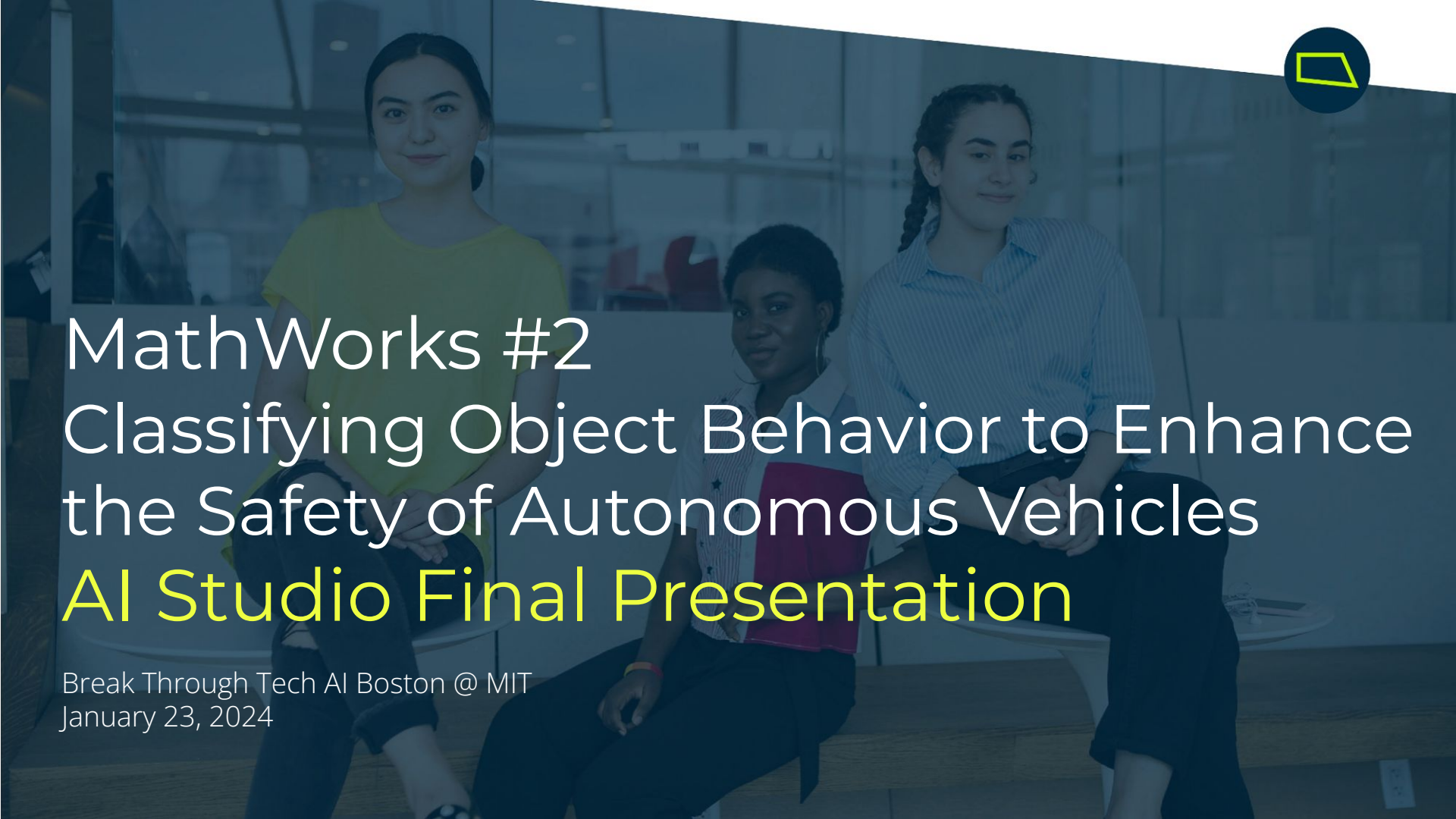


MathWorks #2
Classifying Object Behavior to Enhance the Safety of Autonomous Vehicles
AI Studio Final Presentation

Break Through Tech AI Boston @ MIT
January 23, 2024

Introductions

Meet Our Team!

Pamela Melgar
Tufts University

Elena Wang
Smith College

Kashish Gupta
Worcester Polytechnic Institute

Rachel Ma
Northeastern University

Vanessa Bellotti
Tufts University

Nyah Lalimarmo
Simmons University

Our AI Studio TA and Challenge Advisor

Maria Elena Gavilan Alfonso
Challenge Advisor

Keith Murray
AI Studio TA

Presentation Agenda

1. Project Overview
2. Data Understanding and Data Preparation
3. Transfer Learning & Foundational Model
4. Model Training and Evaluation
5. Final Thoughts
6. Questions

AI Studio Project Overview

Fine-tuning the detection of pedestrians and cyclists near autonomous vehicles using Deep Learning (Transfer Learning)

Business Impact

- Increasing **‘public trust’** in self-driving cars
  - Insight into *potential* dangers
  - Potentially smoother driving, better identification
- Accurate object detection = **safe critical decisions**
- Transfer Learning is **‘cost-effective’**
  - Cut costs of building model from scratch

Our goals

1. Use **real sensor data** collected at Boston to aid the development of autonomous vehicles
2. Explore the efficacy of leveraging **transfer learning** within an autonomous vehicle object classification problem (for pedestrians and cyclists)
    a. Fine-tune an object detection model that demonstrates high accuracy in identifying pedestrians and cyclists
3. Add-in a 3D element to the object detection model by implementing LIDAR data (To explore in the future)

Our Approach

Data preparation

Explored nuImages data and created training and test sets

Training Preparation

Downloaded TensorFlow pretrained models and explored the Object Detection API

Training

Transfer Learning

Evaluation

Used metrics like precision and recall to fine-tune the model

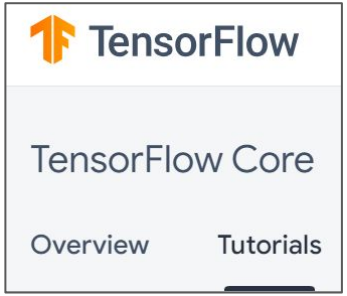
Resources We Leveraged

TensorFlow

MATLAB®

NUSCENES

Training

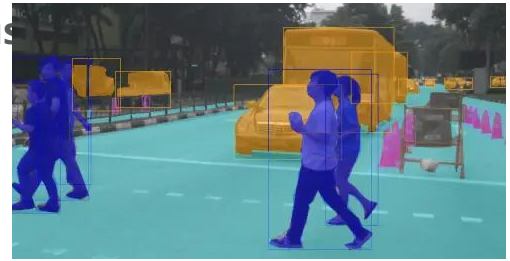


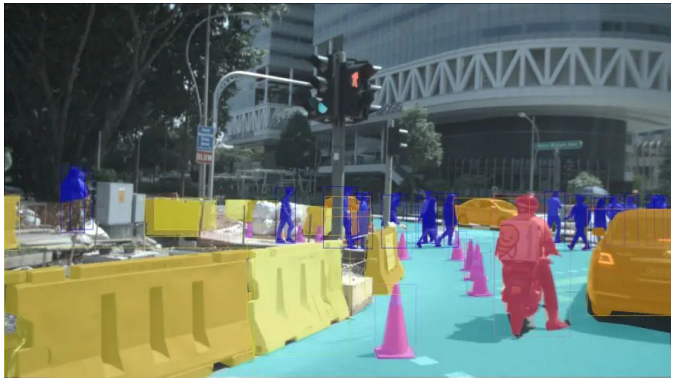
Deep Learning Onramp (MATLAB)

