UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI IFORMATICA

Corso di Laurea Magestrale in Internet of Things

AI-GENERATED ART DETECTION

Supervisor:
Prof.
Carmen BISOGNI

Candidati:
Zakarya BOUDRAF
Mohamed Aziz KHITMI
Melissa OULD BRAHAM

ACADEMIC YEAR 2023/2024

CONTENTS

CHAPTER 1

INTRODUCTION

Identifying and distinguishing AI-generated artworks is crucial for protecting the copyrights of human artists, particularly in the marketing and distribution of artwork where counterfeiting and piracy are prevalent issues. This ability also enhances transparency in the digital art market, allowing buyers and consumers to make informed purchasing decisions based on the origin of the artwork. Moreover, for scholars, art historians, and art enthusiasts, differentiating between AI-generated and human-created art aids in understanding artistic trends, technological evolution, and the impact of AI on art and human creativity.

CHAPTER 2

DATASETS USED

In this chapter, we will discuss the datasets used for our model.

2.1 AI-ArtBench

AI-ArtBench is a dataset that contains 180,000+ art images. 60,000 of them are human-drawn art that was directly taken from ArtBench-10 dataset and the rest is generated equally using Latent Diffusion and Standard Diffusion models. The human-drawn art is in 256x256 resolution and images generated using Latent Diffusion and Standard Diffusion has 256x256 and 768x768 resolutions respectively. [1]

Some of the characteristics of AI-ArtBench are:

- **Class-balanced**: The dataset contains an equal number of images for each of the 10 artistic styles. This is important for training machine learning models, as it prevents any one style from dominating the training process.

- **High-quality**: The images in the dataset are of high quality and have been carefully curated to remove any images that are blurry, low-resolution, or otherwise unsuitable for training machine learning models.

- **Cleanly annotated**: The images in the dataset are cleanly annotated with the correct artistic style. This is important for evaluating the performance of machine learning models.

- **Standardized**: The dataset has been created using standardized data collection, annotation, filtering, and preprocessing procedures. This makes it easy to use the dataset with a variety of machine learning frameworks and image synthesis codebases.

2.1.1 Examples

Here are some examples of images from the AI-ArtBench dataset:

1. Realism:

(a) Abbott Handerson Thayer: A Winged Figure

(b) Realism AI-generated image with Stable Diffusion

Figure 2.1: Real and AI-Generated "Realism" Style Art

2. Renaissance:

(a) Adam Van Noort: Christ Blessing The Little Children

(b) Renaissance AI-generated image with Stable Diffusion

Figure 2.2: Real and AI-Generated "Renaissance" Style Art

3. Ukiyo-e:

(a) Portrait by Adachi Ginko

(b) Ukiyo-e AI-generated image with Latent Diffusion

Figure 2.3: Real and AI-Generated "Ukiyo-e" Style Art

4. Baroque:

(a) Abraham Storck: A Dutch Harbour Scene with Ships and Bathers

(b) Baroque AI-generated image with Latent Diffusion

Figure 2.4: Real and AI-Generated "Baroque" Style Art

2.2 AI and Human Art Classification

The "AI and Human Art Classification" dataset is a collection of images designed to train machine learning models to distinguish between human-made and AI-generated art. Key points of the dataset are:

- **Source**: This dataset is publicly available on Kaggle.[2]
- **Size**: It's a relatively smaller dataset compared to AI-ArtBench, containing around 12,000 images. These images are divided equally between human-made and AI-generated art categories.
- **Content**: There's limited information available regarding the specific artistic styles or AI generation models used for the AI-generated art within the dataset. This differs from AI-ArtBench, which provides a more controlled environment with documented styles and models.

Here are some characteristics of the AI and Human Art Classification dataset:

- **Balanced Classes**: The dataset maintains an equal number of images in both the human-made and AI-generated categories. This is crucial for training machine learning models to avoid bias towards a specific category.

- **Limited Scope**: Compared to AI-ArtBench with its 10 categorized styles, the artistic variety in this dataset might be broader.

- **Focus on Classification**: This dataset is specifically designed for the task of classifying art as human-made or AI-generated. It might not contain additional metadata like the one found in AI-ArtBench.

2.2.1 Examples

(a) AI-generated image 1

(b) AI-generated image 2

(c) AI-generated image 3

(d) AI-generated image 4

Figure 2.5: AI-Generated images

(a) Non AI-generated image 1

(b) Non AI-generated image 2

(c) Non AI-generated image 3

(d) Non AI-generated image 4

Figure 2.6: Non AI-generated images

CHAPTER 3

ARCHITECTURES USED

This chapter will delve into the specific Machine Learning architectures employed in our project.

3.1 CNN

Convolutional Neural Networks (CNNs) are a class of deep neural networks particularly well-suited for image classification tasks. CNNs automatically and adaptively learn spatial hierarchies of features from input images through convolutional layers, pooling layers, and fully connected layers. [3] [4]

3.2 VGGNet

VGGNet is a type of CNN known for its simplicity and depth. It was introduced by the Visual Geometry Group (VGG) at the University of Oxford. VGGNet is characterized by its use of very small (3x3) convolution filters and a deep architecture with many layers. [5]

3.3 ResNet

Residual Networks (ResNet) address the problem of vanishing gradients in deep networks by introducing residual learning. ResNet allows training of extremely deep networks by adding shortcut connections that bypass one or more layers. [6]

3.4 DenseNet

Densely Connected Networks (DenseNet) extend the idea of residual connections in ResNet by connecting each layer to every other layer in a feed-forward manner. This ensures maximum information flow between layers in the network. [7]

CHAPTER 4

DATA PREPROCESSING

In this chapter, we discuss the preprocessing steps applied to the dataset to prepare it for training the convolutional neural network (CNN) models. Preprocessing is a crucial step in machine learning workflows as it ensures that the data is in a suitable format for the model to learn effectively. The preprocessing steps used in this study involve image normalization and data augmentation using the `ImageDataGenerator` class from TensorFlow's Keras API.

