Deep Learning at PlusAI

Xiaoshi Wang

Deep Learning at PlusAI

History

Modeling Procedure at PlusAI

- Problem/ Motivation
- Task Formation
- Data/Label Generation
- Modeling
- Evaluation
- Deployment

Challenges & Future Plans

A Little Bit of History

2017:

- 2D obstacle detection
- 2D lane detection

2020:

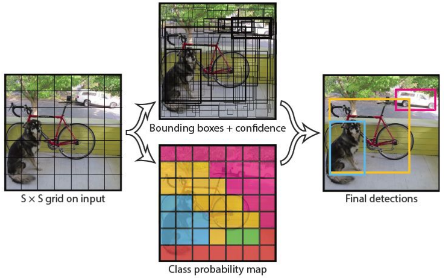
- 2D obstacle detection
- 2D lane detection
- Mono depth estimation
- 3D obstacle detection
- Prediction
- Control (vehicle model)
- ...

A Little Bit of History

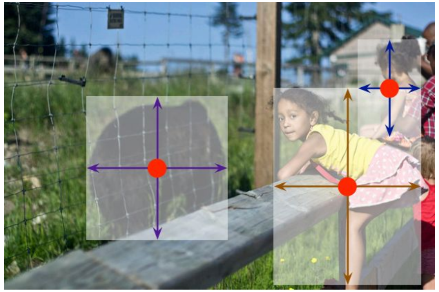
2D obstacle and lane detection

Obstacle: Yolov2 + Multiscale -> Yolov3 + Multiscale ->CenterNet

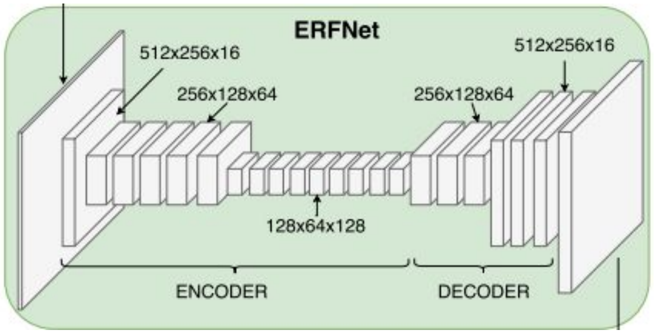
Yolov3: Anchor based



CenterNet: Anchor free



Lane: ERFNet + techniques



A Little Bit of History

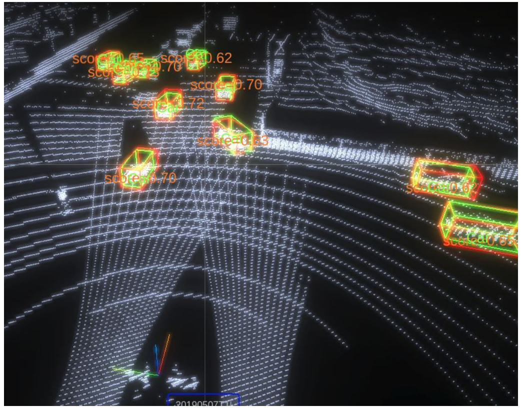
Lidar 3D obstacle detection

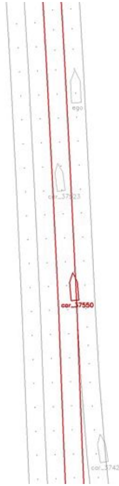
Prediction

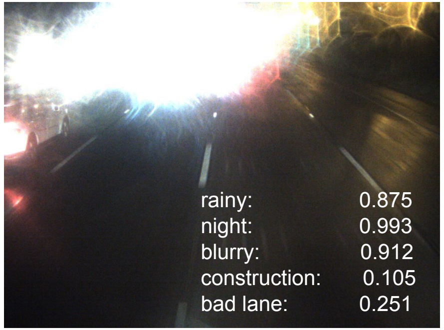
...

WIP tasks

- Object Orientation Detection
- Image Scenario Detection
- Keypoint Detection
- Image Object Tracking
- ...







