LaneAF: Robust Multi-Lane Detection with Affinity Fields

Hala Abualsaud[1†], Sean Liu[1†], David B. Lu[1†], Kenny Situ[1†], Akshay Rangesh[1†], and Mohan M. Trivedi[1]

***Abstract*—This study presents an approach to lane detection involving the prediction of binary segmentation masks and per-pixel affinity fields. These affinity fields, along with the binary masks, can then be used to cluster lane pixels horizontally and vertically into corresponding lane instances in a post-processing step. This clustering is achieved through a simple row-by-row decoding process with little overhead; such an approach allows LaneAF to detect a variable number of lanes without assuming a fixed or maximum number of lanes. Moreover, this form of clustering is more interpretable in comparison to previous visual clustering approaches, and can be analyzed to identify and correct sources of error. Qualitative and quantitative results obtained on popular lane detection datasets demonstrate the model's ability to detect and cluster lanes effectively and robustly. Our proposed approach sets a new state-of-the-art on the challenging CULane dataset and the recently introduced Unsupervised LLAMAS dataset.**

***Index Terms*—Object Detection, Segmentation and Categorization, Deep Learning for Visual Perception**

I. Introduction

LANE detection is the process of automatically perceiving the shape and position of marked lanes and is a crucial component of autonomous driving systems, directly influencing the guidance and steering of vehicles while also aiding the interaction between numerous agents on the road. As the number of drivers on the roads has increased, autonomous driving systems have received considerable attention in the automotive and tech industries as well as in academia [1]. According to the Insurance Institute for Highway Safety (IIHS), in the US alone, car accidents claimed 36,560 lives in 2018, underscoring the importance of any technology that can help prevent crashes.

Since roads commonly have different types of lane lines (solid white, broken white, solid yellow, etc.), each of which have specific implications with regards to how vehicles may interact with them, automated lane detection systems can also help alert drivers when there are changes in lane topology on the road. Furthermore, there are several factors that make lane detection a challenging task. Firstly, there is a wide variety of road infrastructure in use around the world. Additionally, the lane detection system must be able to identify instances where lanes are ending, merging, and splitting. Finally, the lane detection system must possess the ability to discern worn or unclear lane markings. Precise detection of lanes also enable more robust trajectory prediction of surrounding vehicles; as discussed in [2], this is critical for successful path planning in autonomous driving. Therefore, while lane detection is a significant and complex task, it is a key factor in developing any autonomous vehicle system.

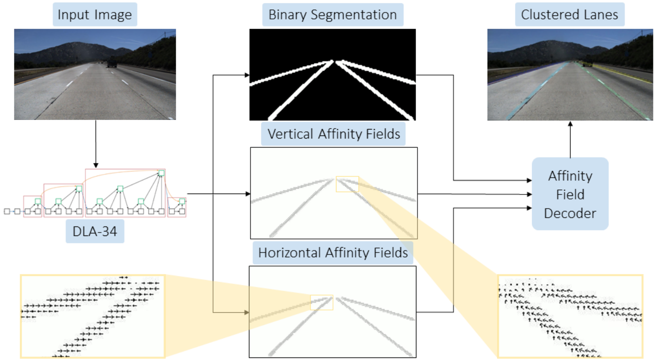


Fig. 1: In our approach, we propose to train a model that outputs binary segmentation masks and affinity fields, which can then be decoded together to produce multiple lane instances. This is opposed to the standard approach to (anchor-free) lane detection that treats each lane as a separate class and trains a model to perform multi-class segmentation.

While binary classification is used for the detection of lanes in our approach, a limitation of this type of classification is that it produces a single-channel output, which does not allow for the identification of separate lane entities. To dissociate different lane instances, we propose a novel clustering scheme based on affinity fields (see Figure 1). Affinity fields were originally introduced in [3] for the purpose of multi-person 2D pose estimation, and are comprised of unit vectors that encode location and orientation. This technique was also used for the detection of hands inside a vehicle, as demonstrated in [4]. In this paper, we have defined two types of affinity fields, the horizontal affinity field (HAF) and vertical affinity field (VAF). It is these affinity fields that enable unique lane instances to be identified and segmented. Since these affinity fields are present wherever there are foreground lane pixels, they are not bound to a pre-determined number of lanes. The model is therefore agnostic to the number of lanes present on the road.

The main contributions of this paper are as follows:

Manuscript received: March 22, 2021; Revised June 15, 2021; Accepted July 11, 2021.

This paper was recommended for publication by Editor Cesar Cadena upon evaluation of the Associate Editor and Reviewers' comments.

[1]The authors are with Laboratory for Intelligent and Safe Automobiles, UC San Diego, CA 92092, USA {habualsa, sel118, dblu, ksitu, arangesh, mtrivedi}@ucsd.edu

† Equal contribution

Code: https://github.com/sel118/LaneAF

Digital Object Identifier (DOI): see top of this page.

1) We show that using an off-the-shelf convolutional neural network (CNN) backbone [5] that intrinsically aggregates and refines multi-scale features can result in superior performance when compared to other bespoke architectures and losses previously proposed for lane detection.
2) We propose affinity fields that are suitable for clustering and associating pixels belonging to amorphous entities like lanes.
3) We detail the procedure and losses to train models that predict binary segmentation masks and affinity fields for the purpose of lane instance segmentation.
4) We introduce efficient methods for generating and decoding such affinity fields into an unknown number of clustered lane instances.

