IDENTIFY ARTIST FROM ART BASED ON NEURAL NETWORKS

6203 Machine Learning II Final Project

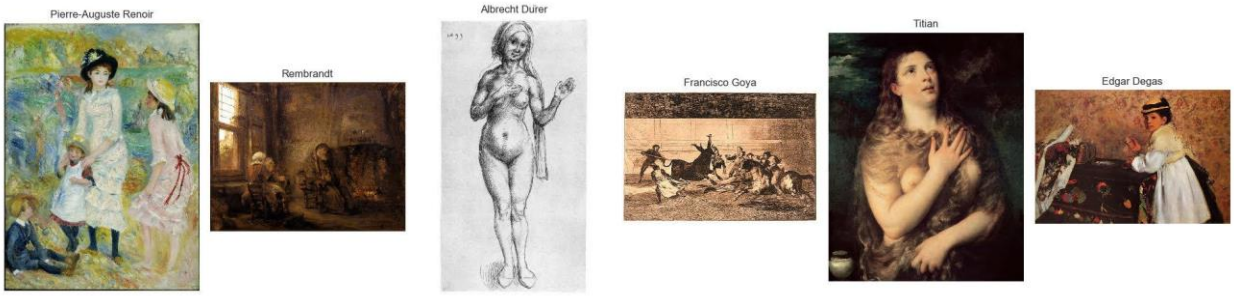
GROUP 3: SIQI JIANG, YUNYUN JIANG, JING LI

1. Introduction

1.1 Topic

Painting is a visual expression of thoughts lying deep within the mind, expression of emotions, perceptions and desires. Using the magic of Machine Learning to identify arts is what our project aims at. The datasets we use gathered a collection of artworks of the 50 most influential artists of all time.

Let's take a look at the following six pictures, can you tell it that they are from different artists just by eyeballing? Yes, most of us could get the confirmed answers by differentiating the color, style and pattern. How about the artists? Could you name any one of them?



As an art amateur, it is impossible to figure out who the artists are. Without the information printed on the top of the following six paintings, I could not even notice the first and the third paintings are both from Pablo Picasso, and the second and the fourth ones are from two different artists. From this project, our team makes use of the power of machine learning to create multiple models, like Multi-Layer Perceptome (MLP), convolutional neural networks (CNN), which would be able to identify the artist of paintings with a great accuracy up to 81%

1.2 Problem Statement

Build neural networks to recognize the corresponding artist of painting pictures based on the colors used and geometric patterns inside the pictures.

1.3 Dataset

This dataset contains three files:

- artists.csv: dataset of information for each artist
- images.zip: collection of images (full size), divided in folders and sequentially numbered
- resized.zip: same collection but images have been resized and extracted from folder structure

Use *resized.zip* allows you to download less data and process faster your model.

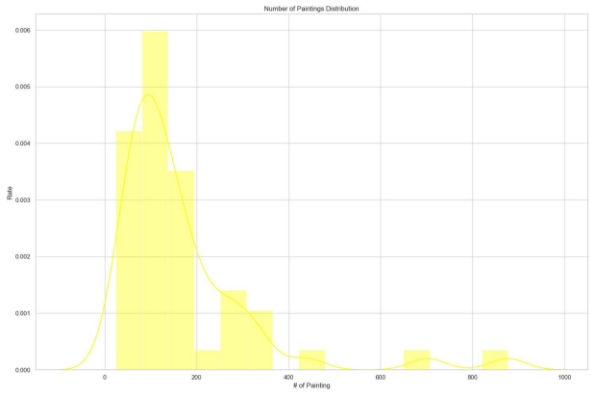
The columns listed in the csv file:

- Id
- name
- years
- genre
- nationality
- bio
- Wikipedia
- paintings

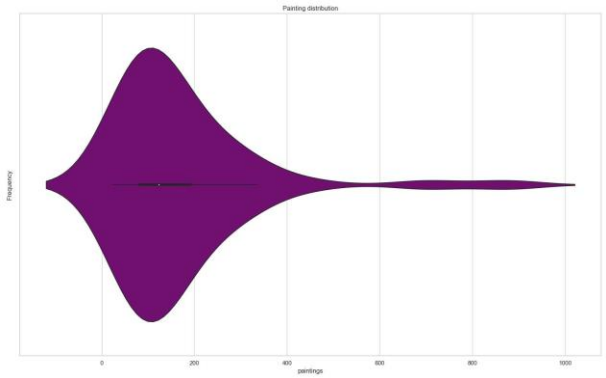
By breaking the column of years, we would be able to create another 3 variables: birth, death and age. Based on the lifetime of each artist, a new variable called age_group is created for further analysis.

2. Exploratory Data Analysis

2.1 Painting Analysis



Plot 2.1-A: Painting Distribution Plot



Plot 2.1-B: Painting Distribution ViolinPlot

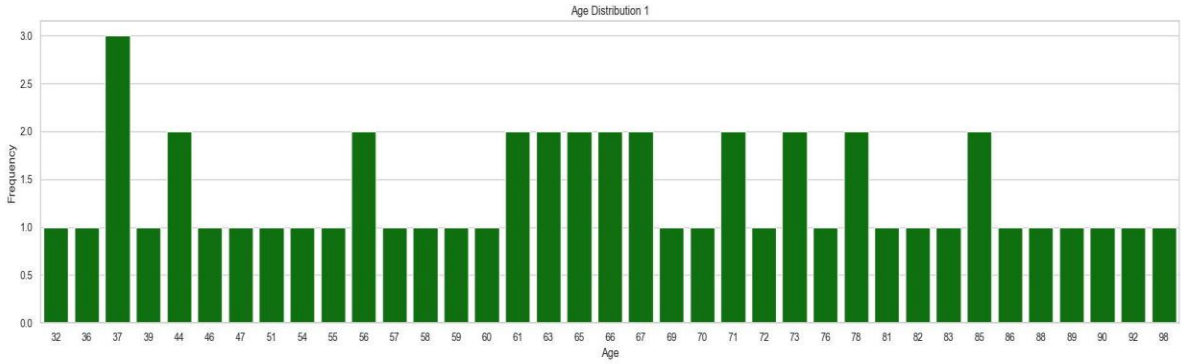
```
Painting Distribution:
 count     50.000000
mean     168.920000
std      157.451105
min       24.000000
25%       81.000000
50%      123.000000
75%      191.750000
max      877.000000
```

Table2. 1: Descriptive Statistics of paintings

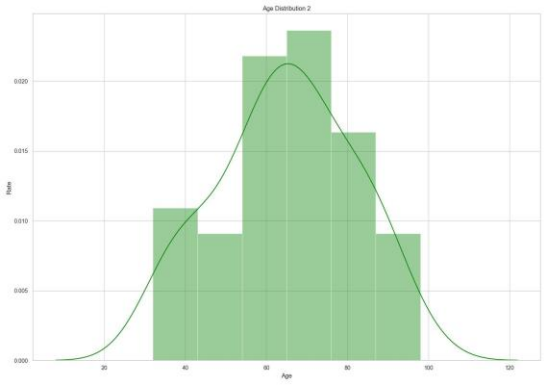
From the above two plots of painting distribution and descriptive statistics, it is clear that artists created 168 paintings averagely in their lifetime. The bigger number of paintings an artist created is 877 and the least is 24.

