AI-Generated Art Detection Project

University of Salerno - C.A.S.A 2023/2024

Prof. Bisogni - Prof. Narducci

Katia M. Perchet
k.perchet@studenti.unisa.it

Franco N. Merenda
f.merenda2@studenti.unisa.it

Abstract

In this **project**[1], we explore the application of deep learning techniques for image classification by making an image classifier that recognizes if a given Art Image is generated by an AI or not, leveraging various Convolutional Neural Network (CNN) architectures. The study is structured in two main phases: the initial phase involves training models on distinct art styles, while the subsequent phase focuses on processing combined art styles. For the first phase, we trained networks using CNN models from scratch, as well as fine-tuned pre-trained VGG16 and VGG19 models, on each individual art style. Each model's accuracy was evaluated to determine its effectiveness in recognizing if the images of a given art styles was AI or human generated. In the second phase, we aggregated all art styles into a single dataset and trained the model with more advanced architectures, including ResNet50 and InceptionV3, to assess their performance in a more complex classification task. Our results indicate significant variations in accuracy across different models and phases, providing insights into the strengths and limitations of each architecture in the context of art style classification. This comprehensive analysis offers valuable contributions to the field of computational art analysis, highlighting the potential and challenges of deep learning in recognizing and differentiating art styles.

1 Introduction

1.1 Background Information

AI-Generated Art is digital artwork created with the assistance of Artificial Intelligence (AI)- based algorithms and techniques.

This process involves algorithms that can learn from data inputs and generated new, original art pieces. The AI-art can include digital images, painting, sculptures, music and/or poetry.

The significance of the study and the goal of this project is to be able to recognize if the image we are seeing is AI-generated or Human-generated. This will allow us to:

- Identify and distinguish AI-generated artworks, helping to protect the copyrights of human artists.
- It can contribute to transparency and awareness in the area of digital art, so buyers and consumers will be aware if the art they are consuming was artificially generated or created by a human artist.
- It may help to understand artistic trends, the evolution of technology and the impact of AI on art and human creativity.

Artworks are generated by Artificial Intelligence applications that are powered by a subset of Machine Learning techniques, called Deep Learning. Deep Learning uses multilayered neural networks to simulate the complex decision-making power of the human brain. Some of the networks used by Deep Learning to generate art styles are:

- **Generative Adversarial Network (GANs)**: they consist of two neural networks, the generator and the discriminator, working together in a competitive process. The generator creates new images from random data, while the discriminator tries to distinguish

between real and generated images. Over time, the generator improves its ability to produce realistic images as it tries to fool the discriminator. GANs have been used to create images, paintings, and even works of art in specific styles.

- **Diffusion Models**: they progressively add noise to the data and learn how to reverse it to create new, similar data. Each one is composed by:
  - **Forward Diffusion**: in this stage, the model starts with an original piece of data and gradually adds random noise through a series of steps.
  - **Training**: the model learns how the noise added during the forward diffusion alters the data.
  - **Reverse diffusion**: once the model is trained, it takes the noisy data and tries to remove the noise to get back to the original data.
  - **Generating new data**: finally, the model can use what it learned in the reverse diffusion process to create new data.
- **Neural Style Transfer**: uses a pre-trained network to analyze visuals and employs additional measures to borrow the style from one image and apply it to another. It is focused on three main losses:
  - **Content loss**: is a measure of how much the content of the generated image differs from the content of the original content image.
  - **Style loss**: measures the style differences, e.g., patterns and textures in the generated image and the style image.
  - **Total loss**: NST combines the content loss and style loss into a single measure called the total loss.

As the optimization progresses, the generated image takes the content and style from different images. The end result is an appealing blend of the two, often bearing a striking resemblance to a piece of art.

Some of the popular AI image generators are:

- **DALL-E 2**: is built on an advanced architecture that employs a diffusion model, integrating data from CLIP (Contrastive Language-Image Pre-training)
- **Midjourney**: it is based on a diffusion model, similar to DALL-E and Stable Diffusion.
- **Stable Diffusion**: utilizes the Latent Diffusion Model (LDM) and OpenClip, a larger version of CLIP, to convert text into embeddings.

1.2 Datasets[2]

The dataset used to do the trials of the models in this project is "AI-Artbench: An AI-generated Artistic Dataset" which contains over a 180.000 art images.

- 60.000 of them are human-drawn art, which were taken from ArtBench-10 dataset
- the rest are 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.

1.3 Art Styles

Some of the art styles included in the dataset are: Art Nouveau, Baroque, Expressionism, Impressionism, Post impressionism, Realism, Renaissance, Romanticism, Surrealism, Ukiyo-e.

Before starting to process the dataset and use it to train our model, the art styles were ordered by complexity according to the team member's perspective, sorted by level of Abstraction of the Art Style:

1. **Realism** - [3]
2. **Renaissance** - [4]
3. **Romanticism** - [5]
4. **Baroque** - [6]
5. **Ukiyo-e** - [7]
6. **Art Nouveau** - [8]
7. **Post impressionism** - [9]
8. **Impressionism** - [10]
9. **Expressionism** - [11]
10. **Surrealism** - [12]

2 Execution Environment

This section outlines the detailed specifications of the hardware and software environment used to run all the models described in this paper.

To ensure the reproducibility of the experiments, the configurations listed below provide insights into the system setup which can significantly impact the performance and outcomes of the model training and inference processes.

2.1 Hardware Specifications

- **Processor**: AMD Ryzen 5 5600X 6-Core Processor
- **GPU**: NVIDIA GeForce RTX 3070 Ti
- **RAM**: 32 GB DDR4
- **Storage**: 1 TB SSD
- **Operating System**: Ubuntu 22.04.4 LTS - Linux 6.5.0-35-generic

2.2 Software Libraries and Versions

- **Python**: 3.10.12
- **TensorFlow**: 2.16.1
- **Keras**: 3.3.3
- **NumPy**: 1.26.4
- **Pandas**: 2.2.2
- **scikit-learn**: 1.4.2
- **Matplotlib**: 3.8.4
- **Seaborn**: 0.13.2

2.3 Additional Tools

- **NVIDIA Driver**: 550.54.15
- **CUDA**: 12.4
- **cuDNN**: 8.9.6

3 Data Pre-Processing

3.1 Data division and labeling

To prepare the data for model training it is first downloaded from AI-ArtBench Kaggle's repository, trying to have the most updated version of the dataset in each art style and model. The directories were organized by art style, which each one of them had a folder identifying it. Each art style folder was divided internally into the training and validation images. Each **"train"** and **"valid"** folder was also subdivided into **"AI_Generated_Art_Style"** and **"art_style"** folders.

After this separation, each directory was iterated to label all the files, splitting an classifying each image in **"AI"** or **"human"** label.

3.2 Data Augmentation

As an additional pre-processing step, each dataset used either for training or validation, were modified by a data augmentation technique, where each dataset had a re-scaling process, applied rotations and flipping of the images to increment the dataset diversity, allowing the model to have more data for a better performance.

For both training and validation datasets the data augmentation used was:

- rescale=1./255: Normalize the image pixel values to the range [0, 1] by scaling by 1/255
- rotation_range=7: Randomly rotate images by up to 7 degrees
- horizontal_flip=True: Randomly flip images horizontally

4 Deep Learning Models

This section will be meant to describe the deep learning models selected for the study and why they were chosen.

4.1 Models overview

4.1.1 Convolutional Neural Network[13]

As first type of network CNN was chosen for image-classification tasks because they are efficient when it comes to handle the complexity and high dimensionality of images.

When applying CNNs to classify human-generated and AI-generated art styles some of the main characteristics used were:

- **Feature Extraction**: CNNs can automatically learn and extract relevant features from the artworks, such as brush strokes, color patterns, and textures, which are crucial for distinguishing between different art styles.
- **Handling Variations**: Art styles can vary greatly in terms of color, texture, and composition. CNNs' ability to learn hierarchical

features makes them well-suited to capture these variations.

4.1.2 VGG16[14]

VGG16 was chosen to train our model because it has an strong performance, availability of pre-train models for transfer learning, ability to handle complex image data and extensive community support. When applying VGG16 to classify human-generated and AI-generated art styles some of the main characteristics used were:

- **Detailed Feature Learning**: VGG16 can learn detailed features from artworks, such as brush strokes, color gradients, and intricate patterns, which are essential for distinguishing between different art styles.
- **Efficient Training**: using a pre-trained VGG16 model on a large dataset like ImageNet, and fine-tuning it on specific art dataset allows to speed up the training and improve the performance of the model, especially when the training data is limited or not abundant.
- **Robust Classification**: The depth and design of VGG16 allow it to perform robust classification, capturing the nuances between human-generated and AI-generated art styles effectively.

4.1.3 VGG19[15]

Although VGG16 was already implemented, VGG19 was chosen for its deeper architecture, proven performance, and potential to learn more detailed features. Implementing VGG19 alongside VGG16 allows performance comparison, improved generalization, and robust model evaluation, helping to determine the most effective model for each art style.

When applying VGG19 to classify human-generated and AI-generated art styles some of the characteristics used were:

- **Enhanced Feature Extraction**: VGG19's deeper architecture might capture more detailed and nuanced features of the artworks, which can be crucial for accurate classification.
- **Improved Performance**: In some cases, VGG19 might outperform VGG16 due to its increased capacity to learn from data, especially if the dataset is rich and varied.

4.1.4 ResNet50[16]

ResNet50's deep architecture, with 50 layers, excels at identifying intricate patterns and details, crucial for distinguishing between human-made and AI-generated art. Its use of residual connections helps it learn effectively, even from smaller datasets, by preventing common training issues. Due to a pre-trained on the extensive ImageNet dataset, leads to ResNet50 have already understanding of many visual features, which speeds up training and improves accuracy.

4.1.5 InceptionV3[17]

InceptionV3 is highly accurate and efficient due to its unique architecture, which includes multiple filter sizes at each layer, allowing it to capture a wide range of features and details in art images. This makes it particularly good at distinguishing subtle differences between human-made and AI-generated art. It is also pre-trained on a large dataset, so it already understands many visual patterns, reducing the amount of data and training time needed for your specific task. Additionally, is designed to be computationally efficient, meaning it can deliver high performance without requiring excessive computational resources, making it an effective and practical choice for AI art detection.

4.2 Model's Architectures

4.2.1 Convolutional Neural Network

The created CNN is composed by a an input layer, 3 convolutional layers, 3 max pooling layer, a flatten layer and 3 fully connected layers. This architecture is designed to progressively extract more complex features from the input images, while reducing the dimensionality and learning to classify the input into one of the two classes. The network processes the input image through several convolutional layers to extract features. Each convolutional layer is followed by a pooling layer to reduce the size and complexity of the data. After the convolutional and pooling layers, the data is flattened into a 1D vector. The flattened data passes through a series of fully connected layers to learn high-level representations. The final layer produces probabilities for each class, allowing the network to make a classification decision.

1. **Input Layer**
    - **Input Shape**: The network expects images of size 32x32 pixels with 3

color channels (typically red, green, and blue).

2. **First Convolutional Layer**
    - **Conv2D Layer**: This layer applies 512 filters (also known as kernels) of size 3x3 to the input image.
    - Activation Function: It uses the ReLU (Rectified Linear Unit) activation function, which helps the network learn complex patterns by introducing non-linearity.
    - **Regularization**: L2 regularization (with a small value of 0.001) is applied to prevent the model from overfitting, which means it helps the model generalize better to new data.
    - **Output**: This layer transforms the input image into 512 feature maps.
3. **First MaxPooling Layer**
    - **MaxPooling2D Layer**: This layer downsamples the input by taking the maximum value over a 2x2 window. It reduces the size of the feature maps, which helps in reducing the computational complexity and extracts dominant features.
    - **Output**: The feature maps are reduced in size by half.
4. **Second Convolutional Layer**
    - **Conv2D Layer**: This layer applies 128 filters of size 3x3 to the output of the previous layer.
    - **Activation Function**: ReLU is used again to introduce non-linearity.
    - **Output**: The transformed feature maps now have 128 channels.
5. **Second MaxPooling Layer**
    - **MaxPooling2D Layer**: Another downsampling step, again reducing the size of the feature maps by half using a 2x2 window.
    - **Output**: The feature maps are further reduced in size.
6. **Third Convolutional Layer**
    - **Conv2D Layer**: This layer applies 32 filters of size 3x3.
    - **Activation Function**: ReLU is used to introduce non-linearity.
    - **Output**: The transformed feature maps now have 32 channels.
7. **Third MaxPooling Layer**
    - **MaxPooling2D Layer**: Another downsampling step, reducing the size of the feature maps using a 2x2 window.
    - **Output**: The feature maps are further reduced in size.
8. **Flatten Layer**
    - **Flatten Layer**: This layer converts the 3D feature maps into a 1D vector, preparing it for the fully connected (dense) layers.
9. **First Dense Layer**
    - **Dense Layer**: This fully connected layer has 32 units (neurons) and uses the ReLU activation function.
    - **Output**: A 1D vector of size 32.
10. **Second Dense Layer**
    - **Dense Layer**: Another fully connected layer with 16 units and ReLU activation.
    - **Output**: A 1D vector of size 16.
11. **Output Layer**
    - **Dense Layer**: The final fully connected layer with 2 units, corresponding to the 2 classes the network aims to classify.
    - **Activation Function**: Sigmoid activation is used here, which converts the output to a value between 0 and 1, suitable for binary classification.
    - **Output**: A 1D vector of size 2, representing the probabilities for each class.

For the neural network described, when the model was created, it ended up with a total of 645.874 parameters, from which all of them were classified as trainable parameters.

4.2.2 VGG16 and VGG19

Both VGGNet models were implemented using the same architecture, with the difference of calling the respective network when loading the pre-trained model. The VGG16 and VGG19 models

created have a pre-trainedlarge dataset called "ImageNet", which contains millions of images across many categories to speed up the performance of the training, taking advantage of the rich feature representation it has learned from a large and diverse dataset.

VGG19 was also selected to train the model because provides extra layers, having a deeper network, which can lead to a better feature extraction capabilities. This can be useful for tasks that require fine-grained details in images, such as distinguishing the subtle differences in art styles.

1. **Load of Pre-Trained VGGNet Model**

   network= VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3)) Or network= VGG19(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

   - **VGGNet Model**: We start with the VGGNet model, pre-trained on the ImageNet dataset.
   - **Weights**: The model uses pre-learned weights from ImageNet.
   - **include_top=False**: This excludes the final classification layer of VGGNet, which is specific to ImageNet's 1000 classes. We will add our own layers for our specific task.
   - **Input Shape**: The model expects input images of size 224x224 pixels with 3 color channels (RGB).

2. **Freezing Layers**

   - This means that the weights of the VGGNet layers will not be updated during training. We do this because we want to use the features learned by VGGNet as they are, without modifying them, making the training faster and requiring less data.

3. **Added Custom Layers on top of the VGGNet Model.**

   Our own layers were made to adapt the model to our specific classification problem.

   - **Flatten Layer**: This layer takes the output of VGGNet and flattens it into a single long vector. It prepares the data for the fully connected layers.
   - **Dense Layer**: A fully connected layer with 256 units and ReLU activation. This layer learns to combine the features extracted by VGGNet in meaningful ways.
   - **Dropout Layer**: This layer randomly drops 50% of the connections during training to prevent overfitting and help the model generalize better.
   - **Prediction Layer**: The final layer with 1 unit and sigmoid activation. It outputs a value between 0 and 1, suitable for binary classification (e.g., distinguishing between two types of art styles).

For the **VGG16** described, when the model was created, it ended up with a total of 14.714.688 parameters, from which all of them were classified as non-trainable parameters. For the **VGG19** described, when the model was created, it ended up with a total of 20.024.384 parameters, from which all of them were classified as non-trainable parameters.

4.3 Techniques to Prevent Overfitting

4.3.1 Regularization

By using regularizers.l2(0.001) in the CNN as kernel regularizer it helps to try to achieve a robust model by having a balance between underfitting and overfitting. It works by adding a penalty to the model's loss function to discourage overly complex models. The value 0.001 controls the strength of the regularization.

4.3.2 Dropout

By adding a Droupout of 0.5 in the VGGNets, RestNet50 and InceptionV3 it randomly drops 50% of the neurons during training, forcing the model to learn more robust features.

4.3.3 Early Stopping

To avoid overfitting during training, the early stopping technique stops the training of the model when it starts to perform worse on a validation set or when it does not see an improvement on the highest validation accuracy, having a patience of 5 epochs in the project.

5 First Phase - Experiments and Results

5.1 Metrics to Measure Models

To compare the performance of the different deep learning models, several key metrics were consid-

ered, helping to evaluate how well the models are performing and where they might be falling short.

1. **Accuracy**: proportion of correctly classified instances amount the total instances.

2. **Precision**: proportion of true positive predictions among all positive predictions.

3. **Recall**: proportion of true positive predictions among all actual positive instances.

4. **F1 Score**: harmonic mean of precision and recall.

5. **Confusion Matrix**: used to describe the performance of a classification model, showing the true positives, true negatives, false positives and false negatives.

6. **Logarithmic Loss**: measures the performance of a classification model where the prediction is a probability value between 0 and 1.

5.2 Experiments

For each art style, as classified based on our criteria, the different models were trained to assess their accuracy and loss, in order to determine which model is best suited for each art style.

