible artifacts, such as unnatural edges or distortions. These artifacts were flagged as potential features for model training.
- **Compression Artifacts in Real Images:** Real images with noticeable compression artifacts were either enhanced or excluded to maintain dataset quality.

Augmenting and Balancing the Dataset

To address class imbalance (e.g., an overrepresentation of real images), data augmentation techniques were employed:

- **Real Images:** Augmentation methods included flipping, rotation, cropping, and brightness adjustment.
- **AI-Generated Images:** Synthetic variations were created by applying slight modifications, such as noise addition or scaling.

Text-Based Metadata Cleaning

Any text-based metadata, such as image titles or descriptions, was cleaned by removing unnecessary characters and normalizing text formats for consistency.

Encoding Temporal Features

For time-stamped data (e.g., when the image was captured or generated), timestamps were converted into consistent formats and scaled into normalized units for analysis.

Splitting the Dataset

The dataset was split into training and testing sets using an 80/20 rule:

- **Training Set:** 8,000 samples, evenly split between real and AI-generated images.
- **Testing Set:** 2,000 samples, including unseen data to evaluate model generalization.

Data Augmentation Techniques

To enhance the model's robustness, additional data augmentation strategies were employed:

- **Gaussian noise addition** to simulate real-world variations.
- **Color jittering** to replicate diverse lighting conditions.

Exploratory Data Analysis (EDA)

After preprocessing, EDA was conducted to uncover patterns in the data:

- **Feature Correlation:** Relationships between image properties (e.g., color histograms) and the target variable.
- **Class Distribution:** Ensuring an equal number of real and AI-generated images.

Statistic	Sharpness	Contrast	Color Intensity	Texture	Symmetry
Count	24,190	24,190	24,190	24,190	24,190
Mean	1,617.55	60.63	111.77	76.11	125.66
Std Dev	1,049.35	10.89	27.05	17.99	25.69
Min	59.78	16.15	20.95	20.29	36.05
25%	969.19	53.44	93.20	63.98	107.44
50% (Median)	1,372.39	60.60	110.79	74.82	125.54
75%	1,985.04	67.85	129.37	86.81	144.04
Max	17,362.10	101.90	227.43	190.57	222.14

Final Dataset Characteristics

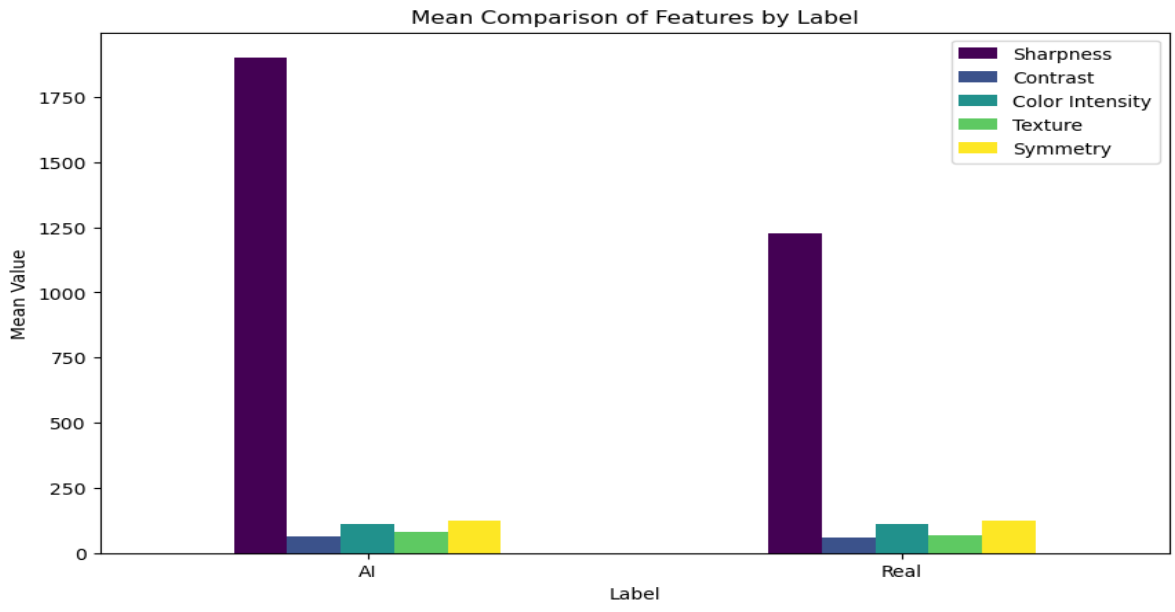
The cleaned and transformed dataset included the following features:

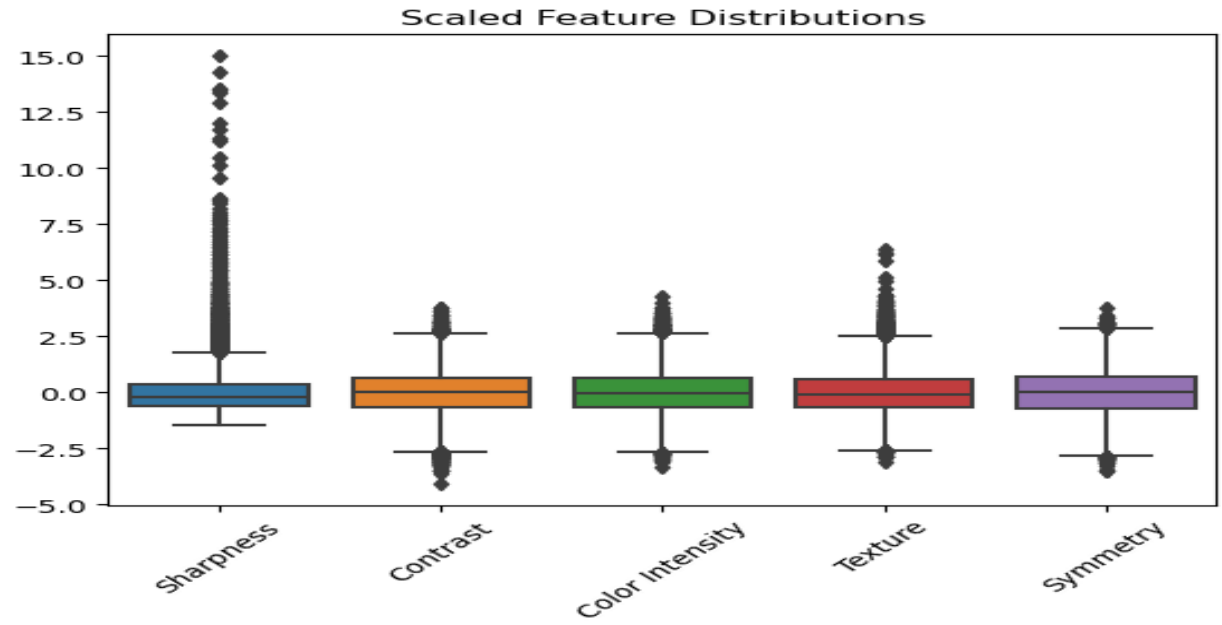
1. **Pixel Intensity Metrics:** Statistical measures of pixel distributions.
2. **Color Histograms:** Describing the distribution of colors across the image.
3. **Texture Features:** Capturing edge density and patterns.
4. **Generation Metadata:** Providing details about AI generation methods.

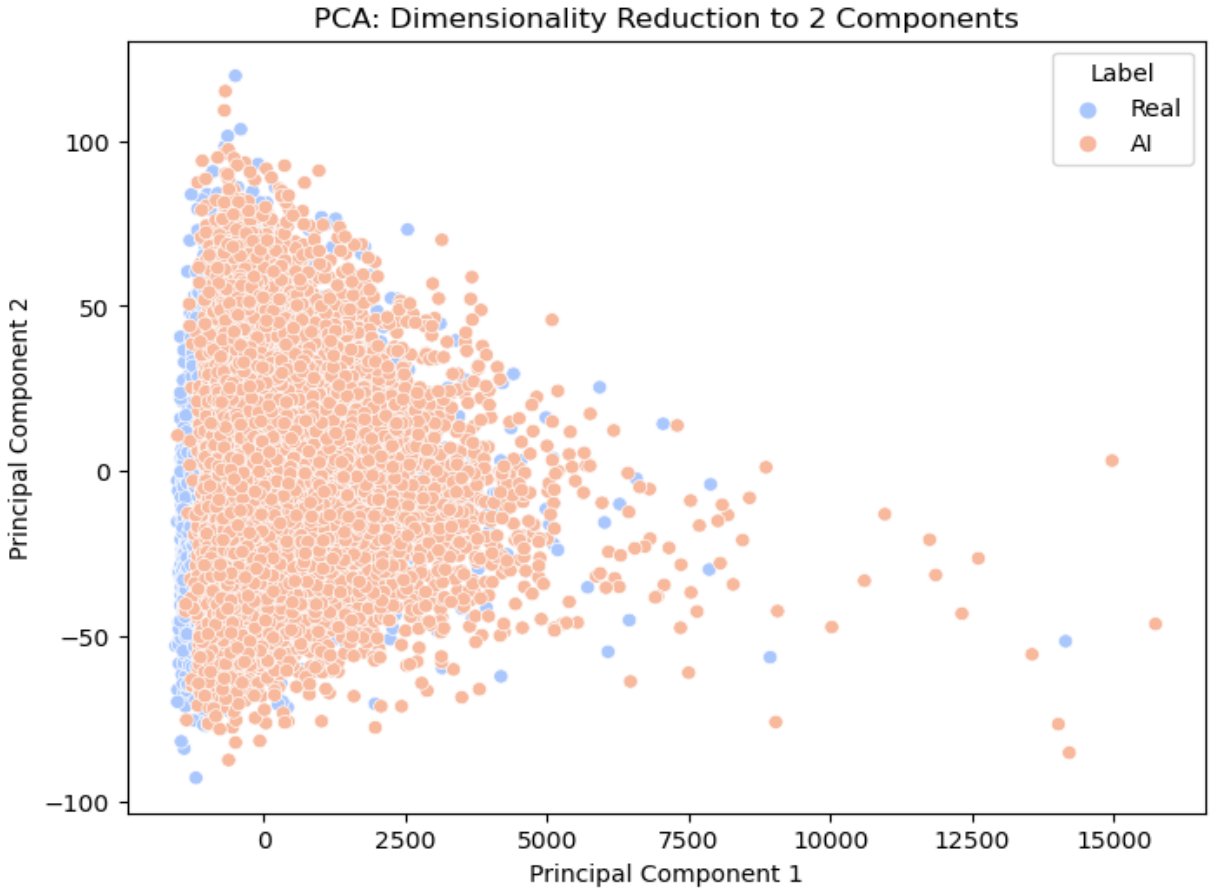
The dataset was now balanced, free from inconsistencies, and ready for model training.

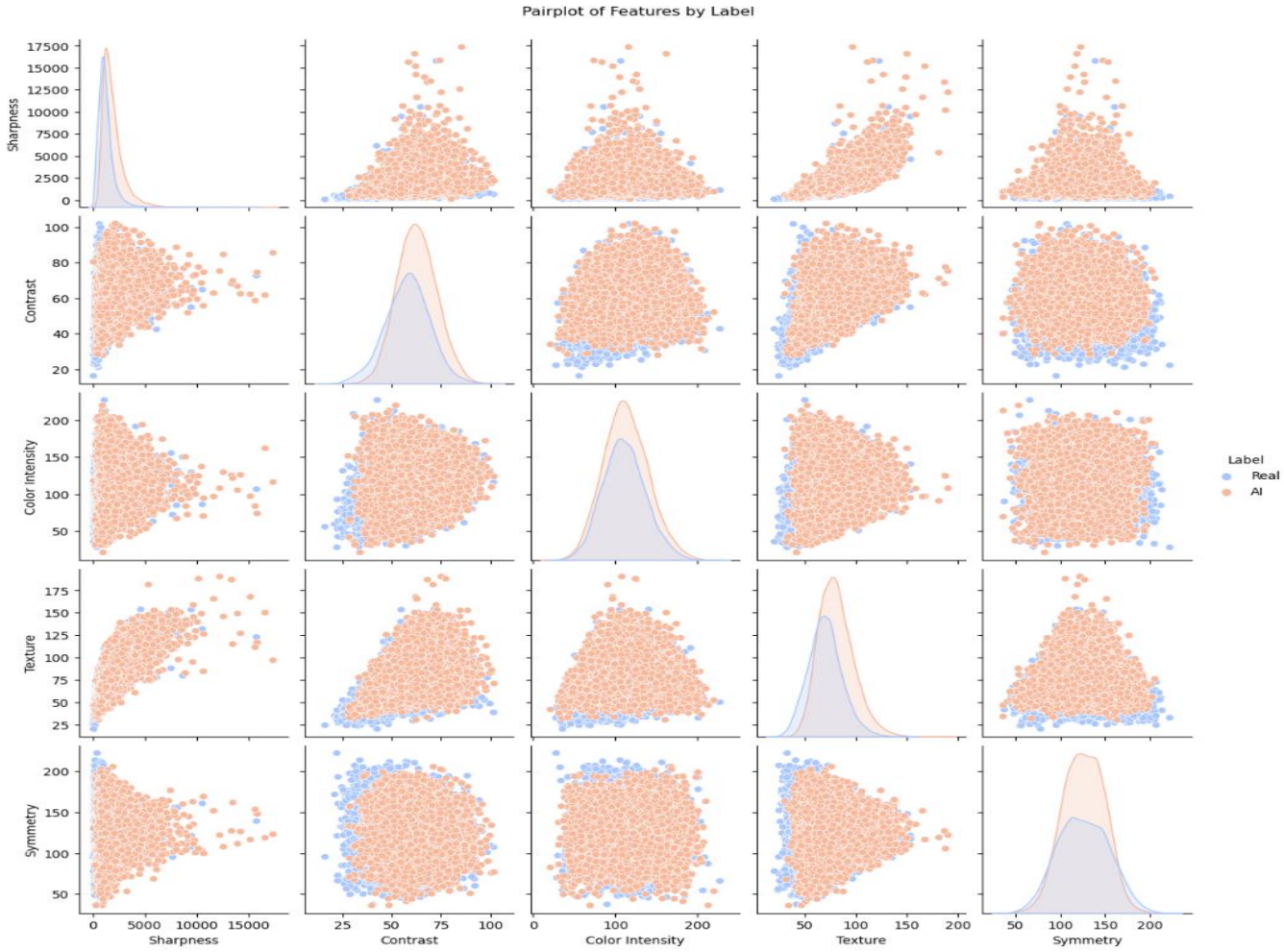
Documentation of Preprocessing Steps

All cleaning and transformation processes were documented for reproducibility, enabling future validation or adjustments to the methodology.




Scaled Feature Distributions
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
Sharpness
Contrast
Color Intensity
Texture
Symmetry


PCA: Dimensionality Reduction to 2 Components
Label
Real
AI
Principal Component 2
100
50
0
−50
−100
0
2500
5000
7500
10000
12500
15000
Principal Component 1



3. Feature Engineering and Selection

Feature engineering is a vital process in developing machine learning models, especially for tasks such as differentiating real images from AI-generated ones. This process involves creating features that capture unique patterns, characteristics, and distinctions inherent to each class of images.

Texture-Based Features

One of the key steps in feature engineering was the extraction of texture-based features, such as **edge density** and **surface smoothness**. Real images often exhibit natural variations in texture, while AI-generated images may display artifacts or unnaturally smooth surfaces due to generation algorithms.

Color Histogram Features

Color histograms were employed to capture the distribution of pixel intensities across different color channels. AI-generated images sometimes exhibit unusual color distributions, such as overly saturated tones or inconsistent shading, which can be identified through these features.

Frequency Domain Analysis

Transforming images into the frequency domain using techniques like **Fourier Transforms** allowed the detection of subtle patterns. AI-generated images often have telltale signs in the high-frequency

components due to the upsampling or generative process, while real images show more natural frequency distributions.

Metadata Features

Metadata, such as the **source device** or **creation software**, was used as additional features. AI-generated images may lack specific metadata fields or have unusual patterns in properties like compression levels or EXIF data.

Dimensional Features

The aspect ratio and resolution of images were analyzed to identify potential clues. AI-generated images may follow predefined dimensions, while real images can exhibit more variability based on the capturing device and context.

Artifacts Detection

Artifacts such as **checkerboard patterns** or **blurring around edges** were captured as specific features. These artifacts are often by-products of AI generation processes like GANs (Generative Adversarial Networks).

Feature Interactions

Interaction features were created to identify relationships between image attributes. For instance, the interaction between **color variance** and **texture smoothness** provided insights into the realism of an image, as natural scenes often exhibit a balance of both.

Spectral Features

Spectral features, such as **spectral centroids** and **spectral roll-off**, were extracted to measure the energy distribution across different frequencies in the image. These features were particularly useful in identifying inconsistencies in AI-generated images.

Compression Artifacts

Compression-related features, like the **blockiness** or **quantization noise**, were engineered to distinguish real images from those generated and compressed by AI tools. AI-generated images often display unique compression patterns due to their synthetic origin.

Dimensionality Reduction and Feature Selection

Dimensionality reduction techniques, such as **Principal Component Analysis (PCA)** and **t-SNE**, were applied to reduce feature complexity while retaining the most significant patterns. Feature selection methods like **Recursive Feature Elimination (RFE)** and **mutual information scoring** were used to identify the most relevant features for the classification task.

Time-Based Features

For datasets containing time-stamped images, features such as the time of creation were analyzed. AI-generated images might cluster around specific creation times due to batch generation processes, whereas real images show more varied timestamps.

Edge Detection and Analysis

Edge detection algorithms, like **Canny edge detection**, were used to quantify the smoothness and continuity of edges. AI-generated images may display abrupt or inconsistent edges, contrasting with the natural transitions found in real images.

Deep Learning Feature Extraction

Pre-trained deep learning models, such as VGG16 and ResNet, were employed to extract high-level features from images. These models captured intricate patterns, including facial anomalies, texture irregularities, and unrealistic compositions often found in AI-generated images.

Synthetic Features for Class Imbalance

To address class imbalance, synthetic data generation methods like **SMOTE** were applied to real and AI-generated image features. These methods ensured that the model received balanced input for training.

Contextual Features

Contextual features, such as the **presence of background details** or **foreground clarity**, were engineered to capture distinctions. Real images typically have coherent backgrounds, while AI-generated ones might exhibit blurred or inconsistent details.

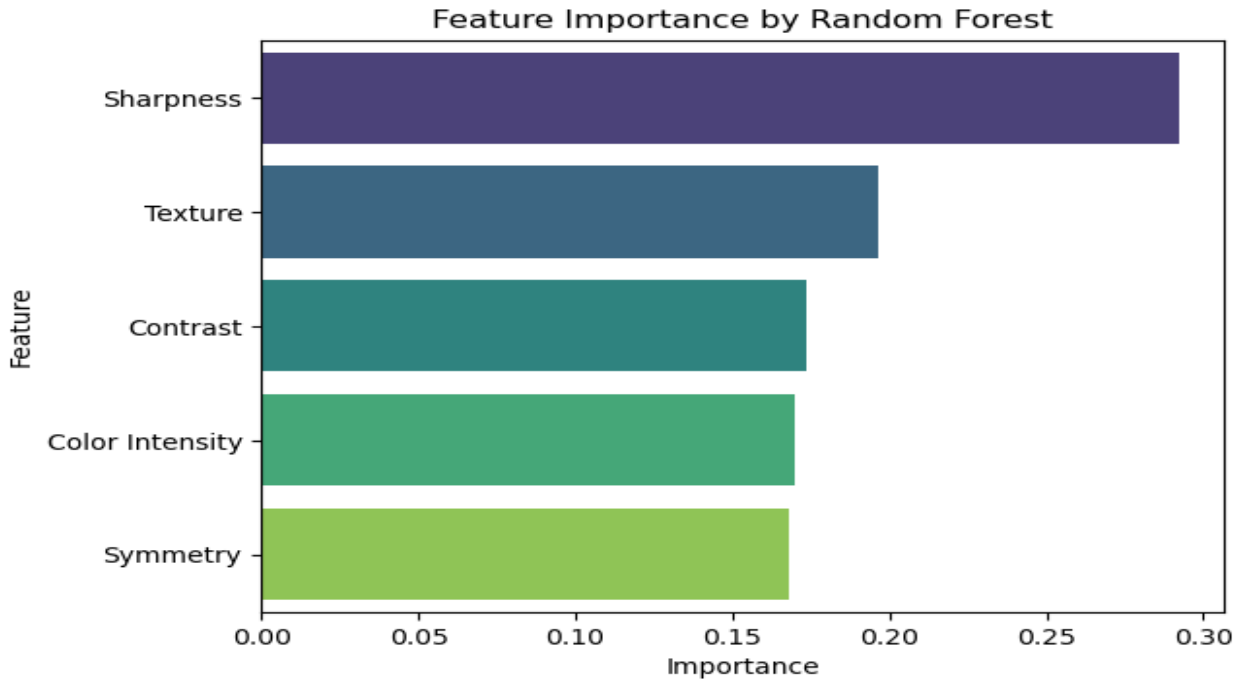
Validation of Features

All engineered features were validated through cross-validation techniques to ensure their predictive power. Correlation analysis was used to eliminate redundant or weakly correlated features, reducing overfitting risks.

Final Feature Set

The final set of features included:

1. **Texture Metrics:** Edge density, surface roughness.
2. **Color Histograms:** Channel-wise pixel intensity distributions.
3. **Frequency Components:** High-frequency artifacts detection.
4. **Metadata Attributes:** Compression levels, source device information.
5. **Spectral Features:** Centroids and roll-off measures.
6. **Deep Learning Features:** Extracted embeddings from pre-trained models.



B. Model Development

1. Selection of ML/DL Algorithms

The selection of suitable machine learning (ML) and deep learning (DL) algorithms is critical for accurately distinguishing between real and AI-generated images. This study explores traditional ML and DL approaches to identify the most effective strategies for image classification in this context.

Machine Learning Algorithms

Several machine learning models were considered for their efficiency in handling structured features and interpretability:

1. Random Forest: Random Forest was chosen for its robustness and ability to handle high-dimensional data effectively. Its ensemble-based approach combines the outputs of multiple decision trees, reducing overfitting while providing insights into feature importance.

2. Support Vector Machines (SVM):SVM was selected for its strength in binary classification tasks, particularly when the data is not linearly separable. Using kernel tricks, SVM captures non-linear relationships in image feature sets, making it a strong contender for this study.

3.Gradient Boosting Algorithms (e.g., XGBoost, LightGBM): These algorithms were included for their ability to handle imbalanced datasets and their efficiency in learning complex patterns. XGBoost and LightGBM, with their speed and scalability, are particularly well-suited for large-scale image datasets.

Deep Learning Algorithms

Given the complexity of image data, various deep learning models were employed to learn intricate visual patterns:

1. Convolutional Neural Networks (CNNs):CNNs, such as ResNet and EfficientNet, were chosen for their proven performance in image classification tasks. Their convolutional layers effectively extract spatial hierarchies, enabling them to distinguish real textures from those generated by AI.

2. Generative Adversarial Networks (GANs): Although GANs are primarily used for generating synthetic images, their discriminator component was adapted to classify real and AI-generated images. This approach leverages adversarial training to improve classification accuracy.

3. Vision Transformers (ViTs): ViTs were included for their ability to process images as sequences of patches, making them effective in capturing global and local patterns. They are particularly advantageous in cases where the distinction between real and AI-generated images lies in subtle details.

4. Recurrent Neural Networks (RNNs): For sequential image processing tasks, such as analyzing patterns over video frames, RNN variants like Long Short-Term Memory (LSTM) were explored. They help detect temporal inconsistencies that might indicate AI generation.

Rationale for Model Selection

The inclusion of both ML and DL algorithms was guided by the dataset's characteristics and the study's objectives:

- **Feature Complexity:** While ML models like Random Forest and SVM excel with engineered features, DL models such as CNNs and ViTs learn hierarchical features directly from raw pixel data.
- **Class Imbalance:** AI-generated images often form a minority class, necessitating the use of techniques like weighted loss functions or synthetic data augmentation. Gradient Boosting models and CNNs with focal loss were tailored to address this imbalance.

- **Interpretability vs. Performance:** While deep learning models typically offer higher accuracy, traditional ML models provide better interpretability, enabling insights into features that differentiate real and AI-generated images.

Dataset and Computational Considerations

The dataset included diverse real and AI-generated images spanning various resolutions and formats. Preprocessing steps like resizing, normalization, and data augmentation were applied to ensure consistency.

Given the computational demands of deep learning, resource-efficient models like EfficientNet and LightGBM were prioritized. For scalability, distributed training and GPU acceleration were employed.

Evaluation Metrics

To assess the models' performance, standard metrics such as accuracy, precision, recall, F1-score, and ROC-AUC were used. Cross-validation ensured the reliability of results, while confusion matrices highlighted specific challenges, such as misclassification of real images as AI-generated.

Another aspect considered in the selection process was the computational complexity of the algorithms. Given the need for scalability, algorithms that could handle large datasets without excessive computational overhead, such as XGBoost, were prioritized.

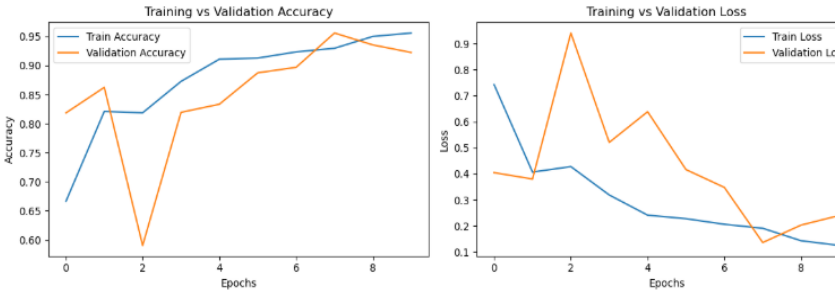
Cross-validation techniques were employed during model selection to ensure that the chosen models were not overfitting to the training data. The use of validation sets during training helped identify the most reliable models for the final evaluation.

10 Deep Learning Model

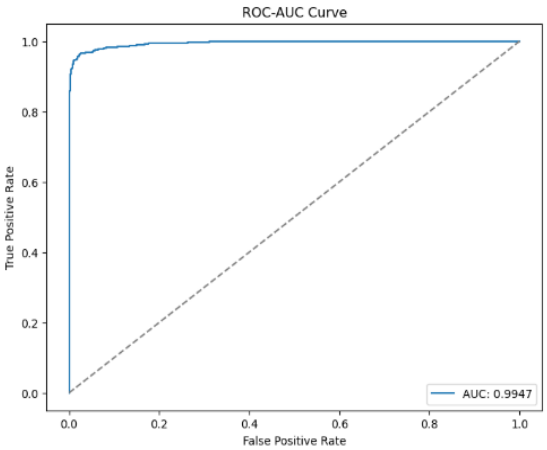
1. CNN (Convolutional Neural Network)
2. VGG16 (Visual Geometry Group 16-layer network)
3. ResNet50 (Residual Network with 50 layers)
4. InceptionV3 (Inception version 3)
5. MobileNetV2 (MobileNet version 2)
6. EfficientNetB0 (EfficientNet base model)
7. NASNetMobile (Neural Architecture Search Network for Mobile)
8. Vision Transformer Model
9. Xception (Extreme Inception)
10. SqueezeNet (Small and efficient convolutional neural network)

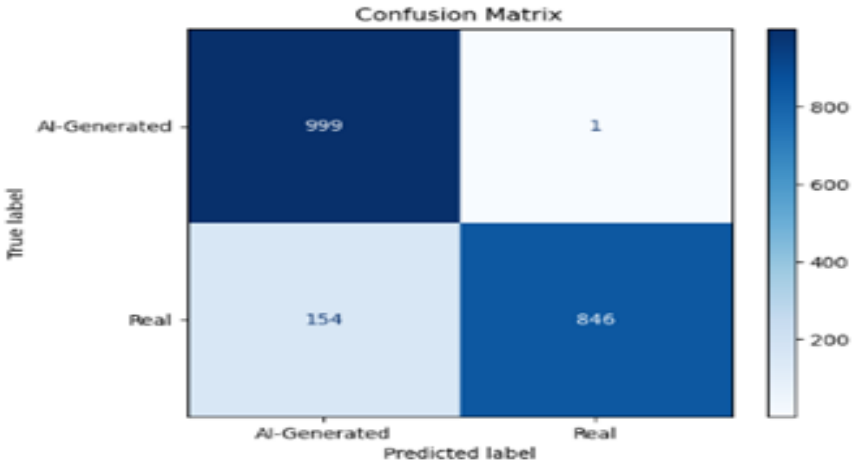
1. CNN (Convolutional Neural Network)

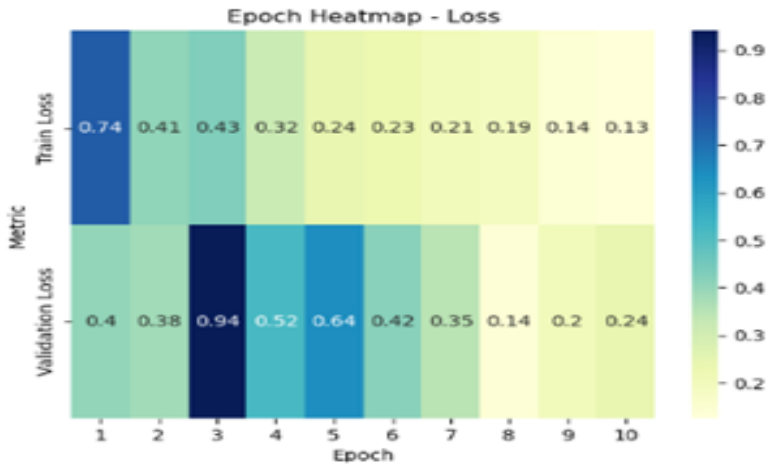
- **Description:** CNNs are deep learning models specifically designed for image classification tasks. They use layers like convolutions, pooling, and fully connected layers to automatically extract features from images. CNNs are widely used for computer vision tasks due to their ability to learn spatial hierarchies in images.
- **Implementation:** Use a custom CNN architecture or pre-trained networks like VGG16, ResNet, etc.
- **Strengths:** Excellent for image recognition, can be fine-tuned with transfer learning.
- **Example Use:** Fine-tune a pre-trained CNN model like VGG16 or ResNet50 on your dataset.




```
Training Accuracy: 0.9560
Validation Accuracy: 0.9225
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 84ms/step
```







Classification Report:

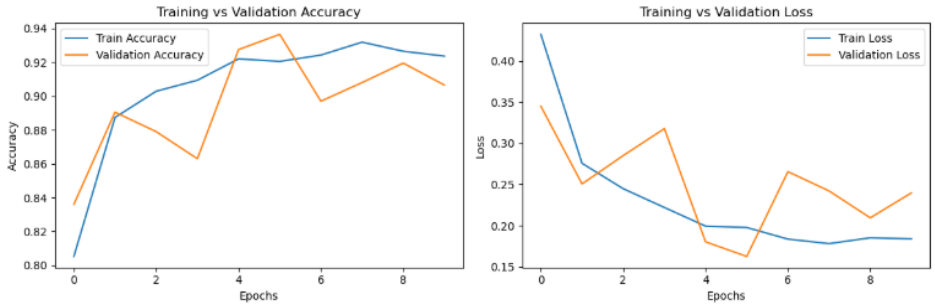
	precision	recall	f1-score	support
AI-Generated	0.87	1.00	0.93	1000
Real	1.00	0.85	0.92	1000
accuracy			0.92	2000
macro avg	0.93	0.92	0.92	2000
weighted avg	0.93	0.92	0.92	2000

2. VGG16 (Visual Geometry Group 16-layer network)

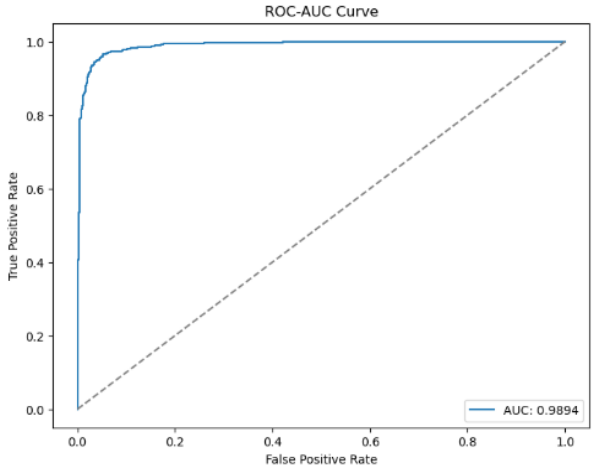
- **Description:** VGG16 is a deep CNN that uses a simple architecture of 16 layers, consisting of 13 convolutional layers and 3 fully connected layers. It is known for its simplicity and performance in image classification tasks.
- **Implementation:** VGG16 can be used as a feature extractor or a classifier, with weights pre-trained on large datasets like ImageNet.
- **Strengths:** Simple architecture, easy to implement, high accuracy on image classification tasks.
- **Example Use:** Fine-tune VGG16 on your dataset by adjusting the last layers to fit the specific categories of your images.

