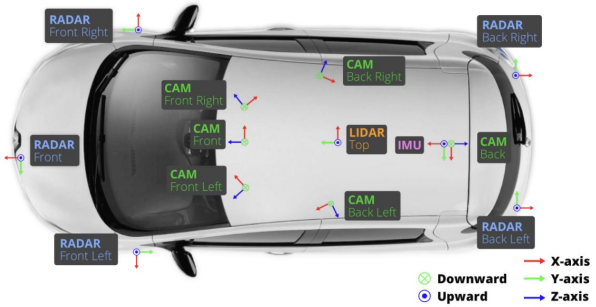
3D object detection from arbitrary RGB camera rigs

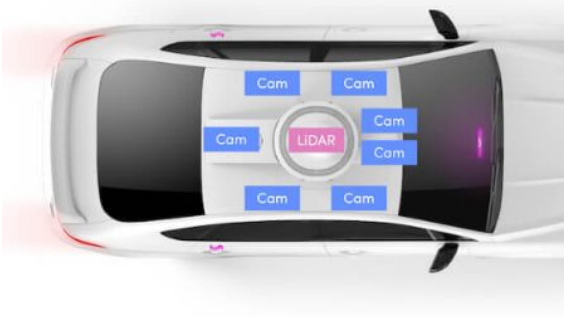
Ayush Baid and Nitish Sontakke

Problem Statement

- Object detection from monocular RGB images
  - Very little overlap between camera frustums



Source: nuScenes



Source: Lyft

Proposed Solution - Lift Splat Detect

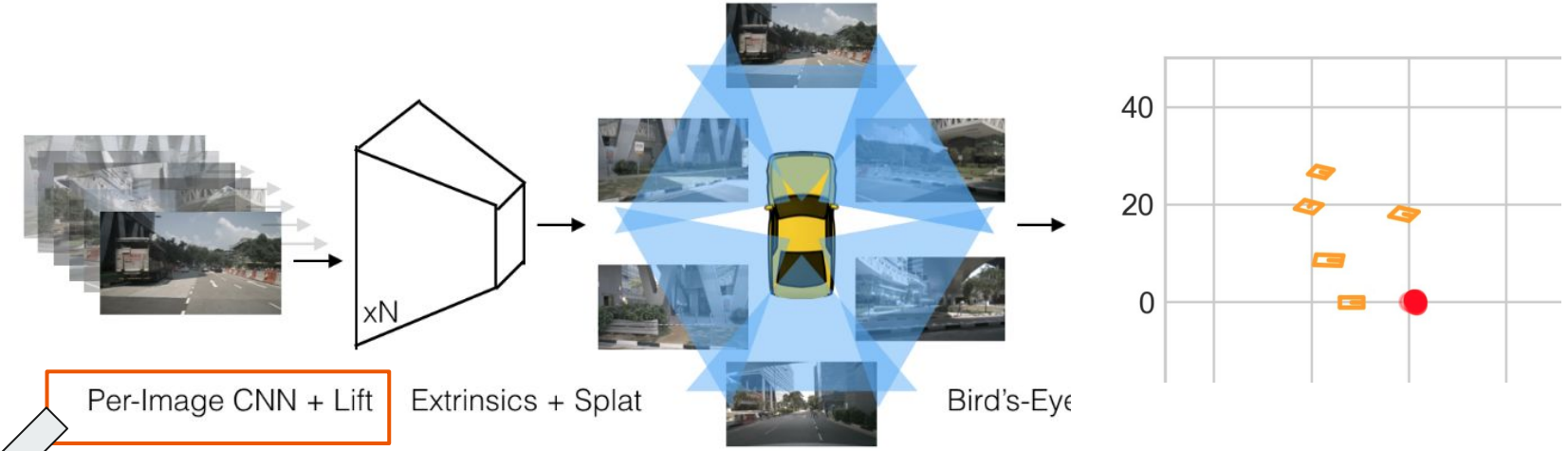
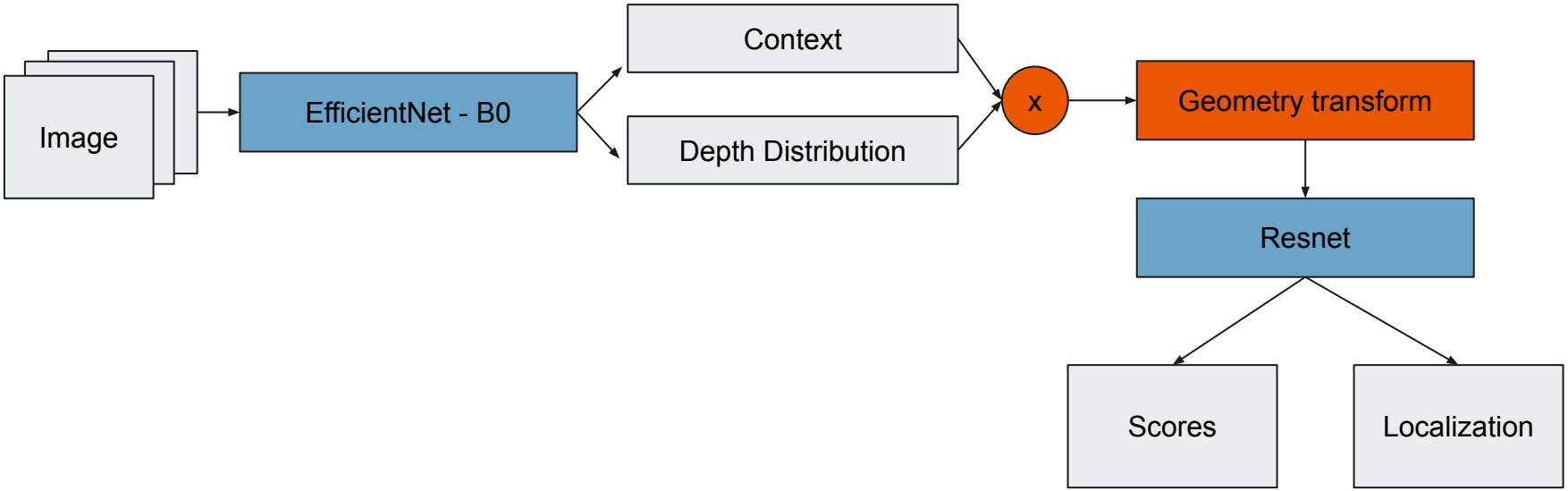


