Deep Learning Report

Lab Assignment - 9

Ayush Abrol
B20AI052

Question 01

Aim:

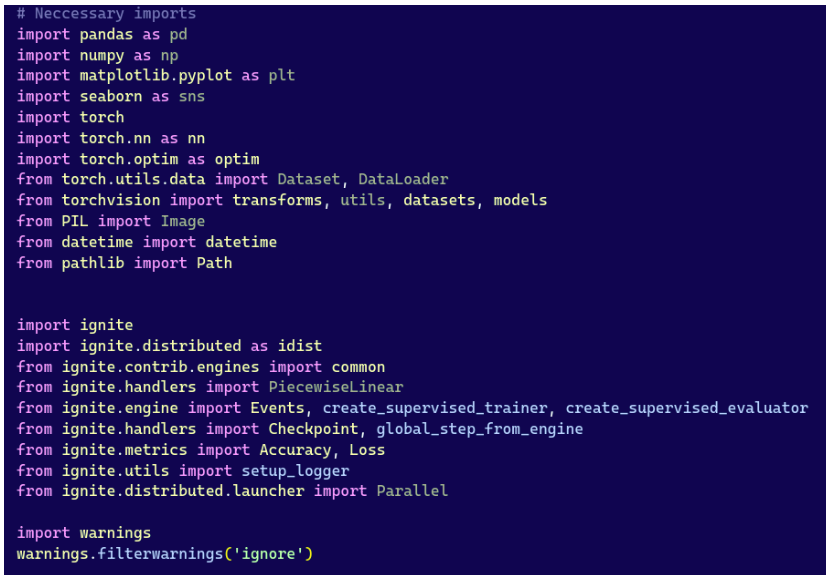
Train a ResNet18 model on SVHN dataset using DataParallelism and DistributedDataParallelism for 30 epochs. You can use any library of your choice for parallel training of your model in PyTorch. [35 marks]

- Use a batch size of 32 and train on a single GPU node.
- Use a batch size of 64 and train and two GPU nodes first using DataParallelism and then using DistributedDataParallelism.
    - Report all the hyper-parameters used for Parallel Training.
    - Compare the time (in seconds) and report the speed up. Show this speed up using a graphical representation.
    - Describe your observations in terms of memory usage for multi-node training.

Reference taken from Colab by Dr. Anush: https://colab.research.google.com/drive/1BRr-nL5pevvrjY1SXAV0XfZ9OHKq1DGW?usp=share_link

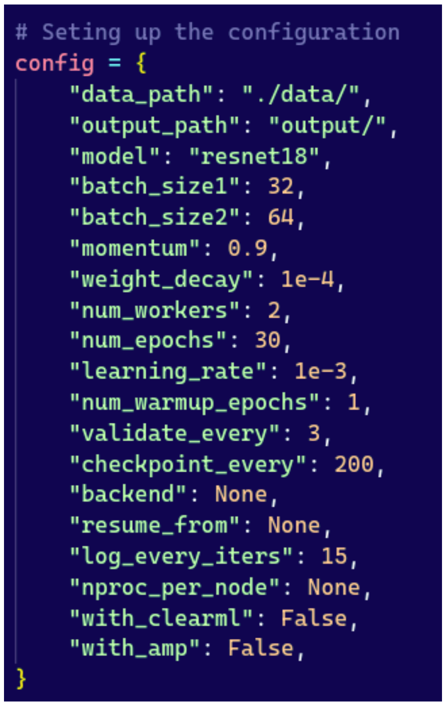
Procedure:

- Imported all the necessary libraries



```
# Neccessary imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms, utils, datasets, models
from PIL import Image
from datetime import datetime
from pathlib import Path


import ignite
import ignite.distributed as idist
from ignite.contrib.engines import common
from ignite.handlers import PiecewiseLinear
from ignite.engine import Events, create_supervised_trainer, create_supervised_evaluator
from ignite.handlers import Checkpoint, global_step_from_engine
from ignite.metrics import Accuracy, Loss
from ignite.utils import setup_logger
from ignite.distributed.launcher import Parallel

import warnings
warnings.filterwarnings('ignore')
```

- Setup the configuration with the following details:



```
# Seting up the configuration
config = {
    "data_path": "./data/",
    "output_path": "output/",
    "model": "resnet18",
    "batch_size1": 32,
    "batch_size2": 64,
    "momentum": 0.9,
    "weight_decay": 1e-4,
    "num_workers": 2,
    "num_epochs": 30,
    "learning_rate": 1e-3,
    "num_warmup_epochs": 1,
    "validate_every": 3,
    "checkpoint_every": 200,
    "backend": None,
    "resume_from": None,
    "log_every_iters": 15,
    "nproc_per_node": None,
    "with_clearml": False,
    "with_amp": False,
}
```

These were the hyperparameters used.

- Took batch_size1 = 32 for part1 and batch_size2 = 64 for part2.
- Then, created a function for importing SVHN dataset with variable batch_size which can be passed as an argument to the function,
- Visualization of the dataset:

Labels: tensor([3, 3, 5, 1, 7, 1, 1, 4, 3, 0, 5, 1, 6, 2, 1, 7, 9, 5, 6, 6, 1, 1, 4, 6,
9, 4, 2, 1, 2, 2, 2, 2])

