LaneAF: Robust Multi-Lane Detection with Affinity Fields

Genze Zhou[†1], Luoyu Chen[†*2], and Fei Wu[†‡2]

[1]Department of Computer Science, The University of Adelaide
[2]Department of Computer Engineering, Australian National University

Abstract

Lane detection is a long-standing task and a basic module in autonomous driving. The task is to detect the lane of the current driving road, and provide relevant information such as the ID, direction, curvature, width, length, with visualization. Our work is based on CNN backbone DLA-34, along with Affinity Fields, aims to achieve robust detection of various lanes without assuming the number of lanes. Besides, we investigate novel decoding methods to achieve more efficient lane detection algorithm.

1. Introduction

Driving safety is the constantly topic in road traffic, with the rapidly increasing number of cars on the road, autonomous driving or auxiliary systems raised considerable attention in both industries and academia. Lane detection is prior and fundamental step for autonomous driving, which correctly localizes the car and obey the driving rules in the lane. In real road there are various types of lane lines (*e.g.*, solid white, double solid white, broken white, *etc.*), and the meaning may vary in different countries. Lane detection enable vehicles to understand the interaction with lanes, predict the driving tracks and alert drivers as an auxiliary system. In applications the lane detection is considered as a computer vision task, from the input images the algorithm need to separate different lane instances.

The difficulties of lane detection lies in extracting semantic information under complicated circumstances, such as the merging, ending, splitting of lanes. The accuracy is affected by various factors: (1) The complexity of lanes, including the stains and degree of wear of the lanes, the obscured parts by coming vehicles; (2) The impact of shading of buildings, trees, viaducts along the road; (3) Constantly changing light condition for night, fog, haze. Those factors largely impede the recall of lane detection algorithms.

From the previous study, the segmentation of lanes is formulated as an instance segmentation problem, the detection algorithm should able to identify pixels for lanes and cluster them into distinct lane instances. In our approach, we separate the segmentation task into two stages: a simple binary segmentation and a clustering process. To identify different lane entities, a novel clustering algorithm based on Affinity Fields is specially designed for lane detection.

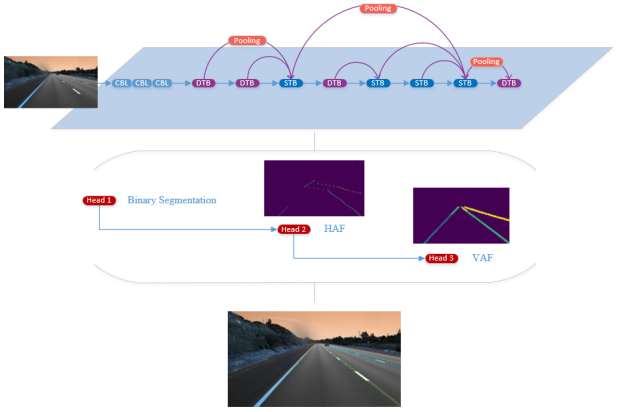

Figure 1. The proposed pipeline for LaneAF. It consists of two parts: (i) Top: DLA-34 backbone for binary segmentation masks along with the predicted vertical affinity fields and horizontal affinity fields. (ii) Down: The affinity field decoder, which infers from the affinity fields to generate lane instances.

The Affinity Fields is a set of vectors at different location representing the proximity between pixels. We divided the Affinity Fields into horizontal field which is organized by rows and vertical fields which is organized by successive rows. By decoding the Affinity Fields we can cluster the foreground pixels into different lanes without assume a fixed number of lanes.

The main contributions of this paper are listed below:

(1) We separate the lane detection into a two-stage pipeline, including semantic segmentation and clustering post-processing. We demonstrate the CNN backbone DLA-34 [18] can perfectly complete the segmentation as well as predicting Affinity Fields for separating lane instances.