Image Credits: Phillion el al. ECCV 2020

Proposed Solution - Network



Proposed Solution - Loss

We then experiment with 3 different loss functions for *scores*:

- **Huber loss**
- **Huber loss with reweighting negative samples** (used in OFT, Roddick et al.)
- **Focal loss** (used in CenterPoint, Yin et al.)

For all other outputs, we use a Huber loss evaluated on just the anchors which have overlap with Ground Truth (GT).

Baseline

We are working with the **nuScenes** dataset.

Since we have limited compute, we cannot run all benchmarks ourselves and will rely on leaderboard results to serve as our baselines. We compare our proposed method to the following camera-only methods:

- CenterNet
- Mono-DIS

Results #1 - Monocular Training

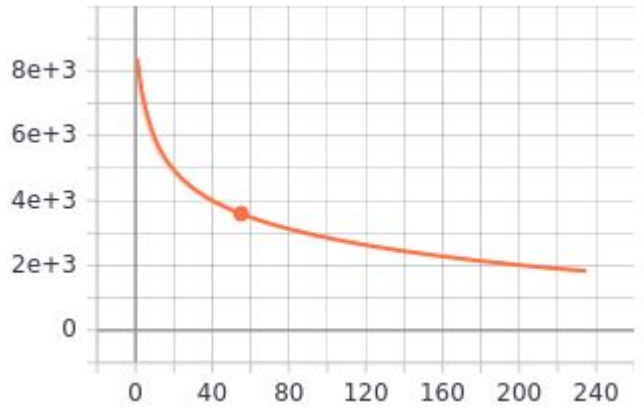
- Training on monocular images for 50 epochs

Method	**mAP (↑)**	**ATE (↓)**	**ASE (↓)**	**AOE (↓)**
Lift-Splat-Detect (50 epochs) (val)	0.22	0.52	0.16	0.15
CenterNet	0.54	0.47	0.14	0.09
Mono-DIS	0.48	0.61	0.15	0.07

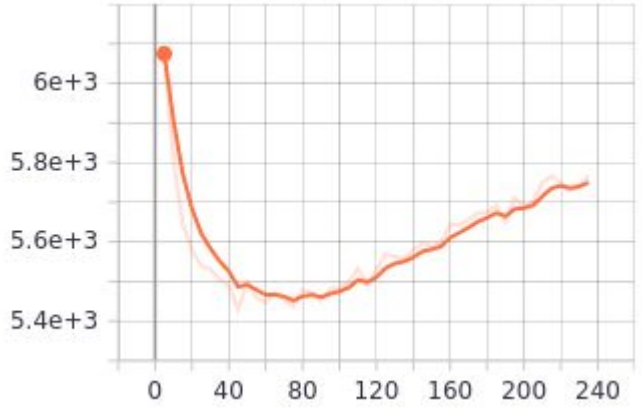
Object Detection for Cars

Issues - Monocular Training

Overfitting!