Sorting by the number of paintings in descending order, we pick up the top 10 artists who created the most paintings for the machine learning process. The top 1 artist with the most paintings is the most famous and influential figures in Western art, Dutch post-impressionist painter Vincent van Gogh. His paintings like landscapes, still life's, portraits and self-portraits, and are characterized by bold colors and dramatic, impulsive and expressive brushwork that contributed to the foundations of modern art.

2.2 Age Analysis



Plot 2.2-A: Age Frequency Plot

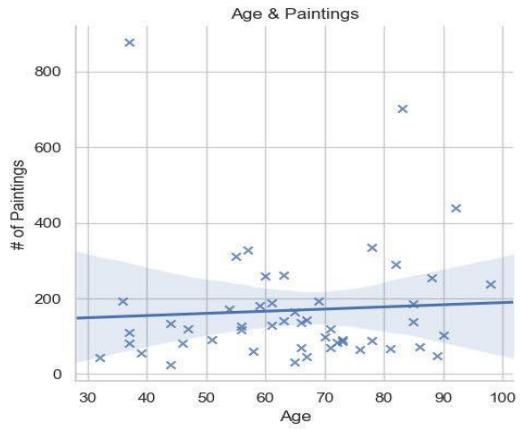


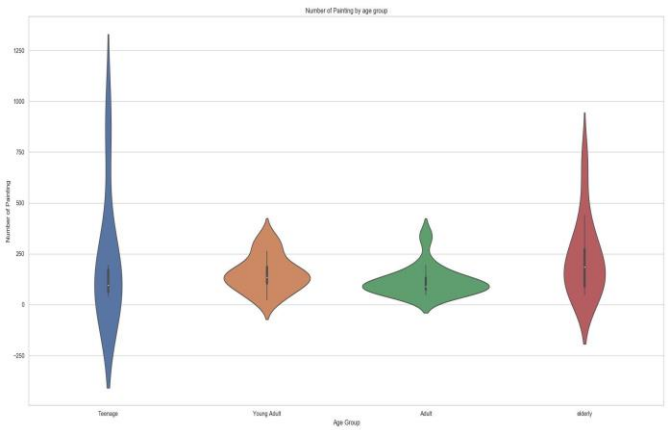
Plot 2.2-B: Age Distribution Plot

Age Distribution:	
mean	64.78000
std	16.74087
min	32.00000
25%	55.25000
50%	65.50000
75%	77.50000
max	98.00000

Table2. 2: Descriptive Statistics of age

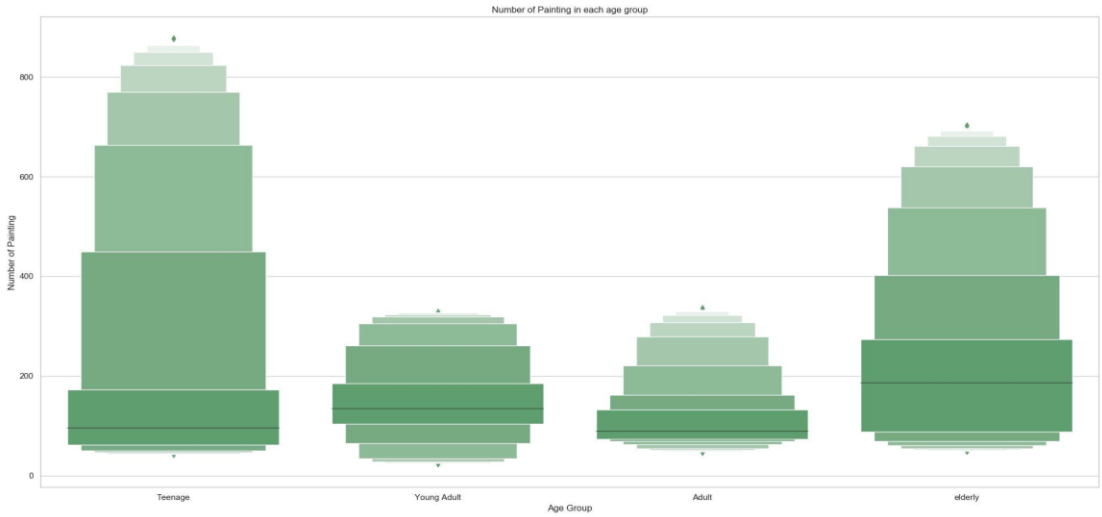
The distribution of artist age is nearly normal distribution given the above plots. The oldest artist was 98-year-old and one artist passed away at the young age of 32. We could create a new variable age_group based on the descriptive statistics table. However, we will define the age breakpoint based on social conventions.





Plot 2.2-C: Age VS. Paintings **Plot 2.2-D: Paintings Distribution by Age Group**

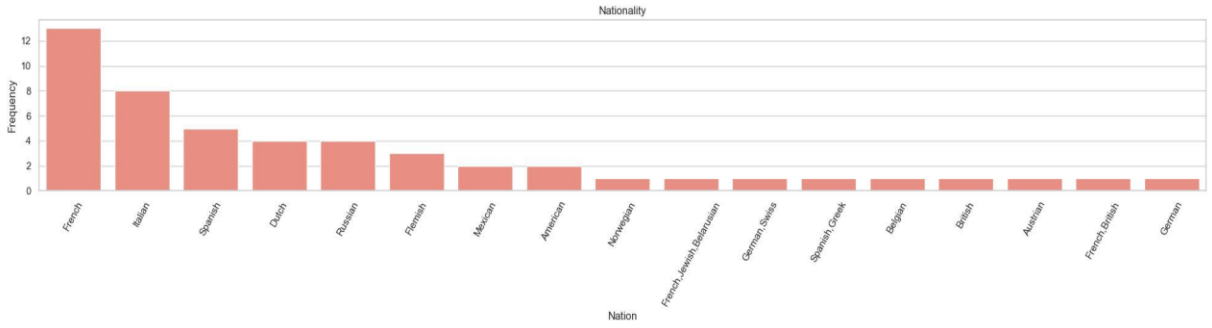
Plot 2.2C shows a simple linear regression of painting and age. The longer lifetime was, the more painting an artist could create, which is very straightforward. There are several outstanding outliers like Dutch artist Vincent van Gogh who created 877 paintings in a lifetime of 34 years, and French artist Edgar Degas creating 702 paintings in his 83 years lifetime. We will keep all for models.



