Running machine learning example (MNIST) on multi-cores/nodes in Graham

Isaac Ye, HPTC @ York University

Isaac@sharcnet.ca

SHARCNET™
GENERAL INTEREST SEMINAR

Objectives

- ✓ Demonstrate how to setup and run python code using GPU on Graham
- ✓ Introduce how to write a job submission script for PyTorch framework
- ✓ Introduce several approaches in using multiple GPUs + multiple nodes
- ✓ Show how to use Tensorboard for PyTorch

compute canada | calcul canada

SHARCNET™
GENERAL INTEREST SEMINAR

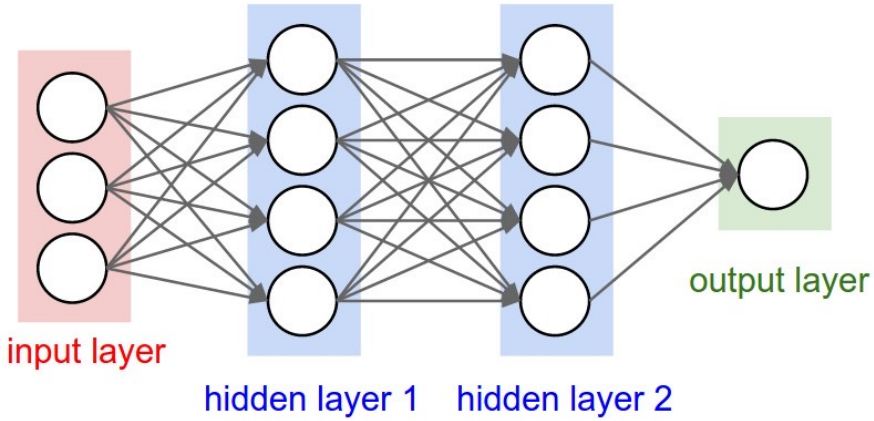
Outline

- DNN & Parallelism (Data vs Model)
- TensorFlow vs PyTorch
- GPUs and Virtual Environment
- Running interactively
- Running in SLURM (Multi-GPUs in single node)
- Running in SLURM (Multi-GPUs in multi-nodes)
- Tensorboard

compute canada | calcul canada

Deep Neural Network (DNN)

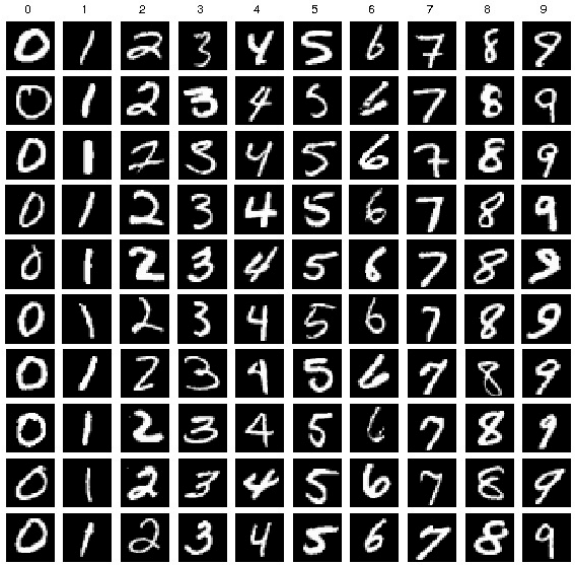
"A family of parametric, non-linear and hierarchical representation learning functions, which are massively optimized with stochastic gradient descent to encode domain knowledge, i.e. domain invariances, stationarity." -- Efstratios Gavves

