Project Progress Report

Title: Painting Classification by Artist

Team 1

Maobin Guo, Xinyi Pan, Pranav Manjunath, Aiman Haider

1. Project Brief

Our current implementation is a binary classification to identify whether two paintings come from the same artist. In the future, we aim to perform a multi-class classification on top 15 artists with the most paintings handed down over the dataset.

1.1 Data Source

The data comes from a kaggle competition : Painter by Numbers, which has been sourced from WikiArt.org.

1.2 Data Preprocessing

The dataset originally consisted of over 100,000 paintings. The first step we looked at was the number of artists, genres, and styles present in the data. We observed that there were 2319 artists and 2062 of them have less than 100 paintings. As the main goal of this project is to classify paintings into its respective artist, we believed it would be best to focus our task on the most representative artists, by which we mean the artists with the most paintings handed down.

In this regard, we observed the top 15 artists with the maximum number of paintings. The table below consists of the number of paintings done by each artist.

```
Zdislav Beksinski            500
Ivan Aivazovsky              500
Pablo Picasso                500
Ilya Repin                   500
Ivan Shishkin                500
Pierre-Auguste Renoir        500
Albrecht Durer               500
John Singer Sargent          500
Gustave Dore                 500
Marc Chagall                 500
Giovanni Battista Piranesi   500
Rembrandt                    500
Martiros Saryan              499
Paul Cezanne                 499
Camille Pissarro             499
```

The subsetted dataset consists of 15 artists and 7497 paintings in total. Once we train a model with this, we will then increase the number of paintings to train the model.

1.3 Methodology

We intend to identify the artist of the painting, given the painting. For this, we first intend to train our model on the training dataset above and then use the model to predict the painting from the test dataset. We first convert the painting images to vectors. We would then use these vectors to train the model and predict. We are presently using the VGG16 pretrained model and working on a baseline classification model. We would then try some more sophisticated architectures to enhance the feature extraction and classification simultaneously.

1.3.1 Present Implementation

The present implementation, to start simple, is to identify whether two paintings come from the same artist. Specifically, two images at a time are taken as inputs, and a binary classifier is used to judge whether these two images come from the same patiner. For that purpose, we used trained convolutional neural networks to extract features from each pair of images, then merge the two feature vectors into one, and finally classify whether the two images belong to the same artist.

1.3.1.1 Feature Generation

Currently, VGG16 is used to extract features from paintings. Every painting is compressed to 224*224 so that it fits the input requirement of VGG16. (VGG16 supports more than one input size, now we only use this size. Other input sizes will be tested in the further works.) We feed an image into VGG16 and get the image's feature vectors from the last convolutional layer's output. In the future, we are going to try image features generated from earlier layers.

1.3.1.2 Feature Merge Operation

In the last step, we would get two vectors *(v1 and v2)*, with each one standing for two different images' features. Before we put them into a classification model, we need to merge them at first. There are many methods can be used, for example:

- Difference : *v1 - v2*
- Sum: *v1 + v2*
- Max : *max(v1, v2)*
- Min : *min(v1, v2)*

Of course, there are also some distance metrics that can be used to calculate the difference of the two features such as euclidean distance, concisin similarity, etc. Currently, we use the vector difference to merge the vectors.

1.3.1.3 Classification Model

At present, we used random forest as a classification model to give us a quick baseline. We are in the process of developing more machine learning algorithms..

1.4 Evaluation Metrics

Since our current work is a binary classification task and there is no obvious difference between false positive and false negative errors, we decided to use accuracy as the evaluation metrics. However, for the final project we would use confusion matrix and micro-F1s.

