MtpNet: Multi-task Panoptic Driving Perception Network

Zheng Li[1], Xiaohui Yuan[2], Bifan Sun[1], Yuting Xia[1], Tingting Jiang[1], Chao Wang[1], Wentao Ma[1], Shuai Yang[1], Siyuan Liu[2], Lichuan Gu[1]

***Abstract*—Panoramic driving systems are crucial for autonomous driving but face challenges in real-time performance and reliability. This paper proposes an end-to-end, multi-tasking MtpNet that reduces latency and enhances detection accuracy. The convolution was upgraded using the Efficient Layer Aggregation Network, and precise multi-task loss functions and more effective training strategies were devised. Our results demonstrate improved performance in small object detection, partial occlusion handling, and drivable area segmentation. The recall of the traffic object detection is 1.3% higher than that of the state-of-the-art model, reaching 94.1%, the mAP^{50} is 6.4% higher, reaching 89.8%, and the mIoU of the drivable area segmentation is 2.7% higher, reaching 95.9%. Additionally, the accuracy of lane detection reached 88.7%. The visual comparison using three datasets TuSimple, CityScapes, and CULane demonstrates that MtpNet has good detection segmentation and strong robustness under various conditions. Codes are available at https://github.com/ErLinErYi/mtpnet**

***Index Terms*—Multi-task network, object detection, segmentation, lane line detection.**

I. Introduction

Panoramic driving awareness systems use sensors such as cameras and lidar to collect data, enabling self-driving vehicles. The system comprises three crucial tasks: object detection provides information about the location and size of other vehicles; lane line detection obtains the direction and shape of each lane line on the road; and drivable area segmentation identifies safe driving areas. The predictive networks help decision-making and control of self-driving vehicles [1] and assist driving via lane keeping [2].

In practice, a balance of speed and accuracy is needed. We propose a network that reduces inference delay that is crucial for autonomous driving with real-time demands. Implementing separate network models for each task in real-time panoramic driving is often infeasible. Running tasks through three distinct network passes may result in out of synchronization due to latency. These tasks are interrelated (see Fig. 1); lane lines represent the boundaries of the drivable area, and obstacles encountered within the driving area must also be considered. In such cases, a multi-task network that shares a backbone for feature extraction achieves information sharing among tasks, leading to improved performance.

Studies have been conducted for multi-tasking in autonomous driving awareness systems [3]. YOLOP incorporates decoder branches for the drivable area segmentation task and lane line detection task, both designed with the same structure. On the other hand, HybridNets use a single decoder branch for both tasks. Additionally, YOLOP and YOLOPv2 adopt feature pyramid networks (FPN) [4] for fusing semantic and spatial features, while HybridNets uses BiFPN [5] to enhance FPN.

Fig. 1. Joint detection of drivable areas, lane lines, and traffic objects.

This paper introduces networks for the drivable area segmentation task and the lane line detection task. We improve Generalized Feature Pyramid Network (GFPN) [6], which employs queen-fusion that introduces a significant number of up-sampling and down-sampling operations, leading to high latency. To address this issue, we extend GFPN by removing certain upsampling operations to achieve a lightweight design and introducing connections based on efficient layer aggregation networks [7]. We integrate techniques such as MetaAconC adaptive activation function [8] for enhanced performance. The main contributions of this paper are as follows:

1) A lightweight, multi-task network that reduces latency and enhances detection. The network jointly learns three tasks and can be trained end-to-end.
2) A loss function and a training strategy to improve the detection of small objects under partial occlusion and inaccurate object segmentation for panoramic driving.

The rest of this paper is organized as follows. Section II reviews the studies of traffic object detection, drivable area segmentation, lane line detection, and multi-tasking networks. In Section III, the structure of the multi-task network model and the loss function of multi-task are introduced. Section IV discusses the comparative experimental results and visualization. Section V, the Ablation Experiment of the proposed model is carried out to evaluate the effectiveness of the proposed network in the joint training of three tasks. Section VI summarizes our findings and presents future work.

II. Related Work

A. Traffic Object Detection

Object detection include two-stage and one-stage approaches. The Region-CNN (R-CNN) family [9]–[11] repre-

[1] School of CS and AI, Anhui Agricultural University, Anhui, China
[2] Department of CSE, University of North Texas, Denton, Texas, USA

sents the two-stage detection algorithms. R-CNN [9] obtains target candidate regions and uses the features in these regions to localize and classify objects. However, it is time-consuming due to overlapping candidate regions, leading to redundant feature extraction. Fast R-CNN [10] improves R-CNN by adding target candidate region boxes to the last layer of the feature map, reducing computational efforts of individual extraction for each candidate region box. Faster R-CNN [11] further enhances the speed of detection frame generation by introducing a Region Proposal Network (RPN) instead of relying on segmentation for candidate frame generation.

The YOLO family is the one-stage detection method. YOLO [12] divides the entire image into grids and feeds them to a CNN, using regression to predict the target. This algorithm achieves real-time detection for the whole image but may suffer from poor localization. SSD [13] improves localization by adding the anchor mechanism, combining region suggestion with regression. YOLOv2 [14] made improvements, including Batch Normalization [15], the usage of a fully convolutional network, and anchor box mechanism. YOLOv3 [16] introduced a multi-scale detection and upgraded the backbone network. Subsequent YOLO algorithms have undergone improvements in network structure, loss functions, and label assignment. GiraffeDet [6] introduces the Generalized Feature Pyramid Network, which enhances feature fusion through a queen-fusion mechanism. This allows the network to handle semantic and spatial information with equal priority in the early stages, making it more effective in object detection tasks. Generally, one-stage algorithms tend to have a faster speed.

B. Drivable Area Segmentation

Semantic segmentation partitions an image into semantic regions. Traditional approaches use image segmentation-based algorithms, often employing clustering, edge detection, and region growing to partition images. In recent years, deep learning has revolutionized semantic segmentation. Full Convolutional Network (FCN) [17] pioneered semantic segmentation by replacing the fully connected layer with a convolutional layer, enabling pixel-level image classification.

The DeepLab models [18] leverages atrous convolution for feature extraction and FCN for segmentation. UNet [19] uses an Encoder-Decoder architecture, where the Encoder employs multilayer pooling to expand the field of perception, and the Decoder uses upsampling to recover the image size. DeepLabv3+ [20] introduces an encoding-decoding architecture, achieving efficient convolution by splitting atrous convolution into two steps. PSPNet [21] uses spatial pyramid pooling for multi-level semantic feature fusion.

In addition to accuracy, inference speed is another factor. To achieve real-time inference, networks such as DFANet [22], and BiSeNetV2 [23] employ lightweight network architecture design and parameter optimization strategies to reduce the number of parameters and computational requirements, enabling real-time, pixel-level semantic segmentation.

C. Lane Line Detection

Lane line detection methods include two types: conventional methods [24], [25] and deep learning-based methods [26], [27]. Conventional methods detect lane lines by extracting features such as edges and textures, whereas deep learning-based methods employ CNN for feature extraction, followed by regression or classification for detecting lane lines.

