AI Generated vs Real Images Classification

Sidiropoulos Michals 2320

MSc Artificial Intelligence

Course: Deep Learning

NCSR Demokritos

Supervising professor: Giorgos Bouritsas

Contents

Introduction

In recent years, the rise of AI-generated content has brought about significant advancements in various fields, including art, entertainment, and media. Among these, AI-generated images have garnered considerable attention for their ability to create visually appealing and highly realistic images that are often indistinguishable from real photographs. This surge in AI-generated imagery has been driven by advancements in deep learning techniques, particularly Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), which have shown remarkable success in generating high-quality images. The proliferation of AI-generated images poses both opportunities and challenges. On one hand, these images have potential applications in creative industries, advertising, and content creation, offering innovative ways to generate visual content rapidly and at scale. On the other hand, the indistinguishability of AI-generated images from real images raises concerns about misinformation, copyright infringement, and the potential for misuse in malicious contexts. The ability to accurately distinguish between AI-generated images and real images is crucial for maintaining the integrity of visual media. It has implications for verifying the authenticity of images in news, social media, and legal contexts, as well as for protecting intellectual property rights. Therefore, developing robust and reliable methods to classify and differentiate between AI-generated and real images is of paramount importance.

Dataset

The dataset for this project is sourced from Kaggle and consists of a curated collection of AI-generated images and real images. The dataset includes diverse image categories and encompasses a range of styles, resolutions, and content types to ensure model robustness and generalization. Specifically, the dataset is a captivating ensemble of images sourced from two distinct channels: web scraping and AI-generated content. The content covers many subjects with special emphasis on the following topics:

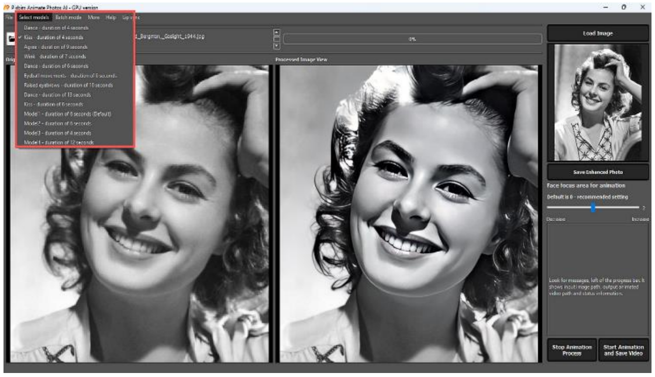
- People
- Animals
- Portraits
- Scenery
- Psychedelics

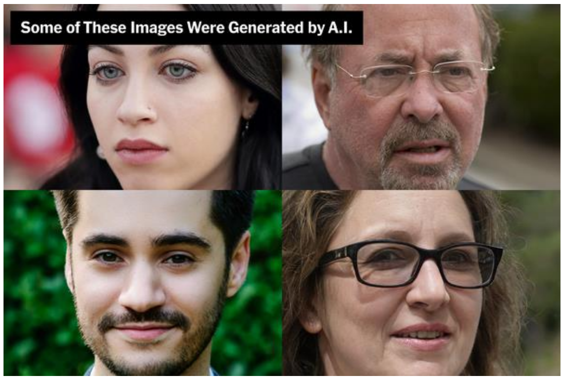
It was carefully curated and processed to ensure a balanced and representative sample for training, validation, and testing.

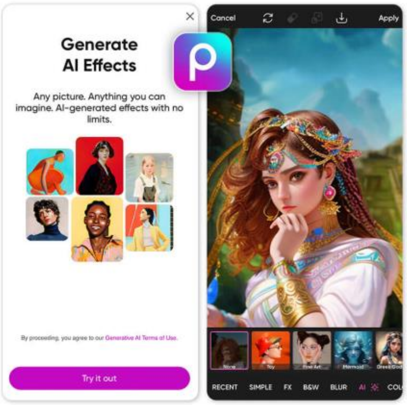
The initial dataset consisted of:

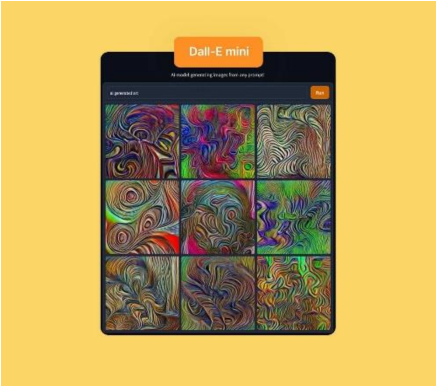
- **436 Real Images**: These images were sourced from web scraping.
- **539 AI-Generated Images**: Generated images from various frameworks, with prompts related to the content.

