Rethinking Efficient Lane Detection via Curve Modeling

Zhengyang Feng[1*], Shaohua Guo[1*], Xin Tan[2,1†], Ke Xu[3†], Min Wang[4], Lizhuang Ma[1,2,5‡]
[1]Shanghai Jiao Tong University [2]East China Normal University [3]City University of Hong Kong
[4]SenseTime Research [5]MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University
zyfeng97@sjtu.edu.cn;guoshaohua@sjtu.edu.cn;tanxin2017@sjtu.edu.cn;kkangwing@gmail.com;
wangmin@sensetime.com;ma-lz@cs.sjtu.edu.cn

Abstract

This paper presents a novel parametric curve-based method for lane detection in RGB images. Unlike state-of-the-art segmentation-based and point detection-based methods that typically require heuristics to either decode predictions or formulate a large sum of anchors, the curve-based methods can learn holistic lane representations naturally. To handle the optimization difficulties of existing polynomial curve methods, we propose to exploit the parametric Bézier curve due to its ease of computation, stability, and high freedom degrees of transformations. In addition, we propose the deformable convolution-based feature flip fusion, for exploiting the symmetry properties of lanes in driving scenes. The proposed method achieves a new state-of-the-art performance on the popular LLAMAS benchmark. It also achieves favorable accuracy on the TuSimple and CULane datasets, while retaining both low latency (>150 FPS) and small model size (<10M). Our method can serve as a new baseline, to shed the light on the parametric curves modeling for lane detection. Codes of our model and PytorchAutoDrive: a unified framework for self-driving perception, are available at: https://github.com/voldemortX/pytorch-auto-drive.

1. Introduction

Lane detection is a fundamental task in autonomous driving systems, which supports the decision-making of lane-keeping, centering and changing, *etc*. Previous lane detection methods [2, 10] typically rely on expensive sensors such as LIDAR. Advanced by the rapid development of deep learning techniques, many works [14, 16, 17, 24, 31] are proposed to detect lane lines from RGB inputs captured by commercial front-mounted cameras.

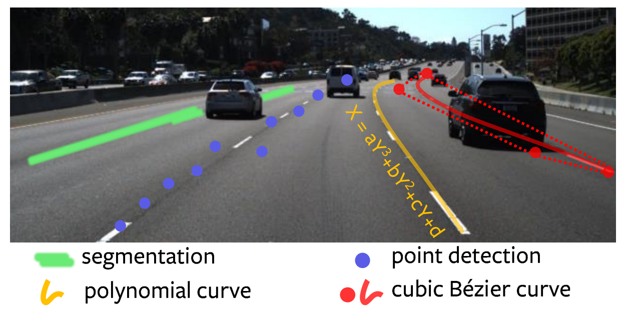

Figure 1. Lane detection strategies. Segmentation-based and point detection-based representations are local and indirect. The abstract coefficients (a, b, c, d) used in polynomial curve are hard to optimize. The cubic Bézier curve is defined by 4 actually existing control points, which roughly fit line shape and wrap the lane line in its convex hull (dashed red lines). Best viewed in color.

Deep lane detection methods can be classified into three categories, *i.e*., segmentation-based, point detection-based, and curve-based methods (Figure 1). Among them, by relying on classic segmentation [5] and object detection [20] networks, the segmentation-based and point detection-based methods typically achieve state-of-the-art lane detection performance. The segmentation-based methods [16, 17, 31] exploit the foreground texture cues to segment the lane pixels and decode these pixels into line instances via heuristics. The point detection-based methods [12, 24, 29] typically adopt the R-CNN framework [8, 20], and detect lane lines by detecting a dense series of points (*e.g*., every 10 pixels in the vertical axis). Both kinds of approaches represent lane lines via indirect proxies (*i.e*., segmentation maps and points). To handle the learning of holistic lane lines, under cases of occlusions or adverse weather/illumination conditions, they have to rely on low-efficiency designs, such as recurrent feature aggregation (too heavy for this real-time task) [17, 31], or a large number of heuristic anchors ($>$ 1000, which may be biased to dataset statistics) [24].

On the other hand, there are only a few methods [14, 23] proposed to model the lane lines as holistic curves (typically the polynomial curves, *e.g*., $x = ay^3 + by^2 + cy + d$).

*Equal Contribution.
†Corresponding Authors.
‡Lizhuang Ma led this project, and he is a member of Qing Yuan Research Institute, Shanghai Jiao Tong University.

While we expect the holistic curve to be a concise and elegant way to model the geometric properties of lane line, the abstract polynomial coefficients are difficult to learn. Previous studies show that their performance lag behind the well-designed segmentation-based and point detection-based methods by a large margin (up to 8% gap to state-of-the-art methods on the CULane [17] dataset). *In this paper, we aim to answer the question of whether it is possible to build a state-of-the-art curve-based lane detector.*

We observe that the classic cubic Bézier curves, with sufficient freedom degrees of parameterizing the deformations of lane lines in driving scenes, remain low computation complexity and high stability. This inspires us to propose to model the thin and long geometric shape properties of lane lines via Bézier curves. The ease of optimization from on-image Bézier control points enables the network to be end-to-end learnable with the bipartite matching loss [28], using a sparse set of lane proposals from simple column-wise Pooling (*e.g.*, 50 proposals on the CULane dataset [17]), without any post-processing steps such as the Non-Maximum Suppression (NMS), or hand-crafted heuristics such as anchors, hence leads to high speed and small model size. In addition, we observe that lane lines appear symmetrically from a front-mounted camera (*e.g.*, between ego lane lines, or immediate left and right lanes). To model this global structure of driving scenes, we further propose the feature flip fusion, to aggregate the feature map with its horizontally flipped version, to strengthen such coexistences. We base our design of feature flip fusion on the deformable convolution [32], for aligning the imperfect symmetries caused by, *e.g.*, rotated camera, changing lane, non-paired lines. We conduct extensive experiments to analyze the properties of our method and show that it performs favorably against state-of-the-art lane detectors on three popular benchmark datasets. Our main contributions are summarized as follows:

- We propose a novel Bézier curve-based deep lane detector, which can model the geometric shapes of lane lines effectively, and be naturally robust to adverse driving conditions.
- We propose a novel deformable convolution-based feature flip fusion module, to exploit the symmetry property of lanes observed from front-view cameras.
- We show that our method is fast, light-weight, and accurate through extensive experiments on three popular lane detection datasets. Specifically, our method outperforms all existing methods on the LLAMAS benchmark [3], with the light-weight ResNet-34 backbone.

