Optimizing Communication Efficiency of Deep Learning Parallelism Techniques in the Inference Phase

Overview

Deep Learning models are becoming increasingly larger so that most of the state-of-the-art model architectures are either too big to be deployed on a single machine or cause performance issues such as undesired delays.

This is not only true for the largest models being deployed in high performance cloud infrastructures but also for smaller and more efficient models that are designed to have fewer parameters (and hence, lower accuracy) to be deployed on edge devices. That said, this project considers the second environment where there are multiple resource constrained machines connected through a network.

When distributing deep learning models across multiple compute nodes, trying to realize parallelism, certain algorithms (e.g., Model Parallelism) are not able to achieve the desired performance in terms of latency, mainly due to (1) communication cost of intermediate tensors; and (2) inter-operator blocking. This project consists of multiple sub-projects each can be taken separately.

In the context of Model Parallelism, two potential modifications can be considered:

1. Pipeline parallelism by delaying the inference of the first few data samples assuming a live stream of input data.

2. Finding certain points in deep learning architectures or modifying the architecture itself so that for each data sample, it becomes possible to filter out some sub-parts of the model, and therefore reducing the transmitted data, and still achieve comparable accuracy.

Class and Variant Parallelism improve inter-node communication significantly. However, the input data needs to be shared between contributing nodes. The goal is to propose a technique to transmit less data, and to find a good trade-off between computation and communication.

Note that this project mainly focuses on the inference (i.e., serving) phase of deep learning algorithms, and although efficiency of the training phase can be considered as well, it has a much lower priority.

Type

- M.Sc. Thesis
- M.Sc. Research Internship
Tasks

1. Replicate one or two parallelism techniques.
   - Can be done as a Research Internship.

2. Evaluate communication overhead in different scenarios and use cases.
   - Video Analytics can be used as the application.

3. In Class Parallelism: Apply a technique similar to Early Exit, and then filter nodes (i.e. do not transmit intermediate data to them) that are unlikely to contribute to the final prediction based on the early results.

4. Find a point and design a bottleneck layer to achieve a balance between communication and computation overhead. This is more valuable if it can be changed dynamically during runtime.

Further Study