Modeling Procedure at PlusAI

Motivation & Task Formation (For new tasks)

Motivation:

- Things we can improve
  in current pipeline

Task Formation:

- Learning/Non learning based.
- Classification

Modeling Procedure at PlusAI

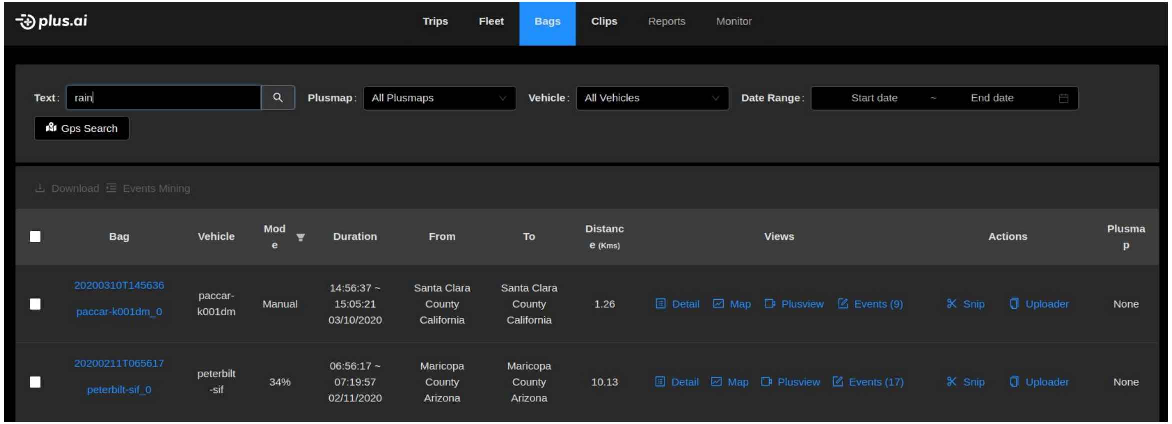
Data Generation : Data gathering

- Data Download (from data1)
- Data Extraction
  - /front_left_camera/image_color/compressed
  - /unified/velodyne_points
  - /navsat/odom
  - ...

Modeling Procedure at PlusAI

Data Generation : Data gathering

- Data Download & Extraction By Events

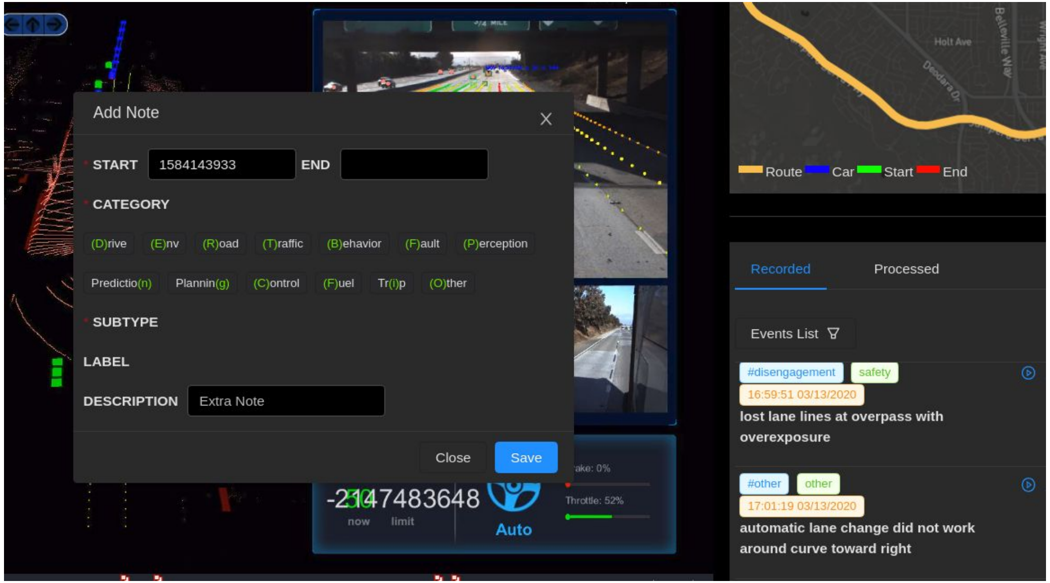


- More Automatic in Future:
  - Advanced Event Processing and Offline Data Mining

Modeling Procedure at PlusAI

Data Generation:

- Manual Annotation
  - Taxonomy
- Event Processing
  - Automatically detect and process events in Bags.