Upon examining the dataset, we observed that many of the AI-generated images contained credentials in the bottom right corner, which necessitated cropping. Additionally, some images were a concatenated result of four separate images, as suggested by the generative frameworks based on prompt engineering. We decided to divide these concatenated images into four distinct images. Moreover, we excluded some images that were deemed out of context or appeared as screenshots showcasing the en tire generative AI framework. For example,





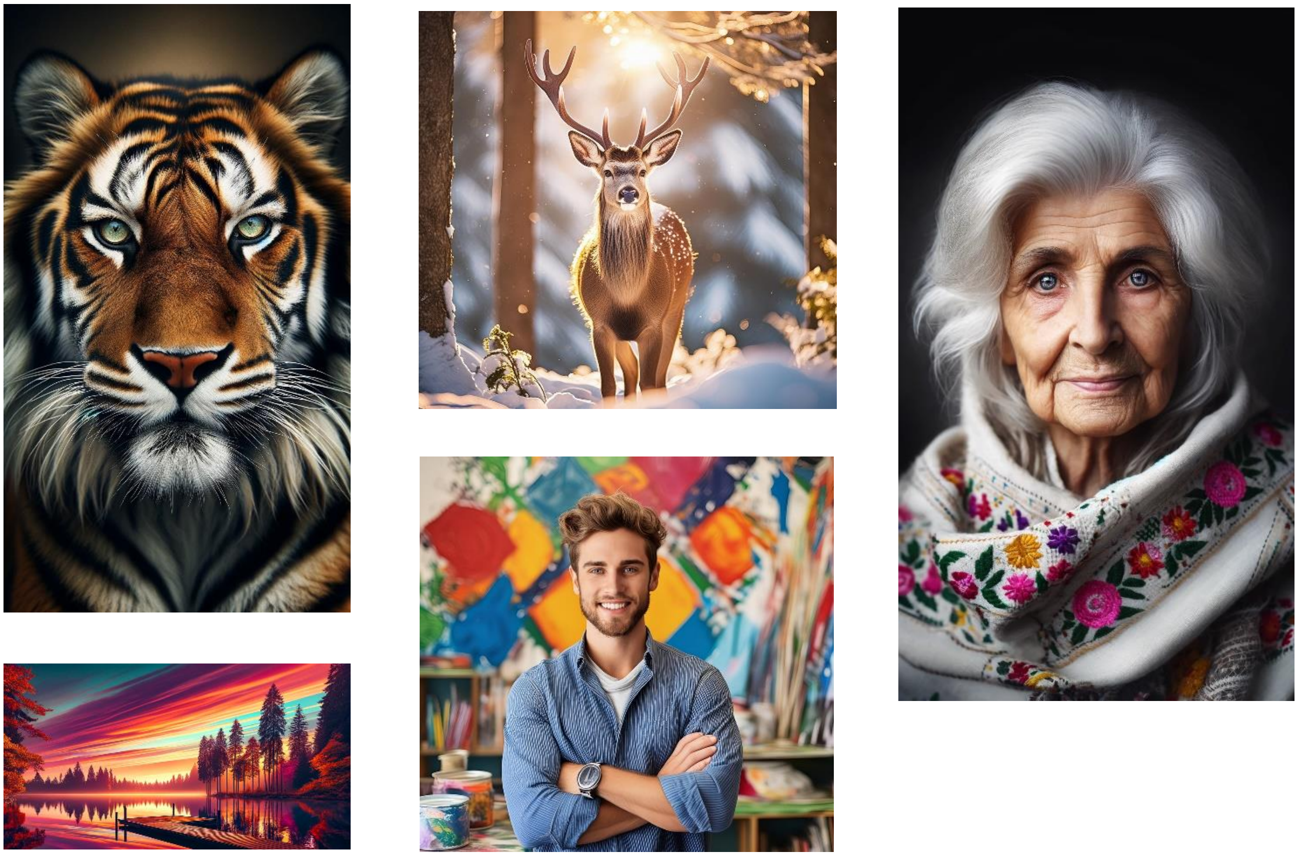

Generate
AI Effects
Any picture. Anything you can imagine. AI-generated effects with no limits.
Try it out
Cancel
Apply
RECENT
SIMPLE
FX
B&W
BLUR
AI