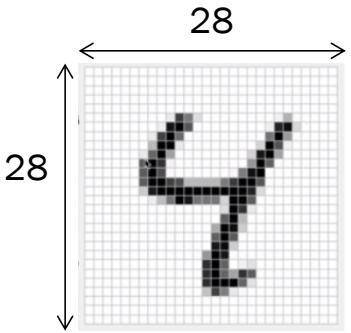


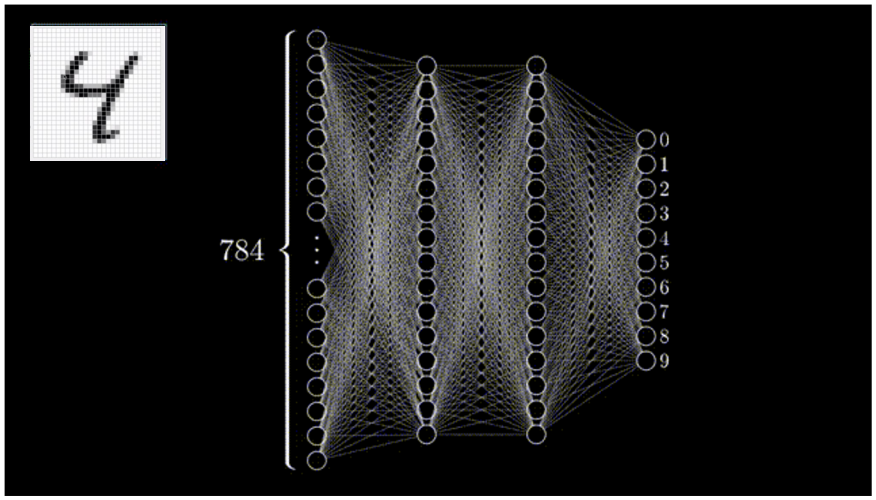
Example of a 3-layer Deep Neural Network (DNN)

http://cs231n.github.io/neural-networks-1/

Classification problem: MNIST





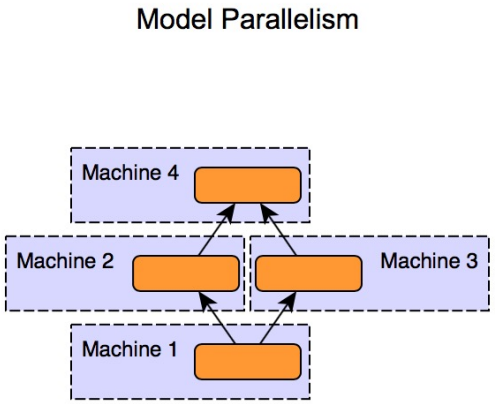


Handwritten data
60K train set and 10K test set
Each image has a size of 28x28 (=784)

Parallelism

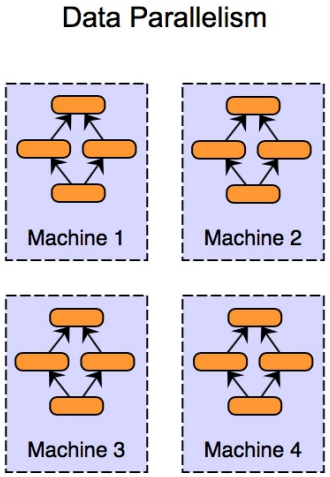
Model parallelism

Use the **same data** for every process but **split the model** among processes



Data parallelism

Use the **same model** for every process but feed it with **split data**



SHARCNET™
GENERAL INTEREST SEMINAR

Outline

- DNN & Parallelism (Data vs Model)
- TensorFlow vs PyTorch
- GPUs and Virtual Environment
- Running interactively
- Running in SLURM (Multi-GPUs in single node)
- Running in SLURM (Multi-GPUs in multi-nodes)
- Tensorboard

compute canada | calcul canada

- Rapidly growing in research community developed by Facebook
- A Python adaptation of Torch
- Caffe2 has been merged to PyTorch
- Define-by-Run type for neural networks
- Ease of expression and use
- https://github.com/pytorch/pytorch
- Version 1.10 is available in Graham

