Accelerating Model Search with Model Batching
Deepak Narayanan, Keshav Santhanam, Matei Zaharia
Stanford University

Overview
- Model Search and Hyperparameter Search for deep learning models extremely resource-intensive (~15k GPU hrs [2])
- GPUs often not saturated while training each model independently
  - Especially for smaller models / recurrent neural networks [1]
- Motivating question: Can we leverage the fact that multiple models are being trained to better utilize GPUs, and hence drive down total search time?
- ModelBatch launches kernels associated with different models concurrently, while also sharing preprocessing among the different models, leading to a throughput increase of up to 9.2x

Evaluation

**Figure 2:** Overhead of preprocessing (percentage of time spent in the preprocessing step) while training a stripped-down AlexNet model on TensorFlow.

**Figure 3:** Throughput increase from using 16 CUDA streams for two different fully connected layers; common minibatch sizes are 16-256.

**Figure 4:** Throughput comparison of ModelBatch with TensorFlow, with (MBS=16) and without model batching, for synthetic (top) and real images (bottom). Results are presented for a 5-layer AlexNet-like model on the ImageNet dataset, for training (left) and inference (right).

Design

ModelBatch has two key components:
- Shared preprocessing: Preprocessing performed on the CPU, and shared among the different models. After preprocessing, processed tensors are moved to the GPU asynchronously; this communication is overlapped with compute using double buffering.
- Parallel model training: Performed on the GPU on a per-model basis, contains operations like convolutions, matrix multiplications, and pooling. Kernels launched in parallel using CUDA streams.

In addition, convolutions (which typically dominate compute time for CNNs) can be implemented using a number of different algorithms, which require different amounts of scratch space:
- Need to pick the right amount of scratch space to give each convolution algorithm based on the number of models run in parallel, and the size of the convolutions.

Future Work

- Evaluation on sequence-to-sequence models like LSTMs and GRUs
  - Especially since these models underutilize GPUs [1]
- Evaluation on end-to-end model search / hyperparameter search applications [2, 3]
  - In particular, how do we batch non-identical models?
- Can we use similar techniques when performing distributed training?
  - Can we use the fact that we’re training multiple models to hide the latency of distributed communication?

[1] Optimizing Performance of Recurrent Neural Networks on GPUs. Appleyard et al.