Homework 1 **Student Name:** Tanmay Samak

AuE 8930: Computing and Simulation for Autonomy
Instructor: Prof. Bing Li, Clemson University, Department of Automotive Engineering

* Refer to Syllabus for homework (late) submission, grading and plagiarism policies;
* Submission due Mon. 9/11/2023 11:59 pm via Canvas, include:
  - This document (with answers), and with your program results/visualization;
  - A .zip file of (modified) source code and data if any, which the TA might run.

Question 1

Training a Pytorch deep learning model on Palmetto cluster (60 points)
(Recommended to use Jupyter Notebook in Palmetto OpenOnDemand for edit/debug/run)

Palmetto Cluster and Setup

- Login into your Palmetto account & request a node with required specifications by specifying a hardware resource configuration, making sure to include GPU. (For below all questions, make sure to use same configuration).
- Transfer the sample code into your account using Globus (if using Terminal) or JupterHub.

Create a Conda virtual environment in the terminal

module add anaconda3/2022.05-gcc/9.5.0

A conda virtual environment allows you to run/install a version of Python and package as needed within it.

This environment, once created/modified is saved and can be accessed later through the code:

conda create -n NAME_OF_ENV python=3.6 # (Create Environment)
source activate NAME_OF_ENV # (Activate Environment)
source deactivate NAME_OF_ENV # (Deactivate Environment)

Install necessary packages in the terminal

Add cuda and cudnn module:

module add cuda/11.1.1-gcc/9.5.0
module add cudnn/8.0.5.39-11.1-gcc/9.5.0-cu11_1

Install Pytorch and Torchvision libraries using conda (reference)

conda install pytorch torchvision torchaudio cudatoolkit=11.1 -c pytorch-lts -c nvidia

Generate Kernel for JupyterHub

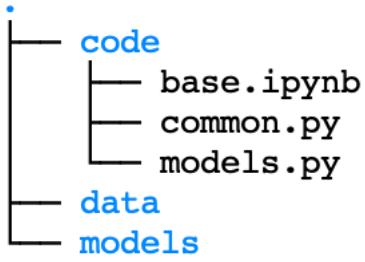
(Attention: if you install those modules under a certain conda environment)
You may encounter this error when running the base.ipynb in Jupyter Hub:

"no module named torch"

It means your Jupyter notebook is running in the default python environment, but your torch module is installed in your Conda virtual environment. You will need to run Jupyter notebook in your virtual env.
Here is a tutorial: https://janakiev.com/blog/jupyter-virtual-envs/

Training deep learning model for Image Classification
Sample code is in Canvas/Files can be downloaded from: Homework_1_sample_code.zip
which includes: base.ipynb, common.py and models.py. The base.ipynb allows you to use your web browser as the GUI to run/edit/debug.

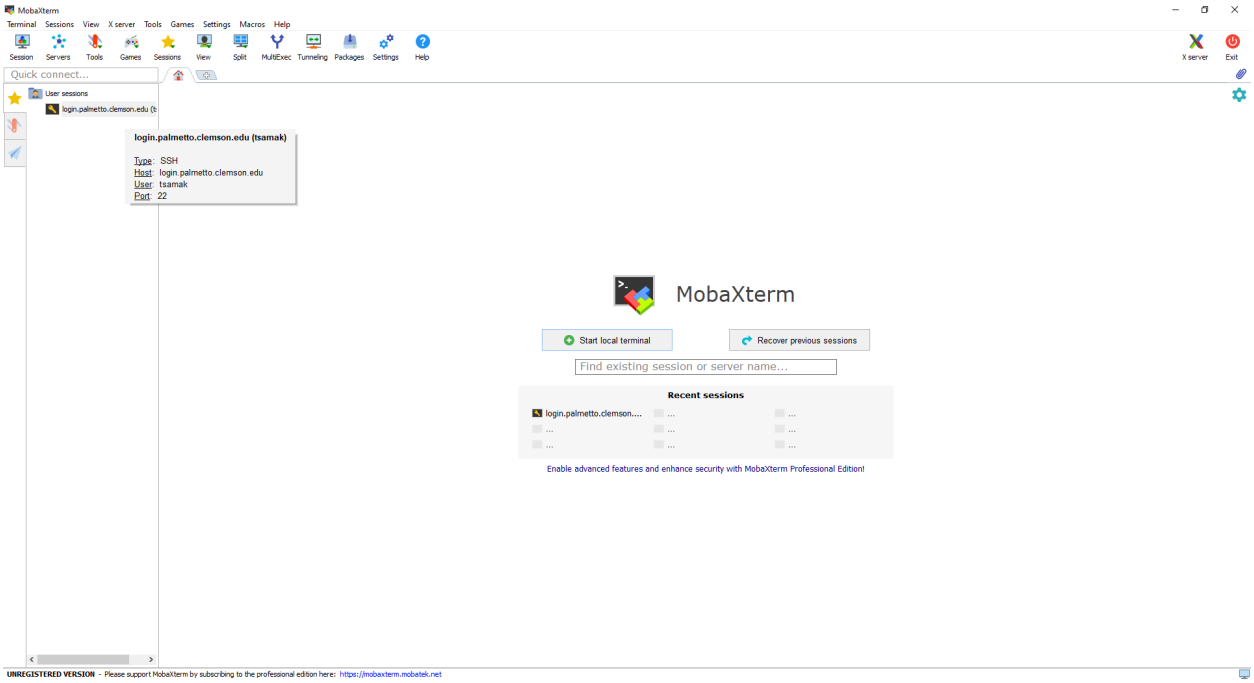
You also need to make 'data' and 'models' folder before running the 'base.ipynb'. The directory structure should look like:



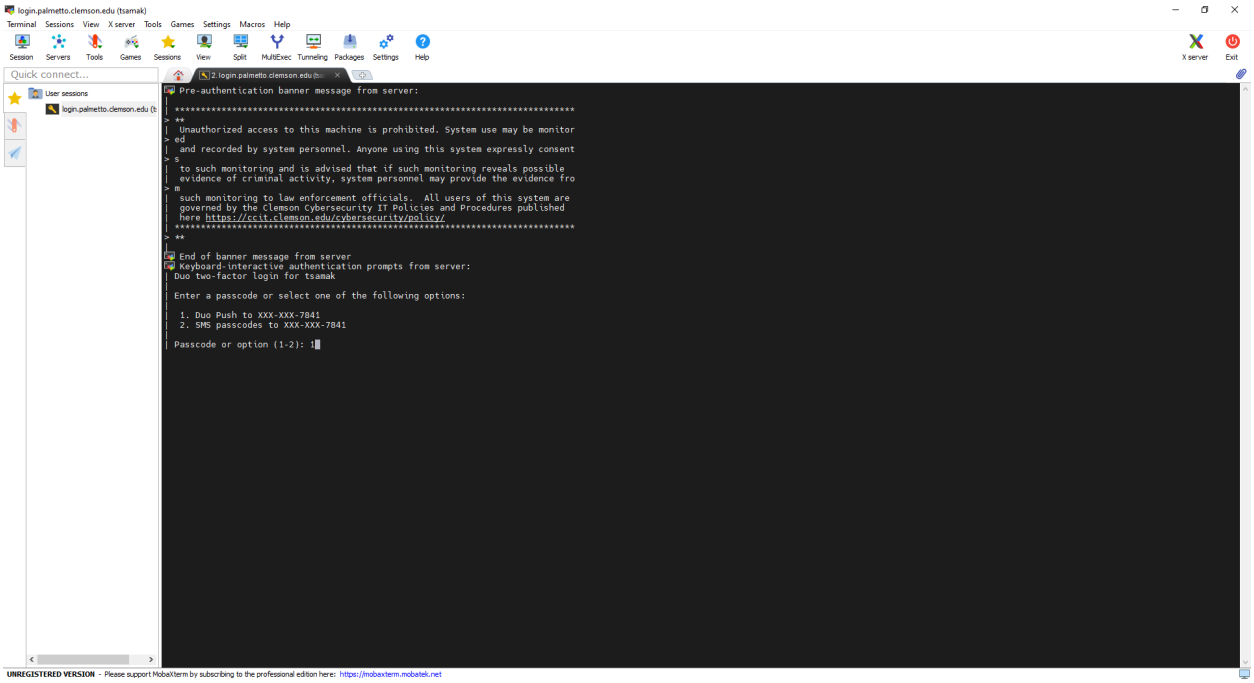
There are multiple steps in the sample code files:

- Load the training and test datasets from torchvision (reference) Training Data can be obtained from various online sources, self-procured or can even be imported from a library like Pytorch.
- Define a Convolutional Neural Network (reference)
- Define a loss function (reference)
- Train the network on the training data with different number of Epochs (reference).

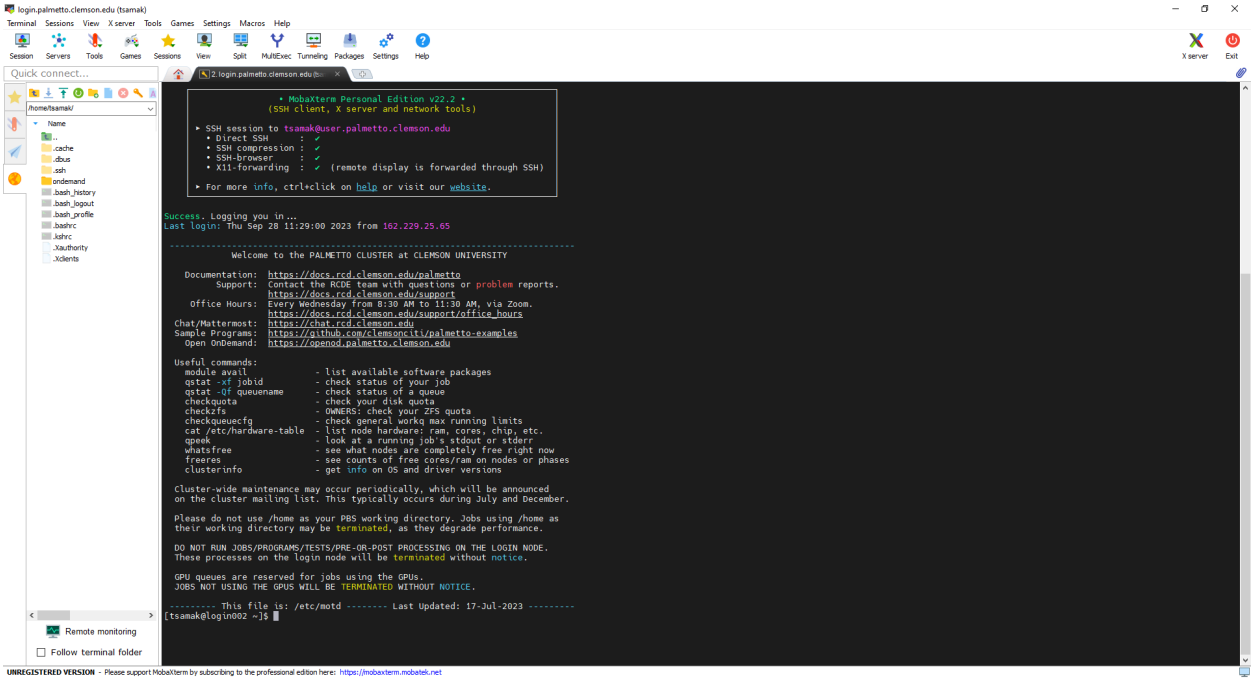
(1) Show screenshots of successful installation and procedure of the setup. (15 points)



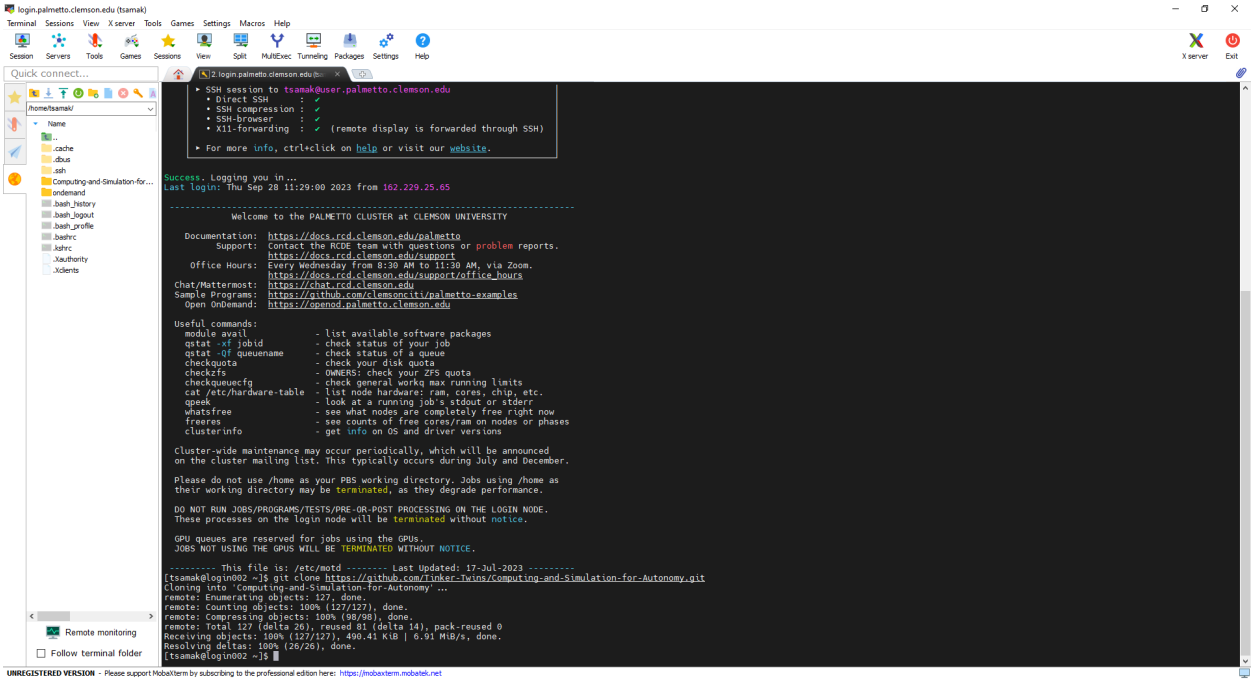
1.1 Open MobaXterm application (Windows) and SSH into Palmetto Cluster.



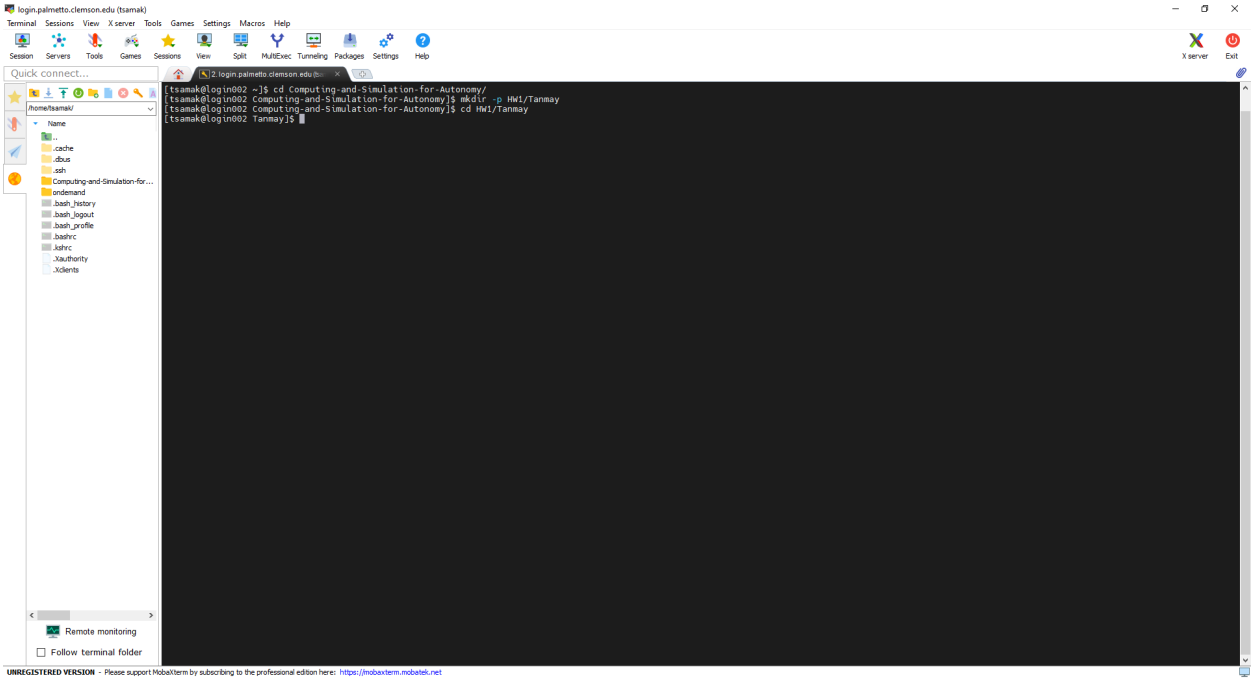
1.2 When prompted, enter the mode of authentication (Duo Push/SMS).



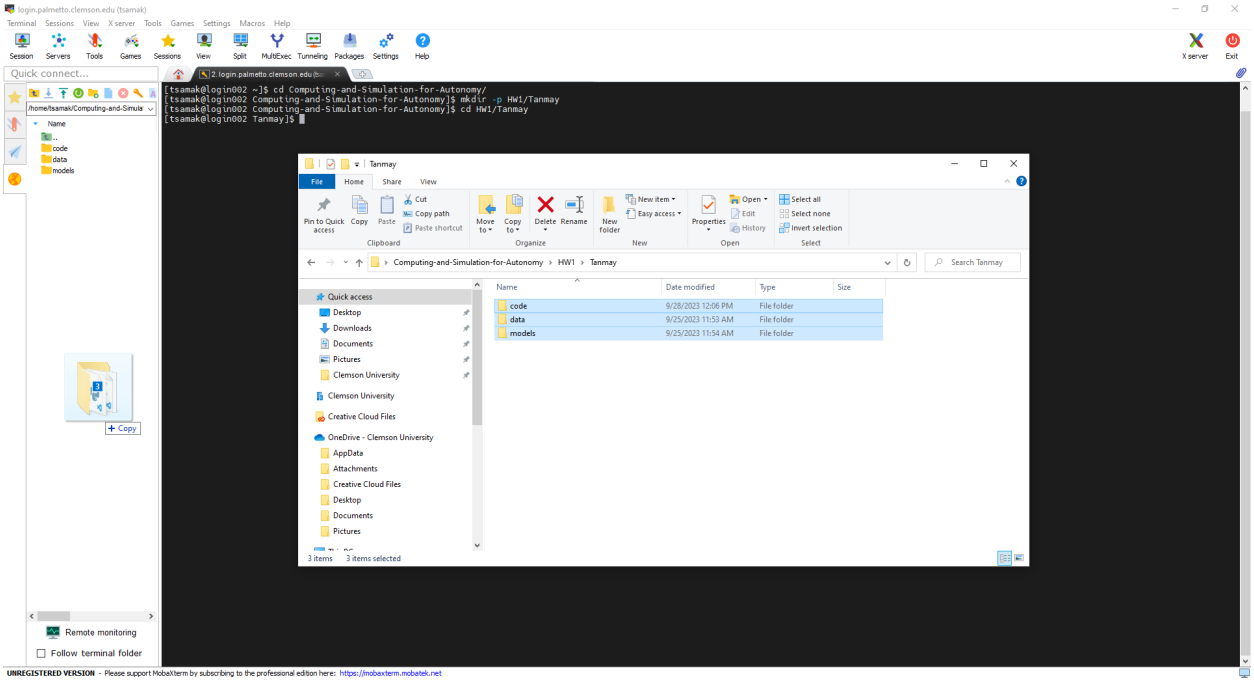
1.3 Login with your credentials (e.g., verify the Duo Push).