Loss on train set



Loss on val set

Proposed Solution #2

- Use LiDAR as privileged information: available during training, absent during test
- How to use PI? **Heteroscedastic Dropout**
  - Multiplicative dropout sampled from a distribution
  - Variance controlled with PI

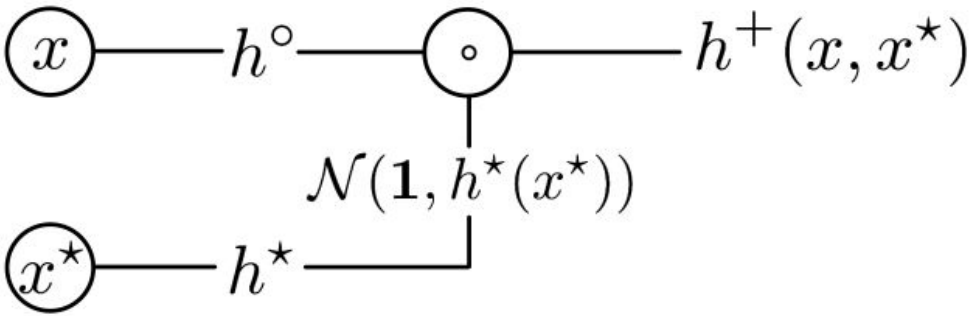
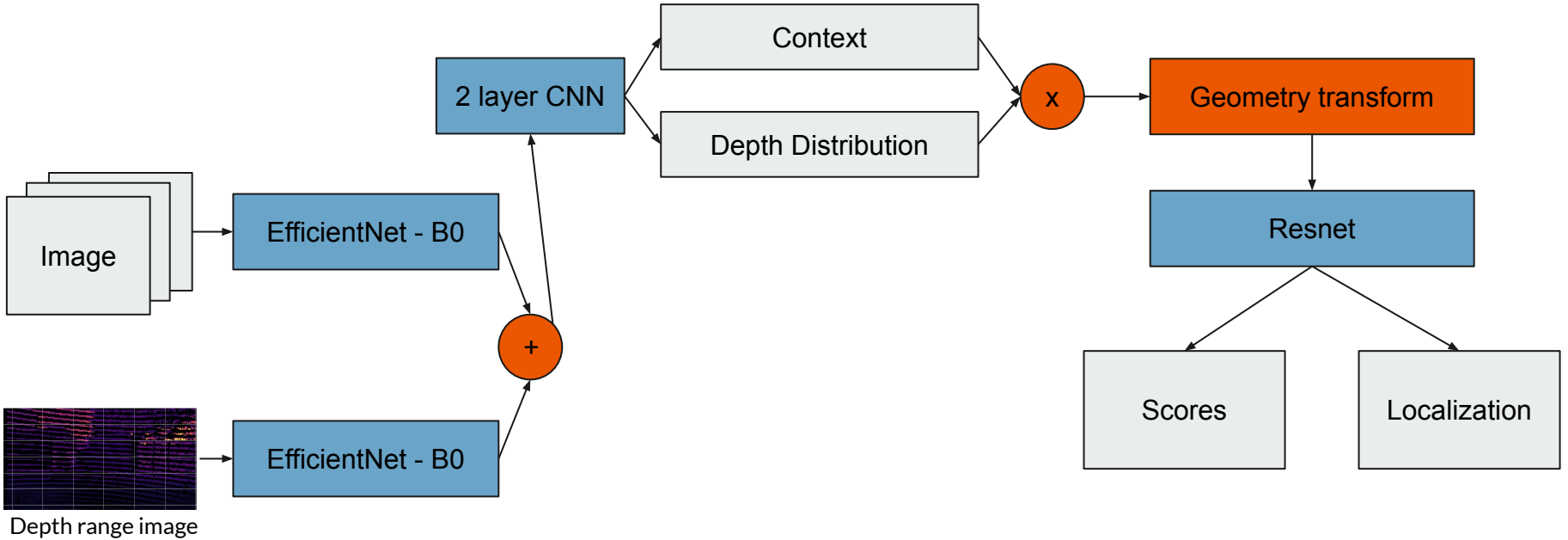


