Who Painted This? Van Gogh, Dali, or Picasso?

Alasia Buschkopf
University of Missouri-St. Louis

May 6, 2024

Abstract

Introduction Vincent Van Gogh, Salvador Dali, and Pablo Picasso painted thousands of works between them and each claims to have been influenced by their predecessor. With similar yet distinct styles drawing from each other, can a convolutional neural network learn to classify these paintings by their artist?

Methods Paintings for each artist were downloaded from wikiart.com in batches. Around 770 images per artist, for a total of over 2,000 images in the dataset. Images were reviewed for quality and accuracy, randomly shuffled, and split into a development and testing dataset (80%/20% split). The development set was further split into a training and validation dataset (80%/20% split). The images were resized, normalized and used to train convolutional neural networks. Various architectures, parameters, and hyperparameters were trained and compared to obtain the best fit model. Once a best fit model was trained, further experiments using augmented data and regularization methods were performed to evaluate the impact on model accuracy. Finally, a pre-trained model (Google's Teachable Machine) was evaluated to determine how it compared to the best fit model.

Results All three artists were evenly represented in the dataset, meaning the baseline accuracy for this classification would be around 33.33%. The best fit model obtained 97.80% accuracy on the test data set. Attempts to improve the model using data augmentation and regularization methods did not result in improved accuracy on the validation dataset. Google's Teachable Machine was able to create a model which achieved between 81% and 89% accuracy between the three classes.

Discussion A best fit convolutional neural network model was developed that could classify paintings by artist with 97.80% accuracy on the test dataset. The baseline accuracy for this dataset was 33.33% and the accuracy obtained by Google's Teachable Machine ranged from 81%-89%, which indicates this is a highly accurate model.

Contents

List of Tables

List of Figures

1 Introduction

Vincent Van Gogh, Salvador Dali, and Pablo Picasso are three well known European artists from the late 19th and early 20th century. Dali claims to have been influenced by Picasso and Picasso claims to have been influenced by Van Gogh, yet they all have unique artistic styles. In light of the influence these painters had on each other and similar subjects in the paintings, it serves as an interesting dataset to attempt to build a multi-class classification convolutional neural network.

2 Methods

2.1 Data Collection

Using wikiart.com, images of paintings from Van Gogh, Picasso, and Dali were downloaded from WikiArt [1] and reviewed to ensure all images in the dataset were paintings and high enough quality images. The final data set includes 2,314 painting images, evenly distributed between the three artists (see **Table 1**). Examples of the images represented in the dataset are also shown (see **Figure 1**) .

Table 1: A table containing the distribution of paintings in the dataset

Artist	Frequency	Percent of Dataset
Van Gogh	773	33.3%
Picasso	770	33.3%
Dali	771	33.3%

2.2 Data Preprocessing and Splitting

The images were reshaped to 256 x 256 scaling using Keras Preprocessing ImageGenerator and the pixel values were divided by 255. Images were mostly RGB, with a few images black and white from the source. Data was split into a development and test set. The images were randomly shuffled, and 20% was set aside for the test dataset. The remaining 80% was used as the development dataset. Within the development dataset, another 80%/20% split was done to create a training and validation dataset (see **Table 2**).

Table 2: A table containing the distribution of paintings in the development and testing dataset

Artist	Training Data Images	Validation Data Images	Test Data Images
Van Gogh	495	125	153
Picasso	495	125	150
Dali	495	125	151

Figure 1: Representative Images of Dataset

3 Modeling and Results

3.1 Model Overfitting

The first step in creating a generalized model is to overfit your model. When a model is overfit, it has learned the training dataset so well that it begins to perform poorly on the validation dataset. Once you have reached overfitting, the model needs to be fine tuned to reach generalization. The first attempt to reach overfitting utilized a CNN model architecture with three Conv2D layers with kernel size 5, activation relu. Each convlutional layer also used a MaxPool2D layer with a 4x4 filter. Then a dense layer with 10 filters and activation relu before the final dense layer with 3 filters and softmax activation. Softmax activation was used in the final layer as this is a multi-class classification problem. Adam was used as the optimizer and categorical-cross entropy was used as the loss function. The batch size was set to 64 and 100 epochs called for training. However, in order to reduce unnecessary computing time, two Keras callback functions were used. One for early stopping and one for model checkpointing. The early stopping function caused the model to stop training at epoch 40 when the validation loss consistently did not improve from 0.28737.

