Final Project Report - Anh Vu

In my final project implementation, I was able to cover two of my chosen topics (GPU Thread Parallelism and Thread Synchronization). However, I was not able to reach my intended third concept: Files and File Systems. This report will provide a high-level overview of my intentions and my envisioned implementations.

Firstly, I intend to use Files and File Systems in the application of saving and loading neural networks. This is a practical use of the concept because saving the training progress of a network has been proven to be beneficial for many purposes, especially transfer learning (using and improving on trained models). For a project that wishes to provide a neural network library for developers, this feature proves to be essential.

In terms of actual implementation, I take inspiration from the "Archive Printer" assignment. We would have a custom binary file with a known format and layout of data, which can then be consumed modularly and systematically in code. For instance, the first section of the file would contain all parameters about the network: dimension of input layer, dimension of output layer, number of layers, learning rate, number of epochs to train, loss function (type `enum`). These parameters are all either integers (type `size_t`) or floats, both of which have a fixed number of bytes used to represent them. Therefore, we would know the total size of this section, and as long as we consume them in the same order they are produced, we can successfully obtain the architecture of the network.

After the first section, the following sections of the file would contain the architecture and learned weights and biases of each layer. Since we have consumed the number of layers present in the network, we know how many such sections exist in the binary file. For each layer, since there is again a set number of fixed-sized parameters, we can safely consume the architecture of each layer. For the weights and biases, they are fortunately again floats that are fixed-sized. To derive the number of weights needed to be consumed at each layer is slightly complicated because it involves the number of neurons in both the current and previous layer, but this parameter is part of the `layer_t` struct and should also be present in the current layer's saved architecture.

In short, following this scheme should allow us to load the network (consuming the file), and the saving process (producing the file) is strictly the reverse. Given more time, I will be able to and genuinely intend to explore and pursue this portion of the project to its completion.