2. Related Work

Segmentation-based Lane Detection. These methods represent lanes as per-pixel segmentation. SCNN [17] formulates lane detection as multi-class semantic segmentation and is the basis of the 1st-place solution in TuSimple challenge [1]. It's core spatial CNN module recurrently aggregates spatial information to complete the discontinuous segmentation predictions, which then requires heuristic post-processing to decode the segmentation map. Hence, it has a high latency, and only struggles to be real-time after an optimization of Zheng *et al*. [31]. Others explore knowledge distillation [11] or generative modeling [7], but their performance merely surpasses the seminal SCNN. Moreover, these methods typically assume a fixed number (*e.g.*, 4) of lines. LaneNet [16] leverages an instance segmentation pipeline to deal with a variable number of lines, but it requires post-inference clustering to generate line instances. Some methods leverage row-wise classification [19, 30], which is a customized down-sampling of per-pixel segmentation so that they still require post-processing. Qin *et al*. [19] propose to trade performance for low latency, but their use of fully-connected layers results in large model size.

In short, segmentation-based methods all require heavy post-processing due to the misalignment of representations. They also suffer from the locality of segmentation task, so that they tend to perform worse under occlusions or extreme lighting conditions.

Point Detection-based Lane Detection. The success of object detection methods drives researchers to formulate lane detection as to detect lanes as a series of points (*e.g.*, every 10 pixels in the vertical axis). Line-CNN [12] adapts classic Faster R-CNN [20] as a one-stage lane line detector, but it has a low inference speed ($<$30 FPS). Later, LaneATT [24] adopts a more general one-stage detection approach that achieves superior performance.

However, these methods have to design heuristic lane anchors, which highly depend on dataset statistics, and require the Non-Maximum Suppression (NMS) as post-processing. On the contrary, we represent lane lines as curves with a fully end-to-end pipeline (anchor-free, NMS-free).

Curve-based Lane Detection. The pioneering work [27] proposes a differentiable least squares fitting module to fit a polynomial curve (*e.g.*, $x = ay^3 + by^2 + cy + d$) to points predicted by a deep neural network. The PolyLaneNet [23] then directly learns to predict the polynomial coefficients with simple fully-connected layers. Recently, LSTR [14] uses transformer blocks to predict polynomials in an end-to-end fasion based on the DETR [4].

Curve is a holistic representation of lane line, which naturally eliminates occlusions, requires no post-processing, and can predict a variable number of lines. However, their performance on large and challenging datasets (*e.g.*, CULane [17] and LLAMAS [3]) still lag behind methods of other categories. They also suffer from slow convergence (over 2000 training epochs on TuSimple), high latency architecture (*e.g.*, LSTR [14] uses transformer blocks which

n	**Bézier**	**Polynomial**
2nd	**0.653**	0.945
3rd	**0.471**	0.558
4th	**0.315**	0.330

Table 1. Comparison of n-order Bézier curves and polynomials ($x = \sum_{i=0}^{n} a_i y^i$) on TuSimple [1] *test* set (**lower is better**). Since the official metrics are too lose to show any meaningful difference, we use the fine-grained LPD metric following [23].

are difficult to optimize for low latency). We attribute their failure to the difficult-to-optimize and abstract polynomial coefficients. We propose to use the parametric Bézier curve, which is defined by actual control points on the image coordinate system[1], to address these problems.

Bézier curve in Deep Learning. To our knowledge, the only known successful application of Bézier curves in deep learning is the ABCNet [15], which uses cubic Bézier curve for text spotting. However, their method cannot be directly used for our tasks. First, this method still uses NMS so that it cannot be end-to-end. We show in our work that NMS is not necessary so that our method can be an end-to-end solution. Second, it calculates L_1 loss directly on the sparse Bézier control points, which results in difficulties of optimization. We address this problem in our work by leveraging a fine-grained sampling loss. In addition, we propose the feature flip fusion module, which is specifically designed for the lane detection task.

3. BézierLaneNet

3.1. Overview

Preliminaries on Bézier Curve. The Bézier curve's formulation is shown in Equation (1), which is a parametric curve defined by $n+1$ control points:

$$\mathcal{B}(t) = \sum_{i=0}^{n} b_{i,n}(t)\mathcal{P}_i,\ 0 \leq t \leq 1, \tag{1}$$

where $\mathcal{P}_i$ is the $i-th$ control point, $b_{i,n}$ are Bernstein basis polynomials of degree n:

$$b_{i,n} = C_n^i t^i (1-t)^{n-i},\ i = 0, ..., n. \tag{2}$$

We use the classic cubic Bézier curve ($n = 3$), which is empirically found sufficient for modeling lane lines. It shows better ground truth fitting ability than 3rd order polynomial (Table 1), which is the base function for previous curve-based methods [14, 23]. Higher-order curves do not bring substantial gains while the high degrees of freedom leads to instability. All coordinates for points discussed here are relative to the image size (*i.e.*, mostly in range $[0, 1]$).

[1] Actually control points of Bézier curves can be outside the image, but statistically that rarely happens in autonomous driving scenes.

