Keep your Eyes on the Lane: Real-time Attention-guided Lane Detection

Lucas Tabelini[1] , Rodrigo Berriel[1], Thiago M. Paixão[2],
Claudine Badue[1], Alberto F. De Souza[1], Thiago Oliveira-Santos[1]
[1]Universidade Federal do Espírito Santo (UFES) [2]Instituto Federal do Espírito Santo (IFES)
tabelini@lcad.inf.ufes.br

Abstract

Modern lane detection methods have achieved remarkable performances in complex real-world scenarios, but many have issues maintaining real-time efficiency, which is important for autonomous vehicles. In this work, we propose LaneATT: an anchor-based deep lane detection model, which, akin to other generic deep object detectors, uses the anchors for the feature pooling step. Since lanes follow a regular pattern and are highly correlated, we hypothesize that in some cases global information may be crucial to infer their positions, especially in conditions such as occlusion, missing lane markers, and others. Thus, this work proposes a novel anchor-based attention mechanism that aggregates global information. The model was evaluated extensively on three of the most widely used datasets in the literature. The results show that our method outperforms the current state-of-the-art methods showing both higher efficacy and efficiency. Moreover, an ablation study is performed along with a discussion on efficiency trade-off options that are useful in practice. Code and models are available at https://github.com/lucastabelini/LaneATT.

1. Introduction

Deep learning has been essential for recent advances in numerous areas, especially in autonomous driving [2]. Many of the deep learning applications in self-driving cars are in their perception systems. To be safe around humans, autonomous vehicles should perceive their surroundings, including the position of other vehicles and themselves. In the end, the more predictable a car's movement is, the safer it will be for its passengers and pedestrians. Thus, it is important for autonomous vehicles to know each lane's exact position, which is the goal of lane detection systems.

Lane detection models have to overcome various challenges. In a real-world scenario, models should be robust to several adverse conditions, such as extreme light and weather. Moreover, lane markings can be occluded by other objects (*e.g.*, cars), which is extremely common for self-driving cars. Some approaches, such as polynomial regression models, may also suffer from a data imbalance problem caused by the long-tail effect since cases with sharper curves are less common. Besides, the model not only has to be robust but also efficient. In several applications, lane detection must perform in real-time, or faster to save processing power for other systems, a requirement that many models struggle to cope with.

There are numerous works in the literature that tackle this problem. Before the advent of deep learning, several methods used more traditional computer vision techniques, such as Hough lines [4, 1]. More recently, focus has shifted to deep learning approaches with the advance of convolutional neural networks (CNNs) [17, 11, 18]. In this context, the lane detection problem is usually formulated as a segmentation task, where, given an input image, the output is a segmentation map with per-pixel predictions [17]. Although recent advances in deep learning have enabled the use of segmentation networks in real-time [22], various models struggle to to achieve real-time performance. Consequently, the number of backbone options for segmentation-based methods is rather limited. Hence, some recent works have proposed solutions in other directions [13, 23]. Apart from that, many other issues are common in works on lane detection, such as the need for a post-processing step (usually a heuristic), long training times, and a lack of publicly available source code, which hinders comparisons and reproducibility.

In this work, we present a method for real-time lane detection that is both faster and more accurate than most state-of-the-art methods. We propose an anchor-based single-stage lane detection model called LaneATT. It uses a novel anchor-based feature pooling method that enables the use of a lightweight backbone CNN while maintaining high accuracy, contrary to an existing method [13]. A novel anchor-based attention mechanism to aggregate global information is also proposed. Extensive experimental results are shown on three benchmarks (TuSimple [24], CULane [17] and LLAMAS [3]), along with a comparison with state-of-the-art methods, a discussion on efficiency trade-offs, and an ablation study of our design choices. In summary, our main contributions are:

- A lane detection method that is more accurate than existing state-of-the-art real-time methods on a large and complex dataset;
- A model that enables faster training and inference times than most other models (reaching 250 FPS and almost an order of magnitude less multiply-accumulate operations (MACs) than the previous state-of-the-art);
- A novel anchor-based attention mechanism for lane detection which is potentially useful in other domains where the objects being detected are correlated.

2. Related work

Although the first lane detection approaches rely on classical computer vision, substantial progress on accuracy and efficiency has been achieved with recent deep learning methods. Thus, this literature review focuses on deep lane detectors. This section first discusses the dominant approaches, which are based on segmentation [17, 11, 29, 15] or row-wise classification [10, 20, 27], and, subsequently, review solutions in other directions. Finally, the lack of reproducibility (a common issue in lane detection works) is discussed.

Segmentation-based methods. In this approach, predictions are made on a per-pixel basis, classifying each pixel as either lane or background. With the segmentation map generated, a post-processing step is necessary to decode it into a set of lanes. In SCNN [17], the authors propose a scheme specifically designed for long thin structures and show its effectiveness in lane detection. However, the method is slow (7.5 FPS), which hinders its applicability in real-world cases. Since larger backbones are one of the main culprits for slower speeds, the authors of [11] propose a self attention distillation (SAD) module to aggregate contextual information. The module allows the use of a more lightweight backbone, achieving a high-performance while maintaining real-time efficiency. In CurveLanes-NAS [26], neural architecture search (NAS) is used to find a better backbone. Although they achieved state-of-the-art results, their NAS is extremely expensive computationally (5,000 GPU hours per dataset).

Row-wise classification methods. The row-wise classification approach is a simple way to detect lanes based on a grid division of the input image. For each row, the model predicts the most probable cell to contain a part of a lane marking. Since only one cell is selected on each row, this process is repeated for each possible lane in an image. Similar to segmentation methods, it also requires a post-processing step to construct the set of lanes. The method was first introduced in E2E-LMD [27], achieving state-of-the-art results on two datasets. In [20], the authors show that it is capable of reaching high speed, although some accuracy is lost. This approach is also used in IntRA-KD [10].

