University of Hertfordshire UH

School of Physics, Engineering and Computer Science

MSc Data Science Project

7PAM2002-0509-2024

Department of Physics, Astronomy and Mathematics

Data Science FINAL PROJECT REPORT

Project Title:

Deep Learning Powered Analysis of AI- Generated vs Human- made Images

Student Name and SRN:

Ammu Ambika Ramachandran ID- 23006390

Supervisor: David Lagattuta

Date Submitted: 27/08/2025

Word Count: 5000

Github link : https://github.com/Ammu-AR/Deep-Learning--powered-Analysis-of-AI-generated-vs-Human-Made-Images

Kaggle Link: https://www.kaggle.com/datasets/saurabhbagchi/deepfake-image-detection

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science **in Data Science** at the University of Hertfordshire.

I have read the detailed guidance to students on academic integrity, misconduct and plagiarism information at Assessment Offences and Academic Misconduct and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6)

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed: AMMU AMBIKA RAMACHANDRAN

Student Name signature Ammu Ambika Ramachandran

Student SRN number: 23006390

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

Abstract

The increase in popularity of AI-created images and images demands a better system to identify them and differentiate them with original images. The three major deep learning architectures chosen to be investigated including the Traditional CNN, VGG16, and Xception in the detection of images generated by AI are assessed in the research activity. The Traditional CNN reflected a gradual rise of the accuracy of training up to 66.3 percent in ten-epochs, and the highest accuracy of 72.2 percent in validation. VGG16, with pre-trained weight, coded at it baseline with a slower initial improvement to achieve a result of the training accuracy of 58.9% training accuracy and validation accuracy was 73.6 percent. Conversely, the best performing model was the Xception model, which has depthwise separable convolutions, which displayed a training accuracy rise of 37.5 percent to 79.6 percent and a validation accuracy high of 76.3 percent. These findings imply that more heavenly setups such as Xception can be better at the classification task, but with a higher computational burden. It is identified in the study that there is a trade-off between efficiency and model complexity and thus a significant need to choose proper architectures depending on requirement specific to applications.

Table of Contents

List of Figures

1. Introduction

1.1. Background and Motivation

1.1.1. Background

With the fast development of deep learning and generative models, the field of image creation has substantially changed. In recent years, generative techniques like Generative Adversarial Networks (GANs), diffusion models, and transformer-based architectures have paved the way to come up with so much realistic images that in most cases seem to be created by human artists and photographers. Digital art, advertising, entertainment, and virtual reality are expanding opportunities provoked by such advances. Nevertheless, they have also presented some complications concerning the difference between images generated by artificial intelligence and those generated by humans.

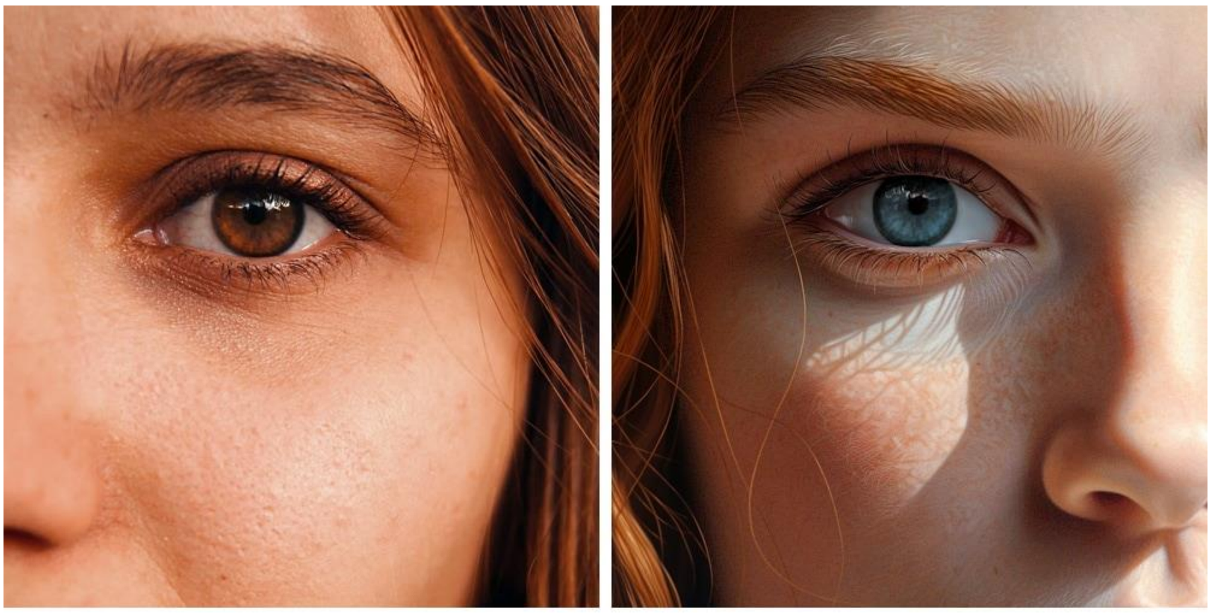


Figure 1 - Real vs. AI generated Image

Traditionally, images made by people have their own characteristic of creativity, deliberateness, and expression. In comparison, AI-produced uses images are based on the statistical learning of huge datasets where machines can mimic textures, colouring, and structures but do not necessarily understand or have intent. Even though the results are usually convincing to the eye, minor differences can occur in composition, detail consistency and representation of abstract concepts. The difference between the images produced with the help of AI and those pictures created by people is not only of academic concern but it is a practical matter as well. As an example, misinformation may be propagated in journalism and media, in case of confusing the synthetic image with the authentic one. The art industry presents challenges about originality and authorship and value when comparing AI-generated works to that of humans. Likewise, in cybersecurity and digital forensics, detection of AI created content is essential in deepfake detection and deterrence of AI-based malfeasance. With the ongoing advancement in deep learning models, these difficulties are further compounded, as further advanced techniques of analysis are needed.

The opportunity to analyze the said differences of the images produced by AI and that produced by people using AI-based approaches of deep learning opens the door of better understanding the advantages and the shortcomings of artificial creativity.

1.1.2. Motivation

The motivation behind this project is the fact that it is increasingly easy to confuse pictures created by AI with those portrayed by people, which affects society in various ways. As generative models have become more powerful their output is not only visually convincing but is also easy to get into the hands of the general audience. This brings I this issue of deception, originality in digital art and ethical questions of originality and ownership of creativity. This project aims at filling the gap between human and artificial creativity through the development of deep learning-based approaches to the analysis and classification of images, which can inform human behaviour with respect to the responsible use of AI and assist in preserving the truth, originality, and trust in visual media.

1.2. Research Questions

Do different deep learning architectures significantly differ in their trade-off between classification accuracy and computational efficiency for AI-Generated Images detection?

Null Hypothesis (H_0):

There is no significant difference in the trade-off between classification accuracy and computational efficiency among architectures for AI-Generated detection.

Alternate Hypothesis (H_1):

There is a significant difference in the trade-off between classification accuracy and computational efficiency among architectures for AI-Generated detection.

1.3. Research Aim

In the scope of the project, a deep learning-based framework is going to be designed and deployed that will be able to reliably and accurately classify AI-generated (fake) and human-made (real) images. The project is based on convolutional neural networks and transfer learning on the pre-trained models (such as VGG16 and Xception) in an attempt to automatically learn discriminative characteristics in the presented image data.

This direction is emphasized when using pre-processing the image, the data distribution balance, and augmentation methods to improve generalization. The project would attempt to test and train models with collections of labelled data of fake and real images to be able to calculate classification performance, compare various architectures, and determine the most effective strategies of detection. The eventual aim, however, is to provide one model that will not just apply significant predictive performance, but also fall under more general arts in digital forensics, media authentication and ethical computing of artificial intelligence in the creative sphere.