(2) We design the clustering process based on Affinity Fields and explain the training and inference process in de-

[†]These authors contributed equally to this work
[*]luoyu.chen.mlcv@gmail.com
[‡]wufei.mlcv@gmail.com

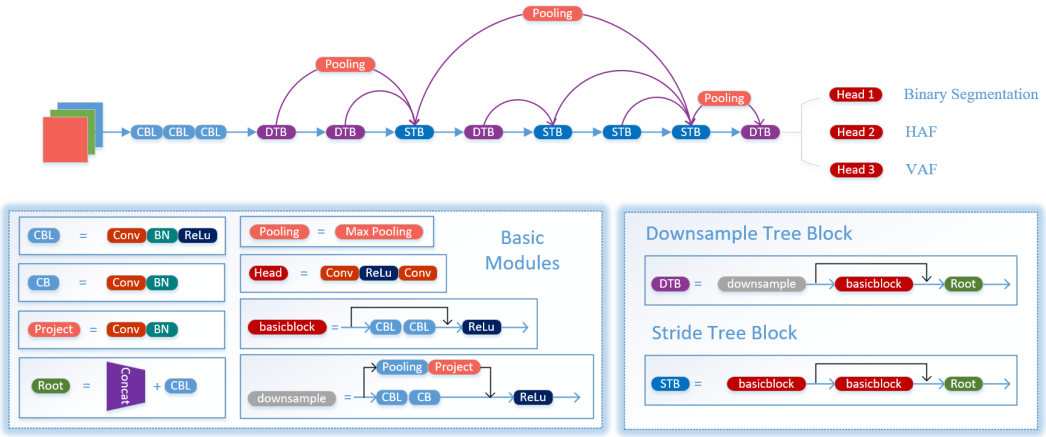

Figure 2. The backbone network of LaneAF.

tail. Affinity Fields prediction branches are added to the backbone network, which generates horizontal vector field (HAF) and vertical vector field (VAF) for each pixel [2].

(3) We illustrate the decoding algorithm for predicted affinity fields, with a revised version to boost the efficiency.

2. Related Work

2.1. Traditional Lane Detection

Traditional lane detection methods are based on camera calibration and traditional edge detection algorithms. The input images are first convert to grayscale images to reduce the computational complexity. Then filters like Gaussian filter, bilateral filter, gabor filter, trilateral filter are applied to remove the noise. S. Srivastava *et al.* compared the effect of using median, wiener, and hybrid median filters for efficient lane detection algorithm, which can provide high quality results for noisy images input [15]. After that, the lane are extracted by edge detectors then regress by line fitting algorithms. Y. Wang *et al.* (2004) used Canny/Hough Estimation of Vanishing Points (CHEVP) to calculate parameters for a B-Snake spline geometric road model [17]. M. Meuter *et al.* (2009) proposed a robust approach for lane recognition by combining two Extended Kalman filter with the Interacting Multiple Models (IMM) algorithm [9]. F. Mariut *et al.* (2012) proposed an algorithm based on Hough transform, they extracts the inner margin of the lane to ensure the accuracy of lane detection [8].

However, the previous algorithms are all limited by hand-crafted features, they may work efficiently for straight roads but fails in various conditions like curved roads or images with shading [8]. Moreover, they are omitting semantic information and based on mathematical transforms.

2.2. Segmentation based Lane Detection

With the researches in deep learning, data driving approaches are introduced in lane detection and model it as a multi-class instance segmentation problem [12, 14, 19, 4, 11]. The segmentation can be executed in real time, Adam Paszke *et al.* propose a novel deep neural network architecture for tasks requiring low latency operation [12]. R. K. Satzoda *et al.* proposed a "drive analysis" was introduced to extract mid-level semantic information from raw sensor data [14]. Qin Zou *et al.* studied the use of multi-frame images of continuous driving scenes and proposed a hybrid deep structure combining CNN and RNN. The information of multiple consecutive frames are extracted through a CNN block with time-series properties, they are input into the RNN block for feature learning and lane prediction [19].