- The most widely used framework open-sourced by Google
- Runs on almost all architectures (CPU/GPU/TPU/etc)
- Define-and-Run type for neural networks
- Version 2.0+ has Define-by-Run component (Eager execution)
- https://github.com/tensorflow/tensorflow
- Version 2.4.1 is available in Graham

SHARCNET™
GENERAL INTEREST SEMINAR

Outline

- DNN & Parallelism (Data vs Model)
- TensorFlow vs PyTorch
- **GPUs and Virtual Environment**
- Running interactively
- Running in SLURM (Multi-GPUs in single node)
- Running in SLURM (Multi-GPUs in multi-nodes)
- Tensorboard

compute canada | calcul canada

SHARCNET™
GENERAL INTEREST SEMINAR

GPU resources in Compute Canada

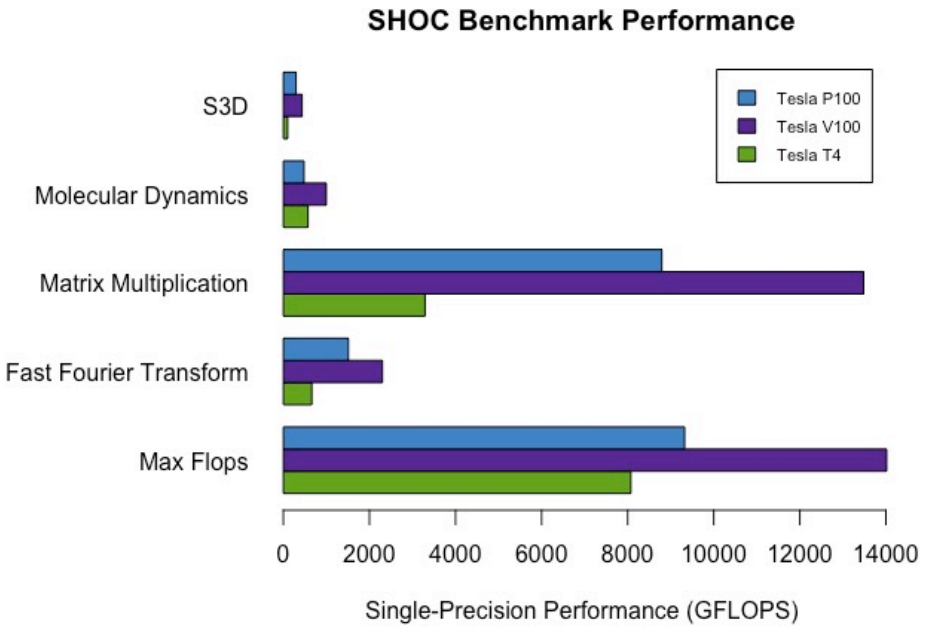
	# of nodes	GPU type	Note
Graham	160	P100 Pascal	--gres=gpu:1
	7	V100 Volta	CPU/GPU ≤ 3.5 --gres=gpu:v100:1
	36	T4 Turing (DL target)	CPU/GPU ≤ 3.5 --gres=gpu:t4:2
Cedar	146	P100 Pascal	--gres=gpu:1
Beluga	172	V100 Volta	CPU/GPU ≤ 3.5 --gres=gpu:v100:1
Niagara	None		
Narval	158	A100 (40G)	In testing

compute canada | calcul canada

Which GPUs?

Available GPUs in Graham

	P100	V100	T4
Availability	Best	Good	Better
Double Pre.	5.3 TF	7.8 TF	N/A
Single Pre.	10.6 TF	15.7 TF	8.1 TF
Tensor core	N/A	620	320



https://www.microway.com/hpc-tech-tips/nvidia-turing-tesla-t4-hpc-performance-benchmarks/tesla_comparison_t4-p100-v100/

