Adversarial Exploits: Classification of AI Generated Human Faces

A study submitted in partial fulfilment
of the requirements for the degree of
MSc in Data Science

at

THE UNIVERSITY OF SHEFFIELD

by

Sarath Tharayil Sreenivasan
220202134

Word Count: 9279

September 2023

Table of Contents

Table of Figures

Table of Tables

ABSTRACT

AIM

This paper develops and evaluates a Convolutional Neural Network (CNN) model for classifying real human faces and faces created by Generative Adversarial Networks (GANs). Its main goal is to accurately distinguish between real human facial images and their AI-generated duplicates.

METHODS

The study uses a comprehensive methodology that includes a literature review to understand the current state of artificial face generation, data preprocessing, exploratory data analysis (EDA), feature extraction using the VGG-16 architecture, model development, and the development of an interactive Streamlit dashboard for real-time image classification. Future development areas include data augmentation, position estimation, object identification, user feedback, and transformable actual faces.

RESULTS

By classifying both real and GAN-generated images with an astonishing 98.5% accuracy, the CNN model demonstrated its impressive generalization performance. Although the model performed exceptionally well in binary classification, difficulties were noted in separating orientations from situations involving many participants. In terms of ethical considerations, AI research has placed an emphasis on privacy, openness, and responsible disclosure.

CONCLUSION

This work successfully demonstrates the capabilities of CNN models in the complex task of identifying real human faces from AI-generated faces. It highlights the importance of ethical obligations associated with AI research efforts and the need for responsible and open practices. Additionally, this project emphasizes the critical value of rigorous data preparation and the importance of creating diverse training data sets that accurately reflect real-world circumstances. This guarantees the generalizability and ability of the models to be applied to a wide range of scenarios. Finally, this study examines the nuances of model performance and possible improvements, exploring options such as data augmentation, pose estimation, and user feedback to further improve the model's capabilities.

1. Introduction

In the age of artificial intelligence (AI), the distinction between reality and digital creation has become increasingly blurred (Jordan, 2009). Due to deep fake videos, AI generated images and text, it is becoming increasingly difficult to tell what is real and what is not. In a world where images generated by AI can be misused, recognizing them appropriately is crucial. Berger (2014) states that all arts are going digital, merging in transdisciplinarity and transmedia, further blurring the line between reality and digital creation. The concept of AI has been around since the 1950s, with the development of computers and the ability to perform tasks that would normally require human intelligence (Simon, 1993). But it was only with the development of neural networks and deep learning algorithms that AI technology really took off. Modern industries such as transportation, agriculture, medicine, and many more have successfully embraced AI (Khan et al., 2021). As a result, AI has been used to develop content that can mimic human speech, movement, and even creativity.

Using AI to create or craft information based on user requests falls under AI-generated content (Du et al., 2023). Images, videos, text and even music are examples of this. AI has a wide range of possible applications, particularly in entertainment and games, but these applications also come with significant difficulties. The ability to produce highly realistic content that is difficult to spot raises questions about its authenticity and the potential for misuse.

In today's digital environment, developing methods to reliably detect AI-generated material is critical. The widespread use of AI-generated content, including deepfakes and synthetic media, poses significant risks to society. False information, malicious impersonations and fake news can undermine trust, damage reputations, and undermine democratic processes (Hu, 2020). The integrity of information and media must be protected by accurately identifying AI-generated content. It gives people, organizations, and platforms the ability to distinguish real content from fake, allowing them to make informed decisions and reduce potential harm (Meel & Vishwakarma, 2020). The ability to properly identify AI-generated content is critical for several reasons like to prevent the spread of misinformation and fake news that can have serious consequences like damage to reputation or dissemination of propaganda (Kreps et al., 2022).

1.1 Aims and Objectives

Deep learning techniques have evolved over the past decade, making it increasingly easy to create images that are extremely photorealistic. As a result, it is very difficult, if not impossible, for the average social media user today to identify fake social media content and profiles in their feed (Rossi et al., 2022).

Research Aim:

RA1. To develop an efficient and optimal method of classifying real human faces from AI generated faces.

It can be difficult to distinguish between what is real and what is fake, making accurate labelling of fabricated content a problem. Complicating the issue further is the possibility that some fake content was intentionally generated to look authentic, making it difficult to pinpoint where it really came from. Also, due to the massive amount of data generated each day, it is impractical to carefully screen each medium for legitimacy. Using a standardized system that uses machine learning algorithms to automatically recognize and classify AI-generated humans would be the ideal approach. The overall goal of this system is to ensure that AI-generated individuals are always identified as virtual and are not used to create false identities for unlawful behaviour that could have serious repercussions on society, such as the loss of trust and influencing public opinion.

Research Objectives:

RO1. Critical review of the literature on AI-generated faces and face classification techniques.
RO2. To identify the features that distinguish real human faces from AI-generated faces.
RO3. Development of a classification model using machine learning algorithms to differentiate between real and AI-generated faces.
RO4. To evaluate the performance of the classification model by testing it on a large data set of both real and artificial faces.

Research Question:

RQ1. How can current real and fabricated face classification methods be improved to develop an optimal model to accurately classify real and AI-generated human faces?

Structure of the Dissertation

The five chapters that make up the remaining parts are the literature review, methodology, results, discussions and conclusion.

2. Literature Review

2.1 Artificial face generation techniques

In recent years, the involvement of artificial intelligence in the creation of images has attracted a lot of attention (Liu & Yu, 2021). The use of AI-generated text, audio, image, and video as a weapon for financial fraud, misinformation campaigns, and non-consensual intimate photography is widespread (Nightingale and Farid 2022). Because of their many uses, face photos rank highly among the various

categories of real images (Abdolahnejad & Liu, 2020). Although face images are very difficult to synthesize due to their highly complex hierarchical structure and the uniqueness of the information contained in each individual face image, recent advances in face synthesis and semantic manipulations have made it possible to create artificial faces that are identical to real faces (Yegemberdiyeva & Amirgaliyev, 2021).

Bank et al. (2020) introduces autoencoder networks, a type of neural network that can meaningfully compress and represent input data before decoding it to produce input that is comparable to the original. By training the autoencoder with a dataset of faces, they can be used to create new faces by sampling from latent space. Variational autoencoders (VAE) that Kingma and Welling (2019) proposes use a Bayesian approach to learn the probability distribution of the input data and apply it to generate new samples. Ozkan and Ozkan (2018) proposes a kinship network that can create a potential child's face by examining a picture of his parents. Sun et al. (2014) demonstrates how deep learning can be used to create useful trait representations that increase interpersonal differences while reducing intrapersonal variability. AI-generated images have become increasingly complex and difficult to distinguish from real-world images due to these breakthroughs in Generative Adversarial Networks (GAN), autoencoders, and other AI technologies (Bang & Woo, 2021). The generated faces show that the AI algorithms have overcome the uncanny valley and are now able to generate indistinguishable and reliable faces (Nightingale & Farid, 2022). This is precisely why realistic faces produced by GANs are the subject of media attention. In studies and surveys that attempted to quantify distinctness, people have consistently been unable to reliably distinguish between all real and synthetic images (Kas et al., 2020).

2.2 Deep learning methods for face generation

Deep learning methods, specifically Generative Adversarial Networks (GANs), are used to create AI-generated human faces (Creswell et al., 2018). This is a method of generative modelling is based on a game between two machine learning models, the discriminator (D) and the generator (G), typically implemented using neural networks (Goodfellow et al., 2020). D has access to both the training set and G's generations, and predicts the probability that the sample is from the training set or from G (Creswell et al., 2018). At the same time, G only learns from interacting with D and generates images without access to the actual images in the training set, and its training procedure is to create generations and maximize the probability that D makes a mistake (Goodfellow et al., 2014). The generator then modifies its output to produce images that are more lifelike until the discriminator is no longer able to tell the difference between actual and synthetic images. The four typical fake face synthesis modes with different GANs are whole face synthesis, facial attribute editing, facial expression manipulation, and Deepfakes (Stehouwer et al., 2019).

When training GANs, a low-variety training set causes the generator to spins through a small range of output types and never learn its way out of the trap, leading to a problem known as mode collapse (Ding et al., 2022). Another problem of vanishing gradients arises when, during training, the gradients required to update the neural network weights are propagated back through the layers of the network and become very low. The network struggles to learn this because the weights are not updated frequently enough (Su, 2018). As a result, we have different GAN types to modify and fix the glitches in the current algorithm, or specially designed types for specific purposes, different architectures or using new techniques.

There are different types of GANs that are made specifically for creating synthetic human faces. These types of GANs use a variety of algorithms and architectures to create highly realistic and detailed images of human faces with different characteristics such as pose, expression, and background. Some of the different types of GANs specifically used to generate human faces are as follows:

1. Progressive growing of GANs (ProGAN): This type initially produces low-resolution images and gradually increases the resolution over time to produce detailed and high-quality images of human faces. Instead of having to learn all the scales at once, the training can first recognize the large-scale structure of the image distribution before focusing on finer and finer scale details (Karras et al., 2017).
2. Style-based GAN architecture (Style GAN): This is currently the most advanced technique for high-resolution image synthesis (Karras et al., 2019). With this method, the image synthesis process is broken down into two distinct steps: creating a low-resolution image to capture fundamental aspects such as posture and expression, and using a different network to encode style information such as colour, texture, and fine detail (Karras et al., 2020).
3. Cycle-Consistent Generative Adversarial Network (CycleGAN): Unlike typical GANs, CycleGAN is designed for image-to-image translation jobs. Two different GANs are trained to fulfil the purpose of CycleGAN to learn a mapping between two different types of images: one to create images in the destination domain from photos in the source domain and another to create images in the source domain from images in the destination domain ensuring that the image translations are consistent in both directions (Wang & Lin, 2018).
4. Star Generative Adversarial Network (StarGAN): StarGAN is a powerful and flexible model that uses a single generator network to produce realistic renderings while changing one aspect of a given image to another, e.g., B. by changing a person's facial expression from a smile to a frown (Choi et al., 2018).
5. Self-Attention Generative Adversarial Network (SAGAN): This can generate high-resolution images of human faces with greater attention to detail. It is

used to manipulate facial attributes, e.g., to remove or put on glasses, and uses a self-attention technique to focus on key elements of the image.

Overall, from the literature available, GANs have become the preferred technique for face manipulation and synthesis due to their generality, diversity, realism, and efficiency.

2.3 Recognizing AI-generated images

Several techniques have been developed to recognize images generated by AI. Rule-based methods and machine learning-based methods are two main groups that these techniques can be divided into. Rule-based methods involve establishing precise guidelines or standards that can be used to recognize images generated by artificial intelligence, e.g., searching for patterns or artifacts that are common in these images (Gu & Angelov, 2018). Techniques based on machine learning involve teaching a machine learning model to recognize patterns and features in a data set of real and AI-generated images. Wang et al. (2019) observe that the imperfection of the upsampling methods it embodies could serve as an important advantage for GAN-synthesized fake image detection and fake location. Much research has been done on manipulations such as face switching, face re-enactment, and expression modification, resulting in the development of a variety of useful techniques for distinguishing between fake and authentic videos and images (Alamayreh & Barni, 2021). Empirically, a human is an animate being, while a machine is inanimate. The line between human and AI seems to be blurred by the human-like artificial entities. Our visual system's ability to recognize life in a face is referred as facial animation perception (Koldewyn et al., 2014) and Xiang et al. (2022) use an adaptation paradigm to explore how humans perceive animated faces.

Misidentification of AI-generated images can have serious repercussions on people, businesses, and society at large. For instance, mistaking an AI-generated image for a real-world image may cause misinformation or fake news to spread, which may have serious repercussions for society (Burton & Soare, 2019). Misidentifying an AI-generated image can also lead to ethical or legal problems, such as when those photographs are used to make deepfake videos of people without their permission. Since most of the research datasets, like Celeb-DF (Li et al., 2020), have only recently been published and GANs were only developed in 2014 (Goodfellow et al., 2014), the field of identifying AI-generated images is still new and there is much room for future research. One promising area of research is the development of more advanced machine learning models that can accurately identify AI-generated images.

2.4 Summary

The literature review provides a thorough analysis of current developments in AI-generated facial synthesis, with a focus on the application of deep learning and GANs. It draws attention to the proliferation of AI-generated content, including text, audio, photos, and videos, that is used for unfortunate exploits like financial fraud

and smear campaigns. Because of its wide range of applications, creating realistic human faces has attracted the most attention. Although facial structures are inherently complicated, and each person's face is unique, recent advances in facial synthesis and semantic manipulation techniques have enabled the creation of artificial faces that are incredibly lifelike.

The use of GANs is the predominant method for creating artificial faces in machine learning. Despite the difficulties that GAN training brings, such as mode collapse and vanishing gradient problems, which can limit the variety and quality of faces generated, algorithmic developments have significantly improved the performance of both discriminator and generator networks.