II. Related Research

Lane detection has traditionally been tackled by feature-based approaches [6] which then evolved to model-based approaches to detect lane boundaries. However, these are not practical in real world scenarios since they require ideal road scenes to work effectively. Currently, data-driven approaches are commonly used to detect both lane boundaries as well as lane regions. While several shortcomings of the traditional lane detection methods (i.e. lane segmentation via hand-crafted features) have been resolved with more robust methods in recent years, there is still room for improvement. In more recent times, deep learning and large-scale datasets have provided solutions to many of these issues. However, lane detection in unconstrained environments and complex scenarios remain a challenge.

Lane detection nowadays is typically modelled as a semantic segmentation problem to extract features using deep learning methods. New approaches tackle lane detection as a multi-class segmentation problem, where each lane forms a separate class. Some of these approaches include: [7], [8], [9], [10], [11], and [12]. In [9], the authors combine a recurrent neural network (RNN) with a CNN for lane prediction and detection. The use of an embedding loss was introduced in [10] which uses generative adversarial networks (GANs) to better preserve the structure of lanes and to mitigate the problem of complex post-processing for the output of semantic segmentation; 96% accuracy on the TuSimple dataset was obtained. In [12], a sequential prediction network has been used to avoid heuristic-based clustering post-processing. Another network architecture was presented in [13] with two elements: a deep network which generates weighted pixel coordinates in addition to a differentiable weighted least-squares fitting module. In [14], the authors introduced Self Attention Distillation (SAD) loss to avoid models that propagate data sequentially and to decrease inference time. However, the fully connected layer that the SAD model employs is computationally expensive and cannot adapt to any number of lanes.

Other lane detection approaches choose to first perform binary segmentation of all lanes, followed by a clustering stage to separate each individual lane instance as in [15], [16], and [17]. Instance segmentation is usually approached with the use of complex pipelines; however, many powerful approaches and research were put to come up with better performance techniques including the approach presented in [18], where they used an end-to-end convolutional neural network to tackle the problem that was inspired by the classical watershed transform. Another method toward instance segmentation was based on using a fully convolutional network to predict semantic labels along with depth and an instance-based encoding. This was implemented by using each pixel's direction toward its corresponding instance center; with the help of low-level computer vision techniques, impressive scene understanding by predicting pixel-wise depth, semantics, and instance-level direction cues was achieved [19]. Lane detection is posed as an instance segmentation problem in [15] so that each lane can be detected in an end-to-end manner, adapting to changing numbers of lanes on the road. In [16], a combination of instance segmentation and classification was used as an end-to-end deep learning real-time method to avoid reliance on two-step detection networks. Although recent methods of lane detection show high accuracy when applied to the popular published datasets, some drawbacks of these current methods are that they are not robust when encountering occlusion and that they require a fixed number of lanes in a scene; thus, they cannot work for a random number of lanes present on the road. Acknowledging this problem in [17], the authors use a key points estimation approach to allow for lane detection of an arbitrary numbers of lanes regardless of orientation.

More recently, some approaches have modelled lane detection as an anchor-based object detection problem such as [20], [21], [22], [23], and [6]. In [23], a spatio-temporal deep learning method was proposed to mitigate the errors that can occur when experiencing harsh weather or other complex problems in the road, jeopardizing the accuracy of detecting a lane in the scene. Meanwhile, in [20], lane markers were tracked temporally. Additionally, [22] presents an anchor-based single-stage deep lane detection model using anchors for feature pooling. In [21], the authors developed 3D-LaneNet, a network that predicts the 3D layout of lanes using a single image. A combination of LiDAR and camera sensors were used in [24] for their network to obtain accurate lane detection in 3D space directly.

III. Methodology

Our proposed methodology involves a feed-forward CNN that is trained to predict binary lane segmentation masks and per-pixel affinity fields. More specifically, the model is trained to predict two affinity fields, which we call the horizontal affinity field (HAF) and vertical affinity field (VAF), respectively. Affinity fields can be thought of as vector fields that map any 2D location on the image plane to a unit vector in 2D. A unit vector in the VAF encodes the direction in which the next set of lane pixels above it is located. On the other hand, a unit vector in the HAF points toward the center of the lane in the current row, thereby allowing us to cluster lanes of arbitrary widths. These two affinity fields, in conjunction with the predicted binary segmentation, can then be used to cluster foreground pixels into lanes as a post-processing step. In the next few subsections, we discuss each individual block in our proposed approach.

Algorithm 1 Creating affinity fields from ground truth data

```
Inputs:
    $SEG(H \times W)$: ground truth segmentation
    $l_{max}$: maximum number of lanes

$HAF, VAF \leftarrow zeros(H, W, 2)$            ▷ initialize affinity fields
for $l \leftarrow 1$ to $L$ do                   ▷ go through each lane
    $prev\_cols \leftarrow nonzero(SEG[H, :] == l)$       ▷ initialize
    /* row-by-row, from bottom to top */
    for $y \leftarrow H - 1$ to $1$ do
        $cols \leftarrow find(SEG[row, :] == l)$      ▷ find lane pixels
        /* horizontal affinity field */
        for $x$ in $cols$ do
            $HAF[y, x] \leftarrow \vec{H}_{gt}(x, y)$            ▷ Eq. 1
        end for
        /* vertical affinity field */
        for $x$ in $prev\_cols$ do
            $VAF[y + 1, x] \leftarrow \vec{V}_{gt}(x, y + 1)$    ▷ Eq. 2
        end for
        $prev\_cols \leftarrow cols$
    end for
end for
return $HAF, VAF$
```

A. Network Backbone

Recent lane detection approaches have made use of a variety of backbone architectures, but most popular among them are usually the ResNet family of architectures [25], ENet [7], and ERFNet [26]. Although these architectures have proven benefits across a variety of tasks, we believe that more recent developments in the field can be leveraged for lane detection. To this end, we make use of the DLA-34 backbone presented in [5].