Plot 2.2-E: Paintings Frequency by Age Group

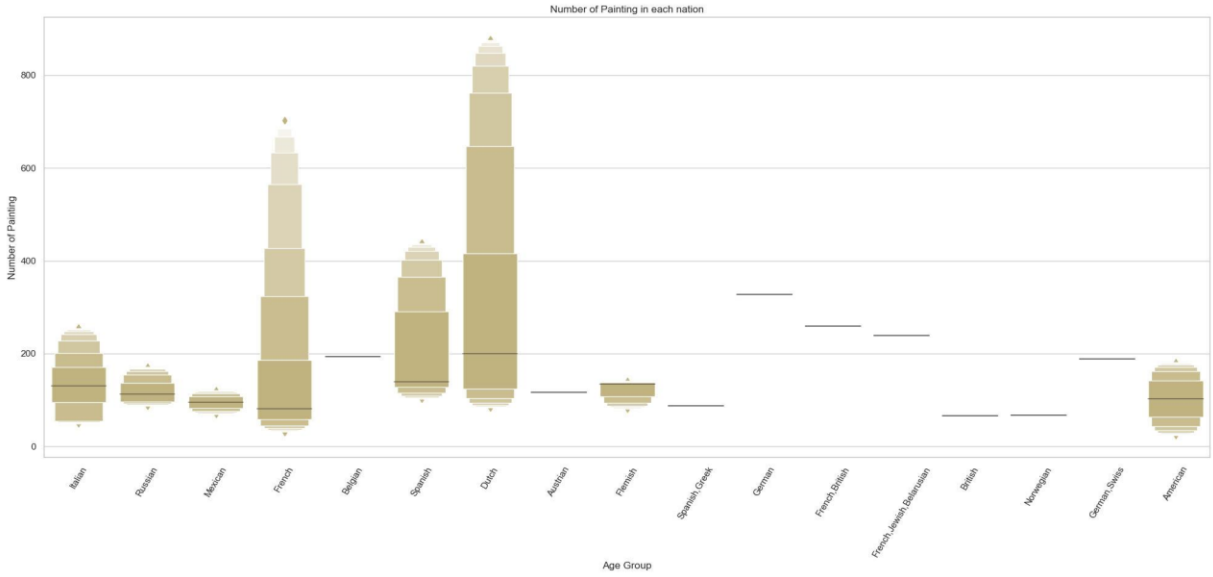
Plot 2.2 C and D are violinplot and boxenplot of painting distribution in each age group respectively. Both plots indicate that the range of painting numbers cross widely in senior and young adults' groups, which is definitely the effect made by the two famous artists Vincent van Gogh and Edgar Degas we mention before.

2.3 Nation Analysis



Plot 2.3-A: Nation Frequency

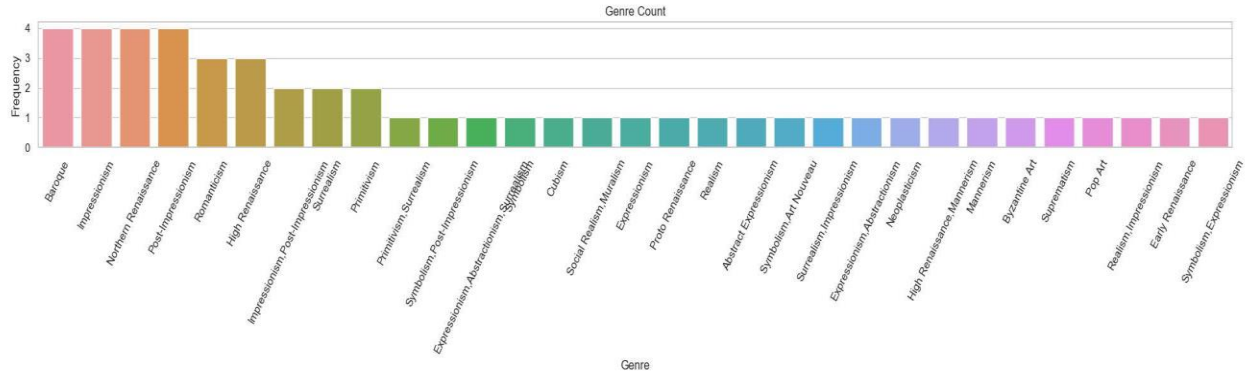
The top 3 countries with the most famous artists are French, Italian and Spanish, in such order. All are in Europe. It is easy to understand given the history of Renaissance art and painting, produced during the 14th, 15th, and 16th centuries in Europe under the combined influences of an increased awareness of nature, a revival of classical learning, and a more individualistic view of man.



Plot 2.3-B: Paintings by Nation

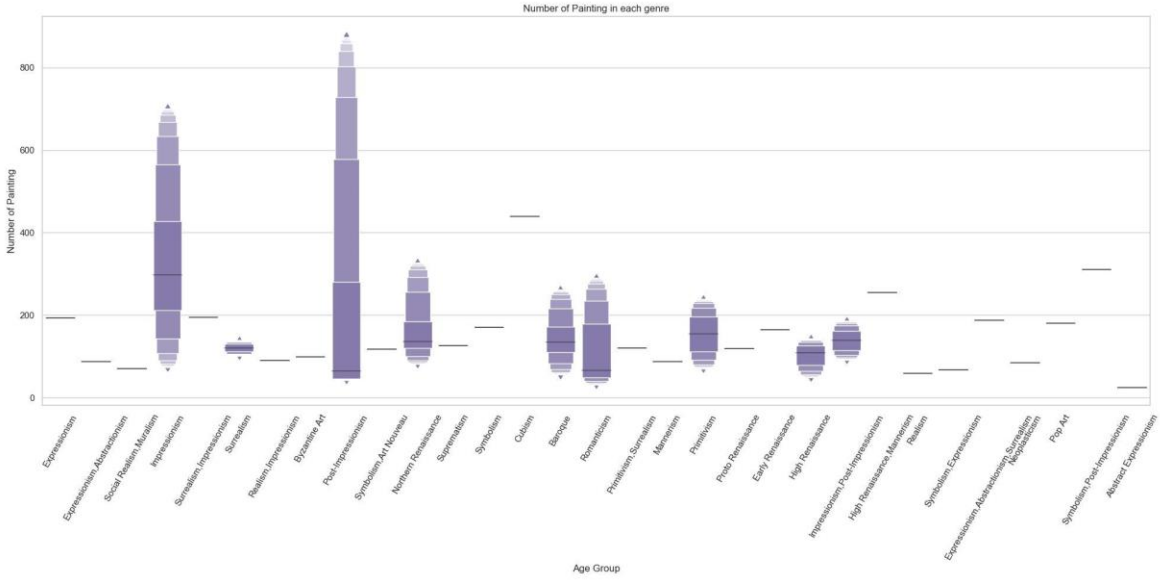
The boxenplot proves what we find out from the frequency table. French, Italian and Spanish have more artists than other counties. The outlier of Vincent van Gogh and Edgar Degas stand out in this plot as well.