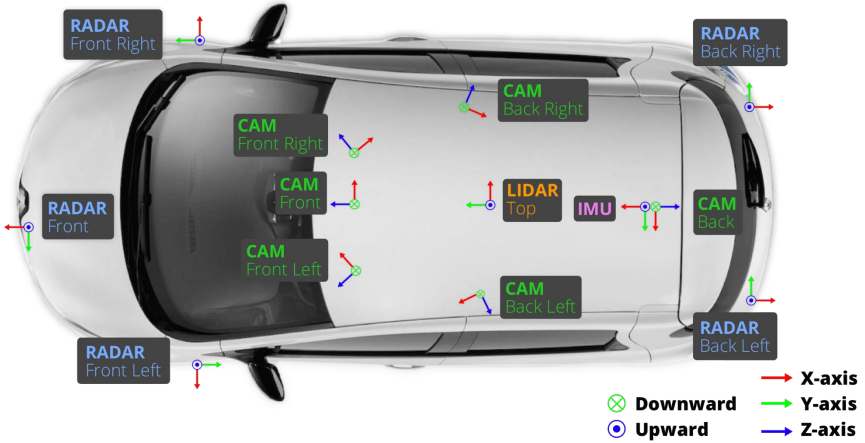
Data Understanding & Data Preparation

Dataset Overview - nuImages

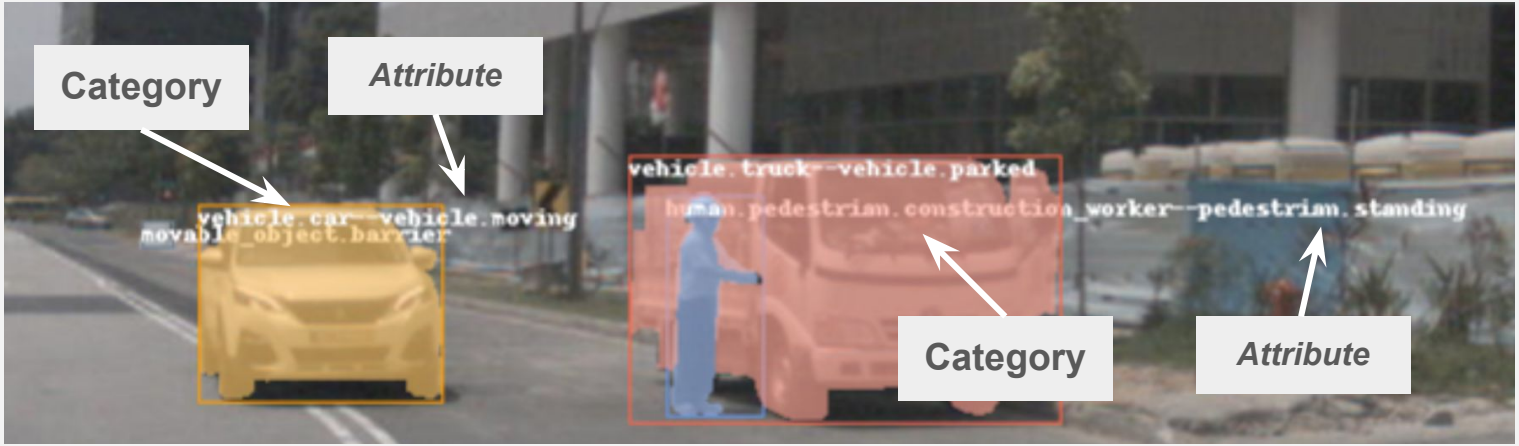
- ***nuImages*** is a subset of ***nuScenes***, a public large-scale **autonomous driving dataset** developed by Motional
- **93k 2D annotated images** mined for diversity
- Full sensor suite (1x LIDAR, 5x RADAR, **6x camera**, IMU, GPS)
- **800k annotated foreground objects** with 2D bounding boxes and instance masks







Example of nuImages




```
def store_single_image(sample_record):
  key_cam_token = sample_record['key_camera_token'] #get token in order to gain access to sample_data
  sample_data_record = nuim.get('sample_data', sample_record['key_camera_token'])

  image_filename = "example_image_name"
  nuim.render_image(key_cam_token, annotation_type='objects', with_category=True, with_attributes=True, out_path='/content/sample_data/images/' + image_fil
```

Data Selection

- Data already cleaned
- Features, annotations and attributes
- Only want pedestrians and cyclists

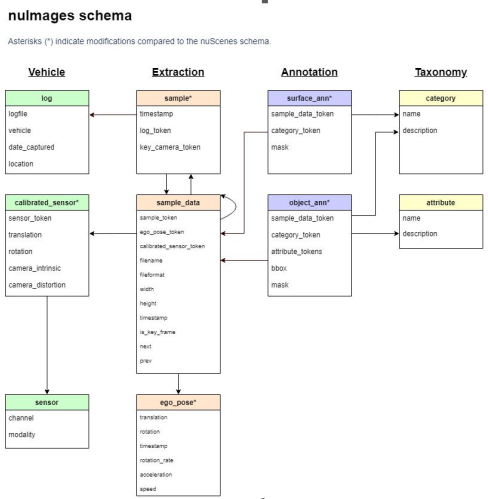
```
for cat in nuim.category:
  print(cat['name'])
```

```
animal
flat.driveable_surface
human.pedestrian.adult
human.pedestrian.child
human.pedestrian.construction_worker
human.pedestrian.personal_mobility
human.pedestrian.police_officer
human.pedestrian.stroller
human.pedestrian.wheelchair
movable_object.barrier
movable_object.debris
movable_object.pushable_pullable
movable_object.trafficcone
static_object.bicycle_rack
vehicle.bicycle
vehicle.bus.bendy
vehicle.bus.rigid
vehicle.car
vehicle.construction
vehicle.ego
vehicle.emergency.ambulance
vehicle.emergency.police
vehicle.motorcycle
vehicle.trailer
vehicle.truck
```

```
for att in nuim.attribute:
  print(att['name'])
```

```
cycle.with_rider
cycle.without_rider
pedestrian.moving
pedestrian.sitting_lying_down
pedestrian.standing
vehicle.moving
vehicle.parked
vehicle.stopped
vehicle_light.emergency.flashing
vehicle_light.emergency.not_flashing
vertical_position.off_ground
vertical_position.on_ground
```

Converting into XML format




```
<annotation>
    <folder>vehicles</folder>
    <filename>ff9435ee-ba7e-4d32-93b
    <path>E:\vehicles\ff9435ee-ba7e-
    <size>
        <width>800</width>
        <height>598</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>truck</name>
        <bndbox>
            <xmin>7</xmin>
            <ymin>119</ymin>
            <xmax>630</xmax>
            <ymax>468</ymax>
        </bndbox>
    </object>
    <object>
        <name>person</name>
        <bndbox>
            <xmin>40</xmin>
            <ymin>90</ymin>
            <xmax>100</xmax>
            <ymax>350</ymax>
        </bndbox>
    </object>
</annotation>
```

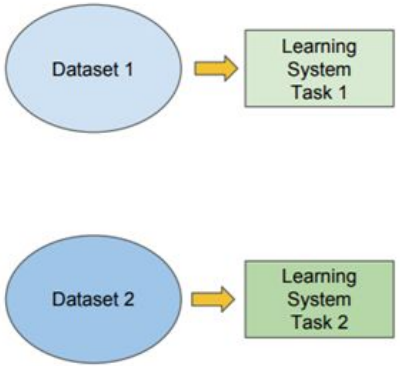
Pascal VOC XML

Transfer Learning & Foundation Model

Why Transfer Learning?

Traditional ML

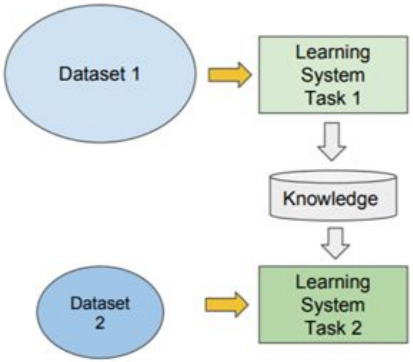
- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