5.2.1 Realism

- **CNN**: Since the "early stopping" was applied to the model, the network trained for a total of 22 epochs.
  - **Training**:
    - Top Accuracy: 0.9478
    - Lowest Loss: 0.1489
  - **Validation**:
    - Top Accuracy: 0.9317
    - Lowest Loss: 0.1764
  - **Overall Accuracy:** 0.9290
  - **Overall Loss:** 0.1875
  - **F1 Score**: 0.8854
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

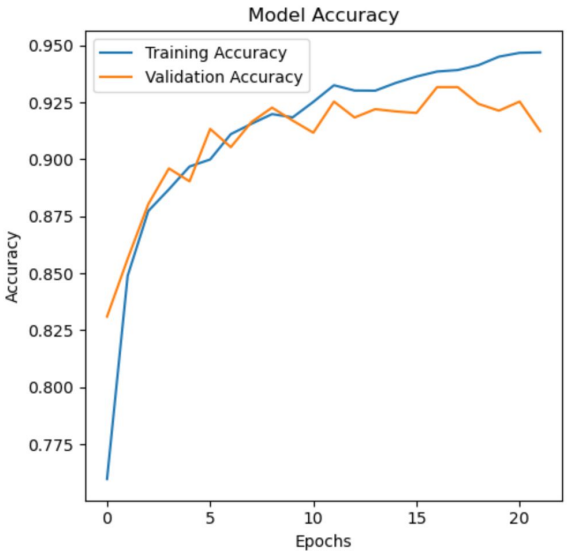

Figure 1: Accuracy-CNN for Realism

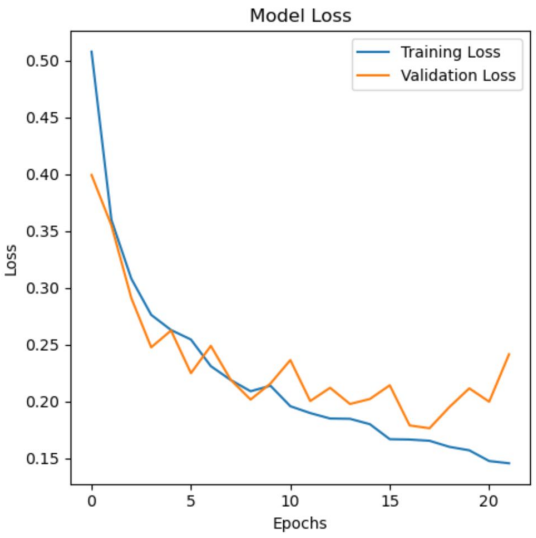

Figure 2: Loss-CNN for Realism

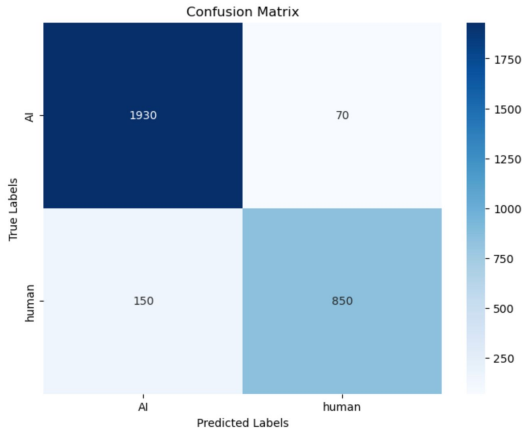

Figure 3: Confusion Matrix for Realism CNN

- **VGG16**: Since the "early stopping" was applied to the model, the network trained for a total of 18 epochs.

  - **Training**:
    - Top Accuracy: 0.9709
    - Lowest Loss: 0.0734
  - **Validation**:
    - Top Accuracy: 0.9640
    - Lowest Loss: 0.1061
  - **Overall Accuracy:** 0.9610
  - **Overall Loss:** 0.1110
  - **F1 Score**: 0.9351
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

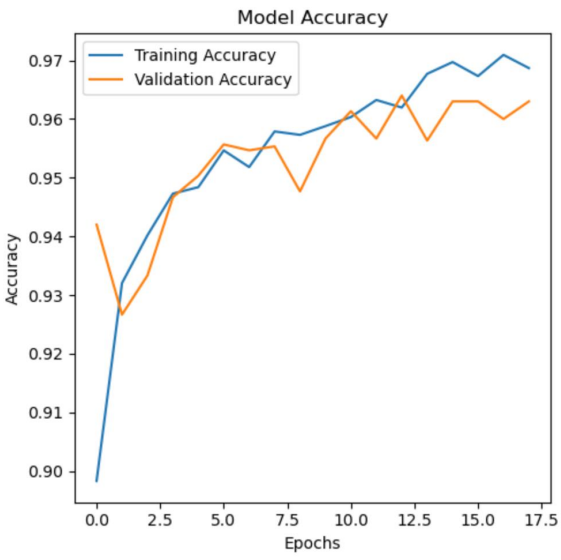

Figure 4: Accuracy-VGG16 for Realism

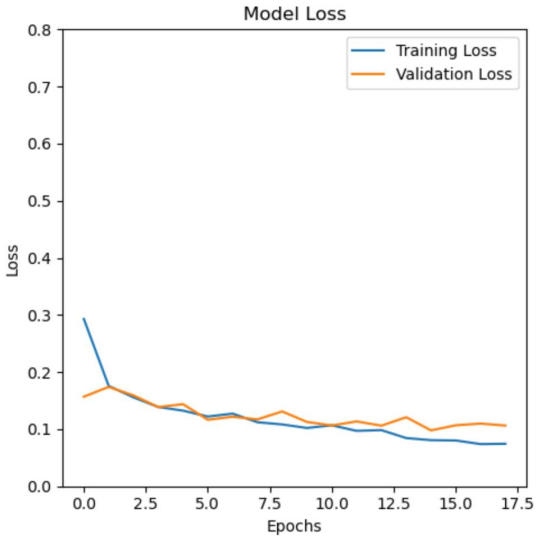

Figure 5: Loss-VGG16 for Realism

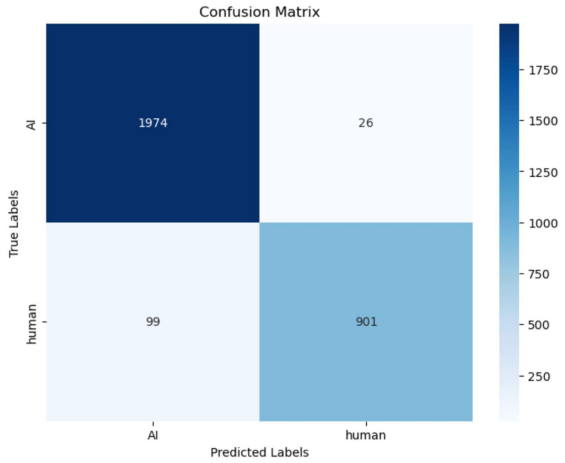

Figure 6: Confusion Matrix for Realism VGG16

- **VGG19**: Since the "early stopping" was applied to the model, the network trained for a total of 13 epochs.
  - **Training**:
    - Top Accuracy: 0.9481
    - Lowest Loss: 0.1350
  - **Validation**:
    - Top Accuracy: 0.9463
    - Lowest Loss: 0.1365
  - **Overall Accuracy:** 0.9459
  - **Overall Loss:** 0.1398
  - **F1 Score**: 0.9151
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

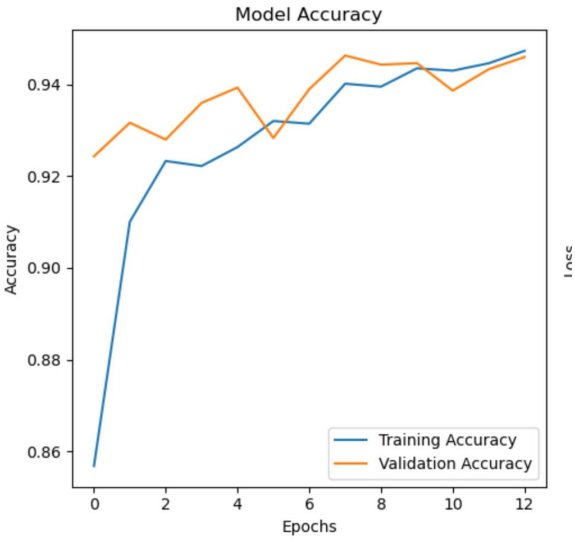

Figure 7: Accuracy-VGG19 for Realism

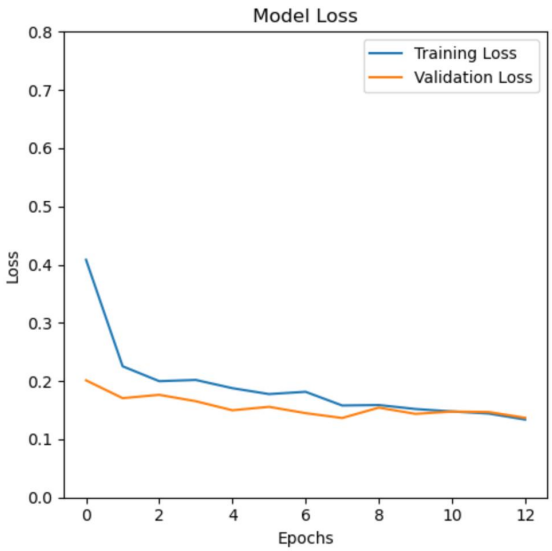

Figure 8: Loss-VGG19 for Realism

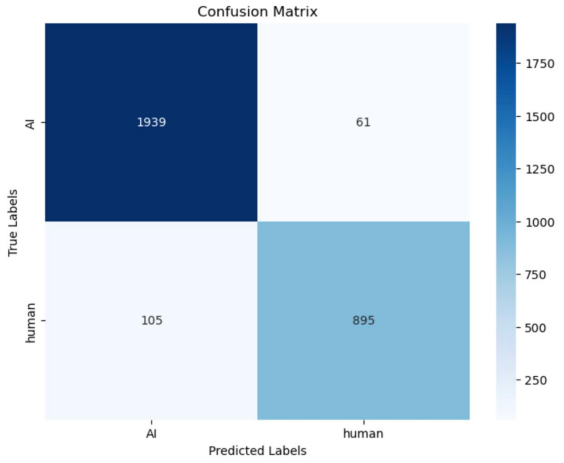

Figure 9: Confusion Matrix for Realism VGG19

5.2.2 Renaissance

- **CNN**: Since the "early stopping" was applied to the model, the network trained for a total of 24 epochs.

  - **Training**:
    - Top Accuracy: 0.9674
    - Lowest Loss: 0.1004
  - **Validation**:
    - Top Accuracy: 0.9490
    - Lowest Loss: 0.1510
  - **Overall Accuracy:** 0.9526
  - **Overall Loss:** 0.1445
  - **F1 Score**: 0.9191
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

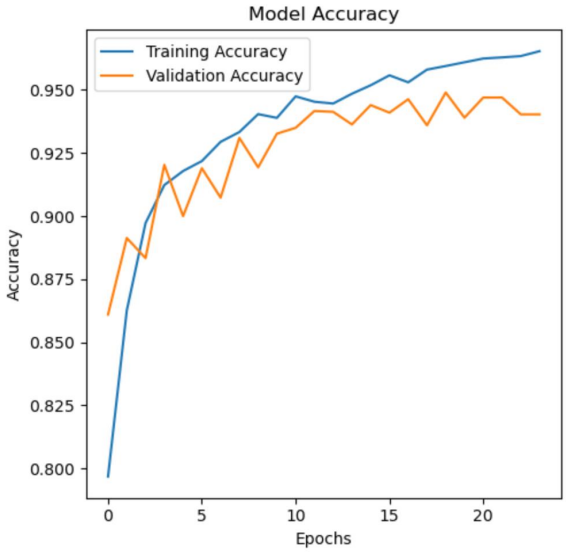

Figure 10: Accuracy-CNN for Renaissance

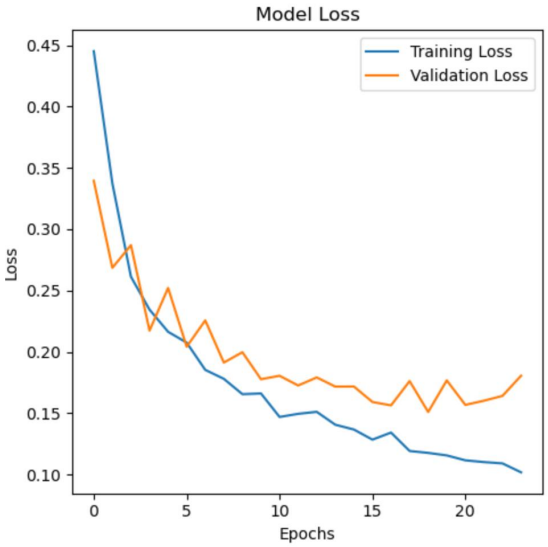

Figure 11: Loss-CNN for Renaissance

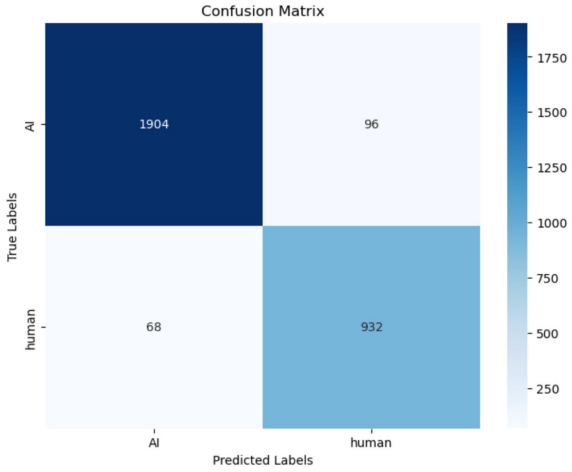

Figure 12: Confusion Matrix for Renaissance CNN

- **VGG16**: Since the ”early stopping” was applied to the model, the network trained for a total of 13 epochs.

  - **Training**:
    - Top Accuracy: 0.9757
    - Lowest Loss: 0.0634
  - **Validation**:
    - Top Accuracy: 0.9730
    - Lowest Loss: 0.0845
  - **Overall Accuracy:** 0.9700
  - **Overall Loss:** 0.0802
  - **F1 Score**: 0.9587
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

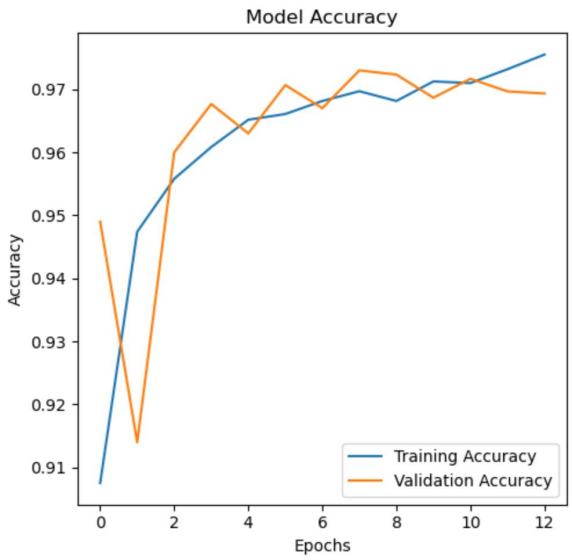

Figure 13: Accuracy-VGG16 for Renaissance

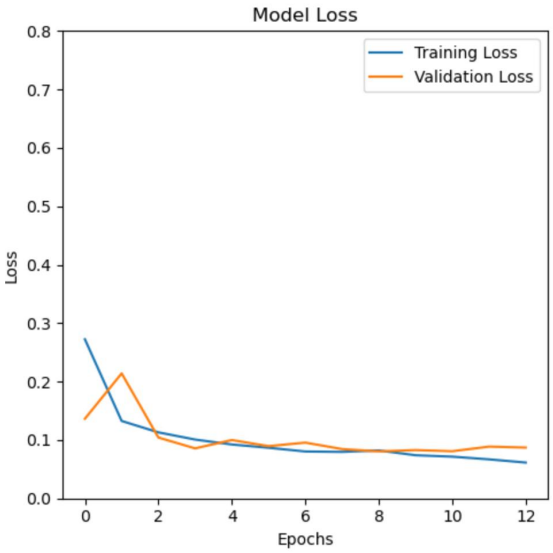

Figure 14: Loss-VGG16 for Renaissance

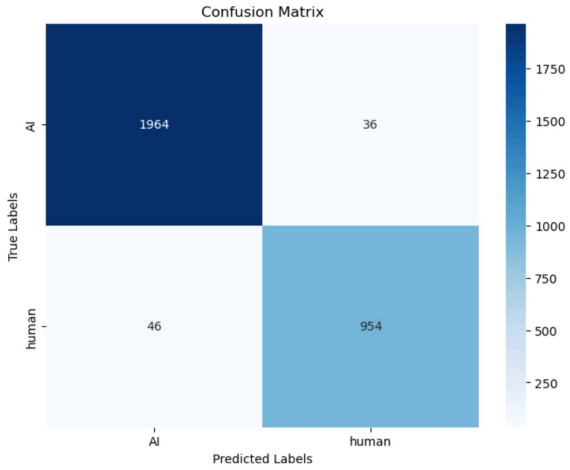

Figure 15: Confusion Matrix for Renaissance VGG16

- **VGG19**: Since the ”early stopping” was applied to the model, the network trained for a total of 17 epochs.

