TESTING THE IMPLEMENTATIONS OF A PARALLEL BATCH TRAINING ALGORITHM FOR DEEP NEURAL NETWORK

YUPING LIN

IFLYTEK LABORATORY FOR NEURAL COMPUTING FOR MACHINE LEARNING

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

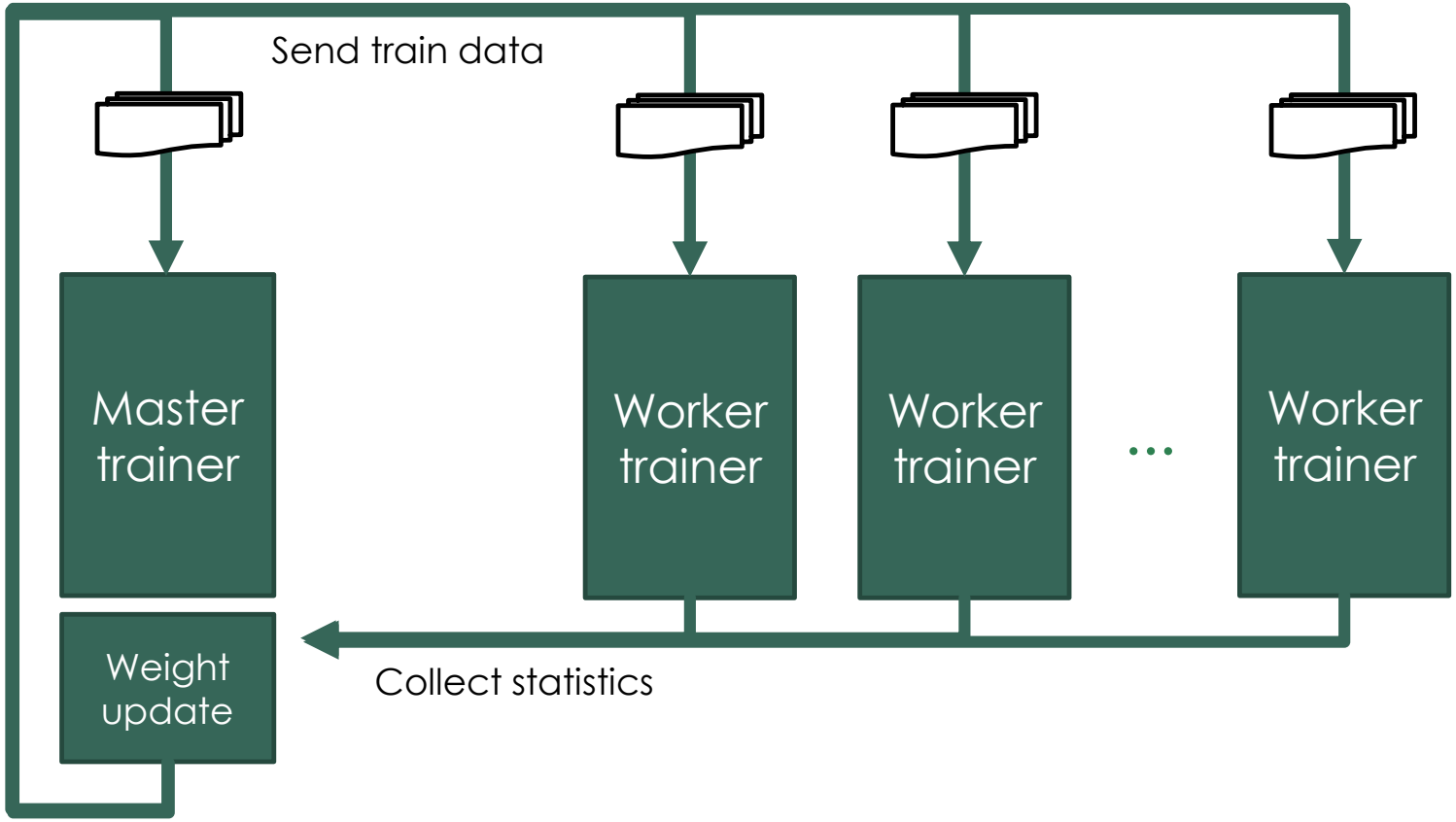
YORK UNIVERSITY, TORONTO

DECEMBER 1, 2015

OUTLINE

- Review
- Testing scheme
  - Metric
  - parameters
- Experiment setup
  - Data set
  - Testing harness
  - Testing environment
  - Implementations to be tested
- Results

REVIEW OF THE ALGORITHM



- Concurrent training

TESTING SCHEME

- Use response time as testing metric
- The response time in this experiment is defined as the time used to train one epoch

- Response time is used because:
  - The training of epochs are serialized steps
  - We are interested in minimizing the training time

TESTING SCHEME

- There are 2 parameters that can affect the response time:
  - Number of threads used
  - Mini-batch size

- Measure
  - Response time vs. number of threads used
  - Response time vs. mini-batch size

EXPERIMENT SETUP

- Use MNIST data set for hand written digits recognition task
- The data set consist of 60,000 28x28 pixel images for training and 10,000 images for testing.
- The samples are classified into 10 categories: 0 ~ 9
- Network model size : 784-300-300-10