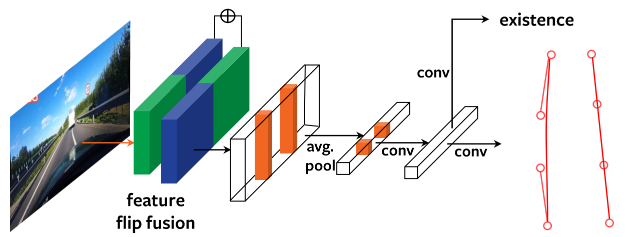

Figure 2. Pipeline. Feature from a typical encoder (*e.g.*, ResNet) is strengthened by feature flip fusion, then pooled to 1D and two 1D convolution layers are applied. At last the network predicts Bézier curves through a classification branch and a regression branch.

The Proposed Architecture. The overall model architecture is shown in Figure 2. Specifically, we use layer-3 feature of ResNets [9] as backbone following RESA [31], but we replace the dilation inside the backbone network by two dilated blocks outside with dilation rates $[4, 8]$ [6]. This strikes a better speed-accuracy trade-off for our method, which leaves a $16\times$ down-sampled feature map with a larger receptive field. We then add the feature flip fusion module (Section 3.2) to aggregate opposite lane features. The enriched feature map ($C \times \frac{H}{16} \times \frac{W}{16}$) is then pooled to ($C \times \frac{W}{16}$) by average pooling, resulting in $\frac{W}{16}$ proposals (50 for CULane [17]). Two 1×3 1D convolutions are used to transform the pooled features, while also conveniently modeling interactions between nearby lane proposals, guiding the network to learn a substitute for the non-maximal suppression (NMS) function. Lastly, the final prediction is obtained by the classification and regression branches (each is only one 1×1 1D convolution). The outputs are $\frac{W}{16} \times 8$ for regression of 4 control points, and $\frac{W}{16} \times 1$ for existence of lane line object.

3.2. Feature Flip Fusion

By modeling lane lines as holistic curves, we focus on the geometric properties of individual lane lines (*e.g.*, thin, long, and continuous). Now we consider the global structure of lanes from a front-mounted camera view in driving scenes. Roads have equally spaced lane lines, which appear symmetrical and this property is worth modeling. For instance, the existence of left ego lane line should very likely indicate its right counterpart, the structure of immediate left lane could help describe the immediate right lane, *etc*.

To exploit this property, we fuse the feature map with its horizontally flipped version (Figure 3). Specifically, two separate convolution and normalization layers transform each feature map, they are then added together before a ReLU activation. With this module, we expect the model to base its predictions on both feature maps.

To account for the slight misalignment of camera captured image (*e.g.*, rotated, turning, non-paired), we apply

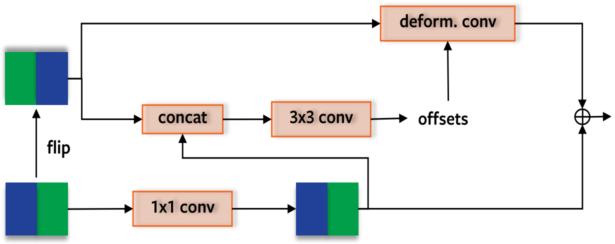

Figure 3. Feature flip fusion. Alignment is achieved by calculating deformable convolution offsets, conditioned on both the flipped and original feature map. Best viewed in color.

deformable convolution [32] with kernel size 3×3 for the flipped feature map while learning the offsets conditioned on the original feature map for feature alignment.

We add an auxiliary binary segmentation branch (to segment lane line or non-lane line areas, which would be removed after training) to the ResNet backbone, and we expect it to enforce the learning of spatial details. Interestingly, we find this auxiliary branch improves the performance only when it works with the feature fusion. This is because the localization of the segmentation task may provide a more spatially-accurate feature map, which in turn supports accurate fusion between the flipped features.

Visualizations are shown in Figure 4, from which we can see that the flipped feature does correct the error caused by the asymmetry introduced by the car (Figure 4(a)).

3.3. End-to-end Fit of a Bézier Curve

Distances Between Bézier Curves. The key to learning Bézier curves is to define a good distance metric measuring the distances between the ground truth curve and prediction. Naively, one can directly calculate the mean L_1 distance between Bézier curve control points, as in ABC-Net [15]. However, as shown in Figure 5(a), a large L_1 error in curvature control points can demonstrate a very small visual distance between Bézier curves, especially on small or medium curvatures (which is often the case for lane lines). Since Bézier curves are parameterized by $t \in [0, 1]$, we propose the more reasonable sampling loss for Bézier curves (Figure 5(b)), by sampling curves at a uniformly spaced set of t values (T), which means equal curve length between adjacent sample points. The t values can be further transformed by a re-parameterization function $f(t)$. Specifically, given Bézier curves $\mathcal{B}(t), \hat{\mathcal{B}}(t)$, the sampling loss $\mathcal{L}_{reg}$ is:

$$\mathcal{L}_{reg} = \frac{1}{n} \sum_{t \in T} ||\mathcal{B}(f(t)) - \hat{\mathcal{B}}(f(t))||_1, \tag{3}$$

where n is the total number of sampled points and is set to 100. We empirically find $f(t) = t$ works well. This simple yet effective loss formulation makes our model easy to converge and less sensitive to hyper-parameters that typically involved in other curved-based or point detection-based methods, *e.g.*, loss weighting for endpoints loss [14] and line length loss [24] (see Figure 5(b,c)).

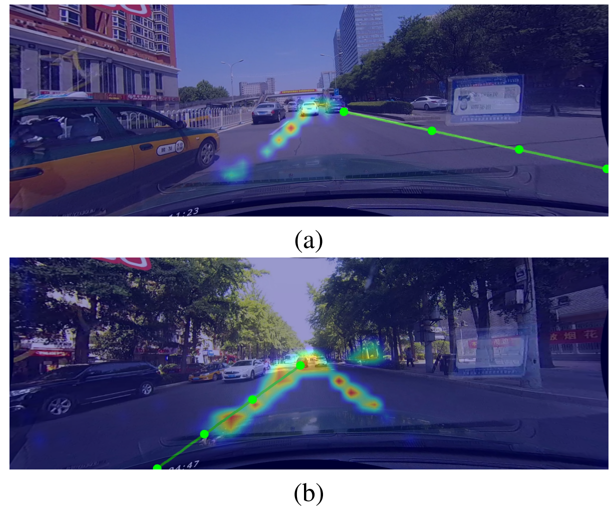

Figure 4. Grad-CAM [22] visualization on the last layer of ResNet backbone. (a) Our model can infer existence of an ill-marked lane line, from clear markings and cars around the opposite line. Note that the car is deviated to the left, this scene was not captured with perfect symmetry. (b) When entire road lacks clear marking, both sides are used for a better prediction. Best viewed in color.