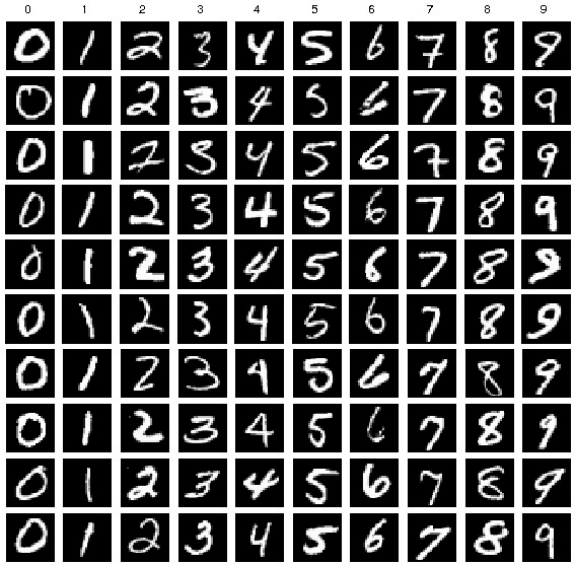
SHARCNET™
GENERAL INTEREST SEMINAR

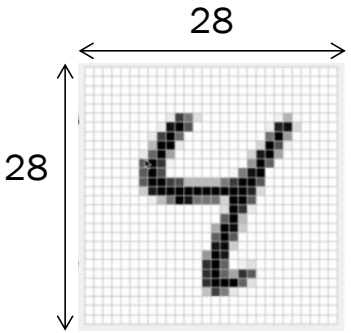
Setting up for PyTorch

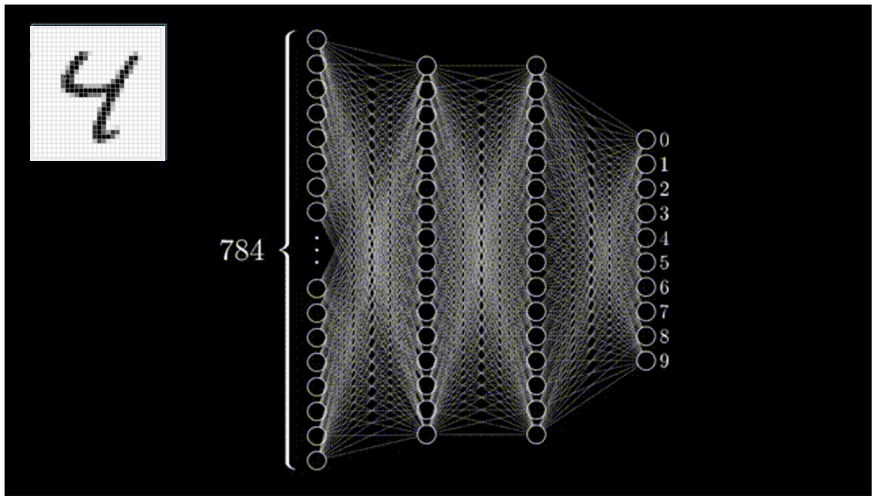
1. Modules
2. Virtual environment
3. Available wheels
4. Installing PyTorch

compute canada | calcul canada

Classification problem: MNIST







Handwritten data
60K train set and 10K test set
Each image has a size of 28x28 (=784)

A little peek in the code

PyTorch

```
import torch
from torchvision import datasets, transforms
import torch.nn as nn
import torch.optim as optim
from sklearn.metrics import accuracy_score
import time
import matplotlib.pyplot as plt

# ====== GPU selection ======= #
if torch.cuda.is_available():
    print('GPU is running')
else:
    print('CPU is running')

device = 'cuda:0' if torch.cuda.is_available() else 'cpu'
model.to(device)
```