2.4 Genre Analysis



Plot 2.4-A: Genre Frequency

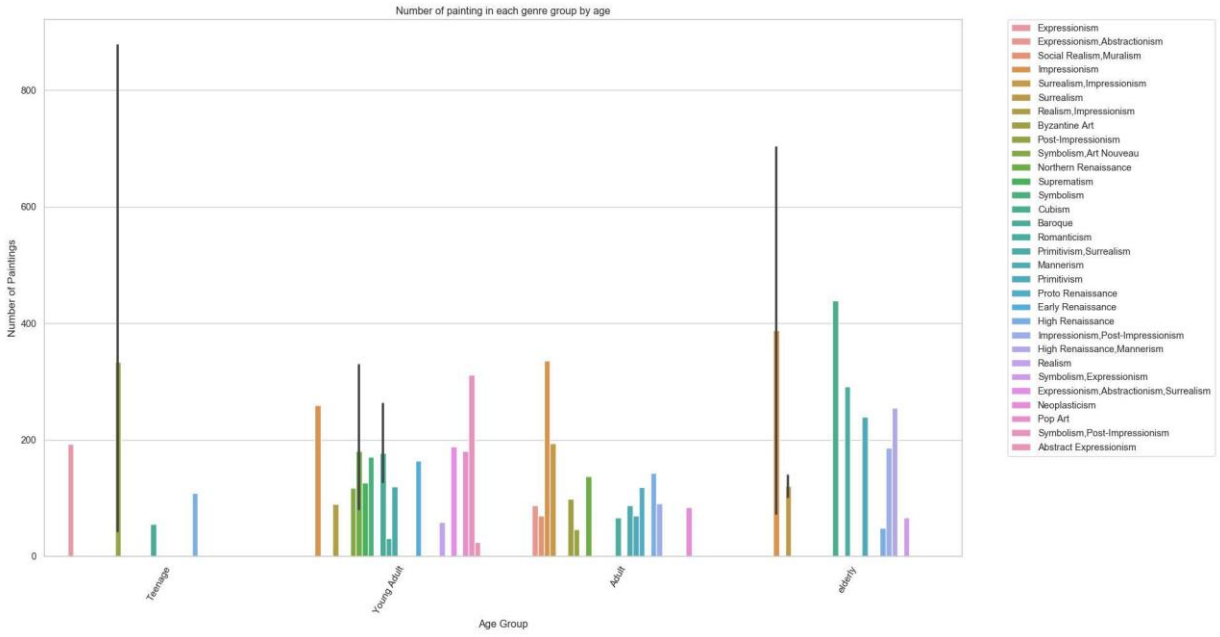
Here are the artists counted by genre. It is nice to see that we have so many different kinds of art genre count up to 24. There are 4 artists in Baroque, Impressionism, Northern Renaissance and Post-Impressionism each.



Plot 2.4-B: Paintings by Genre

Vincent van Gogh was a Dutch post-impressionist painter who is among the most famous and influential figures in the history of Western art and created 877 oil paintings, most of which date from the last two years of his life. So, genre of post-impressionist in this boxenplot is the tallest.

The second tallest bar is the genre of Impression, which has something to do with Edgar Degas who we keep mentioning. Edgar Degas was a French impression artist famous for his pastel drawings and oil paintings of ballerinas.



Plot 2.4-C: Number of Painting by Genre in each Age Group

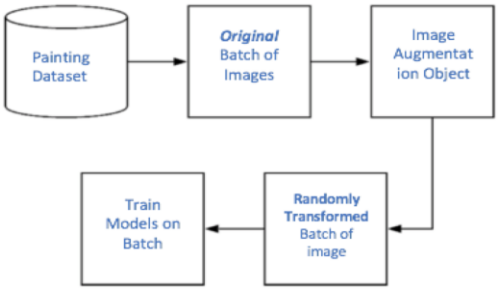
There are way more different kinds of genre among the young adult and adult artist groups, compared to teenage and elderly groups. The most popular genres of impression and Baroque could be found in all 3 age groups except teenagers.

3. Preprocessing

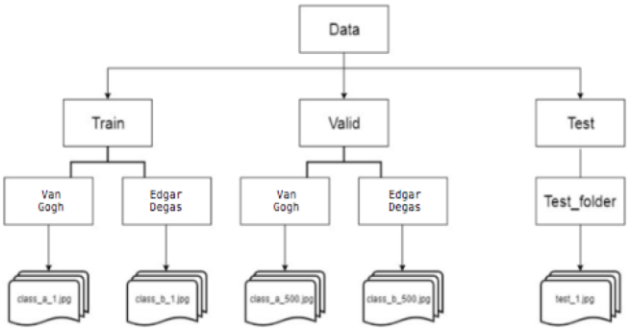
Image Augmentation

Since the training has the imbalance dataset that the largest class is almost 4 times larger than the smallest class, data augmentation was implemented to balance the training set and add more training samples for the under-represented classes. The augmentation was carried out by changes in scale, rotations, shearing, and horizontal & vertical flips etc.

The augmented images were created using the ImageDataGenerator API in Keras. It is the in-place data augmentation that we ensure that our network, when trained, sees new variations of our data at each and every epoch.



Plot 3-A: Data augmentation with Keras



Plot 3-B: flow_from_directory method

We have also applied the flow_from_directory method in the ImageDataGenerator. It takes the path to a directory and generates batches of augmented data. We set up the directory to the path of the directory that contains the sub-directories of the respective classes. Each subdirectory is treated as a different class. The name of the class is inferred from the subdirectory name.

4. Deep Learning Network

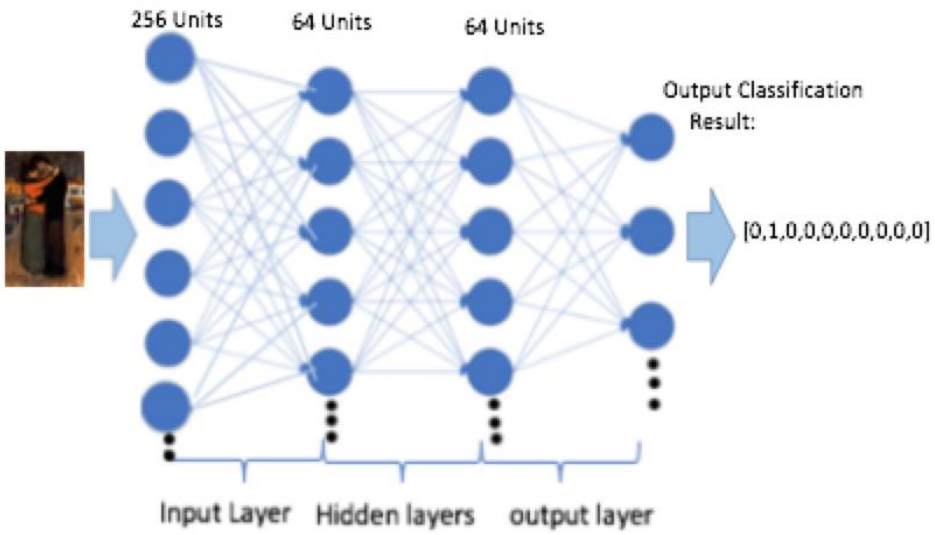
4.1 Model Set-up

There are paintings of 50 artists in the dataset. However, only 10 artists have more than 250 paintings available here. To reduce computation and better training, I decided to use the paintings of these 11 artists only.

name	paintings	class_weight
Vincent van Gogh	877	0.445631
Edgar Degas	702	0.556721
Pablo Picasso	439	0.890246
Pierre-Auguste Renoir	336	1.163149
Albrecht Dürer	328	1.191519
Paul Gauguin	311	1.256650
Francisco Goya	291	1.343018
Rembrandt	262	1.491672
Alfred Sisley	259	1.508951
Titian	255	1.532620

4.2 Multilayer Perceptron

To begin with the model, we chose Multi-layer perceptron as our baseline model. The model we used was a three-layer MLP network. The input dimension is 150528 (224*224*3). The number of neurons for each layer is 125, 64, and 32, respectively. The activation functions used in the hidden layers are Relu and the output layer is used the Softmax activation function, for the purpose of calculating the probability distribution of multi-class outputs. The optimizer is Adam, and the loss function is Categorical Cross Entropy, a function that is specifically targeted at categorical outputs.



