“Classifying two images of painting as belonging to the same artist or not.”

Approach: Convolutional Siamese Network

Deep Learning Final Project – Winter 2023

Mustafa Waked 318577921

Huda Sawaed 318386166

Introduction (Model description and goal)

The art world has always been captivated by the mystery behind the artists and their paintings. Determining the authenticity of a piece of artwork often requires matching it to its corresponding artist, a task that has traditionally been done manually. However, manual methods are not only time-consuming but also prone to errors. In this project, we introduce a revolutionary solution to this age-old problem. Using a Convolutional Siamese Network, we have developed a deep learning model that compares two paintings and predicts with high accuracy if they belong to the same artist. Our model takes the guesswork out of the equation, delivering results that are fast, reliable, and backed by data. In evaluating our model, we achieved good accuracy, making it a good solution for the art world. The Convolutional Siamese Network not only saves time and reduces the risk of errors but also has the potential to uncover new insights into the artists and their work.

Data Preparation

To prepare the data for our image similarity model, we first read in the CSV file containing information about the images and their corresponding artists using the readdatainfo() function. We then reduce the size of the dataset by randomly removing images from the majority class (artists with the most images) to match the size of the minority class (artists with the least images) using the reducedictrandom function. We then split the dataset into training, validation, and testing sets using the splitdict() function. We use a 60/40 split for the training and testing sets and set aside 20% of the data for the validation set. We then use the datagenerator() function to generate batches of image pairs and their corresponding labels for each set. To feed the image data into our Siamese network, we preprocess each image by resizing it to 224x224 pixels, decoding the image, and normalizing the pixel values using the preprocessimage() function. We then use a pair of Input layers to

define the two input images for our Siamese network. And the Siamese network uses the RESNET architecture as a base for feature extraction.

Model architecture

The model architecture of the Siamese Network below consists of two identical convolutional neural networks (CNNs) that share the same weights and architecture. The inputs to the network are two painting images (referred to as Input A and Input B) of shape (224, 224, 3), where the goal is to predict whether the two images are a match or not. The architecture of the CNN is based on the RESNET model, which is pre-trained on the ImageNet. The CNN takes as input a 3-channel image of size (224, 224) and applies a series of convolutional and pooling layers to extract features. The architecture of the CNN consists of 13 convolutional layers, 5 max pooling layers, and 3 fully connected layers. Each convolutional layer is followed by a rectified linear unit (ReLU) activation function, which introduces non-linearity into the network. The final layer of the CNN is a fully connected layer with 4096 units, followed by a ReLU activation function. The outputs of the CNN are two encoded representations of the input images, referred to as Encoded A and Encoded B, respectively. The encoded representations have a shape of (7, 7, 512), which corresponds to a 7 × 7 grid of 512 feature maps. To compare the encoded representations of the two input images, the Siamese Network uses a distance metric based on the absolute difference between the two encoded representations. The distance metric is implemented as a lambda layer that takes the absolute difference between Encoded A and Encoded B as input. The resulting tensor has a shape of (7, 7, 512) and represents the difference between the two encoded representations. The distance metric can be written as: distance = Λ(|Encoded A − Encoded B|) where Λ denotes the lambda layer that applies the absolute value function. The output of the distance metric layer is flattened using a GlobalMaxPooling layer, which converts the tensor into a 1D vector of length 25088. This vector is then goes through a drop-out (regularization) and then fed into a fully connected layer with a single unit, which applies a sigmoid activation function to produce a binary classification output indicating whether the two input images are a match or not. Mathematically, the output of the Siamese Network can be written as:

