A Lightweight Deep Learning Model for Vehicle Viewpoint Estimation from Dashcam Images

Simone Magistri* [1], Francesco Sambo† [2], Fabio Schoen*, Douglas Coimbra de Andrade†, Matteo Simoncini*†, Stefano Caprasecca†, Luca Kubin†, Luca Bravi†, Leonardo Taccari†

***Abstract*— Vehicle viewpoint estimation from vehicle cameras is a crucial component of road scene understanding.**

In this paper, we propose a deep lightweight method to predict vehicle viewpoint from a single RGB dashcam image. To this aim, we customize and adapt state-of-the-art deep learning techniques for general object viewpoint estimation to the vehicle viewpoint estimation task. Furthermore, we define a novel objective function that takes into account errors at different granularity to improve neural network training. To keep the model lightweight and fast, we rely upon MobileNetV2 as backbone.

Tested both on benchmark viewpoint estimation data (Pascal3D+) and on actual vehicle camera data (nuScenes), our method is shown to outperform the state of the art in vehicle viewpoint estimation, in terms of both accuracy and memory footprint.

I. Introduction

Road scene understanding from a moving vehicle is becoming increasingly important in many computer vision fields, like autonomous driving [1], urban scene reconstruction [2], [3] and semantic extraction and classification like lane detection [4], road boundaries detection [5] and crash and near-crash events [6].

Along with traditional environment perception tasks, such as object detection, tracking and segmentation, object viewpoint estimation is a core part of scene understanding [7]. In our case, the focus is on vehicle viewpoint estimation.

Convolutional Neural Networks (CNNs) have been largely used to tackle vehicle viewpoint estimation, but their training requires a huge volume of annotated data that can be collected through the usage of a full suite of sensors like Light Detection and Ranging (LIDAR) or multiple cameras mounted on a moving car [8].

LIDAR is optimal to obtain high quality annotated data and it is the state-of-the-art solution, in terms of precision and accuracy, for autonomous driving systems [9], [10], but it is extremely costly, especially when compared to current dashcam prices. That does not make it suitable for after-market applications, such as fleet management solutions, that require low end-user costs. In this context, monocular dashcams are usually the solution of choice [6]. For this reason, in this paper we present a method for vehicle viewpoint estimation starting from monocular dashcam images.

†Verizon Connect Research, Florence, Italy
*University of Florence, Italy
Email: [1] simone.magistri08@gmail.com
Email: [2] francesco.sambo@verizonconnect.com

Road scene understanding from dashcam images has inherent difficulties such as radial distortion, movement distortion and occlusion. Furthermore, we assume that the cameras are not mounted at exactly the same place in the windshield for every vehicle, leading to non-fixed points of view and lack of extrinsic camera parameters. In light of this, we aim at defining a solution that generalizes to multiple makes and models of vehicles and to *ad-hoc* mounted devices.

Estimating object viewpoint from dashcam images requires first to identify their position within the image and their size. In an application with constrained computational budget, it is common to use single shot object detectors, like YoloV3 [11]. In this paper, we assume to have ground truth level 2D object detection and focus solely on estimating the viewpoint on the road plane, with respect to the camera. We decouple the viewpoint estimation problem from the detection one since training a detector jointly to a viewpoint estimator results in a model that is both less versatile and harder to mantain in a production environment. In addition, recent literature [12] reports that approaches that couple the two tasks do not outperform those that solve them separately for vehicle objects.

This work introduces a lightweight deep learning model that is able to predict vehicle viewpoint from a vehicle image without using any additional information, like extrinsic camera parameters.

Our model is tested on the nuScenes [8] dataset, a large-scale autonomous driving dataset, and on the Pascal3D+ dataset, a benchmark in most of the state-of-the-art papers on object viewpoint estimation.

The main contributions of this work are:

- We developed an accurate, lightweight vehicle viewpoint model which takes as input object detection results provided by an independent detector (such as YoloV3).
- We introduced a new multi-task loss and defined a new model which is able to predict simultaneously finer and coarser viewpoint bins.
- We successfully encapsulated the vehicle type information as input feature, improving the performance on the vehicle viewpoint estimation task.
- We successfully adapted the Siamese approach, developed by [13], in the context of decoupled viewpoint estimation (i.e., the object detector is not jointly trained with the viewpoint estimator).