These developments have led to the creation of exceptionally high-quality human faces, which often confuse the human participants and blur the line between artificial and real images. As a result of this advancement in GAN-based face synthesis, artificial intelligence-generated faces are now essentially indistinguishable from real ones. From a machine learning perspective, these advances highlight the amazing ability of GANs to discover complex patterns and nuanced details in image datasets, ultimately achieving a level of realism that was previously unattainable. These innovations have not only increased the potential of computer vision but have also sparked important discussions about the moral and societal implications of AI-generated material and warrant further study and attention in the machine learning community.

Rule-based and machine learning-based methods can be used to categorize the methods for identifying GAN-generated faces. In recent years, machine learning technologies have replaced rule-based technologies as the preferred strategy. This trend is partly due to the rapid developments in AI-generated image synthesis models, which have made it more difficult to recognize patterns or artifacts typical of AI-generated images based solely on predetermined criteria or standards.

Adopting a machine learning-based strategy is significant due to the problems caused by the increasing number of fake human faces online. The spread of false information, fake news, and moral and legal issues are sometimes due to the false belief that AI-generated images are real. Such misidentifications can have far-reaching effects on people, companies, and society. Additionally, AI-generated image recognition is a relatively young and developing topic. This means that there is still a lot of potential for study and improvement, especially when it comes to creating more sophisticated machine learning models specifically designed for the goal of identifying AI-generated photos. The continued development of AI technology highlights the importance of state-of-the-art machine learning models and the need for continuous research and innovation in the field of AI-generated image identification.

2.5 Gaps in Literature

1. The lack of data on the effectiveness of AI-generated image identification techniques represents a significant research gap. Although the review explores multiple approaches to identifying AI-generated photos, it does not

provide information on how effective these strategies are. Future work should focus on conducting thorough performance evaluations of these detection methods and assess their robustness, accuracy, and false positive rates.

2. The review mentions the potential impact of misclassifying AI-generated photos, but a more thorough analysis of the ethical and societal implications would be beneficial. The impact of misidentification on people, businesses, and society in general should be addressed, as should privacy and security issues.
3. Due to the rapid development of AI-generated image synthesis techniques, many publicly available datasets are no longer useful for training machine learning models designed to categorize AI-generated images. These data sets often do not capture the potential of today's generative models, especially when it comes to creating synthetic content that is incredibly compelling and realistic. There is an urgent need to create and disseminate new ethically sourced datasets consistent with the current synthetic media landscape to address the serious problems posed by the increase in AI-generated images. These datasets should include a wide range of artificial intelligence (AI)-generated content, such as deepfakes, altered faces and other types of synthetic photography.

3 Methodology

This chapter describes the research methods used to achieve the study goals. The thorough methodological framework of this study is detailed below.

3.1 Research Methodology

Research technique is critical to guiding the course of study and providing an organized strategy for answering research questions. This section describes the research methodology used to support study design, data collection, analysis, and interpretation. This study uses a quantitative strategy to ensure a comprehensive understanding of the classification of real and GAN-generated human faces. In quantitative research, numerical data is collected and analysed using statistical methods. In educational research, quantitative methods can be used to collect and analyse data on a range of topics, including surveys, secondary data collection, sampling, and experimental methods (Gorard, 2001). Since this is the classification problem and describes what occurs and not why it occurs, the use of quantitative methods is appropriate in this study.

Table 1 Methodologies in Research

Sl.No	Quantitative Research	Qualitative Research
1	Used to test or confirm theories or assumptions using numbers and graphs	Used to understand concepts and experiences based on accumulated knowledge
2	The analysis is based on mathematics and other statistical analysis	Analysis based on summarizing, categorizing, and interpreting

This dissertation uses an inductive approach with a quantitative technique for research. This study uses two datasets, each with at least 10,000 images, to distinguish between real and artificial human faces. The images are labelled to indicate whether they are real or fake, and this information is used to train a machine learning model. The inductive approach involves collecting data, analysing it, and formulating ideas or generalizations considering the results. Liu (2016) describes a general inductive strategy that can be applied to provide significant interpretive power to make sense of data in educational research. This strategy is practical and adaptable. Because the goal of this study is to uncover patterns and trends in the data that can be used to distinguish between actual and synthetic human faces, this approach is ideal.

3.2 Data Search Strategy and Collection

Identifying and sourcing appropriate datasets that match research goals is the foundation of any data-driven research. Appropriate datasets serve as the foundation upon which models are built and evaluated (Ponce et al., 2006). Choosing diverse and representative datasets is crucial to avoid bias and ensure model fairness (Pagano et al., 2023). The intended approach is to obtain the required datasets containing both real and synthetic faces, and then train a machine learning model in Python to effectively categorize both types of faces. The datasets used, Person Face Dataset and Flickr-Faces-HQ Dataset (FFHQ), were selected after careful consideration of their relevance and suitability. Specific criteria were created to ensure the quality and suitability of the datasets. First, the data sets had to contain many both real human faces and GAN-generated faces. Second, the data sets must be publicly available and well documented so that the research process is reproducible and transparent. Finally, ethical considerations were key, and datasets containing real human faces required appropriate consent and usage rights.

Person Face Dataset contains synthetic images generated by GANs. These photos demonstrate GAN's extraordinary ability to create incredibly compelling humanoid faces. In addition, the FFHQ dataset collected through the Flickr photo sharing site contains legitimate photos of real human faces taken in different environments. The combination of these datasets provides a balanced representation

of both real and GAN-generated human faces. Research ensures the quality, integrity, and representativeness of the datasets collected by adhering to strict criteria.

3.3 Ethical Implications

The advent of machine learning and image categorization techniques has raised several ethical issues, particularly when dealing with sensitive data such as human faces. This section discusses the ethical issues of classifying real and GAN-generated human faces. It emphasizes the importance of responsible data use, possible biases, and the societal consequences of misclassification.

Data ethics addresses a variety of issues, including informed consent, privacy, and data ownership. **This study is categorised low risk** as it uses two publicly available datasets. One obtained from Kaggle under CC0: Public Domain license, consisting of 10,000 photographs of human faces created from a GAN model (Xu, 2021). Despite its synthetic origins, it makes you think about the implications of creating hyper-realistic images of people who don't exist. Such photos could be used for various purposes, including disinformation or identity theft. The ethical application of synthetic data requires a delicate balance between scientific progress and social responsibility. The other dataset is Flickr-Faces-HQ (FFHQ) used in the StyleGAN architecture development by Karras et al. (2019) which contains 70,000 high-quality images of real human faces with a resolution of 1024x1024 under the Creative Commons BY-NC-SA 4.0 license. The use of the FFHQ dataset, a secondary dataset, which consists of publicly uploaded Flickr photos, raises concerns about user consent to share images. Although the data set complies with the Flickr's Terms of Service and only images collected under permissive licenses, it is important to consider potential ethical concerns associated with the use of photos posted in an online community.

Machine learning algorithms are prone to bias in training data, which can lead to skewed or unfair predictions. There may be bias associated with facial classification based on gender, race, or other demographics. Considering this issue, efforts have been made to reduce bias in data collection and model training. The importance of algorithmic fairness has been highlighted in the literature (Mitchell et al., 2021). To counteract prejudice, diverse and representative datasets covering a wide range of human appearances were used. In addition, the study used data augmentation and balanced sampling to ensure that real and GAN-generated faces were represented equally. Using fairness-aware metrics during model evaluation helps reduce bias even further.

Beyond the technical correctness, image categorization models have a social consequence. Misclassification of the human face can have serious consequences, especially in critical situations such as security or law enforcement (Lu, 2023). Misidentification could lead to wrongful allegations or violations of individual rights. When introducing such models, it is crucial to examine potential downstream

impacts as well as accountability measures. The study recognizes the need to provide explicit explanations for model predictions to facilitate human interpretation and monitoring (Gilpin, 2018). Efforts are made to ensure that the model's decisions are interpretable and consistent with human values, thereby increasing trust and accountability.

The cornerstone of this research are ethical considerations that guide the responsible and conscientious use of data and technology. Data set selection, bias reduction measures, and accountability procedures all contribute to the ethical framework of the study. As technology advances, the project aims to contribute to the discussion on the ethical implications of machine learning applications, particularly image classification. The next sections cover the execution platform, data preparation, feature engineering, machine learning models, scoring matrix, and the methodology used, which explains how these components are influenced by and aligned with ethical considerations.

3.4 Execution Platform

Successful application of the research methodology depends on a solid execution platform that enables efficient code development, experimentation, and model evaluation. Research is conducted on a Windows machine utilizing the Python programming language and the PyCharm integrated development environment (IDE) for its versatility and productivity.

The project involves working with large datasets of real and GAN-generated human faces. The work of preprocessing, feature extraction, and model training on such large amounts of data requires a solid execution platform capable of efficiently managing computational resources. While cloud options were considered, practical problems arose when dealing with massive amounts of data. Uploading large files to cloud servers posed challenges in terms of data transport speed and storage space. In addition, the computing power required to process large amounts of data presented difficulties as cloud solutions were limited by the given resources.

Given the difficulties associated with cloud-based solutions, an offline execution strategy was carefully selected as the optimal alternative. This decision is based on the freedom and cost-effectiveness that an offline environment offers. Initial experimentation and model development on a local computer offers the benefit of full control over computing resources and ensuring that processing limitations do not impede progress. When dealing with large datasets, in this case around 30 GB of image data, and when optimal use of local hardware resources is critical, this strategy is invaluable. Additionally, the offline execution strategy is a cost-effective alternative as it does not require costly cloud subscriptions or dedicated infrastructure. While the research starts with an offline execution technique, the approach is scalable in the future. If necessary, the study implementation can be easily transferred to cloud-based technologies. The offline environment acts as a

steppingstone for fundamental experimentation and modelling. As research evolves and data processing requirements increase, the code base and methodology can be moved to cloud platforms with more processing and storage capacity.

Libraries and their role

- NumPy and Pandas are core data manipulation and analysis libraries. NumPy's array-based operations and Pandas data structures ensure efficient data set processing and enable smooth pre-processing and feature development.
- OpenCV is useful for image preprocessing and editing. It offers a wide range of image resizing, normalization, and data expansion operations, ensuring consistency and compatibility across the dataset.
- Keras, a TensorFlow library component, simplifies the creation and training of deep learning models. Its high-level API allows rapid prototyping of neural network architectures and allows the creation of sophisticated models with minimal coding effort.
- Streamlit: This library is used to develop an interactive online application for model visualization and evaluation. Its easy-to-use interface allows researchers and stakeholders to interact with the trained model and gain insights into predictions and performance.
- Matplotlib and Seaborn: These visualization libraries help visualize data distributions, model performance, and function plots by allowing the creation of informative charts and graphs.
- Scikit-learn: Scikit-learn is a library that provides a comprehensive set of machine learning algorithms, tools, and measurements. Its user-friendly interface facilitates the integration of various categorization models and evaluation measures.
- VGG16: The pre-trained deep learning model VGG16 is used for feature extraction. Its convolutional layers collect hierarchical image features, enabling effective learning of the categorization representation. It is a popular option and serves as a solid baseline model in computer vision research due to its simple architecture, compact convolution filters, and layer organization (Tammina, 2019).

3.5 Data Preprocessing

Data preparation is critical to improving the quality and reliability of machine learning models (Njeri, 2022). In the context of this study, data pre-processing refers to a set of activities that transform raw image data into an organized and usable format for further analysis and modelling.

3.5.1 Data Organisation

The research leverages two publicly available datasets: the Person Face Dataset and the FFHQ dataset. Manually labelling each image was a critical step in

this process. Because this is a supervised machine learning problem, manual labelling allows for the creation of accurate ground truth labels. Each image was classified as "Real" or "GAN-generated" based on its source. This manual labelling process provides a solid foundation for model training and evaluation. To do this, a renaming method was used that uniquely identifies each image while retaining its class affiliation. The renaming used a formula-based approach to generate serial numbers that reflect image classification.

Table 2 Individual Image renaming criteria

Class	**Formula to check the class**
Real	(image name - 1) % 9 = 0
GAN-generated	(image name - 2) % 9 = 0

This methodical renaming ensures that each image is clearly identified and organized for later pre-processing and analysis.

3.5.2 Image Standardization and Resizing

The variety of image dimensions is one of the central problems when dealing with image data. The convergence and accuracy of the model can be affected by different image sizes (Viertel & König, 2022). To solve this problem, a crucial step in data preprocessing is the standardization of image size. All photos are resized to a standard resolution of 224 x 224 pixels to ensure they are all the same size.