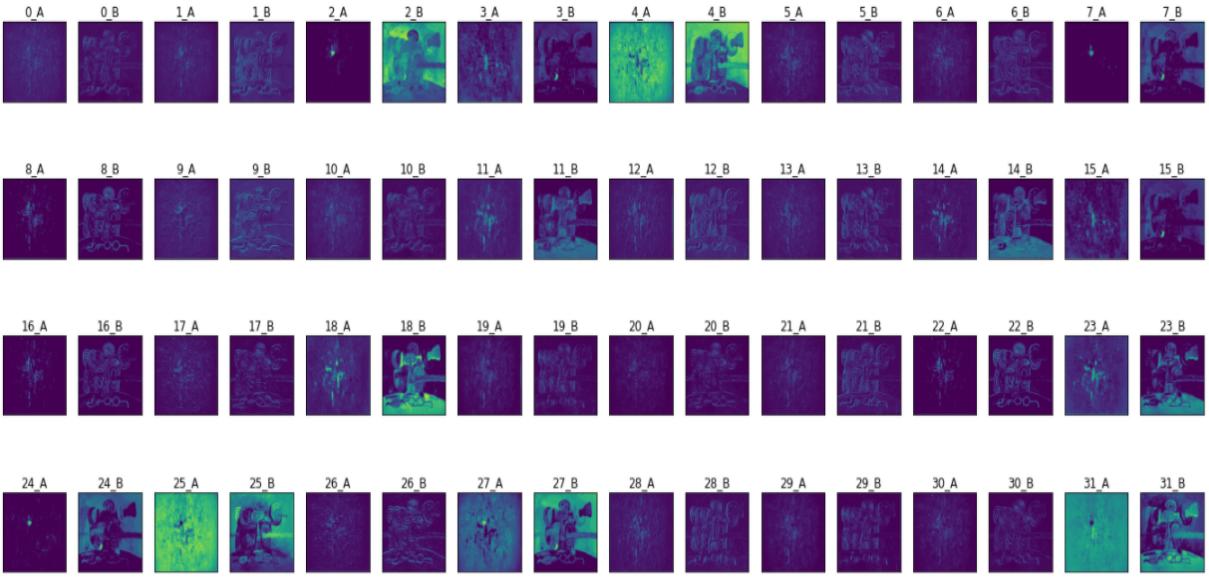
1.5 Machine Learning Tools

We use TensorFlow2 as our deep learning framework. Besides it, Sklearn, XGboot will also be used in this task.

2. Result

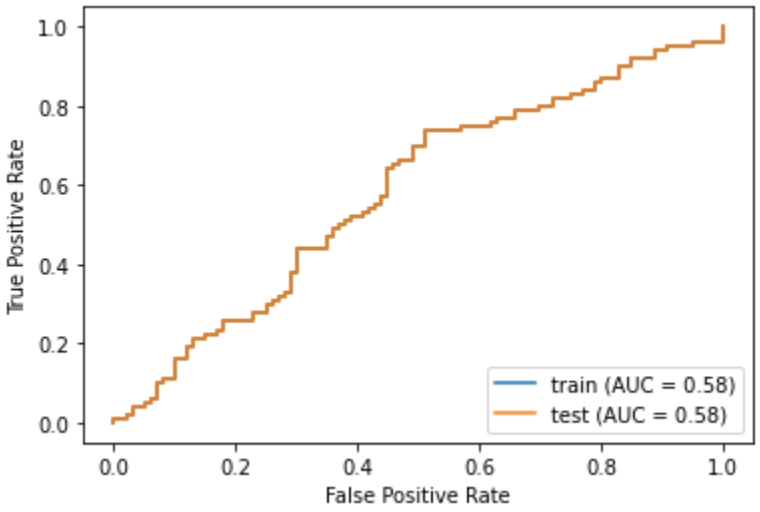
2.1 Feature Extract

For two pictures, we get 64 channel features from each. In the following image matrix, the heading number stands for the id of the channel. A means that it comes from the first picture, and B means that it comes from the seconds picture.



2.2 Model Result

The random forest model trained on 2000 training samples and 500 test samples. The accuracy is 56%.



The ROC curve indicates our model is not far from mature. One direction we want to try is to provide more training data, since this model is trained with only 2000 samples, while each sample has more than 20,000 features. The reason why we didn't feed enough samples to the model currently is explained in the following paragraph.