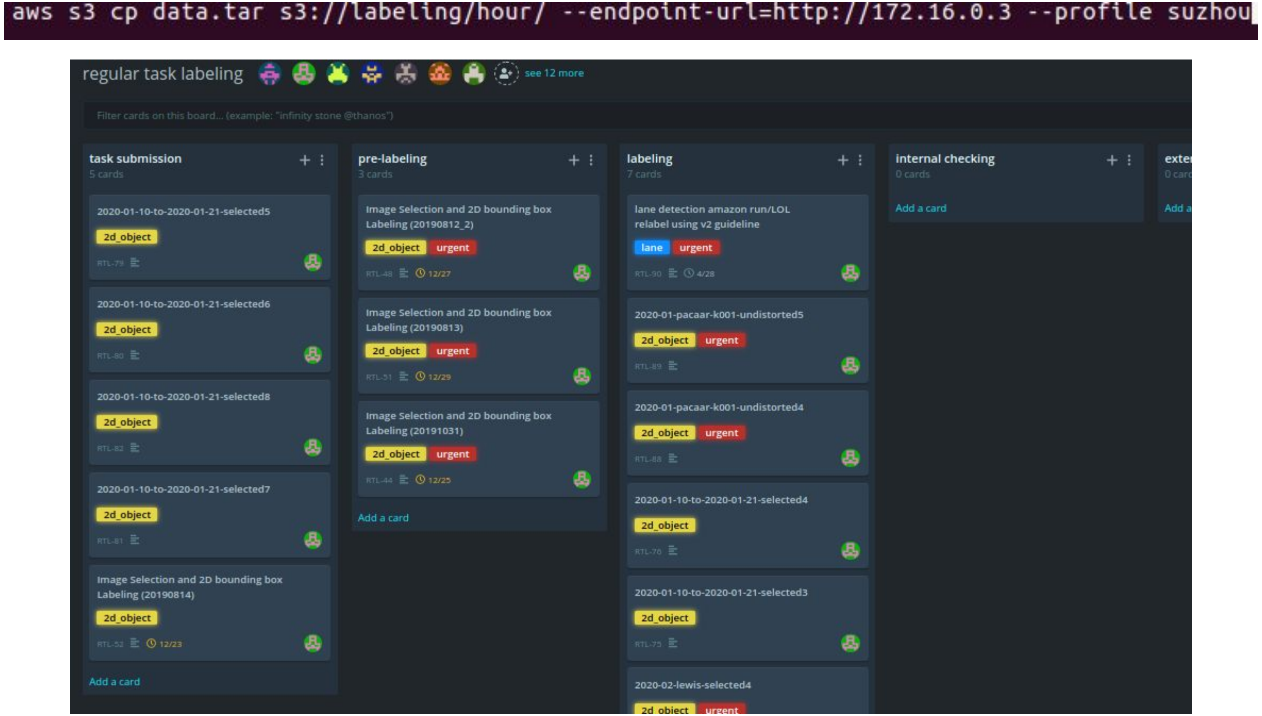
  - https://github.com/PlusAI/event_processing

