The Museum Scanner

Using Convolutional Neural Networks to Identify Artists

By Eduardo Beltrán Herrera

IBM Machine Learning Specialization

10/03/2021

Introduction

Convolutional Neural Networks are particularly effective at classification tasks using images. By learning to identify simple patterns and shapes, CNNs are able to learn the visual elements that form a car, a dog or a cat.

It is interesting to ponder if, by learning these shapes and patterns, CNNs are able to identify art styles, and the artists behind them. This exercise tries to implement a neural net capable of identifying the artist behind some of the most important pieces of art in human history.

The dataset used was obtained from Kaggle (Best Artworks of All Time | Kaggle) and contains a collection of artworks from 50 different artists; to reduce computational time for this project, only the six artists with most pieces of art were considered.

Data cleansing and transformation

The full dataset contains hundreds of artworks from 50 different artists, as well as general information about the artist (nationality, art movement, etc). For the purposes of this project, only the artist name was considered as the target for the neural net.

Figure 1. **Random sample of artworks in the dataset**

The following artists were pulled from the original dataset, as they had the biggest amount of artworks in the set. To balance the training of the neural net, a weight for each artist was calculated.

name	paintings	class_weight
Vincent van Gogh	877	0.568795
Edgar Degas	702	0.710589
Pablo Picasso	439	1.136295
Pierre-Auguste Renoir	336	1.484623
Albrecht Dürer	328	1.520833
Paul Gauguin	311	1.603966

Table 1. **Selected artists, the number of paintings and their weights**

The paintings themselves come in many different sizes. Because the neural net requires a fixed dimension for the input, all pictures were resized to a 224x224 matrix. Additionally, horizontal and vertical flipped images were added to the dataset to better generalize the model:

