Spring 2020 Deep Learning
Object Detection and Road Map Prediction

Jiayi Du (jd4138) Meiyi He (mh5275) Ziyu Lei (zl2350)

Abstract

Autonomous vehicles commonly rely on highly detailed birds-eye-view(BEV) maps of the surrounding environment, which capture both static elements such as road layout, but also dynamic elements such as cars, pedestrian and animals. Instead of using LiDAR-based methods, we have focused on image-based approaches to generate BEV maps, and explore images captured by six different cameras attached to the same car to best inference the surroundings. More specifically, our tasks of generating BEV maps can be broken down into road map prediction and object detection. To this end, we have implemented several state-of-the-art frameworks our way to fit our data, and at the same time combing the technique of pretraining on unlabelled dataset in order to better map image-based features into a top down view of the surrounding area. As demonstrated by our experiments, the high-resolution net (HRNet) and You Only Look Once (YOLO) achieves the best performance for road map prediction and object detection respectively, which achieves 0.75 threat score for road map and 0.002 for bounding box predictions. Link to our Github Repository.

1. Introduction

Detecting objects and predicting road map layout are essential components in Autonomous Driving systems.

Autonomous vehicles requires a rich and detailed representation of their environment which captures geometry and layout of the static world as well as the pose and dimensions of other dynamic elements. These representations often provide the foundation information for all decision making, including path planning, collision avoidance and navigation. Instead of trying to capture the full three dimensional world in its entirety, one plausible solution is to represent the world using a birds-eye view (BEV) map, which provide a compact way to capture the spatial configuration of the scene. Such maps are convenient in that they are simple to visualise and process.

To date the object detection literature for autonomous vehicles has been dominated by approaches which make use of rich LiDAR point clouds, while the performance of image-only methods, which lack the absolute depth information of LiDAR, lags significantly behind. Given the high cost of LiDAR units, the sparsity at long ranges, image-based inference is still of interests and importance in this area. We divide the Bird-view map generation task into road map prediction and object detection(bounding box prediction), and we solve them separately to achieve better performance and take the advantage of different deep learning network architectures.

2. Litearture Review

We conducted literature review in related fields in order to best approach this problem.

2.1. Pretrain

Nowadays, the state-of-the-art computer vision pipelines usually involves two tasks, a pre-text task and a downstream task. The downstream task can be tasks such as classification or detection, with insufficient annotated data samples. The pre-text task is the self-supervised learning task solved to learn the representations, with the aim of using the learned representations, or model weights obtained in the process for downstream task. To learn image representations, many use the idea of exploiting some geometric transformation to get labels, such as Jigsaw Puzzle, Auto-Encoding Transformation. However, the image representations learned can overfit to transformation and not generalize well on downstream tasks. The representations will be covariant with the transformation.

Misra et al. (Chaudhary, 2020) from Facebook AI Research proposed a new method PIRL for learning image representations to overcome this covariance issue. Pretext-Invariant Representation Learning, by its name, the representation is invariant to the pretext task and therefore realizes a better generalization.

2.2. Road Layout Prediction

Schulter et al.(Schulter et al., 2018) proposed one of the first approaches to estimate an occlusion-reasoned bird's eye view road layout from a single color image. They use monocular depth estimation and semantic segmentation to bootstrap a Convolutional Neural Network that predicts the road map layout.

High-resolution representations is also proposed for position-sensitive vision problems including semantic segmentation (Wang et al., 2020). Existing state-of-the-art frameworks first encode the input image as a low-resolution representation through a subnetwork that is formed by connecting high-to-low resolution convolutions such as ResNet and VGGNet, and then recover the high-resolution representation from the encoded low-resolution representation. Instead HRNet, maintains high-resolution representations through the whole process. It extracts more useful information about the image features to build a multi-level representation and has shown top results in detecting objects. This is also the approach which we have achieved the best results in the experiment.

2.3. Object Detection

Detecting bounding boxes in images is a widely studied problem and recent approaches are able to achieve promising results. Existing methods may broadly be divided into two main categories: single stage detectors such as YOLO (Redmon & Farhadi, 2018), SSD (Liu et al., 2016) and RetinaNet (Lin et al., 2017b) which predict object bounding boxes directly and two-stage detectors such as Faster RCNN (Ren et al., 2015) and FPN (Lin et al., 2017a) which add an intermediate region proposal stage.

3. Preprocessing

We are given six images in every sample and each image is of dimension [3, 256, 306]. For the binary road map prediction task, the dataset is already in a usable format, no further preprocessing has been applied. We simply stack the six images together, make it of dimension [batch size, 3*6, 256, 306] and then feed it to our model. For the target, we just used the given 800*800 binary road image format.

For the bounding boxes predictions, we let each of the image go through an encoder. The encoder is basically two convolution layers, where for each layer it go through Relu and max pooling to extract the most significant features out of the image. Then we concatenate the six image vector together as the input for our model. For the target, we transform the given [N, 2, 4] format to [S, S, B*(5+C)] format which is required by YOLO. We also normalized the coordinates by dividing them by the map size of 800.

