Stony Brook University

Comparative Study of Deep Learning Frameworks in HPC Environments

HamidReza Asaadi and Barbara Chapman
Institute for Advanced Computational Science
Stony Brook University,
Stony Brook, NY

New York Scientific Data Summit 2017

Motivation

- Use of machine learning to solve scientific problems is growing rapidly
- As the size of datasets grows, so does potential size of computation
  - Need for ML frameworks that run efficiently on clusters that are setup for typical scientific workload
  - For many scientists, ease-of-use is paramount

- Both HPC community and ML frameworks are investing heavily on GPUs
  - ML frameworks are being deployed in some in HPC environments
  - **But,** how well do the frameworks exploit the resources and how usable are they for domain scientists

Our immediate goal

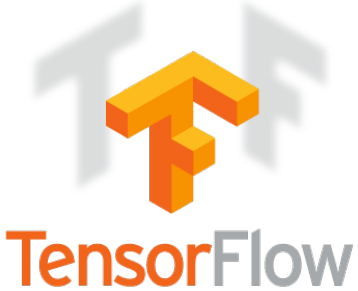
- How compatible ML frameworks are (right-now) with typical HPC clusters?
- What can be done to improve the compatibility of these frameworks?

Stony Brook University

Agenda

- Introduction and Motivation
- Background and Test Setup
  - ML Frameworks
  - Datasets
  - Hardware Infrastructure
- Experimental Results
- Discussion and Future Work

Target Frameworks



Tensorflow

- Developed and maintained by Google
  - Initially released in late 2015
- Supports Linux, macOS, Windows, Android and iOS platforms
- Python API
- Designed to run on multiple CPUs and GPUs
- Multi-node distribution support is an afterthought
  - Modifications in the code is required
  - Launching a multi-node applications is not trivial
  - Performance drop is compare to single node execution

```
import numpy as np
import tensorflow as tf

# Model parameters
W = tf.Variable([.3], dtype=tf.float32)
b = tf.Variable([-.3], dtype=tf.float32)
# Model input and output
x = tf.placeholder(tf.float32)
linear_model = W * x + b
y = tf.placeholder(tf.float32)
# loss
loss = tf.reduce_sum(tf.square(linear_model - y)) # sum of the squares
# optimizer
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)
# training data
x_train = [1,2,3,4]
y_train = [0,-1,-2,-3]
# training loop
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init) # reset values to wrong
for i in range(1000):
  sess.run(train, {x:x_train, y:y_train})

# evaluate training accuracy
curr_W, curr_b, curr_loss = sess.run([W, b, loss], {x:x_train, y:y_train})
print("W: %s b: %s loss: %s"%(curr_W, curr_b, curr_loss))
```

Source: https://www.tensorflow.org/get_started/get_started

```
import numpy as np
import tensorflow as tf

# Model parameters
W = tf.Variable([.3], dtype=tf.float32)
b = tf.Variable([-.3], dtype=tf.float32)
# Model input and output
x = tf.placeholder(tf.float32)
linear_model = W * x + b
y = tf.placeholder(tf.float32)
# loss
loss = tf.reduce_sum(tf.square(linear_model - y)) # sum of the squares
# optimizer
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)
# training data
x_train = [1,2,3,4]
y_train = [0,-1,-2,-3]
# training loop
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init) # reset values to wrong
for i in range(1000):
  sess.run(train, {x:x_train, y:y_train})

# evaluate training accuracy
curr_W, curr_b, curr_loss = sess.run([W, b, loss], {x:x_train, y:y_train})
print("W: %s b: %s loss: %s"%(curr_W, curr_b, curr_loss))
```

Source: https://www.tensorflow.org/get_started/get_started

The Microsoft Cognitive Toolkit (CNTK)