Modeling Procedure at PlusAI

Data Generation: Labeling

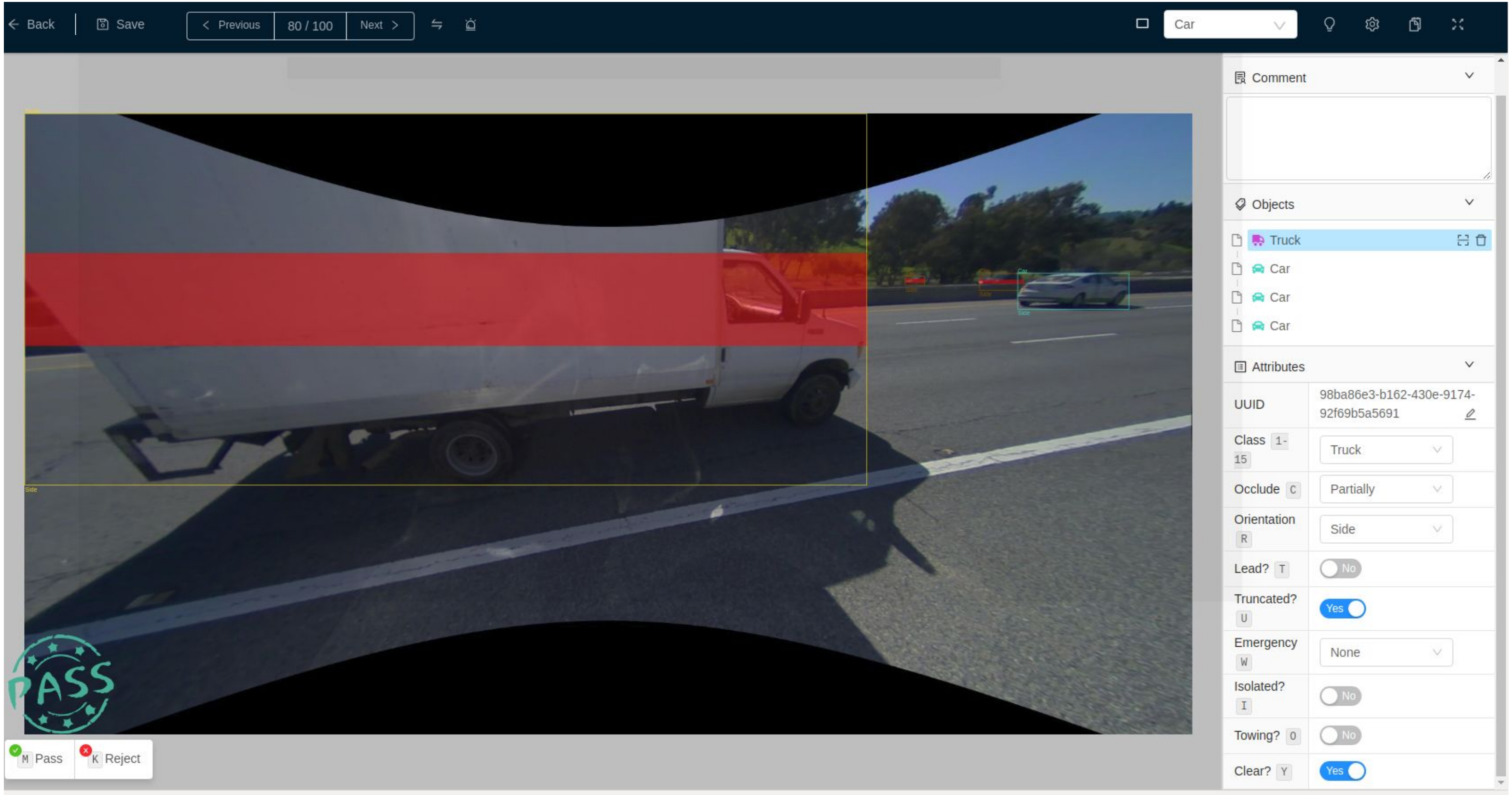
Procedure

- Data Uploading
- Task Generation
  - Data position
  - Guideline
  - Deadline
- Task Tracking
- Data Verification
  - Labeling team
  - Engineer



Modeling Procedure at PlusAI

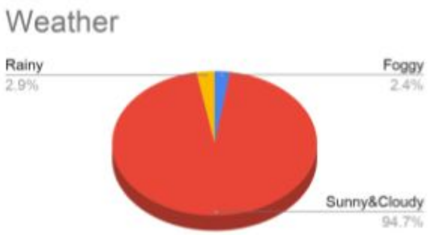
Data Generation: Data Verification (QA)

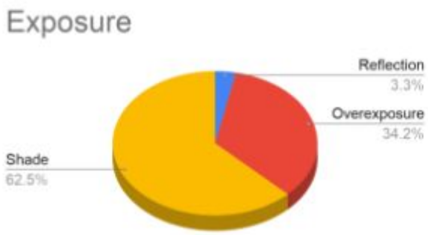


Modeling Procedure at PlusAI

Data Generation: Dataset Split, Benchmark Dataset

- Training, Validation, Test (Benchmark)
- Test set is annotated with scenarios





Modeling Procedure at PlusAI

Data Generation: Existing Datasets

Large Datasets:

