Homework 1 **Student Name: Chinmay 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. 10/02/2023 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:

```
.
├── code
│   ├── base.ipynb
│   ├── common.py
│   └── models.py
├── data
└── models
```

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)

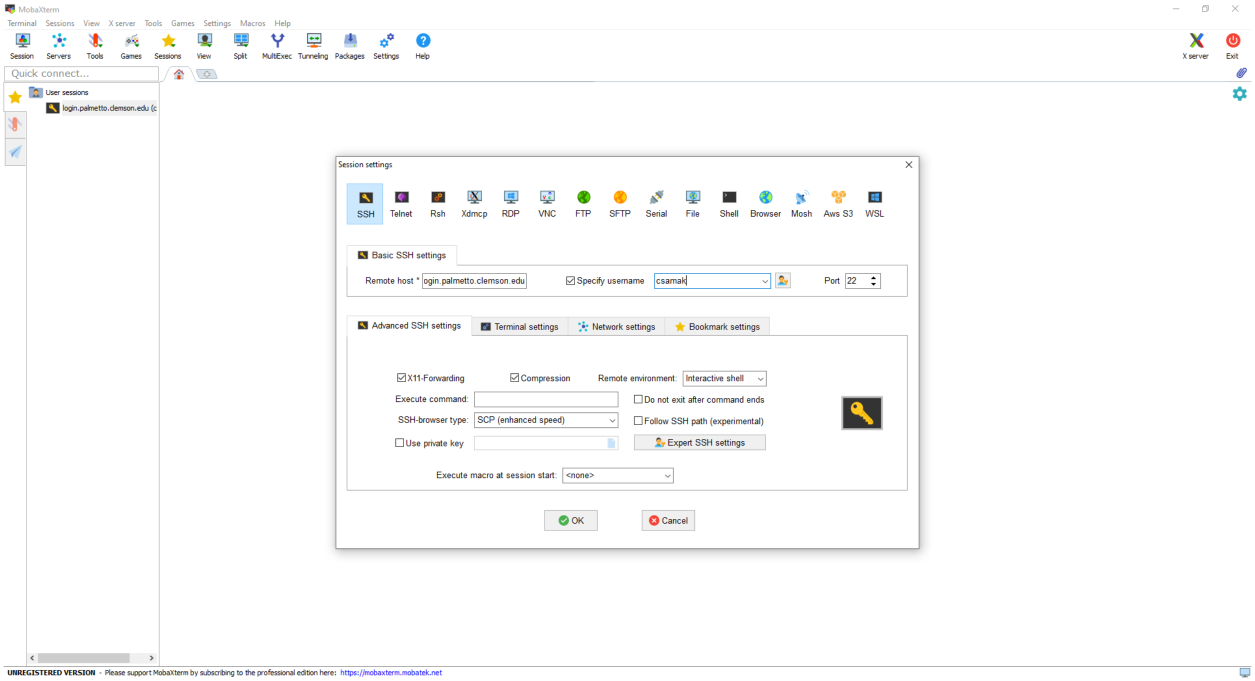


Fig. 1. MobaXterm SSH Setup for Connecting to Palmetto Cluster

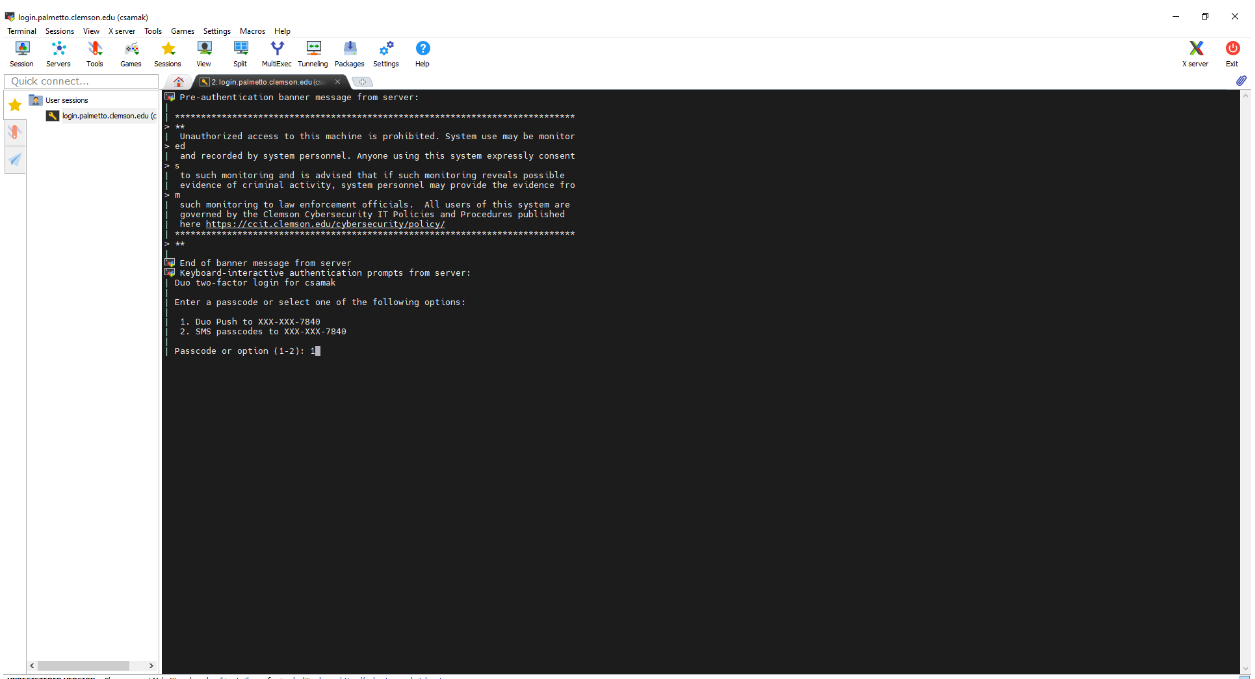


Fig. 2. Login Authentication

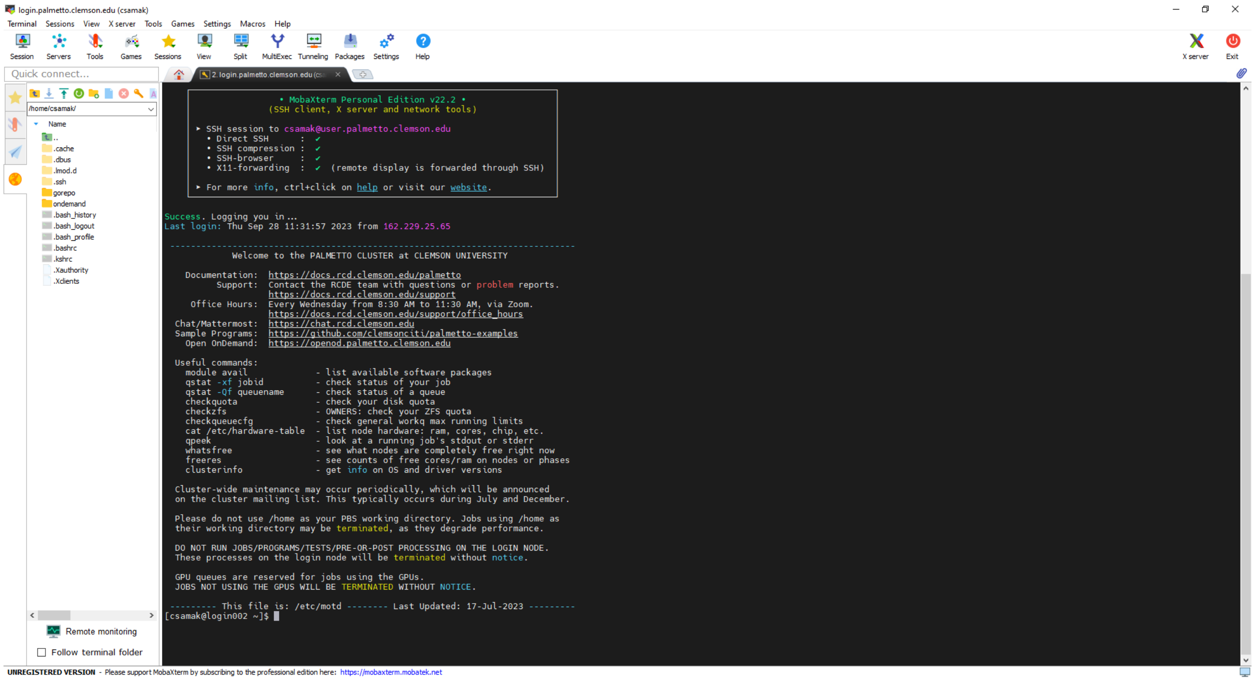


Fig. 3. Successful Login

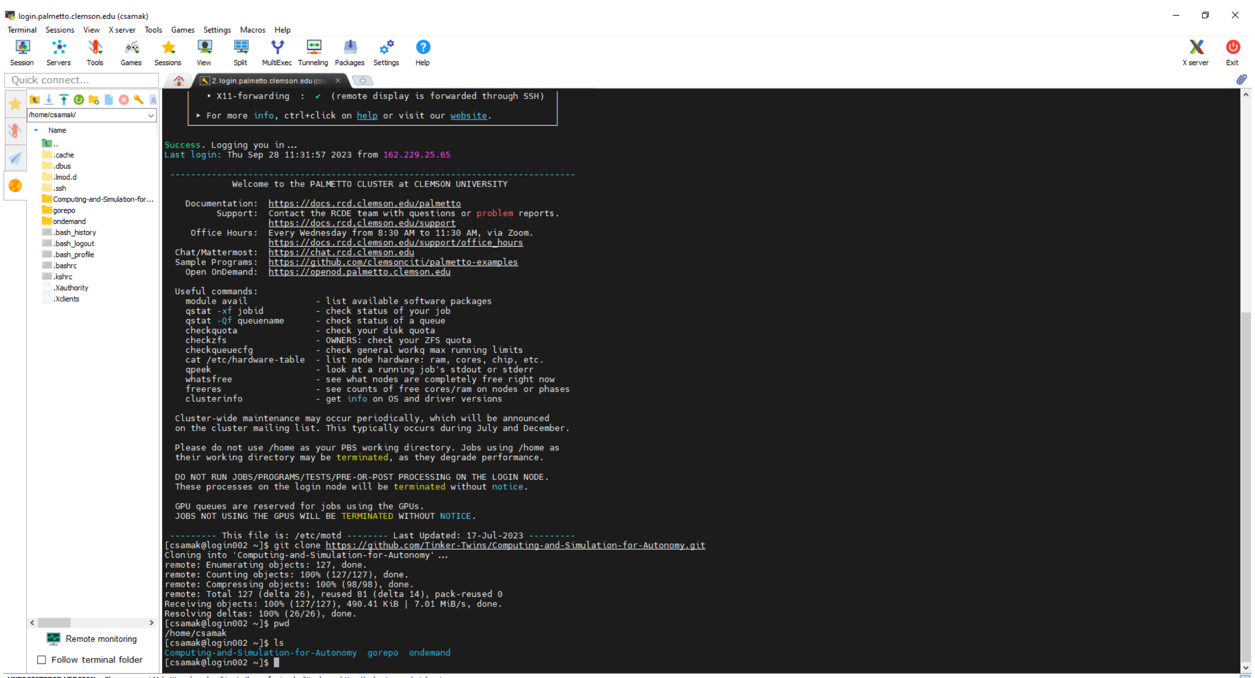


Fig. 4. Clone Course Git Repository and Verify Path to and Existence of the Cloned Repository