4.1 Image Normalization

Image normalization is a technique used to scale the pixel values of images to a specific range. In this study, we rescale the pixel values of all images to the range [0, 1] by dividing each pixel value by 255. This is done using the `rescale` parameter of the `ImageDataGenerator` class. Normalizing the pixel values helps in speeding up the convergence of the neural network during training.

4.2 Creating the Training Dataset

The training dataset is created using the `flow_from_dataframe` method of the `ImageDataGenerator` class. This method generates batches of tensor image data with real-time data augmentation. The following steps were performed to create the training dataset:

1. The `dataframe` parameter is set to `train_data`, which contains the file paths and corresponding labels of the training images.

2. The `x_col` parameter specifies the column in the dataframe that contains the file paths of the images.

3. The `y_col` parameter specifies the column in the dataframe that contains the labels of the images.

4. The `target_size` parameter is set to (32, 32), which resizes all images to 32x32 pixels.

5. The `batch_size` parameter is set to 64, indicating that the images will be processed in batches of 64.

6. The `class_mode` parameter is set to `categorical`, which means that the labels are one-hot encoded.

7. The `shuffle` parameter is set to `True`, ensuring that the images are shuffled before being passed to the model.

4.3 Summary

In summary, the preprocessing steps involve normalizing the pixel values of the images and setting up the training dataset with the desired configurations. Although data augmentation techniques were considered, they were not applied in the current setup. These preprocessing steps ensure that the data is in an appropriate format for the CNN model to learn effectively.

CHAPTER 5

FIRST TRAINING PHASE

In this chapter, we describe the initial training phase where a Convolutional Neural Network (CNN) is employed to classify each artwork into one of two classes: AI-generated or real art. The architecture of the CNN, its training parameters are discussed in detail.

5.1 CNN Architecture

The CNN architecture used in this study is designed to effectively capture and classify features from the input images. The architecture is composed of convolutional layers, pooling layers, and fully connected (dense) layers.

5.1.1 Convolutional Layers

The model begins with two convolutional layers:

- **First Convolutional Layer**: This layer has 64 filters, each of size 3x3. The activation function used is ReLU (Rectified Linear Unit). The input shape is set to (32, 32, 3), corresponding to the resized images.
- **Second Convolutional Layer**: This layer also has 64 filters of size 3x3 with ReLU activation.

5.1.2 Pooling Layers

Each convolutional layer is followed by a max-pooling layer, which reduces the spatial dimensions of the feature maps:

- **MaxPooling2D**: This operation reduces the dimensions, which helps in reducing computational complexity and preventing overfitting.

5.1.3 Fully Connected Layers

After flattening the feature maps, the model includes two fully connected (dense) layers:

- **First Dense Layer**: This layer has 32 units and uses ReLU activation.
- **Second Dense Layer**: This layer has 16 units and also uses ReLU activation.
- **Output Layer**: The final dense layer has 2 units with a softmax activation function to classify the input images into two categories: AI-generated or real art.

5.2 Training the Model

The model is compiled using the Adam optimizer and binary crossentropy loss function. Accuracy is used as the metric to evaluate the performance of the model.

5.2.1 Compilation and Summary

The model summary provides a comprehensive overview of each layer, including the output shape and the number of parameters.

- **Total params:** 113,042 (441.57 KB)
- **Trainable params:** 113,042 (441.57 KB)
- **Non-trainable params:** 0 (0.00 B)

Layer (type)	**Output Shape**	**Param #**
conv2d (Conv2D)	(None, 30, 30, 64)	1,792
max_pooling2d (MaxPooling2D)	(None, 15, 15, 64)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 32)	73,760
dense_1 (Dense)	(None, 16)	528
dense_2 (Dense)	(None, 2)	34

Table 5.1: Summary of the CNN model architecture

5.3 Training Process

First, we ranked art styles from the least abstract to the most abstract, since we felt that it would be easier to classify art in the less abstract art pieces.

Here's the order we went with:

1. Realism
2. Renaissance
3. Ukiyo-e
4. Baroque
5. Post-Impressionism
6. Impressionism
7. Romanticism
8. Expressionism
9. Surrealism
10. Art Nouveau

The model is trained using the training dataset (of each style separately) prepared in the preprocessing phase. The training process involves multiple epochs where the model iteratively learns to classify the input images accurately. During training, the model's performance is monitored using a validation dataset, and a checkpoint callback is used to save the best model based on validation loss.

5.3.1 Training Setup

The training process is initiated with the following parameters:

- **Training dataset:** The preprocessed training data.
- **Validation dataset:** A separate set of data used to evaluate the model's performance during training.
- **Epochs:** The number of complete passes through the training dataset. In this study, we set the number of epochs to a predefined value (e.g., 50).
- **Callbacks:** A list of callback functions that are applied at different stages of training. In this case, we use a model checkpoint callback to save the best model based on validation loss.