Bézier Ground Truth Generation. Now we introduce the generation of Bézier curve ground truth. Since lane datasets are currently annotated by on-line key points, we need the Bézier control points for the above sampling loss. Given the annotated points $\{(k_{x_i}, k_{y_i})\}_{i=1}^{m}$ on one lane line, where (k_{x_i}, k_{y_i}) denotes the 2D-coordinates of the i-th point. Our goal is to obtain control points $\{\mathcal{P}_i(x_i, y_i)\}_{i=1}^{n}$. Similarly to [15], we use standard least squares fitting:

$$\begin{bmatrix} \mathcal{P}_0 \\ \mathcal{P}_1 \\ \vdots \\ \mathcal{P}_n \end{bmatrix} = \begin{bmatrix} k_{x_0} & k_{y_0} \\ k_{x_1} & k_{y_1} \\ \vdots & \vdots \\ k_{x_m} & k_{y_m} \end{bmatrix} \begin{bmatrix} b_{0,n}(t_0) & \cdots & b_{n,n}(t_0) \\ b_{0,n}(t_1) & \cdots & b_{n,n}(t_1) \\ \vdots & \ddots & \vdots \\ b_{0,n}(t_m) & \cdots & b_{n,n}(t_m) \end{bmatrix}^{\mathrm{T}} \tag{4}$$

$\{t_i\}_{i=0}^{m}$ is uniformly sampled from 0 to 1. Different from [15], we do not restrict ground truth to have same endpoints as original annotations, which leads to better quality labels.

Label and Prediction Matching. After obtaining the ground truth, in training, we perform a one-to-one assignment between G labels and N predictions ($G < N$) using optimal bipartite matching, to attain a fully end-to-end pipeline. Following Wang *et al.* [28], we find a G-permutation of N predictions $\pi \in \Pi_G^N$ that formulates the best bipartite matching:

$$\hat{\pi} = \underset{\pi \in \Pi_G^N}{\arg\max} \sum_{i}^{G} Q_{i,\pi(i)}, \tag{5}$$

$$Q_{i,\pi(i)} = \left(\hat{p}_{\pi(i)}\right)^{1-\alpha} \cdot \left(1 - L_1\left(b_i, \hat{b}_{\pi(i)}\right)\right)^{\alpha}, \tag{6}$$

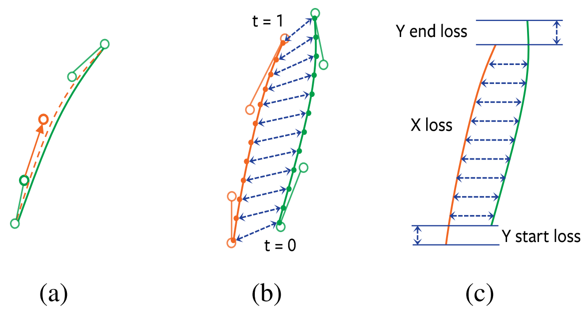

Figure 5. Lane loss functions. (a) The L_1 distance of control points is not highly correlated with the actual distance between curves. (b) The proposed sampling loss is one unified distance metric by t-sampling. (c) Typical loss for polynomial regression [14], at least 3 separate losses are required: y-sampling loss, y start point loss, y end point loss.

where $Q_{i,\pi(i)} \in [0,1]$ represents matching quality of the i-th label with the $\pi(i)$-th prediction, based on L_1 distance between curves $b_i, \hat{b}_{\pi(i)}$ (sampling loss) and class score $\hat{p}_{\pi(i)}$. α is set to 0.8 by default. The above equations can be efficiently solved by the well-known Hungarian algorithm.

Wang *et al*. [28] also use a spatial prior that restricts the matched prediction to a spatial neighborhood of the label (object center distance, the *centerness* prior in FCOS [26]). However, since lots of lanes are long lines with a large slope, this centerness prior is not useful. See Supplementary Section 5 for more investigations on matching priors.

Overall Loss. Other than Bézier curve sampling loss, there is also the classification loss $\mathcal{L}_c$ for the lane object classification (existence) branch. Since the imbalance between positive and negative examples is not as severe in lane detection as in object detection, instead of the focal loss [13], we use the simple weighted binary cross-entropy loss:

$$\mathcal{L}_{cls} = -(y\log(p) + w(1-y)\log(1-p)), \tag{7}$$

where w is the weighting for negative samples, which is set to 0.4 in all experiments. The loss $\mathcal{L}_{seg}$ for the binary segmentation branch (Section 3.2) takes the same format.

The overall loss is a weighted sum of all three losses:

$$\mathcal{L} = \lambda_1\mathcal{L}_{reg} + \lambda_2\mathcal{L}_{cls} + \lambda_3\mathcal{L}_{seg}, \tag{8}$$

where $\lambda_1, \lambda_2, \lambda_3$ are set to $1, 0.1, 0.75$, respectively.

4. Experiments

4.1. Datasets

To evaluate the proposed method, we conduct experiments on three well-known datasets: TuSimple [1], CULane [17] and LLAMAS [3]. TuSimple dataset was collected on highways with high-quality images, under fair weather conditions. CULane dataset contains more complex urban driving scenarios, including shades, extreme illuminations, and road congestion. LLAMAS is a newly formed large-scale dataset, it is the only lane detection benchmark without public *test* set labels. Details of these datasets can be found in Table 2.