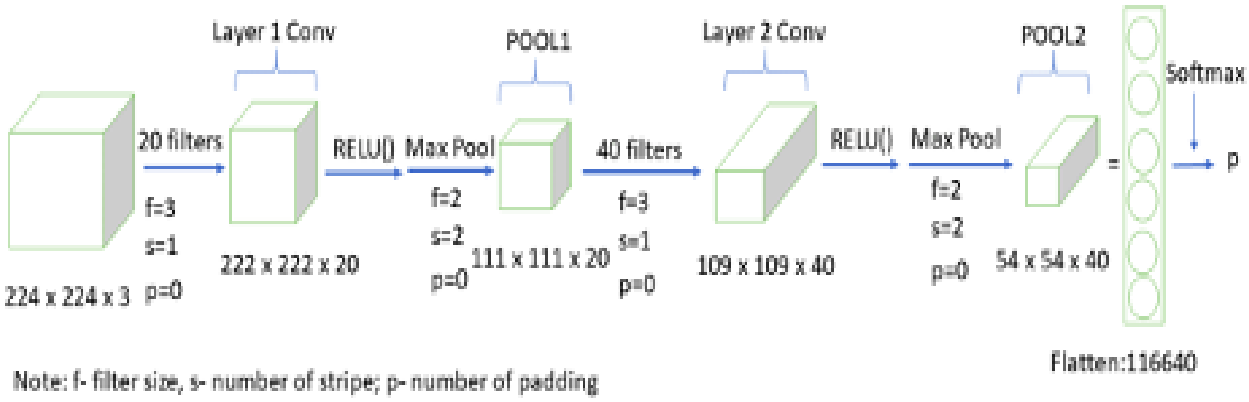

```
model = Sequential([
    Dense(N_NEURONS[0], input_dim=150528, kernel_initializer='uniform'),
    Activation("relu"),
    Dropout(DROPOUT),
    BatchNormalization()])
for n_neurons in N_NEURONS[1:]:
    model.add(Dense(n_neurons, activation="relu", kernel_initializer='uniform'))
    model.add(Dropout(DROPOUT, seed=SEED))
    model.add(BatchNormalization())
model.add(Dense(10, activation="softmax", kernel_initializer=weight_init))
model.compile(optimizer=SGD(lr=LR), loss="categorical_crossentropy", metrics=["accuracy"])
model.fit(x_train, y_train, batch_size=BATCH_SIZE, epochs=N_EPOCHS, validation_data=(x_test, y_test))
```

4.3 Convolution Neural Network

The second model we use is convolution neural network. We train and build the model in Pytorch framework. The self defined neural networks based four major functions in the network and created a ArtCNNclass based on master torch.nn.Module class.

- torch.nn.Conv2d() – applies convolution
- torch.nn.relu() – applies ReLU
- torch.nn.MaxPool2d() – applies max pooling
- torch.nn.Linear() – fully connected layer (multiply inputs by learned weights)

The forward () method computes a forward pass of the ArtCNN (Figure 1).




```
class ArtCNN(nn.Module):
    def __init__(self):
        super(ArtCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 20, (3, 3))  # output (n_examples, 16, 48, 48)
        self.convnorm1 = nn.BatchNorm2d(20)
        self.pool1 = nn.MaxPool2d((2, 2))  # output (n_examples, 16, 24, 24)
        self.conv2 = nn.Conv2d(20, 40, (3, 3))  # output (n_examples, 32, 22, 22)
        self.convnorm2 = nn.BatchNorm2d(40)
        self.pool2 = nn.AvgPool2d((2, 2))  # output (n_examples, 32, 11, 11)
        self.linear1 = nn.Linear(40*54*54, 250)  # input will be flattened to (n_examples, 32 * 5 * 5)
        self.linear1_bn = nn.BatchNorm1d(250)
        self.drop = nn.Dropout(DROPOUT)
        self.linear2 = nn.Linear(250, 10)
        self.act = torch.relu

    def forward(self, x):
        #x = x.reshape(1,-1)
        x = self.pool1(self.convnorm1(self.act(self.conv1(x.float()))))  #### first layer ###
        x = self.pool2(self.convnorm2(self.act(self.conv2(x.float()))))  #### second layer ###
        x = self.drop(self.linear1_bn(self.act(self.linear1(x.view(len(x), -1)))))  # fully connected layer ###
```

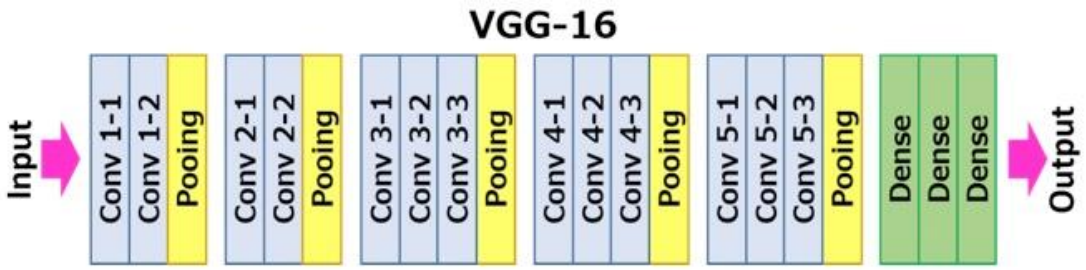
For CNN model training, we use Cross-Entropy loss as our loss function, Adam as the optimizer. We also specify the learning rate of 0.00005. Below also contains a function to train our ArtCNN model using a simple for loop. During each epoch of training, we pass the data to the model in batches and also calculate the loss on the test dataset.

4.4 Pretrained network

We chose VGG-16 and Resnet50 as our pre-trained models. Both models represent the architectures for convolutional neural networks.

VGG16

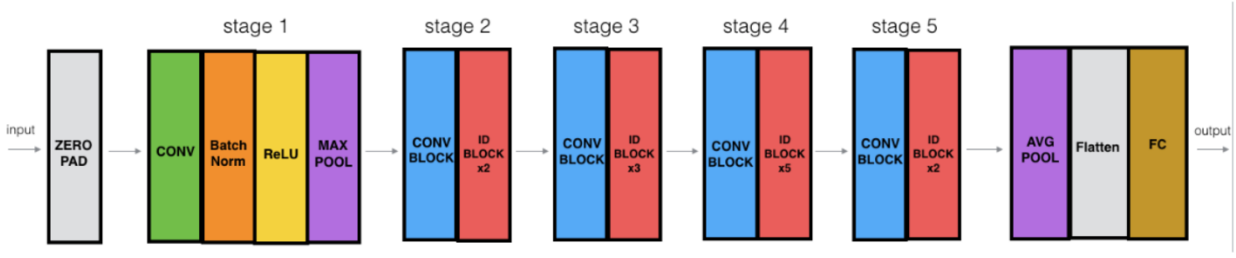
The first pre-trained network model we use is VGG16 from torchvision.models to build and train a new feed-forward classifier using those features. However, a new, untrained feed-forward network is used as a classifier, using ReLU activations and dropout. During the training, we also only update the weights of the feed-forward network not the model parameters from pretrained model.