SCNN [28] proposed a sliced CNN that allows pixel information to flow across ranks and columns in each layer, enabling large targets with spatial correlation but no texture information to be distinguished as different classes. This transforms lane line detection into multi-class segmentation. RESA [26] extends SCNN by adding a transfer of different step sizes between slices. This decouples the temporal dependency between adjacent layers and enhances parallel processing for improved speed. ENet SAD [27] employs semantic segmentation and knowledge distillation to address lane line detection. It introduces a self-attentive distillation that adds SAD modules for shallow features to learn higher-level features, enhancing the overall feature representation. LaneATT [29] applies attention mechanisms and depth-wise separable convolutions in its network architecture, proposing a real-time, one-stage, anchor-based high-performance lane line detection algorithm. Ultra-Fast-Lane-Detection-v2 [30] achieves real-time performance through rapid feature extraction and efficient model design. Both methods satisfy the requirements of real-time detection, with the lightweight version achieving a processing speed of up to 200+ frames per second.

D. Multi-task Approaches

The multitasking networks reduce inference time while enhancing performance by sharing information among tasks. CNN-based multitasking networks enable the convolutional sharing of network structures. Mask-RCNN [31] is a framework built upon Faster-RCNN that adds a fully connected segmentation network, combining instance segmentation and object detection tasks. YOLOP uses a single network to simultaneously handle three tasks. A lightweight CNN serves as the encoder to extract features from images, and these feature maps are sent to three decoders for the respective task. Building upon YOLOP, HybridNets incorporates BiFPN to improve accuracy. It combines the drivable area segmentation and lane line detection into a single task for inference, streamlining the process. YOLOPv2 uses several "bag-of-freebies" techniques to enhance the performance of each task.

Despite significant progress in multitasking methods, there is room for performance improvement. This paper presents a lightweight GFPN method to improve the speed while minimizing potential accuracy loss. Efficient training strategies are proposed and a new loss function is devised. In addition, in the network structure, RFAConv combines the spatial attention with convolution operation to improve the performance of the CNN. In an anchor-based object detector, NWDLoss improves the problem that IOULoss is very sensitive to small target position deviation and can significantly reduce detection performance. Also, the dynamic adaptive activation function MetaAconC can significantly improve network performance.

III. MTPNET

Our MtpNet consists of an encoder and three decoders as shown in Fig. 2. The encoder has a backbone network and a

neck network for multi-level feature extraction. The decoders share the extracted features for reduced computation.

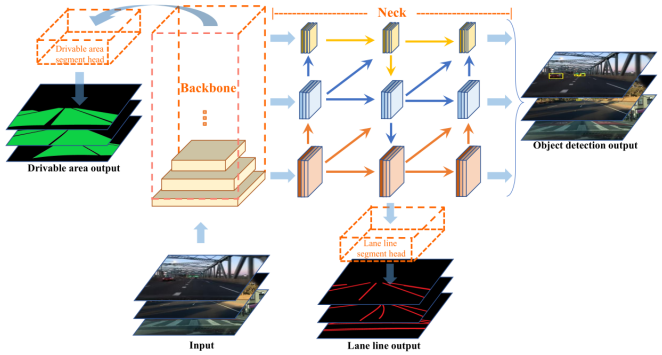

Fig. 2. MtpNet structure. The input is 640×640×3. The output of lane line and drivable area heads is 640×640×2. The output of the detect head is M×85×3, where M (feature map size) is 80, 40, and 20.

A. Backbone

Fig. 3 depicts the backbone network based on ELAN, which employs group convolution for diversified features. This enables the network to learn robust features by controlling the shortest and longest gradient paths. We use Receptive-Field Attention (RFA) to capture global context, allowing the model to understand the image from a larger perspective. RFA enhances spatial features and addresses the inaccurate semantic segmentation of the target region. Despite the negligible increase in computational cost and parameters, RFAConv greatly boosts the network's performance. We introduce the MetaAconC activation function for enhanced network expressiveness. By training multiple sets of learnable parameters for each channel, MetaAconC controls the activation of network neurons based on the adaptive function β. This interaction with spatial information captured by convolution results in a better perceptual field.

The network has four modules, with each module composed of RFAConv, Batch Normalization, and MetaAconC (RBM). The feature map size after the modules becomes 160×160×128. Subsequently, four ELAN modules are applied, and an MP structure is incorporated before the ELAN modules for dimensionality reduction. MP consists of MaxPool and three RBMs (with different kernel sizes and strides shown in parentheses).

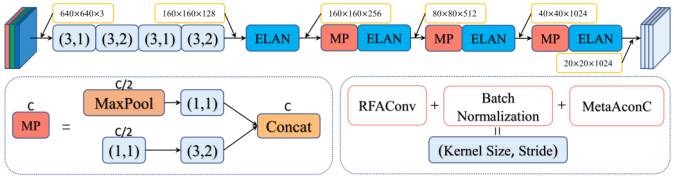

Fig. 3. Backbone network. c is the number of channels and the number near each feature map gives the size.

B. Neck

The FPN aggregates features of different resolutions extracted by the backbone network. While FPN fuses multi-scale features through top-down paths, GFPN [6] serve as generalized neck networks that fully fuse semantic and spatial information. GFPN employs the queen-fusion mechanism to interactively combine features. However, the large number of upsampling and downsampling operations in GFPN introduces high latency, making it unsuitable for real-time requirements. To address this issue, we compare the gains resulting from upsampling and downsampling in GFPN and find that upsampling introduces significantly larger delays with only marginal improvement in accuracy (see Table VIII). Therefore, we remove the additional upsampling operation in GFPN. Fig. 4 depicts the structures of GFPN and LGFPN.

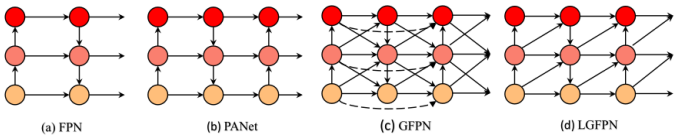

Fig. 4. (a) FPN uses a top-down pathway to fuse features; (b) PANet [32] adds a bottom-up pathway; (c) GFPN has both queen-fusion and skip-layer connection; (d) LGFPN removes the upsampling and skip connections.

We include SPPFCSPC (Fig. 5) in the neck network, which adds MaxPool operations in the convolution. This increases the perceptual field, enabling the algorithm to adapt to images of different resolutions and solve the problem of repetitive feature extractions. SPPFCSPC reduces the number of parameters and obtains a speedup while maintaining the perceptual field. The output of the backbone is the 32-fold downsampled feature map C5. The top-down path and C5, C4, and C3 features are fused to obtain P5, P4, and P3 for subsequent decoder branches.

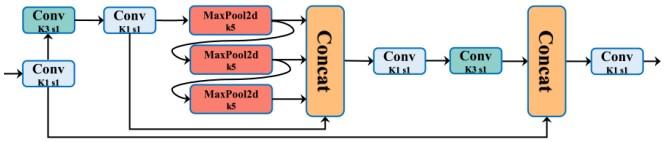

Fig. 5. SPPFCSPC network structure. k: kernel size; s: stride.

C. Decoders

Fig. 6 illustrates the decoders of our MtpNet, each of which accounts for a specific task.