The overfit model reached an accuracy of 0.9067 on the validation set. The learning curves for the overfit models show that the accuracy, precision, and recall on the training data was continuing to climb while the validation accuracy, precision, and recall were all leveling out. (see

Figure 2). This shows that the model has overfit and will not be accurate when applied to the unseen test dataset. At this point, it is time to begin working to generalize the model.

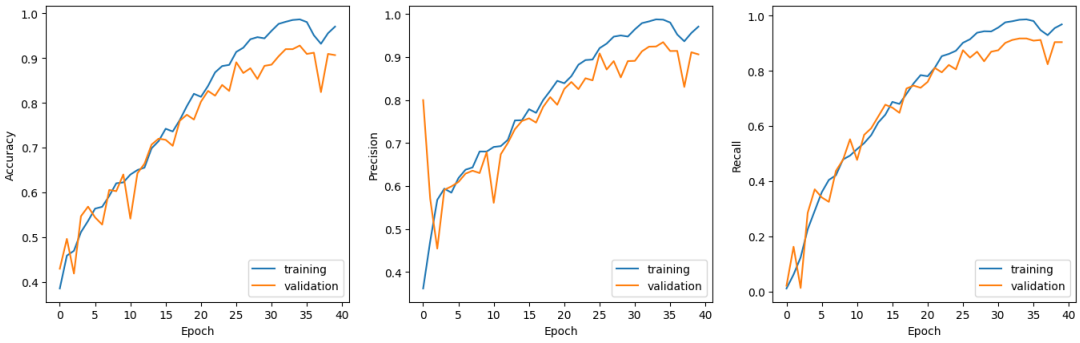

Figure 2: Overfit Model Learning Curves

3.2 Model Fitting

A variety of modifications to the CNN architecture were tested in order to find the best fit model. Some modifications tested included altering the number of convolutional layers and altering the number of convolutional filters in each of the convolutional layers (see **Table 3**). Again, in order to reduce unnecessary computing time and prevent over-fitting, the same early callback functions were used.

Table 3: A table showing how changing the parameters affected the model metrics

Metric - On Validation Dataset	Increased Convolutional Layers	Decreased Convolutional Filters
Accuracy	64.53%	34.40%
Loss	82.74%	110.01%
Recall	56.27%	1.33%
Precision	68.51%	38.46%

After a few iterations of modifications, the best fit model was determined to be a model with only 2 convolutional layers and 2 dense layers. (see **Figure 3** and **Figure 4**). The first convolutional layer had 32 filters with a kernel size of 3. The second convolutional layer had 16 filters with a kernel size of 3. Each layer also had a layer of max pooling with a 4 by 4 filter. The dense layers contained 10 and 3 filters. Relu was used as the activation methods for all layers except the final layer which used softmax. The activation method was Adam and the loss function was categorical cross entropy.
Early stopping with a patience of 15 epochs caused this model to stop training after 39 epochs (see **Figure 5**). With an accuracy of 98.13% on the validation dataset, this model was the highest performing of any iteration tested this far. Reaching the conclusion this model was the best fit, the isolated test dataset was used to evaluate the model and it performed with an

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2  (None, 63, 63, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 61, 61, 16)        4624

 max_pooling2d_1 (MaxPoolin  (None, 15, 15, 16)        0
 g2D)

 flatten (Flatten)           (None, 3600)              0

 dense (Dense)               (None, 10)                36010

 dense_1 (Dense)             (None, 3)                 33

=================================================================
Total params: 41563 (162.36 KB)
Trainable params: 41563 (162.36 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3: Best Fit Model Summary

accuracy of 97.80% on the test dataset (see **Table 4**). The baseline accuracy for a multi-class classification dataset with three evenly distributed classes is 33.33%, so this model performed very well compared to baseline accuracy.

Table 4: A table containing the best fit model metrics on the validation and test datasets

Metric	Validation Dataset	Test Dataset
Accuracy	98.13%	97.80%
Loss	9.54%	10.07%
Recall	98.13%	97.80%
Precision	98.13%	97.80%

3.3 Effects of Data Augmentation

Two different types of data augmentation were utilized to see the impact this would have on the model - flipping images vertically and rotating images 40 degrees. While both augmentations had a negative affect on accuracy, rotating the images rendered the model essentially useless with an accuracy of 33.33% - the same as the baseline accuracy for this dataset (see **Table 5**).