The DLA family of models make use of deep layer aggregation, which unifies semantic and spatial fusion for better localization and semantic interpretation. In particular, this architecture extends densely connected networks [27] and feature pyramid networks with hierarchical and iterative skip connections that deepen the representation and refine resolution. They employ two forms of aggregation: iterative deep aggregation (IDA), focusing on fusing resolutions and scales, and hierarchical deep aggregation (HDA), focusing on merging features from all modules and channels. These architectures also incorporate deformable convolution operations [28] that can adapt the spatial sampling grid for convolutions based on their inputs. We believe these are desirable properties for the tasks of lane detection and instance segmentation.

B. Affinity Fields

In addition to binary lane segmentation masks, our model is trained to predict horizontal and vertical affinity fields (HAFs and VAFs respectively). For any given image, the HAF and VAF can be thought of as vector fields $\vec{H}(\cdot, \cdot)$ and $\vec{V}(\cdot, \cdot)$, that assign a unit vector to each (x, y) location in the image. As we alluded to earlier, the HAF enables us to cluster lane pixels horizontally and the VAF vertically. With the predicted affinity fields and binary mask, clustering lane pixels is achieved through a simple row-by-row decoding process from bottom to top. The rest of this subsection provides details on how to create such affinity fields using the ground truth and how to use the predicted affinity fields to decode individual lanes.

Algorithm 2 Decoding predicted affinity fields into lanes

```
Inputs:
    $BW(H \times W)$: binary segmentation mask
    $HAF(H \times W \times 2)$: horizontal affinity field
    $VAF(H \times W \times 2)$: vertical affinity field
    $\tau$: clustering threshold

$SEG \leftarrow zeros(H, W)$             ▷ initialize segmentation output
$lane\_end\_points \leftarrow []$
                  ▷ keeps track of the latest points added to each lane
$L \leftarrow 0$                       ▷ initialize number of lanes to 0
/* row-by-row, from bottom to top */
for $y \leftarrow H$ to $1$ do
    $cols \leftarrow find(BW[row, :] > 0)$     ▷ find foreground pixels
    /* cluster horizontally */
    $clusters \leftarrow []$
    for $x$ in $cols$ do
        $clusters.update(c^*_{haf}(x, y))$                ▷ Eq. 3
    end for
    /* assign clusters to existing lanes */
    for $l \leftarrow 1$ to $L$ do
        if $d^*(l) <= \tau$ then      ▷ error less than threshold (Eq. 5)
            $lane\_end\_points[l] \leftarrow c^*_{vaf}(l)$        ▷ Eq. 4,Eq. 6
                                  ▷ update latest points added to lane
            for $x$ in $c^*_{vaf}$(l) do
                $SEG[y, x] \leftarrow l$            ▷ assign cluster to lane $l$
            end for
        end if
    end for
    /* spawn new lanes with unassigned clusters */
    for $cluster$ in $clusters$ do
        if $cluster$ is not assigned then
            $L \leftarrow L + 1$
            $lane\_end\_points[L] \leftarrow cluster$
        end if
    end for
end for
return $SEG$
```

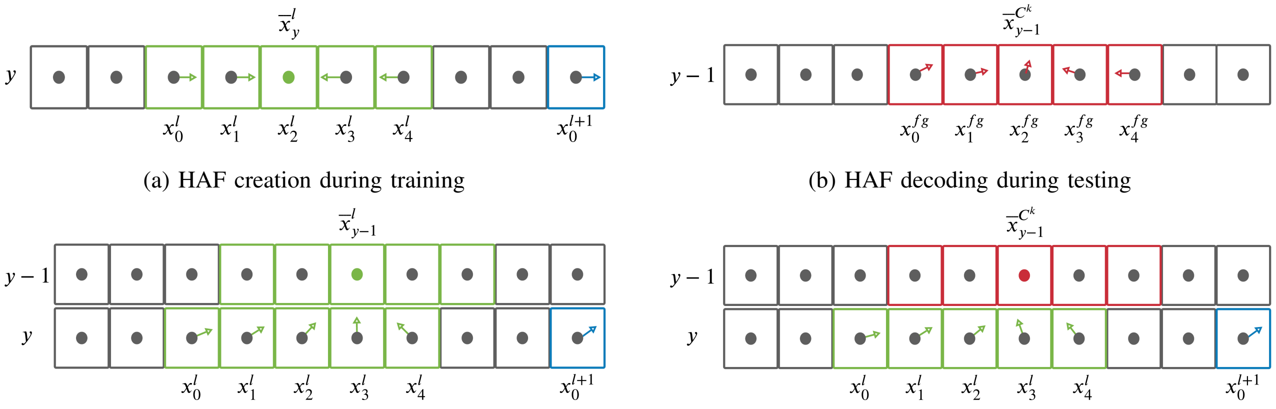
Creating HAFs and VAFs: Affinity fields are created using ground truth segmentation masks on the fly as detailed in Algorithm 1. This proceeds row-by-row, from bottom to top.

For any row y in the image, the HAF vectors are computed for each lane point (x^l_i, y) using the ground truth vector field mapping $\vec{H}_{gt}(\cdot, \cdot)$ as follows:

$$\begin{aligned} \vec{H}_{gt}(x^l_i, y) &= \left(\frac{\overline{x}^l_y - x^l_i}{|\overline{x}^l_y - x^l_i|}, \frac{y - y}{|y - y|} \right)^{\intercal} \\ &= \left(\frac{\overline{x}^l_y - x^l_i}{|\overline{x}^l_y - x^l_i|}, 0 \right)^{\intercal}, \end{aligned} \tag{1}$$

where $\overline{x}^l_y$ is the mean x-coordinate of all points belonging to lane l in row y. This process is illustrated in Figure 2a, where pixels in green and blue represent points belonging to lanes l and $l + 1$ respectively.