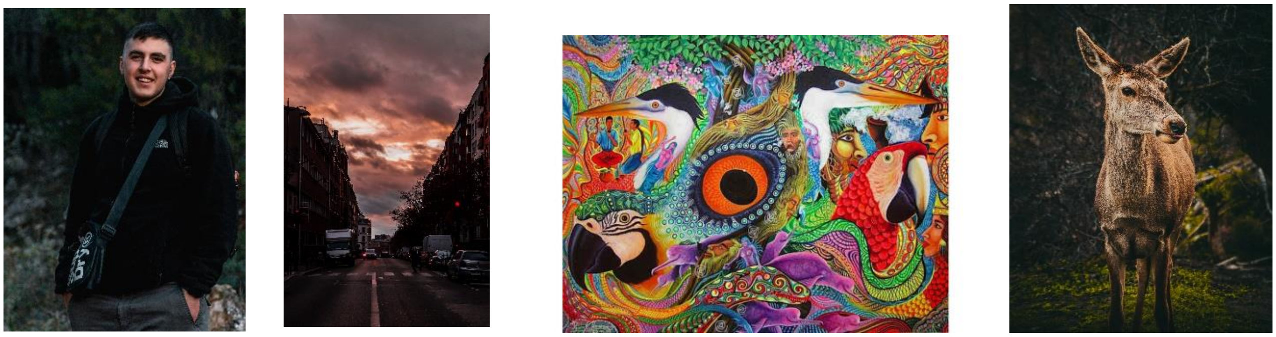
We ended up with a total of 430 real images and 499 AI-generated images, which were split into training and test datasets. The training dataset is composed of 400 real images and 469 AI-generated images, which were split into 80% for training and 20% for validation. The test dataset contains the remaining 30 real images and 30 AI-generated images.

Furthermore, we manually constructed an additional test set using the same sources as the Kaggle dataset to ensure consistency and comprehensiveness in our evaluation. This supplementary dataset includes 35 AI-generated images prompted by us and 35 real images web scraped.

Some of the AI Generated are:



And some of the real images:

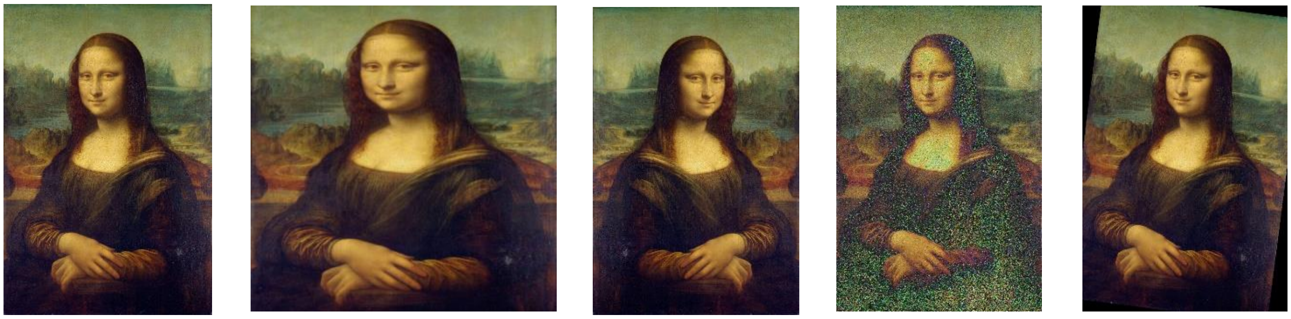


Data Augmentation

Data augmentation is a crucial step in training deep learning models, particularly when the dataset size is limited. It helps to artificially expand the size of the training dataset by creating modified versions of the existing images, which improves the model's ability to generalize to new, unseen data. In this project, various augmentation techniques were applied to the training dataset to enhance the robustness of our convolutional neural network (CNN) model. Initially, we trained our models using the original dataset without augmentation, but this led to early overfitting, where the models performed well on training data but poorly on validation. To mitigate this issue, we implemented data augmentation techniques to increase the variability of the training data.

