Reflection

The aim of this project was to train a explore the use of parallel computing when training a convolutional neural network (CNN) for image classification. Specifically, I wanted to understand how using the parallel computation processes of GPUs can increase the training speed of more complicated convolutional networks. Naturally, the use of these more complicated models necessitated the understanding of many more deep concepts than those of a simple CNN.

The dataset was discovered early on. It did not take much research to discover that convolutional neural networks are at the heart of image classification machine learning systems. The programming was completed entirely in python. Most of the resources used in this project have been cited, including various parts of code. Every code resource had to be understood, and adapted for the my needs. In the literature, I read about the use of ResNet as a more advanced model, and I explain many of its concepts.

Since this was a solo project, I feel that the depth of mathematical understanding in the project is adequately shown in the notebook file itself, since there in so ambiguity as to who wrote it.

The overall method of the project was as follows:

1. Find a suitable dataset.
2. Implement a simple CNN image classifier.
3. Improve this classifier — this requires the use of GPU to speed up computation time.
4. A look at how GPU computation fares against that of CPU.
5. Finally, implementing a model (ResNet) that uses more state-of-the-art concepts to reach a higher accuracy, while also utilising various optimisation concepts.

Main challenges, and overcoming them:

1. Firstly, a challenge of this project was understanding what a CNN does, and how it works. Many concepts of a CNN are subtle, and I try to some of them in the project. Of course, I would have loved to spend many weeks on this project delving further into the nuances and subtleties of creating a successful image classification model, but there was only limited time: another challenge.
2. The timeframe to complete this project is very short. The brief requires an in-depth look into many tricky concepts. The ideas, and mathematics, behind the working and implementation of a CNN are very complicated. That is before considering that the focus of the project also warranted a focus on parallel computing. I tried my best to fit in as much explanation as possible within the timeframe, and I think that the layout out the project was successful.
3. Furthermore, the project requires implementing tricky concepts with code. Luckily, there are many resources online; however, the concepts are complicated, and the code is complicated. I try to provide clarification where possible.
4. Lack of GPU: My laptop does not have a dedicated GPU, and therefore I had find a solution to this, since most parallel computing uses GPUs. It took some

time, but I discovered Google Colab; here, I was able to use a cloud GPU; however, this proved to be financially expansive.

Final remarks:

Various improvements could be made. This includes adding more GPUs and using them in parallel to make more efficient the training of CNNs. This is a change that would be implemented were the data be increased by a factor of 1000. Furthermore, in the real world, image data has often a far larger number of pixels, as well as overlaps in the classes. This would require a more complicated neural network. Such neural networks have been used on the more complicated ImageNet dataset. [1]

Ultimately, I think the project was a success. Various theoretical graphs were replicated from literature using experimental analysis. Many clear and illustrative graphs were also used to provide clarity to the functioning of the code, and the results output.

References

[1] Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM.