Optimization for Deep Learning Final Project: A study on autodiff in PyTorch

b09902033 Hsiu-Hsuan Wang b09902040 Yu-Kai Hong

1 Abstract

We measure the different runtime of the gradient flowing backwards in PyTorch and Tensorflow and give practical suggestions under different scenario. Tensorflow implements static computation graph, which allow its model to be compiled and optimized to run faster. PyTorch library does not explicitly implement the tape, but dynamically generate the computation graph, which allow more flexible and dynamic model structure (e.g. sequence-to-sequence model).

2 Problem Definition

After the forward pass, models would build a computation graph to store each operation and variable. Then the model is updated by the gradient of loss function. The gradient of the last layer output is computed and flow backward to update all the parameters.

In this report, we assume that all the parameters are trainable. Therefore, the time of one back propagation is equal to the computing time for a whole Jacobian matrix with regard to the loss function and model parameters.

Hypothesis 0 does the above interpretation true?

Hypothesis 1 The backward flow of Tensorflow should be faster than PyTorch since the models in Tensorflow are compiled before running.

Next, we measure the running time of several famous basic models including self-attention layer, convolution layer, fully-connected layer with cross-entropy loss and ctc-loss

3 Experiments

Compute the Jacobian of Dense Layer

It is worth noting that we have to wrap the Jacobian function of Tensorflow with the @tf.function decorator to speed up the computation by running in graph mode instead of eager mode. The results are in Table 1

Table 1: Dense layer results.

	Runtime
Tensorflow (eager)	0.20787 (0.03954)
Tensorflow (graph)	0.00139 (0.00369)
PyTorch	0.00062 (0.00022)