Vehicles' Pose Estimation Based on CenterNet

Shengjian Chen, Jingkun Zhang, Yiming Zhang
CSE 599G1 Deep Learning, University of Washington

Introduction

- Self-driving cars have come a long way in recent years. The position of nearby vehicles is a key question for autonomous driving — and it's at the heart of our newest challenge.

Fig.1 Kaggle competition， Peking University/Baidu Autonomous Driving

- The problem we are solving is to estimate the absolute 6DOF (x, y, z, roll, pitch, yaw) poses of vehicles from a single image in a real-world traffic environment.

Dataset

- The dataset contains around 4000 training and 2000 test images. 6DOF poses of cars in training images are provided as training labels.

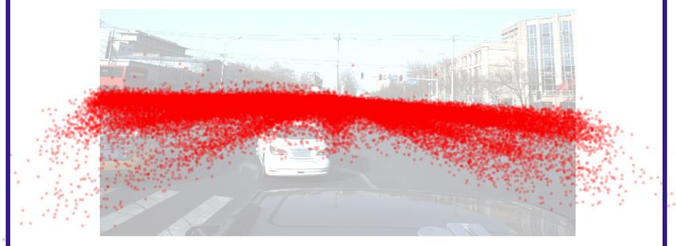


Fig.2 All cars projected to the same image

Methodology

1. Data processing

- ***Image Cropping & Padding:*** Crop upper half of images and pad left and right sides.
- ***Target Processing:*** Change raw targets to the prediction targets in section 3.
- ***Data Augmentation:*** We try horizontal flip but that harms the regression of rotation. Won't use it in our final model.

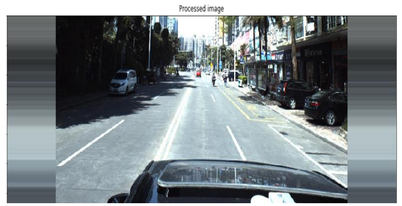


Fig.3 Image cropping and padding

2. Network Architecture

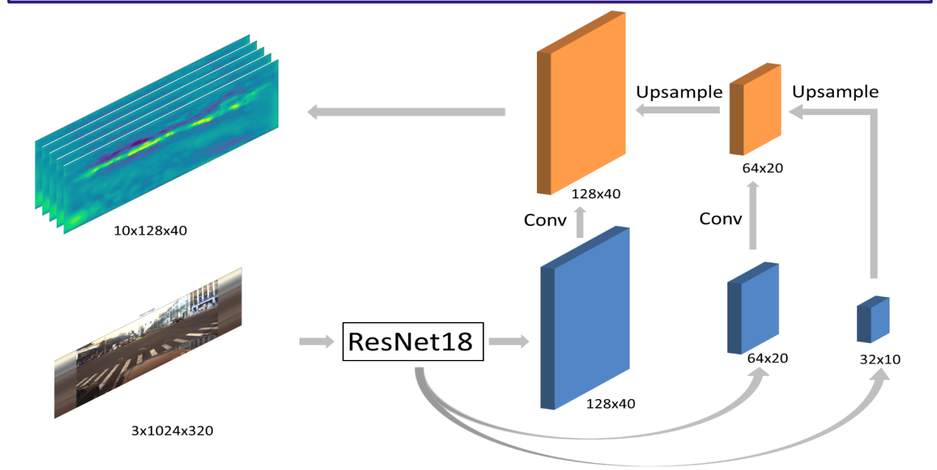


Fig.4 CenterNet Architecture

3. Prediction Target and Criterion Design

- ***Mask*** indicates the probability that there is a car falling on this cell. To address the imbalance between foreground and background, we use focal loss for mask:

$$L_{mask} = -(1-p_t)^2 \log(p_t), \qquad p_t = f(x) = \begin{cases} p, & y = 1 \\ 1-p, & otherwise \end{cases}$$

- ***XY Coordinate*** represents the offset of car's center from cell's center. We use mean squared error to measure XY distance.
- ***Z Coordinate*** is equivalent to the depth of cars. We predict log(z) and use a loss function as:

$$L_Z(z, z^*) = \frac{1}{n}\sum_i d_i^2 - \frac{\lambda}{n^2}(\sum_i d_i)^2$$

where $d_i = \log(z_i) - \log(z_i^*)$ and z_i^* is ground truth, λ = 0.5.