The following augmentations were employed:

- Vertical Flip
- Rotation by ±15 degrees
- Random Crop (random portion of the image and resize to 512x512 pixels)
- Gaussian Noise



Model Development

In this section, we will define the models utilized in this project for comparison.

Lenet – 5

The first model used in this project was LeNet-5, a pioneering convolutional neural network (CNN) architecture developed by Yann LeCun. LeNet-5 was originally designed for handwritten digit classification and is well-known for its simplicity and effectiveness. In its standard form, LeNet-5 takes grayscale images of size 32x32 as input and utilizes the stochastic gradient descent algorithm as its activation function. However, for the purpose of this project, the architecture was adapted to handle RGB images of size 256x256 and ReLU activation function to ensure a fair comparison with other models.

The architecture of the modified LeNet-5 used in this project is as follows:

- **Input Layer**: 32x32x3 RGB images.
- **C1 Convolutional Layer**: 6 filters of size 5x5, stride = 1 followed by a batch normalization and ReLU activation function.
- **S2 Subsampling Layer**: Max pooling layer with a 2x2 filter.
- **C3 Convolutional Layer**: 16 filters of size 5x5, stride = 1 followed by a batch normalization and ReLU activation function.
- **S4 Subsampling Layer**: Max pooling layer with a 2x2 filter.
- **FC5**: Fully Connected layer with input channel 16x5x5 and output channel 120 followed by a ReLU activation function,
- **FC6**: Fully Connected layer with input channel 120 and output channel 84 followed by a ReLU activation function,
- **FC7**: Fully Connected layer with input channel 84 and output channel the number of classes (2).

Support Vector Machines

In addition to the LeNet-5 model, a Support Vector Machine (SVM) was also implemented as a traditional machine learning baseline. SVMs are effective classifiers that work well for a variety of tasks, particularly in cases where the number of dimensions is high. However, SVMs do not inherently work with raw image data. Therefore, a feature extraction step was necessary before applying the SVM.

To extract features, a pre-trained ResNet-18 model was employed. ResNet-18 is a widely recognized deep learning model known for its residual learning framework, which facilitates the training of very deep networks by addressing the vanishing gradient problem. The final fully connected layer of the ResNet-18 model was removed, allowing the output from the penultimate layer (a feature vector) to be utilized as input for the SVM. This output consists of 512 features, which are then fed into the SVM. We implement k - folds validation for statistic efficiency and a manually grid search for hyperparameter tuning where the best hyperparameters were

- C = 1.7
- gamma = 1/n_features = 0.002

This approach leverages the robust feature extraction capabilities of ResNet-18 and the strong classification performance of SVMs, creating a powerful combination for image classification tasks.

Convolutional Neural Network

For the CNN architecture, we drawn inspiration from advanced architectures like ResNet, VGG, and Inception, we design a deep convolutional neural network (CNN) that strategically employs large kernels in the initial layers to capture broad and general features from the images. As the network deepens, smaller kernels are utilized to preserve and extract finer details, ensuring that both high-level abstract features and low-level intricate patterns are effectively captured.

This architectural approach leverages the strengths of various successful paradigms in deep learning. For instance, similar to ResNet, our model incorporates deeper layers to learn complex representations. Inspired by VGG, we use smaller kernels in the deeper layers to maintain detailed information. From Inception, we adopt the concept of using varied kernel sizes to extract multi-scale features. Together, these design choices enable the model to achieve a high degree of accuracy in distinguishing between AI-generated and real images.

The input size of images is 256x256 as we want the most information in such difficult task and RGB channel.