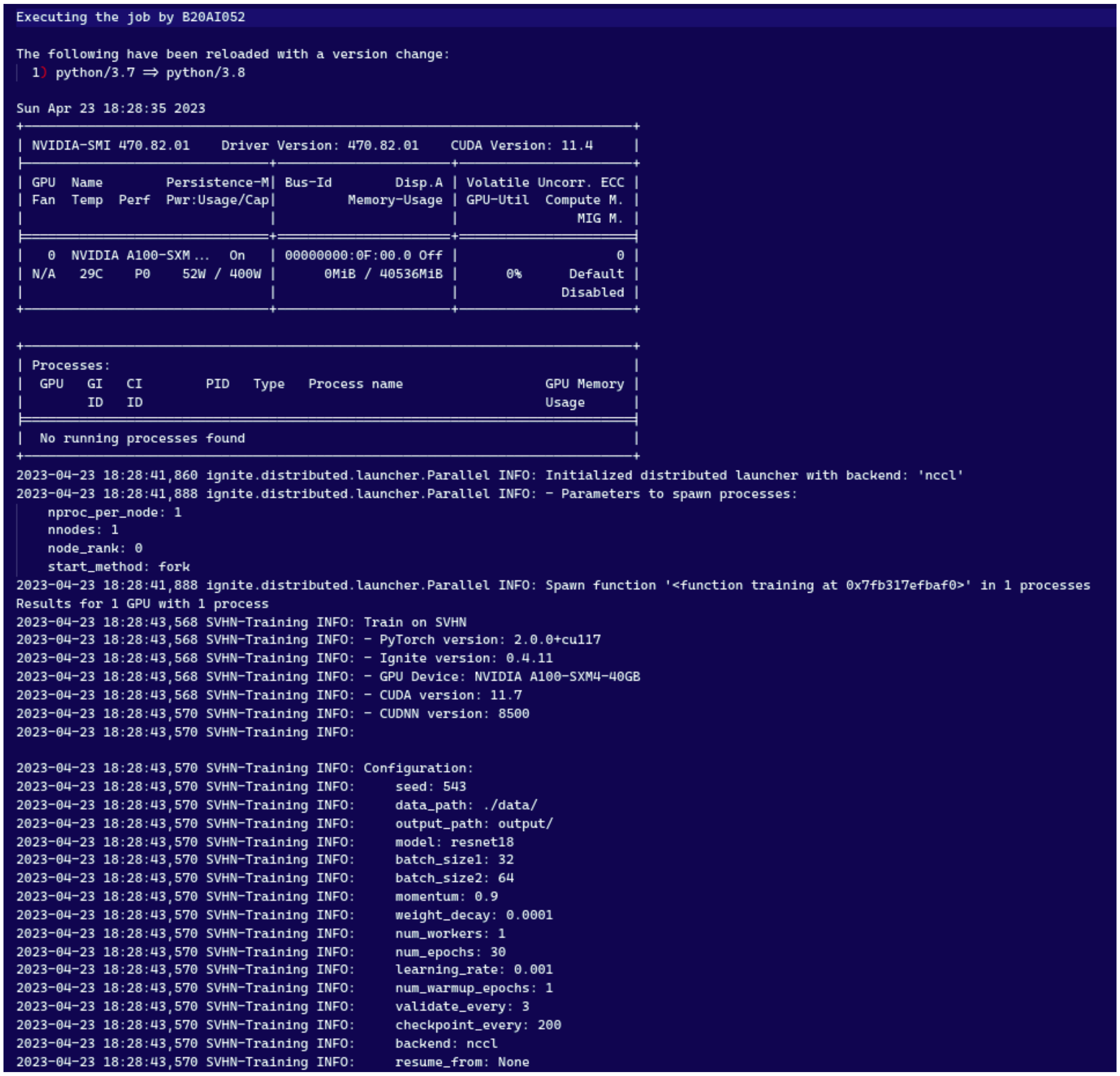
- Wrapped the data loader with ignite.distributed's idist function for DistributedDataParallelism part.

- Defined the torchvision’s pretrained ResNet18 model wrapped inside idist function.
- Similirality defined the optimizer and loss function as SGD and CrossEntropyLoss respectively and also wrapped them using idist.
- Then, I defined two functions: get_save_handler and load_checkpoint. Here's a brief description of what each function does:
    - **get_save_handler(config)**: This function takes a dictionary config as input and returns either a path to a checkpoint file or a ClearMLSaver object. If the with_clearml key in the config dictionary is set to True, the function imports the ClearMLSaver class from the ignite.contrib.handlers.clearml_logger module and returns an instance of this class with the output directory set to the value of the output_path key in the config dictionary. If with_clearml is False, the function simply returns the value of the output_path key in the config dictionary.
    - **load_checkpoint(resume_from)**: This function takes a string resume_from as input, which is the path to a checkpoint file. The function first checks if the file exists and raises an assertion error if it doesn't. If the file exists, the function loads the checkpoint using the torch.load function with the map_location parameter set to "cpu", and returns the resulting dictionary. Note that this function assumes that the checkpoint was saved using PyTorch's torch.save function with the default settings, which saves both the model state dictionary and optimizer state dictionary.
- Then, defined three functions as: create_trainer, create_evaluator, and setup_rank_zero. Here's a brief description of what each function does:
    - **create_trainer(model, optimizer, criterion, train_sampler, config, logger)**: This function creates a PyTorch Ignite trainer object for a given model, optimizer, and criterion function, along with a train_sampler object and a config dictionary. The function sets up several training handlers using the ignite.handlers module, such as Checkpoint, EarlyStopping, and LearningRateScheduler. The function also sets up a logger and returns the trainer object.
    - **create_evaluator(model, metrics, config)**: This function creates a PyTorch Ignite evaluator object for a given model, a list of metrics, and a config dictionary. The function also sets the device and AMP mode for the evaluator, and returns the evaluator object.

    - **setup_rank_zero(logger, config)**: This function sets up the output path and ClearML configuration for the training run. It takes a logger object and a config dictionary as input. The function first sets up the output path based on the current time and a folder name. It then creates the output directory if it doesn't already exist. Finally, if ClearML logging is enabled in the config dictionary, the function creates a ClearML Task object and connects it to the configuration dictionary.
- Then, defined a log_basic_info function which logs basic information about the training configuration, such as PyTorch and Ignite versions, GPU device (if available), and the entire configuration dictionary. It also includes information about the distributed setting if running on multiple nodes.
- Also, the log_metrics function logs evaluation metrics for a given epoch and a given tag (e.g., "validation" or "test"). It includes the elapsed time for the evaluation and the metrics themselves in a formatted output.

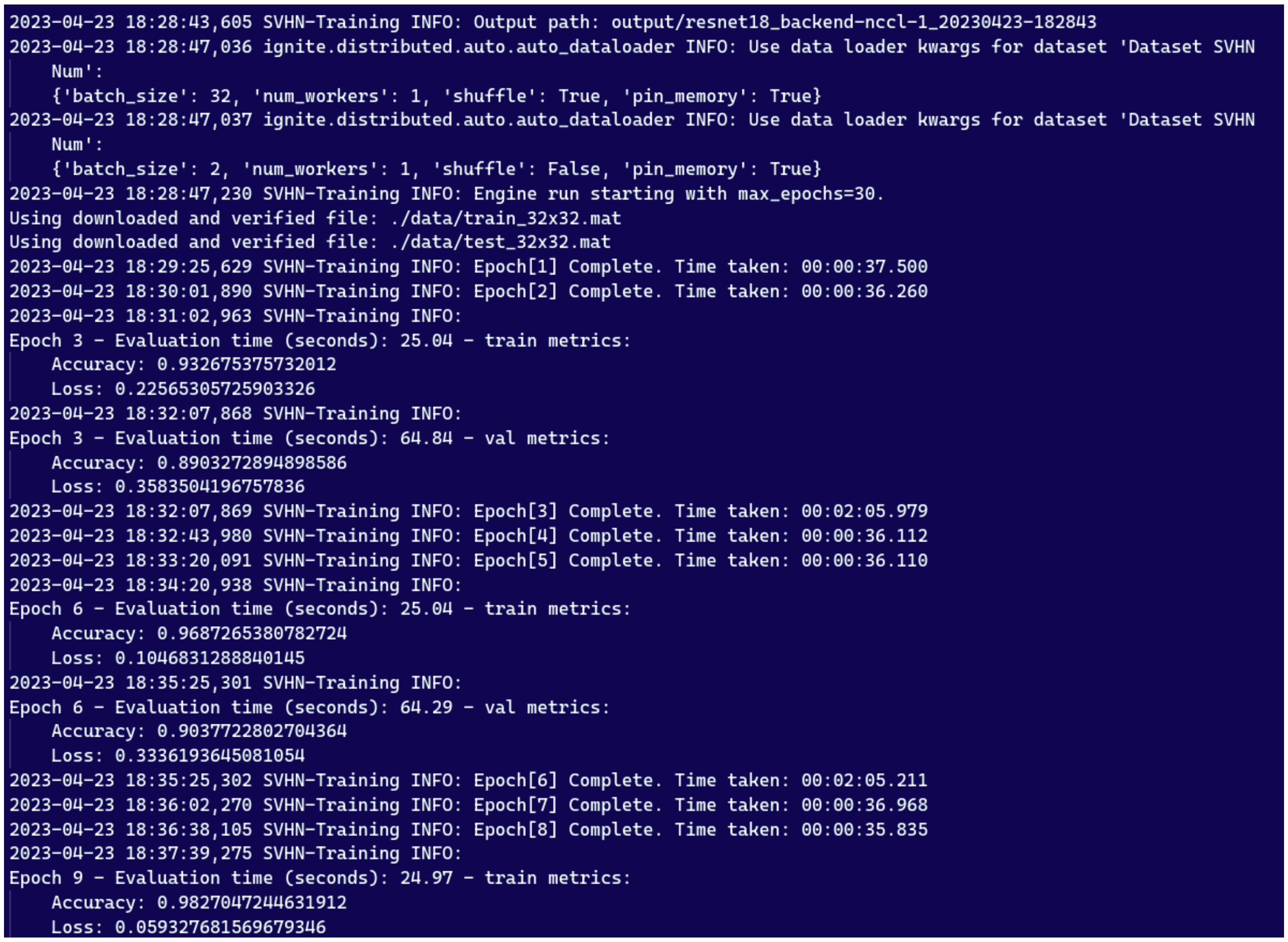
- Then, finally defined the main training loop for a ResNet18 network model on the SVHN dataset. The function sets up the necessary components for training, including loading the data, defining the model, optimizer, and loss function, setting up training and validation evaluators, and setting up logging for metrics and checkpoints. The function then runs the training loop and logs the necessary information. At the end of the function, the trained model is saved. The function is designed to work in a distributed setting and handles different processes/threads appropriately. The code makes use of the PyTorch Ignite library for training and evaluation.