The paper is organized as follows: in Section II we describe related work on vehicle and general object viewpoint estimation; in Section III, we describe the datasets used;

in Section IV we describe the proposed approach and the models that were developed; finally, in Section V the data pre-processing carried out, the results obtained and the error analysis are provided.

II. Related Work

The viewpoint estimation problem has been addressed as either a classification or a regression approach. The first approach is the most popular since it has been proved more effective by Massa et al. [14], [15].

Focusing on the classification approach, Ghodrati et al. [16] extracted features from the feature maps of a CNN to estimate a discretized object viewpoint. Tulsiani et al. [17] directly estimated the viewpoint from the object images, using a CNN taking into account the object class. Su et al. [18] introduced a discretization into 360 bins for viewpoint prediction and they proposed a geometric structure aware loss function. Divon et al. [13] proposed a CNN based architecture that jointly solved detection, classification and viewpoint estimation, introducing a loss function based on the idea of the Siamese Networks.

III. Dataset

Pascal3D+ [19] is a general object dataset. It consists of 12 object classes. It includes Pascal VOC 2012 [20] and a subset of Imagenet [21] objects, enriched with 3D annotations (azimuth, elevation and tilt).

nuScenes [8] is a large-scale autonomous driving datasets. The dataset is a collection of driving scenes from the vehicle point of view, collected through the usage of a single car equipped with a full suite of sensors. Each vehicle is annotated with its 3D bounding box and its viewpoint.

Unlike Pascal3D+, nuScenes dataset provides representative images for the task of identifying object viewpoint using images from a dashcam. All images in nuScenes are acquired by 6 cameras mounted on the front, front-right, front-left, back right, back-left and back of a moving vehicle. Thus, the elevation and tilt of the objects are almost the same for each image. Pascal3D+ images, on the other hand, are captured by a single fixed camera, located at different positions. As a result, elevation and tilt of the objects may significantly vary across different images.

IV. Proposed Approach

The objective of this research is to develop a framework based on CNNs to estimate vehicle viewpoints given a single road view image. In particular, we only focused on azimuth estimation since we assumed that elevation and tilt of the vehicles with respect to a dashcam do not vary much. The azimuth estimation was addressed as a classification problem rather than a regression one, since it has been proven more effective in most recent literature [14], [15]. For this purpose the azimuth was discretized according to the pattern in Figure 1, with four scale levels ranging from coarser to finer.

Performance on azimuth estimation was evaluated using a common model structure: a CNN backbone followed by a global average pooling layer. We enriched this architecture

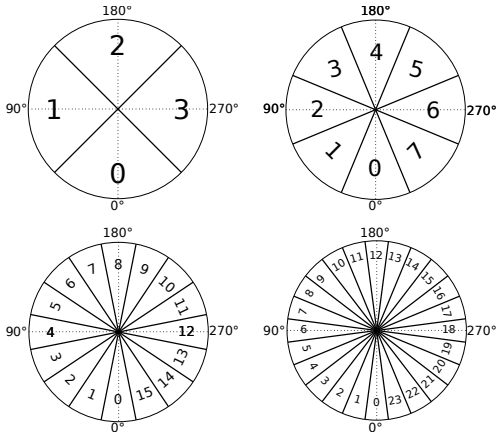

Fig. 1. Azimuth scale levels we adopted to estimate vehicles orientation. The frame of reference assumes slice 0 facing towards the viewpoint.

by adding different output layers and input features. Prior to providing details on the architecture changes, let us define the notation.

Let $\theta \in \mathbb{R}$ be an azimuth angle and let $\mathcal{N}_\alpha : \mathbb{R} \mapsto \{0, .., \alpha - 1\}$ be the discretization function that maps each angle to the corresponding bin, where $\alpha \in \Omega$ and Ω is the set of possible number of bins. Let $y^\alpha \in \mathbb{R}^\alpha$ be the binary indicator vector of the bin $\mathcal{N}_\alpha(\theta)$. Let W be the set of network weights and let $X \in \mathbb{R}^{3\times w\times h}$ be an input RGB image, where w, h are the image width and height respectively, and let $f(W, X)$ be the network output.