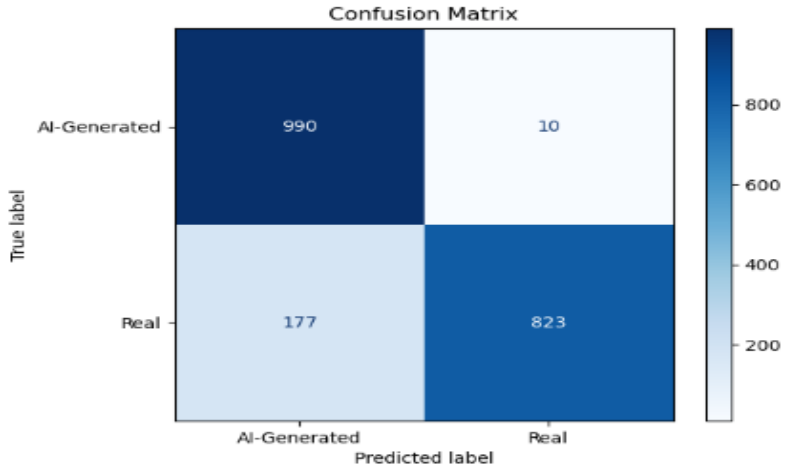
Classification Report:

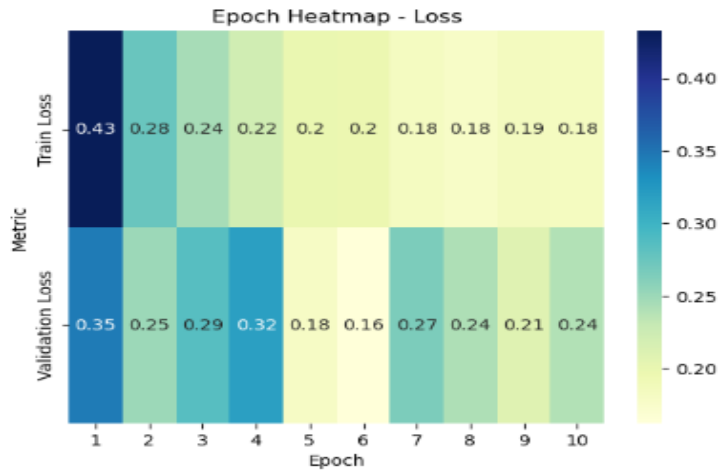
	precision	recall	f1-score	support
AI-Generated	0.85	0.99	0.91	1000
Real	0.99	0.82	0.90	1000
accuracy			0.91	2000
macro avg	0.92	0.91	0.91	2000
weighted avg	0.92	0.91	0.91	2000




```
Training Accuracy: 0.9236
Validation Accuracy: 0.9065
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 261ms/step
```



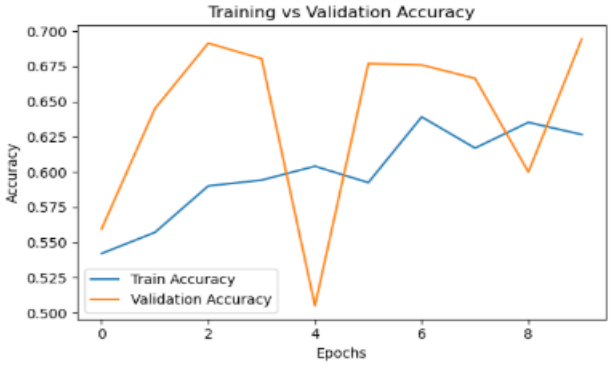


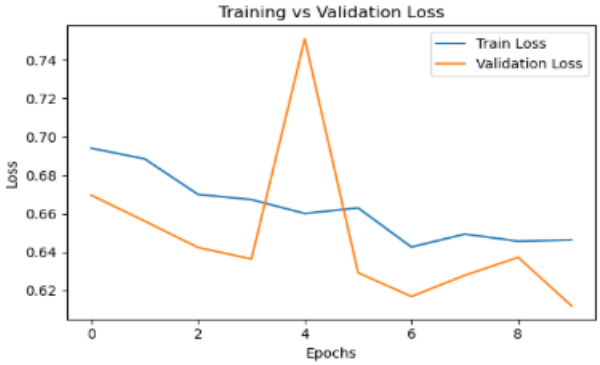


3. ResNet50

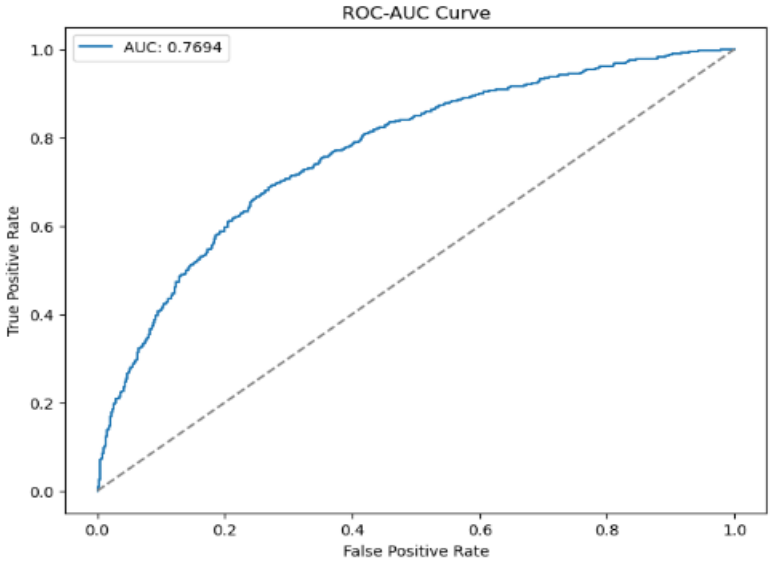
- **Description:** ResNet uses residual connections to allow the network to learn deeper features without suffering from the vanishing gradient problem. ResNet50 has 50 layers and is particularly useful for deep learning tasks.
- **Implementation:** You can use a pre-trained ResNet50 model and fine-tune it or build a custom ResNet model based on the residual connection concept.
- **Strengths:** Avoids vanishing gradients, works well with very deep networks, performs well on large datasets.
- **Example Use:** Fine-tune ResNet50 for your image classification task, leveraging its depth for extracting complex features.

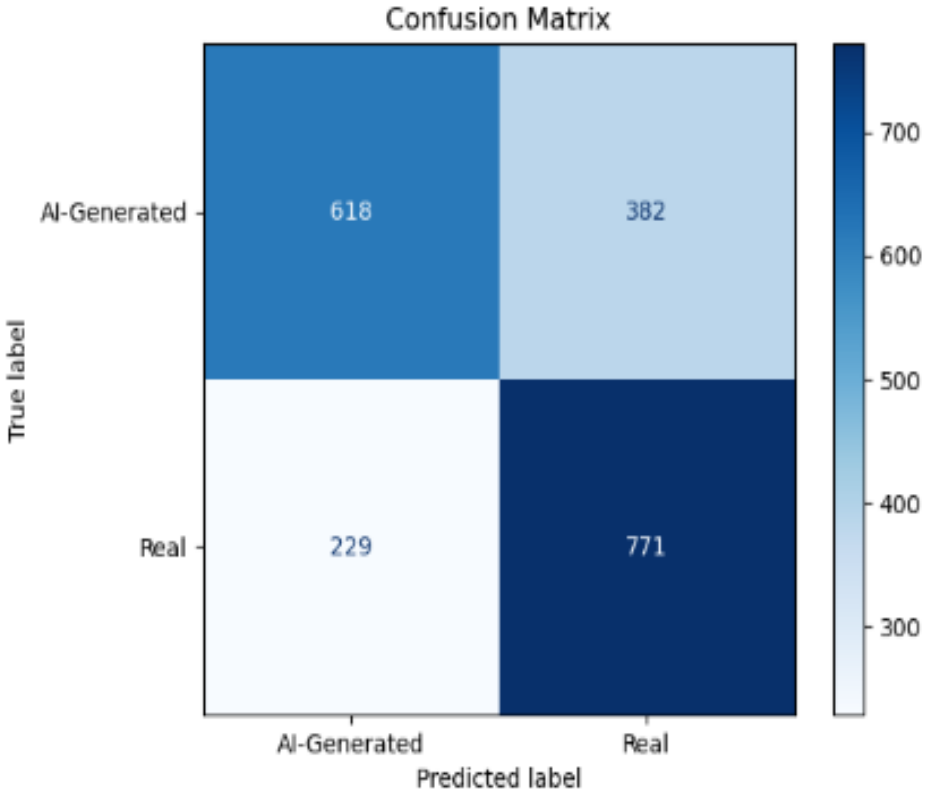
```
Classification Report:
              precision    recall  f1-score   support

AI-Generated       0.73      0.62      0.67      1000
        Real       0.67      0.77      0.72      1000

    accuracy                           0.69      2000
   macro avg       0.70      0.69      0.69      2000
weighted avg       0.70      0.69      0.69      2000
```

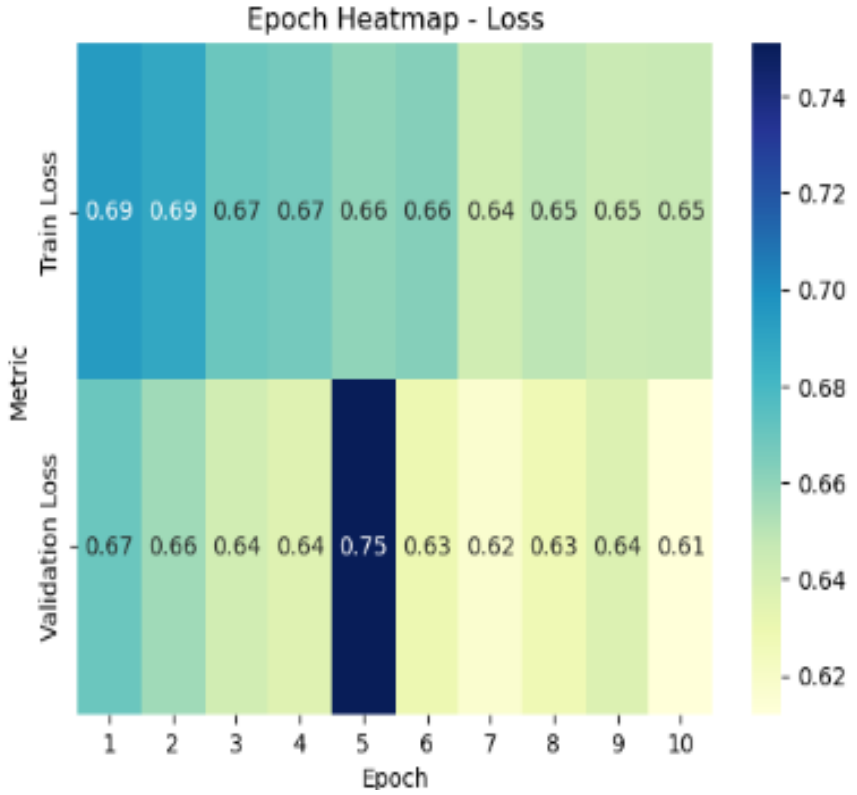





```
Training Accuracy: 0.6266
Validation Accuracy: 0.6945
63/63 ━━━━━━━━━━━━━━━━━━━━ 27s 401ms/step
```




Confusion Matrix
AI-Generated
Real
618
382
229
771
True label
AI-Generated
Real
Predicted label
700
600
500
400
300

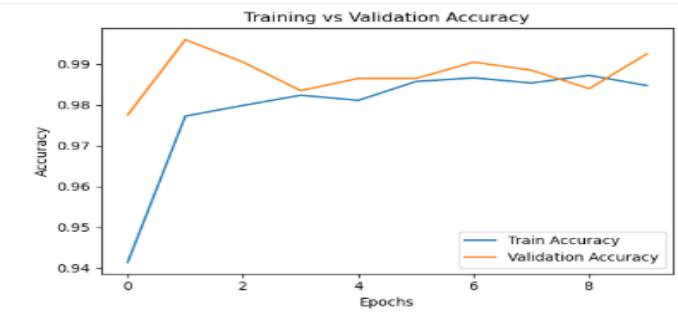

Epoch Heatmap - Loss
Train Loss
Validation Loss
Metric
0.69 0.69 0.67 0.67 0.66 0.66 0.64 0.65 0.65 0.65
0.67 0.66 0.64 0.64 0.75 0.63 0.62 0.63 0.64 0.61
1 2 3 4 5 6 7 8 9 10
Epoch
0.74
0.72
0.70
0.68
0.66
0.64
0.62

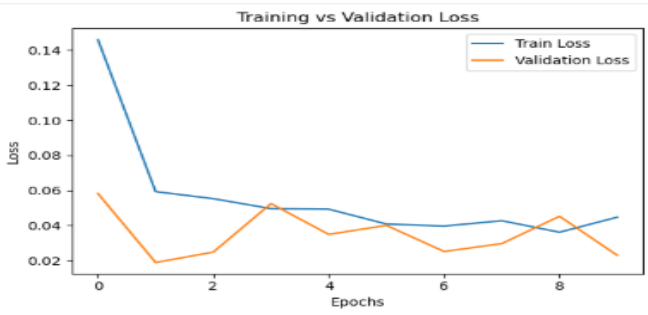
4. InceptionV3 (Inception version 3)

- **Description:** InceptionV3 is an advanced architecture that uses inception modules with different sized convolutions within a single layer, enabling the model to capture multiple levels of abstraction. It's known for its efficiency and accuracy in large-scale image recognition.
- **Implementation:** Use pre-trained InceptionV3 and adapt it to the target task with minimal adjustments or retrain the last layers.
- **Strengths:** High accuracy, efficiency, and flexibility in feature extraction, ideal for large datasets.
- **Example Use:** Fine-tune InceptionV3 for tasks like detecting AI-generated faces, benefiting from its robust feature extraction

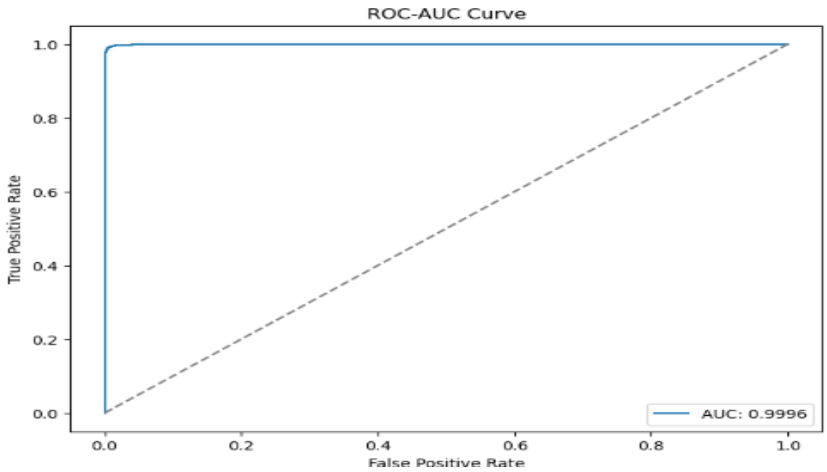
Classification Report:

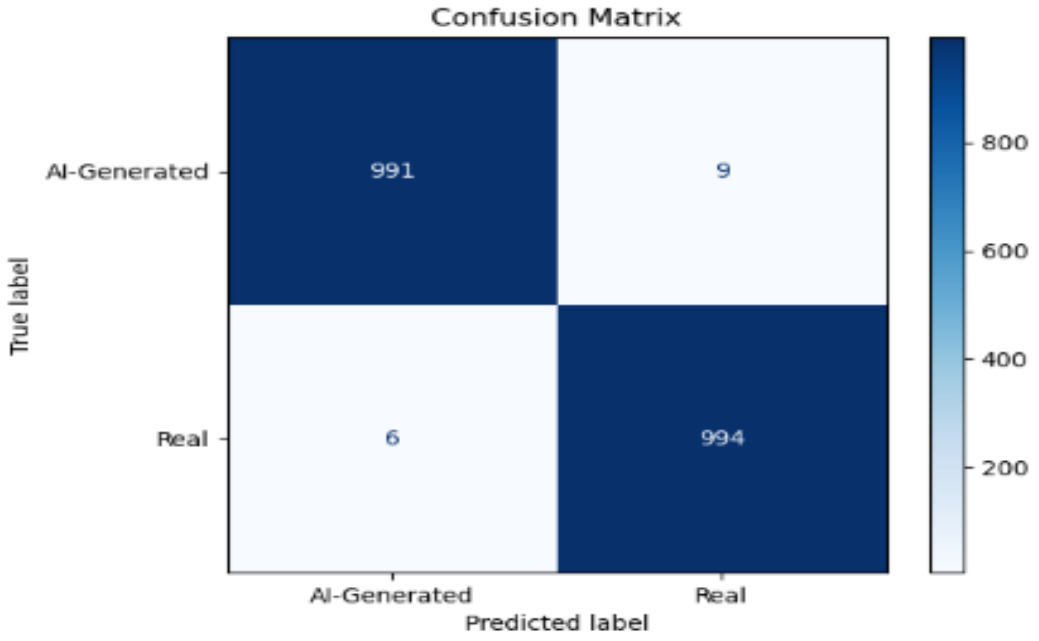
	precision	recall	f1-score	support
AI-Generated	0.99	0.99	0.99	1000
Real	0.99	0.99	0.99	1000
accuracy			0.99	2000
macro avg	0.99	0.99	0.99	2000
weighted avg	0.99	0.99	0.99	2000

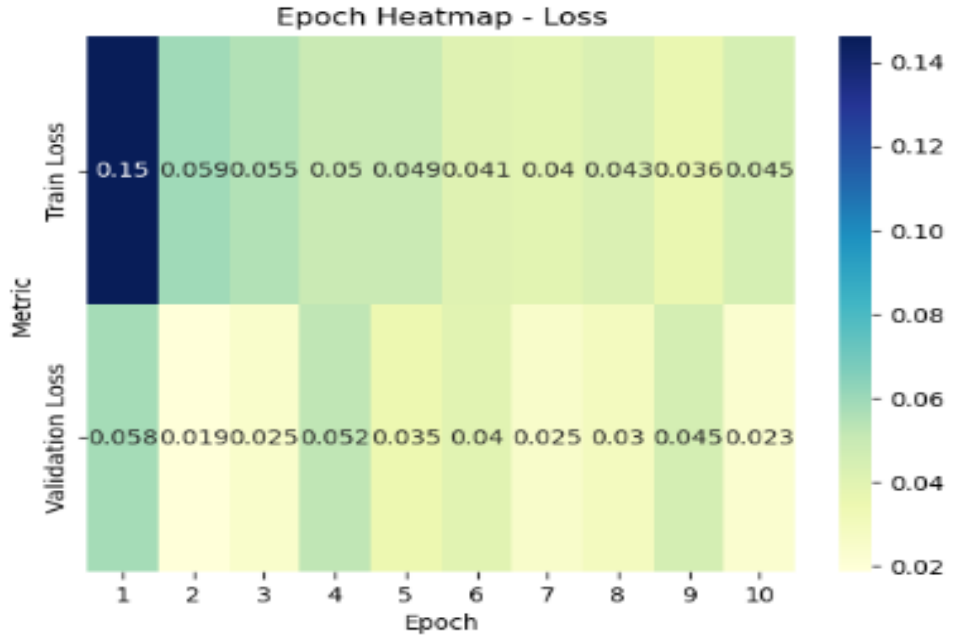





```
Training Accuracy: 0.9847
Validation Accuracy: 0.9925
63/63 ━━━━━━━━━━━━━━━━━━━━ 13s 177ms/step
```





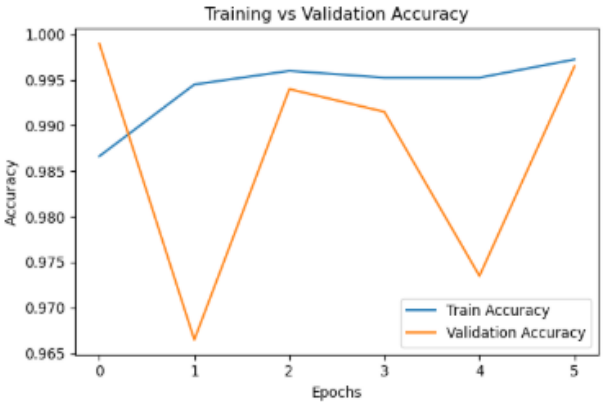


5. MobileNetV2

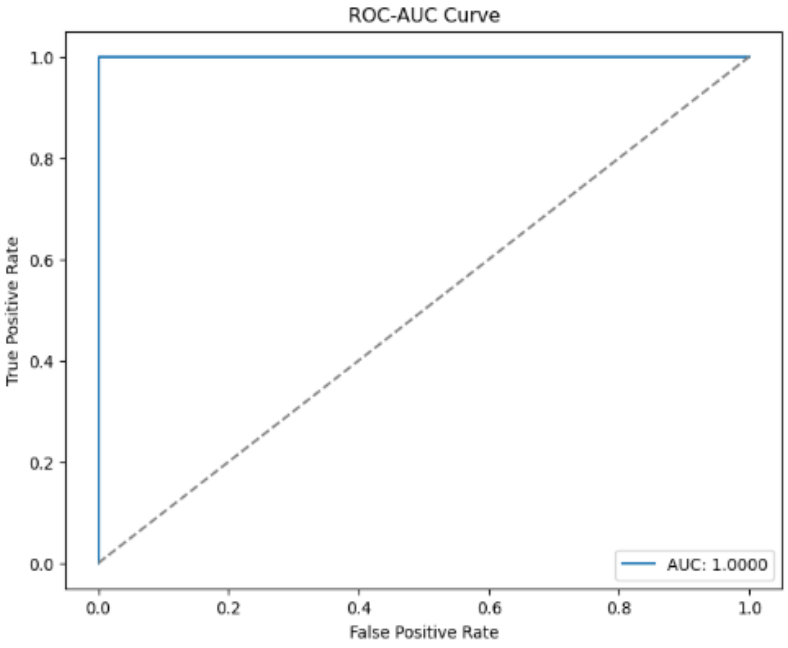
- **Description:** MobileNetV2 is designed for mobile and embedded applications, using depthwise separable convolutions for efficiency. It is lightweight while maintaining good performance on image classification tasks.
- **Implementation:** Use MobileNetV2 pre-trained on large datasets or design a custom version for resource-constrained environments.
- **Strengths:** Extremely efficient, optimized for mobile, lower computational cost.
- **Example Use:** Deploy MobileNetV2 in a mobile app for real-time face detection and classification, or in devices with limited resources.

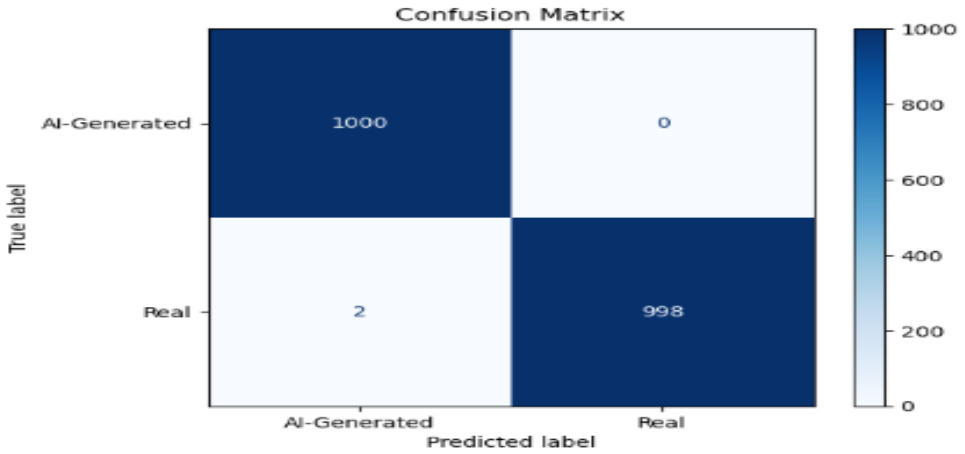
Classification Report:

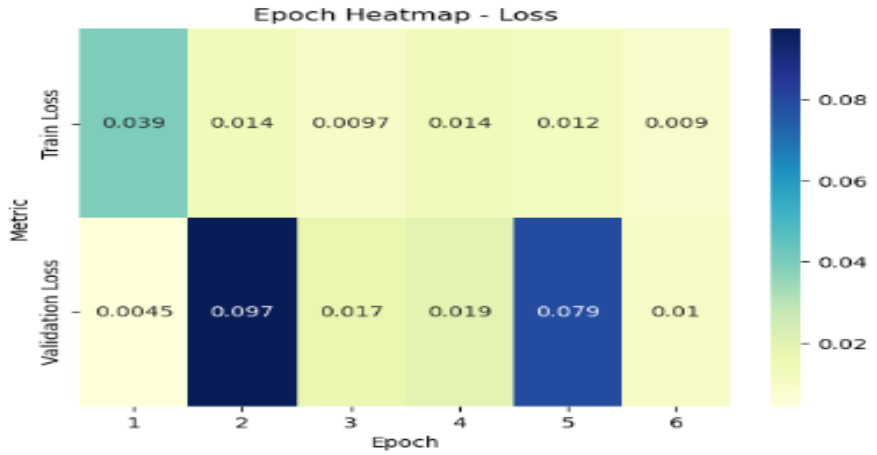
	precision	recall	f1-score	support
AI-Generated	1.00	1.00	1.00	1000
Real	1.00	1.00	1.00	1000
accuracy			1.00	2000
macro avg	1.00	1.00	1.00	2000
weighted avg	1.00	1.00	1.00	2000



Final Training Accuracy: 0.9973
Final Validation Accuracy: 0.9965
63/63 13s 185ms/step



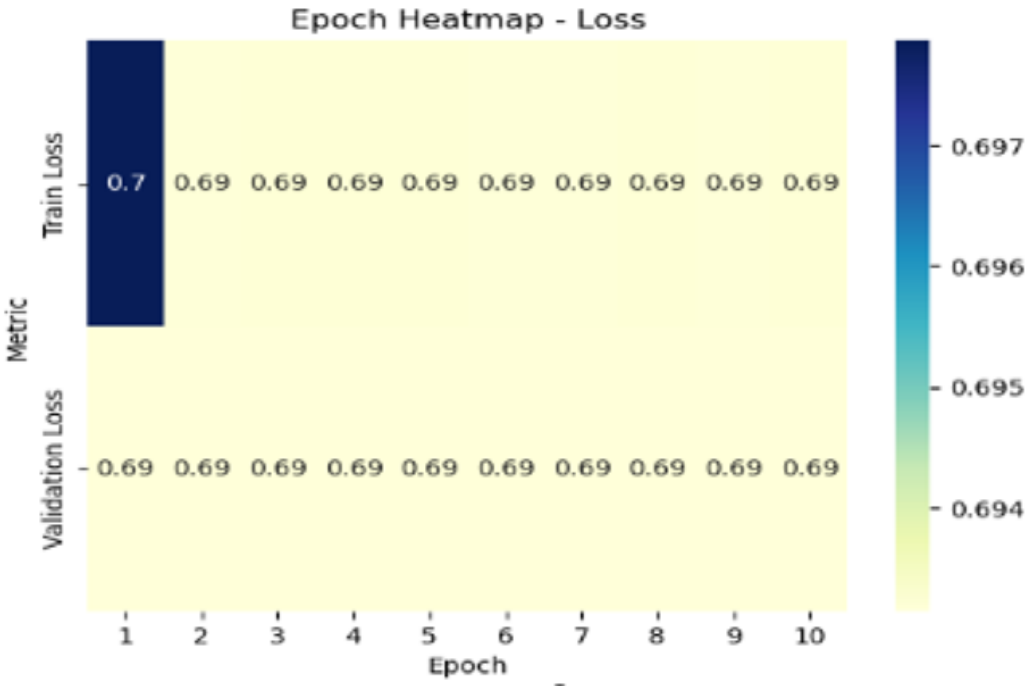


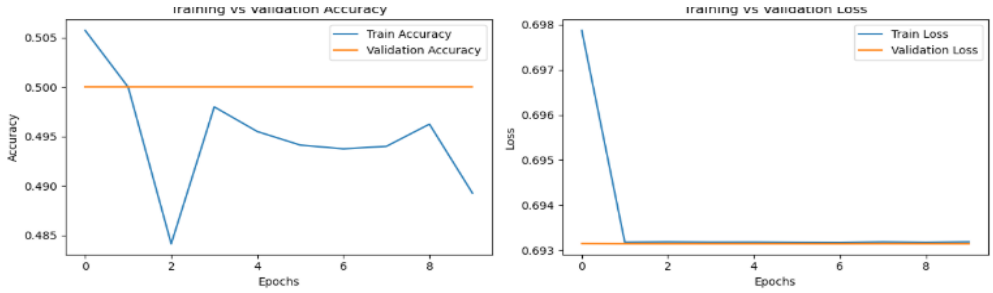


6. EfficientNetB0

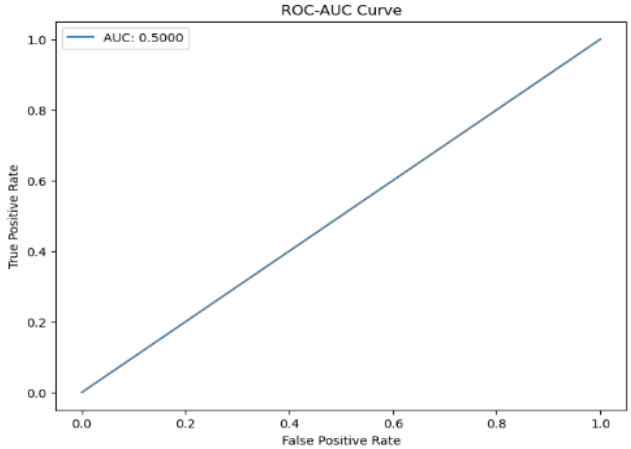
- **Description:** EfficientNet scales the depth, width, and resolution of the network, finding an optimal balance between model size and accuracy. EfficientNetB0 is the smallest and most basic version.
- **Implementation:** EfficientNetB0 can be fine-tuned on smaller datasets with high accuracy, making it a good choice for projects needing efficiency.
- **Strengths:** Highly efficient, reduces computational overhead, scales well across different tasks.
- **Example Use:** Fine-tune EfficientNetB0 for tasks where computational efficiency is crucial, such as deploying on edge devices.

```
Classification Report:
              precision    recall  f1-score   support

AI-Generated       0.00      0.00      0.00      1000
        Real       0.50      1.00      0.67      1000

    accuracy                           0.50      2000
   macro avg       0.25      0.50      0.33      2000
weighted avg       0.25      0.50      0.33      2000
```



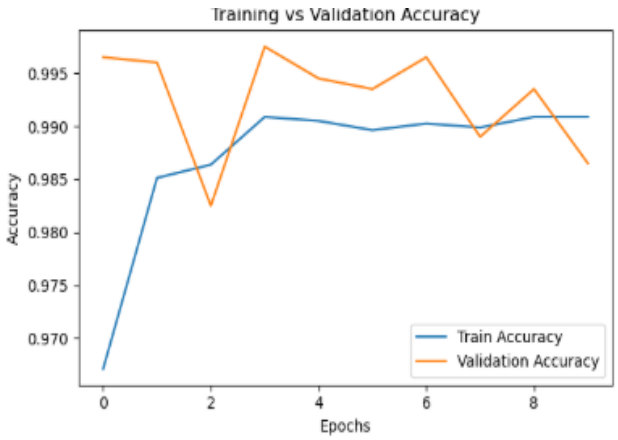



```
Training Accuracy: 0.4893
Validation Accuracy: 0.5000
63/63 ━━━━━━━━━━━━━━━━━━━━ 15s 202ms/step
```

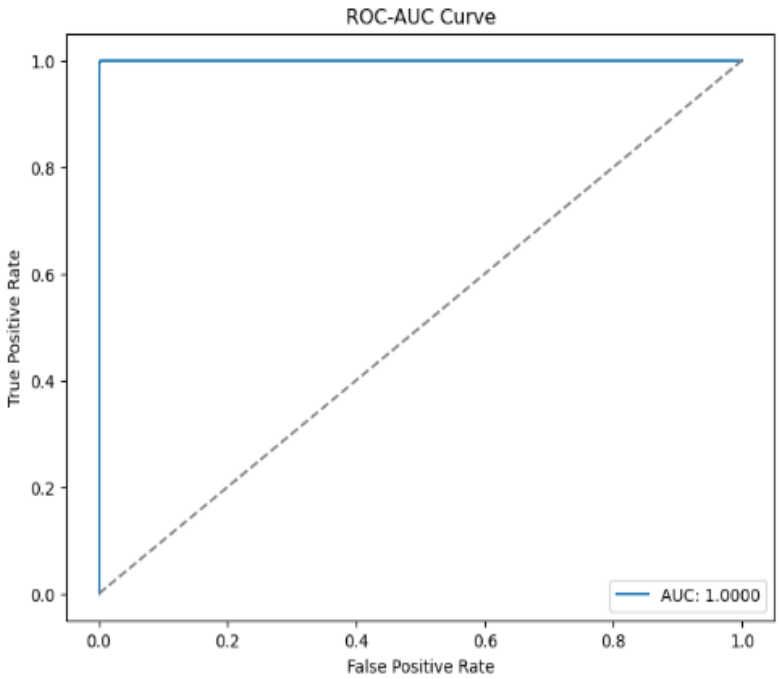


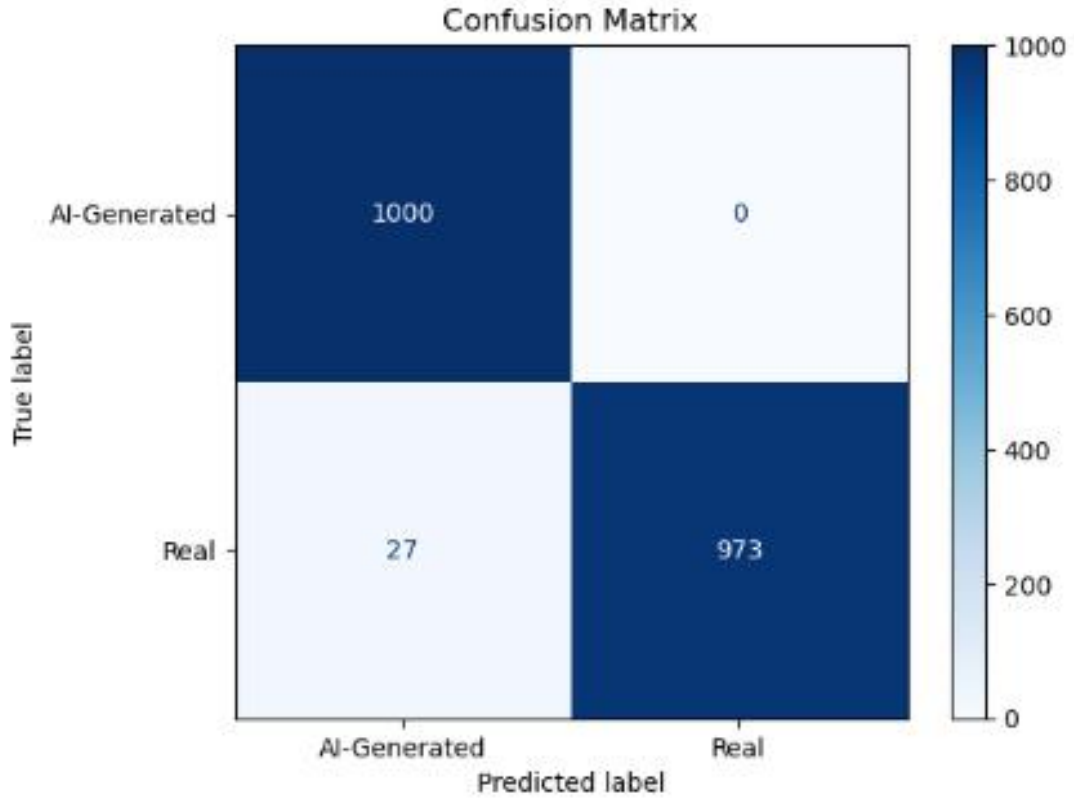
7. NASNetMobile

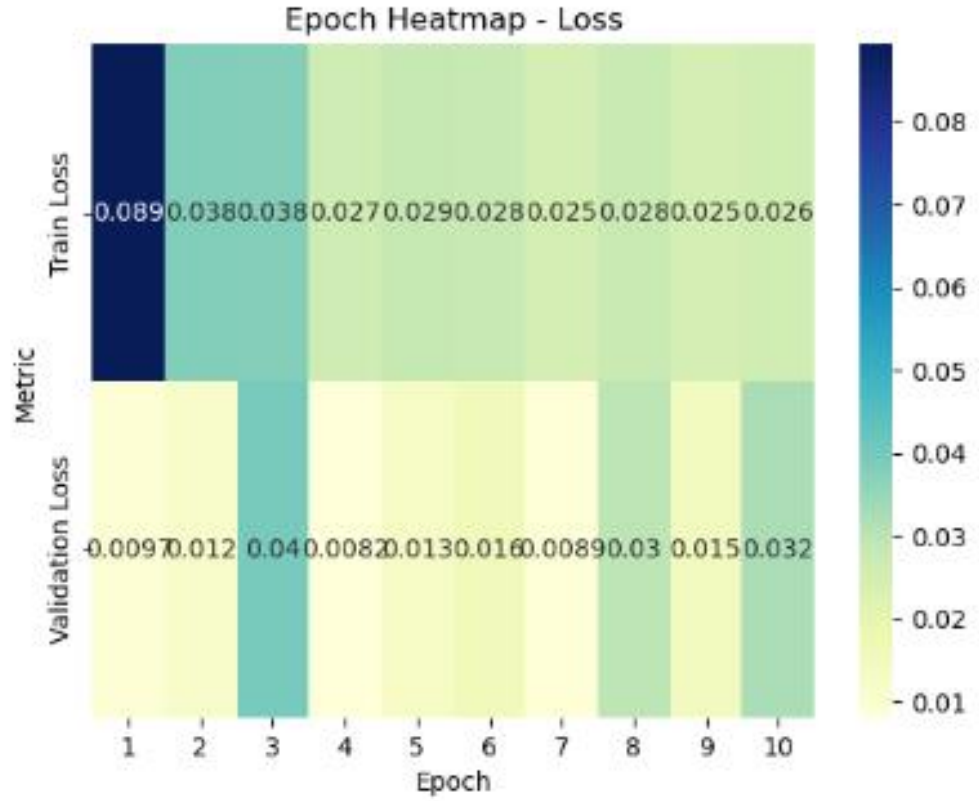
- **Description:** NASNetMobile is a mobile-optimized model found using Neural Architecture Search, a method for automating the design of neural networks. It is efficient and performs well on image classification tasks on mobile devices.
- **Implementation:** Use pre-trained NASNetMobile or adapt the architecture based on specific task needs.
- **Strengths:** Optimized for mobile, automated architecture search, good balance of accuracy and efficiency.
- **Example Use:** Implement NASNetMobile in mobile applications for real-time AI-generated face detection.




```
Training Accuracy: 0.9909
Validation Accuracy: 0.9865
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 293ms/step
```




Confusion Matrix
AI-Generated
Real
True label
1000
0
27
973
AI-Generated
Real
Predicted label
1000
800
600
400
200
0


Epoch Heatmap - Loss
Train Loss
Validation Loss
Metric
0.089 0.038 0.038 0.027 0.029 0.028 0.025 0.028 0.025 0.026
0.0097 0.012 0.040 0.0082 0.013 0.016 0.0089 0.03 0.015 0.032
1 2 3 4 5 6 7 8 9 10
Epoch
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01

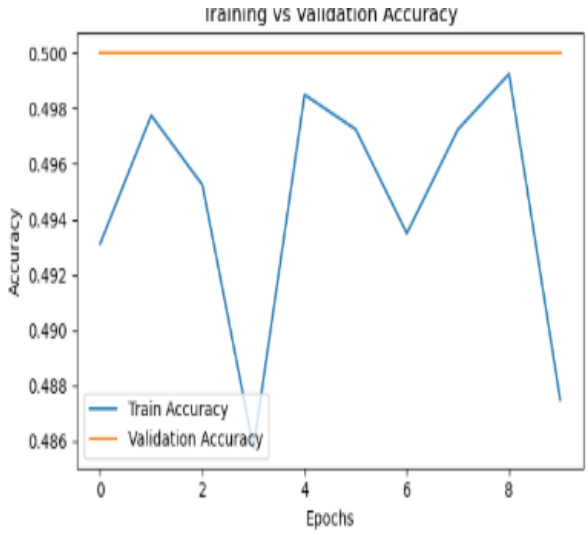
8. Vision Transformer (ViT)

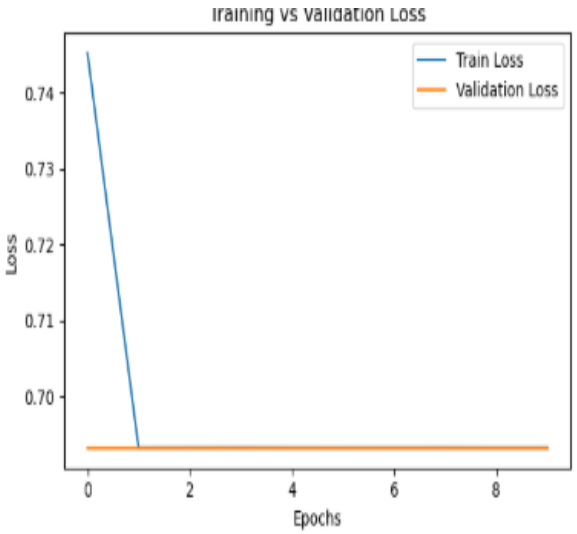
- Description:
  Vision Transformers (ViT) leverage the Transformer architecture from NLP and adapt it to vision tasks. By dividing an image into patches, it treats each patch as a token and applies the self-attention mechanism to model long-range dependencies and global context. ViT excels in processing structured and unstructured visual data while requiring minimal inductive biases.
- Implementation:
  You can use pre-trained ViT models (e.g., from HuggingFace Transformers or TensorFlow Hub) or fine-tune them on your specific dataset. ViT is highly customizable, allowing adjustments to patch size, depth, and number of attention heads to optimize for specific tasks.
- Strengths:
  - Captures global context efficiently using self-attention mechanisms.
  - Requires less task-specific architectural tweaking compared to CNNs.
  - Performs exceptionally well on large-scale datasets and benefits significantly from pretraining.

