Supplementary materials for "CurveLane-NAS: Unifying Lane-Sensitive Architecture Search and Adaptive Point Blending"

In this material, we provide some additional illustrations of the paper. Sec. 2 provides the searched architectures of our CurveLane-NAS in CULane and CurveLanes datasets. More intermediate results as shown in Sec. 1. The performance of the transferred architectures from CULane on TuSimple testing set can be found in Sec. 3. Sec. 4 provides the detailed description of the loss function used in training. The detailed algorithm of adaptive point blending module is presented in Sec. 5. Sec. 6 visualizes the additional qualitative results on CULane, TuSimple and CurveLanes datasets.

1 Intermediate Results for the Architecture Searching on CULane

Figure 1 shows the intermediate results of the searched architectures on CULane. The blue dots are the architectures trained and evaluated during the search. We searched about 500 architectures and it takes about 5 days on 4 computational nodes to complete searching. Red stars on the Pareto front are identified by non-dominate sorting in term of both F1 measure and FLOPS.

2 Searched Architectures of CurveLane-NAS

The searched architectures of CurveLane-NAS in CULane and CurveLanes datasets can be found in Table 1. Our CurveLane-NAS includes the backbone architecture search module and the feature fusion search module that not only is the optimal trade-off between accuracy and different computation constraints but also can find a better fusion of the high-level and low-level features. In each dataset, we search three models that are denoted as CurveLane-S, CurveLane-M, and CurveLane-L by considering three kinds of computational constraints. In backbone architecture, the encoding string looks like "BB_64_14_[3, 5]_[9, 14]" in CurveLane-S in CULane, where the first placeholder encodes the block setting, the base channel size is 64, 14 is the total number of blocks, [3, 5] are the position of downsampling blocks and [9, 14] are the position of doubling the channel size. In the feature fusion search module, we set the fusion layers $M = 2$ due to the computational constraints. The feature fusion module of CurveLane-S in CULane is "$[F_2F_1, R_3]$_$[F_2F_1, R_3]$" where $[F_2F_1, R_3]$ is one fusion layer O_i, F_2F_1 two

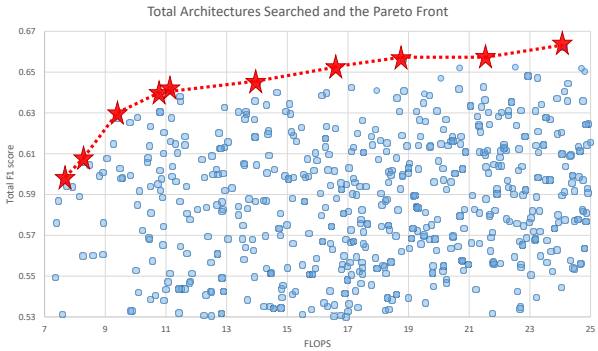

Figure 1: Intermediate Results for the Architecture Searching on CULane. The blue dots are the architectures trained and evaluated during the search. Red stars on the Pareto front are the models with best trade-off for accuracy and FLOPS.

input feature that picked from the output feature levels with different generate by the backbone denoted $\{F_1, F_2, ..., F_t\}$ and R_3 is target output resolution corresponding to the F_3.

Dataset	Models	Backbone Architecture	Feature Fusion Module
CULane	CurveLane-S	BB_64_14_[3, 5]_[9, 14]	$[F_2F_1, R_3]\_[F_2F_1, R_3]$
	CurveLane-M	BB_64_26_[12, 18]_[12, 16]	$[F_1F_2, R_3]\_[F_2F_1, R_3]$
	CurveLane-L	BB_128_25_[3, 13]_[4, 13]	$[F_2F_1, R_3]\_[F_2F_1, R_3]$
CurveLanes	CurveLane-S	BB_64_27_[2, 3]_[23, 27]	$[F_2F_3, R_3]\_[F_1F_2, R_3]$
	CurveLane-M	BB_64_27_[4, 13]_[19, 24]	$[F_2F_1, R_3]\_[F_2F_1, R_3]$
	CurveLane-L	BB_64_25_[3, 13]_[4, 13]	$[F_2F_1, R_3]\_[F_2F_1, R_3]$

Table 1: Searched architectures in CULane and CurveLanes datasets.