Dataset	Train	Val	Test	Resolution	#Lines
TuSimple [1]	3268	358	2782	720×1280	≤ 5
CULane [17]	88880	9675	34680	590×1640	≤ 4
LLAMAS [3]	58269	20844	20929	717×1276	$\leq 4^*$

Table 2. Details of datasets. *Number of lines in LLAMAS dataset is more than 4, but official metric only evaluates 4 lines.

4.2. Evalutaion Metics

For CULane [17] and LLAMAS [3], the official metric is F1 score from [17]:

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{9}$$

where $\text{Precision} = \frac{TP}{TP+FP}$ and $\text{Recall} = \frac{TP}{TP+FN}$. Lines are assumed to be 30 pixels wide, prediction and ground truth lines with pixel IoU over 0.5 are considered a match.

For TuSimple [1] dataset, the official metrics include Accuracy, false positive rate (FPR), and false negative rate (FNR). Accuracy is computed as $\frac{N_{pred}}{N_{gt}}$, where N_{pred} is the number of correctly predicted on-line points and N_{gt} is the number of ground truth on-line points.

4.3. Implementation Details

Fair Comparison. To fairly compare among different state-of-the-art methods, we re-implement representative methods [14, 17, 31] in a unified PyTorch framework. We Also provide a semantic segmentation baseline [5] originally proposed in [17]. All our implementations do **not** use *val* set in training, and tune hyper-parameters **only** on *val* set. Some methods with reliable open-source codes are reported from their own codes [19, 23, 24]. For platform sensitive metric Frames-Per-Second (FPS), we re-evaluate all reported methods on the same RTX 2080 Ti platform. More details for implementations and FPS tests are in Supplementary Sections 1 to 3.

Training. We train 400, 36, 20 epochs for TuSimple, CULane, and LLAMAS, respectively (training of our model takes only 12 GPU hours on a single RTX 2080 Ti), and the input resolution is 288×800 for CULane [17] and 360×640 for others, following common practice. Other than these, all hyper-parameters are tuned on CULane [17] *val* set and remain the same for our method across datasets. We use Adam optimizer with learning rate 6×10^{-4}, weight decay 1×10^{-4}, batch size 20, Cosine Annealing learning rate schedule as in [24]. Data augmentation includes random affine transforms, random horizontal flip, and color jitter.

Method	CULane [17]											TuSimple [1]				
	Ep.	Total	Normal	Crowd	Night	No line	Shadow	Arrow	Dazzle light	Curve	Cross ↓	train+val	Ep.	Acc.	FPR ↓	FNR ↓
Segmentation-based																
Baseline (ResNet-18)*	12	65.30	85.45	62.63	61.04	33.88	51.72	78.15	53.05	59.70	1915		50	94.25	0.088	0.089
Baseline (ResNet-34)*	12	69.92	89.46	66.66	65.38	40.43	62.17	83.18	58.51	63.00	1713		50	95.31	0.064	0.062
Baseline (ResNet-101)*	12	71.37	90.11	67.89	67.01	43.10	70.56	85.09	61.77	65.47	1883		50	95.19	0.062	0.062
SCNN (ResNet-18) [17]*	12	72.19	90.98	70.17	66.54	43.12	66.31	85.62	62.20	65.58	1808		50	94.77	0.075	0.074
SCNN (ResNet-34) [17]*	12	72.70	91.06	70.41	67.75	44.64	68.98	86.50	61.57	65.75	2017		50	95.25	0.063	0.063
SCNN (ResNet-101) [17]*	12	73.58	91.10	71.43	68.53	46.39	72.61	86.87	61.95	67.01	1720		50	**95.69**	0.052	0.050
UFLD (ResNet-18) [19]**	50	68.4	87.7	66.0	62.1	40.2	62.8	81.0	58.4	57.9	1743	-	–	–	–	–
UFLD (ResNet-34) [19]**	50	72.3	90.7	70.2	66.7	44.4	69.3	85.7	59.5	69.5	2037	-	–	–	–	–
RESA (ResNet-18) [31]*	12	72.90	91.23	70.57	67.16	45.24	68.01	86.56	64.32	66.19	1679		50	95.24	0.069	0.057
RESA (ResNet-34) [31]*	12	73.66	91.31	71.80	67.54	46.57	72.74	86.94	64.46	67.31	1701		50	95.15	0.069	0.059
RESA (ResNet-101) [31]*	12	**74.04**	91.45	71.51	69.01	46.54	75.83	87.75	63.90	68.24	1522		50	95.56	0.058	0.051
Point detection-based																
FastDraw (ResNet-18) [18]	–	–	–	–	–	–	–	–	–	–	–	✓	7	94.9	0.061	0.047
CurveLanes-NAS-S [29]	12	71.4	88.3	68.6	66.2	47.9	68.0	82.5	63.2	66.0	2817	-	–	–	–	–
CurveLanes-NAS-M [29]	12	73.5	90.2	70.5	68.2	48.8	69.3	85.7	65.9	67.5	2359	-	–	–	–	–
CurveLanes-NAS-L [29]	12	74.8	90.7	72.3	68.9	49.4	70.1	85.8	67.7	68.4	1746	-	–	–	–	–
LaneATT (ResNet-18) [24]**	15	74.88	90.98	72.78	68.61	48.23	69.68	85.44	65.43	63.18	1163	✓	100	95.57	0.036	0.030
LaneATT (ResNet-34) [24]**	15	76.42	91.94	74.76	70.32	49.17	77.68	88.14	65.92	68.07	1323	✓	100	95.63	0.035	0.029
LaneATT (ResNet-122) [24]**	15	**76.79**	91.50	76.04	70.43	50.29	75.96	86.16	68.99	63.99	1265	✓	100	**96.10**	0.056	0.022
Curve-based																
PolyLaneNet (EfficientNet-B0) [23]**	–	–	–	–	–	–	–	–	–	–	–	✓	2695	93.36	0.094	0.093
LSTR (ResNet-18, 1×) [14]*	–	–	–	–	–	–	–	–	–	–	–		2000	95.06	0.049	0.042
LSTR (ResNet-18, 2×) [14]*	150	68.72	86.78	67.34	59.92	40.10	59.82	78.66	56.63	56.64	1166	-	–	–	–	–
BézierLaneNet (ResNet-18)	36	73.67	90.22	71.55	68.70	45.30	70.91	84.09	62.49	58.98	996		400	95.41	0.053	0.046
BézierLaneNet (ResNet-34)	36	**75.57**	91.59	73.20	69.90	48.05	76.74	87.16	69.20	62.45	888		400	**95.65**	0.051	0.039

Table 3. Results on *test* set of CULane [17] and TuSimple [1]. *reproduced results in our code framework, best performance from three random runs. **reported from reliable open-source codes from the authors.