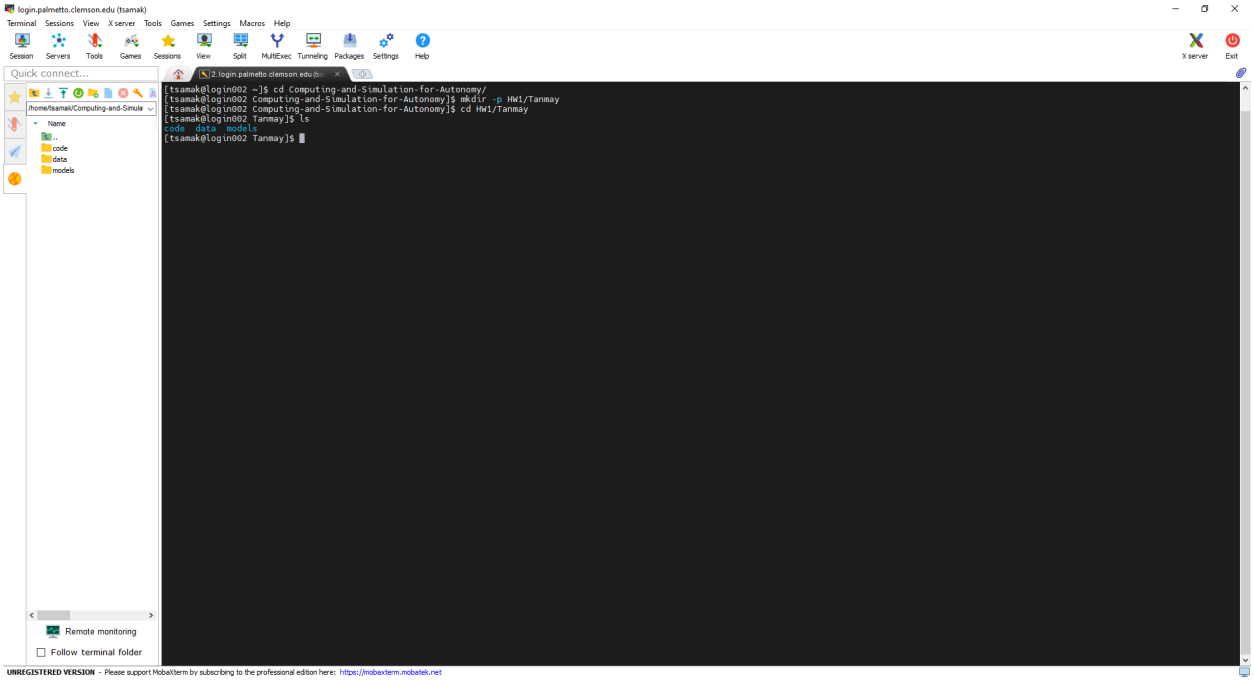
1.4 Clone the GitHub Repository of course to the ~HOME directory on Palmetto Cluster.



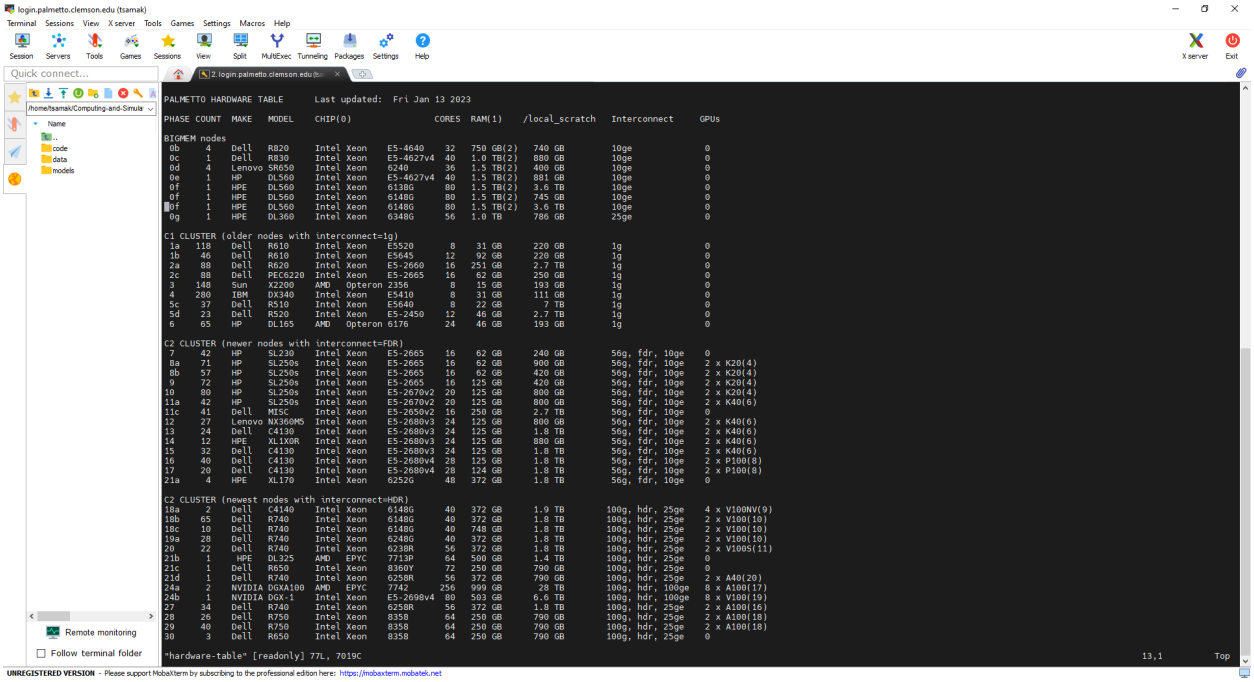
1.5 Create a new directory for HW1 and change directory to that location.



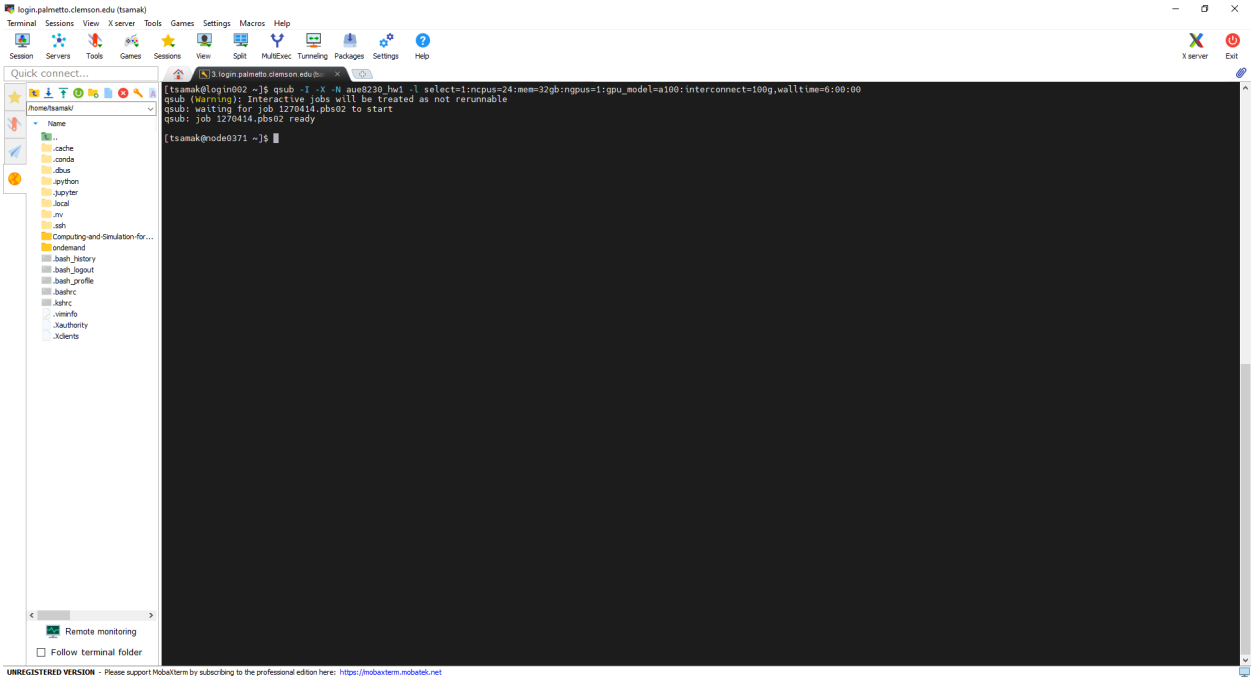
1.6 Move (drag-and-drop) new files for HW1 to `HW1/Tanmay` directory on Palmetto Cluster.



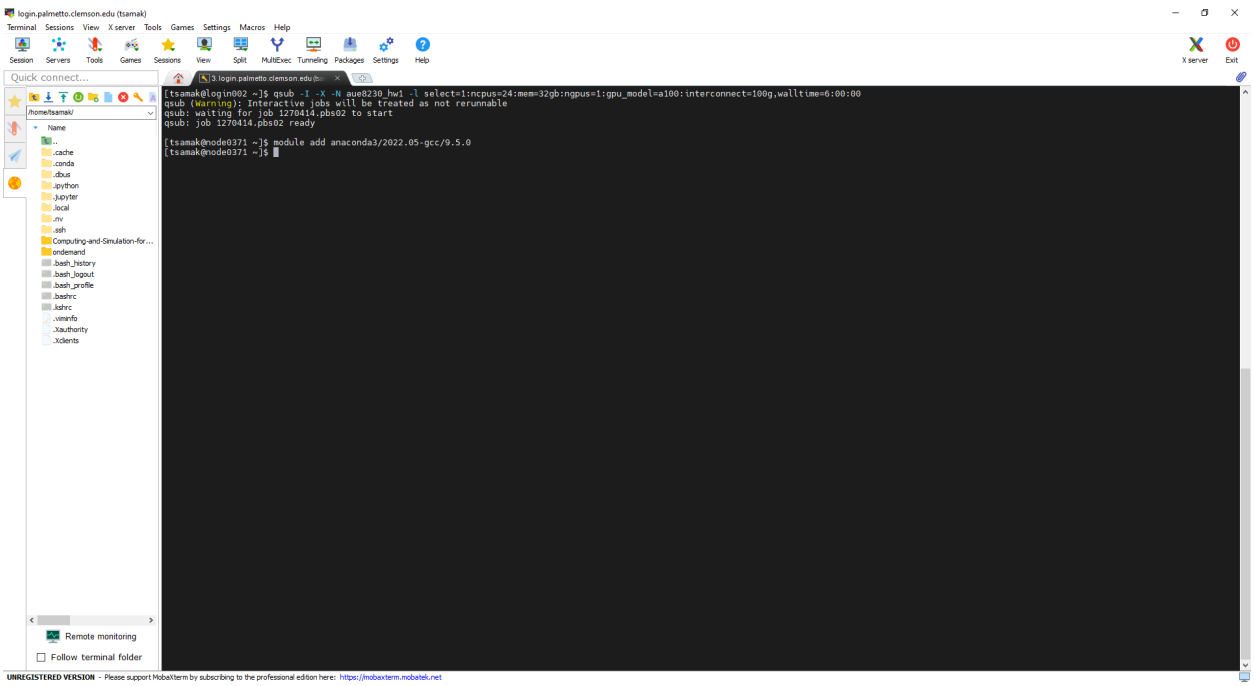
1.7 Confirm uploaded files and folders via terminal.



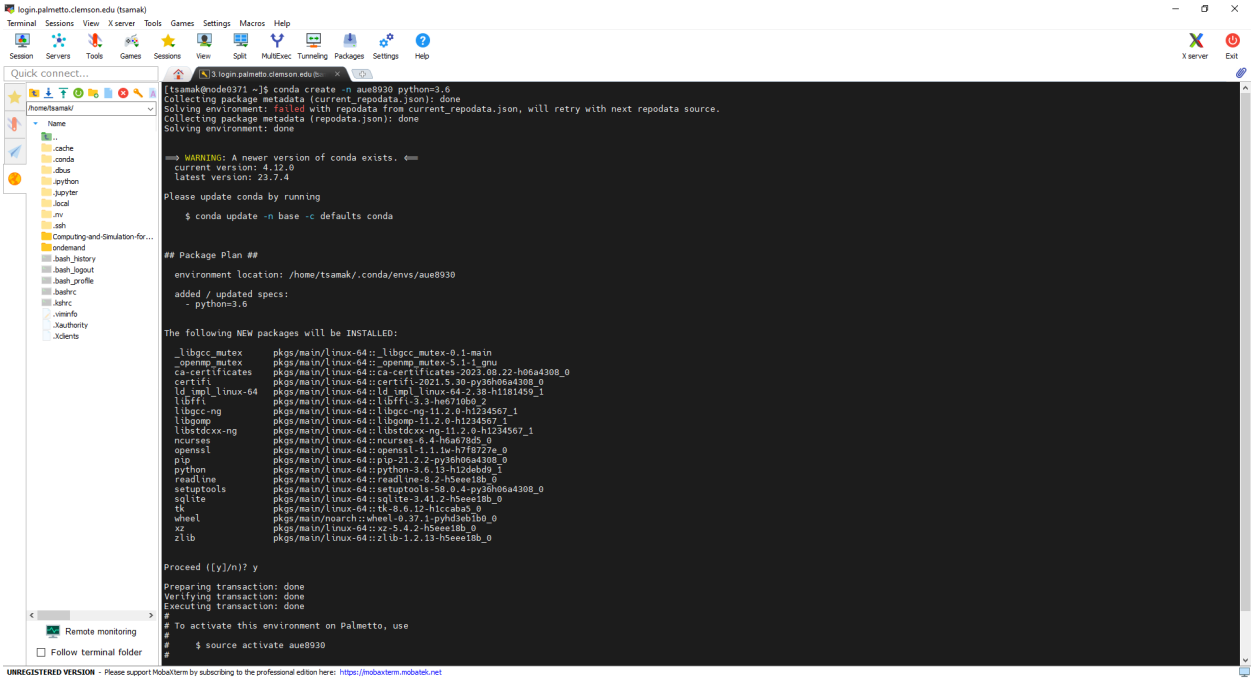
1.8 Refer `/etc/hardware-table` for hardware specification of the available compute resources.



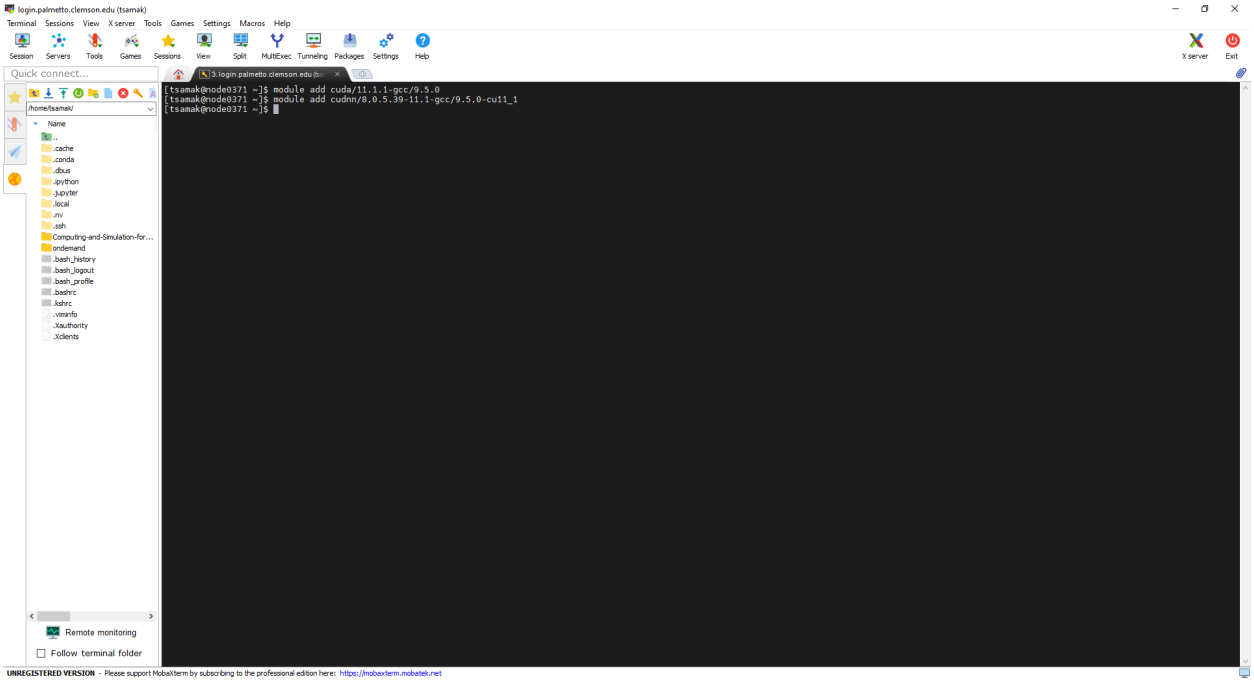
1.9 Submit a new interactive job called `aue8930_hw1` with X11 port forwarding, which has the given hardware specifications.



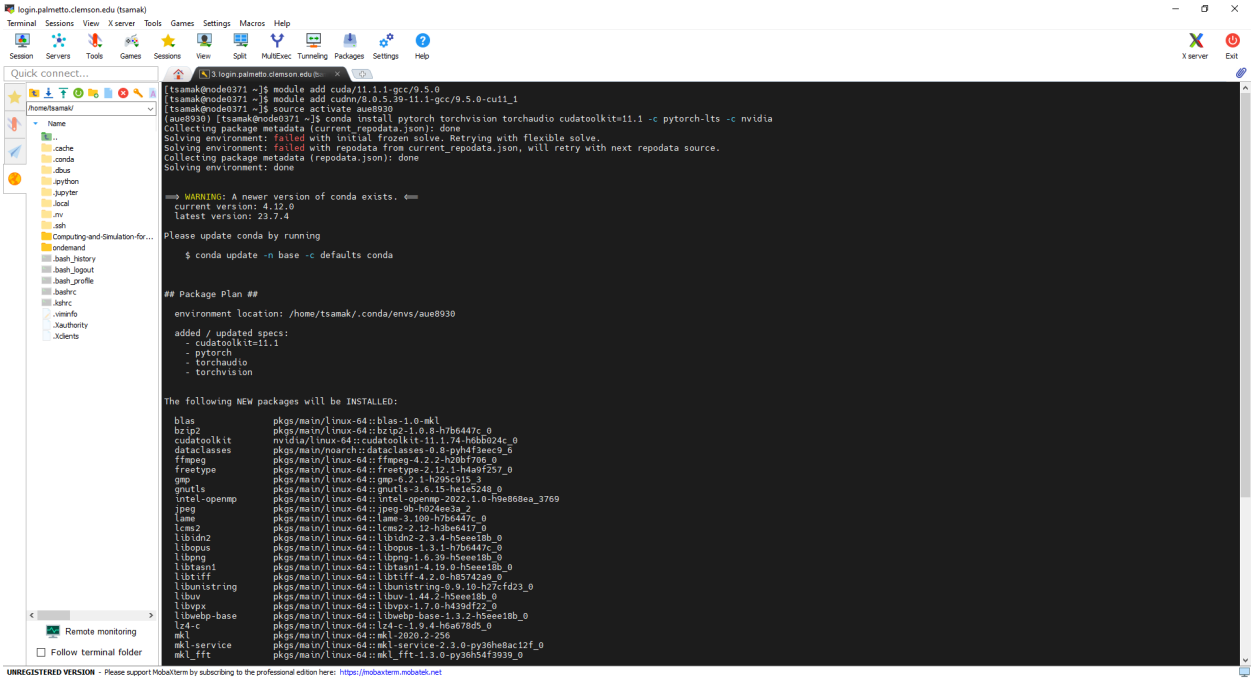
1.10 Add Anaconda module.



