DS-GA 1006 Capstone

Lab Session 4

Instructors: Julia Kempe
Mark Ho
Najoung Kim
Wenda Zhou

TA: Wenxin Zhang (wz2164@nyu.edu)

Lab Session 4

- HW2 - CV-based problem
  - Overview
  - Setup
    - Keep your burst instance alive using Slurm
    - Create base overlay & Create package overlay
    - Optimize data access
    - Start singularity instance & Connect to instance from VSCode
    - Use Notebooks from VSCode
  - Train the provided network
    - 4 epochs | mixed precision training | 2 GPUs
    - Nvidia-smi
    - Tensorboard
  - Testing
    - Smoke test

Overview

- Dataset: Places365
  - A scene recognition dataset
  - Consist of 10 million images comprising 434 scene classes
- Dataset.py:
  - Load the training dataset and the validation dataset
  - The randomized transformations apply the same transformation to all the images of a given batch, but produce different transformations across calls
- Model.py:
  - Build config class: TrainingConfig (ModelConfig, OptimConfig, DataCofing, precision, batch_size, max_epoches, gpus)
  - Choose the CNN model (mobilenet_v3_large), criterion, and metric
  - Forward, Compute Loss, Configure the Optimizers
- Train.py:
  - Train the provided network
  - 4 Epochs & Mixed Precision & 2 gpus

2.1 Setup

Following the instructions in `homework/cv/README.md`, create the overlays containing the required packages for running the homework. Start a singularity instance with the container, and connect VSCode to your instance. Open the `homework/cv/data.ipynb` notebook and run the existing code to display an image from the dataset. How many images are there in total in the training split of the dataset? Write code in a new cell to compute this quantity. Take a screenshot of the notebook with the image example and the total number of images displayed.

Setup

Request GCP instance using Slurm

Slurm

- A cluster management and job scheduling system for Linux clusters, through which we interact with the Greene clusters and the GCP
  - Allocates access to compute nodes to users
  - Provides framework for starting, executing, and monitoring work (parallel job)
  - Arbitrates contention for resources by managing a queue of pending work
- Commands
  - srun: run jobs interactively
  - sbatch *.sh: queue jobs using a bash scripts
  - squeue -u $USER: reports the state of jobs
  - scancel $jobid: cancel pending or running jobs

```
#!/bin/bash
#
#SBATCH --job-name=request_burst_instance
#SBATCH --account=ds_ga_1006_001-2022fa
#SBATCH --partition=n1c16m96-v100-2
#SBATCH --gres=gpu:v100:2
#SBATCH --time=8:00:00


sleep 8h
```

- Account: ds_ga_1006_001-2022fa
- partition: n1c16m96-v100-2
  - Machine: 16 CPU, 96 GB memory, 2 V100 GPU
- GPUs: --gres=gpu:v100:2
  - --gres=gpu:type:count
- Time: 8:00:00
  - Run time of the machine: 8 hours
  - Maximum: 24 hours
- Remember to cancel the running jobs after work

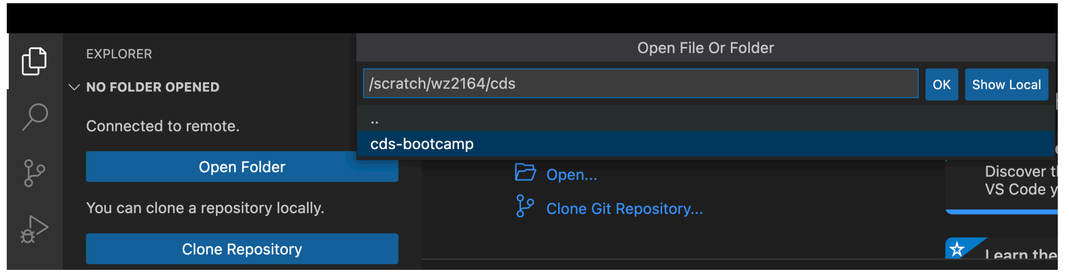
```
(base) [wz2164@log-burst ~]$ sbatch sleep.sh
Submitted batch job 89887
(base) [wz2164@log-burst ~]$ squeue -u wz2164
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             89887 n1c16m96- request_   wz2164 CF       0:03      1 b-17-1
```