EXPERIMENT SETUP

- The testing is only measured on the training of MLP, since the implementation of RBM is similar to the implementation of MLP.
- The testing harness:
  - Read in data
  - For n runs of test
    - Create model and trainer
    - Ask trainer to perform full training process
      - Trainer train epoch by epoch
- Time is measured only at the beginning and the end of the trainEpoch() method.
- Discard the first run of test
- The mean and standard deviation of response time are computed over all the response time measured in the remaining runs of test

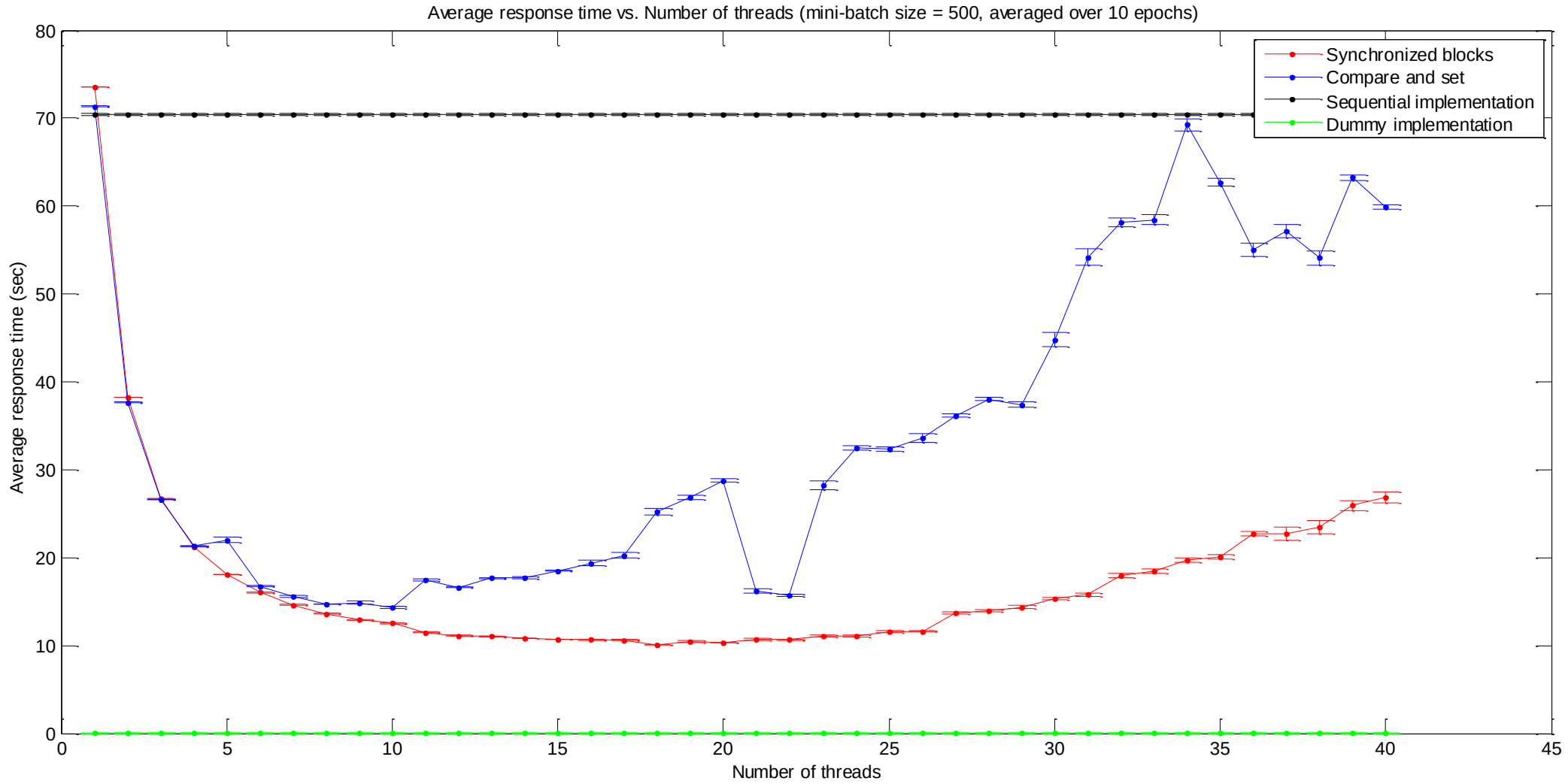
EXPERIMENT SETUP

- Experiment is conducted on the Intel Manycore Testing Lab (MTL)
- Cores available: 40; Cores used: 20
- OS: Linux
- JVM version: 1.7.0_01
- VM argument: -d64 -server -Xms1G -Xmx1G

EXPERIMENT SETUP

- Implementations to be tested:
  - SeqMLPTrainer (baseline): implementation of the sequential training algorithm
  - ConMLPTrainer: implementation of the parallel batch training algorithm that use synchronized blocks to synchronize access to shared variables
  - CASConMLPTrainer: implementation of the parallel batch training algorithm that use compare & set to synchronize access to shared variables
  - DummyMLPTrainer: does nothing in the trainEpoch() method