1.4. Project Objectives

1. Develop a robust deep learning pipeline for classifying AI-generated and human-made images.
2. Implement pre-processing techniques such as resizing, normalization, and augmentation to improve data quality.
3. Explore multiple convolutional architectures, including custom CNNs and transfer learning models like VGG16 and Xception.
4. Evaluate model performance using metrics such as accuracy, loss, and validation accuracy over several epochs.
5. Analyze strengths and limitations of different architectures in handling synthetic versus real images.
6. Address class imbalance through weighted training and augmentation strategies for fairer learning.
7. Contribute insights that support digital media authentication and responsible AI practices.

2. Review of Literatures

2.1 Detection of AI-Generated Images with Synthetic Datasets

Bird et al., (2024) has focused on improving the detection of AI-generated images using computer vision techniques. This study introduced a synthetic dataset created with latent diffusion models to replicate the ten classes of CIFAR-10, enabling direct comparison with real images. The synthetic images featured advanced visual details, such as realistic water reflections, challenging traditional classification methods. The task was framed as a binary classification problem—real versus fake images. A Convolutional Neural Network (CNN) was employed, with extensive experimentation involving 36 network configurations. After hyperparameter tuning, the best-performing model achieved a classification accuracy of 92.98%, demonstrating strong potential for real-world application.

Islam et al., (2025) assesses advanced classification models for detecting AI-generated images using the CIFAKE dataset and introduces two additional benchmark datasets, FakeGPT and PFake, designed to mirror CIFAKE's structure with specific generation prompts. Transfer learning was applied to train models on CIFAKE, with evaluations conducted across all three datasets. The research further investigates ensemble strategies, including stacking and meta-ensemble methods. The proposed Meta Ensemble eXplainable Fake Image Classifier (MEXFIC) outperformed individual models, achieving 94% and 96.61% accuracy on CIFAKE and PFake, respectively. These results emphasize the importance of ensemble learning for robust detection of synthetic content, particularly on complex datasets.

2.2 Data Augmentation and Integration of Synthetic with Real Images

Alabed et al., (2025) explored how integrating synthetic AI-generated images with real-world data affects image classification performance. Focusing on traffic-related datasets—potholes, speed bumps, and traffic lights—the researchers applied data augmentation and tested seven real-to-synthetic image ratios. DenseNet201, optimized using Adam, was used consistently across all tests. The findings revealed that a 1:3 ratio of real to synthetic data significantly improved both accuracy and model generalization, achieving a top validation accuracy of 97.36%. This suggests that carefully balanced synthetic data can effectively supplement real images, offering a practical solution for improving model performance in situations with limited annotated datasets.

Öztürk & Erçelebi, (2021) explores the impact of training deep learning models with synthetic images generated through a game engine to improve real-world image classification. Focusing on distinguishing birds from rotary-wing UAVs, the research evaluates the effectiveness of integrating a Corner Detection and Nearest Three-Point Selection (CDNTS) layer. Two experimental setups were compared: one using standard deep learning models and another incorporating the CDNTS layer. Results revealed significant performance gains, with AUC scores improving from 72% to 88.9% when the CDNTS layer was used. Extensive testing across 432 model configurations highlighted the CDNTS layer's potential to enhance classification accuracy and generalization.

2.3 Feature Extraction and Transfer Learning Approaches

Taspinar & Cinar (2024) addresses the challenge of distinguishing AI-generated images from real photographs using a dataset of 975 samples. Feature extraction was performed using three pre-trained convolutional neural network models: Squeeze Net, InceptionV3, and VGG19. These extracted features were then classified using machine learning techniques, including Artificial Neural Networks (ANN), K-Nearest Neighbours (KNN), and Support Vector Machines (SVM). Among the model combinations tested, InceptionV3 paired with ANN achieved the highest classification performance. While the results show promise for detecting AI-generated visuals, the study emphasizes the need for larger datasets to improve model reliability and tackle the complexity of this emerging classification task.

Killi et al., (2023) addresses the need for a reliable and adaptable model to detect image falsification in large datasets. It highlights the effectiveness of VGG19, a convolutional neural network known for strong performance in image classification tasks. The proposed approach leverages VGG19's deep architecture to analyze pixel-level features, enabling accurate detection of manipulated images. When tested, the model outperformed existing methods, achieving a notable accuracy of 96%. These findings suggest that VGG19 is a practical and effective solution for fake image detection, suitable for real-world applications where distinguishing between authentic and falsified visuals is increasingly critical.

2.4 Domain-Specific Applications (Faces, Art, and Lightweight Models)

Thet et al., (2025) investigates the classification of real versus AI-generated art images using deep learning techniques. Utilizing datasets of paintings, the researchers employed a convolutional neural network (CNN) framework, incorporating the CIFAR-10 architecture for enhanced performance. The synthetic artworks were generated using GANs to replicate diverse artistic styles. Comparative analysis revealed that the CIFAR-10-based model outperformed the standard CNN in classification accuracy. The findings demonstrate the potential of deep learning for identifying AI-generated content in artistic domains, contributing to advancements in digital forensics and automated content verification, particularly in distinguishing authentic human-made works from those produced by generative algorithms.

Kalaimani et al., (2024) focuses on developing an optimally configured Generative Adversarial Network (GAN) to effectively differentiate between real and AI-generated human faces. The research further introduces a robust classifier capable of achieving high accuracy in this binary classification task. Experimental results indicate that the proposed model attains up to 95% accuracy, outperforming several contemporary methods. The study not only demonstrates technical efficacy but also emphasizes the broader implications for AI ethics, identity verification, and digital security. By enhancing detection of synthetic content, the work contributes to combating misinformation and reinforcing public trust in AI-driven systems and facial recognition technologies.

Singh et al., (2024) presents the use of a MobileNet-based convolutional neural network (CNN) optimized for mobile and embedded platforms to differentiate between real and digitally altered images. Trained on a mixed dataset of authentic and artificial visuals, the model achieved a validation accuracy of 95.02% and a low validation loss of 0.1621, indicating strong predictive performance. Precision and recall scores of 0.9494 and 0.9511, respectively, reflect the model's reliability in minimizing false classifications. The findings underscore MobileNet’s effectiveness in detecting manipulated content, providing a lightweight and accurate solution for real-time image verification across diverse digital applications.

3. Ethics, Governance, and Risk

Generative artificial intelligence promises to be both disruptive and promising, especially regarding digital image authenticity. Although detection systems have the potential to increase confidence of online material, their advance reflects a number of ethical, governance and risk issues that have to be considered.

The most important ethical consideration is the gathering, use and treatment of data. On a big-data basis, there may be copyrighted material or images that have been scraped off of the internet with no permission given by the creators. Processing such data with lack of transparency may mean the violation of intellectual property rights and the loss of trust toward AI research among the population. There is also ethical issue of bias: when datasets are biased to specific areas (e.g., faces of specific demographics or scenes of specific regions) detection models would be biased and perform disproportionately, enforcing inequality or misrecognizing specific groups proportionally more. Solid records regarding the origin of the datasets, license agreements and demographic makeup are necessary to reduce these risks.

The governance structures allow the operationalisation of ethical research. Funding agencies and institutions, particularly at the institutional level, are also pressuring conductions to comply with regulations (e.g., data protection regulations, e.g., GDPR), institutional review boards and research transparency. In the case of AI-generated image detection, governance is also extended to responsible disclosure: making detectors publicly available is a broad societal benefit, but there is also a risk of contributing to bad actors finding workarounds by freely distributing detection technology. Therefore, containment measures and responsible communication of its findings constitutes an important aspect of governance.

