MOBIP: A LIGHTWEIGHT MODEL FOR DRIVING PERCEPTION USING MOBILENET

Minghui Ye
School of Mechanical and Electrical Engineering
Guangzhou University
minghuiye@e.gzhu.edu.cn

Jinhua Zhang
School of Mechanical and Electrical Engineering
Guangzhou University
zjhjd@gzhu.edu.cn

ABSTRACT

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Keywords Self-driving · Multi-task learning · Semantic segmentation · Traffic object detection

1 Introduction

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

2 Related Work

3 Methodology

We present a lightweight multi-task learning framework designed for driving perception. As shown in Figure 1, our driving perception model, termed as Mobip, comprises a shared encoder and two decoders. The shared encoder efficiently extracts features, thereby conserving computational resources and facilitating multi-task learning. The decoders consist of two heads: the detect head for traffic object detection, and the segment head for drivable area segmentation and lane detection.

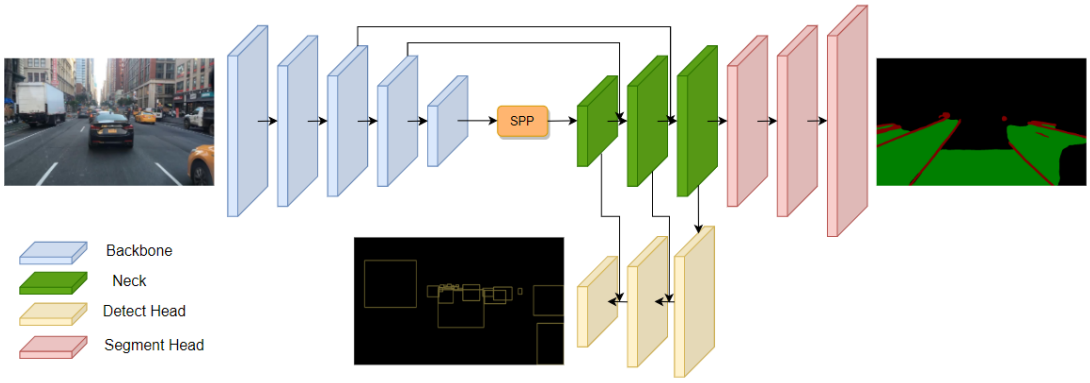


Figure 1: The architecture of Mobip. Mobip contains one shared encoder for feature extraction and two decoders for object detection and semantic segmentation.

3.1 Encoder

The shared encoder comprises two components, namely a backbone network and a neck network.

3.1.1 Backbone

The backbone network serves a critical role in the driving perception model as a feature extractor. Contemporary network architectures often leverage networks that have exhibited high accuracy on the ImageNet dataset for feature extraction purposes. Given the efficient nature of the MobileNetV2 network and its exceptional performance in object detection and semantic segmentation, we specifically select it as the backbone network for our model implementation. Moreover, the utilization of inverted residual as the basic building block enhances our model's efficiency.

3.1.2 Neck

The features extracted by the backbone are fused through the Neck network. The Neck network consists of two modules, including Spatial Pyramid Pooling(SPP) for fusing features of different scales and Feature Pyramid Network(FPN) for fusing features at different semantic layers. Within the FPN module, high-level semantic features are concatenated with low-level features after bilinear interpolation to facilitate aggregation.

3.2 Decoders

The task of driving perception is divided into either a detection task or a segmentation task, subsequently performed by the corresponding task head.

3.2.1 Detect Head

Similar to YOLOv4, our approach employs an anchor-based multi-scale detection scheme. Multiple feature layers within the Feature Pyramid Network (FPN) are passed to the detect head for further feature aggregation. First, the Path Aggregation Network (PAN) is utilized to perform bottom-up feature fusion, enabling better localization feature extraction. The aggregated multi-scale feature from PAN is then used for traffic object detection. Within the multi-scale feature map, every grid is assigned three anchors with varying aspect ratios. The detect head then generates predictions for the offset of position, scaled height and width, as well as the corresponding probabilities and confidences for each class prediction.

