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
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

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

Training and Inferencing flow

**Training**
- Raw Data
- Data Preprocessing
- Training model
- Model validation
- Model finalization

**Inferencing on Embedded Target**
- Raw data
- Data Preprocessing
- DL input
- Prediction
- Semantic Segmentation
LIDAR Semantic Segmentation

DL Training

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

Confusion Matrix

Automated Controlled Environment Vehicle - Velodyne VLP16
Framework independent platform

- Hardware Layers Compatibility
- Model performance
- Open standard model
- Realize missing/incompatible layers
- Retraining

ONNX Conversion
(Open Neural Network Exchange)

Target Code generation
C, *.C++, *.exe

- Embedded Code generation
- Executables
LIDAR Semantic Segmentation
MATLAB realization

- Raw Data
  - Data Preprocessing
  - Training model
  - Model validation

- Data input
  - Lidar Data Selection
  - Importing data set
  - Subsampling of points
  - Parsing of point clouds
Framework independent platform

- Hardware Layers Compatibility
- Model performance
- Open standard model
- Realize missing/incompatible layers
- Retraining
- Embedded Code generation
- Executables

ONNX Conversion
(Open Neural Network Exchange)

TF model
Non-Open format Model

Open format Model

Target Code generation

C, *.C++, *.exe

ONNX

Layer
I/P & O/P

Data

ONNX Conversion

ONNX Conversion

Open standard model

Realize missing/incompatible layers

Retraining

Embedded Code generation

Executables

TF model
Non-Open format Model

Open format Model

Target Code generation

C, *.C++, *.exe

ONNX

Layer
I/P & O/P

Data

ONNX Conversion
(Open Neural Network Exchange)
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-3.09344e+07%f-5.45669e+16%f-1.68519e+07%f-1737.9%f-0.000172468%f-54.3848%f-0.0651428%f-41.9732%f-830.22%f-3.0577e+07%f-2.05931e+07%f-7593.19%f-1.03164e-05%f-1.35903e-05%f-8.99545e-07%f-0.000292669%f-6.159075e-314%f-0.000117189%f-1.59076e-314%f-0.03167e-07%f-1.56832e-314%f-0.000191895%f-3.56522e-314%f-1.59069e-314%f-7.43867e-07%f-1.75476e-08%f-1.59069e-314%f-0.000191895%f-8.10622e-08%f-0.000191895%f-2.1989e-05%f-4.3576e+07%f-1.4885e+07%f-2.0328e+07%f-6.4266e+06%f-678.99%f-6.64842c-05%f-21.048%f-0.0271034%f-17.71714%f-356.68%f-1.24008c+07%f-9.13713e+05%f-3243.67%f-0.010627e-05%f-9.82409e-08%f-6.41512e-05%
Interoperability: HW Target system

Target 1

DL model

Matlab Coder

Validating performance

Code generation

Validating performance

Target interfacing

Enabling Data pipeline

Target 2 config

Target 2...

Reconstructed Organized Point Cloud

Point Cloud Capture

Validating performance

Code generation

Target interfacing

Enabling Data pipeline
Code generation

DL model

Matlab

Validating performance

Matlab Code generation

Host Target

Enabling Data pipeline

Validating performance

Mscripts

Generated Code

```c
8 // Include Files
9 #include "GroundSegmentFuncBin.h"
10 #include "GroundSegmentFuncBin_data.h"
11 #include "GroundSegmentFuncBin_initialize.h"
12 #include "convertFromCartesianToSphericalCoordinate.h"
13 #include "convertFromCartesianToSphericalCoordinate.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 //
```
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