vs

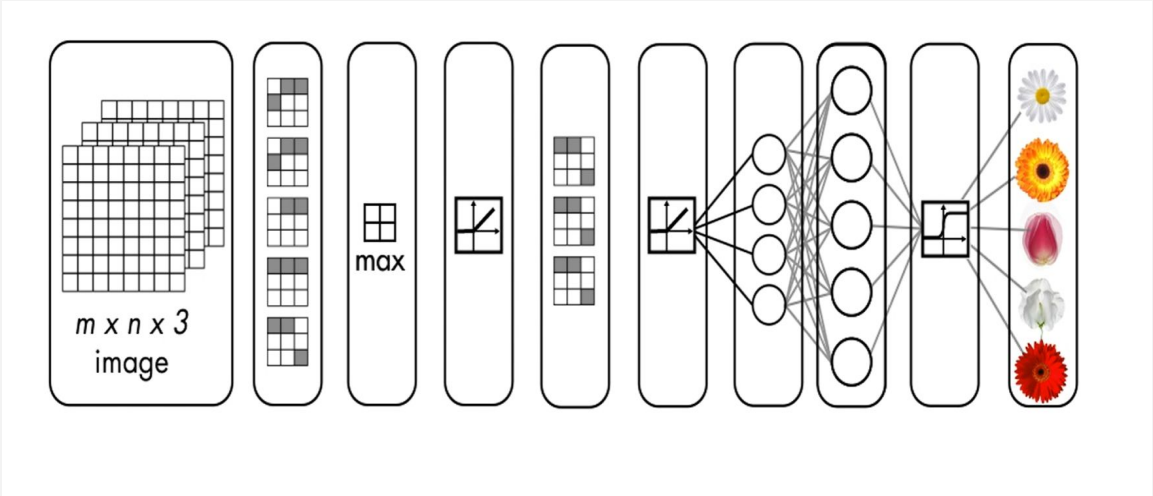
Transfer Learning

- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data

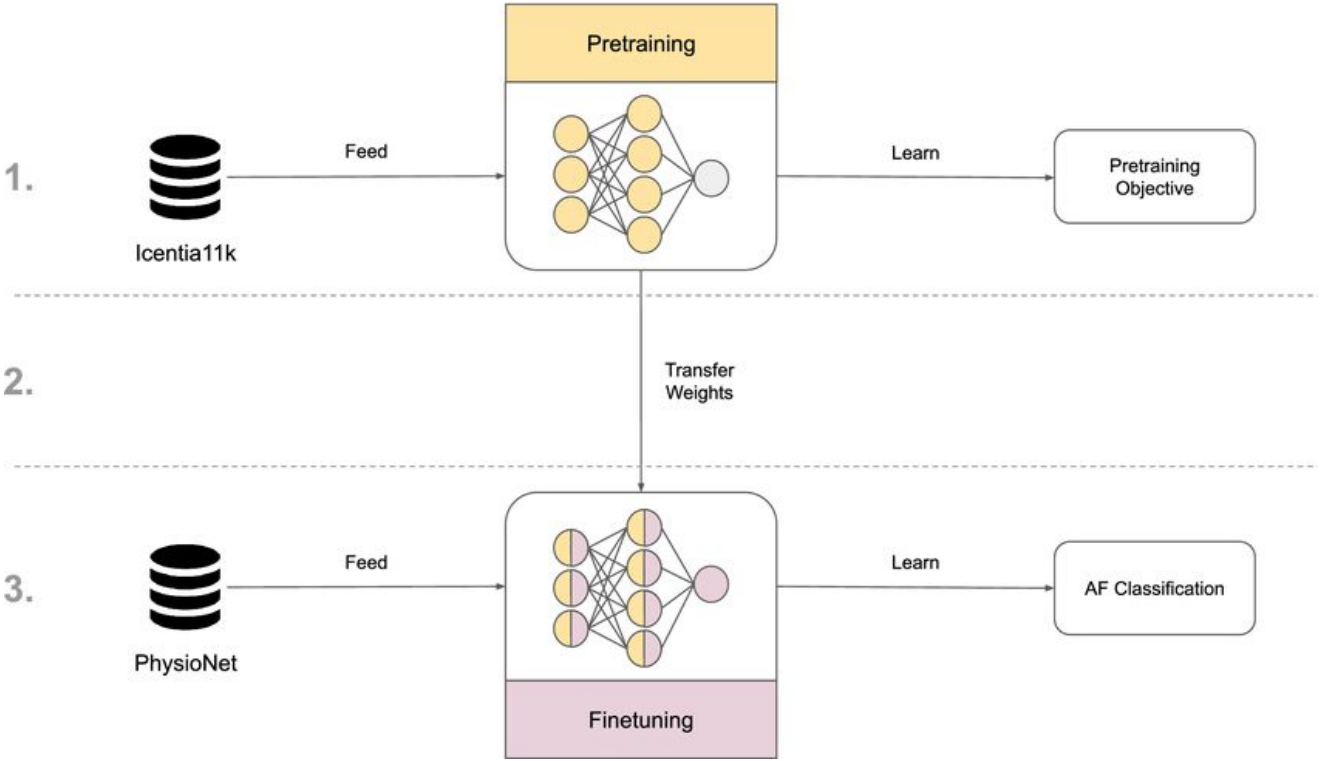


Visual Representation of Transfer Learning

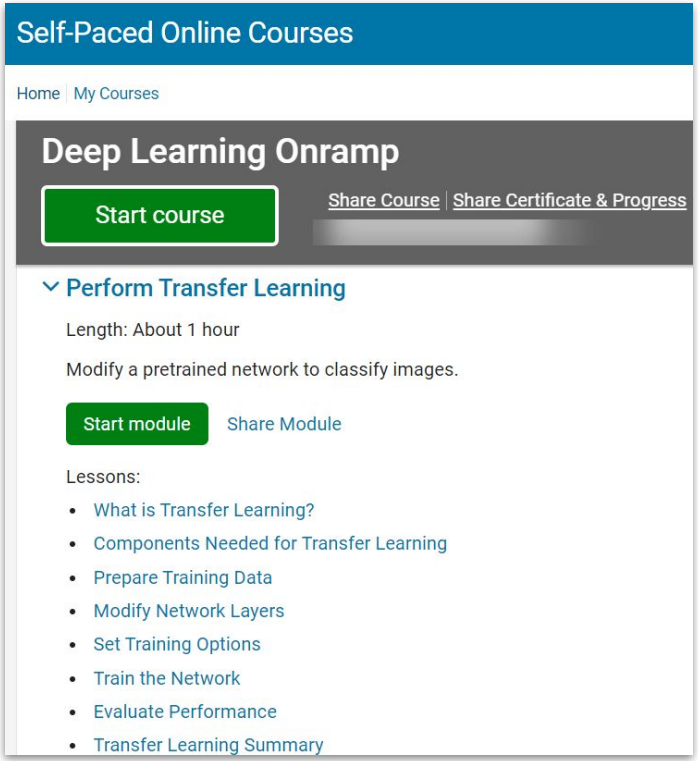
- Take a pre-trained Network
- Modify and re-train it on **new** data

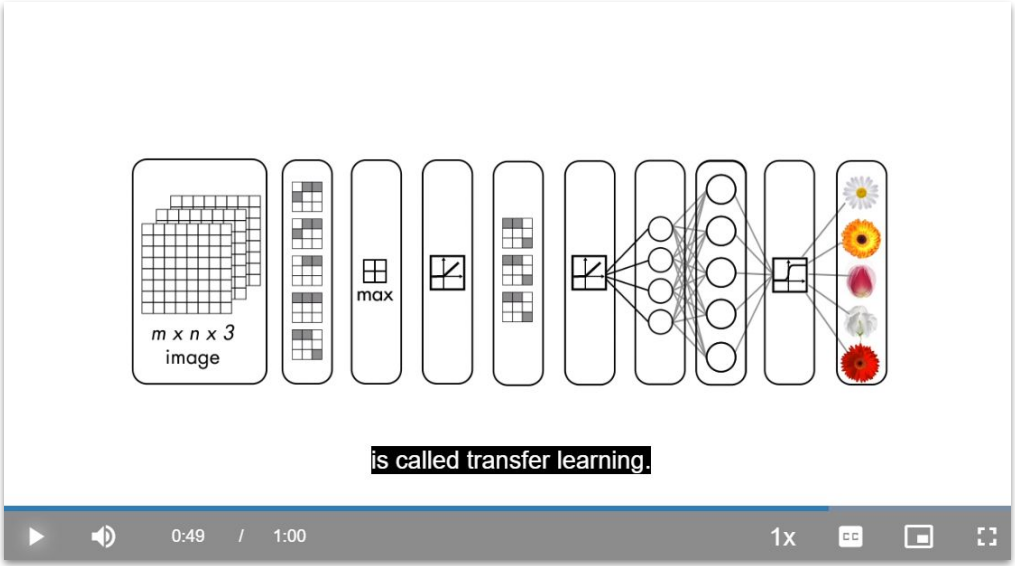


Overview of Transfer Learning



Deep Learning Onramp was useful to get acquainted with Transfer Learning





Model Hubs enable access to several pretrained models

Models available in MATLAB:

Network	Size (MB)	Classes	Accuracy %	Location
googlenet	27	1000	66.25	Doc GitHub
squeezenet	5.2	1000	55.16	Doc
alexnet	227	1000	54.10	Doc
resnet18	44	1000	69.49	Doc GitHub
resnet50	96	1000	74.46	Doc GitHub
resnet101	167	1000	75.96	Doc GitHub

Pretrained EfficientDet Network For Object Detection

This repository provides a pretrained EfficientDet-D0[1] object detection network for MATLAB®.

Open in MATLAB Online

Requirements

- MATLAB® R2021a or later
- Deep Learning Toolbox™
- Computer Vision Toolbox™
- Deep Learning Toolbox Converter for ONNX Model Format™ Support Package

MATLAB Deep Learning Model Hub

TensorFlow 2 Detection Model Zoo

TensorFlow 2.2 Python 3.6

We provide a collection of detection models pre-trained on the COCO 2017 dataset. These models can be useful for out-of-the-box inference if you are interested in categories already in those datasets. You can try it in our inference colab

They are also useful for initializing your models when training on novel datasets. You can try this out on our few-shot training colab.

Please look at this guide for mobile inference.

Finally, if you would like to train these models from scratch, you can find the model configs in this directory (also in the linked `tar.gz`s).