Similar as our method, some methods perform binary segmentation first then separate them into lane instances [10, 13, 7]. F. Pizzati *et al.* present an end-to-end system for real-time lane detection based on two cascaded neural networks [13]. But this model can not adaptive to the number of lanes and have to detect fixed number of lanes. Y. Ko *et al.* improve the model based on the key points estimation and perform instance segmentation with several hourglass models [7]. Anchor based object detection is also used in lane detection [5, 3, 16, 6], some track the lane markers temporally [16], some tend to predict the 3D-layout [3].

In our method, we take the advantage of affinity fields, which was first used in multi-person 2D pose estimation [2], to separate the segmentation masks into different lane instances without pre-defined the number of lanes and can achieve more robust results even the lane is occluded [1].

3. Methods

The LaneAF pipeline includes two stages, the forward process of CNN backbone network and lane instances decoding using Affinity Field, shown in Fig. 1. The CNN network is trained to perform binary segmentation, as well as the prediction of two Affinity Fields. Different lane instances are detected by decoding the predicted affinity field. In section 3.1 we will first discuss the backbone network, then explain the generation of Affinity Fields and the decoding algorithm in section 3.2 and 3.3, as well as the improvements in the decoding process in section 3.4.

3.1. Backbone Network

We choose DLA-34 as the backbone network to classify pixels into the lane or the background. Moreover, two parallel branch are added to predict horizontal vector field (HAF) and vertical vector field (VAF) for foreground pixels.

The Deep Layer Aggregation (DLA) network extracts the abstract structure of VGG, ResNet, DenseNet and proposes two types of aggregation: the iterative deep aggregation (IDA) and hierarchical deep aggregation (HDA). The IDA join neighboring stages to merge features from different scales for spatial fusion. The HDA stands for hierarchical connections with tree structure to merge features different channels for semantic fusion. As shown in Fig. 2

3.2. Supervised Affinity Fields Training

Our backbone network will not only perform binary segmentation, but also trained to predict the horizontal and vertical affinity fields (HAF and VAF). As we discussed before, the HAF and VAF are sets of vectors $\overrightarrow{H}(\cdot,\cdot)$, $\overrightarrow{V}(\cdot,\cdot)$ at different location (x, y) representing the proximity between pixels. The coding process of Affinity Fields is a simple row by row, bottom to top calculation, while decoding the Affinity Fields is a reversed optimization problem, follow a similar row by row schema.

3.2.1. Computation of ground truth Affinity Fields

Before training our model, we need to generate the annotations of HAF and VAF. For each row y in the input image, the ground truth HAF vectors $\overrightarrow{H_{gt}}(\cdot,\cdot)$ are computed only for lane pixels (x_i^l, y) at lane l:

$$\begin{aligned}\overrightarrow{H_{gt}}(x_i^l, y) &= (\frac{\overline{x}_y^l - x_i^l}{|\overline{x}_y^l - x_i^l|}, \frac{y-y}{|y-y|}) \\ &= (\frac{\overline{x}_y^l - x_i^l}{|\overline{x}_y^l - x_i^l|}, 0)\end{aligned} \tag{1}$$

Where $\overline{x}_y^l$ stands for the mean x-coordinate of all points belong to lane l in row y, as the pixel (x_2^l, y) shown in Fig. 3, different colour of pixels are from different lanes.

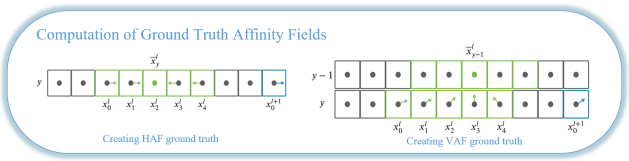

Figure 3. The calculation for ground truth affinity fields.