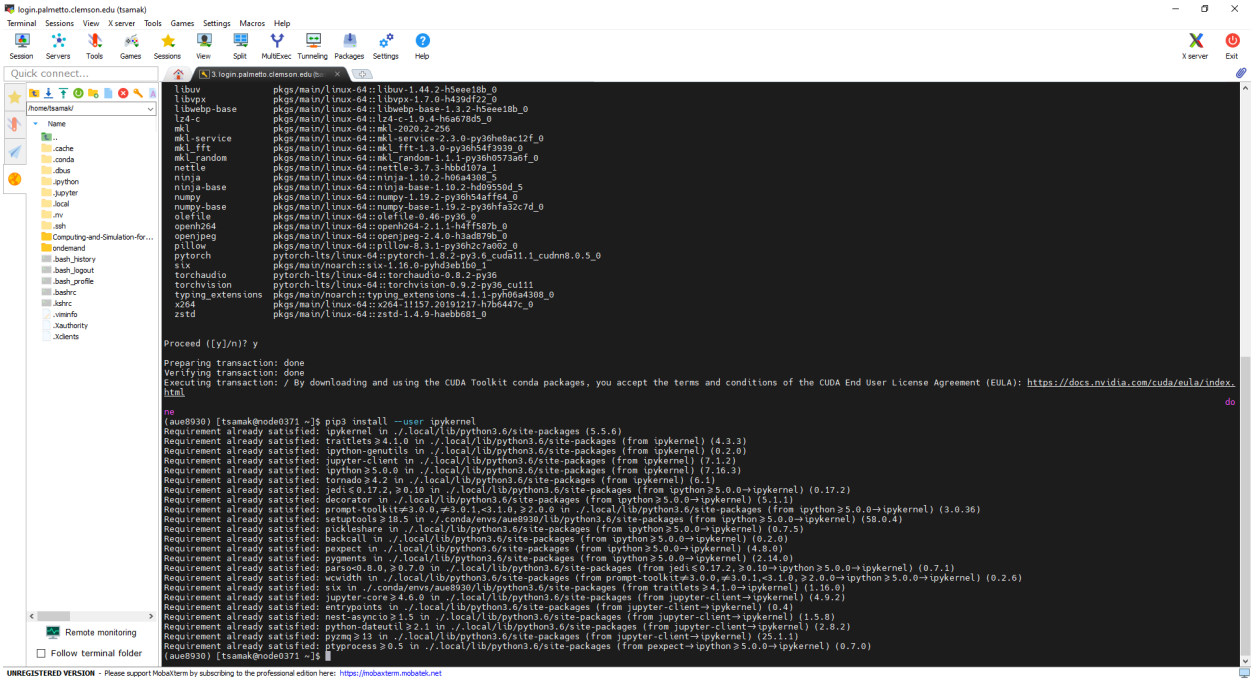
1.11 Create Conda virtual environment called `aue8930`.



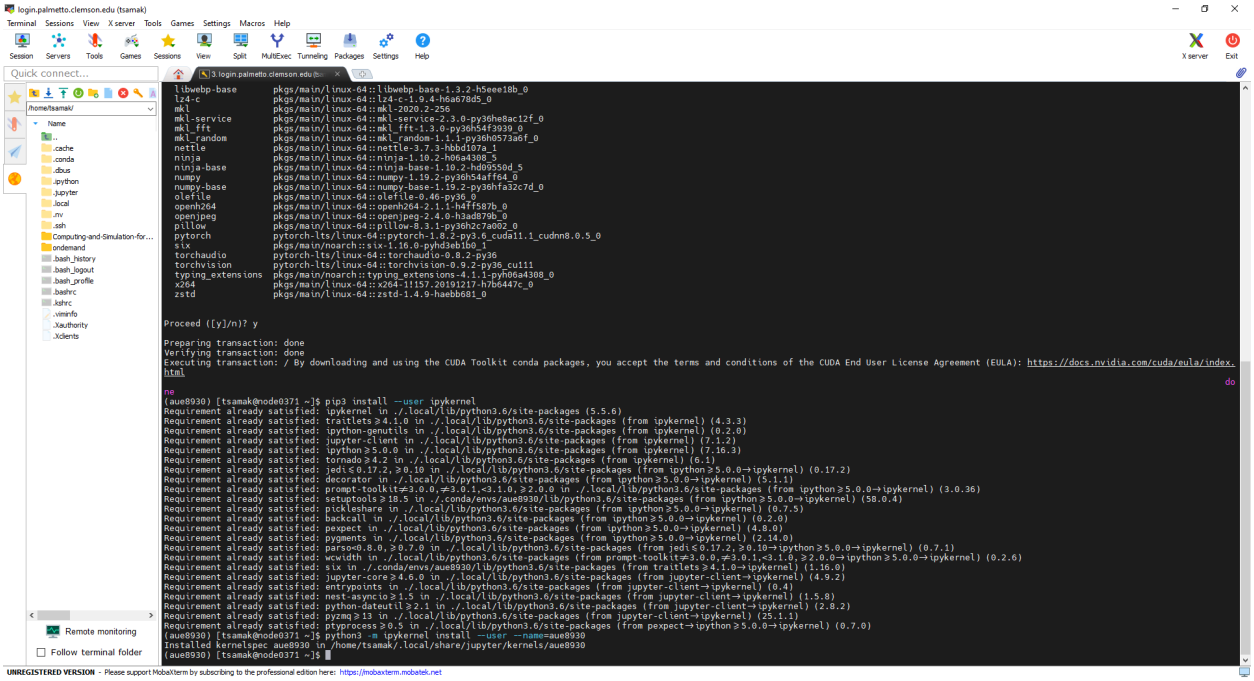
1.12 Add CUDA and CUDNN modules.



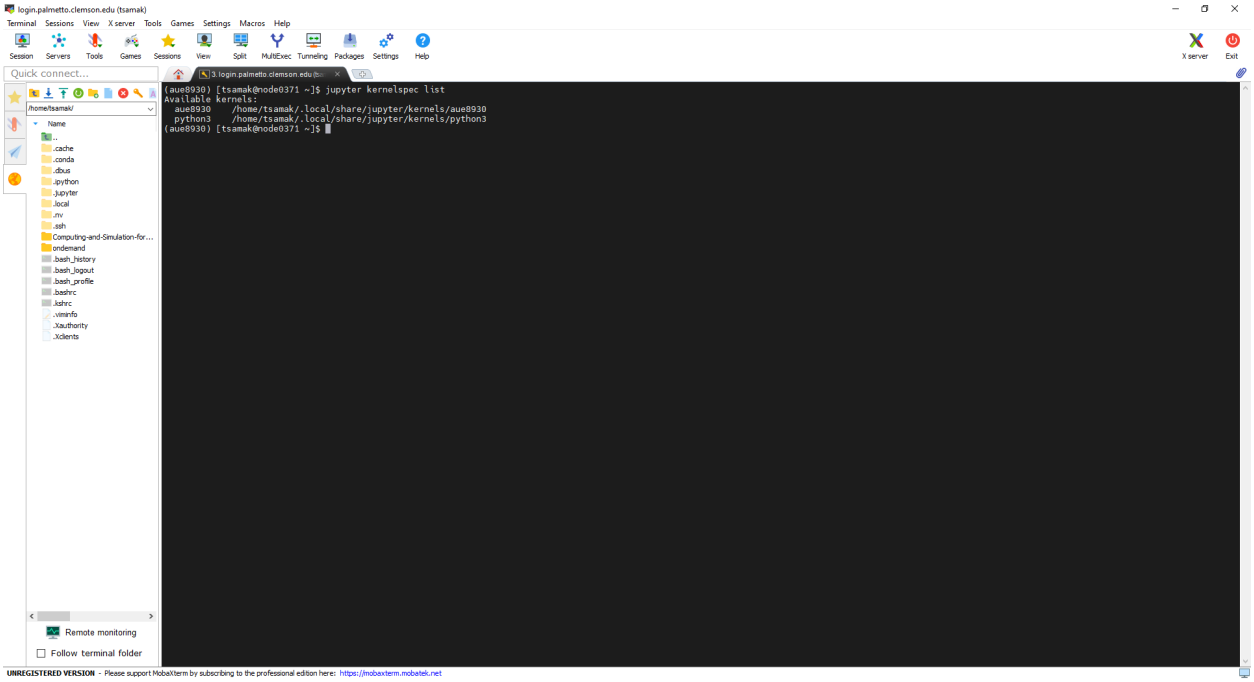
1.13 Activate Conda virtual environment and install `pytorch`.



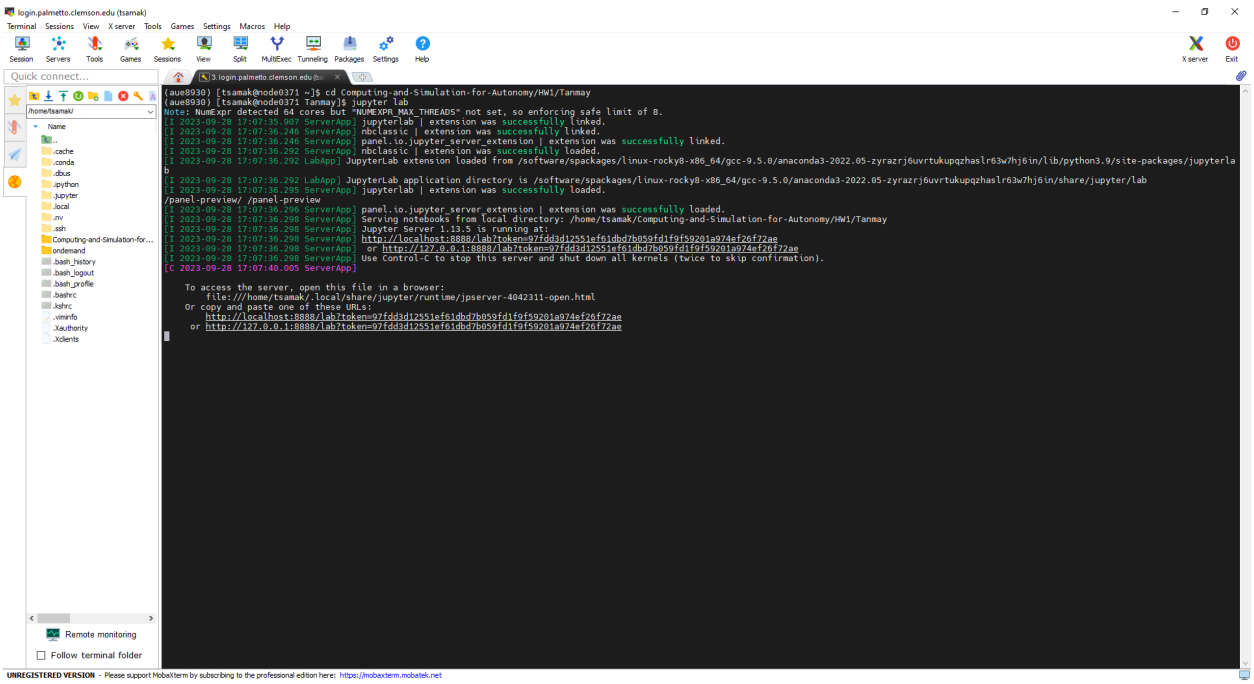
1.14 Install `ipykernel` (required to add the virtual environment to Jupyter).



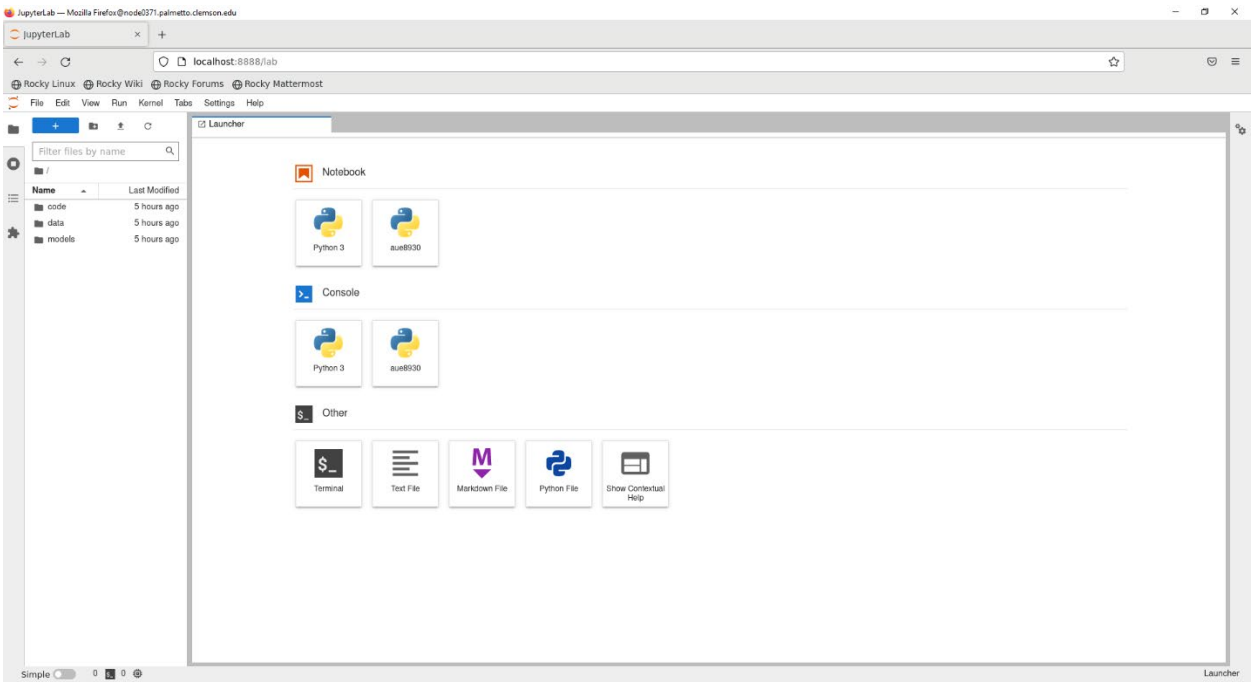
1.15 Add Conda virtual environment to Jupyter.



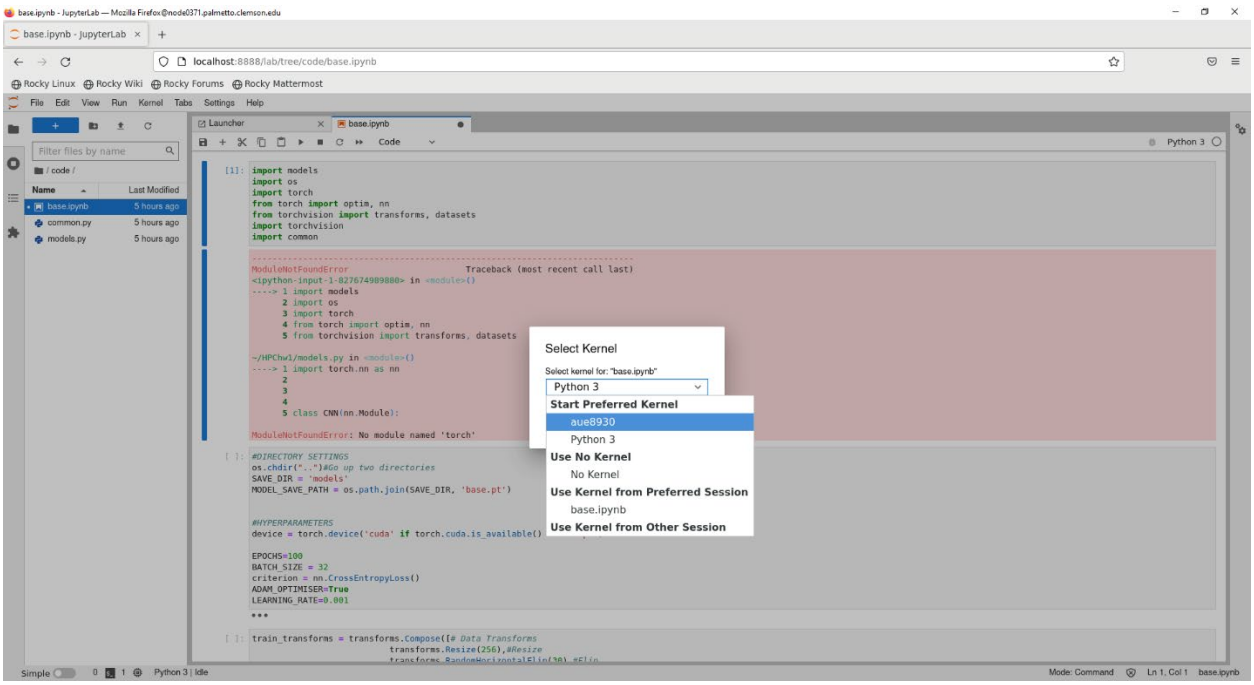
1.16 Verify Jupyter kernels.



1.17 Change directory to `HW1/Tanmay` and launch JupyterLab.



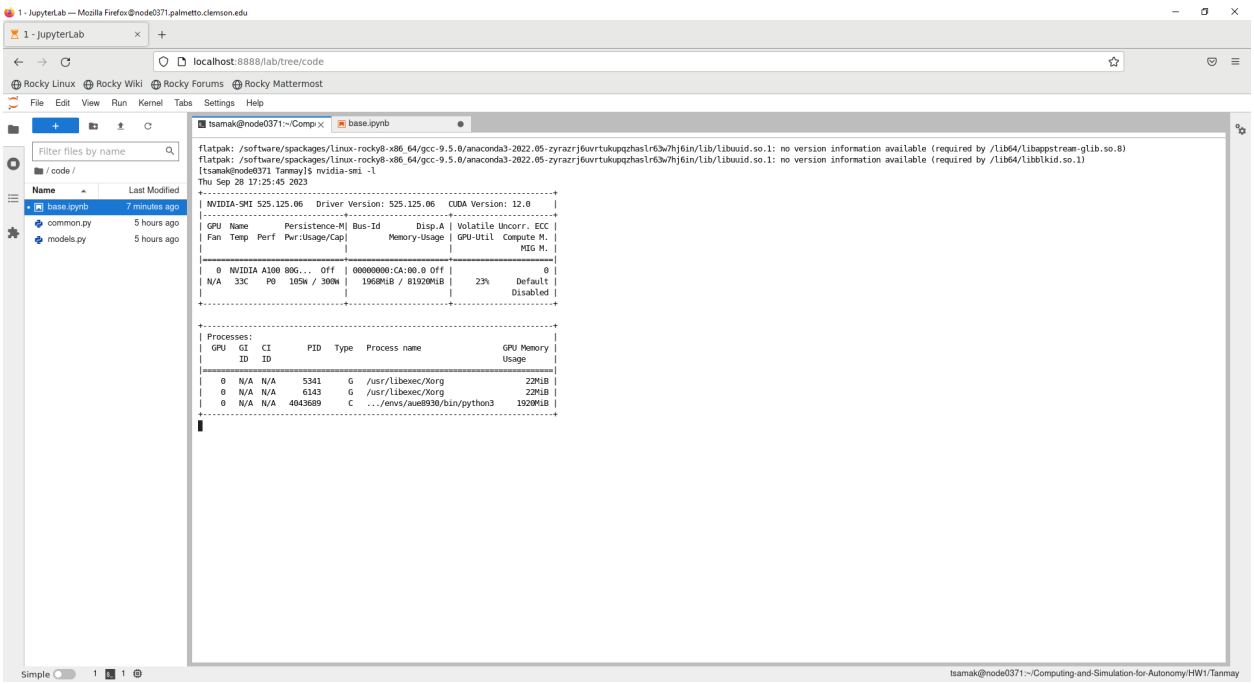
1.18 An interactive browser (e.g., Firefox) window should pop-up (from the Palmetto Node).