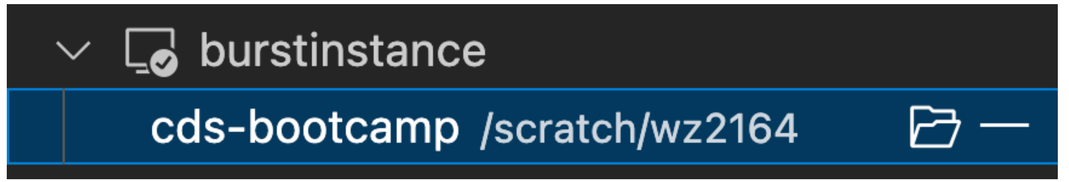
Setup

Create the Overlays

- scripts/
  - create_base_overlay.sh: copy the pytorch environment provided by Nvidia that we can modify
  - create_package_overlay.sh: install additional packages into the environment
- Create the overlays required to work on the homework 2
  - cd /scratch/$USERID
  - git clone https://github.com/wendazhou/cds-bootcamp.git
  - cd cds-bootcamp/homework/cv
  - ./scripts/create_base_overlay.sh
  - ./scripts/create_package_overlay.sh

```
bash-4.4$ git clone https://github.com/wendazhou/cds-bootcamp.git
Cloning into 'cds-bootcamp'...
remote: Enumerating objects: 348, done.
remote: Counting objects: 100% (104/104), done.
remote: Compressing objects: 100% (85/85), done.
remote: Total 348 (delta 37), reused 59 (delta 14), pack-reused 244
Receiving objects: 100% (348/348), 83.50 KiB | 1.21 MiB/s, done.
Resolving deltas: 100% (136/136), done.
```






```
bash-4.4$ ls
bootcamp  data.ipynb  README.md  scripts  setup.py  start_singularity_instance.sh  tests
bash-4.4$ ./scripts/create_base_overlay.sh
Extracting base package overlay
Cloning base packages into overlay
Source:      /opt/conda
Destination: /ext3/conda/bootcamp
The following packages cannot be cloned out of the root environment:
 - conda-forge/linux-64::conda-4.13.0-py38h578d9bd_1
 - conda-forge/linux-64::conda-build-3.21.9-py38h578d9bd_1
Packages: 138
Files: 44425
```

```
bash-4.4$ ls
bootcamp  data.ipynb  overlay-base.ext3  README.md  scripts  setup.py  start_singularity_instance.sh  tests
bash-4.4$ ./scripts/create_package_overlay.sh
Extracting additional package overlay
Installing additional packages
```

Setup

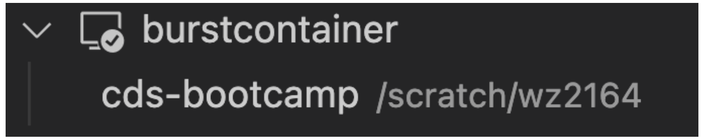
Start singularity instance

- Optimize data access
  - Places365.squashfs is a large dataset of images, comprising of a large number of small files
  - Such accesses are difficult to handle efficiently for networked file systems. It will be helpful to first move the data to a local directory
  - On Greene only, move the data to the $TMPDIR location: rsync --info=progress2 /scratch/wz2247/data/places365.squashfs $TMPDIR
  - On GCP, move the data to your RAM (/dev/shm): rsync --info=progress2 /scratch/wz2247/data/places365.squashfs /dev/shm/
  - The local data will be deleted after your job/instance ends
- Set the DATA_DIRECTORY environment variable before starting the singularity instance:
  - cd /scratch/wz2164/cds-bootcamp/homework/cv
  - DATA_DIRECTORY=/dev/shm ./start_singularity_instance.sh

```
singularity instance start --containall --no-home -B $HOME/.ssh -B /scratch -B $PWD --nv \
    --overlay overlay-temp.ext3 \
    --overlay overlay-base.ext3:ro \
    --overlay overlay-packages.ext3:ro \
    --overlay $DATA_DIRECTORY/places365.squashfs:ro \
    $IMAGE ${INSTANCE_NAME}
```

