Personal Reflections on Individual Assignment

Darragh Sherwin

IT Sligo

Abstract

This is a personal reflection on the learnings from the individual assignment on transfer learning.

1 Process

Initially, I used code from Hands-on Transfer Learning with Keras and the VGG16 Model tutorial (https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/). My initial implements produced high training accuracy but very low testing accuracy (<5%) and took several hours per epoch to execute using Google Colab. This was due to not enabling GPU settings in the notebook. Simultaneously, I tried to get the notebook running on a PC with GTX 970 GPU but did not correctly configure the GPU's driver. Without the adequately configured driver, the time per epoch was 6-7hours on the PC.
Once I enabled GPU settings in Google Colab, the time per epoch was reduced to 7mins.
I tried the following steps to increase the testing accuracy as the tutorial had demonstrated high training accuracy:

- Split the training data into training and validation data sets.
- Adding Keras pre-processing function to the ImageDataGenerator
- Enabling feature normalization in the image data generators as discussed in https://machinelearningmastery.com/how-to-normalize-center-and-standardize-images-with-the-imagedatagenerator-in-keras/.
- Setting the image data generator to resize the images to 64x64 instead of 224x224 pixels.
- Add class weights as described in https://stackoverflow.com/questions/41648129/balancing-an-imbalanced-dataset-with-keras-image-generator.
- Switching the EarlyStopping callback to monitor training loss instead of validation loss.
- Allowing the first five layers to be trainable
- Allowing all layers to be trainable

None of these steps achieved a testing accuracy of about 5% and was hitting GPU usage limits on Google Colab.

I switched to using Vertex AI Workbench on Google Cloud Compute with an Nvidia Tesla GPU. Reading back on my implementation from the tutorial, I had misconfigured the ImageDataGenerator for the testing data with the class_mode mode set to categorical. It should have been assigned to None, so the generator only returned the images.

When I corrected the implementation of the Image Data Generator, I got testing accuracy at 73% with just the output layers changed. I then tried various combinations of layers to be trainable and with and without pre-processing the input. I eventually settled on ten trainable layers, resulting in 97.38% testing accuracy.

2 Overall Reflection

There are quite a few tutorials available on implementing transfer learning; once these tutorials are followed and correctly implemented, you can achieve high accuracy scores. Transfer Learning is a technique I will use in future projects which require classification.