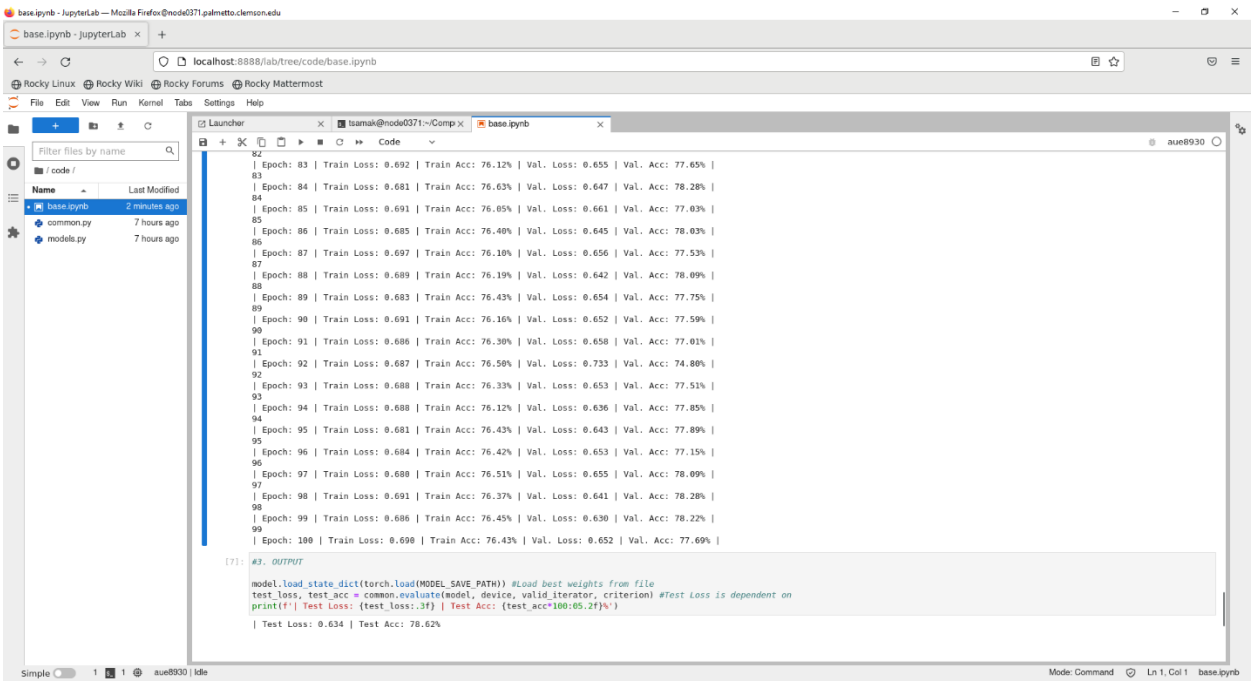
1.19 Open the IPYNB and change the kernel.

(2) Run the existing sample code “base.ipynb” (5 points)

During the training, what’s your GPU usage percentage? (You can open another terminal and use “nvidia-smi –l” to monitor the usage info of GPU and GPU memory.)

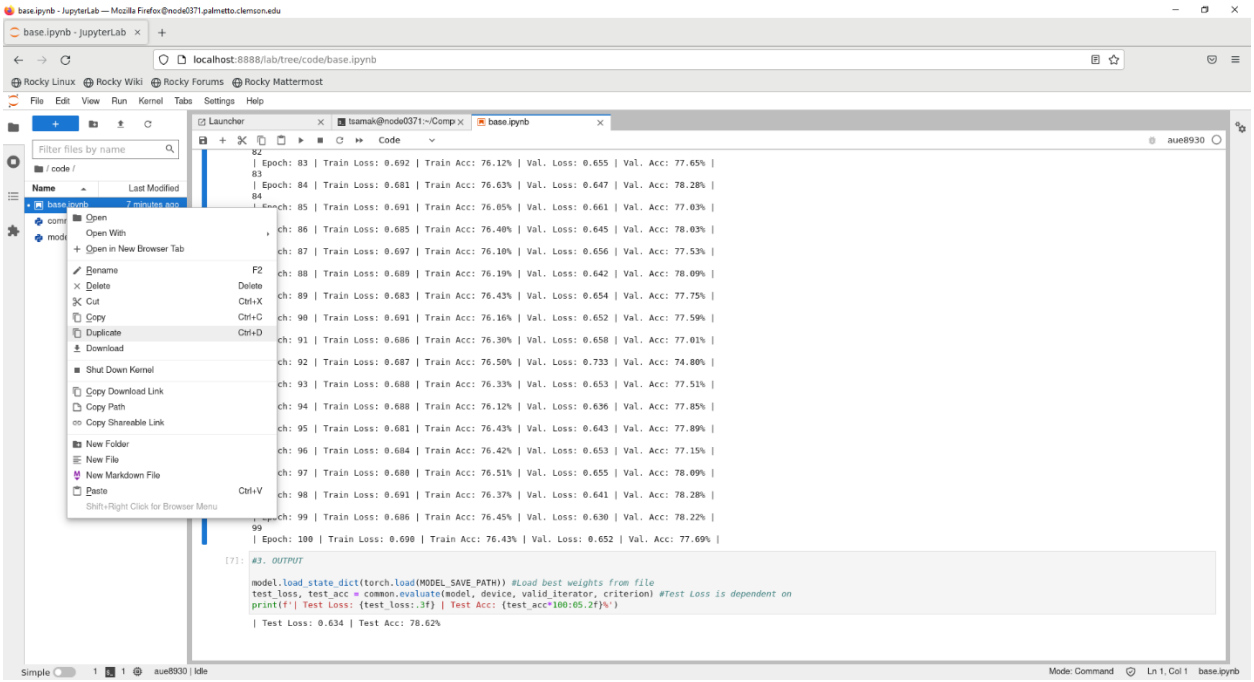


2.1 A100 GPU usage (~23%) and GPU RAM usage (~2.4%) for baseline implementation.

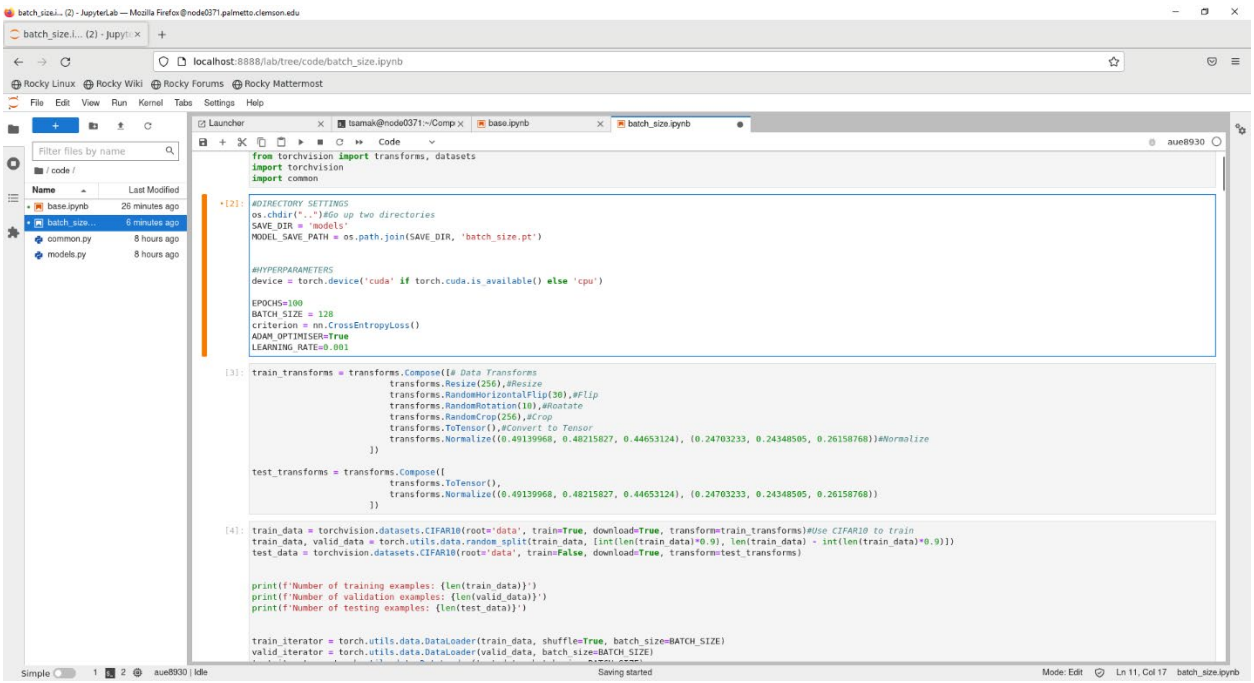


2.2 Accuracy (train=76.43%; val=77.69%; test=78.62%) for baseline implementation.

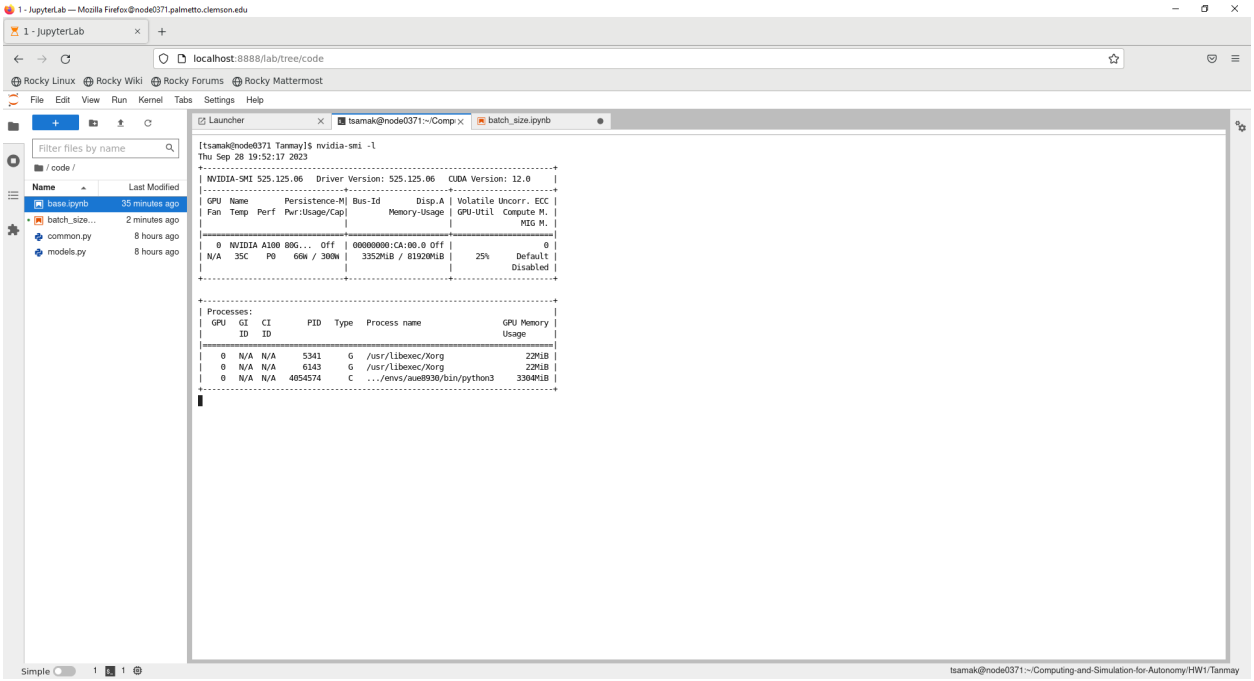
(3) Modify the code for better performance (change the batch size) (10 points)
During the training, what's your GPU usage percentage?



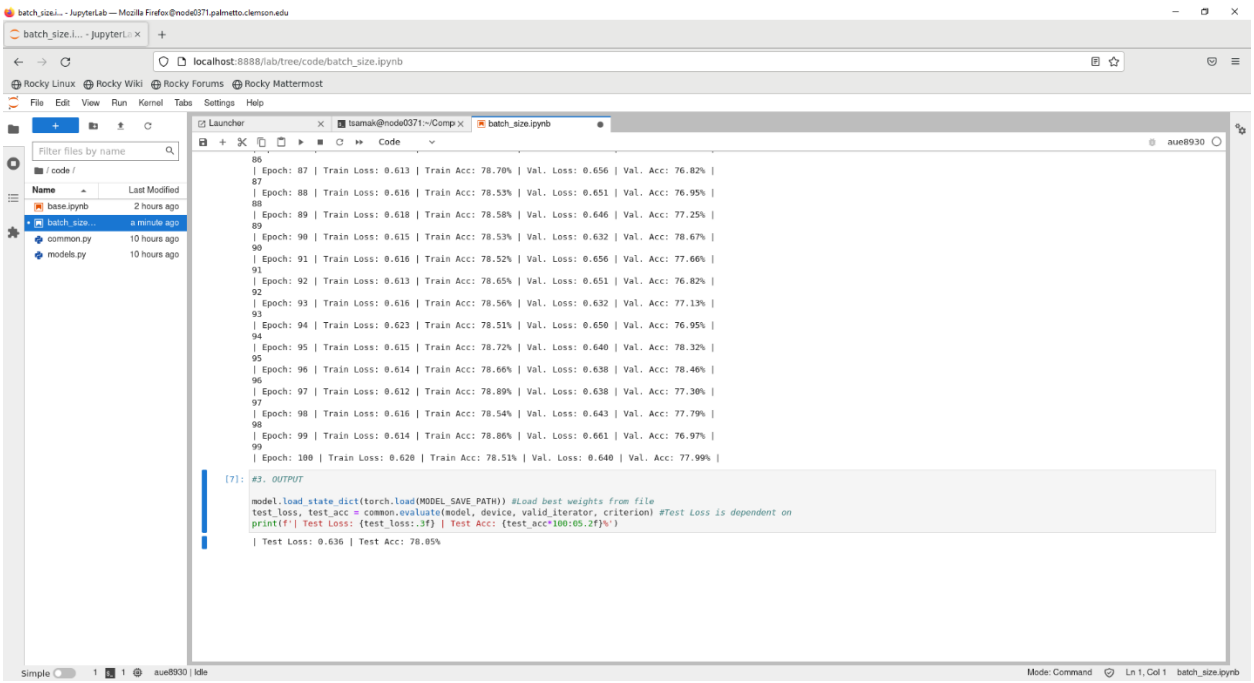
3.1 Duplicate the baseline IPYNB for batch-size experiment.