There are technical and societal aspects that are involved in the risks that are related to this project. Technically, the detectors can overfit to small datasets and thus perform bad generalisation on real world ones. Adversarial examples may lead to genuine media being detected as fake and this may adversely affect individuals or organisations depending on credible evidence. Alternatively, false negative predictions might provide a pass to dangerous synthetic content-- misinformation or identity theft, to name two examples. Misuse of detection technologies in surveillant or censoring people is another danger to society, particularly in an environment of suppressed freedom of expression.

In general, ethical responsibility promotes a balance between disclosure and precaution, and it is critical that datasets are legal, models are thoroughly checked in terms of bias and stability, and governance systems are offered to avoid misuse. The solutions to these concerns are at the foundation of the effective resolution of these issues by addressing the given concerns so that the AI research of image detection can be used both to the benefit of the scholarly academic material as well as the trust of the society at large.

4. Research Methodology

The methodology utilized in the research is aimed at creating a transparent and reproducible framework used in the exploration of the task of detecting images generated by AI and those created by people. It has five key modules dataset curation, choice of model, training, performance testing, and explain ability measures. These elements together will allow the research to answer the formulated questions on accuracy, generalisation, robustness and ethical responsibility.

4.1. Dataset Information

In this research, the Deepfake Image Dataset can be used as the main source of images comprising both original and the ones presented by AI. Curated data is directly targeted at the needs of research in deepfake detection, digital forensics, and the corresponding AI applications. It contains high-quality images that are achieved through the use of the most advanced deepfake algorithms, emulating the complexities and minute artifacts of real-world synthetic data. The set of transformations and manipulations is presented in the dataset and puts the detection models to the test to be effective in generalizing data across various situations. The images come in form of a zip file and each archive is designed in such a way to easily access and combine them into machine learning pipelines. The data are organized according to a uniform naming scheme and directory layout, which enables to directly navigate within, and process the data programmatically, without ambiguity. The reproducibility is also facilitated by this organized method as it can be reproduced and rechecked by the other scientists.

4.2. Image handling and Pre-processing

The pre-processing phase in the deepfake detection pipeline makes the data image clean and consistent and ready to train algorithms. First, the files will be loaded by traversing the directories labelled as real and fake, and obtain the file paths and their categories into a tabular format of a Data Frame. This facilitates the ability to be systematized in accessing and controlling images.

All images are resized to a consistent dimension of 224x224 pixels so that they can be read into convolutional neural networks and with pretrained models such as VGG16 or Xception. Images are also taken to RGB format to keep up with constant colour representation since open CV by default reads images in BGR format. Next normalization is conducted on pixel values so that they maintain a range of 0 to 1 and this stabilizes learning and makes the learning faster.

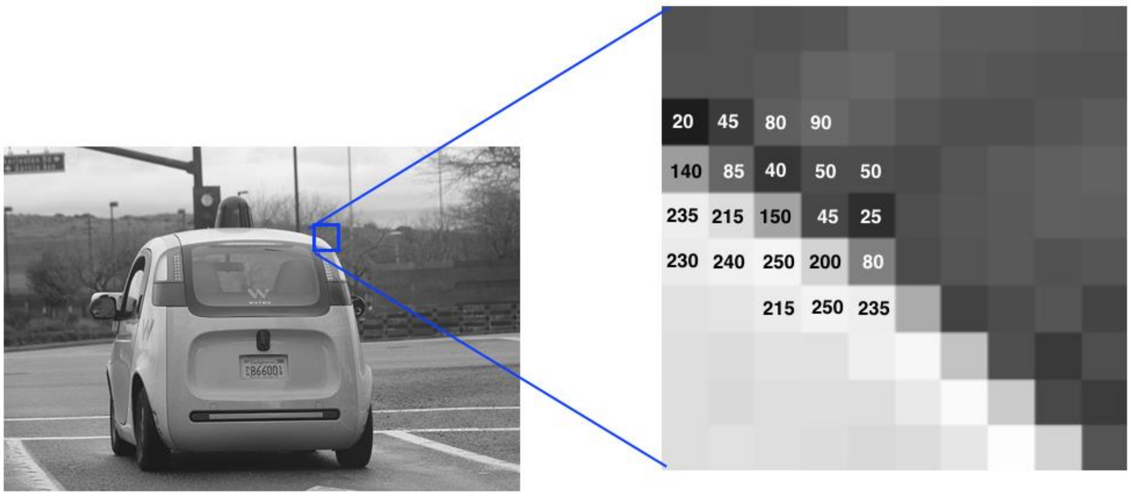


Figure 2 - Images as Grids of Pixels (Camacho C., 2018)

Labels are encoded through Label Encoding and then to one-hot encoding to ensure these are in a numerical format appropriate to multi-class classification. To stimulate model robustness, data augmentation will be added with rotation, zoom, width/height shift, and horizontal flip which artificially increases the variability of the dataset and minimizes overfitting. Lastly, the data is divided into training and test data and class balance is maintained to have unbiased testing.

4.3. Data Augmentation

In this project, data augmentation is used to augment the type of training dataset to learn how to generalize better and avoid overfitting the model. The Image Data Generator of Keras is applied to add real-time modifications to the images as part of training.

All augmentation algorithms are used: rotation, zooming, width and height shift, and flipping horizontally. In particular, at random, the images are subject to a random rotation within a range of 10 degrees, to represent slight motion of the camera or the subject. A zoom in the scale of 10% enables the model to identify objects at varying scales. Horizontal and order variations are accommodated in the network through shift of position and orientation. With these variations, the model is exposed to more input patterns without the use of more raw data. This can be extremely valuable in the area of deepfake detection where even minor shifts in facial characteristics or features can have a significant effect on results. The combination of all of these factors is that data augmentation enhances the robustness and accuracy of the model.

4.4. Traditional CNN Architecture

Convolutional Neural Network (CNN) is a particular kind of neural network employed to operate and examine visual information, i.e., photos or videos. It normally has three distinct varieties of layers, in its architecture which include the convolutional layers, pooling layers, and the fully connected layers.

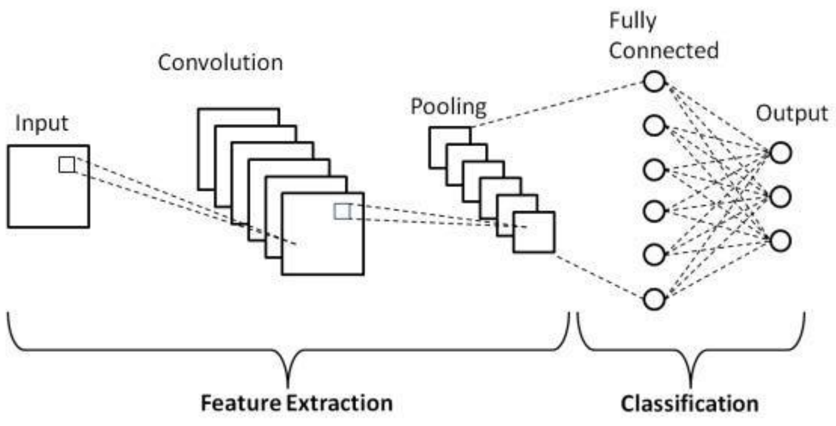


Figure 3 - Architecture of a Traditional CNN (Bajaj G., 2024)

The convolutional layers are the fundamental units, wherein several filters will slide across the input Picture to produce significant details i.e., edges, textures and designs. The layers employ activation functions, typically ReLU, to induce non-linearity, so that the network can capture more complicated relationships. convolutional layers are followed by pooling layers, typically Max-Pooling, which minimize the spatial dimensions, control overfitting, and resource cost. The feature maps are flattened into one-dimensional vector representation after several convolution and pooling steps and are trained with fully connected layers and do high-level reasoning and classification.It is common to prevent overfitting and add Dropout and Batch Normalization layers. The last layer activates functions such as SoftMax to act on class probabilities. The usage of CNNs in computer vision is prevalent owing to their capability to empower hierarchical representation of raw images, automated learning.