Model name	Speed (ms)	COCO mAP	Outputs
CenterNet HourGlass104 512x512	70	41.9	Boxes
CenterNet HourGlass104 Keypoints 512x512	76	40.0/61.4	Boxes/Keypoints
CenterNet HourGlass104 1024x1024	197	44.5	Boxes
CenterNet HourGlass104 Keypoints 1024x1024	211	42.8/64.5	Boxes/Keypoints
CenterNet Resnet50 V1 FPN 512x512	27	31.2	Boxes
CenterNet Resnet50 V1 FPN Keypoints 512x512	30	29.3/50.7	Boxes/Keypoints
CenterNet Resnet101 V1 FPN 512x512	34	34.2	Boxes
CenterNet Resnet50 V2 512x512	27	29.5	Boxes

TensorFlow 2 Model Zoo

Model Architecture Selection

Selected several object detection models from the **TensorFlow 2 Detection Model Zoo**

Picked **4 architectures** to test from each group:

Faster R-CNN ResNet101 V1 1024x1024	CenterNet Resnet101 V1 FPN 512x512	SSD ResNet101 V1 FPN 640x640 (RetinaNet101)	EfficientDet D1 640x640

Based on **3 factors**:

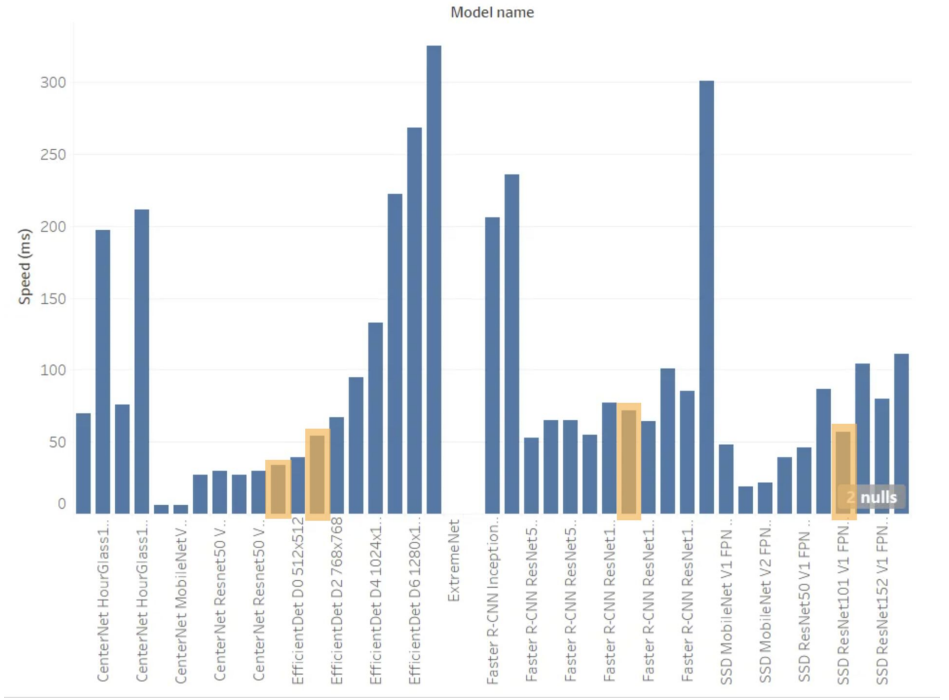
Speed	COCO mAP (mean Average Precision on COCO dataset)	Outputs (keypoints, bounding boxes, masks, or multiple)

After model testing on a neutral dataset, we observed that **Faster R-CNN** is best for our purposes

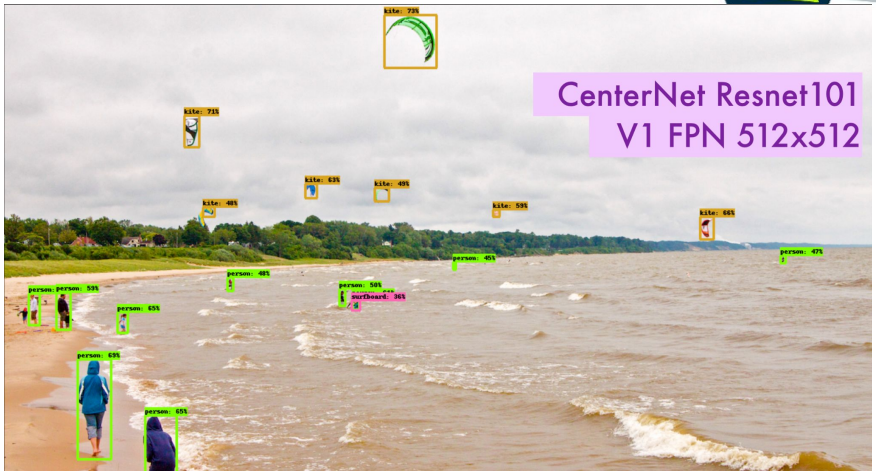
Model Comparison

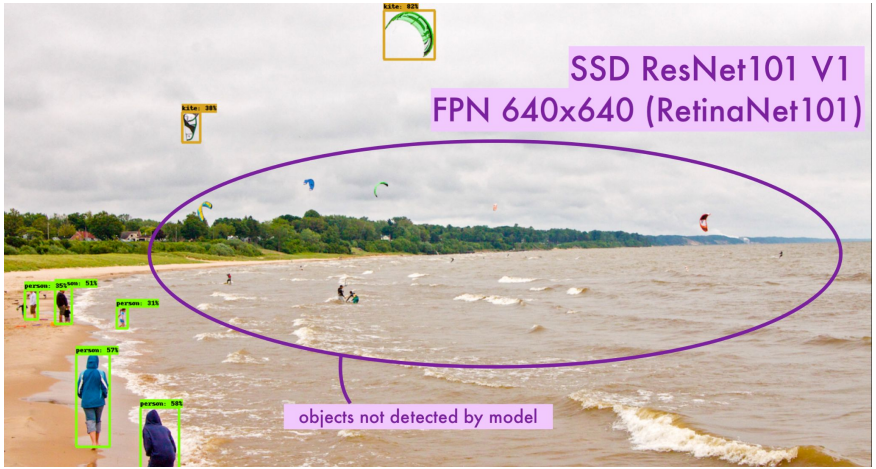
Model Name	Speed (ms)	mAP [precision]	Outputs
Faster R-CNN ResNet101 V1 1024x1024	**72**	**37.1**	**Boxes**
CenterNet Resnet101 V1 FPN 512x512	34	34.2	Boxes
EfficientDet D1 640x640	54	38.4	Boxes
SSD ResNet101 V1 FPN 640x640 (RetinaNet101)	57	35.6	Boxes

