

# MATLAB EXPO

## 2021

### Applying AI to Radar and Lidar Processing

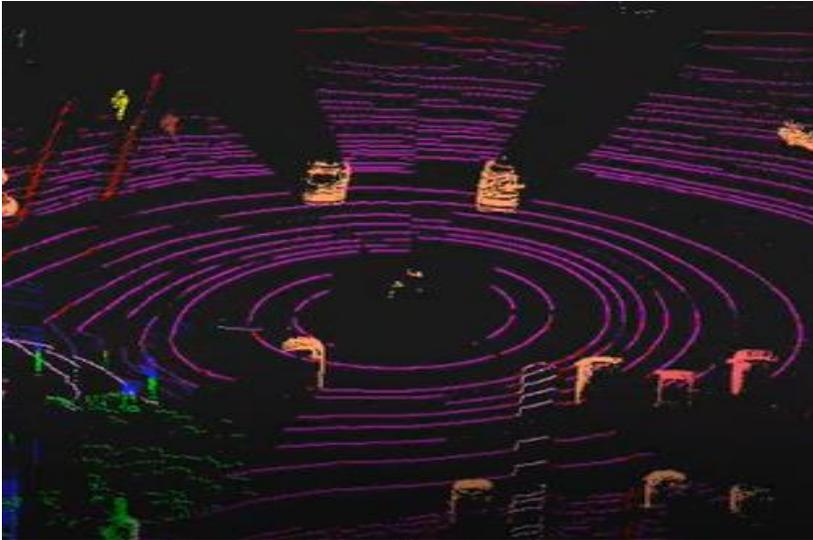
*Rick Gentile*

*Avinash Nehemiah*



# 3 Things We'll Cover Today

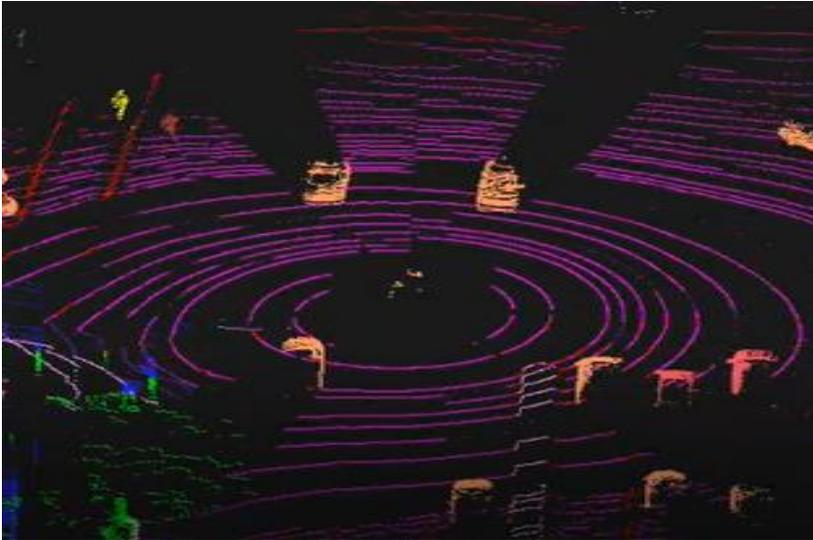
# 3 Things We'll Cover Today



## Insight

*AI Applications for Radar and Lidar*

## 3 Things We'll Cover Today



- Data Synthesis
- Labeling
- Pre-processing
- Model selection and training
- Full system deployment

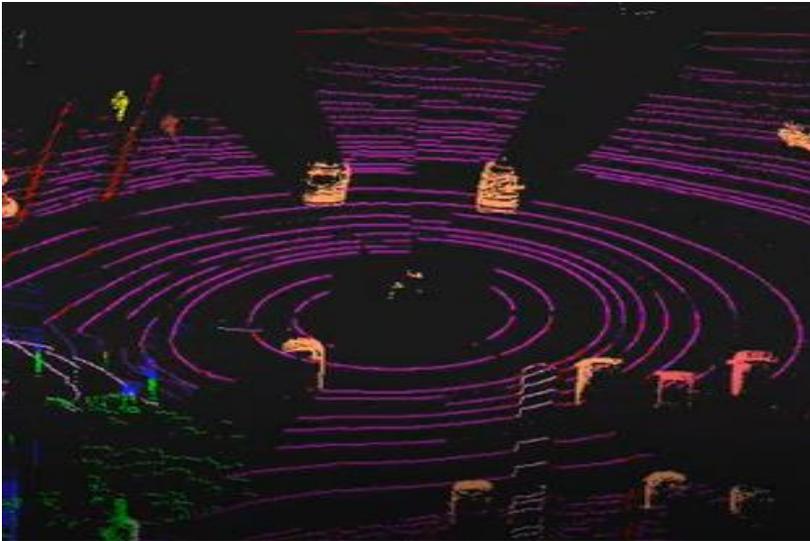
### Insight

*AI Applications for Radar and Lidar*

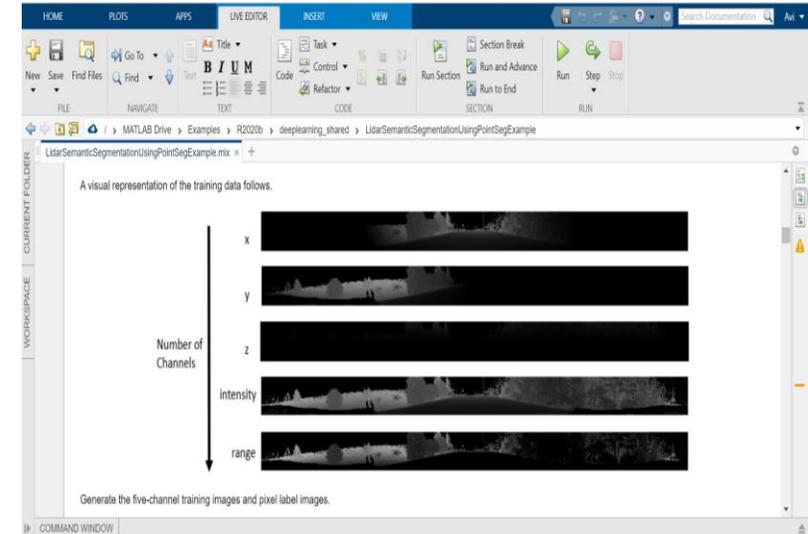
### Challenges

*Common issues engineers face in practice*

# 3 Things We'll Cover Today



- Data Synthesis
- Labeling
- Pre-processing
- Model selection and training
- Full system deployment



## Insight

*AI Applications for Radar and Lidar*

## Challenges

*Common issues engineers face in practice*

## Interaction

*AI models for radar and lidar data*

## What is a lidar sensor and where is AI used ?

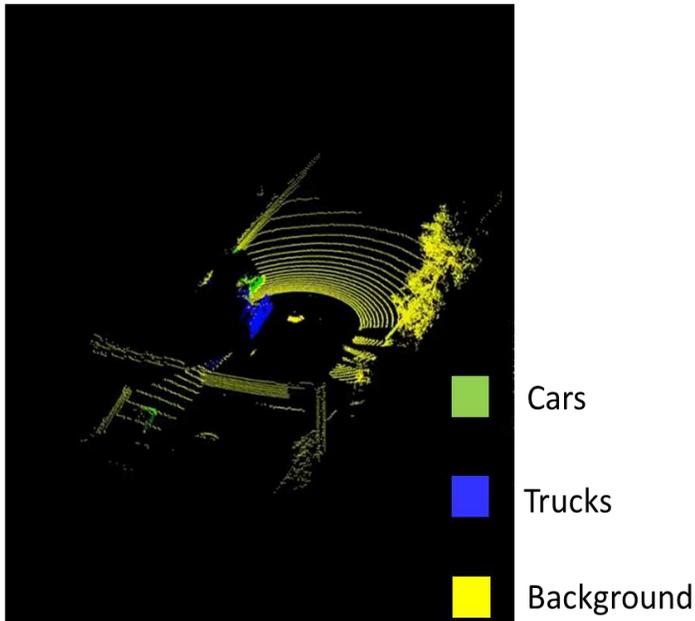
### **Lidar:** *Light detection and ranging*

- Creates 2D or 3D point clouds representing depth using pulsed-light
- Also known as 3D laser scanner, laser scanner

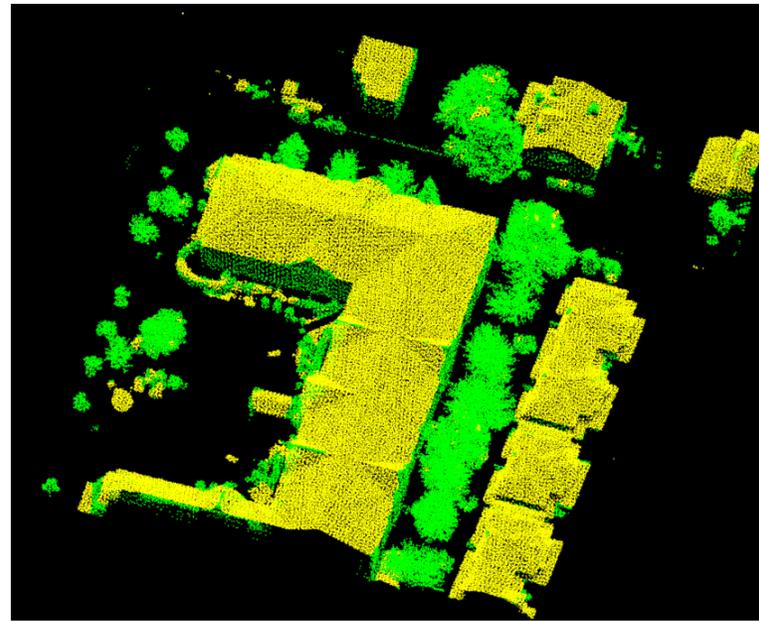
# What is a lidar sensor and where is AI used ?

## Lidar: **L**ight **d**etection **a**nd **r**anging

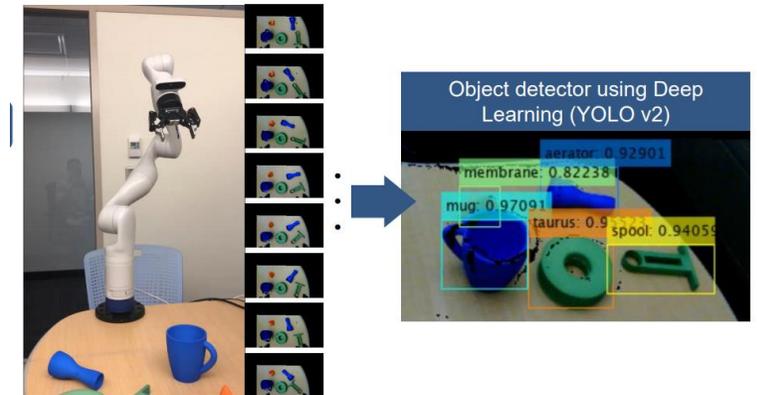
- Creates 2D or 3D point clouds representing depth using pulsed-light
- Also known as 3D laser scanner, laser scanner



Autonomous Perception

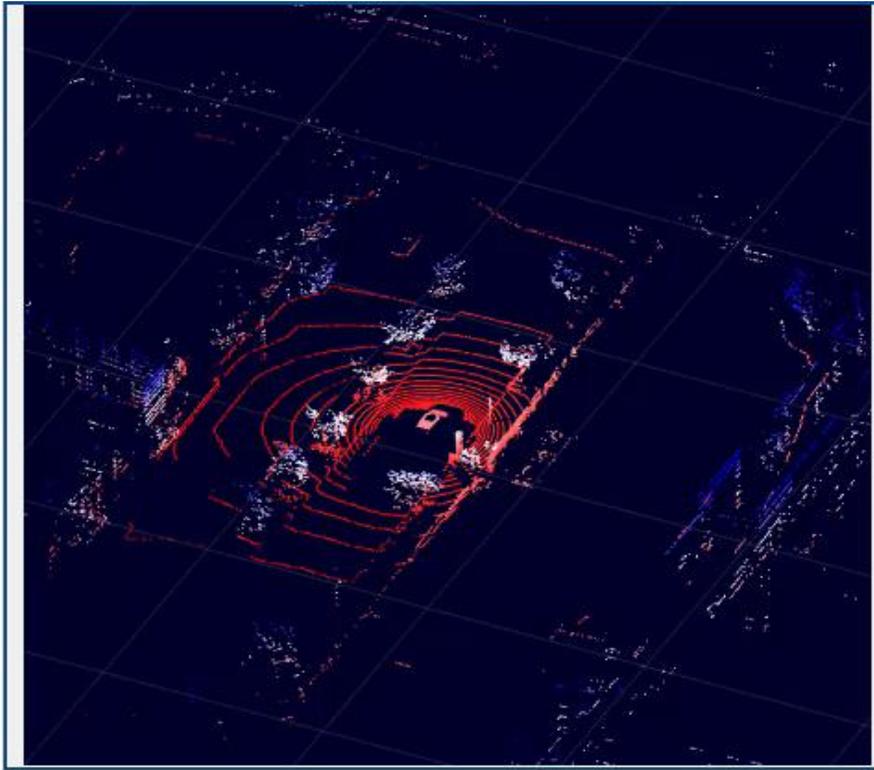


Aerial Imaging and Navigation



Robotics and Augmented Reality

# What are the advantages and disadvantages of lidar sensors ?



Accurate  
Depth



Dense  
Data



## Disadvantages of lidar sensors

- Sensitive to rain, snow and weather effects
- Measurement effected by platform movement/vibration
- Accuracy drops as range increases

## What is a radar sensor and where is AI used ?

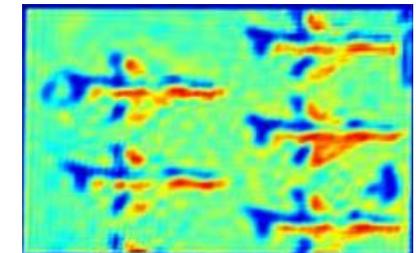
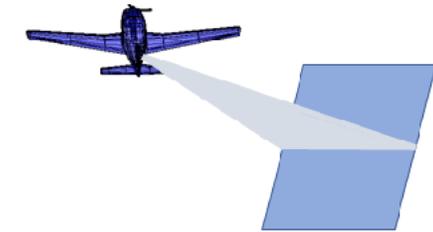
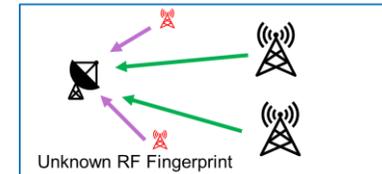
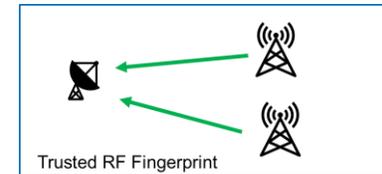
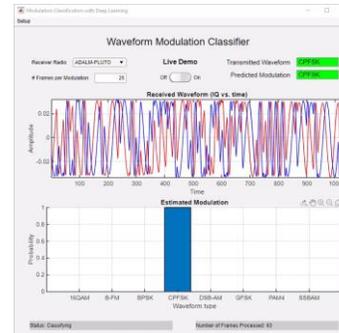
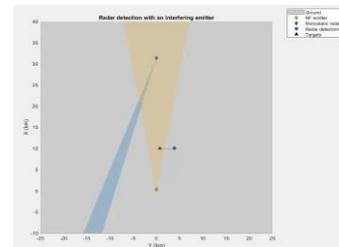
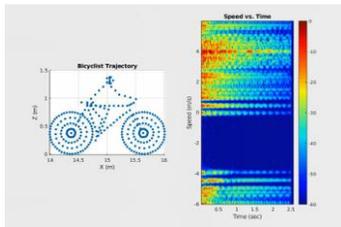
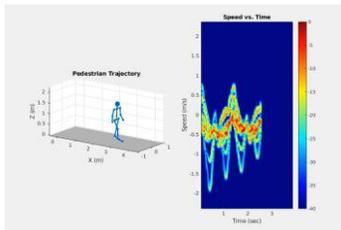
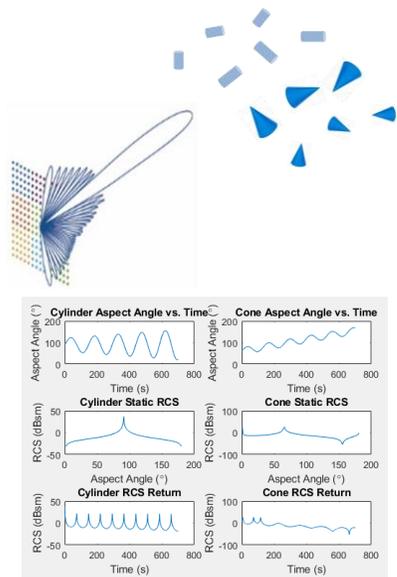
### **Radar: Radio *d*etection *a*nd *r*anging**

- Use radio frequency echos to detect objects at a distance
- Estimate position, Doppler, and micro-Doppler.
- Generate images with 4D radar

# What is a radar sensor and where is AI used ?

## Radar: Radio detection and ranging

- Use radio frequency echos to detect objects at a distance
- Estimate position, Doppler, and micro-Doppler.
- Generate images with 4D radar

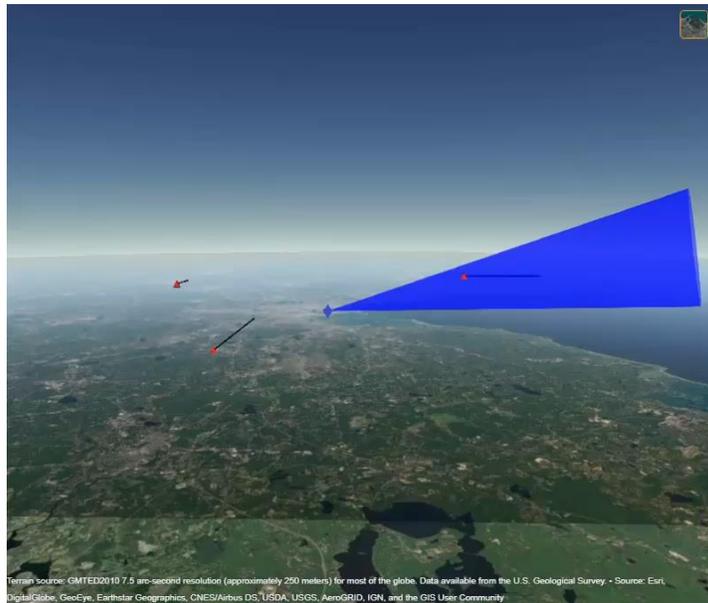


Target classification

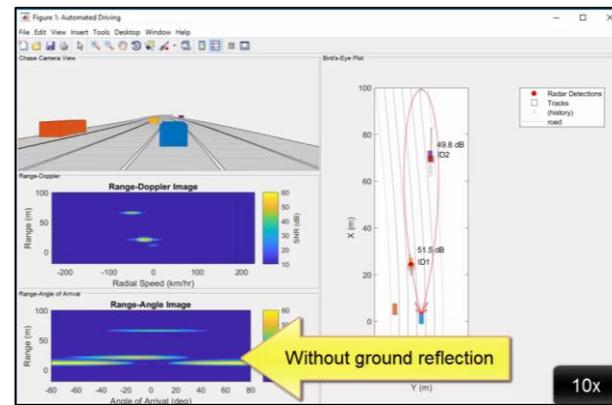
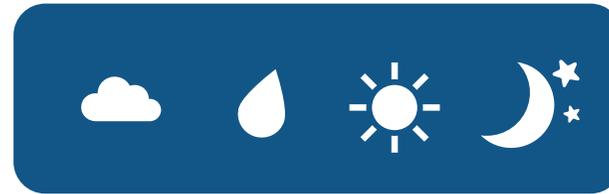
Signal identification

SAR imaging

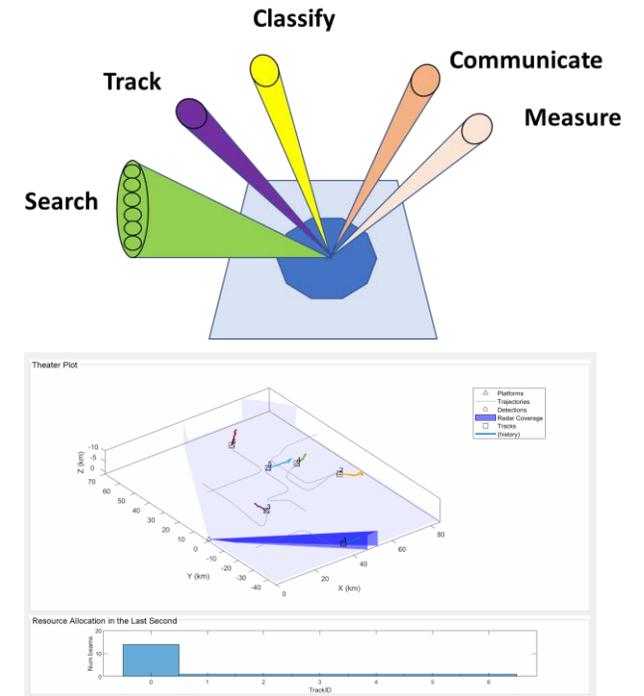
# What are the advantages and disadvantages of radar sensors?