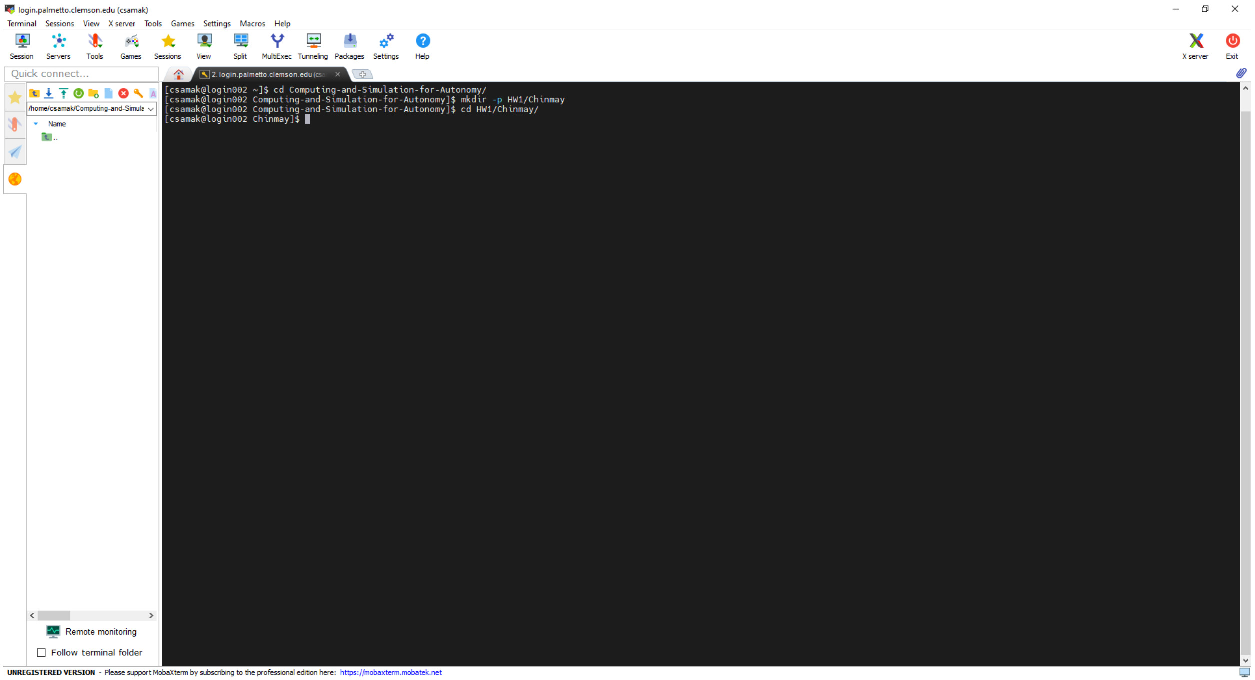


Fig. 5. Create Directory for HW1 and Make it Working Directory

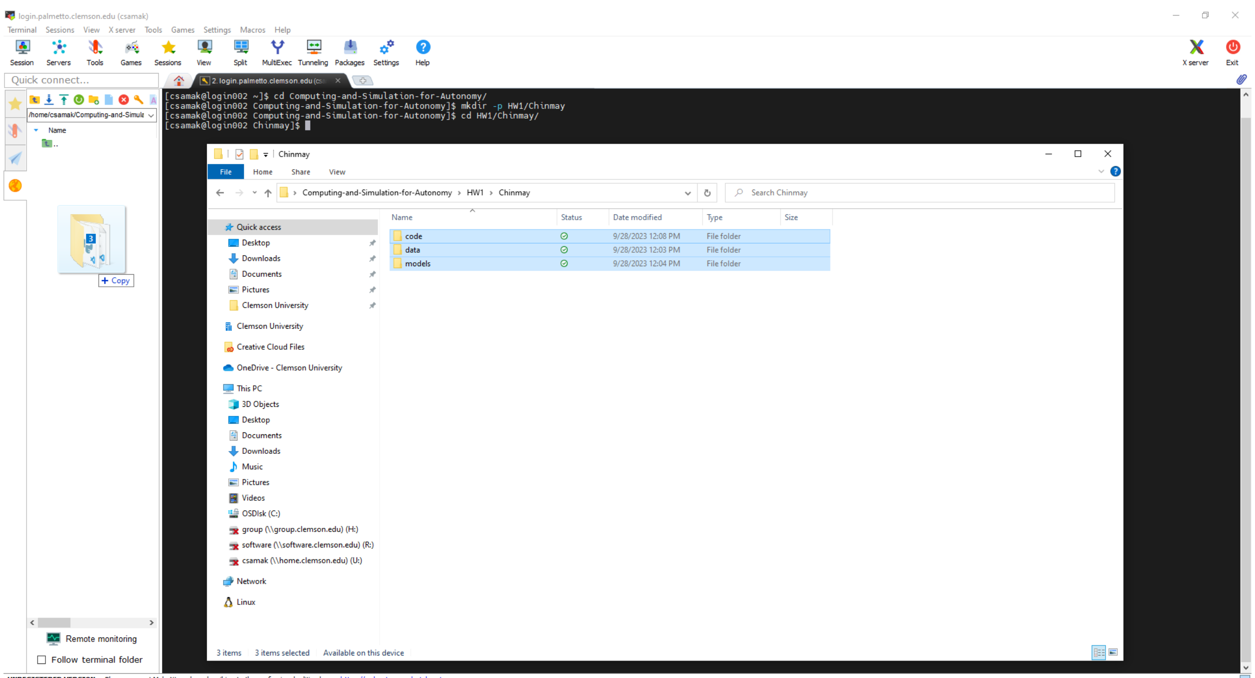


Fig. 6. Copy HW1 Baseline Files to Palmetto Cluster

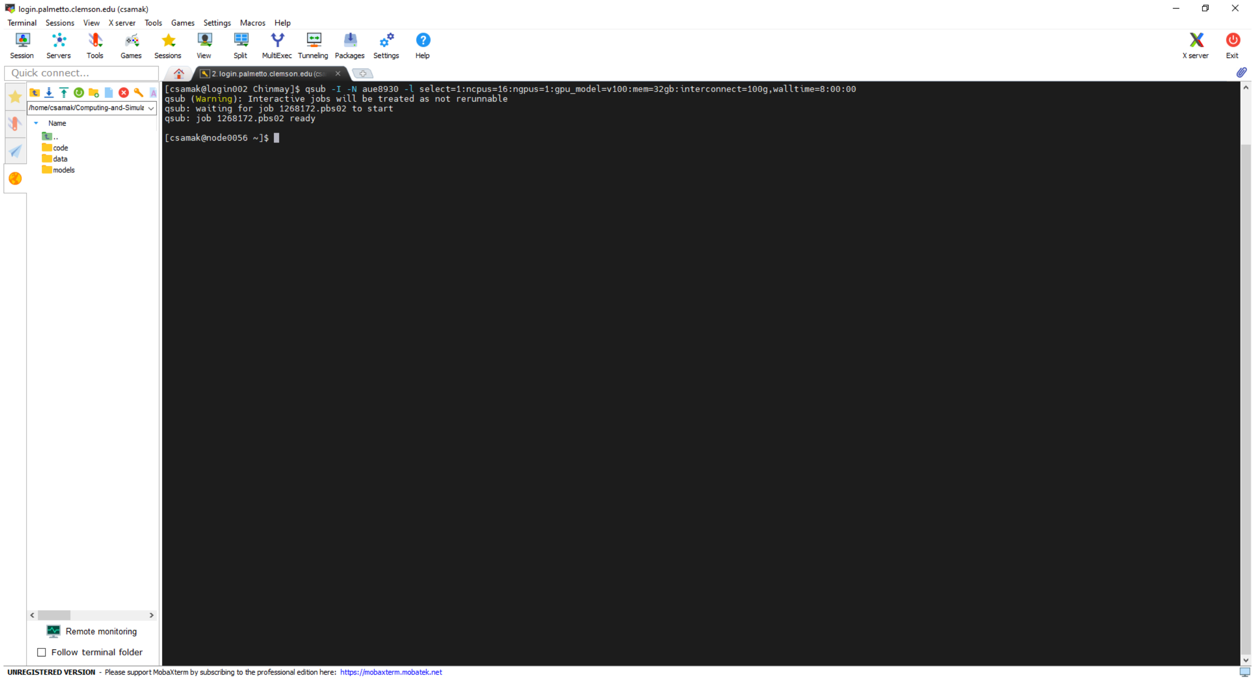


Fig. 7. Submit Interactive Job on Palmetto Cluster with name “aue8930”, 16 CPU Cores, 1 GPU (V100 Model), 32 GB RAM and 100 Gb Interconnect with 8 Hours of Wall Time