4.5. Architecture of VGG16

In the present project, VGG16 is employed as a transfer learning model to identify deepfake pictures. VGG16 is a highly known convolutional neural network attributable to its simplicity and usefulness in image classification. It has a total layer of 16 of which 13 are convolutional layers and 3 fully connected layers employing small sized 3*3 convolutional filters and 2*2 max-pooling filters. In this configuration, the network can learn to extract hierarchical features, with simple edges and textures in the early layers to more complex shapes and patterns with deeper layers.

CNN and VGG16 are employed to extract features in detecting deepfakes. The weights trained on ImageNet will grant the model the general ability to recognize images, which they then can be fine-tuned on the deepfake dataset. The top layers in the fully connected layers are removed here to insert a custom classifier with a dense layer having ReLU activation layout, a dropout layer to regularize it, and a final SoftMax layer to produce the 2 outputs of real or fake. The method exploits the strength of the features in VGG16 as well as tailors it to the peculiarities of deepfake images. VGG16 uses a combination of transfer learning and a specialized output layer which provides the model with a high accuracy and speeds up convergence at the same time allowing it to be trained even faster than normal training.

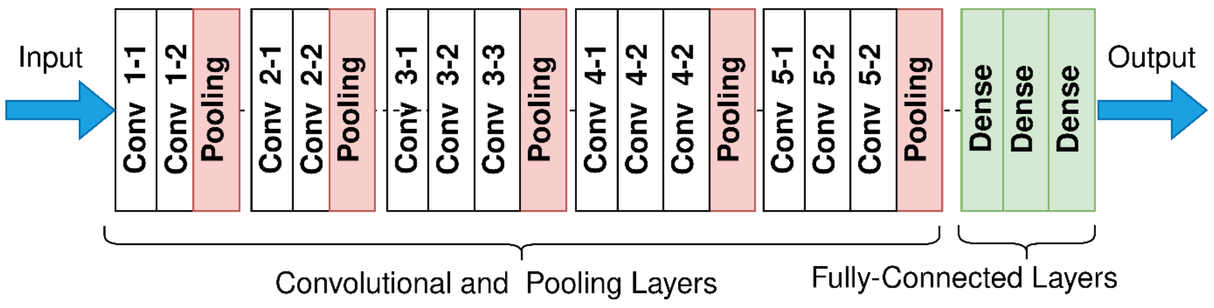


Figure 4 - Architecture of a VGG16 CNN (Zhang et al., 2022)

4.6. Architecture of Xception

Xception-meaning Extreme Inception is the augmentation of the inception architecture where depth wise separable convolutions are implemented and involves compositing standard convolutions into depth wise and pointwise convolutions. This separation enables the model to better model additional features, spatial correlations and channel-wise, in an easy way and it reduces computation and simplifies feature extraction.

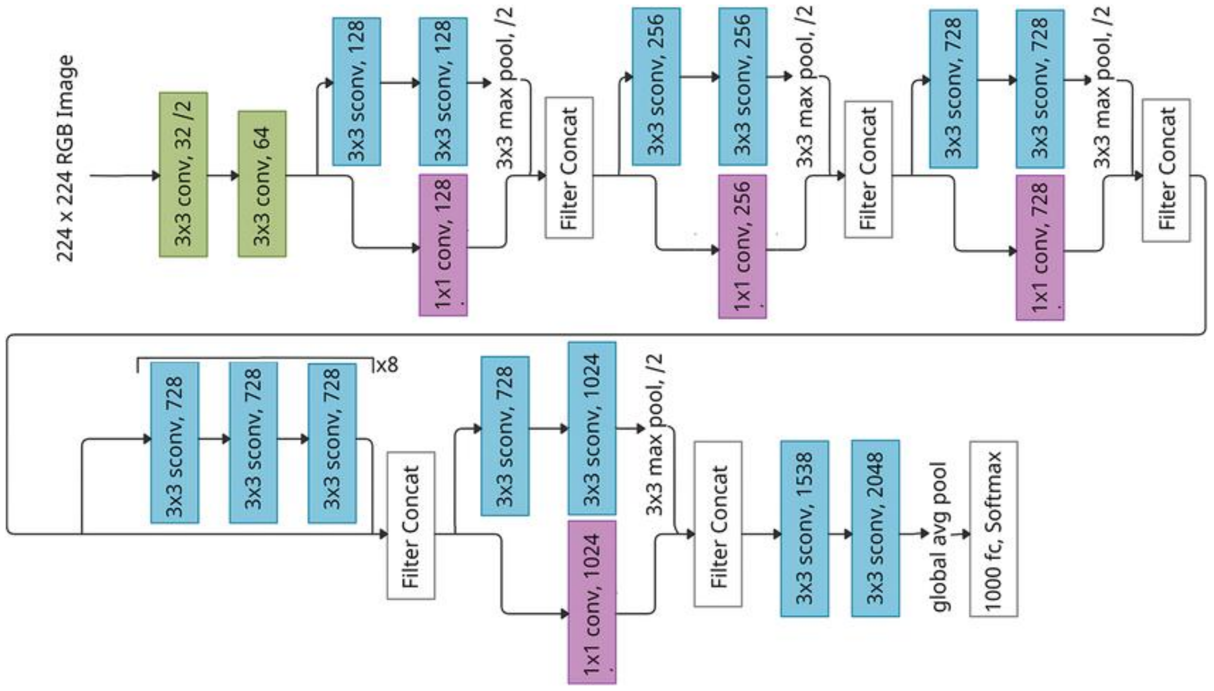


Figure 5 - Architecture of a Xception CNN

The architecture consists of several convolutional layers in entry, middle and exit streams. The low-level features extraction, more complex representations extraction and aggregation of high-level features are realized in the entry flow, the middle flow and the exit flow respectively. The rich feature representations are fundamental in detecting subliminal manipulations in the images that have been deep faked; hence, Xception makes use of pre-trained ImageNet weights to help in this endeavour. The higher layers of the Xception are changed to a bespoke classifier composed of a global average pooling layer, dense layer with ReLU activation, dropout regularization, and a final, SoftMax binary classification layer. This setup enables the network to chopper it learned features to a particular deepfake dataset, increasing the classification robustness and accuracy and without the necessity of retraining the network.

5. Experimental Results

In this study, three different convolutional neural network (CNN) architectures were evaluated for image classification, including a traditional CNN with three convolutional layers, VGG16, and Xception. The models were trained for ten epochs using the same dataset, and their performance was compared based on training accuracy, validation accuracy, and loss metrics.

5.1. Traditional CNN (Three Convolutional Layers) Results

The traditional CNN exhibited a steady increase in training accuracy from 49.1% in the first epoch to 66.3% by the tenth epoch. Validation accuracy started at 69.4% and peaked at 72.2% during epochs six to eight, although it decreased slightly to 63.9% by the last epoch. The loss values followed a similar trend, with training loss reducing from 1.3431 to 0.6407, while validation loss decreased initially but fluctuated slightly toward the end. This behaviour indicates that the network learned general features effectively in early epochs but started to overfit slightly in later epochs, as shown by the divergence between training and validation accuracy.