Long range operations



All weather, night and day



Flexibility

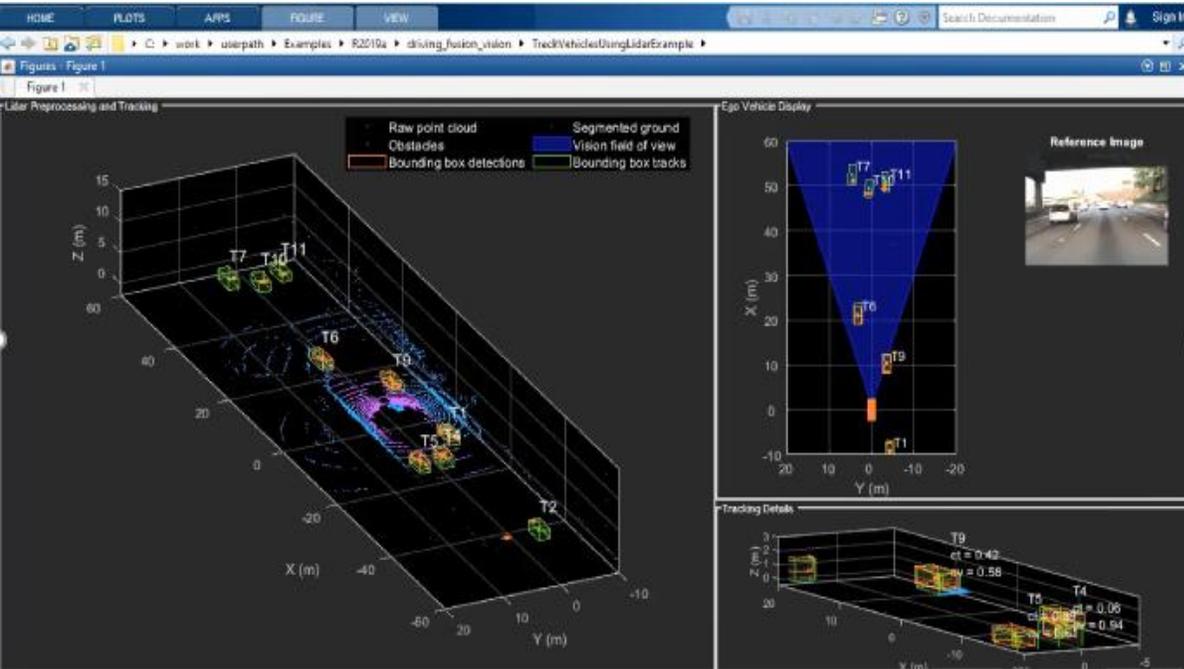
## Disadvantages of radar sensors

- Lower resolution than lidar
- Lower azimuthal resolution at longer ranges
- Multipath and clutter cause ghost detections and false detections

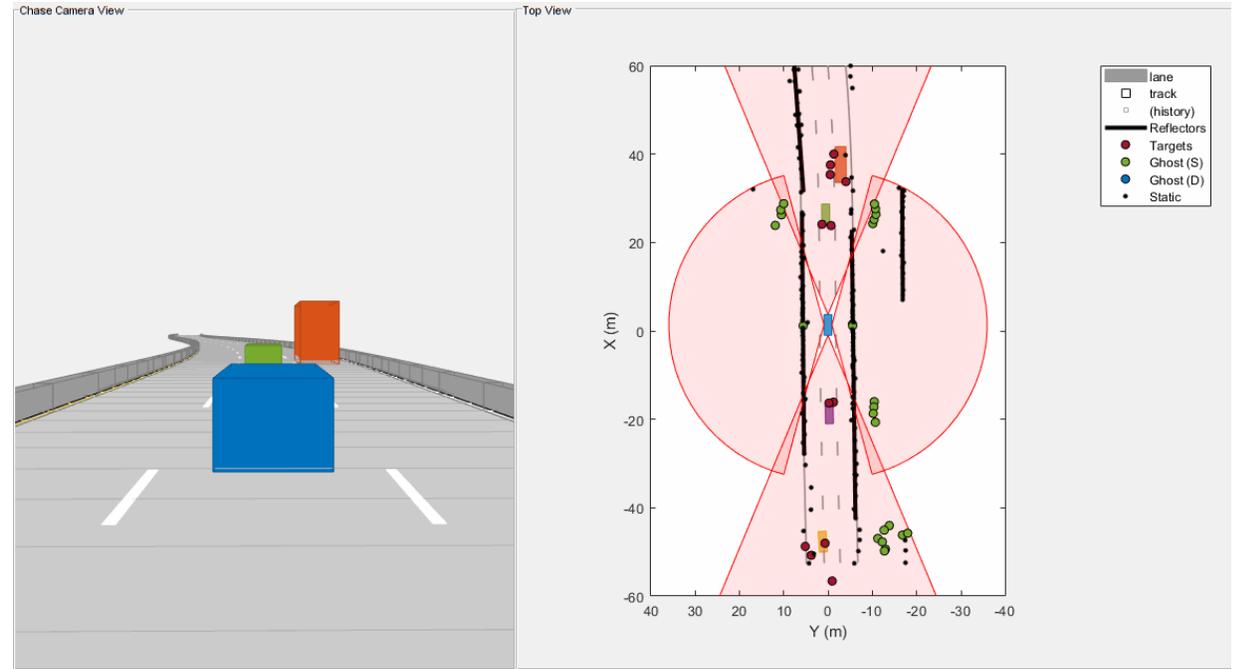
## What are the common challenges engineers face using AI with radar and lidar ?

1. Labeling recorded data for AI training is manual and time consuming
2. Little-no recorded data to train models for safety-critical applications
3. Lack of knowledge on of AI model-type and data formats best results
4. Unclear how to pre-process sensor signals for best results
5. Real-world systems require deployment of more than AI model

# How to overcome challenges using MATLAB and Simulink examples



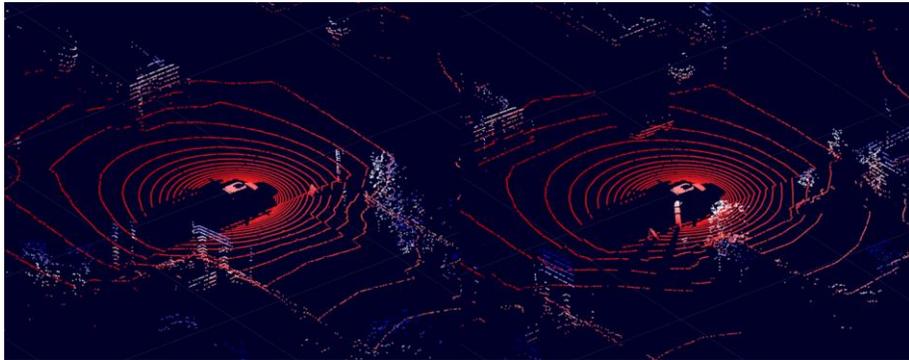
Lidar Detection and Tracking



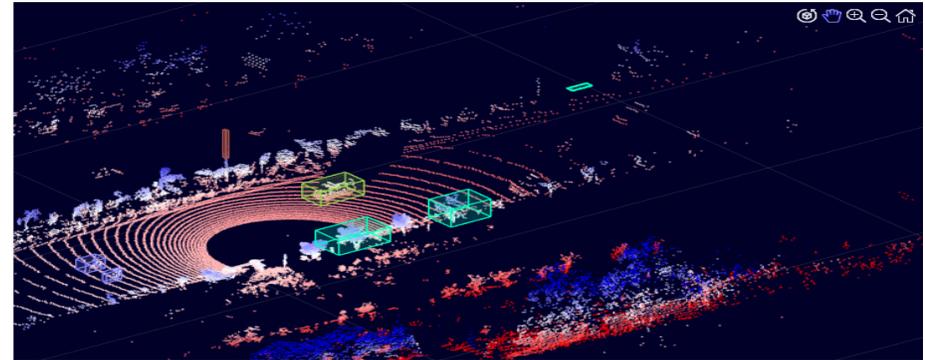
Tracking in the Presence of Radar Multipath

## Challenge

Labeling data is repetitive, manual and time consuming



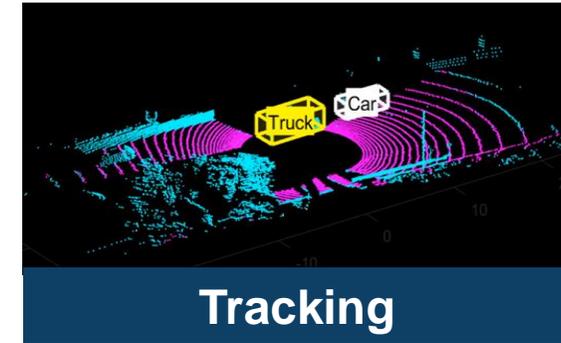
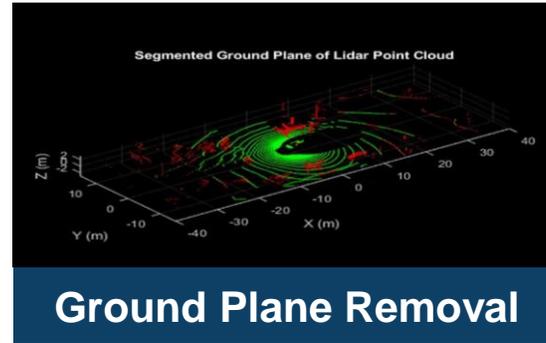
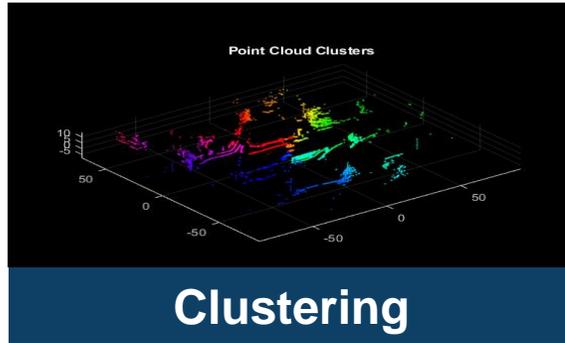
**Repetitive and manual**  
*Very little variation frame-frame*



**Noise**  
*Majority of points not required to train AI model*

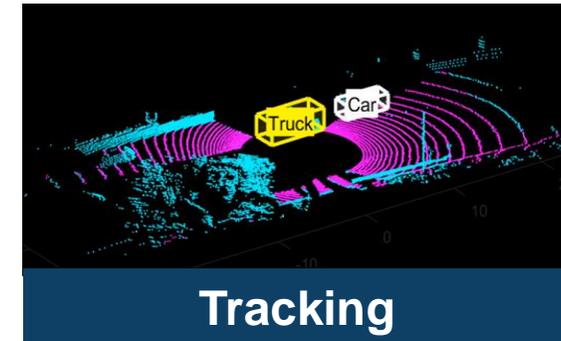
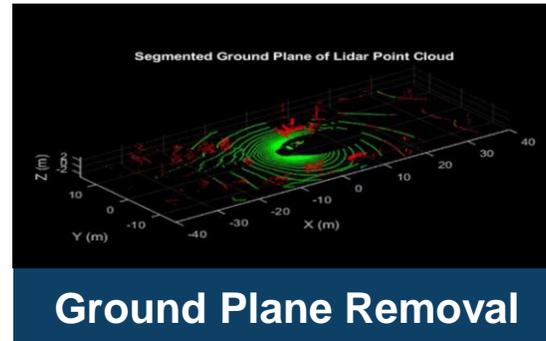
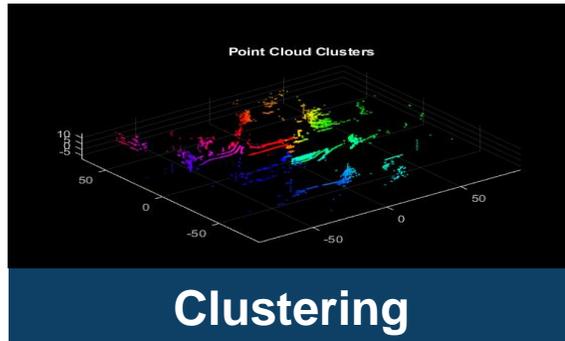
# Two steps to improving accuracy and efficiency of labeling process

## 1. Automation using non-AI techniques

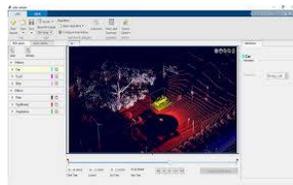


# Two steps to improving accuracy and efficiency of labeling process

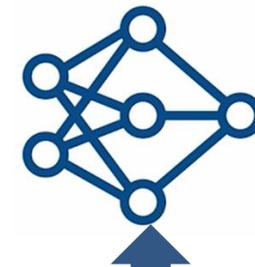
## 1. Automation using non-AI techniques



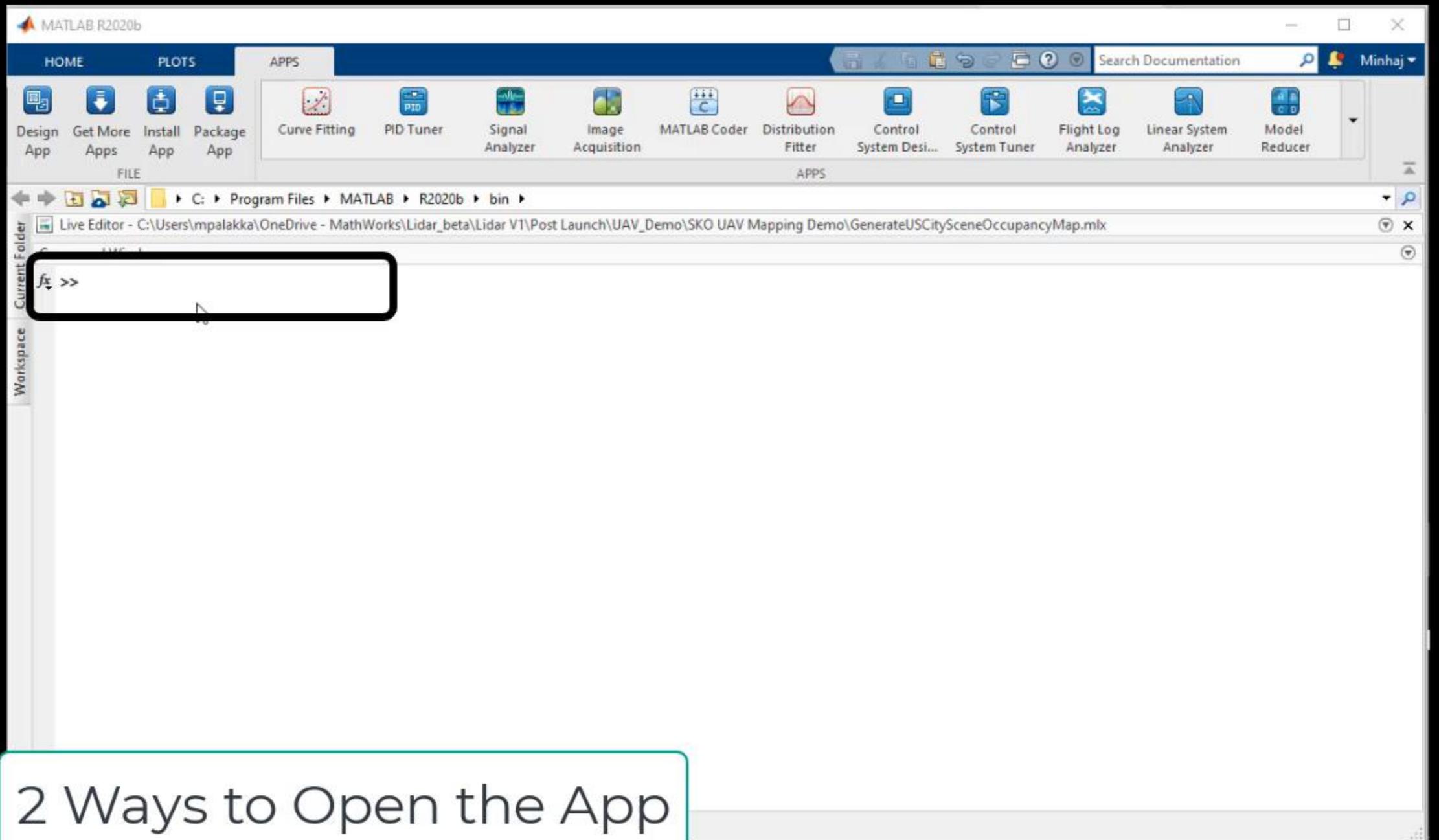
## 2. Iterative training and labeling



Train Model

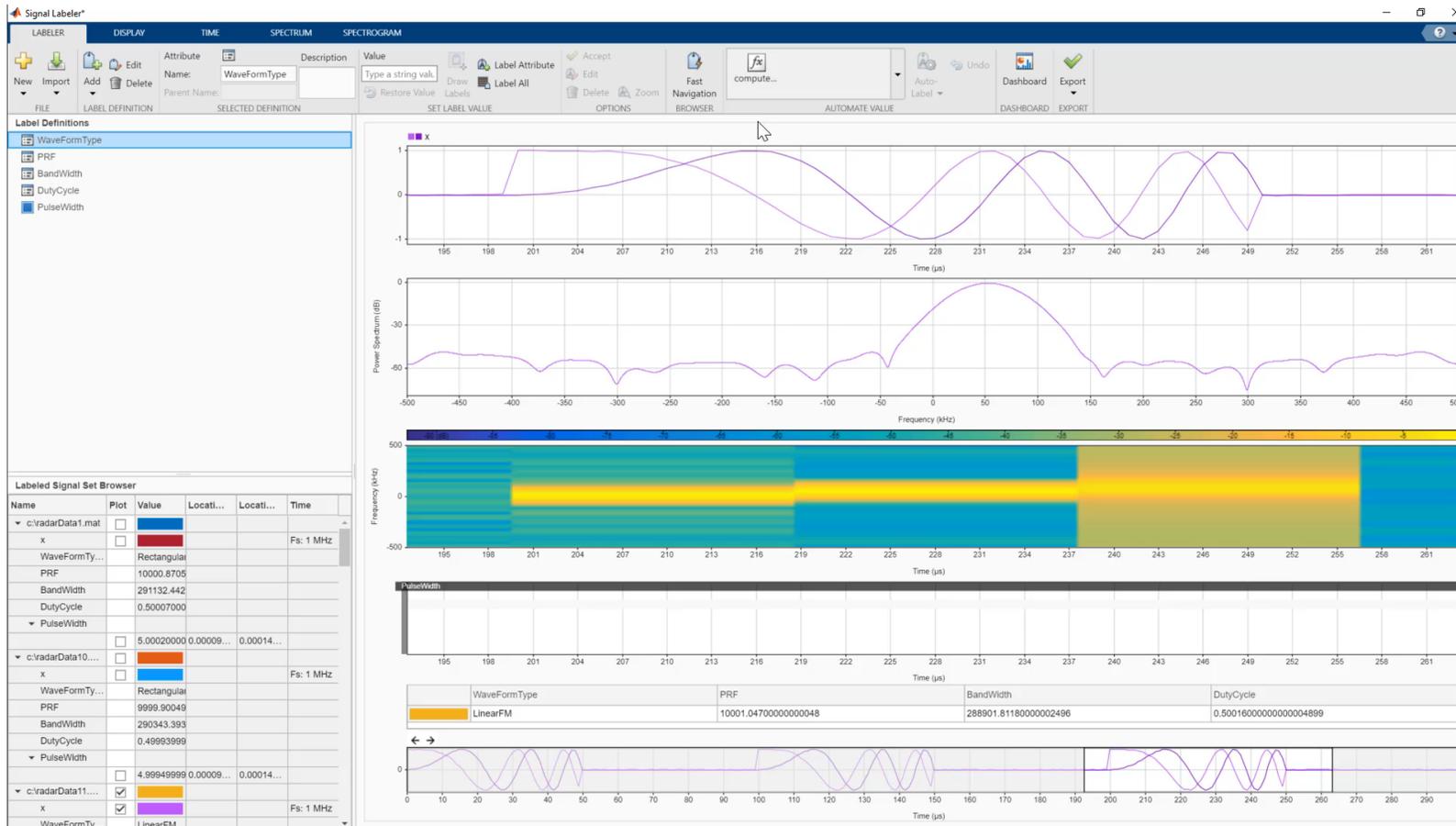


Iteration and Refinement



2 Ways to Open the App

# Labelling radar signals can also be done automatically

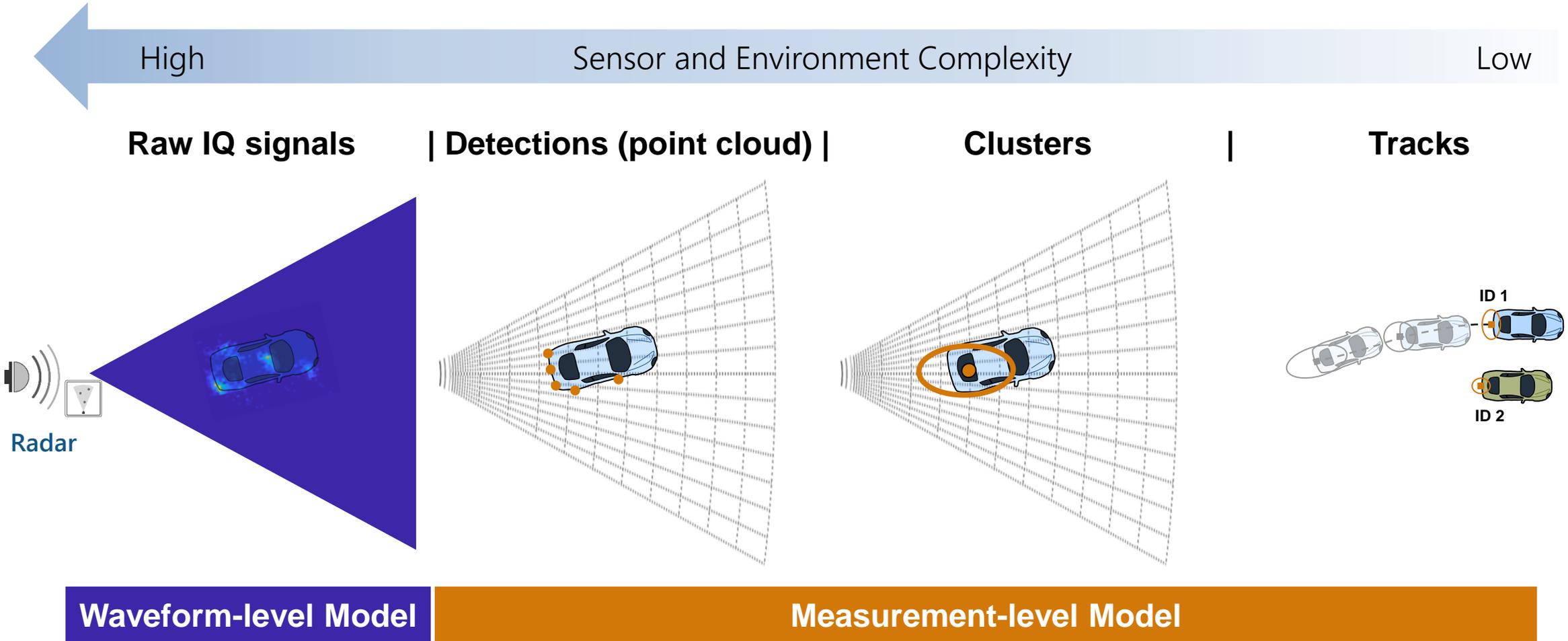


Automatically label signals with custom functions

Explore and label signals with time, frequency, and time-frequency views

Track labelling statistics with integrated dashboards

# Simulating radar data in MATLAB and Simulink



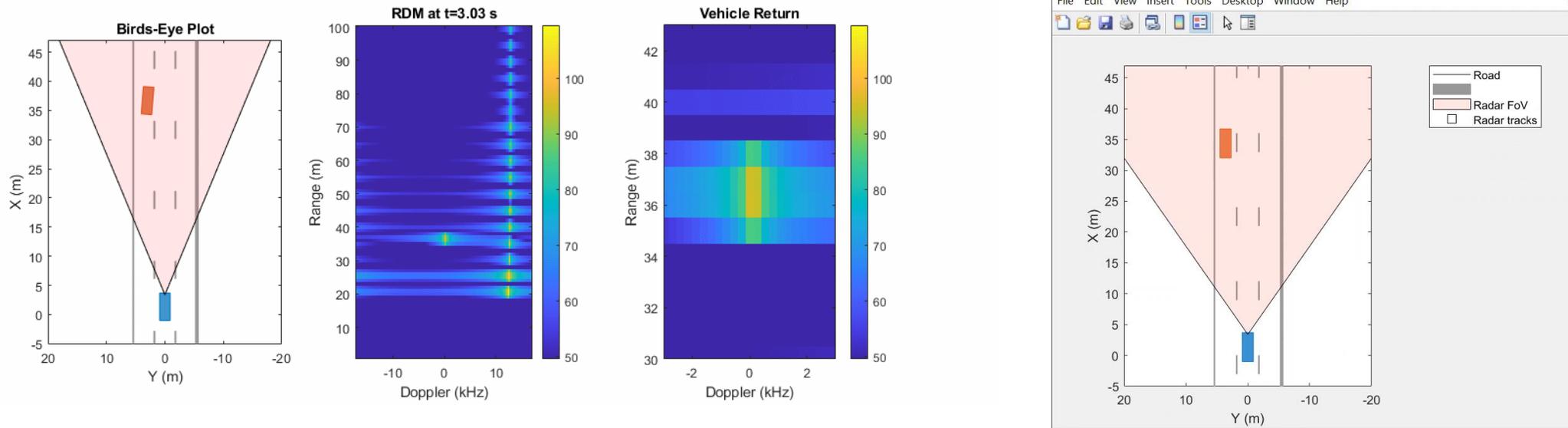
# Simulating radar data in MATLAB and Simulink