- 2D obstacle detection: 100000 frames
- 2D lane detection: 60000 frames
- 3D lidar&image tracking: 50000 frames
- 2D ground touching points (side view): 30000 frames

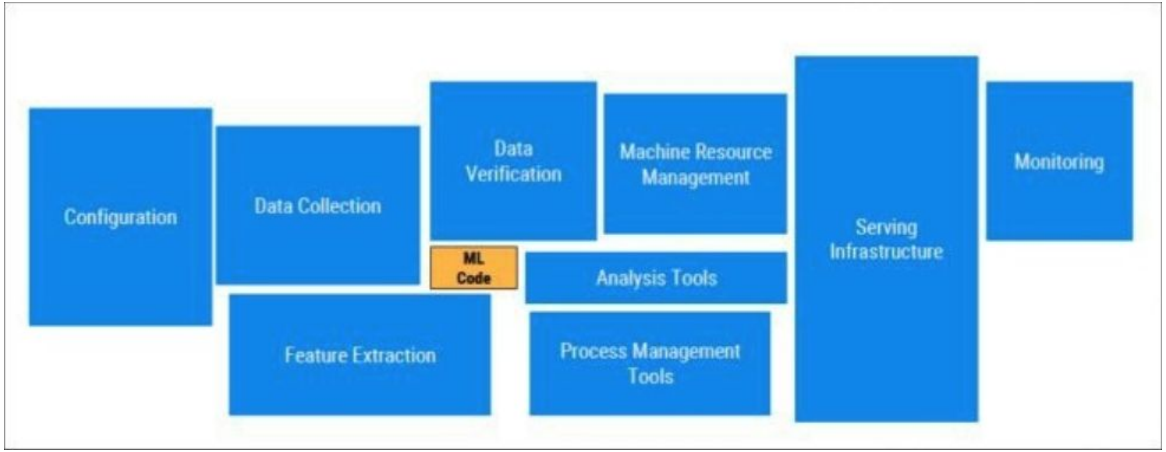
Small Datasets:

- 2D Key point data (2000-4000 frames in May)
- Semantic & instance segmentation data (200-500 in May)

Modeling Procedure at PlusAI

Modeling -- Coding

- Data pipeline
- Model Architecture
- Evaluation Metrics
- Training Script



Modeling Procedure at PlusAI

Modeling -- Training

Hardware: Suzhou Gpu Clusters, AWS

```
xiaoshiw@gpu6:~$ nvidia-smi
Wed Apr 29 11:26:43 2020
+-----------------------------------------------------------------
| NVIDIA-SMI 410.129      Driver Version: 410.129      CUDA V
|-------------------------------+----------------------+-----
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Vola
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-
|===============================+======================+=====
|   0  GeForce RTX 208...  On   | 00000000:1A:00.0 Off |
| 29%   49C    P2    93W / 250W |  10947MiB / 10989MiB |
```

Training Procedure

- Structure & capacity
- Loss
- Scale
- Tricks (hard negative sampling etc.)
- Hyperparameter tuning

Model

Name	Status	Performance	Note
Anchor regeneration	Finished for small scale model; In progress for large scale	Positive	The improved small scale model has been merged.
RefineNet	Finished	Negative	
RetinaNet	Implemented		Planned for detailed examination in future
Yolo v3 (Darknet43) + multiscale output	Implemented	Positive for faraway objects Testing in progress	
Yolo v3 (Darknet26) + multiscale output	Implemented	Positive for faraway objects Testing in progress	
Network random search	Implemented v1		Planned for detailed examination in future
Focal loss	Finished	Positive	

Modeling Procedure at PlusAI

Modeling -- Training ML Platform (in recent future)

Data Repo

→ Extract data by events /rules →

Labeling Platform: Data Labeling & Management

→ Sample labeled data by events /rules →

ML Platform: Distributed Training, Auto Tuning, Logging, Model Storage, Evaluation

→ Pick candidates with best performances →

CI Platform: Candidate Model Evaluation (on whole perception pipeline) Model Deployment

kubernetes

Kubeflow

HOROVOD

Neural Network Intelligence

aws

Azure

Modeling Procedure at PlusAI

Evaluation Metrics:

