Autodetection: An End-to-end Autonomous Driving Detection System

Jiuhong Xiao [1] **Xinmeng Li** [1] **Junrong Zha** [1]

Abstract

Accurate detection of lane and objects on the road is a task of the great importance of intelligent vehicles. However, the error of top-view mapping and multi-viewpoint image stitching limits prediction performance. In this paper, we propose a framework called Autodetection using images captured by six different cameras attached to the same car to detect top-view roadmap and nearby objects. We further introduce rearrange module and Feature Pyramid Network (FPN) to improve accuracy by fusing feature maps with different resolutions. Variational Autoencoder (VAE) is also used to pretrain our model on the unlabeled dataset and make robust prediction. The experiment shows that these modules effectively improve the performance of our framework.

1. Introduction

Accurate detection of lanes and objects on the road is of great importance for self-driving cars since it can provide valuable information for planning the vehicle trajectory and controlling its driving behavior(Yu et al., 2020). As many other computer vision-based tasks, lanes and objects detection accuracy has been significantly improved after introducing convolutional neural networks (CNNs) (LeCun et al., 2015) as the state-of-the-art technology (Pizzati et al., 2019). On the other hand, cameras make it possible to explore different views on the road surface. Rely on visual data, CNNs can process sensorial data and infer high-level information relative to roadmap and objects (LeCun et al., 2015).

To achieve high performance on road and object detection, combined with CNNs and visual data, scientists have come out various methods. In (Liu et al., 2017), they proposed a segmentation network using the ResNet with Pyramid Pooling. In (Tian et al., 2018), they came out a modified version of Faster R-CNN to identify road patches that belong to lane markings which are then joined to obtain a complete representation of the marking. In (Neven et al., 2018), CNNs with fully connected layers are used to obtain a pixelwise representation of lane boundaries. Popular object detection methods like YOLO v3 (Redmon et al., 2016) are developed to identify the objects through sensorial data.

In this work, by combining information from six different cameras to do top-view mapping, we adopted the Encoder-Decoder architecture with convolutional layers and fully-connected layers to detect roadmap and vehicles. Our framework is a roadmap-detection model, using YOLO v3 layers (Redmon et al., 2016) for object detection and pretrained EfficientNet (Chen et al., 2018) for feature extraction. Our innovations are using rearrange module to fuse the feature maps to do top-view mapping and using Feature Pyramid Network (Lin et al., 2017) to integrate different scales of features, which have significantly improved the performance.

2. Methods

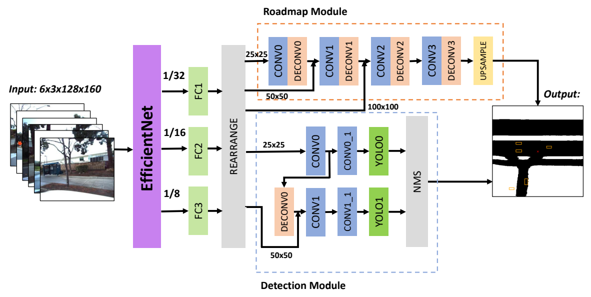


Figure 1. Model Framework

2.1. Overview

Our framework is showed in Fig. 1. We follow an Encoder-Decoder architecture. For encoder, we used Google's EfficientNet (Chen et al., 2018) to extract features. This encoder is pretrained by using VAE (Kingma & Welling, 2013). Three groups of extracted features, which are 1/8, 1/16, 1/32 of the original image size representing objects from small to large, are compressed by FC 1-3 layers and fed into the rearrange module (Fig. 2) for top-view mapping. Then we

[1]New York University, New York, the USA. Correspondence to: Jiuhong Xiao <jx1190@nyu.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

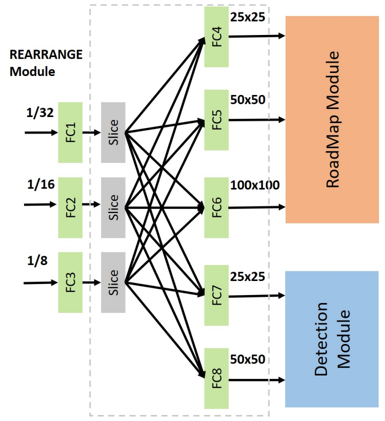


Figure 2. Rearrange Module

incorporate roadmap module and object detection module using the top-view maps to predict the roadmap and objects respectively. The details of each module will be provided in sections below.