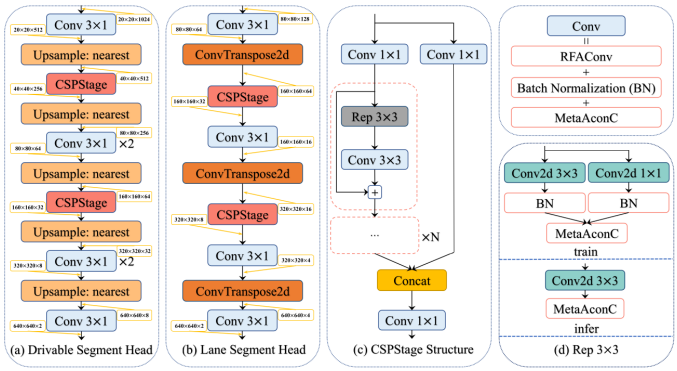

Fig. 6. Decoder structure. (a) Drivable segment head. (b) Lane segment head. (c) CSPStage Structure. (d) Rep 3×3 Structure.

1) Detect Head: The decoder for object detection employs a multi-scale detection [7]. The input image size is (640×640), which is downsampled by 32, 16, and 8 to generate feature maps of sizes 20×20, 40×40, and 80×80. Hence, we have three detection heads of different scales, each of which consists of 3 anchors with different aspect ratios. K-means [33] is used for initializing anchors and a cross-neighborhood grid matching strategy [7] is used, which enables the detection of objects in adjacent grids, thereby improving robustness and reducing false detections. In the matching strategy, simOTA [34] is employed for more accurate prior knowledge.

To balance classification and regression losses, we introduce focal loss [35] and use reparameterized convolution to fuse the convolution and BN layers [7], merging them into one convolution module. By adopting reparameterized convolutions, the inference speed is improved by 1 to 2 frames per second while maintaining performance. This decoder uses the multi-scale feature maps of the GFPN module. The feature maps are adjusted for the number of channels after RepConv [7] and make predictions using 1×1 convolution. By combining the features from the Neck network with a better label assignment strategy and a multi-scale detection scheme, we aim to improve object detection under partial occlusion and, hence, increase the recognition rate.

2) Drivable Area Segment Head: In the drivable area segmentation task, not all passable areas are legal, making the scenarios in the dataset diverse and complex. The network must accurately identify passable areas without including the reverse lane area, which is a common error in existing models. To distinguish between two-way lanes, the network needs to learn specific localization signs during training, such as the direction of travel of the vehicle ahead and lane lines. This requires a strong feature extraction capability, which can be achieved by using the LGFPN structure. The LGFPN can fuse different levels of information at various scales, containing both semantic and spatial location information.

To enhance the accuracy of network segmentation, the network branches that perform the drivable area segmentation task are connected to the last layer of the backbone network. The RFAConv attention mechanism enables the model to understand the image from a global perspective, enhancing spatial feature information. Additionally, the use of the MetaAconC activation function focuses on and captures spatial information, further increasing the perceptual field, and complementing the spatial information captured by convolution. These improvements significantly enhance the accuracy of semantic segmentation of target regions.

The decoder of this task uses the CSPStage [36] twice, optimizing the gradient information repetition and integrating gradient changes into the feature map from beginning to end. This reduces the number of parameters and FLOPS, ensuring both speed and accuracy of inference and reducing the model size. Fig. 6 (a) shows the structure of the decoder for the drivable area segmentation. The feature map size from the end connection of the backbone network is (W/32×H/32×1024). The output feature map undergoes five nearest interpolations upsampling, and the size of the feature map is adjusted to (W×H). The number of channels is adjusted to 2 through six convolutions and two CSPStage modules.

In this task, nearest-neighbor interpolation is used for upsampling for efficiency. The nearest neighbor interpolation maps each pixel position in the target image to the original image and uses the closest original pixel value as the interpolation result. Each pixel in the target image copies the closest pixel value in the original image, thus achieving image enlargement. This method has a small computational load and is efficient. No deconvolution is used because it gives insignificant performance changes in this task.

3) Lane Line Segment Head: The task of lane line detection is challenging since lane lines are often unlabeled in obscured areas, and many are dashed lines. Additionally, issues with lighting and semantic ambiguity complicate the detection. Targets with fragmented and smaller regions require encapsulating more background information in the features since small object detection is highly affected by the background. To enhance the detection, the decoder for this task fuses features of high-level and low-level. The decoder for the lane line detection task is connected to the end of the LGFPN layer of the neck network (see Fig. 6 (b)). The end connections from the LGFPN layer have a feature map size of (W/8×H/8×128). Compared to drivable area segmentation, this task uses three deconvolutions to restore the feature map size of (W×H×2). The deconvolution upsampling introduces more learnable parameters, which convolves the input to generate a larger output. It is a learnable operation and the weights are updated via backpropagation.

D. Loss Function

Our loss function is a weighted sum of three terms: road object detection loss l_{det}, drivable area segmentation loss l_{da-seg}, and lane line detection loss l_{ll-seg}. The road object detection loss l_{det} is the weighted sum of classification loss, target confidence loss, and coordinate loss:

$$l_{det} = \alpha_1 l_c + \alpha_2 l_o + \alpha_3 l_b. \tag{1}$$

l_c and l_{obj} are focal loss [35] to handle the foreground and background imbalance to reduce the loss contribution of simple examples:

$$l_{fl}(p_t) = \begin{cases} -\alpha(1-p_t)^{\gamma} log(p_t) & y = 1 \\ -(1-\alpha)p_t^{\gamma} log(1-p_t) & y = 0 \end{cases}, \tag{2}$$

where α adjusts the weights between positive and negative samples, γ modifies the attention of the focal loss, y is the true label of a sample, and p_t is the prediction probability.

l_b uses normalized Wasserstein distance to improve the detection of small objects:

$$l_{\mathcal{N}WD} = 1 - \exp(-\frac{1}{C}\sqrt{W_2^2(N_p, N_g)}), \tag{3}$$

$$W_2^2(N_p, N_g) = \|[cx_p, cy_p, \frac{\omega_p}{2}, \frac{h_p}{2}], [cx_g, cy_g, \frac{\omega_g}{2}, \frac{h_g}{2}]^T\|_2^2 \tag{4}$$

where $W_2^2(N_p, N_g)$ is the 2nd order Wasserstein distance between N_p and N_g. N_p and N_g are Gaussian distributions modeled according to the bounding box $P = (cx_p, cy_p, w_p, h_p)$ and $G = (cx_g, cy_g, w_g, h_g)$.

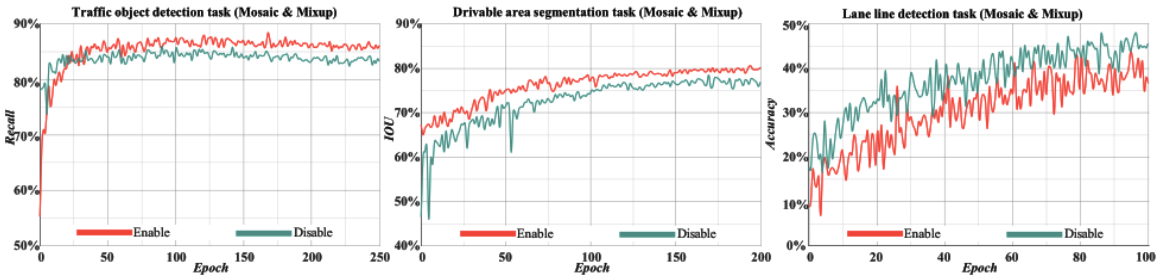

Fig. 7. The performance by enabling/disabling the Mosaic and Mixup across different epochs.

