Neural Networks Project

Introduction

Neural Networks and Deep Learning are the biggest buzzwords in Computer Science today. One of the particular fields that shows a great deal of promise are Convolutional Neural Networks or CNN's these special neural networks use a process known as convolution to learn features in conjunction with traditional Back Propogation. CNNs are particularly useful for processing images because convolution itself treats the data as a 2D matrix. Many different pretrained models exist to process image recognition and they each provide different levels of accuracy and different specialties. There are also a number of APIs featuring pretrained networks to allow more developers to utilize image recognition technology. For this project, these different models and APIs will be compared and contrasted to determine the best use cases.

Dataset

For simplicity purposes, all classifiers will be trained with a simple dataset that consists of cats and dogs. The classifiers will be analyzed based on their ability to differentiate between the two different creatures. The training set imcludes 8000 images, 4000 cats and 4000 dogs. The test set contains 2000 images, 1000 cats and 1000 dogs. While this dataset is simple, the purpose is to learn the process of the CNN and the basic construction, as well as comparing the models.

Technologies Used

Python TensorFlow Keras Numpy

Google Cloud

All computation was performed on a Google Cloud Compute Instance running Ubuntu 18 to speed up computation.

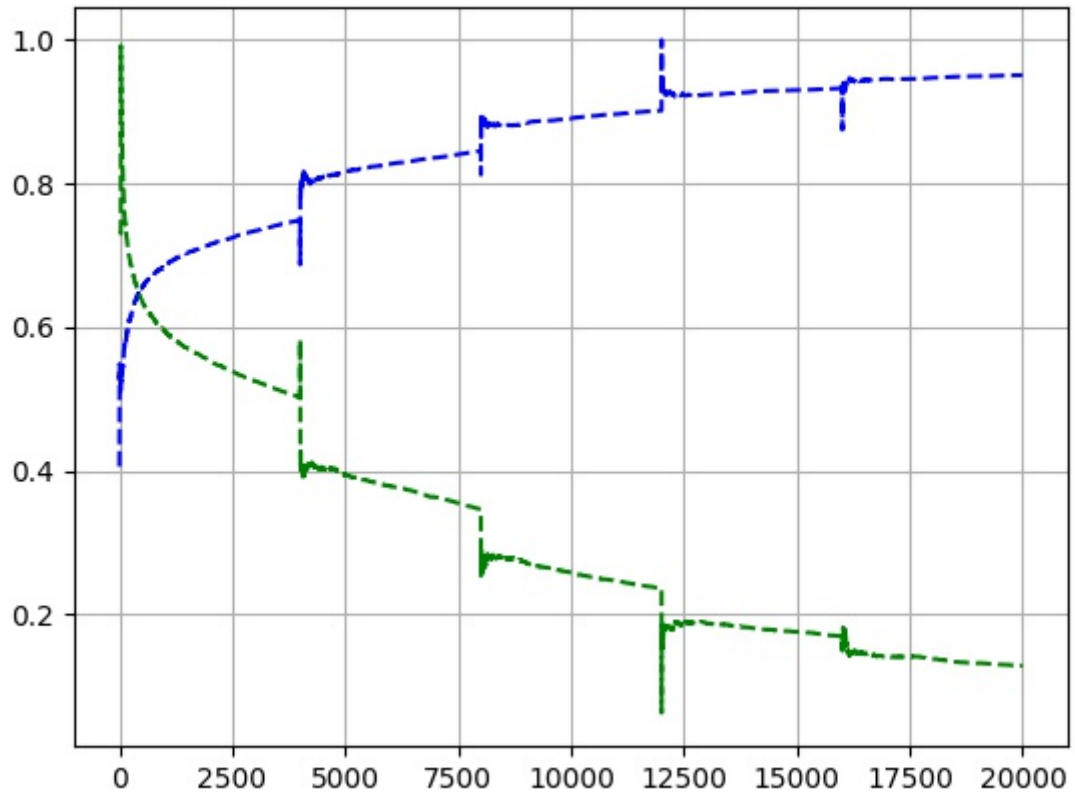
Homemade Image Classification with Keras

For the first attempt at image classification, I used the Keras Library on top of Tensor Flow to create a basic image classifier. This Convolutional Nueral Net contained 5 Layers: Convolution (with 32 filters 3*3 each), Max Pooling, Flattening, Fully Connected and Output (Also Fully Connected)

Training this classifier took quite a while even on a high-equipment machine. The classifier was to be trained for 25 epochs on the dataset. Each Epoch took approximately 27 minutes to complete. I first had an issue of my laptop going into sleep mode and killing the process when the ssh connection went down. To avoid this, I found and utilized the Unix command `nohup` which prevents the program from quitting when the shell is exited. However, I then had another issue of accidentally killing the process myself when working on another experiment, to save time, I reran the homemade algorithm with half the dataset and 5 epochs to get a proof of concept.

Below are the graphs depicting the loss (green) and the accuracy (blue) when training and testing. These values were outputted by keras after each datapoint was plugged into the network. Notice how there is a spike at each epoch (every 4000 datapoints) this makes sense because the data is shuffled at each epoch to allow for more variety in

training and to prevent an overfit. It is also important to note that the CNN got up to 95% accuracy according to Keras.



As great as these results seemed, it was not meant to be. When I attempted to run my own cat images through the network, it would very rarely classify them correctly. Upon further investigation, the way I attempted to limit the training data size in Keras actually only trained the network on dog images. I had to completely start over and retrain the network on the full set of images.

Resnet50

Google Cloud Vision API

Amazon Rekognition

Results