3 Transferability of Our Searched Architectures

Table 2 shows the performance of the transferred architectures from CULane on the TuSimple dataset with a comparison between our methods and other SOTA methods. The transferred architectures reached a comparable performance with most SOTA methods which proves the effectiveness, transferability and robustness of our multi-objective search framework in dealing with lane detection tasks.

Methods	LaneNet[1]	EL-GAN[2]	SCNN[3]	Enet-SAD[4]	PointLaneNet[5]	CurveLane-S	CurveLane-M	CurveLane-L
Accuracy	96.38%	96.39%	96.53%	96.64%	95.15%	95.97%	96.21%	96.52%
FP	0.0780	0.0412	0.0617	0.0602	0.091	0.1145	0.1032	0.0813
FN	0.0244	0.0336	0.0180	0.0205	0.038	0.0316	0.0321	0.0293

Table 2: Performance of the transferred architectures from CULane on TuSimple testing set. CurveLane-S, CurveLane-M, and CurveLane-L are transfered architectures from the searched results of CULane.

4 Loss Function

CurveLane-NAS predicts one line proposal per grid. For each grid g_{ij}, the model will predict a set of offsets Δx_{ijz} and one ending point position to generate line proposals, and corresponding confidence scores. The loss function consists of two parts: the regression loss of line proposals and the classification loss of the confidence scores. The regression loss is smooth L_1 loss between the predicted line proposals and the ground-truth lane line. The cross-entropy loss is designed as the classification loss over two classes: real lane line or not. During training we optimize the following multi-part loss function:

$$Loss = \frac{1}{N_{cls}} \sum_{i=1}^{w} \sum_{j=1}^{h} Loss_{ij}^{cls} + \frac{1}{N_{reg}} \sum_{i=1}^{w} \sum_{j=1}^{h} m_{ij}^{obj} Loss_{ij}^{reg}$$

where

$$Loss_{ij}^{cls} = -\left[y_{ij} ln(p_{ij}) + (1 - y_{ij}) ln(1 - p_{ij})\right]$$

and

$$Loss_{ij}^{reg} = \alpha smoothL_1(ending_{ij}^{p} - ending_{ij}^{gt}) + \beta \sum_{k=1}^{n} smoothL_1(offset_{k}^{p} - offset_{k}^{gt})$$

.

where N_{cls} is the number of the grids. $N_{reg}$is the number of the samples selected. m_{ij}^{obj} is equal to 1 when the grid is positive, otherwise, it is equal to 0. α and β are penalty coefficient of the ending point position and the set of offsets. In this paper, $\alpha = 10$ and $\beta = 1$. In order to improve the performance and to reduce the training time, the Online Hard Example Mining is used in our experiments.

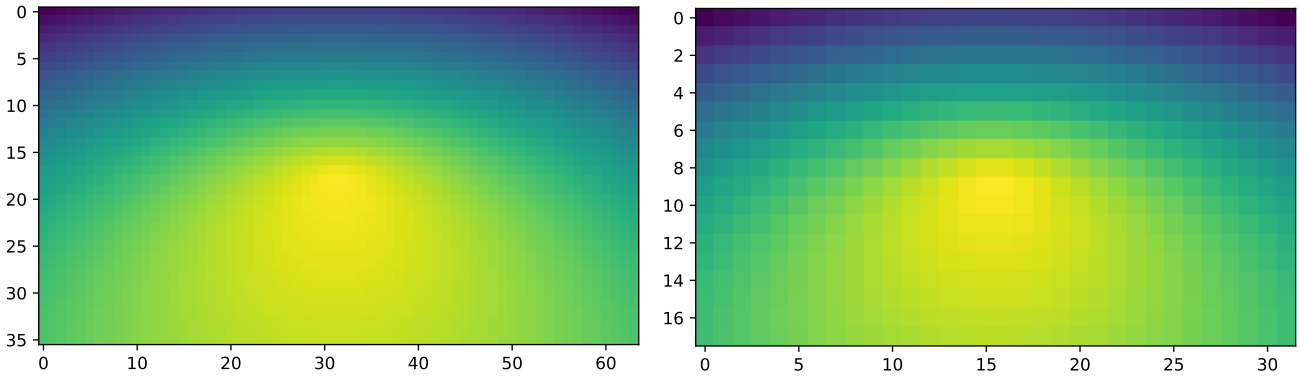

Figure 2: The adaptive score masking m_f of CurveLane-L in CULane dataset including two heads from two feature maps in different stages. The input size is 512×288 in our experiment, so the size of the third stage and fourth stage are 36×64 (Left) and 18×32 (Right) respectively. Since the receptive fields in the fourth stage are larger than the third stage so the adaptive score masking m_f in the fourth stage is emphasis regions where lane lines may exist but the adaptive score masking m_f in the third stage is emphasis center regions more than the fourth stage.

