High Performance Machine Learning
Lab 2

Robert Benke
Department of Computer Architecture
Faculty of Electronics, Telecommunications and Informatics
Gdańsk University of Technology

1 Introduction

The main functionality of a training program in machine learning is executing the training operations, which in TensorFlow are represented by the computational graph. By default tensorflow (2.0+) uses eager execution of computational graphs. Using eager execution is more intuitive and much easier to debug. However, it comes with the cost of performance. There are at least two reasons for that. The computational graph is not optimized and some parts (like if statements or loops) of the code are interpreted and executed by Python.

In this lab we will practice using graph mode and examine the performance (execution time) depending on different methods (Grappler, XLA).

2 Task 0: Measure the execution time of eager mode (1 point)

Measure the execution time of the first 5 iterations of the first epoch for CPU and GPU.

To pass the task: Print the execution time of train_step for the first 5 minibatches. Save results somewhere for further comparison with other methods.

3 Task 1: Use graph mode instead of eager execution (3 points)

Tensorflow uses graph mode internally for some of the core functionality (e.g. model.fit). Whereas, custom functions have to be compiled explicitly. For this purpose we have tf.function decorator (https://www.tensorflow.org/guide/function).

The aim of this task is to change the execution mode from eager to graph and measure the performance. Tracing (building computational graph) can be

expansive and sometimes results in higher execution time than eager mode. For that reason, we want to test graph compilation of the whole training function, training step and only forward pass. More about optimizing graph execution can be found on https://www.tensorflow.org/guide/graph_optimization.

To pass the task: Print the execution time of train_step for first 5 minibatches for compiled (Grappler) train function (0.5+0.5), train_step function (0.5+0.5) and model.__call__ (0.5+0.5) for CPU and GPU. Save the results somewhere for further comparison.

4 Task 2: Optimize computation for particular hardware with XLA (4 points)

Graph mode use only hardware agnostic optimizations. Further improvement can be achieved with XLA (https://www.tensorflow.org/xla).

To pass the task: Print the execution time of train_step for first 5 minibatches for compiled (XLA) train function (0.5+0.5), train_step function (0.5+0.5) and model.call (0.5+0.5). Be able to discuss and explain all results (1 point).