3.2.2 Segment Head

Different from previous methods that handle drivable area segmentation and lane detection separately with two segmentation heads, we adopt a single segmentation branch for multi-class segmentation, which effectively reduces computational redundancy and achieves faster inference speed. The segmentation head is connected to the bottom layer of FPN and up-samples the feature maps for three times with bilinear interpolation method. This process restores the output feature map to the size of original image, generating the semantic segmentation result.

Table 1: Computational cost for various multi-task models and inference speed on TESLA V100 GPU

Network	Input Shape	Params	Multi-adds	Speed(fps)
YOLOP	640x640	7.9M	9.3G	40
HybridNets	640x640	12.8M	7.8G	27
Mobip(Ours)	640x640	10.5M	355.6M	56

3.3 Loss Function

In our multi-task learning approach, the two loss specifically for detect head and segment head are calculated with weights and added together as a final loss as shown in Equation (1).

$$\mathcal{L}_{all} = \gamma_1 \mathcal{L}_{det} + \gamma_2 \mathcal{L}_{seg}, \tag{1}$$

where $\mathcal{L}_{det}$ is the detection loss and $\mathcal{L}_{seg}$ is the segment loss. The detection loss $\mathcal{L}_{det}$ is a weighted sum of classification loss, object loss and bounding box loss, as shown in Equation (2).

$$\mathcal{L}_{det} = \alpha_1 \mathcal{L}_{class} + \alpha_2 \mathcal{L}_{obj} + \alpha_3 \mathcal{L}_{box}, \tag{2}$$

where $\mathcal{L}_{class}$ is classification loss while $\mathcal{L}_{obj}$ is the loss for the confidence of one prediction. $\mathcal{L}_{class}$ and $\mathcal{L}_{obj}$ are calculated with focal loss to force the model to learn the hard example. $\mathcal{L}_{box}$ is $CIoU$ loss which measures the distance of overlap rate, aspect ratio and scale similarity between predicted results and ground truth. For the segment loss $\mathcal{L}_{seg}$, we deploy a hybrid loss including focal loss and dice loss, as shown in Equation (3).

$$\mathcal{L} = \mathcal{L}_{Dice} + \beta \mathcal{L}_{Focal}, \tag{3}$$

The dice loss in the segment head can mitigate the data-imbalance problem since there is a multi-class segmentation where the data of lane line is way less than other segment class. The focal loss can help the model to learn the pixels with poor classification. The values of $\gamma_1, \gamma_2, \gamma_3, \alpha_1, \alpha_2, \beta$ are tuned for the balance of the loss.

4 Experiments

4.1 Dataset and Experimental Setting

The BDD100K dataset contains a variety of driving scenes, in which a large number of data are collected from driving recorders, capturing more "long tail" driving scenes in different environments. The dataset includes various annotations, including drivable areas, object detection, attributes, road types, and lane lanes etc, which can support scientific research on a range of driving perception tasks. Therefore, we validated the effectiveness of the proposed driving perception model by comparing it with state-of-the-art methods on the BDD100K dataset. The BDD100K dataset consists of images at 1280×720 resolution, with a total of 100K images, divided into three splits: 70K for training set, 10K for validation set, and 20K for test set. We evaluated our method following official standards in the literature [].

At the training stage, we use the Adam optimizer, with learning rates, β_1 and β_2 set to 1×10^{-2}, 0.937 and 0.999, respectively. Cosine annealing and warm-up are applied to adjust the learning rate. Data augmentation techniques, including mirror, translation, shearing, rotation, photometric distortion, Mosaic and Mixup, is used to boost the performance. Following [], We resize the input image from $1280 \times 720 \times 3$ to $640 \times 384 \times 3$. All modules are implemented using the PyTorch framework [], and all experiments were run on NVIDIA Tesla V100.

4.2 Result

In this section, We compare Mobip to other representative models on all three tasks. First, we compare Mobip with other multi-tasking methods in terms of parameters, number of computations and inference speed in GPU (NVIDIA Tesla V100). We then evaluate the performance of Mobip on three driving perception task, comparing it with multi-task methods and networks that focus on the single task. Finally, the model is deployed to embedded devices(Raspberry Pi 4B) to test inference performance.