Image Credits: Lambert el al. CVPR 2018

Method #2 - Sensor Fusion

- Establishes the ceiling of our method



Results #2 - Sensor Fusion

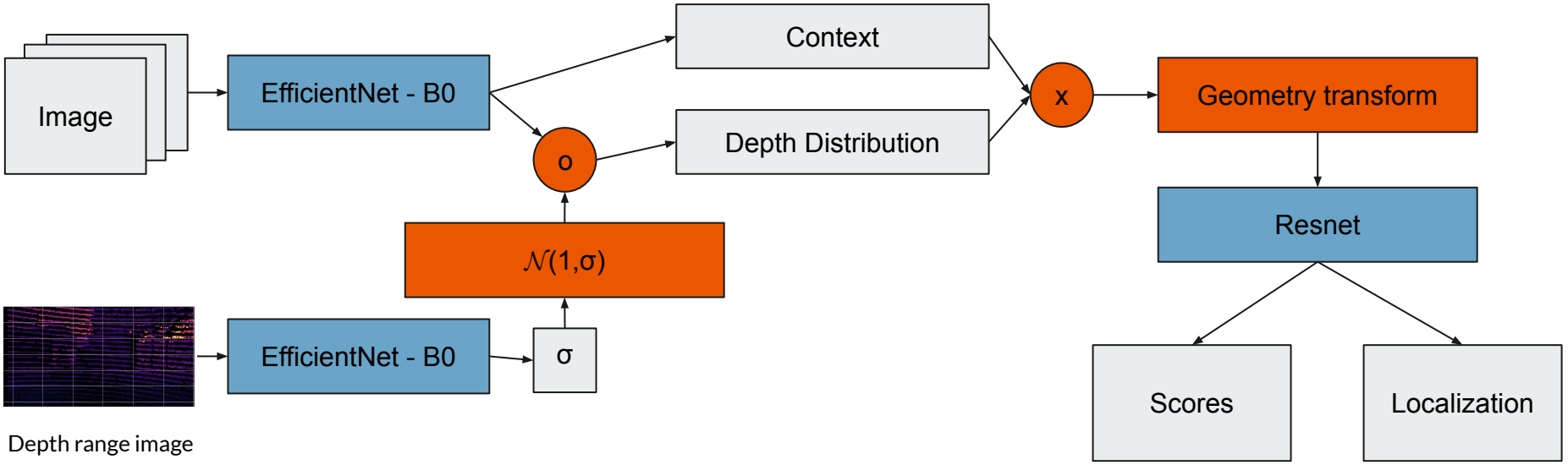
- Trained for 50 epochs, results reported on val split

Method	**mAP (↑)**	**ATE (↓)**	**ASE (↓)**	**AOE (↓)**
RGB only	0.22	0.52	0.16	0.15
RGB + LiDAR depth	0.37	0.44	0.15	0.15

Object Detection for Cars

Method #3 - PI for depth distribution

- Better depth detection leads to huge boost in performance



Depth range image

Loss=L_score + L_localization + α || σ ||^2

Method #3 - PI for depth distribution

- Expectations:
  - Faster training (Lambert et al., CVPR 18)
  - Performance improvement (Kamienny et al. ICLR 20)

Method	mAP (↑)	ATE (↓)	ASE (↓)	AOE (↓)
RGB only	0.134	0.547	0.153	0.189
RGB + LiDAR depth	0.180	0.492	0.156	0.203
RGB + PI (α=1e-3)	0.148	0.551	0.154	0.177
RGB + PI (α=1e-1)	0.135	0.544	0.150	0.198
RGB + PI (α=1e-4)	0.137	0.530	0.152	0.181

