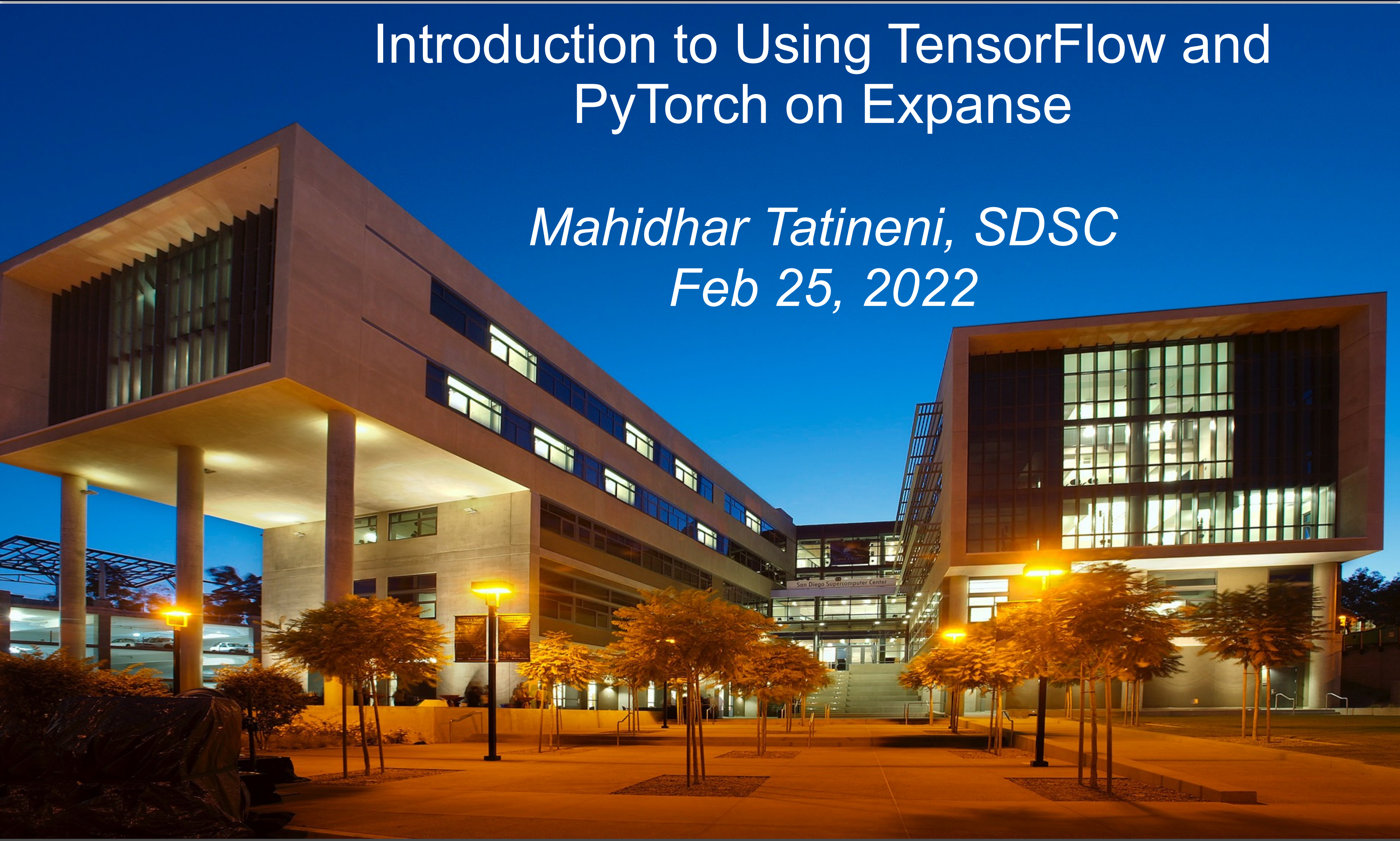
Introduction to Using TensorFlow and PyTorch on Expanse