Similarly, the VAF vectors are computed for each lane point (x^l_i, y) in row y using the ground truth vector field mapping



(a) HAF creation during training

(b) HAF decoding during testing

(c) VAF creation during training

(d) VAF decoding during testing

Fig. 2: Illustrations of HAF and VAF creation and decoding processes during training and testing respectively.

$\vec{V}_{gt}(\cdot,\cdot)$ as follows:

$$\begin{aligned}\vec{V}_{gt}(x_i^l, y) &= \left(\frac{\overline{x}_{y-1}^l - x_i^l}{|\overline{x}_{y-1}^l - x_i^l|}, \frac{y-1-y}{|y-1-y|}\right)^\intercal \\ &= \left(\frac{\overline{x}_{y-1}^l - x_i^l}{|\overline{x}_{y-1}^l - x_i^l|}, -1\right)^\intercal,\end{aligned} \qquad (2)$$

where $\overline{x}_{y-1}^l$ is the mean x-coordinate of all points belonging to lane l in row $y-1$. This process is illustrated in Figure 2c, where pixels in green represent points belonging to lanes l. Note that unlike the HAF, unit vectors in the VAF point to the mean location of the lane in the previous row.

Decoding HAFs and VAFs: After a model is trained to predict the HAFs and VAFs detailed above, a decoding procedure is carried out to cluster foreground pixels into lanes during testing. This procedure is presented in Algorithm 2, and similarly operates row-by-row, from bottom to top.

Assuming $\vec{H}_{pred}(\cdot,\cdot)$ is the vector field corresponding to the predicted HAF, foreground pixels in a row $y-1$ are first assigned to clusters based on the following rule:

$$c_{haf}^*(x_i^{fg}, y-1) = \begin{cases} C^{k+1} & \text{if} \quad \vec{H}_{pred}(x_{i-1}^{fg}, y-1)_0 \leq 0 \\ & \quad \wedge \vec{H}_{pred}(x_i^{fg}, y-1)_0 > 0, \\ C^k & \text{otherwise}, \end{cases} \qquad (3)$$

where $c_{haf}^*(x_i^{fg}, y-1)$ denotes the optimal cluster assignment for a foreground pixel $(x_i^{fg}, y-1)$; C^k and C^{k+1} denote two different clusters indexed by k and $k+1$ respectively. This assignment is illustrated in Figure 2b, where pixels in red are assigned the same cluster.

Next, these horizontal clusters are assigned to existing lanes indexed by l using the vector field $\vec{V}_{pred}(\cdot,\cdot)$ corresponding to the VAF as follows:

$$c_{vaf}^*(l) = \arg\min_{C^k} d^{C^k}(l), \qquad (4)$$

where

$$d^*(l) = \min_{C^k} d^{C^k}(l). \qquad (5)$$

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

$$\begin{aligned} d^{C^k}(l) = & \frac{1}{N_y^l} \sum_{i=0}^{N_y^l - 1} \Bigg\| (\overline{x}^{C^k}, y-1)^\intercal - (x_i^l, y)^\intercal \\ & - \vec{V}_{pred}(x_i^l, y) \cdot ||(\overline{x}^{C^k}, y-1)^\intercal - (x_i^l, y)^\intercal|| \Bigg\|, \end{aligned} \qquad (6)$$

where N_y^l are the number of pixels belonging to lane l in row y. We illustrate this process in Figure 2d, where the cluster in red is assigned to the existing lane in green. By repeating the above steps row-by-row starting from the bottom and working to the top, we are able to assign every foreground pixel to their respective lanes.

C. Losses

To train the proposed model, we use a separate loss at each prediction head. For our binary segmentation branch, we used weighted binary cross-entropy loss, a standard loss for imbalanced binary segmentation tasks. The raw logits produced by the model are first passed through a sigmoid activation for normalization. The loss is then calculated as:

$$L_{BCE} = -\frac{1}{N} \sum_i \Big[w \cdot t_i \cdot log(o_i) + (1-t_i) \cdot log(1-o_i) \Big], \qquad (7)$$

where t_i is the target value for the pixel i and o_i is the sigmoid output. Since this is an unbalanced segmentation task, a weight w was used to increase penalization for foreground pixels. To further account for the imbalanced dataset, an additional intersection over union loss was used for the segmentation branch:

$$L_{IoU} = \frac{1}{N} \sum_i \left[1 - \frac{t_i \cdot o_i}{t_i + o_i - t_i \cdot o_i} \right]. \qquad (8)$$

For the affinity field branches of the model, a simple $L1$ regression loss was applied only to the foreground locations of both the vertical and horizontal affinity fields:

$$L_{AF} = \frac{1}{N_{fg}} \sum_i \Big[|t_i^{haf} - o_i^{haf}| + |t_i^{vaf} - o_i^{vaf}| \Big]. \qquad (9)$$

The total loss applied to the model is a simple summation of the individual losses:

$$L_{total} = L_{BCE} + L_{IoU} + L_{AF}. \tag{10}$$

IV. Experimental Evaluation

A. Implementation Details

Our backbone architecture (DLA-34) is a fully convolutional network that does not retain the original resolution, but rather downsizes the outputs by a factor of 4; thus, we rescaled the input images to one-half their original resolution during run-time and reshaped the ground truth affinity fields and segmentation masks to one-eighth the original resolution (accounting for the model's downsizing factor). This has the added benefit of making our decoding process faster since we now process only an eighth of the original rows. The decoding time typically depends on the number of lanes, the quality of the outputs produced by the model, and the output size. On average, it takes about 15-20ms on a modern CPU without any code optimizations. However, since this an entirely CPU-based operation, it should not affect the overall latency of the approach. We also make use of random rotations, crops, scales and horizontal flips during training.

