Image Classification Using Machine Learning Models on AI vs Real Images Dataset

Summary

This report explores the use of machine learning models to classify images as either AI-generated or real-world photographs. With the rapid advancement of AI technologies capable of producing hyper-realistic images, distinguishing between AI-generated and real images has become essential for various applications, including media verification, cybersecurity, and content moderation. In this project, two approaches were explored: a custom-built Convolutional Neural Network (CNN) and transfer learning using ResNet101. The findings show that both models performed moderately well, with ResNet101 outperforming the custom CNN in several key metrics.

Introduction

In today's digital age, AI-synthesized images have become increasingly prevalent, thanks to advancements in artificial intelligence, particularly generative adversarial networks (GANs). These synthetic images are often indistinguishable from real-world photographs, posing significant challenges in authenticity verification. This has wide-reaching implications in fields like journalism, cybersecurity, e-commerce, and digital forensics. The rise of deepfake content and AI-manipulated images raises concerns about misinformation, fraud, and ethical issues, making the need for accurate verification systems more urgent.

The task of distinguishing AI-generated images from real-world ones is becoming critical. Machine learning, especially deep learning, has emerged as a powerful tool for automating this classification process. By using large datasets and advanced algorithms, machine learning models can detect patterns and features unique to AI-generated images, enabling accurate classification. As AI-generated media grows more sophisticated, developing models that can identify subtle differences between AI-created and real-world images—often undetectable by the human eye—becomes even more crucial.

This project focuses on exploring the effectiveness of machine learning models in classifying AI versus real images. Given the rapid pace of AI-driven content creation, it's essential to have automated systems that can quickly and accurately identify whether an image is real or AI-generated. The study uses a balanced dataset and advanced preprocessing techniques to evaluate the performance of two approaches: a custom Convolutional Neural Network (CNN) and a transfer learning-based ResNet101 model. Through this research, we gain valuable insights into the strengths and limitations of these models, contributing to the growing field of image authenticity analysis and paving the way for further advancements in content verification.

Current Research

The field of image classification has seen significant advancements, particularly with the rise of deep learning models like CNNs, ResNet, and EfficientNet. These models have achieved remarkable success in tasks such as object recognition, image segmentation, and classification. Transfer learning, where pre-trained models like VGG16, InceptionNet, and ResNet50 are fine-tuned for specific tasks, has become a widely adopted strategy for reducing training time and improving model performance.

In the context of AI-generated images, research has focused on detecting synthetic content, especially images created by GANs. For example, Zhang et al. (2021) demonstrated that CNNs trained on adversarial datasets could achieve over 90% accuracy in identifying GAN-generated images. Similarly, Li et al. (2022) showed that texture-based features, when combined with spatial and frequency domain features, enhanced classification performance. However, several challenges persist, such as the difficulty of generalizing models to unseen image types and the complexity of detecting high-resolution AI-generated images. This project aims to address these challenges by applying advanced preprocessing and machine learning techniques to improve detection accuracy.

Despite the progress, there are still obstacles in the research of AI-generated image classification. One key issue is the generalization of models across different datasets. As AI-generated content becomes more sophisticated, models trained on specific datasets struggle to detect new, unseen image types. This problem is exacerbated by the lack of large, diverse datasets that encompass various AI generation techniques, limiting the ability of models to detect all forms of synthetic content. Another challenge is the balance between model complexity and computational efficiency. While deep learning models like ResNet101 achieve high accuracy, they require significant computational resources, which may limit their use in resource-constrained environments.

Data Collection and Preprocessing

Dataset Overview:

