FAST OBJECT DETECTION FOR AUTONOMOUS VEHICLES

Project Report

FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

SUBMITTED BY:

LIKHITHA MANNAM

Abstract:

Real-time object detection is the task of doing object detection in real-time with rapid inference while keeping a base stage of accuracy.

Over time this era has been developed and can be used for diverse purposes , for example use in self sustaining driving. Previously many algorithms and approaches were evolved for this purpose, however self sustaining driving is something wherein precision turns into very crucial.A false positive(fp)[2] can cause deadly accidents. Therefore, in our work, we have tried to look at the vintage works alongside with imposing some changes to improve the accuracy

Keywords: Fast Object,Real time, Faster RCNN, YOLO, YOLOv2, YOLOv3, Gaussian YOLOv3, mAP, fps(frames per second)

1.Introduction:

As we know, in recent years deep learning knowledge has been carried out in diverse different fields , for instance in computer vision. Implementing gps and sensors along with deep getting to know has enabled tendencies in studies inside the discipline of autonomous driving. An independent car is a vehicle with self-riding functionality without any driving force intervention and therefore ought to correctly come across cars, pedestrians, traffic signs, lights, etc. In actual time to make sure safe and correct control decisions . To stumble on such objects, various sensors such as cameras,Lidar and radar are usually used in self sufficient vehicles . Among these numerous styles of sensors, a digicam sensor can pick out the item type based on texture and color capabilities and is greater cost-effective than different sensors. In particular, deep-learning knowledge of based totally item detection using digicam sensors is becoming extra vital in self reliant vehicles as it achieves a higher level of accuracy than humans in phrases of object detection, and therefore it has become an important method in self driving systems.

2. Problem Definition:

Autonomous using has been a promising industry in current years. Both automobile manufactories and IT companies are competitively investing in self-using fields. Companies like Google, Uber, Ford, BMW have already been checking out their self-driving vehicles on the road. Optical imaginative and prescient is an critical component of independent cars. Accurate real-time object detection, such as motors, pedestrians, animals, and road signs, could accelerate the tempo of constructing a self-riding vehicle as secure as human drivers. Image knowledge has been a challenging task for decades. Unlike geometric figures, items in the real global are usually irregular figures. And the identical item seems in numerous shapes when capturing from special angles or the object itself is changing its shape. Besides, pics of objects in the actual global environment are variation to illumination, rotation, scale and occlusion, which makes item detection mission extra challenging. In latest years, a massive improvement in photograph reputation turned into made by means of a chain of CNN based solutions. Faster R-CNN normally show a excessive accuracy however have a disadvantage of a slow detection speed and decrease efficiency. In 2016, a quick object detector, YOLO, become proposed to make bigger item detection to real-time situations. It has a quick detection pace and excessive

efficiency but low accuracy. In current years, to take gain of both varieties of technique and to compensate for their respective disadvantages, item detectors combining numerous schemes were extensively studied. We propose a novel object detection algorithm suitable for self reliant driving based on YOLOv3. YOLOv3 can hit upon multiple objects with a unmarried inference, and its detection pace is consequently extremely fast; in addition, through applying a multi-level detection approach, it may supplement the low accuracy of YOLO and YOLOv2. Based on those advantages, YOLOv3 is appropriate for autonomous using applications, however generally achieves a lower accuracy than a two-degree technique. We are motivated to improve the accuracy while retaining a actual-time object detection capability.

3.Literature Review:

OverFeat is a Convolutional Neural Network version launched in 2013 that jointly performs item recognition, detection, and localization. OverFeat is one in all the most a success detection models triumphing the localization task in the ImageNet Large Scale Visual Recognition Challenge 2013[7]. OverFeat is 8 layers deep, and depends closely on an overlapping scheme that produces detection containers at more than one scales and iteratively aggregates them together into high-self assurance predictions.

VGG16 turned into an try to usurp OverFeat's dominance in object classification and detection by means of exploring the outcomes of intense layer depth. A 16-layer and 19-layer model became produced, setting new benchmarks on localization and classification in the ImageNet ILSVRC 2014 Challenge[6].