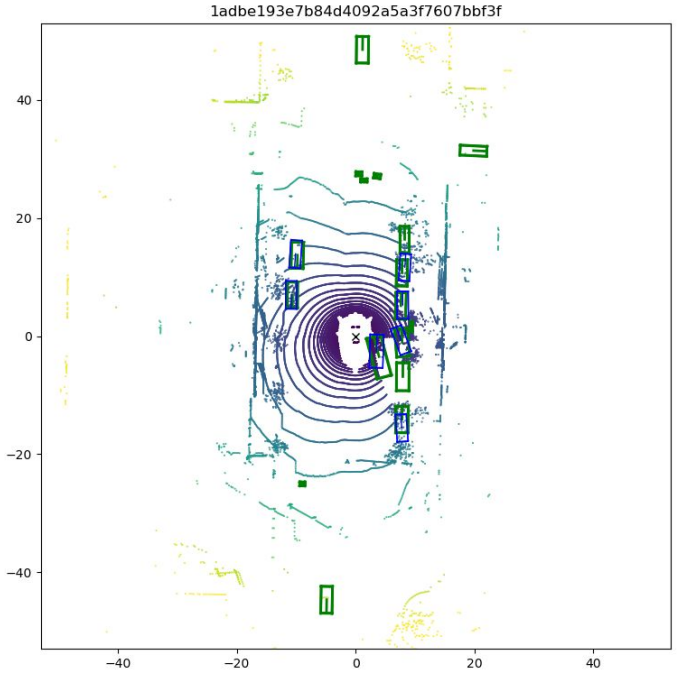
5 epochs

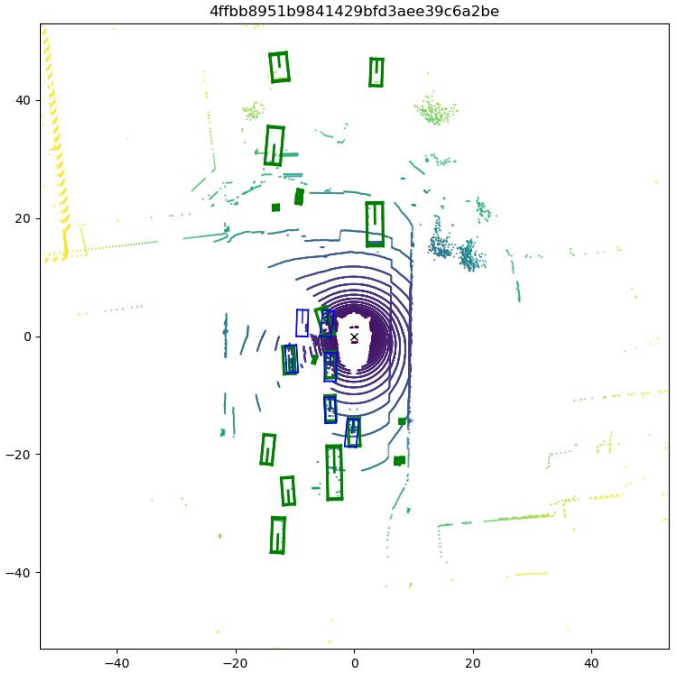
Method #3 - PI for depth distribution

- Performance becomes worse at higher number of epochs

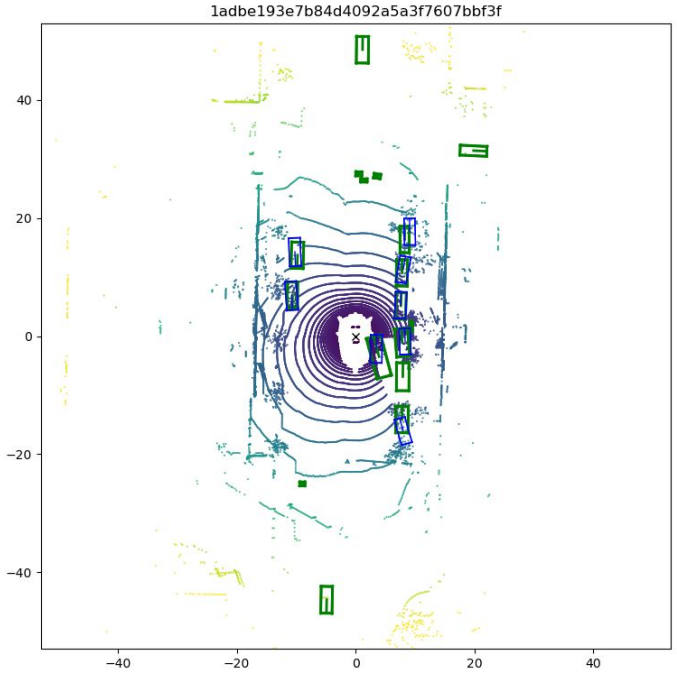
	10 epochs				50 epochs			
Method	**mAP (↑)**	**ATE (↓)**	**ASE (↓)**	**AOE (↓)**	**mAP (↑)**	**ATE (↓)**	**ASE (↓)**	**AOE (↓)**
RGB only	0.191	0.541	0.153	0.150	0.221	0.522	0.159	0.147
RGB + LiDAR depth	0.244	0.474	0.152	0.138	0.372	0.450	0.160	0.146
RGB + PI (α=1e-3)	0.177	0.529	0.156	0.141	0.219	0.533	0.158	0.153
RGB + PI (α=1e-1)	0.173	0.516	0.157	0.177	0.217	0.537	0.161	0.156
RGB + PI (α=1e-4)	0.172	0.527	0.150	0.160	0.205	0.520	0.161	0.160