SHARCNET™
GENERAL INTEREST SEMINAR

Outline

- DNN & Parallelism (Data vs Model)
- TensorFlow vs PyTorch
- GPUs and Virtual Environment
- Running interactively
- Running in SLURM (Multi-GPUs in single node)
- Running in SLURM (Multi-GPUs in multi-nodes)
- Tensorboard

compute canada | calcul canada

Running interactively

```
[isaac@gra-login2 MNIST]$ salloc --time=00:10:00 --ntasks=1 --cpus-per-task=3 --mem=8000M
 --gres=gpu:t4:2 --account=def-isaac
```

```
GPU is running
Number of 159010 parameters
Epoch: 0, Train Loss: 0.9922358669588328, Val Loss: 0.6142751978168005, Test Acc: 90.05%, 9.1
Epoch: 1, Train Loss: 0.46111484544585124, Val Loss: 0.4694672076007988, Test Acc: 90.82000000000001%, 9.0
Epoch: 2, Train Loss: 0.36240459147774046, Val Loss: 0.4909582217282887, Test Acc: 90.62%, 9.0
Epoch: 3, Train Loss: 0.3311268923818455, Val Loss: 0.43367936377283894, Test Acc: 90.05%, 9.1
```


SHARCNET™
GENERAL INTEREST SEMINAR

Outline

- DNN & Parallelism (Data vs Model)
- TensorFlow vs PyTorch
- GPUs and Virtual Environment
- Running interactively
- **Running in SLURM (Multi-GPUs in single node)**
- Running in SLURM (Multi-GPUs in multi-nodes)
- Tensorboard

compute canada | calcul canada

SHARCNET™
GENERAL INTEREST SEMINAR

Running in scheduler (SLURM)

Single GPU in Single Node

```
#!/bin/bash
#
#SBATCH --gres=gpu:t4:1
#SBATCH --cpus-per-task=3
#SBATCH --mem=8000M
#SBATCH --time=00:30:00
#SBATCH --account=def-isaac
#SBATCH --output=slurm.%x.%j.out

module load python
module load scipy-stack

source ~/PT/bin/activate
cd /home/$USER/mnist
python /home/$USER/mnist/mnist.py
```

PyTorch

Multi-GPUs in Single Node

```
#!/bin/bash
#
#SBATCH --nodes=1
#SBATCH --tasks-per-node=2
#SBATCH --gres=gpu:t4:2
#SBATCH --cpus-per-task=3
#SBATCH --mem=20G
#SBATCH --time=00:30:00
#SBATCH --account=def-isaac
#SBATCH --output=slurm.%x.%j.out

module load python
module load scipy-stack

source ~/PT/bin/activate
cd /home/$USER/mnist
srun python /home/$USER/mnist/mnist.py
```

Note: **CPU to GPU** ratio recommended to have **less than 3.5**

compute canada | calcul canada

SHARCNET™
GENERAL INTEREST SEMINAR

Outline

- DNN & Parallelism (Data vs Model)
- TensorFlow vs PyTorch
- GPUs and Virtual Environment
- Running interactively
- Running in SLURM (Multi-GPUs in single node)
- **Running in SLURM (Multi-GPUs in multi-nodes)**
- Tensorboard

compute canada | calcul canada

SHARCNET™
GENERAL INTEREST SEMINAR

PyTorch + DDP

Multiple GPUs in Multi-nodes

Distributed Data Parallel (DDP)

```
ngpus_per_node = torch.cuda.device_count()

print(ngpus_per_node)

rank = int(os.environ.get("SLURM_NODEID"))*ngpus_per_node \
        + int(os.environ.get("SLURM_LOCALID"))

print('From Rank: {}, ==> Initializing Process Group...'.format(rank))

dist.init_process_group(backend=args.dist_backend, init_method=args.init_method, \
        world_size=args.world_size, rank=rank)
print("process group ready!")

print('From Rank: {}, ==> Making model..'.format(rank))


     class Net(nn.Module):

        def __init__(self):
           super(Net, self).__init__()
```