Other approaches. In FastDraw [18], the author proposes a novel learning-based approach to decode the lane structures, which avoids the need for clustering post-processing steps (required in segmentation and row-wise classification methods). Although the proposed method is shown to achieve high speeds, it does not perform better than existing state-of-the-art methods in terms of accuracy. The same effect is shown in PolyLaneNet [23], where an even faster model, based on deep polynomial regression, is proposed. In that approach, the model learns to output a polynomial for each lane. Despite its speed, the model struggles with the imbalanced nature of lane detection datasets, as evidenced by the high bias towards straight lanes in its predictions. In Line-CNN [13], an anchor-based method for lane detection is presented. This model achieves state-of-the-art results on a public dataset and promising results on another that is not publicly available. Despite the real-time efficiency, the model is considerably slower than other approaches since it requires larger backbones (*e.g.*, ResNet-122). In this work, we propose a novel anchor-based pooling method that allows the use of lightweight backbones. Moreover, the code is not public, which makes the results difficult to reproduce. There are also works addressing other parts of the pipeline of a lane detector. In [12], a post-processing method with a focus on occlusion cases is proposed, achieving results considerably higher than other works, but at the cost of notably low speeds (around 4 FPS).

Reproducibility. As noted in [23], many of the cited works do not publish the code to reproduce the results reported [13, 18, 27], or, in some cases, the code is only partially public [11, 10]. This hinders deeper qualitative and quantitative comparisons. For instance, the two most common metrics to measure a model's efficiency are multiply–accumulate operations (MACs) and frames-per-second (FPS). While the first does not depend on the benchmark platform, it is not always a good proxy for the second, which is the true goal. Therefore, FPS comparisons are also hindered by the lack of source code.

Unlike most of the previously proposed methods that managed to achieve high speeds at the cost of accuracy, we propose a method that is both faster and more accurate than existing state-of-the-art ones. In addition, the full code to reproduce the reported results is published for the community.

3. Proposed method

LaneATT is an anchor-based single-stage model (like YOLOv3 [21] or SSD [16]) for lane detection. An overview of the method is shown in Figure 1. It receives as input RGB images $I \in \mathbb{R}^{3\times H_I \times W_I}$ taken from a front-facing camera mounted in a vehicle. The outputs are lane boundary lines (hereafter called lanes, following the usual terminology in the literature). To generate those outputs, a convolutional neural

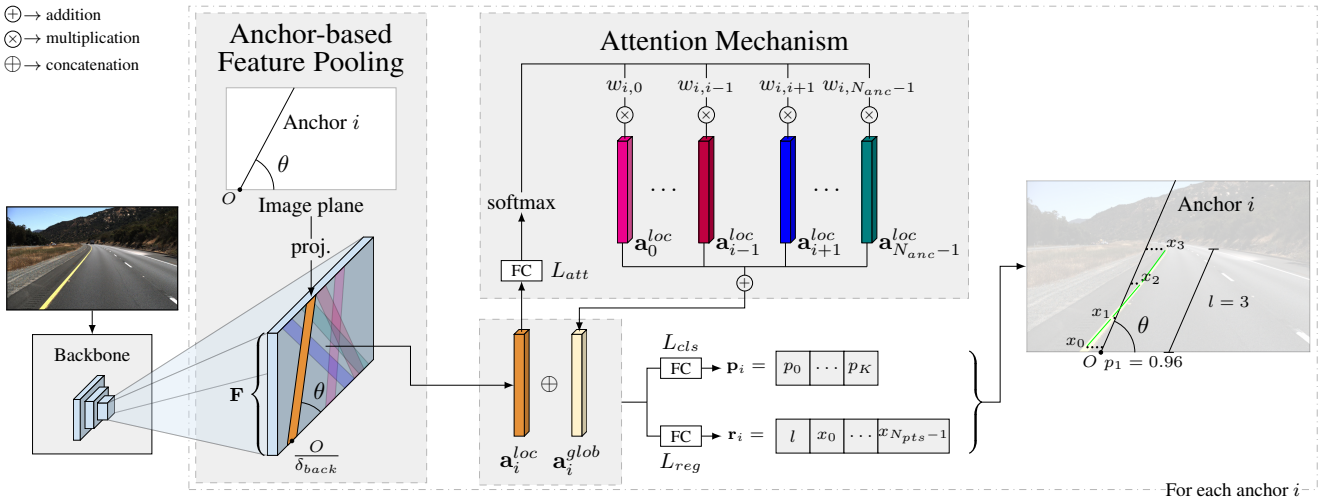

Figure 1. Overview of the proposed method. A backbone generates feature maps from an input image. Subsequently, each anchor is projected onto the feature maps. This projection is used to pool features that are concatenated with another set of features created in the attention module. Finally, using this resulting feature set, two layers, one for classification and another for regression, make the final predictions.

network (CNN), referred to as the backbone, generates a feature map that is then pooled to extract each anchor's features. Those features are combined with a set of global features produced by an attention module. By combining local and global features, the model can use information from other lanes more easily, which might be necessary in cases with conditions such as occlusion or no visible lane markings. Finally, the combined features are passed to fully-connected layers to predict the final output lanes.

3.1. Lane and anchor representation

A lane is represented by 2D-points with equally-spaced y-coordinates $Y = \{y_i\}_{i=0}^{N_{pts}-1}$, where $y_i = i \cdot \frac{H_I}{N_{pts}-1}$. Since Y is fixed, a lane can then be defined only by its x-coordinates $X = \{x_i\}_{i=0}^{N_{pts}-1}$, each x_i associated with the respective $y_i \in Y$. Since most lanes do not cross the whole image vertically, a start-index s and an end-index e are used to define the valid contiguous sequence of X.