2.2. Pretrain

2.2.1. Data Preprocessing

In order to increase the diversity of training samples and have better generalization performance, we apply various transformations on the unlabeled dataset. We change the brightness, contrast, horizontally flip the image, rotate the image, and crop the image with an aspect ratio. The transformation parameters above are randomly selected within appropriate ranges. Due to the computation power limit, we first pad the image with dimension (256,306) to the closet dimension (256,320) that is the multiple of 32, and then resize it to half (128,160).

2.2.2. Train with Backbone Model

For the requirement of autonomous driving, an ideal backbone model should achieve a good efficiency. Compared with other widely used backbone neural nets such as ResNet and MobileNet (Howard et al., 2017), EfficientNet (Chen et al., 2018) reduces the number of parameters and flops by an order of magnitude while still achieves a higher accuracy on the ImageNet.

We initially use AutoEncoder (AE) with a typical architecture, and then switch to VAE to avoid overfitting. We alternate the conv layer and deconv layers in the decoder to avoid the information loss of upsampling by detecting localized features in these layers. We set the loss function of VAE as,

$$\mathcal{L}(y,\hat{y}) = \mathcal{L}_{BCE}(y,\hat{y}) + \frac{1}{2}(\hat{\sigma}^2 - 2\log(\hat{\sigma}) - 1 + \hat{\mu}^2) \quad (1)$$

where y and $\hat{y}$ are ground truth values and predicted values respectively. $\mathcal{L}_{BCE}$ is to calculate binary cross entropy loss of y and $\hat{y}$. $\hat{\sigma}^2$ and $\hat{\mu}$ are the outputs of the encoder.

2.2.3. Fine Tune Model

We initially freeze the whole pretrained backbone (Section 3.1) to speed up the computation and reduce overfitting. However, the data similarity between the labeled training set and the unlabeled images could be low, we need a more flexible structure to adapt to the labeled-dataset-specific characteristics. Therefore, we freeze the first seven layers of the pretrain model and retrain the subsequent layers with a small learning rate, which keeps most of the pretrain weights intact. Therefore, both the universal and the labeled-dataset-specific features are captured. Through this way, we find a balance point between optimization and generalization performance.

2.3. Roadmap Task

In roadmap module, which acts as one decoder, we combined several ConvNet-DeConvNet layers to output the roadmap. In bothmodel (FPN) (Section 3.1), we extract the high resolution features with the size of 25x25, 50x50 and 100x100 from the rearrange module (Fig. 2). Then we concatenate these features with the output of each ConvNet-DeConvNet layer. The loss function is cross entropy loss,

$$\mathcal{L}(y,\hat{y}) = -\frac{1}{2}\sum_{i=1}^{2} y_i \log(\hat{y}_i) \quad (2)$$

In bothmodel (LSTM) (Section 3.1), we also implement LSTM which is used to integrate two frames' information to predict current frame. However, it comes out that the model either fails to converge or overfits. This depends on the setting of step size. We can hardly find the balance point.

2.4. Object Detection

In the object detection module, we first use the K-means method to select proper anchor sizes for detecting objects with different sizes. Five anchor sizes are selected. The top-view maps with the size of 50x50 and 25x25 are fed to detect larger objects and reduce computation cost. YOLO v3 has both great results and high speed (Redmon et al., 2016). Compared with other object detection methods, YOLO v3 is orders of magnitude faster (30 frames per second). Besides, the high resolution features concatenated with the previous layers help preserve the fine-grained features which address the issue of detecting small objects. The loss function is set as YOLO v3 did.