4. Methodology

4.1. Pretrain

Before directly training our model on the labeled dataset, we can make good use of numerous unlabeled data for pretraining. After that, we are able to load the pretrained model to conduct a better feature extraction task and then improve the performance of downstream tasks.

4.1.1. AUTOENCODER

The first method we have tried for pretraining is autoencoder. An autoencoder is, by definition, a technique to encode something automatically. By using a neural network, the autoencoder is able to learn how to decompose data into fairly small bits of data, and then using that representation, reconstruct the original data as closely as it can to the original (Abdelaal, 2019). Moreover, the structure of the autoencoder consists of two components, encoder and decoder. In our case, both encoder and decoder have two convolutional layers with a kernel size of 3. Since our input images are RGB format, we choose the input channel of 3 in the encoder and output channel of 3 in the decoder. In addition, we use the Relu as the activation function and mean square error for computing the loss in our neural network. Thus, our autoencoder can take each image as input and produce a reconstructed image as output. As we have more than 80,000 images in the unlabeled dataset, we have trained our autoencoder for 100 epoch, allowing it to learn more about feature representations of the input image.

4.1.2. PIRL

We have also implemented PIRL: Pretext-Invariant Representation Learning. PIRL proposes a method to tackle the problem of representations being covariant with transformation. It proposes a problem formulation such that representations generated for both the original image and transformed image are similar. In this case, transformation is borrowed from the jigsaw task. We use ResNet-50 as the base ConvNet encoder with 2048-dimensional representation, and then the representations obtained from encoder are passed through a single linear layer to project the representation from 2048 dimension to 128 dimension. After the PIRL model is trained, only the encoder part is used to extract representations and involves the fine tuning process.

4.2. Roadmap Layout

4.2.1. HRNET

We often use the stride and pooling to continuously reduce the size of Feature maps, and then pass it to the classifier or regressor. However; after continuous shrinking, high-resolution information is also continuously lost. Therefore,

the high-resolution net (HRNet) is a framework designed to retain the high-resolution features. Unlike the traditional Convolutional neural network that only has one type of representation extracted from the input images, the modified HRNet is augmented by aggregating the (upsampled) representations from all the parallel convolutions rather than only the representation from the high-resolution convolution(Ke Sun & Wang, 2019), allowing the combined neural net to capture more information from different levels of representations. Moreover, the previous research revealed that the HRNet outperformed other neural networks on semantic segmentation problem, which is a classification task to label every individual pixel. In our case, we can only build a binary classifier to predict the label of each pixel and then concatenate all the predictions to finally produce a road map layout. In this experiment, we choose binary cross-entropy as our loss function. Also, we use the SGD optimizer with an initial learning rate of 0.0001. In addition, the momentum is set as 0.9 and the decay weight is 0.0001. After that, we have trained our HRNet on the training labeled dataset for 10 epochs to get the result.

4.3. Bounding Box Prediction

4.3.1. ResNet

Our first approach is ResNet framework. It is a classic neural network used as a backbone for many computer vision tasks, and it allowed us to train extremely deep neural networks with more than 150 layers successfully. ResNet first introduced the concept of skip connection, and it is applied before the RELU activation to achieve best results. We have tested with RestNet-18 and ResNet-50, and the latter one slightly wins between these two. ResNet-50 model consists of 5 stages each with a convolution and Identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers, it has over 23 million trainable parameters.

Our implementation of ResNet is combined with the PIRL pretrained encoder. For every batch,six images correspond to the same sample are concatenated on channel dimension after going through the same Convolutional layer, and then feed into our pretrained encoder(ResNet-50 is used as ConvNet encoder), then after another ConvNet, upsampling and regressor, we have the final output of size [batch_size, 2, 4]. In the Resnet implementation, we have constructed anchor boxes on our own, the shape and scales are determined by exploring the bounding boxes given by the labeled dataset. Our idea is to constructed a map with all predefined anchor boxes, transform ground truth bounding boxes(gt_boxes) coordinates to corresponding offsets between qualified anchor boxes and gt_boxes, and then use offsets as new targets. It achieved good results but we found it not plausible because it required the information of target information to filter anchor boxes with high quality and it to some extent is information leakage. We have also implemented the framework without giving the target information, but the effects of uncertainty of anchor box quality is hard to solve.

4.3.2. RetinaNet

We have also implemented RetinaNet, use anchor boxes linked to the network instead of predefined ones in the ResNet implementation. The result is not ideal because the anchor boxes are on the original images, as well as the offsets. Without further modification, it may work well for a pure object detection problem, but in our case, the model also need to learn how to map the location from original graph to a BEV map. It is hard for the original RetinaNet to learn the mapping from real world images to BEV maps, the IOU score between predicted bounding boxes and anchors are much smaller than expected since they are not actually in the same space.