**Raw IQ signals**

**| Detections (point cloud) |**

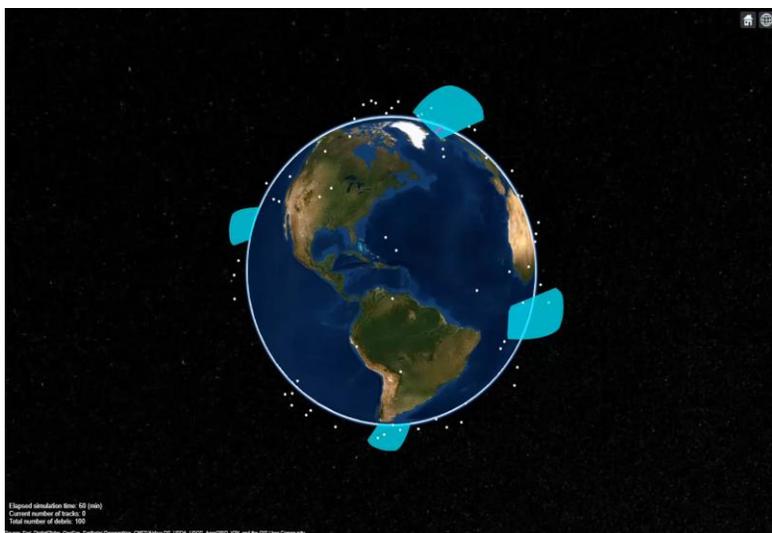
**Tracks**



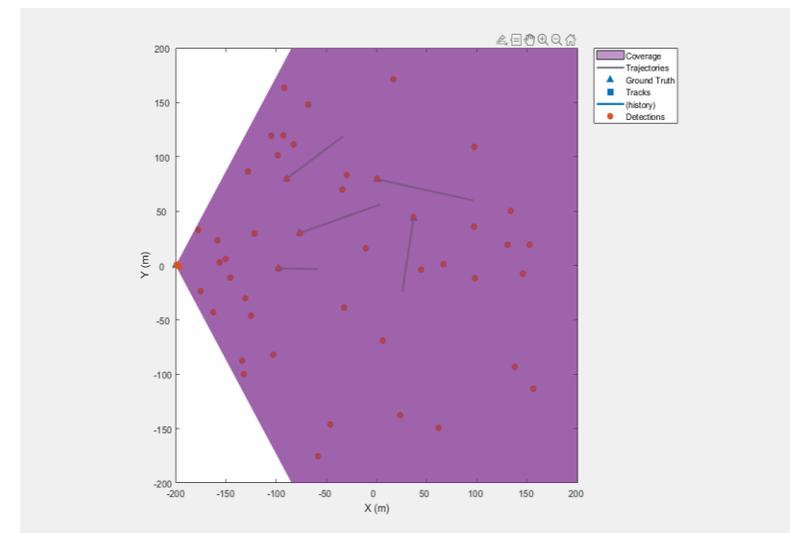
**Waveform-level Model**

**Measurement-level Model**

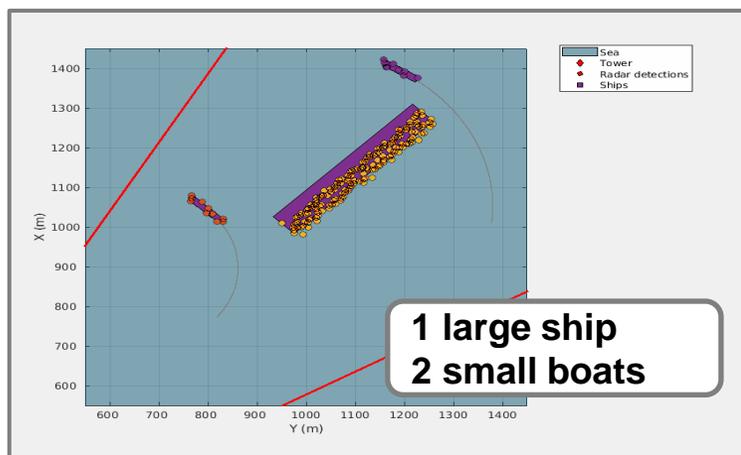
# Wide range of data synthesis options for radar systems



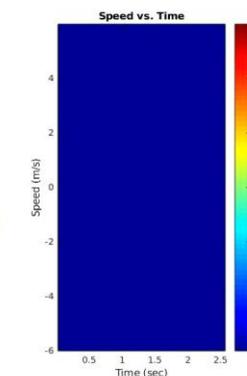
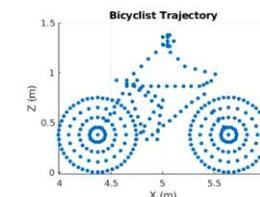
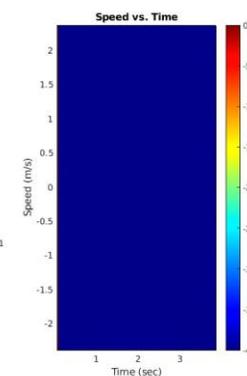
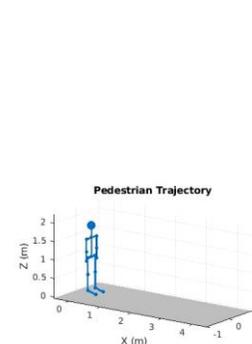
Long distance, multi-object operations



High clutter environments



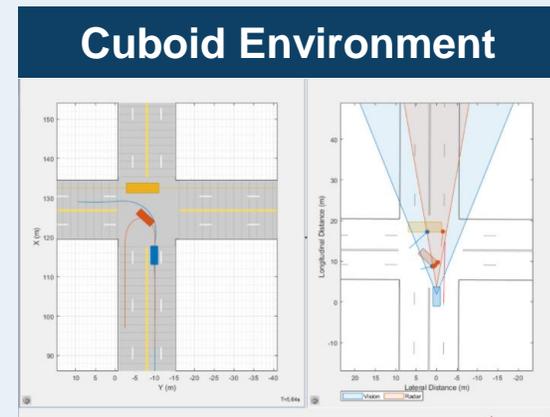
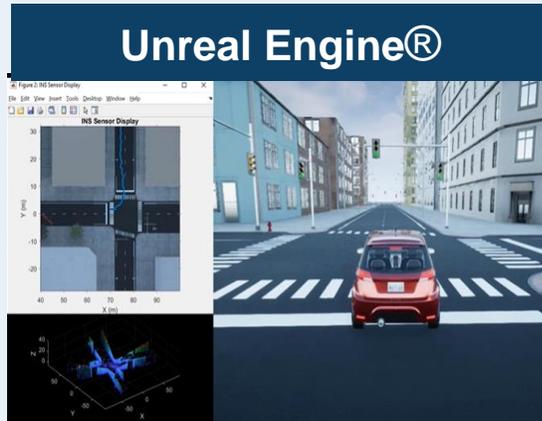
Extended objects



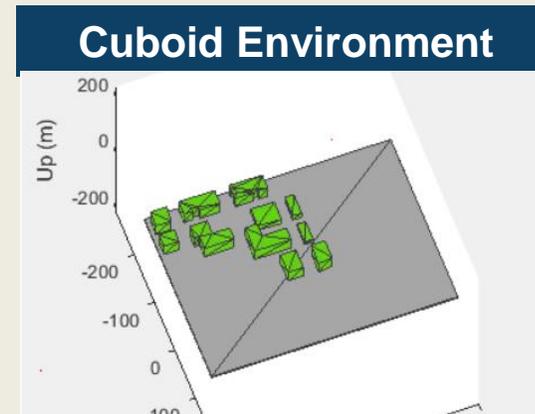
Micro-Doppler signatures

# Simulating lidar sensor data in MATLAB and Simulink

## Automated Driving Toolbox



## UAV Toolbox



## 3D Scene Creation



# Challenge

Lack of knowledge on combination of model-type and data format best results

arXiv:1710.07368v1 [cs.CV] 19 Oct 2017

**SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LIDAR Point Cloud**

Bichen Wu, Alvin Wan, Xiangyu Yue and Kurt Keutzer  
UC Berkeley  
{bichen, alvinwan, yyue, keutzer}@berkeley.edu

**Abstract**—In this paper, we address automatic segmentation of road-objects from 3D LIDAR point clouds. In particular, we wish to detect and categorize instances of interest, such as cars, pedestrians and cyclists. We formulate this problem as a point-wise classification problem, and propose an end-to-end pipeline called SqueezeSeg based on convolutional neural networks (CNN). The CNN takes a transformed LIDAR point cloud as input and directly outputs a point-wise label map, which is then refined by a conditional random field (CRF) implemented as a recurrent layer. Instance-level labels are then obtained by conventional clustering algorithms. Our CNN model is trained on LIDAR point clouds from the KITTI [3] dataset, and our point-wise segmentation labels are derived from 2D bounding boxes from KITTI. To obtain extra training data, we built a LIDAR simulator using Grand Theft Auto V (GTA V) as a popular video game, to synthesize large amounts of realistic training data. Our experiments show that SqueezeSeg achieves high accuracy with substantially fast and stable runtime (0.7 s/100 ms per frame), highly desirable for autonomous driving applications. Furthermore, additionally training on synthesized data boosts validation accuracy on real-world data. Our source code and synthesized data will be open-sourced.

**1. INTRODUCTION**

Autonomous driving systems rely on accurate, real-time and robust perception of the environment. An autonomous vehicle needs to accurately categorize and locate “road-objects”, which we define to be driving-related objects such as cars, pedestrians, cyclists, and other obstacles. Different autonomous driving solutions may have different combinations of sensors, but the 3D LIDAR sensor is one of the most prevalent components. LIDAR scanners directly produce distance measurements of the environment, which are then used by vehicle controllers and planners. Moreover, LIDAR scanners are robust under almost all lighting conditions, whether it be day or night, with or without glare and shadows. As a result, LIDAR based perception tasks have attracted significant research attention.

In this work, we focus on road-object segmentation using (VeloEye) 3D LIDAR point clouds. Given point cloud data output from a LIDAR scanner, the task aims to isolate objects of interest and predict their categories, as shown in Fig. 1. Previous approaches comprise or use parts of the following stages: Remove the ground, cluster the remaining points into instances, extract (hand-crafted) features from each cluster, and classify each cluster based on its features. This workflow, despite its popularity [2], [3], [4], [5] has several disadvantages: a) Ground segmentation in the above

arXiv:1812.05784v2 [cs.LG] 7 May 2019

**PointPillars: Fast Encoders for Object Detection from Point Clouds**

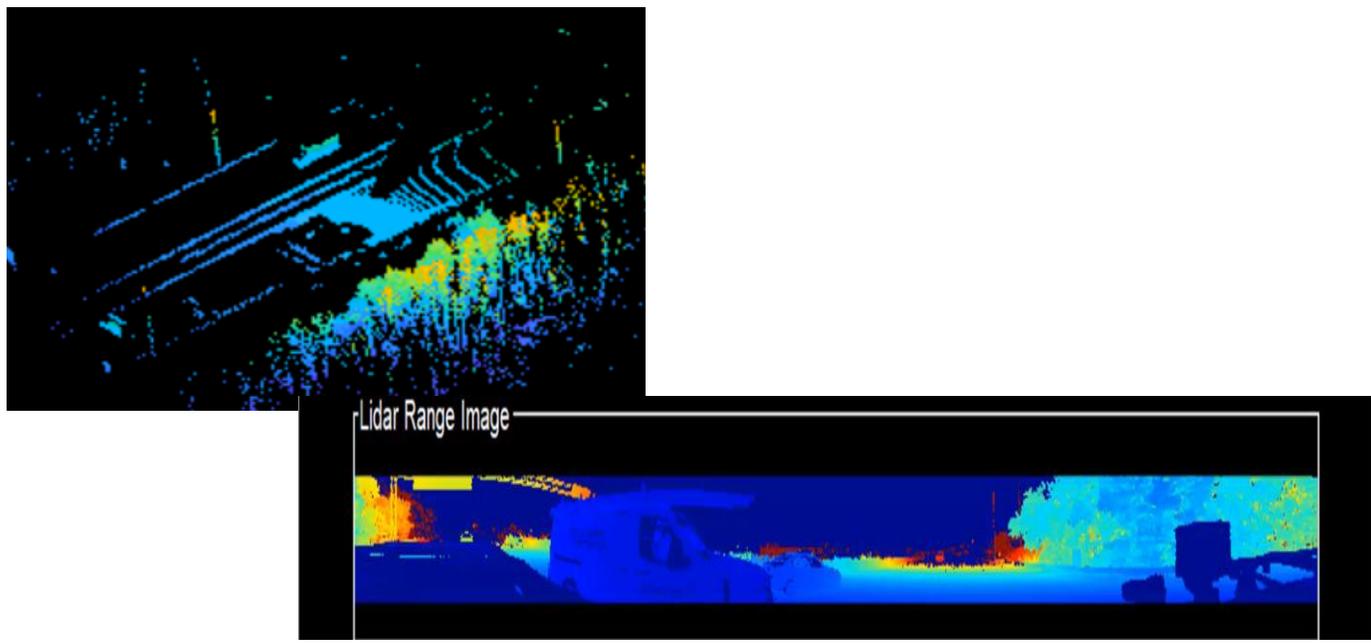
Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, Oscar Beijbom  
MOTOMY, an APTIV company  
{alex, sourabh, holger, lubing, jiong, yang, oscar}@motomy.com

**Abstract**

Object detection in point clouds is an important aspect of many robotics applications such as autonomous driving. In this paper we consider the problem of encoding a point cloud into a format appropriate for a downstream detection pipeline. Recent literature suggests two types of encoders; fast encoders tend to be fast but sacrifice accuracy, while encoders that are learned from data are more accurate but slower. In this work we propose PointPillars, a novel encoder which utilizes PointNet to learn a representation of point clouds organized in vertical columns (pillars). While the encoded features can be used with any standard 2D convolutional detection architecture, we further propose a lean downstream network. Extensive experimentation shows that PointPillars outperforms previous encoders with respect to both speed and accuracy by a large margin. Despite only using lidar, our full detection pipeline significantly outperforms the state of the art, even among fusion methods, with respect to both the 3D and bird’s eye view KITTI benchmarks. This detection performance is achieved while running at 62 Hz: a 2-4 fold runtime improvement. A faster version of our method matches the state of the art at 105 Hz. These benchmarks suggest that PointPillars is an appropriate encoding for object detection in point clouds.

**1. Introduction**

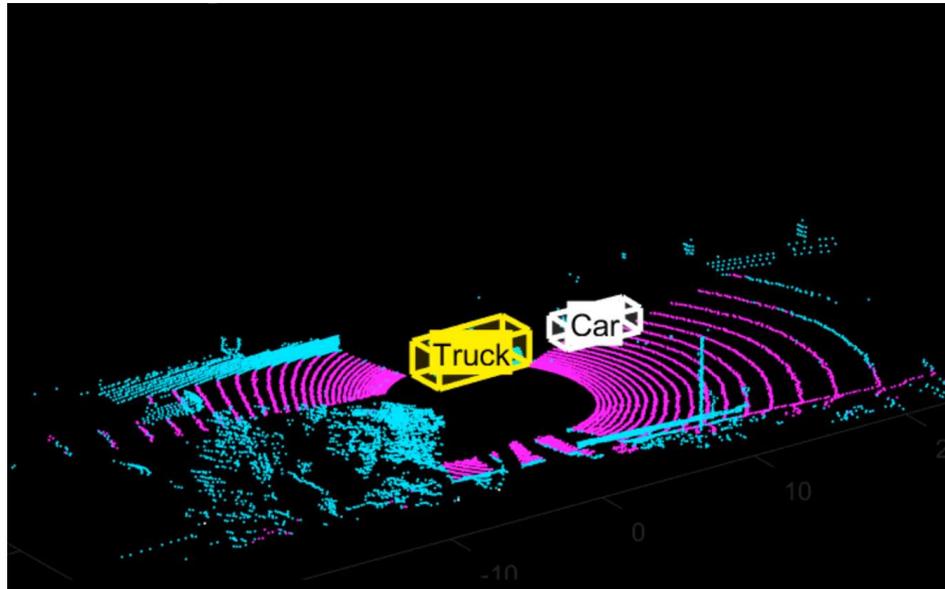
Deploying autonomous vehicles (AVs) in urban environments poses a difficult technological challenge. Among other tasks, AVs need to detect and track moving objects such as vehicles, pedestrians, and cyclists in real-time. To achieve this, autonomous vehicles rely on several sensors out of which the lidar is arguably the most important. A lidar uses a laser scanner to measure the distance to the environment, thus generating a sparse point cloud representation. Traditionally, a lidar robotics pipeline interprets such point clouds as object detections through a bottom-up pipeline involving background subtraction, followed by spatiotemporal clustering and classification [1], [2].



What model do I use ?  
There are so many research papers.

How do I train a model ?  
Raw sensor data or transformed.