Mahidhar Tatineni, SDSC

Feb 25, 2022



SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

Implementing Deep Learning (DL) techniques in Practice

- **Frameworks/Libraries like TensorFlow, PyTorch, and Horovod make it easy to program DL tasks**
  - High level APIs and interfaces from python, C++ etc.
  - Implementations of commonly used neural-network building blocks.
  - Scalable, distributed options

- **This is a quick overview on how to run on Expanse. Lot of training material online if you are interested in details of these frameworks**

TensorFlow

- **Open-source machine learning library originally developed by Google Brain team.**
- **High level Keras API in python**
  - Modular building blocks to create and train DL models.
  - Allows assembly of layers, lot of common use cases supported out of the box
  - Allows for custom blocks, can create new layers, loss functions etc.
- **Eager execution (enabled by default v2.0+)**
- **Pre-made Estimators for training, evaluation, prediction, and export.**

Links: https://www.tensorflow.org/guide/

https://www.tensorflow.org/tutorials/

PyTorch

- **Deep learning platform**
  - Hybrid front-end w/ ease of use/flexibility in eager mode, graph mode for speed, optimization
  - Deep integration with Python
  - C++ front end
- **Pre-trained model repository, can be customized**
- **Ecosystem of tools:** https://pytorch.org/ecosystem
- **Quick intro here:**
  - https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

TensorFlow and PyTorch on Expanse

- **Two main approaches:**
  - **Singularity container images** with all the python packages included, along with GPU drivers, CUDA libraries. Examples:
    - /cm/shared/apps/containers/singularity/tensorflow/tensorflow-2.3.0-gpu-20200929.simg
    - /cm/shared/apps/containers/singularity/pytorch/pytorch-1.5.0-gpu-20200511.simg
    - Today's examples will use the Singularity approach.

  - **Conda/Miniconda based installs** (typically users have their own custom versions).

Machine Learning/Deep Learning on Expanse via Singularity

- Bulk of the Singularity usage on SDSC machines (Expanse, Comet) is for machine learning/deep learning applications.

- Lot of these packages are constantly upgraded, and the dependency list is difficult to update in the standard environment.

- Install options
  - Singularity image provides dependencies and user can compile actual application from source.
  - Entire dependency stack and the application is in the image. e.g TensorFlow and some additional python libraries

- Run options
  - Most cases are run on single GPU nodes (4 GPUs at most)
  - Can access this via Jupyter notebooks
  - Multi-node options are possible but difficult to set up via singularity.

Example TensorFlow run script

- See /cm/shared/examples/sdsc/tensorflow

```
[mahidhar@login01 tensorflow]$ more run-tensorflow-gpu-shared.sh
#!/usr/bin/env bash

#SBATCH --job-name=tensorflow-gpu-shared
#SBATCH --account=use300
#SBATCH --partition=gpu-shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=10
#SBATCH --cpus-per-task=1
#SBATCH --mem=93G
#SBATCH --gpus=1
#SBATCH --time=00:30:00
#SBATCH --output=tensorflow-gpu-shared.o%j.%N

declare -xr SINGUALRITY_MODULE='singularitypro/3.5'

module purge
module load "${SINGUALRITY_MODULE}"
module list
printenv

time -p singularity exec --bind /expanse,/scratch --nv /cm/shared/apps/containers/singularity/tensorflow/tensorflow-2.3.
0-gpu-20200929.simg python3 cnn-cifar.py
```

PyTorch on Expanse

- Enable via Singularity
- Examples: /cm/shared/examples/sdsc/pytorch

```
#!/usr/bin/env bash
#SBATCH --job-name=pytorch-gpu-shared
#### Change account below
#SBATCH --account=use300
#SBATCH --partition=gpu-shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=10
#SBATCH --cpus-per-task=1
#SBATCH --mem=93G
#SBATCH --gpus=1
#SBATCH --time=00:30:00
#SBATCH --output=pytorch-gpu-shared.o%j.%N

declare -xr SINGUALRITY_MODULE='singularitypro/3.5'

module purge
module load "${SINGUALRITY_MODULE}"
module list
printenv

time -p singularity exec --bind /expanse,/scratch --nv /cm/shared/apps/containers/singularity/pytorch/pytorch-1.5.0-gpu
-20200511.simg  python3 /opt/pytorch-1.5.0/examples/mnist/main.py
```