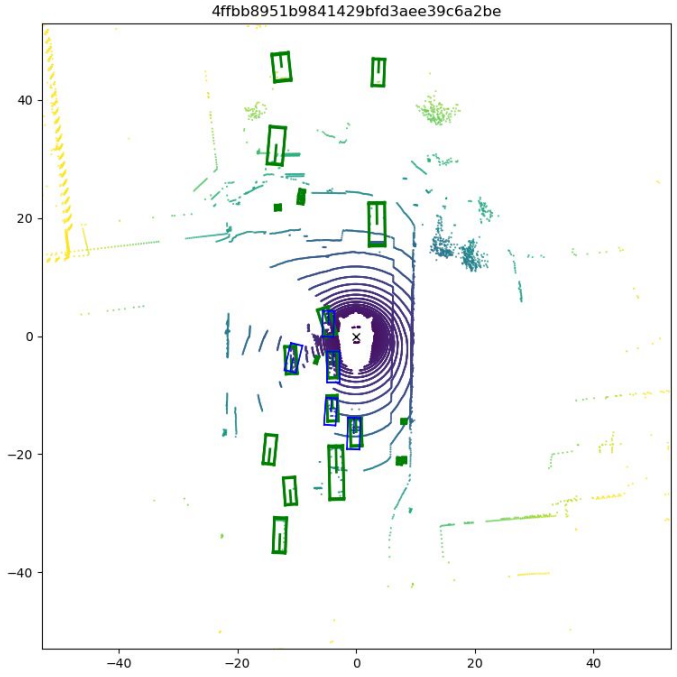
Visualizations - RGB only training





Visualizations - RGB + PI





Improvements to do

- Fix overfitting in RGB-only model and achieve improved performance
- Do multi-class training (help prevent overfitting too)
- Drill down into PI:
  - What are the statistics of the dropout values?
  - What are the regions where dropouts have high variance?
- Have a depth prediction loss?
- Try PI on an established 3D detection algorithm

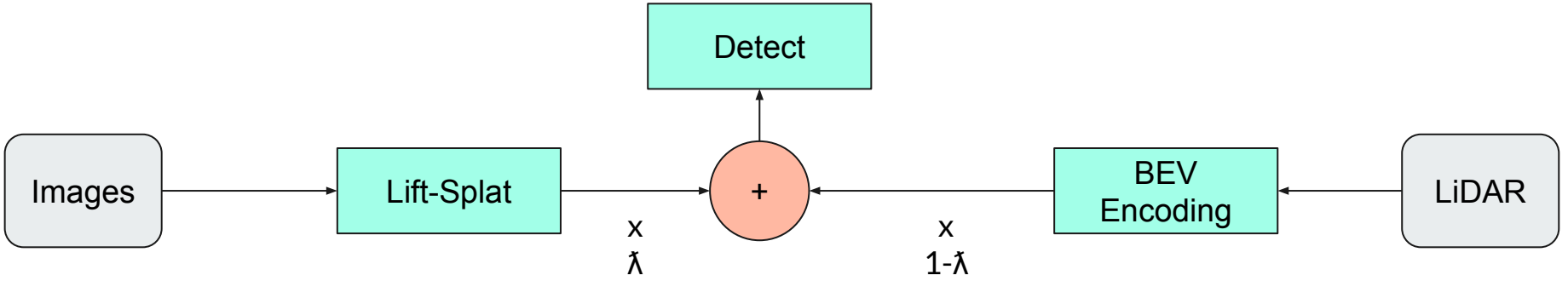
Questions?

Plan for the remainder of the semester

Tasks	Date
Training on full nuScenes using Huber Loss	11/06
Evaluation of mAP and other metrics (BEV or 3D detection?)	11/06
Experiment with other loss functions	11/13
Use a bigger ResNet in the Detect portion of the model (if feasible)	11/20
Training with LiDAR as privileged information	11/27

Performance Improvement

- Stereo-based Object Detection perform exceptionally well (e.g. Deep Stereo Geometry Stereo Network, Chen et al.)
  - AV datasets have very small overlap between cameras
- Use HD-Maps rasterized to BEV frame
- LiDAR as privileged information?
  - Force the LiFT part of the model to learn similar data as LiDAR input



Baseline

The original OFT results are reported on the **KITTI** benchmark.
However, since we're working with nuScenes, we decided to train and evaluate OFT on it. We ran the following experiments for 100 epochs each:

- OFT on **full nuScenes** dataset: 700 scenes, ~21,000 samples
- OFT on **mini nuScenes** dataset: 10 scenes, ~300 samples

However, we noticed the results were very poor. We also noticed a lot of overfitting in the mini dataset.