1. Convolutional Layer 1:
    - Input Channels: 3 (RGB image)
    - Output Channels: 32
    - Kernel Size: 7x7
    - Stride: 1
    - Padding: 3
    - Followed by Batch Normalization, ReLU activation
    - Max Pooling: with a kernel size of 2 and stride of 2.
    - Output Size: 128x128x32

2. Convolutional Layer 2:
    - Input Channels: 32
    - Output Channels: 64
    - Kernel Size: 5x5
    - Stride: 1
    - Padding: 2
    - Followed by Batch Normalization, ReLU activation.
    - Output Size: 128x128x64

3. Convolutional Layer 3:
    - Input Channels: 64
    - Output Channels: 128
    - Kernel Size: 5x5
    - Stride: 1
    - Padding: 2
    - Followed by Batch Normalization, ReLU activation
    - Max Pooling: with a kernel size of 2 and stride of 2.
    - Output Size: 64x64x128

4. Convolutional Layer 4:
    - Input Channels: 128
    - Output Channels: 256
    - Kernel Size: 3x3
    - Stride: 1
    - Padding: 1
    - Followed by Batch Normalization, ReLU activation.
    - Output Size: 64x64x256

5. Convolutional Layer 5:
    - Input Channels: 256
    - Output Channels: 362
    - Kernel Size: 3x3
    - Stride: 1
    - Padding: 1
    - Followed by Batch Normalization, ReLU activation
    - Max Pooling: with a kernel size of 2 and stride of 2.
    - Output Size: 32x32x362

6. Convolutional Layer 6:
    - Input Channels: 362
    - Output Channels: 512
    - Kernel Size: 1x1
    - Stride: 1
    - Padding: 0
    - Followed by Batch Normalization, ReLU activation.
    - Output Size: 32x32x512

7. Convolutional Layer 7:
   - Input Channels: 512
   - Output Channels: 724
   - Kernel Size: 1x1
   - Stride: 1
   - Padding: 0
   - Followed by Batch Normalization, ReLU activation.
   - Output Size: 32x32x724

8. Convolutional Layer 8:
   - Input Channels: 724
   - Output Channels: 1024
   - Kernel Size: 1x1
   - Stride: 1
   - Padding: 0
   - Followed by Batch Normalization, ReLU activation
   - Max Pooling: with a kernel size of 2 and stride of 2.
   - Output Size: 16x16x1024

9. Fully Connected Layer 9:
   - Input Features: 16x16x1024 (flattened from the previous layer)
   - Output Features: 724
   - Followed by ReLU activation and Dropout with a rate of 0.5.

10. Fully Connected Layer 10:
    - Input Features: 724
    - Output Features: 516
    - Followed by ReLU activation and Dropout with a rate of 0.5.

11. Fully Connected Layer 11:
    - Input Features: 516
    - Output Features: 2 (number of classes)
    - This layer outputs the final predictions.

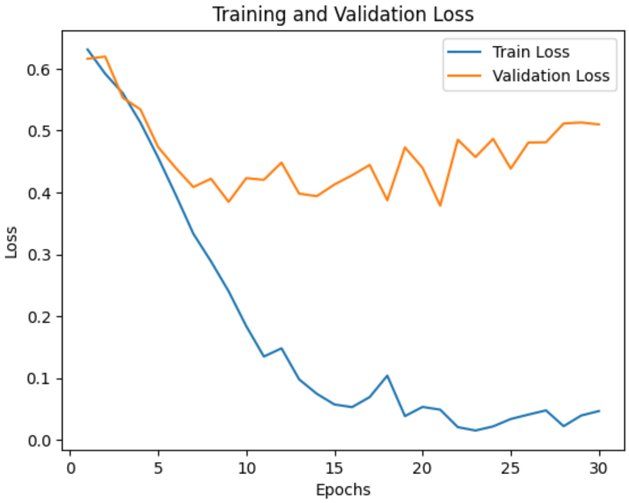
Training Neural Networks

Lenet-5

For the training phase of LeNet-5, we utilized Cross Entropy Loss as the loss function and the Adam optimizer with a learning rate of 0.001. The model was trained for 30 epochs with a batch size of 32.