- FIrstly, trained on a single GPU node with batch_size = 32 (on the hpc server) for which the code is in the python file B20AI052_Lab_Assignment_9_A.py with the batch file as B20AI052_Lab_Assignment_9_A.sh
- Results of the slurm-43579.out file are:



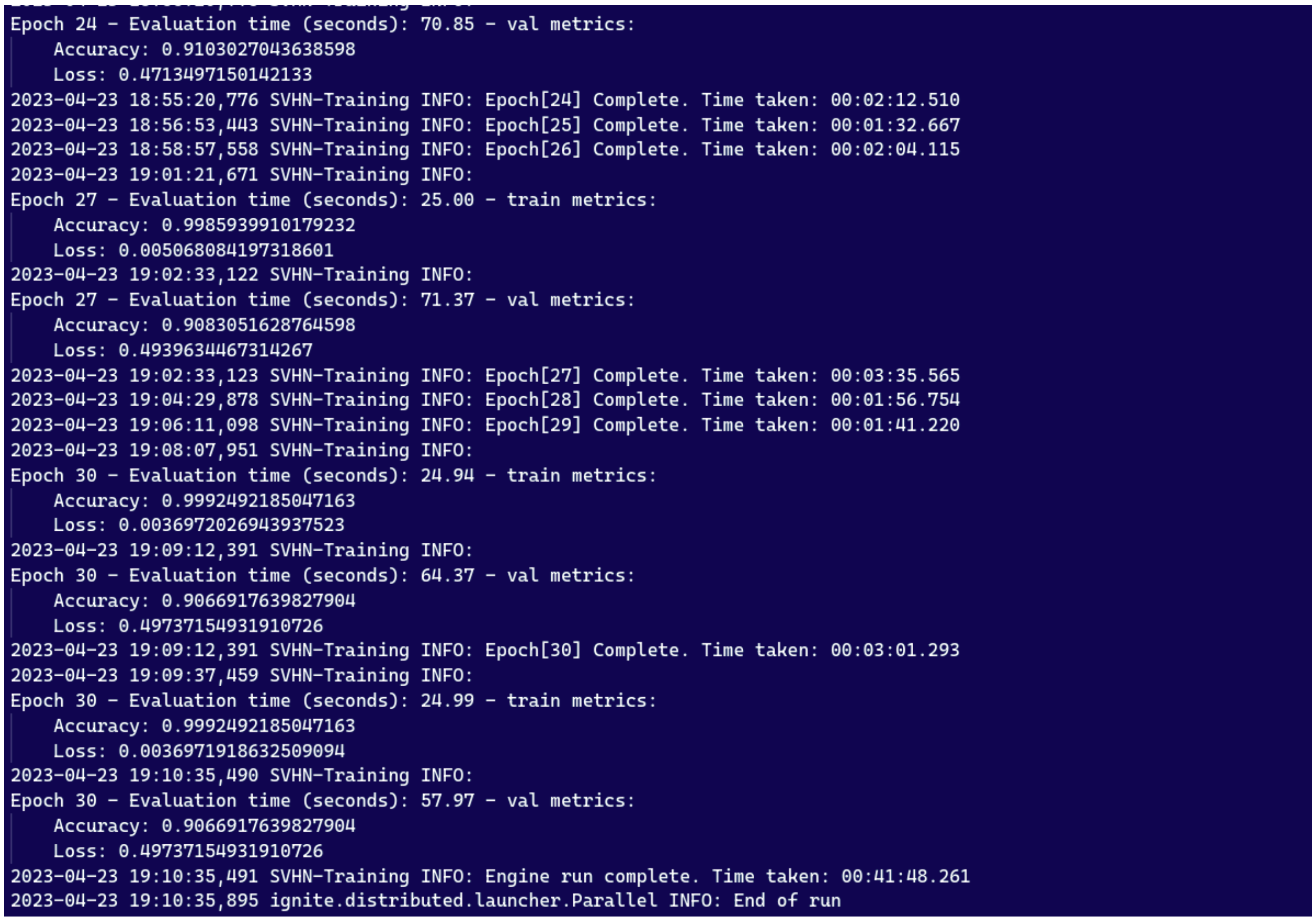
```
Executing the job by B20AI052

The following have been reloaded with a version change:
  1) python/3.7 => python/3.8

Sun Apr 23 18:28:35 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.82.01    Driver Version: 470.82.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| 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  NVIDIA A100-SXM...  On   | 00000000:0F:00.0 Off |                    0 |
| N/A   29C    P0    52W / 400W |      0MiB / 40536MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
2023-04-23 18:28:41,860 ignite.distributed.launcher.Parallel INFO: Initialized distributed launcher with backend: 'nccl'
2023-04-23 18:28:41,888 ignite.distributed.launcher.Parallel INFO: - Parameters to spawn processes:
    nproc_per_node: 1
    nnodes: 1
    node_rank: 0
    start_method: fork
2023-04-23 18:28:41,888 ignite.distributed.launcher.Parallel INFO: Spawn function '<function training at 0x7fb317efbaf0>' in 1 processes
Results for 1 GPU with 1 process
2023-04-23 18:28:43,568 SVHN-Training INFO: Train on SVHN
2023-04-23 18:28:43,568 SVHN-Training INFO: - PyTorch version: 2.0.0+cu117
2023-04-23 18:28:43,568 SVHN-Training INFO: - Ignite version: 0.4.11
2023-04-23 18:28:43,568 SVHN-Training INFO: - GPU Device: NVIDIA A100-SXM4-40GB
2023-04-23 18:28:43,568 SVHN-Training INFO: - CUDA version: 11.7
2023-04-23 18:28:43,570 SVHN-Training INFO: - CUDNN version: 8500
2023-04-23 18:28:43,570 SVHN-Training INFO:

2023-04-23 18:28:43,570 SVHN-Training INFO: Configuration:
2023-04-23 18:28:43,570 SVHN-Training INFO:     seed: 543
2023-04-23 18:28:43,570 SVHN-Training INFO:     data_path: ./data/
2023-04-23 18:28:43,570 SVHN-Training INFO:     output_path: output/
2023-04-23 18:28:43,570 SVHN-Training INFO:     model: resnet18
2023-04-23 18:28:43,570 SVHN-Training INFO:     batch_size1: 32
2023-04-23 18:28:43,570 SVHN-Training INFO:     batch_size2: 64
2023-04-23 18:28:43,570 SVHN-Training INFO:     momentum: 0.9
2023-04-23 18:28:43,570 SVHN-Training INFO:     weight_decay: 0.0001
2023-04-23 18:28:43,570 SVHN-Training INFO:     num_workers: 1
2023-04-23 18:28:43,570 SVHN-Training INFO:     num_epochs: 30
2023-04-23 18:28:43,570 SVHN-Training INFO:     learning_rate: 0.001
2023-04-23 18:28:43,570 SVHN-Training INFO:     num_warmup_epochs: 1
2023-04-23 18:28:43,570 SVHN-Training INFO:     validate_every: 3
2023-04-23 18:28:43,570 SVHN-Training INFO:     checkpoint_every: 200
2023-04-23 18:28:43,570 SVHN-Training INFO:     backend: nccl
2023-04-23 18:28:43,570 SVHN-Training INFO:     resume_from: None
```