  - **Training**:
    - Top Accuracy: 0.9507
    - Lowest Loss: 0.1410
  - **Validation**:
    - Top Accuracy: 0.9509
    - Lowest Loss: 0.1223
  - **Overall Accuracy:** 0.9473
  - **Overall Loss:** 0.1404
  - **F1 Score**: 0.9184
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

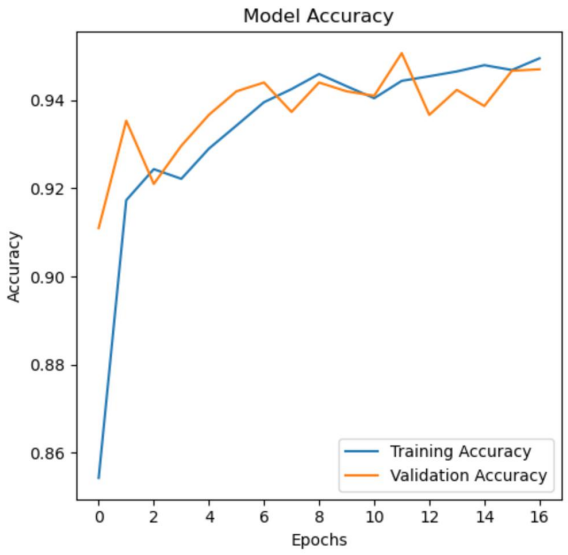

Figure 16: Accuracy-VGG19 for Renaissance

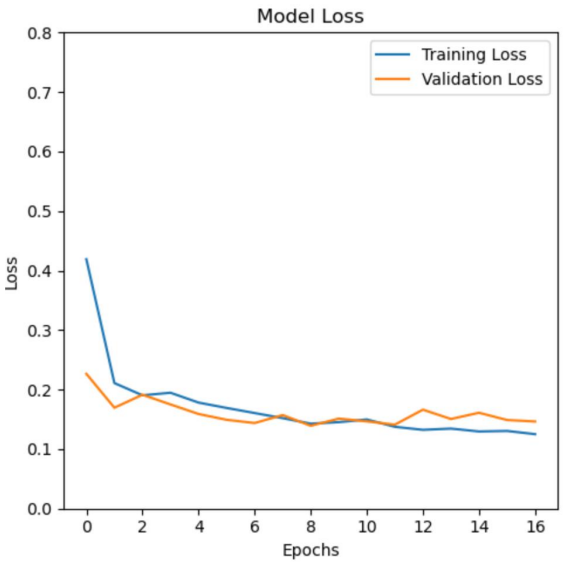

Figure 17: Loss-VGG19 for Renaissance

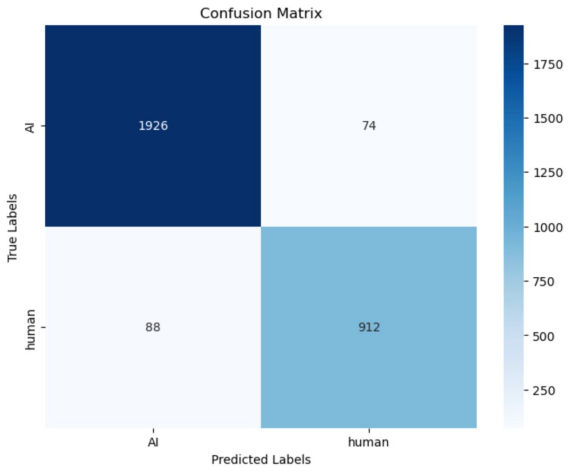

Figure 18: Confusion Matrix for Renaissance VGG19

5.2.3 Romanticism

- **CNN**: Since the ”early stopping” was applied to the model, the network trained for a total of 13 epochs.

  – **Training**:

    * Top Accuracy: 0.9604
    * Lowest Loss: 0.1080

  – **Validation**:

    * Top Accuracy: 0.9620
    * Lowest Loss: 0.1025

  – **Overall Accuracy:** 0.9576

  – **Overall Loss:** 0.1107

  – **F1 Score**: 0.9390

  – **Graphs of Accuracy, Loss. and Confusion Matrix**:

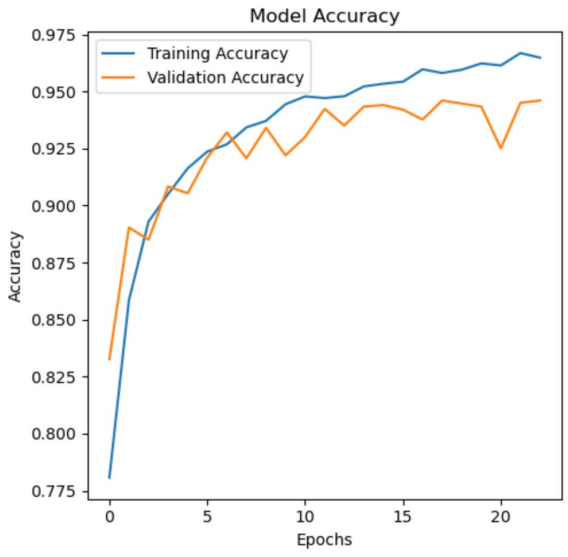

Figure 19: Accuracy-CNN for Romanticism

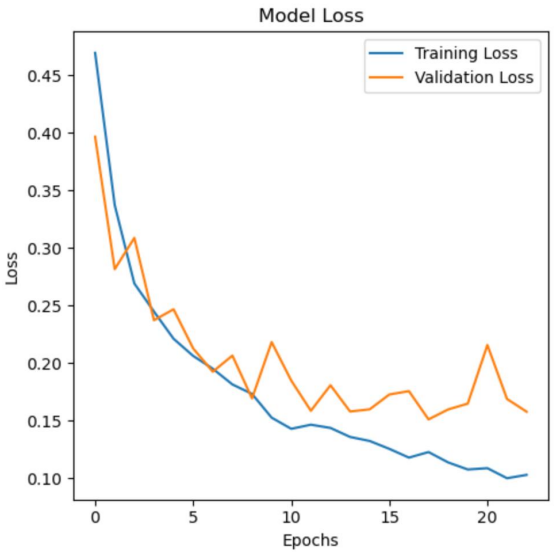

Figure 20: Loss-CNN for Romanticism

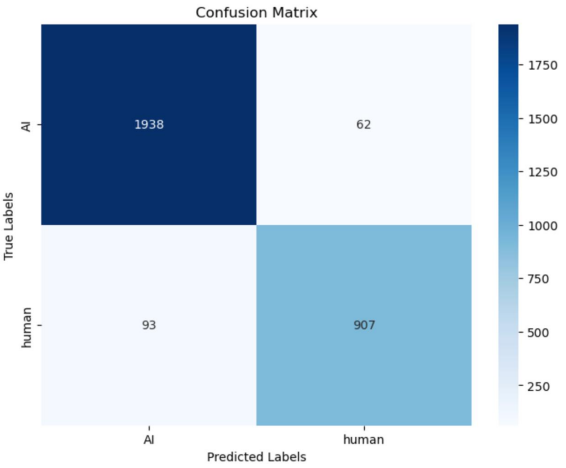

Figure 21: Confusion Matrix for Romanticism CNN

- **VGG16**: Since the ”early stopping” was applied to the model, the network trained for a total of 9 epochs.
  - **Training**:
    - Top Accuracy: 0.9618
    - Lowest Loss: 0.1043
  - **Validation**:
    - Top Accuracy: 0.9577
    - Lowest Loss: 0.1218
  - **Overall Accuracy:** 0.9549
  - **Overall Loss:** 0.1229
  - **F1 Score**:
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

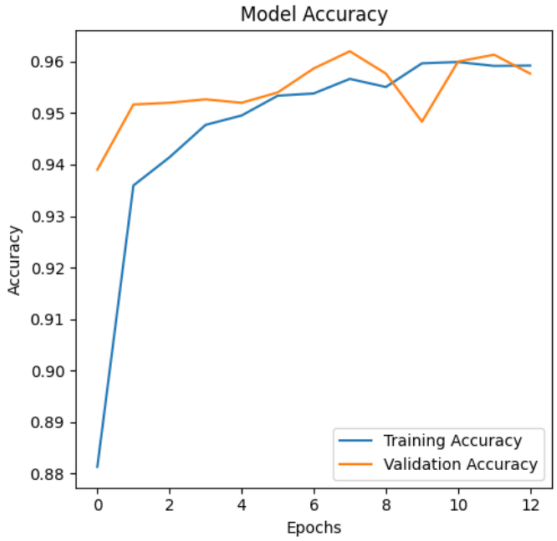

Figure 22: Accuracy-VGG16 for Romanticism

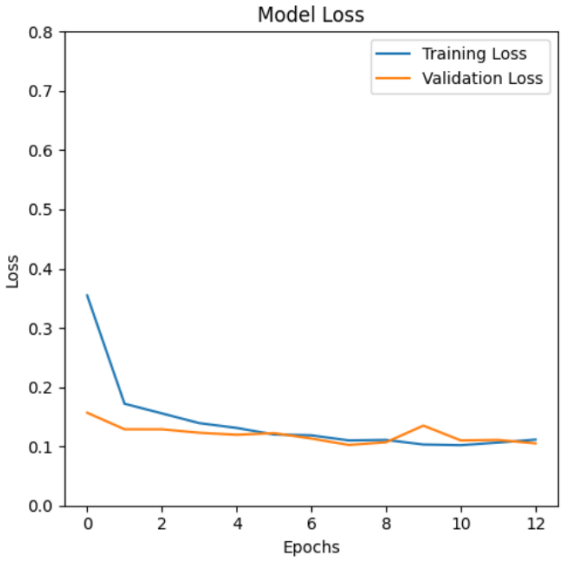

Figure 23: Loss-VGG16 for Romanticism

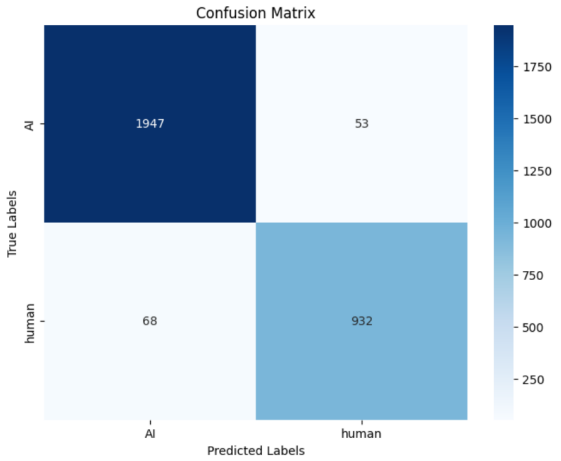

Figure 24: Confusion Matrix for Romanticism VGG16

- **VGG19**: Since the ”early stopping” was applied to the model, the network trained for a total of 12 epochs.

  - **Training**:
    - Top Accuracy: 0.9384
    - Lowest Loss: 0.1521
  - **Validation**:
    - Top Accuracy: 0.9483
    - Lowest Loss: 0.1381
  - **Overall Accuracy:** 0.9496
  - **Overall Loss:** 0.1335
  - **F1 Score**: 0.9189
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

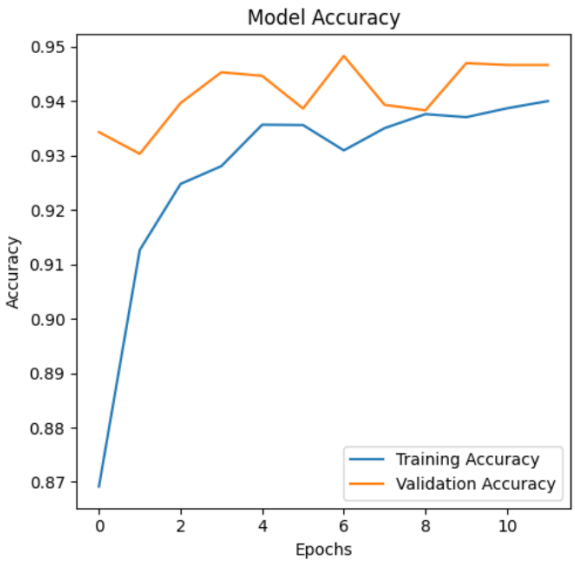

Figure 25: Accuracy-VGG19 for Romanticism VGG19

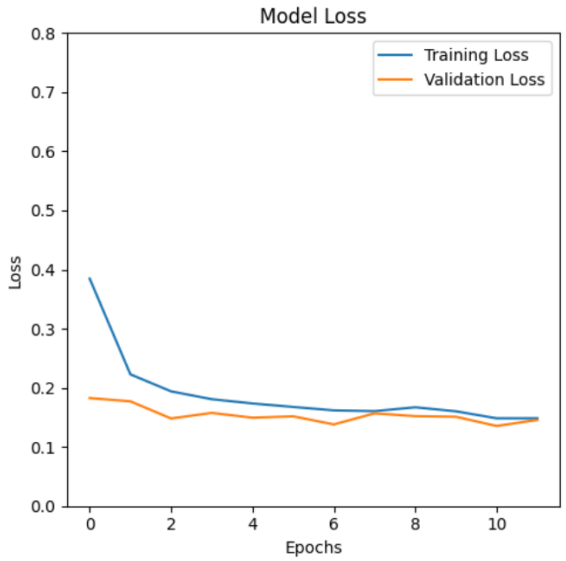

Figure 26: Loss-VGG19 for Romanticism VGG19

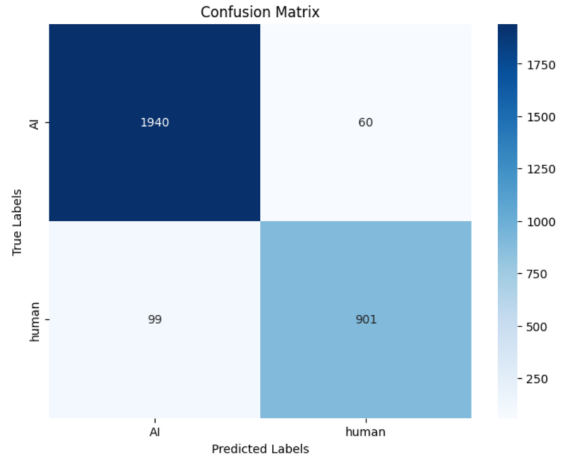

Figure 27: Confusion Matrix for Romanticism VGG19

5.2.4 Baroque

- **CNN**: The network trained for the total of 30 epochs.

  - **Training**:
    - Top Accuracy: 0.9858
    - Lowest Loss: 0.0499
  - **Validation**:
    - Top Accuracy: 0.9683
    - Lowest Loss: 0.1056
  - **Overall Accuracy:** 0.9673
  - **Overall Loss:** 0.1010
  - **F1 Score**: 0.9484
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

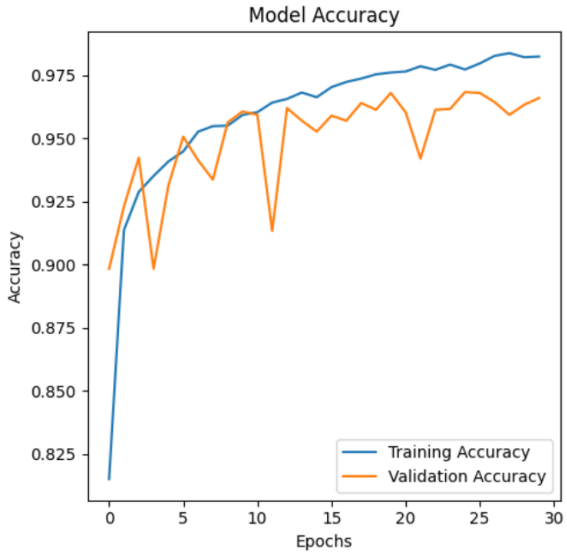

Figure 28: Accuracy-CNN for Baroque

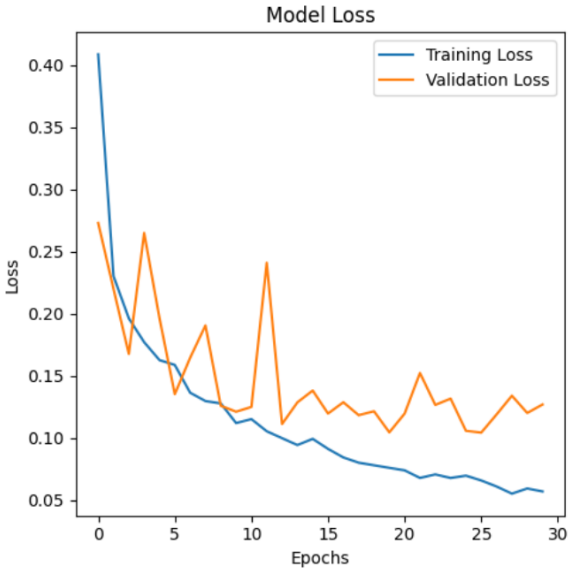

Figure 29: Loss-CNN for Baroque

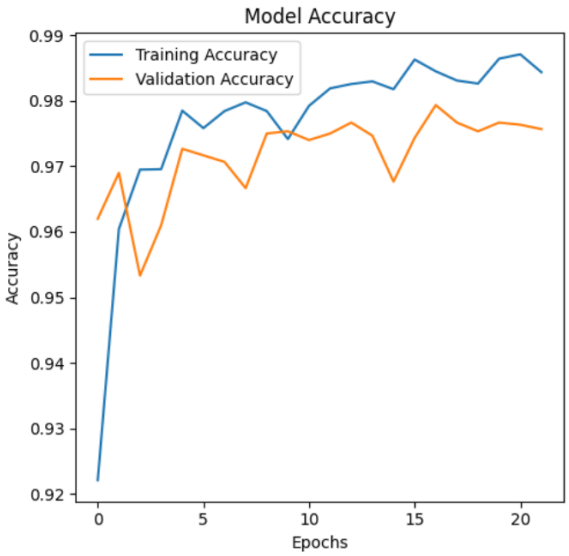

Figure 31: Accuracy-VGG16 for Baroque

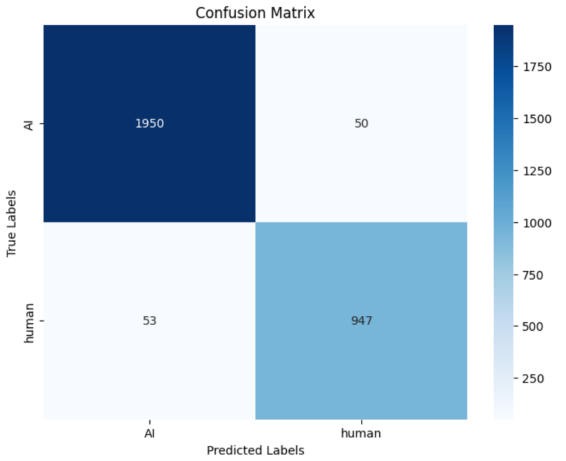

Figure 30: Confusion Matrix for Baroque CNN

- **VGG16**: Since the ”early stopping” was applied to the model, the network trained for a total of 22 epochs.
  - **Training**:
    - Top Accuracy: 0.9894
    - Lowest Loss: 0.0284
  - **Validation**:
    - Top Accuracy: 0.9793
    - Lowest Loss: 0.0754
  - **Overall Accuracy:** 0.9760
  - **Overall Loss:** 0.0770
  - **F1 Score**: 0.9678
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