Likewise Line-CNN [13], our method performs anchor-based detection using lines instead of boxes, which means that lanes' proposals are made having these lines as references. An anchor is a "virtual" line in the image plane defined by (i) an origin point $O = (x_{orig}, y_{orig})$ (with $y_{orig} \in Y$) located in one of the borders of the image (except the top border) and (ii) a direction θ. The proposed method uses the same set of anchors as [13]. This lane and anchor representation satisfies the vast majority of real-world lanes.

3.2. Backbone

The first stage of the proposed method is feature extraction, which can be performed by any generic CNN, such as a ResNet [9]. The output of this stage is a feature map $\mathbf{F}_{back} \in \mathbb{R}^{C'_F \times H_F \times W_F}$ from which the features for each anchor will be extracted through a pooling process, as described in the next section. For dimensionality reduction, a 1×1 convolution is applied onto $\mathbf{F}_{back}$, generating a channel-wise reduced feature map $\mathbf{F} \in \mathbb{R}^{C_F \times H_F \times W_F}$. This reduction is performed to reduce computational costs.

3.3. Anchor-based feature pooling

An anchor defines the points of $\mathbf{F}$ that will be used for the respective proposals. Since the anchors are modeled as lines, the interest points for a given anchor are those that intercept the anchor's virtual line (considering the rasterized line reduced to the feature maps dimensions). For every $y_j = 0, 1, 2, \ldots, H_F - 1$, there will be a single corresponding x-coordinate,

$$x_j = \left\lfloor \frac{1}{\tan\theta} \left(y_j - y_{orig}/\delta_{back}\right) + x_{orig}/\delta_{back} \right\rfloor, \quad (1)$$

where (x_{orig}, y_{orig}) and θ are, respectively, the origin point and slope of the anchor's line, and δ^{back} is the backbone's global stride. Thus, every anchor i will have its corresponding feature vector $\mathbf{a}_i^{loc} \in \mathbb{R}^{C_F \cdot H_F}$ (column-vector notation) pooled from $\mathbf{F}$ that carries local feature information (local features). In cases where a part of the anchor is outside the boundaries of $\mathbf{F}$, $\mathbf{a}_i^{loc}$ is zero-padded.

Notice that the pooling operation is similar to the Fast R-CNN's [8] region of interest (RoI) projection, however, instead of using the proposal for pooling, a single-stage detector is achieved by using the anchor itself. Additionally, the RoI pooling layer (used to generate fixed-size features) is not necessary for our method. Comparing to Line-CNN [13], that leverages only the feature maps' borders, our method can explore all the feature map, which enables the use of more lightweight backbones with smaller receptive field.

3.4. Attention mechanism

Depending on the model architecture, the information carried by the pooled feature vector ends up being mostly local. This is particularly the case for shallower and faster models, which tend to exploit backbones with smaller receptive fields. However, in some cases (such as the ones with occlusion) the local information may not be enough to predict the lane's existence and its position. To address that problem, we propose an attention mechanism that acts on the local features ($\mathbf{a}_\bullet^{loc}$) to produce additional features ($\mathbf{a}_\bullet^{glob}$) that aggregate global information.

Basically, the attention mechanism structure is composed of a fully-connected layer L_{att} which processes a local feature vector $\mathbf{a}_i^{loc}$ and outputs a probability (weight) $w_{i,j}$ for every anchor j, $j \neq i$. Formally,

$$w_{i,j} = \begin{cases} \text{softmax}(L_{att}(\mathbf{a}_i^{loc}))_j, & \text{if } j < i \\ 0, & \text{if } j = i \\ \text{softmax}(L_{att}(\mathbf{a}_i^{loc}))_{j-1}, & \text{if } j > i \end{cases} \quad (2)$$

Afterwards, those weights are combined with the local features to produce a global feature vector of same dimension:

$$\mathbf{a}_i^{glob} = \sum_j w_{i,j}\, \mathbf{a}_j^{loc}. \quad (3)$$

Naturally, the whole process can be implemented efficiently with matrix multiplication, since the same procedure is executed for all anchors. Let N_{anc} be the number of anchors. Let $\mathbf{A}^{loc} = [\mathbf{a}_0^{loc}, \ldots, \mathbf{a}_{N_{anc}-1}^{loc}]^T$ be the matrix containing the local feature vectors (as rows) and $\mathbf{W} = [w_{i,j}]_{N_{anc} \times N_{anc}}$ the weight matrix, $w_{i,j}$ defined in Equation (2). Thus, global features can be computed as:

$$\mathbf{A}^{glob} = \mathbf{W}\, \mathbf{A}^{loc}. \quad (4)$$

Notice that $\mathbf{A}^{glob}$ and $\mathbf{A}^{loc}$ have the same dimensions, *i.e.*, $\mathbf{A}^{glob} \in \mathbb{R}^{N_{anc} \times C_F \cdot H_F}$.

3.5. Proposal prediction

A lane proposal is predicted for each anchor and consists of three main components: (i) $K+1$ probabilities (K lane types and one class for "background" or invalid proposal), (ii) N_{pts} offsets (the horizontal distance between the prediction and the anchor's line), and (iii) the length l of the proposal (the number of valid offsets). The start-index (s) for the proposal is directly determined by the y-coordinate of the anchor's origin (y_{orig}, see Section 3.1). Thus, the end-index can be determined as $e = s + \lfloor l \rceil - 1$.