MNIST Example via Jupyter Notebook

- Step 1: Copy files:

cp -r /cm/shared/examples/sdsc/tensorflow/jupyter $HOME/tensorflow

- Step 2: Launch notebook with instructions on next page

Accessing via Jupyter Notebook

- CPU node:

/cm/shared/apps/sdsc/galyleo/galyleo.sh launch -A use300 -p shared -n 16 -M 32 -t 00:30:00 -e singularitypro/3.5 -s /cm/shared/apps/containers/singularity/tensorflow/tensorflow-2.3.0-gpu-20200929.simg -d /home/$USER/tensorflow

- GPU node:

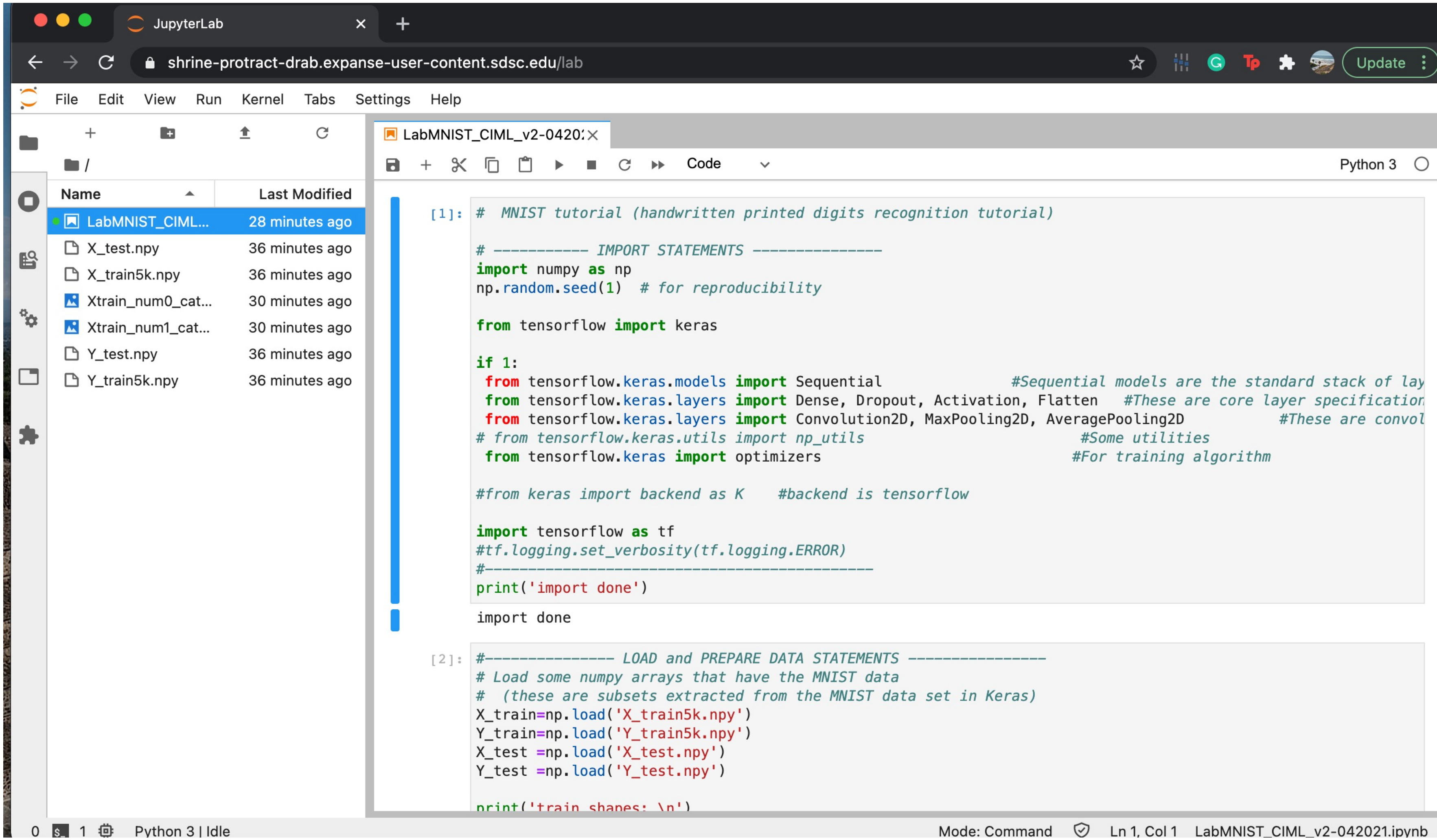
/cm/shared/apps/sdsc/galyleo/galyleo.sh launch -A use300 -p gpu-shared -n 10 -M 93 -G 1 -t 00:30:00 -e singularitypro/3.5 -s /cm/shared/apps/containers/singularity/tensorflow/tensorflow-2.3.0-gpu-20200929.simg -d /home/$USER/tensorflow