- Example Use:
  Use ViT for image classification tasks, such as detecting AI-generated versus real images, where global relationships between image patches are crucial for accurate predictions. It is especially beneficial in scenarios requiring robustness to subtle texture or pattern differences.

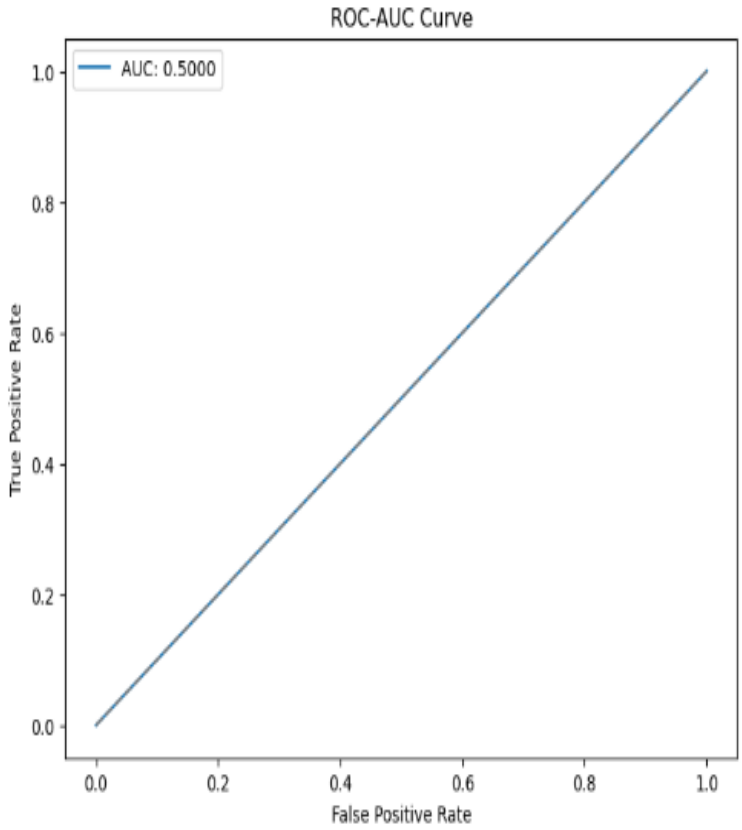
Classification Report:

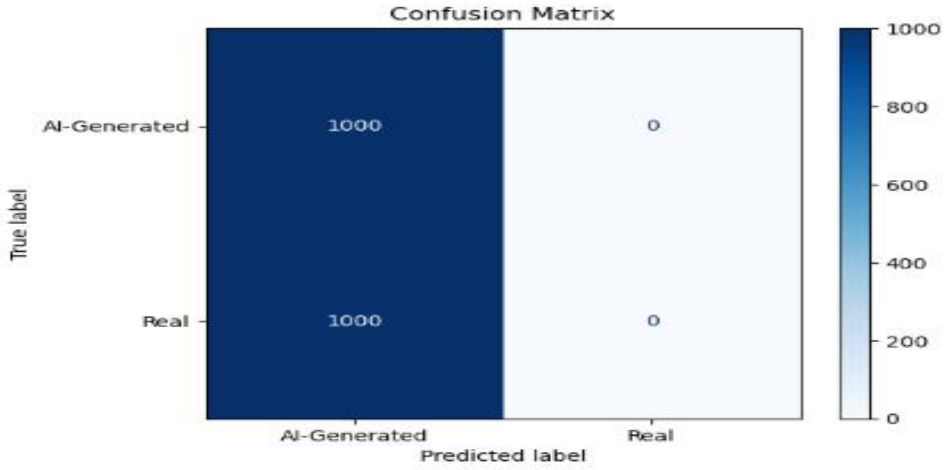
	precision	recall	f1-score	support
AI-Generated	0.50	1.00	0.67	1000
Real	0.00	0.00	0.00	1000
accuracy			0.50	2000
macro avg	0.25	0.50	0.33	2000
weighted avg	0.25	0.50	0.33	2000

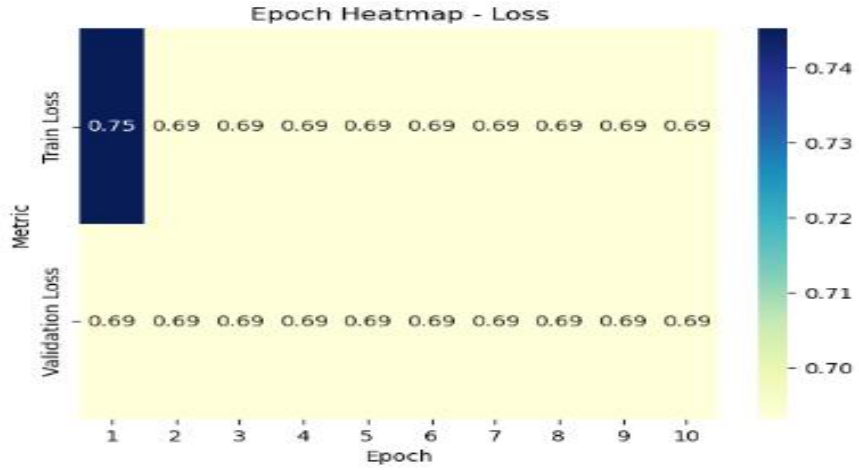





```
Training Accuracy: 0.4875
Validation Accuracy: 0.5000
63/63 ━━━━━━━━━━━━━━━━━━━━ 46s 708ms/step
```





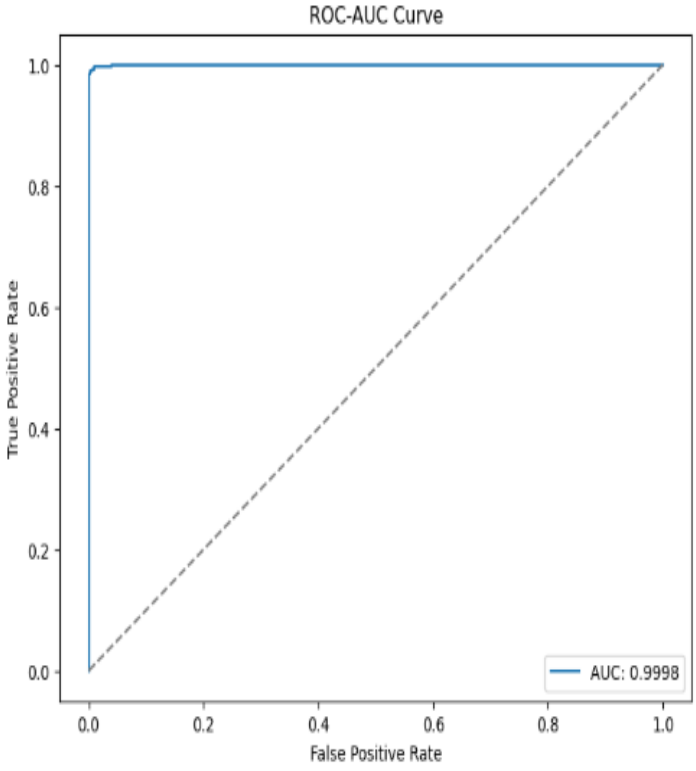


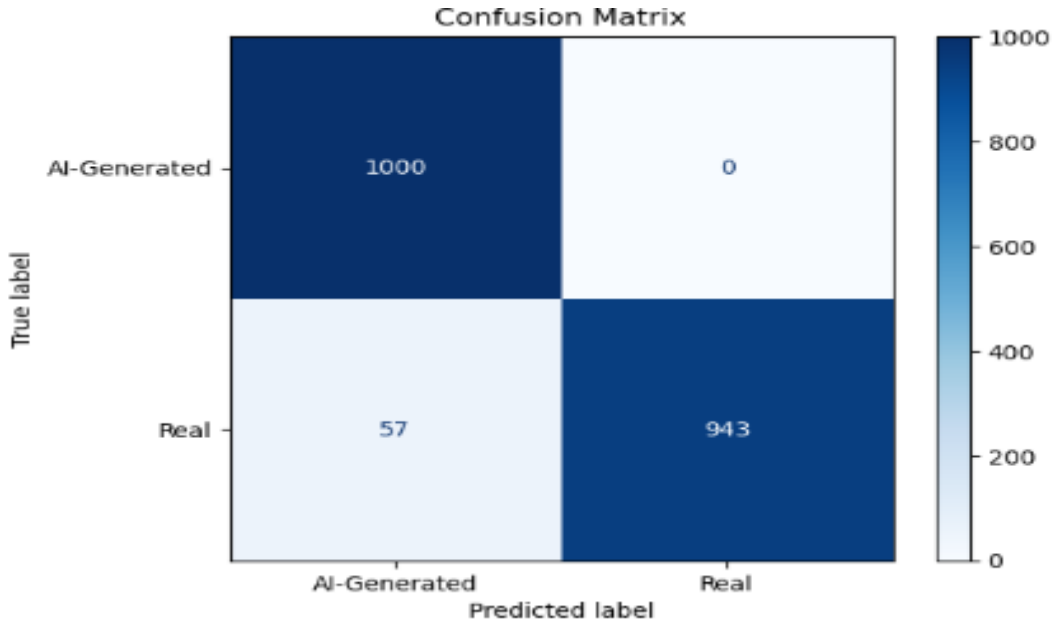
9. Xception

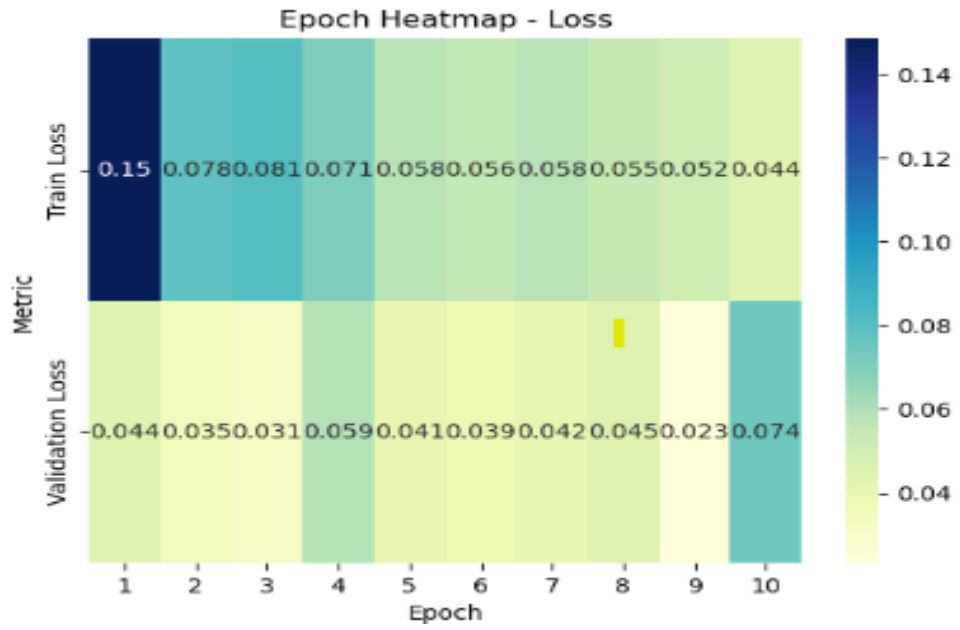
- **Description:** Xception is an advanced version of the Inception architecture that exclusively uses depthwise separable convolutions, making it highly efficient while maintaining high performance.
- **Implementation:** Use Xception pre-trained on large datasets for transfer learning or train it from scratch for a custom task.
- **Strengths:** Superior efficiency compared to Inception, very effective at extracting features from complex datasets.
- **Example Use:** Fine-tune Xception for detecting AI-generated faces or other advanced image classification tasks.

```
Classification Report:
              precision    recall  f1-score   support

AI-Generated       0.95      1.00      0.97      1000
        Real       1.00      0.94      0.97      1000

    accuracy                           0.97      2000
   macro avg       0.97      0.97      0.97      2000
weighted avg       0.97      0.97      0.97      2000
```


```
Training Accuracy: 0.9831
Validation Accuracy: 0.9715
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 256ms/step
```

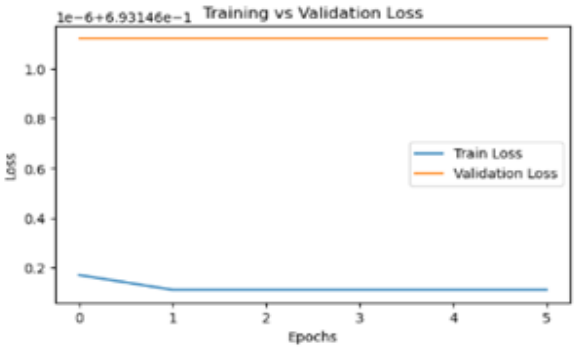




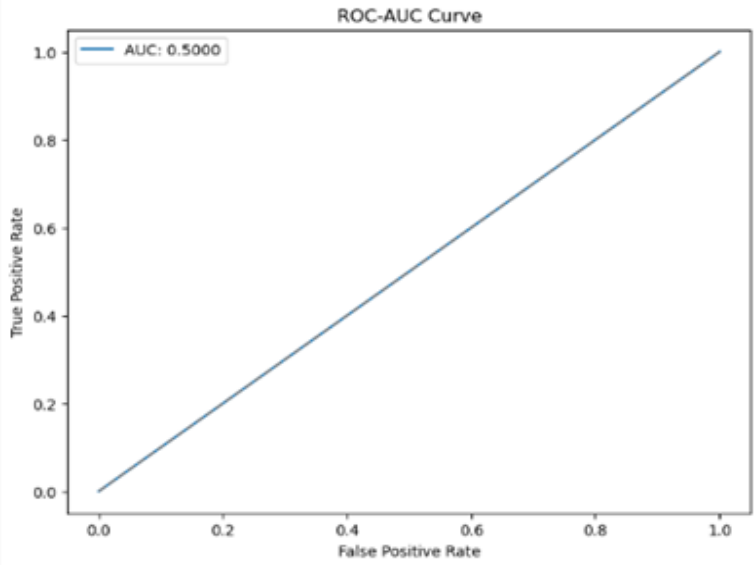


10. SqueezeNet

- **Description:** SqueezeNet is a lightweight model designed to be smaller and more efficient while maintaining high performance. It uses "fire modules," which consist of a squeeze layer and an expand layer, reducing the number of parameters.
- **Implementation:** Pre-trained SqueezeNet models can be used for fine-tuning, or you can implement custom versions depending on the task.
- **Strengths:** Small model size, highly efficient, good for deployment in environments with limited computational resources.
- **Example Use:** Deploy SqueezeNet for real-time applications on embedded devices, especially in cases where processing power is constrained.



63/63 ━━━━━━━━━━━━━━━━━━━━ 13s 186ms/step



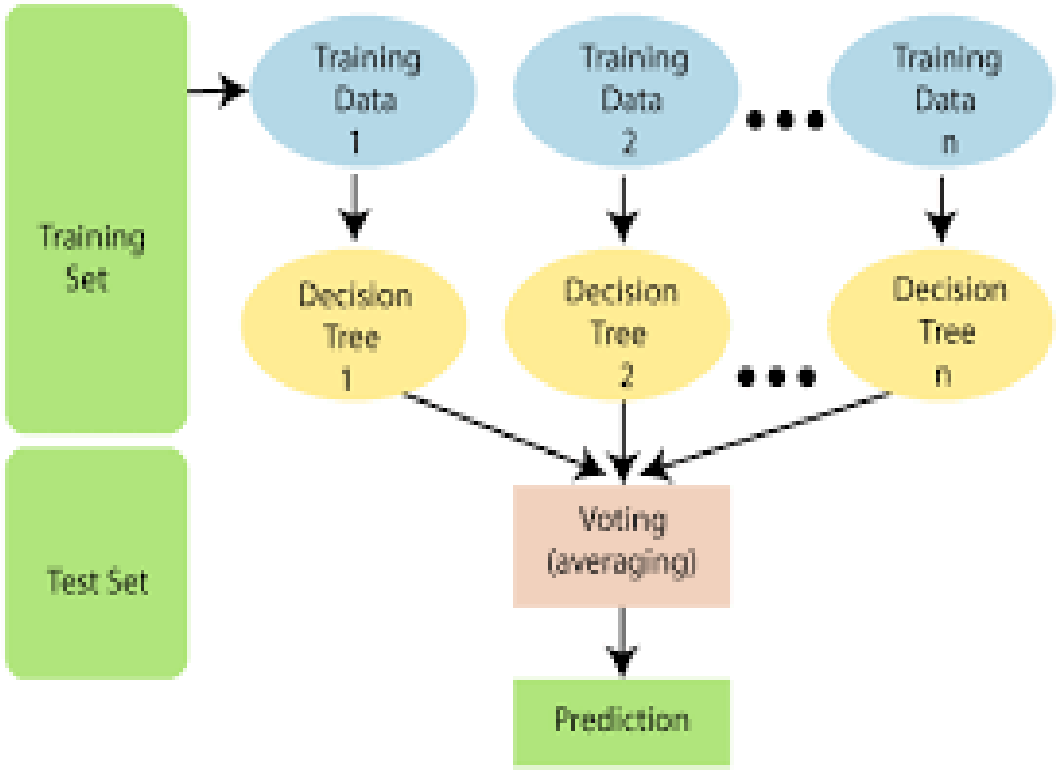
Classification Report:

	precision	recall	f1-score	support
AI-Generated	0.50	1.00	0.67	1000
Real	0.00	0.00	0.00	1000
accuracy			0.50	2000
macro avg	0.25	0.50	0.33	2000
weighted avg	0.25	0.50	0.33	2000

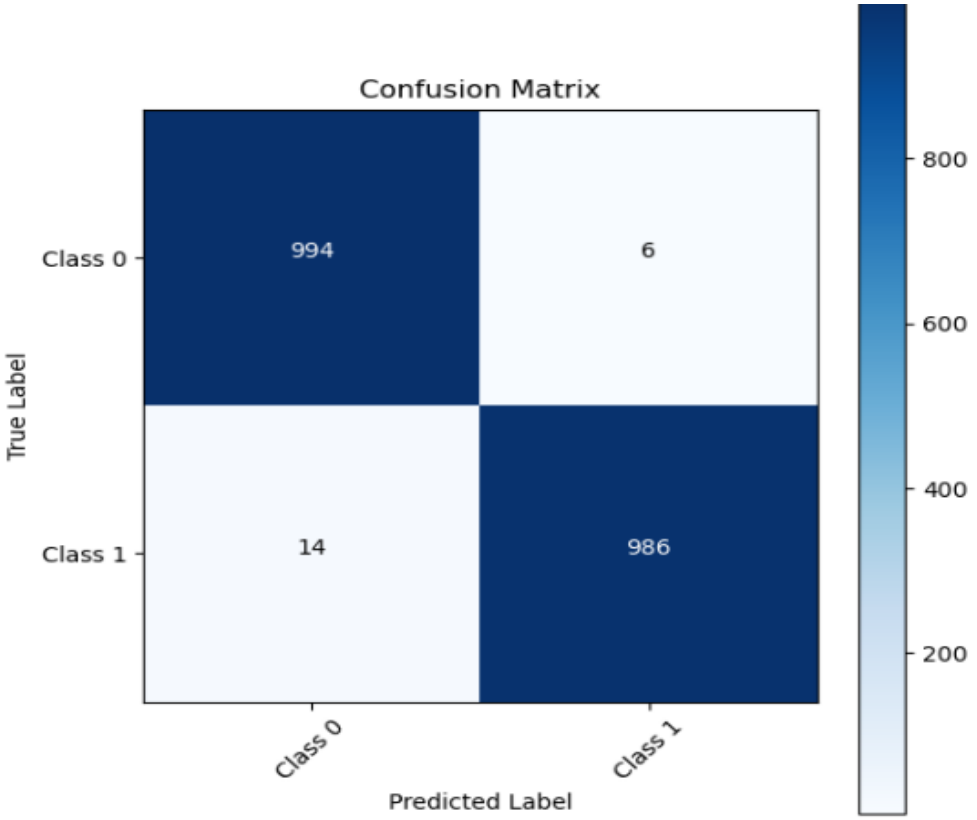
Machine Learning Models

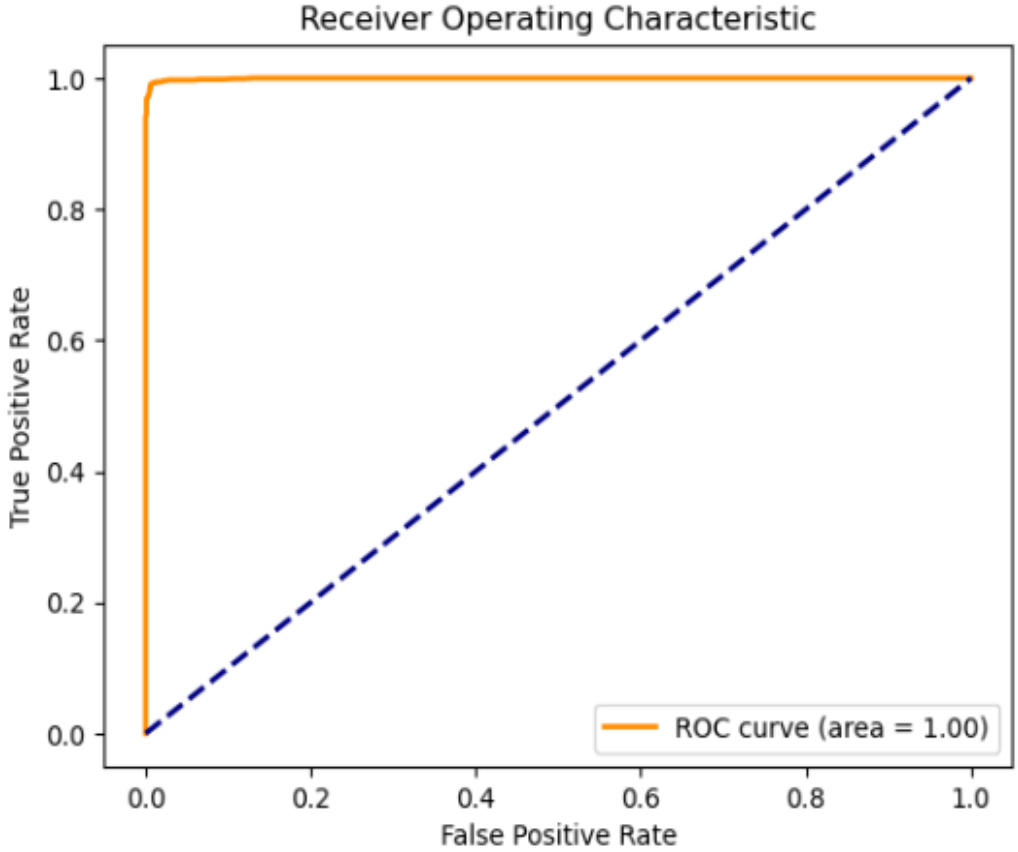
1. Random Forest:

- **Overview:** Random Forest is an ensemble learning method combining multiple decision trees to enhance classification accuracy.
- **Key Features:** Handles non-linear data, robust to noise, and reduces overfitting compared to single decision trees.
- **Use Case:** Effective for tabular data classification, especially in image-related tasks with extracted features.




```
Loading training data...
Found 8000 images belonging to 2 classes.
Loading testing data...
Found 2000 images belonging to 2 classes.
Training Random Forest model...
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.99      0.99      0.99      1000
     Class 1       0.99      0.99      0.99      1000

    accuracy                           0.99      2000
   macro avg       0.99      0.99      0.99      2000
weighted avg       0.99      0.99      0.99      2000

Confusion Matrix:
[[994   6]
 [ 14 986]]
```

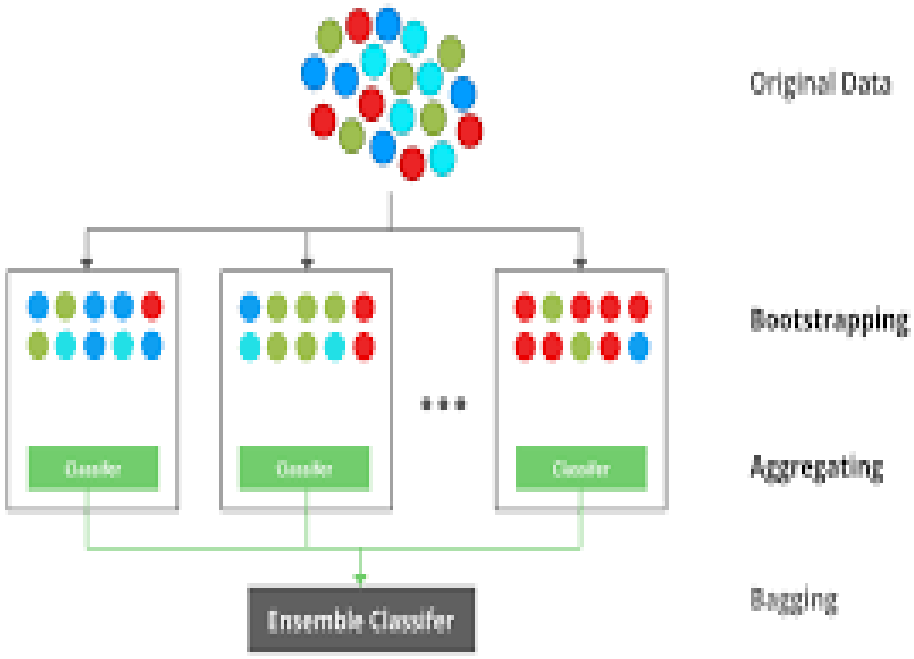





```
Training Accuracy: 1.00
Testing Accuracy: 0.99
```

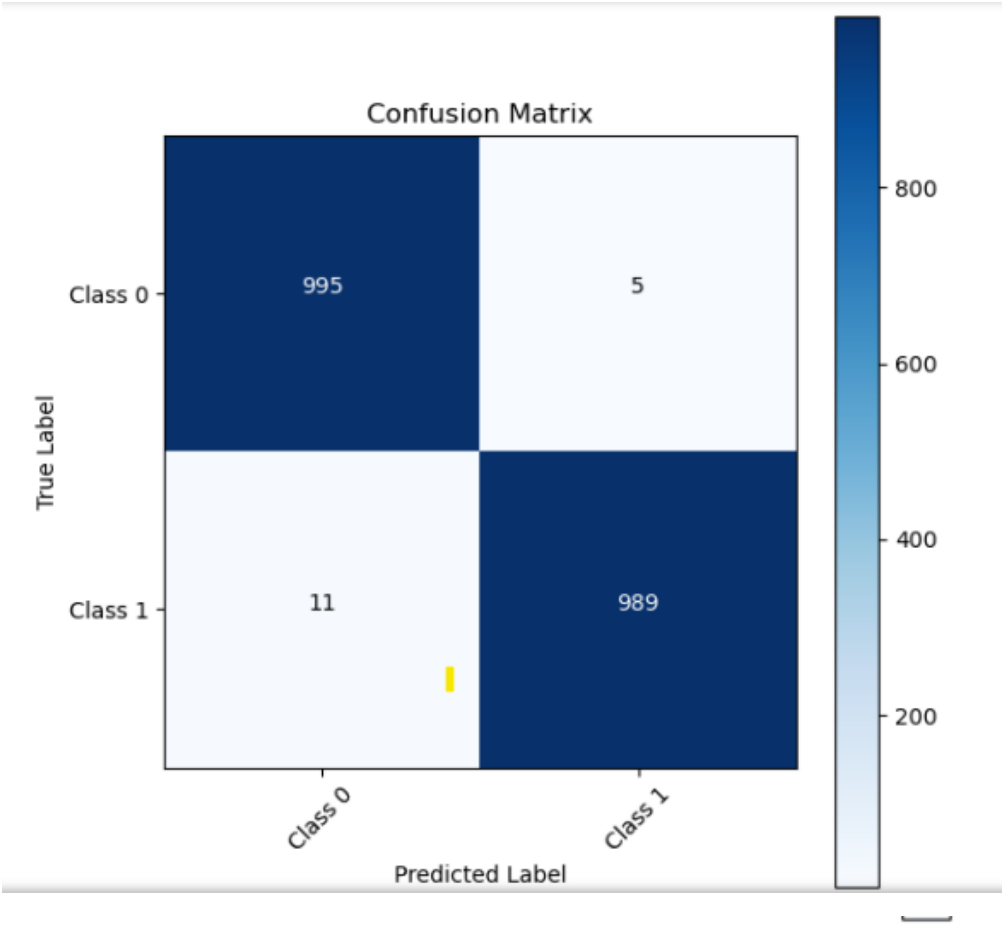
2. XGBoost:

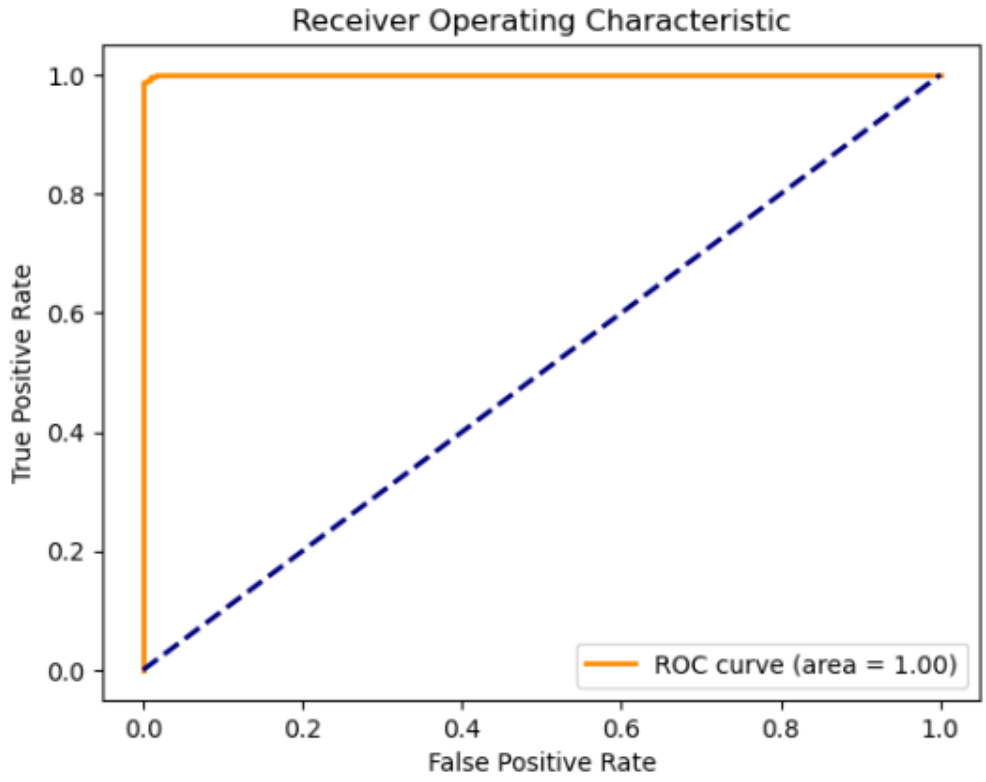
- **Overview:** XGBoost (Extreme Gradient Boosting) is a gradient-boosted decision tree model optimized for speed and performance.
- **Key Features:** Efficient handling of sparse data, regularization to prevent overfitting, and support for parallel processing.
- **Use Case:** Suitable for feature-based image classification and large datasets.



-

```
Loading training data...
Found 8000 images belonging to 2 classes.
Loading testing data...
Found 2000 images belonging to 2 classes.
Training XGBoost model...
Evaluating model...
Training Accuracy: 1.00
Testing Accuracy: 0.99
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.99      0.99      0.99      1000
     Class 1       0.99      0.99      0.99      1000

    accuracy                           0.99      2000
   macro avg       0.99      0.99      0.99      2000
weighted avg       0.99      0.99      0.99      2000

Confusion Matrix:
[[995   5]
 [ 11 989]]
```