Fast R-CNN is a version that attempts to capture the accuracy of deeper models while improving their speed. Fast R-CNN[5] predicts on location proposals, and makes use of shared computation in keeping with location inspiration and truncated SVD factorizations to hurry up the education and prediction time of the model.

YOLO is a recent model that operates at once on photos at the same time as treating object detection as regression in place of category. YOLO has the poorest overall performance out of all of the above models, but more than makes up for it in speed; YOLO is capable of expect at an astounding 45 frames in keeping with second

3.1 CNN:

•**Convolution-**convolution is applied on the input data using a *convolution filter[6]* to produce a *feature map*. Convolution is basically done for feature extraction.

•**Pooling-**After a convolution operation we usually perform *pooling* to reduce the dimensionality.

•In CNN architectures, pooling is typically performed with 2x2 windows, stride 2 and no padding. While convolution is done with 3x3 windows, stride 1 and with padding.

•**Fully Connected-**After the convolution + pooling layers we add a couple of fully connected layers to wrap up the CNN architecture. Fully connected Layer takes the output of convolution and pooling and use them to classify the images to labels.

•**Training**-CNN is trained the same way like ANN, backpropagation with gradient descent.

3.2 RCNN:

In the existing CNN model selective search is used to generate around 2000 different regions from the given image and these regions are cropped and wrapped and then given as input. Also , instead of softmax[3] function, SVM is used.

3.3 Fast RCNN:

Unlike RCNN, Fast RCNN[4] uses a single model which extracts features from the regions, divides them into different classes, and returns the boundary boxes for the identified classes simultaneously.

•Here, the regions of interest are generated after passing the original image through CNN.

3.4 Faster RCNN:

- Here in preference to selective search, region proposed networks are used.
- Faster RCNN takes the characteristic maps from CNN[4] and passes them directly to the Region Proposal Network.
- RPN generates RoI. Then the RoI pooling layer is applied on these proposals to carry down all of the proposals to the same size. Then the proposals are handed to a totally related layer which has a softmax layer and a linear regression layer at its top, to classify and output the bounding packing containers for objects.

  Working of RPN:

- Faster RCNN takes the characteristic maps from CNN and passes them directly to the Region Proposal Network. RPN[3] makes use of a sliding window over these feature maps, and at every window, it generates k Anchor boxes of various shapes and sizes
- For each anchor, RPN predicts two things:the opportunity that an anchor is an object and the bounding container regressor for adjusting the anchors to better shape the item.

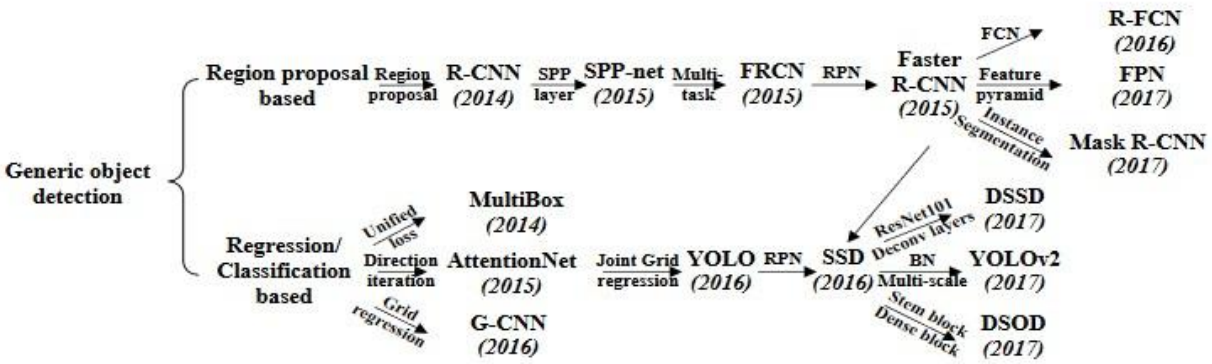
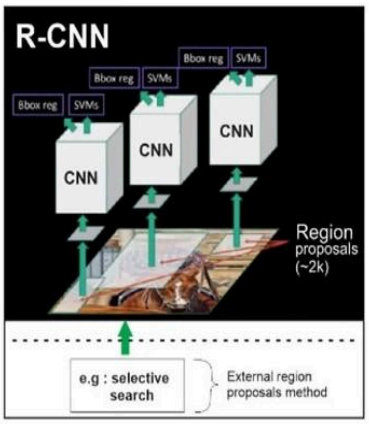
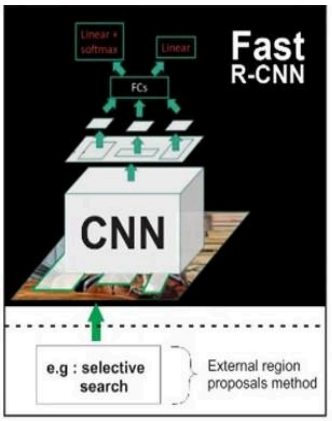
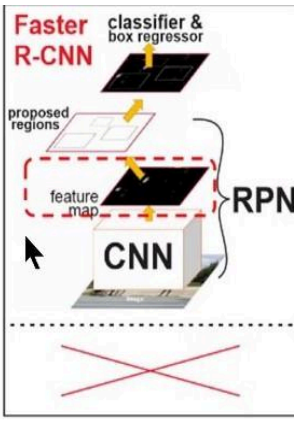