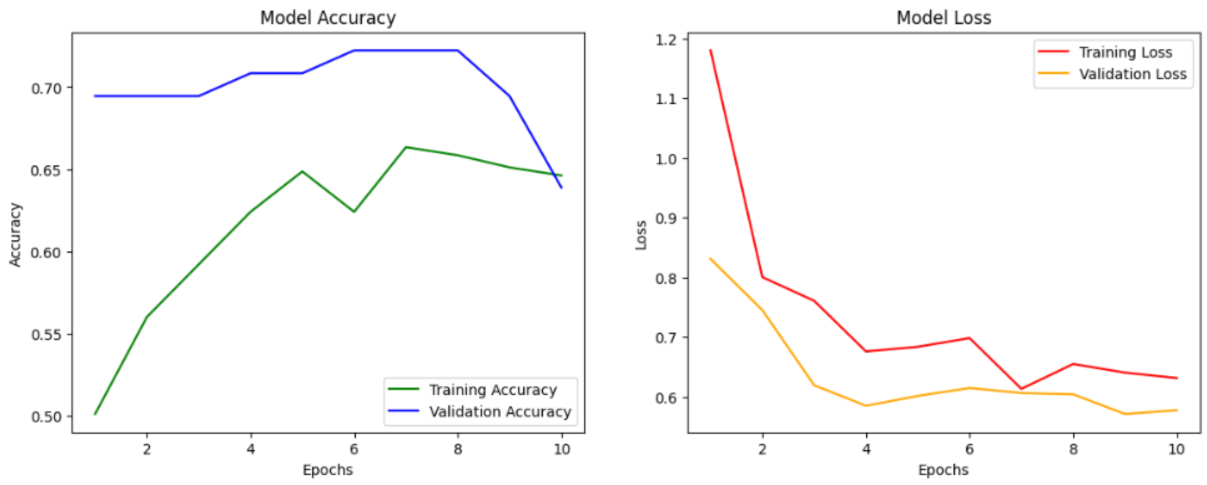


Figure 6 - Accuracy and Loss behaviour of Traditional CNN model

Despite its simplicity, the model achieved reasonable performance, demonstrating that even shallow CNNs can capture meaningful patterns, although they may be limited in representing highly complex features.

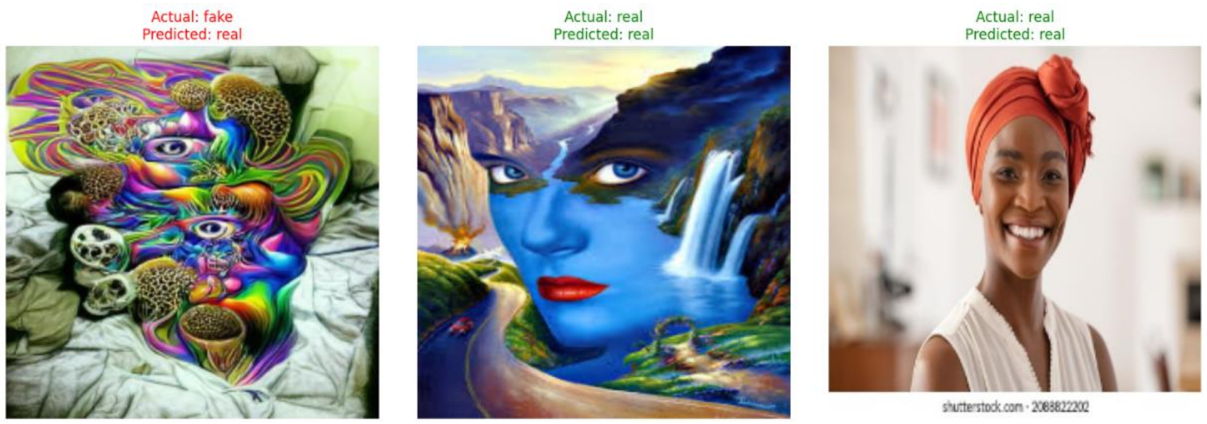


Figure 7 - Traditional CNN predictions made on random images from dataset

5.2. VGG16 Results

VGG16, a deeper network with pre-trained weights, displayed a slower initial improvement in both training and validation accuracy. Training accuracy increased from 44.4% in the first epoch to 58.9% by the tenth epoch, while validation accuracy rose from 40.3% to 73.6%. Training loss decreased steadily from 0.7355 to 0.6749, with validation loss showing a corresponding gradual reduction. The slower initial learning could be attributed to the network's depth, which requires more epochs to fine-tune effectively for a small dataset, even with transfer learning.

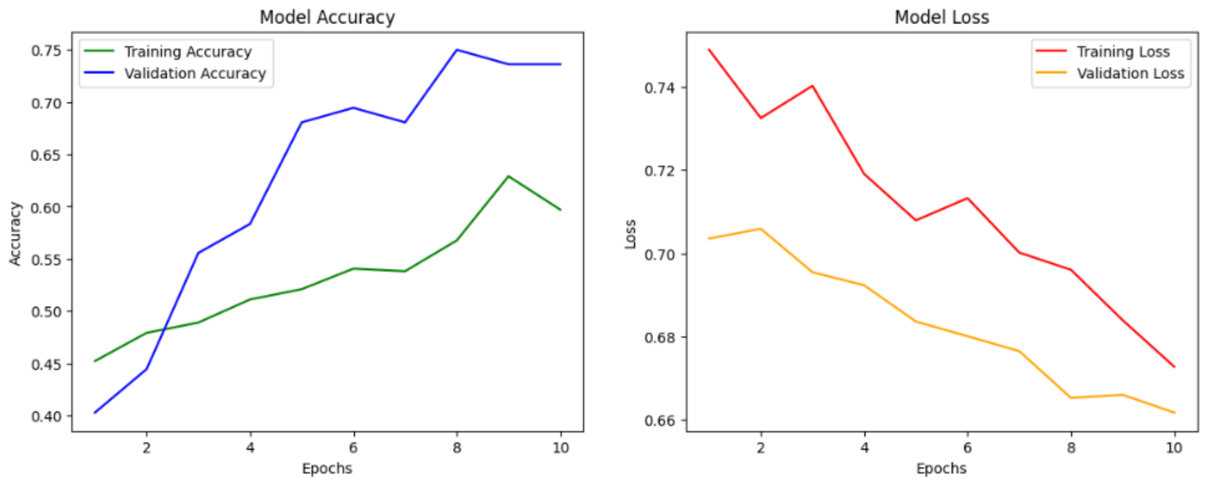


Figure 8 - Accuracy and Loss behaviour of VGG16 model

Notably, the validation accuracy surpassed training accuracy during later epochs, suggesting that pre-trained features provided a strong prior for generalization. However, VGG16 did not outperform the traditional CNN in early epochs, highlighting that deeper network may require careful hyperparameter tuning or larger datasets to achieve optimal performance.

Figure 9 – VGG16 predictions made on random images from dataset

5.3. Xception Results

The Xception model exhibited the most robust performance across all metrics. Training accuracy improved consistently from 37.5% in the first epoch to 79.6% by the tenth epoch, while validation accuracy increased from 68.1% to a peak of 76.3%, stabilizing around 73.6% in the final epoch. Training loss decreased from 0.6955 to 0.5210, and validation loss reduced from 0.6703 to 0.5381. These results indicate that Xception effectively learned both low-level and high-level features, leveraging depth wise separable convolutions to capture spatial and channel-wise correlations efficiently.