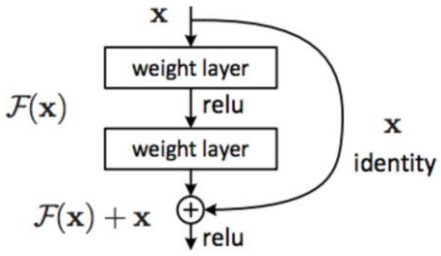

```
#  VGG16  network :
for parameter in model.parameters():
   parameter.requires_grad = False
from collections import OrderedDict
classifier = nn.Sequential(OrderedDict([('fc1', nn.Linear(25088, 5000)),
                                        ('relu', nn.ReLU()),
                                        ('drop', nn.Dropout(p=0.5)),
                                        ('fc2', nn.Linear(5000, 102))]))
```

ResNet

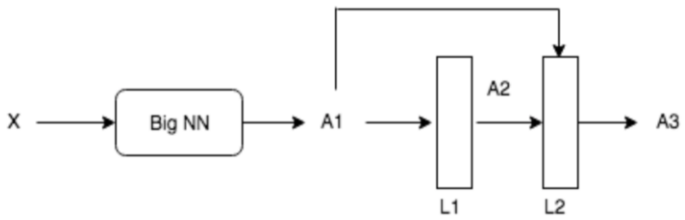
Resnet50 is similar to VGG-16. The unique feature of Resnet is the function of "skip connection", which allows the model to stack additional layers and build a deeper network. By using residual blocks in the network, one can construct networks of any depth with the hypothesis that new layers are actually helping to learn new underlying patterns in the input data. Generally speaking, the Resnet 50 is faster for the training.



ResNet-50 Model



Logical scheme of base building block



Term A1 will be passed to L2.

```
# Load pre-trained model
pre_trained_model = ResNet50(weights='imagenet', include_top=False, input_shape=model_input_shape)
# Add layers at the end
ResNet50_model = pre_trained_model.output
ResNet50_model = Flatten()(ResNet50_model)
N_NEURONS=(512,16)
for n_neurons in N_NEURONS[1:]:
    ResNet50_model = Dense(n_neurons, kernel_initializer='uniform')(ResNet50_model)
    ResNet50_model = BatchNormalization()(ResNet50_model)
    ResNet50_model = Activation('relu')(ResNet50_model)
output = Dense(n_classes, activation='softmax')(ResNet50_model)
model = Model(inputs=pre_trained_model.input, outputs=output)
optimizer = Adam(lr=0.0001)
# Compile the model
model.compile(loss='categorical_crossentropy',
              optimizer=optimizer,
              metrics=['accuracy'])
```

5. Building and training the Neural Network Model

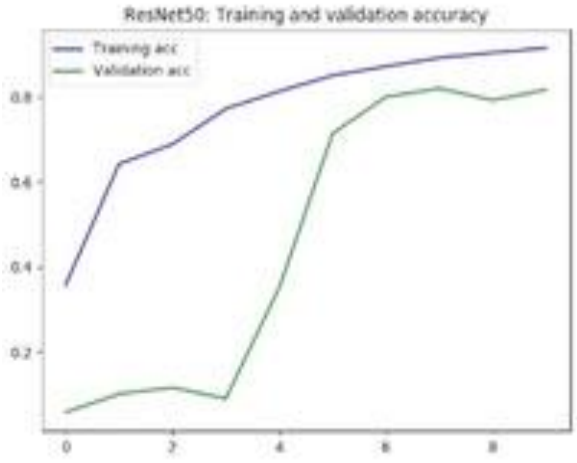
Hyperparameter Tuning and Weight calculation

Once we had established the model architectures and got the initial outcomes, we tried to improve the performance through hyperparameter searches. The following present the main hyperparameters adjusted.

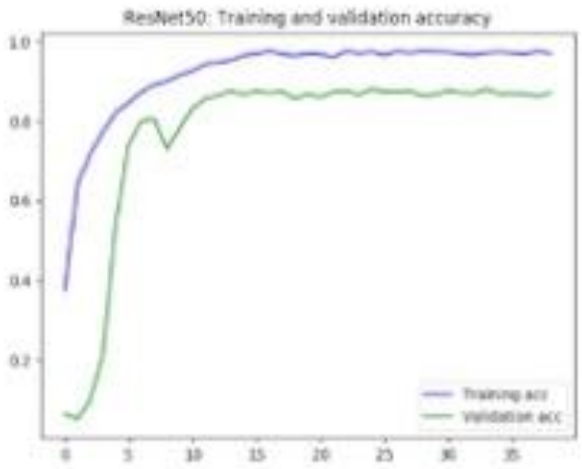
Batch Size: We started our model with batch size of 10. If we increase batch size to 64, the time used to train the pre-train model can be more than one hour. Therefore, for the concern of our computing time, we decided to limit the batch to 16, which we thought gave us relatively good performance.

Learning Rate: We used grid search to find the learning Rate and we ended with the best learning rate of 0.0001.

Epoch: Training the model for more iterations might improve the performance, at the cost of computation resource.



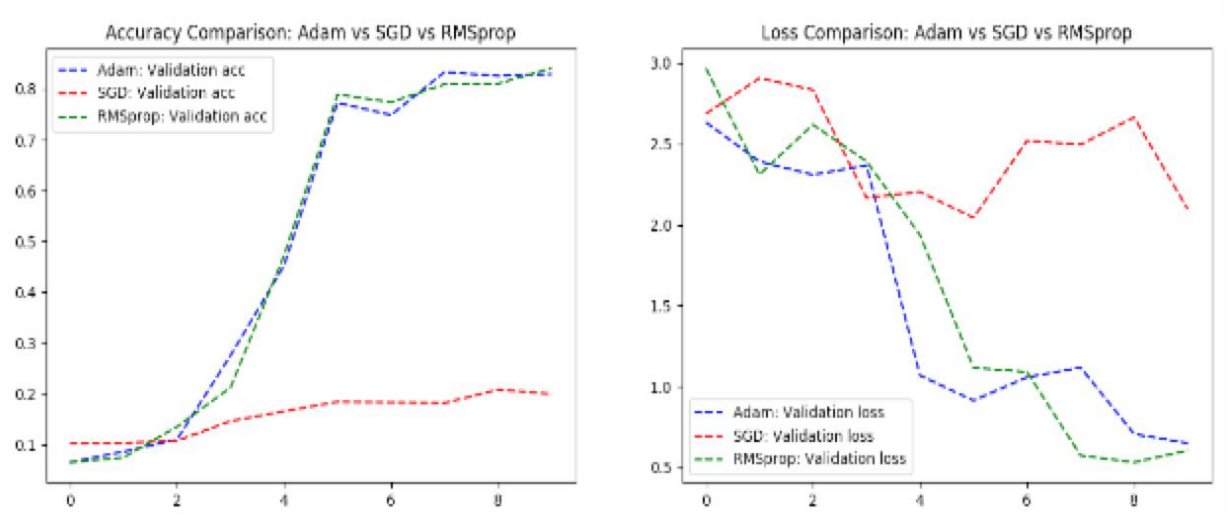
Training with 10 Epoch



Training with 40 Epoch

Optimization algorithm: We choose optimization algorithm instead of the classical stochastic gradient descent procedure to update network weights iteratively in training data due to its advantages: computational efficiency, being appropriate for problems with very noisy or sparse gradients and models that require hyper-parameters.