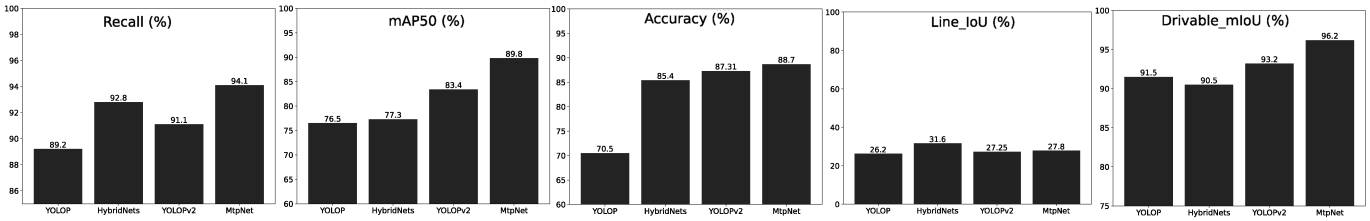

Fig. 8. The performance of our method and the state-of-the-art methods on the BDD100K dataset.

The drivable area segmentation loss l_{da-seg} includes a cross-entropy loss:

$$\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log\left(\sigma\left(p_i\right)\right)+\left(1-y_i\right)\log\left(1-\sigma\left(p_i\right)\right)\right) \quad (5)$$

where $\sigma(x)=\frac{1}{1+e^{-x}}$, N represents the size of the problem, $y_i \in \{0,1\}$ is a binary label, and $p_i \in [0,1]$ is the probability predicted for each sample.

The lane line detection loss l_{ll-seg} uses the weighted sum of Focal Loss and Dice Loss [37] as follows:

$$l_{ll-seg}=\beta_1 l_f+\beta_2 l_d, \quad (6)$$

$$l_d=1-\frac{2\sum_{i=1}^{N} y_i\hat{y}_i}{\sum_{i=1}^{N} y_i+\sum_{i=1}^{N}\hat{y}_i}, \quad (7)$$

where y_i and $\hat{y}_i$ are the label and prediction, respectively, N is the number of pixels. For tasks such as lane line detection, Focal Loss reduces the contribution of simple examples and enhances the focus on correcting misclassified examples. Hence, the model is directed to focus on the misclassified examples. Focal Loss deals with imbalance by assigning greater weights to difficult or misclassified examples, which is appropriate when the lane lines are long and scattered and occupy a small area in the image. Dice Loss alleviates the negative impact of foreground-background imbalance. It weighs more of the foreground region but faces a loss saturation problem.

The loss function of MtpNet is as follows:

$$\psi_1(\alpha_1 l_c+\alpha_2 l_o+\alpha_3 l_b)+\psi_2 l_{BCE}+\psi_3(\beta_1 l_f+\beta_2 l_d), \quad (8)$$

where $\psi_1, \psi_2, \psi_3, \alpha_1, \alpha_2, \beta_1, \beta_2$ are scalar weights.

IV. Experimental Results and Discussion

A. Dataset and Implementation Details

Our experiments use the BDD100K dataset, which contains 100,000 images from the driver's perspective and has three parts: a training set of 70K images, a validation set of 10K images, and a testing set of 20K images. Since the test set has no labels, the validation set is used to evaluate our model. We follow the practices of YOLOP, HybridNets, and YOLOPv2, and merge multiple vehicle categories such as car, truck, bus, train, etc. into a vehicle category, merge direct and alternative into a drivable category, and expand the lane line mask in the training set to 8 pixels while keep the lane lines in the validation set to 2 pixels.

Experiments were performed in a system with an NVIDIA 3090 and torch 1.13.1, cuda 11.7. Adam optimizer and Warm-up and "cosine annealing" strategies were used in model training. The initial learning rate was 0.001, the batch size was 12, the momentum and weight decay parameters were 0.937 and 0.0005, and the number of epochs was 300. Exponential Moving Average (EMA) and group weight decay are used in the network training process, and warm-restart is set with the aim of guiding the model to converge faster and better. The Mosaic and Mixup data enhancement strategies were also used in the training process to obtain stronger robustness, which were enabled or disabled according to the parity of the epoch number during the first 50 epochs. Enabling is performed in odd-numbered epochs, whereas disabling is performed in even-numbered epochs. After 50 epochs, the augmentation strategies were turned off.

We experimented with various training strategies. To minimize training and validation time, we randomly selected 1,000 images for training and 100 images for testing. A comparison is illustrated in Fig. 7. Switching data augmentation strategies is beneficial. Mosaic and Mixup data augmentation strategies [38] demonstrated improved performance in traffic object detection tasks, but a degraded performance in lane line detection. Using Mosaic and Mixup in the early stages enhances the performance of the traffic object while disabling these strategies in the later stages minimizes the negative impact on lane line detection. In addition, both data augmentation

strategies are time-consuming, which increases the training costs. Rotation, scaling, flipping, cropping, and change of brightness and contrast are used to create more cases. Images are resized to 640×640×3.

B. Comparison with the State-of-the-art Methods

Fig. 8 shows comparisons of the traffic object detection using recall and mAP^{50} (mean average precision at 50% IoU). To evaluate the performance of the drivable area segmentation, we use Drivable_mIoU metric. To evaluate the performance of the lane line detection, accuracy and line IoU are used.

1) Traffic Object Detection: In this experiment, the confidence threshold is 0.001 and the NMS threshold is 0.6. Table I presents our results. The MtpNet recall rate improved by 1.3% over the state-of-the-art model HybridNets to 94.1% and mAP^{50} to 89.8%. 6.4% improvement over state-of-the-art model YOLOPv2. This result was achieved by using NWDLoss. IoULoss is sensitive to the size of the objects. Small targets with subtle position deviations will lead to significant IoU degradation, and the sensitivity of IoU to tiny objects makes the network difficult to converge. NWDLoss, on the other hand, is insensitive to objects of different sizes. It is suitable for cases with a large number of small targets.

TABLE I
RESULTS ON TRAFFIC OBJECT DETECTION.