3.2 Change batch size and model name for saving.

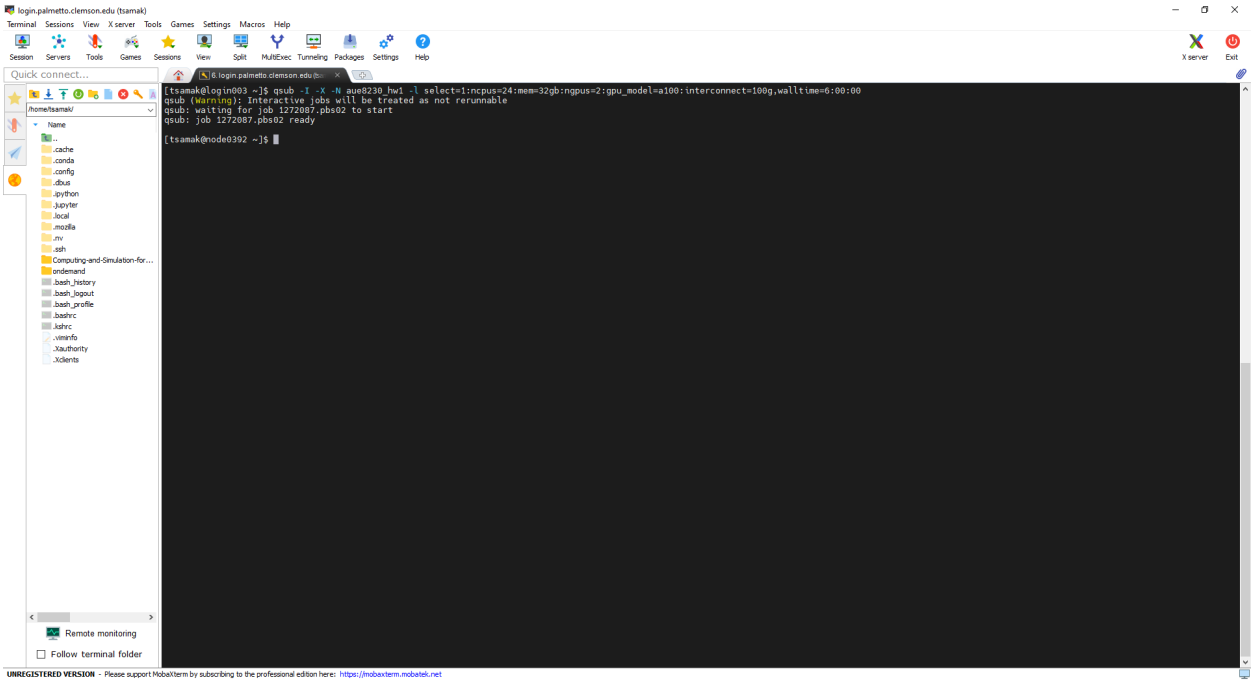


3.3 A100 GPU usage (~25%) and GPU RAM usage (~4.1%) for implementation with increased batch size.

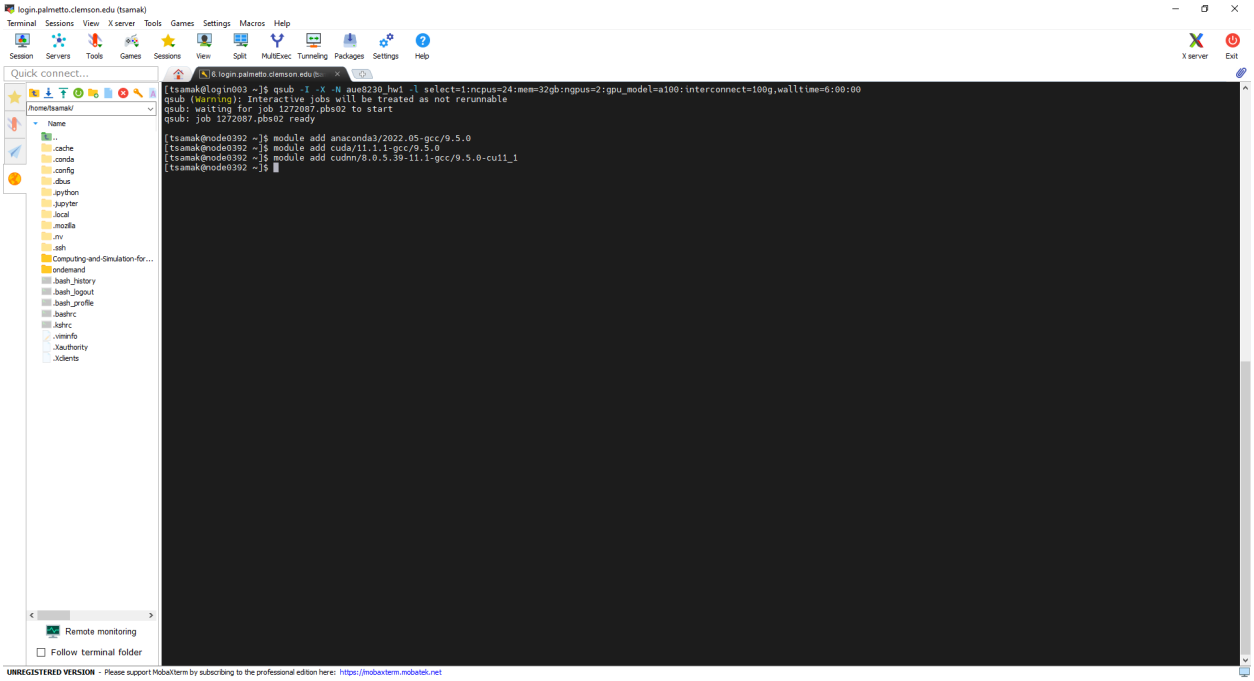


3.4 Accuracy (train=78.51%; val=77.99%; test=78.05%) with increased batch size.

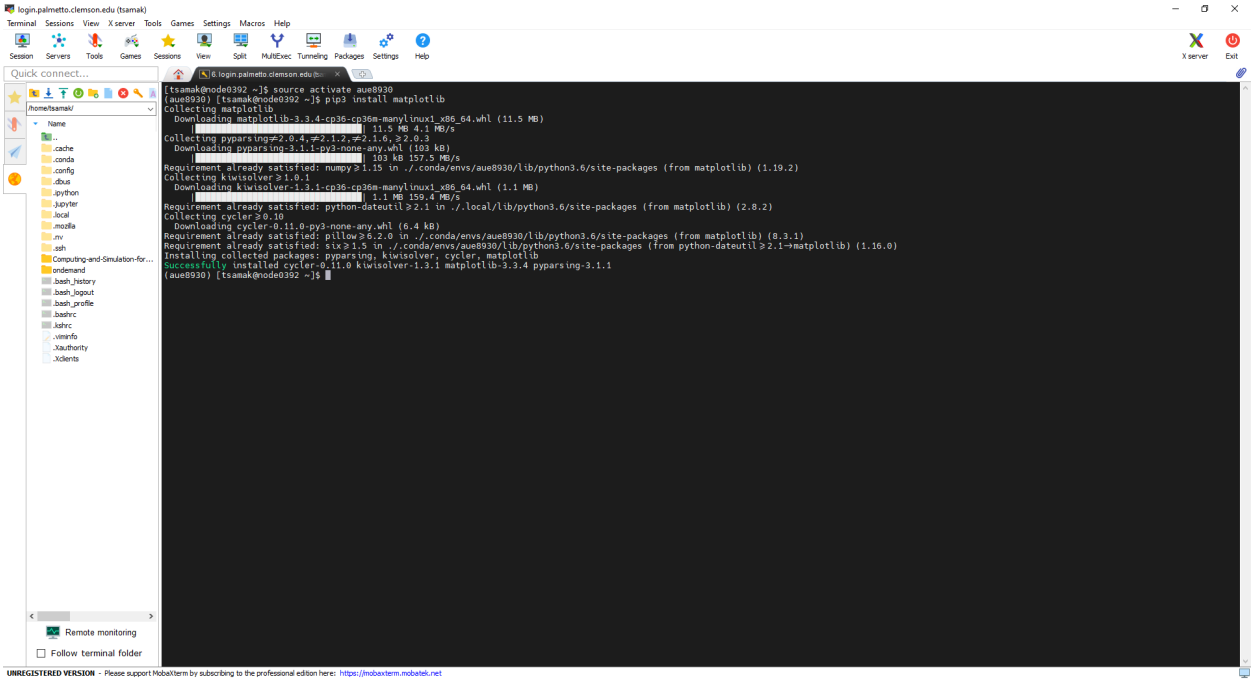
(4) Modify the code for better performance (use two GPUs) (10 points)
During the training, what's your GPU info percentage? (TIPS: reference API)



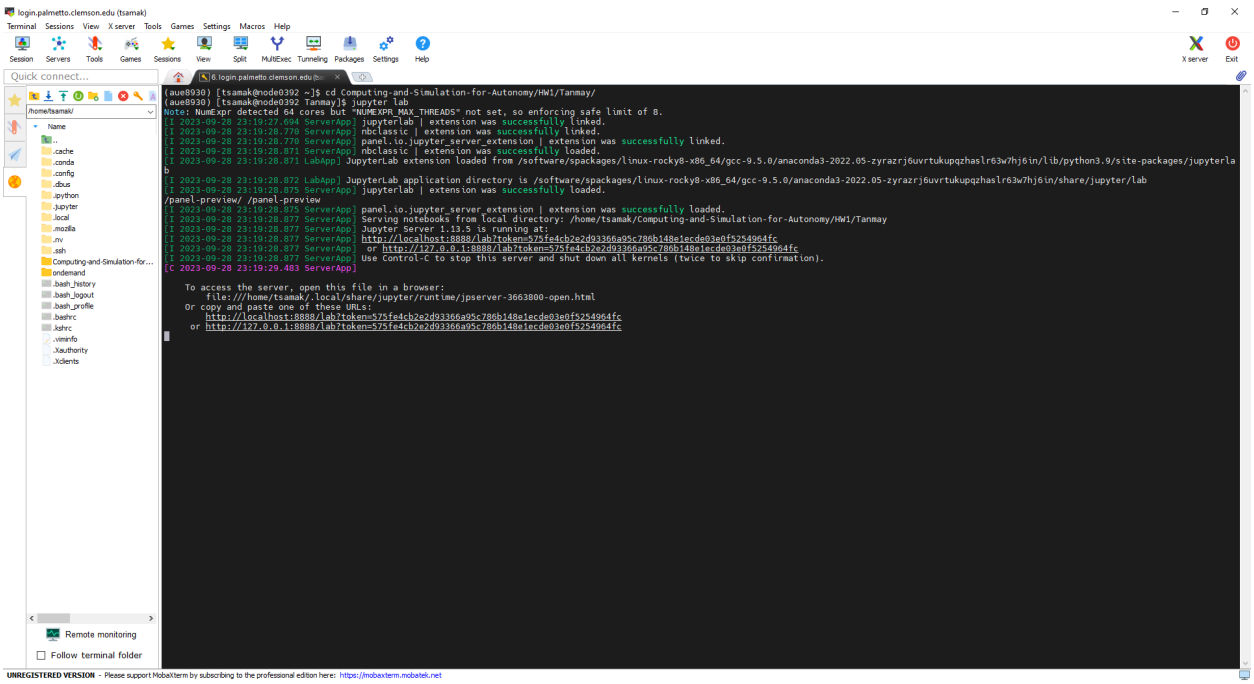
4.1 Submit a dual-GPU interactive job with X11 port forwarding.



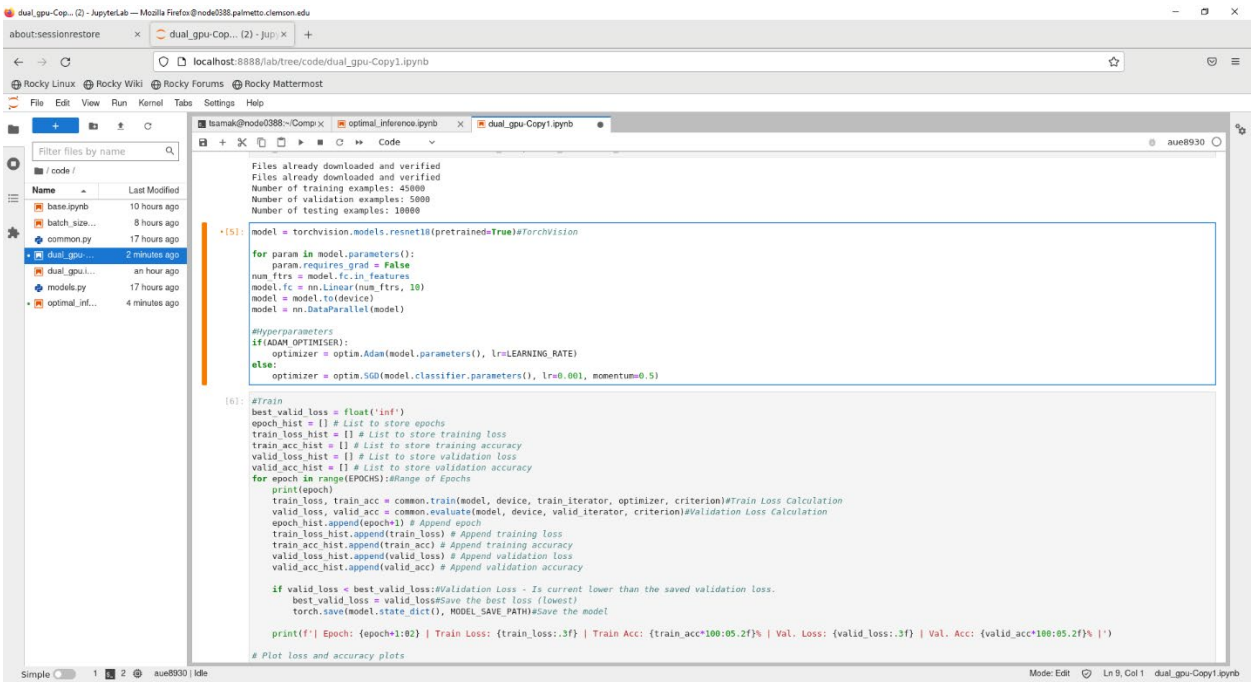
4.2 Add Anaconda, CUDA and CUDNN modules.



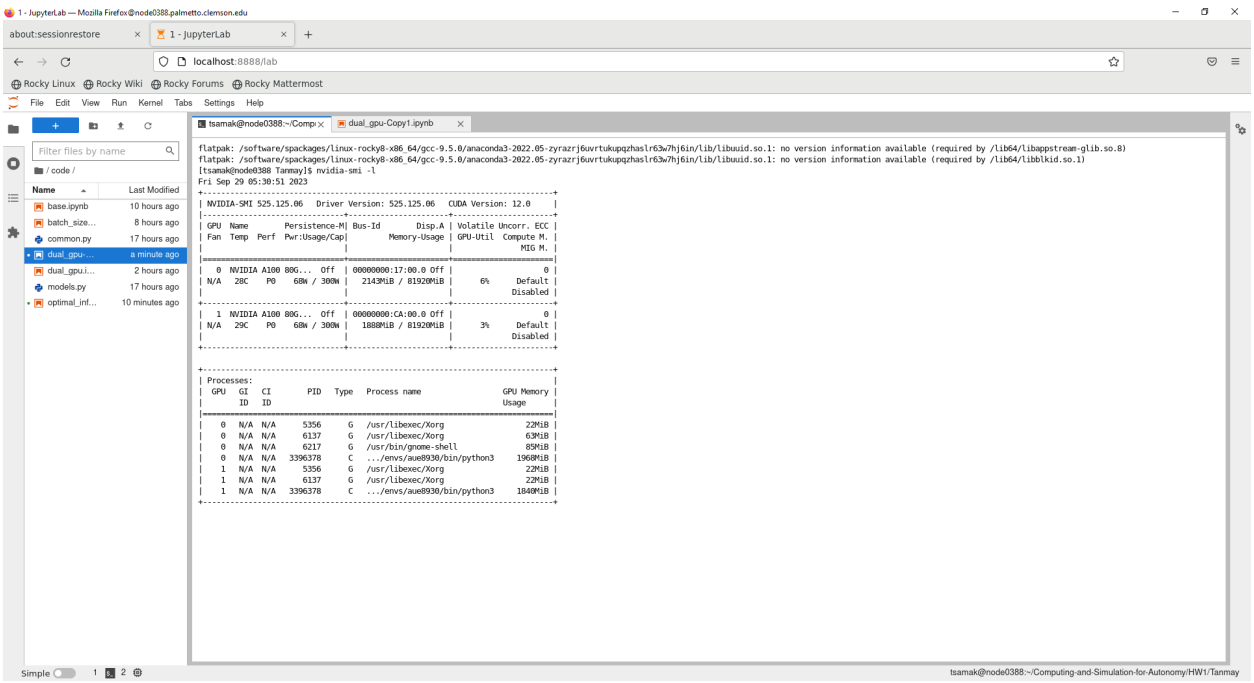
4.3 Activate Conda virtual environment and install `matplotlib`



4.4 Change directory to `HW1/Tanmay` and launch JupyterLab.

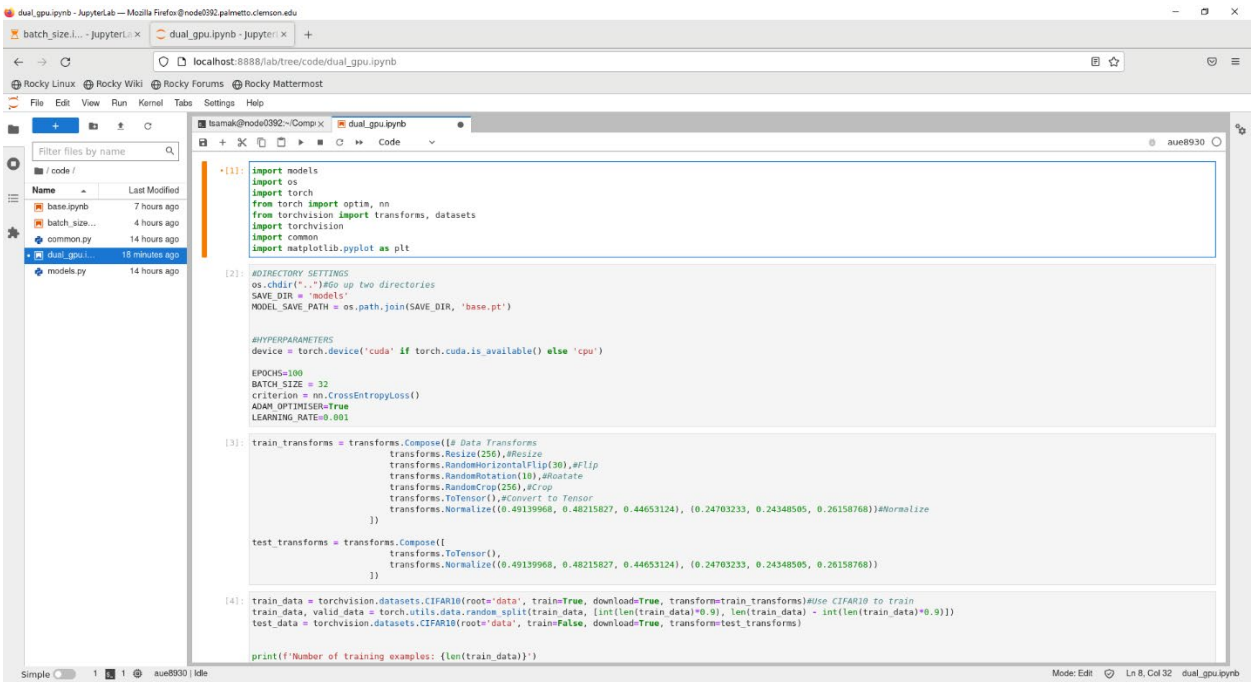


4.5 Modify the model section to allow data parallelism.

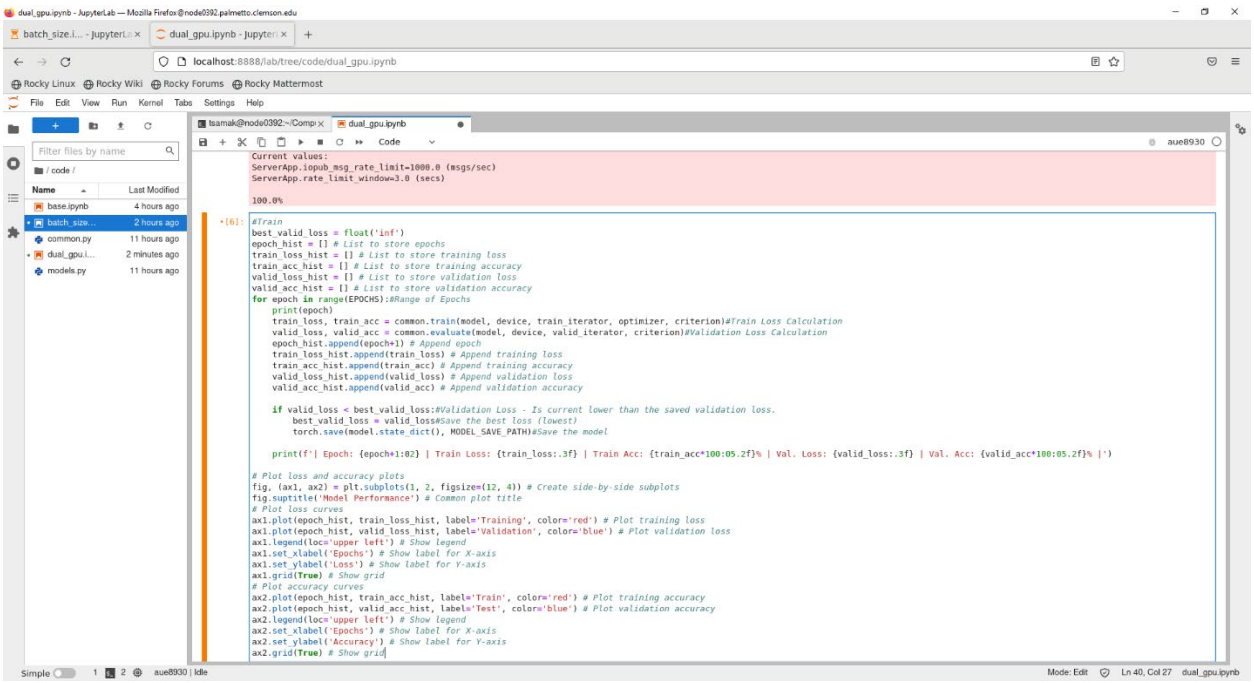


4.6 2xA100 GPU usage (GPU0=~6%; GPU1=~3%) and GPU RAM usage (GPU0=~2.6%; GPU1=~2.3%) for implementation with dual-GPU configuration.