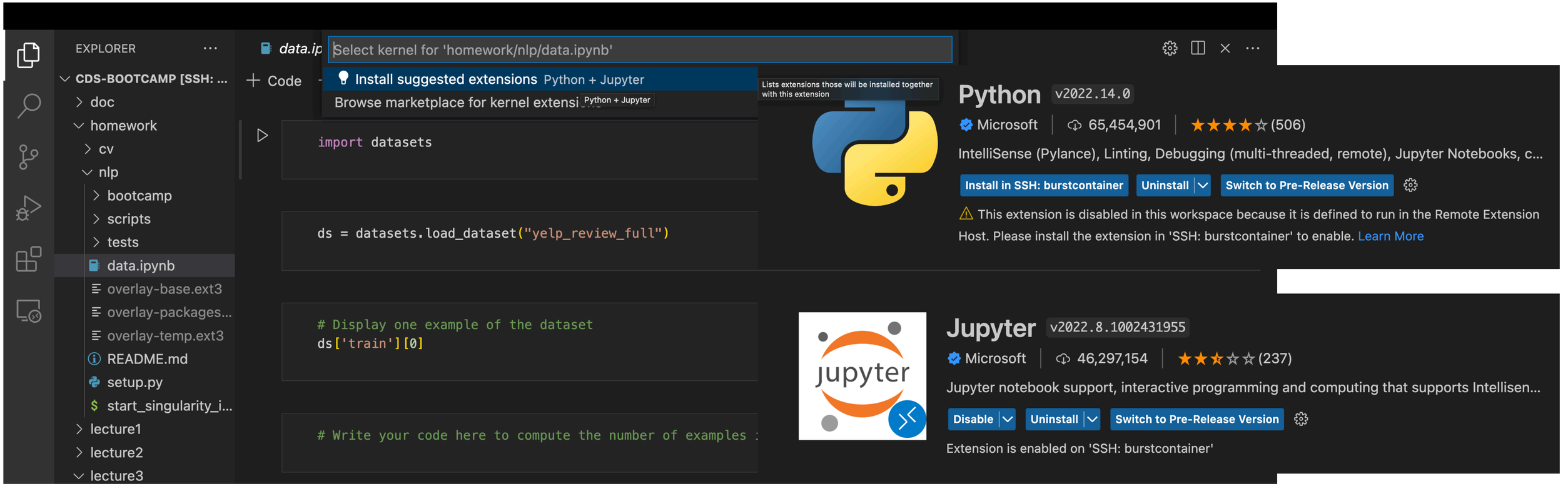
```
TERMINAL
bash-4.4$ ls -lah /dev/shm/
total 0
drwxrwxrwt.  2 root root   40 Oct  1 23:03 .
drwxr-xr-x. 19 root root 3.0K Oct  1 23:03 ..
bash-4.4$ rsync --info=progress2 /scratch/wz2247/data/places365.squashfs /dev/shm/
  24,068,497,408 100%  104.80MB/s    0:03:39 (xfr#1, to-chk=0/1)
bash-4.4$ ls -lah /dev/shm/
total 23G
drwxrwxrwt.  2 root   root     60 Oct  1 23:09 .
drwxr-xr-x. 19 root   root   3.0K Oct  1 23:03 ..
-rw-r--r--.  1 wz2164 wz2164  23G Oct  1 23:09 places365.squashfs
bash-4.4$ cd /scratch/wz2164/cds-bootcamp/homework/cv/
bash-4.4$ DATA_DIRECTORY=/dev/shm ./start_singularity_instance.sh
INFO:    instance started successfully
bash-4.4$
```

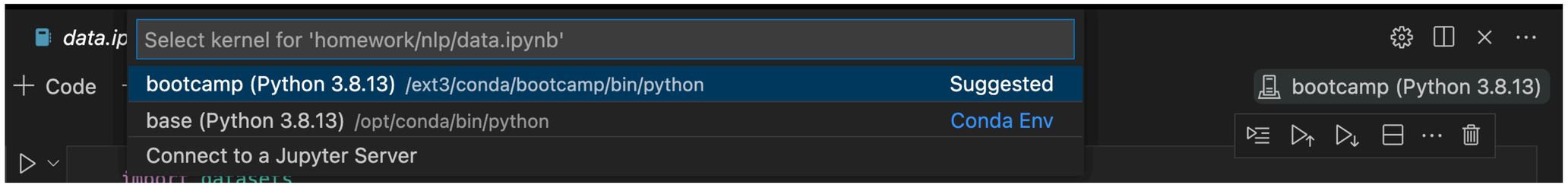
```
Host burstinstance burstcontainer
    User wz2164
    HostName b-17-1
    ForwardAgent yes
    ProxyJump greeneburst
    PasswordAuthentication yes
    ChallengeResponseAuthentication no
    StrictHostKeyChecking=No
    # UserHostsKnownFile=/dev/null


Host burstcontainer
    RemoteCommand singularity shell --containall --shell='/bin/bash' instance://mycontainer
    RequestTTY true
```