# MATLAB provides a curated library of models with different inputs and styles



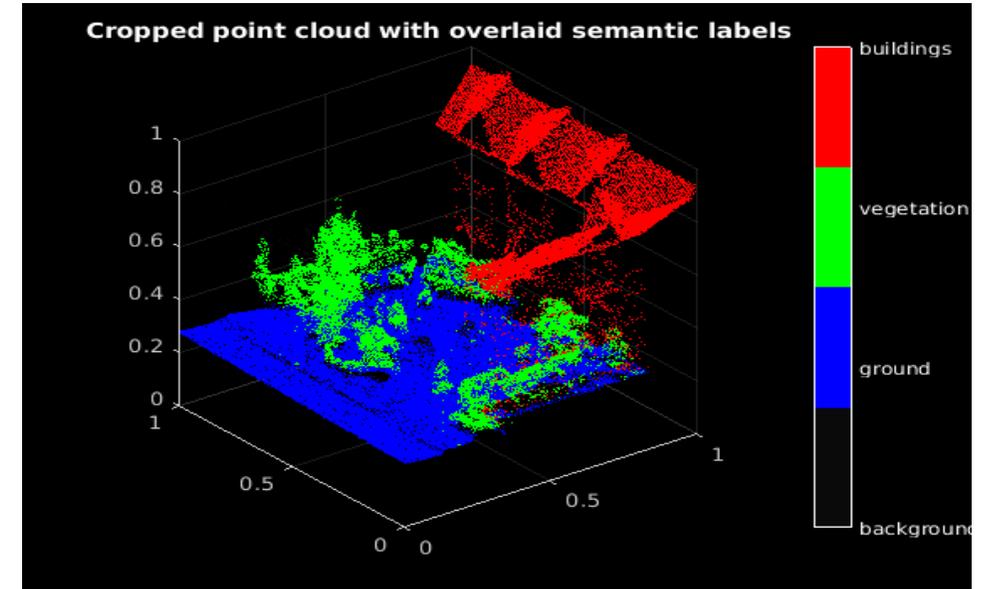
## Object Detection

3D bounding box detection and classification

### Curated Models

1. PointPillars

Raw  
Data



## Semantic Segmentation

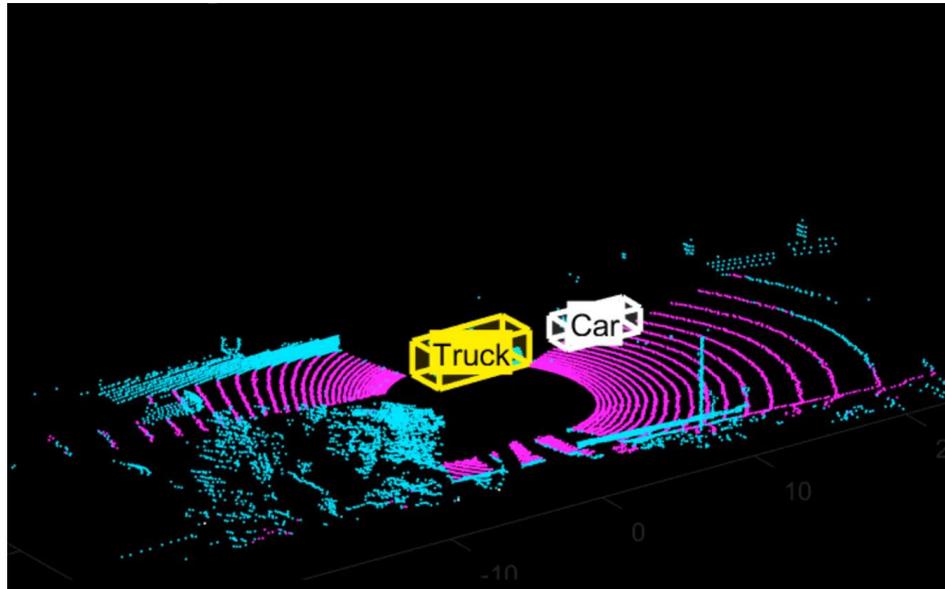
Classify each data point with label

### Curated Models

1. SqueezeSeg v2
2. PointSeg
3. SalsaNext
4. PointNet
5. PointNet++

Image  
Data

# MATLAB provides a curated library of models with different inputs and styles



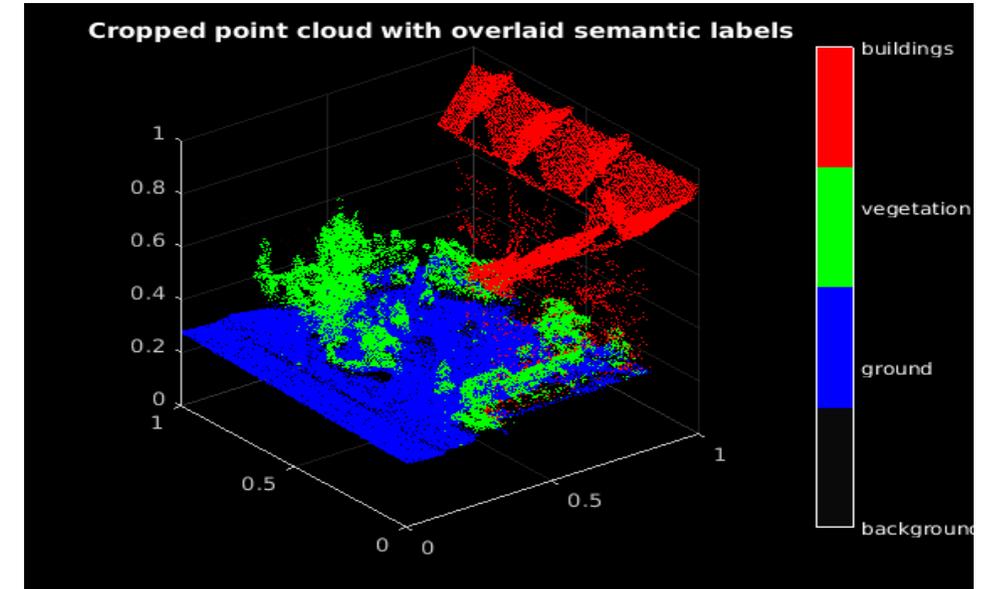
## Object Detection

3D bounding box detection and classification

### Curated Models

1. PointPillars

Raw  
Data



## Semantic Segmentation

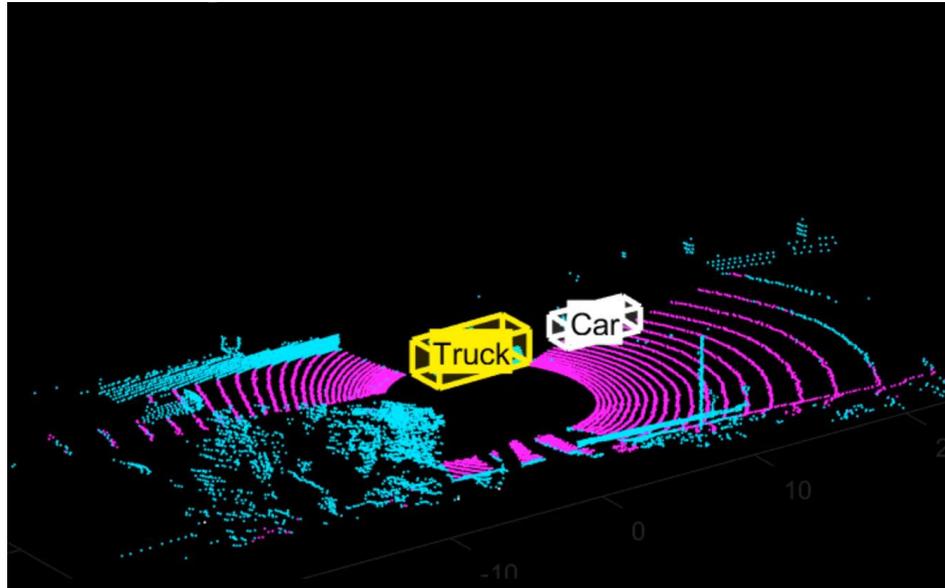
Classify each data point with label

### Curated Models

1. SqueezeSeg v2
2. PointSeg
3. SalsaNext
4. PointNet
5. PointNet++

Image  
Data

# MATLAB provides a curated library of models with different inputs and styles

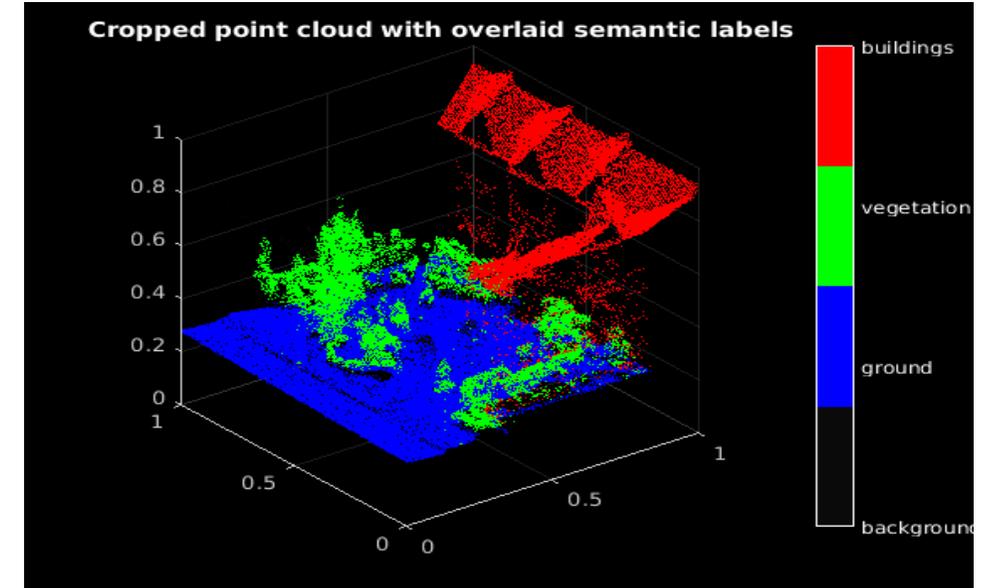


## Object Detection

3D bounding box detection and classification

### Curated Models

1. PointPillars



## Semantic Segmentation

Classify each data point with label

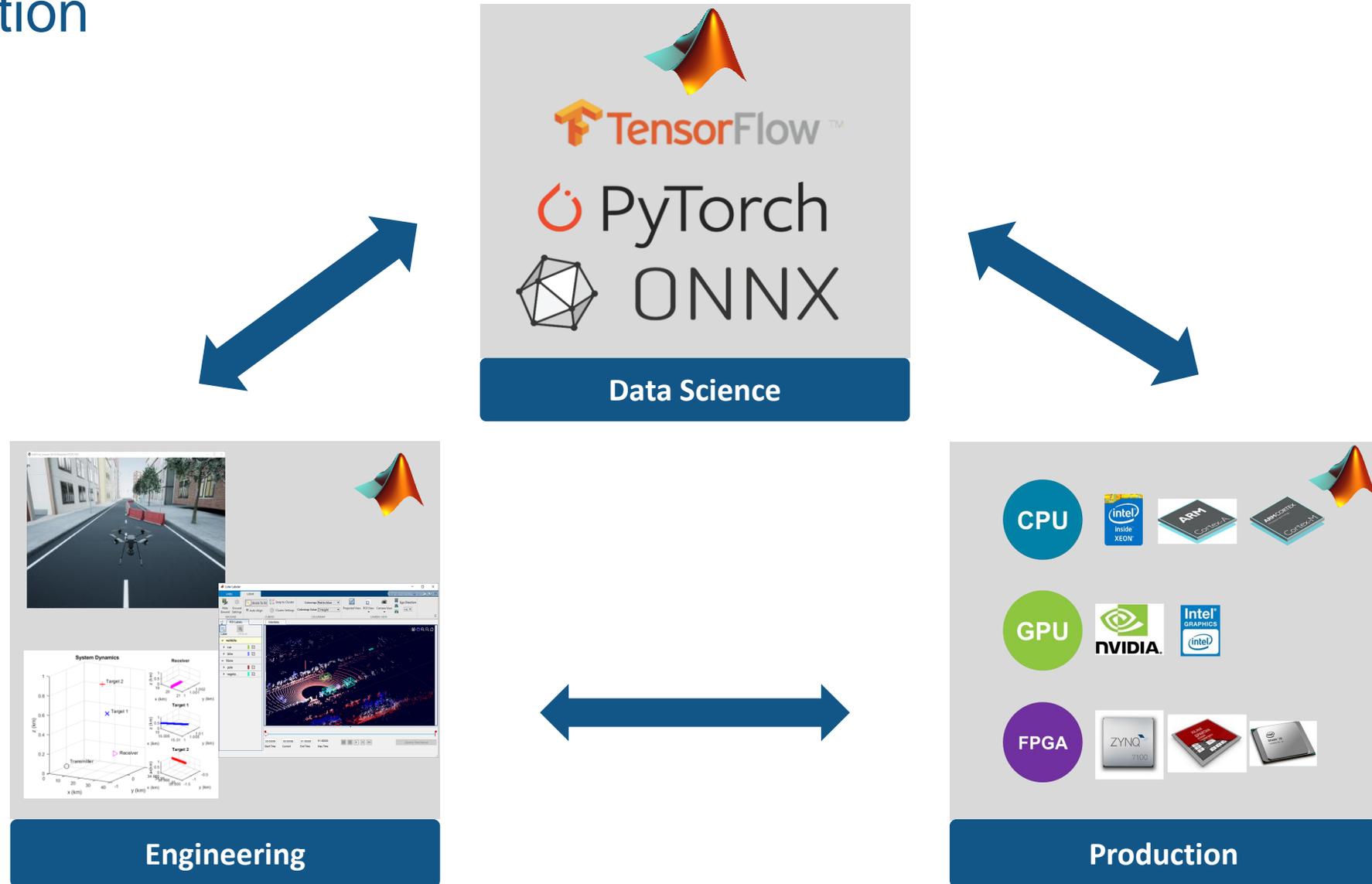
### Curated Models

1. SqueezeSeg v2
2. PointSeg
3. SalsaNext
4. PointNet
5. PointNet++

Interoperability bridges the gap between data science, engineering and production



# Interoperability bridges the gap between data science, engineering and production



# Lidar 3-D Object Detection Using PointPillars Deep Learning

## Load Data

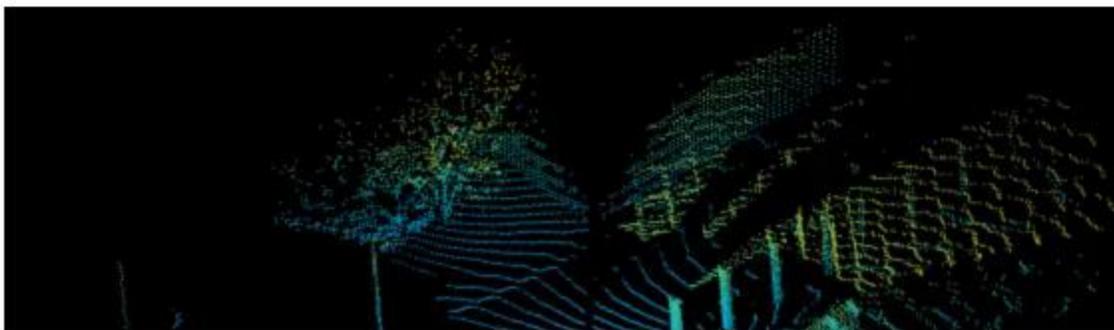
```
1 lidarURL = 'https://www.mathworks.com/supportfiles/lidar/data/WPI_LidarData.tar.gz';  
2 lidarData = downloadWPIData(outputFolder, lidarURL);
```

Load the 3-D bounding box labels.

```
3 load('WPI_LidarGroundTruth.mat', 'bboxGroundTruth');  
4 Labels = timetable2table(bboxGroundTruth);  
5 Labels = Labels(:,2:end);
```

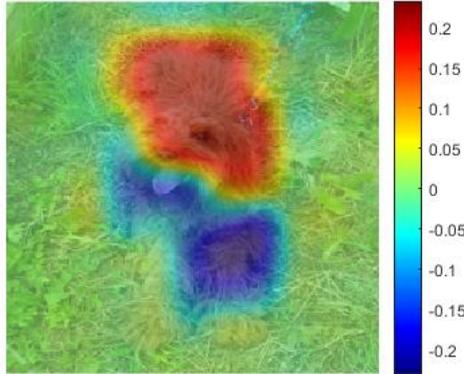
Display the full-view point cloud.

```
6 figure  
7 ax = pcshow(lidarData{1,1}.Location);  
8 set(ax, 'XLim', [-50 50], 'YLim', [-40 40]);  
9 zoom(ax, 2.5);  
10 axis off;
```



# Interpret models and explain network predictions

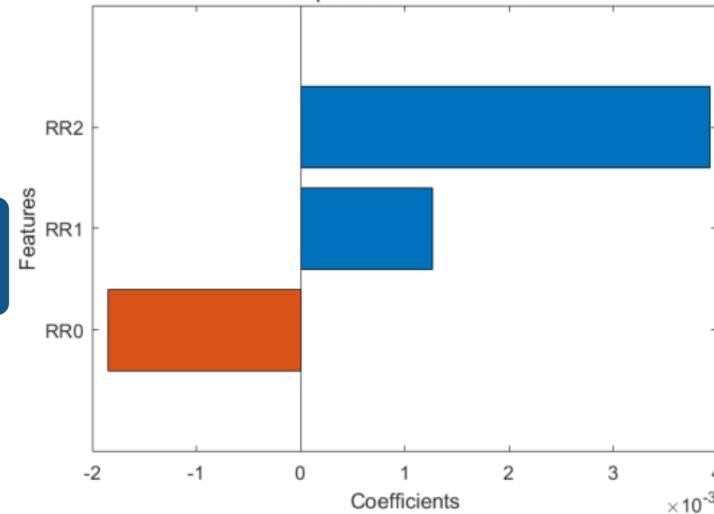
Occlusion sensitivity (miniature poodle)



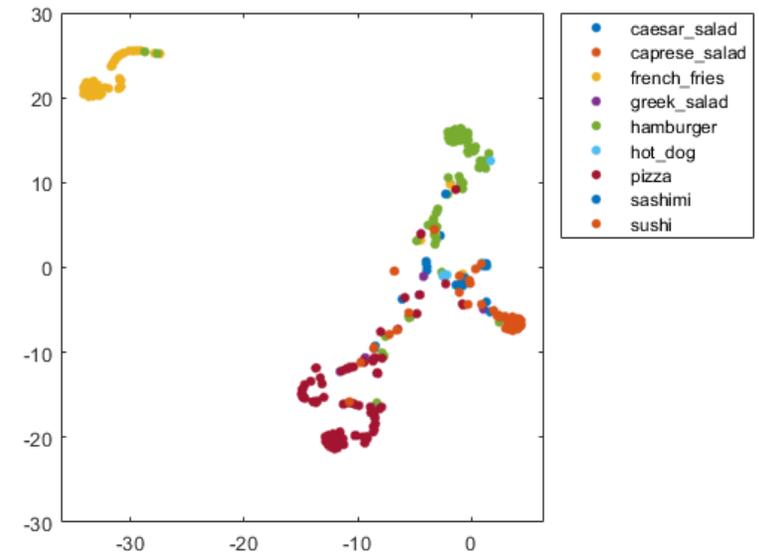
Prediction Explainer Visualization

LIME using Linear Model

Blackbox Model Prediction: 0  
Simple Model Prediction: 0



Model-specific Interpretability



Evaluate Data Separation

# Tune hyperparameters and reproduce training experiments

The screenshot shows the MATLAB Experiment Manager interface. The main window displays the results of a hyperparameter tuning experiment for a DigitsClassifier. The experiment is titled "Baseline Tuning" and is currently running. The results table shows 16 trials, with 7 completed, 1 running, and 8 queued. The table columns include Trial, Status, Progress, Elapsed Time, myInitialLearn..., convFilterSize, Training Accu..., Training Loss, and Validation Ac....

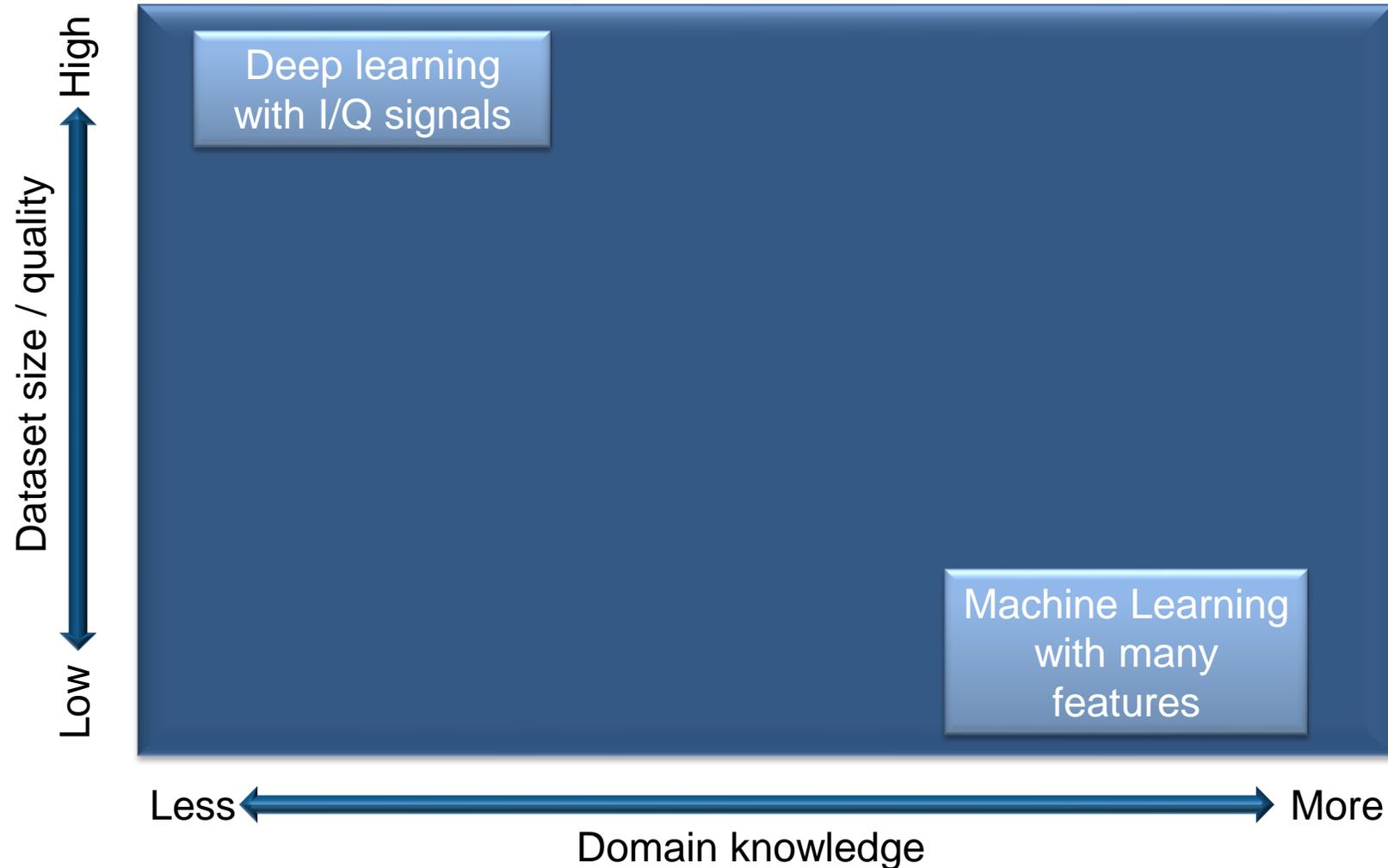
Trial	Status	Progress	Elapsed Time	myInitialLearn...	convFilterSize	Training Accu...	Training Loss	Validation Ac...
1	Complete	100.0%	0 hr 0 min 16 sec	1.0000e-6	3.0000	12.5000	2.6441	10.
2	Complete	100.0%	0 hr 0 min 15 sec	1.0000e-5	3.0000	25.7813	2.1228	20.
3	Complete	100.0%	0 hr 0 min 14 sec	0.0001	3.0000	64.8438	1.0878	42.
4	Complete	100.0%	0 hr 0 min 16 sec	0.0005	3.0000	90.6250	0.4648	49.
5	Complete	100.0%	0 hr 0 min 15 sec	1.0000e-6	4.0000	11.7188	2.4967	6.
6	Complete	100.0%	0 hr 0 min 15 sec	1.0000e-5	4.0000	23.4375	2.1213	14.
7	Complete	100.0%	0 hr 0 min 17 sec	0.0001	4.0000	72.6563	1.0283	39.
8	Running	30.7%	0 hr 0 min 4 sec	0.0005	4.0000			
9	Queued	0.0%		1.0000e-6	5.0000			
10	Queued	0.0%		1.0000e-5	5.0000			
11	Queued	0.0%		0.0001	5.0000			
12	Queued	0.0%		0.0005	5.0000			
13	Queued	0.0%		1.0000e-6	6.0000			
14	Queued	0.0%		1.0000e-5	6.0000			
15	Queued	0.0%		0.0001	6.0000			
16	Queued	0.0%		0.0005	6.0000			