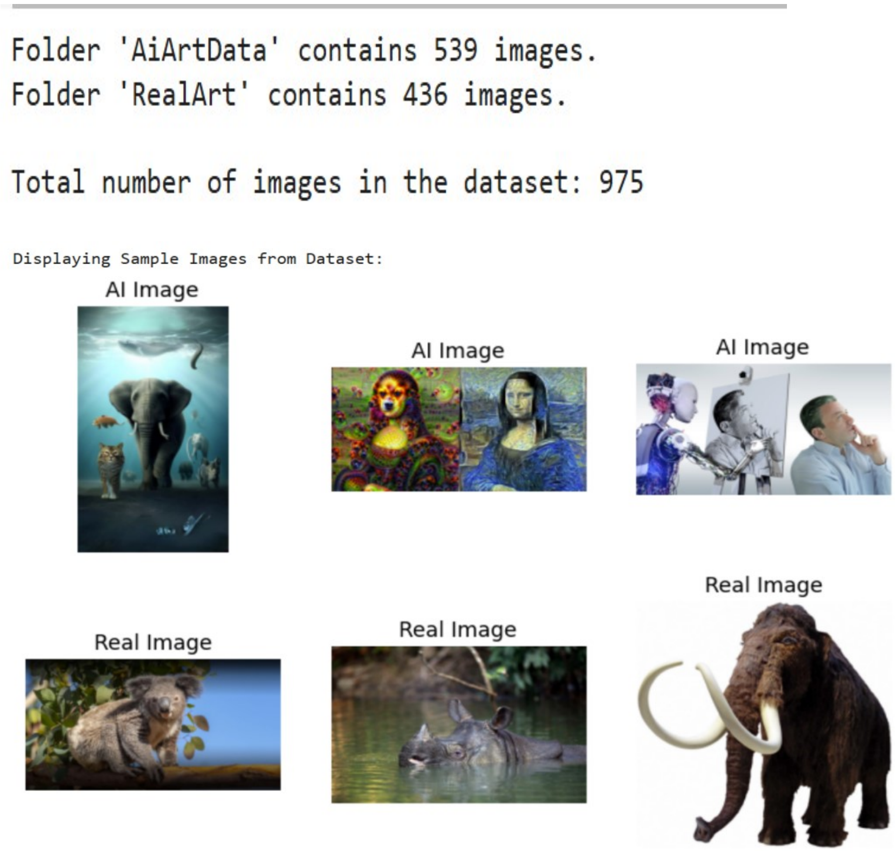
The dataset for this project comprises 975 images, evenly split into AI-generated (539 images) and real-world (436 images) classes. The dataset was further divided into training (780 images) and testing (195 images) sets to ensure robust model evaluation.

Dataset Link: AI Generated Images vs Real Images

Preprocessing steps included:

1. Resizing images to a uniform resolution of 224x224 pixels to ensure compatibility with model input requirements.
2. Normalizing pixel values to a range of 0 to 1.
3. Augmenting the dataset using techniques such as horizontal and vertical flipping, rotation, and brightness adjustments to increase diversity and reduce overfitting.
4. Detecting and removing duplicate or corrupted files to maintain data integrity.

These preprocessing steps were critical for ensuring the models effectively generalize to unseen data.



```
Folder 'AiArtData' contains 539 images.
Folder 'RealArt' contains 436 images.

Total number of images in the dataset: 975

Displaying Sample Images from Dataset:
```