3.5.3 Normalization and Intensity Scaling

The normalization of pixel values, along with the standardization of the image dimensions, is crucial for the stability and convergence of the model (Kusunose, 2022). By dividing each pixel value by 255.0, the image pixel values are normalized to a range of [0,1]. Normalization reduces the impact of changing lighting conditions by ensuring that pixel intensity values are consistent across datasets. This preprocessing phase is critical for achieving optimal performance during model training using gradient-based optimization.

3.5.4 Data Augmentation

Data enrichment is a strategy to improve model generalization while reducing overfitting. Augmentation creates new training samples by applying different transformations to existing images. Techniques such as rotating, mirroring and zooming are used to supplement the data set. Augmentation makes the training set more diverse and variable, allowing the model to acquire robust features and patterns (Shoaib, 2022). By generating enhanced versions of the original photos, the model improves its ability to deal with a variety of real-world circumstances.

3.5.5 Train-Test-Validation Splits

A critical step in the data preprocessing workflow is the partitioning of the data set into discrete subsets. To enable successful model training, hyperparameter tuning, and unbiased evaluation, the datasets are divided into training, validation, and test sets. To ensure a representative sample, the distribution of real and GAN-generated images is carefully maintained across different subsets. The train set is used to train the model, the validation set is used to fine-tune hyperparameters, and the test set is used to evaluate model performance.

The research ensures that the datasets conform to the machine learning models through methodical organization and thorough pre-processing approaches. These pre-processing processes improve the resilience and reliability of the following machine learning phases.

3.6 Feature Engineering

Feature engineering is critical to improving the rendering performance of machine learning models. Feature engineering enables models to identify detailed patterns and correlations by extracting meaningful and relevant features from raw data, improving their predictive capabilities (Pramanik, 2021). In the context of this study, feature engineering is an important step in classifying real and GAN-generated human faces.

Convolutional neural networks (CNNs) have emerged as powerful methods for extracting features from image data. A CNN is a type of deep learning architecture designed to automatically learn hierarchical and discriminatory features from images (Santos et al., 2022). A pre-trained VGG16 model was used for feature extraction in this study. The VGG16 model, is known for its ability to capture complex visual aspects (He, 2020). Features were retrieved from the block5_conv2 layer of the VGG16 model. This layer collects high-level semantically significant features and is therefore useful for distinguishing between actual and GAN-generated human faces.

The feature extraction method is to pass each image through the VGG16 model and retrieve the feature vector generated by the block5_conv2 layer. This vector encapsulates the main visual information of the image. The feature vector is then flattened to create a one-dimensional representation, converting the image into a collection of numerical features.

The derived feature vectors from the VGG16 model are inherently high-dimensional, which can lead to computational inefficiencies and overfitting. To solve these challenges, dimensionality reduction techniques have been employed to minimize the feature space while retaining the most informative components. Dimensionality reduction was achieved using principal component analysis (PCA). The primary axes of variation in the data are identified via PCA and the feature vectors are projected onto a lower dimensional subspace. This subspace preserves

the highest variance of the original data and enables the generation of a compact yet informative feature representation.

Since not all extracted features contribute equally to model performance, feature selection is an important part of feature engineering. A feature importance analysis was performed to identify the most distinctive features for distinguishing between actual and GAN-generated human faces. To rank and select features based on their importance to the classification task, techniques such as recursive feature elimination (RFE) and mutual information were used.

3.7 Machine Learning Model

A crucial step towards an accurate and robust classification of real and GAN-generated human faces is the selection of suitable machine learning models. Machine learning models serve as the backbone of the classification pipelines, using the data retrieved to discover differentiated patterns and make informed predictions. This section describes the machine learning models used in this study, highlighting their architecture, training procedures, and importance in achieving the research goals.

3.7.1 Binary Classification

Given the nature of the problem (a binary classification task), various well-known machine learning models were evaluated for implementation. The main goal was to find models that understand complex relationships and can distinguish between actual and GAN-generated human faces. Because of its demonstrated success in image classification tasks, Convolutional Neural Network (CNN) is chosen for this task.

3.7.2 Convolutional Neural Network (CNN)

CNNs have transformed image categorization by learning hierarchical features directly from pixel values. CNNs are characterized by their ability to capture local patterns via convolutional layers while learning global relationships via pooling and fully connected layers. A custom CNN architecture was created, consisting of numerous layers of convolution and pooling, followed by dense layers of classification. The CNN architecture was designed to take advantage of the spatial hierarchies visible in images to successfully capture differentiators.

Table 3 CNN Architecture

Layer	Type	Output Shape	Param #
dense	Dense	(None, 512)	51380736
batch_normalization	Batch Normalization	(None, 512)	2048
dropout	Dropout	(None, 512)	0
dense_1	Dense	(None, 256)	131328
batch_normalization_1	Batch Normalization	(None, 256)	1024
dropout_1	Dropout	(None, 256)	0
dense_2	Dense	(None, 128)	32896
batch_normalization_2	Batch Normalization	(None, 128)	512
dropout_2	Dropout	(None, 128)	0
dense_3	Dense	(None, 64)	8256
batch_normalization_3	Batch Normalization	(None, 64)	256
dropout_3	Dropout	(None, 64)	0
dense_4	Dense	(None, 1)	65

Total params: 51557121 (196.67 MB), Trainable params: 51555201 (196.67 MB), Non-trainable params: 1920 (7.50 KB)

3.7.3 Model Training and Hyperparameter Tuning

The model underwent extensive training to discover discriminative patterns from manufactured features. Model-specific hyperparameters such as the learning rate in CNN were optimized during the training phase. To determine the ideal parameter values that resulted in higher classification performance, hyperparameter tuning approaches such as grid search and cross-validation were used. Transfer learning and fine-tuning strategies were explored to exploit the potential of pre-trained models and take advantage of their learned representations. The CNN model architecture was built using weights from a pre-trained VGG16 model and then refined with data from the research challenge. The model was fine-tuned to match the learned features to the intricacies of real and GAN-generated human face classification tasks.

3.8 Evaluation Matrix

Scoring machine learning models is an important part of any classification process as it provides insight into how the model is performing and enables more informed decisions. The precise classification of real and GAN-generated human faces depends on the rigorous evaluation of the machine learning models used in this study. This section examines the scoring matrix used to assess model performance, including the metrics used, their interpretation, and their relevance to the research goals.

3.8.1 Evaluation Metrics

To assess the performance of the machine learning models in categorizing real and GAN-generated human faces, a set of comprehensive assessment measures were

used. These measurements include accuracy, precision, recall, F1 score, and area under the Receiver Operating Characteristic Curve (AUC-ROC).

To assess the performance of the machine learning models in categorizing real and GAN-generated human faces, a set of comprehensive assessment measures were used. These measurements include accuracy, precision, recall, F1 score, and area under the Receiver Operating Characteristic Curve (AUC-ROC).

1. Accuracy is a fundamental parameter that quantifies the proportion of correctly identified examples relative to the total number of occurrences. While accuracy provides a general picture of model performance, it can be misleading in unbalanced datasets where one class is dominant (Brabec, 2018).
2. Precision measures the proportion of truly positive predictions out of all positive predictions. This is particularly important when the cost of false positives is significant, such as when GAN-generated images are falsely flagged as genuine (Dinga et al., 2019).
3. Recall (Sensitivity): Recall calculates the proportion of correct positive predictions compared to correct positive cases. This is especially important when the cost of false negatives is significant, such as when actual photos are incorrectly labeled as GAN generated.
4. The F1 score is the harmonious mean of precision and memory and provides a balanced picture of a model's performance. This is useful when a balance between precision and recall is required.
5. AUC-ROC: The AUC-ROC metric evaluates the model's ability to distinguish between two classes at different probability thresholds. It calculates the area under the curve by plotting the true positive rate (TPR) versus the false positive rate (FPR). AUC-ROC is particularly useful for unbalanced datasets as it provides a single value that summarizes the model's discriminative ability.

3.8.2 Interpretation and Analysis

The interpretation of these measurements in the context of this study is crucial. High accuracy can mean good performance, but it can be deceiving with unbalanced data sets. Precision and recall are important indicators of the model's ability to reliably classify real and GAN-generated faces while minimizing false positives and false negatives. The F1 score combines these two factors and provides an overall assessment of the effectiveness of the model. AUC-ROC is resistant to class imbalances and provides information on how well the model discriminates between classes.

3.8.3 Cross-Validation and Test Set Evaluation

Cross-validation techniques were used to confirm the reliability and generalizability of the model. The dataset was divided into training and validation sets using K-fold cross-validation to ensure that each instance was used for both training and validation. Overfitting was avoided and the assessment results were

more robust. In addition, the machine learning models were evaluated using a separate test set that was not used during model building. This provided objectivity in the review and revealed the model's performance on previously undisclosed data, reflecting its potential usefulness in practice.

3.9 Methodology

Once downloaded, the data set is pre-processed by normalizing the pixel values and scaling the photos to a standard size. The model is then trained after a model architecture suitable for the task has been determined. After the model has been trained, its performance is tested on the validation set. The performance of the model can be further improved by adjusting the hyperparameters such as learning rate, stack size, and number of epochs. The quality of the data, the adequacy of the model architecture, and the effectiveness of the hyperparameter tuning have a significant impact on the classifier's performance; As a result, the development process of an image classifier will be iterative, involving numerous rounds of experimentation and refinement.

4. Results

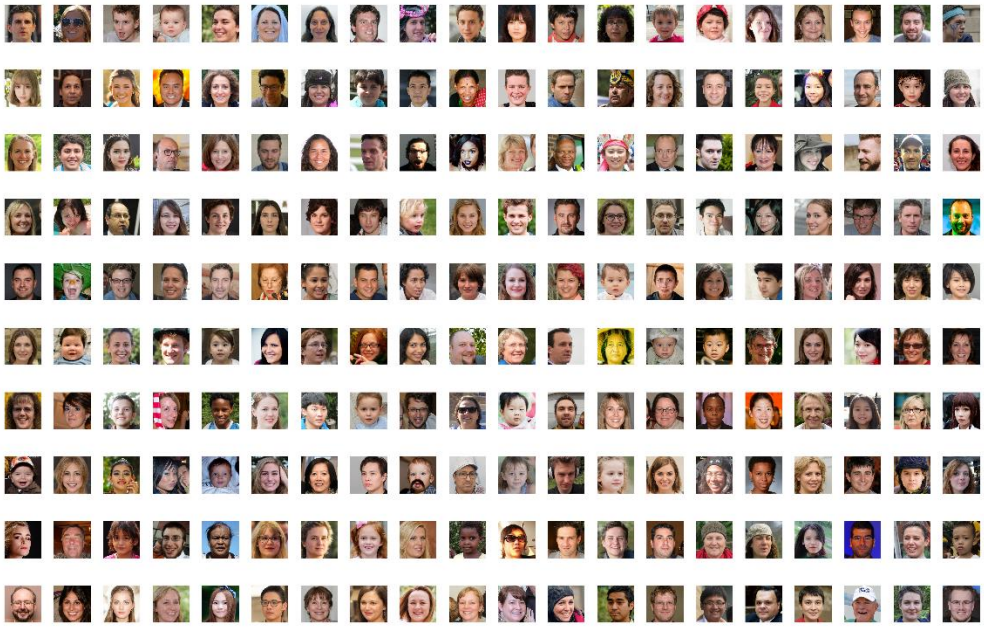
4.1 Exploratory Data Analysis

This section provides the key insights and findings from the Exploratory Data Analysis (EDA) datasets consisting of real and GAN-generated human faces. The following points provide a quick overview of the EDA performed on the data set.

4.1.1 Dataset Visualization

In the first step of EDA, the dataset is visually examined. For closer inspection, random selection of images was selected. Visually examining these selections revealed variations between the images like colour profile, different image orientations, different lighting conditions, and the presence of background noise and artifacts in certain photos. These results typically provide important insights into the real-world nature of the datasets. In the current dataset, these differences in visual properties are not as drastic and required only minor cleaning and standardization.

Figure 1 Random selection of images



Almost 5% of the randomly selected images did not meet the acceptable threshold for direct use in the model. However, the remaining 95% of the photos met the required uniformity and quality standards so they could be immediately integrated into the model. This significant variance in image selection highlights the need for a more rigorous image selection method to ensure that all data inputs consistently meet the model's prescribed standards, improving overall performance and reliability.

The dataset included 10,000 real images and 10,000 images generated by GANs. Photos were methodologically divided into training, testing, and validation sets to support effective machine learning. The training set contained 12,000 unique photos, while the remaining 8,000 were divided equally between the test and validation sets. These splits were made with extreme caution to avoid biased splits. This targeted method protects against imbalances in the training set and improves the robustness and fairness of the performance evaluation and training process of our machine learning models.