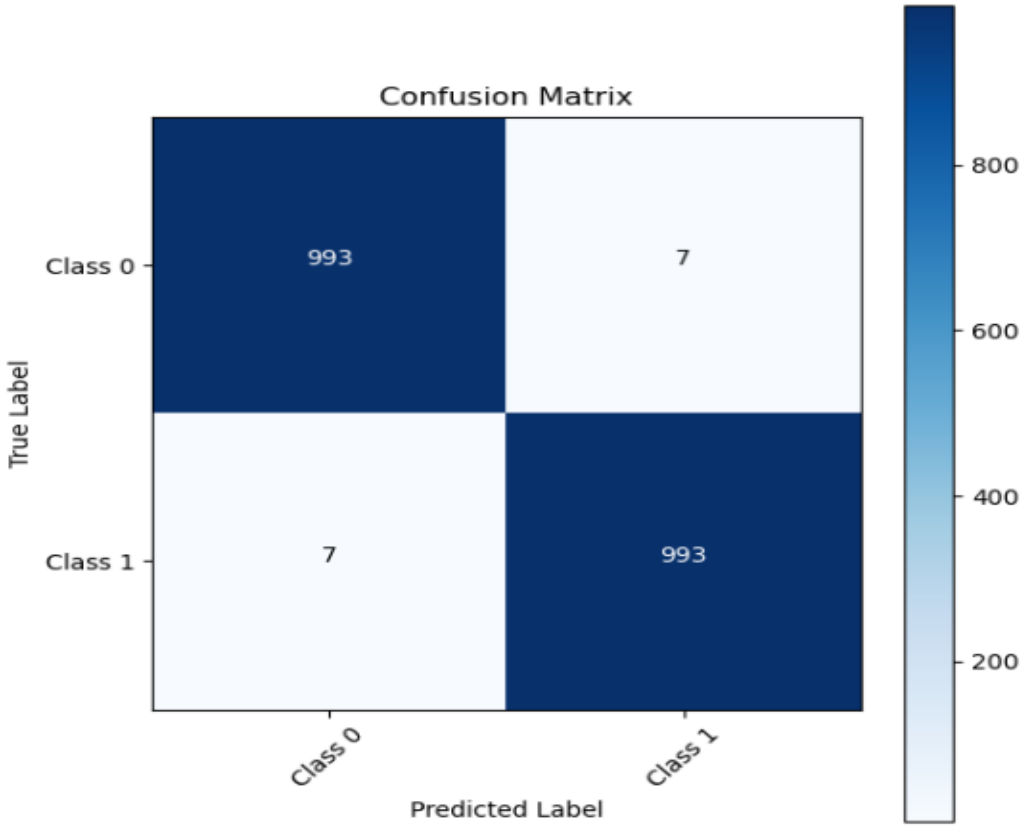

Confusion Matrix
995
5
11
989
Class 0
Class 1
True Label
Class 0
Class 1
Predicted Label
800
600
400
200

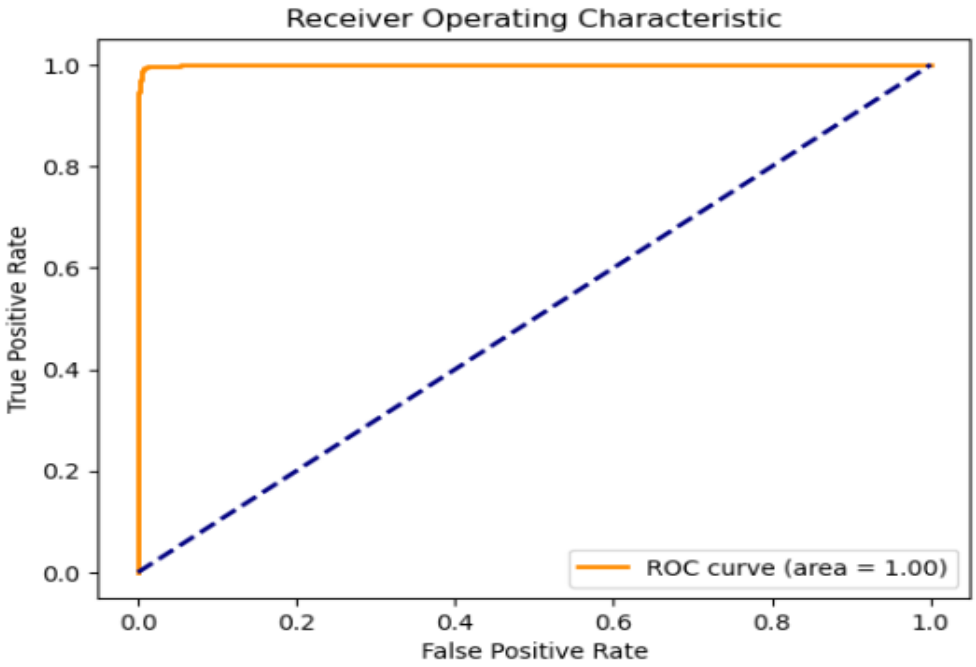

Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 1.00)

3. Logistic Regression:

- **Overview:** Logistic Regression is a simple yet effective method for binary classification problems.
- **Key Features:** Models the probability of an outcome using a logistic function, best for linearly separable data.
- **Use Case:** Preliminary image classification tasks or as a baseline model.

```
Loading training data...
Found 8000 images belonging to 2 classes.
Loading testing data...
Found 2000 images belonging to 2 classes.
Training Logistic Regression model...
Evaluating model...
Training Accuracy: 1.00
Testing Accuracy: 0.99
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.99      0.99      0.99      1000
     Class 1       0.99      0.99      0.99      1000

    accuracy                           0.99      2000
   macro avg       0.99      0.99      0.99      2000
weighted avg       0.99      0.99      0.99      2000

Confusion Matrix:
[[993   7]
 [  7 993]]
```

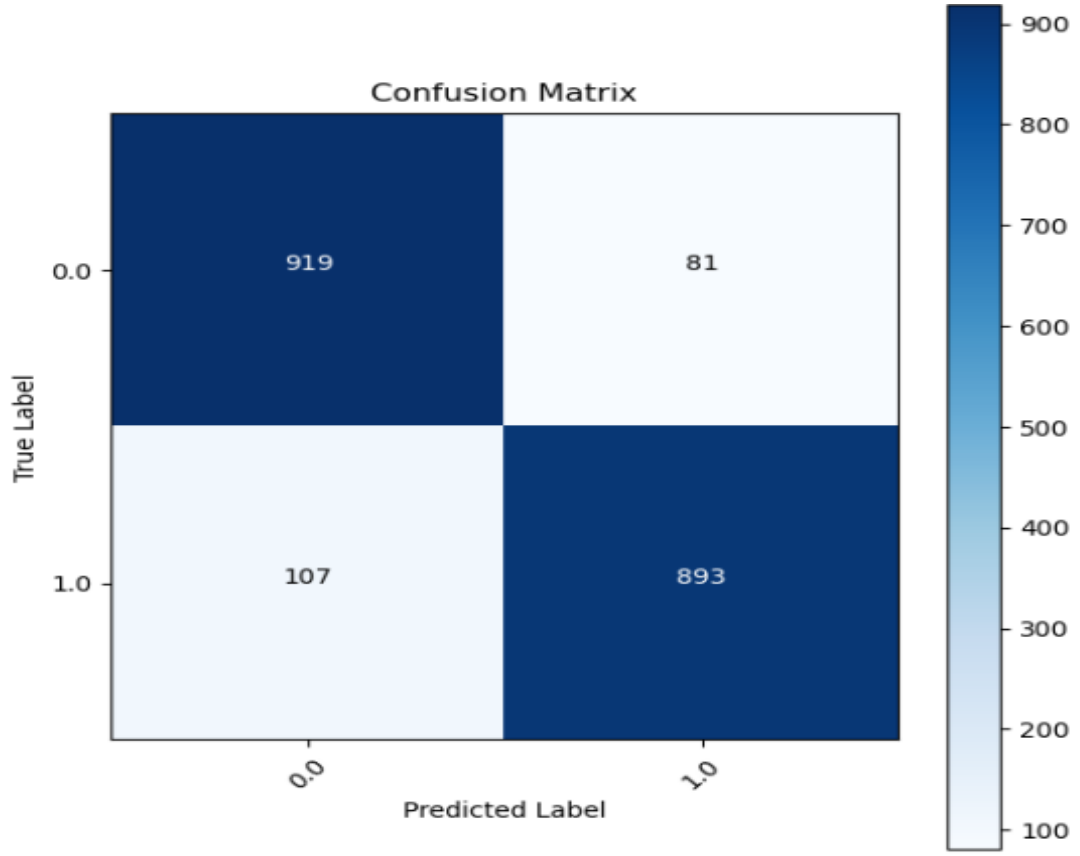


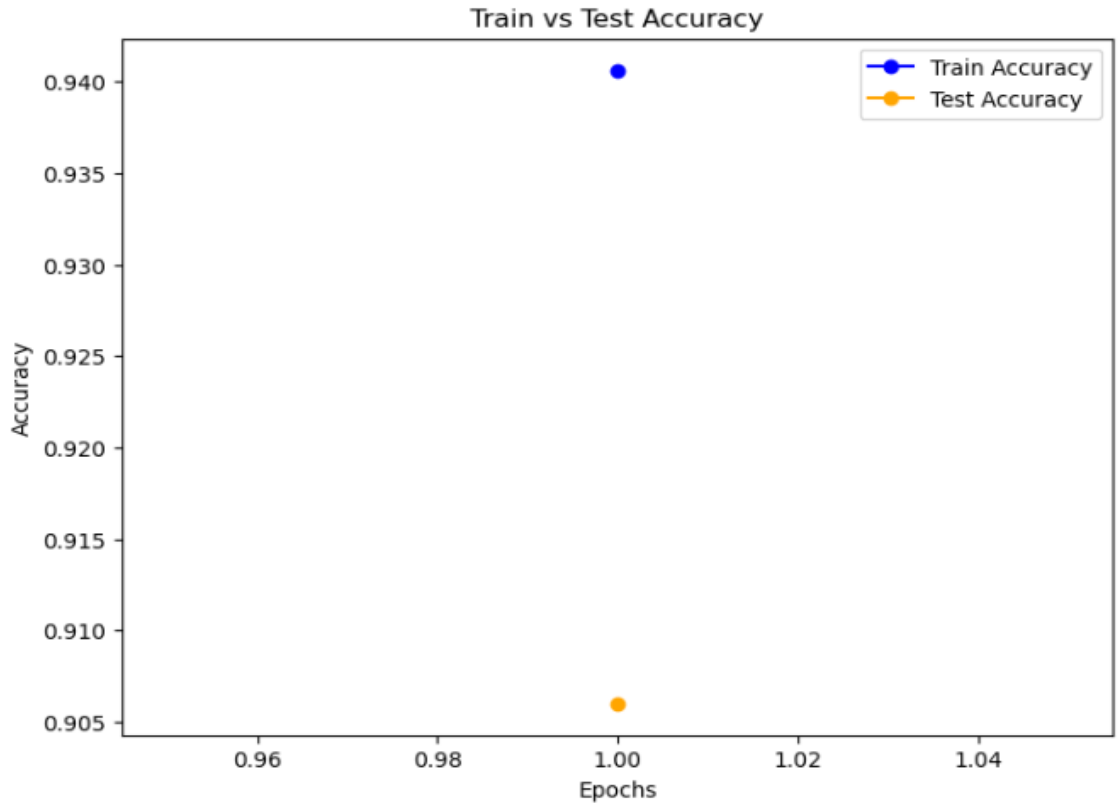


4 k-Nearest Neighbors (kNN):

- **Overview:** kNN is a non-parametric algorithm that classifies based on proximity to neighbors in feature space.
- **Key Features:** Simple implementation, no training phase required, and highly interpretable.
- **Use Case:** Effective for low-dimensional datasets or tasks where computational resources are limited.

```
Loading training data...
Found 8000 images belonging to 2 classes.
Loading testing data...
Found 2000 images belonging to 2 classes.
Training KNN model...
Training Accuracy: 0.94
Testing Accuracy: 0.91
Classification Report:
              precision    recall  f1-score   support

         0.0       0.90      0.92      0.91      1000
         1.0       0.92      0.89      0.90      1000

    accuracy                           0.91      2000
   macro avg       0.91      0.91      0.91      2000
weighted avg       0.91      0.91      0.91      2000

Confusion Matrix:
[[919  81]
 [107 893]]
```

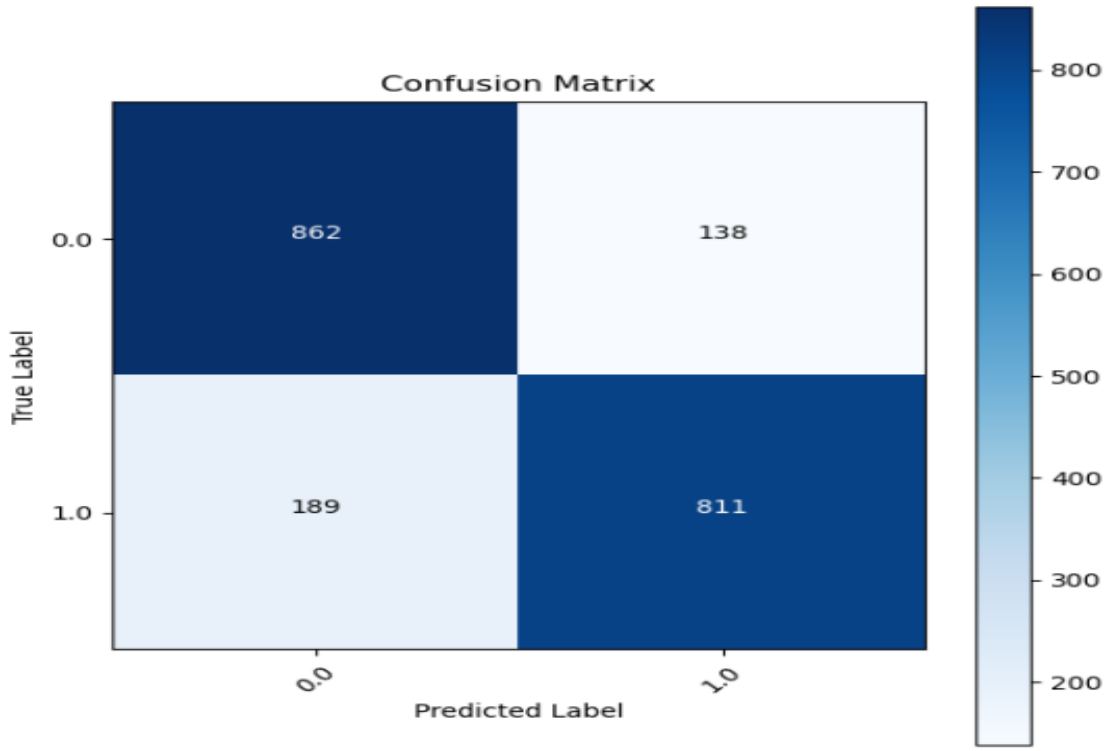

Confusion Matrix
919
81
107
893
0.0
1.0
True Label
0.0
1.0
Predicted Label
900
800
700
600
500
400
300
200
100

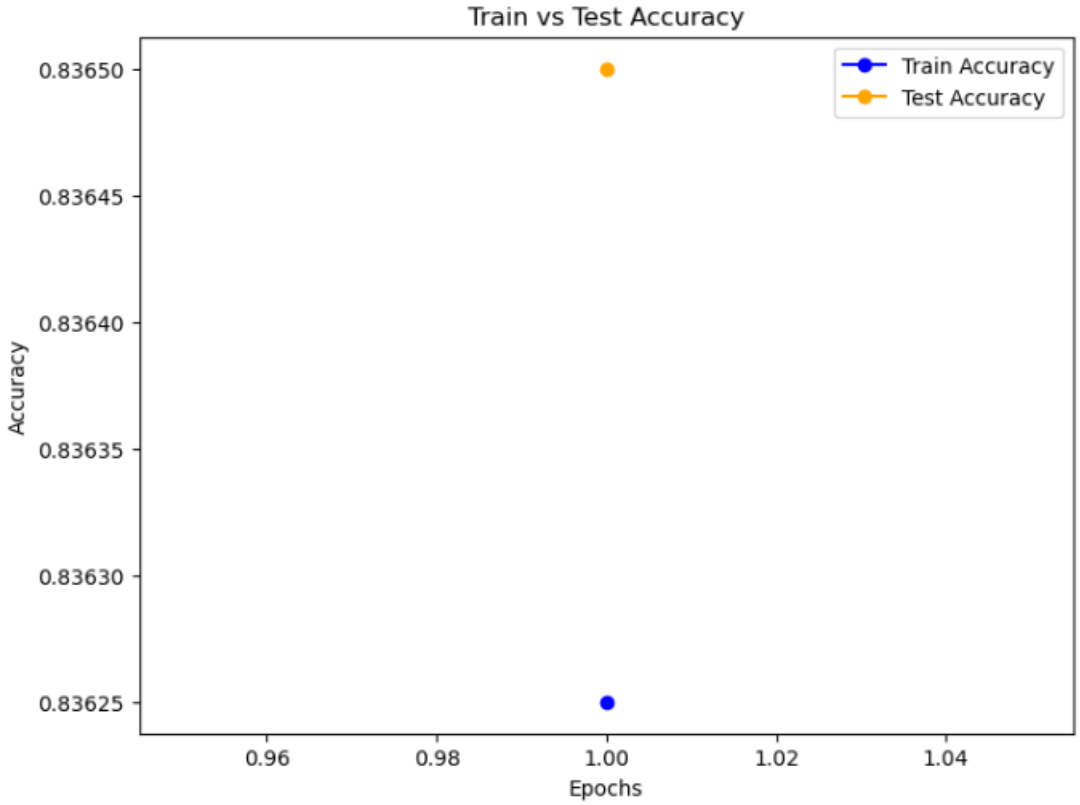

Train vs Test Accuracy
Train Accuracy
Test Accuracy
0.940
0.935
0.930
0.925
0.920
0.915
0.910
0.905
Accuracy
0.96
0.98
1.00
1.02
1.04
Epochs

5. Naive Bayes (Massive Bayes):

- **Overview:** Naive Bayes is a probabilistic classifier based on Bayes' theorem, assuming feature independence.
- **Key Features:** Fast, efficient with small datasets, and handles categorical data well.
- **Use Case:** Works well for textual or categorical data but can be adapted for low-dimensional image data.

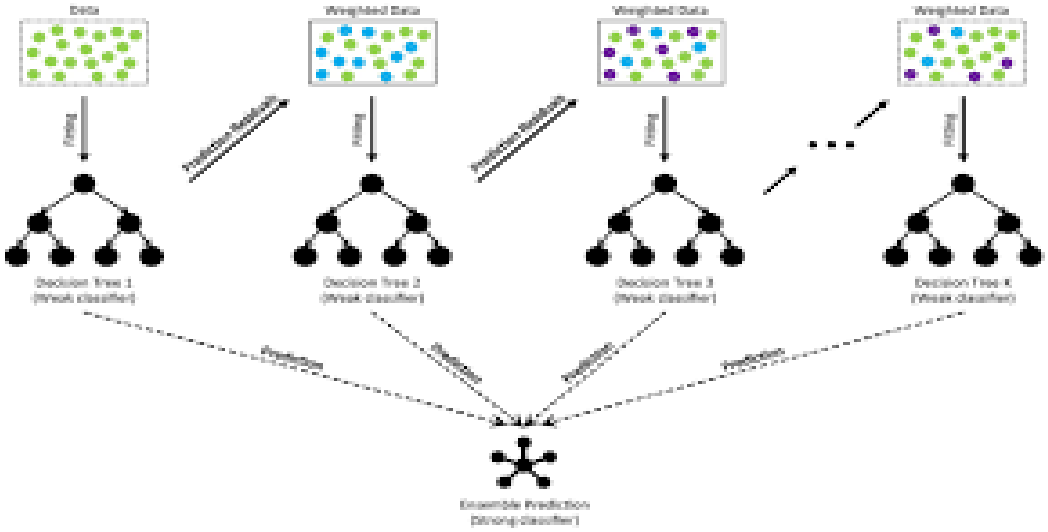
```
Loading training data...
Found 8000 images belonging to 2 classes.
Loading testing data...
Found 2000 images belonging to 2 classes.
Training Naive Bayes model...
Training Accuracy: 0.84
Testing Accuracy: 0.84
Classification Report:
              precision    recall  f1-score   support

         0.0       0.82      0.86      0.84      1000
         1.0       0.85      0.81      0.83      1000

    accuracy                           0.84      2000
   macro avg       0.84      0.84      0.84      2000
weighted avg       0.84      0.84      0.84      2000

Confusion Matrix:
[[862 138]
 [189 811]]
```



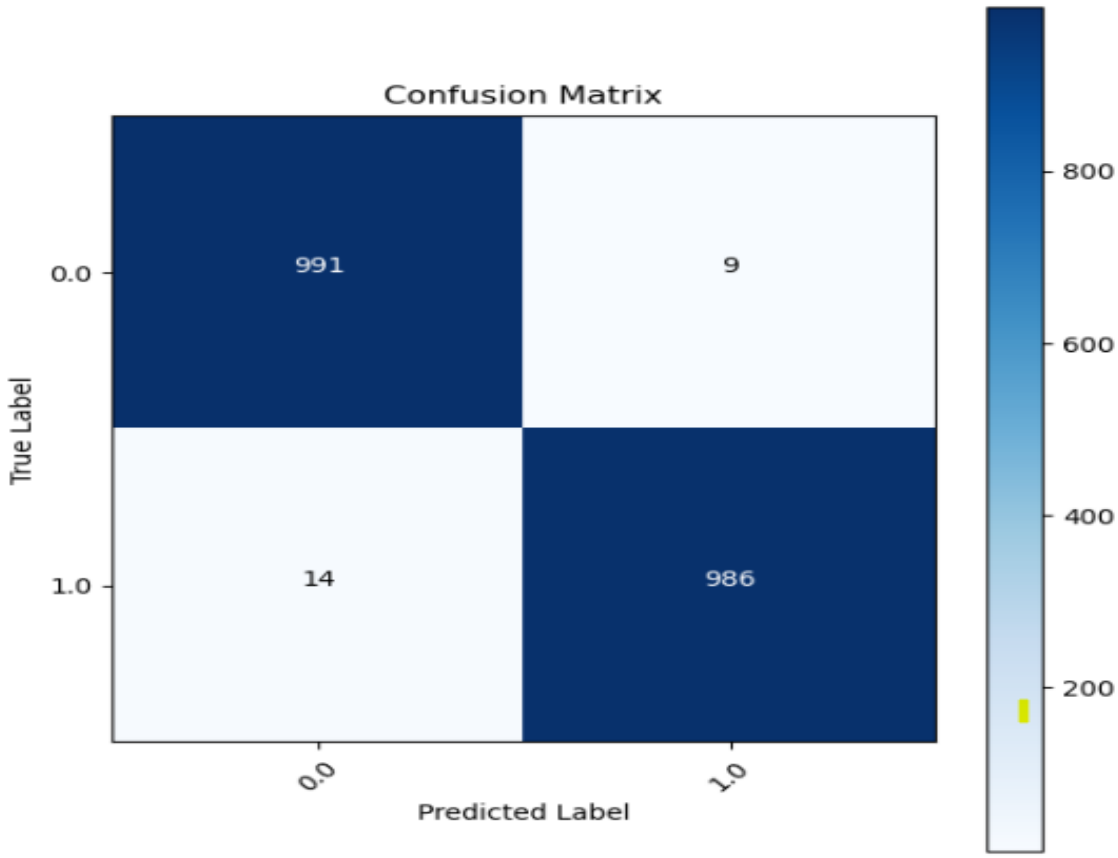


6. Gradient Boosting:

- **Overview:** Gradient Boosting builds an ensemble of weak learners (e.g., decision trees) by iteratively optimizing a loss function.
- **Key Features:** Handles non-linear relationships and complex feature interactions effectively.
- **Use Case:** Robust classification tasks with structured feature data extracted from images.



```
Loading training data...
Found 8000 images belonging to 2 classes.
Loading testing data...
Found 2000 images belonging to 2 classes.
Training Gradient Boosting model...
Training Accuracy: 1.00
Testing Accuracy: 0.99
Classification Report:
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99      1000
         1.0       0.99      0.99      0.99      1000

    accuracy                           0.99      2000
   macro avg       0.99      0.99      0.99      2000
weighted avg       0.99      0.99      0.99      2000

Confusion Matrix:
[[991   9]
 [ 14 986]]
```

- 

7 Ada Boost:

Overview

AdaBoost (Adaptive Boosting) builds an ensemble of weak learners (e.g., decision stumps) by iteratively adjusting the weights of misclassified samples, focusing more on difficult examples in each iteration.

Key Features

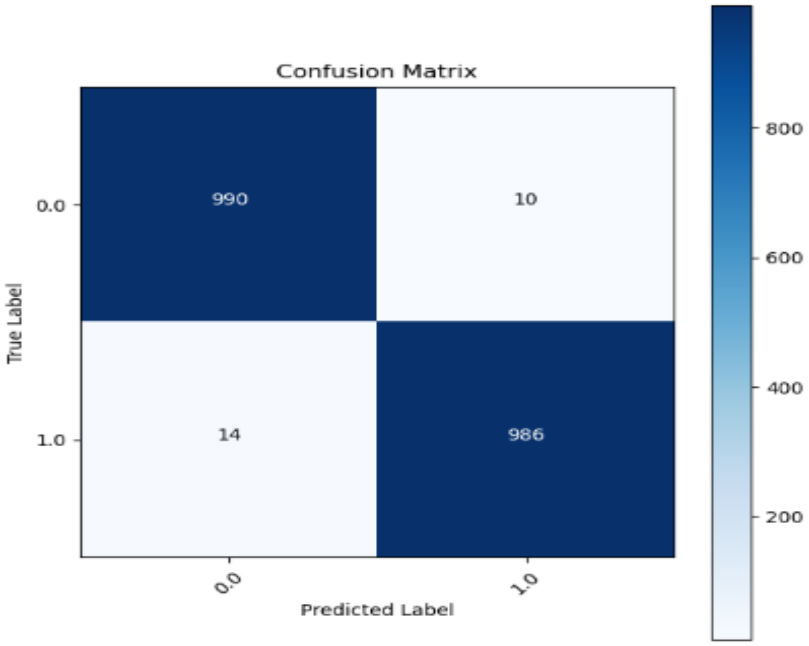
1. Combines multiple weak learners to form a strong classifier through weighted majority voting.
2. Iteratively adapts to the errors of previous learners by focusing on misclassified samples.

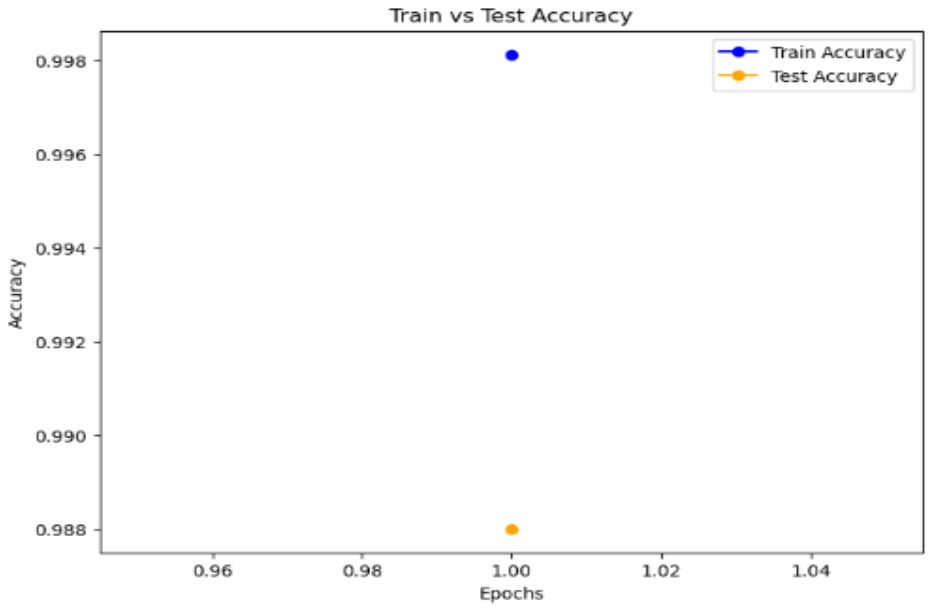
3. Uses exponential weighting to emphasize hard-to-classify instances.
4. Typically employs decision stumps but is flexible to other weak learners.
5. Effective in reducing both bias and variance when properly tuned.

Use Case

Effective in tasks like face detection, where structured feature data is analyzed to differentiate between human faces and non-faces with high accuracy.

```
Training Accuracy: 1.00
Testing Accuracy: 0.99
Classification Report:
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99      1000
         1.0       0.99      0.99      0.99      1000

    accuracy                           0.99      2000
   macro avg       0.99      0.99      0.99      2000
weighted avg       0.99      0.99      0.99      2000

Confusion Matrix:
[[990  10]
 [ 14 986]]
```





Best 4 DL Models

1 CNN:-

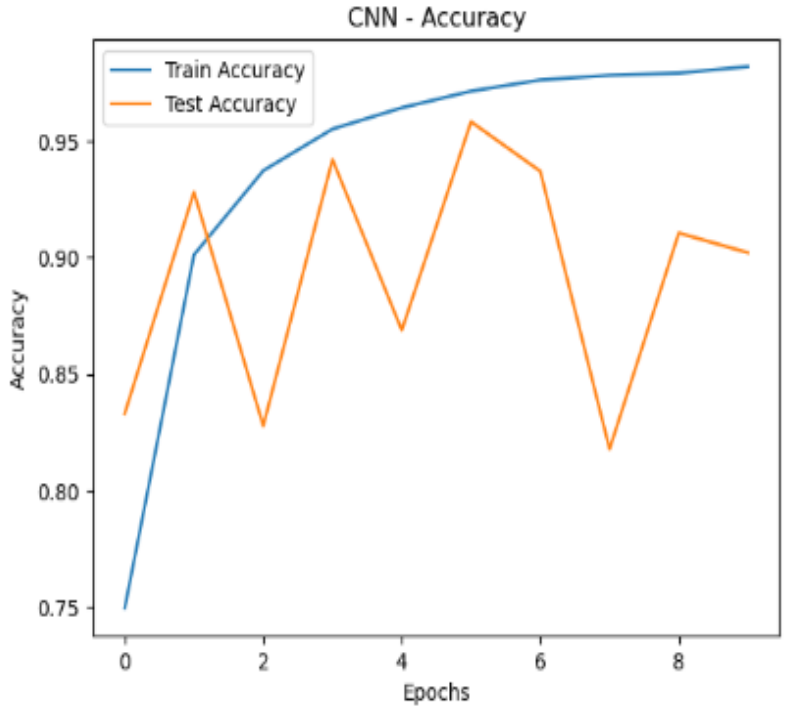
Overview of CNNs

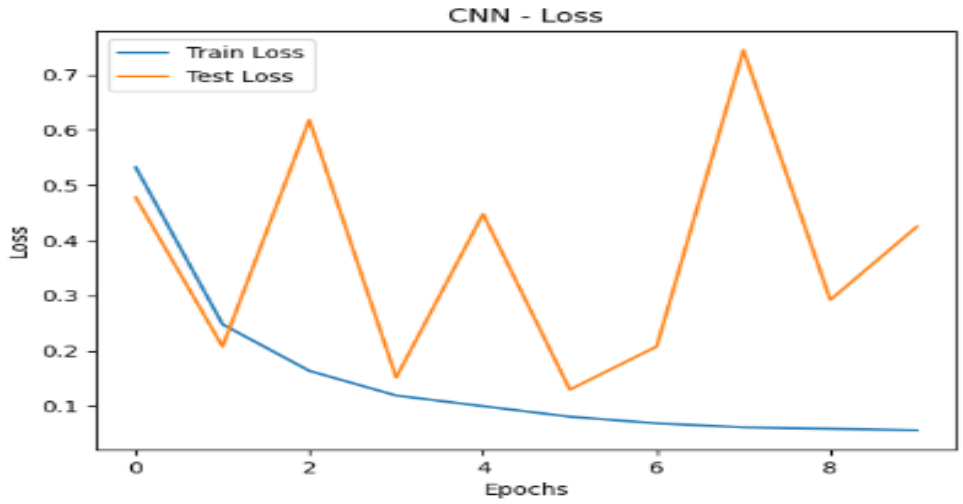
CNNs are designed to process data in grid-like structures, such as images, where each image can be represented as a 2D array (height, width) of pixels. These networks are composed of several layers, each designed to perform specific tasks:

- **Convolutional Layer**: This layer is the core building block of a CNN. It applies a set of filters (kernels) to the input image, generating feature maps that highlight various aspects of the image, such as edges, textures, and patterns.
- **Activation Function (ReLU)**: After the convolution operation, an activation function (usually ReLU) is applied element-wise to the feature maps to introduce non-linearity, enabling the network to model complex patterns.
- **Pooling Layer**: Pooling operations (e.g., max pooling) reduce the spatial dimensions of the feature maps, reducing computational complexity while retaining essential features. This is done by downsampling the feature maps, typically by selecting the maximum or average value from a subset of the map.