Fig 1. Generic Object detection.







	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

Fig. 2 comparision between R-CNN , Fast RCNN and Faster-RCNN

3.5 YOLO:

Faster RCNN offers a regional of interest region for doing convolution while YOLO detects objects by dividing an image into grid units and does detection and classification at the same time.Unlike faster RCNN[7], it's trained to do classification and bounding box regression at the same time.

3.6 YOLOv2:

YOLOv2 is the second version of the YOLO with the objective of improving the accuracy significantly while making it faster.The changes we make corresponding to YOLO is that here we add batch normalization in convolution layers,use High-resolution classifier[8] and use Convolutional with Anchor Boxes.

3.7 YOLOv3:

YOLOv2 used darknet-19(19-layer network+ 11 layers for object detection). YOLOv3 uses darknet-53(53-layer network+53 layers for object detection). It also includes residual blocks,skip connections and upsampling.In YOLO v3, detection is done by applying 1 x 1 detection kernels of shape 1 x 1 x (B x (5 + C)) on feature maps of three different sizes at three different places in the network ,where B is the number of bounding boxes a cell on the feature map can predict, “5” is for the 4 bounding box attributes and one object confidence, and C is the number of classes.
Compared to YOLOv2, it is better at detecting smaller objects, uses far more bounding boxes and uses 9 anchor boxes(3 for each scale) compared to YOLOv2 which uses 5 anchor boxes. It also refrains from using the softmax function for labelling and has a variation in the loss function.

In YOLOv3, we multiply the objectness score and class score is used to detect the object. For class scores, each class score is predicted using logistic regression and a threshold is used to predict multiple labels for an object. Classes which score higher than the threshold are assigned to the box.
For the loss function , squared errors and cross entropy is used. Cross entropy is used in the last 3 terms, that penalize objectness score(0&1) and class scores. Therefore , we use logistic regression[5] for object confidence score and class score.