Table 4 Train-Test-Validation Splits

Counts	**Training Set**	**Test Set**	**Validation Set**	**Total**
Total Image Count	12,000	4,000	4,000	20,000
Real Image Count	6,000	2,000	2000	10,000

Key image statistics were methodologically calculated to improve the understanding of the quantitative characteristics of the dataset. Calculating essential image statistics such as mean pixel values, standard deviation, and colour

distribution analysis is critical when compiling the collection. These statistical indicators are systematically calculated to minimize and eliminate bias in the training set. This ensures that our machine learning models are trained on a fairer and more representative data set, reducing the possibility of biased performance differences, and promoting fair and accurate learning outcomes.

Table 5 Image Statistics

Number of Images	12,000
Average Image Size (KB)	429.442
Mean of Mean Pixel Values	112.699

The colour channels of the image graph provide interesting insights into the data set. A wide range of skin tones, lighting conditions and background settings are represented in the graphic through differences in colour profiles between photos. Although these differences are rather small, they highlight the relevance of the datasets to the real world by reflecting the underlying complexity of human faces in different environments. These variations highlight the need for a strong model that can detect such subtleties.

Figure 2 Colour Channels of Train set

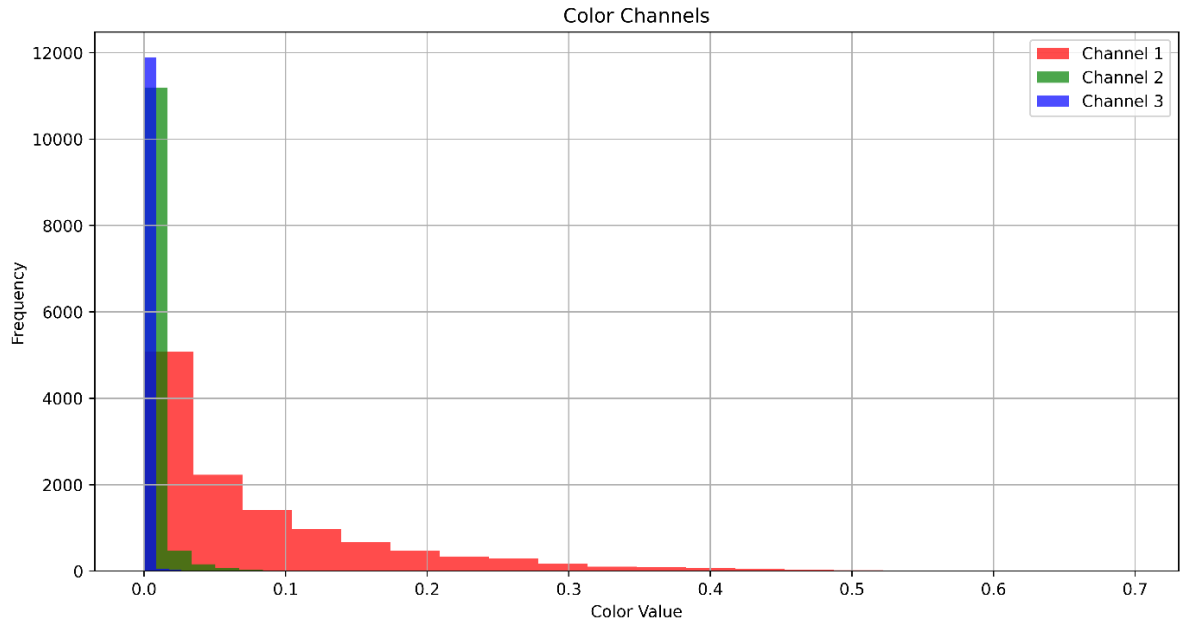
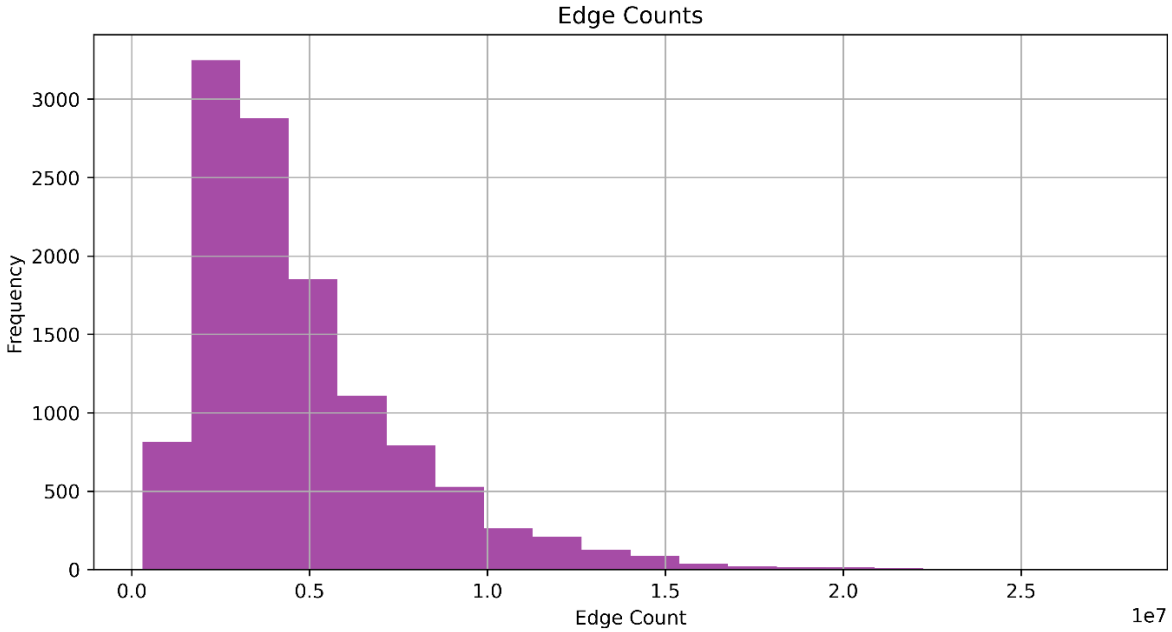
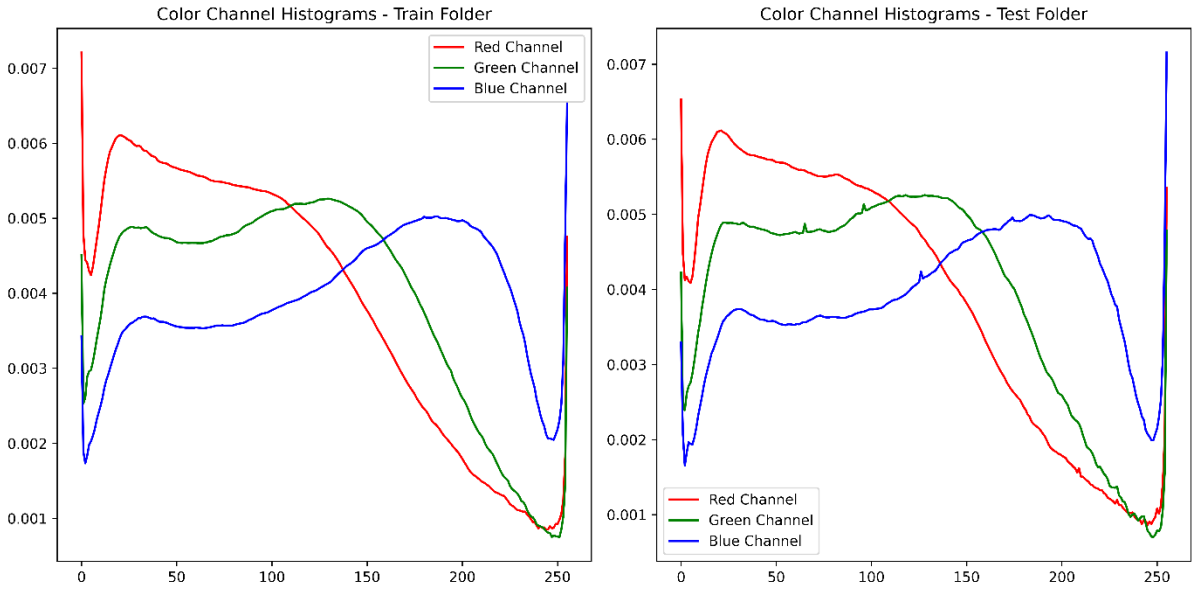


Figure 3 Edge Counts of the Train Set



The image edge count graph provides a unique perspective. It highlights the presence of sharp edges in the images that can represent shapes and features on the face. What is interesting is that the diagram has different numbers of edges, which could indicate different image compositions and orientations. This result highlights the model's ability to reproduce facial representations with varying levels of detail, which is a critical component of classification accuracy.

Figure 4 Colour Channels of Train and Test Splits



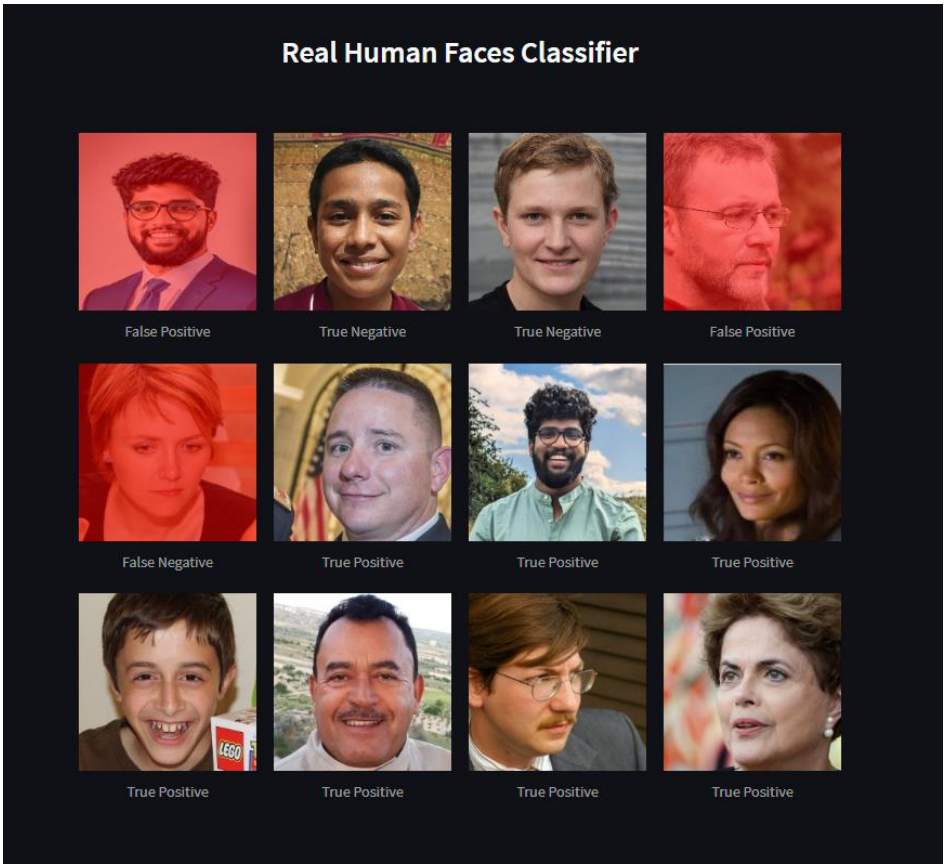
Similarity tests of images were also conducted as part of the EDA to remove similar images from the dataset. Detecting such pairs was critical to identifying likely duplicate or near-duplicate images, which, if ignored, could introduce biases in

model training and evaluation. By identifying these similarities, datasets diversity is managed while avoiding over-representation of individual image examples.

4.2 Streamlit Dashboard

The developed dashboard loads images from the dashboard folder, extract features from images, and then uses the trained model to predict the legitimacy of the images. Images with red overlay are incorrectly classified. The current dataset images as well as new set of test images can also be used fort this dashboard.

Figure 5 Snapshot of the Streamlit Dashboard



4.3 Error Analysis

Error analysis is a crucial component in assessing the performance of a classification model. It requires a thorough investigation of the models' misclassifications and an understanding of why these errors occur. By conducting error analysis, one can learn more about the limitations of the model and potential areas for development.

4.3.1 Types of Misclassifications:

Misclassifications related to the classification of real and synthetic human faces can be broadly divided into the following categories: False positives: Images that were classified as real but were GAN-generated. False negatives are images that have been labelled as synthetic but are real.