(Change use300 to your allocation)

Sample Output From Launch Command

```
[mahidhar@login01 tensorflow]$ /cm/shared/apps/sdsc/galyleo/galyleo.sh launch -A use300 -p shared -n 16 -M 32 -t 00:30:00 -e singularitypro/3.5 -s
/cm/shared/apps/containers/singularity/tensorflow/tensorflow-2.3.0-gpu-20200929.simg -d /home/mahidhar/tensorflow
Preparing galyleo for launch into Jupyter orbit ...
Listing all launch parameters ...
  command-line option      : value
       | --mode            : local
    -A | --account         : use300
    -R | --reservation     :
    -p | --partition       : shared
    -q | --qos             :
    -N | --nodes           : 1
    -n | --ntasks-per-node : 16
    -c | --cpus-per-task   : 1
    -M | --memory-per-node : 32 GB
    -m | --memory-per-cpu  : 2 GB
    -G | --gpus            :
       | --gres            :
    -t | --time-limit      : 00:30:00
    -j | --jupyter         : lab
    -d | --notebook-dir    : /home/mahidhar/tensorflow
    -r | --reverse-proxy   : expanse-user-content.sdsc.edu
    -D | --dns-domain      : eth.cluster
    -s | --sif             : /cm/shared/apps/containers/singularity/tensorflow/tensorflow-2.3.0-gpu-20200929.simg
    -B | --bind            :
       | --nv              :
    -e | --env-modules     : singularitypro/3.5
       | --conda-env       :
    -Q | --quiet           : 1
Generating Jupyter launch script ...
Submitted Jupyter launch script to Slurm. Your SLURM_JOB_ID is 2045884.
Success! Token linked to jobid.
Please copy and paste the HTTPS URL provided below into your web browser.
Do not share this URL with others. It is the password to your Jupyter notebook session.
Your Jupyter notebook session will begin once compute resources are allocated to your Slurm job by the scheduler.
https://runny-yearbook-presoak.expanse-user-content.sdsc.edu?token=3486e40f89adb3bc833336e093161093
```

Browser snapshot of notebook



Horovod

- Distributed training framework for TensorFlow, PyTorch, and MXNet.

- Distributed TensorFlow with parameter servers can be tricky to setup and involve code changes.

- Horovod leverages MPI to make things easier and also picks up performance by using MPI functions which are optimized for the hardware involved.

Typical Horovod code additions

- Initialization (hvd.init())
- Assign GPUs
- Set number of workers (scale up)
- Change optimizer to use Horovod (hvd.DistributedOptimizer). This will allow leverage of MPI functions like AllReduce, AllGather for the gradient computations.
- Broadcast global variables
- Checkpoints written from worker 0.
- MNIST example here:
  - https://github.com/horovod/horovod/blob/master/examples/tensorflow2/tensorflow2_keras_mnist.py

Summary

- Several frameworks available for developing, training, and testing machine learning and deep learning models. TensorFlow, PyTorch commonly used on Expanse.

- Both TensorFlow and PyTorch are available via Singularity on Expanse.

- High level APIs available for use via python. Easy to incorporate into Jupyter notebooks.