Deep Learning Project for CSC 416

Mitchell Anderson
Computer Science Undergraduate
South Dakota School of Mines and Technology
Rapid City, SD 57701
mitchell.anderson@mines.sdsmt.edu

Abstract

This paper covers the process through which a deep neural network was created for the ImageNet dataset. It covers what was hoped to have been accomplish and the setbacks that were encountered.

1 Introduction

In the process of attempting to write the program, several consecutive issues were encountered. The code was largely based off of a tutorial from the TensorFlow website which covered the estimator API[1]. During the extended week, a different approach was attempted, to see if something functioning could quickly be put together, however, this attempt fell short as well.

1.1 What Went Wrong Initially

The tutorial from the TensorFlow website introduced estimators using the iris dataset. This utilized data from a CSV file, which it parsed and fed into a training algorithm. The goal for this project was to modify the TensorFlow code to read and train on images from the ImageNet dataset. During this process, however, there were several issues encountered to which a solution was never found.

Through the process, many of the issues were overcome, but the problem of adapting the code to the ImageNet dataset proved more difficult than expected. The final issue encountered which no solution was found for was to use the data collected from the images and feed it into the `DNNClassifier` function. To do this, the data first had to be used to create a `feature_column`, which was then fed into `DNNClassifier`.

It is unknown whether it was the creating a `feature_column` step which was never solved, or if it was feeding that information into the `DNNClassifier` that caused the issue. It is also possible that there was a bug elsewhere in the program which was spilling over into that section.

When the program (which can be found in the code's repository history from Apr 22, 2018) is run, the most recently encountered error is printed to the screen. Given the error, it is possible for issue to have occurred anywhere in the program. A likely location, aside from the aforementioned areas, could be in the `train_input_fn` function, which was another parameter given to `DNNClassifier`.

1.2 What Went Wrong on the Second Attempt

As mentioned, during the extended week for the project, a second method was attempted to classify images from the ImageNet dataset. Again, the code was based off a tutorial, but this time one given by Chengwei Zhang[2]. This tutorial used the TensorFlow Keras integration to create a TensorFlow Estimator, then used that estimator to train a neural network.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

Similarly to before, the code was modified to accommodate using the ImageNet URL dataset. Once again, there were many errors that were raised one after another, so it was unable to be completed after one short week.

If there had been enough time, the program seemed easily completable had image files been used, rather than image URLs. It appeared fairly straightforward how to feed image files into the program, but decoding image files remotely from a website, then feeding that into the TensorFlow proved to be a more difficult task than anticipated.

One final thing to note from the second attempt is that the tutorial used a TensorFlow function called VGG16 (commented out in the source code). However, when this was used, an exception was raised stating that `h5py` was required. A quick look online revealed that the most likely solution was simply to install `h5py` using `pip`. However, the code was being run on the Linux23 lab machine, so it could not be installed due to privilege levels. Either being unable to use this function or using image URLs rather than image files is likely what caused the most recent error, displayed when the program is run.

1.3 What Was the Goal

As mentioned before, the goal was to modify the tutorial program given by TensorFlow (and later given by Chengwei Zhang) to analyze and classify images from the ImageNet dataset, rather than classifying the iris dataset using CSV files.

Since the program was never finished, different optimizations could not be done. Had there been an opportunity, many different hyperparameters would have been tuned to find the most accurate result. This includes the number of hidden layers and neurons in those layers, the activation function, and the optimization function, among others.

As the program was never completed, the following the sections could not be expanded upon. However, the outline of what would have been discussed is shown below.

2 Neural Network Architecture

In this section, the program's deep neural network architecture would be explained, as well as the reasons for why it was chosen. The current architecture of the program is explained here, however, some of the aspects would likely change should the program be completed, tested, and tweaked.

The program currently has a deep neural network, consisting of two hidden layers, each with ten neurons. The input layer is of the shape `(150, 150, 3)`, so it requires an image with a resolution of 150x150 where the three is the RGB values. The activation functions for both of these layers is ReLU. If the aforementioned `VGG16` function were working, then there would be extra pre-trained layers given by the function which would help with the classification.

The networks optimizer is using the `RMSprop` function provided by Keras. The learning rate is set to 1×10^{-10}, as suggested by the tutorial given by Chengwei Zhang. This model, created in Keras, is then fed into the TensorFlow estimator framework, using the function `model_to_estimator`. The input data is resized to the shape of `(150, 150, 3)`, and the data is then fed into the estimator.

3 Techniques Used to Optimize Results

This section would cover what techniques would have been used to fine tune and produce the best possible results. For example, an explanation behind the tests performed on different activation functions and which one was chosen to have provided the best results. How the training set was split from the test set would also be explained here.

4 The Results

In this section, the results of the deep neural network would be stated and analyzed. Including tables and visualizations of the results and the sets used to train and test.

5 Future Enhancements

In this final section, the parts of the program which could have been further improved would be discussed. Further tests could also be discussed here.

At the moment, the improvements to be made to the program includes, simply, to make the program run without error.

References

[1] The TensorFlow Authors (2016) *Premade Estimators for ML Beginners.* https://www.tensorflow.org/

[2] Zhang, Chengwei (2017) *An Easy Guide to Build New TensorFlow Datasets and Estimator with Keras Model* https://www.dlology.com/