These are the results of the training:

- Accuracy = 88.7 %
- Validation Loss = 0.378
- Precision =0.876
- Recall =0.875
- F1 score = 0.85



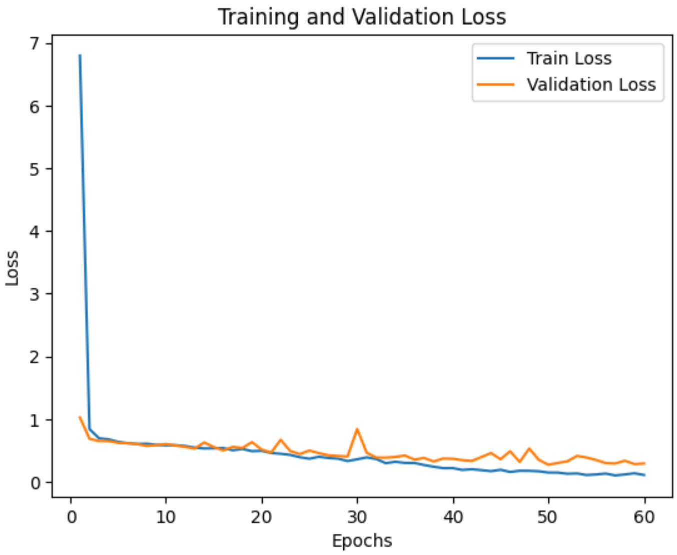
Both the training and validation loss decrease significantly during the first few epochs, indicating that the model is learning effectively from the data. After the initial drop, the training loss continues to decrease and stabilizes at a very low value, nearing zero. In contrast, the validation loss shows a more erratic behavior, decreasing initially but then fluctuating and not following the downward trend of the training loss. This divergence between training and validation loss is a classic sign of overfitting, where the model is learning the training data too well, including its noise and specifics, but is not generalizing effectively to unseen data.

CNN

For the training phase of our custom CNN, we employed Cross Entropy Loss as the loss function and the Adam optimizer with a learning rate of 0.001, as this combination proved to be the most effective in our trials. The model was trained with a batch size of 64 for a maximum of 200 epochs. However, we implemented early stopping, which halted training after 10 consecutive epochs without improvement in validation loss. Consequently, the model training concluded at 60 epochs.

These are the results of the training:

- Accuracy = 89.76 %
- Validation Loss = 0.294
- Precision =0.8976
- Recall =0.8974
- F1 score =0.8974



Similar to LeNet-5, the custom CNN shows a rapid decrease in training loss during the initial epochs. The validation loss decreases significantly in the early epochs and then fluctuates slightly around a lower value, indicating more steady learning with less variance compared to the training loss. Towards the end of the training, the validation loss shows horizontal stabilization, signaling that the model has reached its learning capacity. The implementation of early stopping effectively prevented overfitting, ensuring the model's ability to generalize well without further unnecessary training.

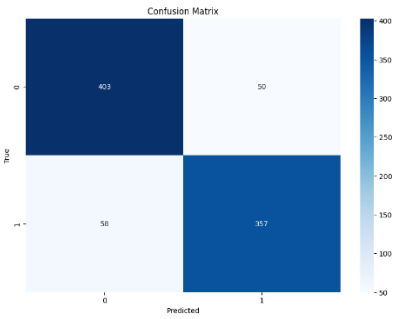
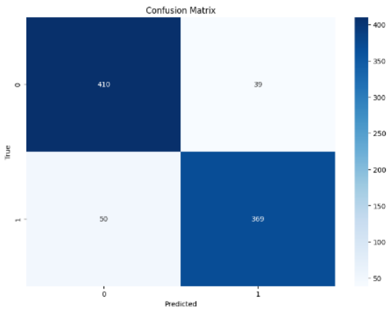
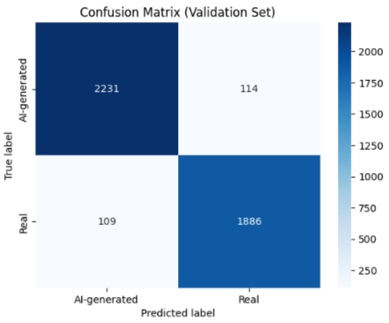
Results