4.3.3. Yolo

Instead of building a complicated classification model to detect objects, You Only Look Once (YOLO) frames object detection as a regression problem to spatially separate bounding boxes and associated class probabilities. Our system divides the input image into an S * S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object (Joseph Redmon, 2016). And then, it will assign N bounding boxes for each grid cell since some objects would overlap in the image. Therefore, each grid cell in the image now has N labels. At the same time, it will also produce the probability for every bounding box assignment. If no object exists in a grid cell, the probability should be zero. After that, the grid cells that have the same bounding box label are combined together to form a complete bounding box of that object. For the architecture of YOLO, it consists of 24 Convolutional layers and 2 fully connected layers. Finally, it outputs an S * S * (N * 5 + C) tensor for object detection, where S is the number of grids, N is the number of bounding boxes assigned to each grid cell, and C is the number of classes. For our experiment, we set S = 7, N = 2, and C = 10. Also, we choose to use the SGD optimizer with the initial learning weight of 0.0001, the momentum of 0.9, and the decay weight of 0.0001. After setting hyperparameters, we have trained our model with the initial weights provided by the autoencoder for 30 epochs.

5. Conclusion

In our project, we have applied the autoencoder and PIRL for pretraining, HRNet for roadmap layout predictions, and ResNet, RetinaNet, and YOLO for bounding box predictions. As a result, take threat score as the measurement in our experiments, the HRNet together with autoencoder

pretraining achieves the best roadmap layout, and at the same time YOLO with autoencoder pretraining has achieved the best performance for bounding box prediction. The visualized predicted road map and bounding boxes paired with their corresponding ground truth are presented in 1 and 2. We can see that our model has an overall good performance on road map predictions, especially in predicting horizontal main roads. However, it does not manage to predict the roads with other orientations precisely. For bounding box predictions, our model successfully managed to detect the cars parked by the roads but performed poorly on detecting objects in the middle. Moreover, our predicted bounding boxes are often smaller than the real boxes. For future improvement, more preprocessing methods should be developed and applied before feeding into models. Instead of stacking six images as an input, we could transform those 2D images into a 3D birds-eye-view. After that, we only need to make the projection to get the road map layout and bounding boxes predictions.

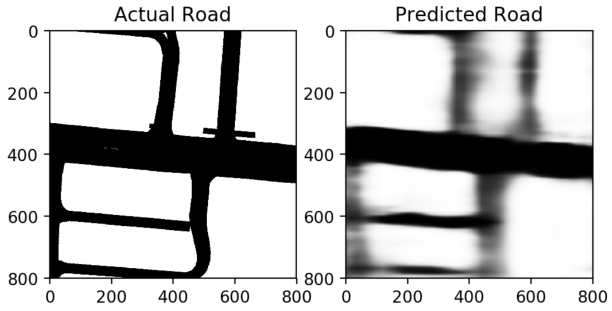

Figure 1. Roadmap Predictions

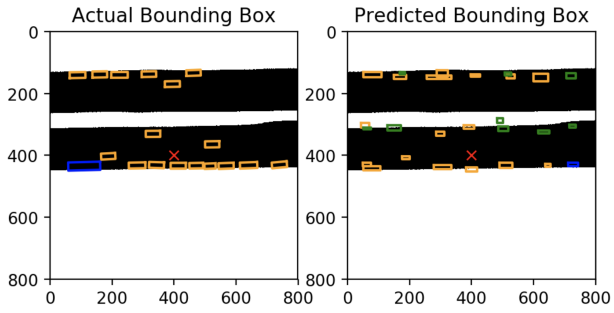

Figure 2. Bounding Box Predictions

References

Abdelaal, A. Autoencoders for image reconstruction in python, 2019. https://stackabuse.com/autoencoders-for-image-reconstruction\-in-python-and-keras/.

Chaudhary, A. The illustrated pirl: Pretext-invariant representation learning, 2020. https://amitness.com/2020/03/illustrated-pirl.

Joseph Redmon, Santosh Divvala, R. G. A. F. You only look once: Unified, real-time object detection. *ArXiv*, abs/1506.02640, 2016.

Ke Sun, Yang Zhao, B. J. T. C. B. X. D. L. Y. M. X. W. W. L. and Wang, J. High-resolution representations for labeling pixels and regions. *ArXiv*, abs/1904.04514, 2019.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Lin, T.-Y., Dollár, P., Girshick, R. B., He, K., Hariharan, B., and Belongie, S. J. Feature pyramid networks for object detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017a.

Lin, T.-Y., Goyal, P., Girshick, R. B., He, K., and Dollár, P. Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, 2017b.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C.-Y., and Berg, A. C. Ssd: Single shot multibox detector. In *ECCV*, 2016.

Redmon, J. and Farhadi, A. Yolov3: An incremental improvement. *ArXiv*, abs/1804.02767, 2018.

Ren, S., He, K., Girshick, R. B., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015.

Schulter, S., Zhai, M., Jacobs, N., and Chandraker, M. Learning to look around objects for top-view representations of outdoor scenes. *CoRR*, abs/1803.10870, 2018. URL http://arxiv.org/abs/1803.10870.

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., and Xiao, B. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020.