During each epoch of training, we pass the data to the model in batches and also calculate the loss on the validation dataset. From the figure below, the validation accuracy is higher for using Adam method than the SGD optimizer.



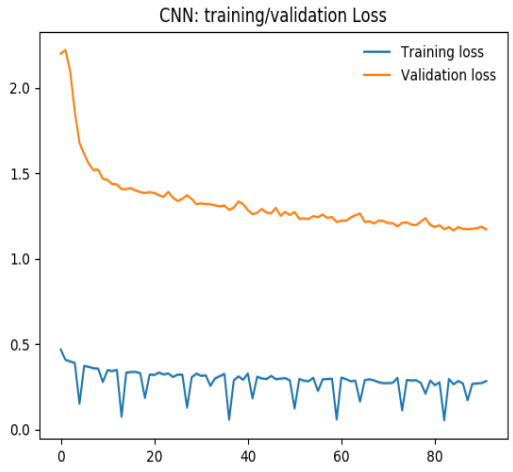
We also analyze the difference of performances between the three optimizers. Based on the graph, Adam learns the fastest and it is more stable than the other optimizers, it doesn't suffer any major decreases in accuracy.

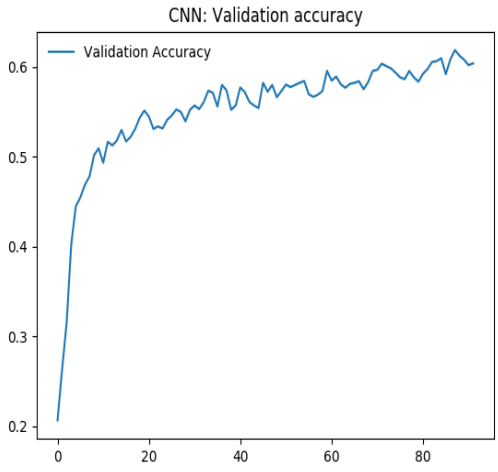
6. Performance Measurement

6.1 Training Loss and Validation Loss Over Time

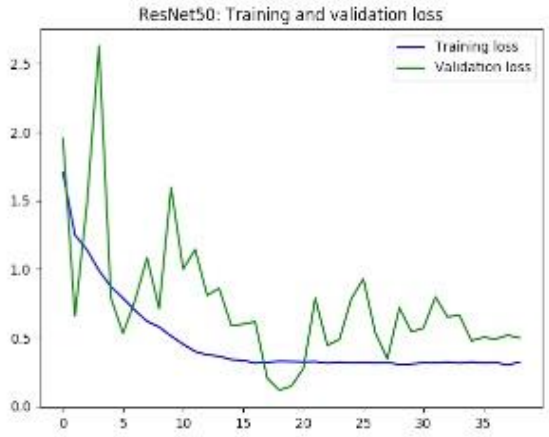
Both Resnet and VGG models have okay loss and accuracy curves. The validation loss decreases quickly from the beginning of epochs and becomes flattened toward the end of the epoch. And the training loss as expected is low. The validation accuracy increases with the epoch. Due to the time constraint, we were not able to train our models with more epochs. However, we still believe that the performance will be better with the training epochs increase.

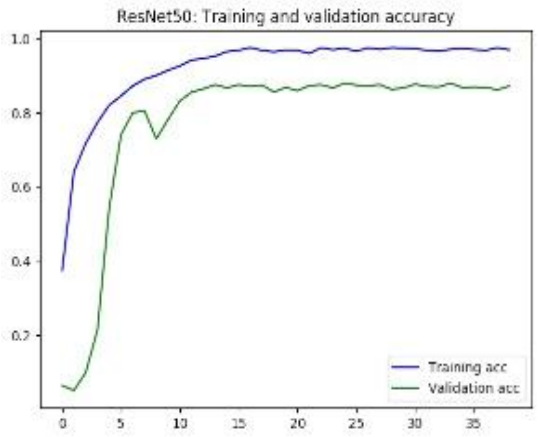
CNN



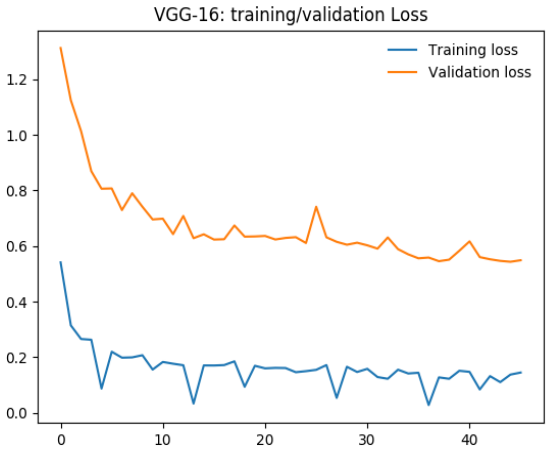


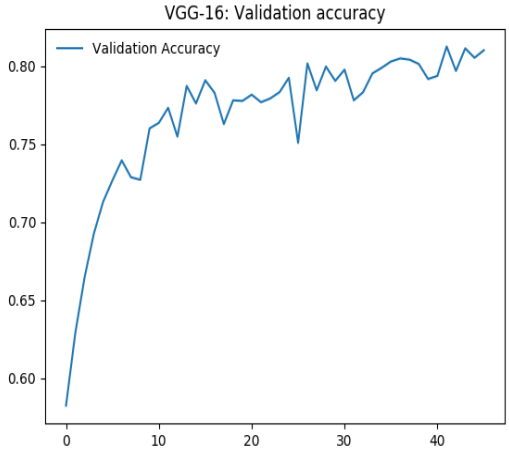
ResNet50





VGG16





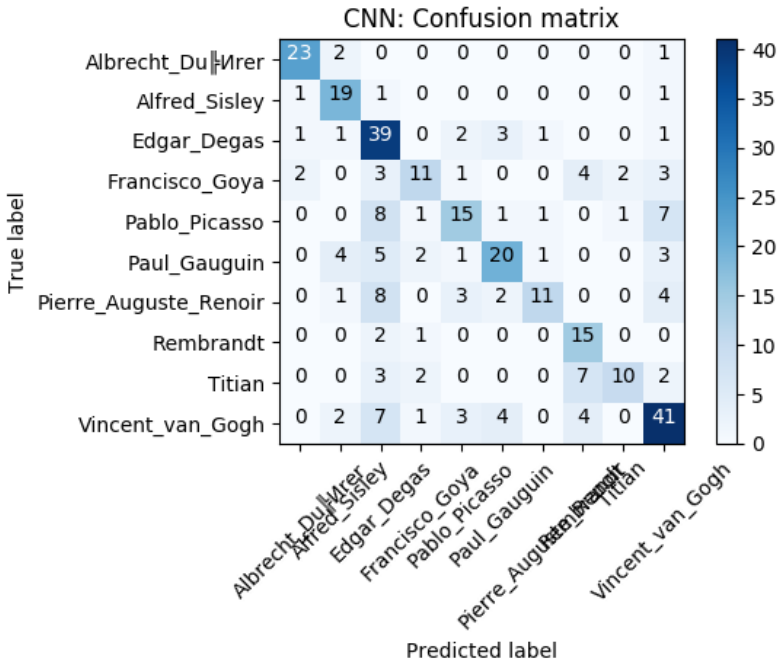
6.2 Model Accuracy

We compared the performance among 4 different models. The ResNet has best performance with accuracy rate of 87%, VGG of 80%, followed by CNN of 76% and MLP of 58%.

