BOSCH – MATLAB EXPO 2022

DESIGNING LIDAR SENSOR CLASSIFIER USING MATLAB FRAMEWORK

Ramakrishnan R
Senior Software expert Vehicle computer
GPU & Safety

Kadengodlu Uthama
Senior Software engineer
Embedded Engineer

Rahul Vikram Porwal
Software engineer
Function Developer
AI & LIDAR Developer

BOSCH

Agenda

1. Who are we
2. Project focus
3. LIDAR @ Deep learning Framework
4. Deep Learning (DL) Approach
5. Lidar Segmentation and MATLAB realization
6. Target realization
7. Summary

Who we are

Our business sectors

Mobility Solutions

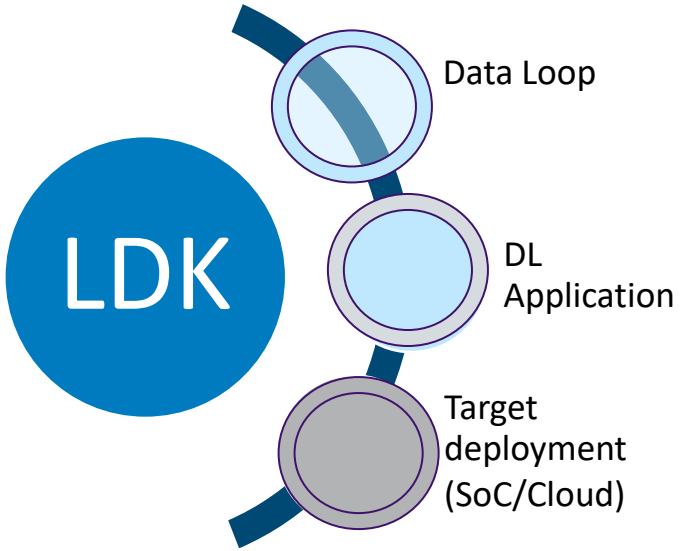
Industrial Technology

Energy & Building Technology

Consumer Goods

Project focus

LiDAR Development Kit (LDK)



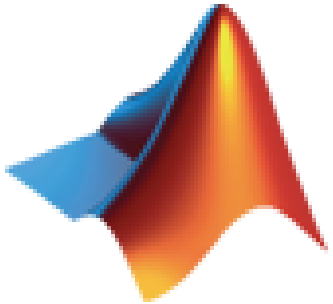
LDK - Connecting industry to Engineering

LiDAR @ Deep learning Framework

- Major LiDAR DL Frameworks
  - Tensorflow
  - Keras
  - PyTorch

Caffe

- Deep learning Challenges
  - Framework independent platform
  - Interoperability between different HW system for development
    - Target code generation



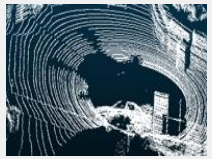
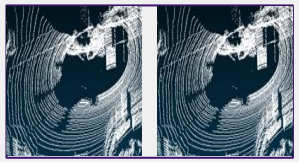
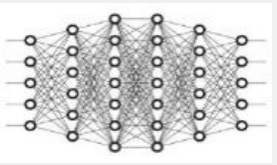
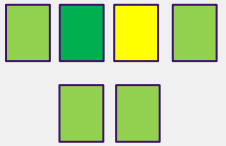
BOSCH

Deep Learning Approach

Training and Inferencing flow

Training

Raw Data → Data Preprocessing → Training model → Model validation → Model finalization