We use the Adam optimizer as our solver with a learning rate of 0.0001, weight decay of 0.001, and train for a total of 40 epochs. We also employ a scheduler that reduces the learning rate by a factor of 5 every 10 epochs. The weight w for the loss in Eq. 7 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 that best performed on the validation set. Using a single GTX Titan X Maxwell GPU, training our model on the CULane dataset until convergence (about 25-30 epochs) takes 2-3 days. Significant speedup can be obtained by using more modern GPUs and by employing multiple GPUs when available.

B. Datasets

To train and benchmark our proposed approach, we make use of the popular TuSimple, CULane [11], and LLAMAS [29] datasets. TuSimple features good and fair weather conditions in various daytime lighting and traffic conditions, employing highways with up to five lanes. Meanwhile, CULane contains significantly more data and also divides test images into nine categories that contain more complex scenarios, including images with challenging lighting conditions. Finally, the LLAMAS dataset is a newer dataset with a sizeable amount of images all obtained using highway recordings and generated from an automated labeling pipeline. A summary of all datasets is compiled in Table I.

TABLE I: Attributes of popular lane detection datasets

Dataset	TuSimple	CULane	LLAMAS
# Frames	6,408	133,325	100,042
Train	3,268	88,880	58,269
Validation	358	9,675	20,844
Test	2,782	34,680	20,929
Resolution	1280 × 720	1640 × 590	1280 × 717
Road Type	highway	urban, rural, highway	highway

C. Metrics

We use the same evaluation metrics used in past literature to make a representative comparison between our approach and prior work. This consists of the official metric of the TuSimple dataset (accuracy), the false positive (FP) rate, and the false negative (FN) rate. The TuSimple accuracy is calculated as:

$$Accuracy = \frac{N_{pred}}{N_{gt}} \tag{11}$$

where N_{pred} is the number of lane points that have been correctly predicted and N_{gt} is the number of ground-truth lane points.

Additionally, we report the $F1$ measure, which is based on the intersection over union (IoU) and is the only metric for CULane. This is calculated as in [11]:

$$F1 = 2 \cdot \left(\frac{precision \cdot recall}{precision + recall} \right) \tag{12}$$

where $precision$ is defined as $\frac{TP}{TP+FP}$, $recall$ is defined as $\frac{TP}{TP+FN}$, TP is the number of lane points that have been correctly predicted, FP is the number of false positives, and FN is the number of false negatives. This same $F1$ measure is also used for the lane approximations benchmark of the LLAMAS dataset.

D. Ablation Experiments

In this subsection, we conduct a series of ablation experiments to validate our design choices. All ablation studies were conducted on the TuSimple validation set and can be seen in Table II. The first row contains the results of the standard LaneAF model, which we denote as the baseline model B. First, we train variants without the IoU loss (B w/o IoU) and the weighted binary cross-entropy loss (B w/o wBCE). Removing these losses decreased accuracy quite drastically while increasing the false positive and false negative rate. In fact, without the weighted binary cross-entropy loss, the F1 score in particular dropped significantly. The same is observed for the baseline model without random transformations during training (B w/o RT), as depicted in the fourth row.

With regards to the down-sampling factor of the outputs, it is clear that the baseline model's factor of 4 achieved the best results; decreasing it to 2 (B (DS-2)) increased runtime and worsened accuracy and F1 slightly, while increasing it to 8 (B (DS-8)) had the most damaging effect on accuracy out of all modifications. We also trained a variant with 128 channels in the output head (B (HC-128)) compared to the original 256, and while this change had the smallest impact with respect to accuracy, it is evident that the baseline's 256 channels yields superior results. Finally, to validate the benefits of our clustering approach over standard multi-class segmentation, we trained a DLA-34 model to directly perform multi-class segmentation of all lanes (DLA-34 multi-class). This model obtained the worst F1 and accuracy scores out of

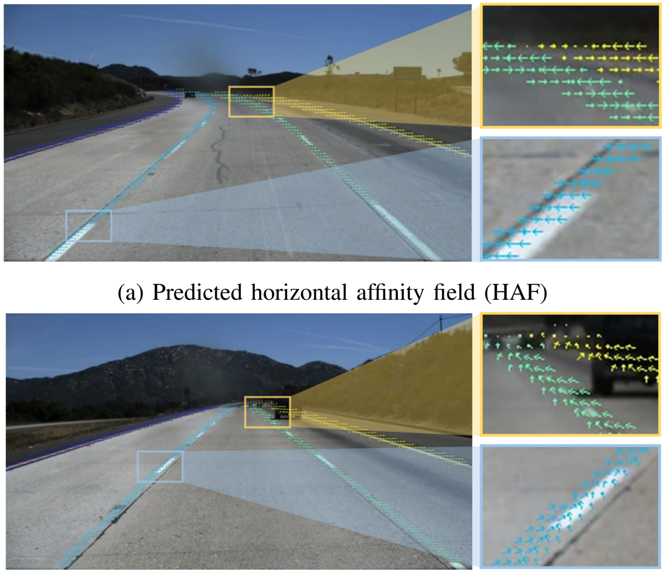
TABLE II: LaneAF ablation experiments on the TuSimple validation set