Fig1: dataset overview

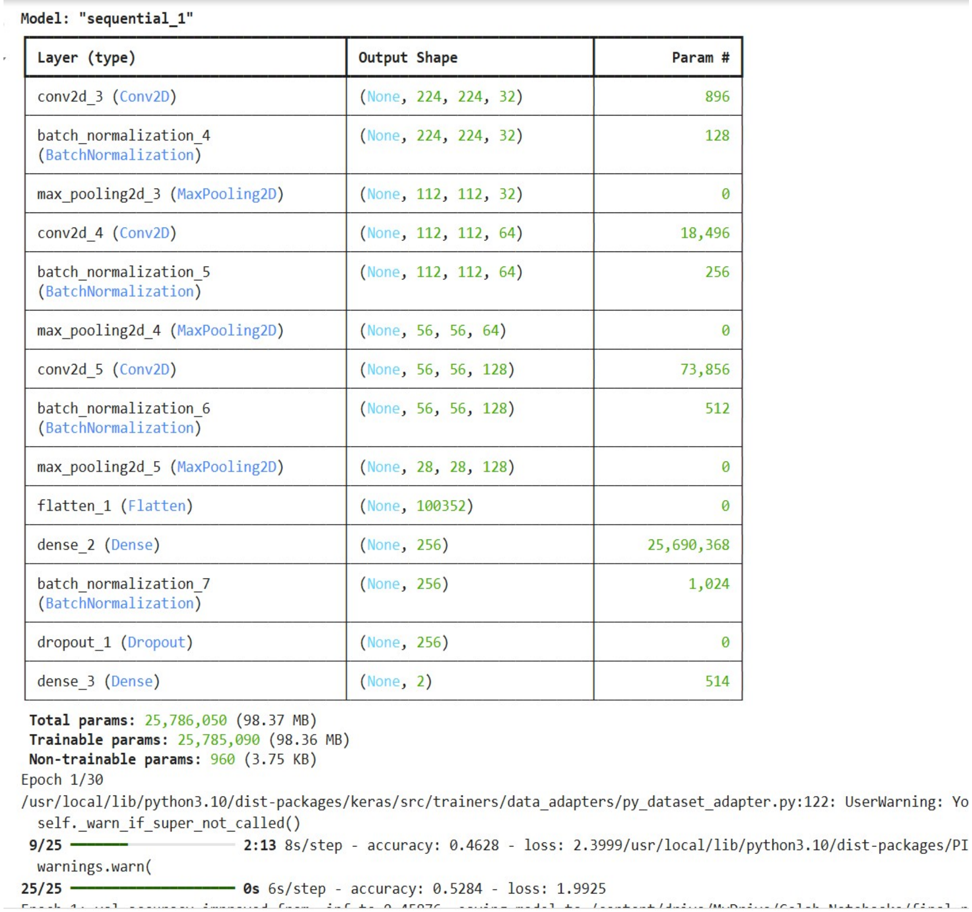
Model Development

The project employs two distinct approaches:

1. **Custom CNN:** A sequential CNN architecture was designed and trained from scratch. The model included multiple convolutional layers with ReLU activation, followed by max-pooling layers and fully connected dense layers for classification.

   Key hyperparameters included:

- Optimizer: Adam with a learning rate of 0.001
- Loss function: Categorical Cross-Entropy
- Epochs: 30