- Thorough understanding of model
- Product orientated

Obstacle Detection:

- Goal: Large objects, nearby objects, ego lane objects, pedestrian, robustness
- Metrics Design:
    - metrics based on obstacle size,
    - truncated obstacles,
    - occluded obstacles
    - front obstacle,
    - ego lane obstacles,
    - Robustness of objects in continuous frames

Modeling Procedure at PlusAI

Evaluation: PlusAI vs KITTI

	PlusAI	KITTI
Metrics	mAP, Precision, Recall, F scores, Confusion Matrix	mAP
Scenarios	Large object (bucketized); Truncated; Occluded; Ego Lane Object (fp); Front Vehicle (recall); Robustness;	Truncated + Occluded
Deployment	Class & size weighted F1 score with precision & recall restrictions	


```
"box_size": "0.003-0.004",
"result": {
  "num_images": 4757,
  "prob_threshold": 0.5,
  "eval_threshold": 0.5,
  "mAPs": {
    "car": 0.9567511980608188,
    "bus": 0.28188229696109174,
    "truck": 0.762556853647838,
    "moto": 0.004201659663970588,
    "pedestrian": 0.9473679224379354,
    "bike": 0.8333291666874998,
    "traffic_cone": 0.8461531952667729,
    "barricade": 0.368865533361575
  },
   "num_vehicles": {
     "car": 1959,
     "bus": 34,
     "truck": 264,
     "moto": 2,
     "pedestrian": 19,
     "bike": 2,
     "traffic_cone": 13,
     "barricade": 332
   },
   "num_front_vehicles": 180,
   "front_car_recalls": {
     "0.8": 0.8944444439475309,
     "0.85": 0.8944444439475309,
     "0.9": 0.8944444439475309,
     "0.95": 0.8944444439475309
   },
   "all_recalls": {
     "0.8": 0.8205714285401687,
     "0.85": 0.814095238064225,
     "0.9": 0.8076190475882812,
     "0.95": 0.7912380952079529
   },
```

```
"F scores": {
  "0.5": 0.9185991387898435,
  "1.0": 0.8420824245937985
},
"precision": {
  "car": 0.9858916478499669,
  "bus": 0.999999995,
  "truck": 0.8989361701649502,
  "pedestrian": 0.9999999993333333,
  "traffic_cone": 0.9999999985714286,
  "barricade": 0.9999999900000002,
  "all": 0.9778337531436886
},
"recall": {
  "car": 0.8917815211797255,
  "bus": 0.0588235293944636 7,
  "truck": 0.640151515127267,
  "pedestrian": 0.7894736837950138,
  "traffic_cone": 0.5384615380473372,
  "barricade": 0.003012048192680359 8,
  "all": 0.7394285714257546
},
"ego_lane_fp/total_fp": 0.6454116572810145,
"ego_lane_fp/total_images": 0.891528274122346,
"confusion matrices on size-matched detection
for gt, col for det)": [
  " 1747,    0,   14,    0,    0,    0",
  "    3,    2,    1,    0,    0,    0",
  "    6,    0,  169,    0,    0,    0",
  "    2,    0,    0,    0,    0,    0",
  "    0,    0,    0,    0,   15,    0",
  "    0,    0,    0,    0,    0,    0"
]
```

Modeling Procedure at PlusAI

Evaluation -- Analysis

Why the new network/technique performs better

Why is CenterNet better than Yolov3?

Hypotheses	Possible effects	How to validate?
CenterNet has a better backbone.	++ small objects ++ large objects	Replace the darknet backbone of Yolov1 with DLA
CenterNet uses soft labels for the locations while Yolov3 uses hard labels.	+ small objects + large objects	Hard to validate
The regression loss function of CenterNet cares more for large objects.	-- small objects ++ large objects	Use a similar loss function for Yolov3
Small bounding boxes are not filtered out during training of CenterNet.	++ small objects - large objects	Train Yolov3 without filtering out small objects. Validated