(5) Plot the accuracy against the number of training Epochs on a Graph. (10 points) (TIPS: you need to import matplotlib, modify the code of "for epoch in range (EPOCHS):" by saving the "epoch" and "train_acc", and plot its relationship in the end)



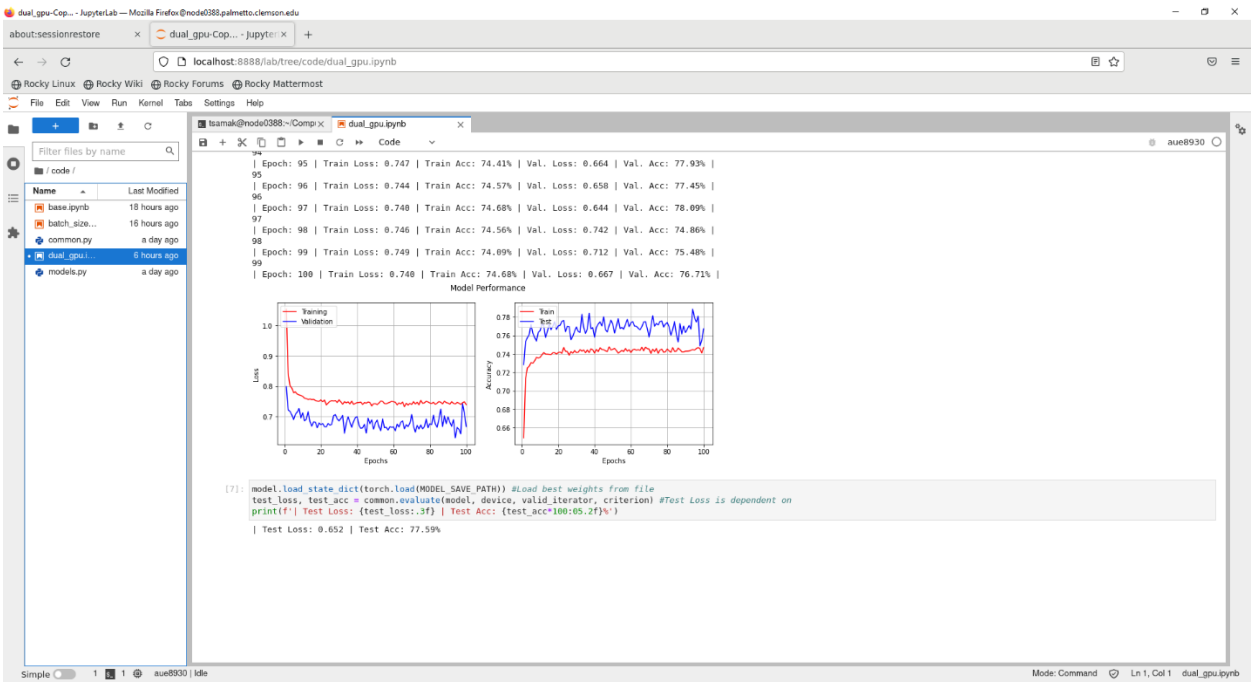
5.1 Import `matplotlib`



```
Current values:
ServerApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
ServerApp.rate_limit_window=3.0 (secs)

100.0%
```

```
#Train
best_valid_loss = float('inf')
epoch_hist = [] # List to store epochs
train_loss_hist = [] # List to store training loss
train_acc_hist = [] # List to store training accuracy
valid_loss_hist = [] # List to store validation loss
valid_acc_hist = [] # List to store validation accuracy
for epoch in range(EPOCHS):#Range of Epochs
    print(epoch)
    train_loss, train_acc = common.train(model, device, train_iterator, optimizer, criterion)#Train Loss Calculation
    valid_loss, valid_acc = common.evaluate(model, device, valid_iterator, criterion)#Validation Loss Calculation
    epoch_hist.append(epoch+1) # Append epoch
    train_loss_hist.append(train_loss) # Append training loss
    train_acc_hist.append(train_acc) # Append training accuracy
    valid_loss_hist.append(valid_loss) # Append validation loss
    valid_acc_hist.append(valid_acc) # Append validation accuracy

    if valid_loss < best_valid_loss:#Validation Loss - Is current lower than the saved validation loss.
        best_valid_loss = valid_loss#Save the best loss (lowest)
        torch.save(model.state_dict(), MODEL_SAVE_PATH)#Save the model

    print(f'| Epoch: {epoch+1:02} | Train Loss: {train_loss:.3f} | Train Acc: {train_acc*100:05.2f}% | Val. Loss: {valid_loss:.3f} | Val. Acc: {valid_acc*100:05.2f}% |')

# Plot loss and accuracy plots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4)) # Create side-by-side subplots
fig.suptitle('Model Performance') # Common plot title
# Plot loss curves
ax1.plot(epoch_hist, train_loss_hist, label='Training', color='red') # Plot training loss
ax1.plot(epoch_hist, valid_loss_hist, label='Validation', color='blue') # Plot validation loss
ax1.legend(loc='upper left') # Show legend
ax1.set_xlabel('Epochs') # Show label for X-axis
ax1.set_ylabel('Loss') # Show label for Y-axis
ax1.grid(True) # Show grid
# Plot accuracy curves
ax2.plot(epoch_hist, train_acc_hist, label='Train', color='red') # Plot training accuracy
ax2.plot(epoch_hist, valid_acc_hist, label='Test', color='blue') # Plot validation accuracy
ax2.legend(loc='upper left') # Show legend
ax2.set_xlabel('Epochs') # Show label for X-axis
ax2.set_ylabel('Accuracy') # Show label for Y-axis
ax2.grid(True) # Show grid
```

5.2 Modify the training section to keep track of loss and accuracy metrics and plot them.



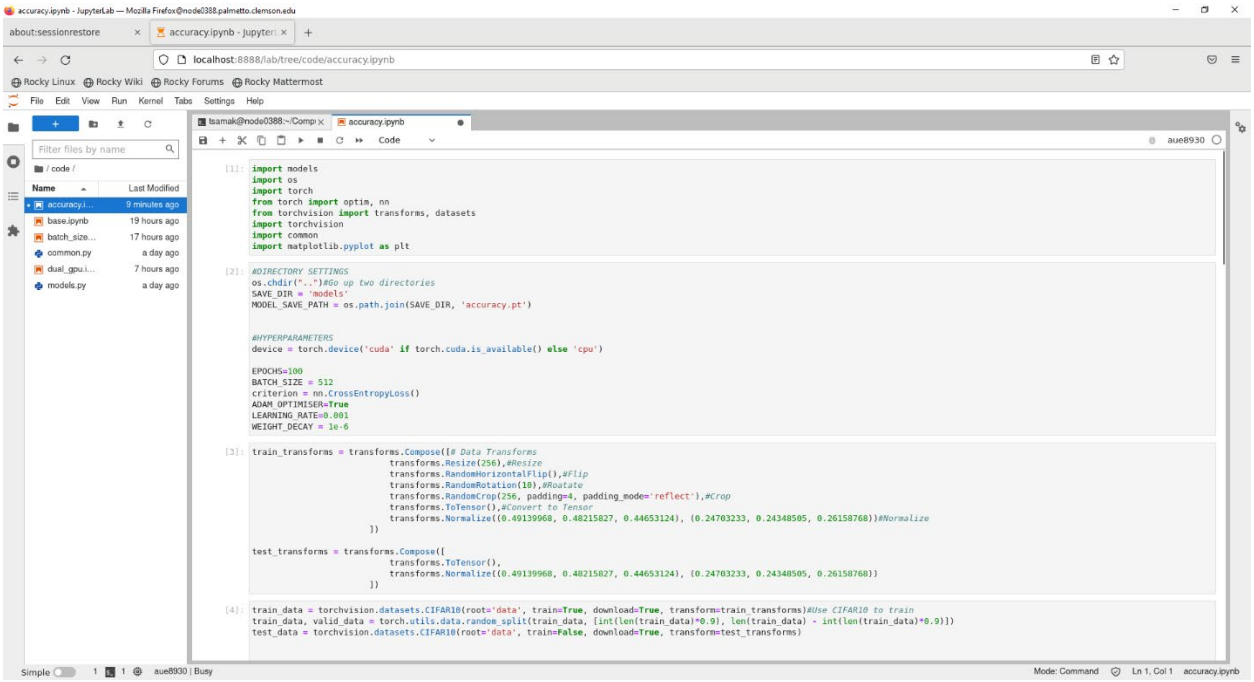
```
| Epoch: 95 | Train Loss: 0.747 | Train Acc: 74.41% | Val. Loss: 0.664 | Val. Acc: 77.93% |
95
| Epoch: 96 | Train Loss: 0.744 | Train Acc: 74.57% | Val. Loss: 0.658 | Val. Acc: 77.45% |
96
| Epoch: 97 | Train Loss: 0.740 | Train Acc: 74.68% | Val. Loss: 0.644 | Val. Acc: 78.09% |
97
| Epoch: 98 | Train Loss: 0.746 | Train Acc: 74.56% | Val. Loss: 0.742 | Val. Acc: 74.86% |
98
| Epoch: 99 | Train Loss: 0.749 | Train Acc: 74.09% | Val. Loss: 0.712 | Val. Acc: 75.48% |
99
| Epoch: 100 | Train Loss: 0.740 | Train Acc: 74.68% | Val. Loss: 0.667 | Val. Acc: 76.71% |
```

```
model.load_state_dict(torch.load(MODEL_SAVE_PATH)) #Load best weights from file
test_loss, test_acc = common.evaluate(model, device, valid_iterator, criterion) #Test Loss is dependent on
print(f'| Test Loss: {test_loss:.3f} | Test Acc: {test_acc*100:05.2f}%')
```

```
| Test Loss: 0.652 | Test Acc: 77.59%
```

5.3 Training and validation loss and accuracy plots for 100 epochs; testing loss and accuracy.

(6) Could you improve on the network model, train it for better accuracy? (optional, 5 points) (This question is optional. Extra 5 points until reach the cap of 100)



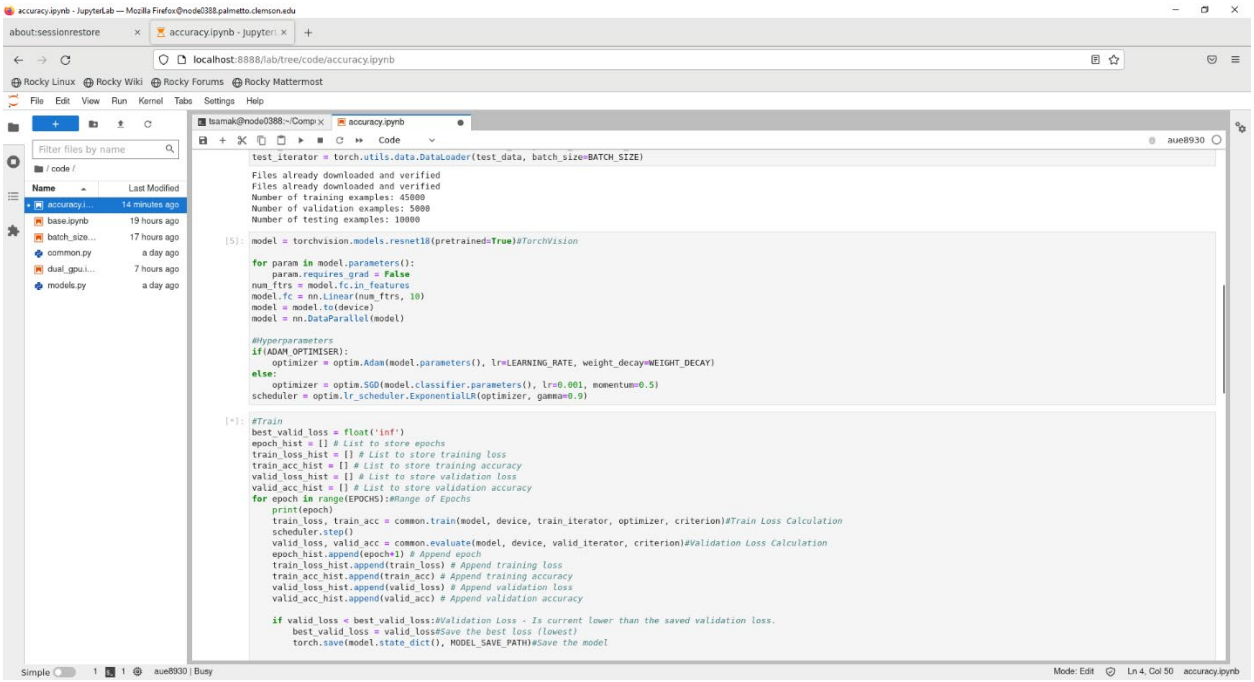
```
import models
import os
import torch
from torch import optim, nn
from torchvision import transforms, datasets
import torchvision
import common
import matplotlib.pyplot as plt
```

```
#DIRECTORY SETTINGS
os.chdir("..")#Go up two directories
SAVE_DIR = 'models'
MODEL_SAVE_PATH = os.path.join(SAVE_DIR, 'accuracy.pt')


#HYPERPARAMETERS
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

EPOCHS=100
BATCH_SIZE = 512
criterion = nn.CrossEntropyLoss()
ADAM_OPTIMISER=True
LEARNING_RATE=0.001
WEIGHT_DECAY = 1e-6
```

```
train_transforms = transforms.Compose([# Data Transforms
                        transforms.Resize(256),#Resize
                        transforms.RandomHorizontalFlip(),#Flip
                        transforms.RandomRotation(10),#Roatate
                        transforms.RandomCrop(256, padding=4, padding_mode='reflect'),#Crop
                        transforms.ToTensor(),#Convert to Tensor
                        transforms.Normalize((0.49139968, 0.48215827, 0.44653124), (0.24703233, 0.24348505, 0.26158768))#Normalize
                    ])

test_transforms = transforms.Compose([
                        transforms.ToTensor(),
                        transforms.Normalize((0.49139968, 0.48215827, 0.44653124), (0.24703233, 0.24348505, 0.26158768))
                    ])
```

```
train_data = torchvision.datasets.CIFAR10(root='data', train=True, download=True, transform=train_transforms)#Use CIFAR10 to train
train_data, valid_data = torch.utils.data.random_split(train_data, [int(len(train_data)*0.9), len(train_data) - int(len(train_data)*0.9)])
test_data = torchvision.datasets.CIFAR10(root='data', train=False, download=True, transform=test_transforms)
```

6.1 Change batch size (512), and implement weight decay and robust augmentations.



```
test_iterator = torch.utils.data.DataLoader(test_data, batch_size=BATCH_SIZE)
```

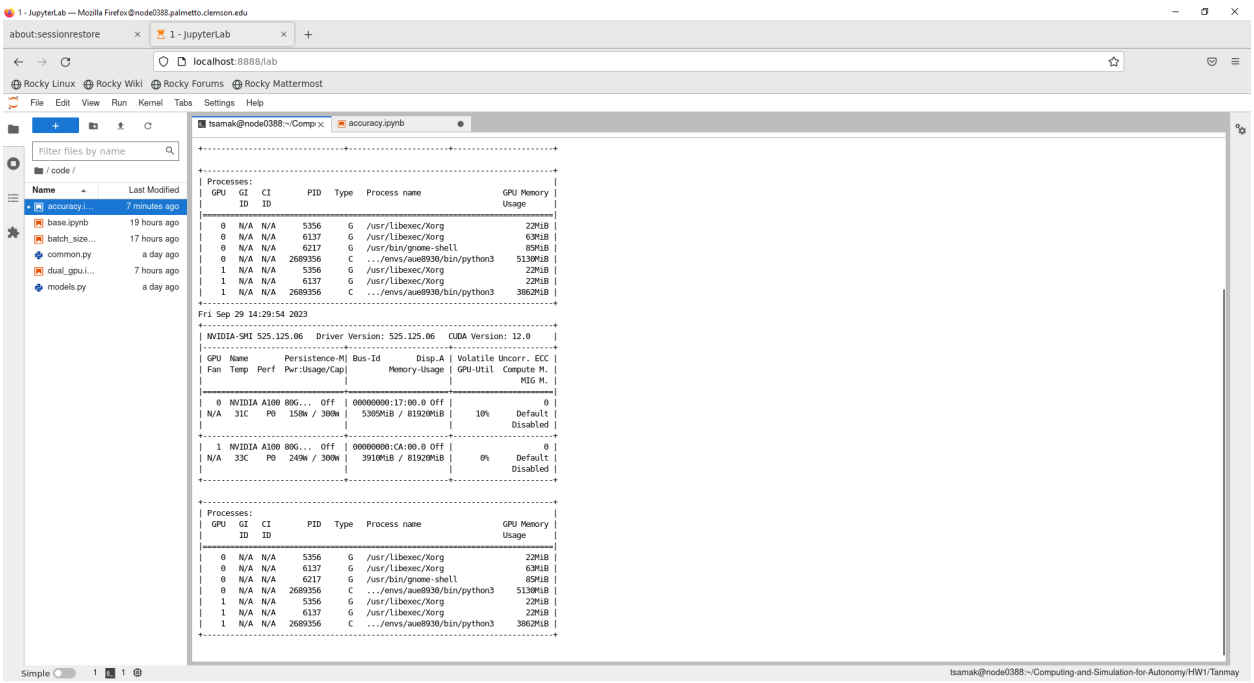
```
Files already downloaded and verified
Files already downloaded and verified
Number of training examples: 45000
Number of validation examples: 5000
Number of testing examples: 10000
```

```
model = torchvision.models.resnet18(pretrained=True)#TorchVision

for param in model.parameters():
    param.requires_grad = False
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, 10)
model = model.to(device)
model = nn.DataParallel(model)

#Hyperparameters
if(ADAM_OPTIMISER):
    optimizer = optim.Adam(model.parameters(), lr=LEARNING_RATE, weight_decay=WEIGHT_DECAY)
else:
    optimizer = optim.SGD(model.classifier.parameters(), lr=0.001, momentum=0.5)
scheduler = optim.lr_scheduler.ExponentialLR(optimizer, gamma=0.9)
```

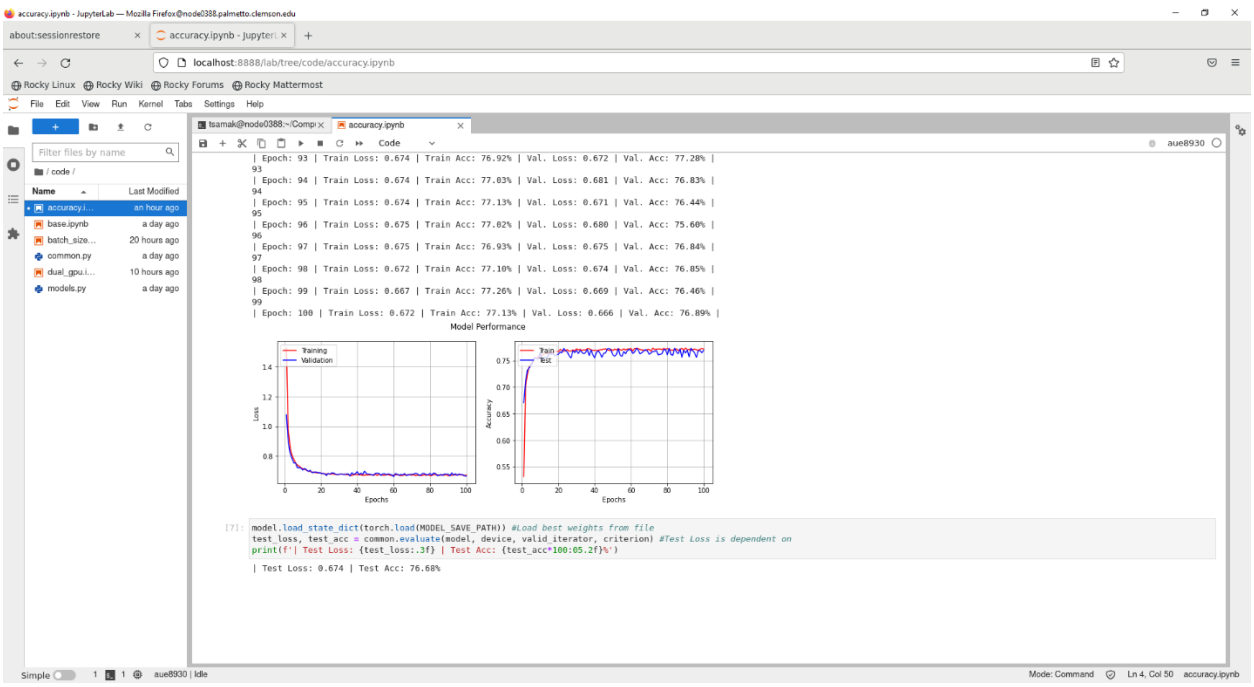
```
#Train
best_valid_loss = float('inf')
epoch_hist = [] # List to store epochs
train_loss_hist = [] # List to store training loss
train_acc_hist = [] # List to store training accuracy
valid_loss_hist = [] # List to store validation loss
valid_acc_hist = [] # List to store validation accuracy
for epoch in range(EPOCHS):#Range of Epochs
    print(epoch)
    train_loss, train_acc = common.train(model, device, train_iterator, optimizer, criterion)#Train Loss Calculation
    scheduler.step()
    valid_loss, valid_acc = common.evaluate(model, device, valid_iterator, criterion)#Validation Loss Calculation
    epoch_hist.append(epoch+1) # Append epoch
    train_loss_hist.append(train_loss) # Append training loss
    train_acc_hist.append(train_acc) # Append training accuracy
    valid_loss_hist.append(valid_loss) # Append validation loss
    valid_acc_hist.append(valid_acc) # Append validation accuracy

    if valid_loss < best_valid_loss:#Validation Loss - Is current lower than the saved validation loss.
        best_valid_loss = valid_loss#Save the best loss (lowest)
        torch.save(model.state_dict(), MODEL_SAVE_PATH)#Save the model
```