Similarly, the computation for ground truth VAF vectors $\overrightarrow{V_{gt}}(\cdot,\cdot)$ are only for lane pixels (x_i^l, y) at lane l, but takes the relationship between rows into account:

Algorithm 1: Calculating affinity fields for training

Input: The ground truth segmentation, $S \in \mathbb{R}^{H\times W}$;
The number of lanes, L.
Output: The horizontal affinity fields, HAF;
The vertical affinity fields, VAF.

initialization: $H, W = shape(S)$,
$HAF, VAF \leftarrow zeros(H, W, 2)$;
for $l = 1$ *to* L **do**
  initialize $cols_p \leftarrow ones(S[H, :] == l)$
  for $row = H - 1$ *to* 1 **do**
    find lane pixels:
    $cols \leftarrow find(S[row, :] == l)$ **for** *col in* $cols$ **do**
      Generate horizontal affinity fields:
      $HAF[row, col] \leftarrow \overrightarrow{H_{gt}}(row, col)$
    end
    for *col in* $cols_p$ **do**
      Generate vertical affinity fields:
      $HAF[row, col + 1] \leftarrow$
      $\overrightarrow{V_{gt}}(row, col + 1)$
    end
    $cols_p \leftarrow cols$
  end
end
return *HAF, VAF*

$$\begin{aligned}\overrightarrow{V_{gt}}(x_i^l, y) &= (\frac{\overline{x}_{y-1}^l - x_i^l}{|\overline{x}_{y-1}^l - x_i^l|}, \frac{y-1-y}{|y-1-y|}) \\ &= (\frac{\overline{x}_{y-1}^l - x_i^l}{|\overline{x}_{y-1}^l - x_i^l|}, -1)\end{aligned} \tag{2}$$

Where $\overline{x}_y^l$ stands for the mean x-coordinate of all points belong to lane l in previous row $y-1$, as the pixel $(x_2^l, y-1)$ shown in Fig. 3. The Calculation process is shown in algorithm 1.

3.2.2. Loss Function

For each branches in our model we adopt different loss, for the binary segmentation branch we used a weighted BCE loss to balance the loss from foreground and background:

$$L_{BCE} = -\frac{1}{N}\sum_i [w \cdot y_i \cdot log(p_i) + (1-y_i) \cdot log(1-p_i)], \tag{3}$$

where y_i is the label for pixel i and p_i is the output of our network with sigmoid activation. w is the balancing coefficient. Moreover, an additional IoU loss is adopted to further regularize the imbalance dataset:

$$L_{IoU} = -\frac{1}{N}\sum_i [1 - \frac{y_i \cdot p_i}{y_i + p_i - y_i \cdot p_i}] \tag{4}$$

Algorithm 2: Decoding affinity fields into lanes

Input: The segmentation mask, $M \in \mathbb{R}^{H\times W}$;
The predicted HAF, $HAF \in \mathbb{R}^{H\times W\times 2}$;
The predicted VAF, $VAF \in \mathbb{R}^{H\times W\times 2}$;
clustering threshold, τ.
Output: The segmentation result, $S \in \mathbb{R}^{H\times W}$.

Initialization: $H, W = shape(M)$,
  $S \leftarrow zeros(H, W)$,
  $lane\_class \leftarrow dict(LIST)$
  $end\_points \leftarrow \varnothing$,
  $L \leftarrow 0$;
for $row = H$ *to* 1 **do**
  find lane pixels: $cols \leftarrow find(S[row, :] > 0)$
  initialize $clusters \leftarrow \varnothing$;
  for *col in cols* **do**
    find lane pixels:
     $cols \leftarrow find(S[row, :] == l)$
    for *col in cols* **do**
      Update cluster for each row:
       $clusters.update(c^*_{HAF}(x_i, y-1))$
    end
    for $l = 1$ *to* L **do**
      if $d^*(l) \leq \tau$ **then**
        update the pixels at the end of lanes:
         $end\_points[l] \leftarrow c^*_{VAF}(l)$
        for *col in* $c^*_{VAF}(l)$ **do**
          $S[row, col] \leftarrow l$
           $lane\_class[l].append((row, col))$
        end
      end
    end
  end
  for *cluster in clusters* **do**
    if *cluster is not assigned* **then**
      Span a new lane: $L \leftarrow L + 1$
      $end\_points[L] \leftarrow cluster$
    end
  end
end
return $S, lane\_class$