3. Experiment and Results

3.1. Experiment

To prove the effectiveness of our framework, we did some ablation experiments using different models. We named these models by the modules they contain (Section 3.2). Roadmap model only contains EfficientNet backbone and roadmap module. Bothmodel contains roadmap module and detection module. Bothmodel (LSTM) replaces parts of FC layers with LSTM layers. Bothmodel (FPN) adds FPN structure and rearrange module. These two models are based on bothmodel.

For pretrain setting, roadmap model is trained from scratch. Bothmodel and bothmodel (LSTM) freeze the whole pretrained backbone network. Bothmodel (FPN) freezes the first 7 layers of backbone network and finetunes the rest layers with initial learning rate = 0.001.

For hyperparameters, the pretrain model is trained with initial learning rate = 0.002 and batch size = 8. For roadmap-detection model, batch size is 8 and initial learning rate is 0.01. Both models use Adam optimizers. A step decay scheduler is applied to the learning rates of roadmap-detection model and pretrain model with decay cycle = 50 epochs and gamma = 0.25. The step size of LSTM is 2.

3.2. Results

For the roadmap-detection model, we compare models by two metrics: mean average threat score (mATS) and mean average precision (mAP). Firstly, we decide a threshold for IoU. If IoU is larger than this threshold, the prediction is correct. Thus, an average precision can be calculated for each threshold. We start with threshold of 0.5, then take step of 0.05 to increase the threshold until it reaches to 0.95 to get several average precisions. Finally, the mean of all these average precisions is mAP.

To calculate mATS for object detection, the step of thresholds is reset to 0.1. We assign a weight for each TS, which is just the precision for each threshold. The ATS is calculated as: $\text{ATS} = \frac{\sum(weight*\text{TS})}{\sum(weight)}$, and $weight = \frac{1.0}{\text{threshold}}$. Finally, we average the ATS of all images to get mATS.

To calculate mATS for roadmap prediction, we calculate the IoU of the ground truth and predicted roadmaps as ATS and average the ATS of all images to get mATS.

Table 1 and Table 2 show the performance results of all of our models. Obviously, the bothmodel with FPN has the highest mATS and mAP on the object detection tasks, which indicates that it performs best on identifying bounding boxes with all of the IoU thresholds. We observe that adding detection module or LSTM has a negative impact on the Roadmap Threat Score. Theoretically, Multi-Task Learning (MTL) by hard parameter sharing reduces overfitting. With MTL, the model learns a more generalized representation, so the performance is likely to be improved. However, since the detection and roadmap module share a common limited feature map, the capacity is not enough for us to train both modules together to achieve a higher accuracy.

Model with LSTM fails to converge on both tasks. This is out of our expectation, since LSTM is supposed to capture the temporal information and therefore improves the object detection accuracy. Two possible reasons for this failure are that the labeled dataset may be too small and the location LSTM layers inject may be inappropriate. Model with FPN has relatively high scores on both roadmap and bounding box.

Consequently, the bothmodel (FPN) is our optimal model. On Figure 3, we observe that the small branches and distant areas are difficult to be detected for the roadmap task. The objects on the other side of the lane are hard to be captured for the object detection task.

For the pretrain model, average test loss of VAE converges to 0.4237 after 200 epochs, and that of AE fluctuates in the range from 0.42 to 0.425. This phenomena implies that VAE generalizes better on the test set. Therefore, we build our pretrain model based on VAE.