- Developed by Microsoft
  - Initially released in early 2016
  - Significant update in April, 2017 (v2.0)
- Supports multiple CPUs and GPUs and distribution using MPI
- Supports Linux and Windows
- API for Python, C++ and Java (added in CNTK 2.0)
- BrainScript DSL

```
TrainConvNet = {
    action = "train"
    BrainScriptNetworkBuilder = [
        imageShape = 32:32:3
        featScale = Constant(1/256)
        labelDim = 3
        model (features) = {
            featNorm = Scale(features, featScale)
            h1 = LinearLayer {100,      init="gaussian", initValueScale=1.5} (featNorm)
            ol = LinearLayer {labelDim, init="gaussian", initValueScale=1.5} (h1)
        }.ol
        # inputs
        features = Input {imageShape}
        regrLabels = Input {labelDim}
        # apply model to features
        ol = model (features)
        # define regression loss
        diff = regrLabels - ol
        sqerr = ReduceSum (diff.*diff, axis=1)
        rmse =  Sqrt (sqerr / labelDim)
        featureNodes    = (features)
        labelNodes      = (regrLabels)
        criterionNodes  = (rmse)
        evaluationNodes = (rmse)
        OutputNodes     = (ol)
    ]
    SGD = {
        epochSize = 0
        maxEpochs = 2
        minibatchSize = 128
        learningRatesPerSample = 0.0005
        momentumAsTimeConstant = 1024
        firstMBsToShowResult = 5 ; numMBsToShowResult = 50
    }
```

Source: https://github.com/Microsoft/CNTK/blob/master/Examples/Image/Regression/RegrSimple_CIFAR10.cntk

Caffe2

- Developed by Facebook research
  - Released in 2017
- Based on UC Berkley Caffe framework
- Supports multi-machine distribution via Gloo library
  - Supports fast interconnects and GPU Direct
- Python API
  - Makes original Caffe neural networks (.proto) unusable in Caffe2

```
def AddLeNetModel(model, data):
    # Image size: 28 x 28 -> 24 x 24
    conv1 = brew.conv(model, data, 'conv1', dim_in=1, dim_out=20, kernel=5)
    # Image size: 24 x 24 -> 12 x 12
    pool1 = brew.max_pool(model, conv1, 'pool1', kernel=2, stride=2)
    # Image size: 12 x 12 -> 8 x 8
    conv2 = brew.conv(model, pool1, 'conv2', dim_in=20, dim_out=50, kernel=5)
    # Image size: 8 x 8 -> 4 x 4
    pool2 = brew.max_pool(model, conv2, 'pool2', kernel=2, stride=2)
    # 50 * 4 * 4 stands for dim_out from previous layer multiplied by the image size
    fc3 = brew.fc(model, pool2, 'fc3', dim_in=50 * 4 * 4, dim_out=500)
    fc3 = brew.relu(model, fc3, fc3)
    pred = brew.fc(model, fc3, 'pred', 500, 10)
    softmax = brew.softmax(model, pred, 'softmax')
    return softmax
```

Comparison Task Scope

- Test case
  - Image classification
  - Inception V3 trainer network (48-layer deep network)
    - Training with domain-specific data is common in scientific analysis
    - E.g. Synchrotron imaging, weather forecast satellite imaging, etc.
  - Inception V3 trainer implementation is not yet available for Caffe2
    - No test results on Caffe2
- Test data
  - CIFAR-10
  - Flowers
  - ImageNet (ILSVRC 2012)
- Hardware platform
  - SBU Seawulf Cluster

Datasets

Dataset	Classes	Training images	Validation images
CIFAR-10	10	50000	10000
ImageNet	1000	1,281,167	50000
Flowers	102	6150	1020

- Data transformation and repackaging

  Original dataset data: folder structured (by label) JPEG/PNG files

  Target input format:

  - Tensorflow: tfrecords
  - Caffe2: LMDB files
  - CNTK: txt list pointer to files

    `Example: dataset/flowers/1/44235_1.png 1`

Seawulf Cluster

Compute Nodes	164
Total CPUs/Total Cores	324/4592
GPU	8 nodes / total 64x GK210 (K40) Cores/ 159,744 CUDA cores
Memory	128 Gb/Node
Interconnect	Infiniband @40Gbps

2 of 3 Frameworks required recompilation + several dependencies

Resource Limitations

- 2 GPU nodes can be assigned to one job (total of 16 GPUs)
- Maximum walltime for GPU nodes is 8 hours
  - Checkpointing must be used to train networks
  - We did not measure checkpointing overheads

