CS4100 Final Project Report: Artist Attribution

By Liam Pavlovic and Brant Wesley

For our final project we developed a deep-learning neural network to identify the artist that produced a particular piece of artwork. The neural network contains a series of convolutional layers that extract features from the image data followed by a feed-forward neural network that classifies the image data based on the features extracted by the convolutions. Our dataset[1] contains 9000 paintings from 50 well known artists, including Van Gogh, Kahlo, and Warhol. The main issues encountered were formatting the data for machine learning and determining the ideal parameters for the layers of the neural network. Although the artworks were categorized into folders corresponding to each artist, the dataset did not provide a train/test/validation split. We fixed this by building our own dataframe of filenames using the os library, splitting that dataframe into training and testing sets, and using the flow_from_dataframe functionality of keras's ImageDataGenerator. This feature of keras was a very useful discovery because it handled all of the image preprocessing for us.

We solved the issue of determining ideal parameters for the neural network's layers by running a set of experiments. The goal of these experiments was to determine if more layers, wider layers (more filters/nodes), or larger kernels would improve performance the most. We used the convolutional neural network[2] that we previously used for the LFW dataset in HW3 as our baseline. Each experimental model was designed and tested multiple times before a final version was selected to ensure that each proposed architecture was performing optimally.

Before beginning our experiments, we did some background research into the topic. One article[3] claimed that deeper convolution networks often outperform wider networks because they tend to generalize better and are less prone to overfitting. The article also noted that deeper neural networks also tend to be less time-consuming to train. Kernel sizes in the range of (1x1 - 5x5) tend to be the most common choices because the performance improvements of larger kernels often do not justify their increased time costs.

The first experiment involved building a model with the same number of layers as the baseline, but with far more filters per layer. The final version quadrupled the number of filters in each convolutional layer.

The second experiment involved building a model that used significantly larger kernels than the baseline. The baseline utilized kernel sizes of 3x3 and 5x5. The modified network used kernel sizes of 8x8, 6x6, 4x4, and 3x3, with each hidden layer having a smaller kernel size than the layer before it.

The final experiment involved building a model that was significantly deeper. The original model contained 2 convolutional layers, 2 pooling layers, and 1 dense layer. The modified model contained 8 convolutional layers, 3 pooling layers, and 3 dense layers. The number of filters in each layer, however, did not exceed 128 (most contained 32 or 64 filters).

After training and comparing the different models, we found that the wide model performed better than the others, with an accuracy of .357 and loss of 2.327 on the testing set. The baseline model also performed well, with an accuracy of .328 and loss of 2.533.

Surprisingly, the deep neural network performed fairly poorly with an accuracy of .197. This may have been due to design flaws, but several different deep models were tested and none performed better than the baseline. That said, the original paper[4] we referenced in our project proposal uses a very deep network (50 layers), so the network may just not be deep enough. We could not emulate such a deep network due to hardware constraints.

One possible source of the models' poor performance is the unbalanced nature of the dataset. There were several artists who had far more paintings in the dataset than others, such as Van Gogh (877 paintings), Picasso (439 paintings), and Degas (702 paintings), while others had relatively few paintings, such as Jackson Pollock who only had 24 artworks in this dataset. This skew may have made some of our neural nets more likely to predict the more prolific artists in the dataset.

The first major thing we learned about was how to use the ImageDataGenerator provided by keras. We learned the necessary arguments to apply random shearing, rotation, and zoom transformations as well as to resize all the images to the correct dimensions. We also learned how to use this ImageDataGenerator to pull image files from a directory. This will be helpful in the future whenever we need to train a neural network on an image dataset located in our local file storage. We also learned about the basics of designing convolutional neural networks and how changing different aspects of the networks can affect their performance. We learned how to build experimental models to determine which architecture works best for a particular problem. We also learned the general benefits and drawbacks of certain architectures (i.e. wide convolutional networks adapt to training data quickly but at the cost of potential overfitting). We also learned how skewed datasets may impact performance, as our models may have had higher rates of accuracy if they had been given a dataset where each artist had the same number of artworks.

Sources:

[1] https://www.kaggle.com/ikarus777/best-artworks-of-all-time
[2] https://keras.io/examples/vision/mnist_convnet/
[3] https://www.sicara.ai/blog/2019-10-31-convolutional-layer-convolution-kernel
[4]https://arxiv.org/pdf/1708.00684.pdf

Appendix: How to Run Our Code

1. Unzip the Liam_Brant_CS4100_FP.zip file

    a. There should be a python notebook, a pdf of the python notebook, a csv, and a images.zip
2. Upload the csv and images.zip to a Google Drive account, place them both in a folder called "artists" located at the home directory of your drive. Leave the artists.zip zipped. The notebook will unzip it when needed
3. Upload the python notebook to Google Colab, enable GPU backend for performance requirements
4. Run the first block, when prompted click the provided link, sign into the Google Drive account containing the "artists" folder and use the authentication key provided
5. Run the remaining cells in order
    a. Note: all the cells for training the models will take a decent amount of time (approximately 30 minutes for each model)