Setup

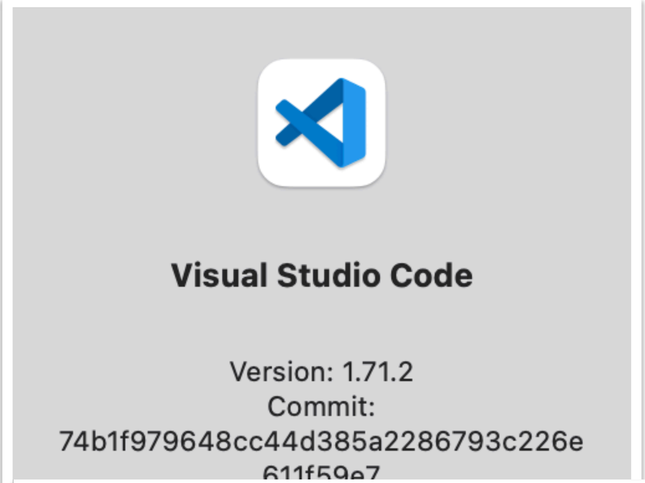
Using Notebook on VSCode

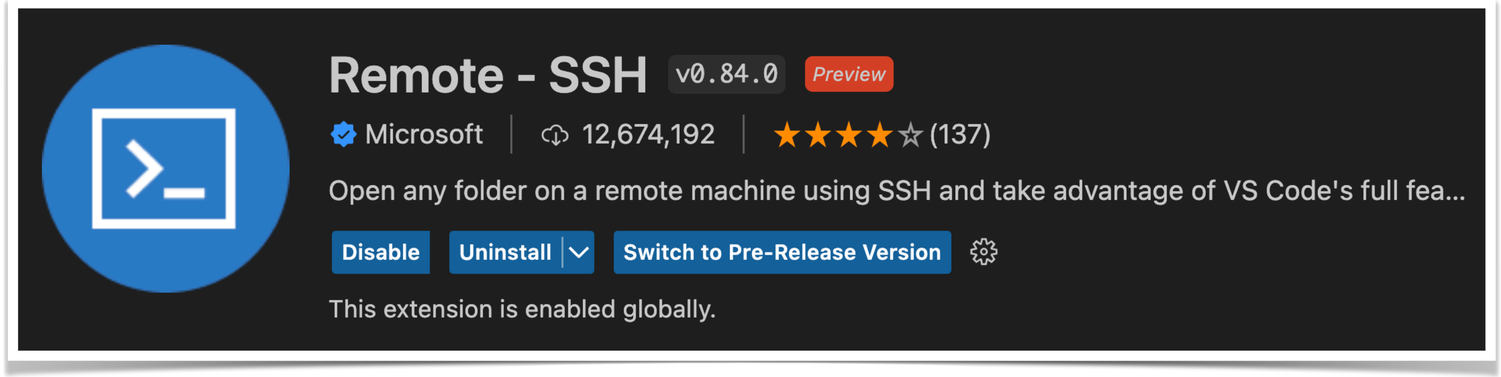


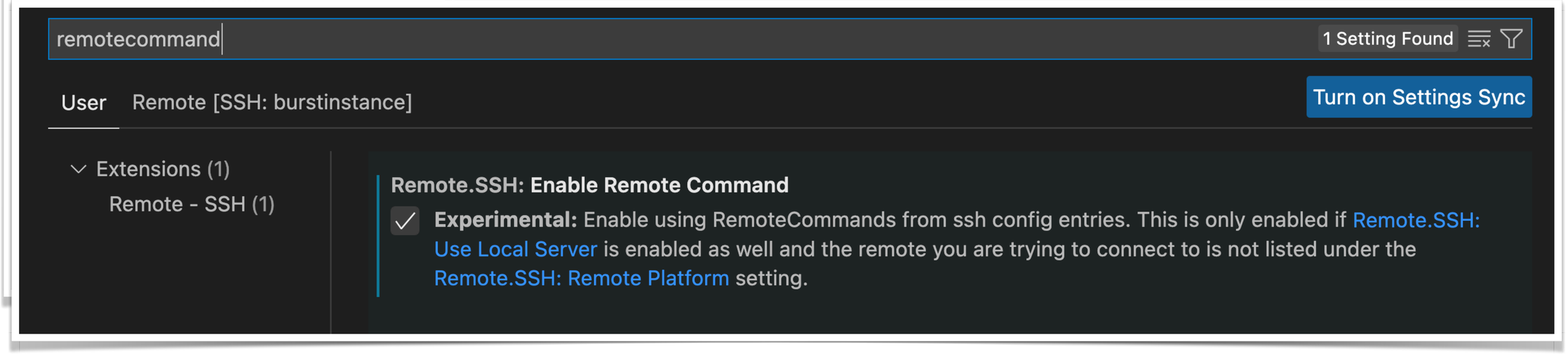


Setup

Enable Remote Command







2.2 Training

Train the provided network (or another network of your choice) on the `places365` dataset for 4 epochs, using mixed-precision training and two GPUs. Ensure that your GPU utilization is close to optimal: run `nvidia-smi` and include a screenshot. How many images are you processing per second (say how you computed this number, and provide a screenshot with your source information)? Note: with proper settings, training 4 epochs should take less than 1 hour. Make sure that you have started your instance and training for best performance as described in the `README.md`.

Open tensorboard (by either port forwarding, or within VSCode), and take a screenshot of the loss and accuracy after 4 epochs.

How big is the data set (in (giga)bytes)? Include a screenshot of the appropriate command to get the size of the dataset.

Train the provided Network

```
@dataclasses.dataclass
class PlacesTrainingConfig:
    data: PlacesDataConfig = PlacesDataConfig()
    model: PlacesModelConfig = PlacesModelConfig()
    optim: PlacesOptimConfig = PlacesOptimConfig()
    lightning: Dict[str, Any] = dataclasses.field(default_factory=dict)
    precision: int = 16
    batch_size: int = 1024
    max_epochs: int = 4
    gpus: int = 2
```

- Precision: Mixed precision combines the use of both 32 and 16-bit floating points to reduce memory footprint during model training, resulting in improved performance
- Max_epochs: Setting max_epochs=4 will ensure that training won't happen after 4 epochs

Nvidia-smi

```
(/ext3/conda/bootcamp) Singularity> nvidia-smi
Sun Oct  2 17:02:30 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.48.07    Driver Version: 515.48.07    CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:04.0 Off |                    0 |
| N/A   45C    P0    61W / 300W |  14492MiB / 16384MiB |     94%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000000:00:05.0 Off |                    0 |
| N/A   46C    P0    65W / 300W |  14492MiB / 16384MiB |     96%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