Test Environment Summary

Frameworks under test	CNTK (v2.0), Tensorflow (v.1.0.1)
Datasets	CIFAR-10, ImageNet (2012), Flowers
Test time	8 hours
Number of nodes	1 or 2

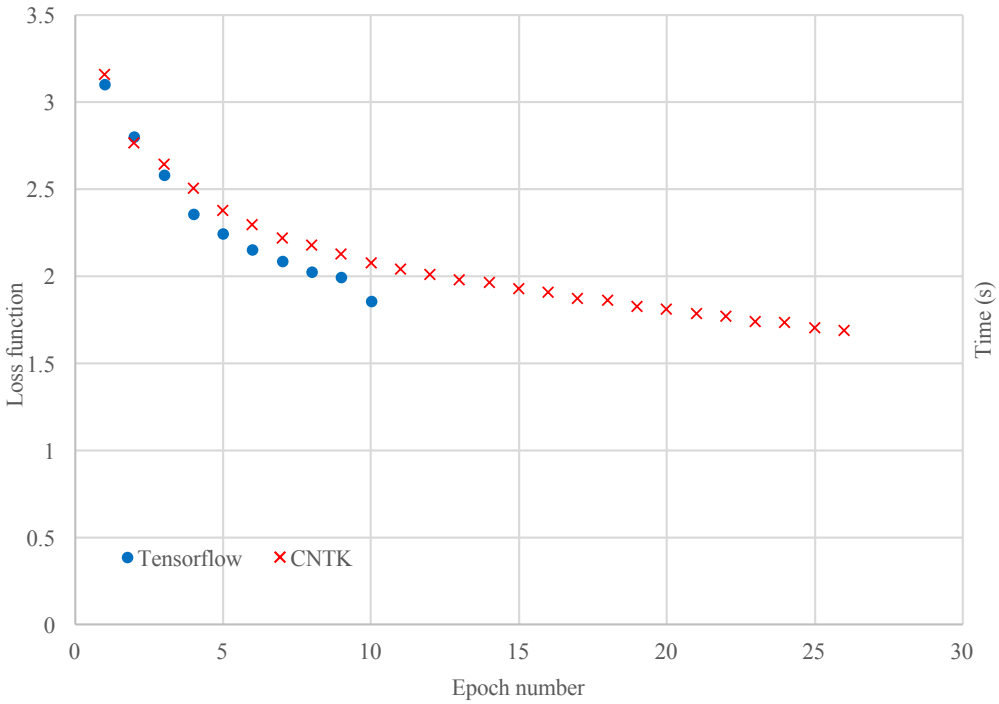
Dataset	Single-Node	Multi-Node
CIFAR-10	Yes	No
ImageNet	Yes	No
Flowers	Yes	Yes

Agenda

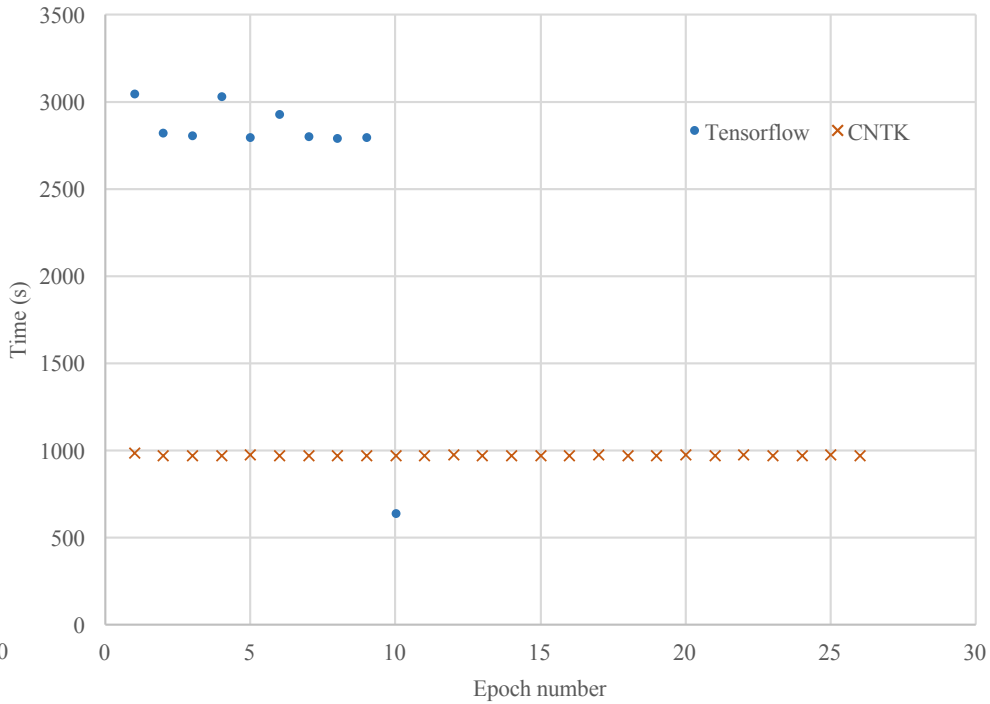
- Introduction and Motivation
- Background and Test Setup
  - ML Frameworks
  - Datasets
  - Hardware Infrastructure
- Experimental Results
- Discussion and Future Work