5 Detailed Algorithm of Adaptive Point Blending Search Module

The adaptive point blending module to search a novel multi-level post-processing refinement strategy to combine multi-level head prediction and allow more robust prediction over the shape variances and remote lanes. The detailed algorithm of the new post-processing adaptive point blending search module can be found in Algorithm 1. For the comparisons, we summary the detailed algorithm of Line-NMS to Algorithm 1. Line-NMS keep only one lane with the highest score in the distance threshold can be found in the Algorithm 1. However, the receptive field is too small to catch the long-rang semantic to predict the whole line, so far away from the anchor is inaccurate, this situation can be found in Figure 3 (f), Figure 4 (a), (b) and (f). In practice, we found that each anchor can only predict the offsets precisely locally around the anchor, so our adaptive point blending search module uses the anchor points of all lanes in the distance threshold to replace the corresponding point in the lane with the highest score. Thus, those remote parts of lanes and curve shape can be mended by the local points and the final results are better.

The distance threshold T of Line-NMS and our adaptive point blending search module is initialized to 10 in CULane and TuSimple. If the number of lane lines is more than five updates the distance threshold $T = T + 10$, this setting mostly follows the PointLaneNet. However, the number of lane lines in CurveLanes is up to 14 and the number is uncertain, so the distance threshold T of Line-NMS and our adaptive point blending search module is set to 90 in CurveLanes according to our experience. The adaptive score masking m_f on the original score prediction for each feature map f can be implemented as shown in Algorithm 1. The adaptive score masking allows different emphasis regions in multi-level prediction. The adaptive score masking m_f of CurveLane-L in CULane dataset as shown in Figure 2. The input size is 512×288 in our experiment, so the size of the third stage and fourth stage are 32×64 and 16×32 respectively. The adaptive score masking m_f in the fourth stage is emphasis regions where lane lines may exist but the adaptive score masking m_f in the third stage is emphasis center regions more than the fourth stage. So the adaptive score masking can emphasize regions where the learned lane's information to improve the performance of lanes detection by adaptive adjusts the detection scores.

6 Additional Qualitative Results

More qualitative comparisons on CULane, TuSimple and CurveLanes datasets between PointLaneNet and our method CurveLane-L can be found in Figure 3, 4 and 5, respectively. From the comparisons, our method performs better in the difficult scenarios such as large-curves, remote lanes, wet roads, night and multi-lanes (the number of lane lines is more than 4), while the PointLaneNet fails to detect them. Results of our method not only are more accurate than the PointLaneNet but also have less false negative lanes due to the help of the CurveLane-NAS to automatically capture both long-ranged coherent and accurate short-range curve information while unifying both architecture search and post-processing on curve lane predictions via point blending. The elastic backbone search module and the feature fusion search module to find a better fusion of the local and global

Algorithm 1 Line-NMS and Adaptive Point Blending Search Module

Input: $L = \{l_1, ..., l_n\}, S = \{s_1, ..., s_n\}, T$

L is $n * z$ matrix of initial detection lanes, where $n = w \times h$ is the number of detection lanes and z is the number of the points of one lane.

S contains corresponding confidence scores.

T is the distance threshold.

$[u_{xf}, u_{yf}]$ denote the center position of the image.

$\alpha_{1f}, \beta_{1f}, \alpha_{2f}$ searched from the adaptive point blending search module.

Output: $D = \{l_1, ..., l_k\}, S = \{s_1, ..., s_n\}$

D is $k * z$ matrix of initial detection lanes, where k is the number of lanes last reserved and z is the number of the points of one lane.

S contains corresponding confidence scores.