PyTorch

compute canada | calcul canada

PyTorch + DDP

Multi-GPU in Multi-Node

- Distributed Data Parallel (DDP)

```
#!/bin/bash
#
#SBATCH --nodes=2
#SBATCH --gres=gpu:t4:2
#SBATCH --tasks-per-node=2
#SBATCH --cpus-per-task=3
#SBATCH --mem=10G
#SBATCH --time=00:30:00
#SBATCH --account=def-isaac
#SBATCH --output=slurm.%x.%j.out

module load python
module load scipy-stack

source ~/PT/bin/activate

export MASTER_ADDR=$(hostname)
echo "r$SLURM_NODEID master: $MASTER_ADDR"
echo "r$SLURM_NODEID Launching python script"

srun python /home/$USER/mnist/mnist_ddp.py --init_method
tcp://$MASTER_ADDR:3456 --world_size $SLURM_NTASKS
```

```
r0 master: gra1181
r0 Launching python script

Starting...
From Rank: 0, ==> Initializing Process Group...
process group ready!
From Rank: 0, ==> Making model..
From Rank: 0, ==> Preparing data..
From Rank: 0, Training time 0:00:00.615626

Starting...
From Rank: 2, ==> Initializing Process Group...
process group ready!
From Rank: 2, ==> Making model..
From Rank: 2, ==> Preparing data..
From Rank: 2, Training time 0:00:03.529344

Starting...
From Rank: 3, ==> Initializing Process Group...
process group ready!
From Rank: 3, ==> Making model..
From Rank: 3, ==> Preparing data..

Starting...
From Rank: 1, ==> Initializing Process Group...
process group ready!
From Rank: 1, ==> Making model..
From Rank: 1, ==> Preparing data..
From Rank: 1, Training time 0:00:00.246557
```


SHARCNET™
GENERAL INTEREST SEMINAR

PyTorch + PyTorch Lighting

Multiple GPUs in Multi-nodes

PyTorch Lightning

```
import pytorch_lightning as pl
```

```
class Net(pl.LightningModule):

    def __init__(self):
        super(Net, self).__init__()

        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)
```

PyTorch

compute canada | calcul canada

PyTorch + PyTorch Lightning

Multi-GPU in Multi-Node

- PyTorch Lightning

```
#!/bin/bash
#
#SBATCH --nodes=2
#SBATCH --gres=gpu:t4:2
#SBATCH --tasks-per-node=2
#SBATCH --cpus-per-task=3
#SBATCH --mem=10G
#SBATCH --time=00:30:00
#SBATCH --account=def-isaac
#SBATCH --output=slurm.%x.%j.out

module load python
module load scipy-stack
module load cuda cudnn

source ~/PT/bin/activate
source ~/.bashrc

srun python /home/$USER/mnist/mnist_lightning.py
```

```
GPU available: True, used: True
TPU available: None, using: 0 TPU cores
Multi-processing is handled by Slurm.
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0,1]
GPU available: True, used: True
TPU available: None, using: 0 TPU cores
Multi-processing is handled by Slurm.
LOCAL_RANK: 1 - CUDA_VISIBLE_DEVICES: [0,1]
GPU available: True, used: True
TPU available: None, using: 0 TPU cores
Multi-processing is handled by Slurm.
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0,1]
GPU available: True, used: True
TPU available: None, using: 0 TPU cores
Multi-processing is handled by Slurm.
LOCAL_RANK: 1 - CUDA_VISIBLE_DEVICES: [0,1]
initializing ddp: GLOBAL_RANK: 0, MEMBER: 1/4
initializing ddp: GLOBAL_RANK: 1, MEMBER: 2/4
initializing ddp: GLOBAL_RANK: 2, MEMBER: 3/4
initializing ddp: GLOBAL_RANK: 3, MEMBER: 4/4
```


PyTorch + HOROVOD

SHARCNET™
GENERAL INTEREST SEMINAR

Multiple GPUs in Multi-nodes

Note: You need to install horovod in your virtual environment
'pip install –no-index horovod'