## Interactivity

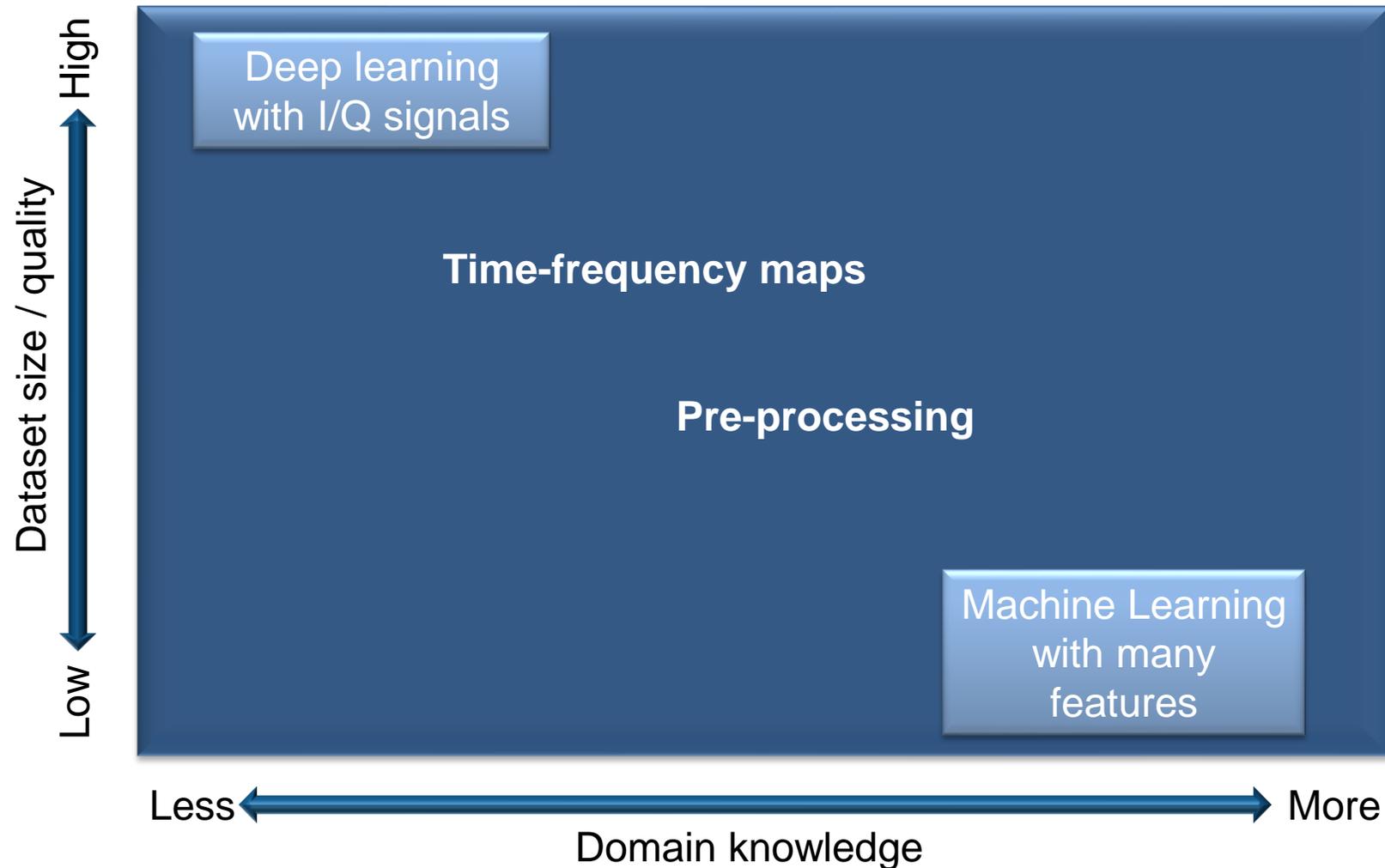
Let's play with an AI model for lidar in MATLAB Online

Pre-processing radar data can improve performance of network

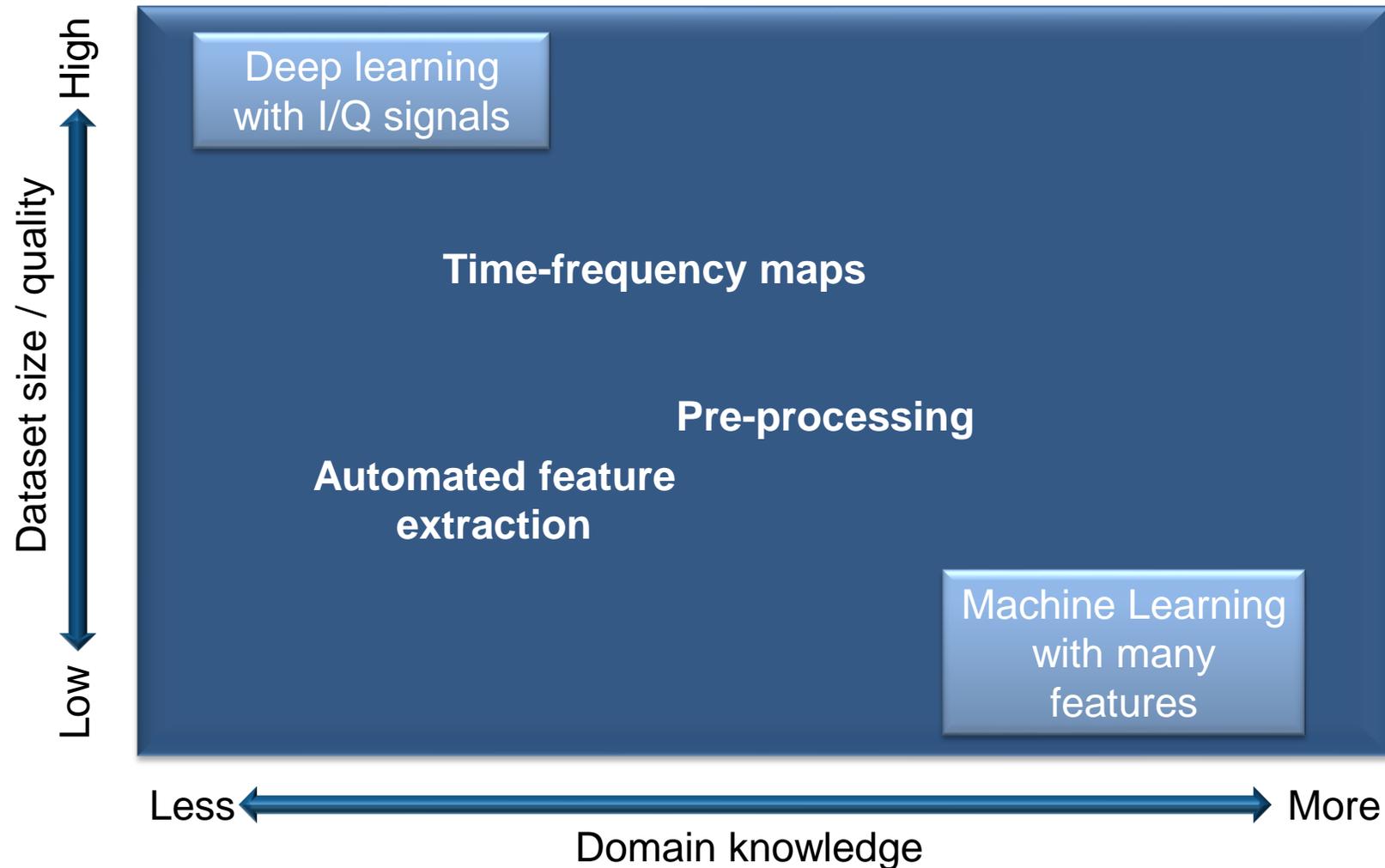
# Pre-processing radar data can improve performance of network



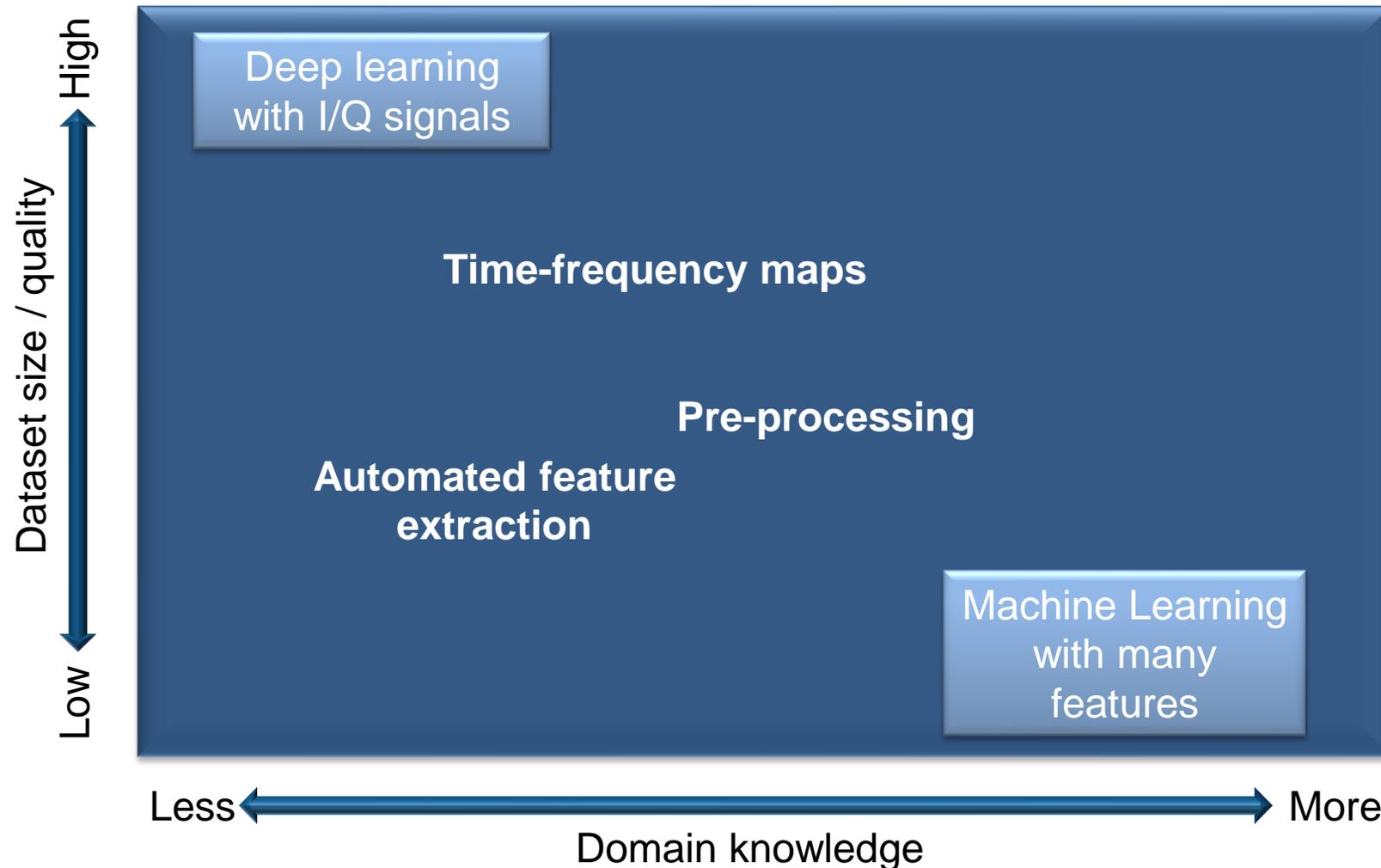
# Pre-processing radar data can improve performance of network



# Pre-processing radar data can improve performance of network

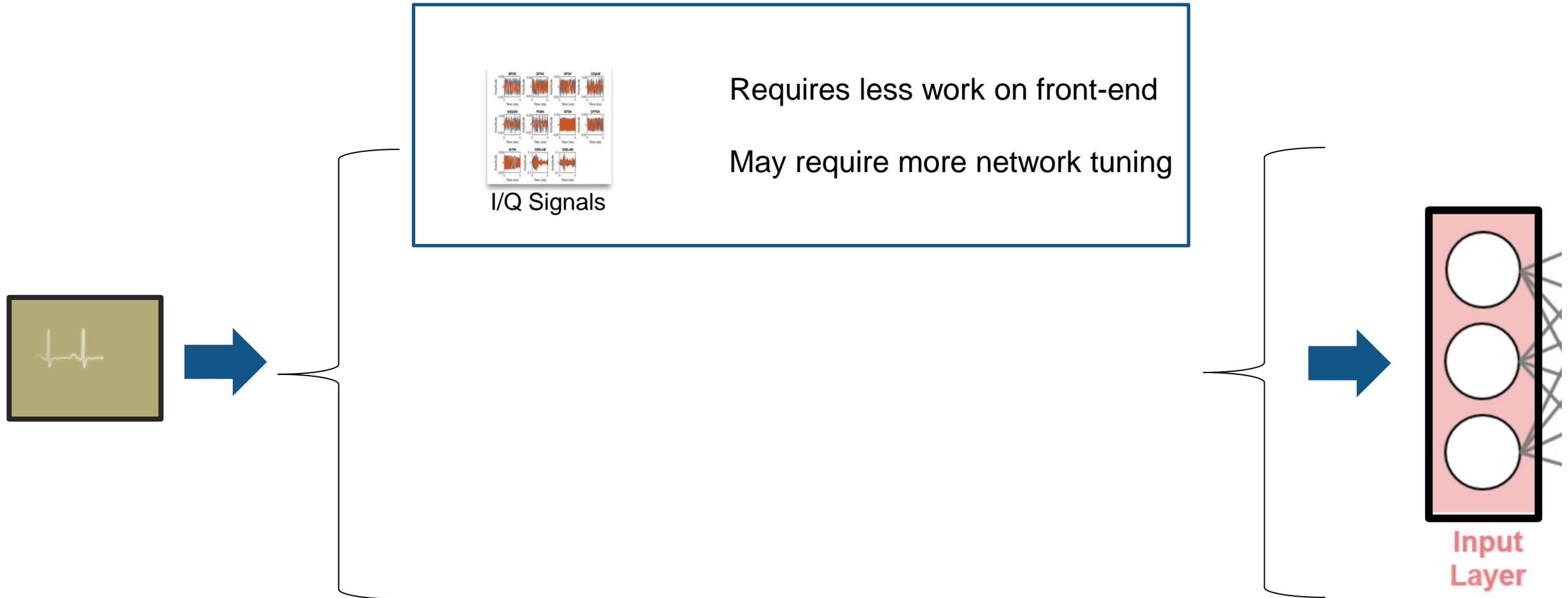


# Pre-processing radar data can improve performance of network

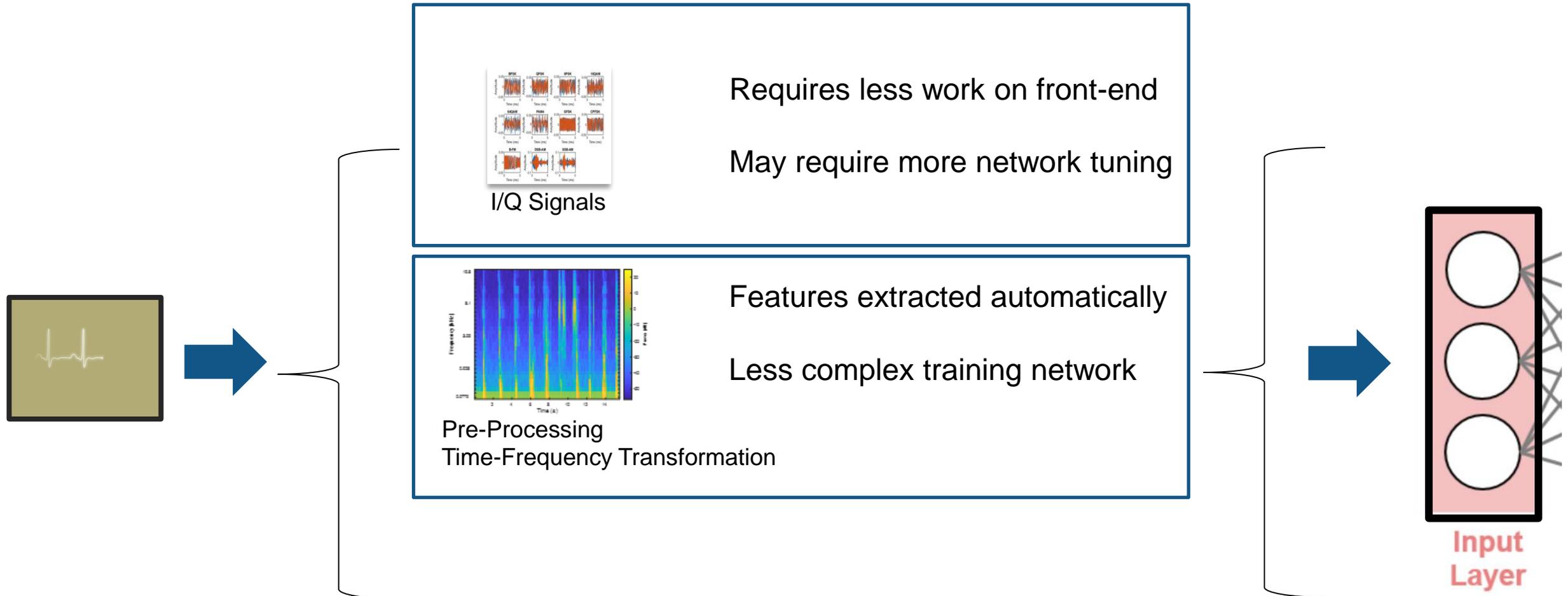


Dataset size vs. domain knowledge vs. compute resources

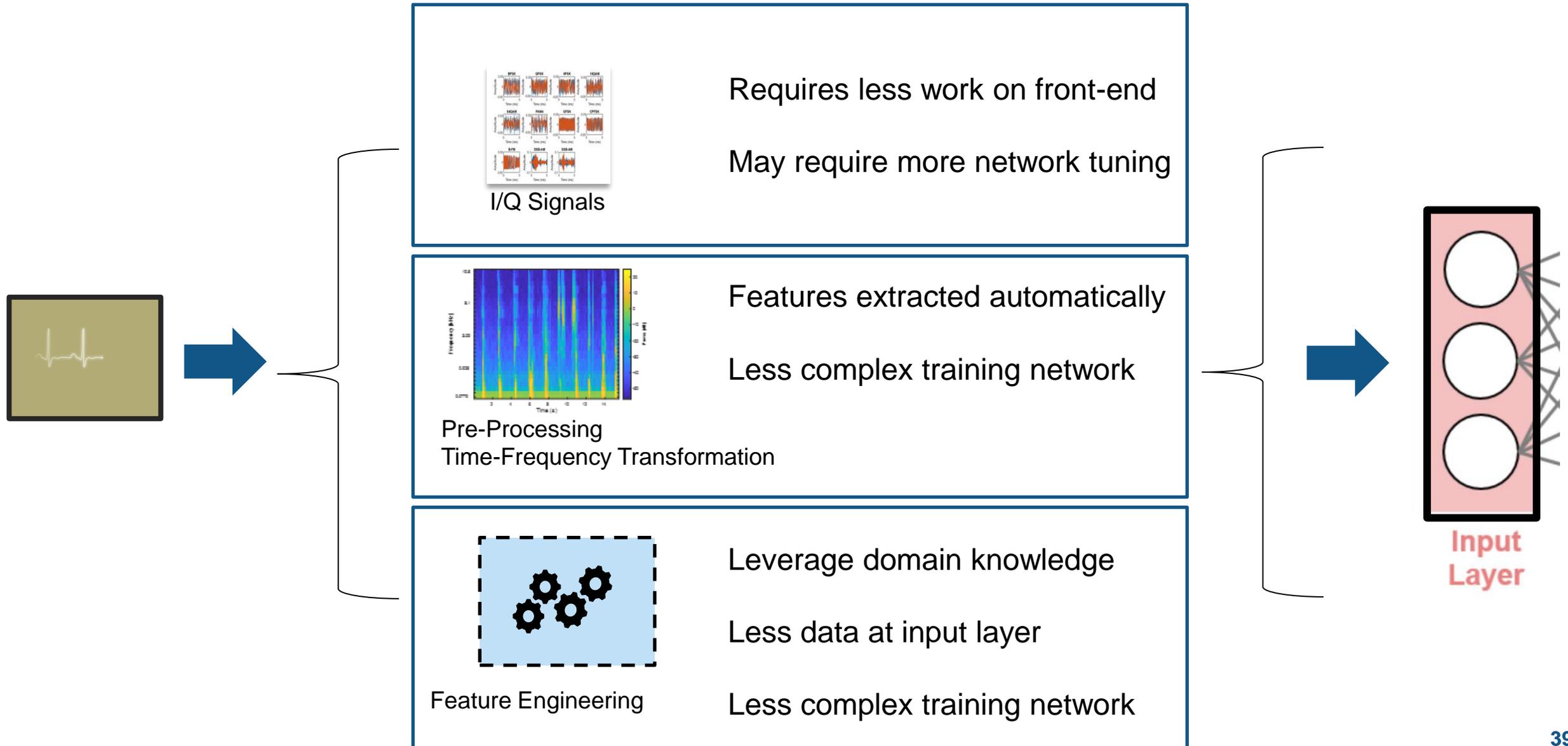
# You can make the trade-off between pre-processing approaches