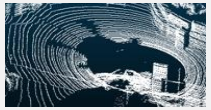

Raw data → Data Preprocessing → DL input → Prediction → Semantic Segmentation

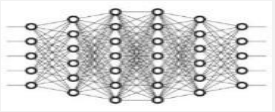
Inferencing on Embedded Target

BOSCH

LIDAR Semantic Segmentation

DL Training

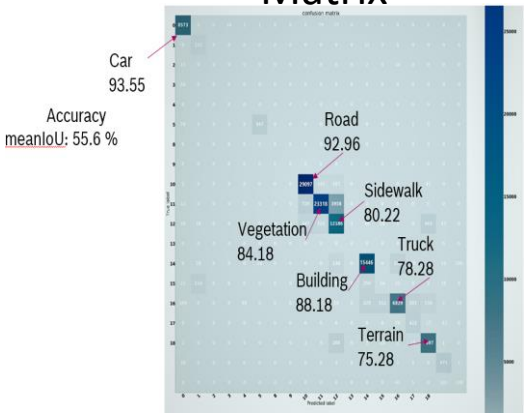


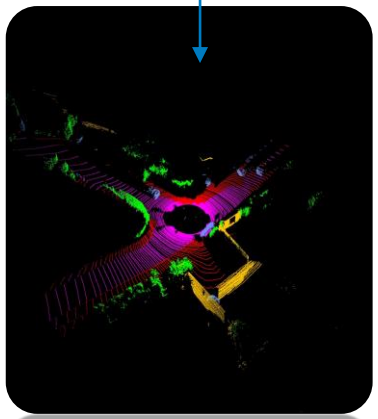


Raw Data → Data Preprocessing → Training model → Model validation → Model finalization

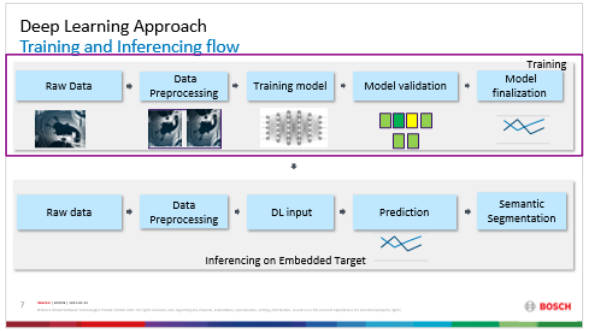
Automated Controlled Environment Vehicle
- Velodyne VLP16

Confusion Matrix





Framework independent platform



TensorFlow

TF model

Non-Open format Model

Layer I/P & O/P

Data

ONNX

Open format Model

ONNX Conversion

(Open Neural Network Exchange)

Target Code generation

C,*.C++, *.exe

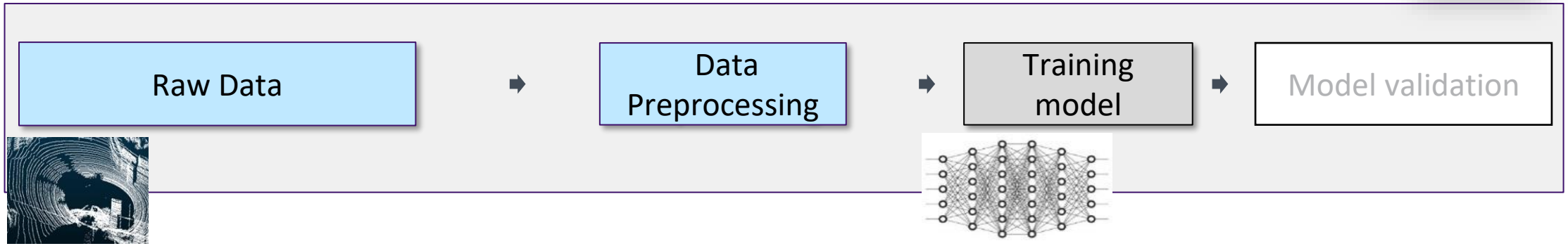
- Hardware Layers Compatibility
- Model performance

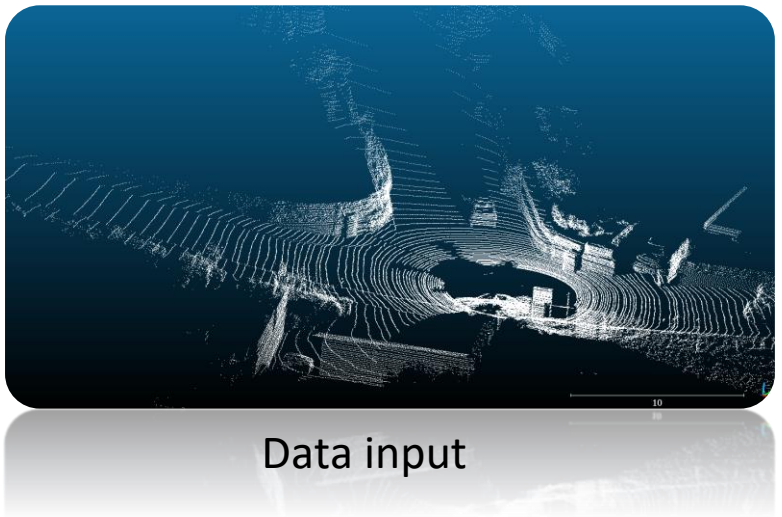
- Open standard model
- Realize missing/incompatible layers
- Retraining