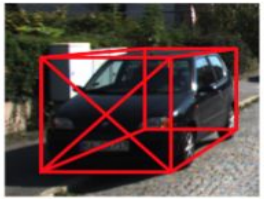
We had the authors correct the bug and reran our experiment, but the trained model failed to detect any objects.

We are therefore trying to replicate OFT results on the KITTI dataset itself as a sanity check.

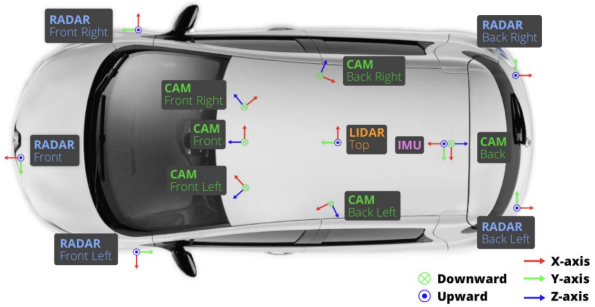
Proposal slides

Motivation

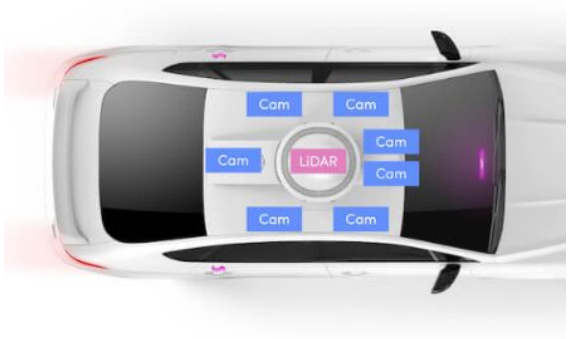
- **3D object detection** in images is a hard problem: **lack of depth** information
- AVs have a camera rig: can we use images from multiple-cameras to help with lack of depth?
  - Can we generalize to arbitrary camera rigs?



Credits: Deep3DBox, Mousavin et. al. CVPR 2017



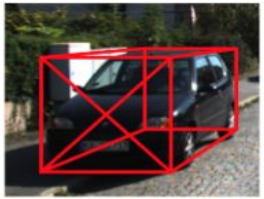
Source: nuScenes



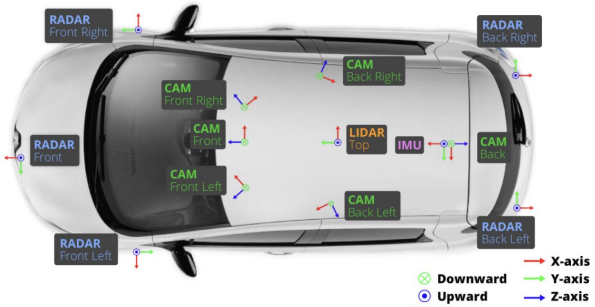
Source: Lyft

Motivation

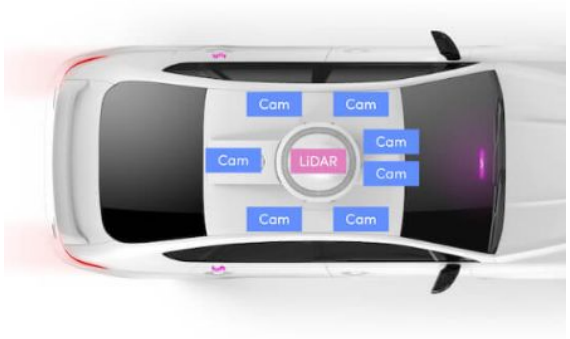
- **3D object detection** in images is a hard problem: **lack of depth** information
- AVs have a **camera rig**: can we use images from multiple-cameras to help with lack of depth?
  - Can we **generalize** to **arbitrary** camera **rigs**?



Credits: Deep3DBox, Mousavin et. al. CVPR 2017



Source: nuScenes



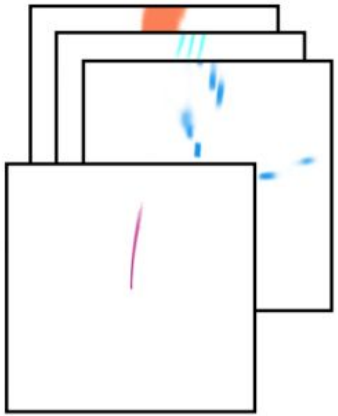
Source: Lyft

Related Work

Lift-Splat-Shoot: Encoding Images from Arbitrary Camera Rigs by Implicitly Unprojecting to 3D