To generate the final proposals, local and global information are aggregated by concatenating $\mathbf{a}_i^{loc}$ and $\mathbf{a}_i^{glob}$, producing an augmented feature vector $\mathbf{a}_i^{aug} \in \mathbb{R}^{2 \cdot C_F \cdot H_F}$. This augmented vector is fed to two parallel fully-connected layers, one for classification (L_{cls}) and one for regression (L_{reg}), which produce the final proposals. L_{cls} predicts $\mathbf{p}_i = \{p_0, \ldots, p_{K+1}\}$ (item i) and L_{reg} predicts $\mathbf{r}_i = \big(l, \{x_0, \ldots, x_{N_{pts}-1}\}\big)$ (items ii and iii).

3.6. Non-maximum Supression (NMS)

As usual in anchor-based deep detection, NMS is paramount to reduce the number of false positives. In the proposed method, this procedure is applied both during training and test phases based on the lane distance metric proposed in [13]. The distance between two lanes $X_a = \{x_i^a\}_{i=1}^{N_{pts}}$ and $X_b = \{x_i^b\}_{i=1}^{N_{pts}}$ is computed based on their common valid indices (or y-coordinates). Let $s' = \max(s_a, s_b)$ and $e' = \min(e_a, e_b)$ define the range of those common indices. Thus, the lane distance metric is defined as

$$D(X_a, X_b) = \begin{cases} \frac{1}{e'-s'+1} \cdot \sum_{i=s'}^{e'} |x_i^a - x_i^b|, & e' \geq s' \\ +\infty, & \text{otherwise.} \end{cases} \quad (5)$$

3.7. Model training

During training, the distance metric in Equation (5) is also used to define the positive and the negative anchors. First, the metric is used to measure the distance between every anchor (those not filtered in NMS) and the ground-truth lanes. Subsequently, the anchors with distance (Eq. 5) lower than a threshold τ_p are considered positives, while those with distance greater than τ_n are considered negatives. Anchors (and their associated proposals) with distance in between those thresholds are disregarded. The remainder $N_{p\&n}$ are used in a multi-task loss defined as:

$$\mathcal{L}(\{\mathbf{p}_i, \mathbf{r}_i\}_{i=0}^{N_{p\&n}-1}) = \lambda \sum_i \mathcal{L}_{cls}(\mathbf{p}_i, \mathbf{p}_i^*) + \sum_i \mathcal{L}_{reg}(\mathbf{r}_i, \mathbf{r}_i^*), \quad (6)$$

where $\mathbf{p}_i$, $\mathbf{r}_i$ are the classification and regression outputs for the anchor i, whereas $\mathbf{p}_i^*$ and $\mathbf{r}_i^*$ are the classification and regression targets for the anchor i. The regression loss is computed only with the length l and the x-coordinates values corresponding to indices common to both the proposal and the ground-truth. The common indices (between s' and e') of the x-coordinates are selected similarly to the lane distance (Equation (5)) but with $e' = e_{gt}$ instead of $e' = \min(e_{prop}, e_{gt})$, where e_{prop} and e_{gt} are the end-indexes for the proposal and its associated ground-truth, respectively. If the end-index predicted in the proposal e_{prop} is used, the training may become unstable by converging to degenerate solutions (*e.g.*, e_{prop} might converge to zero). The functions $\mathcal{L}_{cls}$ and $\mathcal{L}_{reg}$ are the Focal Loss [14] and the Smooth L1, respectively. If the anchor i is considered negative, its corresponding $\mathcal{L}_{reg}$ is equal to 0. The factor λ is used to balance the loss components.

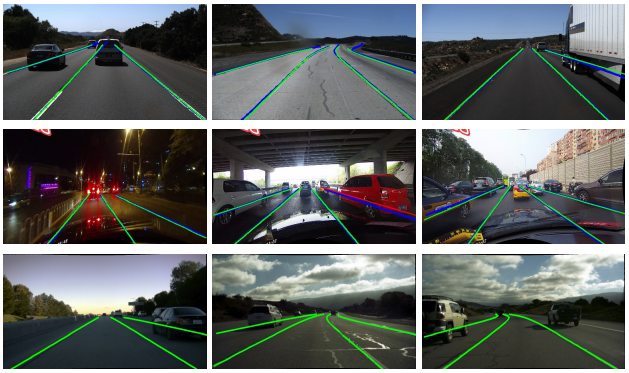

Figure 2. LaneATT qualitative results on TuSimple (top row), CULane (middle row), and LLAMAS (bottom row). Blue lines are ground-truth, while green and red lines are true-positives and false-positives, respectively. See more samples in the videos[1].

3.8. Anchor filtering for speed efficiency

The full set of anchors comprises a total of 2,782 anchors. This elevated number is one of the main factors limiting the model's speed. Since a large number of anchors will not be useful during the training (*e.g.*, some anchors may have a starting point above the horizon line of all images in the training dataset), the set's size can be reduced. To choose which anchors are going to be disregarded in both training and test phases, the method measures the number of times each anchor from the training set is marked as positive (same criteria as in the training). Finally, only the top-N_{anc} marked anchors are kept for further processing (also during test).

4. Experiments

Our method was evaluated on the two most widely used lane detection datasets (TuSimple [24] and CULane [17]) and on the recently released benchmark (LLAMAS [3]). An overview of the datasets can be seen in Table 1. This section starts describing the efficiency metrics and some of the implementation details. All experiments used the default metric parameters set by the dataset's creator, which are the same used by the related works. The three first subsections discusses the experimental results for each dataset (including the dataset description and the evaluation metrics). The two final subsections address experiments on efficiency trade-offs and an ablation study on parts.

Dataset	Train	Val.	Test	Max. # of lanes
TuSimple [24]	3,268	358	2,782	5
CULane [17]	88,880	9,675	34,680	4
LLAMAS [3]	58,269	20,844	20,929	4

Table 1. Overview of the datasets used in this work.

