CS63 Spring 2019
The Paintings Dataset: The Search for Art

Sam Rothstein and Tymoteusz Chrzanowski

1 Introduction

The purpose of this project is to create a computational agent that has the ability to identify objects in paintings. This is accomplished by using convolutional neural networks to learn object-category classifiers given by sources such as ART UK. The dataset used for this project was created by the Visual Geometry Group from the Department of Engineering Science at the University of Oxford. Paintings from this data set date from the early 19th century to the mid 20th century and include a wide range of artistic genres such as rococo and post-impressionism. The ability to identify objects in paintings benefits the art history community and is a prime test of the capabilities of convolutional neural networks.

2 Method and Details

2.1 Dataset Configuration

The dataset began as an Excel spreadsheet that contained the image URL, website URL, subset category, and the object labels for 8629 paintings. Image URL contained a link to the painting on the ART UK website. Website URL is a link to the description of the painting on the ART UK website. Subset category divides data into training data, validation data, and testing data. Each painting is captioned with an object label that indicates what the painting depicts. The nine objects depicted throughout the data set are: ”airplane,” ”bird,” ”boat,” ”chair,” ”cow,” ”dining table,” ”dog,” ”horse,” ”sheep,” and ”train.” We began processing the dataset by removing data marked as N/A and image URl’s that were links to .gifs. We also simplified object labels to include only the most prominent object in the painting. These simplified object labels are represented by one-hot arrays and are the Y output of our convolutional neural network.

	Air	**Bird**	**Boat**	**Chair**	Cow	**Dtable**	**Dog**	**Horse**	Sheep	Train	Total
Train	74	**319**	**862**	**493**	255	**485**	**483**	**656**	270	130	3463
Val	13	**72**	**222**	**140**	52	**130**	**113**	**127**	76	35	865
Test	113	**414**	**1059**	**569**	318	**586**	**549**	**710**	405	164	4301
Total	200	**805**	**2143**	**1202**	625	**1201**	**1145**	**1493**	751	329	8629

Figure 1: Dataset Statistics

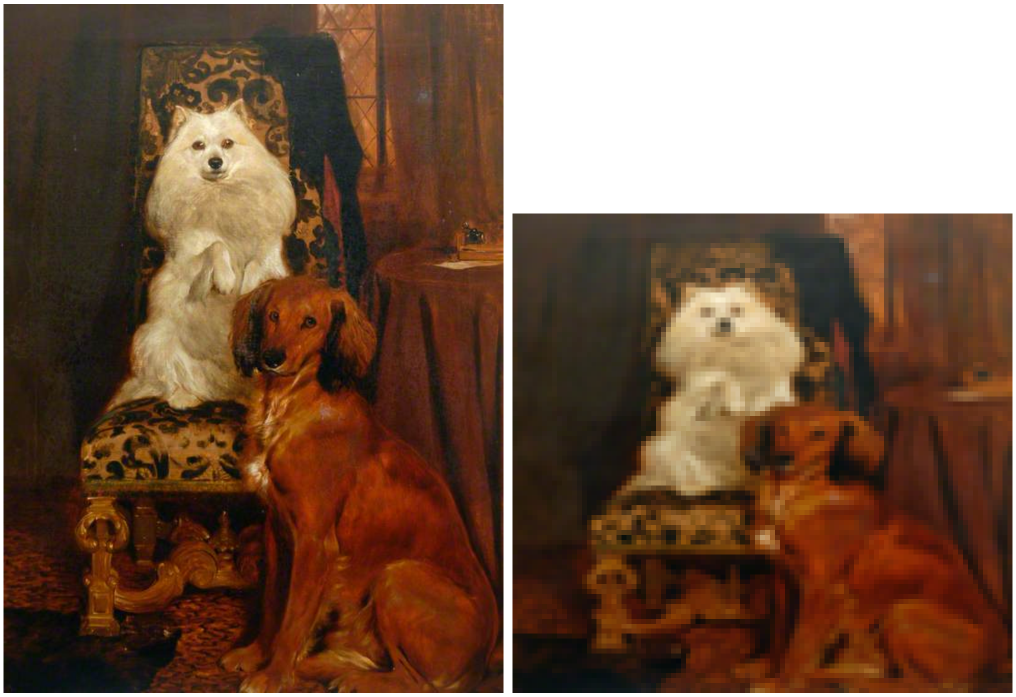
Although the dataset is vast, the distribution of each type of object is not uniform. Objects such as ”boat” are strongly represented and objects such as ”cow” are weakly represented. This is extremely problematic when training a convolutional neural network because it incentivizes laziness: a convolutional neural network can perform reasonably well by simply guessing the same object category for each painting.

To decrease guessing accuracy and encourage learning, we continued modifying the dataset by removing under-represented object categories. The bold entries seen above indicate the data used for our convolutional neural network. This data was selected because each object included is evenly represented, thus preventing the convolutional network from guessing and achieving a high accuracy. Thus, our final dataset contained 5,590 paintings and used 6 object labels.

We then split our dataset into training examples and testing examples. For each object, roughly 70 percent of paintings were categorized as training examples and remaining paintings were categorized as testing examples. The validation subset category was omitted.

2.2 Image Manipulation

The paintings in the original dataset had a fixed width of 550px. While object recognition is easier with larger images, processing paintings of this size proved to be extremely time consuming. To keep our runtime down, we used Pillow (PIL) to pre-process the paintings before inputing them into our convolutional neural network. We began processing the paintings by resizing each image to have a width and hight of 90px. We also applied the ANTIALIAS filter to each resized painting to conserve paint and color texture.



(a) Original Painting

(b) Processed Image

Figure 2: A comparison of raw and processed images

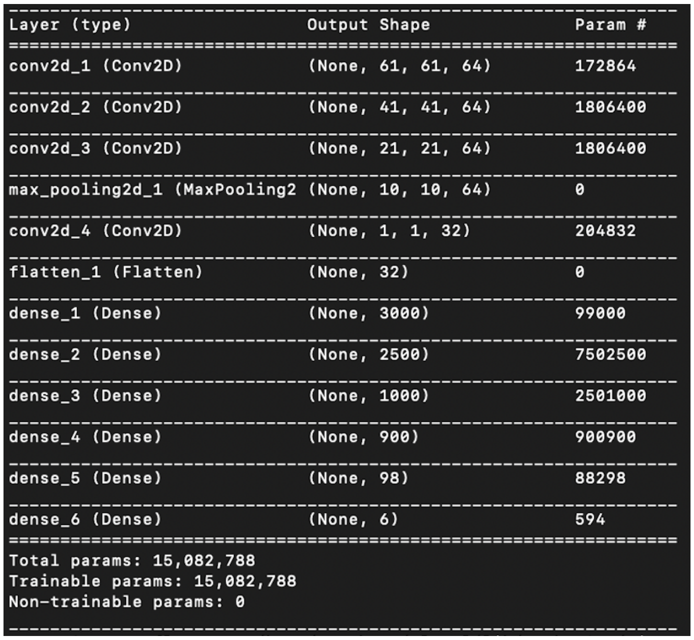
Once paintings were resized and filtered, we created an array containing rgb values for each pixel. This array was the X input to our convolutional neural network.

2.3 Time Constraints and Testing

Runtime is a major factor to consider when training and testing convolutional neural networks. For the purposes of this project, we limited our runtime to 6 hours when processing all 5,590 paintings in our dataset. With this constraint, we were forced to use only 10 epochs. 10 epochs were used in all tests and will all iterations of our convolutional neural network architecture.