Stony Brook University

CIFAR-10 Dataset Results



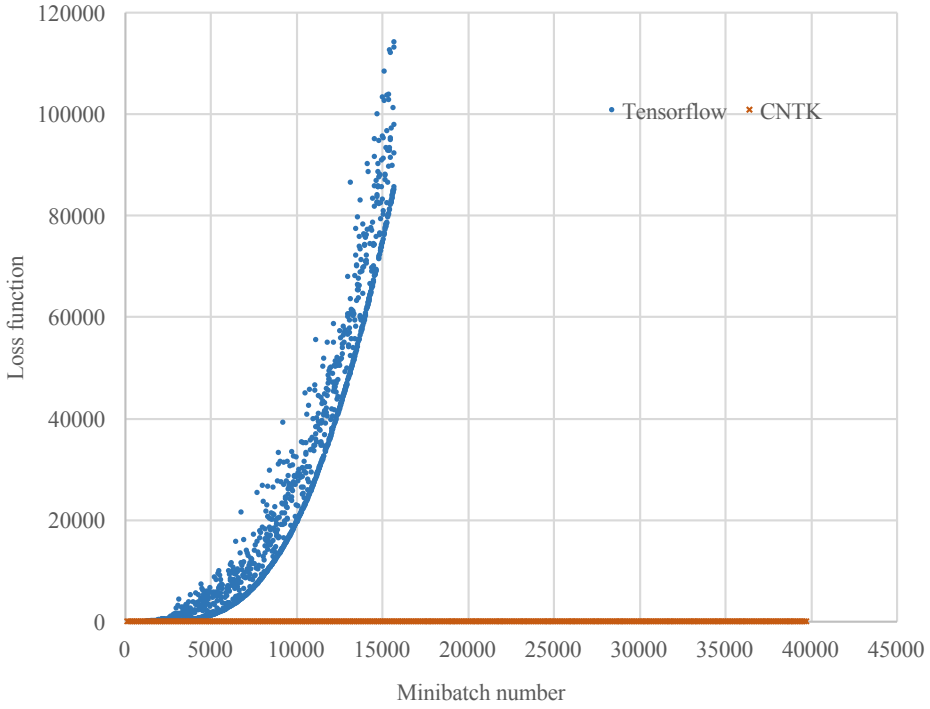
Loss function/Epoch



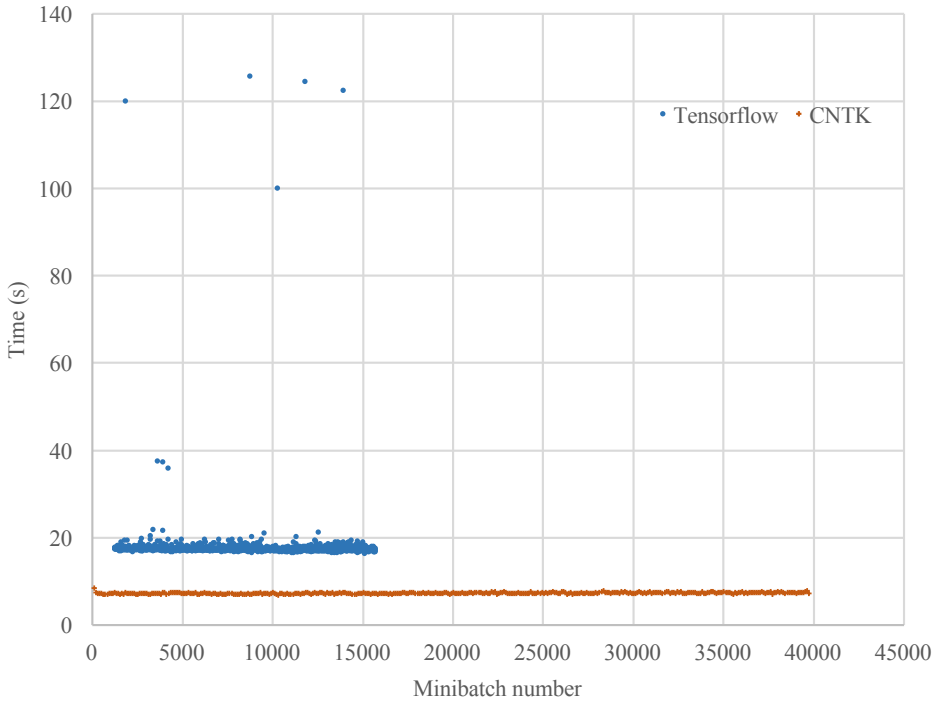
Time/Epoch

Stony Brook University

ImageNet Dataset Results



Loss function/Minibatch

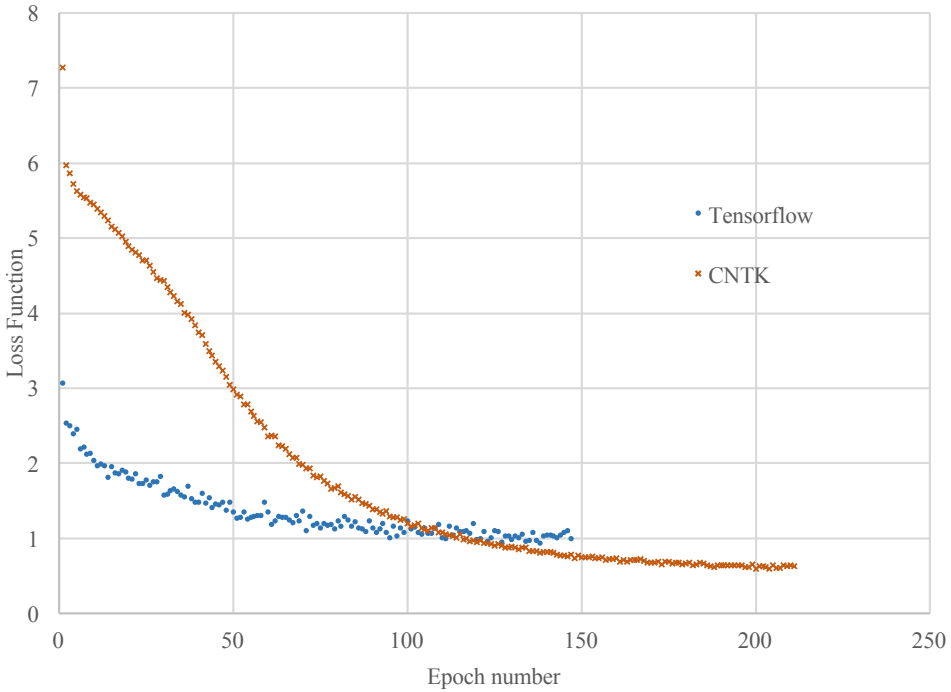


Time/Minibatch

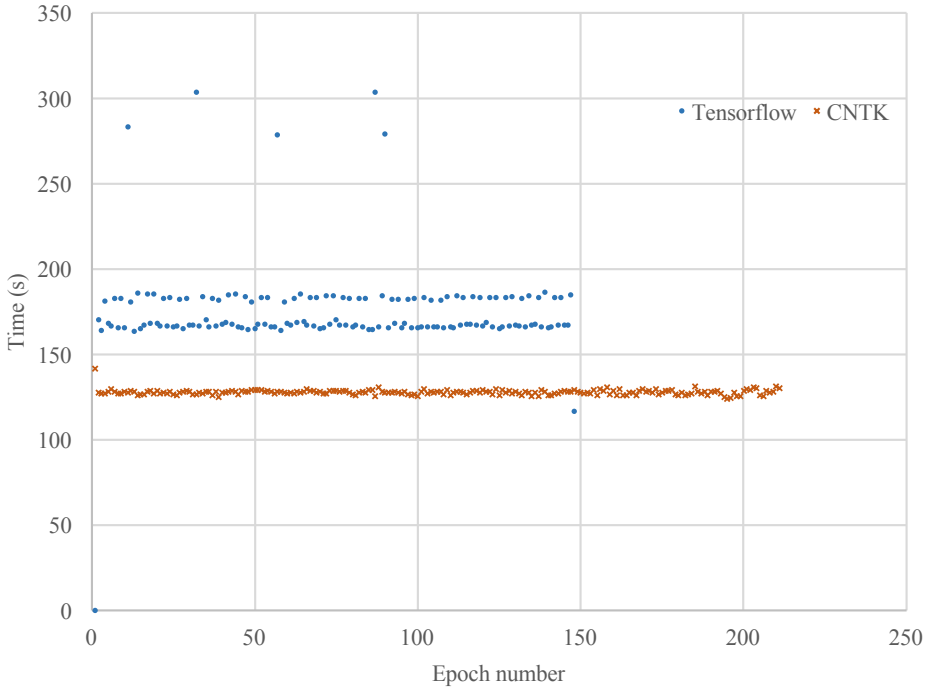
Dataset	Classes	Training images	Validation images
ImageNet	1000	1,281,167	50000

Stony Brook University