5.3.2 Training Execution

The training is executed using the `fit` method of the Keras model, as shown in the following code snippet:

```
history = model.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=[model_checkpoint_callback]
)
```

5.3.3 Checkpointing

During training, the `model_checkpoint_callback` is used to monitor the validation loss and save the model's weights whenever an improvement is detected. This approach ensures that the **best model** is retained, preventing overfitting and ensuring better generalization on the test data.

5.3.4 Training History

The `history` object returned by the `fit` method contains detailed logs of the training process, including the training and validation loss and accuracy for each epoch. This information is useful for analyzing the model's learning curve and identifying any potential issues such as overfitting or underfitting.

5.4 Results

After training, the model's performance is evaluated on the test dataset. Key metrics such as accuracy, precision, recall, and the confusion matrix are used to assess the model's effectiveness in classifying AI-generated and real artworks.

5.4.1 Realism

	precision	recall	f1-score	support
0	0.94	0.96	0.95	2000
1	0.92	0.88	0.90	1000
accuracy			0.93	3000
macro avg	0.93	0.92	0.93	3000
weighted avg	0.93	0.93	0.93	3000

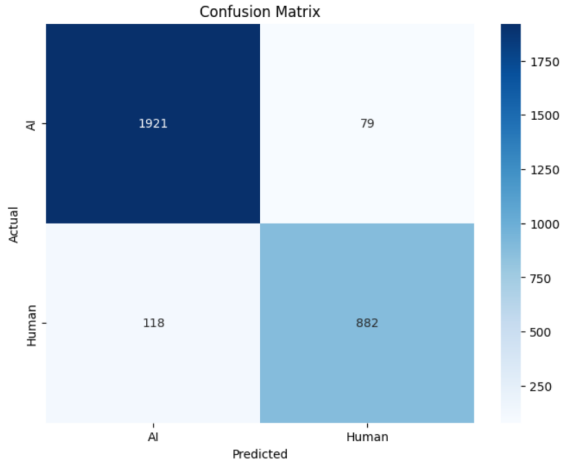

Figure 5.1: Metrics and Confusion Matrix of the CNN on "Realism" Art Style

5.4.2 Renaissance

	precision	recall	f1-score	support
0	0.95	0.98	0.96	2000
1	0.95	0.89	0.92	1001
accuracy			0.95	3001
macro avg	0.95	0.93	0.94	3001
weighted avg	0.95	0.95	0.95	3001

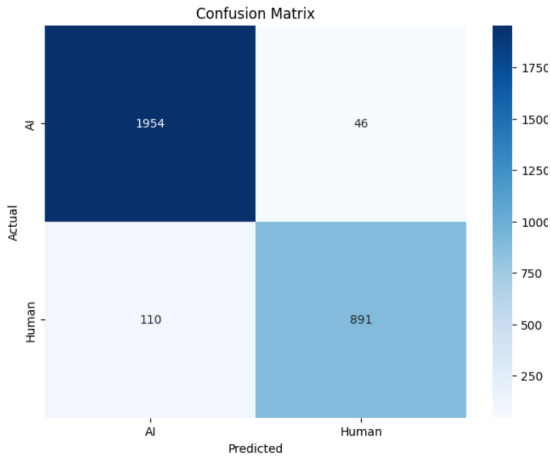

Figure 5.2: Metrics and Confusion Matrix of the CNN on "Renaissance" Art Style

5.4.3 Ukiyo-e

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2000
1	0.97	0.98	0.98	1000
accuracy			0.98	3000
macro avg	0.98	0.98	0.98	3000
weighted avg	0.99	0.98	0.99	3000

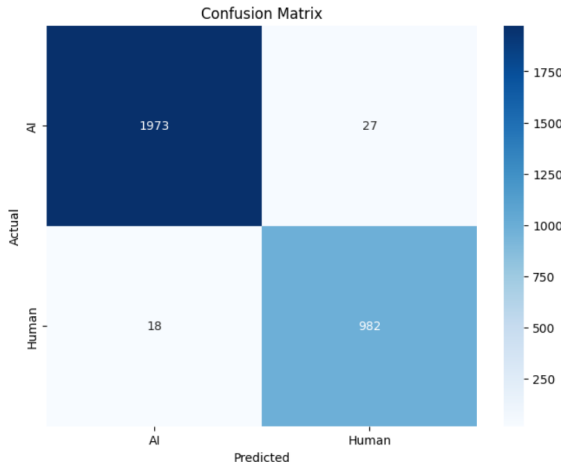

Figure 5.3: Metrics and Confusion Matrix of the CNN on "Ukiyo-e" Art Style

5.4.4 Baroque

	precision	recall	f1-score	support
0	0.97	0.98	0.97	2000
1	0.96	0.93	0.95	1000
accuracy			0.97	3000
macro avg	0.97	0.96	0.96	3000
weighted avg	0.97	0.97	0.97	3000

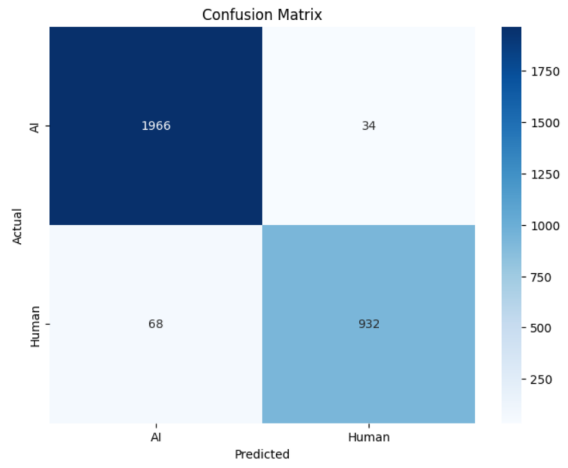

Figure 5.4: Metrics and Confusion Matrix of the CNN on "Baroque" Art Style

5.4.5 Post-Impressionism

	precision	recall	f1-score	support
0	0.97	0.97	0.97	2000
1	0.94	0.94	0.94	1000
accuracy			0.96	3000
macro avg	0.95	0.95	0.95	3000
weighted avg	0.96	0.96	0.96	3000