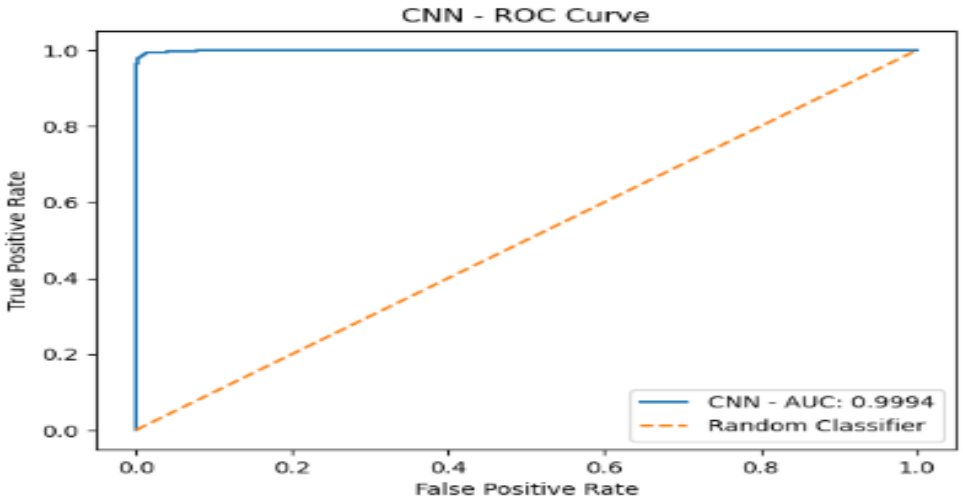
- **Fully Connected Layer**: After several convolutional and pooling layers, the output is flattened and passed through one or more fully connected layers. These layers interpret the extracted features and make the final classification decision.

```
Epoch 1/10
500/500 ━━━━━━━━━━━━━━━━━━━━ 183s 364ms/step - accuracy: 0.6584 - loss: 0.8174 - val_accuracy: 0.8330 - val_loss: 0.4777
Epoch 2/10
500/500 ━━━━━━━━━━━━━━━━━━━━ 186s 373ms/step - accuracy: 0.8766 - loss: 0.2965 - val_accuracy: 0.9280 - val_loss: 0.2074
Epoch 3/10
500/500 ━━━━━━━━━━━━━━━━━━━━ 185s 370ms/step - accuracy: 0.9339 - loss: 0.1692 - val_accuracy: 0.8278 - val_loss: 0.6183
Epoch 4/10
500/500 ━━━━━━━━━━━━━━━━━━━━ 187s 373ms/step - accuracy: 0.9521 - loss: 0.1250 - val_accuracy: 0.9420 - val_loss: 0.1514
Epoch 5/10
500/500 ━━━━━━━━━━━━━━━━━━━━ 186s 372ms/step - accuracy: 0.9660 - loss: 0.0982 - val_accuracy: 0.8687 - val_loss: 0.4478
Epoch 6/10
500/500 ━━━━━━━━━━━━━━━━━━━━ 188s 375ms/step - accuracy: 0.9669 - loss: 0.0936 - val_accuracy: 0.9582 - val_loss: 0.1294
Epoch 7/10
500/500 ━━━━━━━━━━━━━━━━━━━━ 180s 360ms/step - accuracy: 0.9768 - loss: 0.0642 - val_accuracy: 0.9370 - val_loss: 0.2075
Epoch 8/10
500/500 ━━━━━━━━━━━━━━━━━━━━ 186s 372ms/step - accuracy: 0.9825 - loss: 0.0509 - val_accuracy: 0.8177 - val_loss: 0.7445
Epoch 9/10
500/500 ━━━━━━━━━━━━━━━━━━━━ 185s 370ms/step - accuracy: 0.9793 - loss: 0.0613 - val_accuracy: 0.9105 - val_loss: 0.2921
Epoch 10/10
500/500 ━━━━━━━━━━━━━━━━━━━━ 187s 374ms/step - accuracy: 0.9836 - loss: 0.0532 - val_accuracy: 0.9020 - val_loss: 0.4252
```

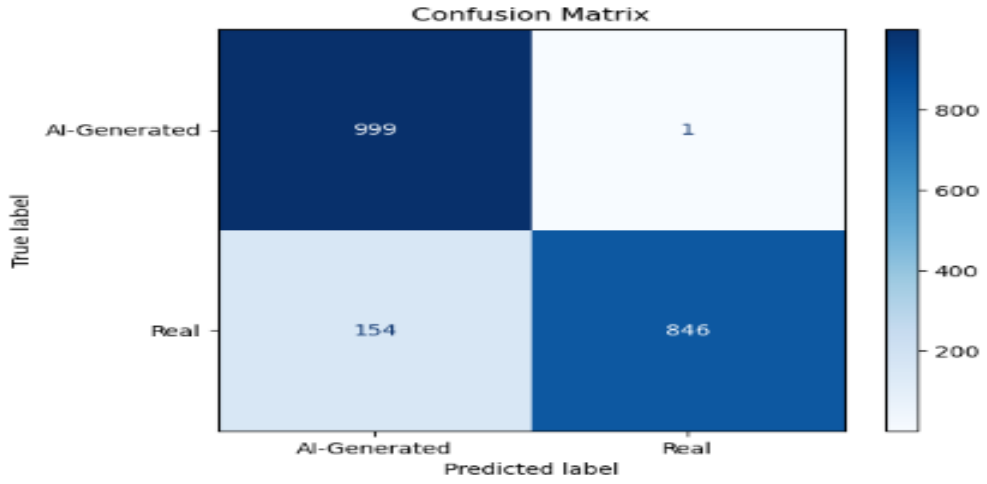





```
125/125 ━━━━━━━━━━━━━━━━━━━━ 11s 83ms/step
```




```
Classification Report:
              precision    recall  f1-score   support

AI-Generated       0.87      1.00      0.93      1000
        Real       1.00      0.85      0.92      1000

    accuracy                           0.92      2000
   macro avg       0.93      0.92      0.92      2000
weighted avg       0.93      0.92      0.92      2000
```



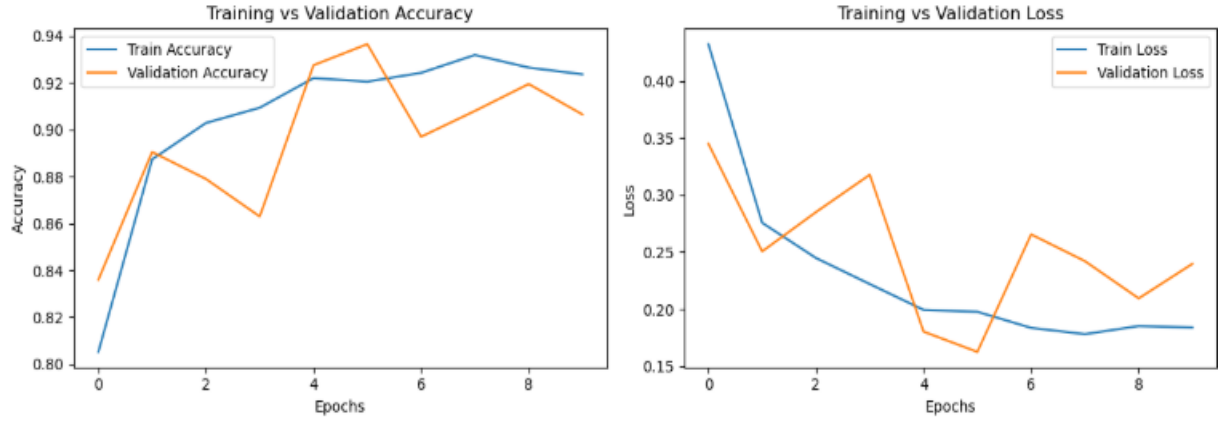
2 .VGG16

In the context of the project focused on detecting real versus AI-generated images, VGG16 can be an effective model for image classification. VGG16, a deep Convolutional Neural Network (CNN) architecture, is known for its simplicity and effectiveness in handling complex image recognition tasks. The model consists of 16 layers, including convolutional and fully connected layers, which helps it learn hierarchical features in images. For this project, VGG16 can be used to classify images as either real or AI-generated by leveraging its ability to extract intricate patterns from image data.

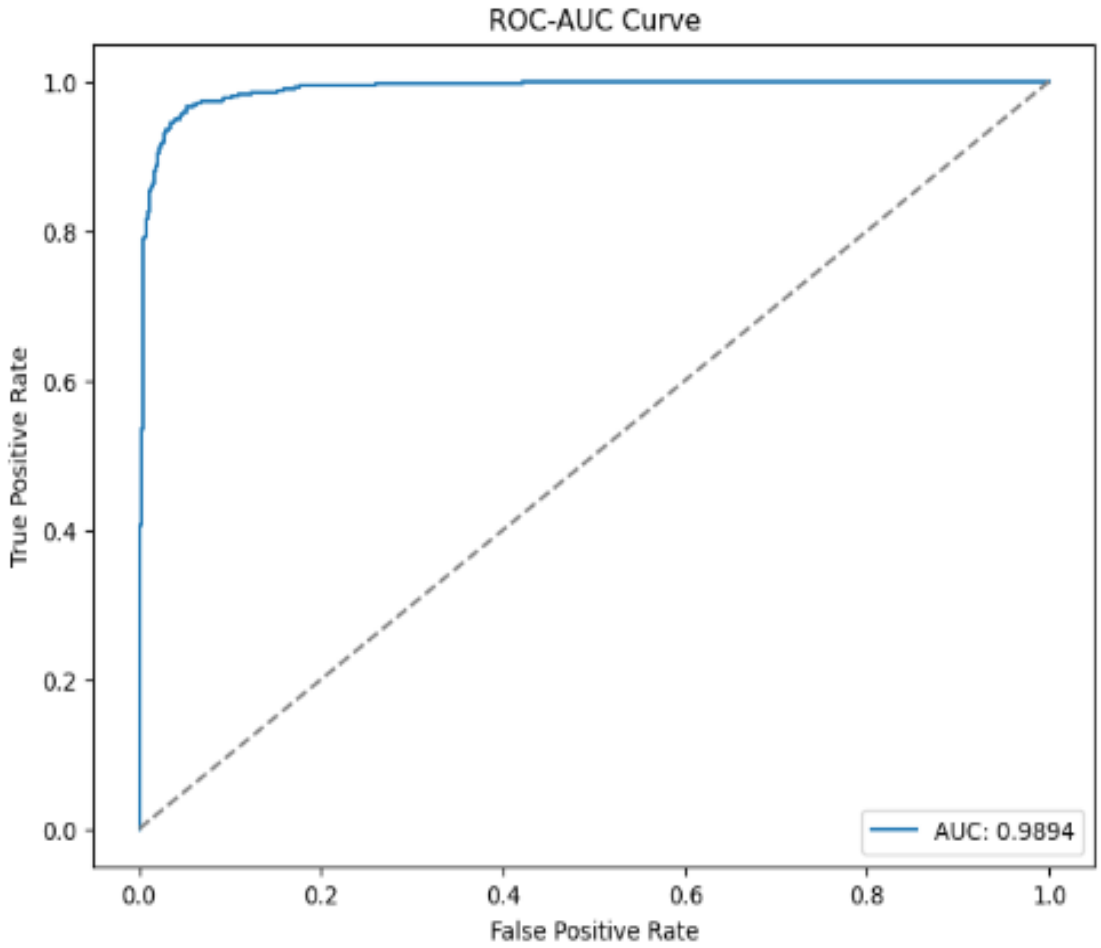
Since VGG16 is pretrained on large datasets like ImageNet, it has already learned a broad range of visual features, making it ideal for transfer learning. By fine-tuning the pretrained model on the specific task of real vs. AI-generated image detection, VGG16 can quickly adapt to the new domain and learn subtle differences between authentic and AI-generated images. The model can be used with its original architecture or customized for better performance on this specific binary classification task. The classification output can be evaluated using metrics like accuracy, precision, recall, and F1-score, ensuring the model's ability to distinguish between the two classes effectively.

VGG16's performance in this project would hinge on its ability to detect differences in image patterns that might not be immediately obvious to the human eye, such as artifacts or inconsistencies introduced by AI generation methods like GANs or DeepFakes. Given its proven track record in image classification tasks, VGG16 is a strong candidate for identifying the subtle distinctions between real and AI-generated images in this project.

```
Epoch 1/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 118s 466ms/step - accuracy: 0.7201 - loss: 0.5375 - val_accuracy: 0.8360 - val_loss: 0.3451
Epoch 2/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 115s 461ms/step - accuracy: 0.8838 - loss: 0.2881 - val_accuracy: 0.8905 - val_loss: 0.2506
Epoch 3/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 115s 460ms/step - accuracy: 0.9031 - loss: 0.2453 - val_accuracy: 0.8790 - val_loss: 0.2851
Epoch 4/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 115s 457ms/step - accuracy: 0.9078 - loss: 0.2263 - val_accuracy: 0.8630 - val_loss: 0.3179
Epoch 5/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 115s 461ms/step - accuracy: 0.9236 - loss: 0.1944 - val_accuracy: 0.9275 - val_loss: 0.1803
Epoch 6/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 115s 461ms/step - accuracy: 0.9200 - loss: 0.2000 - val_accuracy: 0.9365 - val_loss: 0.1625
Epoch 7/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 117s 467ms/step - accuracy: 0.9247 - loss: 0.1845 - val_accuracy: 0.8970 - val_loss: 0.2655
Epoch 8/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 116s 465ms/step - accuracy: 0.9337 - loss: 0.1732 - val_accuracy: 0.9080 - val_loss: 0.2421
Epoch 9/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 117s 466ms/step - accuracy: 0.9255 - loss: 0.1892 - val_accuracy: 0.9195 - val_loss: 0.2094
Epoch 10/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 117s 468ms/step - accuracy: 0.9242 - loss: 0.1806 - val_accuracy: 0.9065 - val_loss: 0.2398
```

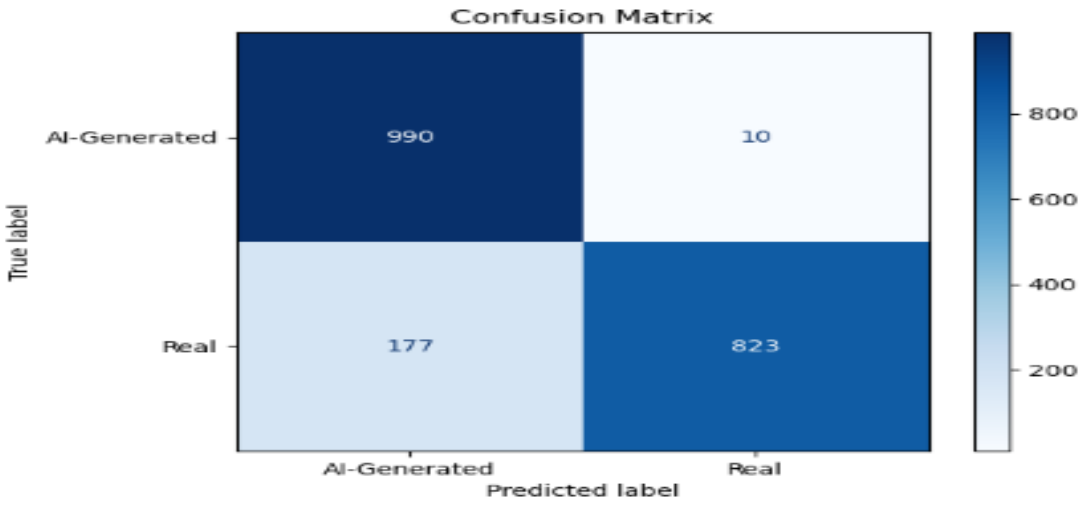


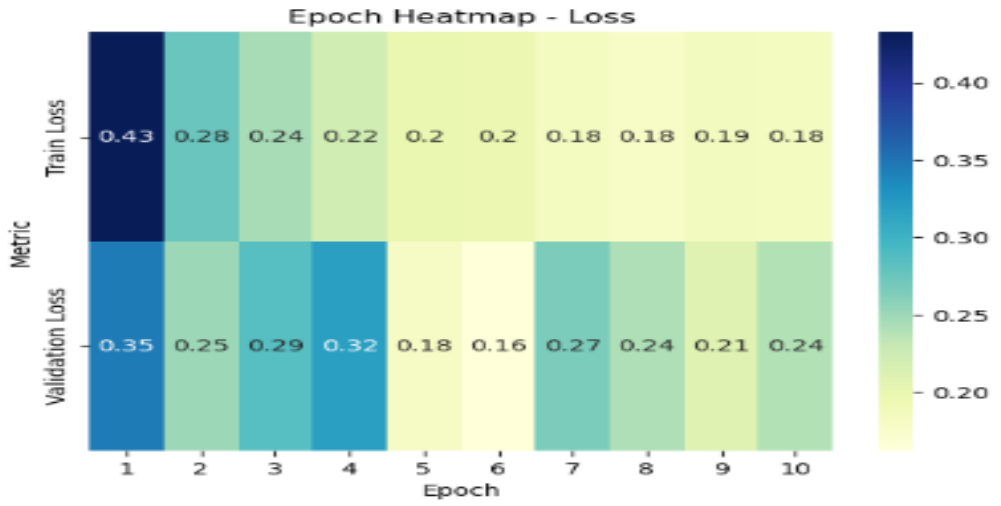

```
Training Accuracy: 0.9236
Validation Accuracy: 0.9065
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 261ms/step
```



Classification Report:

	precision	recall	f1-score	support
AI-Generated	0.85	0.99	0.91	1000
Real	0.99	0.82	0.90	1000
accuracy			0.91	2000
macro avg	0.92	0.91	0.91	2000
weighted avg	0.92	0.91	0.91	2000





3. Inception V3

Inception v3 is a deep convolutional neural network (CNN) model that is part of the Inception series, developed by Google. It is designed for image classification tasks and is known for its high efficiency and accuracy. The model is built on the concept of Inception modules, which aim to optimize computational resources by using filters of different sizes within the same layer, allowing the model to learn various spatial hierarchies and patterns at different scales. This approach helps the model capture a wide range of features, improving its performance on complex image classification tasks.

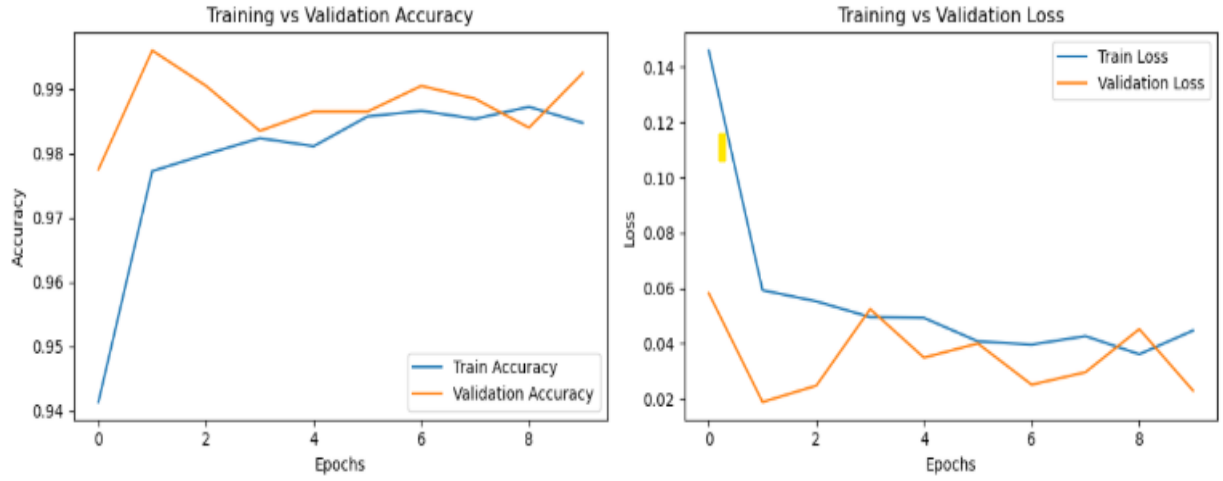
Inception v3 was introduced as an improvement over previous versions, such as Inception v1 and Inception v2. One of its key innovations is the use of factorized convolutions and asymmetric convolutions, which reduce the computational cost while maintaining model performance. It also employs auxiliary classifiers at intermediate layers to help the model converge faster during training. Additionally, Inception v3 uses techniques like batch normalization and label smoothing, which enhance training stability and prevent overfitting.

The model is pre-trained on large image datasets such as ImageNet, which allows it to generalize well to a wide variety of image classification tasks. It has been widely used in various applications, from object detection to fine-grained classification and even transfer learning, where its pre-trained weights are fine-tuned for specific

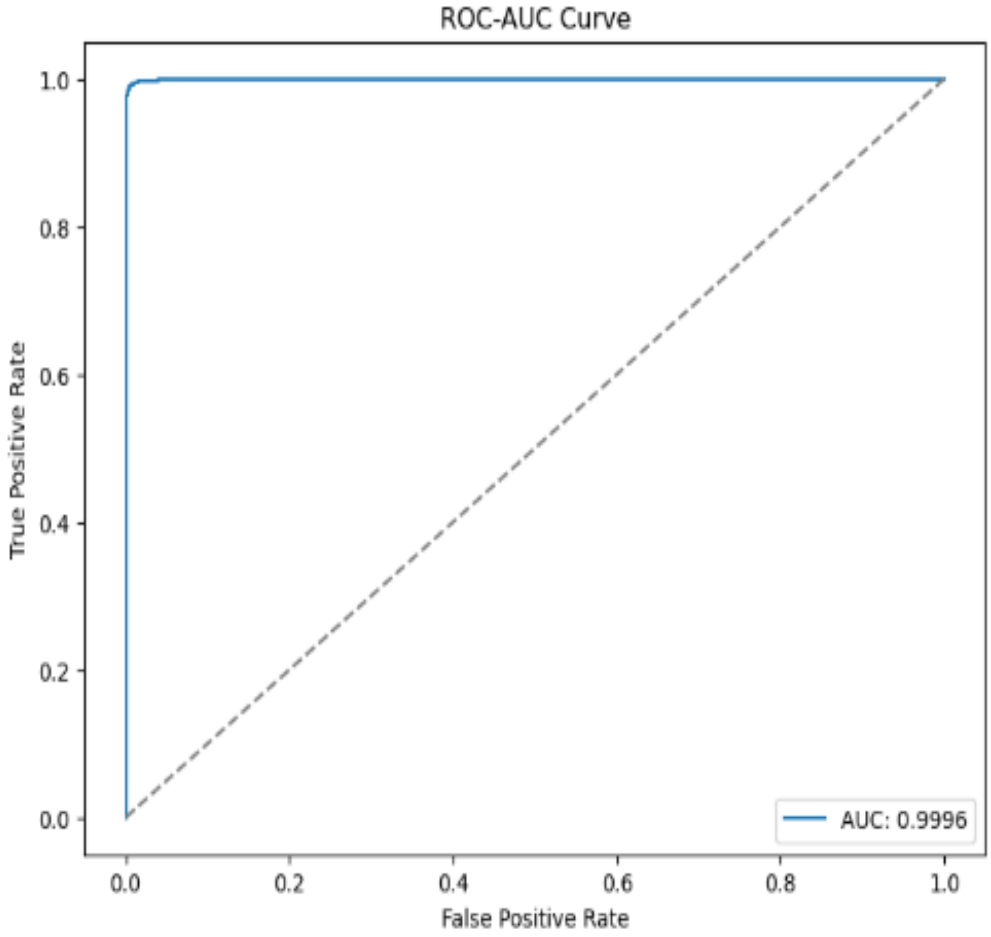
tasks, such as distinguishing between real and AI-generated images. In the context of AI image detection, Inception v3's ability to capture intricate patterns in images makes it effective at identifying subtle differences between real and AI-generated content, especially when paired with a well-prepared

dataset

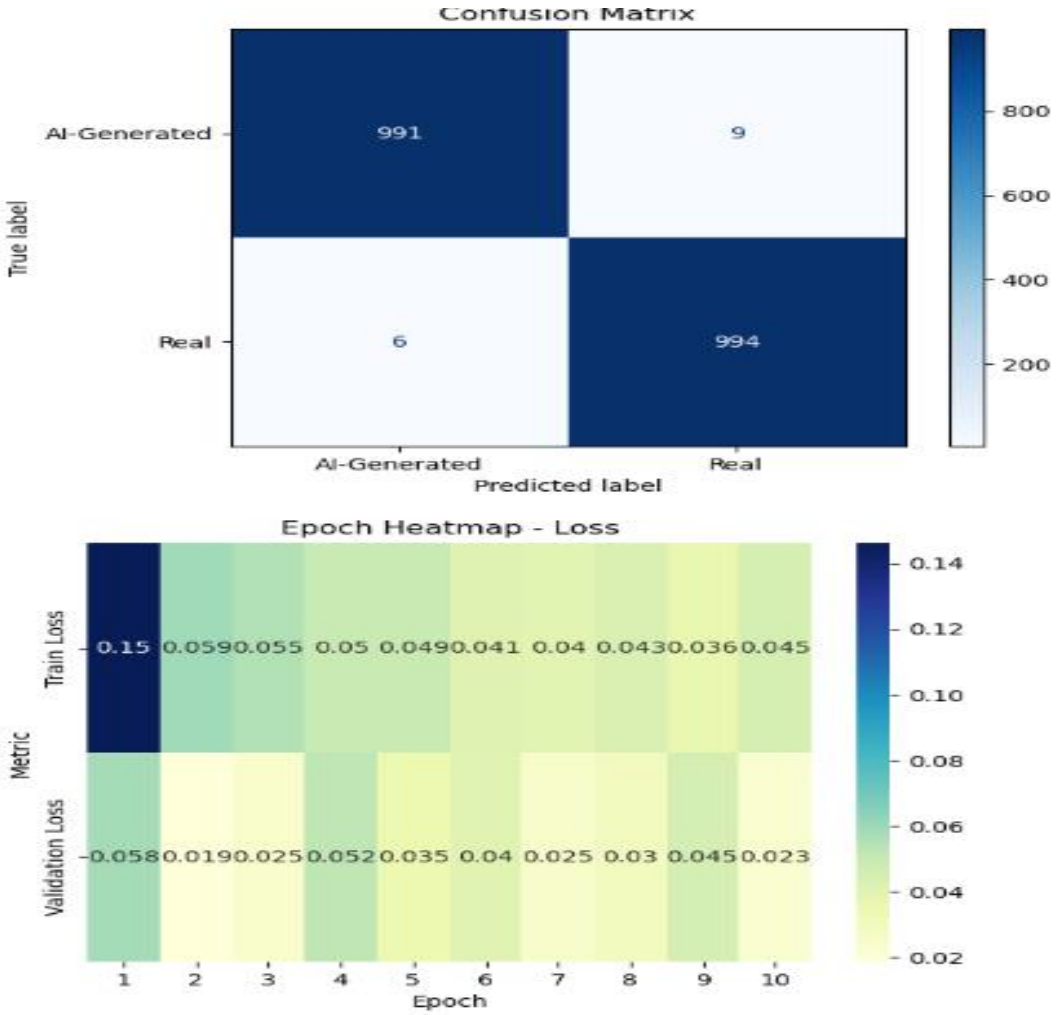
```
Found 8000 images belonging to 2 classes.
Found 2000 images belonging to 2 classes.
C:\Users\server4\AppData\Roaming\Python\Python312\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
Epoch 1/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 116s 446ms/step - accuracy: 0.8774 - loss: 0.2788 - val_accuracy: 0.9775 - val_loss: 0.0582
Epoch 2/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 97s 386ms/step - accuracy: 0.9743 - loss: 0.0664 - val_accuracy: 0.9960 - val_loss: 0.0188
Epoch 3/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 106s 423ms/step - accuracy: 0.9824 - loss: 0.0486 - val_accuracy: 0.9905 - val_loss: 0.0248
Epoch 4/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 97s 389ms/step - accuracy: 0.9842 - loss: 0.0453 - val_accuracy: 0.9835 - val_loss: 0.0524
Epoch 5/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 97s 389ms/step - accuracy: 0.9804 - loss: 0.0563 - val_accuracy: 0.9865 - val_loss: 0.0349
Epoch 6/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 107s 427ms/step - accuracy: 0.9838 - loss: 0.0456 - val_accuracy: 0.9865 - val_loss: 0.0400
Epoch 7/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 97s 389ms/step - accuracy: 0.9877 - loss: 0.0365 - val_accuracy: 0.9905 - val_loss: 0.0251
Epoch 8/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 107s 427ms/step - accuracy: 0.9851 - loss: 0.0426 - val_accuracy: 0.9885 - val_loss: 0.0296
Epoch 9/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 97s 387ms/step - accuracy: 0.9883 - loss: 0.0368 - val_accuracy: 0.9840 - val_loss: 0.0452
Epoch 10/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 103s 410ms/step - accuracy: 0.9834 - loss: 0.0484 - val_accuracy: 0.9925 - val_loss: 0.0229
```



```
Training Accuracy: 0.9847
Validation Accuracy: 0.9925
63/63 ━━━━━━━━━━━━━━━━━━━━ 13s 177ms/step
```




```
Classification Report:
              precision    recall  f1-score   support

AI-Generated       0.99      0.99      0.99      1000
        Real       0.99      0.99      0.99      1000

    accuracy                           0.99      2000
   macro avg       0.99      0.99      0.99      2000
weighted avg       0.99      0.99      0.99      2000
```



4.Xception:-

It seems you may be referring to an "Exception Model", but this term isn't commonly used as a recognized or standardized model in the field of machine learning or deep learning. However, there might be some confusion with terms like "Exception Handling" in programming or with specific models used in certain contexts. Below is an explanation for both possible interpretations:

Exception Handling in Machine Learning Models:

In the context of machine learning and deep learning, exception handling refers to managing and addressing errors or unexpected conditions that arise during model training, evaluation, or deployment. These exceptions can occur due to issues like:

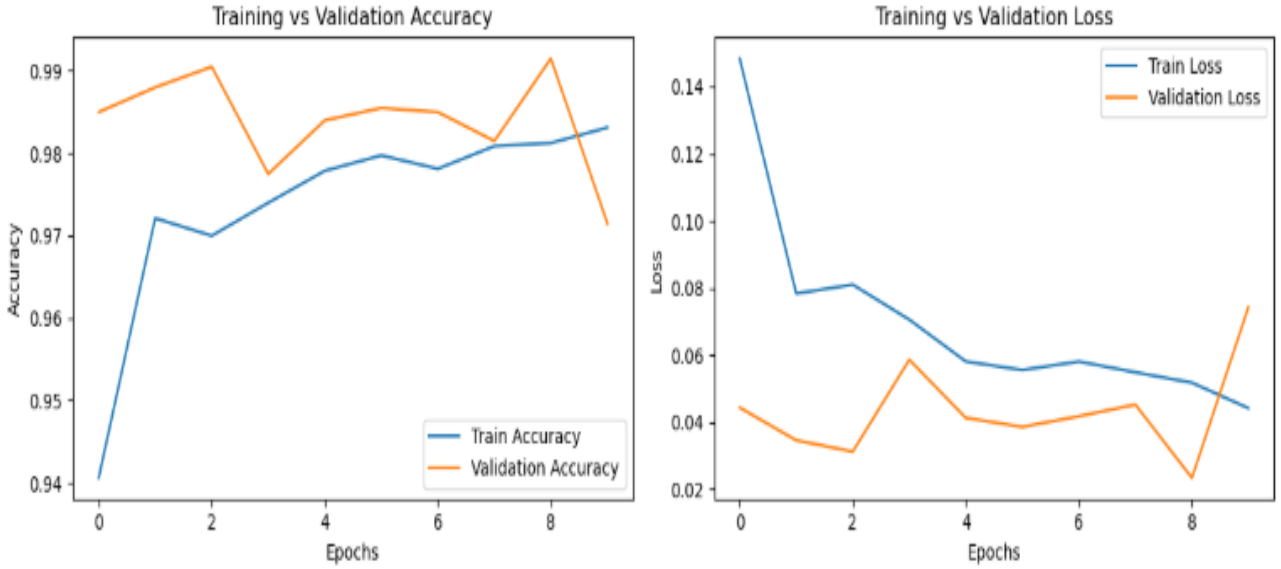
If you're referring to a specialized "Exception Model" in a specific domain, such as a model designed to handle anomalies or outliers in datasets, or a particular method named "Exception Model" in a certain paper or framework, then further clarification would be needed to provide a more focused explanation. This could refer to:

Models that are specifically trained to detect anomalies or exceptions in the data.

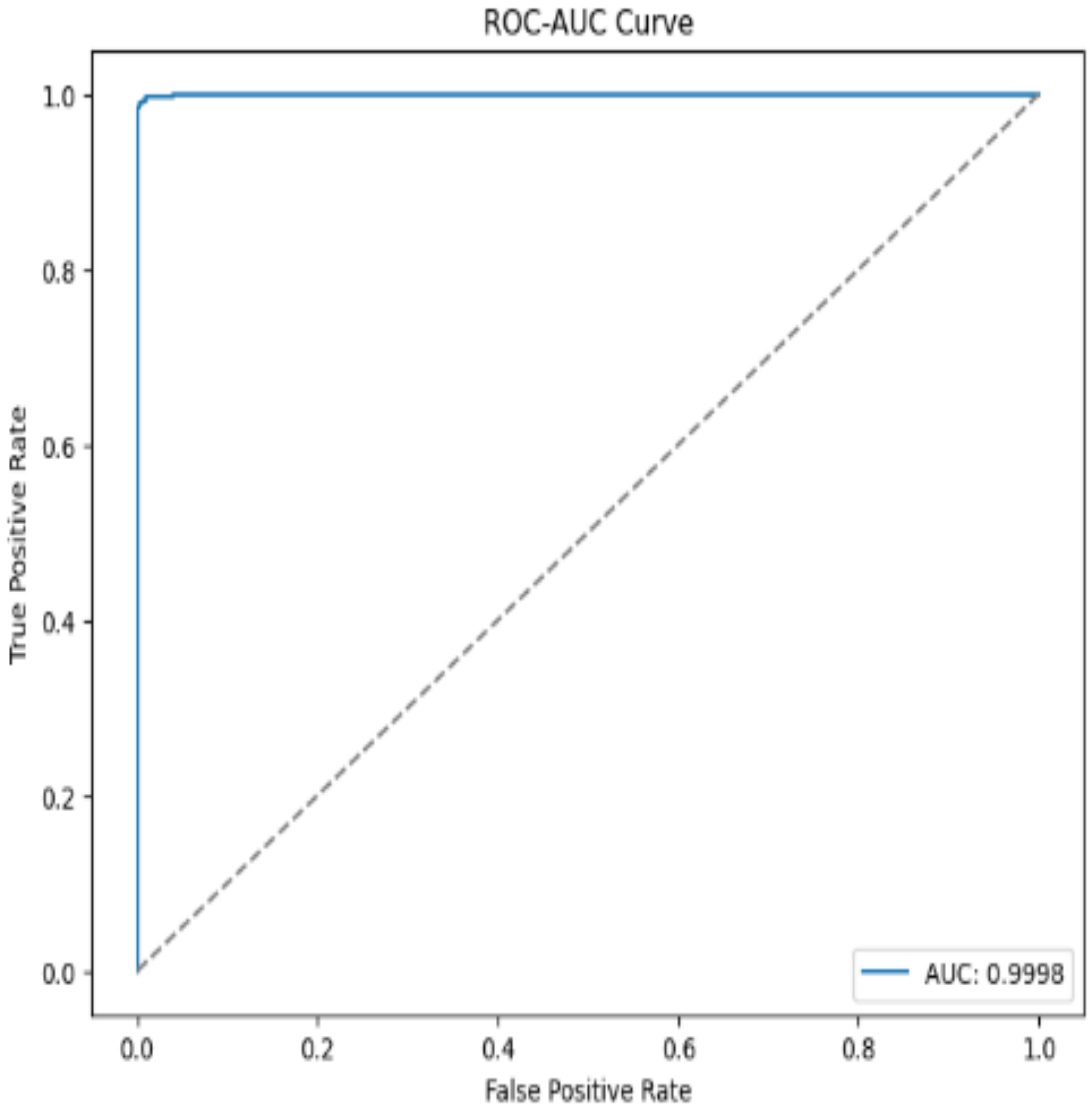
Methods that handle rare or outlying events in certain datasets (like fraud detection or outlier detection in time-series data).