Testing. No post-processing is required for curve methods. Standard Gaussian blur and row selection post-processing is applied to segmentation methods. NMS is used for LaneATT [24], while we remove its post-inference B-Spline interpolation in CULane [17], to align with our framework.

4.4. Comparisons

Overview. Experimental results are shown in Tables 3 and 4. TuSimple [1] is a small dataset that features clear-weather highway scenes and has a relatively easy metric, most methods thrive in this dataset. Thus, we mainly focus on the other two large-scale datasets [3, 17], where there is still a rather clear difference between methods. For high-performance methods ($> 70\%$ F1 on CULane [17]), we also show efficiency metrics (FPS, Parameter count) in Table 5.

Comparison with Curve-based Methods. As shown in Tables 3 and 4, in all datasets, BézierLaneNet outperforms previous curve-based methods [14, 23] by a clear margin, advances the state-of-the-art of curve-based methods by 6.85% on CULane [17] and 6.77% on LLAMAS [3]. Thanks to our fully convolutional and fully end-to-end pipeline, BézierLaneNet runs over $2\times$ faster than LSTR [14]. LSTR has a speed bottleneck from transformer architecture, the $1\times$ and $2\times$ model have FPS 98 and 97, respectively[2]. While curves are difficult to learn, our method converges 4-5$\times$ faster than LSTR. For the first time, an elegant curve-based method can challenge well-designed segmentation methods or point detection methods on these datasets while showing a favorable trade-off, with an acceptable convergence time.

Comparison with Segmentation-based Methods. These methods tend to have a low speed due to recurrent feature aggregation [17, 31], and the use of high-resolution feature map [5, 17, 31]. BézierLaneNet outperforms them in both speed and accuracy. Our small models even compare favorably against RESA [31] and SCNN [17] with large ResNet-101 backbone, surpassing them in CULane [17] with a clear margin ($1 \sim 2\%$). On LLAMAS [3], where the dataset restricts testing on 4 center lines, the segmentation approach shows strong performance (Table 4). Nevertheless, our ResNet-34 model still outperforms SCNN by 0.92%.

UFLD [19] reformulates segmentation to row-wise classification on a down-sampled feature map to achieve fast speed, at the cost of accuracy. Compared to us, UFLD (ResNet-34) is 0.9% lower on CULane **Normal**, while 7.4%, 3.0%, 3.2% worse on **Shadow**, **Crowd**, **Night**, respectively. Overall, our method with the same backbones outperforms UFLD by $3 \sim 5\%$, while being faster on ResNet-34. Besides, UFLD uses large fully-connected layers to optimize latency, which causes a huge model size (the largest in Table 5).

A drawback for all segmentation methods is the weaker performance on **Dazzle Light**. Per-pixel (or per-pixel grid for UFLD [19]) segmentation methods may rely on information from local textures, which is destroyed by extreme exposure to light. While our method predicts lane lines as holistic curves, hence robust to changes in local textures.

Comparison with Point Detection-based Methods. Xu *et al.* [29] finds a series of point detection-based models with

[2] The original 420 FPS report from LSTR paper [14], is throughput with batch size 16, detailed discussions in Supplementary.

Method	LLAMAS [3]			
	Ep.	F1	Precision	Recall
Segmentation-based				
Baseline (ResNet-34)*	18	93.43	92.61	94.27
SCNN (ResNet-34) [17]*	18	94.25	94.11	94.39
Point detection-based				
LaneATT (ResNet-18) [24]**	15	93.46	96.92	90.24
LaneATT (ResNet-34) [24]**	15	93.74	96.79	90.88
LaneATT (ResNet-122) [24]**	15	93.54	96.82	90.47
Curve-based				
PolyLaneNet (EfficientNet-B0) [23]**	75	88.40	88.87	87.93
BézierLaneNet (ResNet-18)	20	94.91	95.71	94.13
BézierLaneNet (ResNet-34)	20	**95.17**	95.89	94.46

Table 4. Results from LLAMAS [3] test server.

neural architecture search techniques called CurveLanes-NAS. Despite its complex pipeline and extensive architecture search for the best accuracy-FLOPs trade-off, our simple ResNet-34 backbone model (29.3 GFLOPs) still surpasses its large model (86.5 GFLOPs) by 0.8% on CULane. CurveLanes-NAS also performs worse under occlusions, a similar drawback as the segmentation methods without recurrent feature fusion [5, 19]. As shown in Table 3, with similar model capacity compared to our ResNet-34 model, CurveLanes-NAS-M (35.7 GFLOPs) is 1.4% worse on **Normal** scenes, but the gap on **Shadow** and **Crowd** are 7.4% and 2.7%.