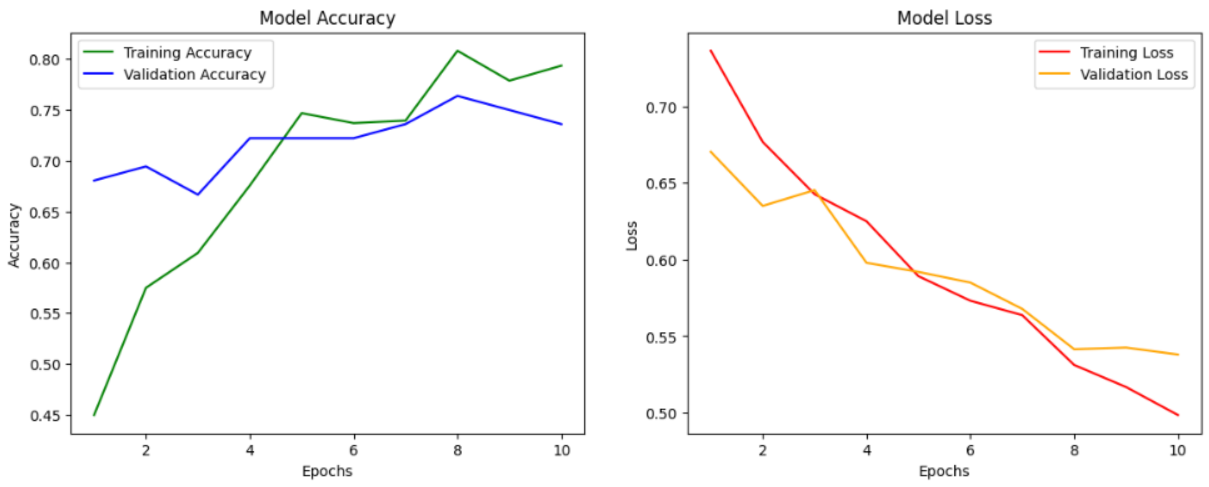


Figure 10 - Accuracy and Loss behaviour of Xception model

Compared to both the traditional CNN and VGG16, Xception achieved the highest training and validation accuracies, highlighting its suitability for detecting subtle differences in images. The network's ability to maintain high validation accuracy without overfitting suggests strong generalization capabilities, which is critical for practical deployment in real-world datasets.

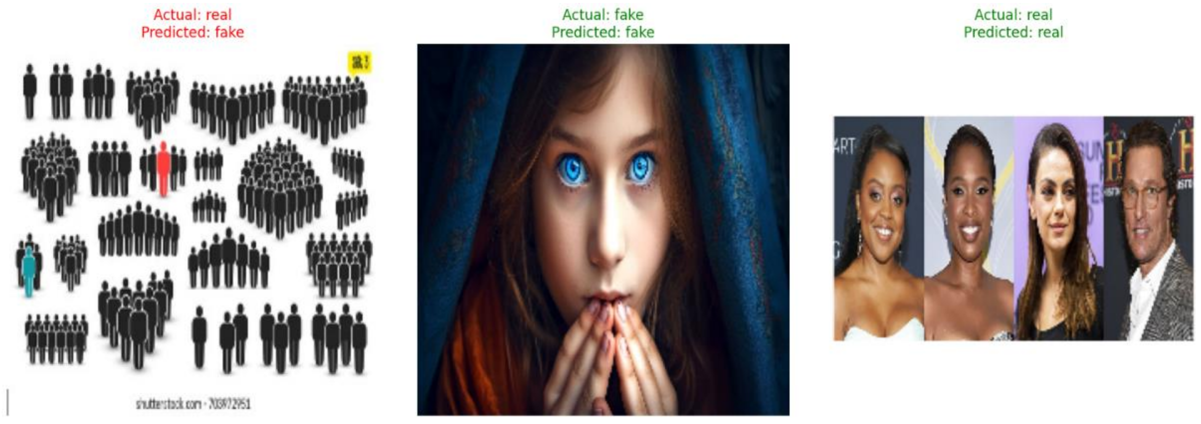
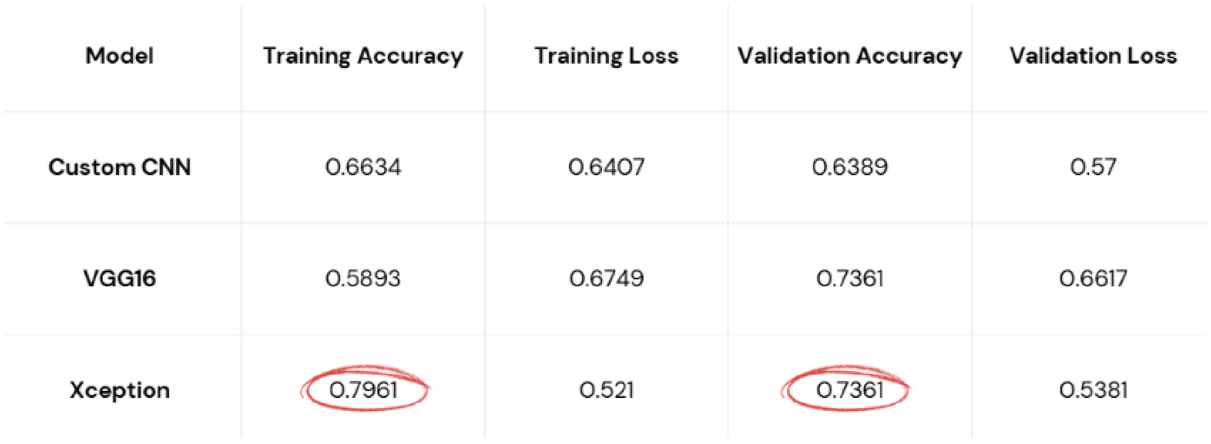


Figure 11 - Xception predictions made on random images from dataset

5.4. Comparative Analysis

Through this comparison of the three models, it is quite evident that Xception model bettered both traditional CNN and VGG16 in the overall accuracy and minimizations of loss values. Although traditional CNN performed well in the initial stages, its depth was limited and could not capture complex features. Although VGG16 utilized pre-trained weights, it has higher number of epochs to fine-tune and this aspect shows slower convergence. The combination of the beneficial features of transfer learning and efficient convolutional operations leads Xception to a higher level of performance and stable learning curves.



Model	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
Custom CNN	0.6634	0.6407	0.6389	0.57
VGG16	0.5893	0.6749	0.7361	0.6617
Xception	0.7961	0.521	0.7361	0.5381

Figure 12 - Results Comparison

These findings indicate that the structure of models has strong influence on the learning capacity and generalization. Simple CNNs may generate adequate results on condensed datasets, although sophisticated designs such as Xception tend to be more precise and reliable and would better suit sections that require nuanced image differences to behave.

6. Discussion

6.1. Limitations

Small Dataset: The dataset size could have been one of the limitations to the performance of the CNN, VGG16, and Xception models. Few data may minimize performance of deep learning models to adequately generalise, leading to overfitting or underrepresentation of classes.

Control Resource Limitations: The deeper models such as VGG16 and Xception took considerable computational resources to train and thus increased the training time and may pose problems. This may have restricted the scale of exploratory work on more in-depth structures or longer-duration training.

Model Interpretability: The deep learning models performed well in accuracy but the decision making is known to be relatively in black-box. Explainable AI methods were not emphasized in the project, and it is therefore hard to understand the predictions that the models made or determine whether the classifications were biased.

6.2. Literature Comparison and Discussion

The literature reviewed points to considerable progress in their detection of the AI-generated and manipulated images with a focus on the variety of methodological approaches, datasets as well as application areas. Bird et al. (2024) and Islam et al. (2025) concentrate on artificial one in which it is possible to show clinical results of CNNs and ensemble learning methodology since they demonstrated a large classification accuracy of real and AI-generated image type. Employment of latent diffusion frameworks and fresh datasets CIFAKE, FakeGPT, and PFake highlight the significance of dataset formulation and benchmarking to provide sound exploration.

Data augmentation and synthetic data emulsification with real images, as studied by Alabed et al. (2025), and Ozturk & Ercelbi (2021), have shown that the ability to combine synthetic data with real-world images can meaningfully increase model generalization and performance. Such techniques as CDNTS layers or streamlined data proportions show the possibility of enhancing performance in domain-specific tasks, such as traffic or UAV-related image classification.

Taspinar & Cinar (2024) and Killi et al. (2023) refer to feature extraction and transfer learning, which demonstrated that machine learning models and transfer learning using pre-trained CNN models such as InceptionV3 and VGG19 may be utilized to identify manipulated content. The benefits of these approaches are less training time and improved feature representation especially on a limited dataset scenario.

