CSCE614: Term Project Individual Report

Keshav Kishore (UIN:335005932)

Preparatory Learning

To contribute effectively to this project, I had to learn the fundamental concepts of deep neural network training. This included understanding the architecture and functioning of neural networks, activation functions like ReLU and sigmoid, optimization techniques such as gradient descent and the Adam optimizer, as well as concepts like backpropagation, loss functions (e.g., mean squared error and cross-entropy), and the importance of hyperparameter tuning. Additionally, I learned about the different types of layers in DNN models, such as fully connected layers, classifier layers, pooling layers, and batch normalization layers.

Contributions

Source Code

- I implemented the baseline VGG13 model and tested it using the CIFAR-10 dataset. For evaluation, I recorded the test accuracy and the total time taken to train the model.
- Using the tensor reorganization function developed by my teammate Eshaan Mandal, I implemented ADA-GP function for VGG13 and MobileNetV2. For both of these implementations, I recorded the test accuracy, layer-wise MSE loss, and the total time taken to train the model. Additionally, I also recorded the time taken in the warm-up phase and the time taken in Phase-GP, which were needed to evaluate the speedup achieved through gradient prediction.
- I also helped in formulating the case study our group performed on VGG11. By varying the hyperparameters, we selected the most optimal values for maintaining high test accuracy and reasonable training time.

Documentation and Presentation

- The sections I worked on include *Proposed Solution*, *Implementation Setup*, *Evaluation Methods*, *Results*, and *Conclusion*.
- I also prepared the presentation slides for the above-mentioned sections.

Problems Faced

- It was difficult to understand the dimensionality of the different layers in VGG13 and MobileNet, and how to map the output tensor of the Predictor model to each layer of the DNN model. To resolve this, we used pooling layers to standardize the dimensions of the input tensor to a fixed size.
- When implementing the gradient-prediction function for more complex DNN models like ResNet-150, I encountered CUDA: Out-of-Memory errors due to the large number of back-propagated gradients that needed to be stored to train the Predictor model. Additionally, for these models, the training time was also very high. Thus, to proceed, I chose less complex models like VGG13 and MobileNet-V2.

Acknowledgement

I would like to thank my teammates, the other groups working on this project, and the TA, Sabuj, for engaging in discussions. These discussions helped improve my understanding of the concept.