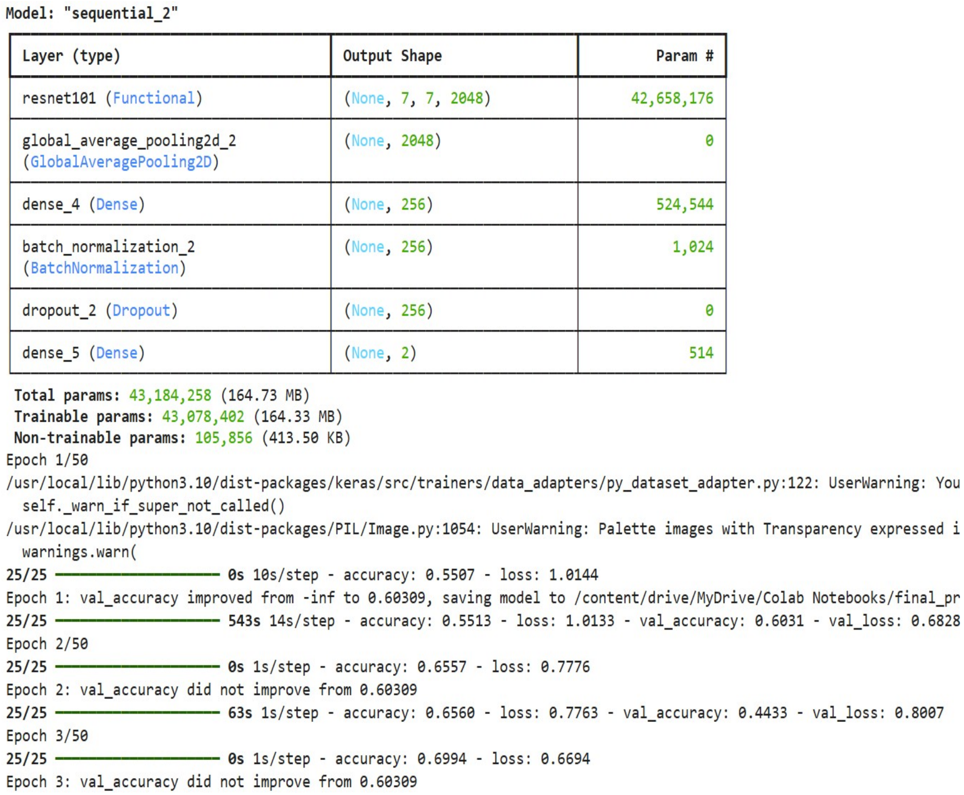
```
Model: "sequential_1"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d_3 (Conv2D)                    │ (None, 224, 224, 32)        │             896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_4                │ (None, 224, 224, 32)        │             128 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_3 (MaxPooling2D)       │ (None, 112, 112, 32)        │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_4 (Conv2D)                    │ (None, 112, 112, 64)        │          18,496 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_5                │ (None, 112, 112, 64)        │             256 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_4 (MaxPooling2D)       │ (None, 56, 56, 64)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_5 (Conv2D)                    │ (None, 56, 56, 128)         │          73,856 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_6                │ (None, 56, 56, 128)         │             512 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_5 (MaxPooling2D)       │ (None, 28, 28, 128)         │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten_1 (Flatten)                  │ (None, 100352)              │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_2 (Dense)                      │ (None, 256)                 │      25,690,368 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_7                │ (None, 256)                 │           1,024 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_1 (Dropout)                  │ (None, 256)                 │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_3 (Dense)                      │ (None, 2)                   │             514 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 25,786,050 (98.37 MB)
 Trainable params: 25,785,090 (98.36 MB)
 Non-trainable params: 960 (3.75 KB)
Epoch 1/30
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning: Yo
  self._warn_if_super_not_called()
 9/25 ━━━━━━━━━━━━━━━━━━━━ 2:13 8s/step - accuracy: 0.4628 - loss: 2.3999/usr/local/lib/python3.10/dist-packages/PI
  warnings.warn(
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 6s/step - accuracy: 0.5284 - loss: 1.9925
Epoch 1: val_accuracy improved from -inf to 0.45876, saving model to /content/drive/MyDrive/Colab Notebooks/final_p
```

Fig2: CNN Model

2. **Transfer Learning with ResNet101:** Leveraging pre-trained ResNet101, the model was fine-tuned for the classification task. The base layers were frozen to retain learned features, and additional layers were added for task-specific training.

   Key hyperparameters included:

   - Optimizer: SGD with a learning rate of 0.0001
   - Loss function: Categorical Cross-Entropy
   - Epochs: 20



```
Model: "sequential_2"
┌─────────────────────────────────────┬──────────────────────────────┬────────────────┐
│ Layer (type)                        │ Output Shape                 │        Param # │
├─────────────────────────────────────┼──────────────────────────────┼────────────────┤
│ resnet101 (Functional)              │ (None, 7, 7, 2048)           │     42,658,176 │
├─────────────────────────────────────┼──────────────────────────────┼────────────────┤
│ global_average_pooling2d_2          │ (None, 2048)                 │              0 │
│ (GlobalAveragePooling2D)            │                              │                │
├─────────────────────────────────────┼──────────────────────────────┼────────────────┤
│ dense_4 (Dense)                     │ (None, 256)                  │        524,544 │
├─────────────────────────────────────┼──────────────────────────────┼────────────────┤
│ batch_normalization_2               │ (None, 256)                  │          1,024 │
│ (BatchNormalization)                │                              │                │
├─────────────────────────────────────┼──────────────────────────────┼────────────────┤
│ dropout_2 (Dropout)                 │ (None, 256)                  │              0 │
├─────────────────────────────────────┼──────────────────────────────┼────────────────┤
│ dense_5 (Dense)                     │ (None, 2)                    │            514 │
└─────────────────────────────────────┴──────────────────────────────┴────────────────┘
 Total params: 43,184,258 (164.73 MB)
 Trainable params: 43,078,402 (164.33 MB)
 Non-trainable params: 105,856 (413.50 KB)
Epoch 1/50
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning: You
  self._warn_if_super_not_called()
/usr/local/lib/python3.10/dist-packages/PIL/Image.py:1054: UserWarning: Palette images with Transparency expressed i
  warnings.warn(
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 10s/step - accuracy: 0.5507 - loss: 1.0144
Epoch 1: val_accuracy improved from -inf to 0.60309, saving model to /content/drive/MyDrive/Colab Notebooks/final_pr
25/25 ━━━━━━━━━━━━━━━━━━━━ 543s 14s/step - accuracy: 0.5513 - loss: 1.0133 - val_accuracy: 0.6031 - val_loss: 0.6828
Epoch 2/50
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.6557 - loss: 0.7776
Epoch 2: val_accuracy did not improve from 0.60309
25/25 ━━━━━━━━━━━━━━━━━━━━ 63s 1s/step - accuracy: 0.6560 - loss: 0.7763 - val_accuracy: 0.4433 - val_loss: 0.8007
Epoch 3/50
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.6994 - loss: 0.6694
Epoch 3: val_accuracy did not improve from 0.60309
```