- Job started executing and Python3.8 was loaded. And results of 1 GPU of the NVIDIA A100 can be seen using nvidia-smi. Then, all the config details are displayed (hyperparameters).



```
2023-04-23 18:28:43,605 SVHN-Training INFO: Output path: output/resnet18_backend-nccl-1_20230423-182843
2023-04-23 18:28:47,036 ignite.distributed.auto.auto_dataloader INFO: Use data loader kwargs for dataset 'Dataset SVHN
    Num':
    {'batch_size': 32, 'num_workers': 1, 'shuffle': True, 'pin_memory': True}
2023-04-23 18:28:47,037 ignite.distributed.auto.auto_dataloader INFO: Use data loader kwargs for dataset 'Dataset SVHN
    Num':
    {'batch_size': 2, 'num_workers': 1, 'shuffle': False, 'pin_memory': True}
2023-04-23 18:28:47,230 SVHN-Training INFO: Engine run starting with max_epochs=30.
Using downloaded and verified file: ./data/train_32x32.mat
Using downloaded and verified file: ./data/test_32x32.mat
2023-04-23 18:29:25,629 SVHN-Training INFO: Epoch[1] Complete. Time taken: 00:00:37.500
2023-04-23 18:30:01,890 SVHN-Training INFO: Epoch[2] Complete. Time taken: 00:00:36.260
2023-04-23 18:31:02,963 SVHN-Training INFO:
Epoch 3 - Evaluation time (seconds): 25.04 - train metrics:
    Accuracy: 0.932675375732012
    Loss: 0.22565305725903326
2023-04-23 18:32:07,868 SVHN-Training INFO:
Epoch 3 - Evaluation time (seconds): 64.84 - val metrics:
    Accuracy: 0.8903272894898586
    Loss: 0.3583504196757836
2023-04-23 18:32:07,869 SVHN-Training INFO: Epoch[3] Complete. Time taken: 00:02:05.979
2023-04-23 18:32:43,980 SVHN-Training INFO: Epoch[4] Complete. Time taken: 00:00:36.112
2023-04-23 18:33:20,091 SVHN-Training INFO: Epoch[5] Complete. Time taken: 00:00:36.110
2023-04-23 18:34:20,938 SVHN-Training INFO:
Epoch 6 - Evaluation time (seconds): 25.04 - train metrics:
    Accuracy: 0.9687265380782724
    Loss: 0.1046831288840145
2023-04-23 18:35:25,301 SVHN-Training INFO:
Epoch 6 - Evaluation time (seconds): 64.29 - val metrics:
    Accuracy: 0.9037722802704364
    Loss: 0.3336193645081054
2023-04-23 18:35:25,302 SVHN-Training INFO: Epoch[6] Complete. Time taken: 00:02:05.211
2023-04-23 18:36:02,270 SVHN-Training INFO: Epoch[7] Complete. Time taken: 00:00:36.968
2023-04-23 18:36:38,105 SVHN-Training INFO: Epoch[8] Complete. Time taken: 00:00:35.835
2023-04-23 18:37:39,275 SVHN-Training INFO:
Epoch 9 - Evaluation time (seconds): 24.97 - train metrics:
    Accuracy: 0.9827047244631912
    Loss: 0.059327681569679346
```