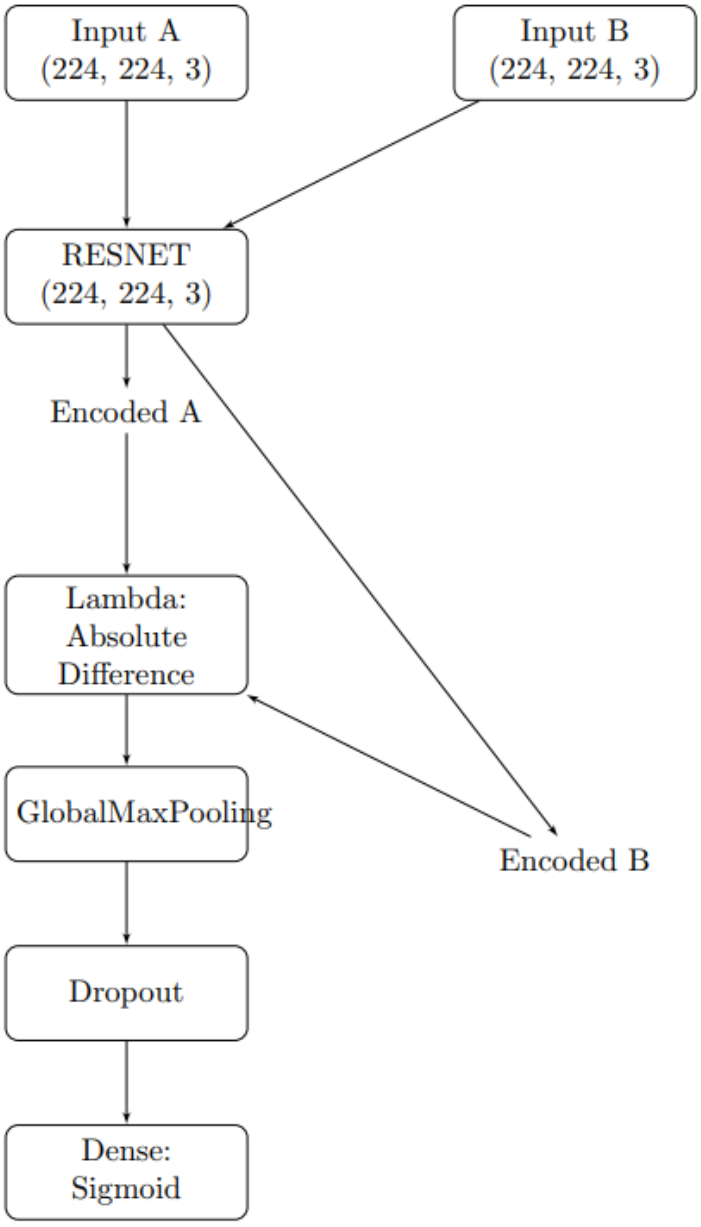
$$\text{output} = \sigma(W\, f(\text{distance}))$$

where W denotes the weight matrix of the fully connected layer, f denotes the GlobalMaxPooling layer, σ denotes the sigmoid activation function, and distance denotes the output tensor of the lambda layer. The Siamese Network is trained using binary cross-entropy loss and the Adam optimizer. The loss function measures the difference between the predicted and true binary labels and updates the network's weights accordingly. The loss function can be written as:

$$L(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

where $\hat{y}$ denotes the predicted binary label, y denotes the true binary label, and log denotes the natural logarithm.

Our Model architecture: (Note that the Dropout here was used for regularization)