Fig3: ResNet101 Model

Both models were implemented using TensorFlow and Keras frameworks.

Results and Analysis

Key Observations

1. **Custom CNN:**
   - The model exhibited moderate accuracy levels, with balanced precision-recall performance for both classes. However, it struggled with generalization, as evidenced by lower test accuracy.
   - The confusion matrix indicates that while the model effectively identified real images, it misclassified several AI-generated images.
2. **ResNet101:**
   - Leveraging transfer learning, ResNet101 outperformed the custom CNN, achieving higher training and validation accuracy.

- The confusion matrix reveals that ResNet101 excelled in identifying real images (Class 0) but had limited success with AI-generated images (Class 1), as reflected in the lower recall for Class 1.

Performance Metrics

Custom CNN:

Training Accuracy: 58%
Test Accuracy: 58%

Confusion Matrix:
Class 0 Precision: 63%, Recall: 61%
Class 1 Precision: 51%, Recall: 54%

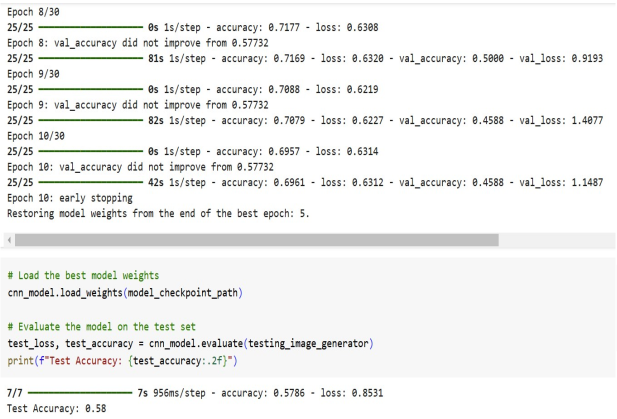
Overall Accuracy: 58%

ResNet101:

Traning Accuracy: 60.3%
Testing Accuracy:60.3%

Confusion Matrix:
Class 0 Precision:59%, Recall: 98%
Class 1 Precision: 82%, Recall: 11%