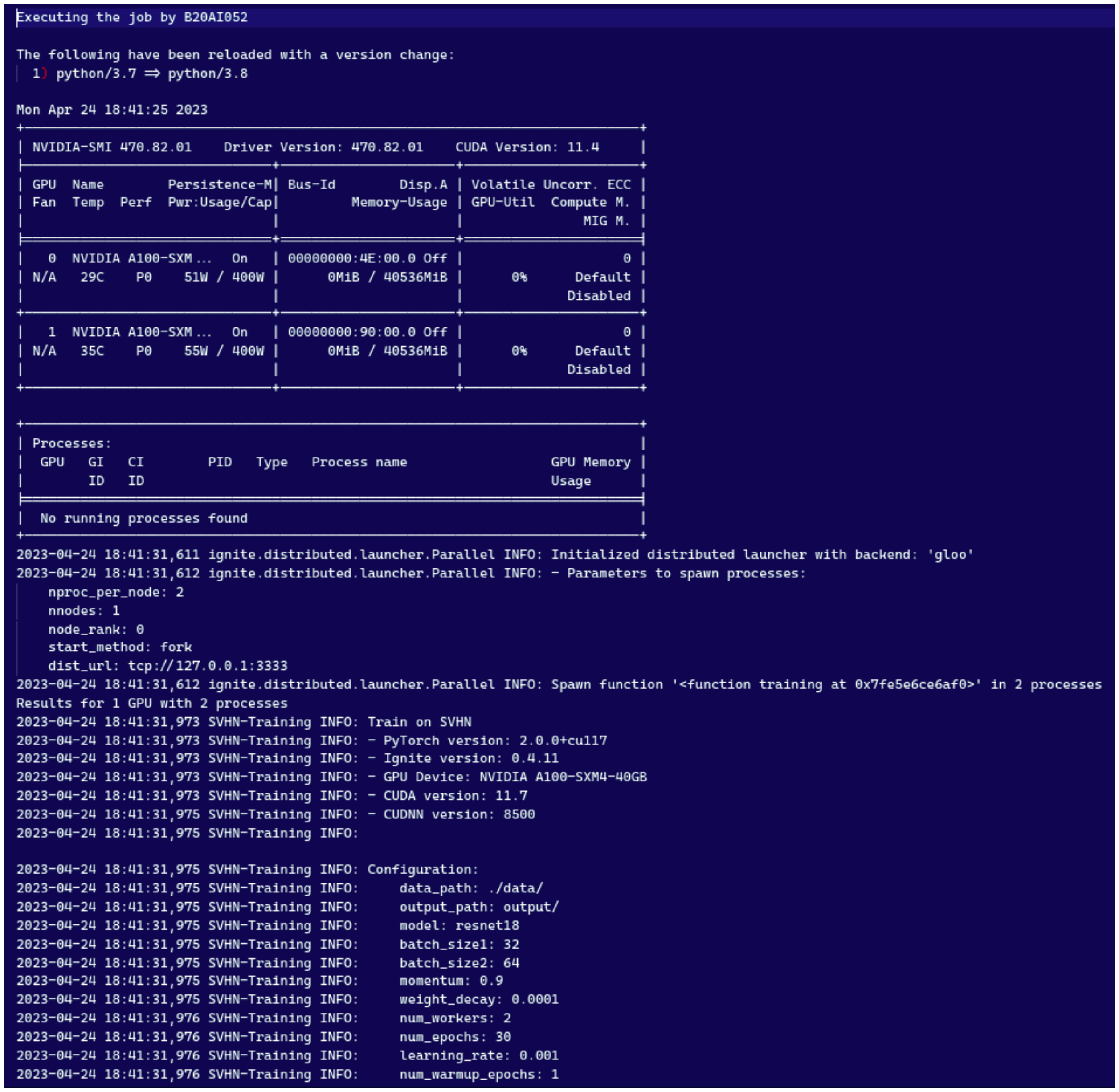
- Training started using 1 GPU only, For each epoch, we see the evaluation time, accuracy and loss for both the training and the validation data.



```
Epoch 24 - Evaluation time (seconds): 70.85 - val metrics:
    Accuracy: 0.9103027043638598
    Loss: 0.4713497150142133
2023-04-23 18:55:20,776 SVHN-Training INFO: Epoch[24] Complete. Time taken: 00:02:12.510
2023-04-23 18:56:53,443 SVHN-Training INFO: Epoch[25] Complete. Time taken: 00:01:32.667
2023-04-23 18:58:57,558 SVHN-Training INFO: Epoch[26] Complete. Time taken: 00:02:04.115
2023-04-23 19:01:21,671 SVHN-Training INFO:
Epoch 27 - Evaluation time (seconds): 25.00 - train metrics:
    Accuracy: 0.9985939910179232
    Loss: 0.005068084197318601
2023-04-23 19:02:33,122 SVHN-Training INFO:
Epoch 27 - Evaluation time (seconds): 71.37 - val metrics:
    Accuracy: 0.9083051628764598
    Loss: 0.4939634467314267
2023-04-23 19:02:33,123 SVHN-Training INFO: Epoch[27] Complete. Time taken: 00:03:35.565
2023-04-23 19:04:29,878 SVHN-Training INFO: Epoch[28] Complete. Time taken: 00:01:56.754
2023-04-23 19:06:11,098 SVHN-Training INFO: Epoch[29] Complete. Time taken: 00:01:41.220
2023-04-23 19:08:07,951 SVHN-Training INFO:
Epoch 30 - Evaluation time (seconds): 24.94 - train metrics:
    Accuracy: 0.9992492185047163
    Loss: 0.003697202694393752З
2023-04-23 19:09:12,391 SVHN-Training INFO:
Epoch 30 - Evaluation time (seconds): 64.37 - val metrics:
    Accuracy: 0.9066917639827904
    Loss: 0.49737154931910726
2023-04-23 19:09:12,391 SVHN-Training INFO: Epoch[30] Complete. Time taken: 00:03:01.293
2023-04-23 19:09:37,459 SVHN-Training INFO:
Epoch 30 - Evaluation time (seconds): 24.99 - train metrics:
    Accuracy: 0.9992492185047163
    Loss: 0.0036971918632509094
2023-04-23 19:10:35,490 SVHN-Training INFO:
Epoch 30 - Evaluation time (seconds): 57.97 - val metrics:
    Accuracy: 0.9066917639827904
    Loss: 0.49737154931910726
2023-04-23 19:10:35,491 SVHN-Training INFO: Engine run complete. Time taken: 00:41:48.261
2023-04-23 19:10:35,895 ignite.distributed.launcher.Parallel INFO: End of run
```

