Creating a Cuda Engine for Fashion MNIST Dataset Classification

Abstract

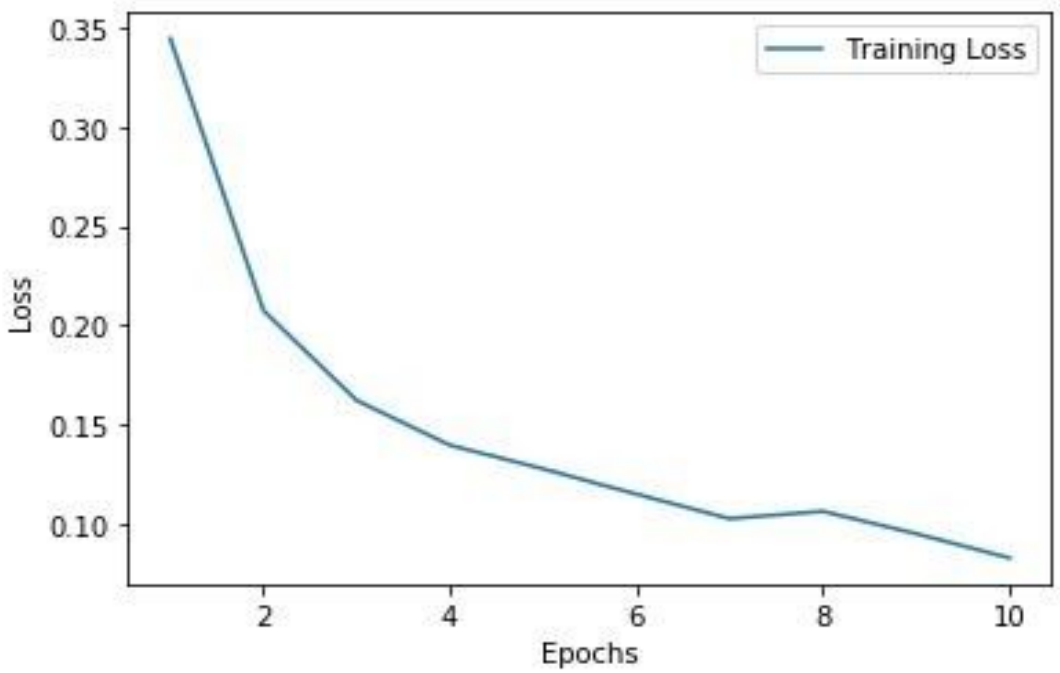
Majority of the projects done in Deep Learning are based on the Python Framework. Python offers a developer a free handed scripting language and easy to use APIs and frameworks. Debugging and visualising the performance has been a crucial requirement of developers which Python satisfies competently. This ease of coding and creating logic might be an optimal way to make the model but during the production and actual working Python falls short of efficiency. This makes them less efficient and impractical at production level and machine level languages like C/C++ are prefered. To combine these two mutually exclusive benefits of Python and C++ we divide the process of a ML project into two. Training part is done by Python and the actual working after deployment is done by a C++.

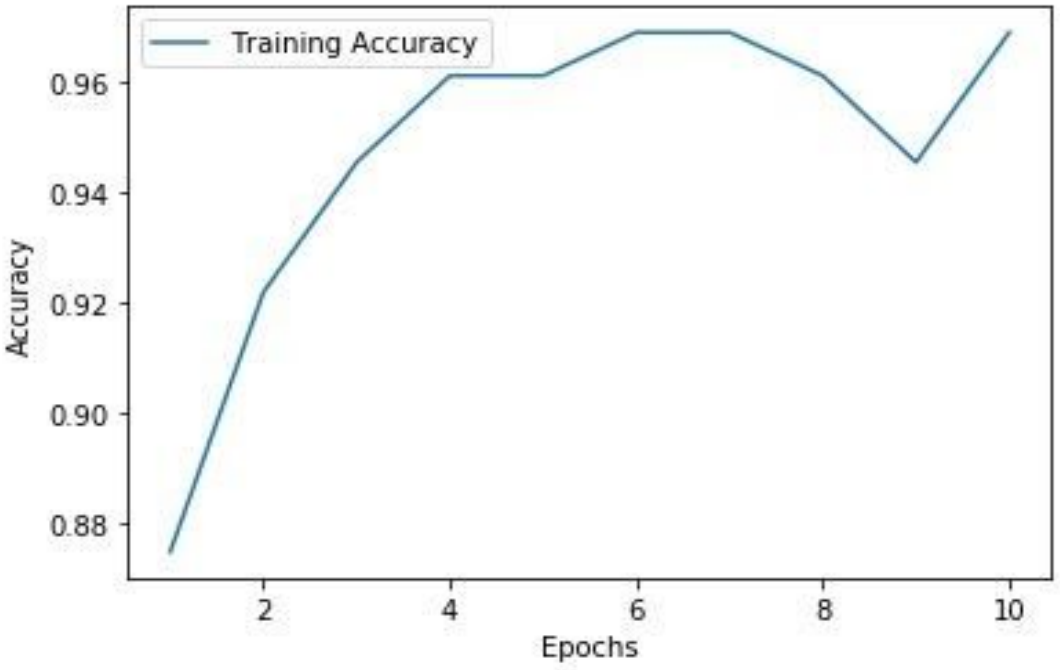
1.0 Introduction

In this project, we have divided the training and testing process of the ML code. We use the FashionMNIST dataset to create a classification model. This model is created and trained on Python using Tensorflow. This gives us the ability to make the model more efficient and accurate as it is easy to do so in Python. This model is then converted to ‘universal file format’ (uff). A cuda engine is created using this model.