Method	mAP^{50} (%)	Recall (%)
MultiNet	*60.2*	*81.3*
DLT-Net	*68.4*	*89.4*
Faster R-CNN	*55.6*	*77.2*
YOLOV5s	*77.2*	*86.8*
YOLOP	*76.5*	*89.2*
Hybridnets	*77.3*	*92.8*
YOLOPv2	*83.4*	*91.1*
MtpNet(our)	***89.8 (+6.4)***	***94.1 (+1.3)***

Fig. 9 shows the Precision-Recall, Precision-Confidence, Recall-Confidence, and F1-Confidence curves of MtpNet and the multitask methods for object detection. The Precision-Recall depicts curves using different thresholds. It visualizes

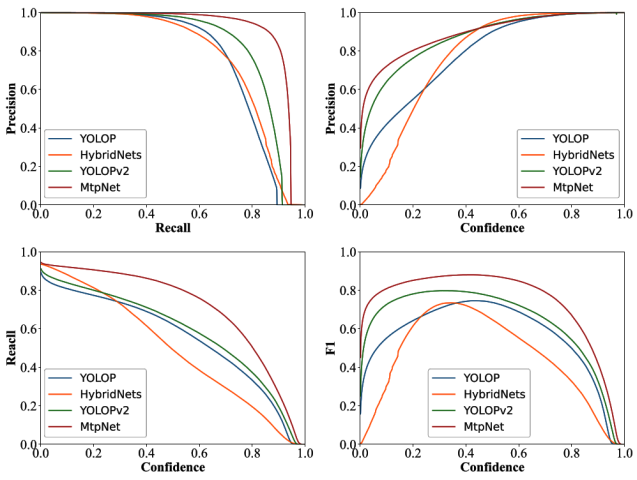

Fig. 9. A comparison of traffic object detection.

the completeness and accuracy of the sample as a whole. The curve enveloped by another has a poor performance. The Precision-Confidence depicts curves using different confidence thresholds. The area under the curve is the total precision. A curve with a larger AUC implies a better performance. The Recall-Confidence depicts curves using different confidence thresholds, the F1-Confidence curve takes both precision and recall into account, and when comparing two curves, if one curve is greater than the other, it is better because at the same confidence, the Recall/F1 value is higher. The results demonstrate improved performance by MtpNet in object detection compared to the state-of-the-art methods.

2) Drivable Area Segmentation: Table II presents the drivable area segmentation results. Among all methods, MtpNet achieves the best results mIoU. It surpasses YOLOPv2 by 2.7%. A separate decoder is designed for this task. The area occupied by the drivable area is relatively large and has clear boundaries. To ensure that the network captures more location and detailed information, the branch connects to the last layer of the backbone network, for which more shallow features are obtained. Using the RFAConv attention and MetaAconC activation function, the attention and capture of spatial information increases the perceptual field, which improves the segmentation in accuracy and speed.

TABLE II
MIOU OF DRIVABLE AREA SEGMENT.

Method	mIoU (%)
MultiNet [39]	*71.6*
DLT-Net [40]	*71.3*
PSPNet	*89.6*
YOLOP	*91.5*
Hybridnets	*90.5*
YOLOPv2	*93.2*
MtpNet	***95.9 (+2.7)***

3) Lane Line Detection: In our experiments, we observe the data processing proposed by YOLOP and preprocess the BDD100K dataset. Following the practice of using the center of the dual lane lines as a new single lane line, a lane line mask map with a width of 8 pixels is drawn for training, while keeping the lane lines in the test set at 2 pixels wide, which is also responsible for the low Lane IoU results for all networks. In this experiment, σ was set to 0.5.

Table III reports the lane line detection results. The model output is 2×640×640, where 640×640 is the image size and 2 is the probability that each pixel is the target area vs the background. YOLOP and YOLOPv2 are used to decide if a pixel is a target or the background. The example heat maps are shown in Fig. 10.

TABLE III
RESULTS OF LANE LINE DETECTION.

Method	Accuracy (%)	IoU (%)
ENet	*34.12*	*14.46*
SCNN	*35.79*	*15.84*
ENet-SAD	*36.56*	*16.02*
YOLOP	*70.50*	*26.20*
Hybridnets	*85.40*	***31.60***
YOLOPv2	***87.31***	*27.25*
MtpNet	*86.0 (-1.31)*	*29.8 (-1.8)*

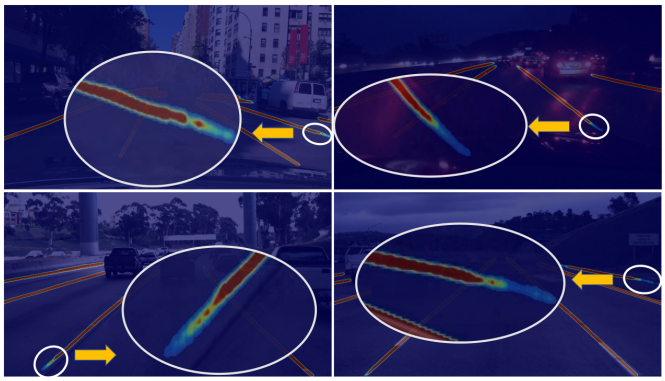

Fig. 10. Heat maps of lane lines detection.

In our method, we employ a threshold σ to distinguish target regions from the background. The task model outputs a probability for each pixel of being background and lane lines. For example, setting a threshold $\sigma \geq 0.5$ means that a pixel is considered part of the background only when the probability being the background is greater than or equal to 0.5. As the threshold increases, the probability of being background decreases. Since the probability of the background is in the range of [0, 1], when setting $\sigma > 1$, all pixels are classified as lane lines; otherwise, all pixels are the background.

Table IV reports the lane accuracy and IoU of MtpNet generated using different σ. Using $\sigma = 0.3$, the accuracy is 85.6%. By increasing σ to 0.98, the accuracy increases to 89.3%. On the other hand, lane IoU drops as σ increases. This is partly because the increase of σ of the background reduces the challenge of deciding the target area, and, hence, more pixels are labeled as the target area. As more pixels of lane lines are marked, the probability of false lane line pixels increases, which leads to a slight decrease in lane IoU. When σ is 0.95, the accuracy improves by 2.7%, and the lane IoU drops by 2%. However, the overall variation is small, which differs by less than 4% in accuracy and 3.5% in IoU.

TABLE IV
LANE ACCURACY AND IOU USING DIFFERENT σ.

σ	Accuracy (%)	IoU (%)
MtpNet ($\sigma = 0.5$)	86.0	29.8
$\sigma \geq 0.30$	85.6 (-0.4)	***30.6 (+0.8)***
$\sigma \geq 0.40$	85.7 (-0.3)	30.2 (+0.4)
$\sigma \geq 0.60$	86.4 (+0.4)	30.0 (+0.2)
$\sigma \geq 0.70$	87.2 (+1.2)	29.3 (-0.5)
$\sigma \geq 0.80$	87.6 (+1.6)	29.0 (-0.8)
$\sigma \geq 0.90$	88.2 (+2.2)	28.3 (-1.5)
$\sigma \geq 0.95$	88.7 (+2.7)	27.8 (-2.0)
$\sigma \geq 0.98$	***89.3 (+3.3)***	27.1 (-2.7)

Increasing σ alleviates lane line discontinuities. Fig. 11 shows the results of lane line detection using different σ. As σ increases, the parts that previously were not recognized as lane lines are recognized as lane lines, connecting the disconnected lines and enhancing the recognition. When σ is greater than or equal to 0.30, pixels are determined as a background when the output probability is greater than 0.30. As shown in the zoomed-in region of the figure, as σ increases, the probability of a pixel being determined as a background becomes smaller,

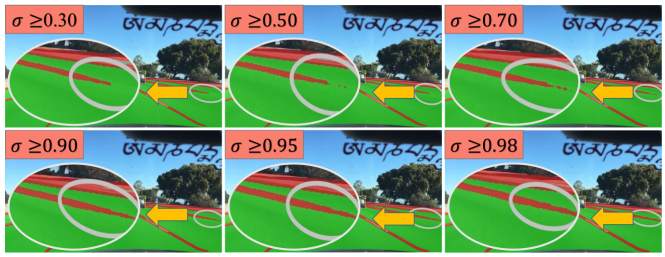

Fig. 11. Results using different σ.

while the probability of it being determined as a lane line becomes larger. More pixels are determined as lane lines, and the number of lane lines in the zoomed-in region increases.