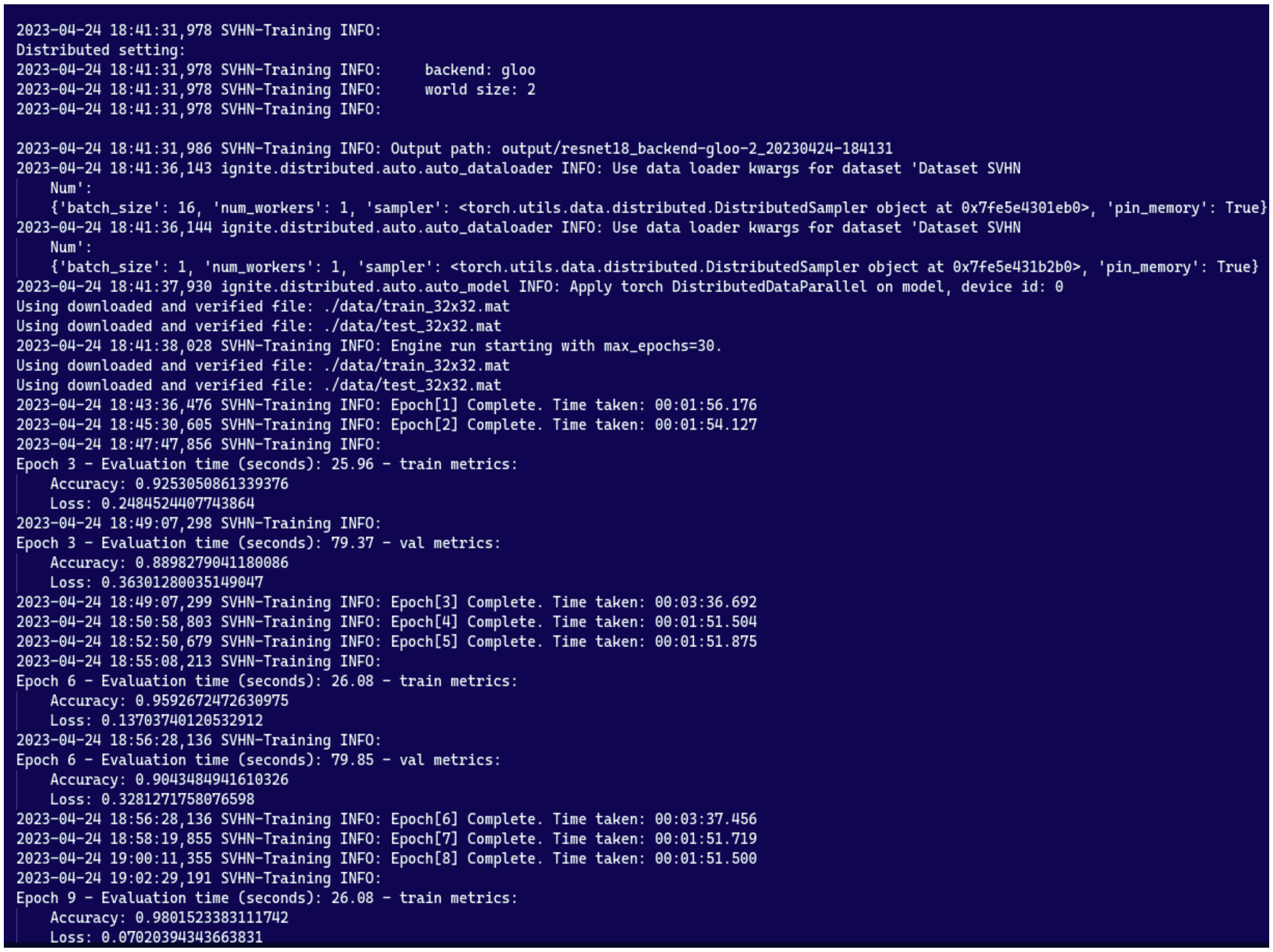
- Training was done till 30 epochs and final results were:
    - Training acc = 99.2492 %
    - Training loss = 0.0036
    - Validation acc = 90.669%
    - Validation loss = 0.4973
    - Total training time = 41 minutes 48 seconds.

- Then, trained on 2 GPU nodes with batch_size = 64 (on the hpc server) for which the code is in the python file B20AI052_Lab_Assignment_9_B.py with the batch file as B20AI052_Lab_Assignment_9_B.sh.
- Results of the slurm-44492.out file are as follows:



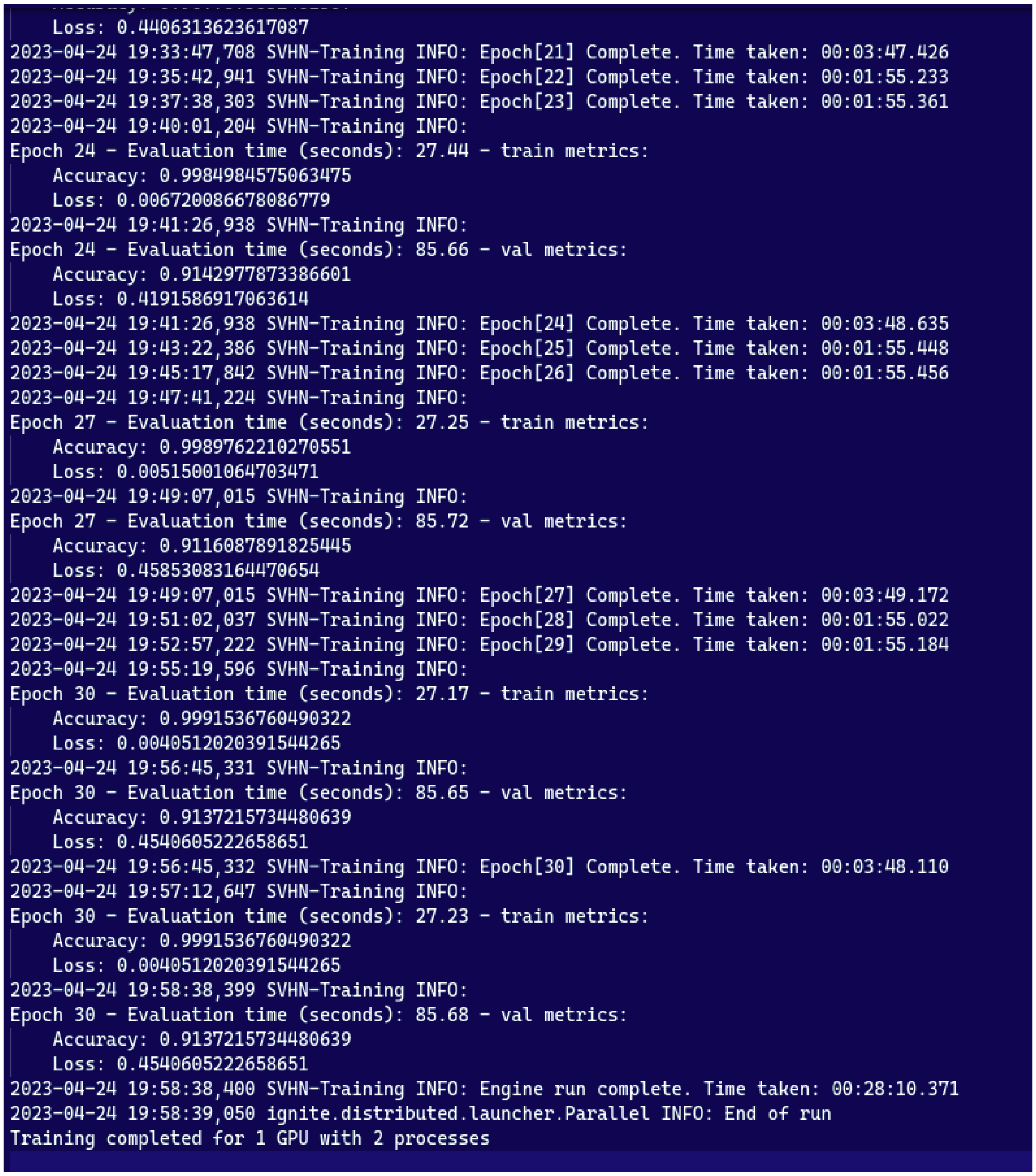
```
Executing the job by B20AI052

The following have been reloaded with a version change:
  1) python/3.7 => python/3.8

Mon Apr 24 18:41:25 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.82.01    Driver Version: 470.82.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| 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  NVIDIA A100-SXM...  On   | 00000000:4E:00.0 Off |                    0 |
| N/A   29C    P0    51W / 400W |      0MiB / 40536MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA A100-SXM...  On   | 00000000:90:00.0 Off |                    0 |
| N/A   35C    P0    55W / 400W |      0MiB / 40536MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
2023-04-24 18:41:31,611 ignite.distributed.launcher.Parallel INFO: Initialized distributed launcher with backend: 'gloo'
2023-04-24 18:41:31,612 ignite.distributed.launcher.Parallel INFO: - Parameters to spawn processes:
    nproc_per_node: 2
    nnodes: 1
    node_rank: 0
    start_method: fork
    dist_url: tcp://127.0.0.1:3333
2023-04-24 18:41:31,612 ignite.distributed.launcher.Parallel INFO: Spawn function '<function training at 0x7fe5e6ce6af0>' in 2 processes
Results for 1 GPU with 2 processes
2023-04-24 18:41:31,973 SVHN-Training INFO: Train on SVHN
2023-04-24 18:41:31,973 SVHN-Training INFO: - PyTorch version: 2.0.0+cu117
2023-04-24 18:41:31,973 SVHN-Training INFO: - Ignite version: 0.4.11
2023-04-24 18:41:31,973 SVHN-Training INFO: - GPU Device: NVIDIA A100-SXM4-40GB
2023-04-24 18:41:31,973 SVHN-Training INFO: - CUDA version: 11.7
2023-04-24 18:41:31,975 SVHN-Training INFO: - CUDNN version: 8500
2023-04-24 18:41:31,975 SVHN-Training INFO: 

2023-04-24 18:41:31,975 SVHN-Training INFO: Configuration:
2023-04-24 18:41:31,975 SVHN-Training INFO:     data_path: ./data/
2023-04-24 18:41:31,975 SVHN-Training INFO:     output_path: output/
2023-04-24 18:41:31,975 SVHN-Training INFO:     model: resnet18
2023-04-24 18:41:31,975 SVHN-Training INFO:     batch_size1: 32
2023-04-24 18:41:31,975 SVHN-Training INFO:     batch_size2: 64
2023-04-24 18:41:31,975 SVHN-Training INFO:     momentum: 0.9
2023-04-24 18:41:31,975 SVHN-Training INFO:     weight_decay: 0.0001
2023-04-24 18:41:31,976 SVHN-Training INFO:     num_workers: 2
2023-04-24 18:41:31,976 SVHN-Training INFO:     num_epochs: 30
2023-04-24 18:41:31,976 SVHN-Training INFO:     learning_rate: 0.001
2023-04-24 18:41:31,976 SVHN-Training INFO:     num_warmup_epochs: 1
```