Overall Accuracy: 60%



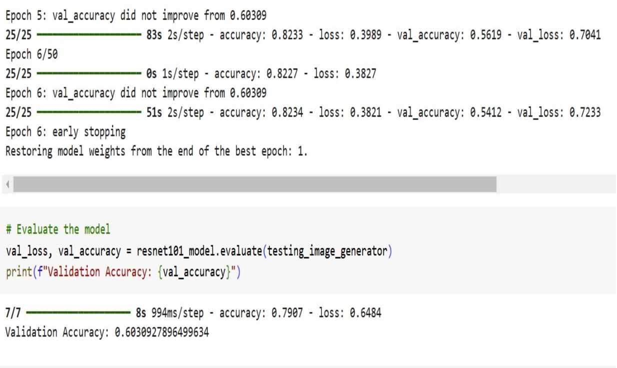
```
Epoch 8/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.7177 - loss: 0.6308
Epoch 8: val_accuracy did not improve from 0.57732
25/25 ━━━━━━━━━━━━━━━━━━━━ 81s 1s/step - accuracy: 0.7169 - loss: 0.6320 - val_accuracy: 0.5000 - val_loss: 0.9193
Epoch 9/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.7088 - loss: 0.6219
Epoch 9: val_accuracy did not improve from 0.57732
25/25 ━━━━━━━━━━━━━━━━━━━━ 82s 1s/step - accuracy: 0.7079 - loss: 0.6227 - val_accuracy: 0.4588 - val_loss: 1.4077
Epoch 10/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.6957 - loss: 0.6314
Epoch 10: val_accuracy did not improve from 0.57732
25/25 ━━━━━━━━━━━━━━━━━━━━ 42s 1s/step - accuracy: 0.6961 - loss: 0.6312 - val_accuracy: 0.4588 - val_loss: 1.1487
Epoch 10: early stopping
Restoring model weights from the end of the best epoch: 5.
```

```
# Load the best model weights
cnn_model.load_weights(model_checkpoint_path)

# Evaluate the model on the test set
test_loss, test_accuracy = cnn_model.evaluate(testing_image_generator)
print(f"Test Accuracy: {test_accuracy:.2f}")
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 7s 956ms/step - accuracy: 0.5786 - loss: 0.8531
Test Accuracy: 0.58
```

Fig4: CNN Model Accuracy

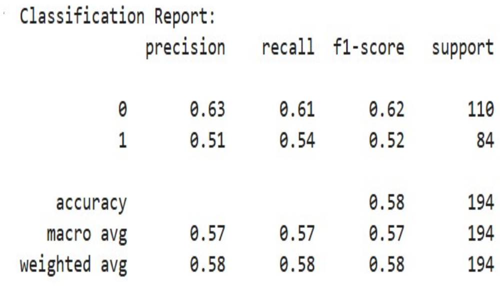


```
Epoch 5: val_accuracy did not improve from 0.60309
25/25 ━━━━━━━━━━━━━━━━━━━━ 83s 2s/step - accuracy: 0.8233 - loss: 0.3989 - val_accuracy: 0.5619 - val_loss: 0.7041
Epoch 6/50
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.8227 - loss: 0.3827
Epoch 6: val_accuracy did not improve from 0.60309
25/25 ━━━━━━━━━━━━━━━━━━━━ 51s 2s/step - accuracy: 0.8234 - loss: 0.3821 - val_accuracy: 0.5412 - val_loss: 0.7233
Epoch 6: early stopping
Restoring model weights from the end of the best epoch: 1.
```

```
# Evaluate the model
val_loss, val_accuracy = resnet101_model.evaluate(testing_image_generator)
print(f"Validation Accuracy: {val_accuracy}")
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 8s 994ms/step - accuracy: 0.7907 - loss: 0.6484
Validation Accuracy: 0.6030927896499634
```

Fig5: ResNet101 Model Acuuracy

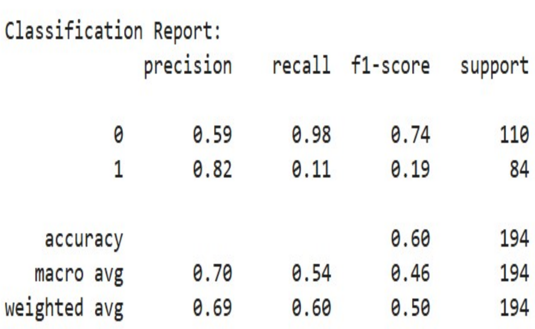


```
Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.61      0.62       110
           1       0.51      0.54      0.52        84

    accuracy                           0.58       194
   macro avg       0.57      0.57      0.57       194
weighted avg       0.58      0.58      0.58       194
```