Begin:

```
# Adaptive Score Masking m_f
for lane in L do
    c_x,c_y = lane_cx, lane_cy
    logit(m_f) = α_1f(c_y) + β_1f + α_2f[(c_x − u_xf)^2 + (c_y − u_yf)^2]^(1/2)
    S = S × logit(m_f)
end
D ← {}
while L ≠ empty do
    m ← argmax S
    L ← L − l_m, S ← S − s_i
    for i in L do
        dis(l_i, l_m) ← Σ_{point=0}^{z} abs(l_i[point] − l_m[point]) / z
        if dis(l_i, l_m) ≤ T then
            # Line-NMS
            L ← L − l_i, S ← S − s_i
            # Adaptive Point Blending
            localpoints ← anchor points of d_i
            l_m[localpoints] ← d_i[localpoints]
            L ← L − l_i, S ← S − s_i
        end
    end
    D ← D ⋃ l_m
end
return D, S
end
```

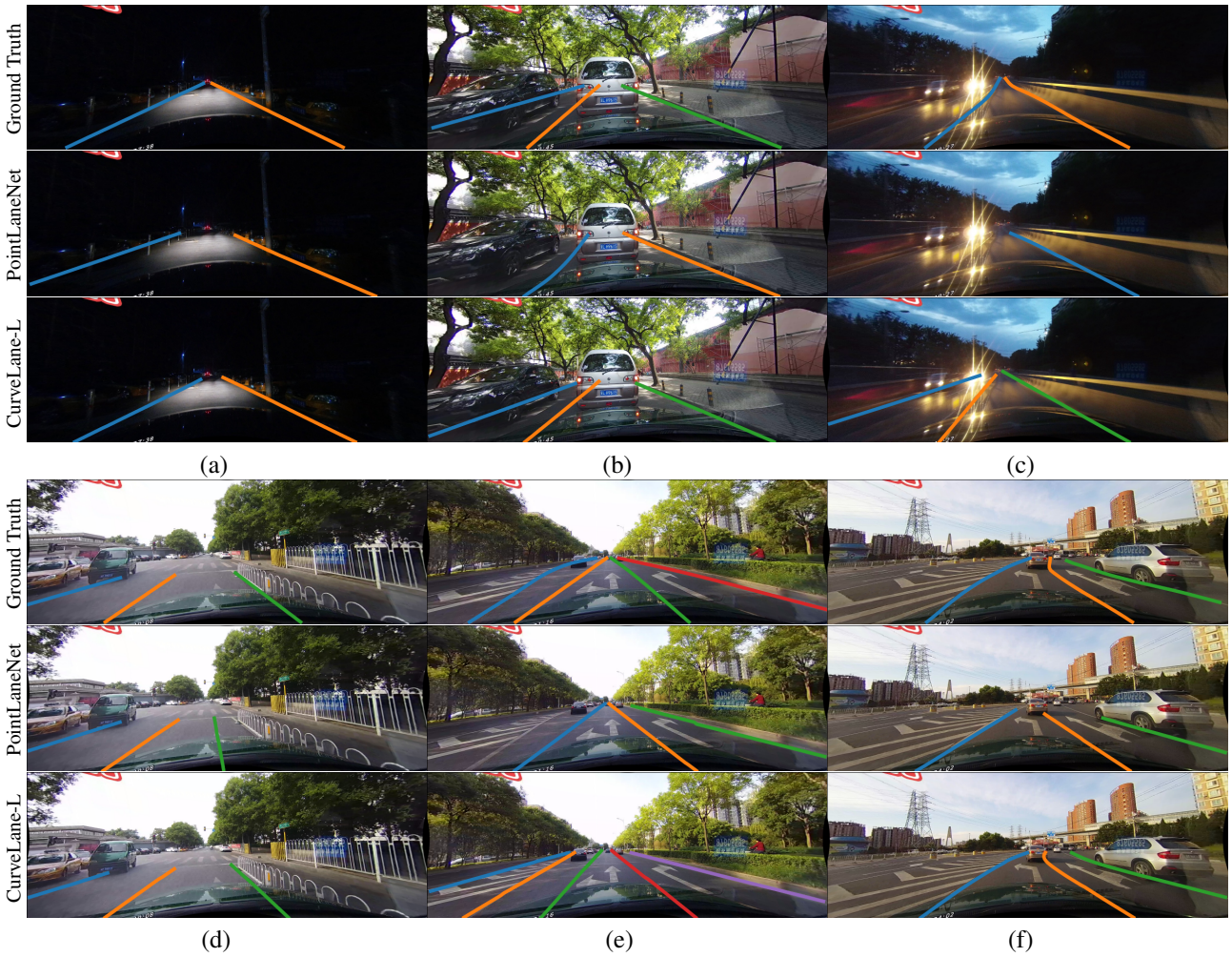

Figure 3: Qualitative results comparison on CULane between PointLaneNet and our method CurveLane-L: Ground-truth(top), PointLaneNet(middle), and our Curve-L(bottom). Our method performs better in the difficult scenarios such as night, crowd and large-curves.

context for multi-level hierarchy features. The adaptive point blending module to search a novel multi-level post-processing refinement strategy to combine multi-level head prediction and allow more robust over the variate shape of the lane and remote lanes.