An original Image of Albrecht Dürer

A transformed Image of Albrecht Dürer

Modeling

The following CNN structure was implemented, loosely based on the *AlexNet* architecture:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 56, 56, 96)        34944
_________________________________________________________________
activation_5 (Activation)    (None, 56, 56, 96)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 27, 27, 96)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 27, 27, 256)       614656
_________________________________________________________________
activation_6 (Activation)    (None, 27, 27, 256)       0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 256)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 13, 13, 384)       885120
_________________________________________________________________
activation_7 (Activation)    (None, 13, 13, 384)       0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 4, 4, 384)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6144)              0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               3146240
_________________________________________________________________
dense_4 (Dense)              (None, 512)               262656
_________________________________________________________________
activation_8 (Activation)    (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 3078
_________________________________________________________________
activation_9 (Activation)    (None, 6)                 0
=================================================================
Total params: 4,946,694
Trainable params: 4,946,694
Non-trainable params: 0
```

The same architecture was tested with three different optimizer settings: two different learning rates for the ADAM optimizer and a single value for RMSdrop.

Test number	**Optimizer**	**Hyperparameters**
1	Adam	Lr = 0.0001
2	Adam	Lr = 0.0005
3	RMSdrop	Lr = 0.0005

Table 2. **List of tested CNNs configurations**

All three models were trained for 10 epochs, with **accuracy** as the metric and **categorical crossentropy** as the loss.

Analysis

The following graphs show the accuracy and loss changes per epoch, across all three models.

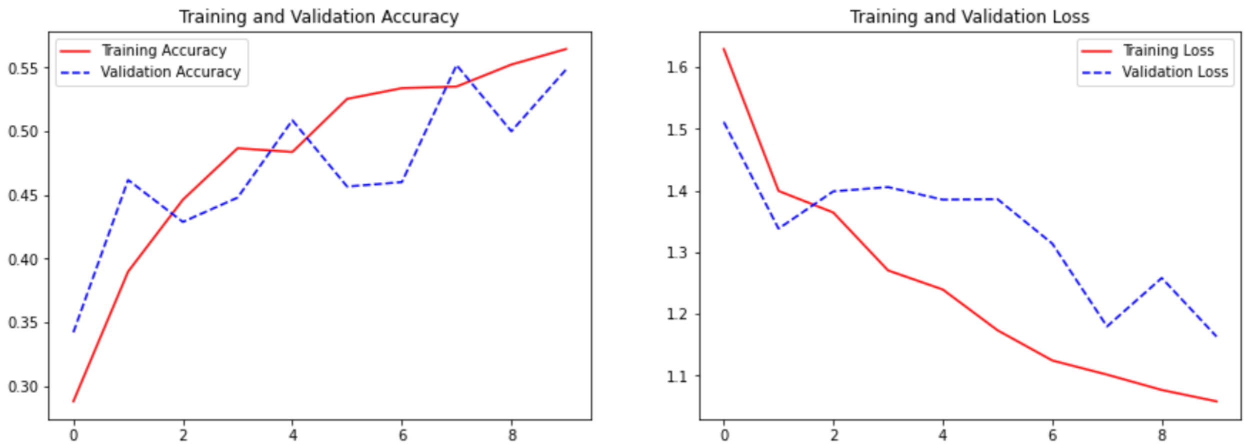


Figure 1. **Adam (lr = 0.0001)**

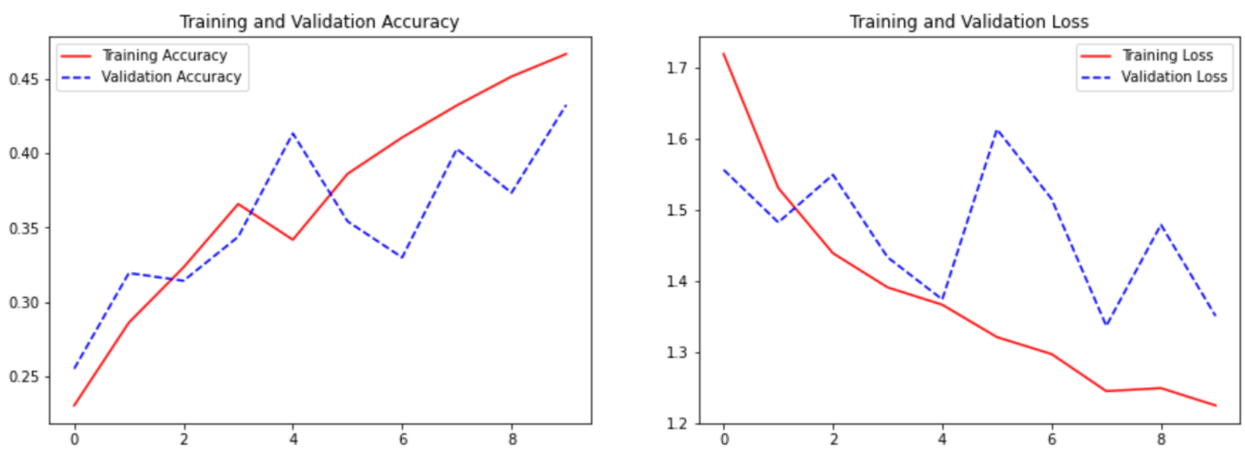


Figure 2. **Adam (lr = 0.0005)**

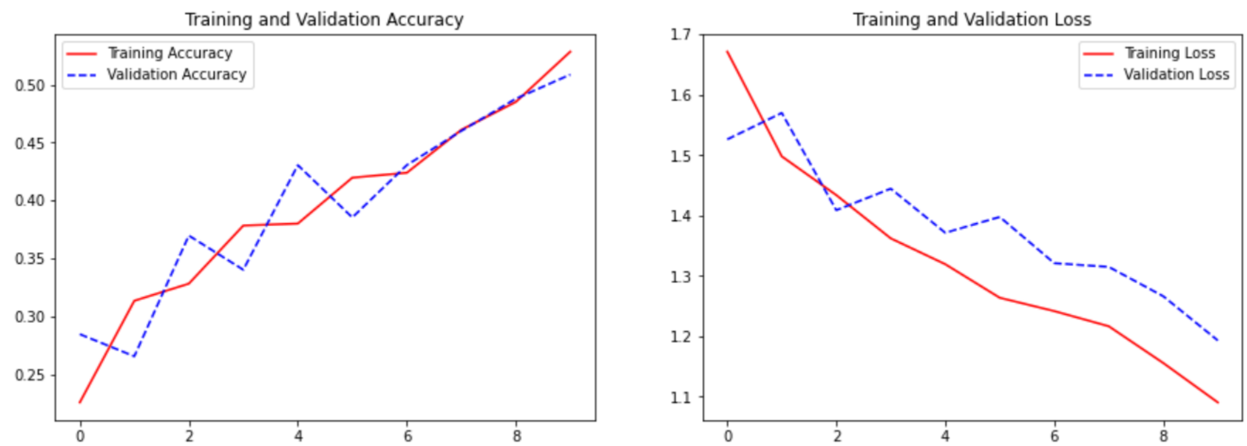
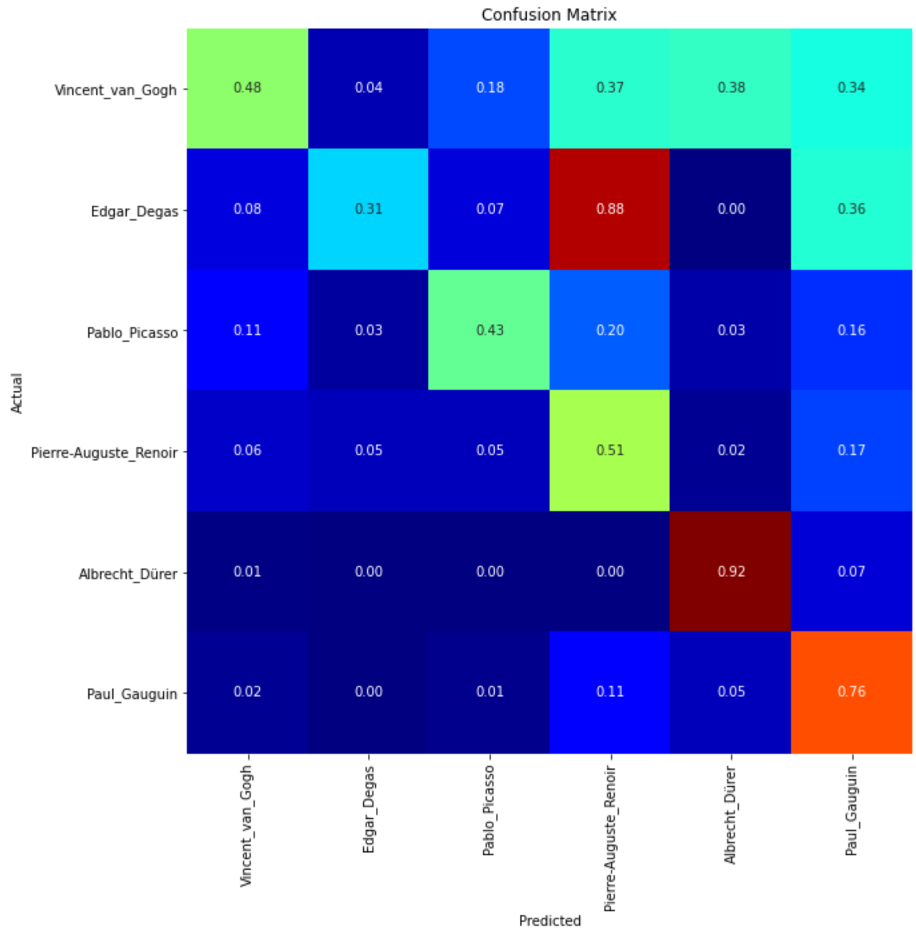


Figure 3. **RMSdrop (lr = 0.0005)**

Test number	Optimizer	Hyperparameters	Validation accuracy (10 epochs)
1	Adam	Lr = 0.0001	0.5486
2	Adam	Lr = 0.0005	0.43
3	RMSdrop	Lr = 0.0005	0.5087

From the previous figures, we can determine that all three models could use more training, as the curves for both accuracy and loss have not yet plateaued. It seems all models performed comparably on the accuracy. Since the training and validation metrics are not too far apart, we can determine that there is no overfitting in our model.

By observing the confusion matrix on all three models, we can determine which artists the models have trouble discerning:



It seems that **Albrecht Dürer** is the easiest artist to identify (0.92 accuracy), and that **Edgar Degas** and **Pierre-August Renoir** are hard to tell apart.

Conclusion

It seems the model with the ADAM optimizer with a learning rate of 0.0001 did the best out of all models-

As discussed previously, all three previous models can be improved by increasing the training time. Models that are more complex could be built to try to increase the overall accuracy in the early stages of training. Finally, pre-trained models could be used to try and accelerate the training of the latter layers of the models and improve accuracy overall.