Deployment

Runtime API

Model Speedup

PR Submission

```cpp
class RTObjectDetector : public ObjectDetector {
  public:
    typedef std::shared_ptr<RTObjectDetector> Ptr;

    // async function. will submit jobs to GPU and return immediately.
    void DetectAsync(const cv::cuda::GpuMat& img) override;
    // this function will block until GPU job finished.
    void GetDetection(std::vector<models::GenericDetection>& detections) override;
    // this function will block until GPU job finished.
    void GetDetection(std::vector<GenericDetection3D>& detections_3d) override;
    // this function will block until GPU job finished.
    void GetFeatures(cv::Mat& features) override;
    bool IsDetectionFinished() override;

    RTObjectDetector(const StereoDetectorParam& conf,
                     const boost::filesystem::path& file_root_path,
                     bool get_image_features = false);

  private:
    std::unique_ptr<TensorrtDetector> _detector;
    const StereoDetectorParam _conf;
    int _gpu_id;
};
```

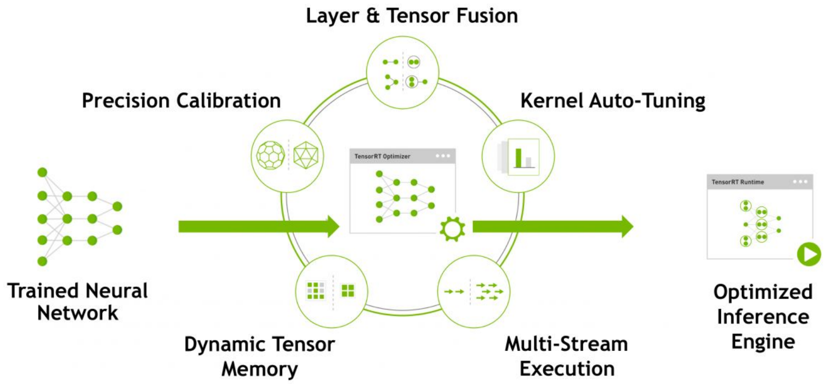
Metric	Equation	Overall	close_front_ego_ego_lane x: (2, 50) , |y|: (0, 2)	close_front_neighbor_lane x: (2, 50) , |y|: (2, 6)	medium_front_ego_lane x: (50, 100) , |y|: (0, 2)	medium_front_neighbor_lane x: (50, 100) , |y|: (2, 6)	far_front_ego_lane x: (100, 150) , |y|: (0, 2)
Recall	TP / (TP + FN)	0.591 -> 0.586	1.000 -> 1.000	0.996 -> 0.995	0.988 -> 0.981	0.961 -> 0.956	0.956 -> 0.959
Precision	TP / (TP + FP)	0.750 -> 0.741	1.000 -> 1.000	0.976 -> 0.974	0.932 -> 0.949	0.964 -> 0.968	0.885 -> 0.888
Matching Rate	1 - MM / TP	1.000 -> 1.000	1.000 -> 1.000	1.000 -> 1.000	1.000 -> 1.000	1.000 -> 1.000	1.000 -> 1.000
Multi Object Tracking Accuracy	1 - (MM + FP + FN) / (TP + FN)	0.394 -> 0.381	1.000 -> 1.000	0.971 -> 0.969	0.915 -> 0.928	0.925 -> 0.925	0.832 -> 0.838
3d cuboid iou	mean(iou3d(tp_has_lidar, gt_has_lidar))	0.193 -> 0.194	0.395 -> 0.391	0.492 -> 0.492	0.193 -> 0.193	0.161 -> 0.162	0.024 -> 0.026
2d box iou	mean(iou2d(tp_has_camera, gt_has_camera))	0.692 -> 0.691	0.866 -> 0.865	0.828 -> 0.828	0.787 -> 0.780	0.782 -> 0.777	0.655 -> 0.656

Deployment

Model Speedup: TensorRT, Cuda post processing

TensorRT

- Layer / Tensor Fusion
- Auto-Tuning
- Precision Calibration
- Multi-Stream Execution
- Dynamic Tensor Memory



Offline: TensorFlow -> UFF, PyTorch -> ONNX

Online: UFF / ONNX -> TensorRT Engine

Cuda post processing

- Run lots of tasks in parallel

Challenges & Future Plans

Data Filtering

-- event processing, online & offline

Model Latency

-- multitask learning

Model Tuning

-- distributed training, AutoML