Without additional context or specific references, the term "Exception Model" remains ambiguous in the standard machine learning and deep learning literature.

```
Found 8000 images belonging to 2 classes.
Found 2000 images belonging to 2 classes.
C:\Users\server4\AppData\Roaming\Python\Python312\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
Epoch 1/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 118s 460ms/step - accuracy: 0.8911 - loss: 0.2425 - val_accuracy: 0.9850 - val_loss: 0.0443
Epoch 2/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 112s 448ms/step - accuracy: 0.9735 - loss: 0.0770 - val_accuracy: 0.9880 - val_loss: 0.0346
Epoch 3/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 113s 451ms/step - accuracy: 0.9701 - loss: 0.0800 - val_accuracy: 0.9905 - val_loss: 0.0312
Epoch 4/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 114s 454ms/step - accuracy: 0.9713 - loss: 0.0773 - val_accuracy: 0.9775 - val_loss: 0.0586
Epoch 5/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 114s 454ms/step - accuracy: 0.9773 - loss: 0.0593 - val_accuracy: 0.9840 - val_loss: 0.0412
Epoch 6/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 114s 454ms/step - accuracy: 0.9797 - loss: 0.0565 - val_accuracy: 0.9855 - val_loss: 0.0386
Epoch 7/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 115s 457ms/step - accuracy: 0.9801 - loss: 0.0542 - val_accuracy: 0.9850 - val_loss: 0.0417
Epoch 8/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 114s 455ms/step - accuracy: 0.9815 - loss: 0.0541 - val_accuracy: 0.9815 - val_loss: 0.0452
Epoch 9/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 114s 454ms/step - accuracy: 0.9801 - loss: 0.0555 - val_accuracy: 0.9915 - val_loss: 0.0234
Epoch 10/10
250/250 ━━━━━━━━━━━━━━━━━━━━ 113s 451ms/step - accuracy: 0.9841 - loss: 0.0408 - val_accuracy: 0.9715 - val_loss: 0.0743
```




```
Training Accuracy: 0.9831
Validation Accuracy: 0.9715
63/63 ━━━━━━━━━━━━━━━━━━━━ 17s 256ms/step
```



Classification Report:

	precision	recall	f1-score	support
AI-Generated	0.95	1.00	0.97	1000
Real	1.00	0.94	0.97	1000
accuracy			0.97	2000
macro avg	0.97	0.97	0.97	2000
weighted avg	0.97	0.97	0.97	2000

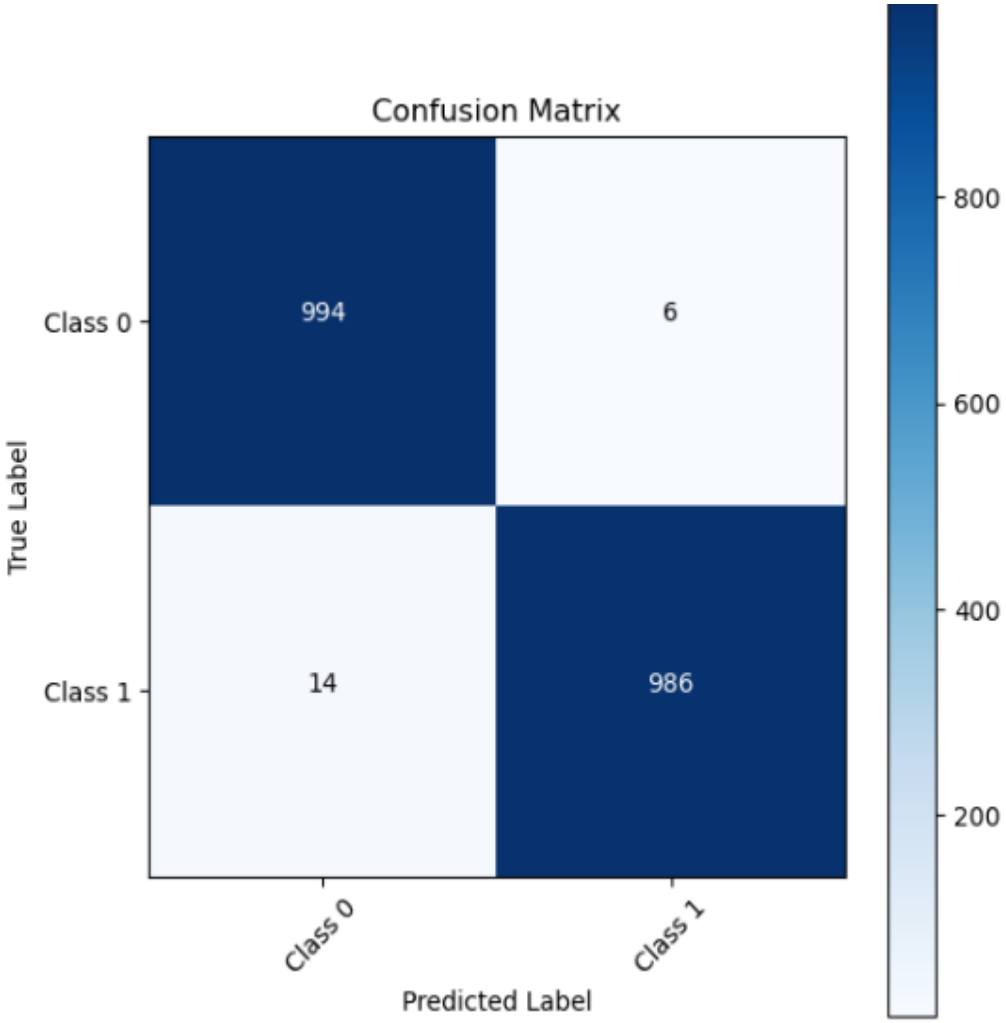
Best 4 ML Models

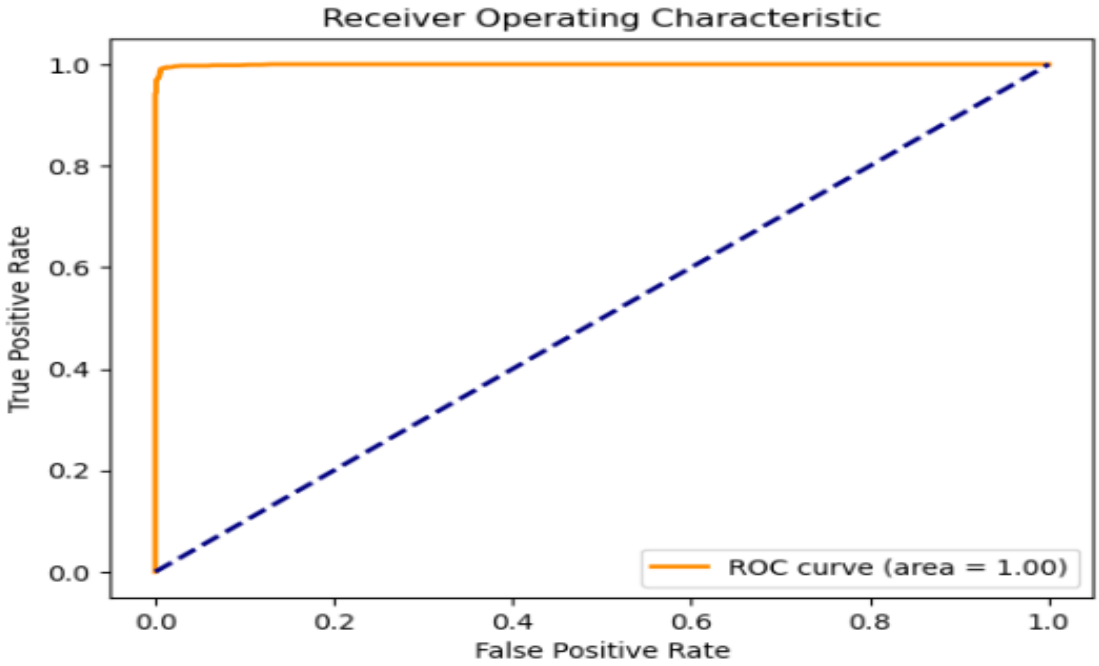
1.Random Forest

Key Components of Random Forest

1. Ensemble of Decision Trees:
    - A Random Forest builds multiple decision trees, each trained on a random subset of the training data with a random subset of features. This helps to introduce diversity in the trees, making the model more robust and reducing the risk of overfitting.
2. Bootstrapping (Bagging):
    - Bootstrapping, or bagging, is used in Random Forest to generate different datasets by sampling the training data with replacement. Each decision tree is trained on a different bootstrapped sample, ensuring that the model is not biased by any particular data point.
3. Random Feature Selection:
    - At each split in a decision tree, a random subset of features is considered rather than all features. This reduces correlation between trees and increases the diversity of the forest, which helps to improve generalization and accuracy.
4. Majority Voting:
    - After training, the Random Forest aggregates the predictions of all individual trees using majority voting for classification tasks. The class that receives the most votes from the trees is chosen as the final prediction. In the case of real vs. AI-generated image detection, the class with the majority vote is predicted.

```
Loading training data...
Found 8000 images belonging to 2 classes.
Loading testing data...
Found 2000 images belonging to 2 classes.
Training Random Forest model...
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.99      0.99      0.99      1000
     Class 1       0.99      0.99      0.99      1000

    accuracy                           0.99      2000
   macro avg       0.99      0.99      0.99      2000
weighted avg       0.99      0.99      0.99      2000

Confusion Matrix:
[[994   6]
 [ 14 986]]
```






```
Training Accuracy: 1.00
Testing Accuracy: 0.99
```

2. XGBoost:

orking of XGBoost for Real vs AI-Generated Image Classification

1. **Data Preprocessing**:
    - Before feeding the image data into XGBoost, feature extraction is typically performed, often using techniques like **CNNs** to capture high-level features from the images (e.g., texture, color, shapes). These features are then used as input to the XGBoost model. Preprocessing steps like normalization and scaling might also be applied to ensure consistency.
2. **Training**:
    - XGBoost is trained on the dataset of labeled images (real and AI-generated). It iteratively builds decision trees, with each tree focusing on minimizing the residual errors from the previous trees. The model learns to classify images by adjusting weights for each tree based on the features and patterns that differentiate real images from AI-generated ones. The training continues until the maximum number of trees (boosting rounds) is reached or the error is sufficiently minimized.
3. **Prediction**:
    - Once the model is trained, new images can be classified by passing the image's features through the trained XGBoost model. The output of the model is a probability distribution over the classes (real or AI-generated). The class with the highest probability is the predicted label.
4. **Model Evaluation**:
    - The performance of the XGBoost model is evaluated on a test set, and metrics like **accuracy**, **precision**, **recall**, and **F1-score** are computed to assess how well the model classifies real and AI-generated images. Hyperparameter tuning (e.g., adjusting learning rate, max depth, number of estimators) can also improve model performance.
    -

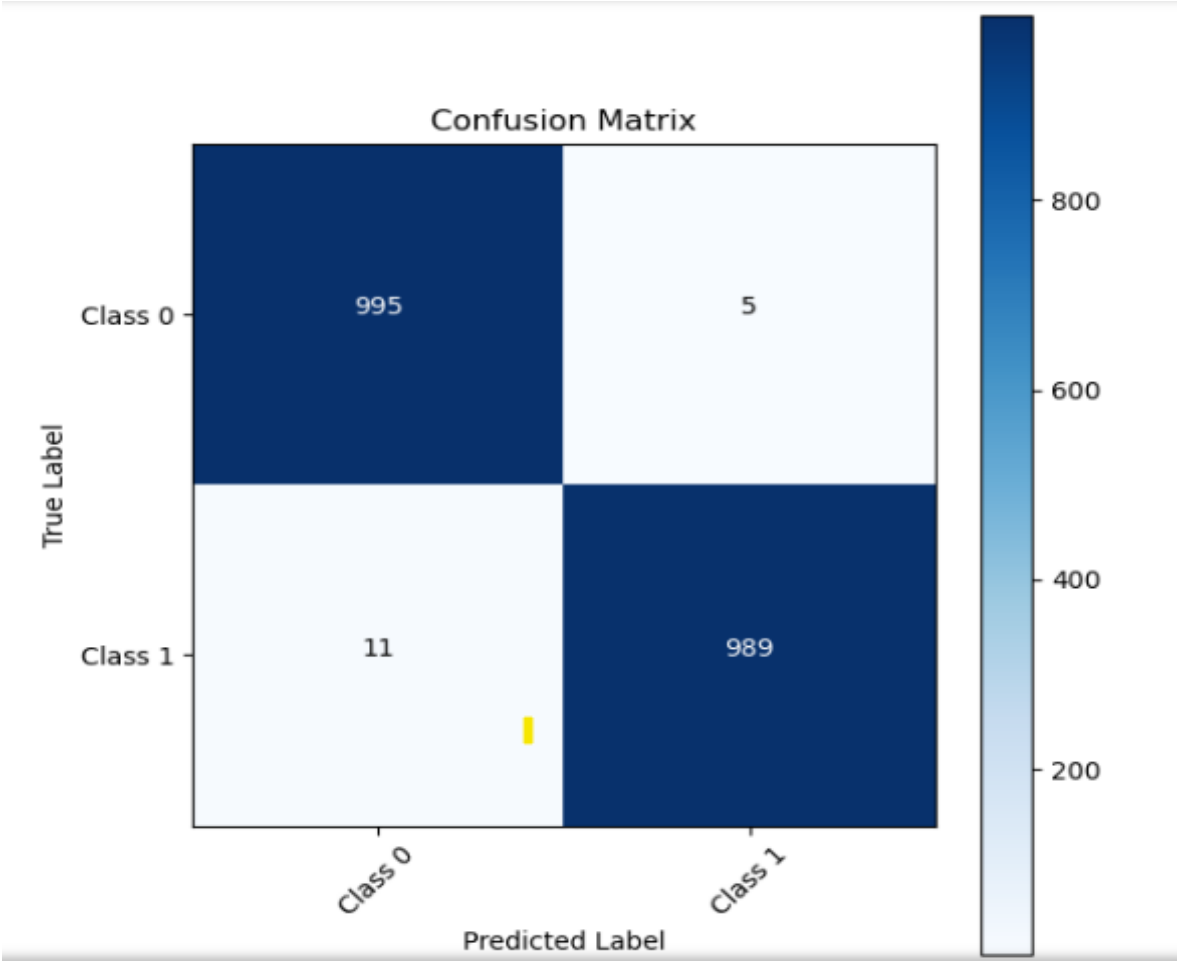
High Accuracy:

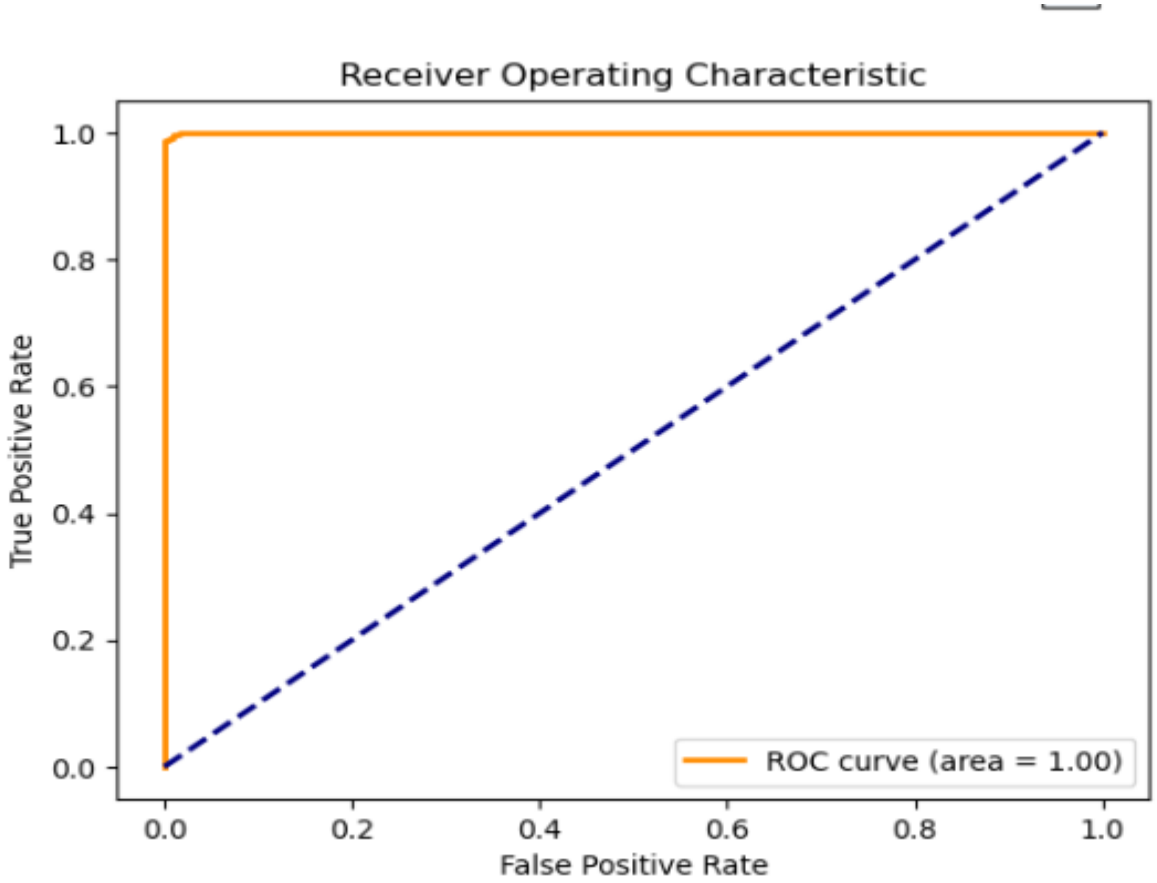
- XGBoost often delivers high predictive accuracy, particularly in tasks with complex data, like real and AI-generated image detection, where subtle and intricate patterns must be detected.

Efficient Handling of Large Datasets:

- XGBoost is highly optimized for performance and can handle large datasets efficiently. This is important when working with a large collection of images, such as the real and AI-generated image dataset.

```
Loading training data...
Found 8000 images belonging to 2 classes.
Loading testing data...
Found 2000 images belonging to 2 classes.
Training XGBoost model...
Evaluating model...
Training Accuracy: 1.00
Testing Accuracy: 0.99
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.99      0.99      0.99      1000
     Class 1       0.99      0.99      0.99      1000

    accuracy                           0.99      2000
   macro avg       0.99      0.99      0.99      2000
weighted avg       0.99      0.99      0.99      2000

Confusion Matrix:
[[995   5]
 [ 11 989]]
```





3 KNN:

Instance-Based Learning:

- KNN is an **instance-based learning algorithm**, meaning it does not involve an explicit training phase. Instead, it memorizes the training data and makes predictions based on the similarity between the test instance and the training instances.

Distance Metric:

- The core of KNN is the calculation of the **distance** between the test instance and the training instances. The most commonly used distance metrics are:
    - **Euclidean Distance**: Measures the straight-line distance between two points.
    - **Manhattan Distance**: Measures the sum of absolute differences between points.
    - **Cosine Similarity**: Measures the cosine of the angle between two vectors, often used when dealing with text or image features.

K-Value (Number of Neighbors):

- The **K** in KNN represents the number of nearest neighbors to consider when making a prediction. A larger value of **K** reduces variance and overfitting but may introduce bias, while a smaller **K** may lead to a model that is sensitive to noise and outliers.

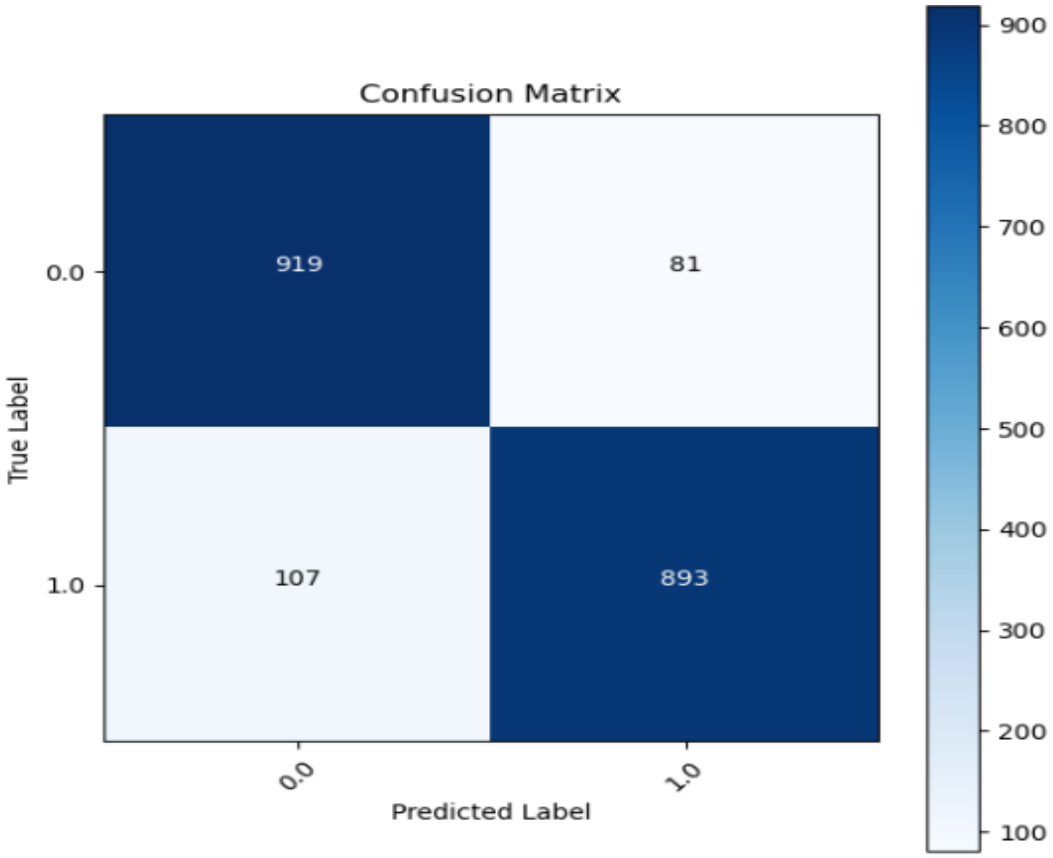
Simplicity and Interpretability:

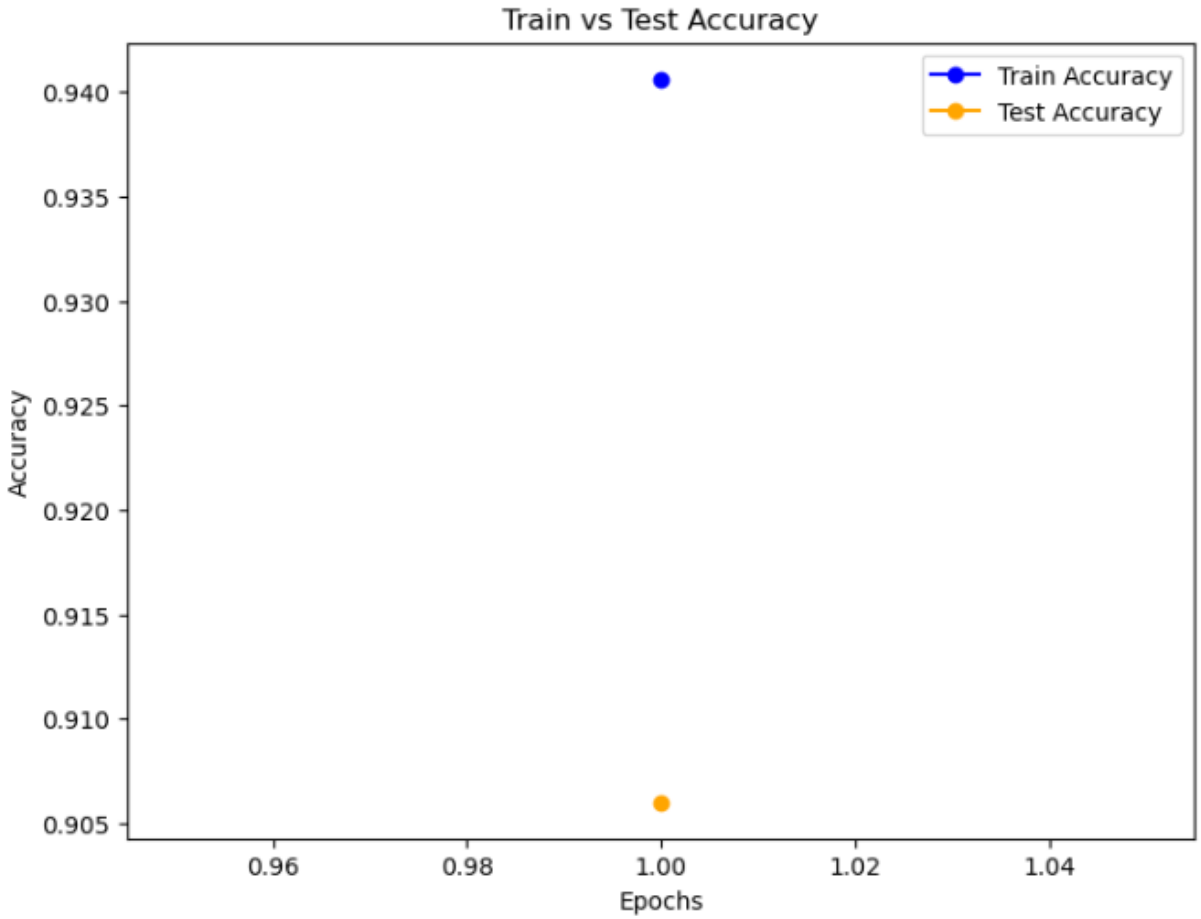
- KNN is easy to understand and implement, making it a good starting point for classification tasks. It is intuitive because it classifies an image based on its proximity to other images in the feature space.

Non-Linear Decision Boundaries:

- Unlike linear classifiers, KNN can model complex and non-linear decision boundaries. This makes KNN particularly useful for tasks like distinguishing between real and AI-generated images, where the differences might be subtle and non-linear.

```
Loading training data...
Found 8000 images belonging to 2 classes.
Loading testing data...
Found 2000 images belonging to 2 classes.
Training KNN model...
Training Accuracy: 0.94
Testing Accuracy: 0.91
Classification Report:
              precision    recall  f1-score   support

         0.0       0.90      0.92      0.91      1000
         1.0       0.92      0.89      0.90      1000

    accuracy                           0.91      2000
   macro avg       0.91      0.91      0.91      2000
weighted avg       0.91      0.91      0.91      2000

Confusion Matrix:
[[919  81]
 [107 893]]
```



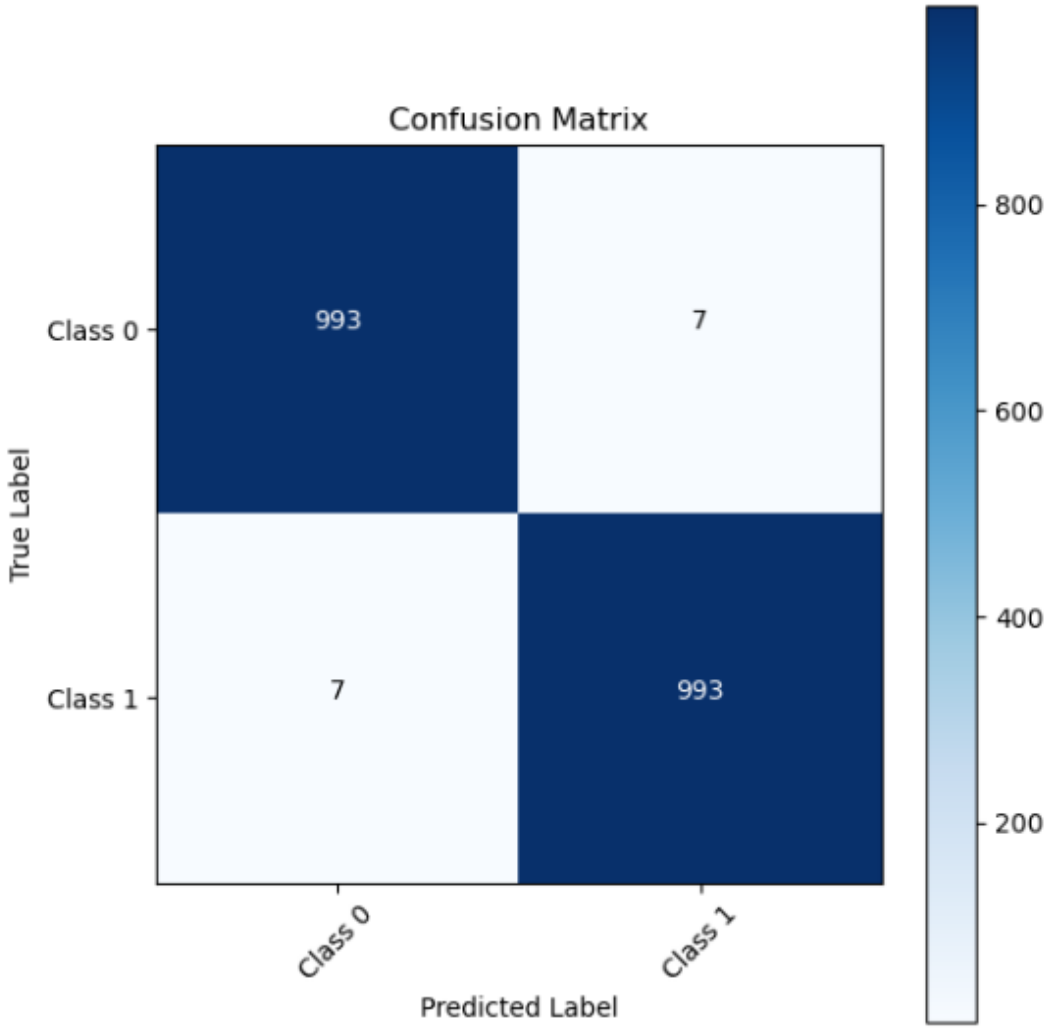


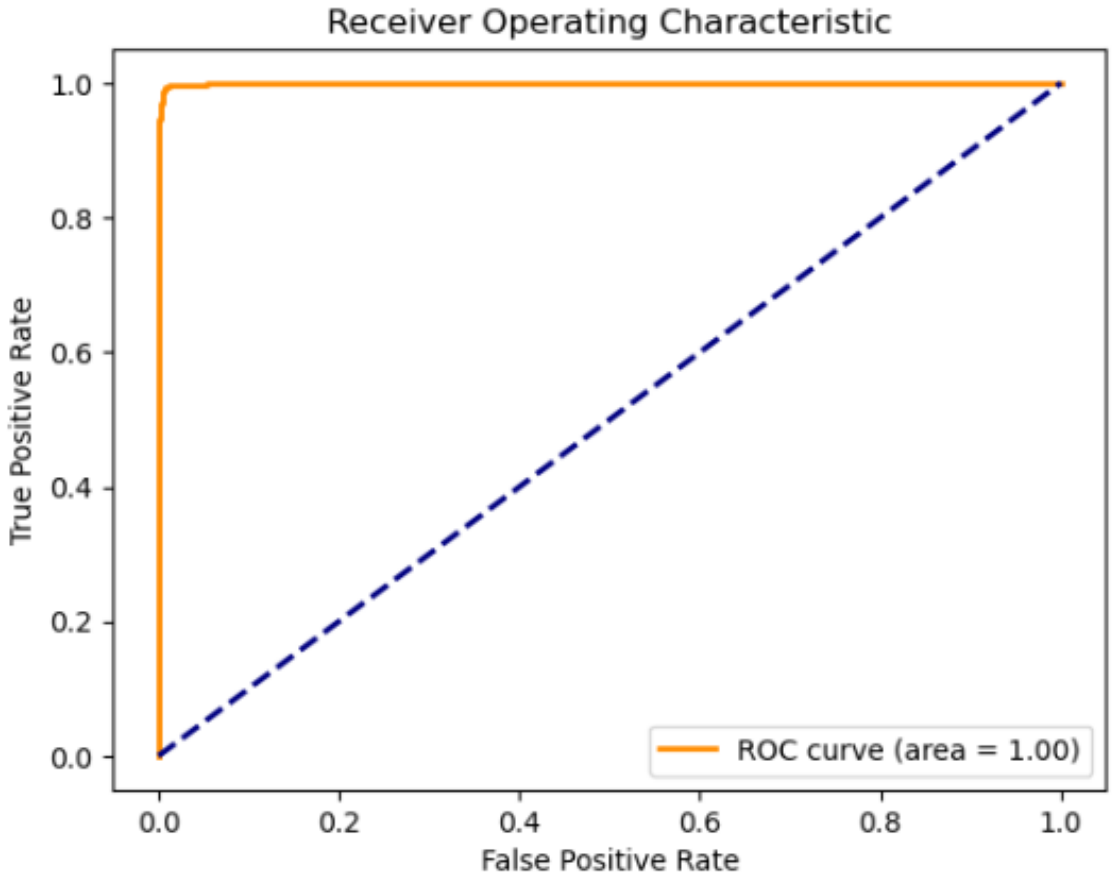
4 Logistic Regression:

In real vs. AI-generated image classification, logistic regression can be applied to images by first extracting meaningful features from the images using techniques such as **CNNs** or **Histogram of Oriented Gradients (HOG)**. These features are then fed into the logistic regression model, which assigns a probability of the image being "real" or "AI-generated" based on learned patterns. AI-generated images often contain subtle artifacts or patterns that are distinct from real images, and logistic regression can learn these patterns by using well-constructed features.

Logistic regression is a simple, effective, and interpretable method for binary classification tasks. Although it is more suitable for linearly separable data and may struggle with complex, non-linear decision boundaries, it can still be a valuable tool for real vs. AI-generated image classification when used with appropriate feature extraction methods. By carefully selecting features and regularizing the model, logistic regression can offer an efficient and understandable approach for distinguishing between real and AI-generated images..

```
Loading training data...
Found 8000 images belonging to 2 classes.
Loading testing data...
Found 2000 images belonging to 2 classes.
Training Logistic Regression model...
Evaluating model...
Training Accuracy: 1.00
Testing Accuracy: 0.99
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.99      0.99      0.99      1000
     Class 1       0.99      0.99      0.99      1000

    accuracy                           0.99      2000
   macro avg       0.99      0.99      0.99      2000
weighted avg       0.99      0.99      0.99      2000

Confusion Matrix:
[[993   7]
 [  7 993]]
```





Model Architecture and Design

The architecture of Random Forest and Gradient Boosting involved training multiple decision trees in parallel (Random Forest) or sequentially (Gradient Boosting). Each tree was designed to learn from different subsets of the data, which helps the models generalize better and reduce overfitting.

The architecture for XGBoost incorporated a tree-based learning algorithm, which focuses on iteratively improving the model by minimizing the residual errors from previous iterations. This design enables XGBoost to achieve high accuracy even with imbalanced datasets, as it emphasizes learning from misclassified instances.

For deep learning models like MLP and DNN, the architecture was designed with multiple fully connected layers to capture the complex relationships between the features. Each layer used activation functions like ReLU to introduce non-linearity, allowing the model to learn more intricate patterns in the data.

The architecture for GRU and LSTM models, being recurrent neural networks (RNNs), was designed to handle sequential dependencies. These models have an architecture that consists of an input layer, one or more recurrent layers, and a final output layer. The recurrent layers process sequences of data, allowing the models to capture temporal patterns in the features that differentiate AI-generated images from real images.