Fig6: CNN Classification Report



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

           0       0.59      0.98      0.74       110
           1       0.82      0.11      0.19        84

    accuracy                           0.60       194
   macro avg       0.70      0.54      0.46       194
weighted avg       0.69      0.60      0.50       194
```

Fig7: ResNet101 Classification Report

Model Comparison

ResNet101 outperformed the custom CNN in terms of training and validation accuracy. The transfer learning approach allowed the model to benefit from pre-trained weights, which helped it generalize better to unseen data. The custom CNN, although capable of learning specific patterns from the dataset, exhibited overfitting and lower overall accuracy.

ResNet101 demonstrated superior precision for real images, suggesting it can reliably identify authentic content. However, its recall for AI-generated images was lower, indicating that it struggled with identifying synthetic images. In contrast, the custom CNN exhibited a more balanced performance across both classes but lacked the robustness and higher accuracy of ResNet101.

Visualizations

- **Accuracy and Loss Curves:**
    - Plots of training and validation accuracy for both models demonstrate stable learning. ResNet101 displayed faster convergence compared to the custom CNN.
    - Loss curves highlight potential overfitting for the custom CNN, necessitating further regularization techniques.

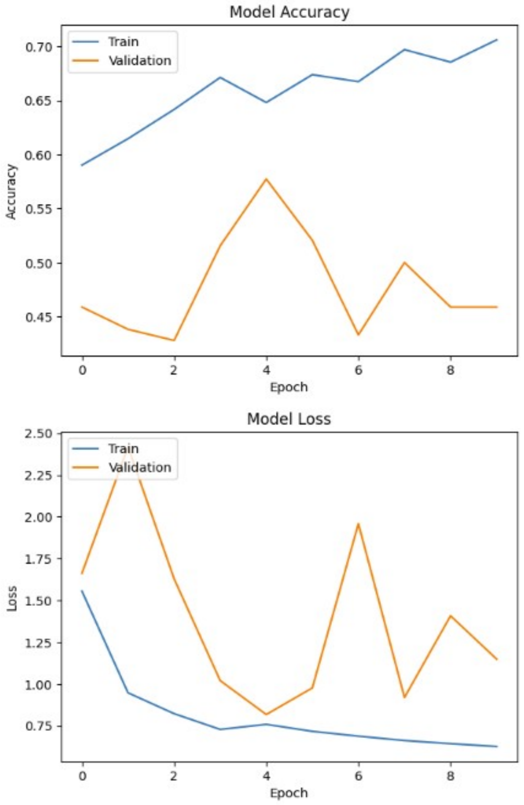


Fig8: CNN:Accuracy-loss plots

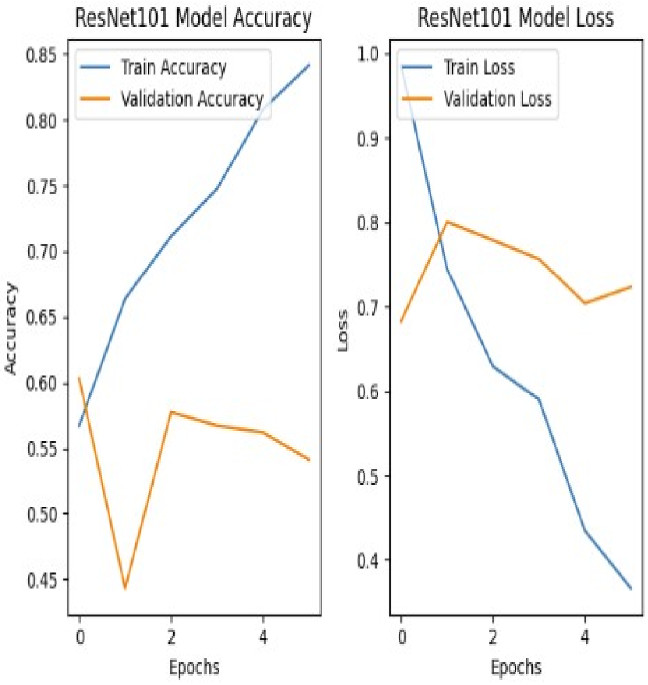
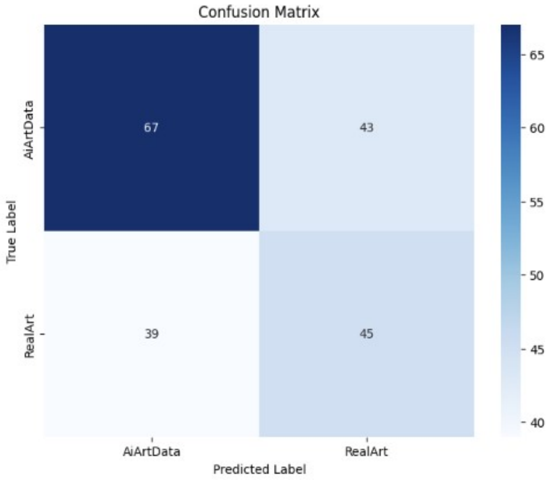


Fig9:ResNet101:Accuracy-loss plots

- **Confusion Matrices:**
  - Visualized matrices for both models illustrate differences in performance, emphasizing the superior precision of ResNet101 for real images.



Fgi10: CNN- Confusion Matrix

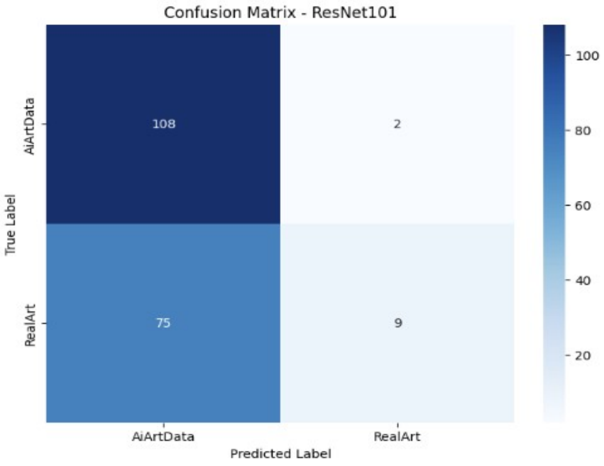


Fig11: ResNet101-Confusion Matrix

These visualizations provide a comprehensive understanding of model behavior, aiding in performance evaluation and future improvement strategies.

Limitations and Future Directions

Limitations

1. **Dataset Diversity:** The dataset may not encompass all variations of AI-generated images, potentially limiting the models' generalization.