4.2.1 Model Parameter and Inference Speed

Table 1 presents the comparison of Mobip, YOLOP and HybridNets in term of parameters, computations and inference speed. By adopting MobileNet as the backbone network and the inverted residual as the building block for the

Table 2: This is a table caption. Tables should be placed in the main text near to the first time they are cited.

Network	Recall(%)	mAP50(%)	Speed(fps)
Faster R-CNN	81.2	64.9	8.8
YOLOv5s	86.8	77.2	82
MultiNet	81.3	60.2	8.6
DLT-Net	89.4	68.4	9.3
YOLOP	89.2	76.5	40
Mobip(Ours)	89.6	75.4	56

Table 3: This is a table caption. Tables should be placed in the main text near to the first time they are cited.

Network	mIoU(%)	Speed(fps)
PSPNet	89.6	11.1
MultiNet	71.6	8.6
DLT-Net	71.3	9.3
YOLOP	91.5	40
Mobip(Ours)	90.0	56

driving perception model, Mobip have a considerable advantage in the amount of calculation (355.6M), achieving the fastest inference speed (56 FPS) in the TESLA V100 GPU. Notably, the performance comparison between Mobip and HybridNets in each driving perception task is not provided in the following passage, as the primary focus of Hybrid-Nets lies in enhancing perception performance, rather than emphasizing lightweight model design and inference speed improvement.

4.2.2 Traffic Object Detection Result

4.2.3 Drivable Area Segmentation Result

4.2.4 Lane Detection Result

4.2.5 Ablation Study

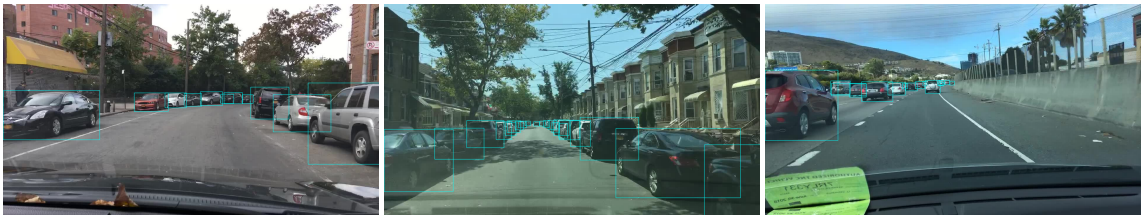
4.2.6 Inference on Embeded Device

5 Conclusion

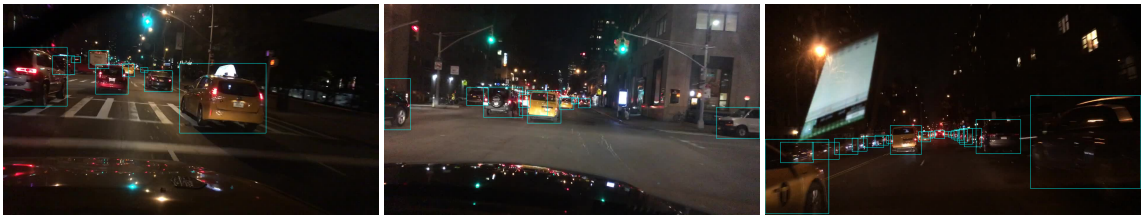
References

Table 4: This is a table caption. Tables should be placed in the main text near to the first time they are cited.

Training Method	Recall(%)	AP	mIoU	Accuracy	IoU	Speed(fps)	Params	Mult-adds
YOLOP	89.2	**76.5**	**91.5**	70.5	26.2	40	7.9M	9.28G
Multi-Class Seg	89.3	76.1	88.9	88.2	27.3	49	7.6M	6.28G
Backbone	87.7	73.2	89.3	88.6	26.4	56	10.5M	355.6M
Mosaic + Mixup	**89.6**	75.4	90.0	**88.9**	**28.5**	**56**	10.5M	**355.6 M**