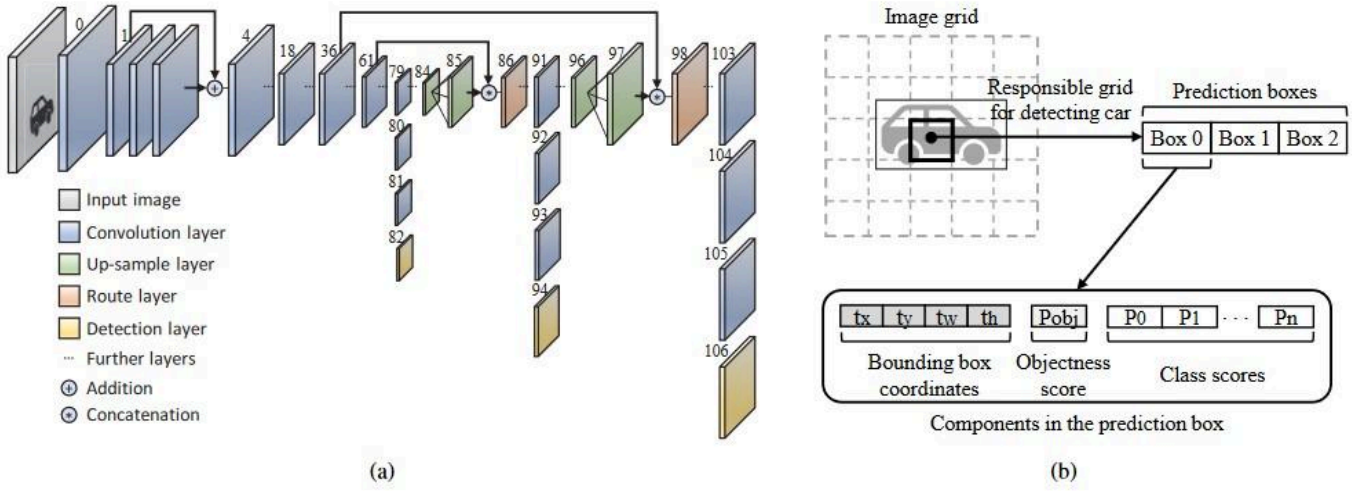


Fig.3 Architecture of yolo v3

.

4.Proposed Methodology:

4.1 By using gaussian modeling :

YOLOv3 , though slower than YOLOv2, is still one of the fastest . Its drawback however is the accuracy. Even though more accurate than the YOLOv2 model, it however isn't one of the most accurate model. In our work , we decided to go through some variations in the YOLOv3 model in order to achieve better accuracy.In our work, we decided to make some changes to the bounding box (bbox) of YOLOv3.

As mentioned earlier, for the loss function, squared errors and cross entropy is used in YOLOv3. Instead we incorporate the gaussian model for the Calculations. Therefore we end up using the negative log likelihood loss.

The gaussian model(or normal) for output y given input x takes in the parameters mean (μ) and variance (Σ) functions. These sigma and variance values are calculated wrt to the bounding box coordinates.

4.2 Reconstructing the loss function

The sum of the squared error loss for boundary boxes , and the binary cross-entropy loss for the objectness and class were used by the YOLOv3 model.
Considering that boundary boxes coordinates are the output as Gaussian parameters so loss function of boundary boxes is reconstructed as a -ve log likelihood loss. By this the loss function for class and objectness is not changed.
Moreover Because YOLOv3 uses the sum of the squared error loss for bbox, it's far not able to cope with noisy information at some stage in training. However, the redesigned loss function of bbox can provide a penalty to the loss via the uncertainty for inconsistent information at some stage in training. That is, the model can be found out by concentrating on consistent records. Therefore, the redesigned loss function of bbox makes the model more robust to noisy statistics . Through this loss attenuation , it's far possible to improve the accuracy of the algorithm

4.3. Utilising localized uncertainty

The proposed Gaussian YOLOv3 can achieve the uncertainty of bbox for every detection item in an image. Because it is not an uncertainty for the complete image, it's far feasible to use uncertainty to every detection result. YOLOv3 considers handiest the objectness rating and class scores in the course of item detection, and cannot don't forget the bbox score at some point of the detection system due to the fact the score statistics for the bbox coordinates is unknown. However, Gaussian YOLOv3 can output the localization uncertainty, that is the score of bbox. Therefore, localization uncertainty may be considered at the side of the objectness rating and sophistication scores throughout the detection technique. The proposed algorithm applies localization uncertainty to the detection standards of YOLOv3 such that bbox with excessive uncertainty among the anticipated results is filtered via the detection procedure. In this way, predictions with high self belief of objectness, class, and bbox are ultimately selected. Thus, Gaussian YOLOv3 can reduce the FP and growth the TP, which results in enhancing the detection accuracy.

