Handover Document

# Introduction

The project aimed to leverage High-Performance Computing (HPC) techniques to parallelize the Contrastive Language-Image Pre-Training (CLIP) model,a ML model designed for learning visual concepts from natural language supervision. This initiative was a collaborative effort between the Deep Learning (DL) and HPC branches, aiming to combine their expertise to optimise and enhance the performance of the CLIP model on a large scale.

# Project Summary

**Objective:** Parallelize the CLIP model using HPC techniques.

**Initial Phase:**

* Collaborative team: 3 DL members and 3 HPC members.
* Focus: Parallelize a CNN model to build foundational knowledge.
* Outcome: Successfully parallelized the CNN model.

**Transition to New Team:**

* New team: All HPC members.
* Requirement: Upskilling to understand deep learning model parallelization.

**Research and Methodology:**

* Explored various parallelization techniques.
* Chose data parallelization as the preferred method.
* Investigated different libraries: PySpark, TensorFlow, PyTorch.

**Resources:**

* Utilized the M3 massive supercomputer from MDN.

**Outcome:**

* Successfully parallelized the CLIP model across multiple nodes.

# Background Information

What is a convolutional neural network for image recognition?

For image recognition, a traditional neural network will use individual pixels, whereas a convolutional neural network (CNN) will take into account larger regions of the image to use the context around pixels for better accuracy.

What is a CLIP model?

A clip model uses text and image embedding to map similar images and prompts in a similar direction while unrelated images and prompts are mapped far from one another which allows for unseen images to be recognised and classified.

# Tasks

The ultimate goal was to parallelise OpenAI’s CLIP model. To do this, we first worked with a convolutional neural network for image recognition. We explored multiple different methods of parallelism and eventually settled on using data parallelism to efficiently train the CNN. We then applied what we had learned to the CLIP model and managed to parallelise it.

# Challenges

Things we tried (and didn’t work):

* Manually parallelising (didn’t work due to GIL)
* CPU parallelisation
* Single GPU and CPU parallelisation
* Pyspark (only machine learning, can’t do CNN because not deep)

Thinks that worked:

* Strudel (access to multiple GPUs)

# Set Up

# Future work

# Resources

# Contact Information

## 