2.4 Convolutional Neural Network Architecture

We used the following layers in our convolutional neural network.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 61, 61, 64)        172864
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 41, 41, 64)        1806400
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 64)        1806400
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 1, 1, 32)          204832
_________________________________________________________________
flatten_1 (Flatten)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3000)              99000
_________________________________________________________________
dense_2 (Dense)              (None, 2500)              7502500
_________________________________________________________________
dense_3 (Dense)              (None, 1000)              2501000
_________________________________________________________________
dense_4 (Dense)              (None, 900)               900900
_________________________________________________________________
dense_5 (Dense)              (None, 98)                88298
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 594
=================================================================
Total params: 15,082,788
Trainable params: 15,082,788
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: The layer summary outputted by Keras

Our network architecture contains 4 convolutional layers and 1 pooling layer. The pooling layer is flattened and sent into 5 dense layers. One output layer follows the dense layers.

This network architecture was developed by testing subsets of the dataset. We tested our network on 1000 random paintings and manipulated the network architecture until we saw the best accuracy. During testing, we noticed that extracting decrementing numbers of features in our convolutional layers increased our accuracy substantially. Dense layers of roughly 1000 nodes also proved to dramatically increase our accuracy.

3 Results

3.1 Overview

To determine the performance of our convolutional neural network, we tested our network on 2000 paintings, 3000 paintings, and our complete dataset of 5590 paintings. The performance of our final network architecture is summarized in figure 4. Our previous network architectures often maxed out at around a 30 percent accuracy. Due to the long runtime of our testing, we did not rigorously test and preserve the statistics for each iteration of our network. However, the final iteration of our network architecture achieved the highest accuracy by a large margin.

Number of Paintings	Accuracy
2000	13.15%
3000	44.43%
5590	50.86%

Figure 4: Results Summary

The percentages in this table reflect the average accuracy of five runs of our program for each specified number of paintings. As mentioned in the previous section, 10 epochs were used for each test.

The accuracies associated with our results are somewhat deceptive. It is important to note that an agent that guessed objects randomly would only achieve an accuracy of around 16 percent. Thus, an accuracy of 50 percent is a significant feat. It is also important to note that many of the paintings in our dataset have abstract qualities. Abstract qualities make object recognition challenging for neural networks and humans alike. Through closely monitoring our networks performance, we also noticed that when testing our entire dataset, network accuracy increased substantially during the final few epochs. With more runtime, we are confident that we could push our accuracy past 50.86 percent.

3.2 Interpreting the Results

One way to gain intuition for the behavior of our convolutional neural network is to construct a confusion matrix. The x axis of the confusion matrix represents the object depicted in the painting that the convolutional neural network receives as input. The y axis of the confusion matrix represents the percentage of time each object is outputted from the convolutional neural network.

	Bird	Boat	Chair	Dtable	Dog	Horse
Bird	39%.	7%	12%	8%	7%	15%
Boat	8%	64%	5%	4%	9%	5%
Chair	12%	4%	59%	16%	13%	6%
Dtable	14%	13%	15%	55%	13%	16%
Dog	11%	9%	4%	7%	43%	13%
Horse	16%	3%	5%	10%	15%	45%

Figure 5: Confusion Matrix

The values in the confusion matrix seem to indicate that cohesive learning occurred in the convolutional neural network. This is evident when comparing distinct column values. For instance, the network often confused dining table and chair. When the network is given a painting of a chair, it most commonly mistakes if with a dining table. Similarly, when the networks is given a painting of a dining table, it often mistakes it for a chair. This can be seen with many combinations of objects. While these confusions demonstrate faults in the network, they indicate that the network has developed a thought process that is reminiscent of that of a human. Chairs and dining tables are indeed similar object, thus the mistake is highly human-perceptible. While these mistakes take away from the accuracy of the network, they are somewhat reassuring because they indicate that the system has learned in a cohesive way.

4 Conclusions

Convolutional neural networks can approach human-level object recognition in paintings. Our convolutional neural network was able to make tremendous strides towards human-level performance on a relatively small data set and with a relatively low runtime. With a larger dataset and a longer runtime, we are certain that an agent could approach human-level performance. Object recognition in paintings can be an extremely valuable tool to the art history community. Object recognition technology could remove the need for people to spend time on mindless, yet important, tasks such as archiving and managing large sets of paintings. While our program is not one which could replace human labelling, it is a step towards something that would enable art historians to make better use of their time and human capabilities.