5. Experimental Results:

We used the KITTI dataset [7] and the BDD dataset [9]. The KITTI dataset includes 3 classes: car, cyclist, and pedestrian, and includes 7,481 pictures for training and 7,518 pictures for testing. The training and validation units are made by randomly dividing the training set in half [8]. The BDD dataset consists of ten classes: bike, bus, car, motor, person, rider, visitors light, site visitors sign, train, and truck. The ratio of training, validation, and test set is 7:1:2. We used a test set for the performance assessment. In general, the IOU threshold (TH) of the KITTI dataset is given as 0.7 for motors and 0.5 for cyclists and pedestrians [7], while the IOU TH of the BDD dataset is 0.75 for all classes [9]. In both YOLOv3 and Gaussian YOLOv3 training, the batch size is 64 and the learning rate is 0.0001.The anchor length is extracted by the usage of k-means clustering for each training set of KITTI and BDD. The anchors used in the training and evaluation are shown in Table 1.

KITTI training set	Anchor 0	Anchor 1	Anchor 2
First detection layer	(49,240)	(82,170)	(118,206)
Second detection layer	(45,76)	(27,172)	(67,116)
Third detection layer	(13,30)	(23,53)	(17,102)

BDD training set			
First detection layer	(73,175)	(141,178)	(144,291)
Second detection layer	(32,97)	(57,64)	(92,109)
Third detection layer	(7,10)	(14,24)	(27,43)

Table 1: Results of anchor boxes of training sets.

Table 2 indicates the overall performance of the proposed algorithm using the KITTI validation set. Row 1 indicates the experimental effects of Gaussian YOLOv3 with a 512 × 512 resolution and row 2 suggests the experimental effects of Gaussian YOLOv3 with a 704 × 704 resolution.

Detection algorithm										mAP(%)	fps	Input size
	E	M	H	E	M	H	E	M	H			
Gaussian YOLOv3	90.61	90.20	81.19	87.84	79.57	72.30	89.31	81.30	80.20	83.61	43.13	512×512
Gaussian YOLOv3	98.74	90.48	89.47	87.85	79.96	76.81	90.08	86.59	81.09	86.79	24.91	704×704

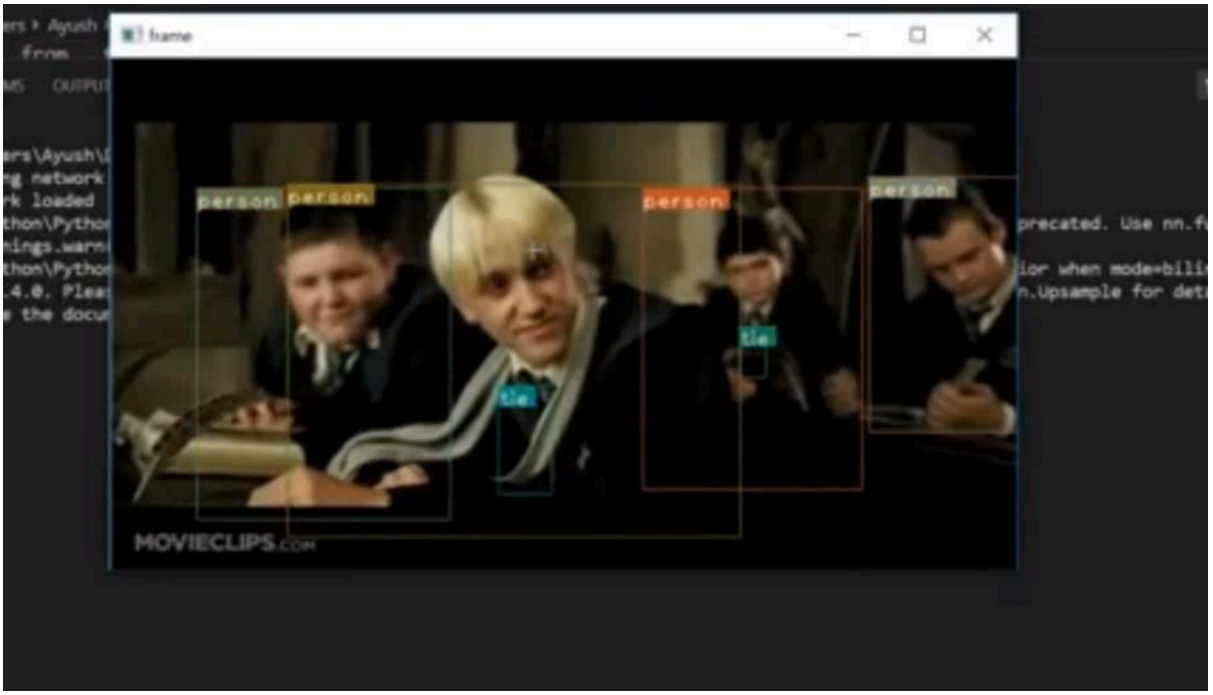
Table 2: Performance results using KITTI validation set. E, M, and H refer to easy, moderate, and hard, respectively