# You can make the trade-off between pre-processing approaches

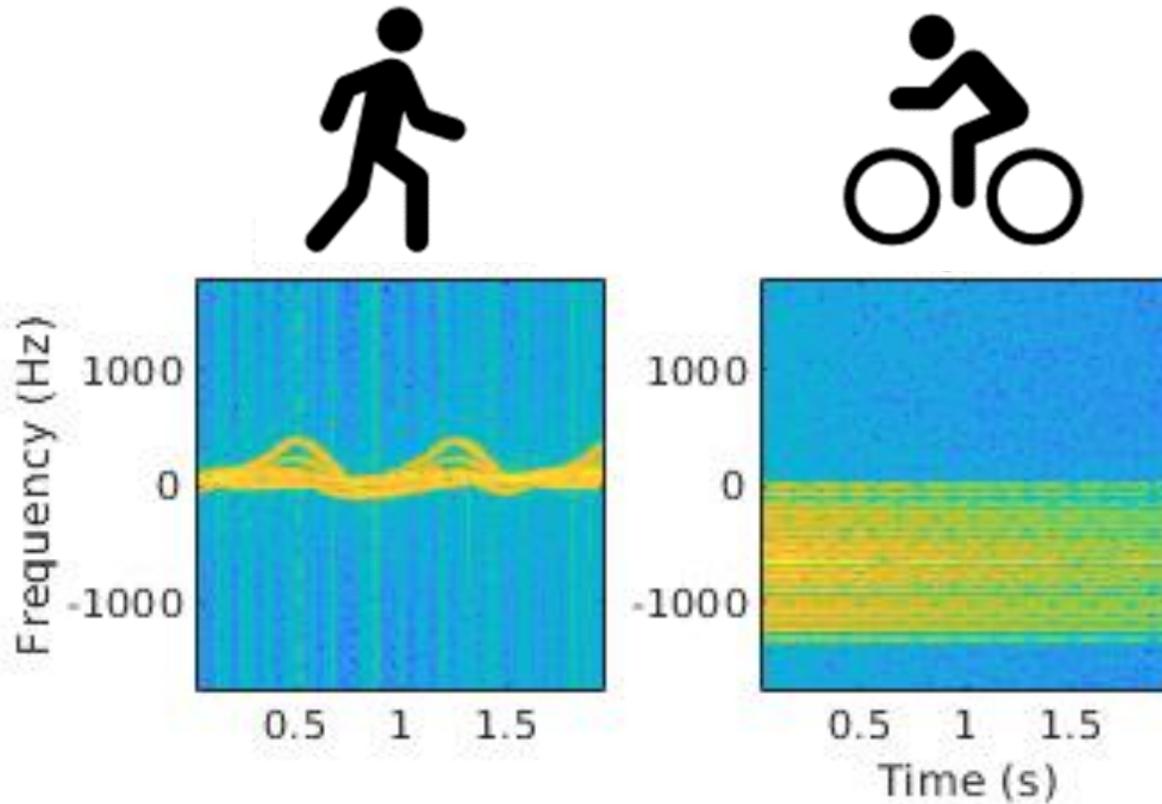


# You can make the trade-off between pre-processing approaches



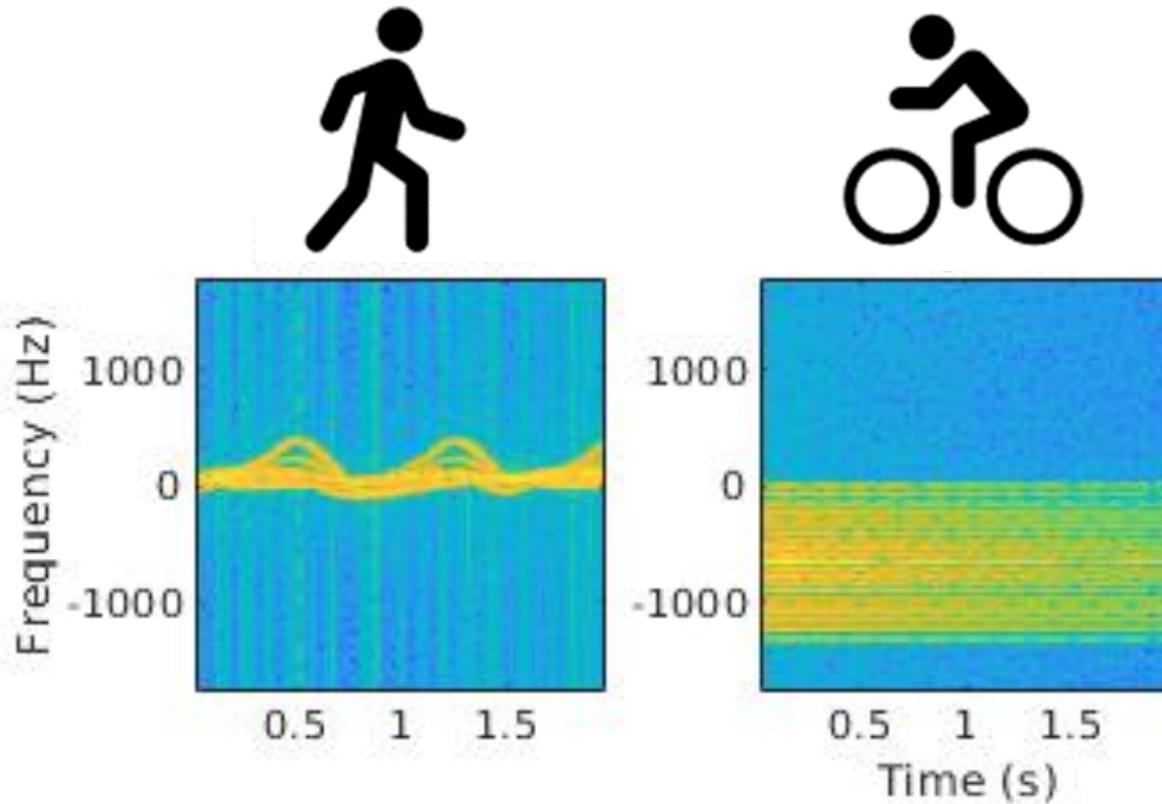
Interactivity: Time to test your ability to classify micro-Doppler returns ...

# Interactivity: Time to test your ability to classify micro-Doppler returns ...

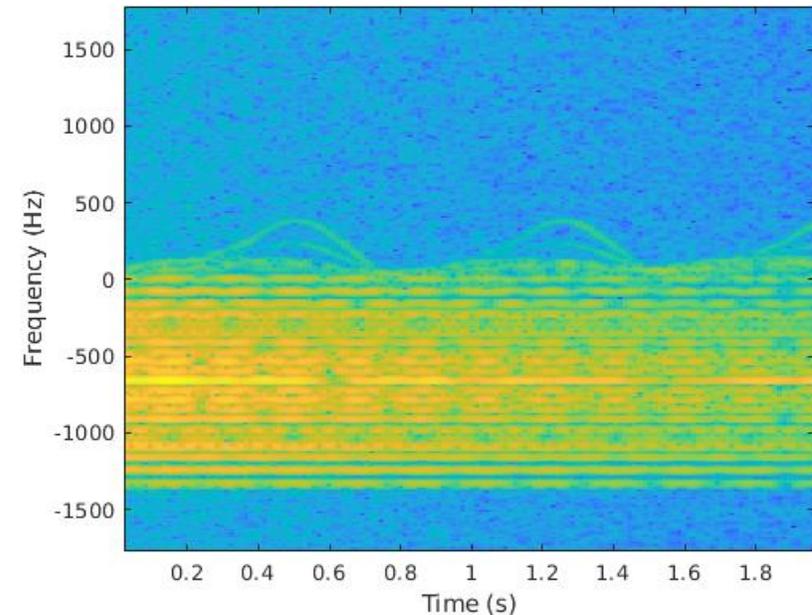


Ground truth – synthesized micro-Doppler

# Interactivity: Time to test your ability to classify micro-Doppler returns ...



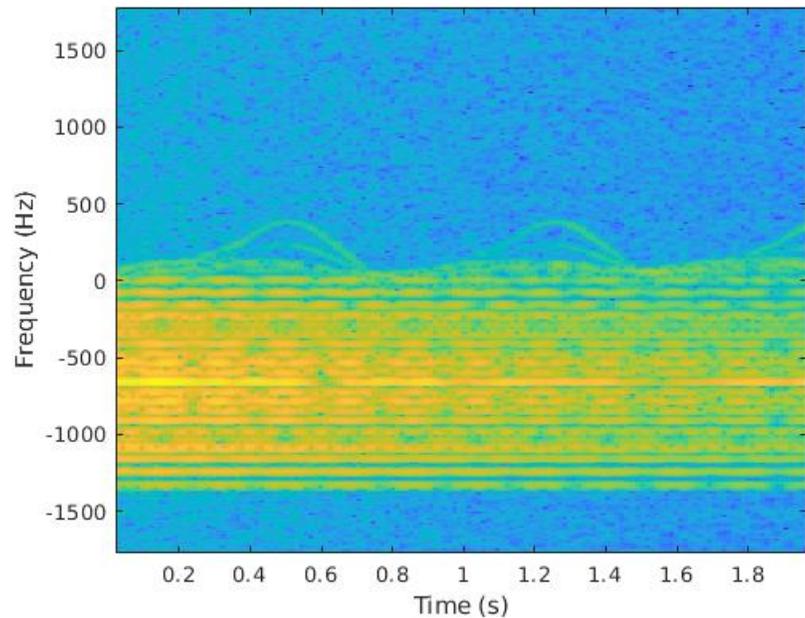
Is this a pedestrian or a bicyclist?



Ground truth – synthesized micro-Doppler

# Poll

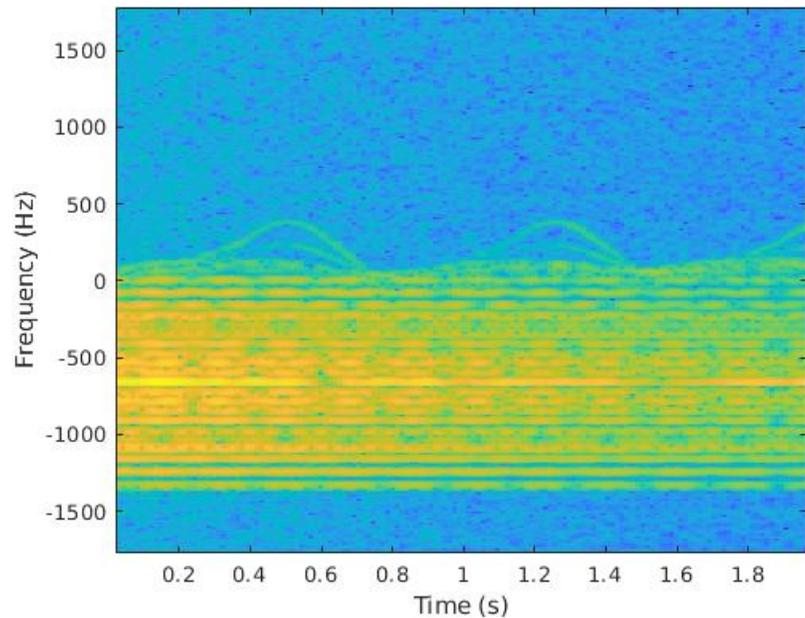
Is this a pedestrian or a bicyclist?



- A. One Pedestrian
- B. One Bicyclist
- C. One of each
- D. Not sure

And the answer is ....

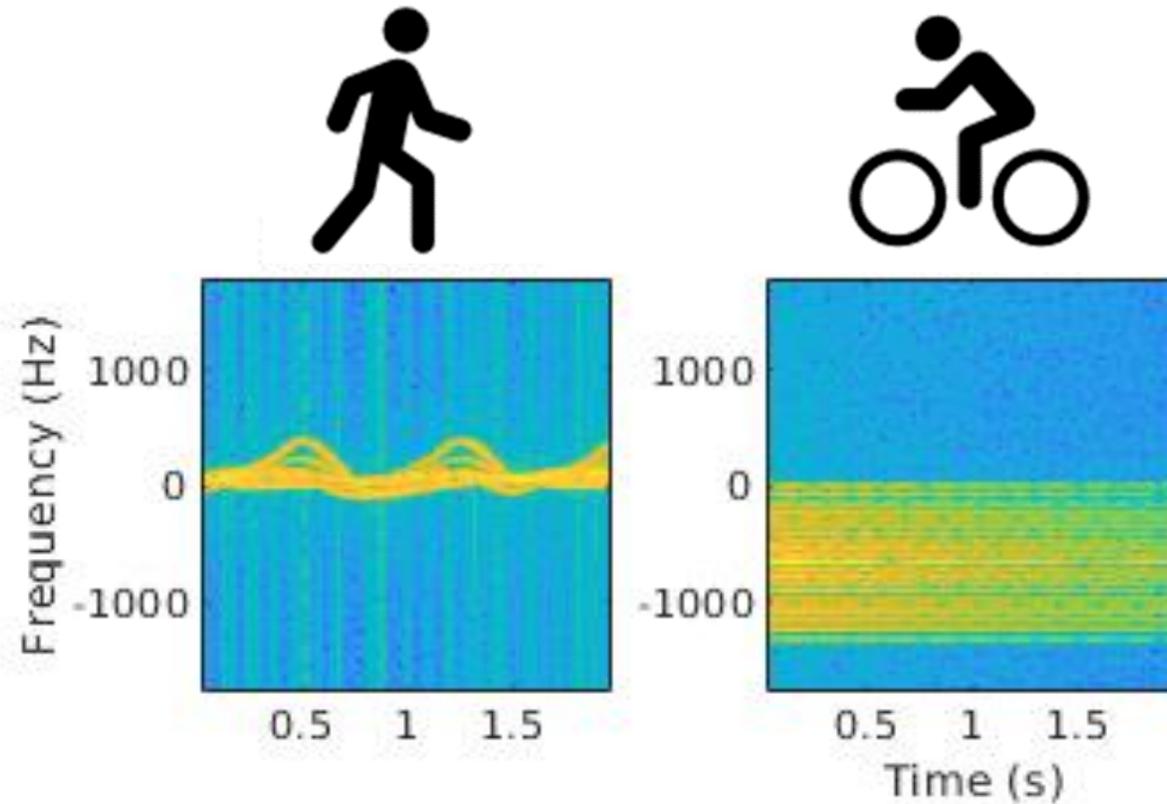
Is this a pedestrian or a bicyclist?



- A. Pedestrian
- B. Bicyclist
- C. One of each
- D. Not sure

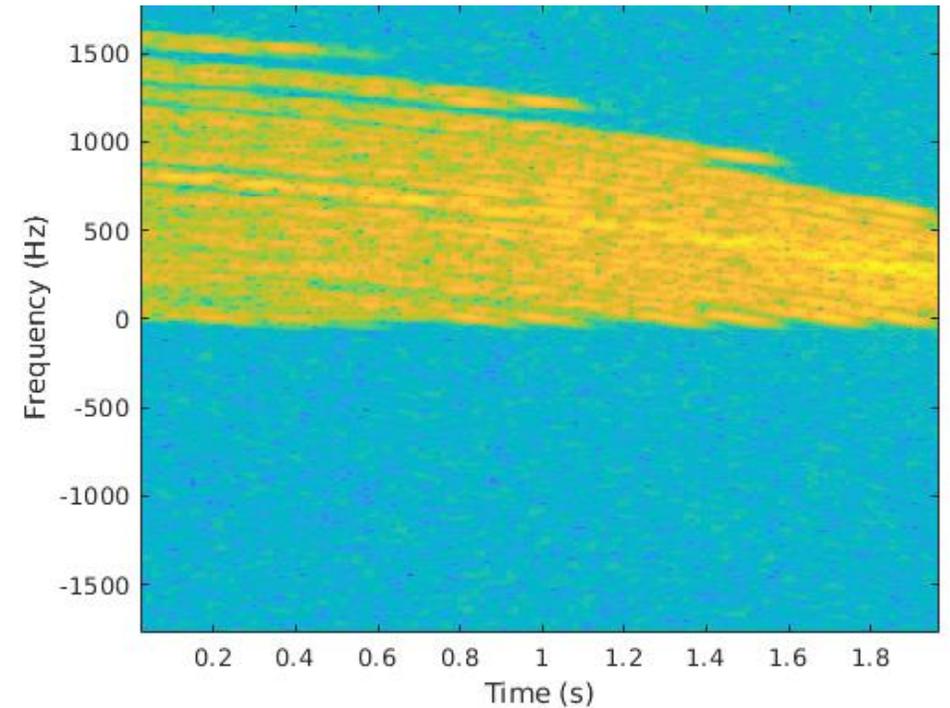
This is a pedestrian and a bicyclist

This one is a bit trickier. The network gets the correct answer

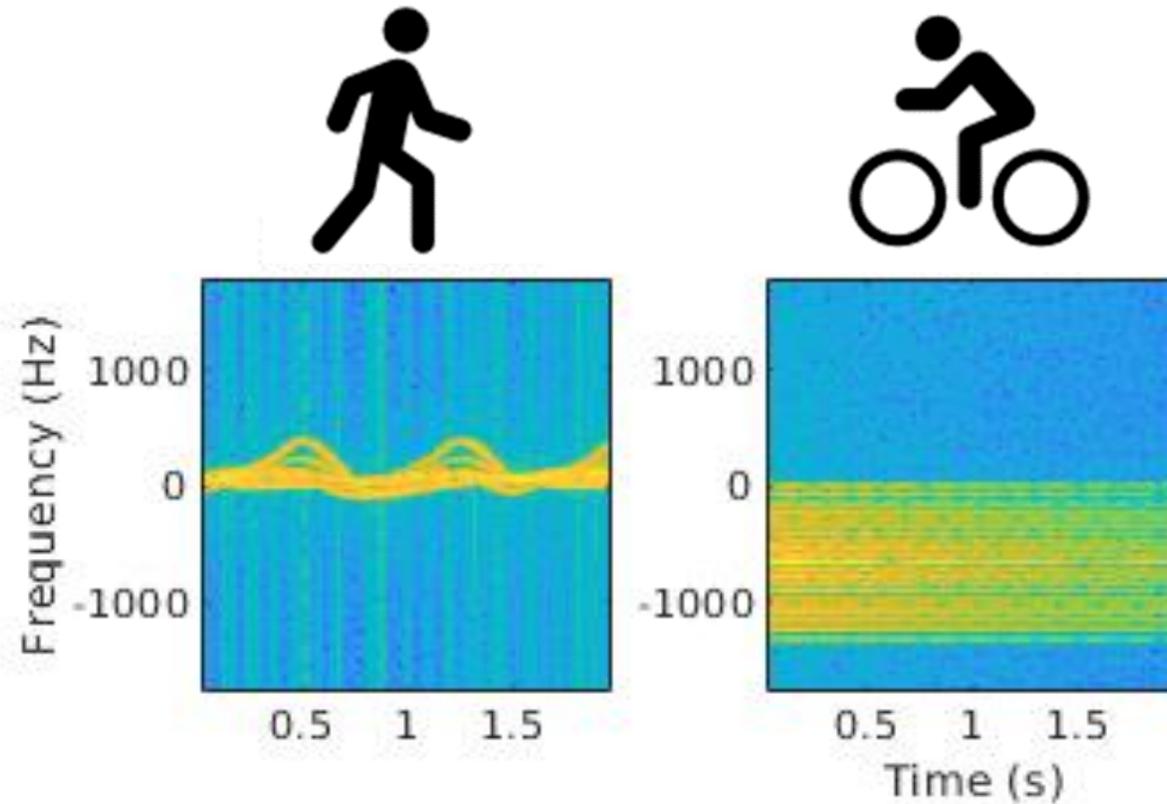


Ground truth – synthesized micro-Doppler

Is this a pedestrian or a bicyclist?

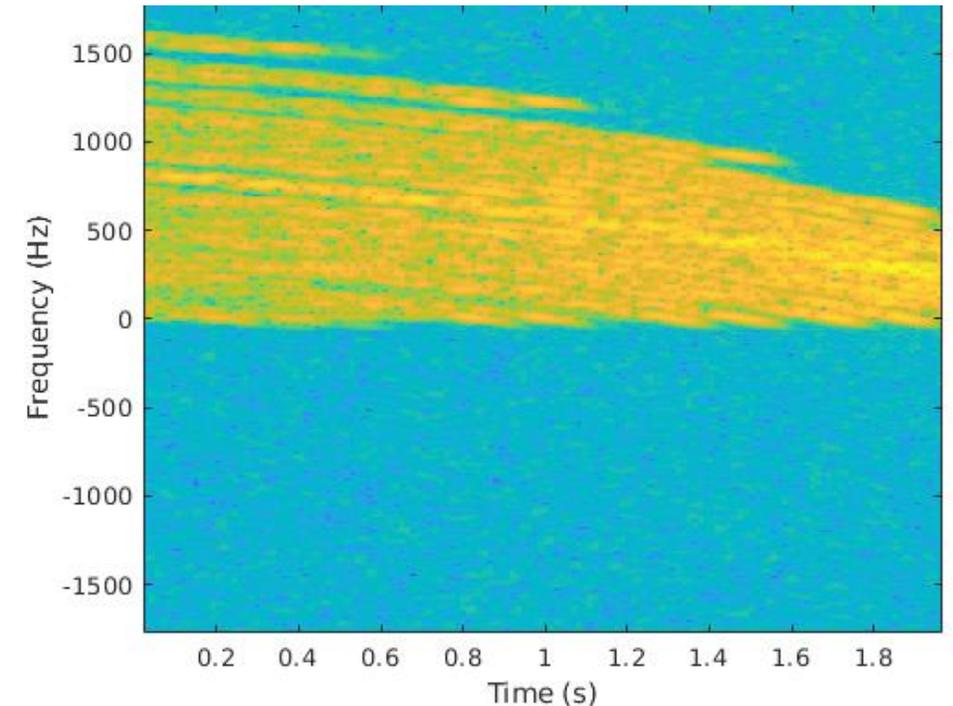


This one is a bit trickier. The network gets the correct answer



Ground truth – synthesized micro-Doppler

Is this a pedestrian or a bicyclist?