For the affinity field branches, we applied a simple L1 regression loss only to the lane pixels of both HAF and VAF:

$$L_{AF} = -\frac{1}{N_{fore}} \sum_i (|y_i^{HAF} - p_i^{HAF}| + |y_i^{VAF} - p_i^{VAF}|) \tag{5}$$

The total loss is the summation of above losses:

$$L_{total} = L_{BCE} + L_{IoU} + L_{AF} \tag{6}$$

3.3. Decoding Affinity Fields

As we discuss in the previous sections, our model will output the segmentation mask for foreground and background, along with the predict HAF and VAF. With the predicted Affinity Fields, our decode process follows a simple row by row, bottom to top schema:

Row by row clustering: Denote $\overrightarrow{H}_{pred}(\cdot,\cdot)$ as the predicted HAF, we utilize it to cluster lane pixels in each row into different entities. Because the HAF ground truth are generated as pointing at the middle pixel, we can assign the foreground pixels in row $y-1$ into different lanes according to the direction of HAF vectors:

$$c^*_{HAF}(x_i, y-1) = \begin{cases} C^{k+1} & \text{if } \overrightarrow{H}_{pred}(x_{i-1}, y)_0 \leq 0 \\ & \wedge\ \overrightarrow{H}_{pred}(x_i, y)_0 > 0, \\ C^k & \text{otherwise,} \end{cases} \tag{7}$$

where $c^*_{HAF}(x_i, y-1)$ denotes the optimal lane assignment for a foreground pixel $(x_i, y-1)$; C_k, C_{k+1} denote two different lanes, as shown in fig 4, the red HAF vectors in a row all pointing at the middle pixel.

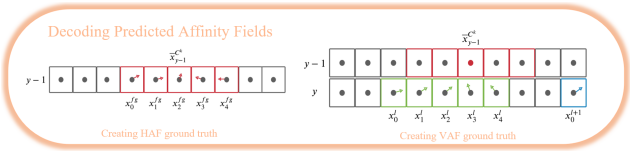

Figure 4. The decoding of Affinity Fields.

Bottom to top concatenate: After assign foreground pixels each row into $c^*_{HAF}(x, y)$, they are connected to one cluster of lane pixels in the previous row from bottom to top according the $\overrightarrow{V}_{pred}$, if the VAF vectors are pointing at the middle pixel in the previous row, which indicates they are more likely to be in one lane. The process can be represented as:

$$c^*_{VAF}(x_i, y-1) = \arg\min_{C^k} d^{C^k}(l). \tag{8}$$

Here, $d^{C^k}(l)$ denotes the error of associating cluster C^k to an existing lane l:

$$\begin{aligned} d^{C^k(l)} = & \frac{1}{N_y^l} \sum_{i=0}^{N_y^l - 1} ||(\overline{x}^{C^k}, y-1) - (x_i^l, y) - \\ & (\overrightarrow{V}_{pred}(x_i^l, y) \cdot ||(\overline{x}^{C^k}, y-1) - (x_i^l, y)||)||, \end{aligned} \tag{9}$$

where N_y^l denote the number of pixels for lane l in row y, as illustrated in fig 4, if the VAF vectors in green are pointing at the middle pixel in the previous row in red, they are concatenated as one lane. The decoding algorithm is shown in algorithm 2.

Algorithm 3: Faster Decoding affinity fields

Input: The segmentation mask, $M \in \mathbb{R}^{H \times W}$;
The predicted HAF, $HAF \in \mathbb{R}^{H \times W \times 2}$;
The predicted VAF, $VAF \in \mathbb{R}^{H \times W \times 2}$;
clustering threshold, τ.
Output: The segmentation result, $S \in \mathbb{R}^{H \times W}$.