4) Model Parameter And Inference Speed: Table V shows the comparison of model parameters and inference speed. The time includes subsequent processing steps such as Non-Maximum Suppression (NMS). The inference speed was conducted on a single NVIDIA 3090 GPU. Different batch sizes were used. Regardless of the batch size, our method exhibited a competitive speed. The average inference speed is 62 FPS, which satisfies the real-time requirements. In addition, large batches (e.g., 16 and 32) resulted in greater speeds by taking advantage of parallel computing.

TABLE V
MODEL PARAMETERS AND INFERENCE SPEED.

Network	Params	Batch				
		1	4	8	16	32
YOLOP	*7.9 M*	*50.58*	*50.24*	*47.77*	*50.41*	*44.27*
Hybridnets	*12.8 M*	*20.87*	*29.76*	*32.05*	*47.80*	*59.88*
YOLOPv2	*38.9 M*	*60.44*	*66.52*	*79.75*	*79.95*	*75.34*
MtpNet	*50.7 M*	*49.31*	*54.91*	*63.23*	*65.04*	*65.57*

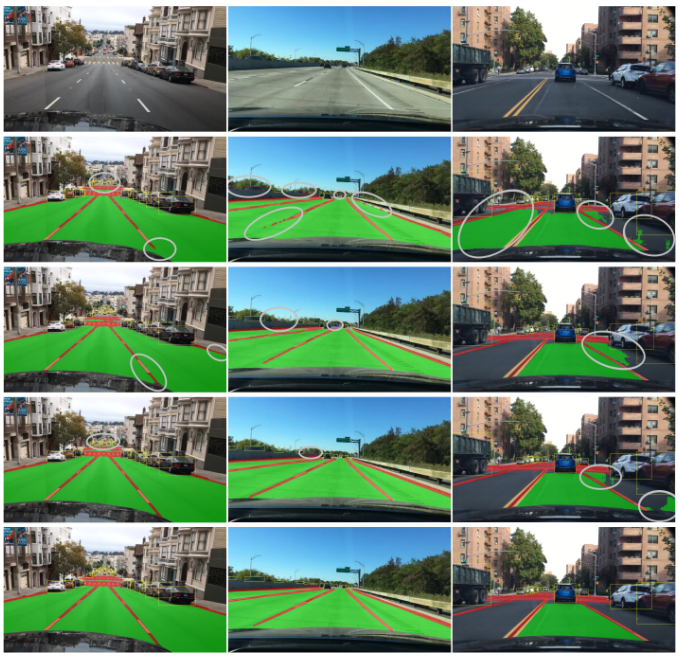

Fig. 12. Daytime results (BDD100K). The ellipses are false detections. The yellow boxes are correct detections, the red boxes are missed, and the blue boxes are false detections.

5) Qualitative Evaluation: Fig. 12 shows the comparison results for daytime images, and the top row shows the input images. The second through fourth rows show the results of YOLOP, HybridNets, and YOLOPv2, respectively. All three methods suffer from similar issues. The left case shows a problem of missing lane line detection and missed small vehicles. The middle case shows a problem of missing partial lane lines, leading to disconnected lines, and the problem of missing the target object with occlusion. The right case shows a problem of missing segmentation and incorrect segmentation of the drivable area. The last row shows the results of MtpNet, which depicts improved performance, especially with small object detection.

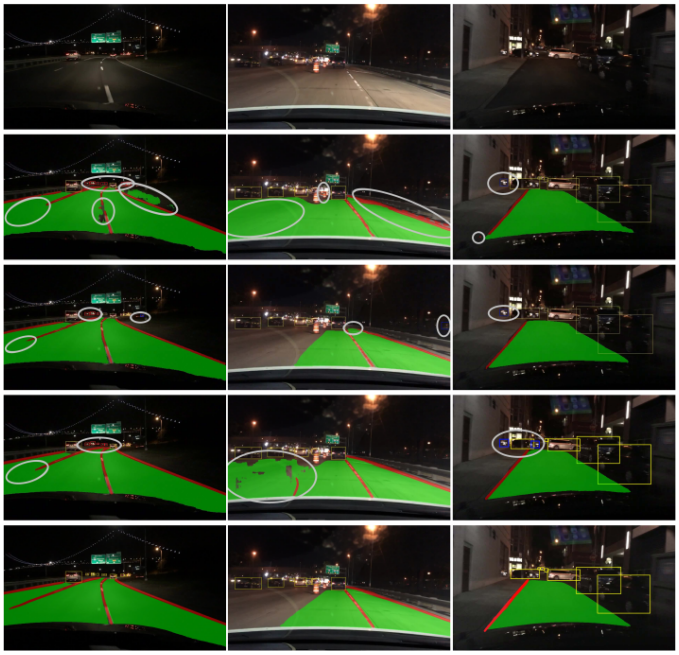

Fig. 13. Night-time results (BDD100K).The ellipses are false detections. The yellow boxes are correct detections, the red boxes are missed, and the blue boxes are false detections.

Fig. 13 depicts the results of nighttime cases. The second through fourth rows show the results of YOLOP, HybridNets, and YOLOPv2, respectively. All three methods suffer from similar issues. In the left case, there are issues with missing lane line detection, failure to detect small vehicles at night, and errors in drivable area segmentation. The middle case shows irregular lane line detection and the division of prohibited driving areas into drivable areas. The cylindrical object is a prohibited traffic sign. The right case is manifested as incorrect vehicle detection, while other models have decreased detection ability in darker scenes and are prone to mistakenly detecting lights as vehicles. Many existing methods fail to correctly detect vehicles in the right case. It is likely to minimize such errors with a larger receptive field. Additional information is integrated for more plausible detection, which is demonstrated with our method. The last row shows the results of MtpNet which exhibits a stronger performance and robustness in nighttime scenarios.

Fig. 14 shows the results of models trained with the BDD100K dataset and tested on the TuSimple, CityScapes, and CULane datasets. The results of YOLOP, HybridNets, and YOLOPv2 show different degrees of incorrect segmentation of the drivable area, inaccurate segmentation edges, failure to detect when the target is partially obscured, as well as poor detection of small targets and partially missing lane line detection. In the results of the TuSimple dataset, signs prohibiting driving are placed in the lanes, whereas YOLOP, HybridNets, and YOLOP classify the area as exercisable. In the CityScapes dataset, the road is two lanes in both directions. YOLOP and HybridNets classify the opposite contraflow lane as a drivable area. YOLOPv2 performs better, but still, some pixels are classified as drivable areas. In the CULane dataset, YOLOP and YOLOPv2 miss the vehicles blocked by railings, and HybridNets detects some vehicles with false positives. In contrast, MtpNet improves both the detection of small vehicles with partial occlusion and the segmentation of drivable areas.