```
Epoch 1:  42%|████████████████▍                | 756/1798 [06:11<08:31,  2.04it/s, loss=2.47, v_num=0, accuracy=0.391]
```

Size of the dataset

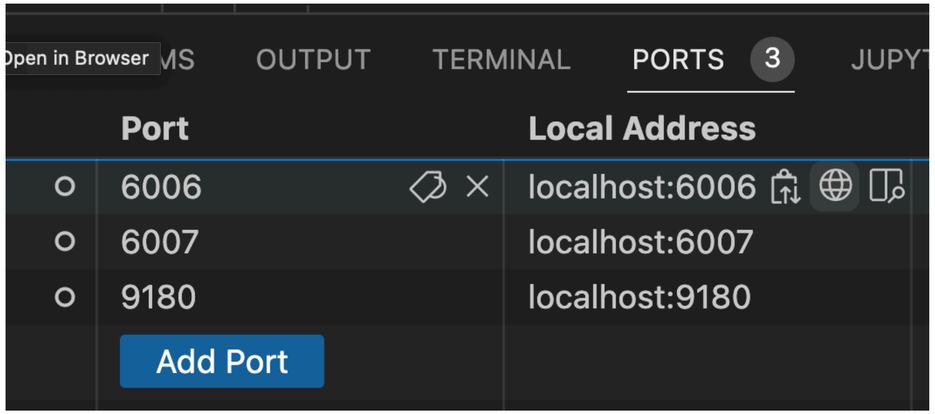
du -h /dev/shm/places365.squashfs

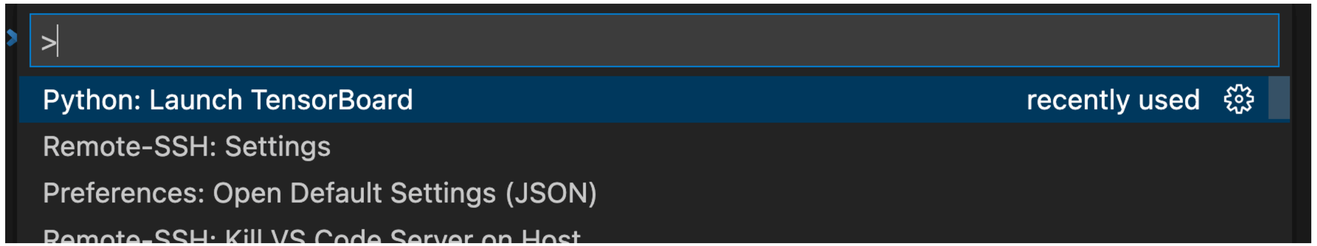
```
bash-4.4$ hostname
b-17-20
bash-4.4$ du -h /dev/shm/places365.squashfs
23G     /dev/shm/places365.squashfs
```

Tensorboard

- Tensorboard
  - Provide measurements and visualizations needed during the machine learning workflow
  - Help track experiment metrics like loss and accuracy
  - /output/2022-09-25: called event file into which Tensorboard saves the summary data
- Port Forwarding
  - To launch the visualization server
  - Run tensorboard --logdir=$EVENT_FILE
  - View your visualization in a web browser
- VSCode
  - Launch Tensorboard

```
Singularity> cd outputs/
Singularity>
Singularity> tensorboard --logdir 2022-10-02
```





2.3 Testing

Create a new file `tests/test_network.py`, and write a smoke test for the network. Parametrize the test so that it runs both on CPU and GPU. Run the test within the VSCode test explorer, and take a screenshot.

Smoke Test (CV)

- Use pytest.mark.parametrize decorator
  - enables parametrization of arguments for a test function
  - ensures the model runs both on GPU and CPU
- Create configuration with a small model
  - width_multiplier: Reduce the number of channels used in each layer, therefore much faster than the full model
- Create model
- Set up dataset: get the first batch from training dataloader; save this batch & load it back for the test
- Make sure the model runs
- Check that we can compute loss

```
@pytest.mark.parametrize(
    'device',
    ['cpu',
     pytest.param('cuda', \
        marks=pytest.mark.skipif(not torch.cuda.is_available(), reason='Cuda required'))])
```

```
config = bootcamp.model.PlacesTrainingConfig()
config.model.width_multiplier = 0.25
```

```
model = bootcamp.model.PlacesModel(config)
model = model.to(device)
```

```
dm = bootcamp.dataset.PlacesDataModule(16, '/places365', 1)
dm.setup()
batch = next(iter(dm.train_dataloader()))
torch.save(batch, 'testdata/batch.pt')
```

```
batch = torch.load('testdata/batch.pt')
batch = [v.to(device) for v in batch]
```

3.3 Testing

Use the huggingface facilities (or otherwise) to save a small sample of data (e.g. 8 review / rating pairs). Write a smoke test which loads that sample and runs it through the model. Run the test in the VSCode test explorer and include a screenshot.

Smoke Test (NLP)

- Create model
- Set up dataset: save a small sample of data & load it back for the test
- Make sure the model runs

```
# Create model
model = bootcamp.model.PretrainedBertModel()
```

```
# Setup dataset
ds = bootcamp.dataset._load_ds()
ds_train = ds["train"].shuffle(seed=42).select(range(8))
ds_test = ds["test"].shuffle(seed=42).select(range(8))
ds_train.save_to_disk("./testdata/train")
ds_test.save_to_disk("./testdata/test")
```

```
# Ideally: save this batch separately and load it back for the test
# as setup for dataset takes a while
ds_train = load_from_disk("./testdata/train")
ds_test = load_from_disk("./testdata/test")
ds_train.set_format('torch')
ds_test.set_format('torch')
```

```
ds_train_batch = ds_train[:8]
ds_test_batch = ds_test[:8]

# Make sure the model runs
model.forward(ds_train_batch)
model.forward(ds_test_batch)
```