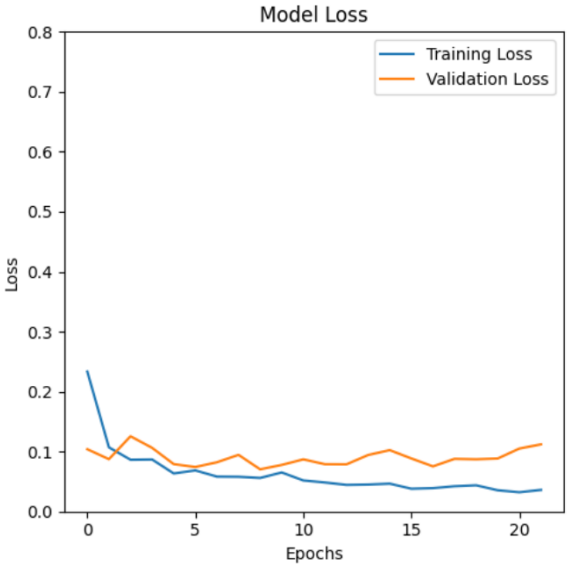

Figure 32: Loss-VGG16 for Baroque

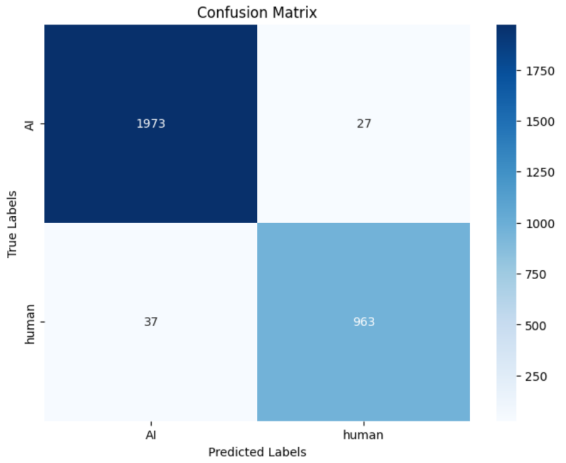

Figure 33: Confusion Matrix for Baroque VGG16

- **VGG19**: Since the ”early stopping” was applied to the model, the network trained for a total of 12 epochs.

  - **Training**:
    - Top Accuracy: 0.9382
    - Lowest Loss: 0.1486
  - **Validation**:
    - Top Accuracy: 0.9487
    - Lowest Loss: 0.1337
  - **Overall Accuracy:** 0.9459
  - **Overall Loss:** 0.1426
  - **F1 Score**: 0.9154
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

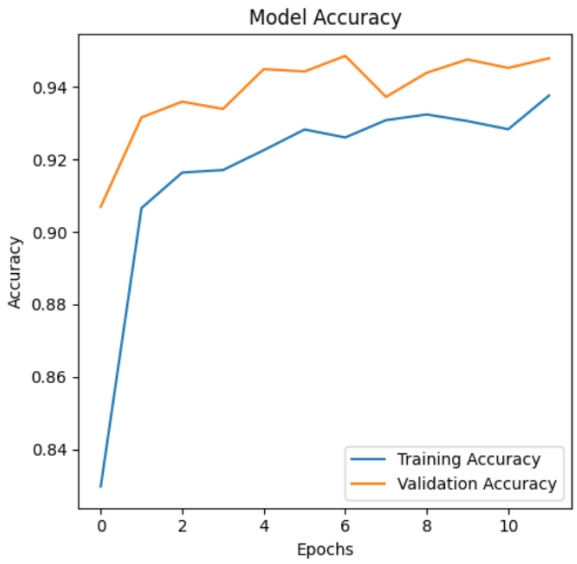

Figure 34: Accuracy-VGG19 for Baroque

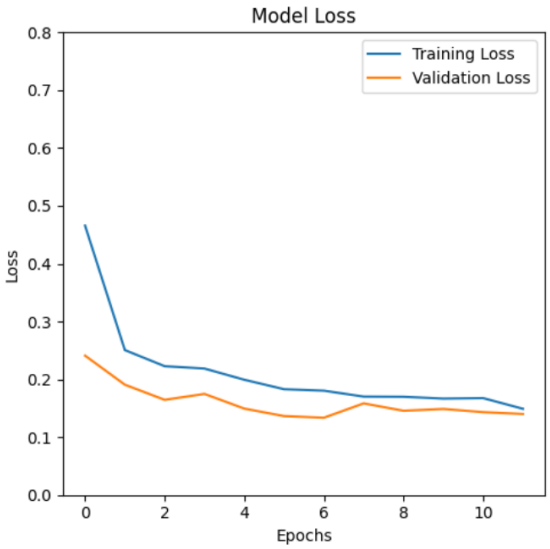

Figure 35: Loss-VGG19 for Baroque

Figure 37: Accuracy-CNN for Ukiyo-E

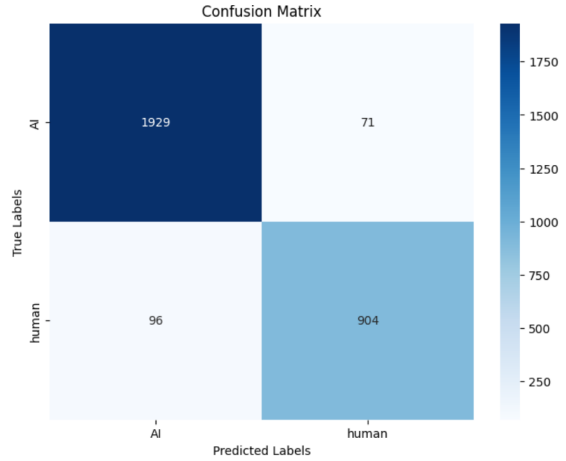

Figure 36: Confusion Matrix for Baroque VGG19

5.2.5 Ukiyo-e

- **CNN**: Since the "early stopping" was applied to the model, the network trained for a total of 26 epochs.
  - **Training**:
    - Top Accuracy: 0.9825
    - Lowest Loss: 0.0561
  - **Validation**:
    - Top Accuracy: 0.9807
    - Lowest Loss: 0.0602
  - **Overall Accuracy:** 0.9786
  - **Overall Loss:** 0.0657
  - **F1 Score**: 0.9728
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

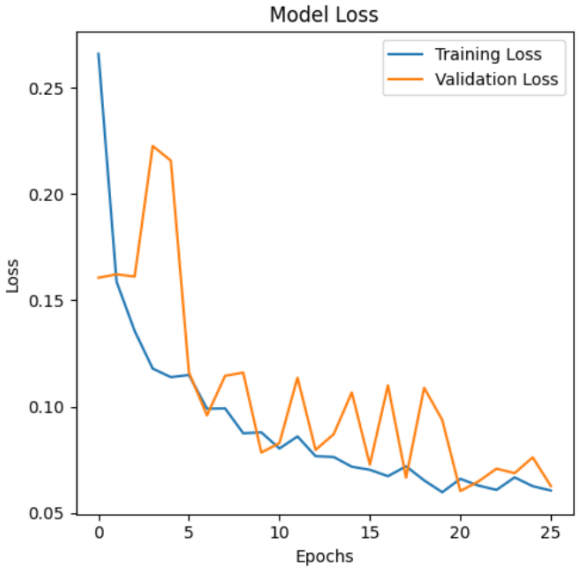

Figure 38: Loss-CNN for Ukiyo-E

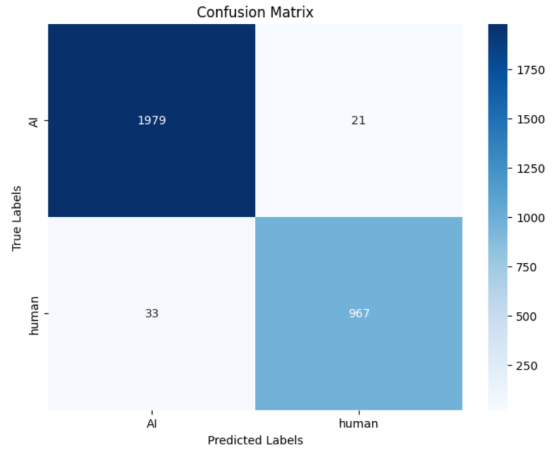

Figure 39: Confusion Matrix for Ukiyo-E CNN

- **VGG16**: Since the "early stopping" was applied to the model, the network trained for a total of 16 epochs.
  - **Training**:
    - Top Accuracy: 0.9961
    - Lowest Loss: 0.0129
  - **Validation**:
    - Top Accuracy: 0.9957
    - Lowest Loss: 0.0128
  - **Overall Accuracy:** 0.9946
  - **Overall Loss:** 0.01837
  - **F1 Score**: 0.9915
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

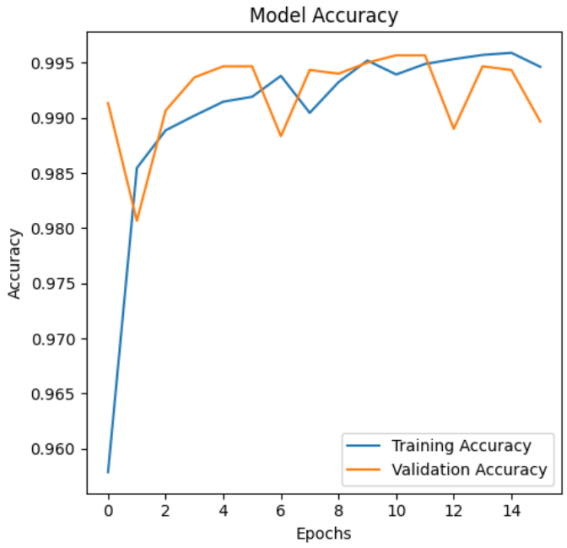

Figure 40: Accuracy-VGG16 for Ukiyo-E

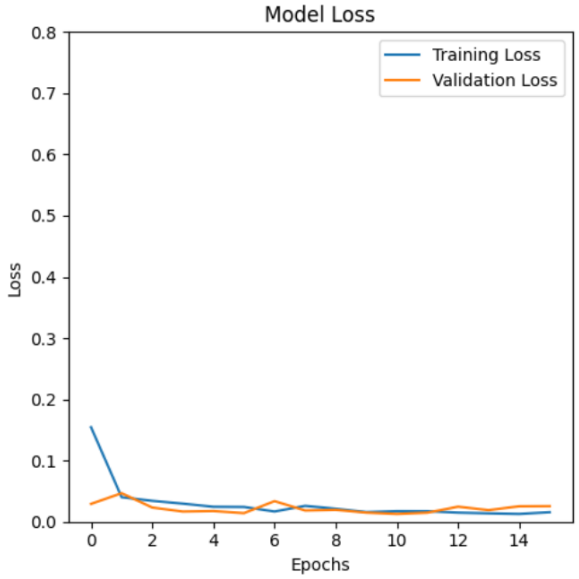

Figure 41: Loss-VGG16 for Ukiyo-E

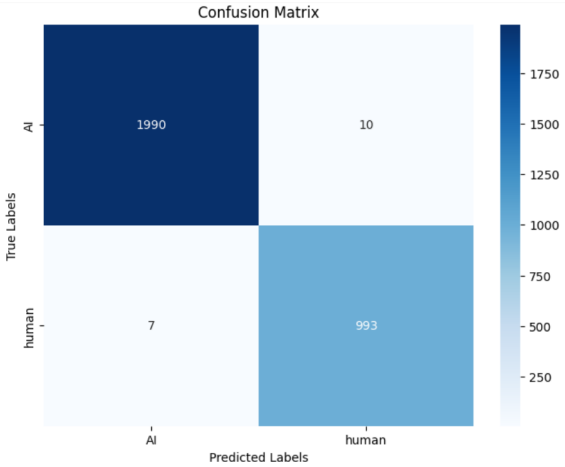

Figure 42: Confusion Matrix for Ukiyo-E

- **VGG19**: Since the ”early stopping” was applied to the model, the network trained for a total of 22 epochs.

  - **Training**:
    - Top Accuracy: 0.9932
    - Lowest Loss: 0.0246
  - **Validation**:
    - Top Accuracy: 0.9937
    - Lowest Loss: 0.0348
  - **Overall Accuracy:** 0.9923
  - **Overall Loss:** 0.0268
  - **F1 Score**: 0.9864
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

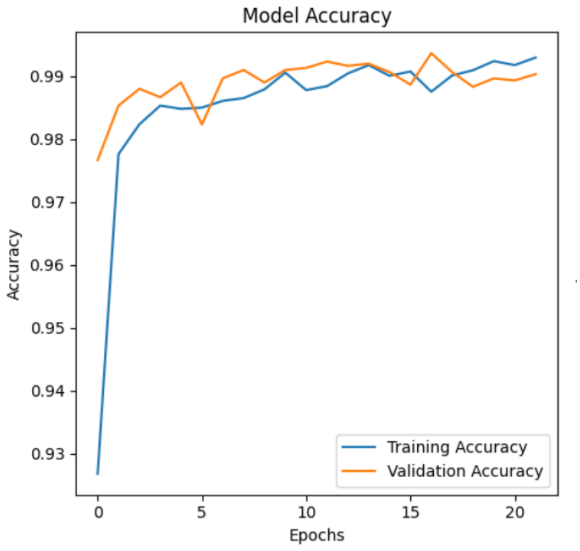

Figure 43: Accuracy-VGG19 for Ukiyo-E

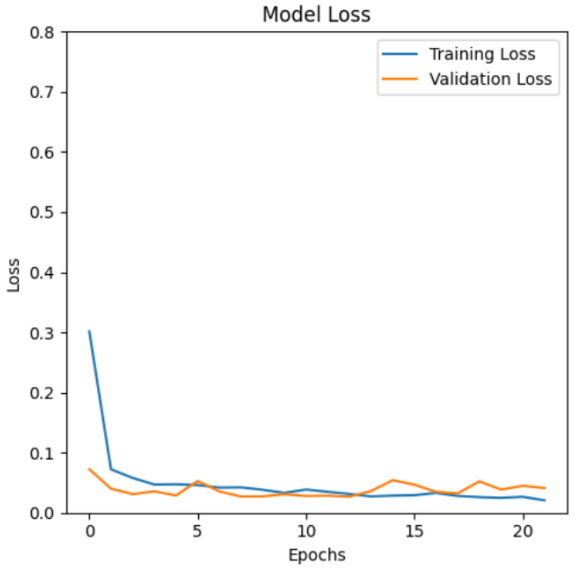

Figure 44: Loss-VGG19 for Ukiyo-E

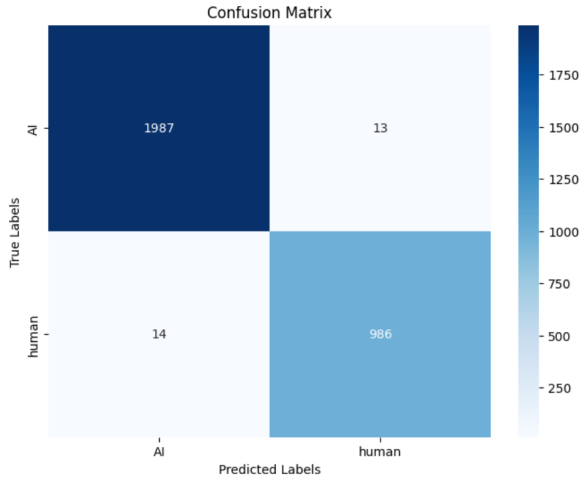

Figure 45: Confusion Matrix for VGG19 Ukiyo-E

5.2.6 Art Nouveau

- **CNN**: Since the ”early stopping” was applied to the model, the network trained for a total of 14 epochs.
  - **Training**:
    - Top Accuracy: 0.9773
    - Lowest Loss: 0.0712
  - **Validation**:
    - Top Accuracy: 0.9730
    - Lowest Loss: 0.0770
  - **Overall Accuracy:** 0.9753
  - **Overall Loss:** 0.0795
  - **F1 Score**: 0.9566