3. Challenges

3.1 Feature extracting

3.1.1 Challenge in computation time

Using VGG16 to extract is a computationally consuming task. At first, we used our CPU based local computer to do the task. On average, each sample would cost 4~6 seconds. Hence, 2000 samples would cost 3~4 hours. If we want 200,000 samples for training, the time would increase to two weeks! To resolve this problem, we are using AWS machine learning EC2, p2.xlarge, server to generate the sample. P2.xlarge has 4 vCPUs, 61G Memory and a NVIDIA K80 GPU with 12G GPU memory. It can generate 4 samples per second. Now we got a training dataset with 10,000 samples. And we are going to use it to explore other classification models.

3.1.2 Challenge in data size

Each sample contains more than 20,000 features, it means sample size is extremely hard to compute, store and transfer. Af first, all the samples were saved in a single file and we quickly found it is hard to manage.

Then we decide to store them in many small files. Each file contains 500 samples. By doing so, we resolve the problem temporarily. However, we are concerned that the data size would bring other problems for model fitting if we decide to use a large training dataset (200,000 for example).

3.1.3 Challenge of limited literature

Our methods are based on dozens of papers, however, almost none of them focus on artist comparison. Most of their research topic is about stype transformation. And this Kaggle competition only has 40 teams. The situation means the experience and existing methods for us to learn is limited.

3.1.4 Challenge in budgets

AWS EC2 p2.xlarge cost 0.9 dollars per hour. It can generate about 10k (10,000) samples per hour. If we want to generate a 200k dataset, we need to pay 18 dollars. To test different feature generation methods (mentioned in 1.2.1), it is necessary to generate millions of samples. It is expected to cost hundreds of dollars on feature extraction.

4. References

4.1 Papers:

1. W. Chu and Y. Wu, "Image Style Classification Based on Learnt Deep Correlation Features," in IEEE Transactions on Multimedia, vol. 20, no. 9, pp. 2491-2502, Sept. 2018, doi: 10.1109/TMM.2018.2801718.
2. Tiancheng Sun, Yulong Wang, Jian Yang, Xiaolin Hu, “Convolution neural networks with two pathways for image style recognition,” IEEE Transactions on Image Processing, vol. 26, no. 9, pp. 4102-4113, 2017.
3. Liu, L. et al. “Advanced deep learning techniques for image style transfer: A survey.” Signal Process. Image Commun. 78 (2019): 465-470.
4. X. Lu, Z. Lin, X. Shen, R. Mech and J. Z. Wang, "Deep Multi-patch Aggregation Network for Image Style, Aesthetics, and Quality Estimation," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 990-998, doi: 10.1109/ICCV.2015.119.
5. K. A. Jangtjik, T. Ho, M. Yeh and K. Hua, "A CNN-LSTM framework for authorship classification of paintings," 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 2017, pp. 2866-2870, doi: 10.1109/ICIP.2017.8296806.
6. Gatys, Leon & Ecker, Alexander & Bethge, Matthias. (2015). A Neural Algorithm of Artistic Style. arXiv. 10.1167/16.12.326.

4.2 Articles:

1. Change input shape dimensions for fine-tuning with Keras（https://www.pyimagesearch.com/2019/06/24/change-input-shape-dimensions-for-fine-tuning-with-keras/）
2. Extract Features, Visualize Filters and Feature Maps in VGG16 and VGG19 CNN Models（https://towardsdatascience.com/extract-features-visualize-filters-and-feature-maps-in-vgg16-and-vgg19-cnn-models-d2da6333edd0）
3. Developing Art Style Embeddings for Visual Similarity Comparison of Artworks (https://towardsdatascience.com/developing-art-style-embeddings-for-visual-similarity-comparison-of-artworks-7a9d4ade2045)
4. Creating an image similarity function with TensorFlow and its application in e-commerce (https://blog.griddynamics.com/create-image-similarity-function-with-tensorflow-for-retail/)
5. Detecting and correcting e-commerce catalog misattribution with image and text classification using Google TensorFlow (https://blog.griddynamics.com/fix-e-commerce-attribution-with-tensorflow-and-image-recognition/)