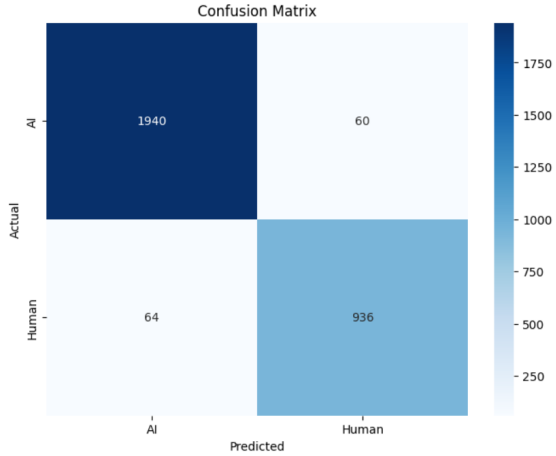

Figure 5.5: Metrics and Confusion Matrix of the CNN on "Post-Impressionism" Art Style

5.4.6 Impressionism

	precision	recall	f1-score	support
0	0.98	0.99	0.98	2000
1	0.97	0.95	0.96	1000
accuracy			0.98	3000
macro avg	0.98	0.97	0.97	3000
weighted avg	0.98	0.98	0.98	3000

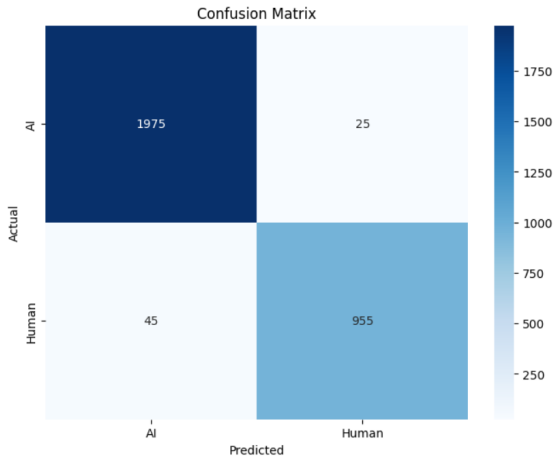

Figure 5.6: Metrics and Confusion Matrix of the CNN on "Impressionism" Art Style

5.4.7 Romanticism

	precision	recall	f1-score	support
0	0.93	0.97	0.95	2000
1	0.93	0.85	0.89	1000
accuracy			0.93	3000
macro avg	0.93	0.91	0.92	3000
weighted avg	0.93	0.93	0.93	3000

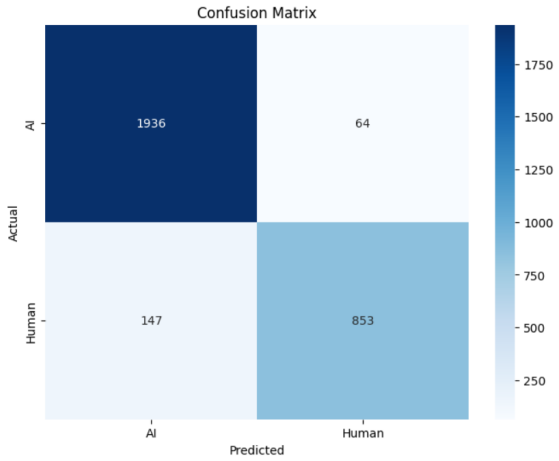

Figure 5.7: Metrics and Confusion Matrix of the CNN on "Romanticism" Art Style

5.4.8 Expressionism

	precision	recall	f1-score	support
0	0.97	0.96	0.97	2000
1	0.93	0.94	0.94	1000
accuracy			0.96	3000
macro avg	0.95	0.95	0.95	3000
weighted avg	0.96	0.96	0.96	3000

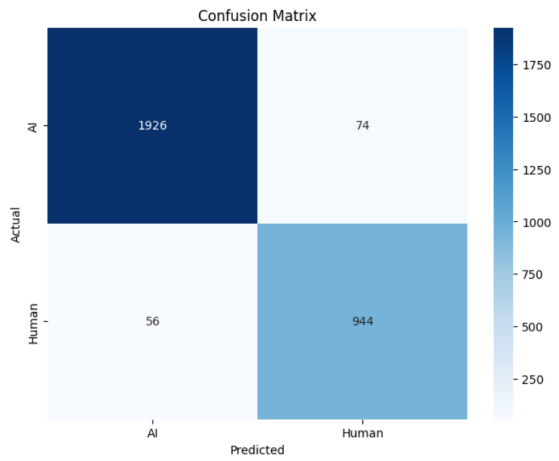

Figure 5.8: Metrics and Confusion Matrix of the CNN on "Expressionism" Art Style

5.4.9 Surrealism

	precision	recall	f1-score	support
0	0.94	0.93	0.93	2000
1	0.86	0.88	0.87	1000
accuracy			0.91	3000
macro avg	0.90	0.90	0.90	3000
weighted avg	0.91	0.91	0.91	3000