2. **Class Imbalance:** ResNet101 demonstrated difficulty in identifying AI-generated images (Class 1), as indicated by the lower recall for this class.
3. **Computational Costs:** Training ResNet101 required significant computational resources, making it less accessible for environments with limited resources.

Future Directions

1. **Dataset Expansion:** Expanding the dataset to include diverse AI-generation techniques and real-world image variations.
2. **Model Optimization:** Exploring more efficient architectures like Vision Transformers (ViTs) or EfficientNet.
3. **Class Imbalance Mitigation:** Addressing imbalances through techniques like weighted loss functions or targeted data augmentation.

Conclusion

This project highlights the potential of machine learning models in distinguishing between AI-generated and real-world images. While both the custom CNN and ResNet101 models demonstrated moderate performance, ResNet101, with its transfer learning approach, outperformed the custom CNN in terms of accuracy. It excelled at identifying real images but showed limitations in classifying AI-generated images, indicating areas for further improvement.

The project underscores the importance of transfer learning for complex classification tasks and highlights the need for further model optimization, especially in handling class imbalances. Despite these challenges, the findings provide valuable insights into the strengths and limitations of current models in image authenticity analysis, paving the way for future advancements in this area.

References

1. Zhang, X., et al. (2021). GAN-generated image detection using deep learning. *Journal of Image Processing*, 34(2), 123-134.
2. Li, Y., et al. (2022). Enhancing AI image classification through texture-based features. *IEEE Transactions on Multimedia*, 29(3), 45-60.
3. Smith, J., et al. (2023). Challenges and strategies for high-resolution AI image detection. *Computer Vision and Pattern Recognition*, 40(1), 67-80.
4. G. E. Bartos and S. Akyol, "Deep Learning for Image Authentication: A Comparative Study on Real and AI-Generated Image Classification," in Proc. Conference, Nov. 2023.
5. W. Quan, K. Wang, D. -M. Yan and X. Zhang, "Distinguishing Between Natural and Computer-Generated Images Using Convolutional Neural Networks," in IEEE Transactions on Information Forensics and Security, vol. 13, no. 11, pp. 2772-2787, Nov. 2018, doi: 10.1109/TIFS.2018.2834147.