Horovod

```
import argparse
import torch.multiprocessing as mp
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms
import torch.utils.data.distributed
import horovod.torch as hvd

if __name__ == '__main__':
    args = parser.parse_args()
    args.cuda = not args.no_cuda and torch.cuda.is_available()

    # Horovod: initialize library.
    hvd.init()
    torch.manual_seed(args.seed)

    if args.cuda:
        # Horovod: pin GPU to local rank.
        torch.cuda.set_device(hvd.local_rank())
        torch.cuda.manual_seed(args.seed)
```

PyTorch

compute canada | calcul canada

HOROVOD

Distributed deep learning training framework

SHARCNET™
GENERAL INTEREST SEMINAR

Installation

```
[isaac@gra-login2 MNIST_tf]$ source ~/TF/bin/activate
(TF) [isaac@gra-login2 MNIST_tf]$ avail_wheels horo*
name     version    build    python    arch
-------  ---------  -------  --------  -------
horovod  0.20.3              cp38      generic
(TF) [isaac@gra-login2 MNIST_tf]$ pip install --no-index horovod
```

Environment

```
[isaac@gra-login2 ~]$ cat .bashrc
export HOROVOD_CUDA_HOME=$CUDA_HOME
export HOROVOD_NCCL_HOME=$EBROOTNCCL
export HOROVOD_GPU_BROADCAST=NCCL
export HOROVOD_GPU_ALLREDUCE=NCCL
export HOROVOD_GPU_OPERATIONS=NCCL
export HOROVOD_WITH_PYTORCH=1
export HOROVOD_WITH_TENSORFLOW=1
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$EBROOTNCCL
export PATH=$PATH:$EBROOTNCCL/include:$EBROOTNCCL/lib
```

PyTorch

compute canada | calcul canada

PyTorch + HOROVOD

Multi-GPU in Multi-Node

- PyTorch Lightning

```
#!/bin/bash
#
#SBATCH --nodes=2
#SBATCH --gres=gpu:t4:2
#SBATCH --tasks-per-node=2
#SBATCH --cpus-per-task=3
#SBATCH --mem=10G
#SBATCH --time=00:30:00
#SBATCH --account=def-isaac
#SBATCH --output=slurm.%x.%j.out

module load python
module load scipy-stack
module load cuda cudnn
module load nccl

source ~/.bashrc
source ~/PT/bin/activate
cd /home/$USER/mnist
srun python /home/$USER/mnist/mnist_horovod.py
```

```
GPU is running
hostname = gra1154
how many gpus in gra1154: 2
which gpu is running:  Tesla T4

Test set: Average loss: 0.2145, Accuracy: 93.66%

GPU is running
hostname = gra1154
how many gpus in gra1154: 2
which gpu is running:  Tesla T4

Test set: Average loss: 0.2145, Accuracy: 93.66%

GPU is running
hostname = gra1155
how many gpus in gra1155: 2
  tensor = torch.tensor(val)

Test set: Average loss: 0.2145, Accuracy: 93.66%

GPU is running
hostname = gra1155
how many gpus in gra1155: 2
which gpu is running:  Tesla T4

Test set: Average loss: 0.2145, Accuracy: 93.66%
```


