Brandon Obasogie

October 21, 2019

**EEN614: Neural Networks  
Project 3 – CNN Implementation for Image Classification**

Introduction:

The goal of this project was to create a network in TensorFlow or Keras to learn how to classify images. In this report the preliminary steps and results are discussed and shown. The training was done using the CIFAR-10 dataset. The CIFAR-10 dataset contains 60,000 color images size 32x32 with 10 classes. Using the architecture of your choice achieve an accuracy of at least 80%. Submit the two best networks you create.

Below is a plot the first 10 images to sample what kind of images are included in the dataset.

A screenshot of a cell phone

Description automatically generated

1) 8-layer CNN as follows: CONV-POOL-CONV-POOL-CONV-POOL-FC-FC. This architecture will use (3 x 3) filters with channels of 32, 64, and 64 with a stride of 1. The pool layers will have a pooling of (2 x 2) with a stride of 2. The Total number of parameters will also be displayed below.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A close up of a map

Description automatically generated

The performance of this model is very good. The model doesn’t over train memorizing the training data. The loss is low compared to the previous iterations of the model I compiled. An accuracy of 82% with a validation on the 10,000 other images with 72% accuracy. I’m extremely proud of this model with only 89,000 parameters

2) The created CNN above architecture has changed to CONV-CONV-POOL-CONV-POOL-FC-FC. The filters will increase from (3 x 3) to (5 x 5). The channel sizes will remain constant at 32, 64, and 64 with a stride of 1. The pool layers will remain (2 x 2) with a constant stride of 2.

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A close up of a map

Description automatically generated

The total number of parameters in #2 is 3 times the amount of number one. The complexity of this network is greater than the first. The larger kernel or filter size makes it get a higher number of weights but less layers. In comparison the 3 x 3 filter yielded a lower number of weights but more layers. This makes this training computationally expensive and less efficient. That was observed by the epoch training times before utilizing the gpu. It took 535 seconds almost 10 minutes per epoch. Yielding a similar result to the previous architecture. However, the loss is higher in this model toward the end along with the validation accuracy. I believe this is due to overfitting the dropout function added to the code probably needs to be higher than 20%.