A. Single-Task model

We followed an approach similar to [15], predicting directly the angle bins. The network output $f(W, X) \in \mathbb{R}^\alpha$ was thus fixed to size α and its softmax was used to compute the standard cross-entropy function:

$$\mathcal{L}_\alpha(y^\alpha, \hat{y}^\alpha) := -\sum_{i=1}^{\alpha} y_i^\alpha \log(\hat{y}_i^\alpha), \tag{1}$$

where $\hat{y}^\alpha = \text{Softmax}(f(W, X))$.

B. Multi-Task model

We defined a model that could estimate simultaneously different angle discretizations, introducing a mapping between the network output $f(W, X) \in \mathbb{R}^{360}$ and $\mathcal{N}_\alpha$. The mapping was obtained summing the network logits corresponding to each discretization level. Let us define $S_\alpha \in \mathbb{R}^\alpha$ the result of summation of $f(W, X)$ for discretization α.

The resulting loss function was:

$$\mathcal{L} = \sum_{\alpha\in\Omega} \mathcal{L}_\alpha(y^\alpha, \hat{y}^{S_\alpha}) \tag{2}$$

where $\hat{y}^{S_\alpha} = \text{Softmax}(S_\alpha)$

There are multiple benefits of using this approach:

- The overall loss was less penalized when the network made mistakes on fewer tasks.
- It significantly reduced the training time with respect to four single-task models training.

In Figure 2 the overall architecture is shown.

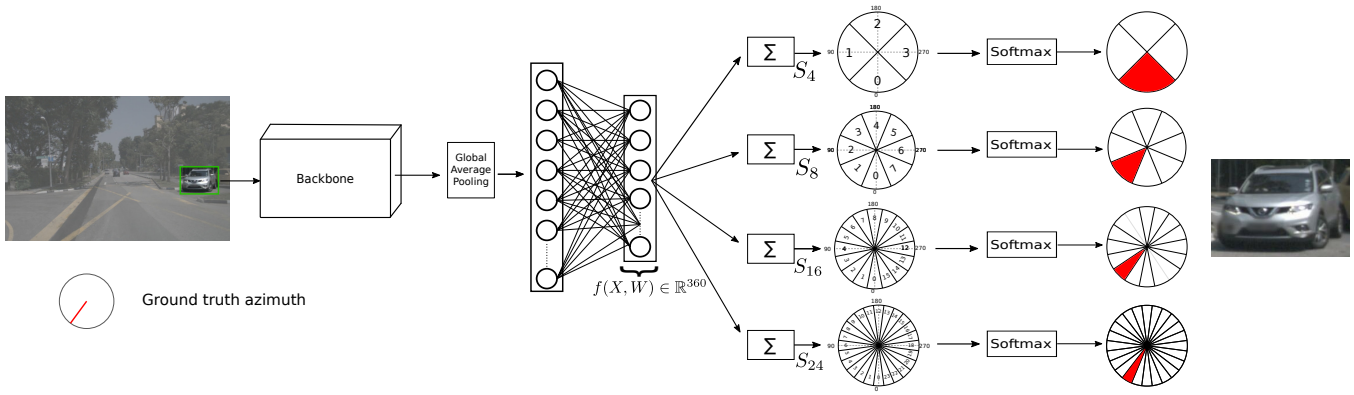

Fig. 2. Multi-task network. Azimuth at different levels of scale is estimated summing the network logits to compute the loss described in Equation 2. The set Ω of the estimated bins is fixed to $\{4, 8, 16, 24\}$.

C. Multi-Task model & Class information

We took into account the vehicle types in two ways:

- Class specific output: it is the approach followed by [18], [17], [15] and consists of defining an output layer characterized by $c \times 360$ units, where c refers to the number of vehicle classes.
  In addition to the previous works, we added the proposed multi-task network output layer for each class c as explained in the previous section IV-B.
- Class as input feature: the class information is encoded in an one-hot-encoded format and it is concatenated to the Global Average Pooling output. In addition, two dense layers of size 512 and 256, respectively, before the output layer are added.

D. Multi-Task model & Siamese network

The Siamese approach was firstly explored by Divon et al. [13] in the context of jointly training an object detector and a viewpoint estimator. We adapted their approach to our architecture.