SHARCNET™
GENERAL INTEREST SEMINAR

Outline

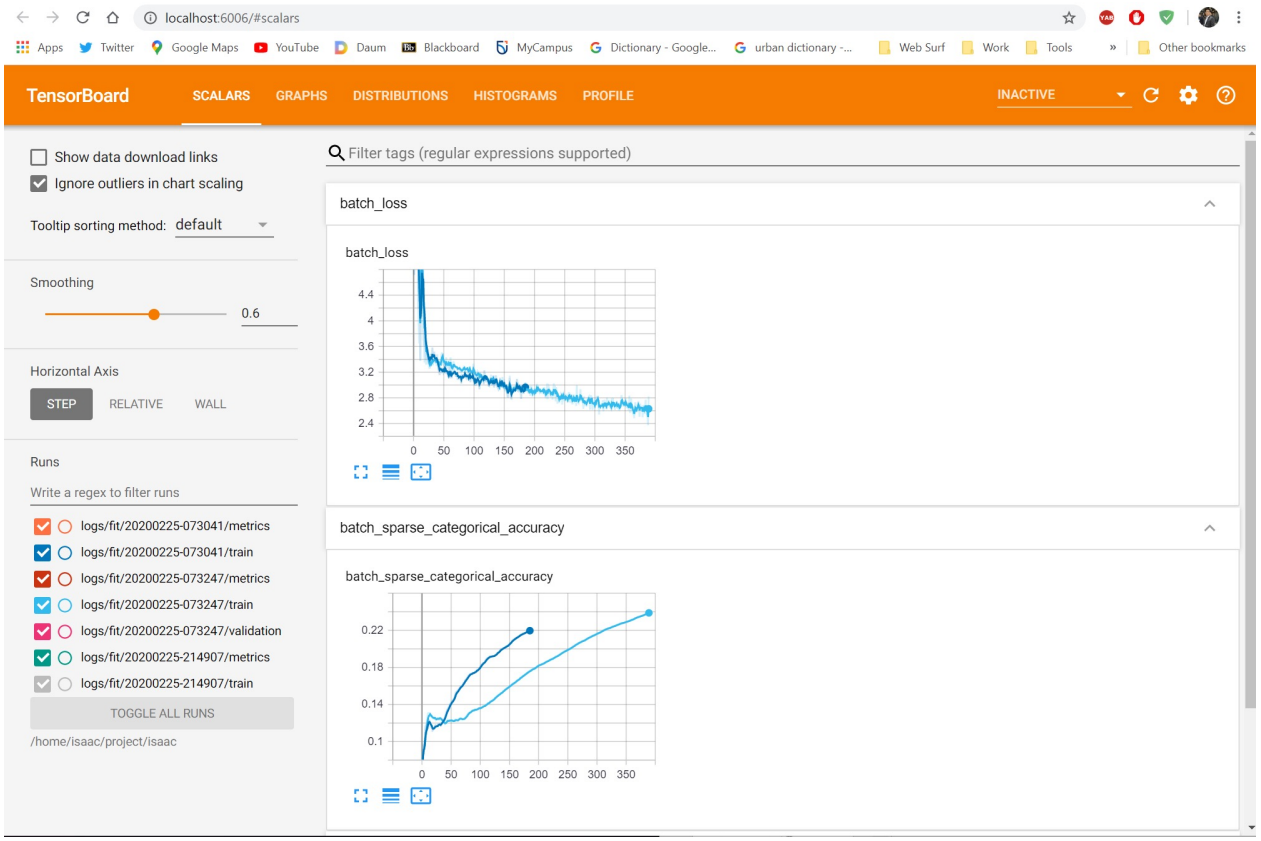
- DNN & Parallelism (Data vs Model)
- TensorFlow vs PyTorch
- GPUs and Virtual Environment
- Running interactively
- Running in SLURM (Multi-GPUs in single node)
- Running in SLURM (Multi-GPUs in multi-nodes)
- Tensorboard

compute canada | calcul canada

SHARCNET™

GENERAL INTEREST SEMINAR

Tensorboard + PyTorch



compute canada | calcul canada

Test code

```
import torch
from torch.utils.tensorboard import SummaryWriter
writer = SummaryWriter()

x = torch.arange(-5, 5, 0.1).view(-1, 1)
y = -5 * x + 0.1 * torch.randn(x.size())

model = torch.nn.Linear(1, 1)
criterion = torch.nn.MSELoss()
optimizer = torch.optim.SGD(model.parameters(), lr = 0.1)

def train_model(iter):
    for epoch in range(iter):
        y1 = model(x)
        loss = criterion(y1, y)
        writer.add_scalar("Loss/train", loss, epoch)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

train_model(100)
writer.flush()

writer.close()
```


Thanks!

Q & A

lsaac@sharcnet.ca