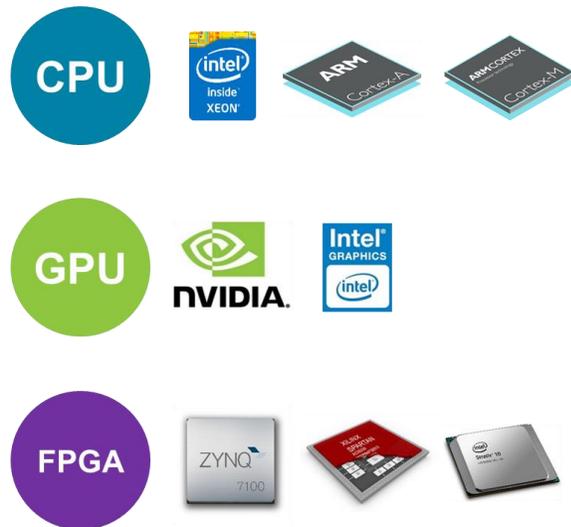
This is two bicyclists

# Challenge

Deploying AI model and application code prototype to a larger system

# Challenge

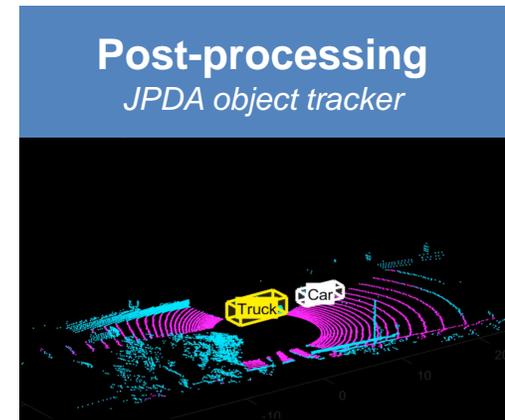
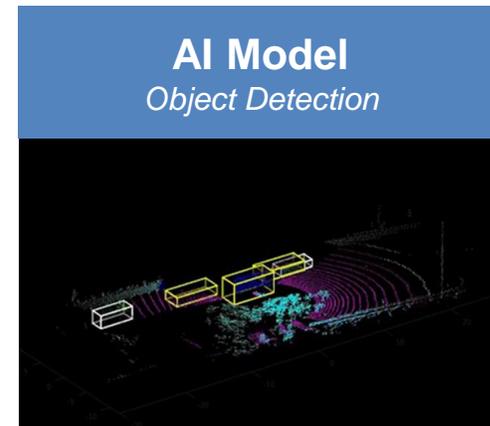
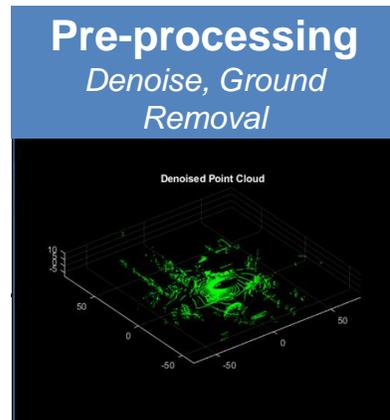
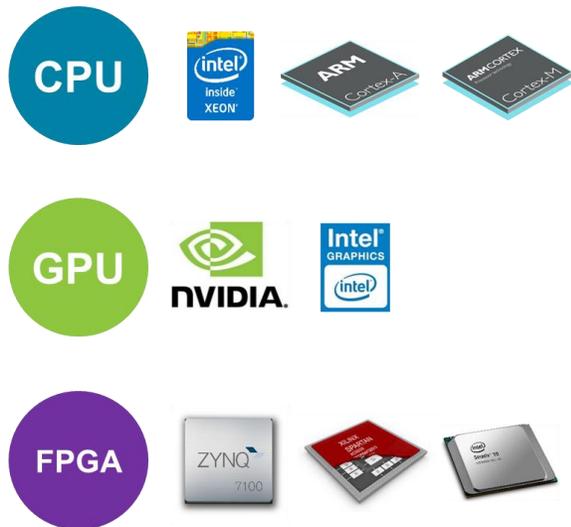
Deploying AI model and application code prototype to a larger system



Multiple options for deployment  
platform  
CPU/GPU/FPGA

# Challenge

Deploying AI model and application code prototype to a larger system

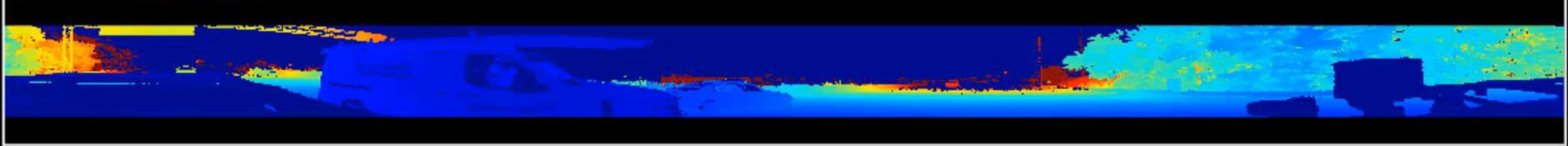


Multiple options for deployment platform  
CPU/GPU/FPGA

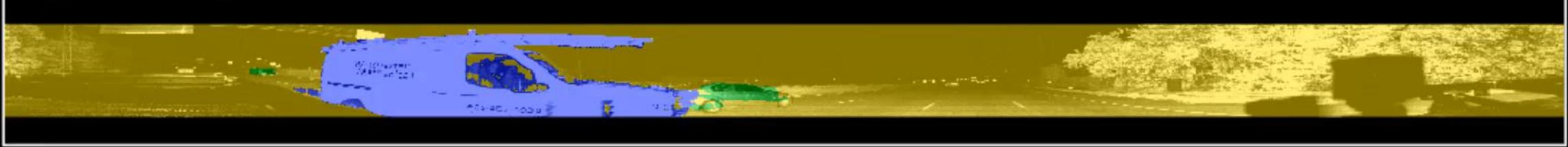
System requires AI model + pre and post processing



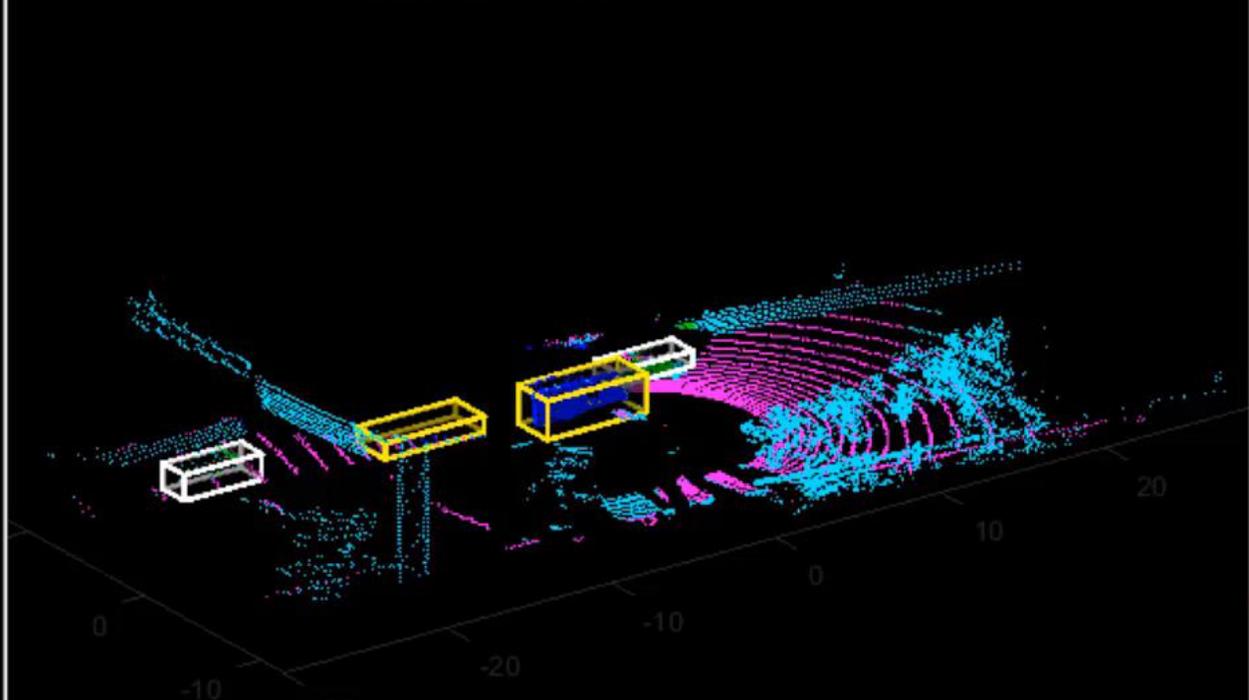
Lidar Range Image



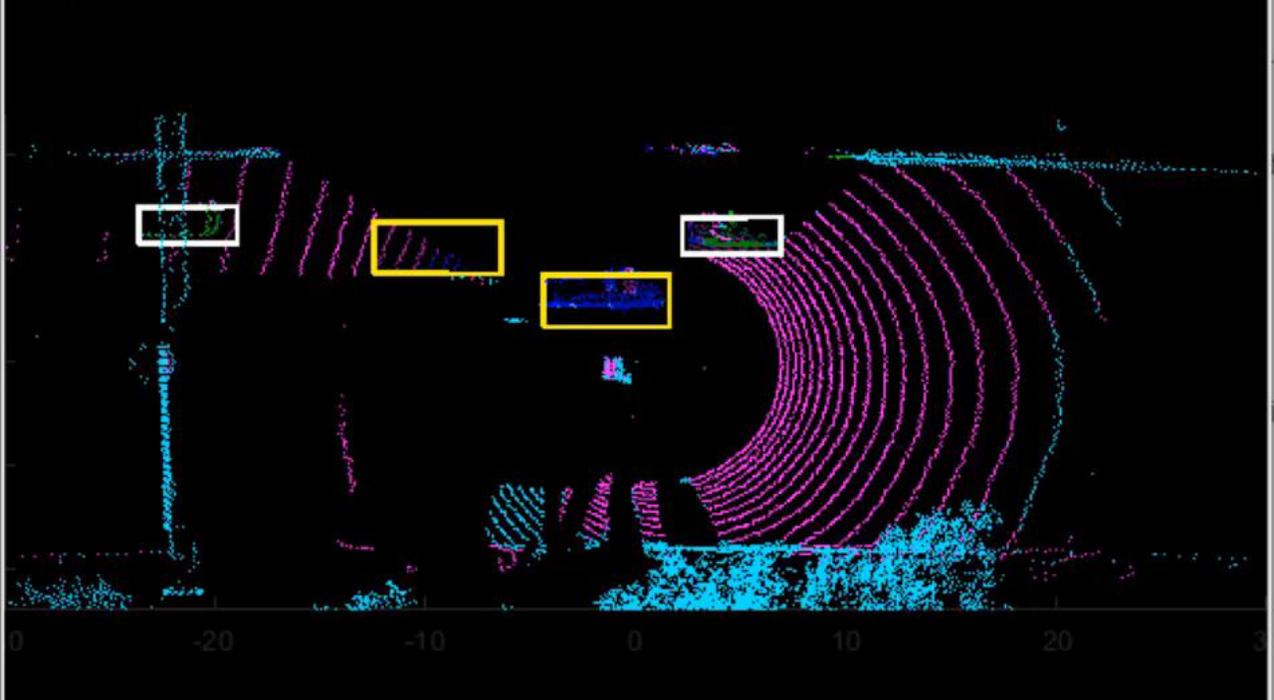
Segmented Image

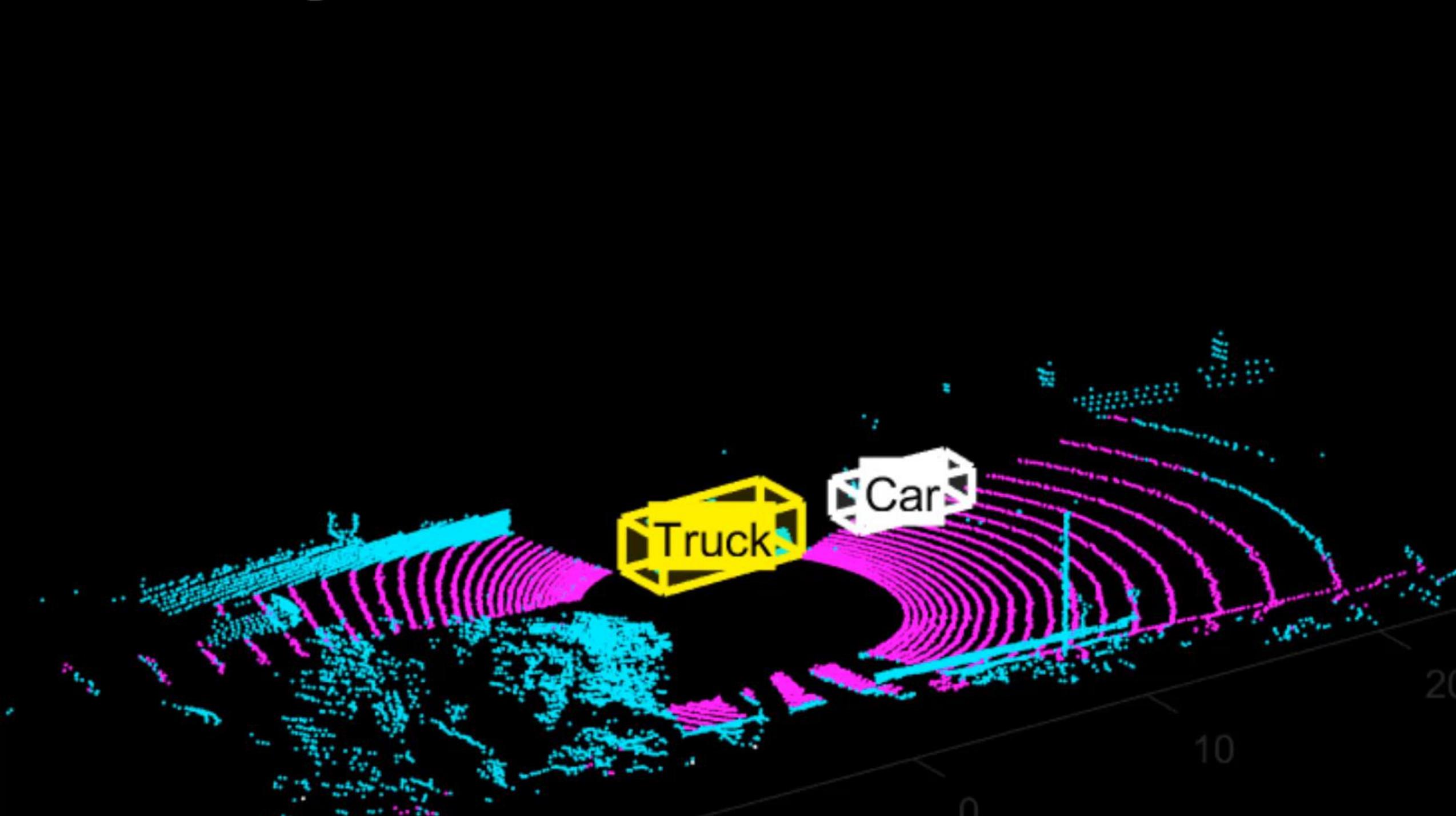


Oriented Bounding Box Detection



Top View





HOME PLOTS APPS LIVE EDITOR INSERT VIEW

Design App Get More Apps Install App Package App

Curve Fitting PID Tuner Signal Analyzer Image Acquisition MATLAB Coder Distribution Fitter Control System Desi... Control System Tuner Flight Log Analyzer Linear System Analyzer Model Reducer

FILE APPS

C:\Users\mpalakka\OneDrive - MathWorks\Documents\MATLAB\Examples\R2020b\shared\_driving\_fusion\_lidar\TrackVehiclesUsingLidarExample

Live Editor - C:\Users\mpalakka\OneDrive - MathWorks\Documents\Demos\DetectClassifyAndTrackOrientedBoundingBoxInLidarExample\DetectClassifyAndTrackOrientedBoundingBoxInLidarExample.mlx

DetectClassifyAndTrackOrientedBoundingBoxInLidarExample.mlx TrackVehiclesUsingLidarExample.m

```
150 filterInitFcn = @helperMultiClassInitIMMFilter;
151
152 % A joint probabilistic data association tracker with IMM filter
153 tracker = trackerJPDA('FilterInitializationFcn',filterInitFcn,...
154     'TrackLogic','History',...
155     'AssignmentThreshold',assignmentGate,...
156     'ClutterDensity',Kc,...
157     'ConfirmationThreshold',confThreshold,...
158     'DeletionThreshold',delThreshold,'InitializationThreshold',0);
159
160 allTracks = struct([]);
161 time = 0;
162 dt = 0.1;
163
164 % Define Measurement Noise
165 measNoise = blkdiag(0.25*eye(3),25,eye(3));
166
167 numTracks = zeros(numFrames, 2);
```

The detected objects are assembled as a cell array of [objectDetection](#) objects using the `helperAssembleDetections` function.

```
168 display = helperLidarObjectDetectionDisplay;
169 initializeDisplay(display);
170
171 for count = 1:numFrames
172     time = time + dt;
173     % Get current data
```

# Reduce memory and power needs of deployed models

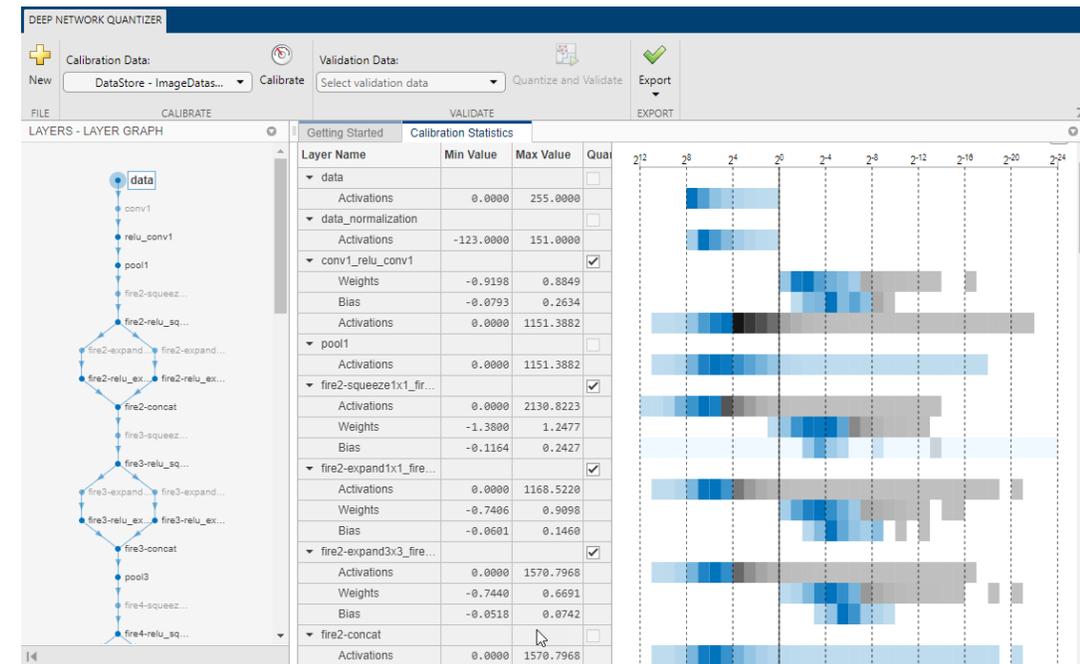
## Quantize & Compress networks to deploy to low-power microcontrollers and FPGA's

Choose and validate the right quantization approach to meet the required accuracy.