Table 1. Roadmap Performance Results

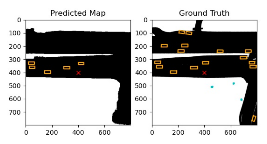
Methods	RmATS (train/val)	RmAP (train/val)	RmATS (test)
Roadmap Model	0.9523	0.9927	0.7821
bothmodel	0.8796	0.8752	0.7193
bothmodel-(LSTM)	0.6876	0.7863	-
bothmodel-(FPN)	0.9269	0.9524	0.7930

RmATS denotes Roadmap mean Average Threat Score at IoU=0.5:0.1:0.9. RmAP denotes Roadmap mAP. mAP denotes the mean Average Precision at IoU=0.50:0.05:0.95. The VAE pretrained model is integrated into bothmodel but not the roadmap model.

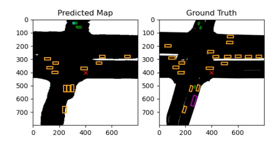
Table 2. Bounding Box Performance Results

Methods	BBmATS (train/val)	BBmAP (train/val)	BBATS (test)
bothmodel	0.1051	0.0915	0.0004
bothmodel-(LSTM)	0.0000	0.0001	-
bothmodel-(FPN)	0.1832	0.2350	0.030

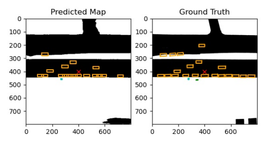
BBmATS denotes Bounding Box mean Average Threat Score. BBmAP denotes Bounding Box mAP.



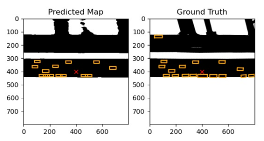
(a)



(b)



(c)



(d)

Figure 3. Compare the Predicted Map and the Ground Truth

4. Discussion

4.1. Challenges

4.1.1. Failed trials

To limit the dimension of outputs of FC layers, we try two methods. First, We have tried to find the affine transformation projecting the input image to the corresponding camera area on the topview by learning Spatial Transformer Networks(STN) in the end to end fashion, but it fails to converge. Second, we have tried to manually define the camera areas on topview by setting the values outside of the camera area to 0, but this results in overfitting.

To capture the temporal information, we try to replace parts of FC layers with LSTM layers. We tried bothModel (LSTM) with step size = 2 and 4. But the first model fails to converge, and the second model overfits.

4.1.2. Limitations

With restricted computation power, we have to shrink the input image, which prevents us from designing and experimenting models with higher complexity. The labeled dataset has a significantly smaller size than the unlabeled dataset, which makes the task result heavily relies on the pretrain model. Otherwise, a model ignoring the pretrain weights is prone to overfit.

4.2. Future Work

We will look for ways to utilize the data about camera intrinsic and action class, to avoid the overfitting occurred in the LSTM, and to tweak the STN. There are overlaps between bounding boxes. Tuning the NMS threshold for IoU might be a way to solve this. The camera intrinsic can be used for the monocular depth estimation. 3d images with the estimated depth as the fourth channel provide more positional information.

References

Chen, P.-R., Lo, S.-Y., Hang, H.-M., Chan, S.-W., and Lin, J.-J. Efficient road lane marking detection with deep learning. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pp. 1–5. IEEE, 2018.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Liu, X., Deng, Z., Lu, H., and Cao, L. Benchmark for road marking detection: Dataset specification and performance baseline. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6. IEEE, 2017.

Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., and Van Gool, L. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE intelligent vehicles symposium (IV)*, pp. 286–291. IEEE, 2018.

Pizzati, F., Allodi, M., Barrera, A., and García, F. Lane detection and classification using cascaded cnns. *arXiv preprint arXiv:1907.01294*, 2019.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

Tian, Y., Gelernter, J., Wang, X., Chen, W., Gao, J., Zhang, Y., and Li, X. Lane marking detection via deep convolutional neural network. *Neurocomputing*, 280:46–55, 2018.

Yu, Z., Ren, X., Huang, Y., Tian, W., and Zhao, J. Detecting lane and road markings at a distance with perspective transformer layers. *arXiv preprint arXiv:2003.08550*, 2020.