Sponsored by the Princeton Institute for Computational Science and Engineering (PICSciE) and co-sponsored by the Center for Statistics and Machine Learning (CSML) and OIT Research Computing

https://researchcomputing.princeton.edu/TensorFlowPyTorchUserGroup
Meeting Calendar

Thursday, July 25 - Today

August - No meeting

Thursday, September 12 - P. Hawkins, Google AI Princeton

October - Regular meeting

November - Regular meeting

December - No meeting

January - Regular meeting
TensorFlow & PyTorch User Group

Thursday, July 25, 4:30-5:30 pm, 138 Lewis Science Library

Running TensorFlow 2.0 (beta) on the PU HPC Clusters
Jonathan Halverson (PICSciE), Princeton University

GPU Computing with R and Keras
Danny Simpson (LSI), Princeton University

Big Data of Big Tissues: Deep Neural Networks to Accelerate Analysis of Collective Cell Behaviors
Julienne LaChance (MAE), Princeton University

Selene: A PyTorch-based Deep Learning Library for Sequence Data
Kathleen Chen (Center for Computational Biology), Flatiron Institute, NY, NY
Special Meeting

TensorFlow and PyTorch User Group

Thursday, September 12, 4:30-5:30 pm, 120 Lewis Science Library

JAX: Accelerated Machine Learning Research via Composable Function Transformations in Python

Peter Hawkins, Google AI Princeton
ON COMPUTING WELL

Installing and Using Tensorflow with R

Posted on June 26, 2019 by Danny Simpson

This tutorial implements the basic MNIST classification tutorial using the R interface to Keras with Tensorflow backend. In particular, this guide is aimed at using the GPU nodes available on Princeton’s HPC Clusters (Adroit and Tiger). This tutorial can also be found on GitHub at: https://github.com/simpsondl/mnist_r_tutorial

Getting Started with TensorFlow 2 (beta) on the PU HPC Clusters

Posted on July 18, 2019 by Jonathan Halverson

In TensorFlow 2, eager execution is now the default, Keras is officially the high-level API and distributed training has been improved. This post explains how to get started with TensorFlow 2 (beta) on the HPC clusters.
Installing and Running TensorFlow version 1.x with Python Interface
https://github.com/PrincetonUniversity/slurm_mnist

Installing and Running TensorFlow version 1.x with Keras R Interface

Installing and Running TensorFlow version 2.0 (beta)

Installing and Running PyTorch
https://github.com/PrincetonUniversity/install_pytorch
NEW CLUSTER SOON

Traverse

- 46 nodes (2x POWER9, 4x NVIDIA V100 (32 GB), GPU Direct)
- mini-Sierra
- Priority to PPPL
- Available around mid-August
TensorFlow 2.0 (beta)

- March 6 (alpha)
- June 7 (beta)
- Official release “by the end of the year”
- Ease of use is top priority
- Keras as official high-level API
- Eager execution is the default
- Remove duplicate functionality
- Better distributed training
Version 1

```python
import tensorflow as tf

mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([tf.keras.layers.Flatten(input_shape=(28, 28)),
                                     tf.keras.layers.Dense(128, activation='relu'),
                                     tf.keras.layers.Dropout(0.2),
                                     tf.keras.layers.Dense(10, activation='softmax')])

model.compile(optimizer='adam',
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Version 2

```python
import tensorflow as tf

mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([tf.keras.layers.Flatten(input_shape=(28, 28)),
                                     tf.keras.layers.Dense(128, activation='relu'),
                                     tf.keras.layers.Dropout(0.2),
                                     tf.keras.layers.Dense(10, activation='softmax')])

model.compile(optimizer='adam',
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```
TF v1.0 Script

```python
learning_rate = 0.01
optimizer = tf.train.MomentumOptimizer(learning_rate=learning_rate, momentum=0.95, use_nesterov=True)
training_op = optimizer.minimize(loss)

with tf.name_scope('my_metrics'):
    thres = 0.5
    y_pred = tf.cast(prob_positive > thres, tf.int32)
    cm = tf.confusion_matrix(labels=y, predictions=y_pred, num_classes=2, dtype=tf.float64)
    accuracy = tf.trace(cm) / tf.reduce_sum(cm)

init = tf.global_variables_initializer()

with tf.Session() as sess:
    init.run()
    for epoch in range(n_epochs + 1):
        for iteration in range(train_paths.size // batch_size):
            X_batch, y_batch = fetch_batch(train_paths, train_labels, batch_size)
            #print max_pool_2.shape
            sess.run(training_op, feed_dict={X:X_batch, y:y_batch, training:True})
        if not (epoch % 10):
            print(epoch, "----------")
            print("Accuracy (train):" , accuracy.eval(feed_dict={X:X_train, y:y_train}))
            print("Accuracy (test):" , accuracy.eval(feed_dict={X:X_test, y:y_test}))
            print("Loss (train):" , loss.eval(feed_dict={X:X_train, y:y_train}))
            print("Loss (test):" , loss.eval(feed_dict={X:X_test, y:y_test}))
```
Converting Scripts

tf_upgrade_v2 --infile version1_script.py --outfile version2_script.py

TensorFlow 2.0 Upgrade Script
-------------------------------
Converted 1 files
Detected 0 issues that require attention
-------------------------------

41:13: INFO: Added keywords to args of function 'tf.nn.max_pool'
41:13: INFO: Renamed keyword argument for tf.nn.max_pool from value to input
41:13: INFO: Renamed 'tf.nn.max_pool' to 'tf.nn.max_pool2d'

54:12: INFO: tf.losses.log_loss requires manual check. tf.losses have been replaced with object oriented versions in TF 2.0 and after. The loss function calls have been converted to compat.v1 for backward compatibility. Please update these calls to the TF 2.0 versions.
54:12: INFO: Renamed 'tf.losses.log_loss' to 'tf.compat.v1.losses.log_loss'
57:12: INFO: Renamed 'tf.train.MomentumOptimizer' to 'tf.compat.v1.train.MomentumOptimizer'
Distributed Training

- tf.keras.utils.multi_gpu_model is working
  - Model copied on different devices

- tf.distribute.MirroredStrategy is working
  - In-graph replication where graph is distributed over devices

- tf.distribute.experimental.MultiWorkerMirrored
Installing TensorFlow 2 (beta) on Adroit/TigerGPU

module load anaconda3
conda create --name tf2b-gpu python=3.6
conda activate tf2b-gpu
pip install --user tensorflow-gpu==2.0.0-beta1

Conda does not support TF2 at the moment so install from PyPI.
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:05:00
#SBATCH --mem-per-cpu=4G
#SBATCH --gres=gpu:1

cudnn=/usr/local/cudnn/cuda-10.1/7.5.0/lib64
cupti=/usr/local/cuda-10.0/extras/CUPTI/lib64
export LD_LIBRARY_PATH=$cudnn:$cupti:$LD_LIBRARY_PATH

module load anaconda3
conda activate tf2b-gpu

srun python mnist_classify.py
References

Installing TF2 (beta) on PU HPC Clusters
https://oncomputingwell.princeton.edu

TensorFlow 2.0 (beta)
https://www.tensorflow.org/beta/tutorials

New book by A. Geron and the accompanying Jupyter notebooks:
https://github.com/ageron/tf2_course