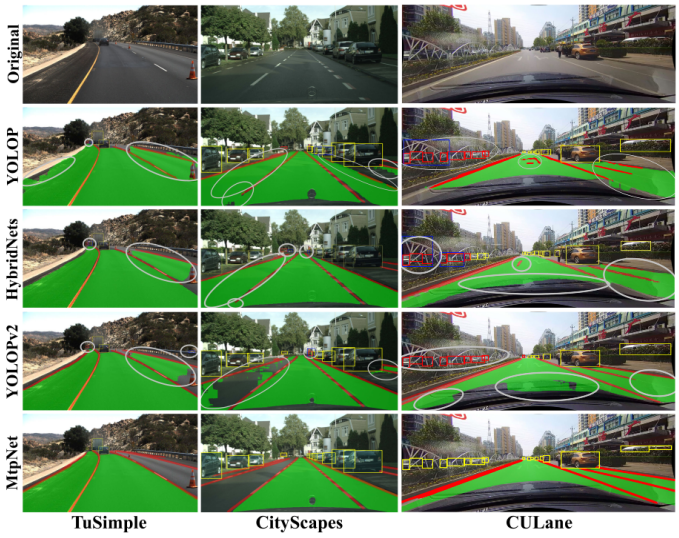

Fig. 14. Results of the TuSimple, Cityscapes, and CULane datasets. The yellow boxes are correct detections, the red boxes are missed, and the blue boxes are false detections.

Fig. 15 shows the results using the CULane night-time scenarios. YOLOP, HybridNets, and YOLOPv2 suffer from inaccurate segmentation of the drivable area and poor lane line detection in difficult scenarios such as backlighting at night to varying degrees. MtpNet improves the accuracy of drivable area segmentation and small object detection in difficult scenes such as backlighting at night, and has stronger robustness.

V. Ablation Study

Multi-task vs Single task: The end-to-end training and separate training of the three tasks are conducted. The speed refers to the inference time, excluding the time of subsequent processing such as NMS. The results are shown in Table VI. Although the inference time for a single task is slightly less than that for multiple tasks, the total time consumed by the three single-task networks is much larger than that of the multi-task network, which indicates that the multi-task network can significantly reduce the inference time for multiple tasks. This is attributed to the shared encoder data among the three tasks,

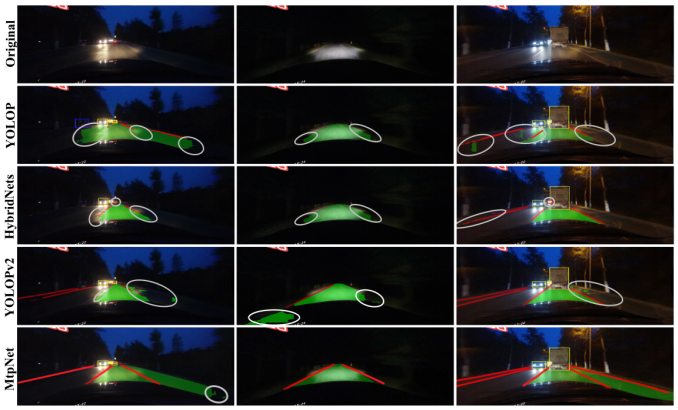

Fig. 15. Night-time results (CULane). The ellipses are false detections. The yellow boxes are correct detections, the red boxes are missed, and the blue boxes are false detections.

TABLE VI
PERFORMANCE OF MULTI-TASK VS SINGLE-TASK LEARNINGS.

Method	Recall	mAP	mIoU	Acc.	lIoU	ms/f	FPS
Detect	*93.9*	*89.5*	-	-	-	*11.4*	*87.7*
Drivable-Seg	-	-	***96.2***	-	-	***10.3***	***97.1***
Lane-Seg	-	-	-	***94.3***	*26.4*	*10.8*	*92.6*
Multi-task	***94.1***	***89.8***	***96.2***	*88.7*	***27.8***	*13.9*	*71.9*

which effectively reduces redundant computations. However, currently, parallelism is not supported in the decoder stage of the three tasks, which results in a slightly longer inference time for multi-tasking compared to the individual task inference time. The individual task performance of the multitask network is comparable to the inference performance of a single task, which shows that the model of the multitask network proposed in this paper is very effective in the joint training of the three tasks mentioned above.

End-to-end vs Step-by-step: Our network is trained end-to-end and step-by-step separately. The results are shown in Table VII, where E, D, A, L, and W represent Training Encoder, Detect Head, Drivable area Segment Head, Lane Line Segment Head, and Whole Network, respectively. The number represents the training rounds. We try a variety of

TABLE VII
RESULTS IN END-TO-END VS STEP-BY-STEP.

Component	Recall	mAP^{50}	mIoU	Acc.	lIoU
EDL_{150}-EAL_{150}	*92.8*	*87.6*	*92.1*	*88.2*	*27.4*
$ED_{50} - W_{150} - AL_{100}$	*88.9*	*78.4*	***96.5***	***90.9***	***28.0***
$ED_{100} - AL_{100} - W_{100}$	*93.1*	*88.2*	*95.4*	*89.2*	*27.6*
$ED_{100} - W_{200}$	*93.6*	*88.9*	*95.2*	*88.7*	*27.2*
$EAL_{150} - W_{150}$	*93.2*	*88.5*	*95.7*	*89.4*	*27.6*
$End - to - end_{300}$	***94.1***	***89.8***	*96.2*	*88.7*	*27.8*

combinations of step-by-step training. We find that the L task converges slowly in the preliminary experiments, while the other D and A tasks converge faster. We choose to train the L task for the whole cycle, while the A and D tasks are trained for only half a cycle. The result was unsatisfactory because the number of training rounds of A and D tasks was too small. After that, we chose to train task D followed by tasks A and L, which resulted in poor results for task D and better results for tasks A and L. This is caused by the late training of only tasks A and L, resulting in network weights that are more biased towards tasks A and L. We train detection and segmentation separately, and then the full task, so that the training results are close to the end-to-end training approach. Finally, the results show that our method performs well using end-to-end training.

Ablation study of GFPN structure: The GFPN uses a queen-fusion mechanism for the interaction and fusion of features. However, the extensive use of upsampling and downsampling operations in GFPN leads to increased latency. Table VIII reports our results of using upsampling and downsampling in GFPN. We compared the inference latency of upsampling and downsampling and their impact on mAP^{50} results. The results reveal that upsampling contributes significantly to the latency compared to downsampling, while the improvement in accuracy from upsampling is relatively minimal.

TABLE VIII
FUSION BRANCH CONNECTIONS IN GFPN.

upsampling	downsampling	$Latency(ms)$	$mAP^{50}(\%)$
✘	✘	*2.78*	*88.6*
✔	✘	*3.35*	*89.0*
✘	✔	***3.04***	***89.8***
✔	✔	*3.77*	*90.2*

Ablation study of MtpNet: We conduct experiments to evaluate network structure, network modules, attention mechanisms, loss functions, activation functions, and data augmentation. The results are shown in Table IX by including each network component and the impact on the results. The '+' represents the addition of a component and the '-' symbol represents the removal of a component. Mosaic and Mixup lead to a decrease in the accuracy of lane detection, while Deconvolution will improve its accuracy. The improved LGFPN and GFPN achieved a frame rate increase of 11 FPS compared to GFPN, with little difference in results.