Recently, LaneATT [24] achieves higher performance with a point detection network. However, their design is not fully end-to-end (requires Non-Maximal Suppression (NMS)), based on heuristic anchors (>1000), which are calculated directly from the dataset's statistics, thus may systematically pose difficulties in generalization. Still, with ResNet-34, our method outperforms LaneATT on the LLAMAS [3] test server (1.43%), with a significantly higher recall (3.58%). We also achieve comparable performance to LaneATT on TuSimple [1] using only the *train* set, and only $\sim 1\%$ worse on CULane. Our method performs significantly better in **Dazzle Light** (3.3% better), comparably in **Night** (0.4% lower). It also has a lower False Positive (FP) rate on Crossroad scenes (**Cross**), even though LaneATT shows an extremely low-FP characteristic (large Precision-Recall gap in Table 4). Methods that rely on heuristic anchors [24] or heuristic decoding process [17,19,29,31] tend to have more false predictions in this scene. Moreover, the NMS is a sequential process that could have unstable runtime in real-world applications. Even when NMS was not evaluated on real inputs, our models are 29%, 28% faster, have $2.9\times$, $2.3\times$ fewer parameters, compared to LaneATT on ResNet-18 and ResNet-34 backbones, respectively.

To summarize, previous curve-based methods (PolyLaneNet [23], LSTR [14]) have significantly worse performance. Fast methods trades either accuracy (UFLD [19]) or model size (UFLD [19], LaneATT [24]) for speed. Accurate methods either discards the end-to-end pipeline (LaneATT [24]), or entirely fails the real-time requirement (SCNN [17], RESA [31]). While our BézierLaneNet is fully end-to-end, fast (>150 FPS), light-weight (<10 million parameters) and maintains consistent high accuracy across datasets.

Method	FPS ↑	Params (M) ↓
Segmentation-based (ignored post-processing time)		
Baseline (ResNet-101)	27	43.56
SCNN (ResNet-18) [17]	21	12.63
SCNN (ResNet-34) [17]	21	22.74
SCNN (ResNet-101) [17]	14	44.15
UFLD (ResNet-34) [19]	144	71.58
RESA (ResNet-18) [31]	68	6.61
RESA (ResNet-34) [31]	54	11.99
RESA (ResNet-101) [31]	25	31.46
Point detection-based (ignored NMS time in real images)		
LaneATT (ResNet-18) [24]	165	12.02
LaneATT (ResNet-34) [24]	117	22.13
LaneATT (ResNet-122) [24]	26	8.55
Curve-based (entirely end-to-end)		
BézierLaneNet (ResNet-18)	**213**	**4.10**
BézierLaneNet (ResNet-34)	150	9.49

Table 5. FPS (*image/s*) and model size. All FPS results are tested with 360×640 random inputs on the same platform. Here only shows models with $> 70\%$ CULane [17] F1 score.

4.5. Analysis

Although we develop our method by tuning on the *val* set, we re-run ablation studies with ResNet-34 backbone (including our full method) and report performance on the CULane *test* set for clear comparison.

Curve representation	**F1**
Cubic Bézier curve baseline	68.89
3rd Polynomial baseline	1.49
BézierLaneNet	75.41
3rd Polynomial from BézierLaneNet	5.01

Table 6. Curve representations. Baselines directly predict curve coefficients without feature flip fusion.

Importance of Parametric Bézier Curve. We first replace the Bézier curve prediction with a 3rd order polynomial, adding auxiliary losses for start and end points. As shown in Table 6, polynomials catastrophically fail to converge in our fully convolutional network, even when trained with 150 epochs (details in Supplementary Section 2). Then we consider modifying the LSTR [14] to predict cubic Bézier curves, the performance is similar to predicting polynomials. We conclude that heavy MLP may be necessary to learn polynomials [14, 23], while predicting Bézier control points from position-aware CNN is the best choice. The transformer-based LSTR decoder destroys the fine spatial information, suppresses the advancement of curve function.
Feature Flip Fusion Design. As shown in Table 7, feature flip fusion brings 4.07% improvement. We also find that the auxiliary segmentation loss can regularize and increase

CP	SP	Flip	Deform	Seg	F1
✓					63.74
	✓				68.89
	✓			✓	65.82
	✓	✓			70.28
	✓	✓	✓		72.96
	✓	✓		✓	73.97
	✓	✓	✓	✓	**75.41**

Table 7. Ablations. **CP**: Control point loss [15]. **SP**: The proposed sampling loss. **Flip**: The feature flip fusion module. **Deform**: Employ the deformable convolution in feature flip fusion. **Seg**: Auxiliary segmentation loss.

the performance further, by 2.45%. It is worth noting that auxiliary loss only works with feature fusion, it can lead to degenerated results when directly applied on the baseline (-3.07%). A standard 3×3 convolution performs worse than deformable convolution, by 2.68% and 1.44%, before and after adding the auxiliary segmentation loss, respectively. We attribute this to the effects of feature alignment.

Bézier Curve Fitting Loss. As shown in Table 7, replacing the sampling loss by direct loss on control points lead to inferior performance (-5.15% in the baseline setup). Inspired by the success of IoU loss in object detection. We also implemented a IoU loss (formulas in Supplementary Section 4) for the convex hull of Bézier control points. However, the convex hull of close-to-straight lane lines are too small, the IoU loss is numerically unstable, thus failing to facilitate the sampling loss.

Model	Aug	F1
LSTR (ResNet-18, 2×) [14]	✓	68.72
LSTR (ResNet-18, 2×) [14]		39.77(-28.95)
BézierLaneNet (ResNet-34)	✓	75.41
BézierLaneNet (ResNet-34)		55.11(-20.30)

Table 8. Augmentation ablations. **Aug**: Strong data augmentation.

Importance of Strong Data Augmentation. Strong data augmentation is defined by a series of affine transforms and color distortions, the exact policy may slightly vary for different methods. For instance, we use random affine transform, random horizontal flip, and color jitter. LSTR [14] also uses random lighting. Default augmentation includes only a small rotation (3 degrees). As shown in Table 8, strong augmentation is essential to avoid over-fitting for curve-based methods.

For segmentation-based methods [5, 17, 31], we fast validated strong augmentation on the smaller TuSimple [1] dataset. All shows a $1 \sim 2\%$ degradation. This suggests that they may be robust due to per-pixel prediction and heuristic post-processing. But they highly rely on learning the distribution of local features such as texture, which could become confusing by strong augmentation.