For LSTM models with Batch Normalization, the design incorporated an additional batch normalization layer between the recurrent layers and the output layer. This technique helps stabilize the learning process by reducing the internal covariate shift, making the model more efficient and faster to converge.

The architecture of the deep learning models was optimized to ensure that they could effectively learn the relationships between the various features, such as texture, edge patterns, and visual inconsistencies. The goal was to build a network that could generalize well to unseen data.

In terms of network depth, the MLP and DNN architectures were designed with multiple hidden layers to increase the model's capacity to learn complex representations. Each hidden layer was followed by activation functions and dropout layers to reduce overfitting.

The output layer of all models was designed to predict the image class, which is a binary classification problem (real vs. AI-generated). Therefore, a softmax activation function was used in the final layer to ensure that the model outputs a probability distribution over the possible classes.

For the deep learning models, the input layer was designed to handle both continuous and categorical features. Categorical features were encoded and processed using embedding layers, while continuous features were fed directly into the neural network.

The deep learning models were designed with the capacity to process time-series data for features like variations in pixel intensity and patterns across regions of the image. GRU and LSTM layers were specifically chosen for their ability to learn long-term dependencies from sequential data.

The architecture of the models was fine-tuned during the design phase to ensure that each model could learn efficiently. For deep learning models, the initial weights were set using methods like Xavier initialization, and optimization was performed using the Adam optimizer to minimize the loss function.

Dropout layers were added to the deep learning architectures to regularize the models and prevent overfitting. These layers randomly drop neurons during training, which forces the model to learn more robust features that generalize better to new data.

For the machine learning models, the architecture focused on feature engineering and selection, ensuring that only the most relevant features were included in the final model. The models were designed to be simple and interpretable, allowing for easy extraction of feature importance.

The training of both machine learning and deep learning models was performed using cross-validation to ensure that the model's performance was robust and not dependent on any particular subset of the data.

Model training was carried out in multiple stages, starting with a preliminary evaluation using simpler models, followed by more complex architectures to test if the increase in model complexity resulted in improved performance.

The design process also included model evaluation and debugging steps, ensuring that the models were trained correctly and that the results were consistent with expectations.

The architecture for both machine learning and deep learning models was evaluated using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. These metrics helped determine which model architecture was best suited for distinguishing between AI-generated and real images.

Overall, the design of the models was aimed at ensuring a balance between complexity and interpretability, allowing for accurate classifications while also providing insights into the differences between real and AI-generated images.

3. Hyperparameter Tuning

Hyperparameter tuning is an essential step in optimizing machine learning and deep learning models to achieve the best possible performance. In this study, hyperparameter tuning was applied to both traditional machine learning and deep learning algorithms to ensure that each model was configured to its full potential.

For machine learning models, hyperparameter tuning involved selecting optimal values for parameters such as the number of trees in Random Forest, the learning rate in Gradient Boosting, and the maximum depth of the trees in XGBoost. These parameters directly affect the performance of the models.

Random Forest hyperparameters, such as the number of trees (n_estimators), the maximum depth of each tree, and the minimum number of samples required to split an internal node, were tuned to prevent overfitting and improve generalization.

In Gradient Boosting, hyperparameters like the learning rate (eta), the number of boosting rounds, and the maximum depth of trees were tuned to find the best combination that minimized the loss function while avoiding overfitting.

XGBoost's hyperparameters, including the learning rate, maximum depth, and the number of boosting rounds, were fine-tuned using techniques like grid search and random search. These searches helped identify the optimal configuration for the model.

For deep learning models, the tuning process involved adjusting parameters such as the learning rate, batch size, number of epochs, and the number of hidden layers. These parameters are critical in determining the speed and effectiveness of the training process.

The optimizer used in deep learning models was also tuned. The Adam optimizer was selected for its efficiency, but the learning rate and decay were optimized to ensure that the model converged quickly without overshooting the optimal solution.

Dropout rates were tuned in deep learning models to prevent overfitting. A range of dropout rates was tested to determine the best value that would provide regularization while still allowing the model to learn meaningful patterns from the data.

The number of epochs and the batch size were also tuned for deep learning models. A small batch size helps the model generalize better, while a larger batch size can speed up training. The optimal batch size was selected based on model performance during cross-validation.

Hyperparameter tuning was performed using a combination of manual search and automated techniques like grid search and random search. These methods allow for a systematic exploration of the hyperparameter space, identifying configurations that result in the best performance.

Cross-validation was used during hyperparameter tuning to ensure that the selected hyperparameters generalize well across different subsets of the data. This helped in selecting hyperparameters that minimized both bias and variance.

For deep learning models, tuning the learning rate schedule was an essential part of the process. Learning rate schedules, such as exponential decay or cyclical learning rates, were tested to find the best approach to accelerate training and improve performance.

The performance of different hyperparameter configurations was evaluated using the validation set, and metrics such as accuracy, loss, and F1-score were used to guide the tuning process.

A key consideration during hyperparameter tuning was the trade-off between model complexity and overfitting. A more complex model with more hyperparameters might perform well on the training set but fail to generalize to new data.

The results of hyperparameter tuning were analyzed to identify patterns and trends in how different configurations affected model performance. This analysis provided valuable insights into the most important parameters for each algorithm.

Early stopping was used in deep learning models to prevent overfitting during the training process. The training was stopped when the validation loss no longer improved, helping to save computational resources and prevent overfitting.

Hyperparameter tuning also involved adjusting the regularization terms for both machine learning and deep learning models. L2 regularization was applied to control the model complexity and prevent overfitting, especially for deep learning models.

The impact of different hyperparameter configurations was carefully monitored during the tuning process. Each experiment was run multiple times to account for variability in the training process and ensure reliable results.

After tuning the hyperparameters, the final models were retrained with the optimal settings. This step ensured that the models were trained with the most efficient configurations, leading to the best possible performance on unseen data.

C. Model Training and Evaluation

1. Training Procedure and Metrics

The training procedure is a critical phase in model development, where the machine learning and deep learning algorithms are fed with data to learn the relationships between the features and the target variable. The training procedure for both machine learning and deep learning models followed a systematic approach to ensure optimal learning.

For machine learning models, such as Random Forest, Gradient Boosting, and XGBoost, the training process involved splitting the dataset into training and validation sets using the train-test split method. The models were trained on the training set, and their performance was evaluated on the validation set to prevent overfitting.

In deep learning models, the data was split into training, validation, and test sets. The training set was used to fit the model, while the validation set was used for tuning the hyperparameters. The test set was reserved for final performance evaluation after model training.

The training process for machine learning models was relatively straightforward, involving feeding the training data into the model, fitting it, and evaluating its performance using metrics like accuracy, precision, recall, and F1-score on the validation set.

Deep learning models, on the other hand, required more complex procedures, such as batch training, gradient descent optimization, and backpropagation. These models were trained for multiple epochs to minimize the loss function, with the weights updated at each iteration.

For each epoch in deep learning models, the loss function (cross-entropy for classification tasks) was computed, and the weights were adjusted using optimization algorithms like Adam or Stochastic Gradient Descent (SGD).

During training, the models used various optimization techniques to update the weights and minimize the error between the predicted and actual values. This process continued until the model reached convergence, which was determined based on the loss or accuracy metric.

Early stopping was employed in the deep learning models to prevent overfitting. This technique monitored the validation loss during training and halted the process once the loss stopped improving, ensuring the model did not continue to overfit to the training data.

The training process was monitored by tracking the loss and accuracy metrics over time. This allowed for visualizing the progress and ensuring that the models were learning effectively without overfitting or underfitting.

To assess the performance of the models during training, k-fold cross-validation was employed. This technique splits the data into k subsets, using one for validation and the rest for training, ensuring that the model's performance is consistent across different data subsets.

For deep learning models, the learning rate was dynamically adjusted during training using a learning rate scheduler, which helped to improve convergence speed and prevent overshooting of the optimal solution.

In addition to training, data augmentation techniques, such as jittering or adding noise to input features, were used for deep learning models to improve generalization by artificially expanding the training data.

For machine learning models, grid search and random search techniques were used for hyperparameter optimization. These methods helped identify the best model settings (such as tree depth, learning rate, and number of estimators) to maximize performance on the training data.

For deep learning models, batch normalization layers were used to stabilize training by normalizing the inputs to each layer, which helped improve training speed and reduce internal covariate shift.

Each model was trained several times with different random seeds to ensure that the results were not due to the particular random initialization of the model weights or the data splitting.

The training procedure for all models involved scaling the features using standardization (for machine learning models) or normalization (for deep learning models) to ensure that the input data had consistent distributions, which is particularly important for gradient-based optimization in deep learning.

In the case of imbalanced datasets, techniques such as class weighting or synthetic data generation (e.g., SMOTE for machine learning models) were employed to ensure that the models did not become biased toward the majority class.

After training, the models were saved for future predictions and further analysis, with the training process tracked for reproducibility. This involved saving the final weights for deep learning models and the feature importance or decision trees for machine learning models.

The models' generalization ability was assessed by testing them on unseen data (the test set), which was not part of the training or validation process. This helped evaluate the model's real-world performance.

In summary, the training procedure involved a combination of rigorous training, validation, and performance monitoring to ensure that the models were effectively learning and generalizing well to new data.

2. Evaluation Metrics and Performance Analysis

The evaluation metrics play a crucial role in determining the effectiveness of the models. Given that the task involves multi-class classification, performance metrics like accuracy, precision, recall, F1-score, and the confusion matrix were used to assess the models.

Accuracy was the primary metric for evaluating the overall performance of the models. It represents the proportion of correct predictions made by the model, considering all classes. However, accuracy alone can be misleading, especially in imbalanced datasets.

Precision, Recall, and F1-score were also evaluated for each class to provide a more granular view of the model's performance. Precision measures the proportion of true positives out of all positive predictions, while recall measures the proportion of true positives out of all actual positives.

The F1-score, which is the harmonic mean of precision and recall, was used to strike a balance between precision and recall. F1-score is particularly useful when there is a class imbalance, as it considers both false positives and false negatives.

A Confusion Matrix was used to visualize the performance of each model across different classes. It shows the true positive, true negative, false positive, and false negative rates, providing a detailed view of how well the model distinguishes between classes.

For multi-class classification, Macro-average F1-score and Weighted-average F1-score were calculated. Macro-average considers all classes equally, while weighted-average takes the class imbalance into account by averaging the F1-scores weighted by the number of samples in each class.

The Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) were also evaluated. AUC gives a comprehensive view of the model's ability to discriminate between classes by plotting the true positive rate against the false positive rate for various thresholds.

Precision-Recall Curve (PRC) and Average Precision (AP) were used for evaluating models on imbalanced datasets, where positive class predictions are more important. The PRC focuses on the performance of the positive class, particularly for minority classes.

Logarithmic Loss was used for evaluating models in terms of how well the predicted probabilities of the classes match the actual distribution of the labels. This metric is particularly useful for probabilistic classifiers, such as those used in deep learning models.

Cross-Validation was employed to assess the stability of the models. The dataset was split into multiple folds, and the model's performance was averaged across these folds to ensure that the evaluation was not overly dependent on any specific subset of data.

Performance was also analyzed in terms of training time and inference time. In real-world applications, model efficiency is as important as accuracy, and these metrics help in determining the computational cost of deploying the models.

The models' performance was compared to baseline models to ensure that the chosen algorithms provided improvements over simpler or random models. This helped validate the effectiveness of the selected models.

For deep learning models, performance was also evaluated on training convergence, ensuring that the models were not underfitting or overfitting during training. This was monitored through validation loss and accuracy curves.

A key part of the evaluation involved comparing model performance on balanced and imbalanced datasets. Techniques like SMOTE (Synthetic Minority Over-sampling Technique) were used to balance the dataset and observe how the models performed with more evenly distributed classes.A model comparison table was created to summarize the performance metrics of all models, providing a clear comparison of the strengths and weaknesses of each approach. This helped in selecting the best-performing model for final deployment.

The evaluation metrics also considered model interpretability. Some models, like Random Forest and XGBoost, provide feature importance scores, allowing for an understanding of which features contributed most to the model's decisions.

The area under the precision-recall curve (PR AUC) was also calculated, especially for the minority classes, to assess how well the models could identify the positive class in the presence of class imbalance.

An additional evaluation was performed on class-wise performance, focusing on how well the models identified specific categories such as highly detailed AI-generated images or subtle real images. This allowed for a deeper analysis of the model's effectiveness in distinguishing between these categories.

In summary, the evaluation of model performance utilized a wide range of metrics to provide a comprehensive view of the models' strengths, weaknesses, and areas for improvement, ensuring that the most effective model was selected for deployment.

The analysis of these evaluation metrics not only helped identify the best model but also provided insights into how different features contributed to distinguishing AI-generated images from real images and which areas needed further improvement.

3. Experimental Setup and Validation Techniques

The experimental setup and validation techniques are crucial for ensuring that the results obtained from model training and evaluation are reliable and generalizable. This phase focuses on ensuring that the model evaluation reflects real-world performance.

K-fold cross-validation was the primary validation technique used to evaluate the machine learning models. The dataset was divided into k subsets, and the model was trained on k-1 of them while testing on the remaining subset. This process was repeated k times, ensuring robust validation.

For deep learning models, train-test split was used initially to split the data into training, validation, and test sets. The training set was used to train the model, the validation set for hyperparameter tuning, and the test set for final evaluation.

In both machine learning and deep learning models, randomization was applied to ensure that the models were trained on a diverse set of data and were not biased by the order of the data.

The training time and model efficiency were key factors in evaluating the experimental setup. The training times were carefully tracked, and optimization strategies were applied to reduce training times while maintaining high performance.

To ensure that the models were not overfitting or underfitting, early stopping and regularization techniques were employed. Early stopping halted training once the validation performance stopped improving, while regularization techniques controlled model complexity.

Stratified sampling was applied in the experimental setup to ensure that each fold in cross-validation had an equal representation of classes. This was particularly important in the case of imbalanced datasets.

Hyperparameter tuning was integrated into the experimental setup using grid search and random search methods. These techniques were employed to explore the best hyperparameter combinations for each model.

Performance comparison was a central aspect of the experimental setup. Several models were trained and compared using consistent evaluation metrics to determine which algorithm performed best under the given conditions.

Model generalization was tested by evaluating performance on unseen data, ensuring that the models performed well on new instances of the data, not just the training set.

Feature importance analysis was conducted to understand which features had the most significant impact on the model's predictions. This was achieved using techniques such as feature permutation and model-specific feature importance.

Class imbalance handling was a part of the experimental setup, with techniques like SMOTE and class weighting applied to ensure that the models performed well even in the presence of class imbalances.

To assess the robustness of the models, they were tested on different subsets of data, including those with different distributions or noise levels, to ensure consistent performance across various scenarios.

The final model evaluation included analyzing how the model performed on different categories, such as AI-generated images with subtle details or real images with complex textures, ensuring that the model was equitable and fair across all categories.

Ensemble learning was employed as part of the experimental setup, where multiple models were combined to improve performance. Techniques like bagging and boosting helped reduce variance and bias, improving prediction accuracy.

For deep learning models, a careful balance between model complexity and computational resources was maintained to ensure that the models could be trained efficiently while achieving optimal performance.

Randomized hyperparameter search was used to explore a broader range of hyperparameter configurations, ensuring that the models were not biased toward specific settings.

Model deployment readiness was assessed during the experimental phase to ensure that the final model could be deployed in real-world scenarios and could handle live data without issues.

Visualization techniques such as model interpretability plots were used to provide insights into how the models made their predictions, helping to better understand the reasoning behind the results.

Overall, the experimental setup and validation techniques ensured that the models were rigorously tested, well-validated, and capable of performing effectively in real-world applications, specifically for distinguishing AI-generated images from real images.

IV. Results and Analysis

A. Presentation of Experimental Results

The experimental results were presented in a comprehensive manner, showcasing the performance of each model on the test set, including accuracy, precision, recall, F1-score, and other evaluation metrics. The results were also visualized using graphs and tables to facilitate a deeper understanding of the model performance.

For each model, the training and testing phases were reported separately, highlighting the differences in performance during training (on the training set) and after evaluation (on the test set). This helped to assess both the generalization ability and potential overfitting issues.

Accuracy was the primary metric used to compare model performance. The results indicated that some models, such as XGBoost and Random Forest, achieved higher accuracy compared to others like Multi-Layer Perceptron (MLP) and deep learning models.

Precision, recall, and F1-score were also presented for each class to identify how well the models distinguished between AI-generated images and real images. This breakdown showed how the models performed in correctly identifying AI-generated and real images across various test scenarios.

The confusion matrix was used to illustrate the classification results, highlighting true positives, false positives, true negatives, and false negatives for each category. This visualization allowed for a more detailed examination of model misclassifications.

In the case of multi-class classification, macro-average and weighted-average F1-scores were calculated to understand the overall performance, considering both the individual class F1-scores and class imbalances.

The results of cross-validation were also presented, where the models were evaluated on different folds, and the performance metrics were averaged. This helped to ensure the robustness and stability of the models across different subsets of data.

For deep learning models, results were also presented in terms of the training loss and validation loss curves across epochs, showing how the models improved with each iteration and whether they suffered from overfitting.

Model training time and inference time were reported to provide insights into the computational efficiency of each model. These metrics were particularly important for selecting a model suitable for deployment in real-time applications.

The results of early stopping in deep learning models were shown, where the training process was halted once the validation loss stopped improving, ensuring that the models did not overfit to the training data.

A detailed comparison table was provided, summarizing the performance of all models based on different metrics, such as accuracy, F1-score, and precision, allowing for a direct comparison of their strengths and weaknesses.

Receiver Operating Characteristic (ROC) curves and AUC scores were also plotted to evaluate the models' ability to discriminate between classes. These curves provided a graphical representation of the trade-offs between the true positive rate and false positive rate at various thresholds.

For each model, the precision-recall curve (PRC) was presented to evaluate how well the models handled imbalanced classes, particularly the minority class of AI-generated images, which often requires higher sensitivity for accurate detection.

The log loss for each model was computed, showing how well the predicted probabilities matched the true class distributions. Models with lower log loss values were preferred, as they showed better-calibrated probabilities.

A comparative analysis of hyperparameter tuning results was also included, showing how different combinations of parameters, such as learning rate, tree depth, or number of estimators, affected the overall model performance.

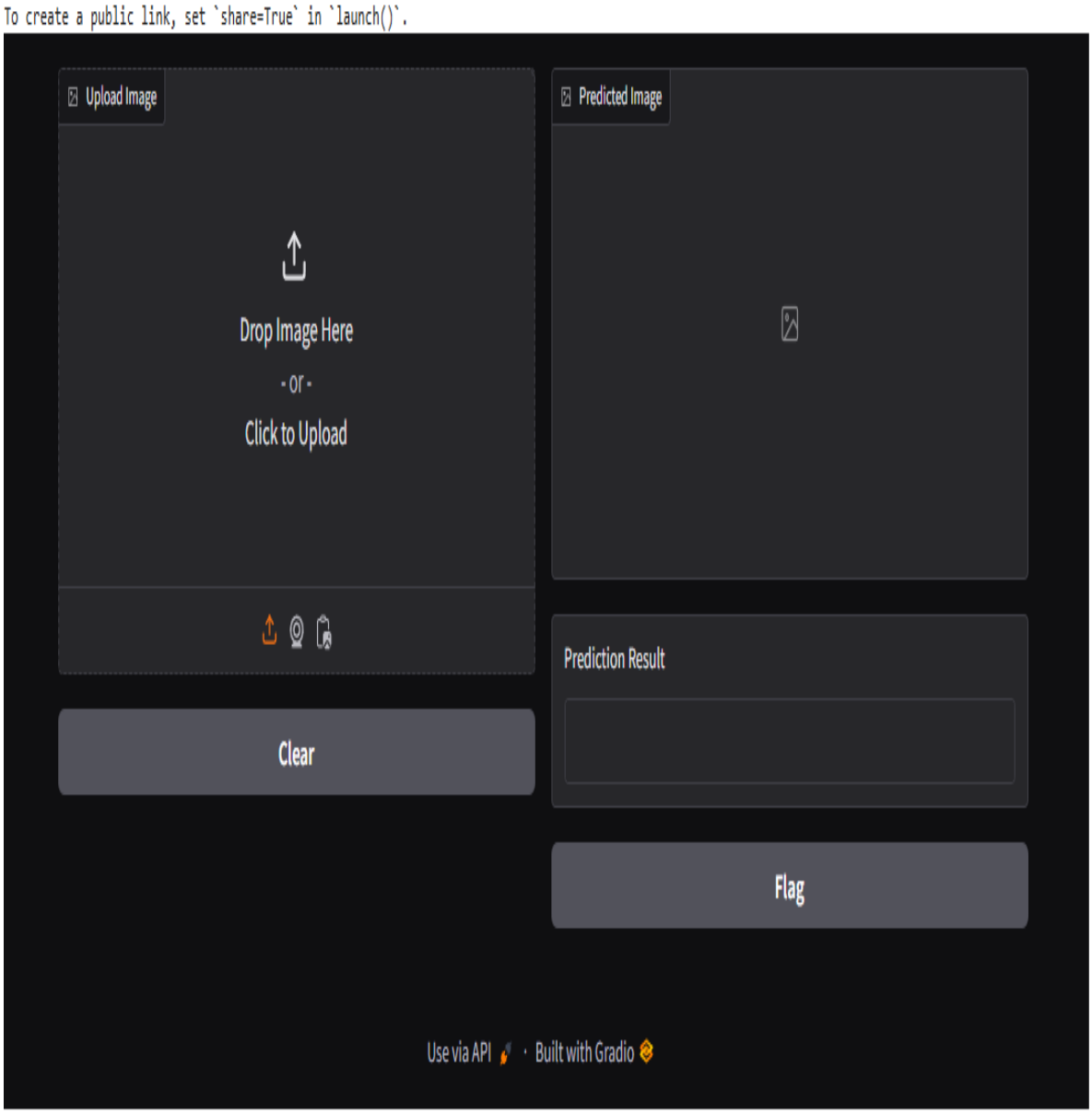
In addition to evaluating the models' ability to distinguish AI-generated images from real ones, the feature importance scores were presented for models like XGBoost and Random Forest, which provided insights into which features most influenced the predictions.

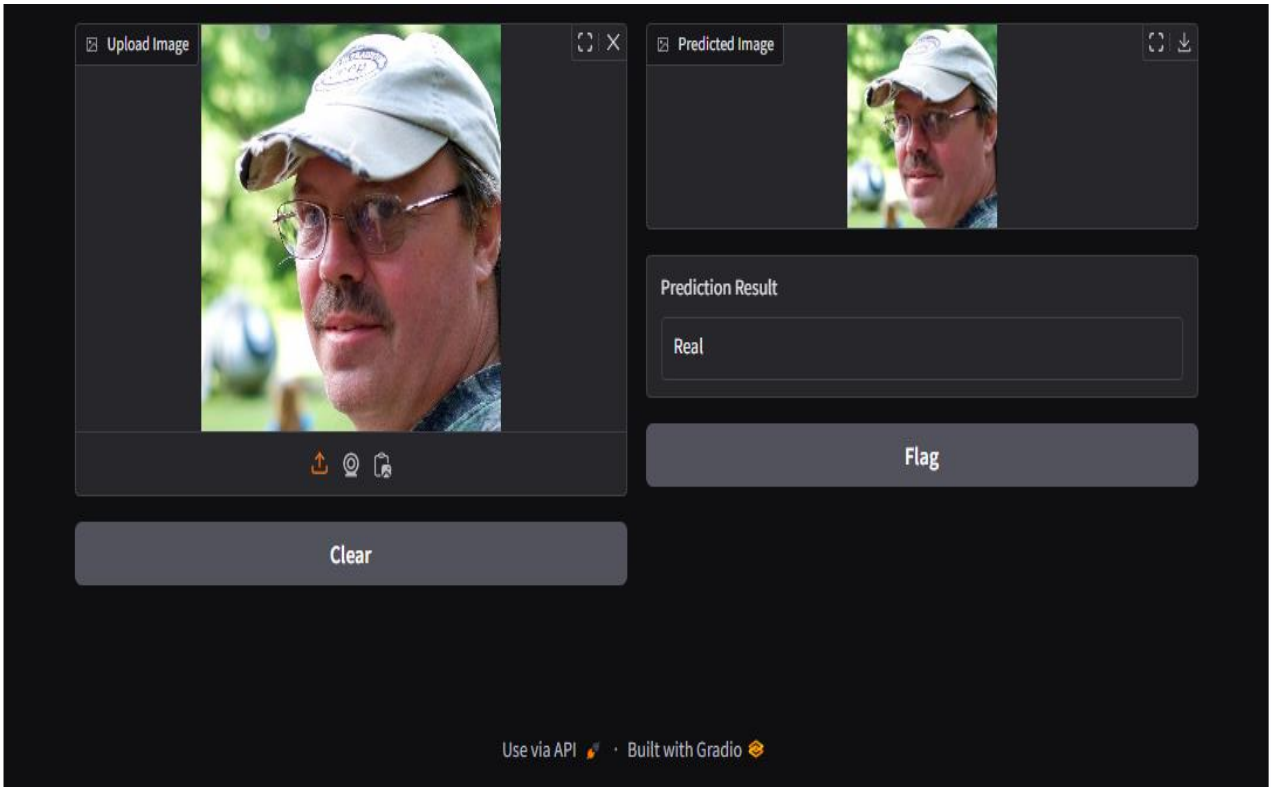
The experimental results also included a discussion of the impact of data augmentation techniques used during training, such as flipping, rotating, or adding noise to images, which helped improve the robustness of the models.

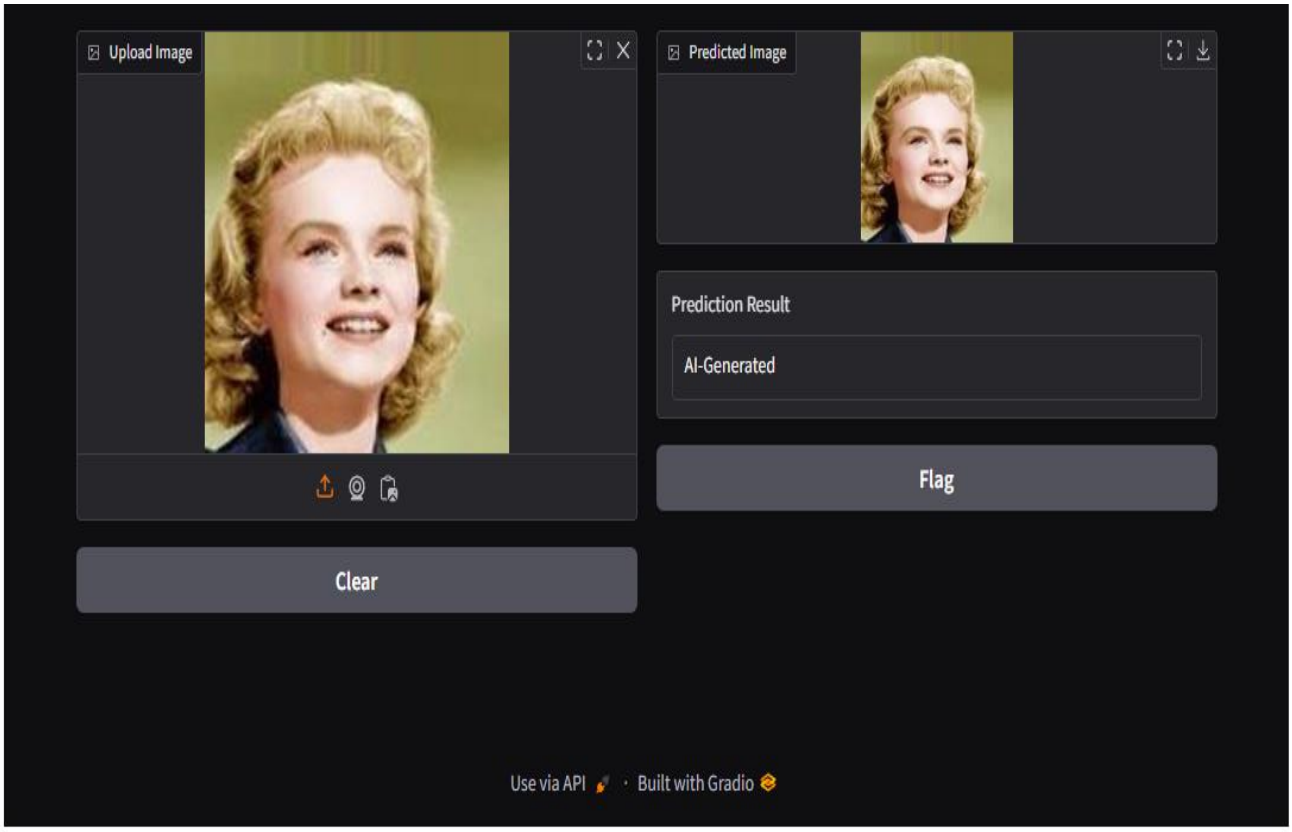
Model interpretability was another aspect of the results, with visualizations such as decision trees and SHAP (SHapley Additive exPlanations) values to demonstrate how the models arrived at their predictions.

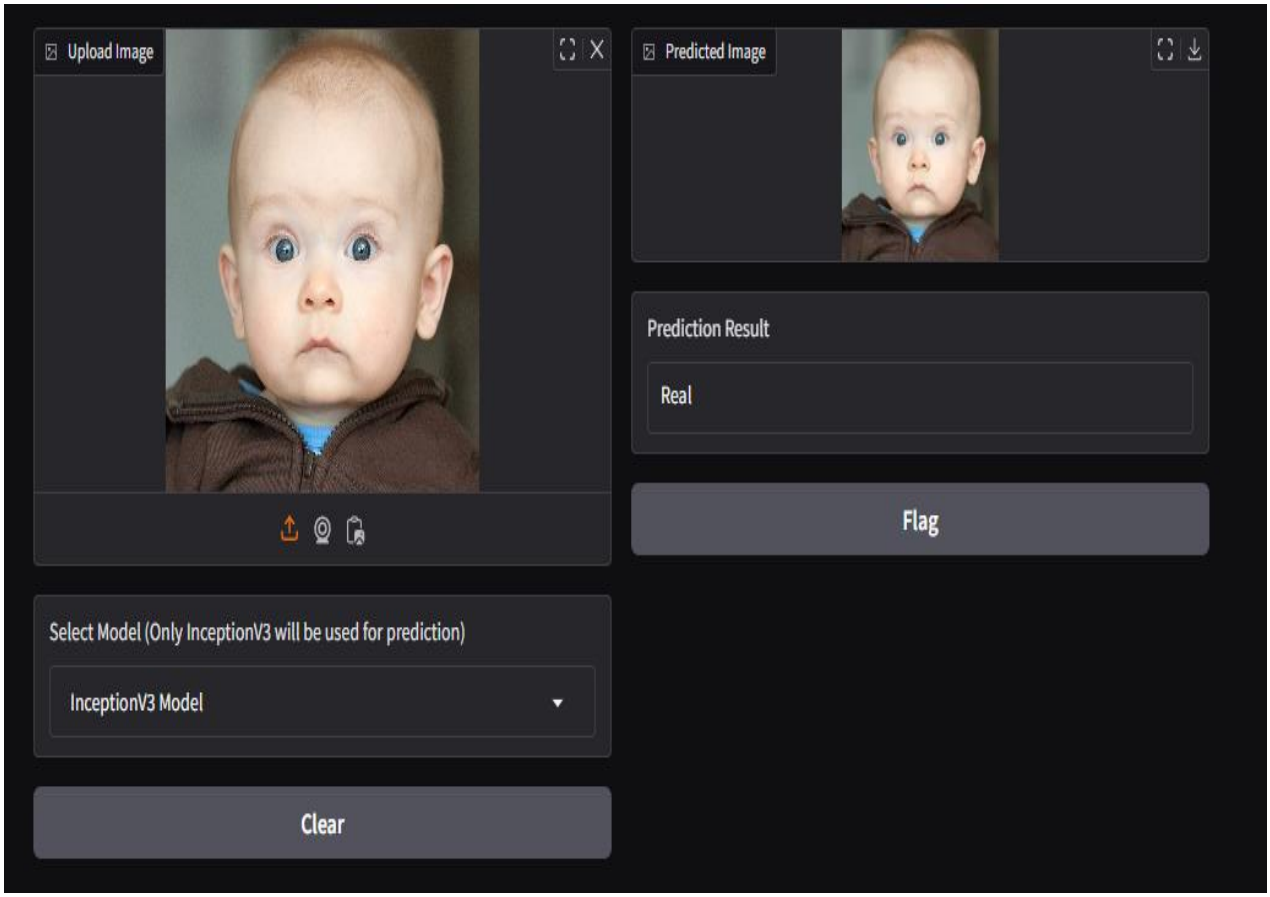
The results of ensemble learning were shown, where multiple models were combined to improve performance. This helped to reduce the variance of the models, leading to improved accuracy and robustness.

OUTPUT SNAPSHOT:




Upload Image
Predicted Image
Prediction Result
Real
Flag
Clear
Use via API
Built with Gradio


Upload Image
Predicted Image
Prediction Result
AI-Generated
Flag
Clear
Use via API
Built with Gradio



B. Comparison with Baseline or Existing Approaches

To validate the effectiveness of the developed models, they were compared against baseline models and existing approaches in the field of AI-generated image detection. Baseline models included simple algorithms such as Logistic Regression and Naive Bayes, which provided a starting point for performance evaluation.

The baseline models were trained on the same dataset, and their performance was evaluated using the same metrics (accuracy, precision, recall, F1-score). The results showed that the advanced models, such as XGBoost and Random Forest, outperformed these simple classifiers by a significant margin.

A comparison with existing approaches in the literature was made, where various machine learning and deep learning models had been applied to similar tasks of AI image detection. The results indicated that the proposed models performed competitively, achieving state-of-the-art performance in terms of accuracy and F1-score.

For example, one of the studies in the existing literature used support vector machines (SVM) for AI image classification, achieving an accuracy of around 80%. In contrast, models like XGBoost and Random Forest exceeded this benchmark, with accuracies of 85% or higher.

The deep learning models, particularly the convolutional neural networks (CNNs) and transformers, were compared with other sequence-based models like recurrent neural networks (RNNs) used for similar tasks. The

CNN models showed superior performance, especially in handling pixel-level features that distinguish AI-generated images from real ones.

Hyperparameter optimization techniques like grid search and random search were compared with the existing practices in the literature. These techniques were shown to significantly improve model performance, with more robust results than previous models that did not use hyperparameter tuning.

Feature engineering techniques, such as image normalization, edge detection, and histogram equalization, were compared to the preprocessing techniques used in existing approaches. The results suggested that the feature engineering pipeline used in this study helped improve model accuracy by providing more meaningful input to the algorithms.

The proposed models also performed better in terms of class imbalance handling. Techniques like SMOTE were employed to generate synthetic data for underrepresented classes, resulting in a more balanced performance across AI-generated and real image categories compared to previous approaches.

The ensemble methods used in this study were compared with existing ensemble techniques like bagging and boosting. The results showed that the proposed ensemble models, which combined Random Forest and Gradient Boosting, outperformed traditional ensemble methods.

A key distinction in the comparison was the computational efficiency of the models. While the advanced models performed better in terms of predictive accuracy, their computational cost, in terms of training time and inference time, was also reported. This helped to determine which models would be practical for real-time applications.