Let us consider the pair (X, θ), where $X \in \mathbb{R}^{3\times w\times h}$ is the input image and $\theta \in [0, 360)$ is the corresponding azimuth label. If the input image X were to be flipped horizontally, we obtain the image X_{flp} and its expected azimuth is mirrored with regards to the Y axis.

Let us define the operator *flip* : $\mathbb{R}^n \mapsto \mathbb{R}^n$, which maps $y = (y_1, y_2.., y_n)$ to $\textit{flip}(y) = (y_n, y_{n-1}, .., y_1)$.

The forward pass of our architecture was the following:

1) Feed the network with X and X_{flp}.
2) Compute the losses $\mathcal{L}$ and $\mathcal{L}_{flp}$ for the pairs $(f(W, X), \theta)$, $(\textit{flip}(f(W, X_{flp})), \theta)$ respectively, using Equation 2.
3) Evaluate the final loss:

$$\mathcal{L}_s = \mathcal{L} + \mathcal{L}_{flp} + \lambda D(f(W, X), \textit{flip}(f(W, X_{flp}))) \quad (3)$$

where $D : \mathbb{R}^{360} \times \mathbb{R}^{360} \mapsto \mathbb{R}$ is a distance function, $\lambda \in \mathbb{R}$ is a regularization term, $\mathcal{L}$ and $\mathcal{L}_{flp}$ are obtained applying the formula 2 to X and X_{flp} respectively.

We considered as D function the square L2 distance, as proposed in [13], and the angular distance:

$$D(X_1, X_2) = \|X_1 - X_2\|_2^2 \quad (4)$$

$$D(X1, X_2) = \frac{1}{\pi} \arccos \frac{X_1 \cdot X_2}{\|X_1\| \, \|X_2\|} \quad (5)$$

where X_1,$X_2 \in \mathbb{R}^n$

V. Experiments

A. Data pre-processing

In the case of nuScenes, we collected the images from the 6 cameras provided, projecting the 3D bounding boxes to each image plane, following the procedure provided in the development kit of [8]. We extracted crops around the vehicles from the projected 3D boxes and we retained the azimuth. While analyzing the nuScenes dataset, we found images that did not contain enough information for azimuth detection because of their size and form factor. For this reason we filtered the dataset removing vehicles satisfying either one of the following conditions:

$$\tfrac{\textit{width}}{\textit{height}} < 0.4, \quad \textit{width} \times \textit{height} \leq 900\textit{px}^2, \quad \textit{visibility} < 60\%$$

where *width* and *height* are the bounding box dimensions and *visibility* is the total visible size of the vehicle in the six cameras, as provided by nuScenes.

In the case of Pascal3D+, we extracted tight crops around the vehicles using the provided 2D bounding boxes and we retrieved the azimuth angle. Furthermore we manually identified and labeled the trucks contained in Pascal3D+ to preserve consistency with nuScenes vehicle classes.

Finally, we defined one training validation split for nuScenes and one for Pascal3D+ and we merged them to obtain a single training and validation set. As for Pascal3D+ we split 50/50 its official training set stratifying by vehicle type, while for nuScenes the split was obtained choosing different scenes for training and validation set. As for the test set, we maintained separated nuScenes and Pascal3D+ test set. Specifically, for Pascal3D+ we used the Pascal VOC 2012 validation set as test set, as specified in [19], including

TABLE I

SAMPLES DISTRIBUTION OF THE PROPOSED SPLIT ON PASCAL3D+ AND NUSCENES.

Split	Bike (%)	Bus (%)	Car (%)	Motorbike (%)	Truck (%)	Total
Training	2.0	3.3	74.4	2.0	18.3	410k
Validation	2.9	3.7	74.0	2.7	16.7	36k
Pascal3D+ test	18.6	14.7	44.4	17.8	4.5	1.9k
nuScenes test	2.1	3.5	72.8	2.2	19.5	90k

TABLE II

PERFORMANCE COMPARISON BETWEEN MULTI-TASK MODEL AND SINGLE-TASK MODEL USING MOBILENETV2 AS BACKBONE.