Table 3 indicates the overall performance of the proposed algorithm using the KITTI validation set. Row 1 indicates the experimental effects of Gaussian YOLOv3 with a 512 × 512 resolution and row 2 suggests the experimental effects of Gaussian YOLOv3 with a 736 × 736 resolution.

Detection algorithm	mAP(%)	fps	Input size
Gaussian YOLOv3	18.4	42.5	512×512
Gaussian YOLOv3	20.8	22.5	736×736

Table 3: Performance results using BDD test set.

(a)



(b)

Figure 4(a): Detection results of Gaussian YOLOv3.The first column shows the detection results of the baseline and proposed algorithms on the KITTI validation set whereas the second column shows on the BDD test set.
Figure 4(b): Algorithm when applied on fast-moving random video

6. Conclusion:

A high-accuracy and real-time evaluation speed of an object detection algorithm are extremely essential for the protection and real-time control of autonomous vehicles. Various research associated with camera-based independent riding have been conducted, but are unsatisfactory primarily based on a trade-off among the accuracy and operation pace. For this reason, our project deals with object detection method that achieve the pleasant trade-off between accuracy and speed for self reliant driving. Through Gaussian modeling, loss function reconstruction, and the usage of localization uncertainty, the proposed method improves the accuracy, will increase the TP, and extensively reduces the FP, while maintaining the real-time capability. Compared to the baseline, the proposed Gaussian YOLOv3[4[algorithm improves the mAP with the aid of 3.09 and 3.5 for the KITTI and BDD datasets, respectively. Furthermore, because the proposed method has a higher accuracy than the previous studies with a similar fps, the proposed algorithm is remarkable in phrases of the trade-off among accuracy and detection speed. As a result, the proposed method can extensively improve the camera based object detection system for autonomous driving, and is consequently predicted to contribute significantly to the huge use of autonomous driving applications.

7. References:

[1] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. **A unified multi-scale deep convolutional neural network for fast object detection.** In European conference on computer vision, pages 354–370. Springer, 2016.

[2] Sungjoon Choi, Kyungjae Lee, Sungbin Lim, and Songhwai Oh. **Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling.** In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 6915–6922. IEEE, 2018.

[3] Aleksa Corovi ´ c, Velibor Ili ´ c, Sini ´ sa Duri ˇ c, Malisa Marijan, ´ and Bogdan Pavkovic. **The real-time detection of traffic participants using yolo algorithm.** In 2018 26th Telecommunications Forum (TELFOR), pages 1–4. IEEE, 2018.

[4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: **Object detection via region-based fully convolutional networks**. In Advances in neural information processing systems, pages 379–387, 2016.

[5] Xuerui Dai. Hybridnet: **A fast vehicle detection system for autonomous driving. Signal Processing**: Image Communication, 70:79–88, 2019.

[6] Di Feng, Lars Rosenbaum, and Klaus Dietmayer. **Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection.** In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pages 3266–3273. IEEE, 2018.

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. **Are we ready for autonomous driving? the kitti vision benchmark suite**. In 2012 IEEE Conference on Computer Vision and Pattern Recognition, pages 3354–3361. IEEE, 2012.

[8] Bichen Wu, Forrest Iandola, Peter H Jin, and Kurt Keutzer. **Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving.** In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 129–137, 2017.

[9] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: **A diverse driving video database with scalable annotation tooling**. arXiv preprint arXiv:1805.04687, 2018.