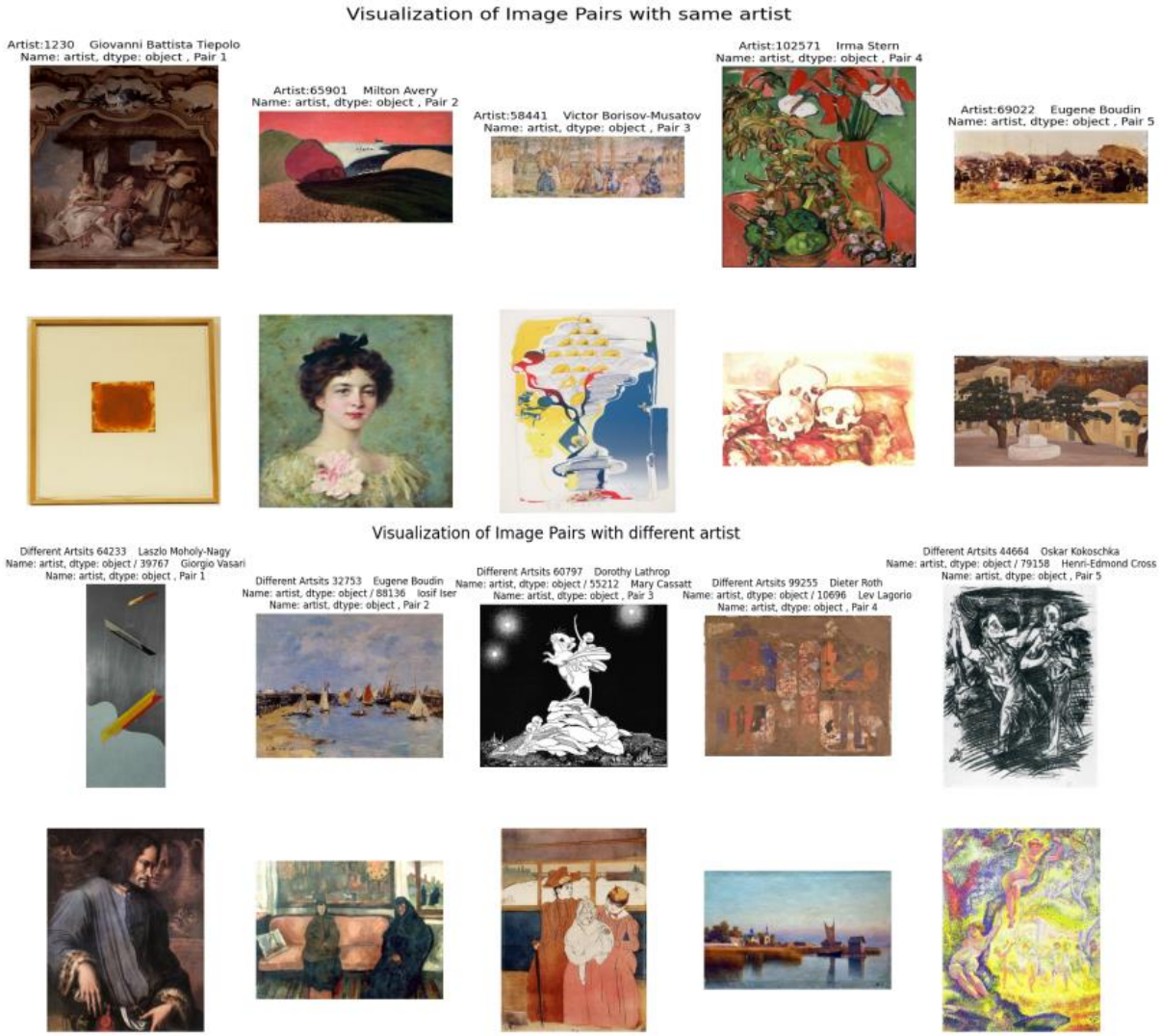
The Adam optimizer is used to minimize the loss function by adjusting the weights of the network.

In summary, the Siamese Network architecture can be represented as:

$$\text{output} = \sigma(W\, f(\Lambda(|\text{Encoded A} - \text{Encoded B}|)))$$

Data visualization

The image below presents displays pairs of paintings, each pair either created by the same artist or by different artists. These painting pairs were utilized as input for our model to estimate the degree of matching between them. By training our model on a dataset of painting pairs with known matching classes, we were able to develop a model that can accurately predict the match of two paintings. As shown in the image, the model was tested on a variety of painting pairs, including those from the same artist and those from different artists. The results of our model's predictions demonstrate its ability to distinguish between paintings created by the same artist and those created by different artists.



How to run the project:

1. Download the Dataset
   Download the following files from the Kaggle dataset link:
   https://www.kaggle.com/competitions/painter-by-numbers/data
   - train_2.zip
   - test.zip
   - replacements_for_corrupted_files.zip
   - all_data_info.csv

2. Put All the Files in the Same Directory(don't unzip the train_2 and test.zip)

3. Download the "ProjectCode.ipynb" and "DataCleaning.ipynb" files and put it in the same directory as the downloaded files.
4. Run the DataCleaning.ipynb using Jupyter Notebook. This file is to be run once, it unzips the data and create new folders for you for crop data. After the code finishes running, run now the ProjectCode.ipynb file that will train your model and saves the results in a new folder called models_1 (model weights).