- ***Rotational Coordinate*** include 3 Euler angles. We predict pitch and roll directly but yaw, also means orientation, is different. As points out in literature, predicting the local orientation (θ_l) is better than predicting global (θ) directly. We use in-bin regression with 2 bins ($\frac{\pi}{2}$ and $-\frac{\pi}{2}$) to regress this local orientation. We used L1 norm for angle regression and cross entropy for bin regression

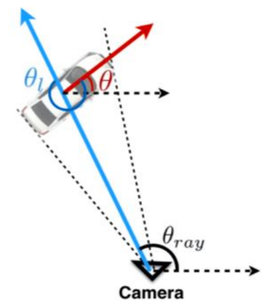


Fig.5 : Illustration of orientation

- **Total loss** is the weighted sum of the above 4 loss components:

$$L = 5 * L_{mask} + 0.5 * (L_{xy} + L_z + L_{rotation})$$

4. Experiment Details

Due to the constrain of devices, we can only use a batch size of 2 but we update weights every 10 iterations with cumulated gradient. So we actually train with a batch size of 20. We use Google Colab to train our model for 20 epochs. Learning rate of the first 10 epoch is 0.001 and the last 10 epoch is 0.0001, weight decay is 0.0005. Each epoch takes around 25 minutes.

Future work

We plan to implement Mask R-CNN model and extend it to estimate car's pose.

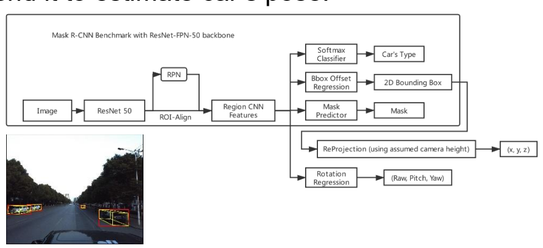
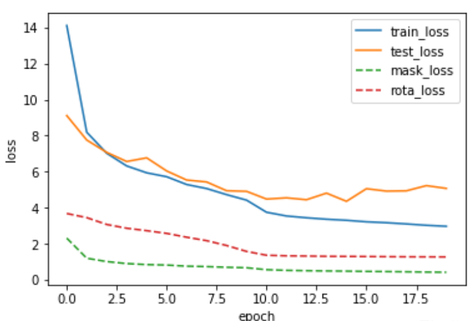


Fig.7 Proposed Network Architecture

Results and Discussion

- Most cars are recognized and pose predictions look reasonable.
- Currently holding the top 13% result in the competition.
- False positive is the main problem of our model but ground truth seems to have many false negatives.
- Mask loss on test set increases in later epochs but that won't be a problem with proper non-max suppression.



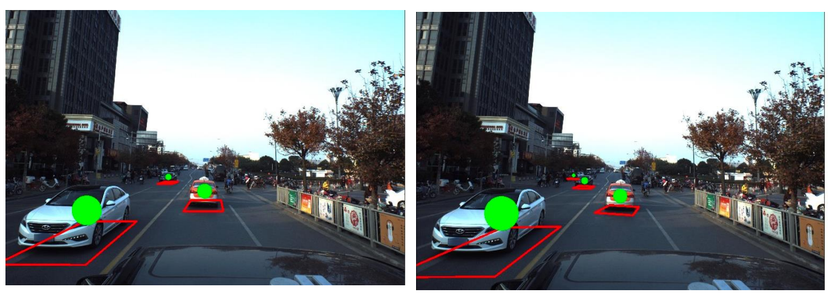


Fig.6 : Left: loss. Middle: ground truth. Right: prediction