Comparison of speed across all of the architectures

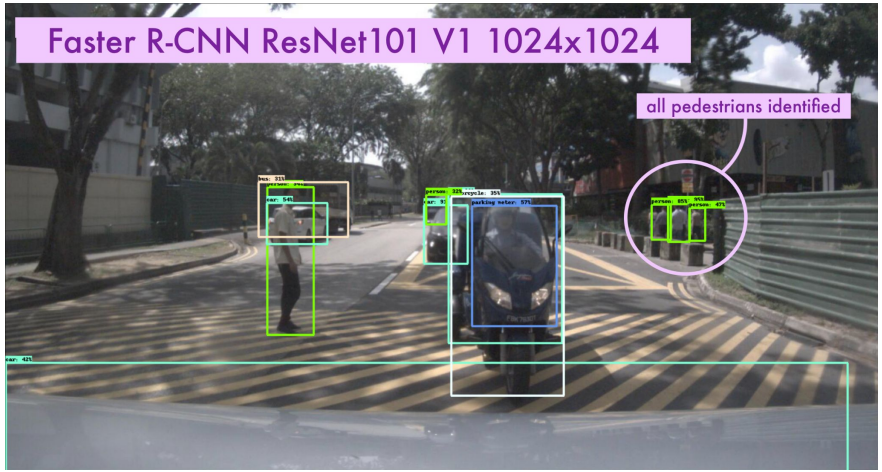


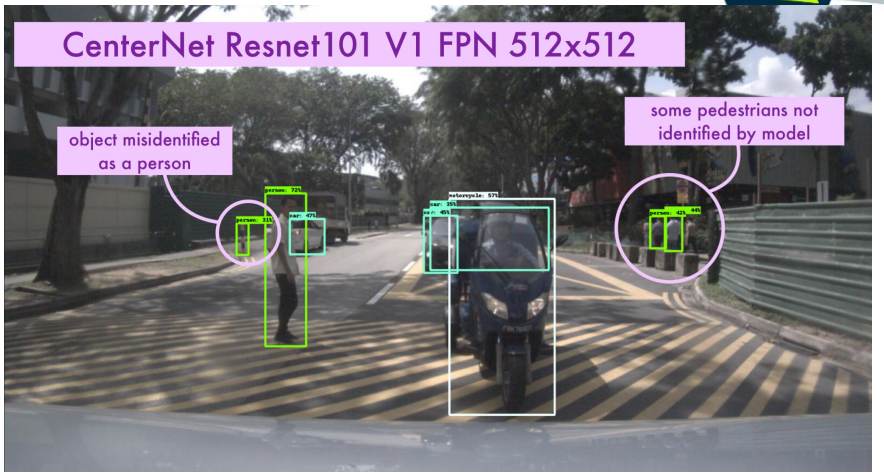
Comparison on beach sample

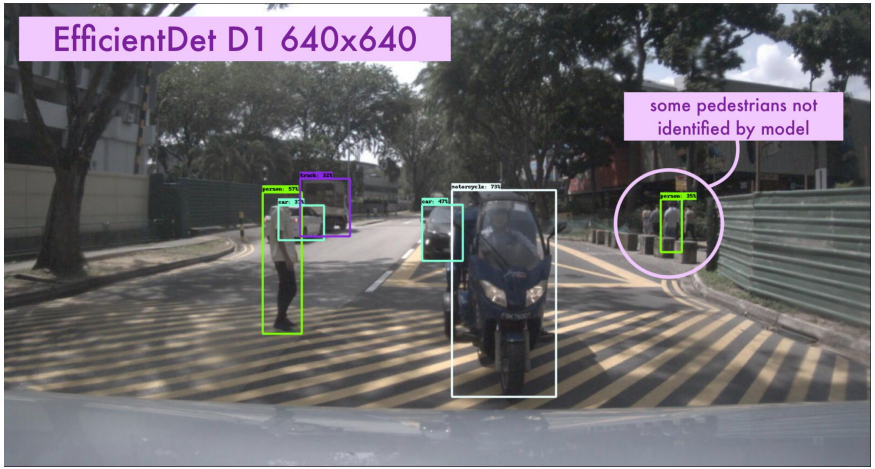


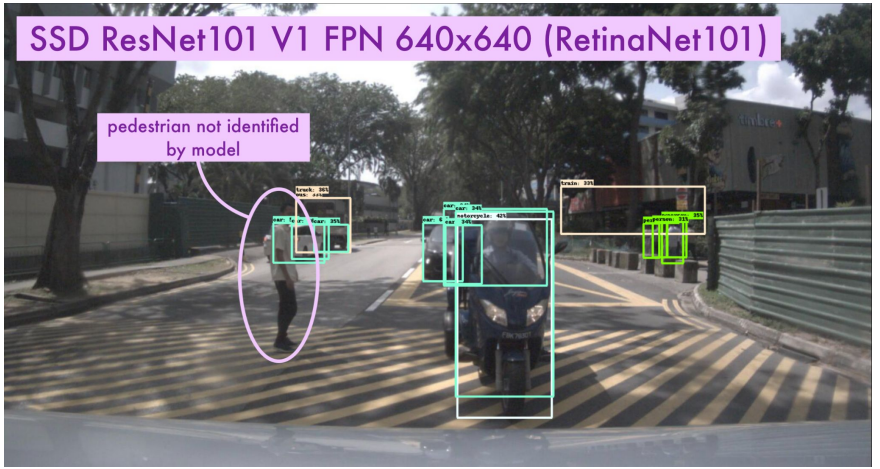


Comparison on nuImages - part 1

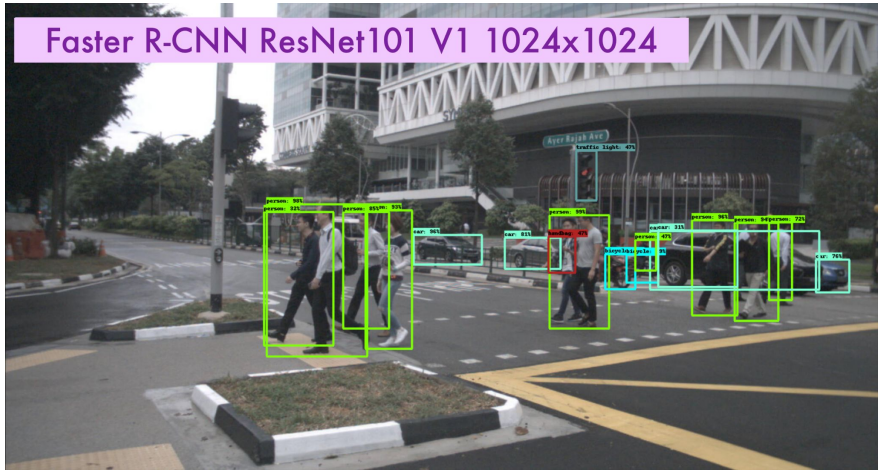


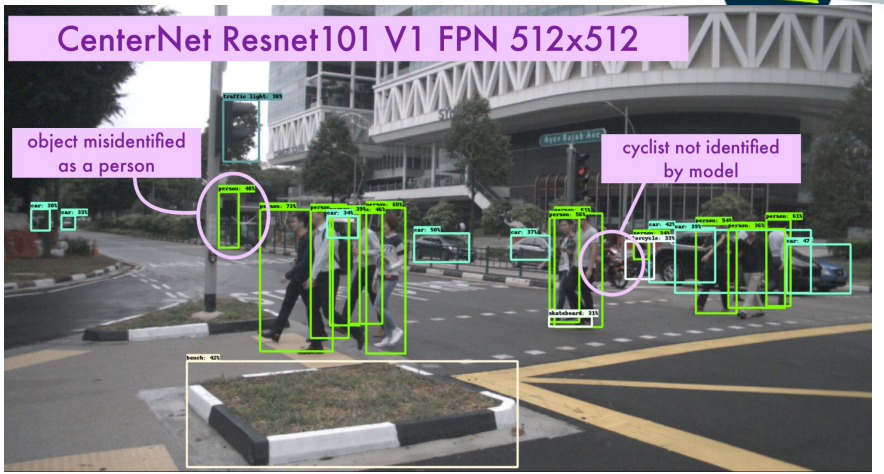


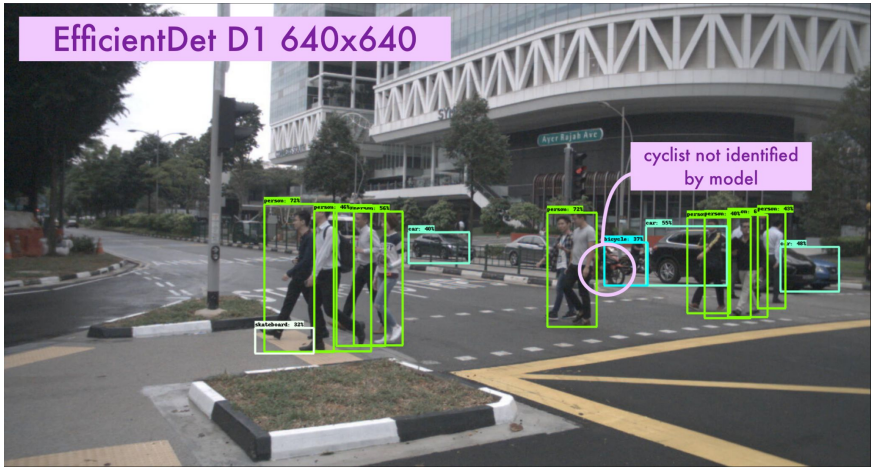


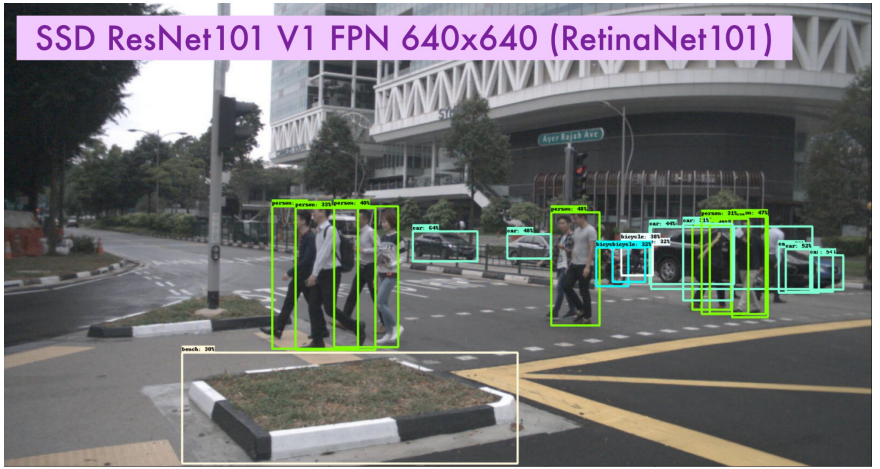


Comparison on nuImages - part 2









Customizing Pretrained Object Detectors

- Existing pre-trained networks for object detection can identify people, bikes and cars, but they provide little to no information to the specifics of the pedestrians (standing, moving, etc.)
- Dataset used in this project contains all the additional details needed to customize the detection of pedestrians and people riding bicycles or motorcycles.
- Architectures pre-selected can be used in the model training step.