4.6. Limitations and Discussions

Curves are indeed a natural representation of lane lines. However, their elegance in modeling inevitably brings a drawback. It is difficult for the curvature coefficients to generalize when the data distribution is highly biased (almost all lane lines are straight lines in CULane). Our Bézier curve approach has already alleviated this problem to some extent and has achieved an acceptable performance (62.45) in CULane **Curve**. On datasets such as TuSimple and LLAMAS [1, 3], where the curvature distribution is fair enough for learning, our method achieves even better performance. To handle broader corner cases, *e.g.*, sharp turns, blockages and bad weather, datasets such as [21, 25, 29] may be useful.

The feature flip fusion is specifically designed for a front-mounted camera, which is the typical use case of deep lane detectors. Nevertheless, there is still a strong inductive bias by assuming scene symmetry. In future work, it would be interesting to find a replacement for this module, to achieve better generalization and to remove the deformable convolution operation, which poses difficulty for effective integration into edge devices such as Jetson.

More discussions in Supplementary Section 7.

5. Conclusions

In this paper, we have proposed BézierLaneNet: a novel fully end-to-end lane detector based on parametric Bézier curves. The on-image Bézier curves are easy to optimize and naturally model the continuous property of lane lines, without heavy designs such as recurrent feature aggregation or heuristic anchors. Besides, a feature flip fusion module is proposed. It efficiently models the symmetry property of the driving scene, while also being robust to slight asymmetries by using deformable convolution. The proposed model has achieved favorable performance on three datasets, defeating all existing methods on the popular LLAMAS benchmark. It is also both fast (>150 FPS) and light-weight (<10 million parameters).

Acknowledgements. This work was supported by the National Key Research and Development Program of China (2019YFC1521104), National Natural Science Foundation of China (72192821, 61972157), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), Shanghai Science and Technology Commission (21511101200, 22YF1420300), and Art major project of National Social Science Fund (I8ZD22), and SenseTime Collaborative Research Grant. We thank Jiaping Qin for guidance on road design and geometry, Yuchen Gong and Pan Chen for helping with CAM visualizations, Zhijun Gong, Jiachen Xu and Jingyu Gong for insightful discussions about math, Fengqi Liu for providing GPUs, Lucas Tabelini for cooperation in evaluating [23, 24], and the CVPR reviewers for constructive comments.

References

[1] TuSimple benchmark. https://github.com/TuSimple/tusimple-benchmark, 2017. 2, 3, 5, 6, 7, 8

[2] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, 2021. 1

[3] Karsten Behrendt and Ryan Soussan. Unsupervised labeled lane markers using maps. In *ICCV*, 2019. 2, 5, 6, 7, 8

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 2

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 1, 5, 6, 7, 8

[6] Qiang Chen, Yingming Wang, Tong Yang, Xiangyu Zhang, Jian Cheng, and Jian Sun. You only look one-level feature. In *CVPR*, 2021. 3

[7] Mohsen Ghafoorian, Cedric Nugteren, Nóra Baka, Olaf Booij, and Michael Hofmann. El-gan: Embedding loss driven generative adversarial networks for lane detection. In *ECCV Workshops*, 2018. 2

[8] Ross Girshick. Fast r-cnn. In *ICCV*, 2015. 1

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[10] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. Recent progress in road and lane detection: a survey. *MVA*, 2014. 1

[11] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning lightweight lane detection cnns by self attention distillation. In *ICCV*, 2019. 2

[12] Xiang Li, Jun Li, Xiaolin Hu, and Jian Yang. Line-cnn: End-to-end traffic line detection with line proposal unit. *ITS*, 2019. 1, 2

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 5

[14] Ruijin Liu, Zejian Yuan, Tie Liu, and Zhiliang Xiong. End-to-end lane shape prediction with transformers. In *WACV*, 2021. 1, 2, 3, 4, 5, 6, 7, 8

[15] Yuliang Liu, Hao Chen, Chunhua Shen, Tong He, Lianwen Jin, and Liangwei Wang. Abcnet: Real-time scene text spotting with adaptive bezier-curve network. In *CVPR*, 2020. 3, 4, 8

[16] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *IV*, 2018. 1, 2

[17] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding. In *AAAI*, 2018. 1, 2, 3, 5, 6, 7, 8

[18] Jonah Philion. Fastdraw: Addressing the long tail of lane detection by adapting a sequential prediction network. In *CVPR*, 2019. 6

[19] Zequn Qin, Huanyu Wang, and Xi Li. Ultra fast structure-aware deep lane detection. In *ECCV*, 2020. 2, 5, 6, 7

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015. 1, 2

[21] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Semantic foggy scene understanding with synthetic data. *IJCV*, 2018. 8

[22] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *CVPR*, 2017. 4

[23] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. Polylanenet: Lane estimation via deep polynomial regression. In *ICPR*, 2020. 1, 2, 3, 5, 6, 7, 8

[24] 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 *CVPR*, 2021. 1, 2, 4, 5, 6, 7, 8

[25] Xin Tan, Ke Xu, Ying Cao, Yiheng Zhang, Lizhuang Ma, and Rynson W. H. Lau. Night-time scene parsing with a large real dataset. *TIP*, 2021. 8

[26] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019. 5

[27] 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 Workshops*, 2019. 2

[28] Jianfeng Wang, Lin Song, Zeming Li, Hongbin Sun, Jian Sun, and Nanning Zheng. End-to-end object detection with fully convolutional network. In *CVPR*, 2021. 2, 4, 5

[29] 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 *ECCV*, 2020. 1, 6, 7, 8

[30] 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 *CVPR Workshops*, 2020. 2

[31] Tu Zheng, Hao Fang, Yi Zhang, Wenjian Tang, Zheng Yang, Haifeng Liu, and Deng Cai. Resa: Recurrent feature-shift aggregator for lane detection. In *AAAI*, 2021. 1, 2, 3, 5, 6, 7, 8

[32] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019. 2, 4