Initialization: $H, W = shape(M)$
$S \leftarrow zeros(H, W)$,
% Set shrinkage ratio:
$\alpha \in \mathcal{N}$
% store classification candidates:
$cands \leftarrow []$
% store regression parameters:
$regs \leftarrow dict()$
for $row = 1$ *to* H **do**
  for $col = 1$ *in* W **do**
    if $M[row, col] > \tau$ **then**
      $cands$.append((row, col))
    end
  end
end
$m, haf, vaf \leftarrow$ M[:: α], HAF[:: α], VAF[:: α]
$s, lane\_class \leftarrow$ DecodeAF(m, haf, vaf)
for $lane$ *in* $lane\_class$ **do**
  $lane\_class[lane] \leftarrow \alpha * lane\_class[lane]$
  $k, b \leftarrow$ HuberLinReg ($lane\_class[lane]$)
  $regs[lane] \leftarrow (k, b)$
  %Interpolate lane points with gap α%
  LinearInterpolate($lane\_class[lane]$)
end
for $lane$ *in* $lane\_class$ **do**
  for $row = 1$ *to* H **do**
    while *True* **do**
      $i \leftarrow 0$
      $bounded \leftarrow row \pm i \in [0, W]$
      $is\_lane \leftarrow M\left[row, lane[row] \pm i\right] > \tau$
      $vacant \leftarrow S[row, lane[row] \pm i] == 0$
      if $bounded$ & is_lane & $vacant$ **then**
        $S[row, lane[row] \pm i] \leftarrow lane$
        $cands$.delete($(row, lane[row] \pm i)$)
        $i \leftarrow i + 1$
      end
      else break
    end
  end
end
for $cand$ *in* $cands$ **do**
  $D \leftarrow [\frac{|k * cand_x - cand_y + b|}{\sqrt{k^2+1}}$ **for** k, b *in* $regs]$
  $S[cand] \leftarrow lane\_class.key([$ argmin $D])$
end
return S

3.4. Faster Decoding

The previous decoding process follow a simple row-by-row schema on the entire segmentation mask, HAF and VAF, which could be costly. We proposed a faster decoding method on the basis of algorithm 2, shown in algorithm 3. First we pick a shrinkage ratio $\alpha \in \mathcal{N}$. Since this is not a lossless decoding process, from our practice, when $\alpha \in \{2, 3, 4, 5\}$, good lane decoding quality can be preserved. The larger α is, the shorter decoding time it will be. Then we down sample the segmentation mask, HAF, VAF respectively and feed them into the decoding process as shown in algorithm 2.

This is a much faster decoding process since down sampling means both length and width will shrink by α, the image size to decode will be $1/\alpha^2$ as before. After decoding on shrunk images, we can get pixel sequences of lane with its class label. We approximate each lane as a straight line for simplicity, then by using pixel sequences, perform Huber linear regression as a robust line estimator to regress each lane to get their slope k and intercept b, and this will be used later. To recover the entire decoded lane, we need to up sample to their original sizes, but here we only inversely scale up the pixel sequences by α and linearly interpolate the gaps in between for each pixel sequence. Therefore, till now we have got the mid line of each lane, to complete the segmentation task, we need to identify the specific class for all pixels which are identified as lane area. Now we have already identify the mid line of each lane, next we only have to horizontally expand the mid line until a lane has been segmented.

We expand a lane base on the following stopping criterion: (1) expansion is limited within the image size. (2) Expansion should stop when touching the boundary of lane areas. (3) Expansion should stop when touching pixel that have already been classified as a part of another lane. (4) expansion happens on left and right direction at the same time, assume the width on the left side and right side are the same, if one side stops expansion the other side should stop at the same time. After expansion phase is finished, it is very likely there are still some pixels in lane areas have not been classified. To classify these remaining pixels, we choose to classify them based on their point-line distances to each lane. We use the already regressed lane as straight line. Finally we can assign each unclassified pixel by picking the lane with the least point-line distance and decoding phase is completed.