Flowers Dataset Results (Single Node)



Loss function/Epoch



Time/Epoch

Stony Brook University

Launching Tensorflow and CNTK on 2 nodes

Tensorflow

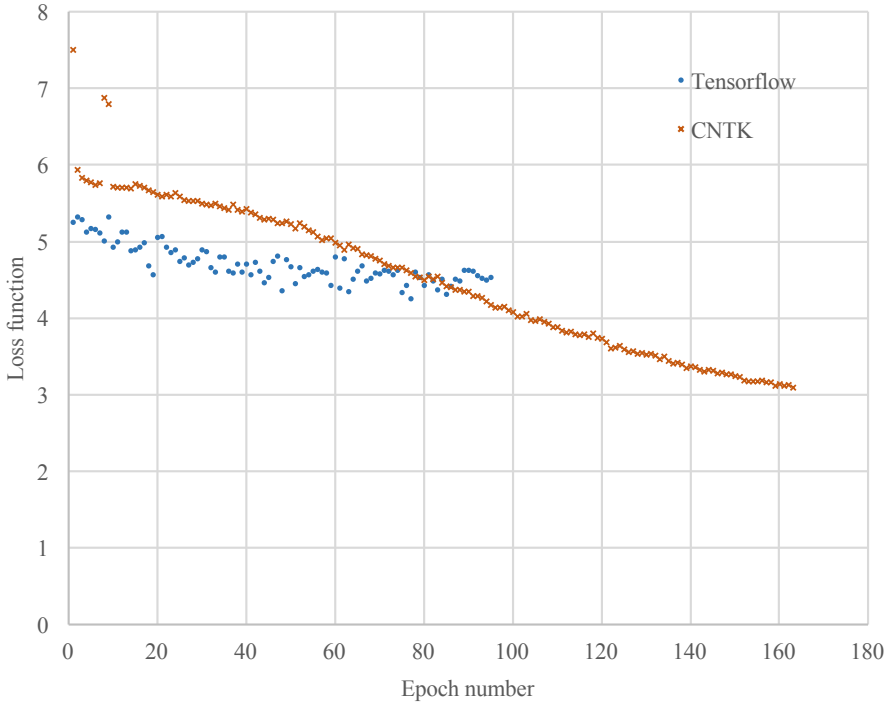
```
while IFS='' read -r line || [[ -n "$line" ]]; do
  if [ $NODES ]; then
        NODES=$NODES","
        SLAVE_NODE=$line
  else
        PS_NODE=$line #this is the ps node as well (num = 0)
  fi
  NODES=$NODES"$line:2222"
done < "$PBS_NODEFILE"

source activate tensorflow
SLAVE_CMD="python /gpfs/home/hasaadi/tensorflow/models/inception/flowers_distributed_train.py
--batch_size=32 --data_dir=/gpfs/scratch/hasaadi/flowers \
--train_dir=/gpfs/scratch/hasaadi/flowers-train --job_name='worker' \
--task_id=1 --ps_hosts=$PS_NODE:2223 --worker_hosts=$NODES 2>&1 >> slave.out"

ssh $SLAVE_NODE "module add cudnn/5.1; module add cuda80/toolkit/8.0.44; module add anaconda/3;
source activate tensorflow; $SLAVE_CMD" 2>&1 >> ssh.out &
```