7. Conclusion

7.1. Summary of Model Performances

These findings give room to concluding that there are significant differences between various deep learning architectures in terms of their performance and efficiency when deployed to the AI-generated image detection. The very simple structure of conventional CNN showed enough learning potential with a gradual increase in the accuracy of training figures, starting as high as 49.1% and rising to 66.3% and the widest accuracy in the validation reached to 72.2%. The training and the validation accuracy at higher epochs reveal slight overfitting, which exemplifies the inability of shallow networks to pick up complicated features that are intrinsic to AI generated images. Although the model was satisfactory with simple image patterns, it had poor abilities to generalise on fine differences.

VGG16, which used pre-trained weights, initially learned at a slower rate, and the training accuracy improved over ten epochs by 44.4% to 58.9% and that of validation to 73.6%. Its greater validation accuracy compared to its training accuracy proves the usefulness of transfer learning in generalization. Nevertheless, deeper architecture was much more sensitive to tuning and needed more data to perform well and its deeper depth is not synonymous to faster slope or better performance on small datasets.

Xception became the most effective architecture as it scored the best training 79.6% and validation 76.3% accuracy as well as sustaining the most consistent loss drops. The depth wise separable convolutions of the model allowed extracting both low-level and high-level information and, therefore, highlighted the fine-grained disparity between real and AI-generated images. In contrast to the conventional CNN and VGG16, Xception performed well on validation through a substantial validation accuracy and acceptable overfitting levels leading to robust generalization capacities that rendered it quite dependable into practical tasks.

7.2. Hypothesis Testing

Rejection of the null hypothesis (H0) is clearly evident when conducting comparative analysis since there is a notable difference between the chosen architectures in terms of the trade-off between the computation efficiency and appropriate classification accuracy. The simple CNNs are efficient computationally and have a low level of feature extraction, VGG16 are the best of both worlds in terms of pre-trained features and being slow rendered and hence Xception is superior in terms of accuracy combined with generalization at middle-range computation costs. This result highlights model architecture to be very important in the balancing of accuracy, learning speed, and resource requirements in AI generated image detection.

7.3. Future Works

Future research can focus on the combination of lightweight with high-performing structures, e.g., Efficient Net or MobileNetV3 to promote computational cost with reduced accuracy. Besides, the influence of more and diverse data, such as higher-resolution AI-generated pictures, is to be examined to enhance model resilience. Techniques that would make use of combination of ensemble learning with transfer learning would be a great avenue to pursue and find an improvement in the detection performance. Lastly, analysing its performance in live deployments and energy consumption will give useful information as to which architectures to choose that best minimize speed, precision, and energy usage within an AI-generated image detection system of huge proportions.

References

Alabed, A.T. et al. (2025) ‘Bridging reality and synthetics: Optimizing Image Classification with hybrid AI-generated and real-world datasets’, SN Computer Science, 6(6). doi:10.1007/s42979-025-04181-0.

Bird, J.J. and Lotfi, A. (2024) ‘CIFAKE: Image classification and explainable identification of AI-generated synthetic images’, IEEE Access, 12, pp. 15642–15650. doi:10.1109/access.2024.3356122.

Islam, M.T. et al. (2025) ‘MEXFIC: A Meta Ensemble explainable approach for AI-synthesized fake image classification’, Alexandria Engineering Journal, 116, pp. 351–363. doi:10.1016/j.aej.2024.12.031.

Kalaimani, G. et al. (2024) ‘Optimally configured generative adversarial networks to distinguish real and ai-generated human faces’, Signal, Image and Video Processing, 18(11), pp. 7921–7938. doi:10.1007/s11760-024-03440-6.

Khan, S.B. et al. (2025) ‘Deepfake detection: Evaluating the performance of EFFICIENTNETV2-B2 on real vs. fake image classification’, IET Image Processing, 19(1). doi:10.1049/ipr2.70152.

Killi, C.B., Balakrishnan, N. and Rao, C.S. (2023) ‘Deep fake image classification using VGG-19 model’, Ingénierie des systèmes d information, 28(2), pp. 509–515. doi:10.18280/isi.280228.

Oo, S.T. et al. (2025) ‘Classifying real versus AI generated images using CIFAR-10 Network and painting image data’, 2025 IEEE Conference on Computer Applications (ICCA), pp. 1–5. doi:10.1109/icca65395.2025.11011094.

Singh, G., Guleria, K. and Sharma, S. (2024) ‘A fine-tuned MobileNetV3 model for real and fake image classification’, 2024 Second International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI), pp. 1–5. doi:10.1109/icoici62503.2024.10696448.

Taspinar, Y.S. and Cinar, I. (2024) ‘Distinguishing between AI images and real images with hybrid image classification methods’, 2024 13th Mediterranean Conference on Embedded Computing (MECO), pp. 1–4. doi:10.1109/meco62516.2024.10577770.

Öztürk, A.E. and Erçelebi, E. (2021) ‘Real UAV-bird image classification using CNN with a synthetic dataset’, Applied Sciences, 11(9), p. 3863. doi:10.3390/app11093863.

Öztürk, A.E. & Erçelebi, E., 2021. Real UAV-Bird Image Classification Using CNN with a Synthetic Dataset. MDPI Applied Sciences, 11(9), pp. 3863-3875. Available at: https://doi.org/10.3390/app11093863

Taspinar, Y.S. & Cinar, S., 2024. Deep Fake Image Classification Using VGG-19 Model. IIETA. Available at: https://doi.org/10.18280/isi.280228

Killi, C.B.R., et al., 2023. Deep Fake Image Detection Using GAN. International Journal of Advanced Research in Computer Science and Technology, 11(2), pp. 275-280. Available at: https://doi.org/10.3934/mbe.2024071

Baraheem, S.S. and Nguyen, T.V. (2023) ‘Ai vs. AI: Can ai detect AI-generated images?’, Journal of Imaging, 9(10), p. 199. doi:10.3390/jimaging9100199.

Chinta, D.S. et al. (2024) ‘Analyzing image classification on AI-generated art vs human created art using Deep Learning Models’, 2024 Third International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), pp. 1–6. doi:10.1109/iceeict61591.2024.10718485.

Khan, S. and Singh, K.K. (2024) ‘Leveraging AI-generated image for scene classification: A transfer learning approach’, 2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp. 1–6. doi:10.1109/icrito61523.2024.10522260.

Lokner Lađević, A. et al. (2024) ‘Detection of AI-generated synthetic images with a lightweight CNN’, AI, 5(3), pp. 1575–1593. doi:10.3390/ai5030076.

Mistry, J., Koshti, N. and Gautam, G.K. (2025) ‘Efficientnetb0 for AI-generated and Real Image Classification’, Lecture Notes in Networks and Systems, pp. 307–316. doi:10.1007/978-981-96-5751-3_26.

Park, D., Na, H. and Choi, D. (2024) ‘Performance comparison and visualization of AI-generated-image detection methods’, IEEE Access, 12, pp. 62609–62627. doi:10.1109/access.2024.3394250.