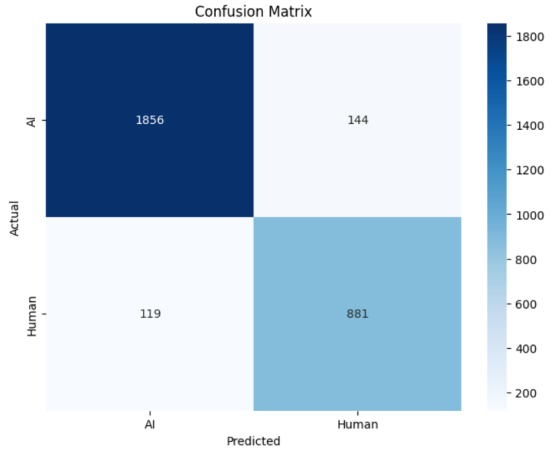

Figure 5.9: Metrics and Confusion Matrix of the CNN on "Surrealism" Art Style

5.4.10 Art Nouveau

	precision	recall	f1-score	support
0	0.99	0.98	0.98	2000
1	0.96	0.97	0.97	1000
accuracy			0.98	3000
macro avg	0.98	0.98	0.98	3000
weighted avg	0.98	0.98	0.98	3000

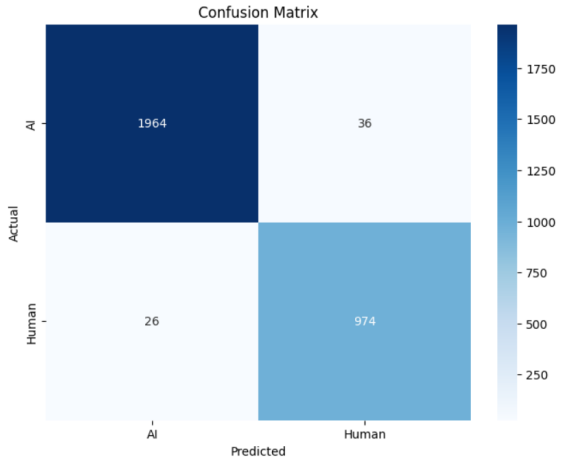

Figure 5.10: Metrics and Confusion Matrix of the CNN on "Art Nouveau" Art Style

5.5 Model Performance

Table 5.2: Classification Reports for Different Models

	Fake			Real			Overall
Art Style	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**	**Acc.**
Ukiyo-e	**0.99**	**0.99**	**0.99**	**0.97**	**0.98**	**0.98**	**0.98**
Art Nouveau	**0.99**	0.98	0.98	0.96	0.97	0.97	**0.98**
Impressionism	0.98	**0.99**	0.98	**0.97**	0.95	0.96	**0.98**
Baroque	0.97	0.98	0.97	0.96	0.93	0.95	0.97
Expressionism	0.97	0.96	0.97	0.93	0.94	0.94	0.96
Post-Impressionism	0.97	0.97	0.97	0.94	0.94	0.94	0.96
Renaissance	0.95	0.98	0.96	0.95	0.89	0.92	0.95
Realism	0.94	0.96	0.95	0.92	0.88	0.90	0.93
Romanticism	0.93	0.97	0.95	0.93	0.85	0.89	0.93
Surrealism	0.94	0.93	0.93	0.86	0.88	0.87	0.91

The performance metrics and visualizations, confusion matrix, and classification report, provide insights into the model's strengths and areas for improvement. These results are crucial for understanding how well the model generalizes to new, unseen data and for guiding future enhancements to the model architecture and training process.

5.6 Discussion about the Results vs the Expectations

According to the results we obtained, the art styles from lowest to highest error rate are as follows:

1. Ukiyo-e
2. Art Nouveau

3. Impressionism
4. Baroque
5. Expressionism
6. Post-Impressionism
7. Renaissance
8. Realism
9. Romanticism
10. Surrealism

5.6.1 Ukiyo-e

The easiest art style to identify was Ukiyo-e, probably due to its unique 2D format and colors. From the dataset, we notice that the AI-generated Ukiyo-e images have high pigmentations, contrary to real paintings that are more muted.

5.6.2 Art Nouveau

This style was easy to detect probably due to the variety of the dataset, but from a human perspective, it is difficult to identify due to the inconsistency in the patterns.

5.6.3 Impressionism

AI finds it easier to classify probably due to the AI's ability to identify patterns, even detailed ones, and impressionism is known for its brushwork.

5.6.4 Baroque

Both human and AI find this style relatively easy to classify.

5.6.5 Expressionism

It's not clear why the AI finds it easier to identify, but perhaps it is due to the GANs failing to recreate a proper Expressionism painting.

5.6.6 Post-Impressionism

This style was at the same level of difficulty to identify for both humans and AI, perhaps due to its closeness to Impressionism.

5.6.7 Renaissance and Realism

These styles are found to be more difficult to identify than expected, likely due to the GAN's ability to recreate them.

5.6.8 Romanticism and Surrealism

Both human and AI find these styles hard to identify because of the inconsistencies in their style.

5.6.9 Conclusion

We conclude that the patterns learned by AI contribute significantly to the classification of art styles.

5.7 Conclusion

The first training phase demonstrates the ability of a CNN to learn and classify artworks into AI-generated and real categories. The model's architecture, training parameters, and initial results provide a foundation for further improvements and fine-tuning in subsequent phases.