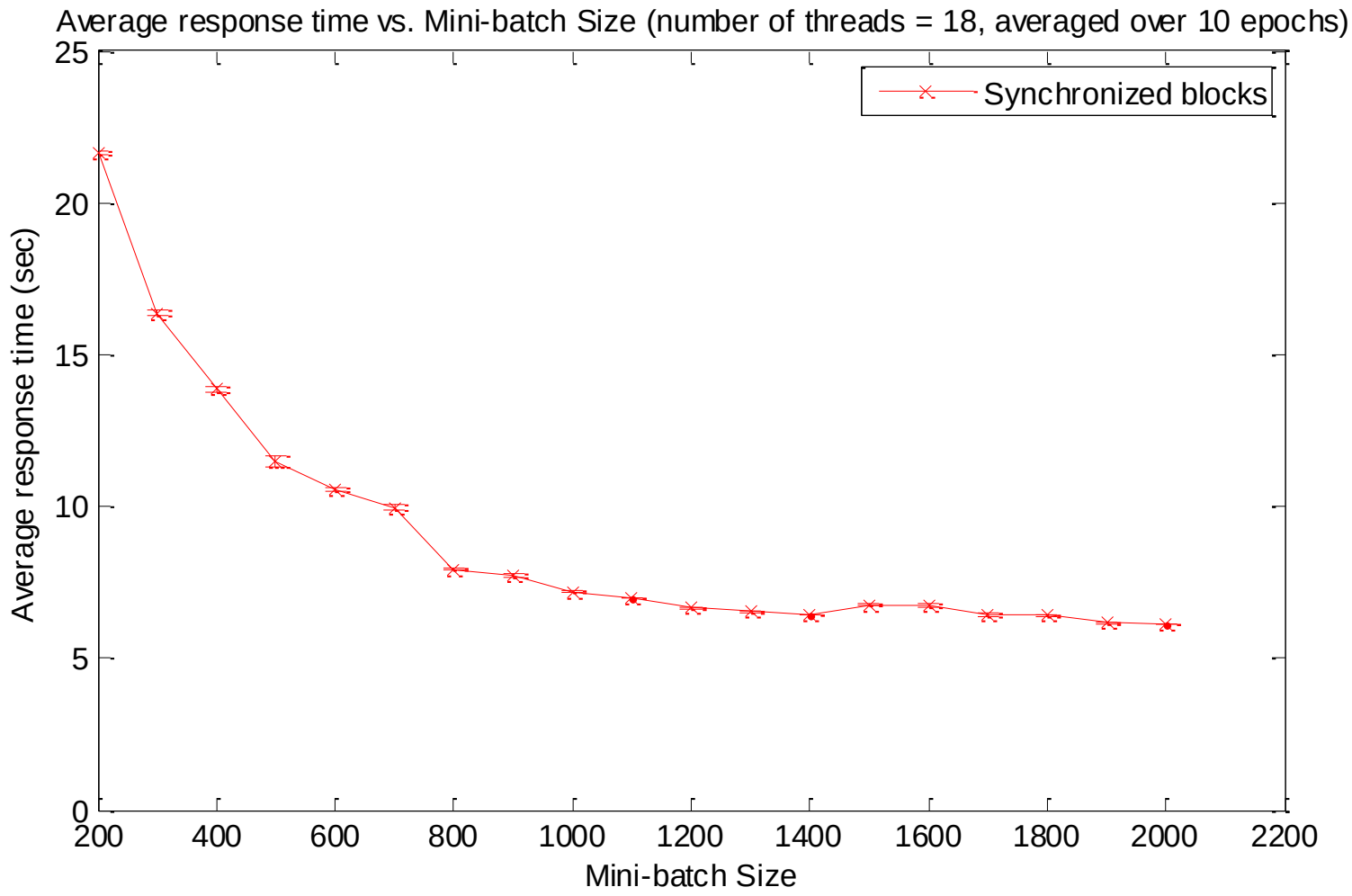
EXPERIMENT RESULTS



EXPERIMENT RESULTS

- The synchronized blocks implementation performs generally better than the compare and set implementation.
- This phenomenon is probably due the increased contention when using more threads.
- The best average response time of 10.07 sec/epoch is reached by the synchronized blocks implementation when using 18 threads.

EXPERIMENT RESULTS



EXPERIMENT RESULTS

- The average response time decreases as the size of mini-batch increases.
- The average response time asymptotically approaching around 6 sec/epoch, which is more that 10 times faster than the sequential implementation.

EXPERIMENT RESULTS

- Parallelization efficiency:
- $E = \frac{t_{seq}/t_{con}}{nThr} = \frac{70.41\ sec/10.07\ sec}{18} = 38.84\%$

EXPERIMENT RESULTS

- The classification performance of the trained model:
  - Only reaches 11.35% accuracy without pre-trained by RBM (trained 100 epochs)
  - Achieved 94.52% accuracy when pre-trained by RBM (trained 100 epochs)

- Stat-of-the-art: 99.77% accuracy by Multi-column DNN (D. Ciresan et al., 2012)

THANK YOU