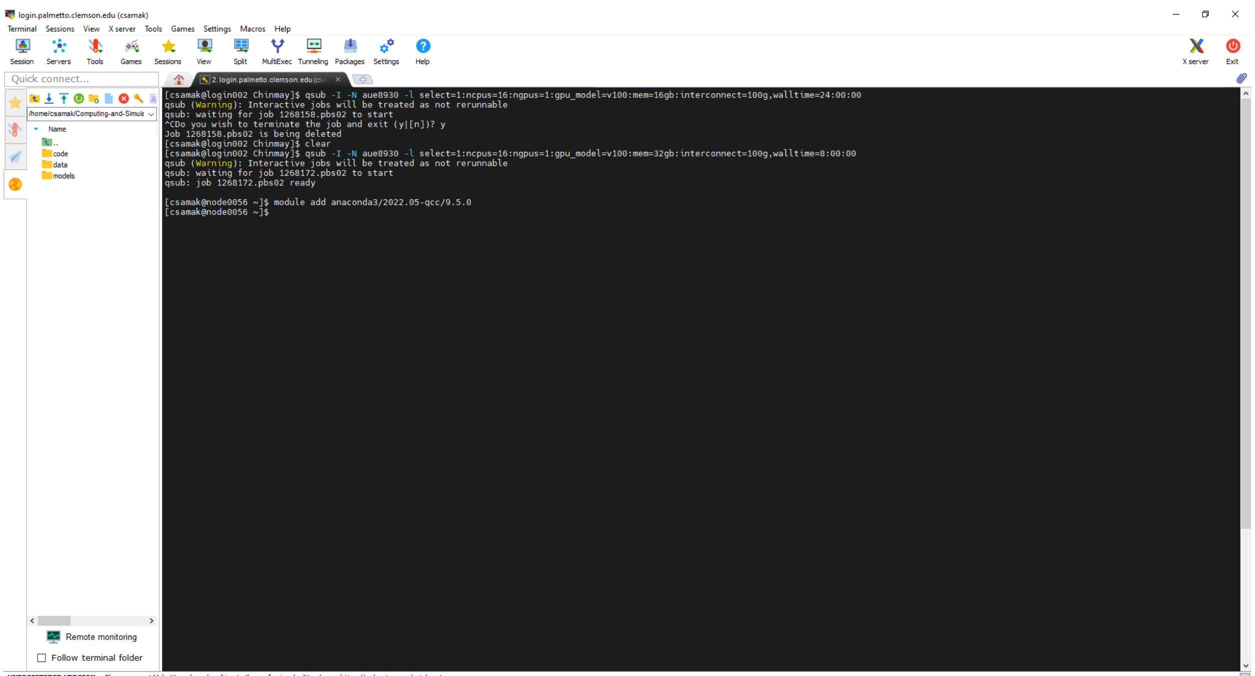


Fig. 8. Add Anaconda Module

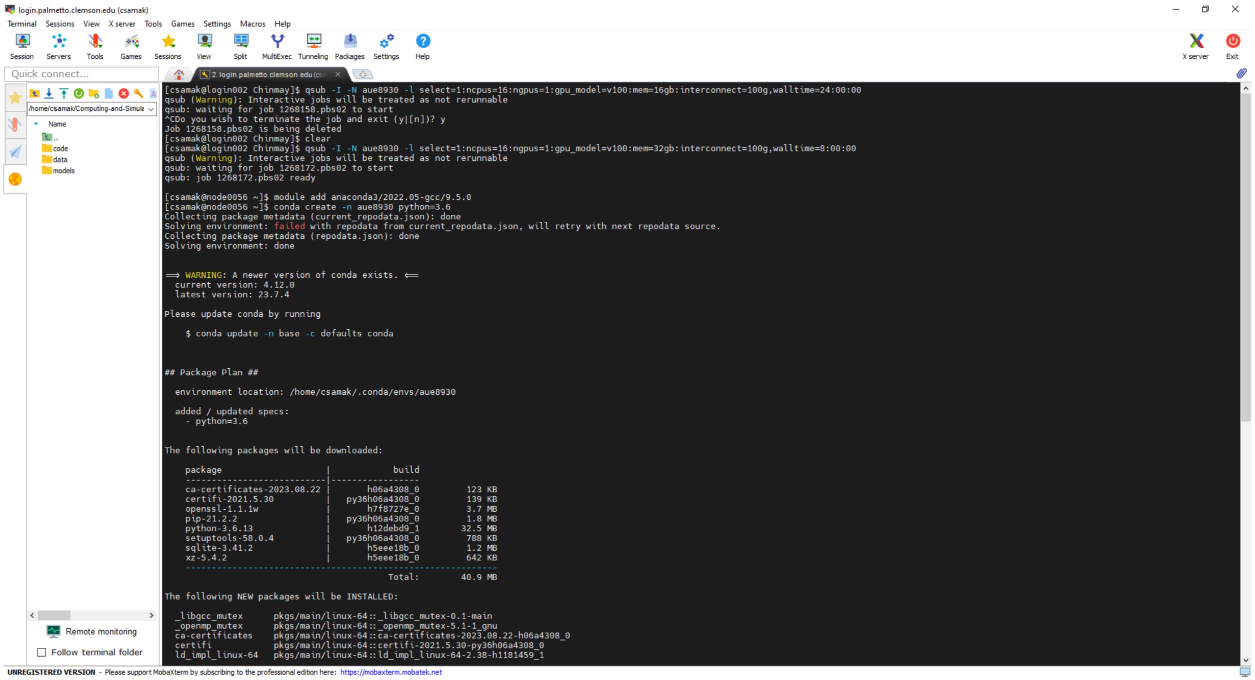


Fig. 9. Create Conda Environment

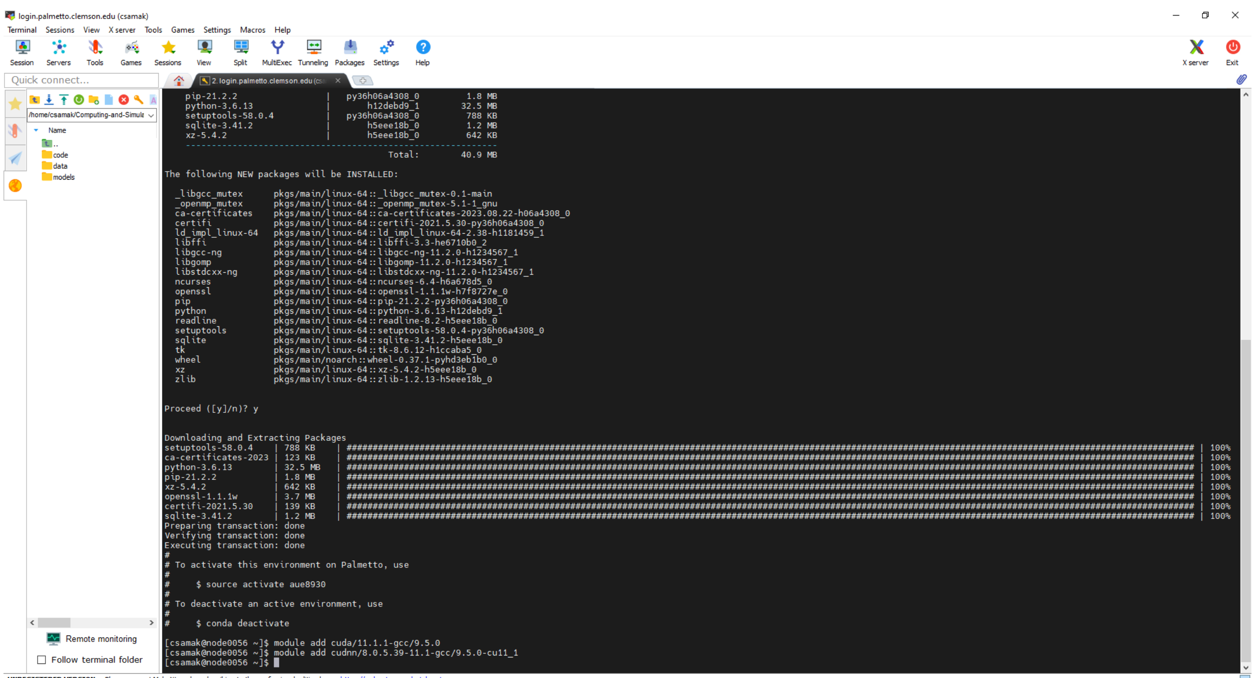


Fig. 10. Add CUDA and cuDNN Modules

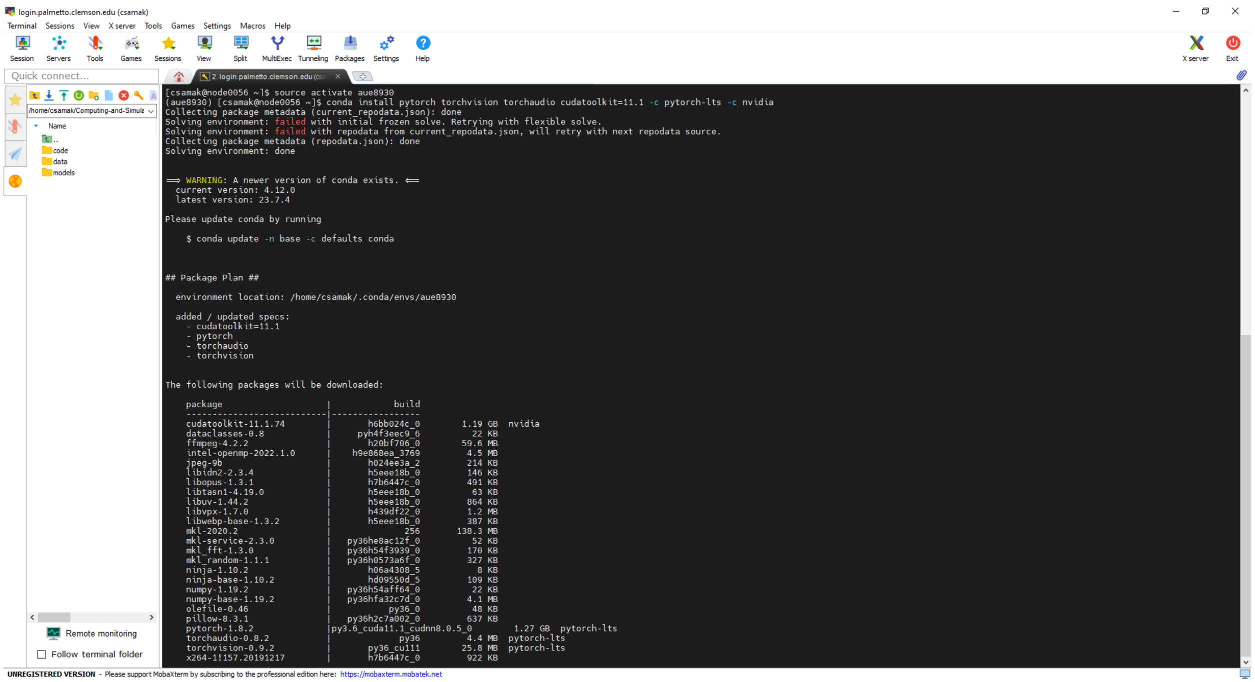


Fig. 11. Activate Conda Environment and Install Pytorch

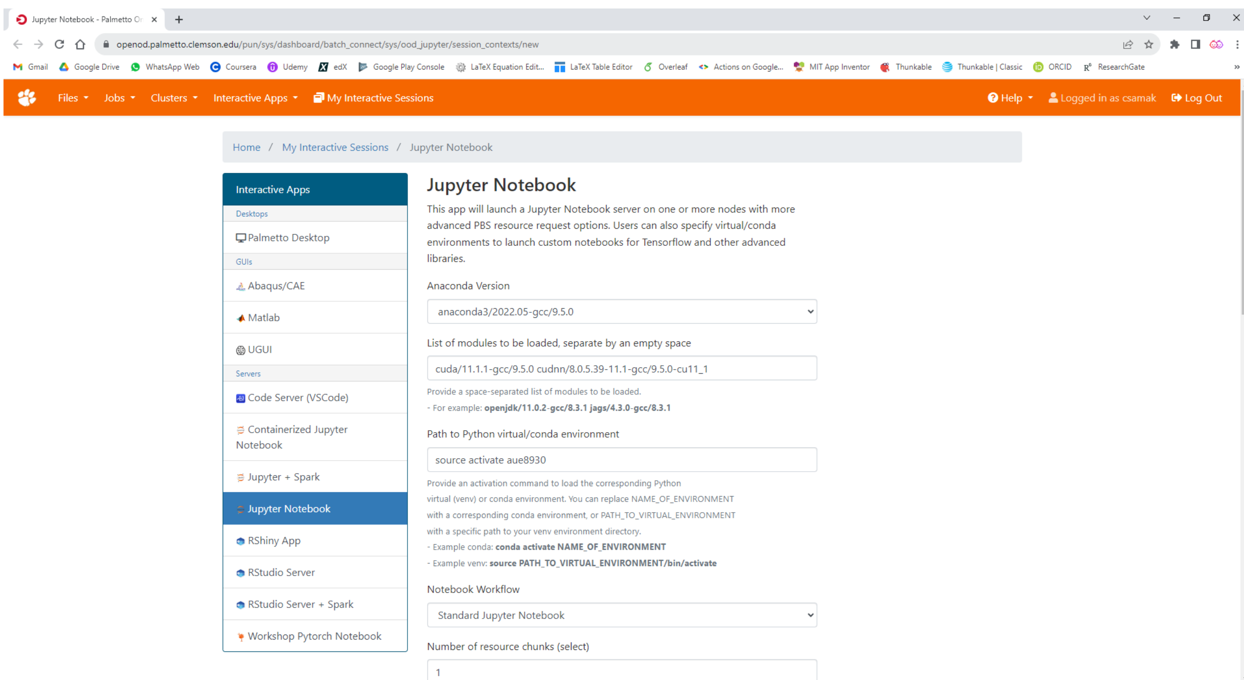


Fig. 12. Continue to Palmetto OnDemand for Launching Jupyter Notebooks Easily with Lag-Free Experience (PFA the detailed job configuration in Appendix 1 at the end of this document)

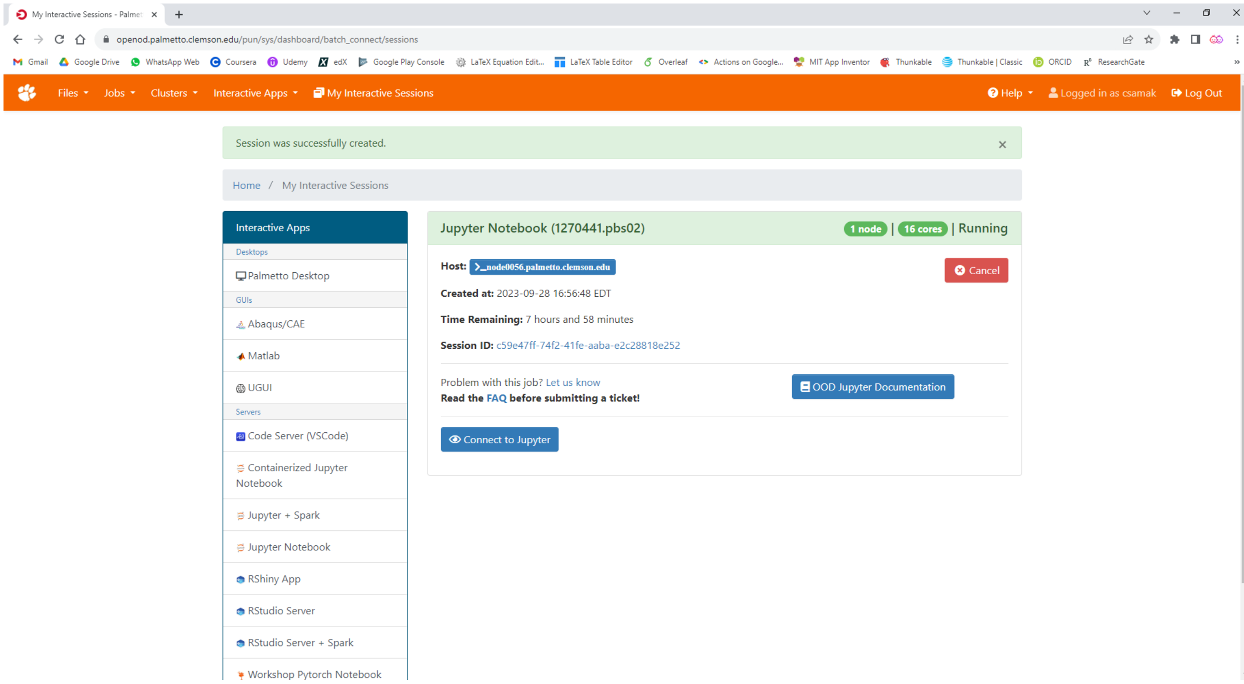


Fig. 13. Create Jupyter Notebook Session on Palmetto OnDemand (wait for the job to start "Running" and then hit "Connect to Jupyter")

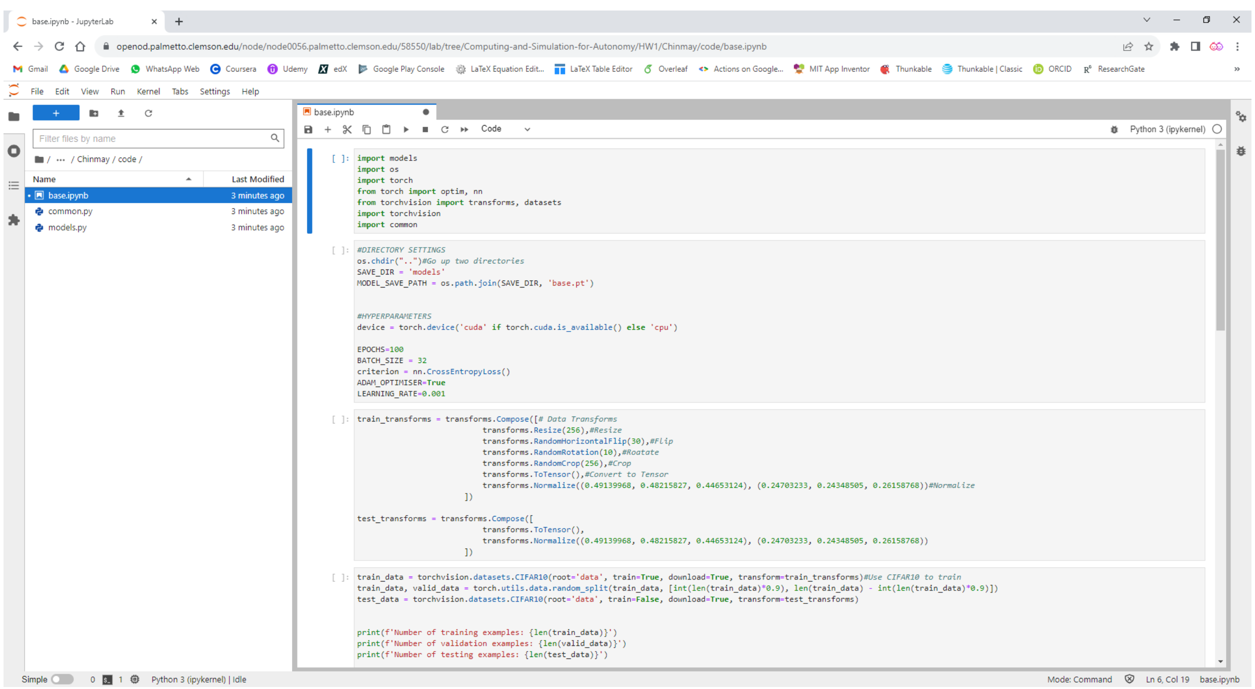


Fig. 14. Open the Baseline HW1 Code

(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.)

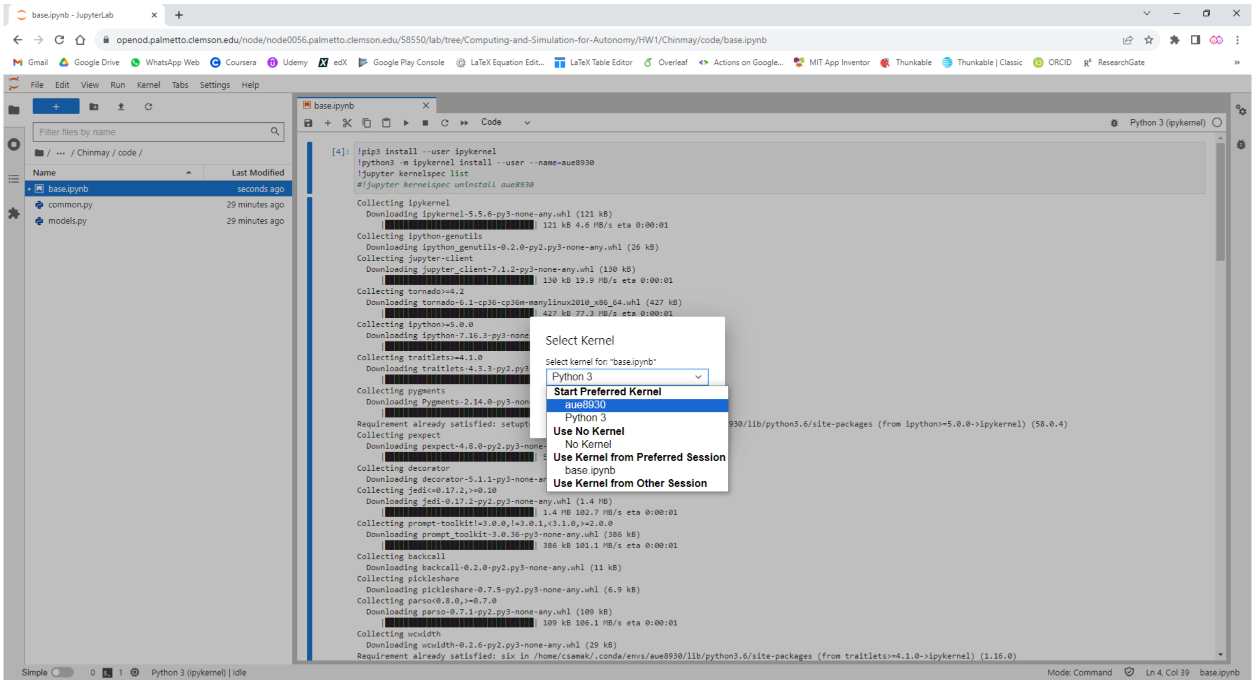


Fig. 15. Install “ipykernel”, Add “aue8930” Conda Environment and Run Jupyter Notebook in this Environment