- Embedded Code generation
- Executables

BOSCH

LIDAR Semantic Segmentation
MATLAB realization

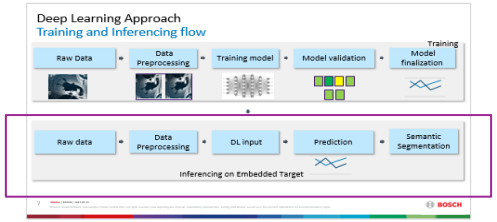




Data input

- Lidar Data Selection
- Importing data set
- Subsampling of points
- Parsing of point clouds

Framework independent platform



TensorFlow

TF model

Non-Open format Model

Layer I/P & O/P

Data

ONNX

Open format Model

ONNX Conversion

(Open Neural Network Exchange)

Target Code generation

C,*.C++, *.exe

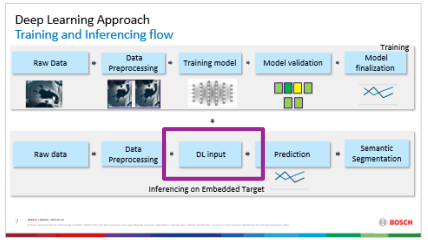
- Hardware Layers Compatibility
- Model performance

- Open standard model
- Realize missing/incompatible layers
- Retraining

- Embedded Code generation
- Executables

BOSCH

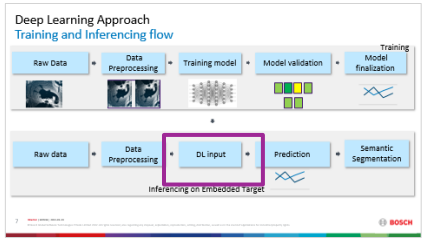
Target Realization @ X86




```
>> test
Input File
 semantic01.bin
Code generation successful.

File read start
File read end
Conversion to organised point cloud started
Conversion to organised point cloud done
Segmentation done
Inside function to save point cloud
%f-1.23228e+08%f-3.93344e+07%f-5.45696%f-1.68519e+07%f-1737.9%f-0.000172499%f-54.3848%f-0.0651428%f-41.9732%f-830.29%f-3.05776e+07%f-2.05391e+07%f-7093.19%fnan%fnan%f-1.03164e-05%fnan%fnan%fnan%fnan%fnan%f-2.19093e-05%f-1.39603e-05%
%f8.96454e-07%f0.000292969%f1.59075e-314%f0.000117188%f1.59076e-314%f3.05176e-07%f1.58632e-314%f0.000131836%f3.56522e-12%f1.59066e-314%f7.43867e-07%f1.75476e-06%f1.5906e-314%fnan%fnan%f8.10623e-08%fnan%fnan%fnan%fnan%fnan%f2.3365e-0
%f-4.35707e+07%f-1.43953e+07%f-2.03209%f-6.42606e+06%f-678.939%f-6.84412e-05%f-21.943%f-0.0271034%f-17.7171%f-356.88%f-1.34908e+07%f-9.13713e+06%f-3243.67%fnan%fnan%f-4.10627e-06%fnan%fnan%fnan%fnan%fnan%f-9.82409e-06%f-6.41512e-06%
```