TABLE IX
ABLATION STUDY OF MTPNET. MAP IS THE mAP^{50}, MIOU IS DRIVABLE MIOU, AND LIOU IS THE LANE IOU.

Method	Recall	mAP	mIoU	Acc.	lIoU	FPS
YOLOP (Baseline)	*89.2*	*76.5*	*91.5*	*70.5*	*26.2*	*48*
New Network	*90.1*	*82.2*	*92.9*	*76.3*	*26.5*	***73***
+Mosaic&Mixup	*90.6*	*83.5*	*93.1*	*74.1*	*24.9*	*73*
+Deconvolution	*90.4*	*83.7*	*93.3*	*77.7*	*26.7*	*70*
+Focal loss	*91.6*	*85.4*	*93.6*	*83.2*	*27.5*	*70*
+Dice loss	*90.9*	*84.5*	*93.4*	*86.3*	*27.1*	*70*
+NWDLoss	*91.8*	*86.8*	*93.3*	*86.1*	*27.2*	*69*
+RFAConv	*92.5*	*87.7*	*94.9*	*87.8*	*27.1*	*67*
+MetaAconC	*93.2*	*88.6*	*95.5*	*88.4*	*27.4*	*66*
+GFPN	*94.1*	***90.2***	*96.1*	***88.8***	*27.6*	*51*
+LGFPN&-GFPN	***94.1***	*89.8*	***96.2***	*88.7*	***27.8***	*62*

VI. CONCLUSION

This paper presents a multitasking MtpNet for object detection, drivable area segmentation, and lane line detection. Our method achieves improved performance with strong robustness. The recall of the traffic object detection is 1.3% higher than that of the state-of-the-art model, reaching 94.1%, the mAP^{50} is 6.4% higher, reaching 89.8%, and the mIoU of the

drivable area segmentation is 2.7% higher, reaching 95.9%. Additionally, the accuracy of lane detection has reached 88.7%. Pixel-level semantic segmentation is unnecessary if the two segmentation tasks, drivable area, and lane line detection, are performed for object detection before the detected targets go through pixel-level semantic segmentation. MtpNet processes images instead of videos, which can be optimized given the similarity among adjacent frames. In our future work, we will extend our method to process videos to make the system practical. Moreover, multi-task panoptic driving perception networks be extended and applied to interaction-aware systems in intelligent vehicles [41], [42].

REFERENCES

[1] Z. Zhou, Y. Wang, G. Zhou, K. Nam, Z. Ji, and C. Yin, "A twisted gaussian risk model considering target vehicle longitudinal-lateral motion states for host vehicle trajectory planning," *IEEE Trans. Intell. Transp. Syst.*, 2023.

[2] F. Ding, X. Han, C. Jiang, J. Liu, and C. Peng, "Fuzzy dynamic output feedback force security control for hysteretic leaf spring hydro-suspension with servo valve opening predictive management under deception attack," *Trans. Fuzzy Syst.*, vol. 30, no. 9, pp. 3736–3747, 2021.

[3] D. Wu, M.-W. Liao, W.-T. Zhang, X.-G. Wang, X. Bai, W.-Q. Cheng, and W.-Y. Liu, "Yolop: You only look once for panoptic driving perception," *Mach. Intell. Res.*, vol. 19, no. 6, pp. 550–562, 2022.

[4] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2117–2125.

[5] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10 781–10 790.

[6] Y. Jiang, Z. Tan, J. Wang, X. Sun, M. Lin, and H. Li, "Giraffedet: A heavy-neck paradigm for object detection," in *Int. Conf. Learn. Represent.*, 2022.

[7] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 7464–7475.

[8] N. Ma, X. Zhang, M. Liu, and J. Sun, "Activate or not: Learning customized activation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8032–8042.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 580–587.

[10] R. Girshick, "Fast r-cnn," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 21–37.

[14] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7263–7271.

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.* pmlr, 2015, pp. 448–456.

[16] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," Apr. 2018, a*rXiv:1804.02767*. [Online]. Available: https://arxiv.org/abs/1804.02767

[17] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.

[18] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, APR 2018.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Med. Image Comput. Comput. Assist. Interv.* Springer, 2015, pp. 234–241.

[20] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 801–818.

[21] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2881–2890.

[22] H. Li, P. Xiong, H. Fan, and J. Sun, "Dfanet: Deep feature aggregation for real-time semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9522–9531.

[23] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation," *Int. J. Comput. Vis.*, vol. 129, pp. 3051–3068, 2021.

[24] S. Y. Win and H. H. Lwin, "Lane boundaries detection algorithm based on retinex with line segments angles computation," in *Proc. 18th Int. Symp. Commun. Inf. Technol.* IEEE, 2018, pp. 160–164.

[25] J. Wang, W. Hong, and L. Gong, "Lane detection algorithm based on density clustering and ransac," in *Proc. CCDC*. IEEE, 2018, pp. 919–924.

[26] T. Zheng, H. Fang, Y. Zhang, W. Tang, Z. Yang, H. Liu, and D. Cai, "Resa: Recurrent feature-shift aggregator for lane detection," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 4, 2021, pp. 3547–3554.

[27] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, "Learning lightweight lane detection cnns by self attention distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1013–1021.

[28] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.

[29] L. Tabelini, R. Berriel, T. M. Paixao, C. Badue, A. F. De Souza, and T. Oliveira-Santos, "Keep your eyes on the lane: Real-time attention-guided lane detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 294–302.

[30] Z. Qin, P. Zhang, and X. Li, "Ultra fast deep lane detection with hybrid anchor driven ordinal classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022.

[31] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.

[32] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8759–8768.

[33] S. Lloyd, "Least squares quantization in pcm," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[34] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," Sep. 2022, a*rXiv:2209.02976*. [Online]. Available: https://arxiv.org/abs/2209.02976

[35] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.

[36] X. Xu, Y. Jiang, W. Chen, Y. Huang, Y. Zhang, and X. Sun, "Damo-yolo: A report on real-time object detection design," Apr. 2023, a*rXiv:2211.15444*. [Online]. Available: https://arxiv.org/abs/2211.15444

[37] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *Proc. 4th Int. Conf. 3D Vis.* Ieee, 2016, pp. 565–571.

[38] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," Apr. 2020, a*rXiv:2004.10934*. [Online]. Available: https://arxiv.org/abs/2004.10934

[39] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *Proc. IEEE Int. Veh. Symp.* IEEE, 2018, pp. 1013–1020.

[40] Y. Qian, J. M. Dolan, and M. Yang, "Dlt-net: Joint detection of drivable areas, lane lines, and traffic objects," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4670–4679, 2019.

[41] L. Crosato, H. P. Shum, E. S. Ho, and C. Wei, "Interaction-aware decision-making for automated vehicles using social value orientation," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1339–1349, 2022.

[42] L. Crosato, K. Tian, H. P. Shum, E. S. Ho, Y. Wang, and C. Wei, "Social interaction-aware dynamical models and decision-making for autonomous vehicles," *Advanced Intelligent Systems*, vol. 6, no. 3, p. 2300575, 2024.