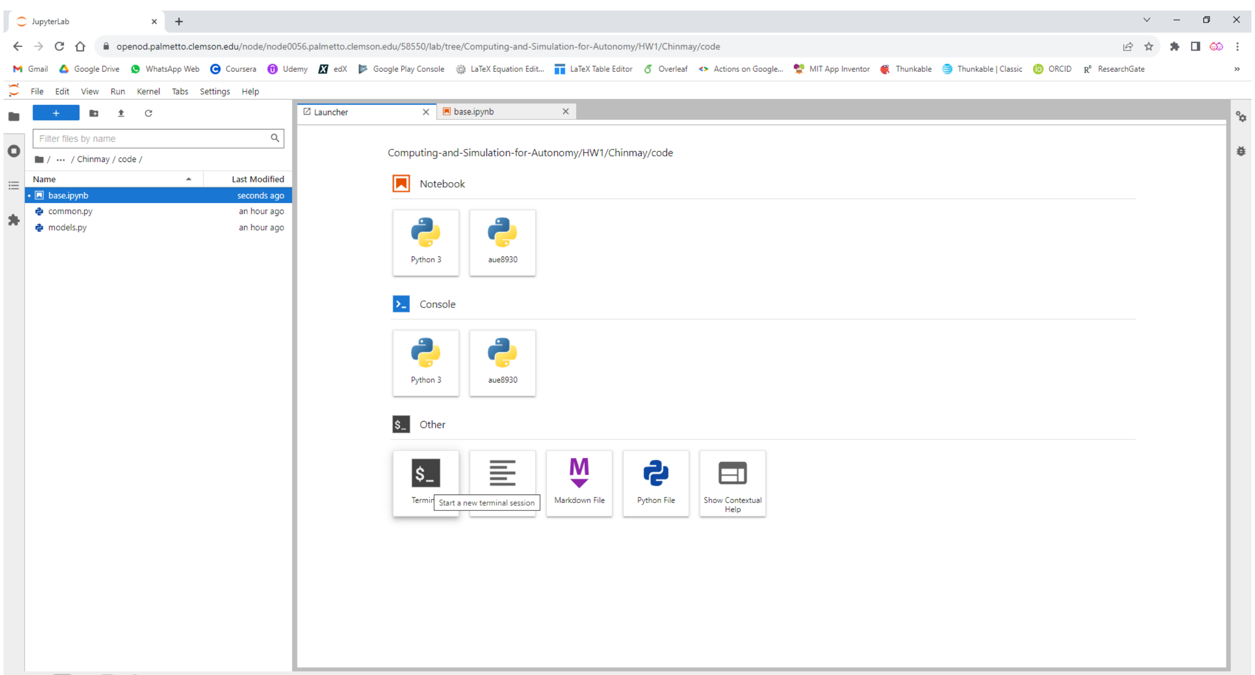


Fig. 16. Start New Terminal Session from Launcher

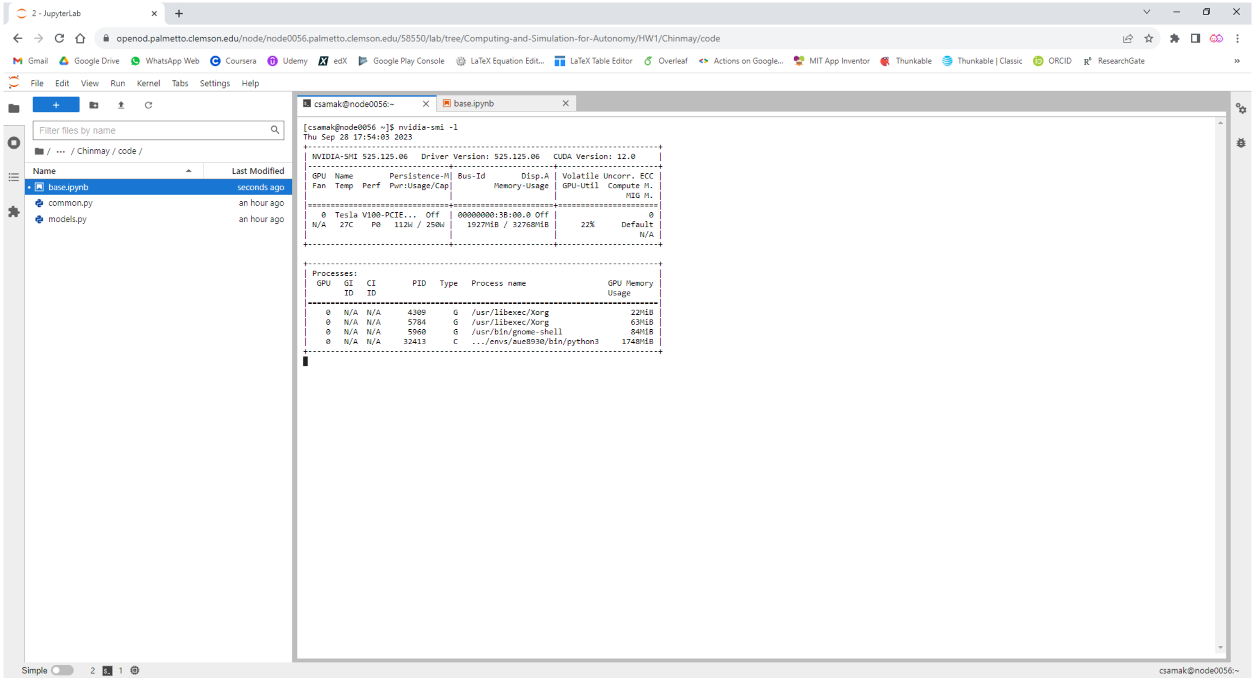


Fig. 17. Check the GPU Usage [One GPU] as the Network Trains

The GPU Usage Percentage was: 22% (GPU RAM: 1927 ÷ 32768 × 100 = 5.88%)

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

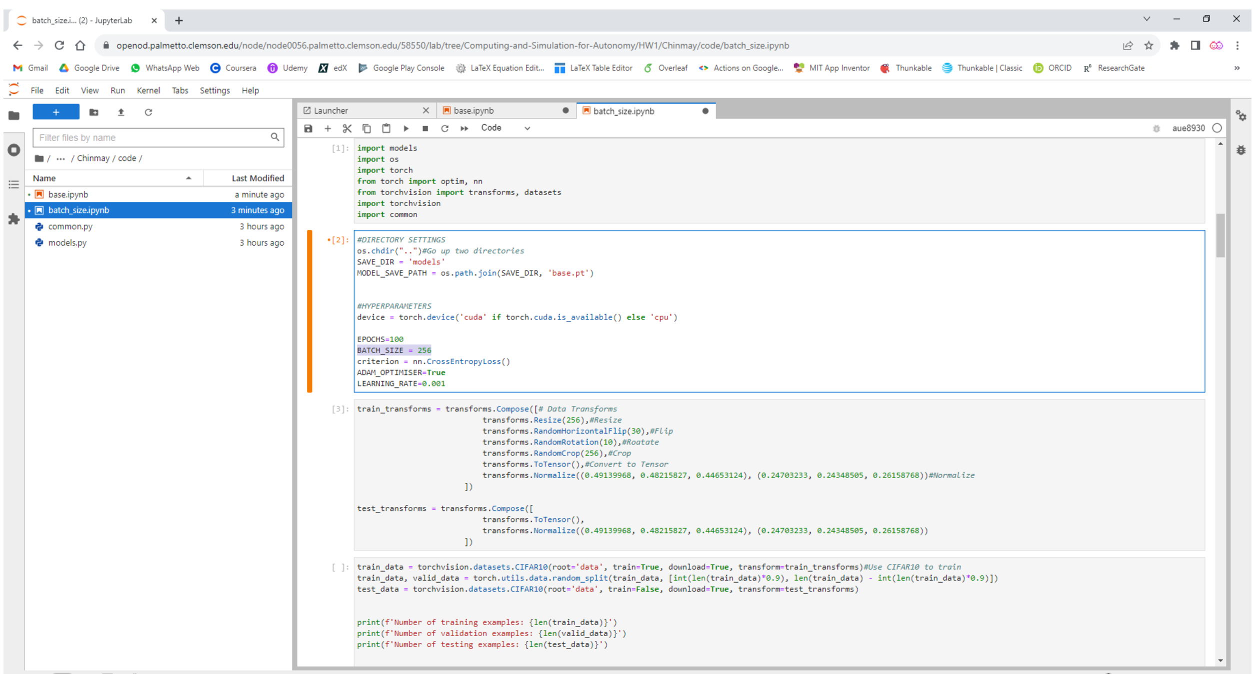


Fig. 18. Duplicate "base.ipynb" and Modify Batch Size to 256 for Faster Training

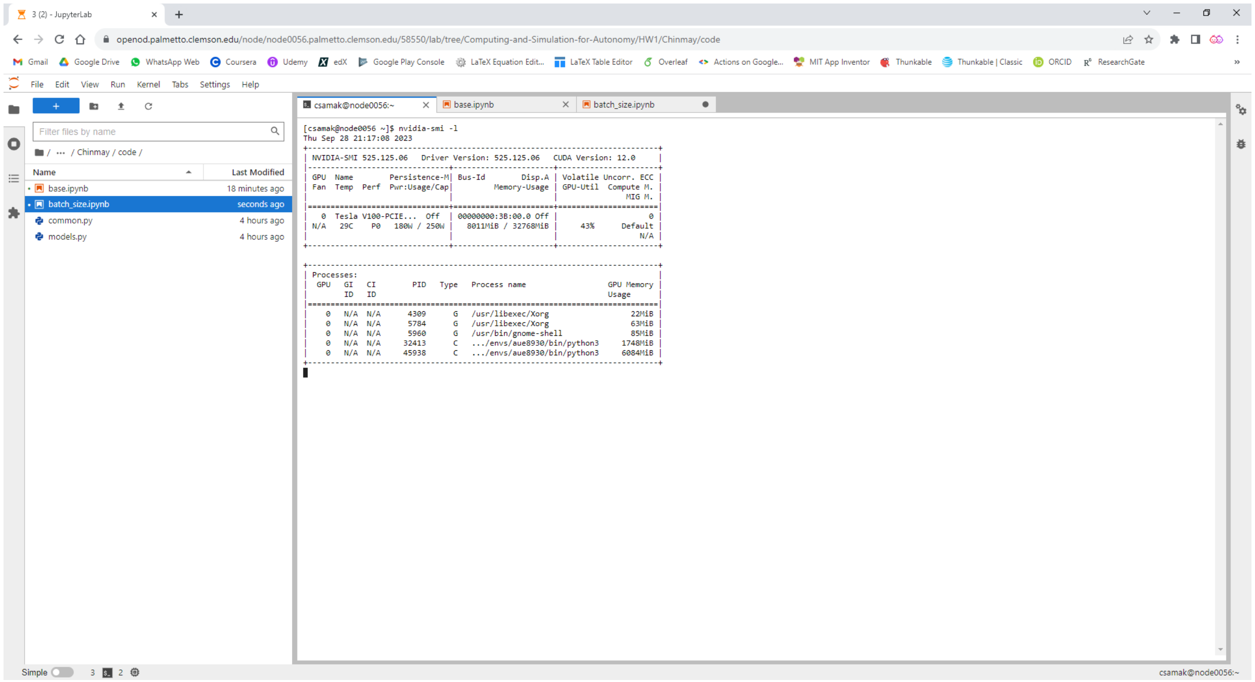


Fig. 19. Check the GPU Usage [Batch Size = 256] as the Network Trains

The GPU Usage Percentage was: 43% (GPU RAM: 8011 ÷ 32768 × 100 = 24.44%)

(4) Modify the code for better performance (use two GPUs) (10 points)

During the training, what's your GPU usage percentage? (TIPS: [reference API])