3.4 Effects of Regularization Methods

In an attempt to increase the current best fit model's accuracy, two different regularization methods were applied - dropout and batch normalization. To implement dropout, a dropout layer of 0.25 was added between the two existing convolutional layers. This did not have a drastic impact on model accuracy, and ultimately resulted in a slightly lower accuracy than the

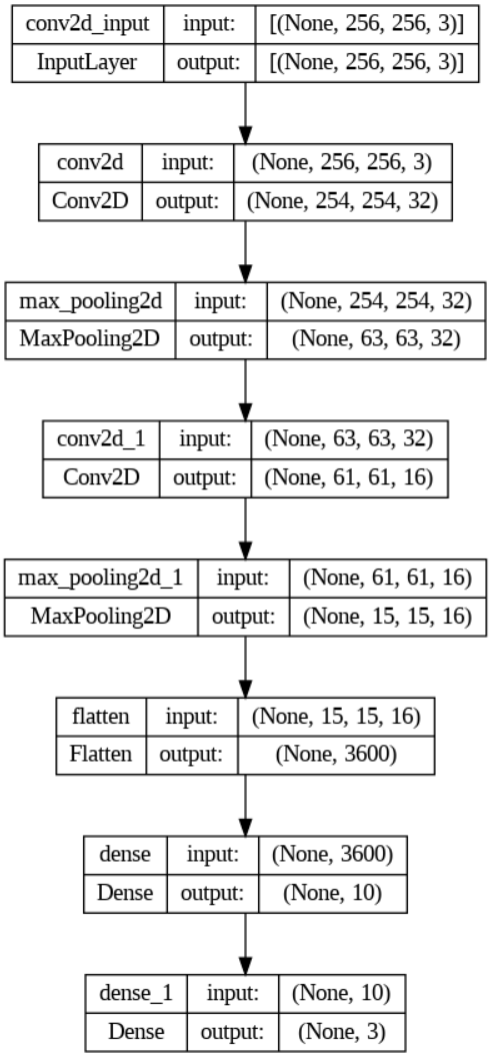

Figure 4: Best Fit Model Layers

same model not utilizing dropout. Batch normalization surprisingly caused a significant decrease in accuracy. (see **Table 6**). Because of these results, the previous best fit model was retained.

3.5 Using More Powerful Architecture

While the current best fit CNN was able to achieve high accuracy, a more powerful architecture was trained to see if it could improve the accuracy. Using a ResNet model that utilized padding, MaxPooling, convolutional layers, dense layers, and 2 residual blocks, accuracy on the validation set only reached 94.4%.

3.6 Using "Off-the-Shelf" Models

The best fit model achieved an accuracy of 98.13% on the validation dataset and 97.80% on the test dataset. While this can be compared to the baseline accuracy of 33.33%, it is also helpful to compare this accuracy to the level of accuracy that off-the-shel models can obtain on the same

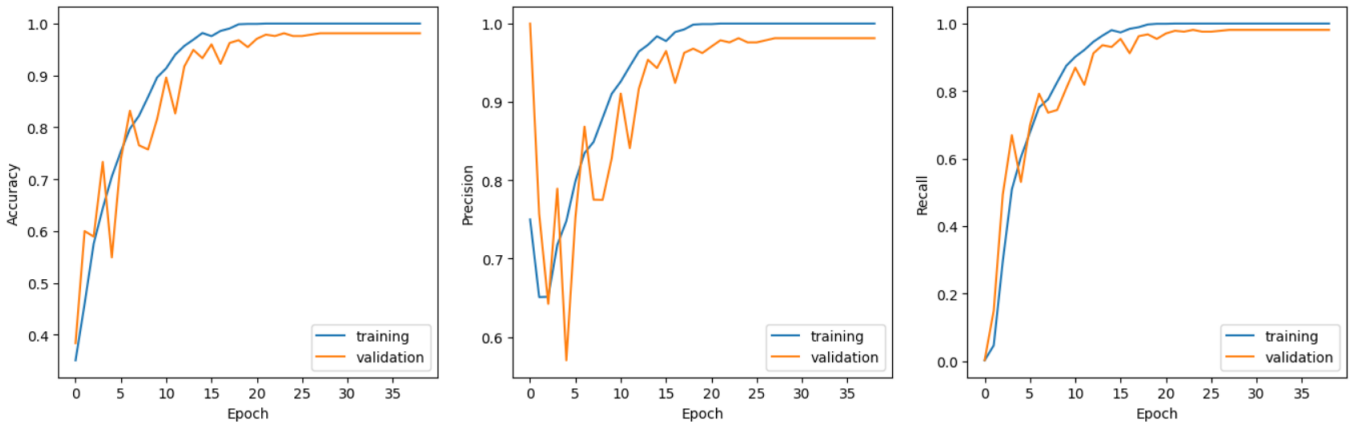

Figure 5: Best Fit Model Learning Curves

Table 5: A table showing how augmenting the images affected the model metrics

Metric - On Validation Dataset	Unaltered Images	Flipped Images	Rotated Images
Accuracy	98.13%	91.47%	33.33%
Loss	9.54%	25.19%	109.86%
Recall	98.13%	91.20%	0.00%
Precision	98.13%	91.69%	0.00%