CNTK

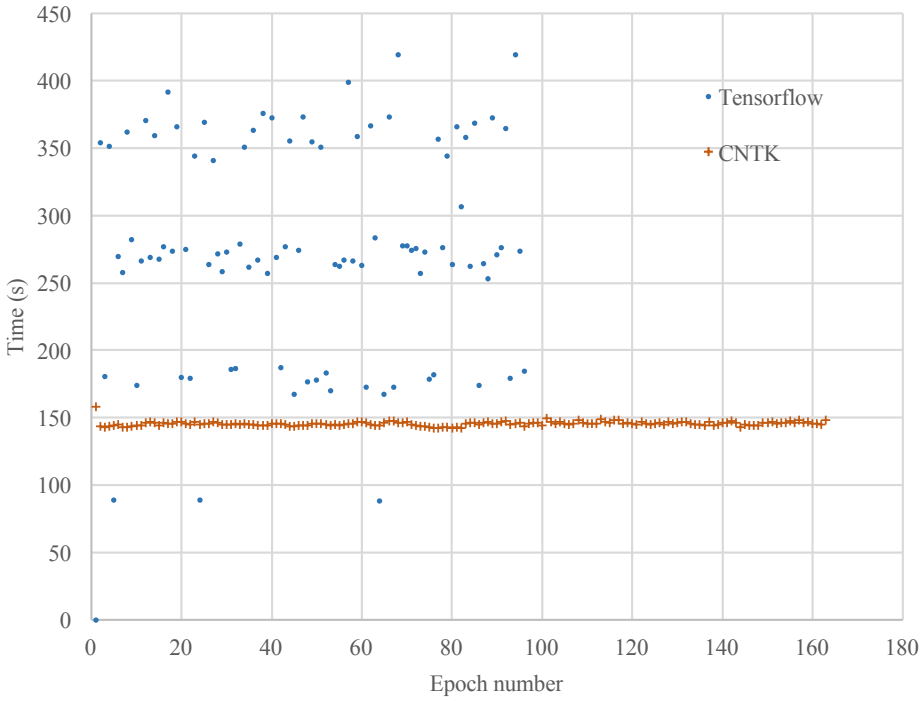
```
mpiexec --npernode 8 cntk configFile=InceptionV3.cntk
```

Stony Brook University

Flowers Dataset Results (2 Nodes)



Loss function/Epoch



Time/Epoch

Findings and conclusion

- CNTK provides broader programming language support in addition to higher convenience for non-expert users
- CNTK demonstrated the best performance in our tests
- Both tested frameworks showed poor performance when running on multiple machines
- Out of 3 frameworks, only CNTK supports launching jobs using 'mpiexec'
  - Cluster resource managers cannot used out-of-the-box to manage ML and BigData frameworks
  - Better tool support is required for BigData usecases
- All three frameworks are under very active development

Future work

- Provide binary distributions compatible with typical HPC environments
- Develop tools to facilitate launching and management of ML and BigData frameworks on HPC clusters

Longer term goal

To provide domain scientists with tools required for solving very large deep learning problems at scale.

Stony Brook University

Comparative Study of Deep Learning Frameworks in HPC Environments

HamidReza Asaadi and Barbara Chapman
Institute for Advanced Computational Science
Stony Brook University,
Stony Brook, NY

New York Scientific Data Summit 2017