Model type	F1 (%)	Acc (%)	FP	FN
B[1]	95.31	94.62	0.0435	0.0500
B w/o IoU[2]	95.17	94.31	0.0456	0.0507
B w/o wBCE[3]	93.75	94.19	0.0598	0.0649
B w/o RT[4]	93.56	94.29	0.0614	0.0670
B (DS-2)[5]	94.76	94.22	0.0484	0.0559
B (DS-8)	93.94	92.73	0.0549	0.0656
B (HC-128)[6]	94.80	94.56	0.0503	0.0535
DLA-34 multi-class[7]	88.86	92.59	0.1115	0.1114

[1] B: baseline model with down-sampling factor 4 and 256 channels in output head [2] IoU: IoU loss [3] wBCE: weighted BCE loss [4] RT: random transformations during training
[5] DS-x: down-sampling factor for outputs [6] HC-x: number of channels in output head [7] DLA-34 multi-class: DLA-34 model trained for lane detection via multi-class segmentation

TABLE III: LaneAF results on the TuSimple benchmark

Method	F1 (%)	Acc (%)	FP	FN	MACs (G)
ResNet-18 [25]	87.87	92.69	0.0948	0.0822	-
PolyLaneNet [30]	90.62	93.36	0.0942	0.0933	**1.7**
Cascaded-CNN [16]	90.82	95.24	0.1197	0.0620	-
LaneNet [15]	94.80	96.38	0.0780	0.0244	-
ENet-SAD [14]	95.92	96.64	0.0602	0.0205	-
SCNN [11]	96.53	96.53	0.0617	**0.0180**	-
ResNet-34 [25]	96.77	92.84	0.0918	0.0796	-
PINet [17]	**97.20**	**96.75**	0.0310	0.0250	-
LaneATT(ResNet-18) [22]	96.71	95.57	0.0356	0.0301	9.3
LaneATT (ResNet-34) [22]	96.77	95.63	0.0353	0.0292	18.0
LaneATT (ResNet-122) [22]	96.06	96.10	0.0564	0.0217	70.5
LaneAF (DLA-34)	96.49	95.62	**0.0280**	0.0418	22.2


(a) Predicted horizontal affinity field (HAF)

(b) Predicted vertical affinity field (VAF)

Fig. 3: Example outputs produced by LaneAF with color coded affinity fields; each color represents a unique lane instance based on affinity field decoding. Of note is the successful discrimination of lane instances even as the lanes converge.

all variants. This result clearly illustrates the effectiveness of binary segmentation followed by a separate affinity field-based clustering approach.

E. Results

Performance results from LaneAF on the TuSimple benchmark are shown in Table III. It can be seen that our false positive rate sets a new standard (0.0280) among the current state-of-the-art. This demonstrates that our model does not incorrectly detect a lane pixel as often as other networks and that LaneAF's multi-branch approach leads to confident lane pixel predictions. While we obtain superior accuracy to other backbone architectures such as ResNet-18 and -34 [25], our approach falls slightly short of current state-of-the-art models such as PINet [17], ENet-SAD [14], and SCNN [11]. However, our false negative rate is only marginally higher, signifying that the incorrectly classified lane pixels are most likely at the very ends of the lanes. Additionally, six training runs were conducted on this dataset with different random seeds, producing a standard deviation of 0.12 on the accuracy metric. From the consistency of these results, we can see that our proposed method is robust.

Table IV displays the state-of-the-art results of our model on the CULane benchmark. With this significantly larger and more complex dataset, we can see that LaneAF's performance improves greatly with respect to other models and demonstrates our network's ability to generalize. LaneAF (with DLA-34) outperforms the current state-of-the-art with an F1 score of 77.41%, surpassing models of similar size and even LaneATT [22] with its largest backbone, ResNet-122. Moreover, LaneAF sets a new benchmark in a majority of categories, including difficult ones such as Dazzle, Shadow, No line, Curve, and Night, exhibiting our model's high adaptability to curving roads and challenging lighting conditions.

For the CULane dataset, we additionally trained LaneAF models with ENet [7] and ERFNet [26] backbones, where we forego the final few upsampling/transposed convolution layers to ensure a downsampling factor of 4 (same as the DLA-34 variant). This allows us to make direct comparisons between our approach and other approaches using the same backbone architecture. For instance, LaneAF with the ENet backbone outperforms ENet-SAD [14] by over 3% with respect to the F1 score. When using ERFNet as the backbone network, LaneAF's F1 score eclipses other ERFNet-based models such as ERFNet-E2E [34] and ERFNet-Intra-KD [31] by 1.63% and 3.23%, respectively. These comparisons confirm that the performance gains of LaneAF are achieved through a combination of the DLA-34 backbone and our proposed affinity fields based clustering.

Furthermore, LaneAF again achieves state-of-the-art performance on the LLAMAS dataset with an F1 score of 96.07%. This surpasses LaneATT's [22] best model by over 2%, as shown in Table V. This gap in performance is due to LaneAF's high Recall score, which indicates that the model is more adept at retrieving true lane pixels.