– **Graphs of Accuracy, Loss. and Confusion Matrix**:

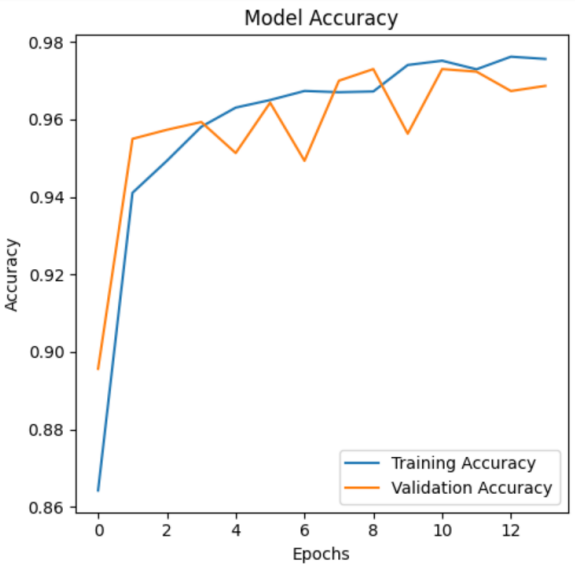

Figure 46: Accuracy-CNN for Art Nouveau

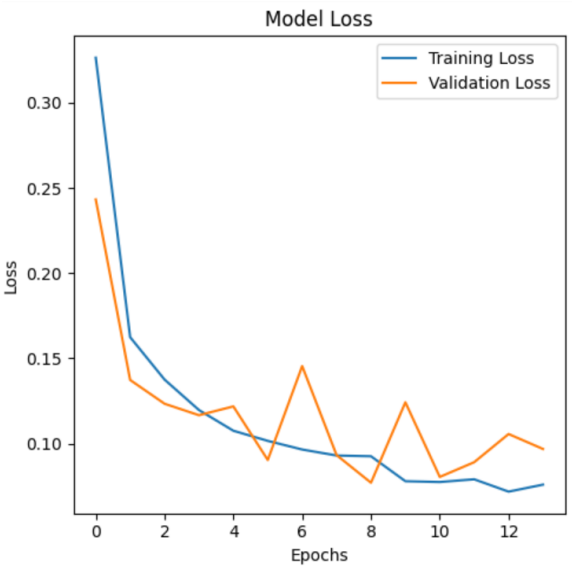

Figure 47: Loss-CNN for Art Nouveau

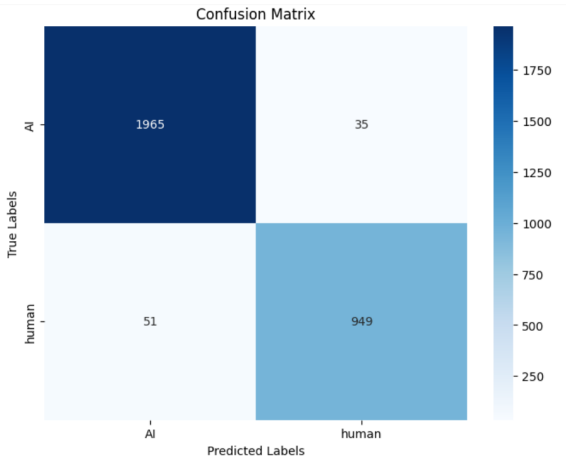

Figure 48: Confusion Matrix for Art Nouveau CNN

- **VGG16**: Since the ”early stopping” was applied to the model, the network trained for a total of 9 epochs.
  - **Training**:
    - Top Accuracy: 0.9953
    - Lowest Loss: 0.0152
  - **Validation**:
    - Top Accuracy: 0.9907
    - Lowest Loss: 0.0323
  - **Overall Accuracy:** 0.9890
  - **Overall Loss:** 0.0346
  - **F1 Score**: 0.9854
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

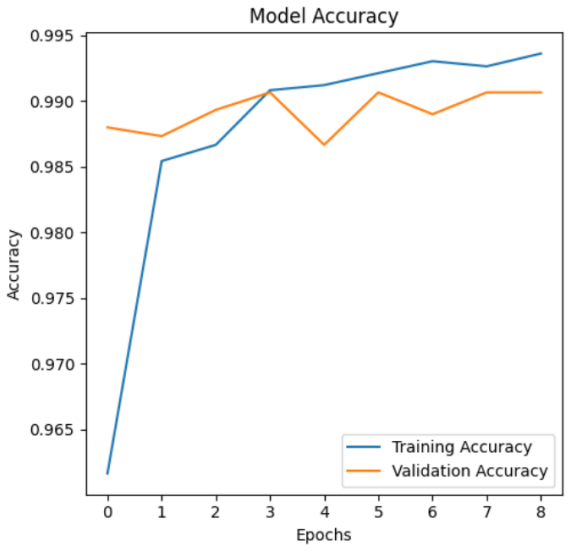

Figure 49: Accuracy-VGG16 for Art Nouveau

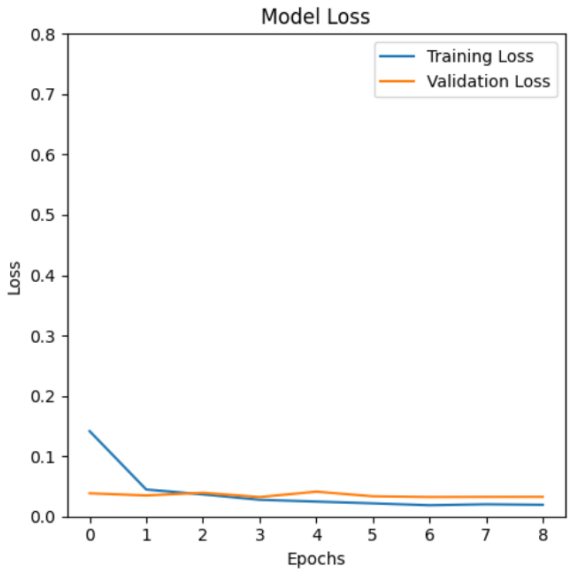

Figure 50: Loss-VGG16 for Art Nouveau

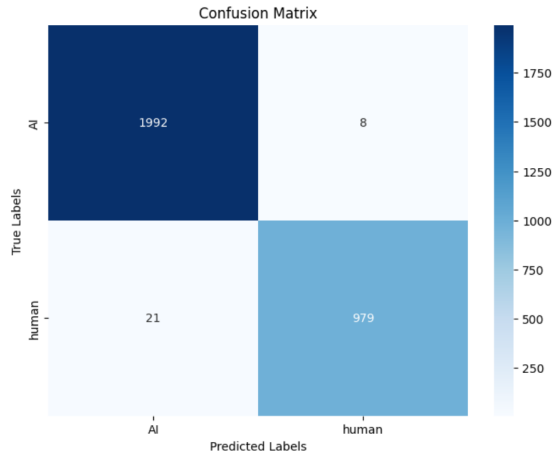

Figure 51: Confusion Matrix for Art Nouveau VGG16

- **VGG19**: Since the ”early stopping” was applied to the model, the network trained for a total of 18 epochs.
  - **Training**:
    - Top Accuracy: 0.9939
    - Lowest Loss: 0.0158
  - **Validation**:
    - Top Accuracy: 0.9907
    - Lowest Loss: 0.0277
  - **Overall Accuracy:** 0.9900
  - **Overall Loss:** 0.0323
  - **F1 Score**: 0.9855
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

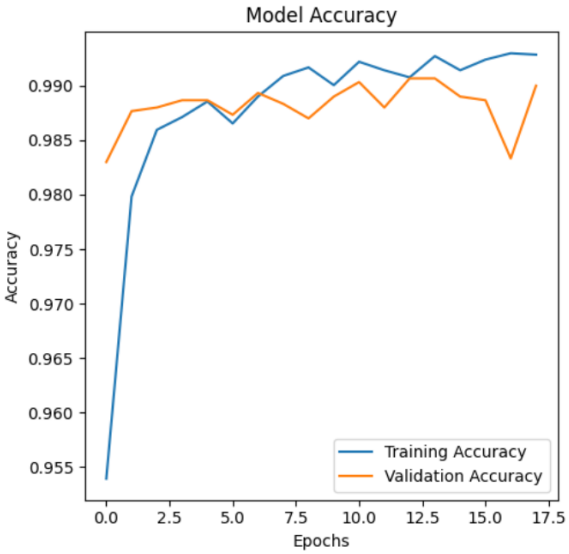

Figure 52: Accuracy-VGG19 for Art Nouveau

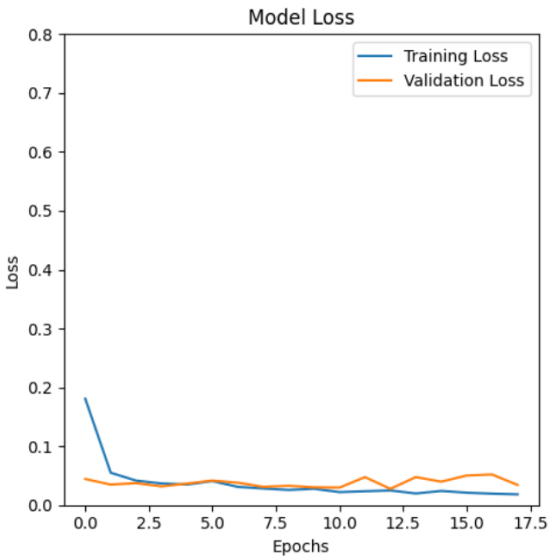

Figure 53: Loss-VGG19 for Art Nouveau

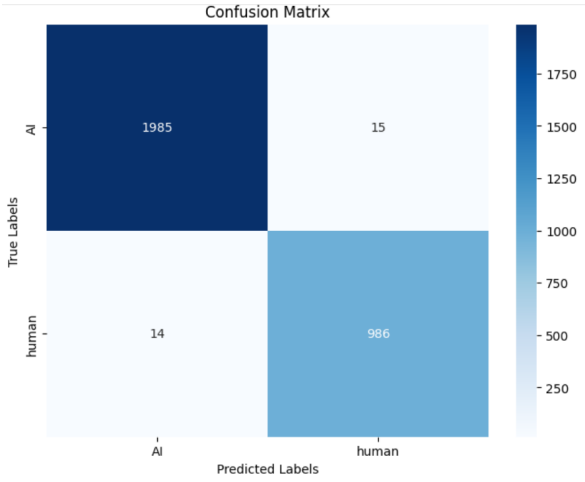

Figure 54: Confusion Matrix for Art Nouveau VGG19

5.2.7 Post Impressionism

- **CNN**: Since the "early stopping" was applied to the model, the network trained for a total of 25 epochs.
  - **Training**:
    - Top Accuracy: 0.9706
    - Lowest Loss: 0.0864
  - **Validation**:
    - Top Accuracy: 0.9547
    - Lowest Loss: 0.1406
  - **Overall Accuracy:** 0.9533
  - **Overall Loss:** 0.1392
  - **F1 Score**: 0.9254
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

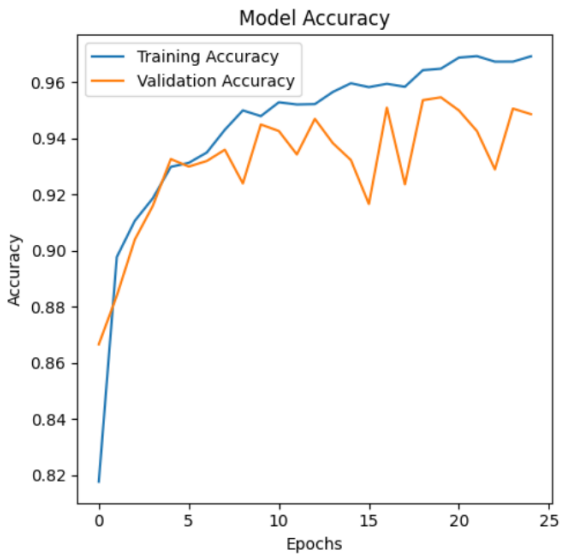

Figure 55: Accuracy-CNN for Post Impressionism

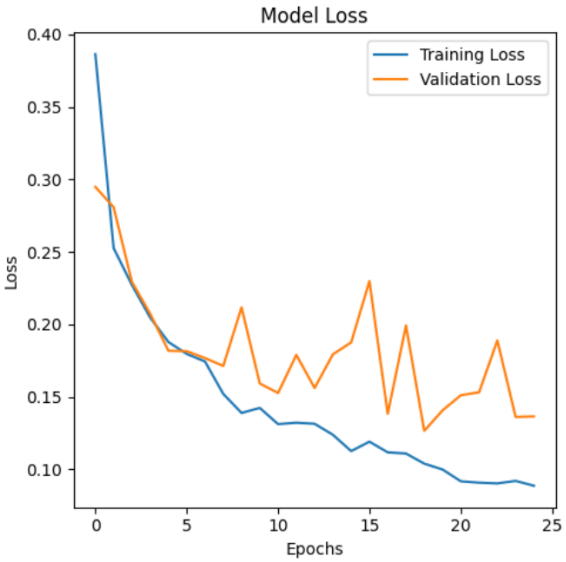

Figure 56: Loss-CNN for Post Impressionism

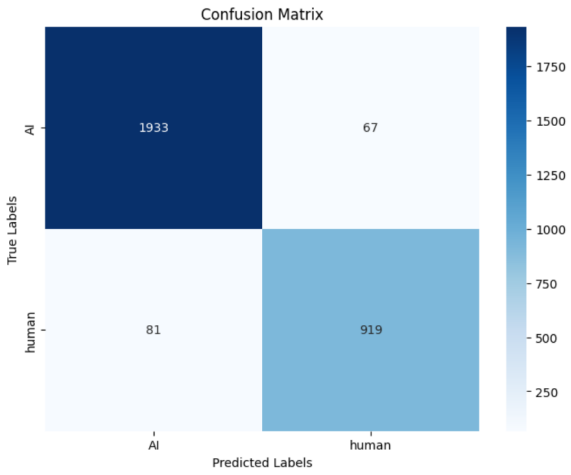

Figure 57: Confusion Matrix for Post Impressionism CNN

- **VGG16**: Since the "early stopping" was applied to the model, the network trained for a total of 9 epochs.
  - **Training**:
    - Top Accuracy: 0.9618
    - Lowest Loss: 0.1004
  - **Validation**:
    - Top Accuracy: 0.9563
    - Lowest Loss: 0.1177
  - **Overall Accuracy:** 0.9556
  - **Overall Loss:** 0.1160
  - **F1 Score**: 0.938
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

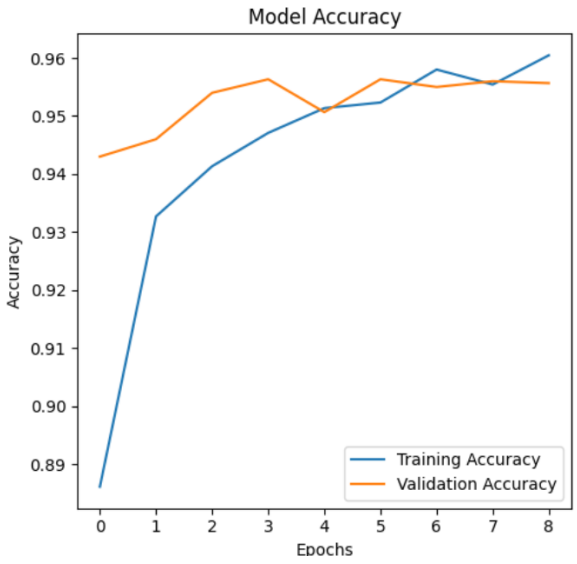

Figure 58: Accuracy-VGG16 for Post Impressionism

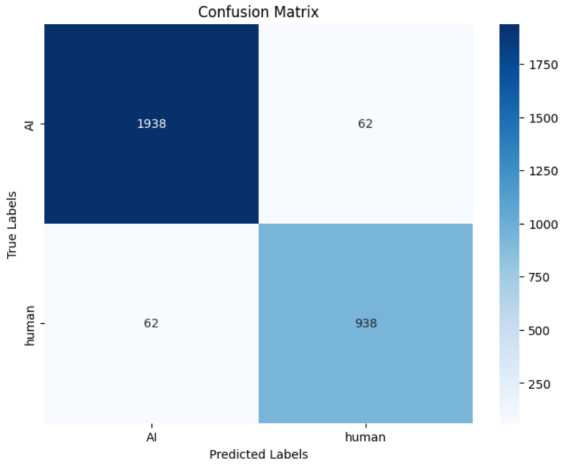

Figure 60: Confusion Matrix for VGG16 Post Impressionism

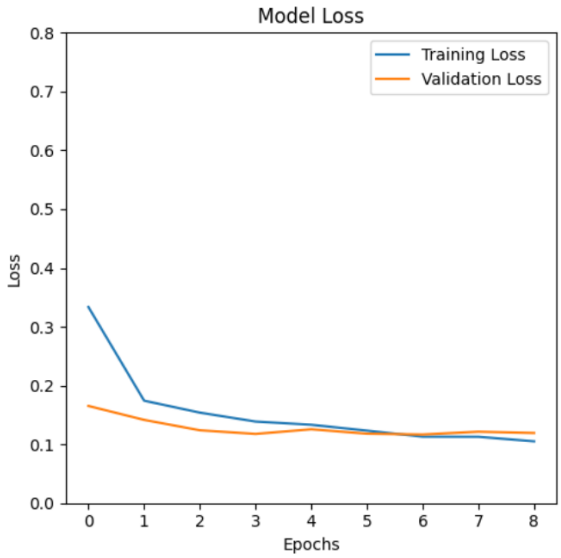

Figure 59: Loss-VGG16 for Post Impressionism

- **VGG19**: Since the ”early stopping” was applied to the model, the network trained for a total of 20 epochs.