5. Discussions

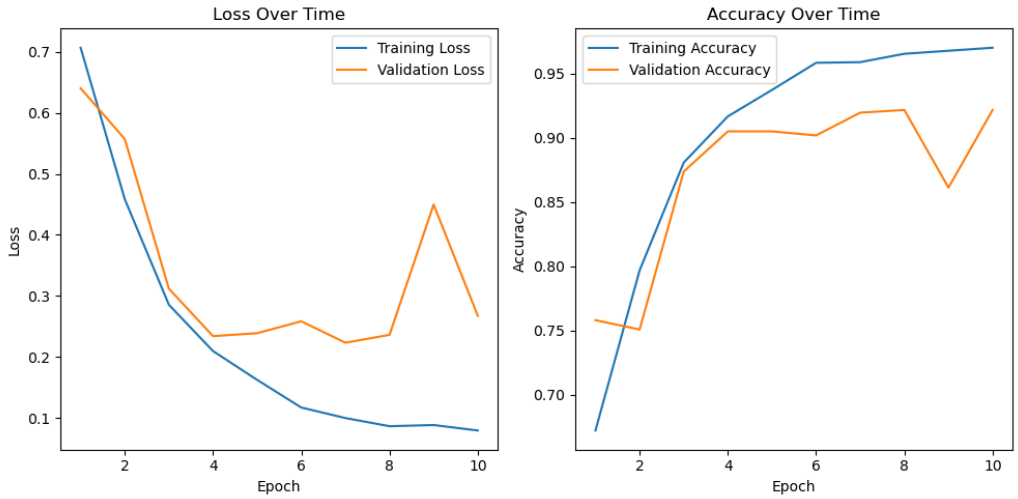
5.1 Accuracy and Generalization

The model's astonishing 98.5% accuracy in image categorization is a remarkable, especially for tasks with complex visual inputs. This level of precision demonstrates the model's ability to recognize detailed patterns and features in photos, making precise predictions on previously unseen data. Importantly, the model's ability to automatically extract relevant visual information, rather than the names of the images, is crucial to success. Using the VGG-16 model as a feature extractor is crucial to the performance of our model. VGG-16 was trained on large datasets to capture basic visual properties. This allows images to be categorized into numerous categories, including distinguishing between real and fake images.

Table 6 Model Metrics

Metric	**Real Images**	**GAN-Generated**
Accuracy	0.985	0.985
Precision	0.982	0.988
Recall	0.988	0.982
F1 Score	0.985	0.985

Figure 6 Accuracies and Loss Over Epochs



In particular, the precision of the model not necessarily indicate overfitting. Overfitting is a typical problem in machine learning where a model is heavily fitted to the training data, resulting in poor generalization. On the other hand, the consistent accuracy of our model across training, testing, and validation datasets demonstrates its robustness and lack of overfitting.

5.2 Data Distribution and Imbalance

Before diving into the model's performance, it's essential to consider the data distribution across different datasets. We have a total of 20,000 images, with 10,000 being real images and the remaining 10,000 being of another class. This balanced dataset ensures that the model is not biased toward any class, contributing to its generalization ability.

5.3 Challenges in Classifying Orientation

While our model excels in classifying images as real or another class, it encounters difficulties when classifying images based on the orientation of the person in the picture. Specifically, it struggles to correctly identify instances where there is another person in the image or when the person is facing at an angle.

Figure 7 Random Misclassified Images

5.3.1 Multiple Persons in the Image

One of the difficulties the model faces is distinguishing between photos with a single person and those with many people. Because it focuses on identifying the presence of a person rather than their number, the model may misclassify photos containing more than one person as real images. To solve this problem, future improvements could include using object detection techniques to accurately count the number of people in an image.

5.3.2 Angled Orientation

Another area where the model struggles is that the person in the image is not looking directly at the camera, but at an angle. This can lead to misclassifications because the model may not correctly capture people's distinguishing characteristics. Advanced approaches such as posture estimation could be incorporated into the model to better understand an individual's orientation toward improving performance in such cases.

5.4 Ethical Implication of the research

Using secondary datasets downloaded from the internet to categorize actual and AI-generated human faces raises several ethical questions. The use of existing datasets initially raises concerns about the consent and privacy of the people depicted in such photos. The privacy rights of people whose photos are included in the datasets could be violated without the necessary authorization or anonymization. It is important and hence the data used in this project is ethically sourced and sufficiently anonymized, as maintaining privacy is a key ethical concept.

Ethical requirements for AI initiatives also include transparency and responsible disclosure. To promote ethical AI development, it is critical to clearly describe the sources and procedures used in collecting and preparing datasets and provide insight into model limitations. In summary, although the experiment hopefully helps to improve image categorization, it is important to recognize the moral implications of using online secondary datasets. To maintain the highest ethical standards in AI research and development, ethical issues such as privacy, bias, openness, and responsible disclosure is carefully considered.

6. Conclusion

6.1 Summary

In this study, an efficient and optimal method of classifying real human faces from AI generated faces, a CNN model was built to categorize real and GAN-produced human faces. The model managed to classify real and GAN-generated images with an accuracy of around 98.5%. Literature review was conducted to identify the current trends in artificial face generation, its problems, and methods for recognizing the faces generated by AI. The VGG-16 model was used to identify the distinguishing features. Thorough data preprocessing, comprehensive exploratory data analysis, and intelligent error analysis have resulted in the creation of a strong machine learning model with excellent accuracy. The model is evaluated using various metrics to evaluate the performance of the model on the test data. Possible ways to improvise the current methods of AI-generated faces were evaluated and the final model was implemented. The final phase of the model development process involved creating a dynamic dashboard using Streamlit. This is intended to carry out image categorization in real time. This interactive dashboard combines the power of our trained model with a pre-trained VGG-16 model to process a random selection of existing photos from the dataset as well as completely new images, demonstrating the model's capabilities in an easy-to-use interface. A balanced dataset and solid training enabled our model to achieve outstanding accuracy. Overall, the performance of our model is promising for practical image classification tasks, but there is still room for development in dealing with tricky situations.

6.2 Future Directions and Improvements

To further enhance our model's performance, several avenues for improvement can be explored:

1. Data Augmentation: The model can better handle orientation differences and multiple people in photos by expanding the diversity of training data through data augmentation approaches.
2. Pose Estimation: Incorporating pose estimation models can increase classification accuracy by correctly identifying the orientation of people in photos.
3. Object detection: The use of methods to count the number of people in photos can help minimize misclassifications.
4. Human Feedback: To pinpoint misclassifications and improve the model accordingly, feedback can be obtained from human annotators to identify misclassifications.
5. Faces transformed from real images: Instead of creating fake faces, techniques such as using machine learning algorithms to completely alter the structure of real faces can go unnoticed.
6. Explainability and interpretability: While high accuracy is undoubtedly a useful indicator, it is also important to consider how interpretable and explainable our models' predictions are. To build confidence in the model's predictions, one must understand why it made a particular choice, especially in complex situations. To see which areas of the image the model focuses on when predicting, strategies such as gradient-based class activation maps (CAM) can be used. This can provide insights into the model's decision-making process.

6.3 Critical Reflection

Developing a project to categorize real and AI-generated human faces using a CNN model was an exciting and challenging adventure that led to critical self-reflection on my learning process. Recognizing the dynamic nature of the field is one of the key takeaways from this endeavour. Throughout the project, I experienced the rapid development of AI-generated material, forcing me to constantly update my knowledge and change my approach. This experience highlighted the importance of staying current in the rapidly changing field of artificial intelligence.

The initiative also made me aware of the difficulties and moral dilemmas associated with AI-generated material. The more I delved into the intricacies of distinguishing between real and artificial photos, the more I became aware of the potential impacts of misclassification, such as the spread of false information and privacy issues. This made me more aware of the ethical obligations that come with working in deep learning and AI and motivated me to approach future initiatives with more ethical rigor. The project also demonstrated the value of thorough data preparation. A crucial part of the model development was collecting and curating a data set that accurately represented both actual and AI-generated faces. I discovered the importance of diverse and high-quality data because it has a direct impact on the success of the model. This project improved my technological skills while expanding my understanding of the moral and practical implications of using AI-generated material.

7. References

Abdolahnejad, M., & Liu, P. (2020). Deep learning for face image synthesis and semantic manipulations: a review and future perspectives. Artificial Intelligence Review, 53(8), 5847–5880. https://doi.org/10.1007/s10462-020-09835-4

Alamayreh, O. (2021, September 10). Detection of GAN-synthesized street videos. arXiv.org. https://arxiv.org/abs/2109.04991

Bang, Y. O. (2021, December 22). DA-FDFtNet: Dual Attention Fake Detection Fine-tuning Network to Detect Various AI-Generated Fake Images. arXiv.org. https://arxiv.org/abs/2112.12001

Bank, D. (2020, March 12). Autoencoders. arXiv.org. https://arxiv.org/abs/2003.05991

Brabec, J. (2018, December 4). Bad practices in evaluation methodology relevant to class-imbalanced problems. arXiv.org. https://arxiv.org/abs/1812.01388

Burton, J. W., & Soare, S. R. (2019). Understanding the Strategic Implications of the Weaponization of Artificial Intelligence. IEEE. https://doi.org/10.23919/cycon.2019.8756866

Choi, Y. (2017, November 24). StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. arXiv.org. https://arxiv.org/abs/1711.09020

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative Adversarial Networks: An Overview. IEEE Signal Processing Magazine, 35(1), 53–65. https://doi.org/10.1109/msp.2017.2765202

Dang, H. (2019, October 3). On the Detection of Digital Face Manipulation. arXiv.org. https://arxiv.org/abs/1910.01717

Ding, Z., Jiang, S., & Zhao, J. (2022). Take a close look at mode collapse and vanishing gradient in GAN. 2022 IEEE 2nd International Conference on Electronic Technology, Communication and Information (ICETCI). https://doi.org/10.1109/icetci55101.2022.9832406

Dinga, R., Bw, P., Dj, V., Schmaal, L., & Af, M. (2019). Beyond accuracy: Measures for assessing machine learning models, pitfalls and guidelines. bioRxiv (Cold Spring Harbor Laboratory). https://doi.org/10.1101/743138

Du, H. (2023, January 9). Enabling AI-Generated Content (AIGC) Services in Wireless Edge Networks. arXiv.org. https://arxiv.org/abs/2301.03220

Gilpin, L. H. (2018, May 31). Explaining Explanations: An Overview of Interpretability of Machine Learning. arXiv.org. https://arxiv.org/abs/1806.00069

Goodfellow, I. (2014). Generative Adversarial Nets. https://papers.nips.cc/paper_files/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html

Gorard, S. (2001). Quantitative Methods in Educational Research: The Role of Numbers Made Easy. ResearchGate. https://www.researchgate.net/publication/44832401_Quantitative_Methods_in_Educational_Research_The_Role_of_Numbers_Made_Easy

Gu, X., & Angelov, P. (2018). Semi-supervised deep rule-based approach for image classification. Applied Soft Computing, 68, 53–68. https://doi.org/10.1016/j.asoc.2018.03.032

He, X. (2020). Heterogeneous Transfer Learning for Hyperspectral Image Classification Based on Convolutional Neural Network. https://www.semanticscholar.org/paper/Heterogeneous-Transfer-Learning-for-Hyperspectral-He-Chen/84e5ef965a7cec991912d70ef2222f0f8a9fe517

Jordan, B. (2009). Blurring Boundaries: The “Real” and the “Virtual” in Hybrid Spaces. Human Organization, 68(2), 181–193. https://doi.org/10.17730/humo.68.2.7x4406g270801284

Karras, T. (2017, October 27). Progressive Growing of GANs for Improved Quality, Stability, and Variation. arXiv.org. https://arxiv.org/abs/1710.10196

Karras, T. (2019, December 3). Analyzing and Improving the Image Quality of StyleGAN. arXiv.org. https://arxiv.org/abs/1912.04958

Karras, T., Laine, S., & Aila, T. (2019). A Style-Based Generator Architecture for Generative Adversarial Networks. IEEE. https://doi.org/10.1109/cvpr.2019.00453

Kas, S. (n.d.). Do you know if I'm real? An experiment to benchmark human recognition of AI-generated faces. AIS Electronic Library (AISeL). https://aisel.aisnet.org/bled2020/7/

Khan, F., Pasha, M. A., & Masud, S. (2021). Advancements in Microprocessor Architecture for Ubiquitous AI—An Overview on History, Evolution, and Upcoming Challenges in AI Implementation. Micromachines, 12(6), 665. https://doi.org/10.3390/mi12060665

Kingma, D. P., & Welling, M. (2019). An Introduction to Variational Autoencoders. Foundations and Trends in Machine Learning, 12(4), 307–392. https://doi.org/10.1561/2200000056

Koldewyn, K., Hanus, P., & Balas, B. (2013). Visual adaptation of the perception of "life": Animacy is a basic perceptual dimension of faces. Psychonomic Bulletin & Review, 21(4), 969–975. https://doi.org/10.3758/s13423-013-0562-5

Kreps, S. E., McCain, R. M., & Brundage, M. (2020). All the News That's Fit to Fabricate: AI-Generated Text as a Tool of Media Misinformation. Journal of Experimental Political Science (Print), 9(1), 104–117. https://doi.org/10.1017/xps.2020.37