- Job started executing and Python3.8 was loaded. And results of 2 GPUs of the NVIDIA A100 can be seen using nvidia-smi. Then, all the config details (hyperparameters) are displayed.



```
2023-04-24 18:41:31,978 SVHN-Training INFO:
Distributed setting:
2023-04-24 18:41:31,978 SVHN-Training INFO:     backend: gloo
2023-04-24 18:41:31,978 SVHN-Training INFO:     world size: 2
2023-04-24 18:41:31,978 SVHN-Training INFO:

2023-04-24 18:41:31,986 SVHN-Training INFO: Output path: output/resnet18_backend-gloo-2_20230424-184131
2023-04-24 18:41:36,143 ignite.distributed.auto.auto_dataloader INFO: Use data loader kwargs for dataset 'Dataset SVHN
    Num':
    {'batch_size': 16, 'num_workers': 1, 'sampler': <torch.utils.data.distributed.DistributedSampler object at 0x7fe5e4301eb0>, 'pin_memory': True}
2023-04-24 18:41:36,144 ignite.distributed.auto.auto_dataloader INFO: Use data loader kwargs for dataset 'Dataset SVHN
    Num':
    {'batch_size': 1, 'num_workers': 1, 'sampler': <torch.utils.data.distributed.DistributedSampler object at 0x7fe5e431b2b0>, 'pin_memory': True}
2023-04-24 18:41:37,930 ignite.distributed.auto.auto_model INFO: Apply torch DistributedDataParallel on model, device id: 0
Using downloaded and verified file: ./data/train_32x32.mat
Using downloaded and verified file: ./data/test_32x32.mat
2023-04-24 18:41:38,028 SVHN-Training INFO: Engine run starting with max_epochs=30.
Using downloaded and verified file: ./data/train_32x32.mat
Using downloaded and verified file: ./data/test_32x32.mat
2023-04-24 18:43:36,476 SVHN-Training INFO: Epoch[1] Complete. Time taken: 00:01:56.176
2023-04-24 18:45:30,605 SVHN-Training INFO: Epoch[2] Complete. Time taken: 00:01:54.127
2023-04-24 18:47:47,856 SVHN-Training INFO:
Epoch 3 - Evaluation time (seconds): 25.96 - train metrics:
    Accuracy: 0.9253050861339376
    Loss: 0.2484524407743864
2023-04-24 18:49:07,298 SVHN-Training INFO:
Epoch 3 - Evaluation time (seconds): 79.37 - val metrics:
    Accuracy: 0.8898279041180086
    Loss: 0.36301280035149047
2023-04-24 18:49:07,299 SVHN-Training INFO: Epoch[3] Complete. Time taken: 00:03:36.692
2023-04-24 18:50:58,803 SVHN-Training INFO: Epoch[4] Complete. Time taken: 00:01:51.504
2023-04-24 18:52:50,679 SVHN-Training INFO: Epoch[5] Complete. Time taken: 00:01:51.875
2023-04-24 18:55:08,213 SVHN-Training INFO:
Epoch 6 - Evaluation time (seconds): 26.08 - train metrics:
    Accuracy: 0.9592672472630975
    Loss: 0.13703740120532912
2023-04-24 18:56:28,136 SVHN-Training INFO:
Epoch 6 - Evaluation time (seconds): 79.85 - val metrics:
    Accuracy: 0.9043484941610326
    Loss: 0.3281271758076598
2023-04-24 18:56:28,136 SVHN-Training INFO: Epoch[6] Complete. Time taken: 00:03:37.456
2023-04-24 18:58:19,855 SVHN-Training INFO: Epoch[7] Complete. Time taken: 00:01:51.719
2023-04-24 19:00:11,355 SVHN-Training INFO: Epoch[8] Complete. Time taken: 00:01:51.500
2023-04-24 19:02:29,191 SVHN-Training INFO:
Epoch 9 - Evaluation time (seconds): 26.08 - train metrics:
    Accuracy: 0.9801523383111742
    Loss: 0.07020394343663831
```