Appendices

```
base_dir = Path('../input/deepfake-image-detection/train-20250112T065955Z-001/train')

classes = ['real', 'fake']


filepaths = []

labels = []


for label in classes:

    class_dir = base_dir / label

    image_files = class_dir.glob('*')


    for file in image_files:

        if file.is_file():

            filepaths.append(str(file))

            labels.append(label)


Files=pd.Series(filepaths, name='filepaths')

Label=pd.Series(labels, name='labels')


df=pd.concat([Files,Label], axis=1)
```

```
df.sample(10)


def sample_images(label, df, n=5):
    """
    Displays n images in a single row from the specified label.


    Parameters:
    - label (str): 'real' or 'fake'
    - df (pd.DataFrame): DataFrame with 'filepaths' and 'labels'
    - n (int): Number of images to display (default is 5)
    """
    subset = df[df['labels'] == label]


    if subset.empty:
        print(f"No images found with label: {label}")
        return


    sample_files = subset.sample(min(n, len(subset)))['filepaths'].tolist()


    plt.figure(figsize=(15, 5))
    for i, file_path in enumerate(sample_files):
        try:
            img = Image.open(file_path)
            plt.subplot(1, n, i + 1)

            plt.imshow(img)

            plt.axis('off')

            plt.title(f"{label.capitalize()}")

        except Exception as e:

            print(f"Error loading image {file_path}: {e}")

    plt.tight_layout()

    plt.show()


sample_images('real', df)


sample_images('fake', df)


def label_distribution(df):

    """

    Plots a bar chart and a pie chart showing the distribution of labels.

    Bar chart shows counts with text on top of bars.

    Pie chart shows percentages.

    """

    label_counts = df['labels'].value_counts()

    labels = label_counts.index

    counts = label_counts.values

    percentages = label_counts / label_counts.sum() * 100
```

```
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# Bar plot
axes[0].bar(labels, counts, color=['skyblue', 'salmon'])
axes[0].set_title('Label Counts')
axes[0].set_ylabel('Count')

for i, count in enumerate(counts):
    axes[0].text(i, count + 1, str(count), ha='center', va='bottom', fontsize=10)

axes[1].pie(
    counts,
    labels=labels,
    autopct='%1.1f%%',
    startangle=90,
    colors=['skyblue', 'salmon']
)
axes[1].set_title('Label Distribution (%)')

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

```
label_distribution(df)


image_size = 224
images = []
processed_labels = []


for path, label in zip(df['filepaths'], df['labels']):
    img = cv2.imread(path)  # Load as BGR
    if img is None:
        print(f"Warning: Could not read image {path}")
        continue


    img = cv2.resize(img, (image_size, image_size))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    images.append(img)
    processed_labels.append(label)


images[:1]


processed_labels[:5]


x = np.array(images, dtype='float32') / 255.0
```

```
x.shape

le = LabelEncoder()
y_encoded = le.fit_transform(processed_labels)
y = to_categorical(y_encoded, num_classes=2)

y.shape

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.15, shuffle = True,
random_state=42)

x_train.shape

y_train.shape

class_weights = class_weight.compute_class_weight(class_weight='balanced',
                                                  classes=np.unique(y_encoded),
                                                  y=y_encoded)
class_weights_dict = dict(enumerate(class_weights))

epochs = 10
input_shape = (225, 225, 3)

model = Sequential()
```

```
model.add(Conv2D(32, (3, 3), padding='same', activation='relu', input_shape=input_shape))


model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))


model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))


model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))


model.add(Dense(2, activation='softmax'))


model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy',
          metrics=['accuracy'])


model.summary()


from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
datagen = ImageDataGenerator(rotation_range=10, zoom_range=0.1, width_shift_range=0.1,
                             height_shift_range=0.1, horizontal_flip=True)

history = model.fit(datagen.flow(x_train, y_train, batch_size=32), epochs=epochs, validation_data=(x_test, y_test),
                    class_weight=class_weights_dict)

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(1, len(acc) + 1)

plt.figure(figsize=(14, 5))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy', color='green')
plt.plot(epochs_range, val_acc, label='Validation Accuracy', color='blue')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()


# Loss plot

plt.subplot(1, 2, 2)

plt.plot(epochs_range, loss, label='Training Loss', color='red')

plt.plot(epochs_range, val_loss, label='Validation Loss', color='orange')

plt.title('Model Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()


plt.show()


label_map = {i: label for i, label in enumerate(le.classes_)}


random_indices = random.sample(range(x_test.shape[0]), 3)

sample_images = x_test[random_indices]

sample_labels = y_test[random_indices]


predictions = model.predict(sample_images)

predicted_classes = np.argmax(predictions, axis=1)

true_classes = np.argmax(sample_labels, axis=1)
```

```
plt.figure(figsize=(15, 5))

for i in range(3):

    plt.subplot(1, 3, i+1)

    plt.imshow(sample_images[i])

    plt.axis('off')

    plt.title(

        f"Actual: {label_map[true_classes[i]]}\nPredicted: {label_map[predicted_classes[i]]}",

        color='green' if predicted_classes[i] == true_classes[i] else 'red')


plt.tight_layout()

plt.show()


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


base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

base_model.trainable = False


model = Sequential([base_model, GlobalAveragePooling2D(),

                    Dense(256, activation='relu'), Dropout(0.5),

                    Dense(2, activation='softmax')])


model.compile(optimizer=Adam(learning_rate=1e-4),                 loss='binary_crossentropy',
metrics=['accuracy'])
```

```
model.summary()

history = model.fit(datagen.flow(x_train, y_train, batch_size=32), epochs=epochs,
validation_data=(x_test, y_test),
                    class_weight=class_weights_dict)

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(1, len(acc) + 1)

plt.figure(figsize=(14, 5))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy', color='green')
plt.plot(epochs_range, val_acc, label='Validation Accuracy', color='blue')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Loss plot
```

```
plt.subplot(1, 2, 2)

plt.plot(epochs_range, loss, label='Training Loss', color='red')

plt.plot(epochs_range, val_loss, label='Validation Loss', color='orange')

plt.title('Model Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()


plt.show()


label_map = {i: label for i, label in enumerate(le.classes_)}


random_indices = random.sample(range(x_test.shape[0]), 3)

sample_images = x_test[random_indices]

sample_labels = y_test[random_indices]


predictions = model.predict(sample_images)

predicted_classes = np.argmax(predictions, axis=1)

true_classes = np.argmax(sample_labels, axis=1)


plt.figure(figsize=(15, 5))

for i in range(3):

    plt.subplot(1, 3, i+1)
    plt.imshow(sample_images[i])

    plt.axis('off')

    plt.title(

        f"Actual: {label_map[true_classes[i]]}\nPredicted: {label_map[predicted_classes[i]]}",

        color='green' if predicted_classes[i] == true_classes[i] else 'red')


plt.tight_layout()

plt.show()


from tensorflow.keras.applications import Xception

from tensorflow.keras.layers import GlobalAveragePooling2D


base_model = Xception(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

base_model.trainable = False


model = Sequential([base_model, GlobalAveragePooling2D(),

                    Dense(256, activation='relu'), Dropout(0.5),

                    Dense(2, activation='softmax')])


model.compile(optimizer=Adam(learning_rate=1e-4),                    loss='binary_crossentropy',
metrics=['accuracy'])


model.summary()
```

```
history = model.fit(datagen.flow(x_train, y_train, batch_size=32), epochs=epochs,
validation_data=(x_test, y_test),
                    class_weight=class_weights_dict)

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(1, len(acc) + 1)

plt.figure(figsize=(14, 5))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy', color='green')
plt.plot(epochs_range, val_acc, label='Validation Accuracy', color='blue')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Loss plot
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss', color='red')
plt.plot(epochs_range, val_loss, label='Validation Loss', color='orange')

plt.title('Model Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()


plt.show()


label_map = {i: label for i, label in enumerate(le.classes_)}


random_indices = random.sample(range(x_test.shape[0]), 3)

sample_images = x_test[random_indices]

sample_labels = y_test[random_indices]


predictions = model.predict(sample_images)

predicted_classes = np.argmax(predictions, axis=1)

true_classes = np.argmax(sample_labels, axis=1)


plt.figure(figsize=(15, 5))

for i in range(3):

    plt.subplot(1, 3, i+1)

    plt.imshow(sample_images[i])

    plt.axis('off')
    plt.title(
        f"Actual: {label_map[true_classes[i]]}\nPredicted: {label_map[predicted_classes[i]]}",
        color='green' if predicted_classes[i] == true_classes[i] else 'red')

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