CHAPTER 6

SECOND TRAINING PHASE

In this phase of the project, we employed several advanced convolutional neural network (CNN) architectures, namely VGGNet, ResNet, and DenseNet, to classify art styles using the entire dataset. Unlike the previous phase where each art style was trained separately, here the entire dataset is utilized at once, allowing the models to learn more generalized features across different art styles.

6.0.1 Model Architectures

We used Four different CNN architectures for training:

Two different CNNs (our own architectures)

Layer (type)	**Output Shape**	**Param #**
conv2d (Conv2D)	(None, 30, 30, 64)	1,792
max_pooling2d (MaxPooling2D)	(None, 15, 15, 64)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 32)	73,760
dense_1 (Dense)	(None, 16)	528
dense_2 (Dense)	(None, 2)	34

Table 6.1: Summary of the First CNN model architecture

Layer (type)	**Output Shape**	**Param #**
conv2d_2 (Conv2D)	(None, 30, 30, 512)	14,336
max_pooling2d_2 (MaxPooling2D)	(None, 15, 15, 512)	0
conv2d_3 (Conv2D)	(None, 13, 13, 128)	589,952
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 128)	0
conv2d_4 (Conv2D)	(None, 4, 4, 32)	36,896
max_pooling2d_4 (MaxPooling2D)	(None, 2, 2, 32)	0
flatten_1 (Flatten)	(None, 128)	0
dense_3 (Dense)	(None, 32)	4,128
dense_4 (Dense)	(None, 16)	528
dense_5 (Dense)	(None, 2)	34

Table 6.2: Summary of the Second CNN model architecture

VGGNet19, VGGNet16

ResNet34

DenseNet121

6.1 Results

We evaluated the performance of each model using metrics such as accuracy, precision, recall, F1-score, and confusion matrices.

6.1.1 CNN First

	precision	recall	f1-score	support
0	0.95	0.95	0.95	20000
1	0.90	0.90	0.90	10000
accuracy			0.94	30000
macro avg	0.93	0.93	0.93	30000
weighted avg	0.94	0.94	0.94	30000

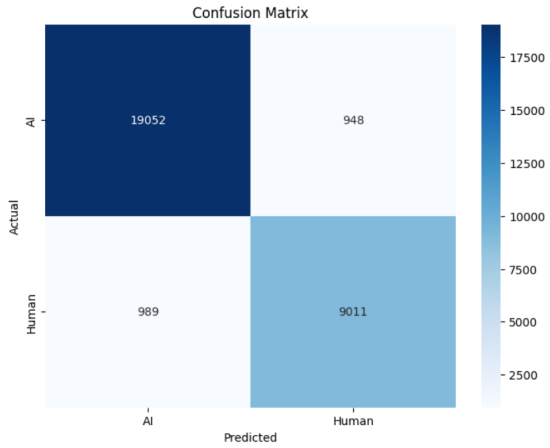

Figure 6.1: Metrics and Confusion Matrix of CNN First on the Entire Dataset

6.1.2 CNN Second

	precision	recall	f1-score	support
0	0.95	0.96	0.96	20000
1	0.91	0.91	0.91	10000
accuracy			0.94	30000
macro avg	0.93	0.93	0.93	30000
weighted avg	0.94	0.94	0.94	30000

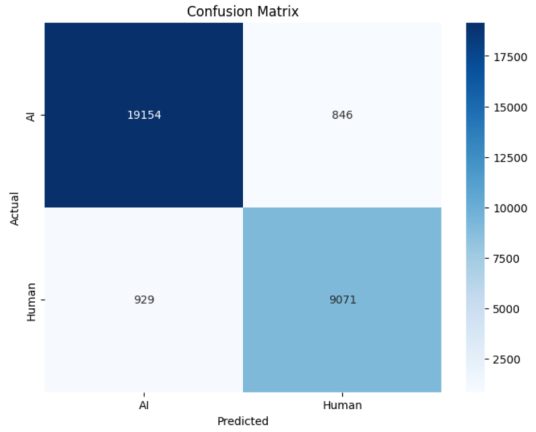

Figure 6.2: Metrics and Confusion Matrix of CNN Second on the Entire Dataset

6.1.3 VGGNet 19

	precision	recall	f1-score	support
0	0.93	0.95	0.94	20000
1	0.90	0.87	0.88	10000
accuracy			0.92	30000
macro avg	0.91	0.91	0.91	30000
weighted avg	0.92	0.92	0.92	30000

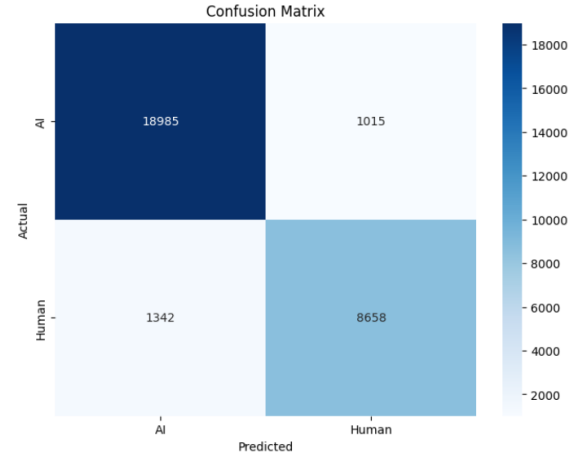

Figure 6.3: Metrics and Confusion Matrix of VGGNet on the Entire Dataset

6.1.4 ResNet 34

	precision	recall	f1-score	support
0	0.96	0.93	0.94	20000
1	0.86	0.91	0.89	10000
accuracy			0.92	30000
macro avg	0.91	0.92	0.91	30000
weighted avg	0.92	0.92	0.92	30000