[1] http://youtu.be/1f_y4A-muMg and http://youtu.be/ghs93acwkBQ

Efficiency metrics. Two efficiency-related metrics are reported: frames-per-second (FPS) and multiply-accumulate operations (MACs). One MAC is approx. two floating operations (FLOPs). The FPS is computed using a single image per batch and constant inputs, so the metric is not dependent on I/O operations but only on the model's efficiency.

Implementation details. Except when explicitly indicated, all input images are resized to $H_I \times W_I = 360 \times 640$ pixels. For all training sessions, the Adam optimizer is used for 15 epochs on CULane and 100 epochs on TuSimple (the large discrepancy is due to the large difference between the datasets' sizes). For data augmentation, a random affine transformation is performed (with translation, rotation, and scaling) along with random horizontal flips. Most experiments and all FPS measures were computed on a machine with an Intel i9-9900KS and an RTX 2080 Ti. The model parameters were $N_{pts} = 72$, $N_{anc} = 1000$, $\tau_p = 15$ and $\tau_n = 20$. The used datasets do not provide the lane type (*e.g.*, dashed or solid), thus, we set $K = 1$ (see Section 3.5). For more details and parameter values, the code[2] can be accessed, along with each experiment's configuration.

4.1. TuSimple

Dataset description. TuSimple [24] is a lane detection dataset containing only highway scenes, a scenario that is usually considered easier than street scenes. Despite that, it is one of the most widely used datasets in lane detection works. All images have 1280×720 pixels, with at most 5 lanes.

Evaluation metrics. On TuSimple, the three standard metrics are false discovery rate (FDR), false negative rate (FNR), and accuracy. The accuracy is defined as

$$\text{Acc} = \frac{\sum_{clip} C_{clip}}{\sum_{clip} S_{clip}}, \tag{7}$$

where C_{clip} is the number of lane points predicted correctly in the clip and S_{clip} is the total number of points in the clip (or image). For a point prediction to be considered correct, the prediction has to be within 20 pixels the ground truth. For a lane prediction to be considered a true positive (for the FDR and FNR metrics), its number of correct predicted points has to be greater than 85%. We also report the F1 score (hereafter called F1), which is the harmonic mean of the precision and the recall.

Results. The results of LaneATT on TuSimple, along with other state-of-the-art methods, are shown in Table 2 and in Figure 3 (left side). Qualitative results are shown in Figure 2 (top row). As demonstrated, accuracy-wise LaneATT is

[2] https://github.com/lucastabelini/LaneATT

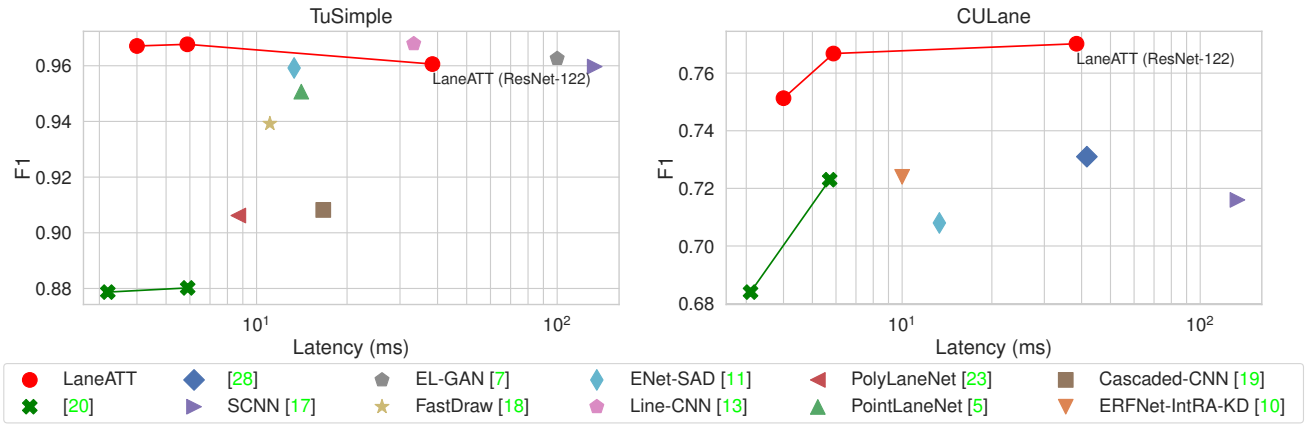

Figure 3. Model latency vs. F1 of state-of-the-art methods on CULane and TuSimple.

Method	F1 (%)	Acc (%)	FDR (%)	FNR (%)	FPS	MACs (G)
Source-code unavailable						
EL-GAN [7]	96.26	94.90	4.12	3.36	10.0	
Line-CNN [13]	96.79	96.87	4.42	1.97	30.0	
FastDraw (ResNet-18) [18]	94.59	94.90	6.10	4.70		
PointLaneNet [5]	95.07	96.34	4.67	5.18	71.0	
[25]	95.80				71.5	
R-18-E2E [27]	96.40	96.04	3.11	4.09		
R-34-E2E [27]	96.58	96.22	3.08	3.76		
Source-code available						
SCNN [17]	95.97	96.53	6.17	**1.80**	7.5	
Cascaded-CNN [19]	90.82	95.24	11.97	6.20	60.0	
ENet-SAD [11]	95.92	**96.64**	6.02	2.05	75.0	
[20] (ResNet-18)	87.87	95.82	19.05	3.92	**312.5**	
[20] (ResNet-34)	88.02	95.86	18.91	3.75	169.5	
PolyLaneNet [23]	90.62	93.36	9.42	9.33	115.0	**1.7**
LaneATT (ResNet-18)	96.71	95.57	3.56	3.01	250	9.3
LaneATT (ResNet-34)	**96.77**	95.63	**3.53**	2.92	171	18.0
LaneATT (ResNet-122)	96.06	96.10	5.64	2.17	26	70.5