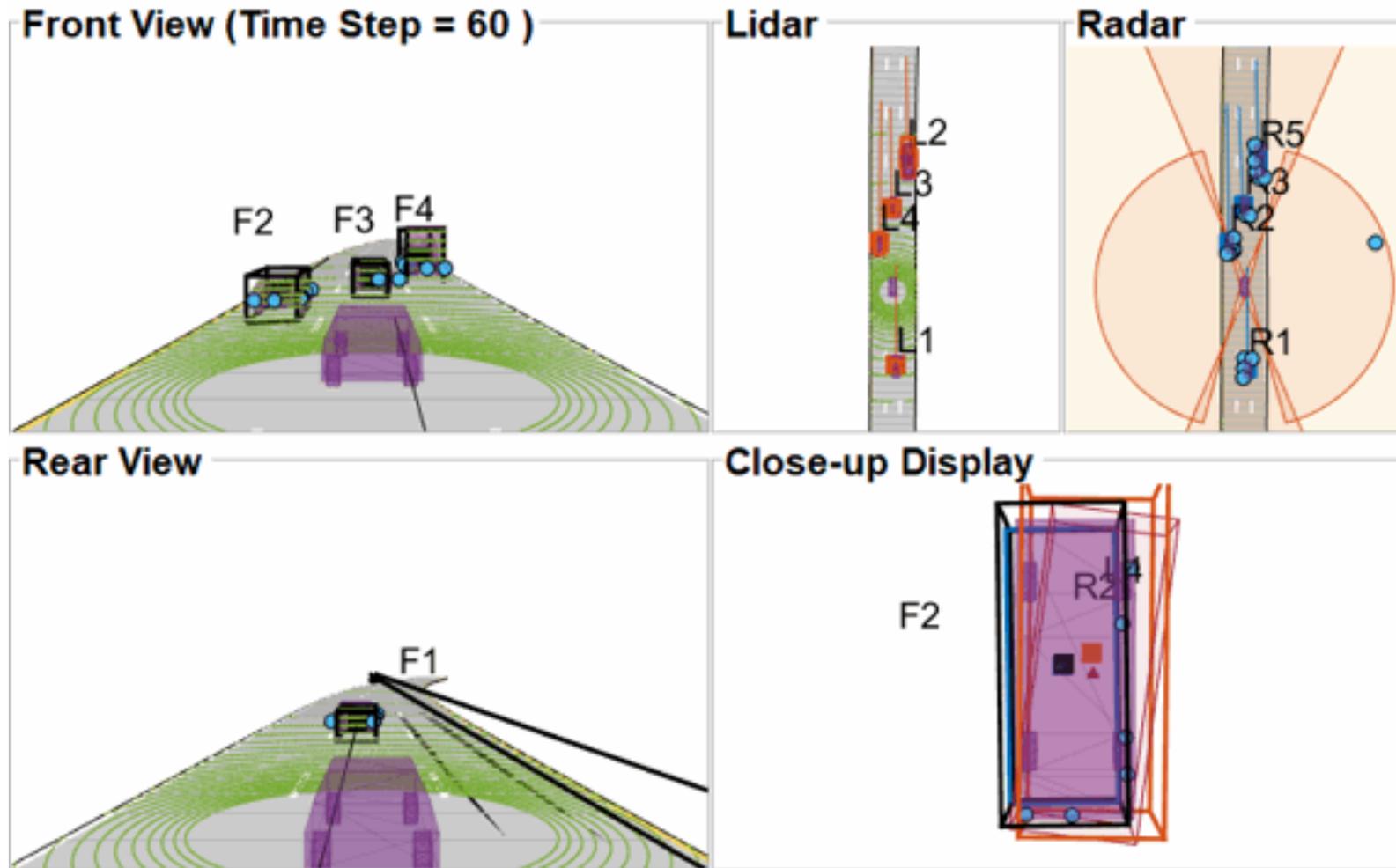
### Deep Network Quantizer App

R2020a

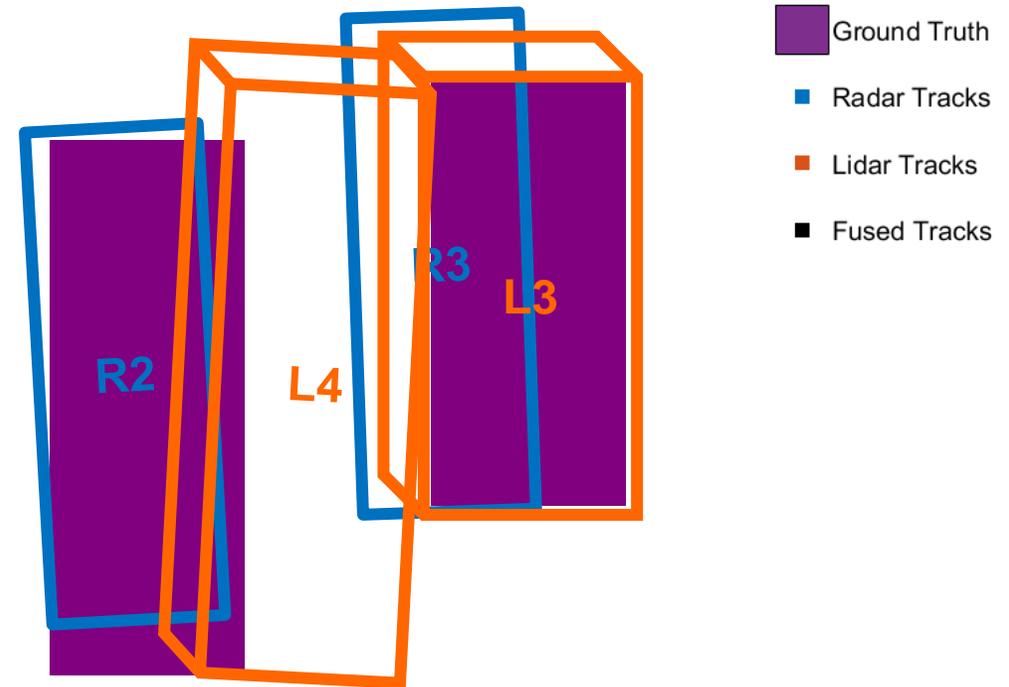
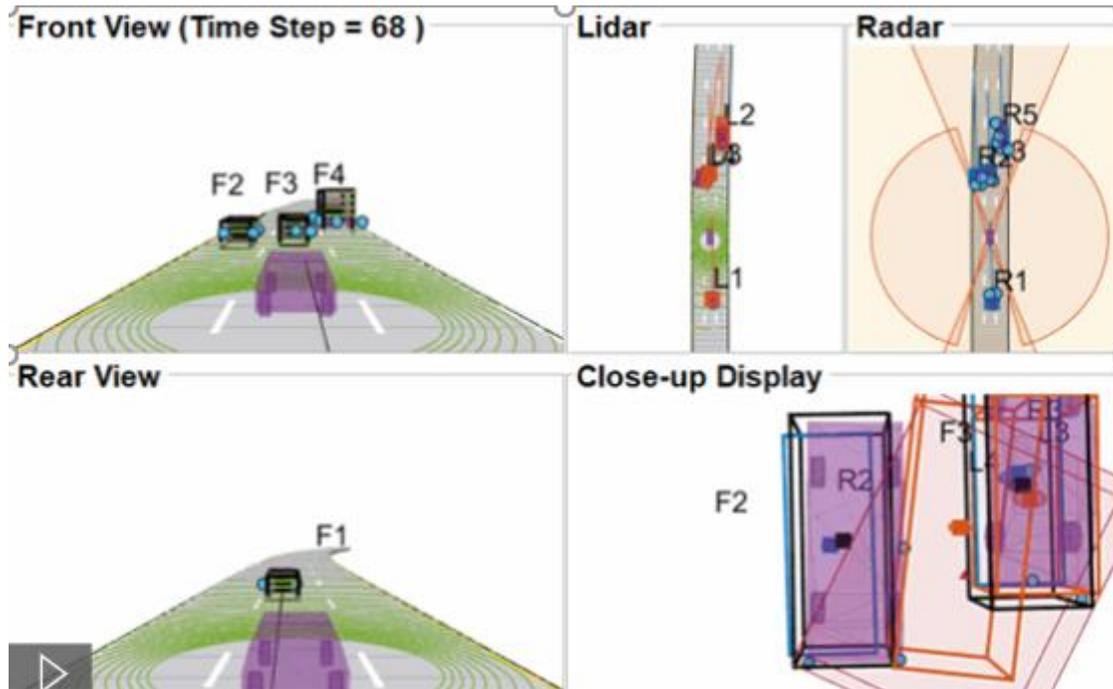
- Visualize the dynamic ranges of convolution layers.
- Select individual network layers to quantize.
- Assess the performance.
- Generate GPU code to deploy



We can improve our results when we fuse the two sensors



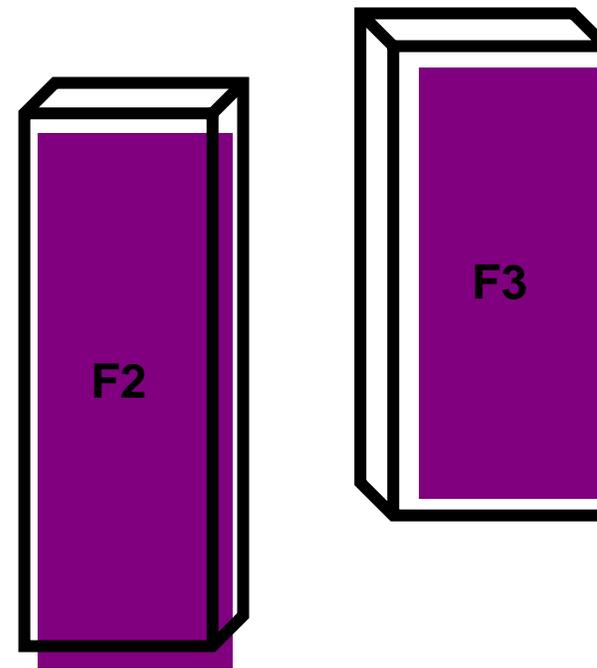
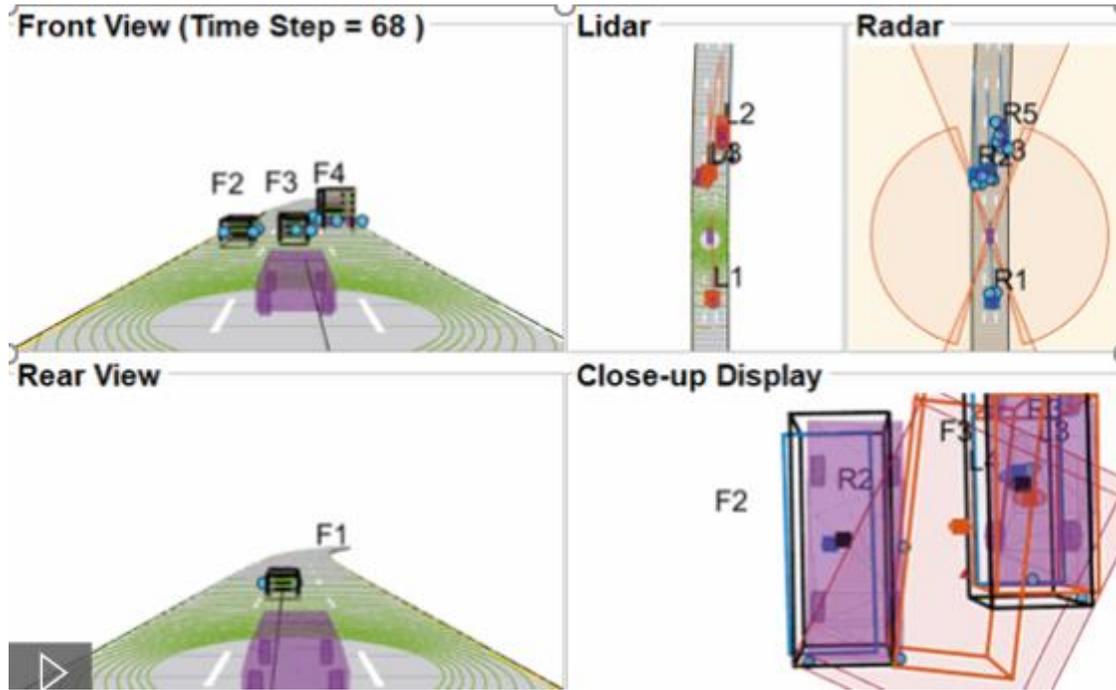
# Let's take a closer look ...



- Ground Truth
- Radar Tracks
- Lidar Tracks
- Fused Tracks

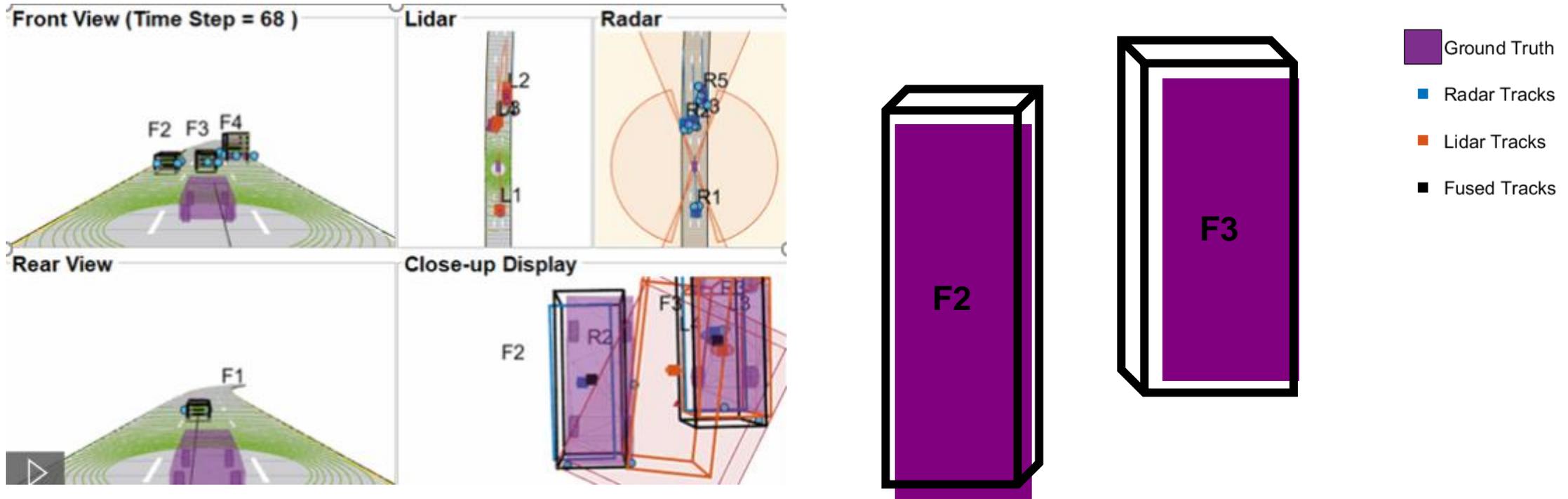
Fused tracks more accurate than individual sensor tracks

# Let's take a closer look ...



- Ground Truth
- Radar Tracks
- Lidar Tracks
- Fused Tracks

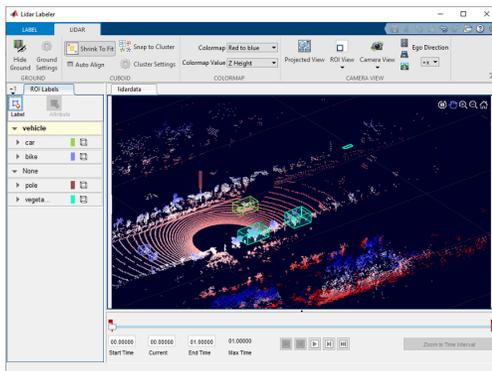
# Let's take a closer look ...



Fused tracks more accurate than individual sensor tracks

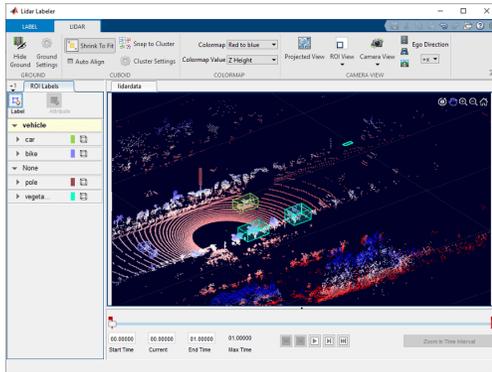
# How MATLAB and Simulink help create AI-driven radar and lidar processing systems

# How MATLAB and Simulink help create AI-driven radar and lidar processing systems

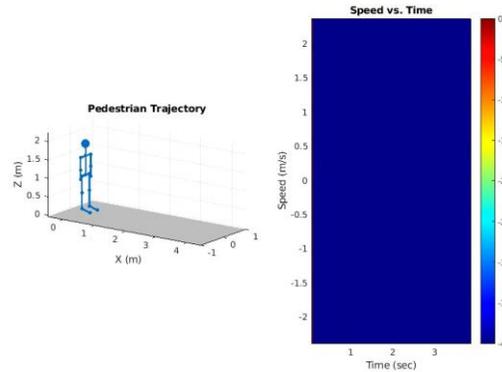


**Labeling Automation**

# How MATLAB and Simulink help create AI-driven radar and lidar processing systems

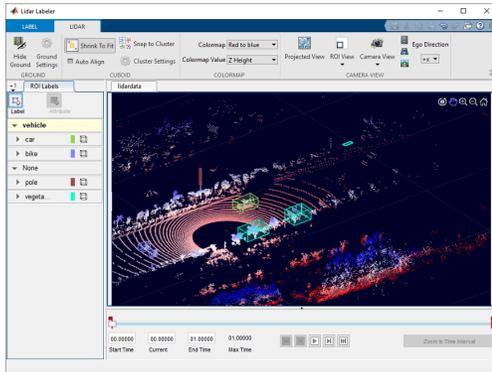


Labeling Automation

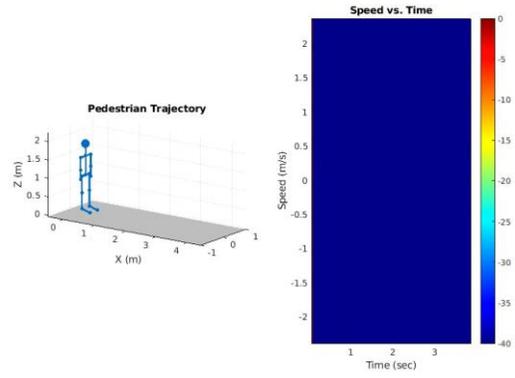


Data Synthesis

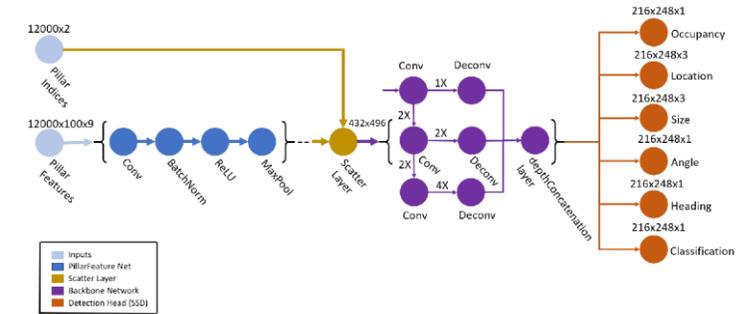
# How MATLAB and Simulink help create AI-driven radar and lidar processing systems



**Labeling Automation**

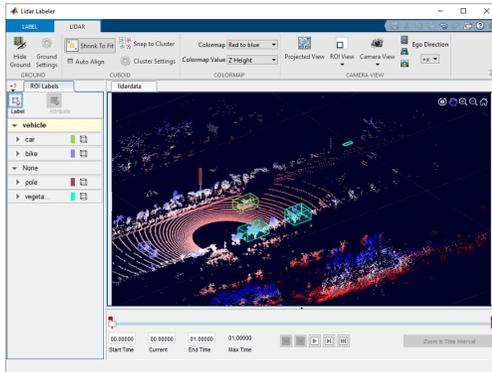


**Data Synthesis**

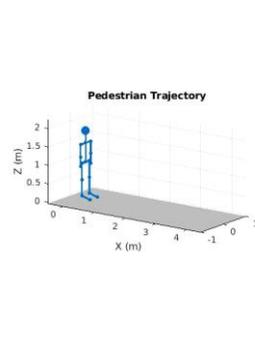


**AI Workflow**  
Pre-trained models, training, evaluation, validation

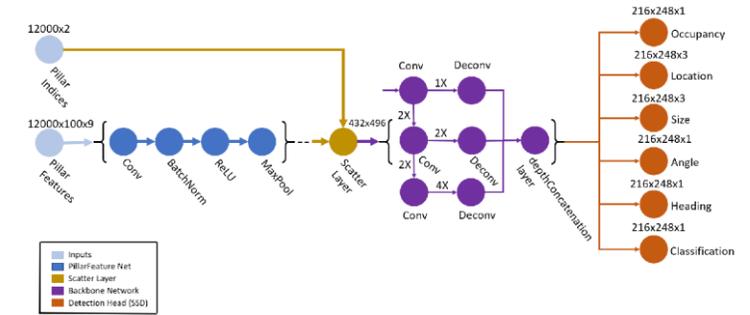
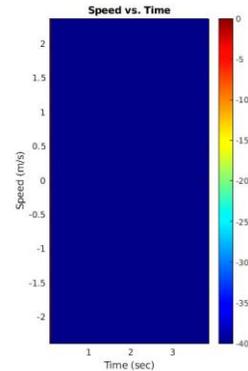
# How MATLAB and Simulink help create AI-driven radar and lidar processing systems



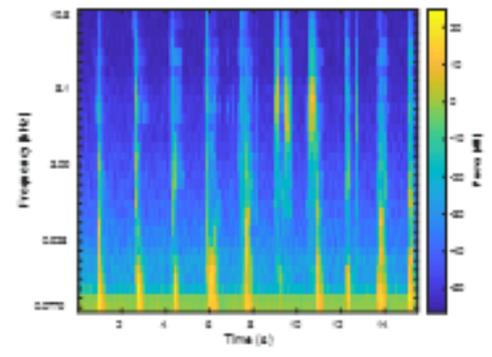
**Labeling Automation**



**Data Synthesis**

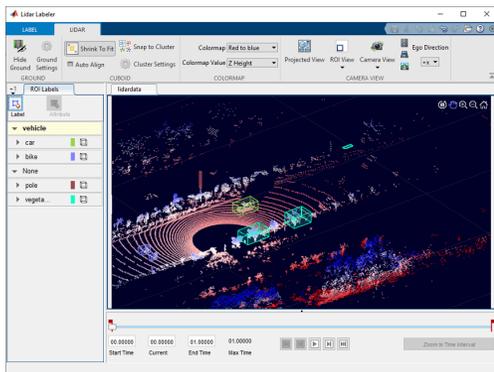


**AI Workflow**  
Pre-trained models, training, evaluation, validation

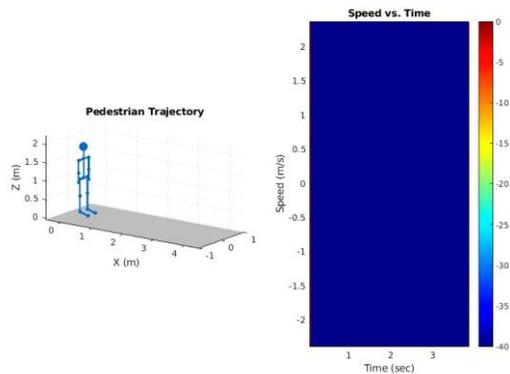


**Pre-processing**

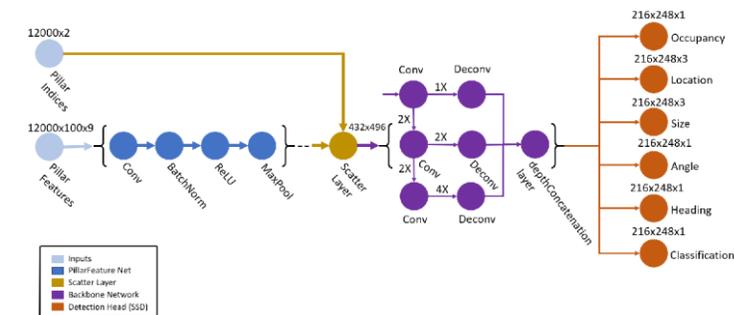
# How MATLAB and Simulink help create AI-driven radar and lidar processing systems



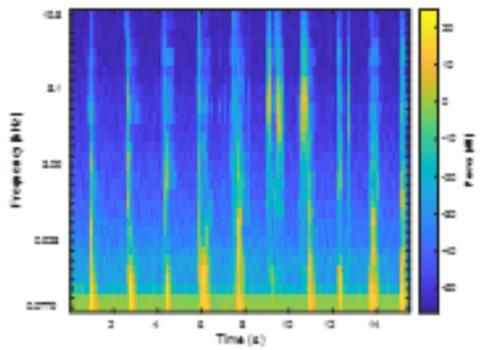
**Labeling Automation**



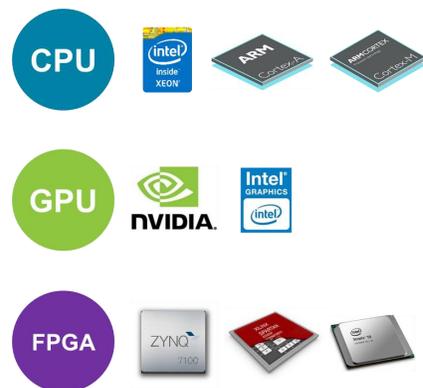
**Data Synthesis**



**AI Workflow**  
Pre-trained models, training, evaluation, validation



**Pre-processing**



**Full Application Deployment**

# MATLAB EXPO

## 2021

Thank you