Kusunose, T. (2022). Facial Expression Emotion Recognition Based on Transfer Learning and Generative Model. https://www.semanticscholar.org/paper/Facial-Expression-Emotion-Recognition-Based-on-and-Kusunose-Kang/c9c64e374b543208fcb5b26fd357c90cb7da26e3

Li, L. (2016). Using Generic Inductive Approach in Qualitative Educational Research: A Case Study Analysis. Journal of Education and Learning, 5(2), 129. https://doi.org/10.5539/jel.v5n2p129

Li, Y. (2019, September 27). Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics. arXiv.org. https://arxiv.org/abs/1909.12962

Liu, C. (2021, December 17). AI-Empowered Persuasive Video Generation: A Survey. arXiv.org. https://arxiv.org/abs/2112.09401

Lu, Z. (2023). Seeing is not always believing: Benchmarking Human and Model Perception of AI-Generated Images. https://www.semanticscholar.org/paper/Seeing-is-not-always-believing%3A-Benchmarking-Human-Lu-Huang/b9b308428283e4899afabee10fab6cf234de8fd7

Meel, P., & Vishwakarma, D. K. (2020). Fake news, rumor, information pollution in social media and web: A contemporary survey of state-of-the-arts, challenges and opportunities. Expert Systems With Applications, 153, 112986. https://doi.org/10.1016/j.eswa.2019.112986

Mitchell, S., Potash, E., Barocas, S., D'Amour, A., & Lum, K. (2021). Algorithmic Fairness: Choices, Assumptions, and Definitions. Annual Review of Statistics and Its Application, 8(1), 141–163. https://doi.org/10.1146/annurev-statistics-042720-125902

Nightingale, S. J. (2022). AI-synthesized faces are indistinguishable from real faces and more trustworthy. PNAS. https://doi.org/10.1073/pnas.2120481119

Njeri, N. R. (2022). Data Preparation For Machine Learning Modelling. https://www.semanticscholar.org/paper/Data-Preparation-For-Machine-Learning-Modelling-Njeri/c968aee2d44a304f5a3c1058b768792887cd820f

Ozkan, S. (2018, June 22). KinshipGAN: Synthesizing of Kinship Faces From Family Photos by Regularizing a Deep Face Network. arXiv.org. https://arxiv.org/abs/1806.08600

Pagano, T. P., Loureiro, R. B., Lisboa, F. V. N., Peixoto, R. M., De Sousa Guimarães, G. A., Cruz, G. O. R., Araujo, M. M., Santos, L. L. D., Cruz, M. a. S., De Oliveira, E. L. S., Winkler, I., & Nascimento, E. G. S. (2023). Bias and Unfairness in Machine Learning Models: A Systematic

Review on Datasets, Tools, Fairness Metrics, and Identification and Mitigation Methods. Big Data and Cognitive Computing, 7(1), 15. https://doi.org/10.3390/bdcc7010015

Person Face Dataset (thispersondoesnotexist). (2021, December 12). Kaggle. https://www.kaggle.com/datasets/almightyj/person-face-dataset-thispersondoesnotexist

Pierre Berger Diccan.com, A Impasse Marie Louise, Maisons-Laffitte (France). (n.d.). Digital creation | Proceedings of the 2014 Virtual Reality International Conference. ACM Other Conferences. https://dl.acm.org/doi/10.1145/2617841.2620704

Ponce, J., Berg, T. L., Everingham, M., Forsyth, D., Hebert, M., Lazebnik, S., Marszałek, M., Schmid, C., Russell, B., Torralba, A., Williams, C. K. I., Zhang, J., & Zisserman, A. (2006). Dataset Issues in Object Recognition. In Lecture Notes in Computer Science (pp. 29–48). https://doi.org/10.1007/11957959_2

Pramanik, R. (2021). Comparative Analysis of Mobile Price Classification Using Feature Engineering Techniques. https://www.semanticscholar.org/paper/Comparative-Analysis-of-Mobile-Price-Classification-Pramanik-Agrawal/87225a4b14a5b126fa01aa2a1442470d27c21b58

Rossi, S. (2022, September 15). Are Deep Learning-Generated Social Media Profiles Indistinguishable from Real Profiles? arXiv.org. https://arxiv.org/abs/2209.07214

SAGE PublicationsSage UK: London, England. (n.d.). Cambridge Analytica's black box - Margaret Hu, 2020. Sage Journals. https://doi.org/10.1177/2053951720938091

Santos, C. F. G. D., De Souza Oliveira, D., Passos, L. A., Pires, R. G., Santos, D. O., Valem, L. P., Moreira, T. P., Santana, M. C. S., Roder, M., Papa, J. P., & Colombo, D. (2022). Gait Recognition Based on Deep Learning: A Survey. ACM Computing Surveys, 55(2), 1–34. https://doi.org/10.1145/3490235

Shoaib, M. R. (2022). Efficient deep learning models for brain tumor detection with segmentation and data augmentation techniques. https://www.semanticscholar.org/paper/Efficient-deep-learning-models-for-brain-tumor-with-Shoaib-Elshamy/f33557f39b5af448f24078d5703288bb19676904

Simon, H. A. (1993). Anecdotes-a very early expert system. IEEE Annals of the History of Computing, 15(3), 64–68. https://doi.org/10.1109/85.222851

Su, J. (2018, November 18). GAN-QP: A Novel GAN Framework without Gradient Vanishing and Lipschitz Constraint. arXiv.org. https://arxiv.org/abs/1811.07296

Sun, Y. (2014, June 18). Deep Learning Face Representation by Joint Identification-Verification. arXiv.org. https://arxiv.org/abs/1406.4773

Tammina, S. (2019). Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images. International Journal of Scientific and Research Publications, 9(10), p9420. https://doi.org/10.29322/ijsrp.9.10.2019.p9420

Viertel, P., & König, M. (2022). Pattern recognition methodologies for pollen grain image classification: a survey. Journal of Machine Vision and Applications, 33(1). https://doi.org/10.1007/s00138-021-01271-w

Wang, R. (2019, September 13). FakeSpotter: A Simple yet Robust Baseline for Spotting AI-Synthesized Fake Faces. arXiv.org. https://arxiv.org/abs/1909.06122

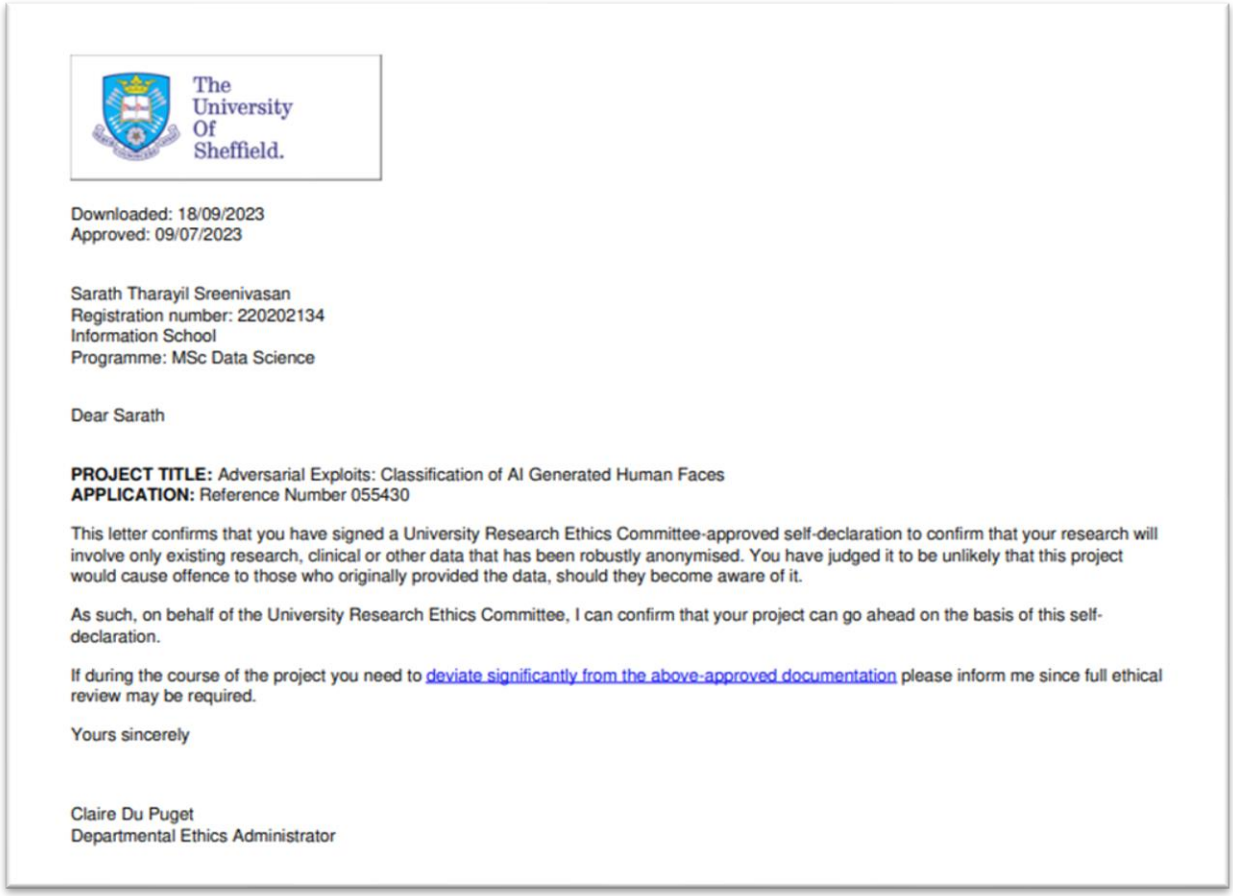
Wang, T., & Lin, Y. (n.d.). CycleGAN with Better Cycles. www.tongzhouwang.info.

Xiang, J., Mi, T., & Wang, X. (2022). Adaptation in face animacy perception: An event-related potential study. Neuropsychologia, 165, 108118. https://doi.org/10.1016/j.neuropsychologia.2021.108118

Yegemberdiyeva, G., & Amirgaliyev, B. (2021). Study Of AI Generated And Real Face Perception. 2021 IEEE International Conference on Smart Information Systems and Technologies (SIST). https://doi.org/10.1109/sist50301.2021.9465908

8. Appendix

8.1 Appendix 1 (Ethics Approval)



The University Of Sheffield.

Downloaded: 18/09/2023
Approved: 09/07/2023

Sarath Tharayil Sreenivasan
Registration number: 220202134
Information School
Programme: MSc Data Science

Dear Sarath

PROJECT TITLE: Adversarial Exploits: Classification of AI Generated Human Faces
APPLICATION: Reference Number 055430

This letter confirms that you have signed a University Research Ethics Committee-approved self-declaration to confirm that your research will involve only existing research, clinical or other data that has been robustly anonymised. You have judged it to be unlikely that this project would cause offence to those who originally provided the data, should they become aware of it.

As such, on behalf of the University Research Ethics Committee, I can confirm that your project can go ahead on the basis of this self-declaration.

If during the course of the project you need to deviate significantly from the above-approved documentation please inform me since full ethical review may be required.