Validation set

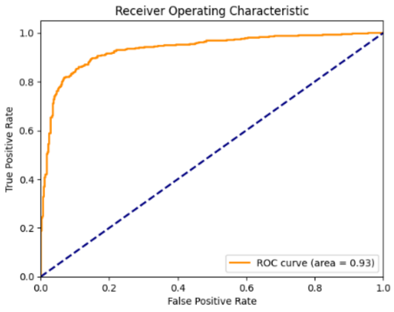
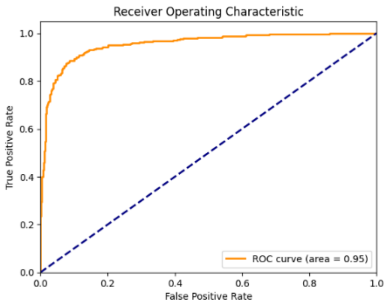
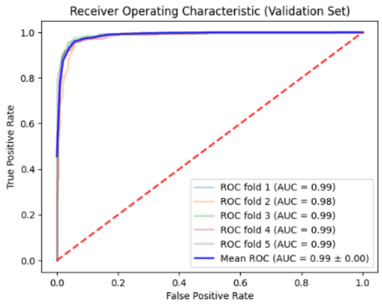
The metrics from Lenet – 5, our CNN and SVM are easily compared in the following table

	Lenet – 5	CNN	SVM
Accuracy (%)	87.56	89.76	95 ± 0.0
Validation Loss	0.51	0.294	-
Precision	0.876	0.8976	0.94 ± 0.02
Recall	0.875	0.8974	0.95 ± 0.01
F1 Score	0.875	0.8974	0.94 ± 0.0

The confusion matrix of the validation set of each classifier for the validation set.

	CNN	SVM
		

The ROC curve of the validation set of each classifier for the validation set.

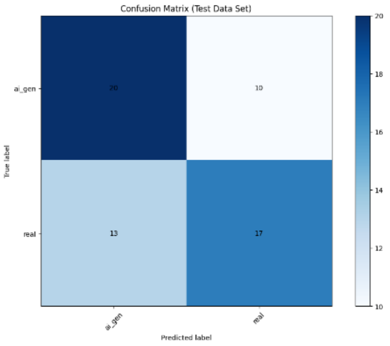
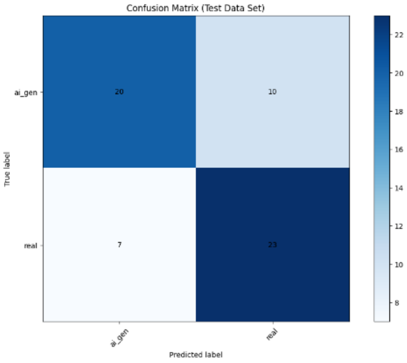
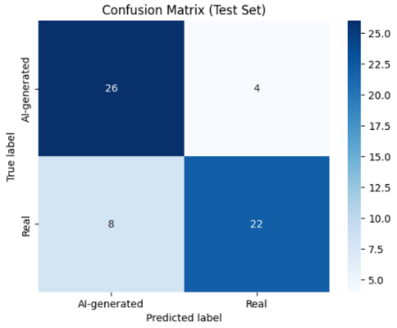
Lenet – 5	CNN	SVM
		

Test – Set

The metrics from Lenet – 5, our CNN and SVM are easily compared in the following table

	Lenet – 5	CNN	SVM
Accuracy (%)	62	72	80
Precision	62	72	0.85
Recall	62	72	0.73
F1 Score	62	72	0.79

The confusion matrix of the validation set of each classifier for the validation set.

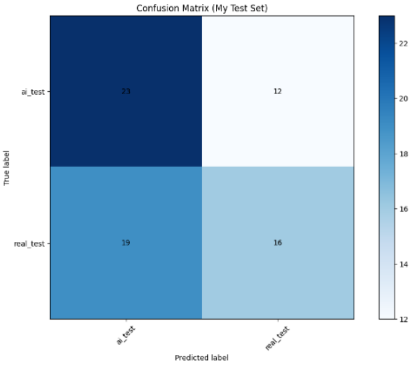
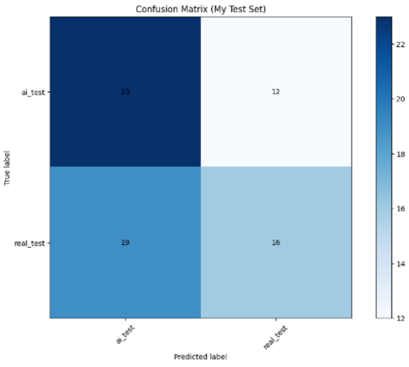
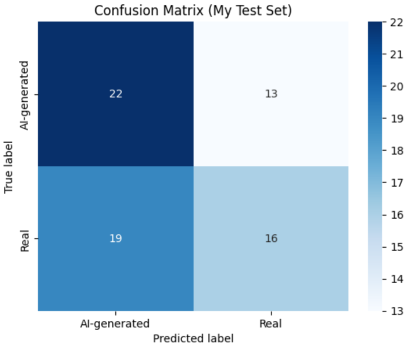
Lenet – 5	CNN	SVM
		