6.2 Implement exponential learning rate scheduling with gamma = 0.9.

Hyperparameters:	Augmentations:
BATCH_SZIE = 512 *WEIGHT_DECAY = 1e-6* *LR_SCHEDULER.EXPONENTIAL_LR(gamma=0.9)*	*RANDOM_HORIZONTAL_FLIP(50%)* *RANDOM_CROP(256, padding=4, padding_model='reflect')*



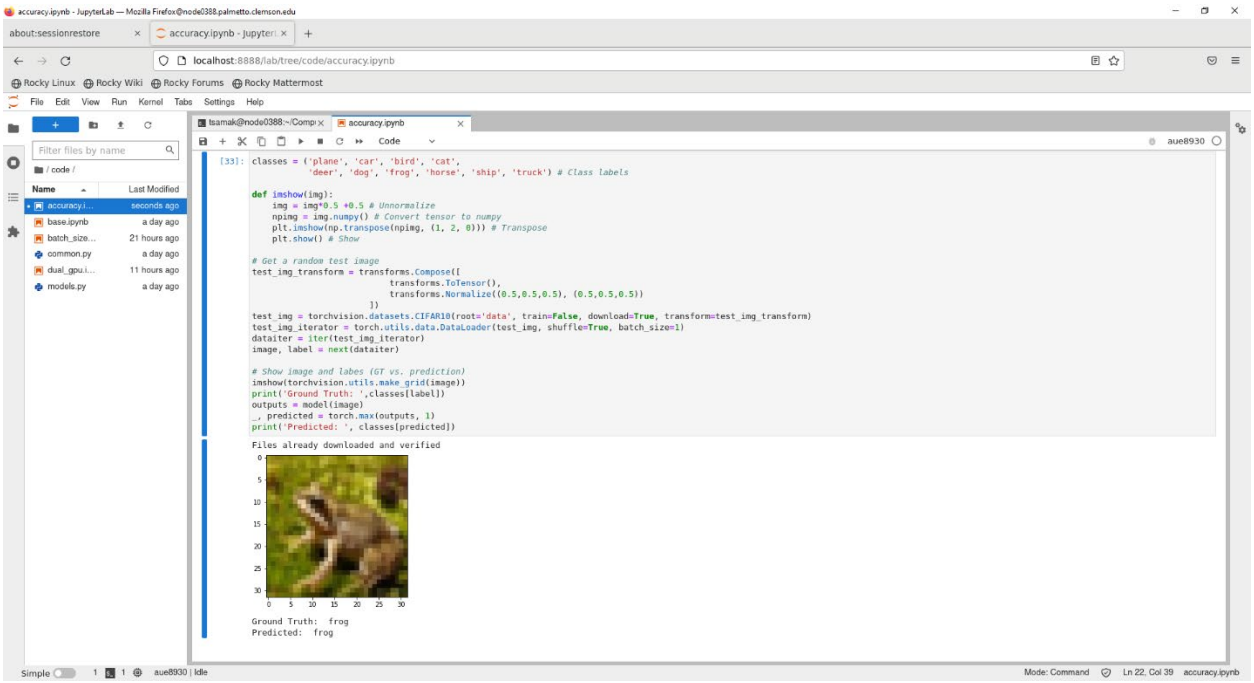
6.3 2xA100 GPU usage (GPU0=~10%; GPU1=~0%) and GPU RAM usage (GPU0=~6.4%; GPU1=~4.8%) for implementation with dual-GPU configuration.



6.4 Training, validation and testing loss and accuracy metrics for 100 epochs.

Note: The training became significantly stable with the modification made to the pipeline. As a result, the training, validation and testing loss and accuracy metrics fall really close, which ultimately points towards superior performance of this model (as compared to the baseline implementation.

(7) Perform a model inference for a certain image, which you can choose from anywhere. The image shall include the object which belongs to the category of the training dataset. (10 points) (TIPS: if you are using CIFAR10 datasets, its categories are shown in this reference)



7.1 Mode inference on a test image with ground truth and predicted labels.

Question 2

Write a 2~3 pages survey report on a particular High-Performance-Computing application related to engineering/vehicles (40 points). The grading of this question will be based on the contents which the survey covers:

- What is the problem to be solved (5 points);
- The importance of the problem to be solved (5 points);
- The challenges of solving this problem (10 points);
- Existing solutions of solving this problem (15 points);
- Other grading factors (such as novelty, organization, etc.) (5);

* You are encouraged to include any drawing/table in the report;
* Attention: use like [1] to cite a content you referred to, with reference list in the end. You should never literally copy contents from other places;

TIPS: you should survey and read multiple academic papers, academic papers. Then, summarize for the above.

[PTO]

High Performance Computing for Developing Hierarchically Consistent Digital Twins for Autonomous Vehicles using the Real2Sim Approach

Tanmay Samak

1. Introduction

Modeling and simulation form an integral part of moving from reality to simulation (real2sim). These models could be first principles models (white-box) including rigid-body – kinematics, dynamics and contact models; soft-body – continuum, truss and finite-element models, multi-physics – bond-graph and object-oriented models; small-scale statistical models (grey-box) including system identification – parameter estimation; curve fitting – polynomials, splines, Gaussian process based models; large-scale neural models (black-box) using machine learning techniques (supervised/un-supervised/reinforcement learning).

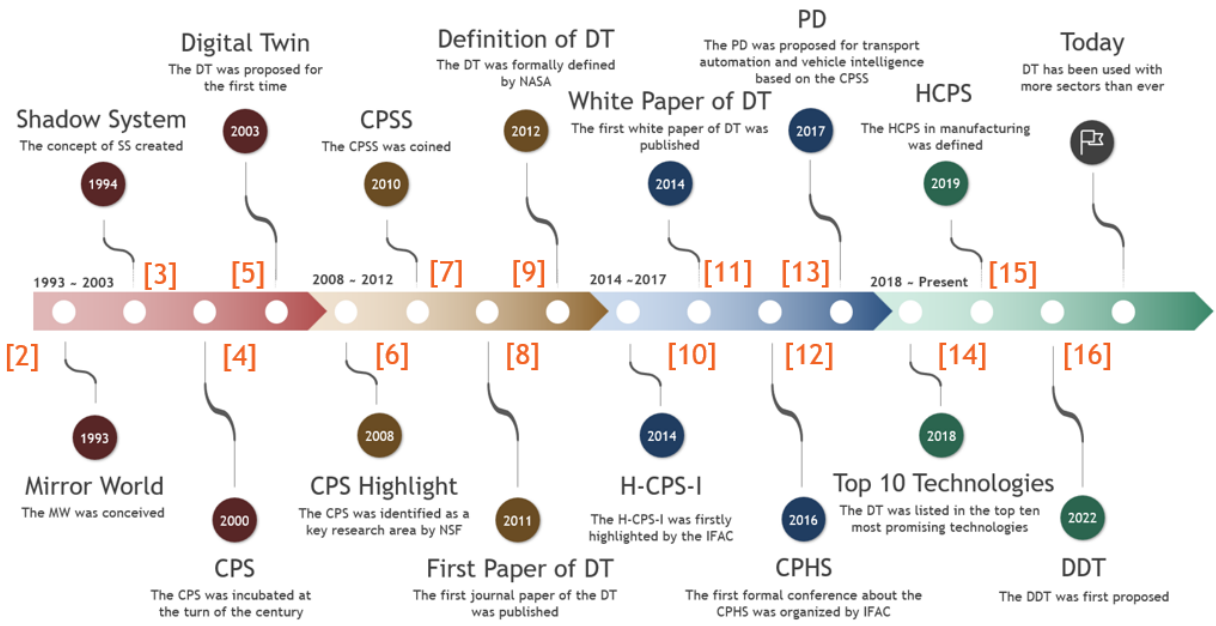


Figure 1. Origins of digital twin technology [1]

Although modeling was practiced since a long time, it was John von Neumann and Stanislaw Ulam who first used these “models” and computationally simulated the behavior of neutrons in a nuclear shielding problem for the first time. This way, in addition to just having models, solvers that performed numerical time-stepping and integrating the evolution of their states over time became common. Later, kinematic and dynamic simulations for mechanism design and analysis started becoming common wherein the results were often mere numbers or time/frequency domain plots describing the system behavior. Going further, as the complexities of these models grew and the number of states exploded exponentially, it became highly intractable to analyze their behavior and benchmark their performance using just numbers or plots. This is when advanced visualization features such as motion analysis, animation and photorealistic

rendering started becoming very popular and demanding. Finally, especially in the context of autonomous mobile robots and vehicles, features such as extended application programming interfaces (APIs) to multiple software frameworks and programming languages as well as user-friendly graphical user interfaces (GUIs) have started to become a recent popular demand.

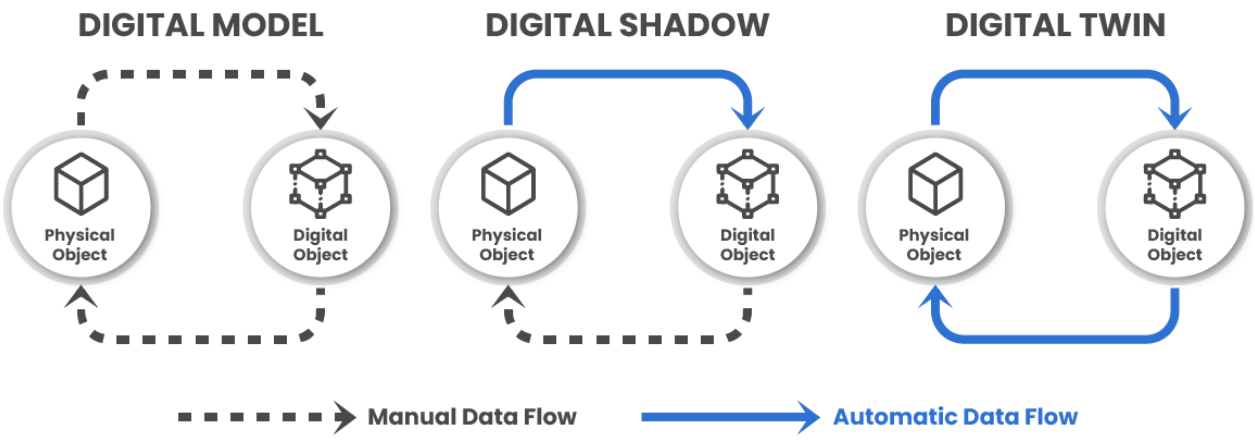


Figure 1. Distinction between different stages of the digital thread [17-19].

Moreover, recent technological developments have given birth to and popularized the notion of digital twins. Origins of digital twins can be traced back to the concept of mirror world [2] for real-time geographically accurate representations with data-driven updates, which was followed by shadow systems [3] for engineering applications with one-way data flow & updates, succeeded by cyber-physical systems (CPS) [4] and cyber-physical social systems CPSS [7] involving social interactions, feedback and cooperation, finally leading to the modern-day digital twins [5], [9], [11] encompassing two-way data flow and updates in real-time.

2. Motivation

Specifically, in the context of autonomous vehicles, high performance computing (HPC) plays a crucial role in developing hierarchically consistent digital twins using the real2sim approach. To reiterate, digital twins are virtual models that replicate the behavior and performance of physical entities in real-time [20]. They have been successfully applied in various industries, including medicine, manufacturing, and now in the autonomous vehicle industry [21]. The real2sim approach involves creating a digital twin that closely resembles the real-world vehicle and the environment its operating in and using it to simulate and test different scenarios and conditions [22].

One of the key challenges in developing digital twins for autonomous vehicles is ensuring the consistency between the digital twin and the real vehicle. This consistency is crucial for accurate simulation and testing. [23] discuss the importance of consistency in digital twin test methods and real vehicle site validation for intelligent vehicles. They propose a digital twin parallel test system that combines real-time parallel simulation and 5G cellular mobile technology to achieve more challenging tests. This approach accelerates the research, development, and evaluation of autonomous vehicles and reduces the possibility of human error.

To enable efficient task offloading in autonomous vehicles, [24] propose a Digital Twin (DT) empowered task offloading framework for the Internet of Vehicles. This framework leverages the high mobility of vehicles, the dynamics of wireless conditions, and the uncertainty of computing tasks to determine the optimal offloading strategy. By using digital twins, vehicles can offload computing tasks to mobile edge computing infrastructure, improving performance and reducing latency.

High-performance computing is essential for developing hierarchically consistent digital twins for autonomous vehicles. As described in [25], the development of an automatic driving simulation test system based on digital twin technology. This system utilizes the rapid development of 5G infrastructure and cloud computing to test and evaluate autonomous vehicles safely and efficiently. The digital twin-based simulation allows for extensive testing and evaluation before deploying vehicles on real public roads.

3. Literature Survey

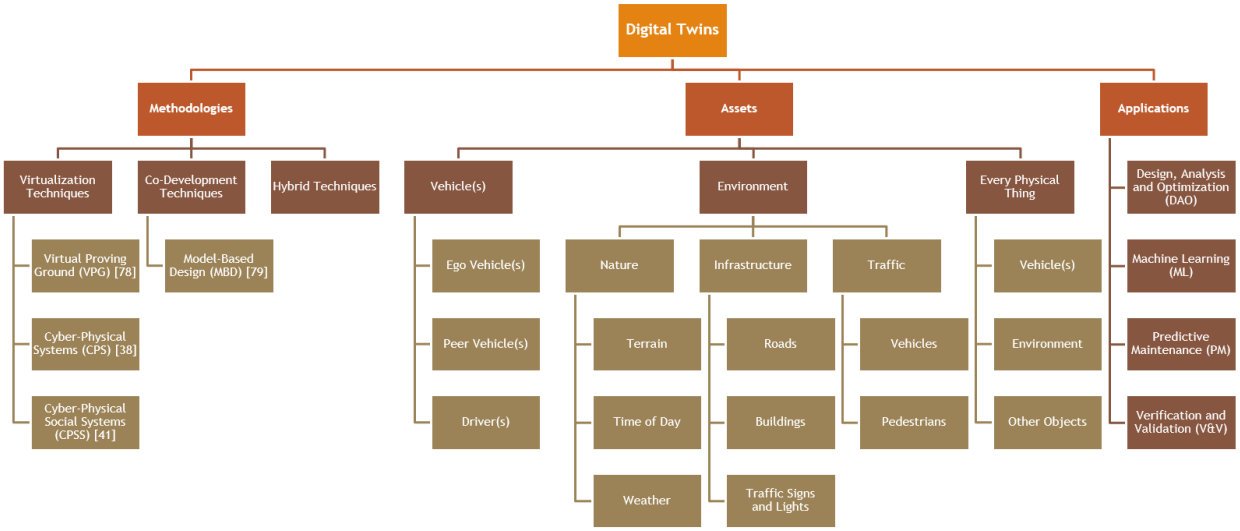


Figure 2. Breakdown of current state-of-the-art literature

The rich state-of-the-art literature in this domain can be broken down based on the methodology adopted for creating the digital twins, the assets (vehicle, environment, etc.) that are under consideration for developing digital twins, and the application space of these digital twin models. Here, I have presented a literature survey primarily considering the method of conceiving digital twin models with a secondary focus on the assets under consideration and their applications in autonomous vehicle domain.

- **Virtual Proving Ground (VPG):** This technique involves accurately modeling, estimating and simulating an existing real-world scenario, with a prominent focus on variability testing, corner-case analysis, and design optimization.
- **Cyber-Physical Systems (CPS):** This method involves instrumenting a physical asset with sensors and communication equipment to measure and transmit real-world data through the digital thread to update the digital twin, and vice-versa.

- **Cyber-Physical Social Systems (CPSS):** This method uses CPS as a substrate and focuses on modeling, estimating and simulating social interactions, feedbacks and coordination amongst multiple agents.
- **Model Based Design (MBD):** This technique proposes simultaneous development of the physical and digital twins withing a concurrent engineering framework, wherein the virtual prototype developed in the first half of its lifecycle is converted and employed as a digital twin in the other half of its lifecycle.