  - **Training**:
    - Top Accuracy: 0.9561
    - Lowest Loss: 0.1169
  - **Validation**:
    - Top Accuracy: 0.9510
    - Lowest Loss: 0.1387
  - **Overall Accuracy:** 0.9533
  - **Overall Loss:** 0.1397
  - **F1 Score**: 0.9216
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

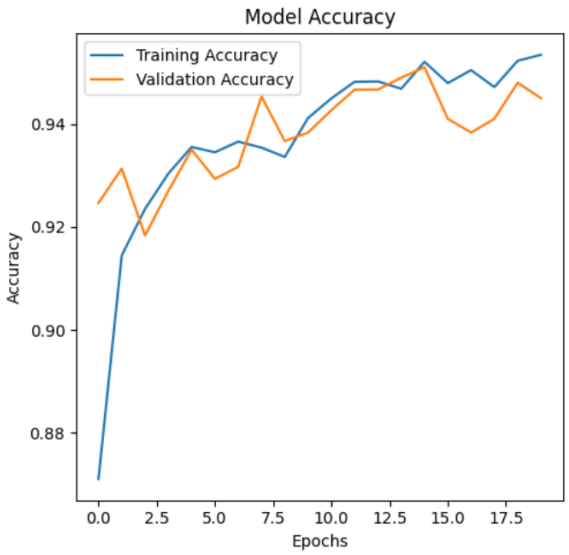

Figure 61: Accuracy-VGG19 for Post Impressionism

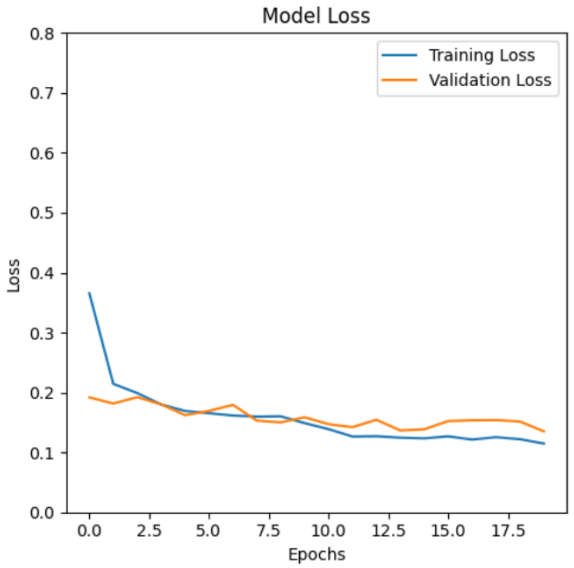

Figure 62: Loss-VGG19 for Post Impressionism

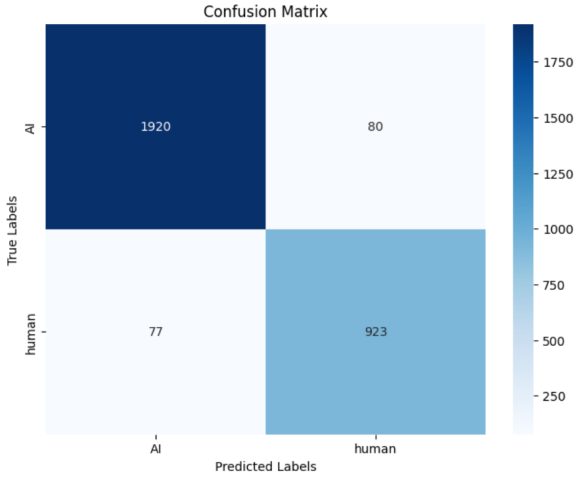

Figure 63: Confusion Matrix for Post Impressionism VGG19

5.2.8 Impressionism

- **CNN**: Since the ”early stopping” was applied to the model, the network trained for a total of 21 epochs.
  - **Training**:
    - Top Accuracy: 0.9489
    - Lowest Loss: 0.1454
  - **Validation**:
    - Top Accuracy: 0.9337
    - Lowest Loss: 0.1877
  - **Overall Accuracy:** 0.9330
  - **Overall Loss:** 0.1880
  - **F1 Score**: 0.8980
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

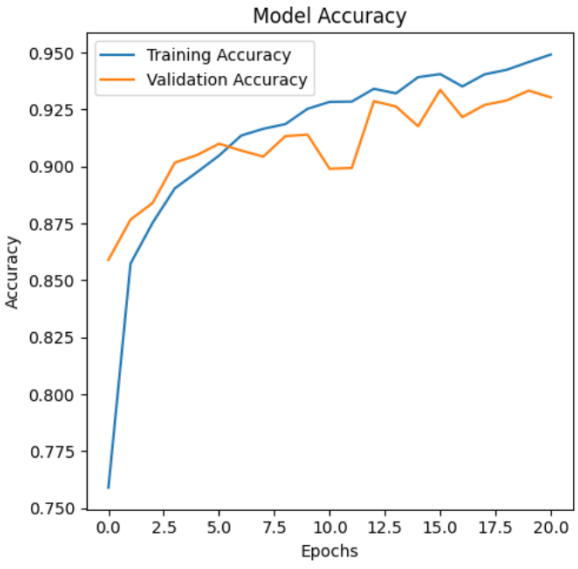

Figure 64: Accuracy-CNN for Impressionism

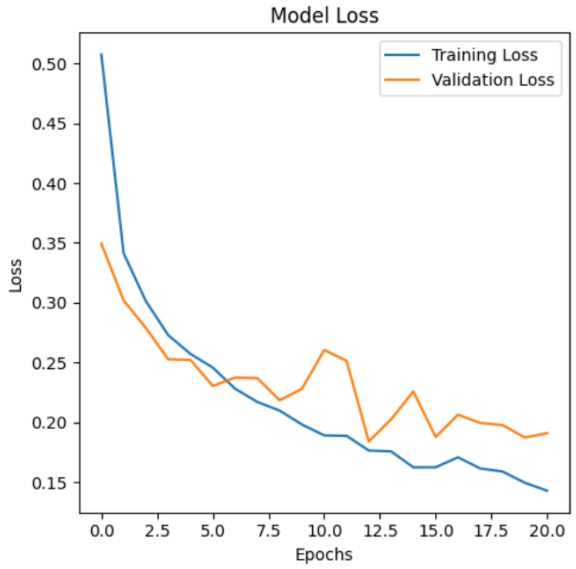

Figure 65: Loss-CNN for Impressionism

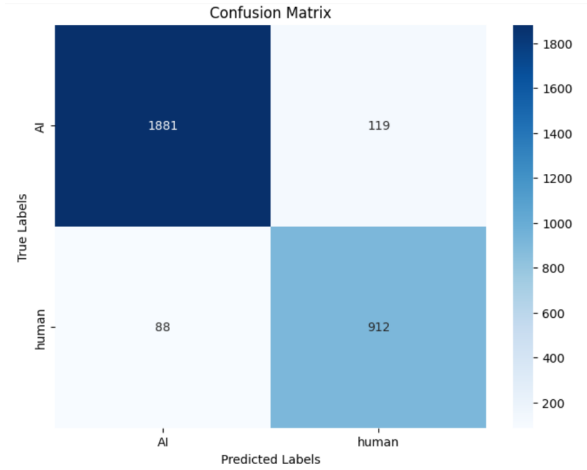

Figure 66: Confusion Matrix for Impressionism CNN

- **VGG16**: Since the ”early stopping” was applied to the model, the network trained for a total of 13 epochs.
  - **Training**:
    - Top Accuracy: 0.9887
    - Lowest Loss: 0.0286
  - **Validation**:
    - Top Accuracy: 0.9900
    - Lowest Loss: 0.0284
  - **Overall Accuracy:** 0.9876
  - **Overall Loss:** 0.0345
  - **F1 Score**: 0.9834
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

Figure 67: Accuracy-VGG16 for Impressionism

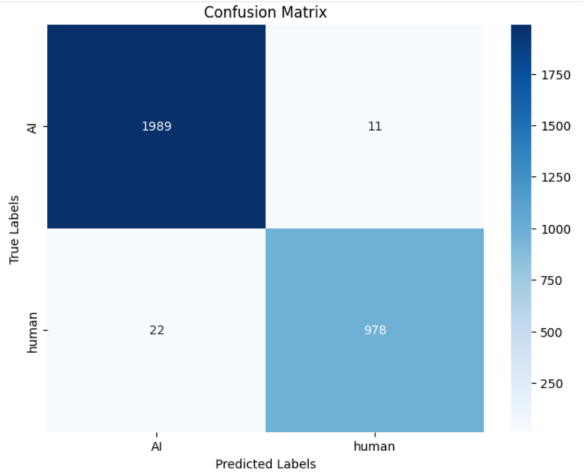

Figure 69: Confusion Matrix for Impressionism VGG16

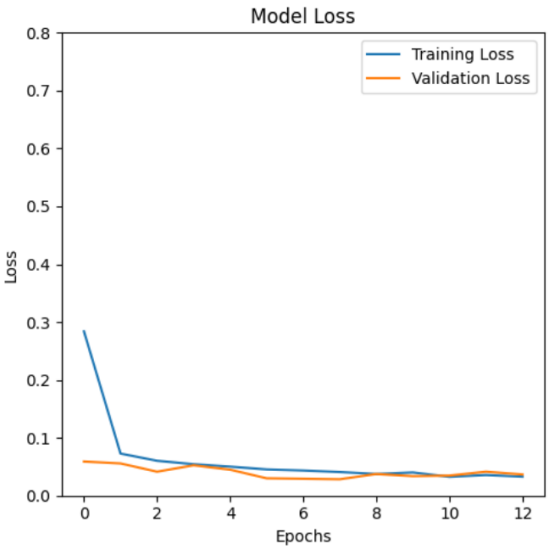

Figure 68: Loss-VGG16 for Impressionism

- **VGG19**: Since the ”early stopping” was applied to the model, the network trained for a total of 16 epochs.

  - **Training**:
    - Top Accuracy: 0.9525
    - Lowest Loss: 0.1255
  - **Validation**:
    - Top Accuracy: 0.9480
    - Lowest Loss: 0.1404
  - **Overall Accuracy:** 0.9433
  - **Overall Loss:** 0.1411
  - **F1 Score**: 0.9208
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

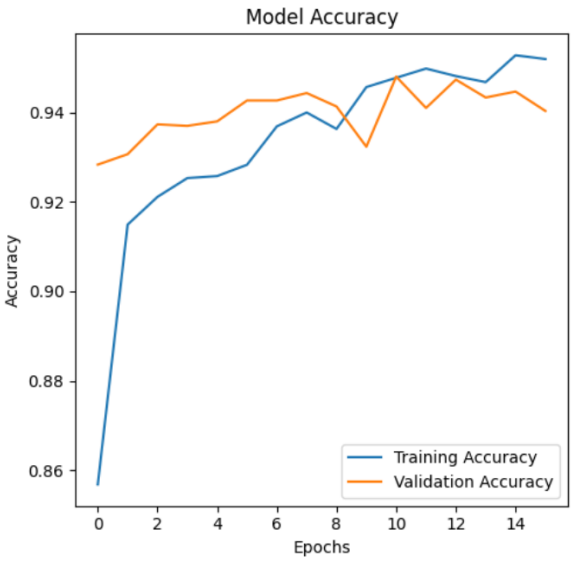

Figure 70: Accuracy-VGG19 for Impressionism

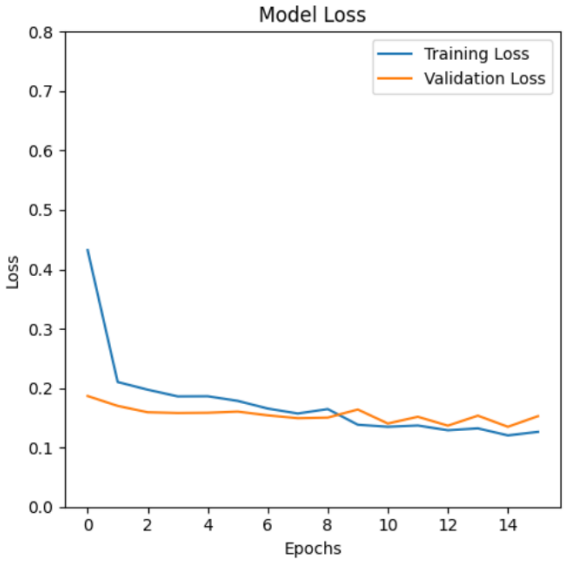

Figure 71: Loss-VGG19 for Impressionism

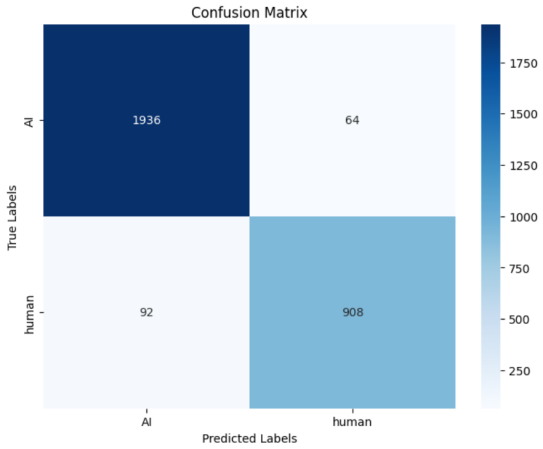

Figure 72: Confusion Matrix for Impressionism VGG19

5.2.9 Expressionism

- **CNN**: Since the ”early stopping” was applied to the model, the network trained for a total of 23 epochs.
  - **Training**:
    - Top Accuracy: 0.9494
    - Lowest Loss: 0.1385
  - **Validation**:
    - Top Accuracy: 0.9297
    - Lowest Loss: 0.1966
  - **Overall Accuracy:** 0.9313
  - **Overall Loss:** 0.1905
  - **F1 Score**: 0.8848
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

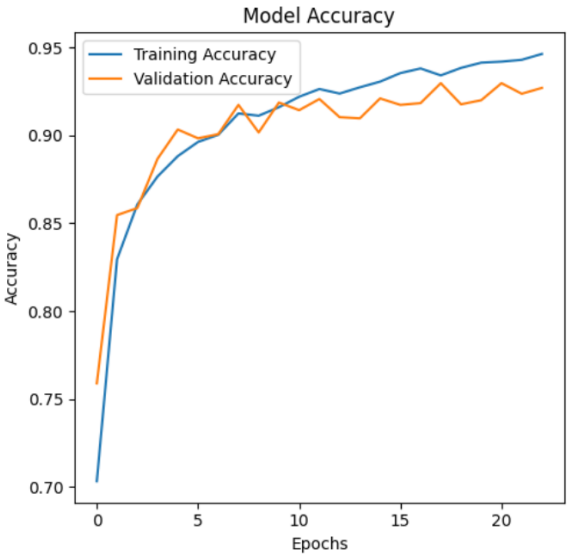

Figure 73: Accuracy-CNN for Expressionism

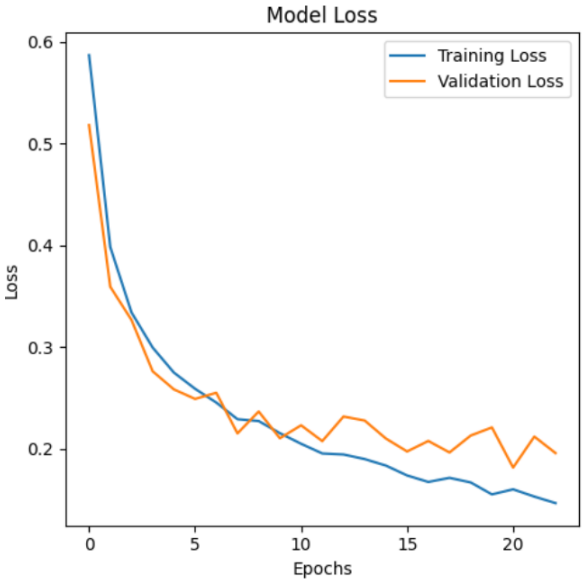

Figure 74: Loss-CNN for Expressionism

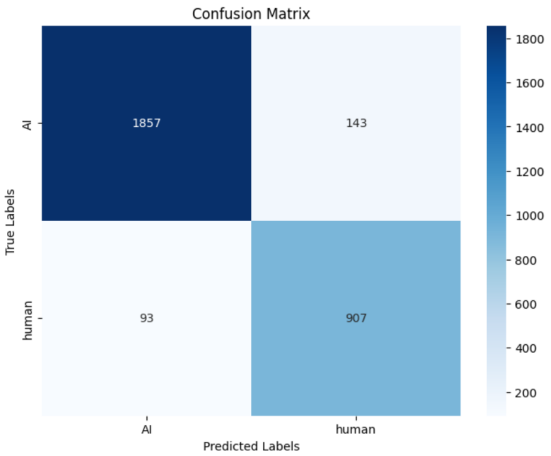

Figure 75: Confusion Matrix for Expressionism CNN

- **VGG16**: Since the ”early stopping” was applied to the model, the network trained for a total of 10 epochs.
  - **Training**:
    - Top Accuracy: 0.9635
    - Lowest Loss: 0.0984
  - **Validation**:
    - Top Accuracy: 0.9607
    - Lowest Loss: 0.1144
  - **Overall Accuracy:** 0.9606
  - **Overall Loss:** 0.1117
  - **F1 Score**: 0.9390
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