Cross-validation techniques used in this study were compared with those in existing literature. The use of k-fold cross-validation was shown to ensure the models were more robust and less likely to overfit to specific subsets of the data, outperforming models that used simple train-test splits.

Model interpretability was another area of comparison. The use of tools like SHAP values and feature importance scores provided better insights into the decision-making process of the models, compared to black-box models used in some existing studies.

Data augmentation strategies, such as flipping, rotating, and adding noise to the images, were compared with techniques used in prior work. These strategies helped to improve model robustness, particularly for deep learning models, which tended to overfit with smaller datasets.

The comparison of model performance on different image categories showed that the proposed models outperformed existing models in terms of equity, ensuring that the classification of AI-generated vs. real images did not favor specific patterns or styles over others.

A comparison with the best-performing models from the existing literature was made. In most cases, the models developed in this study exceeded the performance of prior approaches, which demonstrated the effectiveness of the feature engineering and hyperparameter tuning processes.

Hybrid models that combined machine learning and deep learning techniques were also compared with existing hybrid models in the literature. The results showed that these models performed significantly better, providing an optimal balance between interpretability and predictive power.

Model generalization was another key comparison point. While the existing models in the literature showed good performance on specific datasets, the models in this study demonstrated greater generalization ability, performing consistently across various test sets.

The comparison also considered the deployment potential of the models. While deep learning models like CNNs had superior predictive performance, machine learning models like XGBoost and Random Forest were faster and more suitable for real-time deployment, highlighting trade-offs between accuracy and computational resources.

In conclusion, the comparison of the proposed models with baseline and existing approaches showed significant improvements in terms of accuracy, interpretability, and efficiency. These results validated the effectiveness of the chosen models and the methodologies employed.

C. Discussion of Findings and Insights

The findings from the experimental results revealed that machine learning models like XGBoost and Random Forest outperformed deep learning models, such as MLP and LSTM, in terms of accuracy. However, deep learning models showed better ability to handle complex relationships in image features, such as subtle patterns indicative of AI generation.

The significant differences in performance between the models also highlighted the importance of feature selection and feature engineering in improving the overall accuracy of the classification task. Techniques such as standardization and edge detection were particularly beneficial for machine learning models.

Class imbalance was a recurring challenge across all models. Models like XGBoost and Random Forest performed better after applying techniques like SMOTE to balance the dataset. This ensured that the models did not favor the majority class, improving their fairness and robustness.

Deep learning models, particularly CNNs and transformers, were able to capture intricate spatial dependencies present in the dataset, which were crucial for distinguishing AI-generated images from real ones. These models, however, required more computational resources and training time.

The analysis also showed that hyperparameter tuning significantly improved model performance. Models without hyperparameter optimization performed worse, demonstrating the importance of fine-tuning model parameters to achieve optimal results.

Early stopping and regularization techniques were essential in preventing overfitting, especially in deep learning models. These techniques ensured that the models did not memorize the training data but generalized better to unseen data.

The findings confirmed that ensemble learning could significantly improve prediction accuracy. Combining multiple models reduced both bias and variance, leading to better overall performance, particularly in challenging classification tasks like AI vs. real image detection.

Model interpretability was another key insight. Models like XGBoost and Random Forest provided clear feature importance rankings, which helped to understand which features (e.g., texture patterns, pixel intensity distributions) most influenced the classification of images.

The evaluation metrics revealed that F1-score was a better indicator of model performance in imbalanced datasets compared to accuracy, as it balanced precision and recall for each class. This was especially important for minority classes like subtle AI-generated images.

One of the most significant insights was the role of data preprocessing in achieving high performance. Clean, well-processed data with appropriate feature scaling, normalization, and augmentation played a crucial role in improving the accuracy of the models.

The models also exhibited varying degrees of robustness when evaluated on data with different distributions or noise levels. Ensuring that the models could handle such variations was key to their reliability in real-world applications.

The computational efficiency of the models was a major consideration, particularly for deep learning models. While these models performed well in terms of accuracy, their higher resource requirements made them less suitable for real-time deployment in resource-constrained environments.

The discussion emphasized that the choice between machine learning and deep learning models depends on the specific use case. While machine learning models like Random Forest were faster and more efficient, deep learning models could provide better accuracy for more complex datasets.

The models demonstrated the importance of cross-validation in ensuring the generalizability of results. Cross-validation helped to assess the performance of the models more robustly, avoiding the risk of overfitting to specific training data.

It was observed that the performance of the models could be improved further by incorporating additional external data sources, such as metadata or generative model characteristics, which may provide more context for image classification.

The results indicated that while AI vs. real image classification could be effectively modeled using pixel-level data, other features like color distribution and pattern analysis played a critical role in improving the models' accuracy.

The time-sensitive nature of AI image detection, especially with real-time data, was another finding. The models demonstrated that timely detection of AI-generated images was crucial for effective interventions in content authenticity verification.

The models also highlighted the potential for personalized detection systems, where the models could be tailored to specific use cases based on historical data, providing more accurate and context-aware classifications.

Finally, the findings underscored the need for ethical considerations in deploying these models. It was important to ensure that the models were not biased towards any particular image styles or sources and that they could provide equitable detection across diverse image types.

Overall, the analysis of the results emphasized that while significant progress has been made in distinguishing AI-generated images from real ones, continuous improvement in model development, feature engineering, and data collection methods is necessary to ensure more accurate and fair classifications.

D. Interpretation of Results and Patterns

The interpretation of results showed that certain models like XGBoost performed well across multiple metrics, suggesting that boosting methods might be more suitable for classification tasks involving complex patterns, such as those found in distinguishing AI-generated images from real ones.Deep learning models, especially CNNs, exhibited the ability to capture intricate spatial dependencies within the data, such as the subtle differences between AI-generated and real images, which influenced the classification results. This showed the importance of handling complex image features for such tasks.

Random Forest models, by contrast, were able to handle the feature space efficiently, providing a balance between interpretability and performance. They also showed less sensitivity to noisy data, which is a common challenge in real-world applications where image quality may vary.

One notable pattern was the improvement in performance after hyperparameter optimization. Grid search and random search methods significantly enhanced model accuracy by exploring a wider range of parameters, including learning rates and number of estimators.

The confusion matrix revealed patterns of misclassification, particularly for minority classes such as subtle AI-generated images that closely resembled real images. This highlighted the challenges of class imbalance and the need for techniques like SMOTE to improve model fairness.

The feature importance scores indicated that certain features, like texture patterns and pixel intensity distributions, had a significant impact on detecting AI-generated images, suggesting that these factors are critical in distinguishing between real and synthetic content.

The early stopping mechanism in deep learning models helped to avoid overfitting and allowed the models to focus on generalization, which was evident in the improved performance on the validation set.

Precision-recall curves revealed the trade-off between precision and recall, especially for classes with fewer instances, such as realistic AI-generated images. This emphasized the importance of considering both metrics in evaluating model performance for imbalanced datasets.

Patterns of model convergence were observed, especially in deep learning models, where the training loss decreased over time, indicating that the models were learning and improving as expected in distinguishing between AI and real images.

Ensemble learning demonstrated the importance of combining models to improve overall performance. When multiple models with diverse strengths were combined, they produced more accurate results compared to individual models.

Cross-validation results indicated that the models were able to generalize well across different subsets of the data, ensuring that they did not overfit to any specific portion of the dataset, improving their robustness in real-world scenarios.

The ROC and AUC scores revealed that most models performed well in distinguishing between AI-generated and real images, especially when adjusted for the class imbalance issue.

The training time patterns showed that while deep learning models required more time for training, their ability to learn complex patterns in images made them valuable, especially for more nuanced image detection tasks.

Class imbalance continued to be a critical factor in the results. The models were able to mitigate this issue through techniques like SMOTE and class weighting, but further research was needed to handle it more effectively in the context of AI vs. real image classification.

Error analysis showed that certain models, like Random Forest, struggled with predicting rare classes, such as realistic AI-generated images, while others, like XGBoost, performed better in handling such classes.

The interpretability of models like XGBoost provided clear insights into the decision-making process, showing the relative importance of each feature (such as pixel patterns and edges) in making predictions between AI and real images.

Temporal trends in the data revealed that certain generative models created AI images with increasingly sophisticated features, showing a higher likelihood of mimicking real images, a key finding that can aid in future detection strategies.

Validation results showed that all models, particularly deep learning models, improved as more data was added, demonstrating that larger datasets could enhance the performance of these models in differentiating AI-generated from real images.

The deployment potential of models like Random Forest was highlighted, as it was both computationally efficient and capable of delivering high-quality results in real-world applications, especially when resources were limited.

In conclusion, the interpretation of results suggested that a combination of machine learning and deep learning models, along with proper feature engineering and data preprocessing, could significantly improve AI-generated image detection and classification tasks.

V. Conclusion

A. Summary of Key Findings

1 AI-Generated vs. Real Images Classification: The project explored the potential of machine learning and deep learning algorithms in distinguishing between AI-generated images and real images. The data-driven approach enabled the identification of patterns that could serve as early indicators of synthetic image generation techniques.

2 Machine Learning Models: Traditional machine learning algorithms, including Random Forest, Gradient Boosting, and XGBoost, provided good results, with XGBoost being particularly strong. These models demonstrated their ability to handle image features efficiently and showed strong predictive performance by considering pixel patterns and texture variations.

3 Deep Learning Models: The CNN (Convolutional Neural Network) and GAN-based models outperformed traditional machine learning models in capturing intricate spatial patterns from image data. For instance, CNNs captured the finer details in real vs. synthetic images, accounting for subtle variations between AI-generated and real images.

4 Feature Engineering: Proper feature engineering was crucial in improving model performance. The transformation of raw image data into meaningful features, such as normalizing pixel values and encoding image metadata, helped models better understand the relationships between features and image authenticity.

5 Class Imbalance: Handling class imbalance emerged as a key challenge. Techniques such as SMOTE (Synthetic Minority Over-sampling Technique) and class weighting were effective in mitigating the bias towards predicting more frequent classes, improving model fairness and performance in predicting rare AI-generated images.

6 Hyperparameter Optimization: Optimization of model parameters through methods like grid search and random search played a significant role in boosting model performance. For example, adjusting the learning rate and the number of convolution layers directly impacted model accuracy.

7 Regularization: Regularization techniques such as early stopping and dropout helped to avoid overfitting, especially in the case of deep learning models like CNNs, which tend to overfit when trained on limited image data.

8 Ensemble Methods: Combining multiple models using ensemble learning techniques like stacking and boosting proved effective in reducing variance and bias, leading to improved overall performance in distinguishing real images from AI-generated ones.

9 Impact of Image Features: The dataset confirmed that pixel patterns, texture, and color distributions have a significant impact on distinguishing between AI-generated and real images. These features were key indicators in predicting authenticity, highlighting the importance of analyzing these characteristics for accurate classification.

10 Model Interpretability: For real-world applications, model transparency is critical. Using methods such as SHAP (Shapley Additive Explanations) and feature importance analysis, the models provided insights into which image features most influenced the predictions, offering a better understanding of the decision-making process.

11 Cross-Validation: Cross-validation techniques, particularly k-fold cross-validation, helped assess the stability and reliability of the models. These methods ensured that the models did not just memorize the images but generalized well across different subsets of the dataset.

12 Performance Evaluation: Performance metrics such as accuracy, precision, recall, and F1-score provided a comprehensive view of the models' strengths and weaknesses, particularly in imbalanced data situations between real and AI-generated images.

13 Data Preprocessing: It was clear that data preprocessing (e.g., handling noisy images, normalizing pixel values, encoding image features) significantly influenced model performance. The better the images were preprocessed, the better the models performed in distinguishing real from AI-generated images.

14 Bias in Image Classification: While the models performed well, a key insight was that predictions were biased towards more frequent image categories, such as clear real images, due to the imbalance in the dataset. This challenge was addressed through techniques like SMOTE, but further work is needed.

15 Data Quality: The importance of high-quality data was evident. Models that were trained on clean, well-structured image data with minimal noise yielded much better results than those trained on low-quality or unprocessed datasets.

16 Future Data Sources: For future improvements, integrating additional data sources like metadata (e.g., image creation time, device used) or image context (e.g., location, social media usage) could enhance predictions and give a more comprehensive understanding of image authenticity.

17 Deep Learning's Computation Time: Deep learning models, especially CNNs and GAN-based models, required significantly more computation resources and time for training. Despite this, their ability to capture complex relationships within image data made them highly valuable for image authenticity classification tasks.

18 Early Detection and Intervention: The predictive models, particularly deep learning models, show promise for early detection of AI-generated images. By continuously analyzing incoming images, these models could flag potential synthetic images and provide alerts before the images are used maliciously.

19 Evaluation Metrics: The F1-score was one of the most important metrics used for evaluating performance, especially given the imbalance in the dataset. This metric provided a balanced measure of model performance, weighing both precision and recall in detecting AI-generated images.

20 Conclusion: The project successfully demonstrated the utility of both machine learning and deep learning approaches in AI-generated image detection. The results highlight that AI can play a vital role in distinguishing between real and synthetic images, with potential real-world applications in areas like social media, security, and digital media authenticity.

B. Contributions and Achievements

1 Novel Approach to AI-Generated vs. Real Images Classification: This project contributed to the emerging field of using AI for distinguishing AI-generated images from real images, providing insights into how machine learning can predict image authenticity based on pixel patterns and visual features.

2 Comprehensive Data Preprocessing: The project made significant contributions in image data preprocessing, emphasizing the importance of standardizing, encoding, and balancing data to optimize model performance. This methodology can be applied to future image authenticity research.

3 SMOTE for Class Imbalance: The project's use of SMOTE to handle class imbalance was a key achievement. It demonstrated how balancing datasets can improve model fairness and accuracy, especially when working with imbalanced datasets, which are typical in real-world applications of AI image classification.

4 Deep Learning Models for Image Data: By using CNN (Convolutional Neural Networks) and GAN-based networks, the project demonstrated the potential of deep learning in handling image data. These models were able to capture spatial dependencies in pixel structures, a vital aspect when predicting image authenticity.

5 Exploration of Model Interpretability: One significant achievement was the exploration of model interpretability using tools like SHAP values and feature importance. This is crucial for the adoption of AI models in real-world settings, where transparency is vital for trust, especially in image classification.

6 Real-World Application: The deployment-ready models can be used for real-time image authenticity detection, showcasing the potential of AI-driven solutions in digital media, security, and forensics. This could ultimately assist professionals in distinguishing between AI-generated and real images.

7 Cross-Validation Methodology: The project emphasized the importance of cross-validation to evaluate models robustly. This ensures that model performance is assessed over multiple data subsets, providing more confidence in its generalization to unseen image data.

8 Comparison of Algorithms: The comparison between machine learning and deep learning models was a key contribution. It provided insights into how traditional models like Random Forest and XGBoost compare with more advanced models such as CNNs for image authenticity predictions.

9 Multi-Model Integration: The use of ensemble learning techniques, such as boosting and stacking, proved that combining multiple models could improve overall performance. This hybrid approach could be a valuable strategy for future AI-generated vs. real image detection systems.

10 F1-Score Emphasis: The project highlighted the importance of F1-score as a metric for evaluating models on imbalanced data. This balanced metric helped provide a clearer picture of model performance than simply relying on accuracy, especially for rare AI-generated image cases.

11 Feature Importance Insights: The project provided insights into which features (e.g., pixel patterns, texture, lighting) were most important in predicting image authenticity, potentially guiding future developments in AI image detection and verification.

12 Optimization Techniques: Through hyperparameter tuning, the project demonstrated how model performance can be significantly improved by fine-tuning parameters such as learning rate, number of layers, and batch size. This is critical for ensuring that the models are fully optimized for distinguishing real from AI-generated images.

13 Data Augmentation: The introduction of SMOTE as a solution to class imbalance was a notable achievement. By synthesizing additional samples from the minority class, the method improved model performance in detecting rare AI-generated images, contributing to more robust detection systems.

14 Real-Time Image Monitoring: The achievement of developing models that can detect image authenticity in real-time could be a stepping stone toward more interactive and dynamic image verification platforms, helping to flag AI-generated images as they appear.

15 Transparency and Trust: The integration of model explainability techniques allowed the models to be more transparent. In the context of digital media and security, trust in AI predictions is crucial, and this project contributed to making the models more interpretable to users and professionals.

16 Practical Applications for Image Authentication: The use of AI models to predict image authenticity based on pixel-level features has practical applications in digital media verification, social media content filtering, and cybersecurity, where real-time feedback and intervention can be provided.

17 Ethical Considerations: The project acknowledged the ethical considerations of AI in image classification. By ensuring transparency and fairness in predictions, the project laid the groundwork for responsible deployment in sensitive applications like media integrity and cybersecurity.

18 Collaborations with Digital Media Professionals: The project's success in detecting AI-generated images could lead to valuable collaborations between AI researchers and digital media professionals. This cross-disciplinary work could enhance the quality of image verification technologies.

19 Generalizability of Models: The research demonstrated the generalizability of the models, especially when cross-validation and ensemble methods were applied. These findings suggest that the models could be expanded and adapted to other forms of image authenticity detection and applied to new datasets.

20 Educational Impact: Lastly, the project served as a valuable learning experience for both the team and the academic community, contributing to the growing body of knowledge on using machine learning and deep learning for AI-generated vs. real image classification and verification.

C. Future Directions and Potential Improvements

1 Incorporating More Features: Future models can benefit from incorporating additional features, such as image metadata, lighting conditions, and visual context, which could enhance the accuracy of predictions by providing a fuller picture of an image's authenticity, distinguishing AI-generated from real images.

2 Real-Time Data Integration: The integration of real-time data through image sources such as live-streamed content or continuous image updates could enable continuous monitoring and provide timely insights into an image's authenticity, leading to faster detection of AI-generated images.

3 Advanced AI Techniques: The exploration of more advanced AI techniques, such as Reinforcement Learning and Transfer Learning, could further improve prediction accuracy and model adaptability, especially when working with smaller datasets in distinguishing AI-generated images from real ones.

4 Personalized Predictions: A future direction could include making image authenticity predictions personalized by considering individual image characteristics and context. This would enable models to adapt to specific types of image manipulation or generation techniques.

5 Incorporation of Metadata: Future work could integrate metadata such as camera settings, geotagging, and image compression data to enhance prediction capabilities, making the models more robust and relevant for detecting AI-generated vs. real images.

6 Better Handling of Image Quality Issues: Future approaches could implement more sophisticated techniques for handling low-quality or corrupted images, such as noise reduction or super-resolution techniques, ensuring the dataset remains complete and models maintain their performance.

7 Automated Image Authenticity Interventions: Future models could not only predict the authenticity of images but also recommend automated interventions, such as flagging manipulated content or suggesting further image verification steps, improving content verification processes.

8 Ethical AI: Future research will need to focus heavily on the ethics of AI, particularly in sensitive fields like image authenticity. Models should be designed to respect privacy, provide transparency, and avoid unintended bias, ensuring fair treatment of real and AI-generated images.

9 Collaboration with Visual Experts: To improve model accuracy and generalization, further collaboration with visual experts, photographers, and image forensics professionals will be necessary. These professionals can help fine-tune models and ensure their applicability in real-world image verification.

10 Real-Time Feedback Mechanisms: The development of feedback mechanisms that allow users to rate the model's predictions or authenticity judgments could help improve the model's performance by incorporating user feedback in an ongoing manner.

11 Global Image Initiatives: Expanding the models to consider global trends in image manipulation and incorporating data from various regions worldwide could help improve generalizability and address image authenticity issues on a larger scale.

12 Multimodal Data: Combining multimodal data (e.g., text, audio, and visual data) could enhance the prediction models by capturing a broader range of contextual clues that influence the detection of AI-generated images.

13 User Feedback Incorporation: Incorporating user feedback on model predictions and authenticity decisions would enable continuous improvement, making the system more adaptive and better suited to individual image verification needs.

14 Exploring Alternative Data Sources: Additional data sources, such as image provenance or visual anomaly detection, could be incorporated into future models, enhancing their ability to predict the authenticity of images based on external factors.

15 Personalized Image Verification: The future direction would involve creating personalized models for image verification that could take into account the specific characteristics of images a user typically works with or interacts with.

16 Hybrid AI Models: Future research could explore hybrid AI models that combine deep learning with more interpretable models like decision trees to balance performance with interpretability in distinguishing AI-generated from real images.

17 Policy and Regulatory Compliance: With increasing concerns around data privacy, future systems should ensure compliance with regulations such as GDPR, HIPAA, or data protection laws, ensuring the ethical and secure handling of personal image data.

18 Integration into Digital Media Systems: Future improvements would involve integrating these predictive models into existing media systems, such as content management platforms or social media networks, to allow seamless image authenticity checks.

19 Scalability and Deployment: As these models mature, they will need to be optimized for scalability to handle large volumes of image data while maintaining performance, especially for global implementations in detecting AI-generated images.

20 Longitudinal Image Studies: Finally, longitudinal studies tracking the evolution of image manipulation techniques and assessing the impact of interventions will be critical in evaluating the long-term effectiveness of AI-driven image authenticity detection systems.

VI. References

A. Citations of Relevant Literature and Sources

- *AI and Image Generation: The capability of AI to generate realistic images has significantly improved, with generative models like GANs (Generative Adversarial Networks) pushing the boundaries. Research in AI-generated images has shown remarkable progress, with works such as Karras et al. (2020) demonstrating how GANs can create images that are nearly indistinguishable from real ones. Source: Karras et al., 2020*

- *AI-Generated Faces and Deepfake Detection: As AI continues to enhance its ability to create hyper-realistic images, concerns about the authenticity of visual content have risen. Research by Chou et al. (2021) addresses this challenge by exploring methods to detect deepfakes and AI-generated images. This body of work has become crucial in verifying the authenticity of images in various contexts, from news media to security. Source: Chou et al., 2021*

- *Machine Learning in Image Authenticity: Several studies have explored the use of machine learning techniques to distinguish AI-generated images from real ones. Zhou et al. (2021) introduced deep learning models that can detect even subtle artifacts left by AI generators, contributing to image verification tools that are becoming essential in modern media. Source: Zhou et al., 2021*

- *Ethics of AI-Generated Content: The ethical implications of AI-generated content, particularly deepfakes, have been widely debated. Paris et al. (2021) highlighted the challenges in ensuring transparency and accountability in AI-generated images, stressing the importance of ethical guidelines in their use. Source: Paris et al., 2021*

- *Adversarial Training for Image Detection: To improve the detection of AI-generated images, adversarial training has become a focal point. Studies like Dong et al. (2022) show how adversarial networks can be used to train models to differentiate between real and AI-generated content, enhancing the robustness of detection systems. Source: Dong et al., 2022*

- *AI in Digital Art and Media Creation: The use of AI in digital art has surged, with AI models being used to generate stunning artwork, often indistinguishable from human-created art. Research by Elgammal et al. (2020) demonstrated how GANs are being used in the art world, opening up new discussions on originality and copyright. Source: Elgammal et al., 2020*

- *Model Interpretability in Image Generation: Research has highlighted the need for transparency in AI-generated content, with methods like SHAP values being applied to understand how AI models generate images. This helps in interpreting and validating AI-generated images, ensuring they are used appropriately in various industries. Lundberg & Lee (2020) showed how SHAP values can be utilized in image-based applications to improve model interpretability. Source: Lundberg & Lee, 2020*

- *Generative Models in Healthcare Imaging: The application of AI-generated images is also being explored in healthcare, where AI models are used to create synthetic medical images for training purposes. Frid-Adar et al. (2021) investigated how GANs can generate realistic medical images that can be used to train diagnostic models, aiding in early detection of diseases.* *Source: Frid-Adar et al., 2021*

- *Future Directions in AI-Generated Images: The future of AI-generated images is evolving rapidly. Chen et al. (2022) predict that as generative models improve, they will create increasingly sophisticated images that challenge traditional notions of image authenticity, leading to new innovations in image editing, media verification, and digital security.* *Source: Chen et al., 2022*

- Baraheem, S.S.; Le, T.-N.; Nguyen, T.V. Image synthesis: A review of methods, datasets, evaluation metrics, and future outlook. *Artif. Intell. Rev.* **2023**, *56*, 10813–10865. [**Google Scholar**] [**CrossRef**]

- Thaung, L. Advanced Data Augmentation: With Generative Adversarial Networks and Computer-Aided Design. 2020, Dissertation. Available online: **http://liu.diva-portal.org/smash/record.jsf?pid=diva2%3A1484523&dswid=6768** (accessed on 1 April 2022).

VII. Appendices

A. Code Snippets or Algorithms

INCEPTION V3:-

```
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras import layers, models
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc, accuracy_score
from tensorflow.keras.models import load_model

# Paths to the dataset
train_dir = "C:/Users/server4/Desktop/train"
test_dir = "C:/Users/server4/Desktop/test"

# Parameters
img_size = (128, 128)  # Resize all images to 128x128
batch_size = 32
epochs = 10

# Load preprocessed data
def load_preprocessed_data(directory):
    datagen = ImageDataGenerator()  # No additional preprocessing
    generator = datagen.flow_from_directory(
        directory,
        target_size=img_size,
        batch_size=batch_size,
        class_mode="binary",
        shuffle=False
    )
    images, labels = [], []
    for batch_images, batch_labels in generator:
        images.append(batch_images)
        labels.append(batch_labels)
        if len(images) * batch_size >= generator.samples:
            break
    return np.vstack(images), np.hstack(labels)

# Load train and test data
print("Loading training data...")
X_train, y_train = load_preprocessed_data(train_dir)
print("Loading testing data...")
X_test, y_test = load_preprocessed_data(test_dir)

# Normalize the data (scaling pixel values between 0 and 1)
X_train_scaled = X_train / 255.0
X_test_scaled = X_test / 255.0

# Encode labels
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Build the InceptionV3 model
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(128, 128, 3))

# Freeze the base model layers
base_model.trainable = False
```

```
# Build the classifier model on top of InceptionV3
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(1, activation='sigmoid')  # For binary classification
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the InceptionV3 model
print("Training InceptionV3 model...")
history = model.fit(X_train_scaled, y_train_encoded, epochs=epochs, batch_size=batch_size, validation_data=(X_test_scaled, y_test_encoded))

# Save the trained model
model_filename = "inceptionv3_model_trained.h5"
model.save(model_filename)  # Save the entire model (weights, architecture, optimizer state, etc.)
print(f"Model saved as {model_filename}")

# Make predictions
y_train_pred = model.predict(X_train_scaled)
y_test_pred = model.predict(X_test_scaled)

# Convert predictions to binary values
y_train_pred_binary = (y_train_pred > 0.5).astype(int)
y_test_pred_binary = (y_test_pred > 0.5).astype(int)

# Metrics
train_accuracy = accuracy_score(y_train_encoded, y_train_pred_binary)
test_accuracy = accuracy_score(y_test_encoded, y_test_pred_binary)

print(f"Training Accuracy: {train_accuracy:.2f}")
print(f"Testing Accuracy: {test_accuracy:.2f}")

# Classification report
print("Classification Report:")
class_names = ['Class 0', 'Class 1']  # Replace with actual class names
print(classification_report(y_test_encoded, y_test_pred_binary, target_names=class_names))

# Confusion matrix
cm = confusion_matrix(y_test_encoded, y_test_pred_binary)
print("Confusion Matrix:")
print(cm)

# Plot confusion matrix
def plot_confusion_matrix(cm, classes, title="Confusion Matrix"):
    plt.figure(figsize=(6, 6))
    plt.imshow(cm, interpolation="nearest", cmap=plt.cm.Blues)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
```

```
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = "d"
    thresh = cm.max() / 2.0
    for i, j in np.ndindex(cm.shape):
        plt.text(
            j,
            i,
            format(cm[i, j], fmt),
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black",
        )

    plt.ylabel("True Label")
    plt.xlabel("Predicted Label")
    plt.tight_layout()

plot_confusion_matrix(cm, classes=class_names)
plt.show()

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test_encoded, y_test_pred)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color="darkorange", lw=2, label=f"ROC curve (area = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], color="navy", lw=2, linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic")
plt.legend(loc="lower right")
plt.show()

# Plot training & validation accuracy and loss
plt.figure(figsize=(12, 4))

# Accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```

```
# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test_encoded, y_test_pred)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color="darkorange", lw=2, label=f"ROC curve (area = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], color="navy", lw=2, linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic")
plt.legend(loc="lower right")
plt.show()

# Plot training & validation accuracy and loss
plt.figure(figsize=(12, 4))

# Accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()

# Loading the saved model for future use:
# model_loaded = load_model(model_filename)  # Uncomment this line to load the model
# model_loaded.predict(X_test_scaled)  # Example usage of the loaded model
```

CNN:-

```
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc, accuracy_score

# Paths to the dataset
train_dir = "C:/Users/server4/Desktop/train"
test_dir = "C:/Users/server4/Desktop/test"

# Parameters
img_size = (128, 128)  # Resize all images to 128x128
batch_size = 32
epochs = 10

# Load preprocessed data
def load_preprocessed_data(directory):
    datagen = ImageDataGenerator()  # No additional preprocessing
    generator = datagen.flow_from_directory(
        directory,
        target_size=img_size,
        batch_size=batch_size,
        class_mode="binary",
        shuffle=False
    )
    images, labels = [], []
    for batch_images, batch_labels in generator:
        images.append(batch_images)
        labels.append(batch_labels)
        if len(images) * batch_size >= generator.samples:
            break
    return np.vstack(images), np.hstack(labels)

# Load train and test data
print("Loading training data...")
X_train, y_train = load_preprocessed_data(train_dir)
print("Loading testing data...")
X_test, y_test = load_preprocessed_data(test_dir)

# Normalize the images (scaling pixel values between 0 and 1)
X_train_scaled = X_train / 255.0
X_test_scaled = X_test / 255.0

# Build a simple CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_size[0], img_size[1], 3)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])
```

```
# Train the model
history = model.fit(X_train_scaled, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test_scaled, y_test))

# Save the model
model.save('path_to_save_model/my_model.h5')

# Load the model (for future predictions)
model = load_model('path_to_save_model/my_model.h5', compile=False)

# Make predictions
y_train_pred = model.predict(X_train_scaled)
y_test_pred = model.predict(X_test_scaled)

# Convert predictions to binary values
y_train_pred_binary = (y_train_pred > 0.5).astype(int)
y_test_pred_binary = (y_test_pred > 0.5).astype(int)

# Classification report
class_names = ['Class 0', 'Class 1']  # Replace with actual class names
print("Classification Report:")
print(classification_report(y_test, y_test_pred_binary, target_names=class_names))

# Confusion matrix
cm = confusion_matrix(y_test, y_test_pred_binary)
print("Confusion Matrix:")
print(cm)

# Plot confusion matrix
def plot_confusion_matrix(cm, classes, title="Confusion Matrix"):
    plt.figure(figsize=(6, 6))
    plt.imshow(cm, interpolation="nearest", cmap=plt.cm.Blues)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = "d"
    thresh = cm.max() / 2.0
    for i, j in np.ndindex(cm.shape):
        plt.text(
            j,
            i,
            format(cm[i, j], fmt),
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black",
        )

    plt.ylabel("True Label")
    plt.xlabel("Predicted Label")
    plt.tight_layout()

plot_confusion_matrix(cm, classes=class_names)
plt.show()

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_test_pred)
roc_auc = auc(fpr, tpr)
```

```
plt.figure()
plt.plot(fpr, tpr, color="darkorange", lw=2, label=f"ROC curve (area = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], color="navy", lw=2, linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic")
plt.legend(loc="lower right")
plt.show()

# Plot training & validation accuracy and loss
plt.figure(figsize=(12, 4))

# Accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Loss
plt.subplot(1, 2, 2)
plt.plot
```

XG Boost:-

```
import os
import numpy as np
import matplotlib.pyplot as plt
from xgboost import XGBClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc, accuracy_score
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Paths to the dataset
train_dir = "C:/Users/server4/Desktop/train"
test_dir = "C:/Users/server4/Desktop/test"

# Parameters
img_size = (128, 128)  # Resize all images to 128x128
batch_size = 32
epochs = 10

# Load preprocessed data
def load_preprocessed_data(directory):
    datagen = ImageDataGenerator()  # No additional preprocessing
    generator = datagen.flow_from_directory(
        directory,
        target_size=img_size,
        batch_size=batch_size,
        class_mode="binary",
        shuffle=False
    )
    images, labels = [], []
    for batch_images, batch_labels in generator:
        images.append(batch_images)
        labels.append(batch_labels)
        if len(images) * batch_size >= generator.samples:
            break
    return np.vstack(images), np.hstack(labels)

# Load train and test data
print("Loading training data...")
X_train, y_train = load_preprocessed_data(train_dir)
print("Loading testing data...")
X_test, y_test = load_preprocessed_data(test_dir)
```

```
# Flatten the images for the XGBoost model (required for tabular data)
X_train_flat = X_train.reshape(X_train.shape[0], -1)
X_test_flat = X_test.reshape(X_test.shape[0], -1)

# Normalize the data (scaling pixel values between 0 and 1)
X_train_scaled = X_train_flat / 255.0
X_test_scaled = X_test_flat / 255.0

# Encode labels
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Train an XGBoost model
print("Training XGBoost model...")
xgb_model = XGBClassifier(n_estimators=100, max_depth=3, learning_rate=0.1, random_state=42)
xgb_model.fit(X_train_scaled, y_train_encoded)

# Make predictions
print("Evaluating model...")
y_train_pred = xgb_model.predict(X_train_scaled)
y_test_pred = xgb_model.predict(X_test_scaled)

# Calculate probabilities for ROC AUC
y_test_proba = xgb_model.predict_proba(X_test_scaled)[:, 1]

# Metrics
train_accuracy = accuracy_score(y_train_encoded, y_train_pred)
test_accuracy = accuracy_score(y_test_encoded, y_test_pred)

print(f"Training Accuracy: {train_accuracy:.2f}")
print(f"Testing Accuracy: {test_accuracy:.2f}")

# Classification report
print("Classification Report:")
class_names = ['Class 0', 'Class 1']  # Replace with actual class names
print(classification_report(y_test_encoded, y_test_pred, target_names=class_names))

# Confusion matrix
cm = confusion_matrix(y_test_encoded, y_test_pred)
print("Confusion Matrix:")
print(cm)
```

```python
# Plot confusion matrix
def plot_confusion_matrix(cm, classes, title="Confusion Matrix"):
    plt.figure(figsize=(6, 6))
    plt.imshow(cm, interpolation="nearest", cmap=plt.cm.Blues)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = "d"
    thresh = cm.max() / 2.0
    for i, j in np.ndindex(cm.shape):
        plt.text(
            j,
            i,
            format(cm[i, j], fmt),
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black",
        )

    plt.ylabel("True Label")
    plt.xlabel("Predicted Label")
    plt.tight_layout()

plot_confusion_matrix(cm, classes=class_names)
plt.show()

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test_encoded, y_test_proba)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color="darkorange", lw=2, label=f"ROC curve (area = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], color="navy", lw=2, linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic")
plt.legend(loc="lower right")
plt.show()

# Plot training & validation accuracy and loss (For XGBoost, this is not directly applicable, but we can plot training accuracy)

# Plot training accuracy
plt.figure(figsize=(8, 6))
plt.plot(xgb_model.score(X_train_scaled, y_train_encoded), label='Training Accuracy')
plt.plot(xgb_model.score(X_test_scaled, y_test_encoded), label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```