Model Training & Evaluation

Environments We Utilized

Tufts UNIVERSITY

Data Decisions for Training

500 total images

Limited number of images out of the 87k available

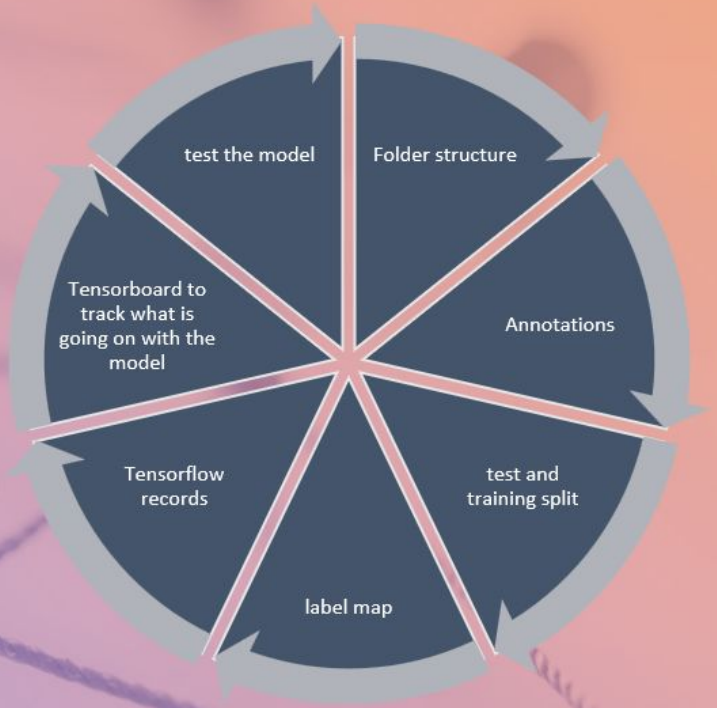
FRONT CAM

Images used from FRONT Cameras only

Moving

Must include at least one moving pedestrian or cycle with rider

The training process



The training process - New Classes for Detection

Examples of classes to train on:

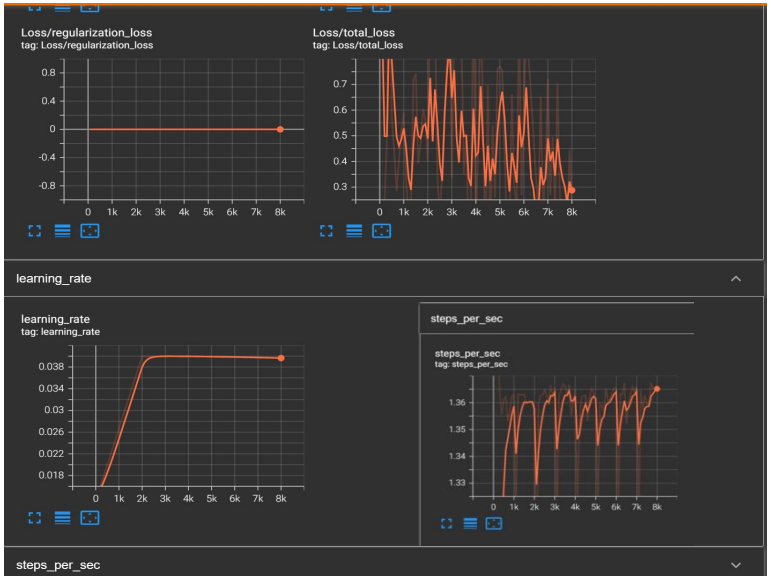
'Human.pedestrian.adult-pedestrian.moving'

'Human.pedestrian.child-pedestrian.moving'

A combination of category and attribute to make the classes distinctive enough. In total, we trained the object detector to detect 10 different types of humans moving in the image.

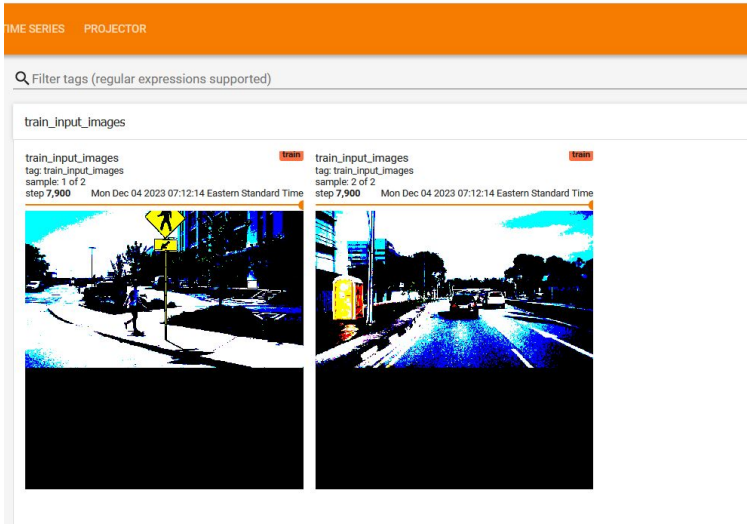
Training Results - Improper Data

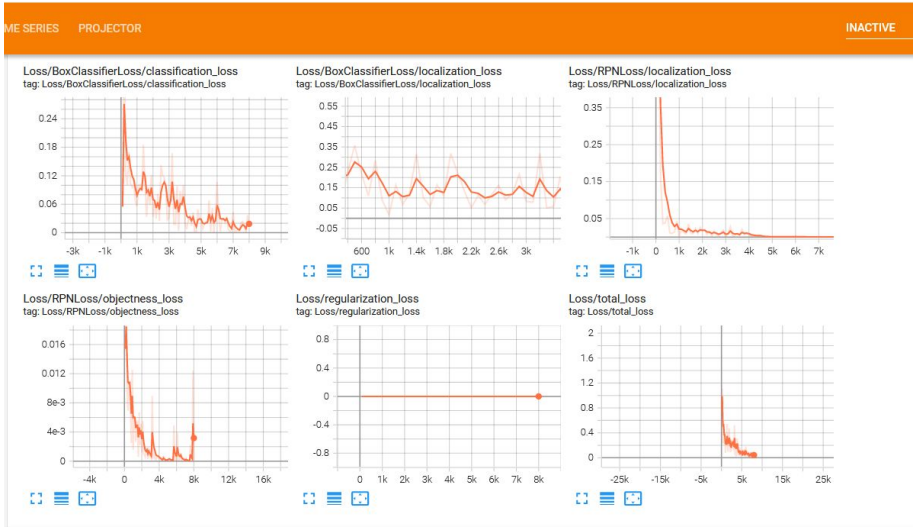
Fast RCNN ResNet 101(1024x1024)



Training Results - First pretrained model

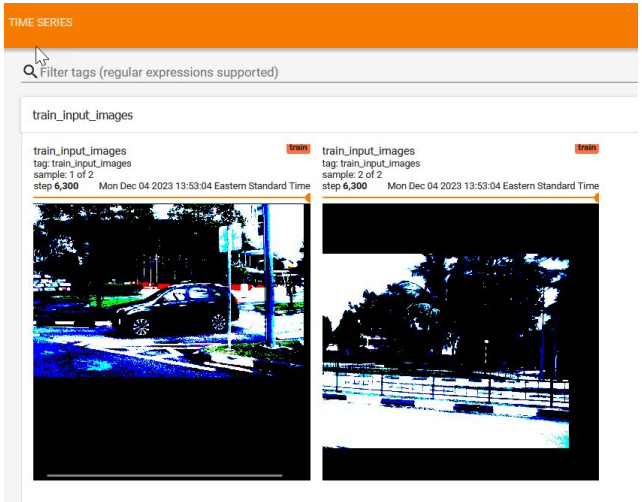
Fast RCNN ResNet 50 (640x640)