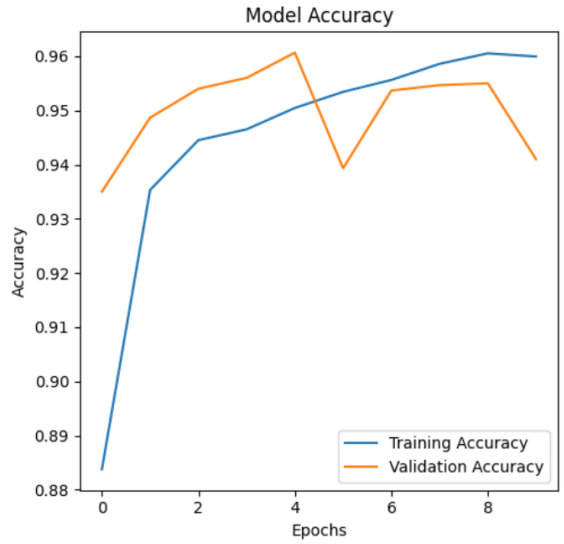

Figure 76: Accuracy-VGG16 for Expressionism

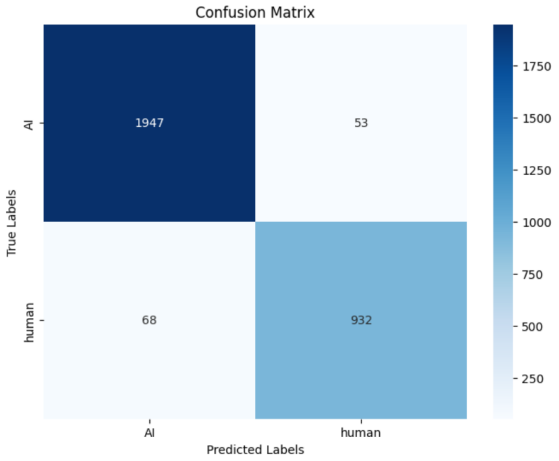

Figure 78: Confusion Matrix for Expressionism VGG16

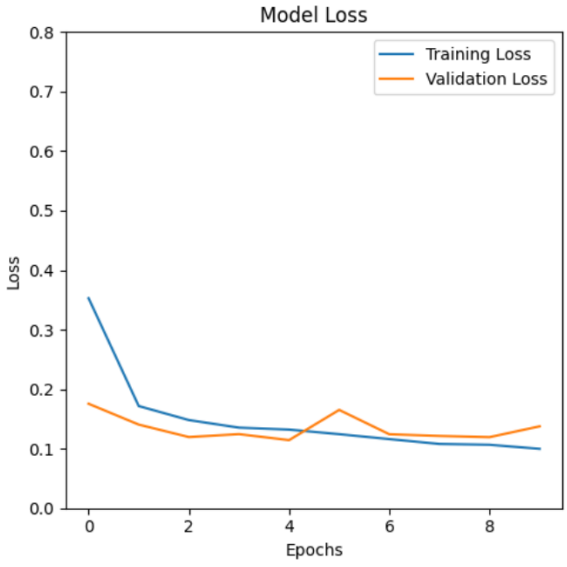

Figure 77: Loss-VGG16 for Expressionism

- **VGG19**: Since the ”early stopping” was applied to the model, the network trained for a total of 21 epochs.

  - **Training**:

    * Top Accuracy: 0.9535
    * Lowest Loss: 0.119

  - **Validation**:

    * Top Accuracy: 0.9513
    * Lowest Loss: 0.1436

  - **Overall Accuracy:** 0.9486
  - **Overall Loss:** 0.1463
  - **F1 Score**: 0.9219
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

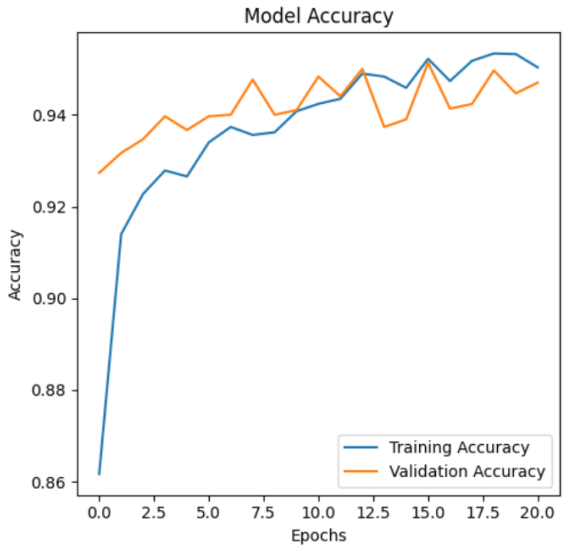

Figure 79: Accuracy-VGG19 for Expressionism

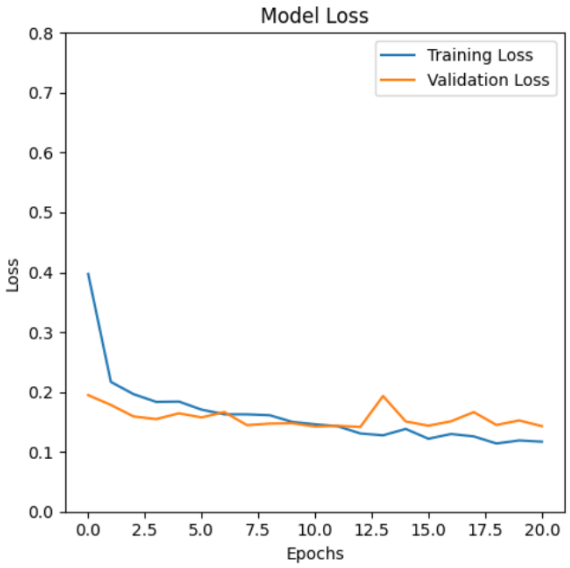

Figure 80: Loss-VGG19 for Expressionism

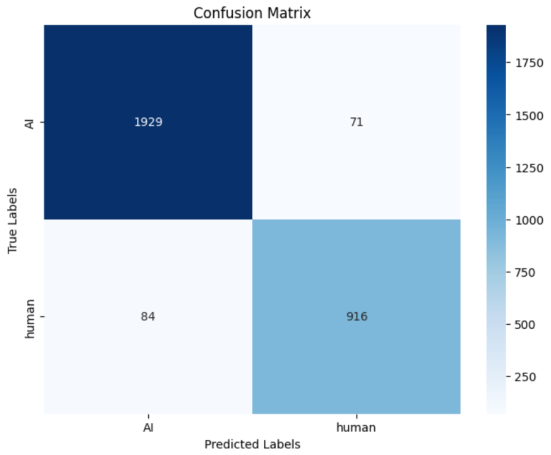

Figure 81: Confusion Matrix for Expressionism VGG19

5.2.10 Surrealism

- **CNN**: Since the ”early stopping” was applied to the model, the network trained for a total of 17 epochs.
  - **Training**:
    - Top Accuracy: 0.9386
    - Lowest Loss: 0.1620
  - **Validation**:
    - Top Accuracy: 0.9310
    - Lowest Loss: 0.1919
  - **Overall Accuracy:** 0.9309
  - **Overall Loss:** 0.1944
  - **F1 Score**: 0.8873
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

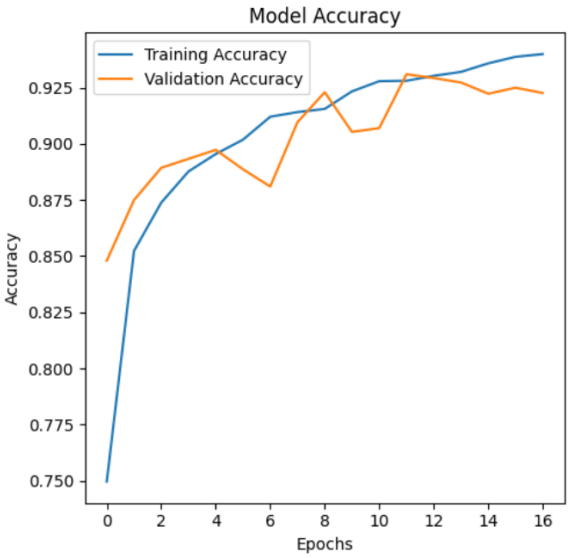

Figure 82: Accuracy-CNN for Surrealism

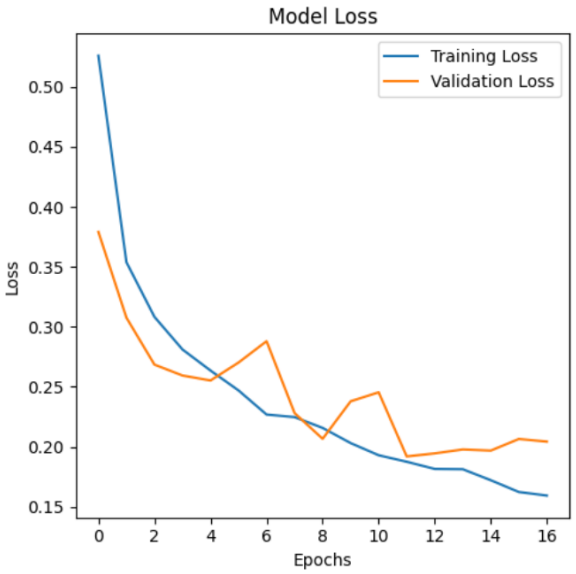

Figure 83: Loss-CNN for Surrealism

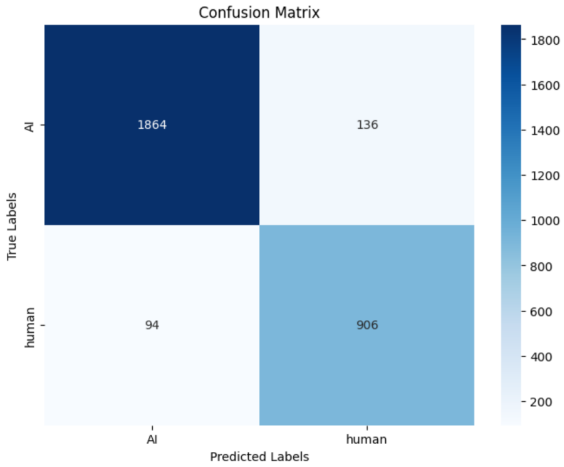

Figure 84: Confusion Matrix for Surrealism CNN

- **VGG16**: Since the ”early stopping” was applied to the model, the network trained for a total of 22 epochs.
  - **Training**:
    - Top Accuracy: 0.9747
    - Lowest Loss: 0.0660
  - **Validation**:
    - Top Accuracy: 0.9657
    - Lowest Loss: 0.1163
  - **Overall Accuracy:** 0.9610
  - **Overall Loss:** 0.1063
  - **F1 Score**: 0.9439
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

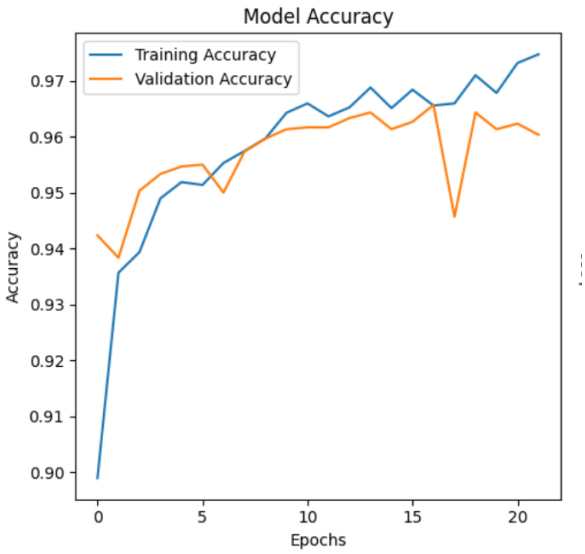

Figure 85: Accuracy-VGG16 for Surrealism

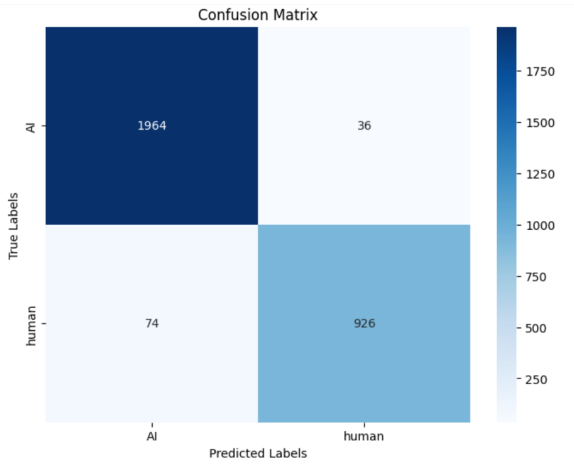

Figure 87: Confusion Matrix for Surrealism VGG16

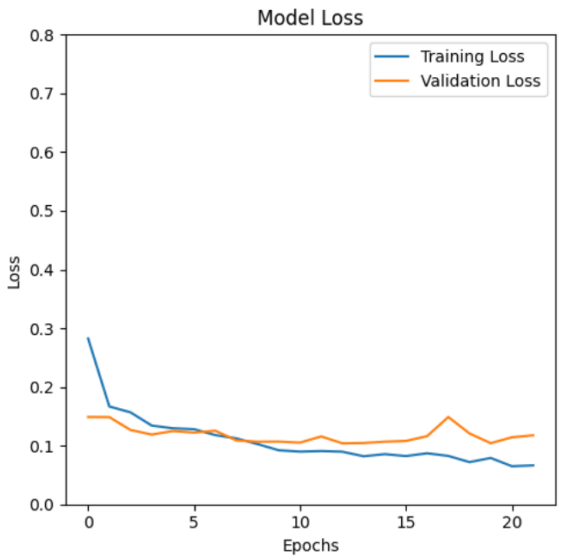

Figure 86: Loss-VGG16 for Surrealism

- **VGG19**: Since the ”early stopping” was applied to the model, the network trained for a total of 22 epochs.
  - **Training**:
    - Top Accuracy: 0.9552
    - Lowest Loss: 0.1163
  - **Validation**:
    - Top Accuracy: 0.9530
    - Lowest Loss: 0.1256
  - **Overall Accuracy:** 0.9506
  - **Overall Loss:** 0.1321
  - **F1 Score**: 0.9204
  - **Graphs of Accuracy, Loss. and Confusion Matrix**:

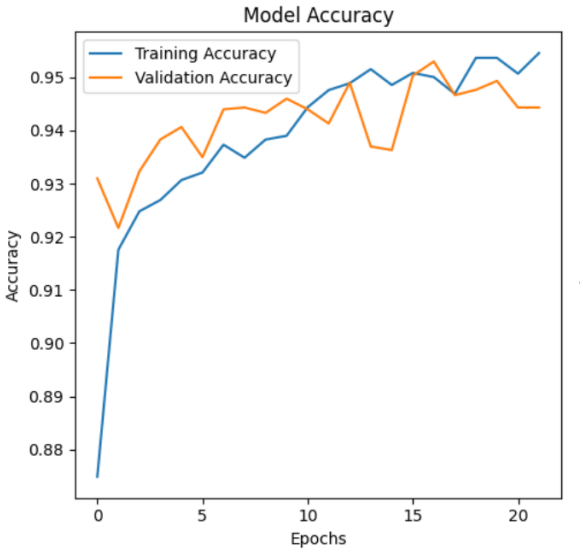

Figure 88: Accuracy-VGG19 for Surrealism

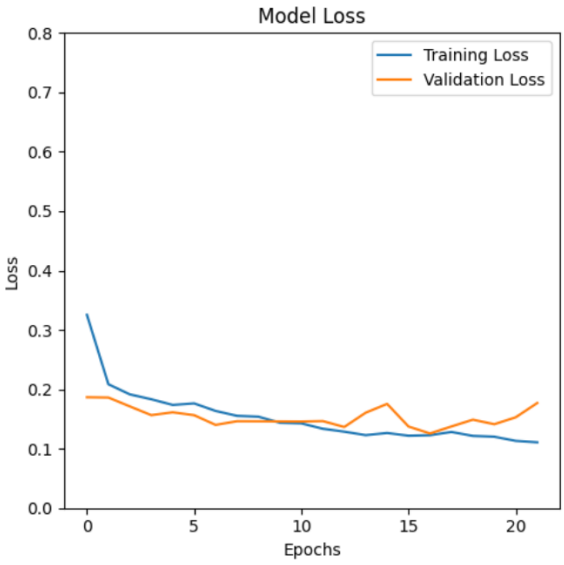

Figure 89: Loss-VGG19 for Surrealism

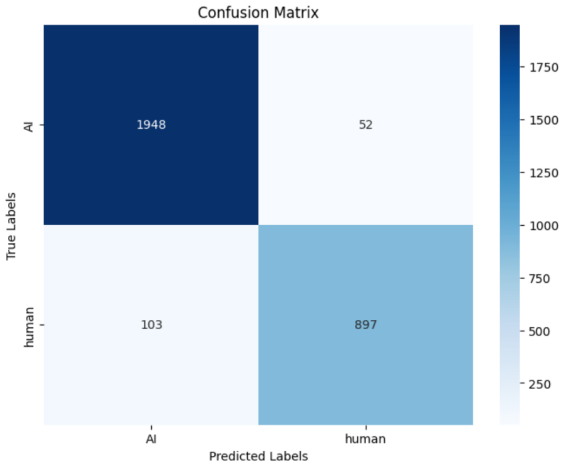

Figure 90: Confusion Matrix for Surrealism VGG19

5.3 Final Conclusions of First Phase

Our phase of comparing art styles using a custom CNN, VGG16, and VGG19 revealed the limitations of these models in handling complex art styles. As it is shown in the accuracy's, number of epochs and graphics, the models are not well suited for some of the art styles, due to the complexity of the images and the lack of enough complexity of the model, such as the Convolutional Neural Network it was created from scratch. Certain art styles, such as Baroque, Art Nouveau, Post-Impressionism, Impressionism, Expressionism, and Ukiyo-e, exhibited lower classification accuracy. These styles contain intricate details and varied patterns that the current models struggled to capture effectively. Having analyzed the results of our experiments, we can reorganize the art styles based on the accuracy achieved by the classifiers.