Table 2. State-of-the-art results on TuSimple. For a fairer comparison, the FPS of the fastest method ([20]) was measured on the same machine and conditions as our method. Additionally, all metrics for this method were computed using the official source code, since only the accuracy was available in the paper. The best and second-best results across methods with source-code available are in bold and underlined, respectively.

on par with other state-of-the-art methods. However, it is also clear that the results in this dataset are saturated (high-values) already, probably because its scenes are not complex and the metric is permissive [23]. This is evidenced by the small difference in performance across methods, in contrast to results in more complex datasets and less permissive metrics (as shown in Section 4.2). Nonetheless, our method is much faster than others. The method proposed in [20] is the only with speed comparable to ours. Since the FDR and FNR metrics were not reported in their work, we computed them using the published code and reported those metrics. Although they achieved high accuracy, the FDR is notably high. For instance, our highest FDR is 5.64% (ResNet-122), whereas their lowest is 18.91%, almost four times higher.

4.2. CULane

Dataset description. CULane [17] is one of the largest publicly available lane detection datasets, and also one of the most complexes. All the images have 1640×590 pixels, and all test images are divided into nine categories, such as crowded, night, absence of visible lines, etc.

Method	Total	Normal	Crowded	Dazzle	Shadow	No line	Arrow	Curve	Cross	Night	FPS	MACs (G)
Source-code unavailable												
[28]	73.10	89.70	76.50	67.40	65.50	35.10	82.20	63.20		68.70	24.0	
FastDraw (ResNet-50) [18]		85.90	63.60	57.00	59.90	40.60	79.40	65.20	7013	57.80	90.3	
PointLaneNet [5]		90.10									71.0	
SpinNet [6]	74.20	90.50	71.70	62.00	72.90	43.20	85.00	50.70		68.10		
R-18-E2E [27]	70.80	90.00	69.70	60.20	62.50	43.20	83.20	70.30	2296	63.30		
R-34-E2E [27]	71.50	90.40	69.90	61.50	68.10	45.00	83.70	69.80	2077	63.20		
R-101-E2E [27]	71.90	90.10	71.20	60.90	68.10	44.90	84.30	70.20	2333	65.20		
ERFNet-E2E [27]	74.00	91.00	73.10	64.50	74.10	46.60	85.80	71.90	2022	67.90		
Source-code available												
SCNN [17]	71.60	90.60	69.70	58.50	66.90	43.40	84.10	64.40	1990	66.10	7.5	
ENet-SAD [11]	70.80	90.10	68.80	60.20	65.90	41.60	84.00	65.70	1998	66.00	75	
[20] (ResNet-18)	68.40	87.70	66.00	58.40	62.80	40.20	81.00	57.90	1743	62.10	**322.5**	
[20] (ResNet-34)	72.30	90.70	70.20	59.50	69.30	44.40	85.70	**69.50**	2037	66.70	175.0	
ERFNet-IntRA-KD [10]	72.40										100.0	
SIM-CycleGAN [15]	73.90	91.80	71.80	66.40	76.20	46.10	87.80	67.10	2346	69.40		
CurveLanes-NAS-S [26]	71.40	88.30	68.60	63.20	68.00	47.90	82.50	66.00	2817	66.20		**9.0**
CurveLanes-NAS-M [26]	73.50	90.20	70.50	65.90	69.30	48.80	85.70	67.50	2359	68.20		33.7
CurveLanes-NAS-L [26]	74.80	90.70	72.30	67.70	70.10	49.40	85.80	68.40	1746	68.90		86.5
LaneATT (ResNet-18)	75.09	91.11	72.96	65.72	70.91	48.35	85.49	63.37	**1170**	68.95	250	9.3
LaneATT (ResNet-34)	76.68	**92.14**	75.03	66.47	**78.15**	49.39	**88.38**	67.72	1330	70.72	171	18.0
LaneATT (ResNet-122)	**77.02**	91.74	**76.16**	**69.47**	76.31	**50.46**	86.29	64.05	1264	**70.81**	26	70.5

Table 3. State-of-the-art results on CULane. Since the images in the "Cross" category have no lanes, the reported number is the amount of false-positives. For a fairer comparison, we measured the FPS of the fastest method ([20]) under the same machine and conditions as ours. The best and second-best results across methods with source-code available are in bold and underlined, respectively.

Evaluation metrics. The only metric is the F1, which is based on the intersection over union (IoU). Since the IoU relies on areas instead of points, a lane is represented as a thick line connecting the respective lane's points. In particular, the dataset's official metric considers the lanes as 30-pixels-thick lines. If a prediction has an IoU greater than 0.5 with a ground-truth lane, it is considered a true positive.

Results. The results of LaneATT on CULane, along with other state-of-the-art methods, are shown in Table 3 and in Figure 3 (right side). Qualitative results are shown in Figure 2 (middle row). We do not compare to the results shown in [12], as the main contribution is a post-processing method that could easily be incorporated to our method, but the source-code is not public. Moreover, it is remarkably slow (less than 10 FPS, as reported in their work), which makes the model impractical in real-world applications. In this context, LaneATT achieves the highest F1 while maintaining a high efficiency (+170 FPS) on CULane, a dataset with highly complex scenes. Compared to [20], our most lightweight model (with ResNet-18) surpasses their largest (with ResNet-34) by almost 3% of F1 while being much faster (250 vs. 175 FPS on the same machine). Additionally, in "Night" and "Shadow" scenes, our method outperforms all others, including SIM-CycleGAN [15], specifically designed for those scenarios. These results demonstrate both the efficacy and the efficiency of LaneATT.

4.3. LLAMAS

Dataset description. LLAMAS [3] is a large lane detection dataset with over 100k images. The annotations were not manually made, instead, they were generated using high definition maps. All images are from highway scenarios. The evaluation is based on the CULane's F1, which was computed by the author of the LLAMAS benchmark since the testing set's annotations are not public.