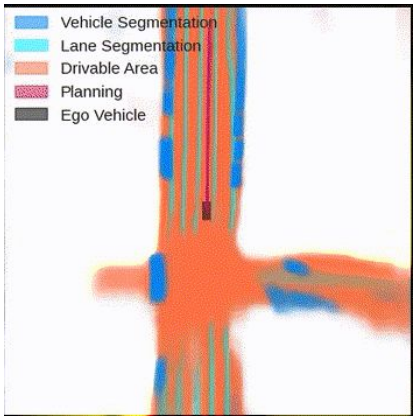
Input:
RGB images, camera intrinsics and extrinsics

Output:
BEV representation



Input:
BEV representation

Output:
BEV segmentation
BEV planning

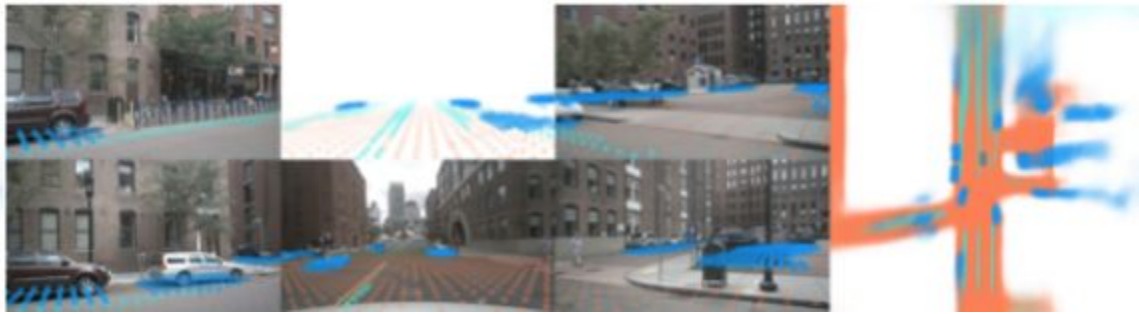


Source: Lift-Splat-Shoot, Philion et. al., ECCV 2020

Related Work

Why lift-splat-shoot?

- Robustness built in:
  - Camera dropout
  - Arbitrary rig geometry

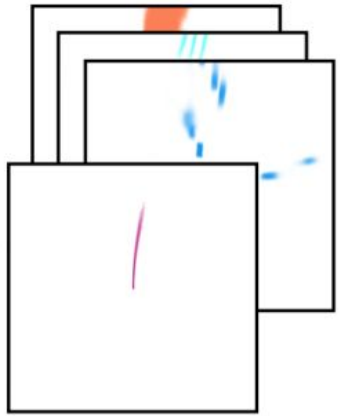


Source: Lift-Splat-Shoot, Philion et. al., ECCV 2020

Proposed Method

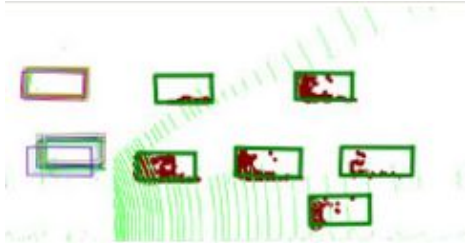
Input:
RGB images

Output:
BEV representation



Input:
BEV representation

Output:
BEV object detection



Source: Lift-Splat-Shoot, Philion et. al., ECCV 2020

Project Plan

- **Action plan**
  - Ask the authors for pretrained models; run it to reproduce lift-splat-shoot
  - Obtain BEV detection **baselines** for **lidar-only** and **lidar+image** methods
  - Add a detection network and train for detection task
  - [Optional] Utilize LiDAR data as privileged information
  - [Optional] Run 2D detection on mono-images and apply lift-splat to them too
- **Dataset**: nuScenes
- **Compute resources**: Personal machine, and Google Colab if needed

Thank you!

Presentation Discussion

- Compare with MV3D
- Compare with Panorama based methods (Panorama360)
- Project detections back in image to get 2D (or 3D detections)

Doubts

- Is the Lift, Splat module trained end-to-end with the motion planning (Shoot) part?
- If not, what loss is used to train it?
- If we were to use just Lift, Splat with an object detection loss, would it work? Or is the BEV segmentation learnt because of the Shoot part? Will training in a multi-task manner where we include object detection hamper the training?
- When do you plan on releasing the code? Will there be a pretrained model available?

Whacky ideas

- Get 2D detections in mono images; lift-splat the detections (as probabilistic 3D bounding boxes)
- Use LiDAR information as prior/privileged:
  - Before lift-splat
  - In the lift-splatted BEV frame
- Use HD maps in lift-splat to improve BEV transformation
  - If we use it, we become country specific
-