Our Test – Set

The metrics from Lenet – 5, our CNN and SVM are easily compared in the following table

	Lenet – 5	CNN	SVM
Accuracy (%)	56	50	54
Precision	0.56	0.5	0.55
Recall	0.56	0.5	0.46
F1 Score	0.55	0.5	0.5

The confusion matrix of our test set of each classifier for the validation set.

	CNN	SVM
		

Observations and Comments on the Results

The SVM model's superior performance can be attributed to the high-quality features extracted by ResNet-18. This combination leverages the deep learning model's ability to learn complex representations while relying on SVM's strength in classification. The custom CNN demonstrated strong performance, indicating that the architecture and data augmentation strategies were effective. However, its slightly lower performance compared to the SVM suggests that the CNN could benefit from further tuning and optimization. In contrast, LeNet-5, while showing decent results, exhibited signs of quick overfitting. This outcome can be attributed to the inherent limitations of a smaller architecture like LeNet-5, which struggles to effectively learn from data with high information complexity. It is worth noting that, despite LeNet-5 achieving an accuracy comparable to the custom CNN, there are significant differences in the loss values between the two models. This discrepancy indicates the varying levels of confidence each model has in its predictions. The higher loss value for LeNet-5 suggests less confidence in its predictions compared to the custom CNN, which demonstrates higher confidence through its lower loss value.

The test set results demonstrate a drop in performance across all models compared to the validation set, which is expected due to the unseen nature of the test data. However, SVM still maintained the highest accuracy, highlighting its robustness. The custom CNN exhibited a moderate drop in performance, underscoring its ability to classify accurately without overfitting to the training data. This result is particularly noteworthy when compared to the state-of-the-art ResNet combined with SVM. On the other hand, LeNet-5 demonstrated a significant drop in performance, reinforcing its tendency to overfit in training data and its limitations in handling the complex information contained.

The custom test set, which was manually constructed to ensure consistency and comprehensiveness, highlighted the challenges of creating an "on point" test dataset within the context of an existing dataset. This difficulty is evident as both the CNN and SVM, which performed well on the test set, failed to achieve noteworthy accuracy on the custom test set. Interestingly, by chance, LeNet-5 achieved the best accuracy on this custom set, despite its generally lower performance, further illustrating the unpredictability and complexity involved in dataset creation and model evaluation.

Conclusion

The project successfully demonstrated the capability of both deep learning and traditional machine learning methods in distinguishing AI-generated images from real images. The custom CNN achieved promising results. However, the project also highlighted the challenges associated with generalizing the model, fine-tuning its parameters, and achieving high performance metrics with limited resources. This underscores the importance of comprehensive data quality, extensive computational power, and robust model tuning in developing high-performing deep learning models. The use of a pre-trained ResNet-18 for feature extraction, combined with an SVM classifier, demonstrated the effectiveness of state-of-the-art. This approach not only leverages the powerful feature extraction capabilities of ResNet-18 but also underscores the necessity of extensive data for achieving superior performance in image classification tasks.

In addition, this project highlighted the critical importance of having good quality data. The initial exploration of the dataset revealed the need for rigorous preprocessing, such as cropping out irrelevant parts and splitting concatenated images. Ensuring the quality of data through meticulous preprocessing steps significantly contributed to the effectiveness of the models.

Overall, this project emphasizes the importance of comprehensive data preprocessing, the effectiveness of data augmentation, and the potential of deep learning models in image classification tasks.