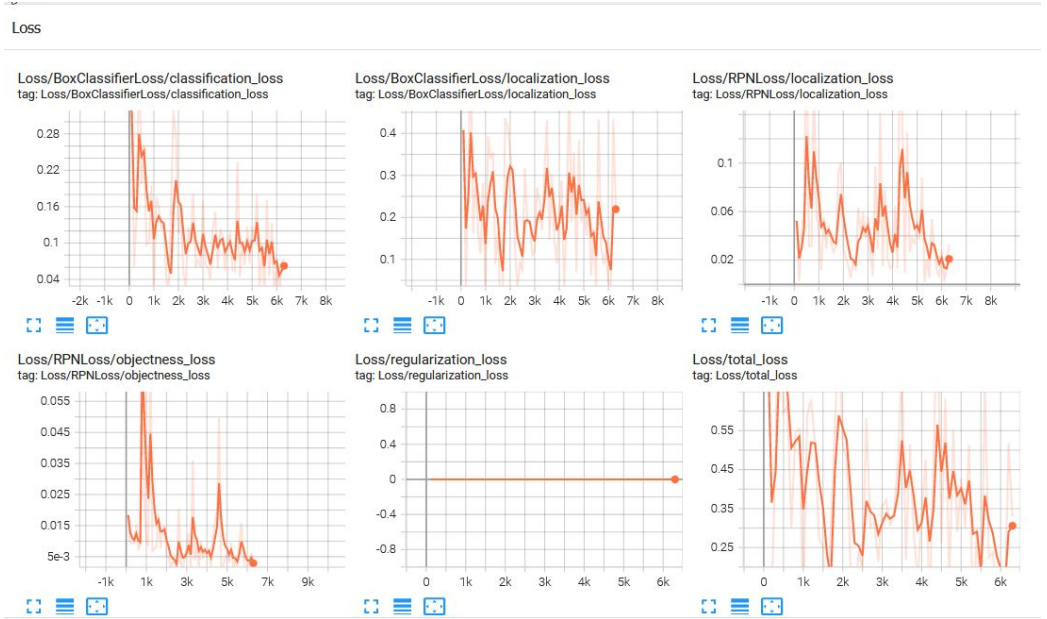




Training Results - Second pretrained model

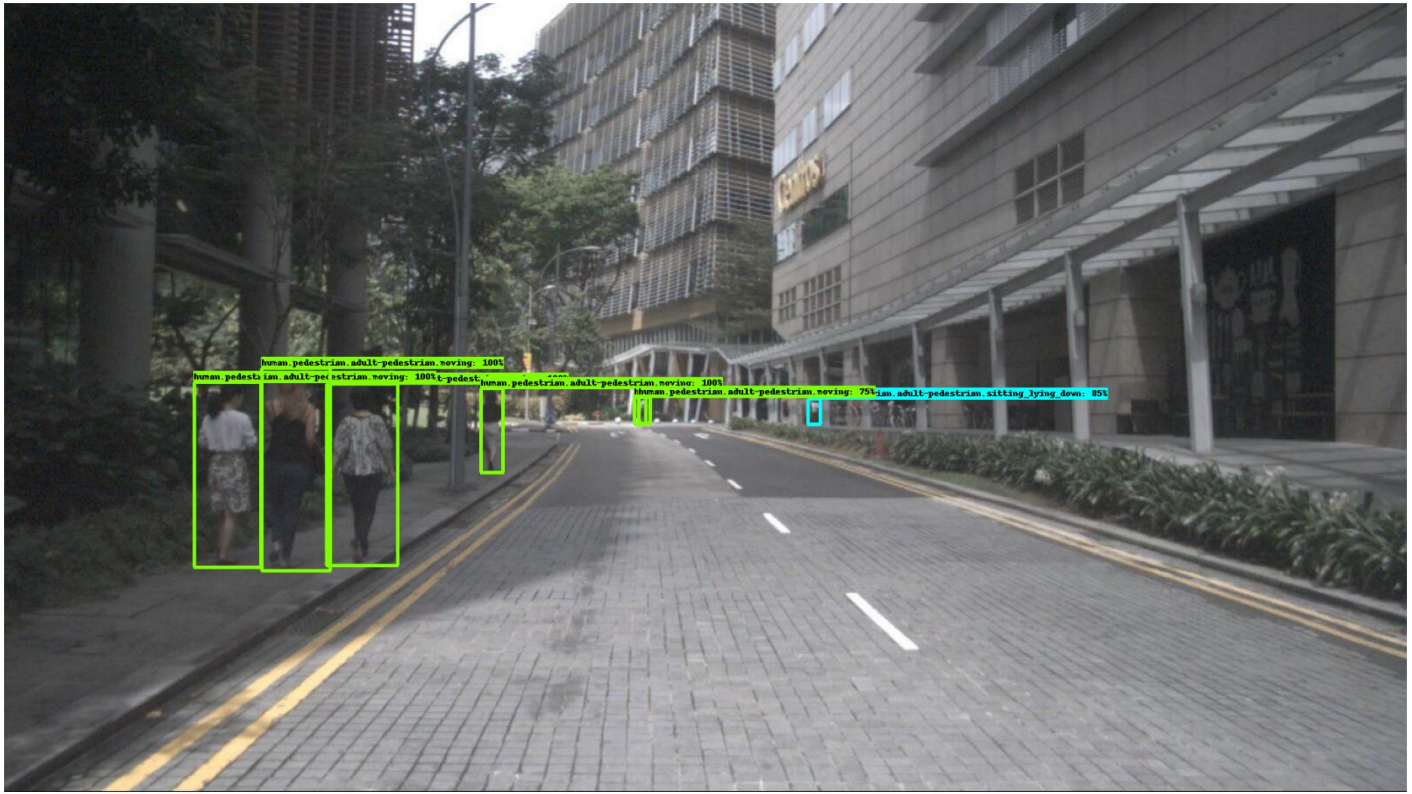
Fast RCNN ResNet 101(1024x1024)



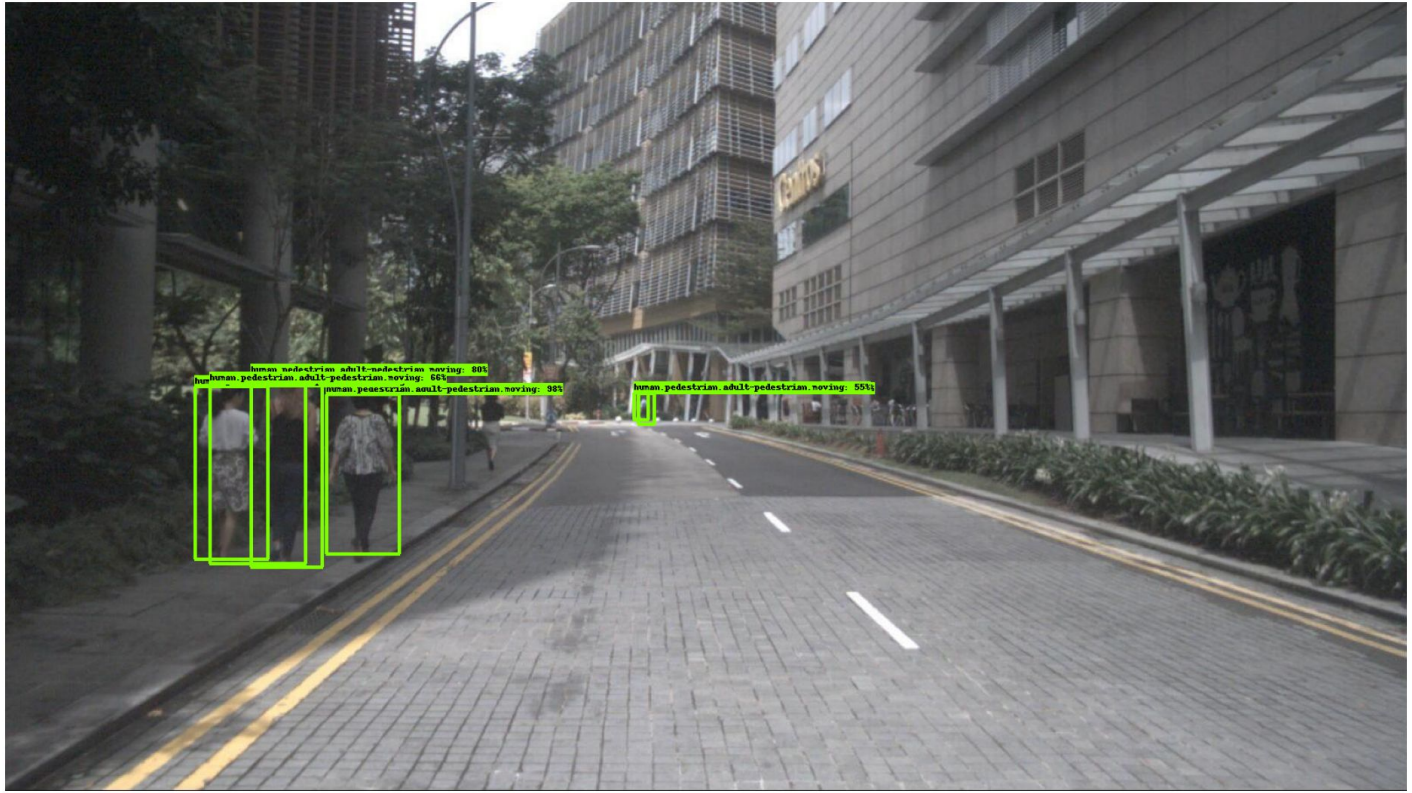


Testing the Model (before detection)

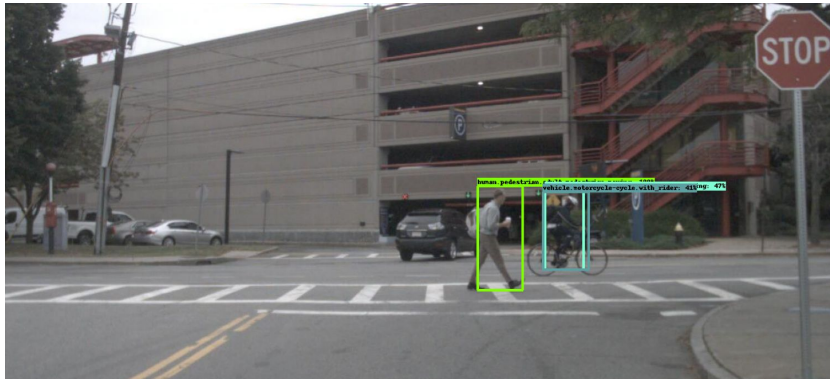
Testing the Model (with model trained #1)