Yours sincerely

Claire Du Puget
Departmental Ethics Administrator

8.2 Appendix 2 (Code)

00_files_rename.py

```
import os

# Define the paths to the folders containing the images
real_people_folder = r"C:\Users\iamsa\Documents\Dissertation\Datasets\real"
gan_faces_folder = r"C:\Users\iamsa\Documents\Dissertation\Datasets\fake"
real_factor = 1
fake_factor = 2
# Function to rename images based on series
def rename_images(folder_path, series_number):
    for idx, filename in enumerate(sorted(os.listdir(folder_path))):
        if filename.endswith('.jpg') or filename.endswith('.png'):
            new_name = f'{series_number + 9 * idx:09d}.jpg'
            os.rename(os.path.join(folder_path, filename),
os.path.join(folder_path, new_name))


# Rename images in the real people folder using Series 1
rename_images(real_people_folder, real_factor)
```

```
# Rename images in the GAN generated faces folder using Series 2
rename_images(gan_faces_folder, fake_factor)
```

01_test_train_val_splits.py

```
import os
import random
import shutil

# Set a random seed for reproducibility
random_seed = 42
random.seed(random_seed)

# Define the paths to the folders containing real and fake images
real_people_folder = r"C:\Users\iamsa\Documents\Dissertation\Datasets\real"
gan_faces_folder = r"C:\Users\iamsa\Documents\Dissertation\Datasets\fake"

# Define paths for train, validation, and test folders
train_folder = r"C:\Users\iamsa\Documents\Dissertation\Directories\train"
val_folder = r"C:\Users\iamsa\Documents\Dissertation\Directories\val"
test_folder = r"C:\Users\iamsa\Documents\Dissertation\Directories\test"

# Total number of images desired in each set
total_images_per_set = 20000

# Train-validation-test split ratios (adjust as needed)
train_ratio = 0.6  # 60% of the data for training
val_ratio = 0.2    # 20% of the data for validation, the rest for testing

# Function to clean and create folders
def clean_and_create_folder(folder_path):
    if os.path.exists(folder_path):
        shutil.rmtree(folder_path)
    os.makedirs(folder_path)

# Clean and create train, validation, and test folders
clean_and_create_folder(train_folder)
clean_and_create_folder(val_folder)
clean_and_create_folder(test_folder)

# Function to copy and split images
def split_images(src_folder, dest_folder, num_images):
    image_files = os.listdir(src_folder)
    selected_images = random.sample(image_files, num_images)

    for image in selected_images:
        src_path = os.path.join(src_folder, image)
        dest_path = os.path.join(dest_folder, image)
        shutil.copy(src_path, dest_path)

# Determine the number of real and fake images needed based on the total
num_real_images = total_images_per_set // 2
num_fake_images = total_images_per_set // 2

# Calculate the number of images for train, validation, and test sets
num_real_train = int(num_real_images * train_ratio)
num_real_val = int(num_real_images * val_ratio)
num_real_test = num_real_images - num_real_train - num_real_val

num_fake_train = int(num_fake_images * train_ratio)
```

```
num_fake_val = int(num_fake_images * val_ratio)
num_fake_test = num_fake_images - num_fake_train - num_fake_val

# Copy and split real images
split_images(real_people_folder, train_folder, num_real_train)
split_images(real_people_folder, val_folder, num_real_val)
split_images(real_people_folder, test_folder, num_real_test)

# Copy and split fake images
split_images(gan_faces_folder, train_folder, num_fake_train)
split_images(gan_faces_folder, val_folder, num_fake_val)
split_images(gan_faces_folder, test_folder, num_fake_test)

print("Images successfully split into train, validation, and test sets.")
```

02_00_split_balance_check.py

```
import os

# Define a list of paths to the folders containing test images
test_folders = [
    r"C:\Users\iamsa\Documents\Dissertation\Directories\val",
    r"C:\Users\iamsa\Documents\Dissertation\Directories\test",
    r"C:\Users\iamsa\Documents\Dissertation\Directories\train",
    # Add more folder paths here as needed
]

# Function to check if an image is real or fake based on its name
def check_real_or_fake(image_name):
    number = int(image_name[:-4])  # Remove the file extension and convert
the remaining part to an integer

    if (number - 1) % 9 == 0:
        return "Real"  # Images from Series 1 are real
    elif (number - 2) % 9 == 0:
        return "Fake"  # Images from Series 2 are fake
    else:
        return "Unknown"  # Images with other numbers are not assigned to a
series

# Loop through each test folder
for test_folder in test_folders:
    num_real = 0
    num_fake = 0

    # Check real or fake for images in the current test folder
    for filename in os.listdir(test_folder):
        if filename.endswith('.jpg') or filename.endswith('.png'):  #
Adjust the file extensions as needed
            result = check_real_or_fake(filename)
            if result == "Real":
                num_real += 1
            elif result == "Fake":
                num_fake += 1
            #print(f"{filename}: {result}")

    # Calculate the total number of test images in the current folder
    total_test_images = num_real + num_fake

    # Calculate the percentage of real and fake images in the current
```

```
folder
    percentage_real = (num_real / total_test_images) * 100
    percentage_fake = (num_fake / total_test_images) * 100

    # Print the results for the current folder
    print(f"Folder: {test_folder}")
    print(f"Total Real Images: {num_real}")
    print(f"Total Fake Images: {num_fake}")
    print(f"Percentage of Real Images: {percentage_real:.2f}%")
    print(f"Percentage of Fake Images: {percentage_fake:.2f}%")
    print()
```

02_01_rgb_edges.py

```
import os
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import cv2
import pandas as pd

# Define the path to the folder containing your standardized face image
dataset
dataset_folder = r"C:\Users\iamsa\Documents\Dissertation\Directories\train"

# Initialize lists to store image statistics
image_files = []
color_histograms = []
edge_counts = []

# Process each face image in the dataset folder
for image_file in os.listdir(dataset_folder):
    if image_file.endswith(('.png', '.jpg', '.jpeg')):
        image_path = os.path.join(dataset_folder, image_file)

        # Open the image using Pillow (PIL)
        image = Image.open(image_path)
        image = np.array(image)  # Convert to numpy array

        # Calculate color histogram
        color_hist = cv2.calcHist([image], [0, 1, 2], None, [8, 8, 8], [0,
256, 0, 256, 0, 256])
        color_hist = color_hist.flatten()
        color_hist = color_hist / np.sum(color_hist)  # Normalize the
histogram

        # Calculate edge count using Canny edge detection
        gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
        edges = cv2.Canny(gray_image, 100, 200)
        edge_count = np.sum(edges)

        # Store image statistics in lists
        image_files.append(image_file)
        color_histograms.append(color_hist)
        edge_counts.append(edge_count)

# Create a DataFrame to store the image statistics
image_stats_df = pd.DataFrame({
    'Image File': image_files,
    'Color Histogram': color_histograms,
    'Edge Count': edge_counts
})
```

```
# Visualize color histograms as a stacked bar chart
colors = ['red', 'green', 'blue']
plt.figure(figsize=(12, 6))
for i, color in enumerate(colors):
    color_values = [hist[i] for hist in image_stats_df['Color Histogram']]
    plt.hist(color_values, bins=20, alpha=0.7, color=color, label=f'Channel
{i + 1}')

plt.xlabel('Color Value')
plt.ylabel('Frequency')
plt.title('Color Channels')
plt.legend()
plt.grid()
plt.savefig('Color_Channels.png', dpi=600, bbox_inches='tight')
plt.show()

# Histogram of Edge Counts
plt.figure(figsize=(10, 5))
plt.hist(image_stats_df['Edge Count'], bins=20, color='purple', alpha=0.7)
plt.xlabel('Edge Count')
plt.ylabel('Frequency')
plt.title('Edge Counts')
plt.grid()
plt.savefig('Edge_Counts.png', dpi=600, bbox_inches='tight')
plt.show()
```

02_02_color_channels.py

```
import os
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Function to calculate and plot color histograms
def plot_color_histograms(folder_path, folder_name, ax):
    # Initialize lists to store histograms for each channel
    red_histogram = []
    green_histogram = []
    blue_histogram = []

    # Iterate through images in the folder
    for filename in os.listdir(folder_path):
        if filename.endswith('.jpg'):
            image_path = os.path.join(folder_path, filename)
            image = cv2.imread(image_path)

            # Calculate histograms for each color channel
            hist_red = cv2.calcHist([image], [0], None, [256], [0, 256])
            hist_green = cv2.calcHist([image], [1], None, [256], [0, 256])
            hist_blue = cv2.calcHist([image], [2], None, [256], [0, 256])

            # Normalize histograms
            hist_red /= hist_red.sum()
            hist_green /= hist_green.sum()
            hist_blue /= hist_blue.sum()

            # Append histograms to the lists
            red_histogram.append(hist_red)
            green_histogram.append(hist_green)
            blue_histogram.append(hist_blue)
```

```
    # Convert lists to NumPy arrays for plotting
    red_histogram = np.array(red_histogram)
    green_histogram = np.array(green_histogram)
    blue_histogram = np.array(blue_histogram)

    # Plot histograms
    ax.plot(red_histogram.mean(axis=0), color='red', label='Red Channel')
    ax.plot(green_histogram.mean(axis=0), color='green', label='Green
Channel')
    ax.plot(blue_histogram.mean(axis=0), color='blue', label='Blue
Channel')

    ax.set_title(f'Color Channel Histograms - {folder_name}')
    ax.legend()

# Path to your image folders
folder1_path = r"C:\Users\iamsa\Documents\Dissertation\Directories\train"
folder2_path = r"C:\Users\iamsa\Documents\Dissertation\Directories\test"

# Create a single chart with color histograms for both folders
plt.figure(figsize=(12, 6))
ax1 = plt.subplot(1, 2, 1)
plot_color_histograms(folder1_path, 'Train Folder', ax1)

ax2 = plt.subplot(1, 2, 2)
plot_color_histograms(folder2_path, 'Test Folder', ax2)

plt.tight_layout()
plt.savefig('colorchannel.png', dpi=600, bbox_inches='tight')
plt.show()
```

03_feature_engineering.py

```
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.applications.vgg16 import VGG16, preprocess_input
from keras.models import Model
import cv2

# Load the VGG16 model without the top (classification) layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,
224, 3))

# Extract features from a specific layer
layer_name = 'block5_conv2'
feature_extractor = Model(inputs=base_model.input,
outputs=base_model.get_layer(layer_name).output)

# Set the path to your image folders and feature save folder
train_images_path   =
r"C:\Users\iamsa\Documents\Dissertation\Directories\train"
val_images_path     =
r"C:\Users\iamsa\Documents\Dissertation\Directories\val"
test_images_path    =
r"C:\Users\iamsa\Documents\Dissertation\Directories\test"
features_save_path  =
r"C:\Users\iamsa\Documents\Dissertation\Directories\features"

# Function to extract features from an image
```

```
def extract_features(image_path, feature_extractor):
    img = load_img(image_path, target_size=(224, 224))  # Load image and
resize
    img_array = img_to_array(img)  # Convert image to array
    img_array = np.expand_dims(img_array, axis=0)  # Expand dimensions for
the model
    img_array = preprocess_input(img_array)  # Preprocess the image
    features = feature_extractor.predict(img_array)  # Extract features
    return features.flatten()  # Flatten the features into a 1D array


# Set the target size for resizing
target_size = (224, 224)

# Function to resize images in a folder
def resize_images(folder_path, target_size):
    for filename in os.listdir(folder_path):
        image_path = os.path.join(folder_path, filename)
        img = cv2.imread(image_path)
        img_resized = cv2.resize(img, target_size)
        cv2.imwrite(image_path, img_resized)

# Resize images in train folder
resize_images(train_images_path, target_size)

# Resize images in test folder
resize_images(test_images_path, target_size)

# Resize images in validation folder
resize_images(val_images_path, target_size)

print("Images resized to 256x256.")


# Create directories for saving features
os.makedirs(features_save_path, exist_ok=True)

# Extract and save features for train images
for filename in os.listdir(train_images_path):
    image_path = os.path.join(train_images_path, filename)
    features = extract_features(image_path, feature_extractor)
    save_path = os.path.join(features_save_path,
f"{filename.split('.')[0]}.npy")
    np.save(save_path, features)

print("Features extracted and saved for train images.")


# Extract and save features for validation images
for filename in os.listdir(val_images_path):
    image_path = os.path.join(val_images_path, filename)
    features = extract_features(image_path, feature_extractor)
    save_path = os.path.join(features_save_path,
f"{filename.split('.')[0]}.npy")
    np.save(save_path, features)

print("Features extracted and saved for validation images.")


# Extract and save features for test images
for filename in os.listdir(test_images_path):
```

```
    image_path = os.path.join(test_images_path, filename)
    features = extract_features(image_path, feature_extractor)
    save_path = os.path.join(features_save_path,
f"{filename.split('.')[0]}.npy")
    np.save(save_path, features)

print("Features extracted and saved for test images.")
```