How to test the project:

Two ways:

1. The validation data was used at the training phase to test the model:

   #Define the early stopping callback

   early_stopping = EarlyStopping(monitor='val_accuracy', patience=100, mode='min', verbose=1).

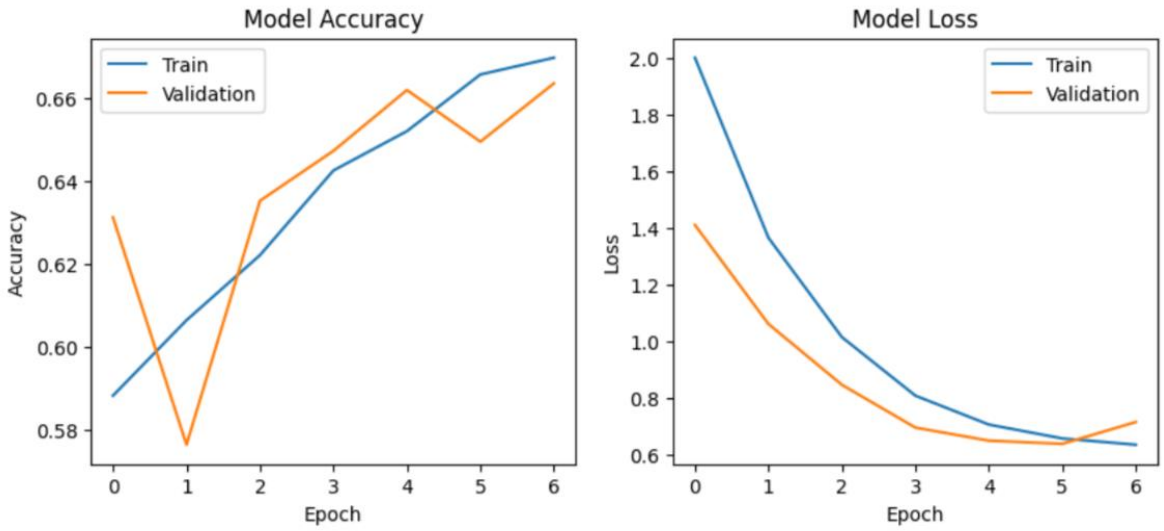
   # Train the model with early stopping

   h = siamese_net.fit(train_data_generator, epochs=1000, steps_per_epoch=500, validation_data=val_data_generator, validation_steps=300, callbacks=[early_stopping], verbose=1)

2. The second way is to test the model manually, to insert the images paths manually (2 images at a time) to check if they belong to the same artist (Check the last couple code cells in ProjectCode.ipynb.

Prediction Results:

In this section, we present the results of our machine learning model's predictions. We evaluated the performance of our model using the accuracy metric. Our model achieved an overall train accuracy of ~70%, with a validation set accuracy of 67.45% which indicates the ratio of the samples correctly classified in our dataset. This is a promising result and suggests that our model has learned to accurately identify patterns in the data.



Conclusion:

In conclusion, our project has presented a novel approach for automating the task of matching paintings to their corresponding artists using a combination of transfer learning and a Convolutional Siamese Network. The proposed method has demonstrated promising results in providing a fast and reliable solution to the problem of artist attribution. However, it is important to note that there are still limitations to our approach, including not achieving very high accuracy rates. Future work should focus on building, developing, and tuning the model to overcome these limitations and improve its performance. Our work contributes to the field of art history and preservation by offering a promising direction for research in the area of image recognition and deep learning. The potential for a more efficient and accurate method for artist attribution using such techniques is an exciting prospect for the field. We believe that our work can serve as a foundation for continued progress in this area and we look forward to future advancements that will further advance the field of art history and preservation.