References

[1] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: an instance segmentation approach," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 286–291.

[2] M. Ghafoorian, C. Nugteren, N. Baka, O. Booij, and M. Hofmann, "El-gan: embedding loss driven generative adversarial networks for lane detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[3] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[4] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, "Learning lightweight lane detection cnns by self attention distillation," *arXiv preprint arXiv:1908.00821*, 2019.

[5] Z. Chen, Q. Liu, and C. Lian, "Pointlanenet: Efficient end-to-end cnns for accurate real-time lane detection," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 2563–2568.

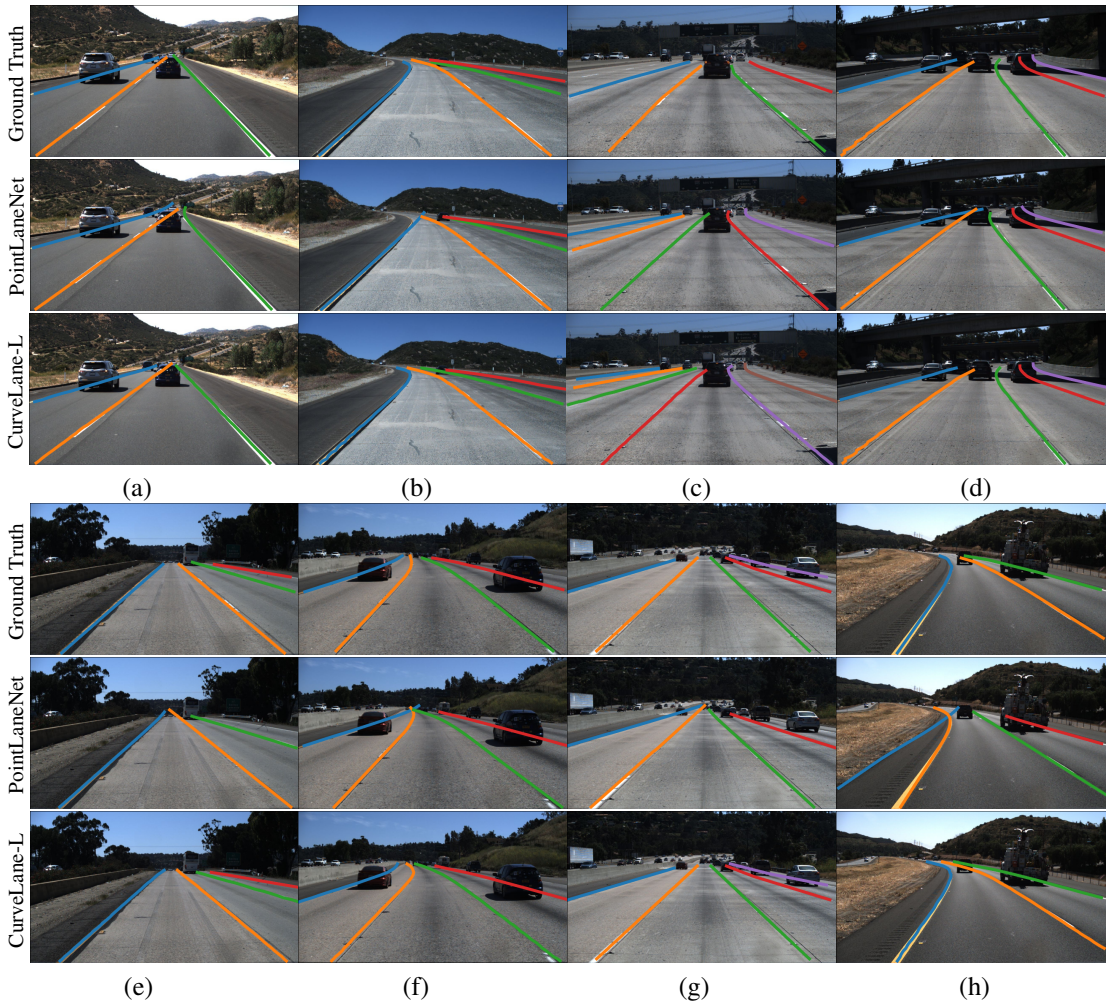

Figure 4: Qualitative results comparison on TuSimple between PointLaneNet and our method CurveLane-L: Ground-truth(top), PointLaneNet(middle), and our CurveLane-L(bottom). Our method performs better in the difficult scenarios such as large-curves and remote lanes.