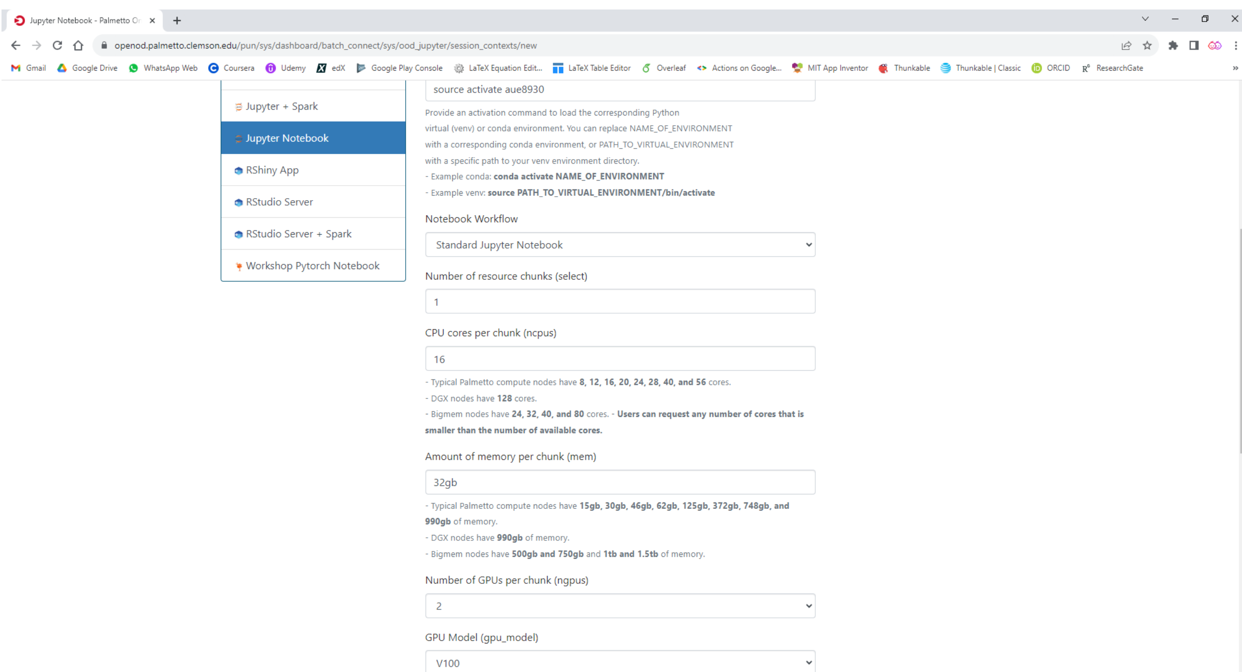


Fig. 20. Start New Palmetto OnDemand Session with 2 GPUs

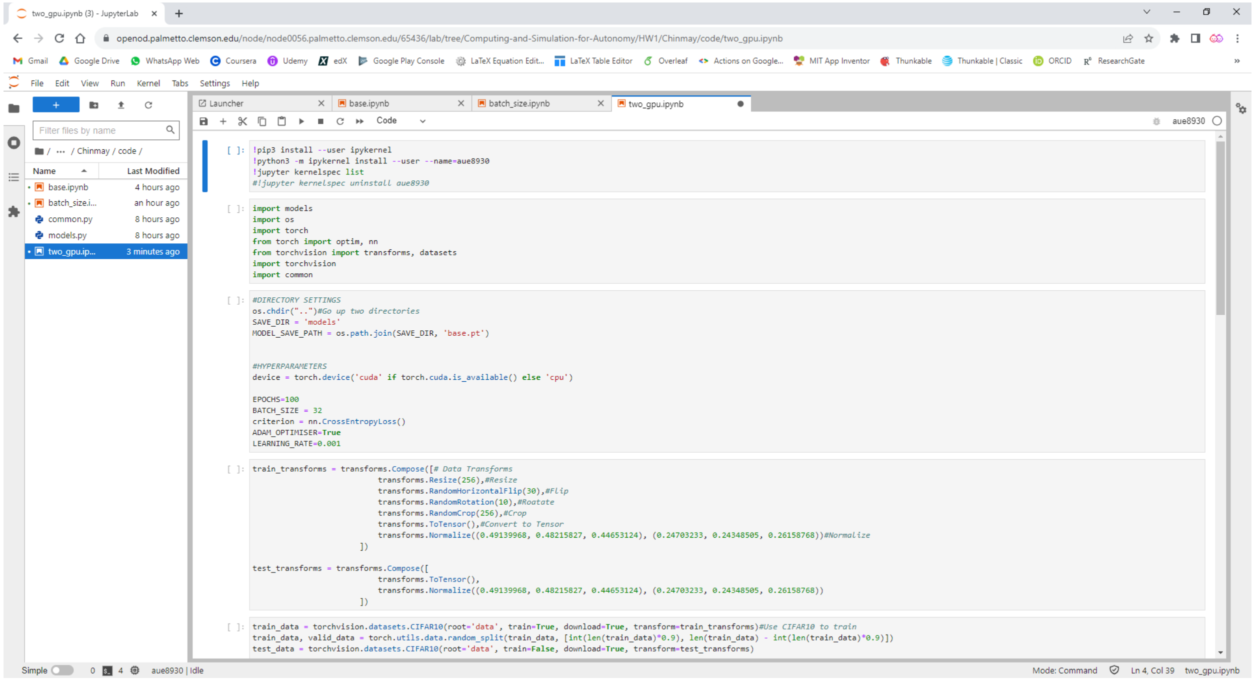


Fig. 21. Duplicate base.ipynb and Execute Training with 2 GPU Cores

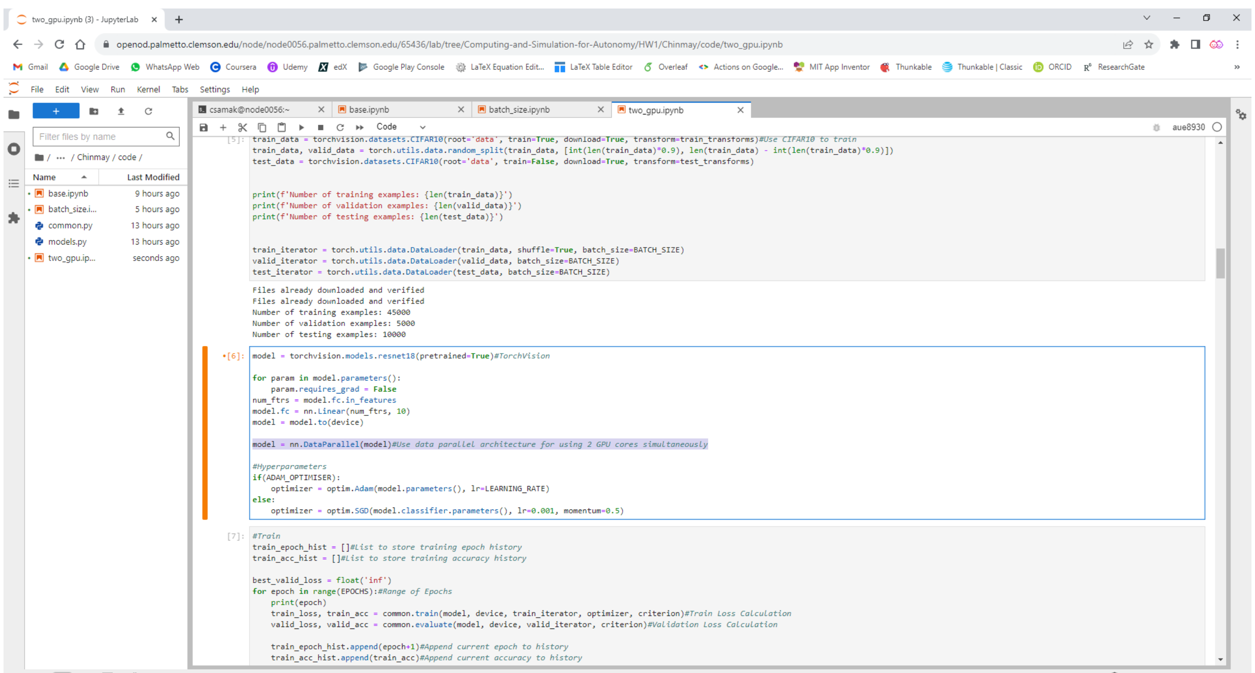


Fig. 22. Use DataParallel to Utilize 2 GPU Cores Simultaneously

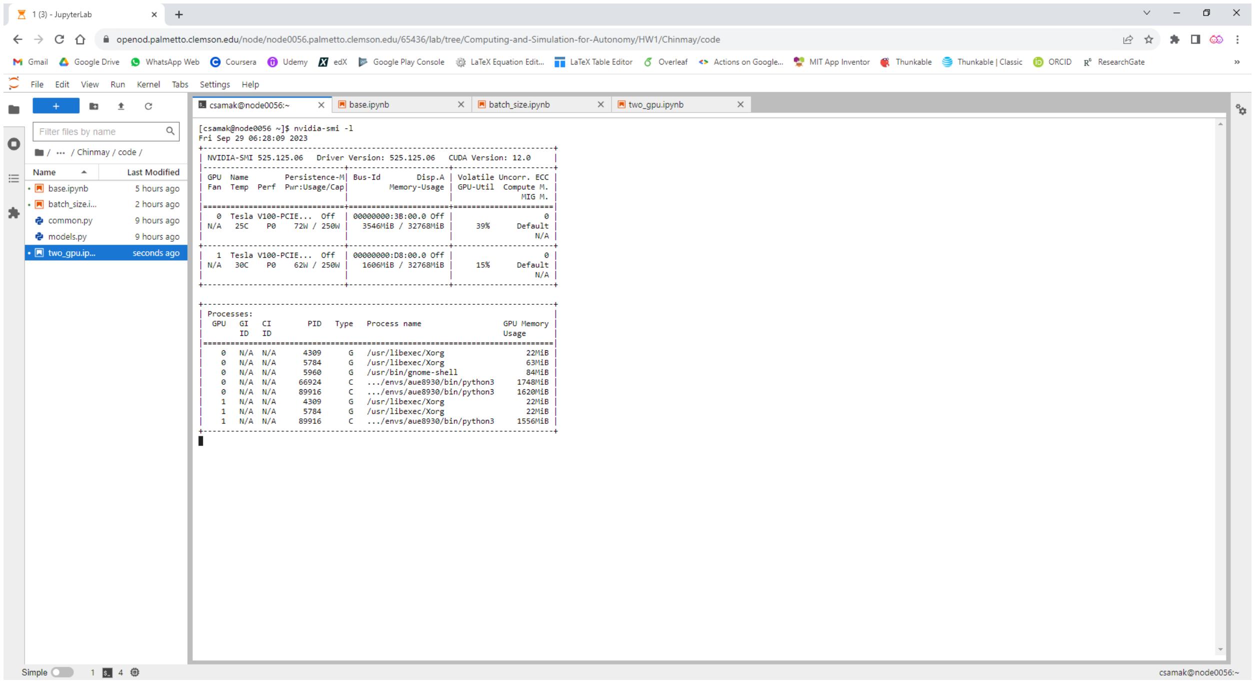


Fig. 23. Check the GPU Usage [Two GPUs] as the Network Trains

The GPU 1 Usage Percentage was: 39% (GPU RAM: 3546 ÷ 32768 × 100 = 10.82%)

The GPU 2 Usage Percentage was: 15% (GPU RAM: 1606 ÷ 32768 × 100 = 4.9%)

(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)

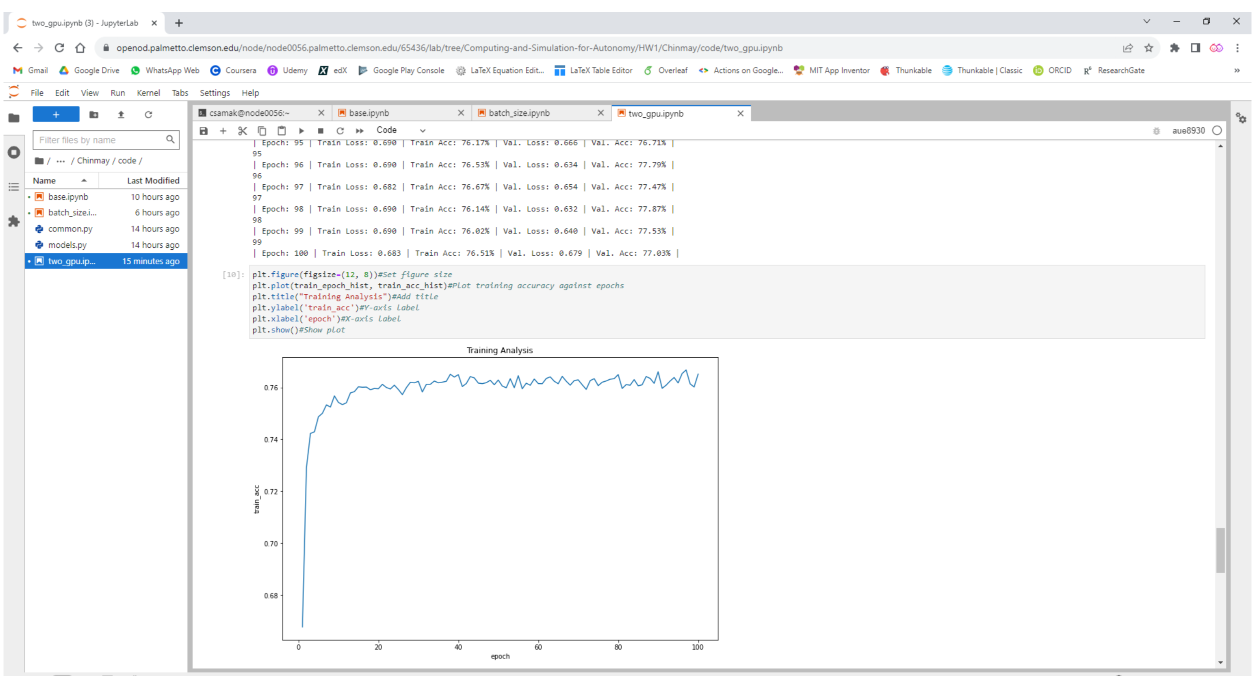


Fig. 24. Modify Code to Plot Training Accuracy Against Epochs

(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)