Results. The results of LaneATT on LLAMAS, along with PolyLaneNet's [23] results, are shown in Table 4. Qualitative results are shown in Figure 2 (bottom row). Since the benchmark is recent and only PolyLaneNet provided the necessary source code to be evaluated on LLAMAS, it is the only comparable method. As evidenced, LaneATT is able to achieve an F1 greater than 90% in all three backbones. The results can also be seen in the benchmark's website[3].

Method	F1 (%)	Prec. (%)	Rec. (%)
PolyLaneNet [23]	88.40	88.87	87.93
LaneATT (ResNet-18)	93.46	96.92	90.24
LaneATT (ResNet-34)	93.74	96.79	90.88
LaneATT (ResNet-122)	93.54	96.82	90.47

Table 4. State-of-the-art results on LLAMAS.

[3] https://unsupervised-llamas.com/llamas/benchmark_splines

Modification	F1 (%)	FPS	MACs (G)	TT (h)
$N_{anc} = 250$	68.68	196	17.3	5.7
$N_{anc} = 500$	75.45	190	17.4	6.4
$N_{anc} = 750$	75.80	181	17.7	7.8
$N_{anc} = 1000$	76.66	171	18.0	11.1
$N_{anc} = 1250$	75.91	156	18.4	11.5
$H_I \times W_I = 180 \times 320$	66.74	195	4.8	4.3
$H_I \times W_I = 288 \times 512$	75.02	186	11.5	7.3
$H_I \times W_I = 360 \times 640$	76.66	171	18.0	11.1

Table 5. Efficiency trade-offs on CULane using the ResNet-34 backbone. "TT" stands for training time in hours.

4.4. Efficiency trade-offs

Being efficient is crucial for a lane detection model. In some cases, it might even be necessary to trade some accuracy to achieve the application's requirement. In this experiment, some of the possible trade-offs are shown. In particular, different settings of image input size ($H_I \times W_I$) and number of anchors (N_{anc}, as described in Section 3.8). The results are shown in Table 5. They show that the number of anchors can be reduced for a slight improvement in terms of efficiency without a large F1 drop. However, if the reduction is too large, the F1 starts to drop considerably. Moreover, if too many anchors are used, the efficacy can also degrade, which might be a consequence of unnecessary anchors disturbing the training. The results are similar for the input size, although the MACs drops are larger. The largest impact of both the number of anchors and the input size is on the training time. During the inference, the proposals are filtered (using a confidence threshold) before the NMS procedure. During the training, there's no such filtering. Since the NMS is one the main bottlenecks of the model, and its running time depends directly on the number of objects, the number of anchors has a much higher impact on the training.

4.5. Ablation study

This experiment evaluates the impact of each major part (one at a time) of the proposed method: anchor-based pooling, shared layers, focal loss, and the attention mechanism. The results are shown in Table 6. The first row comprises the results for the standard LaneATT, while the following rows show the results for slightly modified versions. In the second row, the anchor-based pooling was removed and the same procedure to select features of Line-CNN [13] was used (*i.e.*, only features from a single point in the feature map were used for each anchor). In the third one, instead of using a single pair of fully-connected layers (L_{reg} and L_{cls}) for the final prediction, three pairs (six layers) were used, one pair for each boundary (left, bottom, or right). That is, all anchors starting in the left boundary of the image had its proposals generated by the same pair of layers L^L_{reg} and L^L_{cls}

Model	F1 (%)	FPS	Params. (M)
LaneATT (ResNet-34)	76.68	171	22.13
− anchor-based pooling	64.89	188	21.39
− shared layers	75.45	142	22.34
− focal loss	75.54	171	22.13
− attention mechanism	75.78	196	21.37

Table 6. Ablation study results on CULane.

and similarly for the bottom (L^B_{reg} and L^B_{cls}) and the right (L^R_{reg} and L^R_{cls}) boundaries. In the fourth one, the Focal Loss was replaced with the Cross Entropy, and in the last one, the attention mechanism was removed.

The massive drop of performance when the anchor-based pooling procedure is removed shows its importance. This procedure enabled the use of a more lightweight backbone, which was not possible in Line-CNN [13] without a large performance drop. The results also show that a layer for each boundary of the image (as done in [13]) is not only unnecessary, but also detrimental to the model's efficiency. Furthermore, using the Focal Loss instead of the Cross Entropy was also shown to be beneficial. Besides, it also eliminates the need for one hyperparameter (the number of negative samples to be used in the loss computation). Finally, the proposed attention mechanism is another modification that significantly increases the model performance.

5. Conclusion

We proposed a real-time single-stage deep lane detection model that outperforms state-of-the-art models, as shown by an extensive comparison with the literature. The model is not only effective but also efficient. On TuSimple, the method achieves the second-highest reported F1 (a difference of only 0.02%) while being much faster than the top-F1 method (171 vs. 30 FPS). On CULane, one of the largest and most complex lane detection datasets, LaneATT establishes a new state-of-the-art among real-time methods in terms of both speed and accuracy (+4.38% of F1 compared to the state-of-the-art method with a similar speed of around 170 FPS). Moreover, LaneATT achieves over 93% of F1 on LLAMAS for all three backbones evaluated. To achieve those results, along with other modifications, a novel anchor-based attention mechanism was also proposed. The ablation study showed that this addition increased the model's performance (F1 score) significantly compared to the gains obtained by the literature in recent years. Additionally, some efficiency trade-offs that are useful in practice were also shown.

Acknowledgments. This study was financed in part by the Coordenacão de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, FAPES (grant 594/2018) and PIIC UFES.