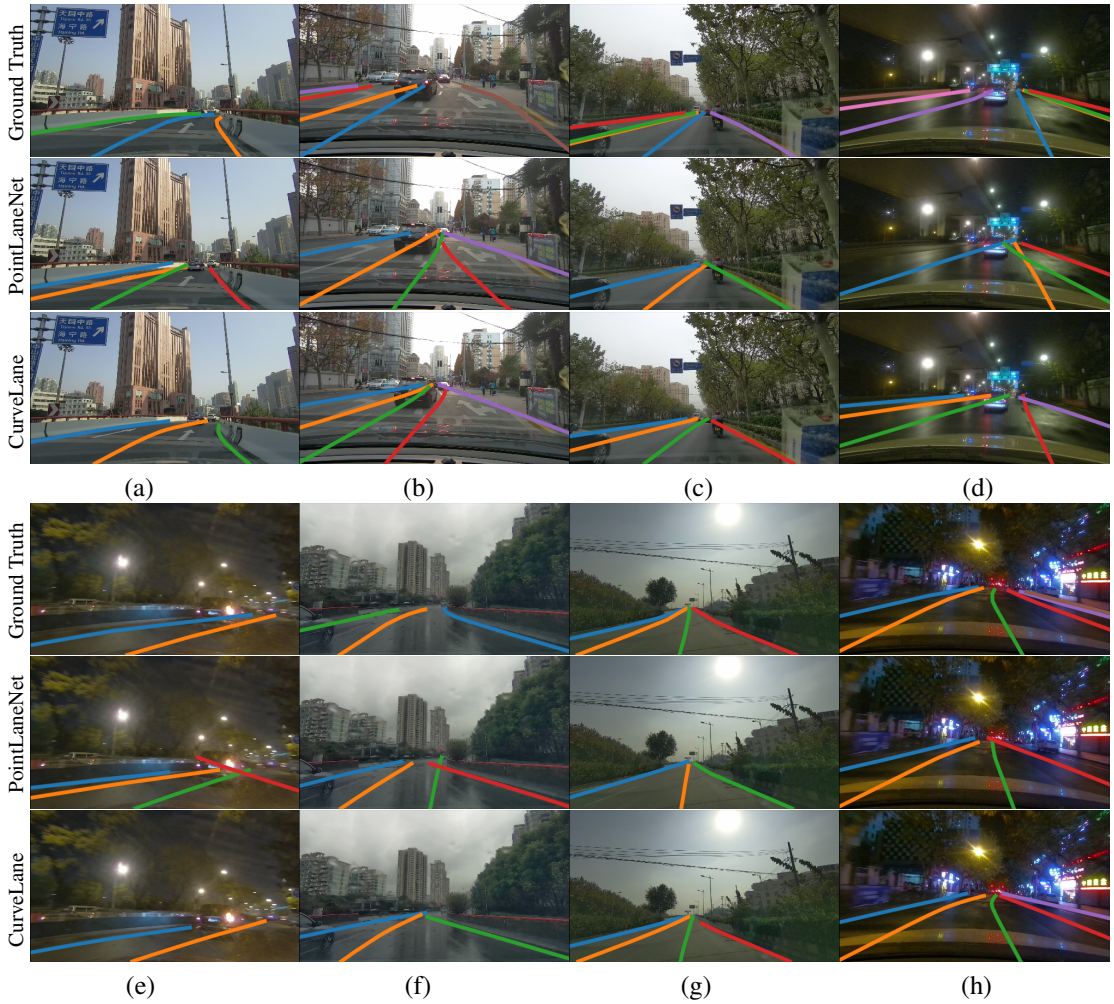

Figure 5: Qualitative results comparison on CurveLanes between PointLaneNet and our method CurveLane-L: Ground-truth(top), PointLaneNet(middle), and our CurveLane-L(bottom). Our method performs better in the difficult scenarios such as night, crowd and multi-lanes.