Qualitative examples of the predicted affinity fields from our approach are depicted in Figures 3a and 3b. The clustered outputs shown here were created using the affinity field decoder, outlined in Algorithm 2. In Figure 3a, the HAF vectors point towards the center of their respective lane lines for each

TABLE IV: LaneAF state-of-the-art results on the CULane benchmark

Method	Total	Normal	Crowded	Dazzle	Shadow	No line	Arrow	Curve	Cross	Night	MACs (G)
ResNet-18 [25]	68.40	87.70	66.00	58.40	62.80	40.20	81.00	57.90	1743	62.10	-
ENet-SAD [14]	70.80	90.10	68.80	60.20	65.90	41.60	84.00	65.70	1998	66.00	-
SCNN [11]	71.60	90.60	69.70	58.50	66.90	43.40	84.10	64.40	1990	66.10	-
ResNet-34 [25]	72.30	90.70	70.20	59.50	69.30	44.40	85.70	69.50	2037	66.70	-
ERFNet-Intra-KD [31]	72.40	-	-	-	-	-	-	-	-	-	-
CurveLanes-NAS-M [32]	73.50	90.20	70.50	65.90	69.30	48.80	85.70	67.50	2359	68.20	33.7
SIM-CycleGAN [33]	73.90	91.80	71.80	66.40	76.20	46.10	87.80	67.10	2346	69.40	-
ERFNet-E2E [34]	74.00	91.00	73.10	64.50	74.10	46.60	85.80	71.90	2022	67.90	-
PINet [17]	74.40	90.30	72.30	66.30	68.40	49.80	83.70	65.60	1427	67.70	-
LaneATT (ResNet-18) [22]	75.09	91.11	72.96	65.72	70.91	48.35	85.49	63.37	**1170**	68.95	9.3
RESA-50 [35]	75.30	92.10	73.10	69.20	72.80	47.70	88.30	70.30	1503	69.9	-
LaneATT (ResNet-34) [22]	76.68	**92.14**	75.03	66.47	78.15	49.39	**88.38**	67.72	1330	70.72	18.0
LaneATT (ResNet-122) [22]	77.02	91.74	**76.16**	69.47	76.31	50.46	86.29	64.05	1264	70.81	70.5
LaneAF (ENet)	74.24	90.12	72.19	68.70	76.34	49.13	85.13	64.40	1934	68.67	**2.2**
LaneAF (ERFNet)	75.63	91.10	73.32	69.71	75.81	50.62	86.86	65.02	1844	70.90	22.2
LaneAF (DLA-34)	**77.41**	91.80	75.61	**71.78**	**79.12**	**51.38**	86.88	**72.70**	1360	**73.03**	23.6

TABLE V: LaneAF results on the LLAMAS benchmark

Method	F1 (%)	Prec (%)	Recall (%)
PolyLaneNet [30]	88.40	88.87	87.93
LaneATT(ResNet-18) [22]	93.46	**96.92**	90.24
LaneATT (ResNet-34) [22]	93.74	96.79	90.88
LaneATT (ResNet-122) [22]	93.54	96.82	90.47
LaneAF (DLA-34)	**96.07**	96.91	**95.26**

row of the output image. Lane clusters are still successfully separated despite being closely located for numerous rows, demonstrated in yellow box of Figure 3a. Likewise, in Figure 3b, the VAF vectors point along the lane towards the mean location of the next row's lane pixels. This is visualized in the yellow box of Figure 3b, where for each unique lane instance, the unit vector points towards the next row's mean lane pixel location. For both Figures 3a and 3b, the blue boxes clearly display how the HAF and VAF are implemented for a single detected lane instance.

Another key point to note is the accuracy of the model at lane points that are farther away from the camera. Since the ground truth segmentation masks for each lane are of approximately the same thickness in the image plane from top to bottom, the model is trained to predict thick foreground masks for lane points even if they are farther away. This results in little to no degradation for lane points that are far away. However, at the horizon, some clusters will be occasionally assigned to non-optimal lanes due to the close proximity of the lane lines.

In Figure 4, we show additional qualitative results from the TuSimple dataset (row 1), the CULane dataset (rows 2-4), and the LLAMAS dataset (row 4). The TuSimple examples demonstrate LaneAF's high performance on curved highways and on lanes that are merging and splitting due to entrances and exits, highlighting our model's flexibility to the number of lanes present on a given road. Also notable in the middle image of the first row is the false detection of a lane line due to an airplane contrail. The displayed results from CULane include challenging scenarios that illustrate LaneAF's robustness on curved roads and in very poor lighting conditions. This diverse set of examples exhibit characteristics of the Dazzle, Shadow, Curve, and Night categories of the CULane dataset. Finally, the LLAMAS samples show excellent performance on additional highway scenes similar to the TuSimple examples.

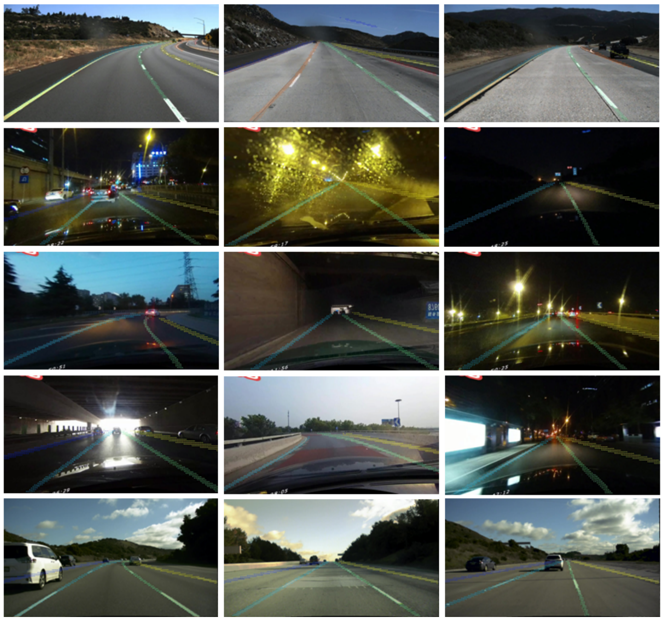


Fig. 4: LaneAF qualitative results on TuSimple (row 1), CULane (rows 2-4), and LLAMAS (row 5).