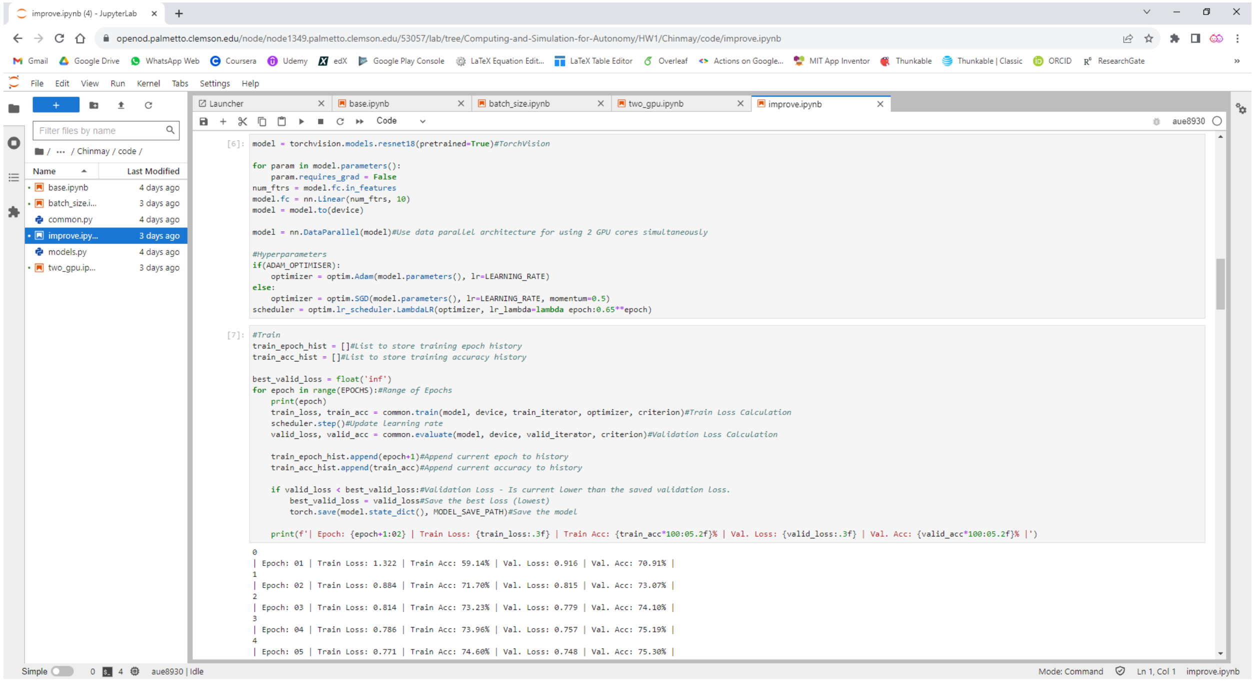


Fig. 25. Hyperparameter Tuning

The best accuracy was obtained with the following set of hyperparameters:

- Epochs = 100
- Batch Size = 256
- Optimizer = Adam
- Learning Rate = 0.001

Furthermore, following hyperparameters resulted in a more stable training in general and a much faster convergence:

- Epochs = 100
- Batch Size = 256
- Optimizer = Adam
- Learning Rate = 0.001
- Scheduler = LambdaLR (Lambda = $0.65^{\#\text{Epoch}}$)

A more thorough hyperparameter tuning could be done systematically using full-factorial or Latin hypercube sampling methods so as to cover a broader hyperparameter set with many permutations (which is beyond the scope of this assignment).

(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)

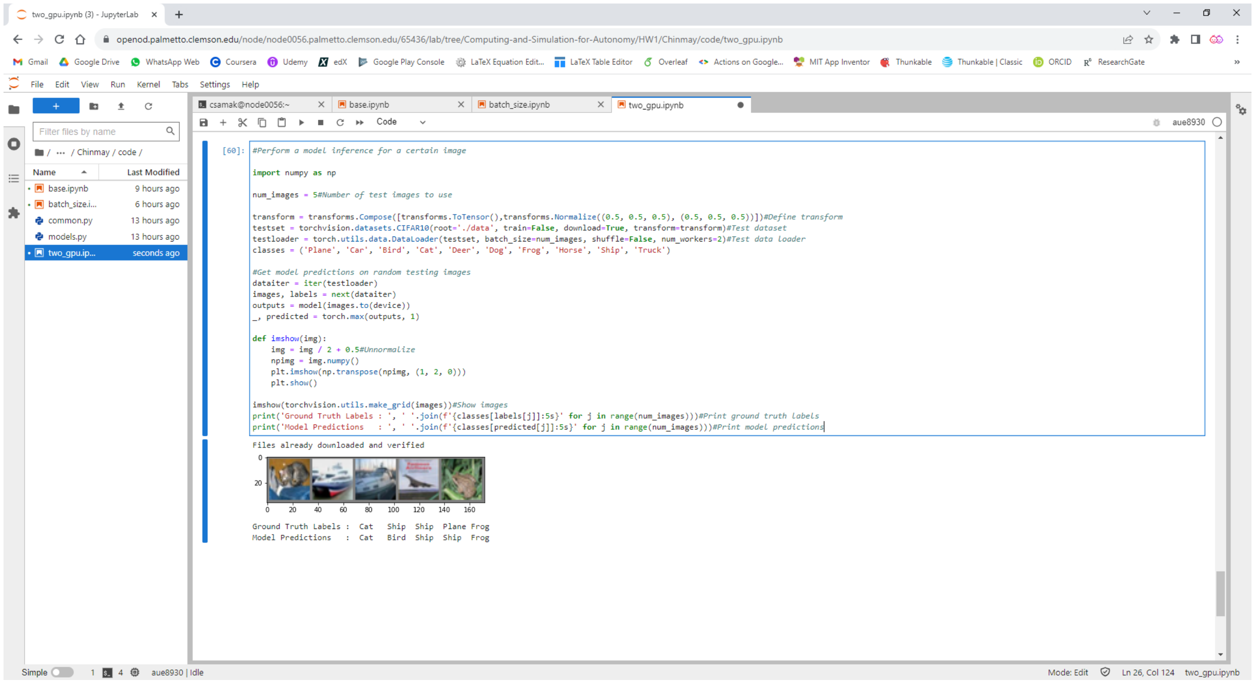


Fig. 26. Perform Model Inference for Test Images (from CIFAR-10 dataset)

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. Then, summarize for the above.

[PTO]

High Performance Computing for Bridging the Sim2Real Gap using Autonomy Oriented Digital Twins: A Survey

Author: Chinmay Samak

Introduction

Simulation-based design and verification offers various benefits such as cost-effective space (in terms of monetary, safety, spatial, temporal constraints) for prototyping/testing [1], controlled settings for variability testing [2-3], control over test case generation and execution, comprehensive corner-case analysis [4-6], safety-critical testing with social and situational variability, rapid evaluation of alternate design choices [7-9], holistic mechatronic design optimization, simulation-as-a-service (SAAS) [10-12], parallel training/testing workloads for faster execution.

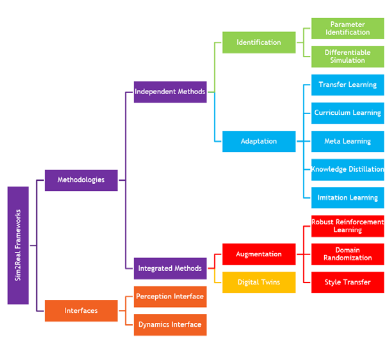
However, all these benefits are rendered moot due to the sim2sim [13] and sim2real gap [14-17]. In essence, non-repeatability within same simulation tool and non-uniformity across different simulation tools (sim2sim gap) as well as unmodeled/mismatched dynamics and perception interfaces for real and virtual worlds (sim2real gap) ultimately questions the trustworthiness of simulation-based design and verification.

High performance computing plays a crucial role in bridging the sim2real gap using autonomy-oriented digital twins. Autonomy-oriented digital twins aim to replicate real-world systems and environments in a virtual simulation, allowing for testing and development of autonomous systems without the need for physical prototypes. However, there is often a significant gap between the simulated environment and the real-world conditions, which can limit the effectiveness and applicability of the digital twin.

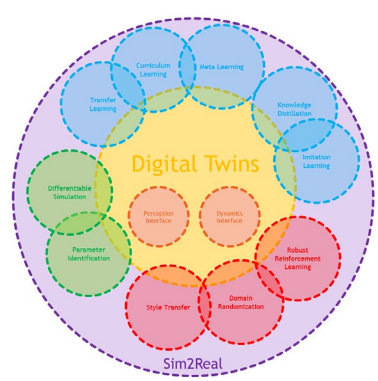
Literature Survey

This survey is primarily categorized based on the method adopted for sim2real transfer, while also highlighting the dynamics/perception interfaces.

- **Independent Sim2Real Frameworks:** These include techniques that have to be applied pre/post-development of the autonomy algorithms.
- **Integrated Sim2Real Frameworks:** These include techniques, which can be applied on-the-fly while designing/training the autonomy algorithms.



(a) Literature survey overlay and basis of classification.



(b) Overlaps of techniques to break the barriers of sim2real transfer.

One approach to bridging this gap is through system **identification**. This method involves calibrating the simulator against real-world system under test (SUT) through varying grades of parameter/system identification. More recently, the concept of differentiable simulation shows promise to update simulation parameters through gradient-based optimization.

Article/Author	Category	Dataset/Simulator	Implemented Tasks	Description	Interface
Tan et al. [18]	Parameter identification	PyBullet	Learning quadruped locomotion	Narrow reality gap by improving the simulator physics and learning robust policies.	Dynamics
Kaspar et al. [19]	Parameter identification	PyBullet	RL for peg-in-hole task using KUKA LBR iiwa	Perform system identification prior to learning for aligning the simulation environment as far as possible with the dynamics of real robot.	Dynamics
Mehta et al. [20]	Parameter identification	MuJoCo	Several tasks like pushing a block to goal, sliding a puck to goal, picking and placing a block onto another, moving a ball to goal with hand, opening a door with hand, as well as simple and difficult locomotion of half-cheetah and a humanoid	Use various methods like linear regression (LR), Bayesian optimization (BayesOpt), model-agnostic meta-learning (MAML), simulation parameter distribution optimization (SimOpt) and active domain randomization (ADR) for calibrating robotic simulators.	Dynamics
Sontakke et al. [21]	Parameter identification	MuJoCo	Walking and turning policies for buoyancy assisted legged robots	Model the nonlinear dynamics of the actuators by collecting hardware data and optimizing the simulation parameters.	Dynamics
Lee et al. [22]	Parameter identification	Custom	Simultaneous identification of intrinsic and extrinsic parameters of a laser-vision sensor	Use particle swarm optimization (PSO) for sensor model parameter estimation.	Perception
Krishna et al. [23]	Differentiable simulation	gradSim	Motion of rigid and soft body objects upon impulse input	Identification of a variety of physical parameters such as mass, friction and elasticity of rigid and soft body objects using gradient-based optimization for minimizing variation between true and estimated image/video observations.	Dynamics
Le Lidec et al. [24]	Differentiable simulation	Custom	Sliding motion as well as elastic collision of rigid body cubes	Identification of a physical parameters such as object mass, coefficient of kinetic friction and coefficient of restitution of rigid bodies using gradient-based optimization for minimizing variation between true and estimated object trajectories.	Dynamics
Heiden et al. [25]	Differentiable simulation	DiSECt	Cutting of natural soft materials such as fruits (apple) and vegetables (potato) using knife blade	Identification of simulation parameters such as vertical knife velocity, cut spring softness, cut spring stiffness, contact stiffness, contact friction and contact damping using gradient-based optimization for minimizing variation between true and estimated knife blade force.	Dynamics

Another approach to bridging this gap is through various **adaption techniques**. This involves adapting the simulation domain to match real-world data distribution using certain pre/post-processing methods such as transfer learning, curriculum learning, meta learning, knowledge distillation or imitation learning.

Article/Author	Category	Dataset/Simulator	Implemented Tasks	Description	Interface
Kim et al. [26]	Transfer learning	Kinect	End-to-end autonomous driving with lane semantic segmentation	A continuous end-to-end transfer learning approach which uses two transfer learning steps.	Perception
Akhauri et al. [27]	Transfer learning	Unity, deepDrive	Autonomous driving domain transfer	Adding the idea of robust RL to transfer learning, learning both parameters and strategies, and transferring the correspondence between the two to the execution environment.	Perception
Wu et al. [28]	Transfer learning	Blender (BlenSor, BlenderProc)	Automated disassembly of different variants of actuators in vehicle manufacturing	Pre-train the network model on synthetic data, fine-tune the network model on real-world data, and post-process semantically segmented point-cloud for predicting screw locations and orientations.	Perception
Qiao et al. [29]	Curriculum learning	PyBullet	Sequential goal tracking	Map low-fidelity (grid-world) simulation to high-fidelity (physics-based) simulation or real-world, using curriculum learning.	Dynamics
Bae et al. [30]	Curriculum learning	Isaac Sim	Construct task-independent trajectories for point-to-point motions of robot manipulator	Sim2Real transfer, augmented by curriculum learning, highlights that the robots behave in the same way in the real world as in the simulation.	Dynamics
Xiao et al. [31]	Curriculum learning	Custom	Flying quadrotor through narrow gaps	Curriculum learning for searching dynamically feasible flight trajectories with a sim2real framework that can transfer control commands to a real quadrotor without using real flight data.	Dynamics
Qin et al. [32]	Curriculum learning	V-REP (CoppeliaSim)	Gait planning of six-legged robot to adapt to complex terrain	Train a robot to safely arrive to the target point through complex terrains and use curriculum learning to speed up and optimize the training. Verify the reliability of the method in simulation platform and finally transfer the learned model to real robot.	Dynamics
Nagabandi et al. [33]	Meta learning	MuJoCo	Simulation of dynamical models for online adaptation of real scenarios	An online adaptive learning method for high-capacity dynamic models to address the simulation to reality problem.	Dynamics
Jaafra et al. [34]	Meta learning	CARLA	Autonomous driving strategy	A meta reinforcement learning approach to embedding adaptive neural network controllers on top of adaptive meta-learning.	Both
Kar et al. [35]	Meta learning	KITTI	Autonomous driving scene generation and rendering	Meta-Sim environment, where images and their corresponding realistic ground images are acquired through a graphics engine.	Perception
Arndt et al. [36]	Meta learning	MuJoCo	Hitting a hockey puck to a target using robot manipulator (KUKA LBR 4+)	Use meta learning to train a policy that can adapt to a variety of dynamic conditions and use a task-specific trajectory generation model to provide an action space that facilitates quick exploration.	Dynamics

Saputra et al. [37]	Knowledge distillation	KITTI, Malaga	Autonomous driving trajectory prediction	Learning teacher's intermediate representations through attentional imitation loss and attentional cue training methods.	Perception
Zhang et al. [38]	Knowledge distillation	KITTI, nuScenes	Point cloud map feature extraction	A knowledge distillation method based on point cloud map.	Perception
Sautier et al. [39]	Knowledge distillation	SemanticKITTI, nuScenes	3D image generation for multimodal autonomous driving	A self-supervised knowledge distillation method.	Perception
Li et al. [40]	Knowledge distillation	SemanticKITTI, nuScenes	Semantic segmentation of autonomous driving radar data	Transformer-based voxel feature encoder for robust LIDAR semantic segmentation in autonomous driving.	Perception
Zhu et al. [41]	Imitation learning	MuJoCo	Robotic manipulation for a wide variety of visuomotor tasks	Model-free deep reinforcement learning method that leverages a small amount of demonstration data to assist a reinforcement learning agent.	Both
Desai et al. [42]	Imitation learning	OpenAI Gym	Experiments in several domains with mismatched dynamics	Generative adversarial reinforced action transformation (GARAT) for grounded transfer learning.	Dynamics
Javed et al. [43]	Imitation learning	OpenAI Gym	Robotic fabric manipulation task	Novel policy gradient-style robust optimization approach, PG-BROIL, that optimizes a soft-robust objective that balances expected performance and risk.	Both
Tsinganos et al. [44]	Imitation learning	DART	Multi-stage, multi-object manipulation tasks	Two pipelines for learning a robust robot policy with sim2real: (a) imitation of solution sketches (states alone); and (b) imitation from voxel-based scene representation; and transferring of it in the physical environment.	Both

Yet another way of bridging the sim2real gap is using **augmentation methods**, which expand the simulation domain to "hopefully" match the real-world data distribution. This includes techniques such as robust reinforcement learning, domain randomization as well as style transfer.