2.0 Training

We use a simple tensorflow model to train the dataset. The training accuracy reaches around 96%

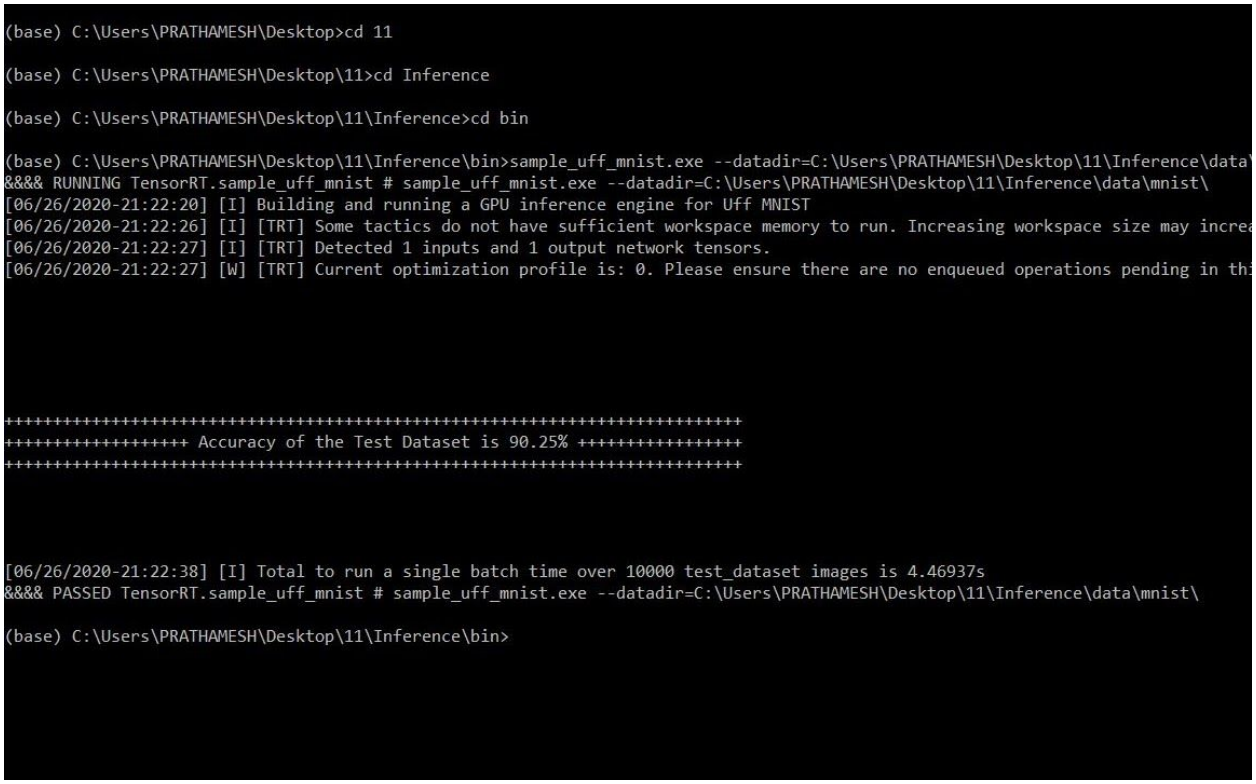




3.0 Testing the Tensorflow Model

Inorder to get a better understanding and do a comparative study of the latency and accuracy of both the Tensorflow model and the C++ Cuda Engine, model was tested on fashion_mnist test dataset of 10,000 images.
An **accuracy of 90%** was achieved by the model and the **latency was 10 seconds.**

Building a Cuda Engine using TensorRT's C++API
In order to test the dataset in the C++ model, a Cuda Engine was built. We used the .uff file as weights for this engine.
After building the engine we fed it the same 10,000 images of the fashion_mnist test dataset and tested the model. An **accuracy of 90%** was achieved and the **latency was 4.46 seconds**



4.0 Inference

The latency of the model decreases significantly in the C++ API when compared to the tensorflow model in Python while the accuracy remains intact.

5.0 Findings:

---

1. Python is a good language for coding and optimising your model.
2. It however lacks the speed and efficiency of a machine level language setup.
3. CUDA Engine is a powerful production tool.