Table 1. State-of-the-art literature for virtual proving ground (VPG) methodology

Article/Author	Methodology	Tools/Frameworks	Twin objects	Application	Summary	Year
Atorf et al. [26]	VPG	Experimentable Digital Twin (EDT)	Vehicles, environment, traffic	PM	Interactive analysis and visualization for vehicles and robots	2018
Chen et al. [27]	VPG	Generative Adversarial Network (GAN)	Environment, traffic	V&V	Safety-critical scenario generation for motion planning	2019
Culley et al. [28]	VPG	Gazebo, Robot Operating System (ROS)	Vehicle, environment	DAO	System design for autonomous racing vehicle	2020
Fremont et al. [29]	VPG	LGSVL Simulator	Environment	V&V	Scenario-based testing of autonomous vehicles	2020
Wu et al. [30]	VPG	CARLA Simulator	Environment, traffic	ML	Model-based RL in autonomous driving	2021
Wang et al. [31]	VPG	Unity, SUMO, MATLAB, Python, AWS	Vehicles, environment	DAO	Personalized adaptive cruise control (P-ACC)	2021
Malayjerdi et al. [32]	VPG	Unity, Metashape, Autoware	Environment	V&V	VIrtual testing of autonomous vehicles	2021

Table 2. State-of-the-art literature for cyber physical systems (CPS) methodology

Article/Author	Methodology	Tools/Frameworks	Twin objects	Application	Summary	Year
Schwarz et al. [33]	CPS	National Advanced Driving Simulator (NADS)	Environment	DAO	Digital map aware enhancement of electronic stability control (ESC)	2010
Eleonora et al. [34]	CPS	Gazebo, ROS	Vehicles, environment	DAO	AGV logistics action optimization	2017
Chen et al. [35]	CPS	Unity Engine	Drivers	V&V	Predict future actions of neighboring vehicles	2018
Veledar et al. [36]	CPS	IoT4CPS Framework	Vehicles	V&V	Safe and secure integration of IoT into AD	2019
Liu et al. [37]	CPS	Unity Engine	Vehicles, drivers	V&V	Multi-sensor fusion for vehicle recognition	2020
Liu et al. [38]	CPS	Unreal Engine	Environment	V&V	Infrastructure-vehicle cooperative driving	2021

Wang et al. [39]	CPS	Unity Engine	Vehicles, drivers, environment, traffic	V&V	Cooperation at non-signalized intersections	2021
Staczek et al. [40]	CPS	Gazebo, ROS	Vehicle, environment	V&V	AGV logistics action testing	2021

Table 3. State-of-the-art literature for cyber physical social systems (CPSS) methodology

Article/Author	Methodology	Tools/Frameworks	Twin objects	Application	Summary	Year
Wang et al. [41]	CPSS	Artificial societies, Computational experiments and Parallel execution (ACP)	Vehicles, environment	DAO	Parallel transportation management systems	2010
Wang et al. [42]	CPSS	ACP	Every physical thing	DAO	Smart society	2016
Liu et al. [43]	CPSS	Panosim, ACP	Vehicles	DAO	Parallel driving using descriptive, predictive, prescriptive and real vehicles	2019
Lu et al. [44]	CPSS	EuArtisan framework	Every physical thing	DAO	Parallel factories	2022
Wang et al. [45]	CPSS	CARLA, SUMO, Unity, Redis	Vehicles, traffic environment, pedestrians	V&V	Vehicle-to-Pedestrian (V2P) warning system	2023

Table 4. State-of-the-art literature for model-based design (MBD) methodology

Article/Author	Methodology	Tools/Frameworks	Twin objects	Application	Summary	Year
Laschinsky et al. [46]	MBD	Virtual Test Drive (VTD)	Vehicles	V&V	Vehicle-in-the-Loop (VIL) validation of active safety lights	2010
Shikata et al. [47]	MBD	Unity Engine	Vehicles	V&V	Electric Vehicle (EV) automatic parking and charging design and test	2019
Dygalo et al. [48]	MBD	Custom Testbenches	Vehicle sub-system, system and system-of-systems	V&V	Vehicle active safety technology system	2020
Wagg et al. [49]	MBD	MATLAB	Infrastructure	DAO	Shaking three-store building	2020

Table 5. State-of-the-art literature for hybrid methodologies

Article/Author	Methodology	Tools/Frameworks	Twin objects	Application	Summary	Year
Wang et al. [13]	CPSS, MBD	ACP, iHorizon	Vehicle, driver, environment, traffic	DAO	Intelligent energy management for autonomous EVs	2017

Rassolkin et al. [50]	CPS, MBD	ISEAUTO autonomous shuttle bus	Vehicle sub-systems	V&V	Test stand for electric propulsion drive systems (EPDS) of self-driving EVs	2019
Ge et al. [51]	VPG, CPS, MBD	LTE-V2X framework	Vehicles, environment	V&V	Virtual, hybrid and physical testing of autonomous vehicles	2019
Szalai et al. [52]	CPSS, VPG	Unity, SUMO	Vehicles, traffic, pedestrians	V&V	Mixed-reality ADAS/AD validation	2020
Yu et al. [53]	MBD, CPS, CPSS	Structural, physical and logical twin framework	Environment, sensors, traffic	V&V	ADAS/AD software design and test	2022

4. Conclusion

In conclusion, high-performance computing is crucial for developing hierarchically consistent digital twins for autonomous vehicles using the Real2Sim approach. Digital twins enable accurate simulation and testing of autonomous vehicles, reducing the possibility of human error and improving safety. The Real2Sim approach, combined with parallel testing systems and task offloading frameworks, accelerates the research and development of autonomous vehicles. With the rapid development of technologies such as 5G and cloud computing, digital twin-based simulation and testing systems are becoming more efficient and effective in ensuring the safety and performance of autonomous vehicles.

References

[1] S. M. Mostaq Hossain, S. Kumar Saha, S. Banik and T. Banik, "A New Era of Mobility: Exploring Digital Twin Applications in Autonomous Vehicular Systems," *2023 IEEE World AI IoT Congress (AIIoT)*, Seattle, WA, USA, 2023, pp. 0493-0499, doi: 10.1109/AIIoT58121.2023.10174376

[2] D. Gelernter, "Mirror Worlds: The Day Software Puts the Universe In a Shoebox... How it Will Happen and What It Will Mean?," Oxford University Press, New York, NY, USA, 1991.

[3] F. Wang, "Shadow systems: A new concept for nested and embedded co-simulation for intelligent systems," Univ. Arizona, Tucson, AZ, USA, 1994.

[4] S. Sarma, D.L. Brock and K. Ashton, "The Networked Physical World: Proposals for Engineering the Next Generation of Computing, Commerce & Automatic-Identification," Massachusetts Inst. Technol., Cambridge, MA, USA, White Paper, vol. 1, pp. 1-16, Oct 2000. [Online]. Available: https://cocoa.ethz.ch/downloads/2014/06/None_MIT-AUTOID-WH-001.pdf

[4] M. Grieves, "Product Lifecycle Management (PLM)," University of Michigan, MI, USA, Executive Course, 2003.

[6] H. Gill and S.F. Midkiff, "Cyber-Physical Systems (CPS)," NSF 08-611, National Science Foundation (NSF), Sep 2008. [Online]. Available: https://www.nsf.gov/publications/pub_summ.jsp?ods_key=nsf08611

[7] F. -Y. Wang, "The Emergence of Intelligent Enterprises: From CPS to CPSS," in IEEE Intelligent Systems, vol. 25, no. 4, pp. 85-88, Jul-Aug 2010, doi: 10.1109/MIS.2010.104

[8] E.J. Tuegel, A.R. Ingraffea, T.G. Eason, S.M. Spottswood, "Reengineering Aircraft Structural Life Prediction Using a Digital Twin," International Journal of Aerospace Engineering, vol. 2011, Article ID 154798, 14 pages, 2011, doi: https://doi.org/10.1155/2011/154798

[9] E. Glaessgen and D. Stargel, "The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles," in Proc. 53rd AIAA/ASME/ASCE/AHS/ASC Struct., Struct. Dyn. Mater. Conf. 20th AIAA/ASME/AHS Adapt. Struct. Conf. 14th AIAA, Article ID 1818, Apr. 2012, doi: 10.2514/6.2012-1818

[10] IFAC/IEEE, “H-CPS-I : Human Cyber Physical System Interaction - Control for the Human Welfare,” Website of the IFAC IEEE CSS Workshop Human Cyber Physical System Interaction: Control for Human Welfare H-CPS-I 2014, Paris, France, September 22-23, 2014. [Online]. Available: https://h-cps-i.sciencesconf.org

[11 M. Grieves, “Digital Twin: Manufacturing Excellence through Virtual Factory Replication,” Florida Inst. Technol., Melbourne, FL, USA, White Paper, vol. 1, pp. 1-7, 2014. [Online]. Available: https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/DELMIA/PDF/Whitepaper/DELMIA-APRISO-Digital-Twin-Whitepaper.pdf

[12] IFAC/IEEE, “Cyber-Physical & Human-Systems - 1st CPHS 2016,” Website of the IFAC IEEE, EECI, NSF Conference on Cyber-Physical & Human-Systems - 1st CPHS 2016, Florianopolis, Brazil, December 7-9, 2016. [Online]. Available: https://www.ifac-control.org/conferences/cyber-physical-human-systems-1st-cphs-2016

[13] F. -Y. Wang, N. -N. Zheng, D. Cao, C. M. Martinez, L. Li and T. Liu, “Parallel driving in CPSS: a unified approach for transport automation and vehicle intelligence,” in IEEE/CAA Journal of Automatica Sinica, vol. 4, no. 4, pp. 577-587, 2017, doi: 10.1109/JAS.2017.7510598

[14] Kasey Panetta, “Gartner Top 10 Strategic Technology Trends for 2018,” Gartner Inc., Oct 2017. [Online]. Available: https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2018

[15] J. Zhou, P. Li, Y. Zhou, B. Wang, J. Zang and L. Meng, “Toward New-Generation Intelligent Manufacturing,” Engineering, vol. 4, no. 1, 2018, pp. 11-20, doi: 10.1016/j.eng.2018.01.002

[16] Z. Hu, S. Lou, Y. Xing, X. Wang, D. Cao and C. Lv, “Review and Perspectives on Driver Digital Twin and Its Enabling Technologies for Intelligent Vehicles,” in IEEE Transactions on Intelligent Vehicles, vol. 7, no. 3, pp. 417-440, Sept. 2022, doi: 10.1109/TIV.2022.3195635

[17] A. Andrade, “The 3 Levels of the Digital Twin Technology,” Vidya Blog, Vidya Technology, Aug 2022. [Online]. Available: https://vidyatec.com/blog/the-3-levels-of-the-digital-twin-technology-2/

[18] R. Nikula, M. Paavola, M. Ruusunen and J. Keski-Rahkonen, “Towards Online Adaptation of Digital Twins,” Open Engineering, vol. 10, no. 1, 2020, pp. 776-783, doi: 10.1515/eng-2020-0088

[19] G. Barbieri, et al., “A virtual commissioning based methodology to integrate digital twins into manufacturing systems,” in Prod. Eng. Res. Devel., vol. 15, pp. 397-412, 2021, doi: 10.1007/s11740-021-01037-3

[20] Yu, G., Zhang, S., Hu, M., & Wang, Y. (2020). Prediction of highway tunnel pavement performance based on digital twin and multiple time series stacking. Advances in Civil Engineering, 2020, 1-21, doi: 10.1155/2020/8824135

[21] Laubenbacher, R., Niarakis, A., Helikar, T., An, G., Shapiro, B., Sheriff, R., … & Glazier, J. (2022). Building digital twins of the human immune system: toward a roadmap. NPJ Digital Medicine, 5(1), doi: 10.1038/s41746-022-00610-z

[22] Almeaibed, S., Al-Rubaye, S., Tsourdos, A., & Avdelidis, N. (2021). Digital twin analysis to promote safety and security in autonomous vehicles. Ieee Communications Standards Magazine, 5(1), 40-46, doi: 10.1109/mcomstd.011.2100004

[23] Zhao, S., Wu, S., Tang, Y., Meng, Z., Tian, H., & Xu, G. (2023). Research on consistency of digital twin test method and real vehicle site validation for intelligent vehicle, doi: doi: 10.1117/12.2668577

[24] Zheng, J., Luan, T., Gao, L., Zhang, Y., & Yuan, W. (2022). Learning based task offloading in digital twin empowered internet of vehicles, doi: https://doi.org/10.48550/arxiv.2201.09076

[25] Chen, X., Jin, Z., Li, P., Zhang, S., Sun, J., Tian, X., … & Zhang, J. (2022). Research on automatic driving simulation test system based on digital twin. Journal of Physics Conference Series, 2170(1), 012039, doi: 10.1088/1742-6596/2170/1/012039

[26] L. Atorf and J. Roßmann, “Interactive Analysis and Visualization of Digital Twins in High-Dimensional State Spaces,” 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, 2018, pp. 241-246, doi: 10.1109/ICARCV.2018.8581126

[27] L. Chen, X. Hu, B. Tang and D. Cao, “Parallel Motion Planning: Learning a Deep Planning Model against Emergencies,” in IEEE Intelligent Transportation Systems Magazine, vol. 11, no. 1, pp. 36-41, Spring 2019, doi: 10.1109/MITS.2018.2884515

[28] J. Culley et al., “System Design for a Driverless Autonomous Racing Vehicle,” 2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), Porto, Portugal, 2020, pp. 1-6, doi: 10.1109/CSNDSP49049.2020.9249626

[29] D. J. Fremont et al., "Formal Scenario-Based Testing of Autonomous Vehicles: From Simulation to the Real World," 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), Rhodes, Greece, 2020, pp. 1-8, doi: 10.1109/ITSC45102.2020.9294368

[30] J. Wu, Z. Huang, P. Hang, C. Huang, N. De Boer and C. Lv, “Digital Twin-enabled Reinforcement Learning for End-to-end Autonomous Driving,” 2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI), Beijing, China, 2021, pp. 62-65, doi: 10.1109/DTPI52967.2021.9540179

[31] Z. Wang, K. Han and P. Tiwari, “Digital Twin Simulation of Connected and Automated Vehicles with the Unity Game Engine,” 2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI), Beijing, China, 2021, pp. 1-4, doi: 10.1109/DTPI52967.2021.9540074

[32] M. Malayjerdi, V. Kuts, R. Sell, T. Otto and B.C. Baykara, “Virtual Simulations Environment Development for Autonomous Vehicles Interaction,” Proceedings of the ASME 2020 International Mechanical Engineering Congress and Exposition, vol. 2B: Advanced Manufacturing, Virtual, Online, November 16–19, 2020, Article ID: IMECE2020-23362, V02BT02A009, ASME, doi: 10.1115/IMECE2020-23362

[33] K. Moran and C. Schwarz, “Digital Map Enhancements of Electronic Stability Control,” in SAE 2010 World Congress & Exhibition, SAE Technical Paper 2010-01-0089, SAE International, Apr 2010, doi: 10.4271/2010-01-0089

[34] E. Bottani, A. Cammardella, T. Murino and S. Vespoli, “From the Cyber-Physical System to the Digital Twin: the process development for behaviour modelling of a Cyber Guided Vehicle in M2M logic,” XXII Summer School “Francesco Turco” - Industrial Systems Engineering, 2017, pp. 1-7. [Online]. Available: https://www.summerschool-aidi.it/edition-2017/cms/extra/papers/75-%20Bottani%20et%20al-with-numbers.pdf

[35] X. Chen, E. Kang, S. Shiraishi, V.M. Preciado and Z. Jiang, “Digital Behavioral Twins for Safe Connected Cars,” in Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS '18), Association for Computing Machinery, New York, NY, USA, 2018, pp. 144-153, doi: 10.1145/3239372.3239401

[36] O. Veledar, V. Damjanovic-Behrendt and G. Macher, “Digital Twins for Dependability Improvement of Autonomous Driving,” in A. Walker, R. O’Connor and R. Messnarz (eds), Systems, Software and Services Process Improvement, EuroSPI 2019, Communications in Computer and Information Science, vol. 1060, Springer, Cham., 2019, doi: 10.1007/978-3-030-28005-5_32

[37] Y. Liu, Z. Wang, K. Han, Z. Shou, P. Tiwari and J. H. L. Hansen, "Sensor Fusion of Camera and Cloud Digital Twin Information for Intelligent Vehicles," 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 2020, pp. 182-187, doi: 10.1109/IV47402.2020.9304643

[38] S. Liu, B. Yu, J. Tang and Q. Zhu, “Invited: Towards Fully Intelligent Transportation through Infrastructure-Vehicle Cooperative Autonomous Driving: Challenges and Opportunities,” 2021 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 2021, pp. 1323-1326, doi: 10.1109/DAC18074.2021.9586317

[39] Z. Wang, K. Han and P. Tiwari, “Digital Twin-Assisted Cooperative Driving at Non-Signalized Intersections,” in IEEE Transactions on Intelligent Vehicles, vol. 7, no. 2, pp. 198-209, June 2022, doi: 10.1109/TIV.2021.3100465

[40] P. Stączek, J. Pizoń, W. Danilczuk, and A. Gola, “A Digital Twin Approach for the Improvement of an Autonomous Mobile Robots (AMR’s) Operating Environment—A Case Study,” Sensors, vol. 21, no. 23, p. 7830, Nov. 2021, doi: 10.3390/s21237830

[41] F.-Y. Wang, “Parallel Control and Management for Intelligent Transportation Systems: Concepts, Architectures, and Applications,” in IEEE Transactions on Intelligent Transportation Systems, vol. 11, no. 3, pp. 630-638, Sept. 2010, doi: 10.1109/TITS.2010.2060218

[42] X. Wang, L. Li, Y. Yuan, P. Ye and F.-Y. Wang, “ACP-based social computing and parallel intelligence: Societies 5.0 and beyond,” in CAAI Transactions on Intelligence Technology, vol. 1, no. 4, 2016, pp. 377-393, doi: 10.1016/j.trit.2016.11.005

[43] T. Liu, Y. Xing, X. Tang, H. Wang, H. Yu and F. -Y. Wang, “Cyber-Physical-Social System for Parallel Driving: From Concept to Application,” in IEEE Intelligent Transportation Systems Magazine, vol. 13, no. 1, pp. 59-69, Spring 2021, doi: 10.1109/MITS.2020.3014079

[44] J. Lu, X. Wang, X. Cheng, J. Yang, O. Kwan and X. Wang, "Parallel Factories for Smart Industrial Operations: From Big AI Models to Field Foundational Models and Scenarios Engineering," in IEEE/CAA Journal of Automatica Sinica, vol. 9, no. 12, pp. 2079-2086, December 2022, doi: 10.1109/JAS.2022.106094

[45] Z. Wang, O. Zheng, L. Li, M. Abdel-Aty, C. Cruz-Neira and Z. Islam, “Towards Next Generation of Pedestrian and Connected Vehicle In-the-Loop Research: A Digital Twin Co-Simulation Framework,” in IEEE Transactions on Intelligent Vehicles, vol. 8, no. 4, pp. 2674-2683, April 2023, doi: 10.1109/TIV.2023.3250353

[46] Y. Laschinsky, K. von Neumann-Cosel, M. Gonter, C. Wegwerth, R. Dubitzky and A. Knoll, "Evaluation of an Active Safety Light using Virtual Test Drive within Vehicle in the Loop," 2010 IEEE International Conference on Industrial Technology, Via del Mar, Chile, 2010, pp. 1119-1112, doi: 10.1109/ICIT.2010.5472583

[47] H. Shikata, T. Yamashita, K. Arai, T. Nakano, K. Hatanaka and H. Fujikawa, “Digital Twin Environment to Integrate Vehicle Simulation and Physical Verification,” SEI Technical Review, vol. 88, 2019, pp. 18-21. [Online]. Available: https://global-sei.com/technology/tr/bn88/pdf/E88-04.pdf

[48] V. Dygalo, A. Keller and A. Shcherbin, “Principles of application of virtual and physical simulation technology in production of digital twin of active vehicle safety systems,” in Transportation Research Procedia, vol. 50, 2020, pp. 121-129, doi: 10.1016/j.trpro.2020.10.015

[49] D.J. Wagg, K. Worden, R.J. Barthorpe and P. Gardner, “Digital Twins: State-of-the-Art and Future Directions for Modeling and Simulation in Engineering Dynamics Applications,” ASME J. Risk Uncertainty Part B, vol. 6, no. 3, pp. 030901, Sep 2020, doi: 10.1115/1.4046739

[50] A. Rassõlkin, T. Vaimann, A. Kallaste and V. Kuts, “Digital twin for propulsion drive of autonomous electric vehicle,” 2019 IEEE 60th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), Riga, Latvia, 2019, pp. 1-4, doi: 10.1109/RTUCON48111.2019.8982326

[51] Y. Ge, Y. Wang, R. Yu, Q. Han and Y. Chen, “Demo:Research on test method of autonomous driving based on digital twin,” 2019 IEEE Vehicular Networking Conference (VNC), Los Angeles, CA, USA, 2019, pp. 1-2, doi: 10.1109/VNC48660.2019.9062813

[52] M. Szalai, B. Varga, T. Tettamanti and V. Tihanyi, “Mixed reality test environment for autonomous cars using Unity 3D and SUMO,” 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, 2020, pp. 73-78, doi: 10.1109/SAMI48414.2020.9108745

[53] B. Yu, C. Chen, J. Tang, S. Liu and J.-L. Gaudiot, “Autonomous Vehicles Digital Twin: A Practical Paradigm for Autonomous Driving System Development,” in Computer, vol. 55, no. 9, pp. 26-34, Sept. 2022, doi: 10.1109/MC.2022.3159500