The new order of the Art Styles is based on an average of the 3 test accuracies in the trained models, ending up with:

1. **Ukiyo-E** - Acc: 0.9885
2. **Art Nouveau** - Acc: 0.9847
3. **Baroque** - Acc: 0.9630
4. **Renaissance** - Acc: 0.9566
5. **Impressionism** - Acc: 0.9546
6. **Post Impressionism** - Acc: 0.9540
7. **Romanticism** - Acc: 0.9495
8. **Surrealism** - Acc: 0.9475
9. **Expressionism** - Acc: 0.9468
10. **Realism** - Acc: 0.9453

6 Second Phase - All Art Styles AI-Detection

For the second phase, all art styles together were used to train different models in order to find the most efficient and accurate network able to distinguish if the art is AI generated or human generated.

6.1 Implemented Models

6.1.1 ResNet50

- **Architecture**:
  - **Load Pre-trained ResNet50 Model**:
    - **ResNet50**: We are using a model called ResNet50, which is a deep learning model with 50 layers.
    - **Pre-trained on ImageNet**: The model is already trained on a large dataset of images called ImageNet, which helps it recognize general patterns in images.
    - **include_top=False**: We are not including the final classification layers of the ResNet50 model because we will add our own layers for the specific task of AI art detection.
    - **input_shape=(224, 224, 3)**: The input images are resized to 224x224 pixels with 3 color channels (RGB).
  - **Freeze the Base Model Layers:**
    - We are freezing the layers of the ResNet50 model, which means their weights will not be updated during training. This helps retain the knowledge the model has from the ImageNet dataset.
  - **Add New Layers**
    - **Sequential Model**: We are creating a new model by adding layers on top of the frozen ResNet50 model.
    - **Flatten**: This layer flattens the output from the ResNet50 model into a 1D array, making it suitable for the dense layers that follow.
    - **Dense(256, activation='relu')**: This layer has 256 neurons and uses the ReLU activation function, which helps the model learn complex patterns.
    - **Dropout(0.5)**: This layer randomly drops 50% of the neurons during training to prevent overfitting, helping the model generalize better to new data.
    - **Dense(1, activation='sigmoid')**: The final layer has 1 neuron and uses the sigmoid activation function, making it suitable for binary classification (AI-generated vs. human-generated art).
  - **Results**: Since the "early stopping" was applied to the model, the network trained for a total of 26 epochs.
    - **Training**

- Accuracy: 0.8157
- Loss: 0.4000

* **Validation**
  - Accuracy: 0.8632
  - Loss: 0.3500
* **Overall Accuracy:** 0.8615
* **Overall Loss:** 0.3511
* **Graphs of Accuracy, Loss. and Confusion Matrix**

Figure 91: Accuracy of ResNet50

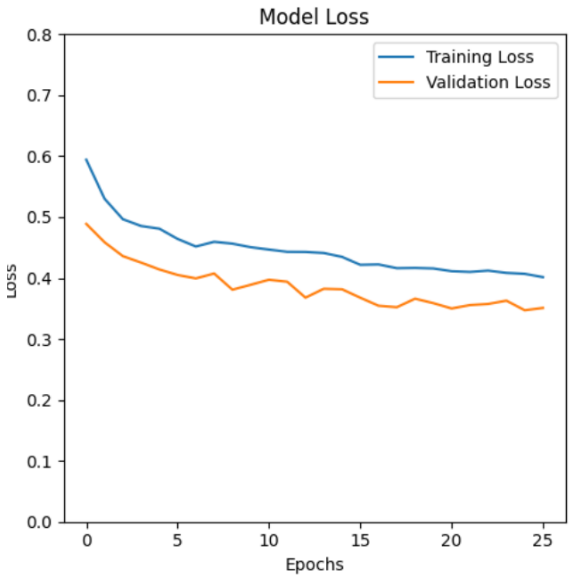

Figure 92: Loss of ResNet50

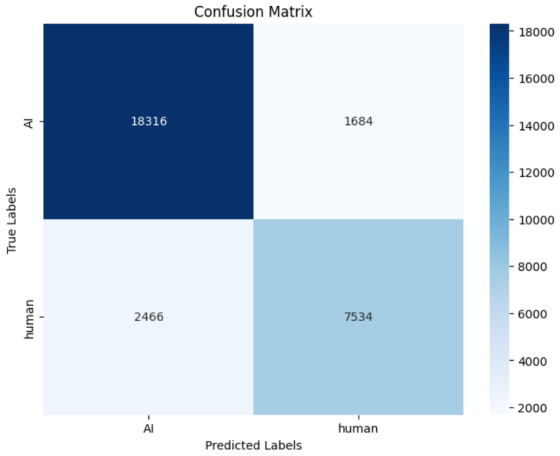

Figure 93: Confusion Matrix of ResNet50

6.1.2 InceptionV3

- **Architecture**:
  - **Load Pre-trained ResNet50 Model**:
    * **InceptionV3**: We are using a model called InceptionV3, which is a deep learning model with a sophisticated architecture that can capture intricate details in images.
    * **Pre-trained on ImageNet**: The model is already trained on a large dataset of images called ImageNet, which helps it recognize general patterns in images.
    * **include_top=False**: We are not including the final classification layers of the ResNet50 model because we will add our own layers for the specific task of AI art detection.
    * **input_shape=(224, 224, 3)**: The input images are resized to 224x224 pixels with 3 color channels (RGB).
  - **Freeze the Base Model Layers:**
    * We are freezing the layers of the InceptionV3 model, which means their weights will not be updated during training. This helps retain the knowledge the model has from the ImageNet dataset.
  - **Add New Layers**
    * **Sequential Model**: We are creating a new model by adding layers on top of the frozen InceptionV3 model.
    * **Flatten**: This layer flattens the output from the InceptionV3 model into

a 1D array, making it suitable for the dense layers that follow.

  - **Dense(256, activation='relu')**: This layer has 256 neurons and uses the ReLU activation function, which helps the model learn complex patterns.
  - **Dropout(0.5)**: This layer randomly drops 50% of the neurons during training to prevent overfitting, helping the model generalize better to new data.
  - **Dense(1, activation='sigmoid')**: The final layer has 1 neuron and uses the sigmoid activation function, making it suitable for binary classification (AI-generated vs. human-generated art).

– **Results**: Since the "early stopping" was applied to the model, the network trained for a total of 17 epochs.

  - **Training**
    - Accuracy: 0.9146
    - Loss: 0.1938
  - **Validation**
    - Accuracy: 0.9465
    - Loss: 0.1525
  - **Overall Accuracy:** 0.9453
  - **Overall Loss:** 0.1518
  - **Graphs of Accuracy, Loss. and Confusion Matrix**

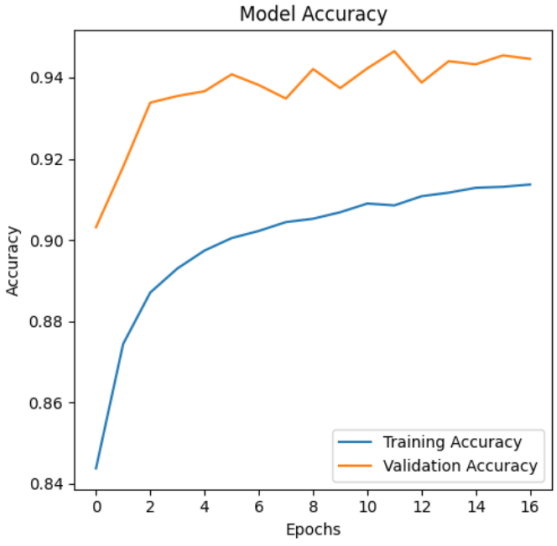

Figure 94: Accuracy of InceptionV3

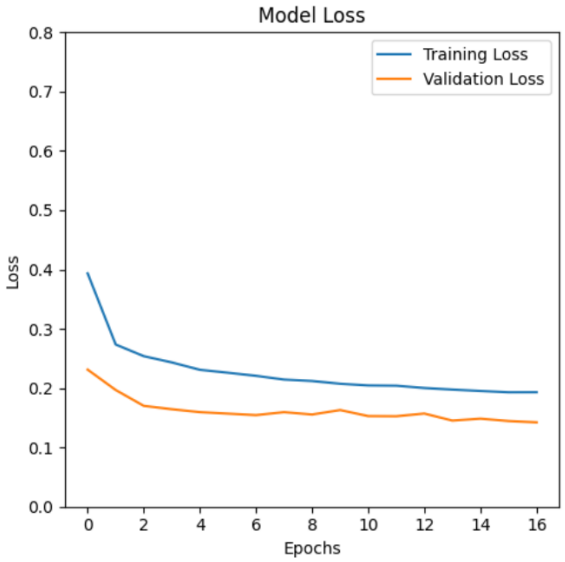

Figure 95: Loss of InceptionV3

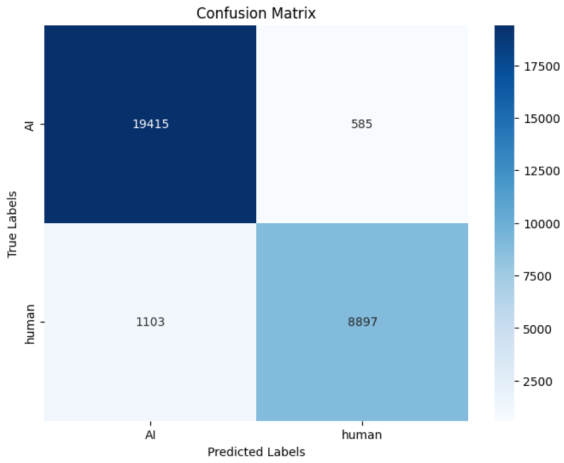

Figure 96: Confusion Matrix of InceptionV3

7 Final Conclusions of Second Phase

Based on the results, InceptionV3 outperformed ResNet50 in detecting whether art styles were AI-generated or human-generated. The key findings are:

- **Higher Accuracy**: InceptionV3 achieved an accuracy of 0.9453, significantly higher than ResNet50's 0.8615. This indicates that InceptionV3 was more effective in correctly classifying the images.

- **Lower Loss**: InceptionV3 had a lower loss value of 0.1518 compared to ResNet50's 0.3511. Lower loss signifies better model performance in terms of prediction errors.

InceptionV3 had fewer **false positives** (AI incorrectly identified as human) and **false negatives** (human incorrectly identified as AI) compared to ResNet50. Specifically, InceptionV3 had 585 false positives and 1,103 **false negatives**, whereas ResNet50 had 1,684 false positives and 2,466 **false negatives**. In summary, InceptionV3 demonstrated superior performance in identifying AI-generated and human-generated art styles, this suggests that InceptionV3's complex architecture is better suited for capturing the intricate features required for this classification task.

8 Conclusion of the Project

In this project, we aimed to classify art styles and distinguish between AI-generated and human-generated artworks using various deep learning models. The project was divided into two phases, each providing critical insights into model performance and the challenges of image classification in art.

8.1 Phase One: CNN, VGG16 and VGG19

Our first phase involved comparing the performance of a custom Convolutional Neural Network (CNN), VGG16, and VGG19 models on different art styles. Key findings from this phase included:

- **Model Limitations:** The custom CNN, due to its simpler architecture, struggled with the complexity of certain art styles. Both VGG16 and VGG19 showed improved performance but still faced challenges with highly intricate art styles.
- **Art Style Complexity:** Art styles such as Baroque, Art Nouveau, Post-Impressionism, Impressionism, Expressionism, and Ukiyo-e exhibited lower classification accuracy. These styles contain complex details and patterns that were not adequately captured by the models.
- **Dataset Size and Diversity:** We learned that the small size and limited diversity of our dataset negatively impacted the models' performance. More complex models with access to larger and more diverse datasets performed better, highlighting the importance of data quantity and variety in training robust models.

8.2 Phase Two: ResNet50 and InceptionV3

The second phase focused on comparing ResNet50 and InceptionV3 models for identifying AI-generated and human-generated from all art styles. InceptionV3 outperformed ResNet50 by achieving higher accuracy, lower loss, and fewer misclassifications. The complex architecture of InceptionV3 allowed it to capture the intricate features necessary for accurately distinguishing between AI-generated and human-generated art.

8.3 Summary.

We clearly saw that there is a correlation between the amount of data and the complexity of it, with the accuracy of the CNN model. It came to our attention that complex CNNs that were trained with small datasets had worst results than models with larger datasets, since they require a vast amount of parameters to have a robust performance and the ability to accurately capture and classify complex image features. This project highlights the importance of model selection and architecture design in achieving high accuracy in image classification tasks by considering three major points:

- Data complexity
- Dataset Size
- Network Complexity

Bibliography

[1] GitHub. Github. https://github.com/katiaperchet/ai-art-casa, 2024. Accessed on 2024-07-22.

[2] Kaggle. Dataset: Ai-artbench. https://www.kaggle.com/datasets/ravidussilva/real-ai-art, 2023. Accessed on 2024-07-22.

[3] Wikipedia. Realism. https://en.wikipedia.org/wiki/Realism, 2024. Accessed on 2024-07-22.

[4] Wikipedia. Renaissance. https://en.wikipedia.org/wiki/Renaissance, 2024. Accessed on 2024-07-22.

[5] Wikipedia. Romanticism. https://en.wikipedia.org/wiki/Romanticism, 2024. Accessed on 2024-07-22.

[6] Wikipedia. Baroque. https://en.wikipedia.org/wiki/Baroque, 2024. Accessed on 2024-07-22.

[7] Wikipedia. Ukiyo-e. https://en.wikipedia.org/wiki/Ukiyo-e, 2024. Accessed on 2024-07-22.

[8] Wikipedia. Art nouveau. https://en.wikipedia.org/wiki/Art_Nouveau, 2024. Accessed on 2024-07-22.

[9] Wikipedia. Post impressionism. https://en.wikipedia.org/wiki/Post-Impressionism, 2024. Accessed on 2024-07-22.

[10] Wikipedia. Impressionism. https://en.wikipedia.org/wiki/Impressionism, 2024. Accessed on 2024-07-22.

[11] Wikipedia. Expressionism. https://en.wikipedia.org/wiki/Expressionism, 2024. Accessed on 2024-07-22.

[12] Wikipedia. Surrealism. https://en.wikipedia.org/wiki/Surrealism, 2024. Accessed on 2024-07-22.

[13] TensorFlow. Cnn. https://www.tensorflow.org/tutorials/images/cnn, 2024. Accessed on 2024-07-22.

[14] Keras. Vgg16. https://keras.io/api/applications/vgg/, 2024. Accessed on 2024-07-22.

[15] Keras. Vgg19. https://keras.io/api/applications/vgg/, 2024. Accessed on 2024-07-22.

[16] Keras. Resnet50. https://keras.io/api/applications/resnet/, 2024. Accessed on 2024-07-22.

[17] Wikipedia. Inceptionv3. https://keras.io/api/applications/inceptionv3/, 2024. Accessed on 2024-07-22.