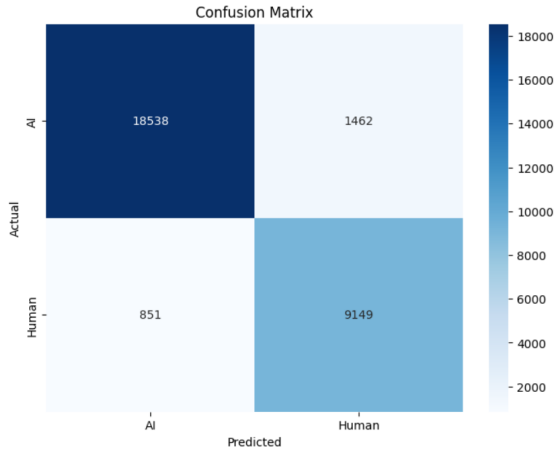

Figure 6.4: Metrics and Confusion Matrix of ResNet on the Entire Dataset

6.1.5 DenseNet 121

	precision	recall	f1-score	support
0	0.94	0.97	0.96	20000
1	0.95	0.88	0.91	10000
accuracy			0.94	30000
macro avg	0.94	0.93	0.93	30000
weighted avg	0.94	0.94	0.94	30000

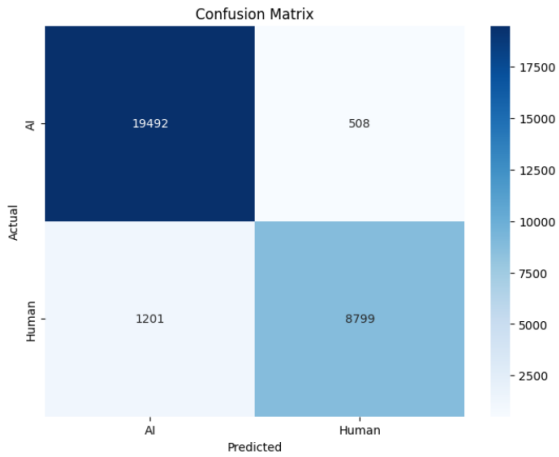

Figure 6.5: Metrics and Confusion Matrix of DenseNet on the Entire Dataset

6.2 Discussion

Our CNNs and ResNet get a better result at 94% where DenseNet under perform at 92%.

6.3 Conclusion

Training on the entire dataset using advanced CNN architectures like VGGNet, ResNet, and DenseNet has allows generalization compared to training on individual art styles separately. Future work may include further fine-tuning these models and exploring additional architectures for even better results.

CHAPTER 7

TESTING ON THE SECOND DATASET

7.1 Evaluation Methodology

The evaluation was conducted by directly applying the previously trained models to the "AI and Human Art Classification" dataset. We measured the performance using the same metrics: accuracy, precision, recall, F1-score, and confusion matrices.

7.2 Results

7.2.1 CNN First

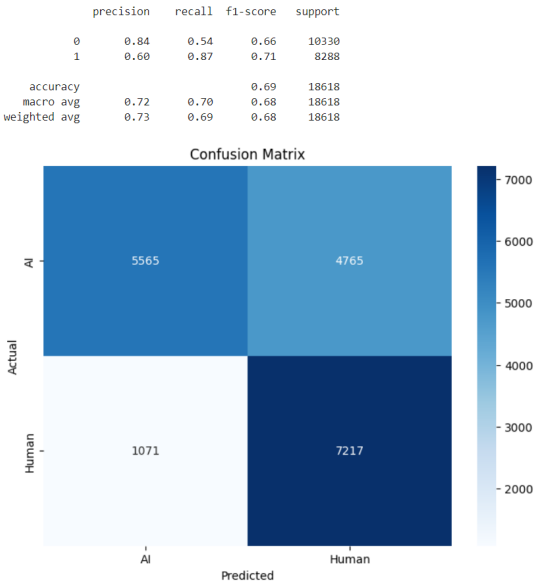

Figure 7.1: Metrics and Confusion Matrix of CNN First on the Second Dataset

7.2.2 CNN Second

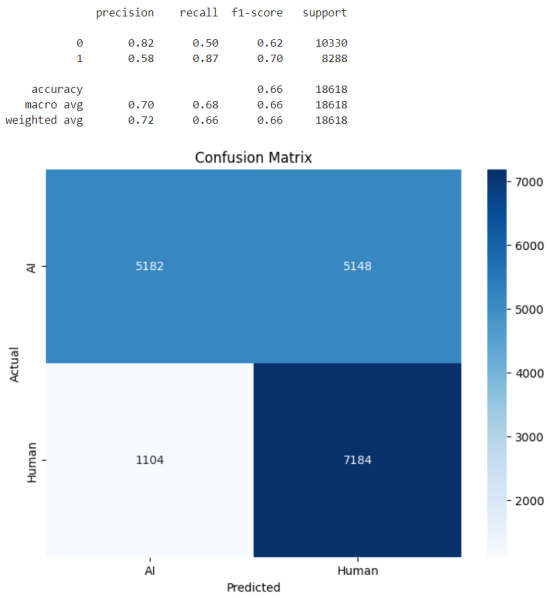

Figure 7.2: Metrics and Confusion Matrix of CNN Second on the Entire Dataset

7.2.3 VGGNet 19

	precision	recall	f1-score	support
0	0.55	1.00	0.71	10330
1	0.00	0.00	0.00	8288
accuracy			0.55	18618
macro avg	0.28	0.50	0.36	18618
weighted avg	0.31	0.55	0.40	18618