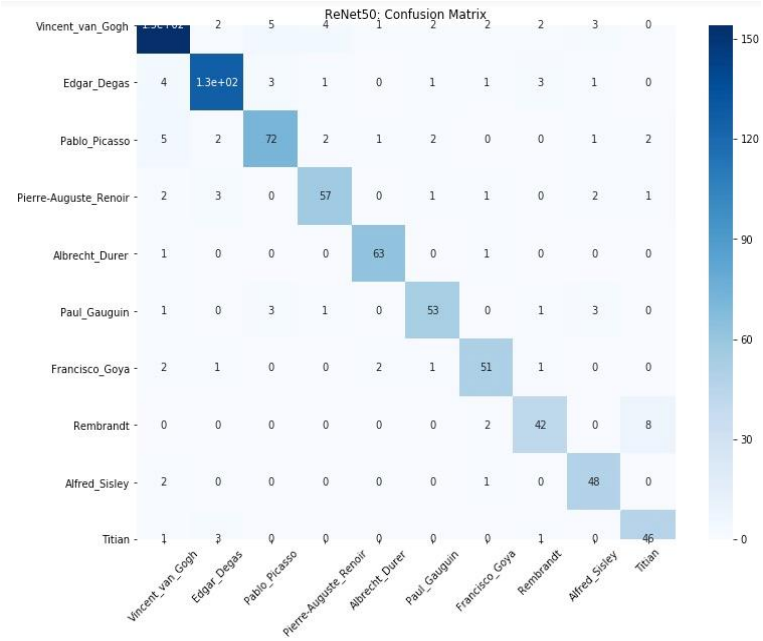
Model	Accuracy
MLP	0.58
CNN	0.64
ResNet	0.87
VGG16	0.81

We created a confusion matrix to describe the performance of a CNN classification model on the test dataset. The diagonal elements represent the number of points for which the predicted labels are equal to the true label, the off diagonal elements are those that are not equal.

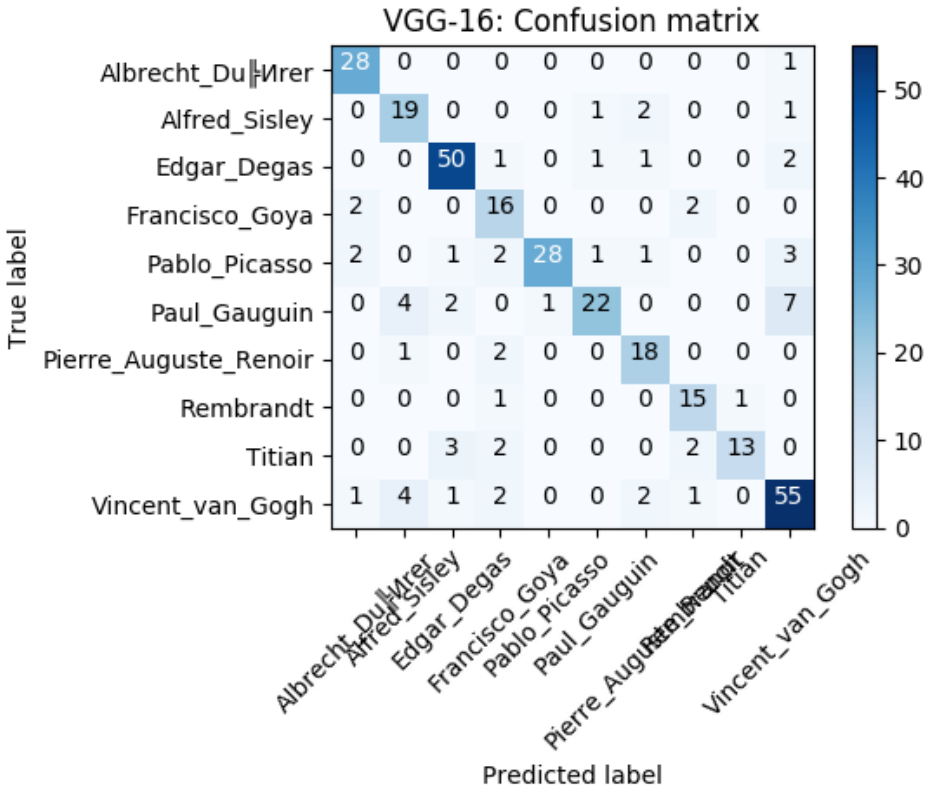
The performance of the CNN model is relatively well as 64% of the pairs of predicted and targeted labels are correctly matched.



For ResNet 50 model, where accuracy is 87%, the numbers on the diagonal line are very high, which reflects the high accuracy.



For VGG-16 model, where the accuracy is 81%, the numbers on the diagonal line are very high, which reflects the high accuracy.



7. Conclusion

In this project, we successfully built four networks that help us to recognize the artist of a painting picture based on the colors used and geometric patterns. Overall, pretrained models have significantly better performance than the self-designed network. We will use the trained network for inference. That is, we will pass a new image into the network and predict the corresponding artist of the painting in the image.

To further improve model performance, we would like to implement more epochs to train our models. And, we also hope to apply model ensemble techniques to incorporate the strength from our models. Last but not least, we could use text mining from the word document provided in the dataset to help better predict the painting author.

8. Reference

https://www.kaggle.com/ikarus777/best-artworks-of-all-time

https://help.healthycities.org/hc/en-us/articles/219556208-How-are-the-different-age-groups-defined-

https://keras.io/preprocessing/image/

https://keras.io/applications/#vgg16

https://keras.io/applications/#resnet

https://en.wikipedia.org/wiki/Convolutional_neural_network

https://en.wikipedia.org/wiki/Multilayer_perceptron

https://wiki.python.org/moin/PyQt4

https://scikit-learn.org/stable/

http://en.wikipedia.org/wiki/Vincent_van_Gogh

http://en.wikipedia.org/wiki/Pablo_Picasso

http://en.wikipedia.org/wiki/Francisco_Goya

http://en.wikipedia.org/wiki/Edgar_Degas

9. Appendix

For this project, personal laptops will be used. If more computation power is required then the project will use a cloud platform like Amazon Web Services or Google public cloud.

- Machine Configuration: I5 Intel chip and 64 GB RAM
- SDK: Pycharm Community Version
- Python: 3.6
- Numpy and pandas
- MatplotLib
- Keras
- Pytorch
- Sci-kit
- PyQt4