04_train_model.py

```
import os
import numpy as np
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense, BatchNormalization, Dropout
from keras.optimizers import Adam
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Set the path to your feature data
features_save_path =
r"C:\Users\iamsa\Documents\Dissertation\Directories\features"

# Load the features and generate labels based on image names
train_features = []
train_labels = []

for filename in os.listdir(features_save_path):
    if filename.endswith(".npy"):
        feature_path = os.path.join(features_save_path, filename)
        label = filename.split('.')[0]
        features = np.load(feature_path)
        train_features.append(features)

        # Generate labels based on the image name
        image_number = int(label)
        if (image_number - 1) % 9 == 0:
            train_labels.append(1)  # Real image
        elif (image_number - 2) % 9 == 0:
            train_labels.append(0)  # GAN-generated image

# Convert lists to numpy arrays
train_features = np.array(train_features)
train_labels = np.array(train_labels)

# Split the data into train and validation sets
X_train, X_val, y_train, y_val = train_test_split(train_features,
train_labels, test_size=0.2, random_state=42)

# Build the model
model = Sequential()
model.add(Dense(512, activation='relu',
input_shape=(train_features.shape[1],)))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(128, activation='relu'))
```

```
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(64, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer=Adam(lr=0.001), loss='binary_crossentropy',
metrics=['accuracy'])

# Print the model architecture
model.summary()

# Initialize lists to store training history
train_loss_history = []
train_accuracy_history = []
val_loss_history = []
val_accuracy_history = []

# Train the model and record history
epochs = 10
for epoch in range(epochs):
    history = model.fit(X_train, y_train, epochs=1, batch_size=32,
validation_data=(X_val, y_val))

    # Record training and validation loss and accuracy
    train_loss_history.append(history.history['loss'][0])
    train_accuracy_history.append(history.history['accuracy'][0])
    val_loss_history.append(history.history['val_loss'][0])
    val_accuracy_history.append(history.history['val_accuracy'][0])

    print(
        f"Epoch {epoch + 1}/{epochs} - Loss: {train_loss_history[-1]:.4f} -
Accuracy: {train_accuracy_history[-1]:.4f} - Val Loss: {val_loss_history[-
1]:.4f} - Val Accuracy: {val_accuracy_history[-1]:.4f}")

# Plot training and validation loss
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(range(1, epochs + 1), train_loss_history, label='Training Loss')
plt.plot(range(1, epochs + 1), val_loss_history, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Loss Over Time')
plt.legend()

# Plot training and validation accuracy
plt.subplot(1, 2, 2)
plt.plot(range(1, epochs + 1), train_accuracy_history, label='Training
Accuracy')
plt.plot(range(1, epochs + 1), val_accuracy_history, label='Validation
Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Accuracy Over Time')
plt.legend()

plt.tight_layout()
```

```
plt.show()

# Evaluate the model
val_predictions = model.predict(X_val)
val_predictions_classes = np.round(val_predictions).flatten()  # Round to 0
or 1
val_accuracy = accuracy_score(y_val, val_predictions_classes)
print("Validation Accuracy:", val_accuracy)

# Save the trained model
model_save_path =
r"C:\Users\iamsa\Documents\Dissertation\Directories\model"
os.makedirs(model_save_path, exist_ok=True)

model.save(os.path.join(model_save_path, 'model.h5'))
print("Trained model saved.")
```

05_perf_metrics.py

```
import os
import numpy as np
import matplotlib.pyplot as plt
from keras.models import load_model
from keras.preprocessing.image import load_img, img_to_array
from keras.applications.vgg16 import VGG16, preprocess_input
from keras.models import Model
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, precision_score, recall_score, f1_score, roc_curve, auc

# Load the trained model
model_path =
r"C:\Users\iamsa\Documents\Dissertation\Directories\model\model.h5"
loaded_model = load_model(model_path)

# Load the VGG16 model without the top (classification) layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,
224, 3))

# Set the path to your test images
test_images_path =
r"C:\Users\iamsa\Documents\Dissertation\Directories\test"

# Extract features from VGG16
def extract_features(image_array, feature_extractor):
    features = feature_extractor.predict(image_array)
    flattened_features = features.reshape(features.shape[0], -1)
    return flattened_features

layer_name = 'block5_conv2'

# Initialize lists for ROC curve
roc_labels = []
roc_scores = []

# Initialize counters for metrics
tp_real = 0
tn_real = 0
fp_real = 0
fn_real = 0
tp_gan = 0
tn_gan = 0
```

```
fp_gan = 0
fn_gan = 0

# Process test images and calculate metrics
for filename in os.listdir(test_images_path):
    if filename.endswith(".jpg") or filename.endswith(".png"):
        image_path = os.path.join(test_images_path, filename)

        # Determine ground truth label based on the given formula
        image_number = int(filename.split('.')[0])
        actual_label_real = (image_number - 1) % 9 == 0
        actual_label_gan = (image_number - 2) % 9 == 0

        # Load and preprocess the image for VGG16
        target_size = (224, 224)
        image = load_img(image_path, target_size=target_size)
        image_array = img_to_array(image)
        image_array = np.expand_dims(image_array, axis=0)
        image_array = preprocess_input(image_array)

        # Extract features using the VGG16 model
        feature_extractor = Model(inputs=base_model.input,
outputs=base_model.get_layer(layer_name).output)
        features = extract_features(image_array, feature_extractor)

        # Make predictions using the loaded model
        prediction = loaded_model.predict(features)

        # Compare predictions with actual labels
        predicted_label_real = prediction[0][0] > 0.95
        predicted_label_gan = not predicted_label_real

        if actual_label_real:
            if predicted_label_real:
                tp_real += 1
            else:
                fn_real += 1
        else:
            if predicted_label_real:
                fp_real += 1
            else:
                tn_real += 1

        if actual_label_gan:
            if predicted_label_gan:
                tp_gan += 1
            else:
                fn_gan += 1
        else:
            if predicted_label_gan:
                fp_gan += 1
            else:
                tn_gan += 1

        # Append ground truth label and prediction score for ROC curve
        roc_labels.append(actual_label_real)
        roc_scores.append(prediction[0][0])

# Calculate metrics for real images
accuracy_real = (tp_real + tn_real) / (tp_real + tn_real + fp_real +
fn_real) if (tp_real + tn_real + fp_real + fn_real) > 0 else 0
```

```
precision_real = tp_real / (tp_real + fp_real) if (tp_real + fp_real) > 0
else 0
recall_real = tp_real / (tp_real + fn_real) if (tp_real + fn_real) > 0 else
0
f1_score_real = 2 * (precision_real * recall_real) / (precision_real +
recall_real) if (precision_real + recall_real) > 0 else 0

# Calculate metrics for GAN-generated images
accuracy_gan = (tp_gan + tn_gan) / (tp_gan + tn_gan + fp_gan + fn_gan) if
(tp_gan + tn_gan + fp_gan + fn_gan) > 0 else 0
precision_gan = tp_gan / (tp_gan + fp_gan) if (tp_gan + fp_gan) > 0 else 0
recall_gan = tp_gan / (tp_gan + fn_gan) if (tp_gan + fn_gan) > 0 else 0
f1_score_gan = 2 * (precision_gan * recall_gan) / (precision_gan +
recall_gan) if (precision_gan + recall_gan) > 0 else 0

# Calculate ROC curve for all images
fpr, tpr, _ = roc_curve(roc_labels, roc_scores)
roc_auc = auc(fpr, tpr)

# Display metrics for real images
print("Metrics for Real Images:")
print("Accuracy:", accuracy_real)
print("Precision:", precision_real)
print("Recall:", recall_real)
print("F1 Score:", f1_score_real)

# Display metrics for GAN-generated images
print("\nMetrics for GAN-Generated Images:")
print("Accuracy:", accuracy_gan)
print("Precision:", precision_gan)
print("Recall:", recall_gan)
print("F1 Score:", f1_score_gan)

# Display ROC curve for all images
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area =
{:.2f})'.format(roc_auc))
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for All Images')
plt.legend(loc='lower right')
plt.show()
```

06_streamlit_app.py

```
import streamlit as st
import numpy as np
import os
from keras.models import load_model
from keras.preprocessing.image import load_img, img_to_array
from keras.applications.vgg16 import VGG16, preprocess_input
from keras.models import Model
from PIL import Image, ImageDraw

# Load the trained model
model_path =
r"C:\Users\iamsa\Documents\Dissertation\Directories\model\model.h5"
loaded_model = load_model(model_path)
```

```
# Load the VGG16 model without the top (classification) layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,
224, 3))


# Extract features from VGG16
def extract_features(image_array, feature_extractor):
    features = feature_extractor.predict(image_array)
    flattened_features = features.reshape(features.shape[0], -1)
    return flattened_features


layer_name = 'block5_conv2'


# Function to make predictions and display results
def predict_and_display(image_path):
    target_size = (224, 224)
    image = Image.open(image_path)
    image = image.resize(target_size)
    image_array = img_to_array(image)
    image_array = np.expand_dims(image_array, axis=0)
    image_array = preprocess_input(image_array)

    feature_extractor = Model(inputs=base_model.input,
outputs=base_model.get_layer(layer_name).output)
    features = extract_features(image_array, feature_extractor)

    prediction = loaded_model.predict(features)
    predicted_label = 1 if prediction[0][0] > 0.5 else 0

    # Get the image number from the filename
    filename = os.path.basename(image_path)
    image_number = int(filename.split('.')[0])

    # Determine actual class based on the formula
    actual_label_real = (image_number - 1) % 9 == 0
    actual_label_gan = (image_number - 2) % 9 == 0
    actual_label = 1 if actual_label_real else 0

    return image, predicted_label, actual_label


# Streamlit app
def main():
    import streamlit as st

    # Centered title using Markdown
    st.markdown("<h3 style='text-align: center;'>Real Human Faces
Classifier</h3>", unsafe_allow_html=True)
    st.markdown("<br>", unsafe_allow_html=True)

    # Provide the folder path manually
    folder_path =
r"C:\Users\iamsa\Documents\Dissertation\Directories\test_sample"

    image_files = [os.path.join(folder_path, filename) for filename in
os.listdir(folder_path) if
                   filename.endswith((".jpg", ".png"))]
    st.markdown(
```

```
        """
        <style>
            button[title^=Exit]+div [data-testid=stImage]{
                text-align: center;
                display: block;
                margin-left: auto;
                margin-right: auto;
                width: 100%;
            }
        </style>
        """, unsafe_allow_html=True
    )
    num_images = len(image_files)
    num_rows = min((num_images + 3) // 4, 4)

    for row in range(num_rows):
        col1, col2, col3, col4 = st.columns(4)

        for col, image_path in zip((col1, col2, col3, col4),
image_files[row * 4: (row + 1) * 4]):
            image, predicted_label, actual_label =
predict_and_display(image_path)

            true_positive = (predicted_label == 1 and actual_label == 1)
            true_negative = (predicted_label == 0 and actual_label == 0)
            false_positive = (predicted_label == 1 and actual_label == 0)
            false_negative = (predicted_label == 0 and actual_label == 1)

            overlay = Image.new("RGBA", image.size, (255, 0, 0, 128)) if
false_positive or false_negative else None
            if overlay:
                image = Image.alpha_composite(image.convert("RGBA"),
overlay)

            caption = ""
            if true_positive:
                caption += "True Positive\n"
            if true_negative:
                caption += "True Negative\n"
            if false_positive:
                caption += "False Positive\n"
            if false_negative:
                caption += "False Negative\n"

            col.image(image, caption=caption, use_column_width=True)


if __name__ == "__main__":
    main()
```

07_error_analysis.py

```
import os
import numpy as np
import shutil  # To move files
from keras.models import load_model
from keras.preprocessing.image import load_img, img_to_array
from keras.applications.vgg16 import VGG16, preprocess_input
from keras.models import Model
```

```
# Load the trained model
model_path =
r"C:\Users\iamsa\Documents\Dissertation\Directories\model\model.h5"
loaded_model = load_model(model_path)

# Load the VGG16 model without the top (classification) layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,
224, 3))

# Set the path to your test images
test_images_path =
r"C:\Users\iamsa\Documents\Dissertation\Directories\test"

# Define output folders for misclassified images
misclassified_real_folder =
r"C:\Users\iamsa\Documents\Dissertation\Directories\misclassified_real"
misclassified_gan_folder =
r"C:\Users\iamsa\Documents\Dissertation\Directories\misclassified_gan"

# Create the output folders if they don't exist
os.makedirs(misclassified_real_folder, exist_ok=True)
os.makedirs(misclassified_gan_folder, exist_ok=True)

# Extract features from VGG16
def extract_features(image_array, feature_extractor):
    features = feature_extractor.predict(image_array)
    flattened_features = features.reshape(features.shape[0], -1)
    return flattened_features

layer_name = 'block5_conv2'

# Process test images and find misclassified images
for filename in os.listdir(test_images_path):
    if filename.endswith(".jpg") or filename.endswith(".png"):
        image_path = os.path.join(test_images_path, filename)

        # Determine ground truth label based on the given formula
        image_number = int(filename.split('.')[0])
        actual_label_real = (image_number - 1) % 9 == 0
        actual_label_gan = (image_number - 2) % 9 == 0

        # Load and preprocess the image for VGG16
        target_size = (224, 224)
        image = load_img(image_path, target_size=target_size)
        image_array = img_to_array(image)
        image_array = np.expand_dims(image_array, axis=0)
        image_array = preprocess_input(image_array)

        # Extract features using the VGG16 model
        feature_extractor = Model(inputs=base_model.input,
outputs=base_model.get_layer(layer_name).output)
        features = extract_features(image_array, feature_extractor)

        # Make predictions using the loaded model
        prediction = loaded_model.predict(features)

        # Compare predictions with actual labels
        predicted_label_real = prediction[0][0] > 0.7
        predicted_label_gan = not predicted_label_real

        # Move misclassified images to respective folders
```

```
        if actual_label_real and not predicted_label_real:
            shutil.copy(image_path, os.path.join(misclassified_real_folder,
filename))
        elif actual_label_gan and not predicted_label_gan:
            shutil.copy(image_path, os.path.join(misclassified_gan_folder,
filename))
```