Article/Author	Category	Dataset/Simulator	Implemented Tasks	Description	Interface
Malmir et al. [45]	Robust reinforcement learning	DART	Robotic reaching task	Disturbance-augmented Markov decision process in delayed settings to incorporate disturbance estimation in training on-policy reinforcement learning algorithms.	Dynamics
Kim et al. [46]	Robust reinforcement learning	Custom	Control of a simple pendulum	Handling parametric uncertainty and/or input disturbance of simulated vs. real plants by utilizing disturbance observer (DOB).	Dynamics
Josifovski et al. [47]	Robust reinforcement learning	Unity	Robotic reach-and-balance manipulator task	Analyze the effect of randomization: more randomization helps in sim2real transfer, yet it can also harm the ability of the algorithm to find a good policy in simulation.	Dynamics
Yue et al. [48]	Domain randomization	GTA	Semantic segmentation of autonomous driving scenarios	A new method of domain randomization and pyramid consistency is proposed to learn models with high generalization ability.	Perception
Kontes et al. [49]	Domain randomization	CARLA	ADAS obstacle-avoidance	More complex road and high-speed traffic situations are considered, and the sim2real transformation is accomplished by training	Both

				several variants of the complex problem using domain randomization.	
Pouyanfar et al. [50]	Domain randomization	Unity, KITTI	ADAS obstacle-avoidance	Static domain randomization uses real data to solve the end-to-end collision-free depth drive problem.	Perception
Zhang et al. [51]	Style transfer	Gazebo, CARLA	Visual navigation in indoor and outdoor scenarios	Generative adversarial network (GAN) with cyclic loss, semantic loss as well as shift loss for consistent style transfer.	Perception
Zhang et al. [52]	Style transfer	Cruise Morpheus Simulator	Mimic the real images as closely as possible in simulation	Utilize “approximately-paired” data that shares contextual information like camera pose, map location, scene composition and lighting, while allowing some variations in assets, textures, appearance and shapes.	Perception
Tripathy et al. [53]	Style transfer	Cityscapes, Google Maps	Map input label maps to realistic images (and vice versa) for driving scenes and aerial maps	General purpose image-to-image translation model that can utilize both paired and unpaired training data simultaneously.	Perception
Bewley et al. [54]	Style transfer	Custom (procedural generation)	Visually-aided lane-following autonomous driving	Learning to translate between simulated and real-world imagery, while jointly learning a control policy from this common latent space using labels from an expert driver in simulation.	Both

Finally, digital twins have a great potential of applying any or all of the aforementioned techniques before/after/while developing the autonomy algorithms and behaviors.

Article/Author	**Category**	**Dataset/Simulator**	**Implemented Tasks**	**Description**	**Interface**
Xiong et al. [55]	Identification + augmentation	Unity	Autonomous vehicle in a manned vehicle following scenario with V2V communication	A new digital twin framework of autonomous vehicles consisting of physical entity components, virtual simulation components, and simulation evaluation components is proposed.	Dynamics
Voogd et al. [56]	Augmentation + adaptation	Simcenter Prescan, Simcenter Amesim	Autonomous vehicle in a manned vehicle following scenario with V2V communication	Combining virtual and real-world data to train a path following DRL agent for an autonomous electric vehicle.	Dynamics
Allamaa et al. [57]	Augmentation + adaptation	Simcenter Amesim	Nonlinear model predictive control for autonomous vehicle	A new method combining adaptation and augmentation in an online setting for optimizing a nonlinear model predictive control framework for autonomous vehicles saving tedious time and labor consuming tuning.	Dynamics

In addition to these approaches, there are advancements in information and communication technologies (ICT) that play a crucial role in implementing service-oriented digital twins. Groshev et al. [58] highlight the importance of ICT advancements such as edge computing, network function virtualization (NFV), and 5G in satisfying the required key performance indicators (KPIs) of latency, reliability, bandwidth, and more. These technologies enable the efficient and reliable operation of digital twins, enhancing their performance and applicability.

Furthermore, Lu et al. [59] propose the use of low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks. This approach leverages the power of edge computing and blockchain technology to enable efficient and secure collaboration between digital twins in a distributed network. By reducing latency and ensuring data integrity, this approach enhances the performance and reliability of digital twins in complex control algorithms and mental models.

Conclusion

In summary, high-performance computing plays a crucial role in bridging the sim2real gap in autonomy-oriented digital twins. Techniques such as identification, adaptation, augmentation and digital twinning enable the transfer of control and behavior from simulation to the real world. Advancements in information and communication technologies, such as edge computing and network function virtualization, enhance the performance and reliability of digital twins. Additionally, low-latency federated learning and blockchain technology enable efficient collaboration between digital twins in distributed networks. These advancements in high performance computing contribute to the development and application of autonomy-oriented digital twins.

References

[1] HeeSun Choi, et al., "On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward," Proceedings of the National Academy of Sciences (PNAS), vol. 118, no. 1, pp. e1907856118, doi: 10.1073/pnas.1907856118

[2] H. Kagalwala, S. Srivastava, M. Venkatesan, S. Srinivasan and V. Krovi, "Implementation Methodologies for Simulation as a Service (SaaS) to Develop ADAS Applications," SAE Int. J. Adv. & Curr. Prac. in Mobility 3(4):2123-2135, 2021, doi: 10.4271/2021-01-0116

[3] Y. Koroglu and F. Wotawa, "Towards a Review on Simulated ADAS/AD Testing," 2023 IEEE/ACM International Conference on Automation of Software Test (AST), Melbourne, Australia, 2023, pp. 112-122, doi: 10.1109/AST58925.2023.00015

[4] H. Sun, S. Feng, X. Yan and H. Liu, "Corner Case Generation and Analysis for Safety Assessment of Autonomous Vehicles," in Transportation Research Record, vol. 2675, no. 11, pp. 587-600, 2021, doi: 10.1177/03611981211018697

[5] S. Feng, et al., "Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment," in Nat. Commun., vol. 12, no. 748, 2021, doi: 10.1038/s41467-021-21007-8

[6] S. Guneshka, "Ontology-based corner case scenario simulation for autonomous driving," Abschlussarbeit - Bachelor, Karlsruher Institut für Technologie (KIT), 2022, doi: 10.5445/IR/1000144811

[7] P. He, C.A. Mader, J.R.R.A. Martins, K.J. Maki, "An aerodynamic design optimization framework using a discrete adjoint approach with OpenFOAM," Computers & Fluids, vol. 168, pp. 285-303, 2018, doi: 10.1016/j.compfluid.2018.04.012

[8] Ford Inc., "MAKE WAY FOR HOLOGRAMS: NEW MIXED REALITY TECHNOLOGY MEETS CAR DESIGN AS FORD TESTS MICROSOFT HOLOLENS GLOBALLY," Ford Media Center, Sep 2017. [Online]. Available: https://media.ford.com/content/fordmedia/fna/us/en/news/2017/09/21/ford-tests-microsoft-hololens-globally.html

[9] Toyota Research Institute (TRI), "Toyota Research Institute Unveils New Generative AI Technique for Vehicle Design," in Toyota Newsroom, Jun 2023. [Online]. Available: https://pressroom.toyota.com/toyota-research-institute-unveils-new-generative-ai-technique-for-vehicle-design/

[10] [19] N. Rudin, D. Hoeller, P. Reist and M. Hutter, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning," in Proceedings of the 5th Conference on Robot Learning, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu and G. Neumann (eds), vol. 164. PMLR, 8-11 Nov 2021, pp. 91-100. [Online]. Available: https://proceedings.mlr.press/v164/rudin22a.html

[11] V. Makoviychuk, et al., "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning," arXiv Preprint, 2021, doi: 10.48550/arXiv.2108.10470

[12] A. Nair, et al., "Massively Parallel Methods for Deep Reinforcement Learning," arXiv Preprint, 2015, doi: 10.48550/arXiv.1507.04296

[13] A. Farley, J. Wang, J.A. Marshall, "How to pick a mobile robot simulator: A quantitative comparison of CoppeliaSim, Gazebo, MORSE and Webots with a focus on accuracy of motion," in Simulation Modelling Practice and Theory, vol. 120(102629), 2022, doi: 10.1016/j.simpat.2022.102629

[14] S. Höfer, et al., “Perspectives on Sim2Real Transfer for Robotics: A Summary of the R:SS 2020 Workshop,” arXiv Preprint, 2020, doi: 10.48550/arXiv.2012.03806

[15] S. Höfer et al., “Sim2Real in Robotics and Automation: Applications and Challenges,” in IEEE Transactions on Automation Science and Engineering, vol. 18, no. 2, pp. 398-400, April 2021, doi: 10.1109/TASE.2021.3064065

[16] H. Ju, et al., “Transferring policy of deep reinforcement learning from simulation to reality for robotics,” in Nat. Mach. Intell., vol. 4, pp. 1077-1087, 2022, doi: 10.1038/s42256-022-00573-6

[17] Q.H. Miao, Y.S. Lv, M. Huang, X. Wang, and F.-Y. Wang, “Parallel learning: Overview and perspective for computational learning across Syn2Real and Sim2Real,” IEEE/CAA J. Autom. Sinica, vol. 10, no. 3, pp. 603–631, 2023, doi: 10.1109/JAS.2023.123375

[18] J. Tan, et al., “Sim-to-Real: Learning Agile Locomotion For Quadruped Robots,” in Proceedings of Robotics: Science and Systems, Jun 2018, Pittsburgh, PA, USA, doi: 10.15607/RSS.2018.XIV.010

[19] M. Kaspar, J. D. Muñoz Osorio and J. Bock, “Sim2Real Transfer for Reinforcement Learning without Dynamics Randomization,” 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 4383-4388, doi: 10.1109/IROS45743.2020.9341260

[20] B. Mehta, A. Handa, D. Fox and F. Ramos, “A User’s Guide to Calibrating Robotic Simulators” Proceedings of the 2020 Conference on Robot Learning, in Proceedings of Machine Learning Research, 2021, vol. 155, pp. 1326-1340. [Online]. Available: https://proceedings.mlr.press/v155/mehta21a.html

[21] N. Sontakke, H. Chae, S. Lee, T. Huang, D.W. Hong and S. Ha, “Residual Physics Learning and System Identification for Sim-to-real Transfer of Policies on Buoyancy Assisted Legged Robots,” arXiv Preprint, 2023, doi: 10.48550/arXiv.2303.09597

[22] J. K. Lee, K. Kim, Y. Lee, and T. Jeong, “Simultaneous Intrinsic and Extrinsic Parameter Identification of a Hand-Mounted Laser-Vision Sensor,” Sensors, vol. 11, no. 9, pp. 8751–8768, Sep. 2011, doi: 10.3390/s110908751

[23] J. Krishna, et al., “gradSim: Differentiable simulation for system identification and visuomotor control,” in International Conference on Learning Representations (ICLR), 2021. [Online]. Available: https://openreview.net/forum?id=c_E8kFWfhp0

[24] Q. Le Lidec, I. Kalevatykh, I. Laptev, C. Schmid and J. Carpentier, “Differentiable Simulation for Physical System Identification,” in IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 3413-3420, April 2021, doi: 10.1109/LRA.2021.3062323

[25] E. Heiden, et al., “DiSECt: a differentiable simulator for parameter inference and control in robotic cutting,” in Auton. Robot., vol. 47, pp. 549-578, 2023, doi: 10.1007/s10514-023-10094-9

[26] J. Kim and C. Park, “End-To-End Ego Lane Estimation Based on Sequential Transfer Learning for Self-Driving Cars,” 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 2017, pp. 1194-1202, doi: 10.1109/CVPRW.2017.158

[27] S. Akhauri, L. Y. Zheng and M. C. Lin, “Enhanced Transfer Learning for Autonomous Driving with Systematic Accident Simulation,” 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 5986-5993, doi: 10.1109/IROS45743.2020.9341538

[28] C. Wu, X. Bi, J. Pfrommer, A. Cebulla, S. Mangold and J. Beyerer, “Sim2real Transfer Learning for Point Cloud Segmentation: An Industrial Application Case on Autonomous Disassembly,” 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 2023, pp. 4520-4529, doi: 10.1109/WACV56688.2023.00451

[29] Y. Shukla, C. Thierauf, R. Hosseini, G. Tatiya and J. Sinapov, “ACuTE: Automatic Curriculum Transfer from Simple to Complex Environments,” in Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS '22), International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, May 2022, pp. 1192-1200, doi: 10.5555/3535850.3535983

[30] L. Væhrens, D. D. Álvarez, U. Berger and S. Bøgh, “Learning Task-independent Joint Control for Robotic Manipulators with Reinforcement Learning and Curriculum Learning,” 2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA), Nassau, Bahamas, 2022, pp. 1250-1257, doi: 10.1109/ICMLA55696.2022.00201