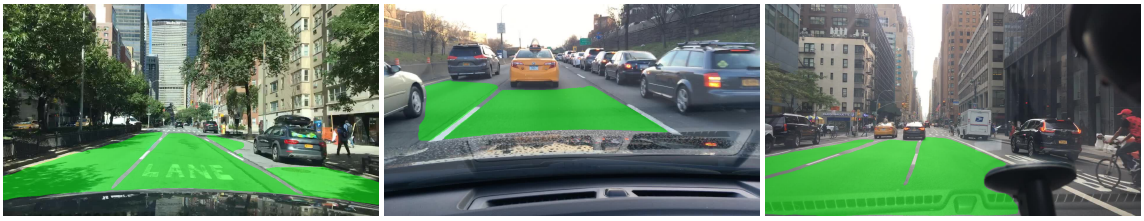


(a) Day-time result

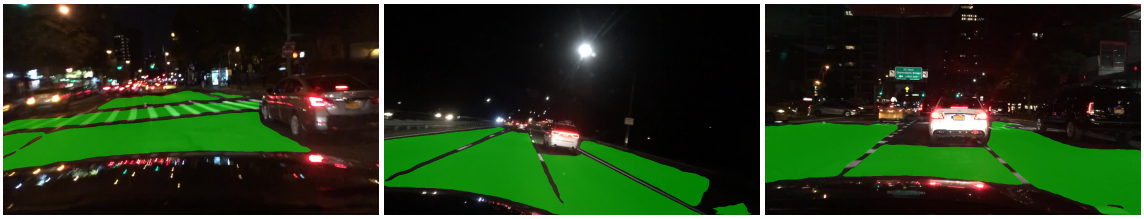


(b) Night-time result

Figure 2: Visualization of traffic object detection results of Mobip. (a) Results in day conditions and (b) Results in night conditions



(a) Day-time result

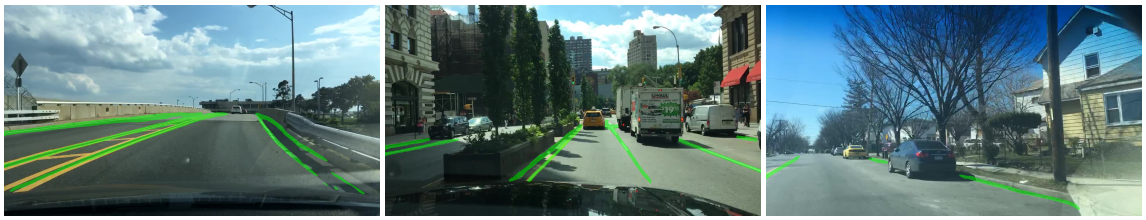


(b) Night-time result

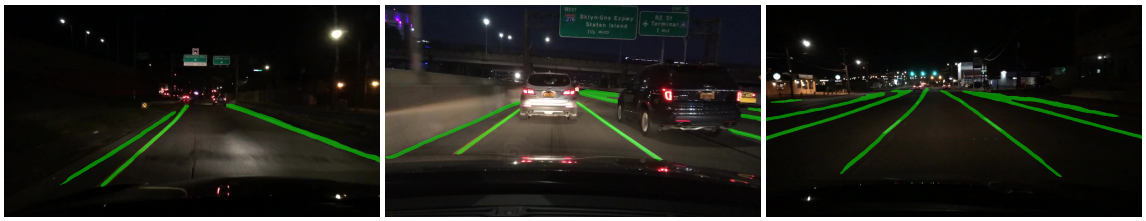
Figure 3: The convergence of the DNN based solver for inversely computing molecular parameters.

Table 5: This is a table caption. Tables should be placed in the main text near to the first time they are cited.

Network	**Accuracy(%)**	**IoU(%)**	**Speed(fps)**
ENet	34.12	14.64	100
SCNN	35.79	15.84	19.8
ENet-SAD	36.56	16.02	50.6
YOLOP	70.5	26.20	40
Mobip(Ours)	**88.9**	**28.5**	**56**



(a) Day-time result



(b) Night-time result

Figure 4: The convergence of the DNN based solver for inversely computing molecular parameters.