dataset. Teachable Machine by Google [2] describes itself as "a web-based tool that makes creating machine learning models fast, easy, and accessible to everyone." After uploading the images from each artist into three separate classes on their website, the user can train a model using Google's default parameters. (see **Figure 6**). The model took only a few moments to train and could then be exported as a Keras model for further use. The Teachable Machine generated model was not able to obtain the same level of accuracy as the best fit model obtained from this project (see **Figure 7** and **Figure 8**).

4 Conclusions

This project attempted to train and compare deep learning models to classify paintings of three well known artists. A variety of convolutional neural networks were developed, trained, and tested to try to obtain the highest accuracy. Additionally, existing models were evaluated to compare with the best fit model.

The best fit model was a neural network with two convolutional layers and two dense layers. The first two layers were convolutional layers with 32 and 16 filters and a kernel size of 3. Each convolutional layer also utilized 4x4 maxpooling. The two dense layers contained 10 and 3 filters. The first three layers applied relu activation while the final layer applied softmax activation. Adam optimizer, categorical cross entropy loss function, and Keras callback functions were also used. Early stopping caused the model to finish training after 39 epochs with a validation

Table 6: A table showing how regularization methods affect model metrics

Metric - On Validation Dataset	Current Best Fit Model	With Dropout	With Batch Normalization
Accuracy	98.13%	97.33%	78.93%
Loss	9.54%	11.33%	91.43%
Recall	98.13%	97.33%	78.67%
Precision	98.13%	97.33%	78.88%

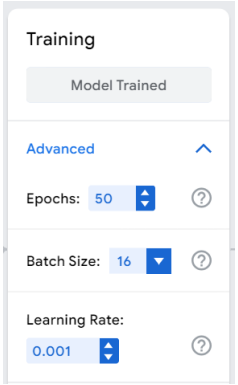

Figure 6: Google Teachable Machine Parameters

accuracy of 98.13%. With this being the highest validation accuracy obtained by any model, this model was determined to be the best fit. When evaluated on the test dataset, the model obtained an accuracy of 97.80%. The baseline accuracy for an evenly distributed dataset with three classes is 33.33%, so this model was able to reach high accuracy when compared to baseline.

The best fit model was further modified using data augmentation and regularization methods, but no improvements in accuracy were noted. To further evaluate the best fit model, a model was trained using Google's Teachable Machine. This model was able to reach 81%-89% accuracy. This model learned very quickly in the first few epochs, but never reached greater than 90% accuracy.

A deep learning model can accurately classify paintings of Van Gogh, Dali, and Picasso and pretrained networks such as Google's Teachable Machine are also relatively successful. Further projects to include more artists with even more similar styles would prove to be an interesting experiment.

Link to Google Colab Project: `https://colab.research.google.com/drive/1__3GOuyrNr6Qvg3ENi6Qw7JPwZf-iicX?usp=sharing`

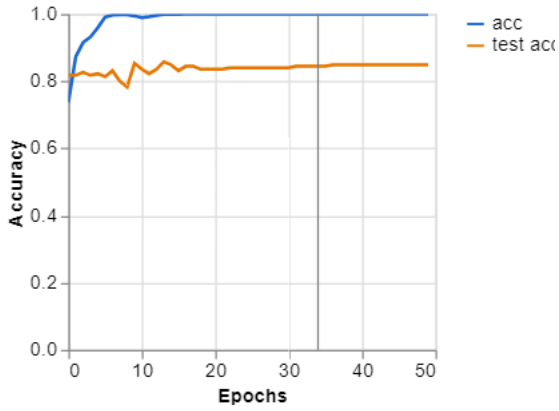

Figure 7: Google Teachable Machine Learning Curve

Accuracy per class

CLASS	ACCURACY	# SAMPLES
Van Gogh	0.84	75
Dali	0.89	75
Picasso	0.81	75

Figure 8: Google Teachable Machine Accuracy

References

[1] WikiArt. `https://www.wikiart.org/`. Accessed: 2024-04-24.

[2] Teachable Machine. `https://teachablemachine.withgoogle.com/`. Accessed: 2024-04-28.