References

[1] A. A. Assidiq, O. O. Khalifa, M. R. Islam, and S. Khan. Real time lane detection for autonomous vehicles. In *Int. Conf. on Computer and Communication Engineering*, 2008. 1

[2] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B. Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M. Paixão, Filipe Mutz, Lucas de Paula Veronese, Thiago Oliveira-Santos, and Alberto F. De Souza. Self-driving cars: A survey. *Expert Systems with Applications*, 165:113816, 2021. 1

[3] Karsten Behrendt and Ryan Soussan. Unsupervised labeled lane marker dataset generation using maps. In *International Conference on Computer Vision (ICCV)*, 2019. 1, 5, 7

[4] Rodrigo F. Berriel, Edilson de Aguiar, Alberto F. De Souza, and Thiago Oliveira-Santos. Ego-Lane Analysis System (ELAS): Dataset and Algorithms. *Image and Vision Computing*, 68:64–75, 2017. 1

[5] Zhenpeng Chen, Qianfei Liu, and Chenfan Lian. PointLaneNet: Efficient end-to-end CNNs for Accurate Real-Time Lane Detection. In *Intell. Vehicles Symposium*, 2019. 6, 7

[6] Ruochen Fan, Xuanrun Wang, Qibin Hou, Hanchao Liu, and Tai-Jiang Mu. SpinNet: Spinning Convolutional Network for Lane Boundary Detection. *Computational Visual Media*, 5(4):417–428, 2019. 7

[7] Mohsen Ghafoorian, Cedric Nugteren, Nóra Baka, Olaf Booij, and Michael Hofmann. EL-GAN: Embedding Loss Driven Generative Adversarial Networks for Lane Detection. In *European Conference on Computer Vision (ECCV)*, 2018. 6

[8] Ross Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, 2015. 3

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[10] Yuenan Hou, Zheng Ma, Chunxiao Liu, Tak-Wai Hui, and Chen Change Loy. Inter-Region Affinity Distillation for Road Marking Segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 6, 7

[11] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning lightweight Lane Detection CNNs by Self Attention Distillation. In *International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 6, 7

[12] Hussam Ullah Khan, Afsheen Rafaqat Ali, Ali Hassan, Ahmed Ali, Wajahat Kazmi, and Aamer Zaheer. Lane Detection using Lane Boundary Marker Network with Road Geometry Constraints. In *Winter Conference on Applications of Computer Vision (WACV)*, 2020. 2, 7

[13] Xiang Li, Jun Li, Xiaolin Hu, and Jian Yang. Line-CNN: End-to-end traffic line detection with line proposal unit. *IEEE Transactions on Intelligent Transportation Systems*, 21:248–258, 2019. 1, 2, 3, 4, 6, 8

[14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4

[15] Tong Liu, Zhaowei Chen, Yi Yang, Zehao Wu, and Haowei Li. Lane Detection in Low-light Conditions Using an Efficient Data Enhancement: Light Conditions Style Transfer. In *Intell. Vehicles Symposium*, 2020. 2, 7

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single Shot Multibox Detector. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[17] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial As Deep: Spatial CNN for Traffic Scene Understanding. In *Conference on Artificial Intelligence (AAAI)*, February 2018. 1, 2, 5, 6, 7

[18] Jonah Philion. FastDraw: Addressing the Long Tail of Lane Detection by Adapting a Sequential Prediction Network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 6, 7

[19] Fabio Pizzati, Marco Allodi, Alejandro Barrera, and Fernando García. Lane Detection and Classification using Cascaded CNNs. In *International Conference on Computer Aided Systems Theory (EUROCAST)*, 2019. 6

[20] Zequn Qin, Huanyu Wang, and Xi Li. Ultra Fast Structure-aware Deep Lane Detection. In *European Conference on Computer Vision (ECCV)*, 2020. 2, 6, 7

[21] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[22] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation. *IEEE Trans. Intell. Transp. Syst.*, 19(1):263–272, 2017. 1

[23] Lucas Tabelini, Rodrigo Berriel, Thiago M. Paixão, Claudine Badue, Alberto F. De Souza, and Thiago Oliveira-Santos. PolyLaneNet: Lane Estimation via Deep Polynomial Regression. In *ICPR*, 2020. 1, 2, 6, 7

[24] TuSimple. Tusimple benchmark. https://github.com/TuSimple/tusimple-benchmark. Accessed September, 2020. 1, 5

[25] Wouter Van Gansbeke, Bert De Brabandere, Davy Neven, Marc Proesmans, and Luc Van Gool. End-to-end Lane Detection through Differentiable Least-Squares Fitting. In *ICCV Workshop*, 2019. 6

[26] Hang Xu, Shaoju Wang, Xinyue Cai, Wei Zhang, Xiaodan Liang, and Zhenguo Li. CurveLane-NAS: Unifying Lane-Sensitive Architecture Search and Adaptive Point Blending. In *European Conference on Computer Vision (ECCV)*, 2020. 2, 7

[27] Seungwoo Yoo, Hee Seok Lee, Heesoo Myeong, Sungrack Yun, Hyoungwoo Park, Janghoon Cho, and Duck Hoon Kim. End-to-End Lane Marker Detection via Row-wise Classification. In *IEEE CVPR Workshop*, 2020. 2, 6, 7

[28] Jie Zhang, Yi Xu, Bingbing Ni, and Zhenyu Duan. Geometric Constrained Joint Lane Segmentation and Lane Boundary Detection. In *European Conference on Computer Vision (ECCV)*, 2018. 6, 7

[29] Qin Zou, Hanwen Jiang, Qiyu Dai, Yuanhao Yue, Long Chen, and Qian Wang. Robust Lane Detection from Continuous Driving Scenes using Deep Neural Networks. *Transactions on Vehicular Technology*, 69(1):41–54, 2019. 2