[31] C. Xiao, P. Lu and Q. He, “Flying Through a Narrow Gap Using End-to-End Deep Reinforcement Learning Augmented With Curriculum Learning and Sim2Real,” in IEEE Transactions on Neural Networks and Learning Systems, vol. 34, no. 5, pp. 2701-2708, May 2023, doi: 10.1109/TNNLS.2021.3107742

[32] B. Qin, Y. Gao and Y. Bai, “Sim-to-real: Six-legged Robot Control with Deep Reinforcement Learning and Curriculum Learning,” 2019 4th International Conference on Robotics and Automation Engineering (ICRAE), Singapore, 2019, pp. 1-5, doi: 10.1109/ICRAE48301.2019.9043822

[33] A. Nagabandi, et al., “Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning,” International Conference on Learning Representations (ICLR), 2019. [Online]. Available: https://openreview.net/forum?id=HyztsoC5Y7

[34] Y. Jaafra, A. Deruyver, J. L. Laurent and M. S. Naceur, “Context-Aware Autonomous Driving Using Meta-Reinforcement Learning,” 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, USA, 2019, pp. 450-455, doi: 10.1109/ICMLA.2019.00084

[35] A. Kar et al., "Meta-Sim: Learning to Generate Synthetic Datasets," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 4550-4559, doi: 10.1109/ICCV.2019.00465

[36] K. Arndt, M. Hazara, A. Ghadirzadeh and V. Kyrki, “Meta Reinforcement Learning for Sim-to-real Domain Adaptation,” 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 2020, pp. 2725-2731, doi: 10.1109/ICRA40945.2020.9196540

[37] M. R. U. Saputra, P. Gusmao, Y. Almalioglu, A. Markham and N. Trigoni, “Distilling Knowledge From a Deep Pose Regressor Network,” 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 263-272, doi: 10.1109/ICCV.2019.00035

[38] L. Zhang, R. Dong, H.-S. Tai and K. Ma, “PointDistiller: Structured Knowledge Distillation Towards Efficient and Compact 3D Detection,” 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, Canada, 2023, pp. 21791-21801, doi: 10.48550/arXiv.2205.11098

[39] C. Sautier, G. Puy, S. Gidaris, A. Boulch, A. Bursuc and R. Marlet, “Image-to-Lidar Self-Supervised Distillation for Autonomous Driving Data,” 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 2022, pp. 9881-9891, doi: 10.1109/CVPR52688.2022.00966

[40] J. Li, H. Dai and Y. Ding, “Self-Distillation for Robust LiDAR Semantic Segmentation in Autonomous Driving,” in S. Avidan, G. Brostow, M. Cissé, G.M. Farinella and T. Hassner (eds), Computer Vision – ECCV 2022, Lecture Notes in Computer Science, vol. 13688, Springer, Cham, 2022, pp 659-676, doi: 10.1007/978-3-031-19815-1_38

[41] Y. Zhu, et al., “Reinforcement and Imitation Learning for Diverse Visuomotor Skills,” in Proceedings of Robotics: Science and Systems, vol. XIV, Pittsburgh, Pennsylvania, Jun 2018, doi: 10.15607/RSS.2018.XIV.009

[42] S. Desai, I. Durugkar, H. Karnan, G. Warnell, J. Hanna and P. Stone, “An Imitation from Observation Approach to Transfer Learning with Dynamics Mismatch,” in H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan and H. Lin (eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc., vol. 33, pp. 3917-3929, 2020. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/28f248e9279ac845995c4e9f8af35c2b-Paper.pdf

[43] Z. Javed, “Robust Imitation Learning for Risk-Aware Behavior and Sim2Real Transfer,” Master’s Thesis, Technical Report No. UCB/EECS-2022-48, University of California, Berkeley, 2022. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-48.html

[44] K. Tsinganos, “Sim2Real Methods for Visual-based Imitation Learning,” Master’s Thesis, University of Patras, 2021. [Online]. Available: http://hdl.handle.net/10889/15316

[45] M. Malmir, J. Josifovski, N. Klarmann and A. Knoll, “DiAReL: Reinforcement Learning with Disturbance Awareness for Robust Sim2Real Policy Transfer in Robot Control,” arXiv Preprint, 2023, doi: 10.48550/arXiv.2306.09010

[46] J. W. Kim, H. Shim and I. Yang, “On Improving the Robustness of Reinforcement Learning-based Controllers using Disturbance Observer,” 2019 IEEE 58th Conference on Decision and Control (CDC), Nice, France, 2019, pp. 847-852, doi: 10.1109/CDC40024.2019.9028930

[47] J. Josifovski, M. Malmir, N. Klarmann, B. L. Žagar, N. Navarro-Guerrero and A. Knoll, “Analysis of Randomization Effects on Sim2Real Transfer in Reinforcement Learning for Robotic Manipulation Tasks,” 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 2022, pp. 10193-10200, doi: 10.1109/IROS47612.2022.9981951

[48] X. Yue, Y. Zhang, S. Zhao, A. Sangiovanni-Vincentelli, K. Keutzer and B. Gong, “Domain Randomization and Pyramid Consistency: Simulation-to-Real Generalization Without Accessing Target Domain Data,” 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 2100-2110, doi: 10.1109/ICCV.2019.00219

[49] G. D. Kontes, D. D. Scherer, T. Nisslbeck, J. Fischer and C. Mutschler, “High-Speed Collision Avoidance using Deep Reinforcement Learning and Domain Randomization for Autonomous Vehicles,” 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), Rhodes, Greece, 2020, pp. 1-8, doi: 10.1109/ITSC45102.2020.9294396

[50] S. Pouyanfar, M. Saleem, N. George and S. -C. Chen, “ROADS: Randomization for Obstacle Avoidance and Driving in Simulation,” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 2019, pp. 1267-1276, doi: 10.1109/CVPRW.2019.00166

[51] J. Zhang et al., “VR-Goggles for Robots: Real-to-Sim Domain Adaptation for Visual Control,” in IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 1148-1155, April 2019, doi: 10.1109/LRA.2019.2894216

[52] C.Y. Zhang and A. Shrivastava, “AptSim2Real: Approximately-Paired Sim-to-Real Image Translation,” arXiv Preprint, 2023, doi: 10.48550/arXiv.2303.12704

[53] S. Tripathy, J. Kannala and E. Rahtu, “Learning Image-to-Image Translation Using Paired and Unpaired Training Samples,” in: C. Jawahar, H. Li, G. Mori and K. Schindler (eds), Computer Vision – ACCV 2018, Lecture Notes in Computer Science, vol. 11362, Springer, Cham., pp. 51–66, 2018, doi: 10.1007/978-3-030-20890-5_4

[54] A. Bewley et al., “Learning to Drive from Simulation without Real World Labels,” 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 4818-4824, doi: 10.1109/ICRA.2019.8793668

[55] H. Xiong, Z. Wang, G. Wu and Y. Pan “Design and Implementation of Digital Twin-Assisted Simulation Method for Autonomous Vehicle in Car-Following Scenario,” in Journal of Sensors, H. Jerbi, Eds., Hindawi, 2022, doi: 10.1155/2022/4879490

[56] K. L. Voogd, J. P. Allamaa, J. Alonso-Mora and T. Son, "Reinforcement Learning from Simulation to Real World Autonomous Driving using Digital Twin", in Proc. IFAC World Congress, July 2023, doi: 10.48550/arXiv.2211.14874

[57] J. P. Allamaa, P. Patrinos, H. Van der Auweraer, T. D. Son, “Sim2Real for Autonomous Vehicle Control using Executable Digital Twin,” in IFAC-PapersOnLine, vol. 55, no. 24, 2022, pp. 385-391, doi: 10.1016/j.ifacol.2022.10.314

[58] M. Groshev, C. Guimarães, A. Oliva, & R. Gazda, “Dissecting the impact of information and communication technologies on digital twins as a service,” IEEE Access, vol. 9, pp. 102862-102876, 2021, doi: 10.1109/access.2021.3098109

[59] Y. Lu, X. Huang, K. Zhang, S. Maharjan, & Y. Zhang, “Low-latency federated learning and blockchain for edge association in digital twin empowered 6g networks,” IEEE Transactions on Industrial Informatics, vol. 17, no. 7, pp. 5098-5107, 2021, doi: 10.1109/tii.2020.3017668

Appendix 1

Home / My Interactive Sessions / Jupyter Notebook

Interactive Apps

Desktops

Palmetto Desktop

GUIs

Abaqus/CAE

Matlab

UGUI

Servers

Code Server (VSCode)

Containerized Jupyter Notebook

Jupyter + Spark

Jupyter Notebook

RShiny App

RStudio Server

RStudio Server + Spark

Workshop Pytorch Notebook

Jupyter Notebook

This app will launch a Jupyter Notebook server on one or more nodes with more advanced PBS resource request options. Users can also specify virtual/conda environments to launch custom notebooks for Tensorflow and other advanced libraries.

Anaconda Version

anaconda3/2022.05-gcc/9.5.0

List of modules to be loaded, separate by an empty space

cuda/11.1.1-gcc/9.5.0 cudnn/8.0.5.39-11.1-gcc/9.5.0-cu11

Provide a space-separated list of modules to be loaded.
- For example: **openjdk/11.0.2-gcc/8.3.1 jags/4.3.0-gcc/8.3.1**

Path to Python virtual/conda environment

source activate aue8930

Provide an activation command to load the corresponding Python virtual (venv) or conda environment. You can replace
NAME_OF_ENVIRONMENT
with a corresponding conda environment, or
PATH_TO_VIRTUAL_ENVIRONMENT
with a specific path to your venv environment directory.
- Example conda: **conda activate NAME_OF_ENVIRONMENT**
- Example venv: **source PATH_TO_VIRTUAL_ENVIRONMENT/bin/activate**

Notebook Workflow

Standard Jupyter Notebook

Number of resource chunks (select)

1

CPU cores per chunk (ncpus)

16

- Typical Palmetto compute nodes have **8, 12, 16, 20, 24, 28, 40, and 56** cores.
- DGX nodes have **128** cores.
- Bigmem nodes have **24, 32, 40, and 80** cores. - **Users can request any number of cores that is smaller than the number of available cores.**

Amount of memory per chunk (mem)

32gb

- Typical Palmetto compute nodes have **15gb, 30gb, 46gb, 62gb, 125gb, 372gb, 748gb, and 990gb** of memory.
- DGX nodes have **990gb** of memory.
- Bigmem nodes have **500gb and 750gb** and **1tb and 1.5tb** of memory.

Number of GPUs per chunk (ngpus)

1

GPU Model (gpu_model)

V100

Interconnect

100g - Ethernet phase 18 and above

Extra PBS resource allocation request

- Enter the additional resource request just like how you would in a command line environment.
- Each request should start with a colon **:** sign.
- For example: **:chip_type=e5-2665**

Walltime

08:00:00

- Walltime format is **hh:mm:ss**.
- Phase 1 through 6 nodes can be reserved up to 336 hours.
- Phase 7 through 27 nodes can be reserved up to 72 hours.

Queue

work1

Queue to submit the job to

Absolute path to working directory

Select your project directory; defaults to $HOME

I would like to receive an email when the session starts

Launch

* The Jupyter Notebook session data for this session can be accessed under the data root directory.

Appendix 2

Home / My Interactive Sessions / Jupyter Notebook

Interactive Apps

Desktops

Palmetto Desktop

GUIs

Abaqus/CAE

Matlab

UGUI

Servers

Code Server (VSCode)

Containerized Jupyter Notebook

Jupyter + Spark

Jupyter Notebook

RShiny App

RStudio Server

RStudio Server + Spark

Workshop Pytorch Notebook

Jupyter Notebook

This app will launch a Jupyter Notebook server on one or more nodes with more advanced PBS resource request options. Users can also specify virtual/conda environments to launch custom notebooks for Tensorflow and other advanced libraries.

Anaconda Version

anaconda3/2022.05-gcc/9.5.0

List of modules to be loaded, separate by an empty space

cuda/11.1.1-gcc/9.5.0 cudnn/8.0.5.39-11.1-gcc/9.5.0-cu11

Provide a space-separated list of modules to be loaded.
- For example: **openjdk/11.0.2-gcc/8.3.1 jags/4.3.0-gcc/8.3.1**

Path to Python virtual/conda environment

source activate aue8930

Provide an activation command to load the corresponding Python virtual (venv) or conda environment. You can replace
NAME_OF_ENVIRONMENT
with a corresponding conda environment, or
PATH_TO_VIRTUAL_ENVIRONMENT
with a specific path to your venv environment directory.
- Example conda: **conda activate NAME_OF_ENVIRONMENT**
- Example venv: **source PATH_TO_VIRTUAL_ENVIRONMENT/bin/activate**

Notebook Workflow

Standard Jupyter Notebook

Number of resource chunks (select)

1

CPU cores per chunk (ncpus)

16

- Typical Palmetto compute nodes have **8, 12, 16, 20, 24, 28, 40, and 56** cores.
- DGX nodes have **128** cores.
- Bigmem nodes have **24, 32, 40, and 80** cores. - **Users can request any number of cores that is smaller than the number of available cores.**

Amount of memory per chunk (mem)

32gb

- Typical Palmetto compute nodes have **15gb, 30gb, 46gb, 62gb, 125gb, 372gb, 748gb, and 990gb** of memory.
- DGX nodes have **990gb** of memory.
- Bigmem nodes have **500gb and 750gb** and **1tb and 1.5tb** of memory.

Number of GPUs per chunk (ngpus)

2

GPU Model (gpu_model)

V100

Interconnect

100g - Ethernet phase 18 and above

Extra PBS resource allocation request

- Enter the additional resource request just like how you would in a command line environment.
- Each request should start with a colon **:** sign.
- For example: **:chip_type=e5-2665**

Walltime

08:00:00

- Walltime format is **hh:mm:ss**.
- Phase 1 through 6 nodes can be reserved up to 336 hours.
- Phase 7 through 27 nodes can be reserved up to 72 hours.

Queue

work1

Queue to submit the job to

Absolute path to working directory

Select your project directory; defaults to $HOME

I would like to receive an email when the session starts

Launch

* The Jupyter Notebook session data for this session can be accessed under the data root directory.