V. Concluding Remarks

In this paper, we proposed a novel approach to lane detection and instance segmentation through the use of binary segmentation masks and per-pixel affinity fields. The horizontal and vertical affinity fields, along with the predicted binary masks were demonstrated to successfully cluster lane pixels into unique lane instances in a post-processing step. This is accomplished using a simple row-by-row decoding process with little overhead, and enables LaneAF to detect a variable number of lanes of arbitrary width without assuming a fixed or maximum number of lanes. This form of clustering is also more interpretable in comparison to previous visual approaches since it can be analyzed to easily identify and correct sources of error. The ablation study conducted also validated the effectiveness of the approach over standard multi-class segmentation. Our proposed method achieves the lowest reported false positive rate (0.0280) on the TuSimple benchmark; on the larger and more comprehensive CULane dataset, LaneAF sets a new state-of-the-art result with a total F1 score of 77.41%, surpassing much deeper and more complex models. LaneAF also achieves a state-of-the-art F1 score on the LLAMAS benchmark by a significant margin (+2%), underscoring its robust performance.

VI. Acknowledgements

We are grateful to the reviewers for their valuable comments. We also thank our sponsors and colleagues at LISA, UC San Diego for their support.

References

[1] M. Daily, S. Medasani, R. Behringer, and M. Trivedi, "Self-driving cars," *Computer*, vol. 50, no. 12, pp. 18–23, 2017.

[2] N. Deo, A. Rangesh, and M. M. Trivedi, "How would surround vehicles move? a unified framework for maneuver classification and motion prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, pp. 129–140, 2018.

[3] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299.

[4] K. Yuen and M. M. Trivedi, "Looking at hands in autonomous vehicles: A convnet approach using part affinity fields," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 3, pp. 361–371, 2019.

[5] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2403–2412.

[6] R. K. Satzoda and M. M. Trivedi, "On enhancing lane estimation using contextual cues," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 11, pp. 1870–1881, 2015.

[7] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv e-prints*, pp. arXiv–1606, 2016.

[8] R. K. Satzoda and M. M. Trivedi, "Drive analysis using vehicle dynamics and vision-based lane semantics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 9–18, 2014.

[9] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, and Q. Wang, "Robust lane detection from continuous driving scenes using deep neural networks," *IEEE transactions on vehicular technology*, vol. 69, no. 1, pp. 41–54, 2019.

[10] 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) Workshops*, 2018, pp. 0–0.

[11] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[12] J. Philion, "Fastdraw: Addressing the long tail of lane detection by adapting a sequential prediction network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 582–11 591.

[13] W. Van Gansbeke, B. De Brabandere, D. Neven, M. Proesmans, and L. Van Gool, "End-to-end lane detection through differentiable least-squares fitting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. IEEE, 2019, pp. 905–913.

[14] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, "Learning lightweight lane detection cnns by self attention distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1013–1021.

[15] 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.

[16] F. Pizzati, M. Allodi, A. Barrera, and F. García, "Lane detection and classification using cascaded cnns," in *International Conference on Computer Aided Systems Theory*. Springer, 2019, pp. 95–103.

[17] Y. Ko, Y. Lee, S. Azam, F. Munir, M. Jeon, and W. Pedrycz, "Key points estimation and point instance segmentation approach for lane detection," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[18] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5221–5229.

[19] J. Uhrig, M. Cordts, U. Franke, and T. Brox, "Pixel-level encoding and depth layering for instance-level semantic labeling," in *German Conference on Pattern Recognition*. Springer, 2016, pp. 14–25.

[20] T. Gupta, H. S. Sikchi, and D. Charkravarty, "Robust lane detection using multiple features," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1470–1475.

[21] N. Garnett, R. Cohen, T. Pe'er, R. Lahav, and D. Levi, "3d-lanenet: end-to-end 3d multiple lane detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2921–2930.

[22] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 294–302.

[23] Y. Huang, S. Chen, Y. Chen, Z. Jian, and N. Zheng, "Spatial-temproal based lane detection using deep learning," in *IFIP International conference on artificial Intelligence applications and innovations*. Springer, 2018, pp. 143–154.

[24] M. Bai, G. Mattyus, N. Homayounfar, S. Wang, S. K. Lakshmikanth, and R. Urtasun, "Deep multi-sensor lane detection," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3102–3109.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2017.

[27] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[28] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 764–773.

[29] K. Behrendt and R. Soussan, "Unsupervised labeled lane markers using maps," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[30] L. Tabelini, R. Berriel, T. M. Paixao, C. Badue, A. F. De Souza, and T. Oliveira-Santos, "Polylanenet: Lane estimation via deep polynomial regression," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 6150–6156.

[31] Y. Hou, Z. Ma, C. Liu, T.-W. Hui, and C. C. Loy, "Inter-region affinity distillation for road marking segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 486–12 495.

[32] Z. Li, "Curvelane-nas: Unifying lane-sensitive architecture search and adaptive point blending," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020.

[33] T. Liu, Z. Chen, Y. Yang, Z. Wu, and H. Li, "Lane detection in low-light conditions using an efficient data enhancement: Light conditions style transfer," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1394–1399.

[34] S. Yoo, H. S. Lee, H. Myeong, S. Yun, H. Park, J. Cho, and D. H. Kim, "End-to-end lane marker detection via row-wise classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 1006–1007.

[35] T. Zheng, H. Fang, Y. Zhang, W. Tang, Z. Yang, H. Liu, and D. Cai, "Resa: Recurrent feature-shift aggregator for lane detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 4, 2021, pp. 3547–3554.