Interoperability : HW Target system



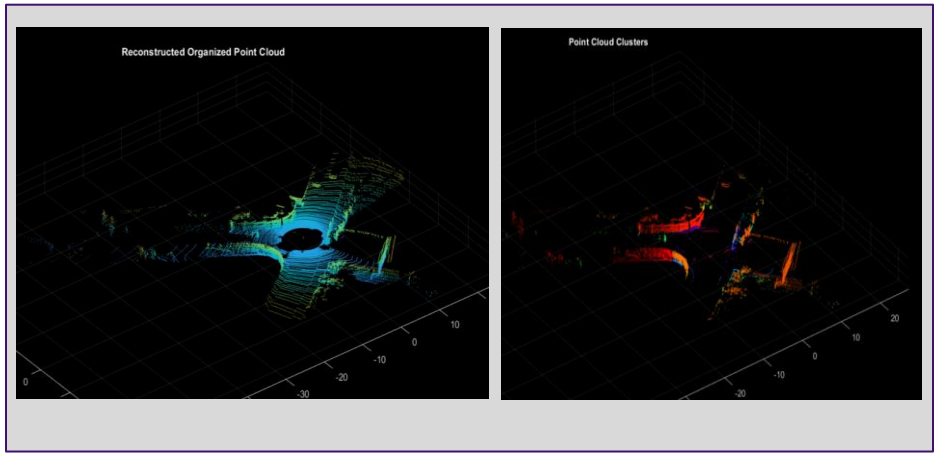
DL model

Matlab Coder → Validating performance → Code generation → Target interfacing → Enabling Data pipeline

Validating performance

Target 1

Target 2 config

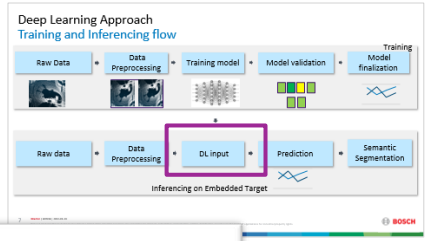


Code generation → Target interfacing → Enabling Data pipeline

Validating performance

Target 2…

Code generation



DL

Matla

et 2

Output Files

- convertFromCartesianToSphericalCoordinate.cpp
- convertUnorgToOrg.cpp
- fileManager.cpp
- fread.cpp
- GroundSegmentFuncBin_data.cpp
- GroundSegmentFuncBin_initialize.cpp
- GroundSegmentFuncBin_rtwutil.cpp
- GroundSegmentFuncBin_terminate.cpp
- GroundSegmentFuncBin.cpp
- histcounts.cpp
- labelRangeDataGroundRemovalUtilsCore.cpp
- main.cpp
- matlabCodegenHandle.cpp
- pcdenoise.cpp
- pointCloud.cpp
- pointCloudImpl.cpp
- rt_nonfinite.cpp
- rtGetInf.cpp
- rtGetNaN.cpp
- strcmp.cpp
- coder_array.h
- convertFromCartesianToSphericalCoordinate.h
- convertUnorgToOrg.h
- fileManager.h
- fread.h
- GroundSegmentFuncBin_
- GroundSegmentFuncBin_initialize.h

```
 8 // Include Files
 9 #include "GroundSegmentFuncBin.h"
10 #include "GroundSegmentFuncBin_data.h"
11 #include "GroundSegmentFuncBin_initialize.h"
12 #include "convertFromCartesianToSphericalCoordinate.h"
13 #include "convertUnorgToOrg.h"
14 #include "fileManager.h"
15 #include "fread.h"
16 #include "labelRangeDataGroundRemovalUtilsCore_api.hpp"
17 #include "matlabCodegenHandle.h"
18 #include "pcdenoise.h"
19 #include "pointCloud.h"
20 #include "pointCloudImpl.h"
21 #include "rt_nonfinite.h"
22 #include "strcmp.h"
23 #include <string.h>
24
25 // Function Definitions
26
27 //
28 // %% For reading bin files
```

Target Build Log | Variables

Variable	Type	Size
Input		
file	char	1 x 24
sensor	char	1 x 6
Output		
groundPtCloud > 1	double	:Inf x 3

2

Generated Code

BOSCH

Summary

- Collaborative Faster layer standardization
  - Fine tuning of models
  - Usage of Model constructors in MATLAB helps faster realization of model

- Collaborative Realization of Multi target workflow
  - Support from MathWorks engineering team to accelerate project
  - Code generation support from Mscript to target binary/embedded code

THANK YOU