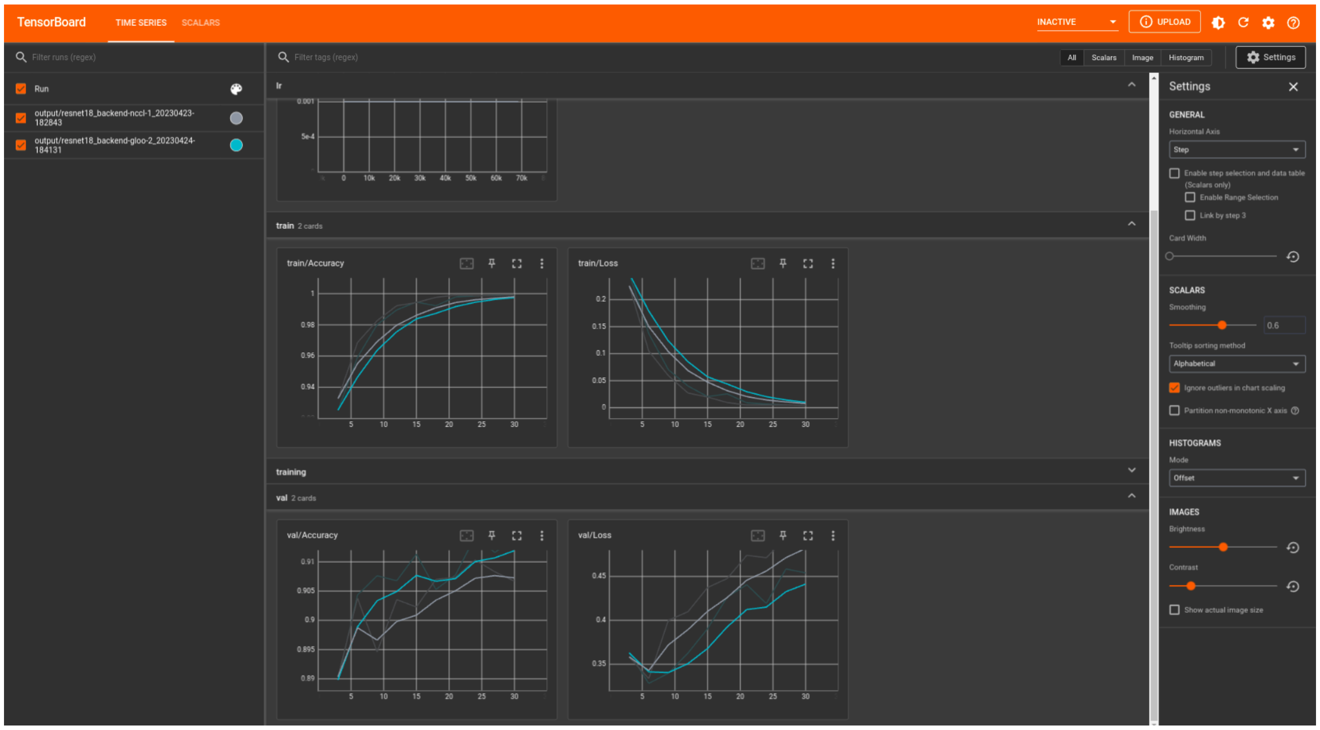
- Training started using 2 GPUs (actually CPUs because I am using “gloo” in this case instead of “nccl” because I was getting socket address issues in case of using “nccl”).
- For each epoch, we see the evaluation time, accuracy and loss for both the training and the validation data.



```
    Loss: 0.4406313623617087
2023-04-24 19:33:47,708 SVHN-Training INFO: Epoch[21] Complete. Time taken: 00:03:47.426
2023-04-24 19:35:42,941 SVHN-Training INFO: Epoch[22] Complete. Time taken: 00:01:55.233
2023-04-24 19:37:38,303 SVHN-Training INFO: Epoch[23] Complete. Time taken: 00:01:55.361
2023-04-24 19:40:01,204 SVHN-Training INFO:
Epoch 24 - Evaluation time (seconds): 27.44 - train metrics:
    Accuracy: 0.9984984575063475
    Loss: 0.006720086678086779
2023-04-24 19:41:26,938 SVHN-Training INFO:
Epoch 24 - Evaluation time (seconds): 85.66 - val metrics:
    Accuracy: 0.9142977873386601
    Loss: 0.4191586917063614
2023-04-24 19:41:26,938 SVHN-Training INFO: Epoch[24] Complete. Time taken: 00:03:48.635
2023-04-24 19:43:22,386 SVHN-Training INFO: Epoch[25] Complete. Time taken: 00:01:55.448
2023-04-24 19:45:17,842 SVHN-Training INFO: Epoch[26] Complete. Time taken: 00:01:55.456
2023-04-24 19:47:41,224 SVHN-Training INFO:
Epoch 27 - Evaluation time (seconds): 27.25 - train metrics:
    Accuracy: 0.9989762210270551
    Loss: 0.005150010647034 71
2023-04-24 19:49:07,015 SVHN-Training INFO:
Epoch 27 - Evaluation time (seconds): 85.72 - val metrics:
    Accuracy: 0.9116087891825445
    Loss: 0.45853083164470654
2023-04-24 19:49:07,015 SVHN-Training INFO: Epoch[27] Complete. Time taken: 00:03:49.172
2023-04-24 19:51:02,037 SVHN-Training INFO: Epoch[28] Complete. Time taken: 00:01:55.022
2023-04-24 19:52:57,222 SVHN-Training INFO: Epoch[29] Complete. Time taken: 00:01:55.184
2023-04-24 19:55:19,596 SVHN-Training INFO:
Epoch 30 - Evaluation time (seconds): 27.17 - train metrics:
    Accuracy: 0.9991536760490322
    Loss: 0.004051202039154 4265
2023-04-24 19:56:45,331 SVHN-Training INFO:
Epoch 30 - Evaluation time (seconds): 85.65 - val metrics:
    Accuracy: 0.9137215734480639
    Loss: 0.4540605222658651
2023-04-24 19:56:45,332 SVHN-Training INFO: Epoch[30] Complete. Time taken: 00:03:48.110
2023-04-24 19:57:12,647 SVHN-Training INFO:
Epoch 30 - Evaluation time (seconds): 27.23 - train metrics:
    Accuracy: 0.9991536760490322
    Loss: 0.004051202039154 4265
2023-04-24 19:58:38,399 SVHN-Training INFO:
Epoch 30 - Evaluation time (seconds): 85.68 - val metrics:
    Accuracy: 0.9137215734480639
    Loss: 0.4540605222658651
2023-04-24 19:58:38,400 SVHN-Training INFO: Engine run complete. Time taken: 00:28:10.371
2023-04-24 19:58:39,050 ignite.distributed.launcher.Parallel INFO: End of run
Training completed for 1 GPU with 2 processes
```

- Training was done till 30 epochs and final results were:
    - Training acc = 99.9153 %
    - Training loss = 0.00405
    - Validation acc = 91.372%
    - Validation loss = 0.4540
    - Total training time = 28 minutes 10 seconds.

- Along with this, a tensorboard file was also generated (events.out file for both the models trained):
- Results are as follows:



- Observations in terms of memory-usage in multi-node training:
    - In multi-node training, the memory usage is significantly higher compared to single-node training. This is because in multi-node training, each node needs to maintain a copy of the model, optimizer, gradients, and other variables in its own memory, which can quickly add up as the number of nodes and batch size increases.
    - Using DataParallelism on multiple GPUs within a single node led to increased memory usage, as each GPU needs to store a copy of the model and gradients. However, the memory usage is less than that of multi-node training, as the data does not need to be sent over the network.
    - When using DistributedDataParallelism, the memory usage is urther reduced compared to DataParallelism, as each node only needs to store a

portion of the model and gradients. However, communication overhead between nodes may increase, as each node needs to communicate with the others to update its model.