Measuring Artistic Similarity of Paintings

Jay Whang, Yancheng Xiao and Buhuang Liu

Stanford Center for Professional Development / Google

Objective

In this project, we survey various techniques for the task of artist similarity detection, where a machine is asked to determine if two given paintings are by the same artist. Specifically, we compare the performance of manually-engineered features to that of a neural network in both online and offline settings.

Data

The data is obtained from the official Kaggle competition "Painter by Numbers" [1]. The training data includes 79433 paintings coming from 1584 artists in JPEG format, and 23817 paintings in the test set. While the dataset contains other metadata such as title, genre and style (e.g. Cubism), only the actual image data was used for training and testing.

Challenges

We found the training data size to be the biggest challenge. Out of 79433 paintings in the training data, there are over 3.1 billion distinct pairs of paintings that can be generated – a prohibitively large number given our resources. To cope with this, we generated a smaller version of the training set using top 100 painters by number of paintings. We then sampled 70 paintings from each painter, where 50 of them were used to generate training data, and the remaining 20 were used as validation set. Also, as the number of different-artist pairs far outnumbered same-artist pairs, we oversampled same-artist pairs to have 1-to-1 ratio in the test set.

Experiment Setup

For all techniques we evaluated, images were transformed into "feature vectors" in some way. Then, a model was learned on top of the feature vectors to determine whether the two paintings were by the same artist or not.

Experiment Setup (cont'd)

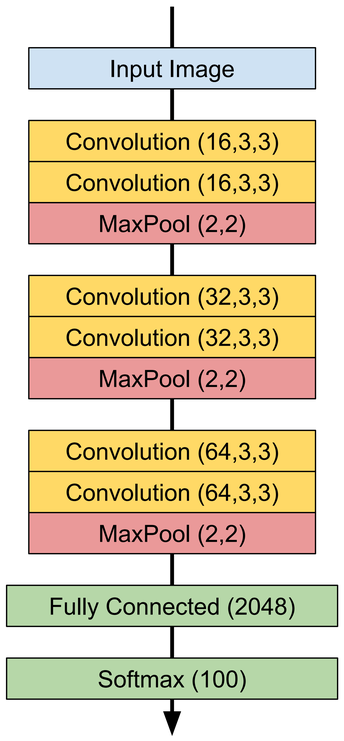
Here are the algorithms that we evaluated:

- Manually-crafted rules
- Logistic Regression, Linear SVM
- Random Forest
- Convolutional Neural Network (CNN)

Except for the CNN, we converted the images into feature vectors using various techniques from Computer Vision, including: edge detection, pixel intensity, hue intensity, aspect ratio, etc. For implementing the CNN, we used Keras [2] and Numpy [3].

CNN Architecture

The following convolutional neural network architecture was used to train a classifier that predicts the artist label on individual paintings. Then we use the output of the network as a feature vector, and use cosine similarity for final classification. Batch Normalization and Dropout were used as appropriate.



Results & Discoveries

The following table includes accuracy values on train/test sets from the best run for each algorithm.

Algorithm	Train (%)	Test (%)
Cosine Similarity	N/A	50.1
Logistic Regression	52.5	54.5
Linear SVM	55.6	57.4
Random Forest	94.6	67.7
ConvNet	N/A	59.9

Table 1: Performance of Different Algorithms

Combining Feature Vectors

We noticed that simply concatenating two feature vectors and training a classifier does not work at all, even doing worse than random guess (50%) in some cases. Linear models cannot learn the interaction between features across two paintings, so this was expected. Also, notice that a single feature from one painting reveals nothing about whether the two paintings are similar or not. For this reason, we postulate that Random Forest is unable to create a good decision stumps and thus does poorly as well. After multiple experiments, we discovered that using the positive difference (i.e. $\text{abs}(v_1 - v_2)$) worked the best. Conceptually, this quantity captures the "relative distance" between two paintings.

Better Embedding from Fewer Classes

For the CNN, we trained the model on different numbers of artists and found out that the model that was trained on to predict fewer (50) artists had better similarity detection performance. We believe that this is because predicting one artist out of 100 is likely more difficult than for 50 artists. This may have caused the trained model to be less optimal for the 100 case, affecting the efficacy of embeddings from it. Also, as the following plot shows, the CNN models had a lot of trouble generalizing to the validation set. Regardless of the regularization coefficient along with Batch Normalization and Dropout, validation loss always stayed higher than train loss. This suggests that the model was overfitting due to its small capacity.

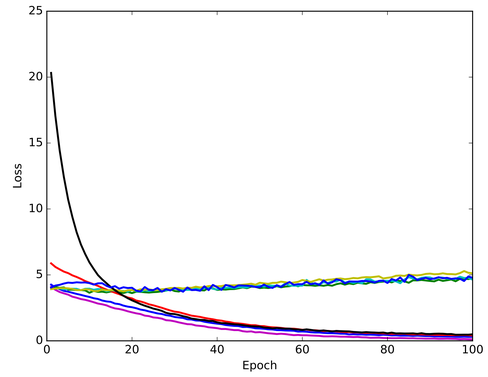


Figure 1: Plot of Train/Validation Loss for CNN

Conclusion

The task of artistic similarity detection was a surprisingly difficult challenge, as evident from many low-accuracy submissions on Kaggle [1]. Many subtle changes were required just to make training work, and even then the test accuracy is still below 70%. Although Random Forest outperformed other models, we strongly believe that a larger CNN that is trained on the entire data set will be the best model. This also suggests that a Siamese Network trained on top of two pre-trained CNNs could potentially do better than simply resorting to cosine similarity.

References

[1] Painter by numbers | kaggle.
`https://www.kaggle.com/c/painter-by-numbers`.
Accessed: 2016-11-20.

[2] François Chollet.
Keras.
`https://github.com/fchollet/keras`, 2015.

[3] S. Chris Colbert StÃlfan van der Walt and GaÃńl Varoquaux.
The numpy array: A structure for efficient numerical computation, 2001.
Accessed: 2016-12-12.