Painting vs. Photograph Classifier

Naveen Kumar Rajesh, Jacob Borello

ABSTRACT

The project is a simple classifier that analyzes a given input and either classifies it as a Painting or a Photograph. The python programming language was used in addition to libraries such as PIL, OS, and more. The project involved using various methods of image manipulation including image resizing, pixel manipulation and coloring. The neural net was trained using the perceptron algorithm with varying techniques such as 1-pattern and 2-pattern.

KEYWORDS

Perceptron, Image processing, RGB, 1-pattern, 2-pattern

INTRODUCTION

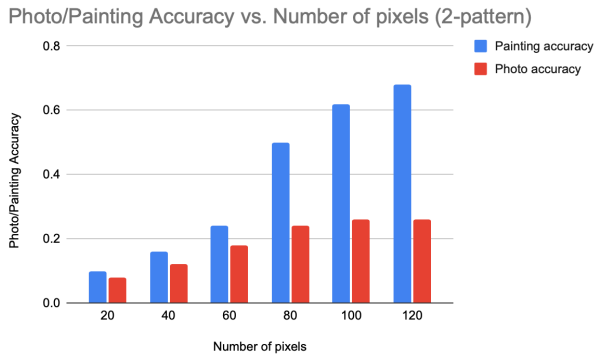

Figure 1: **Graph to represent the accuracy of Photograph and Painting classification from 2-pattern**

This table displays how the accuracy of categorizing images is affected by the number of square pixels. There was a significant jump in accuracy between 60 and 80 square pixels analyzed. The photograph categorization accuracy seemed to stop increasing around 30%, despite an increase in square pixels being analyzed. The painting categorization on the other hand had a steady increase in accuracy depending on the number of pixels being analyzed.

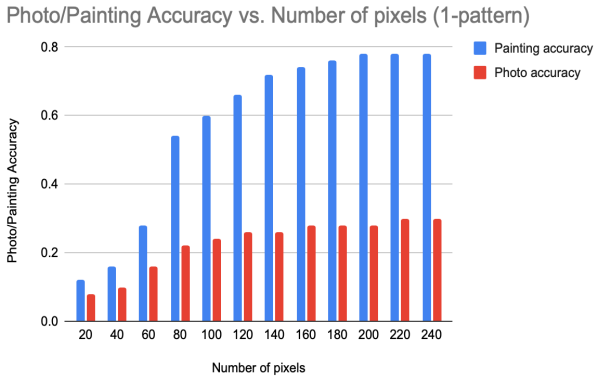

Figure 2: **Graph to represent the accuracy of Photograph and Painting classification from 1-pattern perceptron net**

The painting accuracy had a steady increase up until 200 pixels being analyzed, when the categorization accuracy stopped around 80%. The photograph categorization did not receive large increases in accuracy past 30%, despite an increase in pixel data.

BACKGROUND

To tackle this classification, a greater understanding of image manipulation was required. One library that was found was the Python Imaging Library(PIL), which allowed for greater options in terms of being able to resize and pixel management for the program.

APPROACH

The first step of the program involved iterating through a folder of photographs or paintings and changing all of the pixels in each picture to black and white based on how close each RGB value is to either black or white.

Before we could convert an image into a vector, we needed to determine which pixels we would sample to train our weight matrix. Our images were all of different shapes and sizes, so we were forced to take only a subset of the images pixels. We did this by allowing the user to specify how big they wanted their sample “square” to be. Although this feature was initially intended only to be used for avoiding errors when reading images, it became a

requirement due to computing issues which are explained further down.

After converting to black and white, a vector matrix was needed to pass on the training function. To obtain the vector for each image, the program iterates through each pixel in the black and white image and assigns a value of 1 or 0 and appends it to a matrix.

The matrix was then saved to a file with all the corresponding vector matrices with a marker of “-1” or”1” in a empty line at the end of the file so that program could later identify whether the values are for a painting or a photograph.

Initially we had wanted to use a 2-pattern net, but we lacked the computing pattern to analyze more than 120 x 120 pixels at the same time. To overcome this, we decided to use a 1-pattern perceptron net instead. Initially we would run the image vector through two sets of weights, and depending on the output ((-1, 1) or (1, -1)) we would determine if there was a successful categorization or not. Because there were only two outcomes, a 1 or a -1, we decided that a 1 pattern perceptron could achieve the same outcome and would require less computing power.

We would first train a weight matrix based off of picture vectors loaded into a file. Then we would read that weight matrix whenever we wanted to test the contents of a vector image file. The result of a test would be the percentage of successfully categorized images in a vector file.

CONCLUDING REMARKS

From this project a lot was learned in terms of image processing and performance limiters. While the program was accurate and performed as intended, a large dataset or a high pixel count would result in very high run times and even just lead to a computer crash.

For future upgrades, we want to be able to classify between a painting or a photograph without having to convert the pixels to black and white. To achieve that, we would use convolution neural networks so that a lot of detailed information can be derived from each of the input data, which would allow for greater accuracy. We would also like to be able to improve the data processing speed to take in more pixels by using a more powerful GPU.

In the future we'd like to go about the project a different way. First we would find the smallest height and width of the pictures being vectorized. Then we would use this to determine how many “squares” we could evenly break each image into during the first stage of testing. After looking at each "square" by using a perceptron net to determine if it resembled a photograph or a painting. Using this new matrix, we would then train and use a separate net to determine the categorization of the entire image instead of only using a sample from each image. In addition to creating a net which would hopefully produce more accurate categorizations, we would also be able to test the entire image instead of a small fraction of it. This would also allow us to avoid computing limitations which prevented us from using the 2-pattern perceptron net.

Something which might help the photograph accuracy would be to use a data set which had less skies or black space near the top. As a result, the trained weights were likely limited by the uninteresting data from the subset of pixels we took. If we were to conduct the experiment again, we would take a sample from the middle of each image instead of the top right corner. This would diversify our data, preventing too many image vectors from being dominantly black or white because of solid colors.

REFERENCES

[1] Sambasivam, Siddesh. “Painting vs Photograph Classification Dataset.” *Kaggle*,11Mar.2020 ,https://www.kaggle.com/iiplutocrat45ii/painting-vs-photograph-classification-dataset.

APPENDIX A : TEAM CONTRIBUTIONS

Naveen Kumar Rajesh was responsible for converting the images in the folder to black and white and then forming the vector matrix by iterating through each pixel in an image and appropriately assigning values of ones and zeros. He also made sure that the vector matrices were saved to a “.txt” file that can be read in later. He also worked on putting together the presentation slides with the help of his partner Jacob Borello.

Jacob Borello was in charge of writing and testing the code to read from vector files made by Naveen, and creating and testing the perceptron nets.