4. Experimental Results

We use the Adam optimizer as our optimizer with a learning rate of 0.0001, weight decay of 0.001, and train for a total of 20 epochs. The weight w for the loss was set to 9.6 because there are approximately 9.6 times as many background pixels than there are foreground (lane) pixels in most public datasets. To avoid overfitting, early stopping was implemented by retaining the model parameters

Method	RunTime (ms/img)	Accuracy	F1-score	FP	FN
DecodeAF	105	0.956	0.965	0.028	0.042
DecodeAF, $\alpha = 2$	**75**	0.944	0.956	0.037	0.050
DecodeFaster, $\alpha = 2$	86	**0.953**	**0.962**	**0.030**	**0.043**
DecodeAF, $\alpha = 3$	**65**	0.924	0.937	0.054	0.070
DecodeFaster, $\alpha = 3$	72	**0.945**	**0.960**	**0.034**	**0.047**
DecodeAF, $\alpha = 4$	**58**	0.763	0.335	0.663	0.670
DecodeFaster, $\alpha = 4$	65	**0.936**	**0.947**	**0.046**	**0.060**
DecodeAF, $\alpha = 5$	**54**	0.710	0.170	0.829	0.833
DecodeFaster, $\alpha = 5$	62	**0.916**	**0.887**	**0.106**	**0.122**

Table 1. ablation study by choosing different α

that best performed on the validation set. Using a single NVIDIA V100 GPU, training our model on the TuSimple dataset until convergence and takes 10 hours.

4.1. Dataset and Metrics

We trained and tested our model on the popular TuSimple dataset for lane detection. The TuSimple dataset for lane segmentation contain multiple situations: (1) Good and fair weather conditions; (2) Various daytime lighting conditions; (3) 2/3/4 or more lanes; (4) different traffic conditions. The training set combine 3626 video clips, each video is clipped into a image per second, 20 images in total for each video. Each video clips have corresponding annotations. The test set is made up of 2944 video clips, each images is of 1280×720.

The metric we uses are simply Accuracy, F1-score, False Negative, False Positive to measure model performance. Accuracy measures the predictability on pixels classification, F1-score measures the predictability on correct lane area classification and incorrect lane area misclassification prevention. False Negative measures the misclassification rate while False Positive measures the omission rate.

$$F1 = 2(\frac{precision \cdot recall}{precision + recall}) \tag{10}$$

4.2. Ablation Studies

To test if our decode faster algorithm does work, we conducted ablation study with decode AF with original size of segmentation mask, HAF, and VAF as the best quality achievable. Then we compare decode faster with decode AF on shrunk size of segmentaion mask, HAF and VAF, with respect to their decoding quality and running time. Quantitative results are shown in **Table** 1. As we can see from this table, decode faster does have shorter decoding time compared to decode AF on original size of input, and decoding quality deteriorates much slower than decode AF on the same input size. Especially when shrinkage ratio $\alpha \in \{2, 3\}$, the decoding quality is almost lossless. When $\alpha = 4$, the decoding quality is better than decoding AF when $\alpha = 3$ with less running time. When $\alpha = 5$, the decoding quality is still usable while decoding AF becomes unusable when $\alpha \geq 4$. Qualitative comparison is shown in **Figure** 6, it is obvious that our method is superior than decode AF on shrunk images.

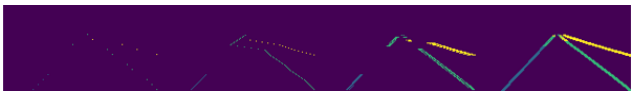

Figure 5. Decode faster lane forming process ($\alpha = 5$): (1) Get point anchors; (2) Connect; (3) Expand; (4) Assign unclassified pixels.

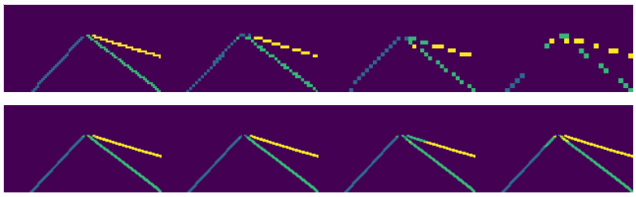

Figure 6. comparison between decode faster and decode AF for $\alpha \in \{2, 3, 4, 5\}$, row 1 are for decode AF results, row 2 are for decode faster results

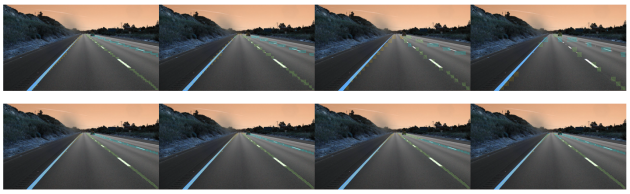

Figure 7. The output segmentation results.