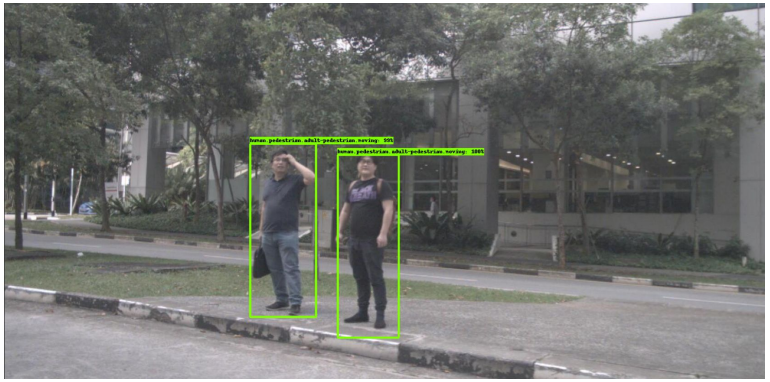


Testing the Model (with model trained #2)



Testing the Model





Insights and Key Findings

TRANSFER LEARNING ENABLES THE TRAINING OF HIGHLY SPECIALIZED DEEP LEARNING NETWORKS THAT DO CUSTOM OBJECT DETECTION

HIGHLY PRECISE MODELS CAN BE TRANSFER LEARNED USING A SMALLER DATASET AND LESS COMPUTE THAN IF WE HAD TRAINED A BASE DEEP NEURAL NETWORK FROM SCRATCH

DATA INFRASTRUCTURE AND THE IMPORTANCE OF REPRODUCIBILITY IN PIPELINES

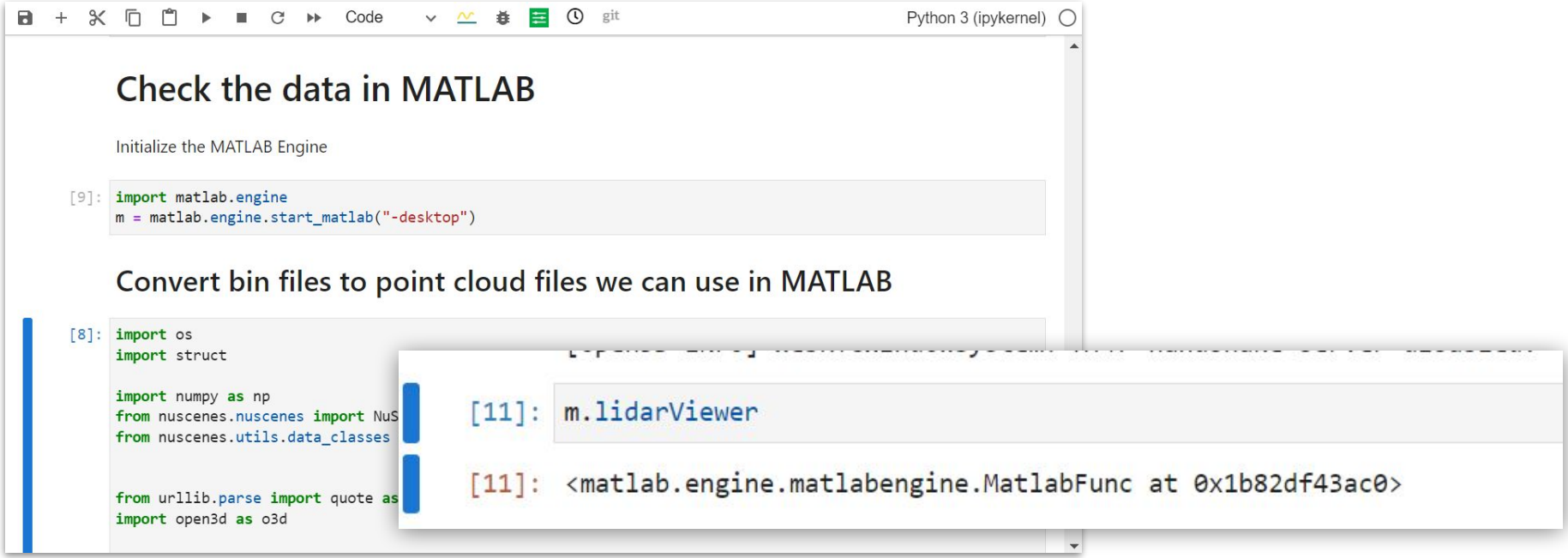
Further exploration: Using the model in MATLAB

- Using the Deep Learning Toolbox, TensorFlow models can be imported into MATLAB

Puzzle: TensorFlow 2 models with “SavedModel” format into MATLAB - How best to import models with this format?

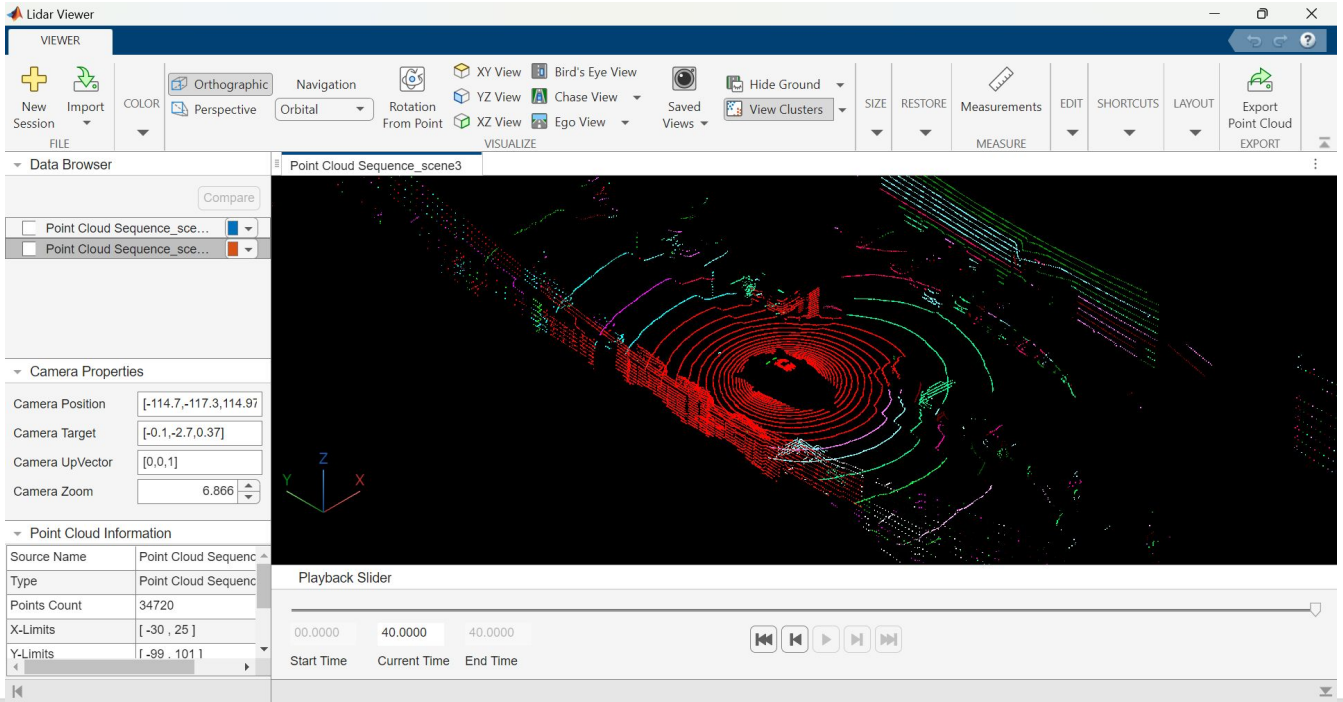
Next Step: Access LIDAR Data from nuScenes

- From Python we can export the LIDAR data as PCD (point cloud data) and open the results in MATLAB (using the MATLAB Engine API) for further exploration



Next Step: Access LIDAR Data from nuScenes

- From Python we can export the LIDAR data as PCD (point cloud data) and open the results in MATLAB (using the MATLAB Engine API) for further exploration



Final Thoughts

Summary

Fine-tune an object detection model that demonstrates high accuracy in detecting pedestrians and cyclists

Used 15GB of annotated image data from nuImages

Had to convert these into XML format in order to use TF records

Transfer Learning, object detection model zoo

Training on Tufts HPC

Next Steps: Using the model in MATLAB

- Using the Deep Learning Toolbox, we can now import the trained TensorFlow model into MATLAB

- In MATLAB, use the Lidar Toolbox, Automated Driving Toolbox and Sensor Fusion Toolbox to use both **camera images** and **LIDAR data** from the nuScenes dataset

- This integration with MATLAB would enable a more precise pedestrian detection, as well as getting depth information

Potential Next Steps

- Using LIDAR and distance data, classify the **safety** of the pedestrians according to how far they are from the car, where they are in relation to the street, etc.
    - Involves neural network to learn **intent** of pedestrians and cyclists
- This open dataset is very comprehensive and can help computer science and engineering students learn more about deep learning and using multiple sensors to make better predictions.

What We Learned

- Image processing & computer vision
- Deep Learning (both in MATLAB and Python)
- TensorFlow
- Object Detection Models
- XML format (data infrastructure & pipelining)
- Transfer Learning
- HPC
- Teamwork / delegation / collaboration / time management
- Accountability / keeping everyone on the same page
- Working with people from different experience levels


Questions?