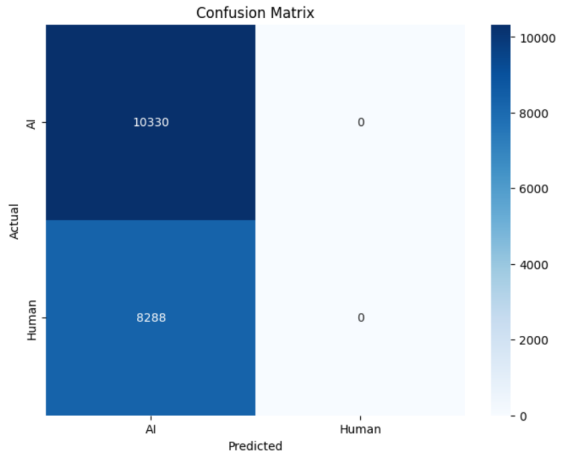

Figure 7.3: Metrics and Confusion Matrix of VGGNet on the Second Dataset

7.2.4 ResNet 34

	precision	recall	f1-score	support
0	0.83	0.42	0.56	10330
1	0.55	0.89	0.68	8288
accuracy			0.63	18618
macro avg	0.69	0.66	0.62	18618
weighted avg	0.71	0.63	0.61	18618

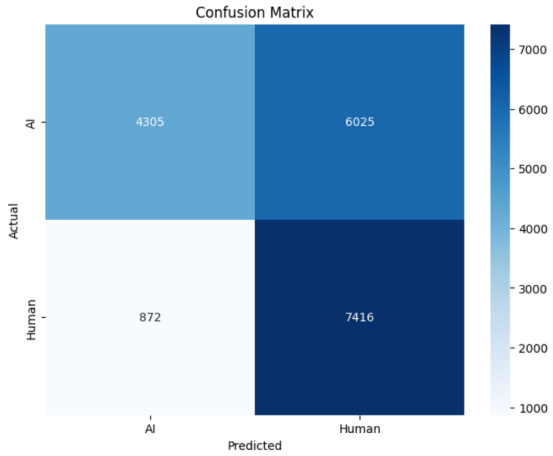

Figure 7.4: Metrics and Confusion Matrix of ResNet on the Second Dataset

7.2.5 DenseNet 121

	precision	recall	f1-score	support
0	0.84	0.54	0.66	10330
1	0.60	0.87	0.71	8288
accuracy			0.69	18618
macro avg	0.72	0.70	0.68	18618
weighted avg	0.73	0.69	0.68	18618

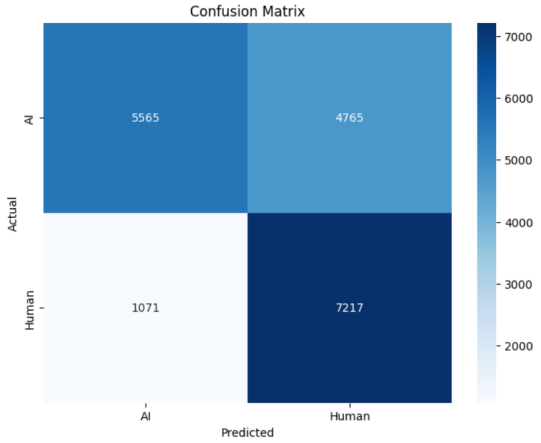

Figure 7.5: Metrics and Confusion Matrix of DenseNet on the Second Dataset

7.3 Discussion

The results of evaluating the models on the "AI and Human Art Classification" dataset show that even though the models perform relatively well on the human art, they fail on the classification of AI generated art almost 50% split.

With the exception of VGGNet19 which completely failed at classifying art in the second dataset.

CHAPTER 8

GENERAL CONCLUSION

In this study, we explored the use of various convolutional neural network (CNN) architectures, two of our own architectures, specifically VGGNet, ResNet, and DenseNet, for the classification of art styles. The training was performed on separate styles, then the entire dataset, allowing the models to learn from a comprehensive set of art style features.

Our results demonstrated that the our Second CNN architecture achieved the best performance across all evaluation metrics, including accuracy, precision, recall, and F1-score. ResNet got similar results.The residual connections in ResNet effectively mitigated the vanishing gradient problem, enabling the network to learn deeper and more complex representations of the data. This resulted in superior generalization and classification performance compared to VGGNet and DenseNet.

DenseNet also performed well, benefiting from its dense connections that facilitate maximum information and gradient flow between layers. However, it was ResNet that consistently delivered the highest accuracy and most reliable results across different art styles.

In conclusion, training on the entire dataset with advanced CNN architectures significantly enhanced the classification performance. Among the architectures tested, our Second CNN emerged as the most effective model, providing the best

results at **94% accuracy** in classifying the diverse art styles in our dataset. Future research may focus on further optimizing these models and exploring additional architectures to push the boundaries of art style classification even further.

REFERENCES

[1] Ravi Dussilva. *Real AI Art*. `https://www.kaggle.com/datasets/ravidussilva/real-ai-art`. Accessed: 2024-06-10. 2024.

[2] Kausthub Kannan. *AI and Human Art Classification with MobileNetV2*. `https://www.kaggle.com/code/kausthubkannan/ai-human-art-classification-mobilenetv2-91#Imports`. Accessed: 2024-06-10. 2024.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[5] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[6] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[7] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

LIST OF FIGURES

LIST OF TABLES