5. Conclusion

We separate the lane detection into a two-stage pipeline, including semantic segmentation and clustering post-processing. We illustrate the decoding algorithm for predicted affinity fields, with a revised version to boost the efficiency. In the original paper the author simply process a downsampled input in the decoding process. In our method, we decode the AF skipping rows, and the lanes are all decently segmented even with large alpha. But there are problems for the far distance pixels, some of them fail to be correctly segmented.

References

[1] Hala Abualsaud, Sean Liu, David Lu, Kenny Situ, Akshay Rangesh, and Mohan M Trivedi. Laneaf: Robust multi-lane detection with affinity fields. *arXiv preprint arXiv:2103.12040*, 2021. 2

[2] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017. 2

[3] Noa Garnett, Rafi Cohen, Tomer Pe'er, Roee Lahav, and Dan Levi. 3d-lanenet: end-to-end 3d multiple lane detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2921–2930, 2019. 2

[4] Mohsen Ghafoorian, Cedric Nugteren, Nóra Baka, Olaf Booij, and Michael Hofmann. El-gan: Embedding loss driven generative adversarial networks for lane detection. In *proceedings of the european conference on computer vision (ECCV) Workshops*, pages 0–0, 2018. 2

[5] Tejus Gupta, Harshit S Sikchi, and Debashish Charkravarty. Robust lane detection using multiple features. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 1470–1475. IEEE, 2018. 2

[6] Yuhao Huang, Shitao Chen, Yu Chen, Zhiqiang Jian, and Nanning Zheng. Spatial-temproal based lane detection using deep learning. In *IFIP International conference on artificial Intelligence applications and innovations*, pages 143–154. Springer, 2018. 2

[7] Yeongmin Ko, Younkwan Lee, Shoaib Azam, Farzeen Munir, Moongu Jeon, and Witold Pedrycz. Key points estimation and point instance segmentation approach for lane detection. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 2

[8] Felix Măriut, Cristian Foşalău, and Daniel Petrisor. Lane mark detection using hough transform. In *2012 International Conference and Exposition on Electrical and Power Engineering*, pages 871–875. IEEE, 2012. 2

[9] Mirko Meuter, Stefan Muller-Schneiders, Adalbert Mika, Stephanie Hold, Christian Nunn, and Anton Kummert. A novel approach to lane detection and tracking. In *2009 12th International IEEE Conference on Intelligent Transportation Systems*, pages 1–6. IEEE, 2009. 2

[10] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 286–291. IEEE, 2018. 2

[11] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2

[12] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016. 2

[13] Fabio Pizzati, Marco Allodi, Alejandro Barrera, and Fernando García. Lane detection and classification using cascaded cnns. *arXiv preprint arXiv:1907.01294*, 2019. 2

[14] Ravi Kumar Satzoda and Mohan Manubhai Trivedi. Drive analysis using vehicle dynamics and vision-based lane semantics. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):9–18, 2014. 2

[15] Sukriti Srivastava, Ritika Singal, and Manisha Lumba. Efficient lane detection algorithm using different filtering techniques. *International Journal of Computer Applications*, 88(3), 2014. 2

[16] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. Keep your eyes on the lane: Real-time attention-guided lane detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 294–302, 2021. 2

[17] Yue Wang, Eam Khwang Teoh, and Dinggang Shen. Lane detection and tracking using b-snake. *Image and Vision computing*, 22(4):269–280, 2004. 2

[18] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2018. 1

[19] Qin Zou, Hanwen Jiang, Qiyu Dai, Yuanhao Yue, Long Chen, and Qian Wang. Robust lane detection from continuous driving scenes using deep neural networks. *IEEE transactions on vehicular technology*, 69(1):41–54, 2019. 2