Bins	Task	Bike(%)	Bus(%)	Car(%)	Motorbike(%)	Truck(%)	Avg(%)	Total(%)
4	Single	58.81	82.36	90.98	70.85	81.66	76.93	87.75
	Multiple	63.27	83.79	92.35	72.13	82.88	**78.89**	**89.15**
24	Single	27.06	62.24	69.38	35.45	58.82	**50.59**	65.44
	Multiple	27.95	62.24	70.11	35.45	56.92	50.53	**65.62**

the vehicles annotated as *difficult*, *occluded* and *truncated*, while for nuScenes, we used the official validation set as test set. In Table I the percentage of vehicle types in each split and the effective size of datasets are shown.

B. Implementation details

Our code was written exploiting the Pytorch Framework [22].

Optimizer. We used Adam [23] as optimizer with the default parameters and weight decay 10^{-5}.

CNN backbone. We used backbones, with different size and structure, to evaluate the performance on our datasets: MobileNetV2 ([24]), Resnet50 ([25]) and VGG16 ([26]). Pre-trained Imagenet weights were used for all models. We fine-tuned all the layers of MobileNetV2 and Resnet50, whereas for VGG16, we fine-tuned the last two convolutional blocks and the fully connected layers.

Data Preparation. Images were resized to 224×224 pixels and were normalized by subtracting the mean and dividing by the standard deviation of the Imagenet dataset. During the model training, we applied Random Horizontal Flipping with probability 0.5 to each image, except for the training of the model described in Section IV-D.

Metrics. The training of each model was stopped when the accuracy on the validation set stopped to increase. On the test set, we measured the accuracy per class, the average accuracy across classes and the total accuracy.

The majority of the experimental results are reported on the nuScenes dataset, as it is more representative of the task we want to solve (vehicle viewpoint estimation from dashcam images). For completeness, at the end of the session we evaluate our model also on the Pascal3D+ test set.

C. Multi-task model vs Single-task model.

We compared the performance of the multi-task model (IV-B) with the single-task model (IV-A), using MobileNetV2 as backbone.

In Table II the performance on the easiest (4 bins) and hardest (24 bins) tasks are provided. The results show that the multi-task model outperforms the single-task one, in terms of total accuracy for both tasks and of average accuracy for four bins.

TABLE III

MULTI-TASK MODEL PERFORMANCE VARYING THE BACKBONE USED.

Bins	Backbone	Bike(%)	Bus(%)	Car(%)	Motorbike(%)	Truck(%)	Avg(%)	Total(%)
4	Resnet50	66.39	84.27	92.00	71.88	83.29	**79.56**	89.05
	VGG16	57.09	82.29	90.20	65.16	80.34	75.02	86.76
	MobileNetV2	63.27	83.79	92.35	72.13	82.88	78.89	**89.15**
24	Resnet50	24.78	59.73	68.13	32.02	55.24	47.98	63.63
	VGG16	22.44	55.12	64.88	27.77	51.05	44.25	60.14
	MobileNetV2	27.95	62.24	70.11	35.45	56.92	**50.53**	**65.62**

TABLE IV

MULTI-TASK MODEL PERFORMANCE TAKING INTO ACCOUNT THE VEHICLE CLASSES.

Bins	Architecture	Bike(%)	Bus(%)	Car(%)	Motorbike(%)	Truck(%)	Avg(%)	Total(%)
4	No Class	63.27	83.79	92.35	72.13	82.88	78.89	89.15
	Class Specific output	64.57	84.58	92.01	76.84	83.07	80.22	89.10
	Class input feature	65.77	84.84	92.10	75.31	83.73	**80.35**	**89.29**
8	No Class	50.91	79.12	85.14	63.27	73.67	70.42	81.50
	Class Specific output	49.92	79.24	85.10	61.32	74.00	69.92	81.47
	Class input feature	53.14	80.01	85.20	64.24	74.60	**71.44**	**81.82**
16	No Class	36.78	69.77	75.87	43.14	64.28	57.97	71.86
	Class Specific output	34.29	69.61	75.70	46.52	64.57	58.14	71.81
	Class input feature	39.95	69.20	75.64	45.95	65.22	**59.19**	**71.98**
24	No Class	27.95	62.24	70.11	35.45	56.92	50.53	65.62
	Class Specific output	24.78	63.10	70.10	34.73	57.54	50.05	**65.68**
	Class input feature	28.16	63.51	69.73	37.04	58.09	**51.30**	65.66

D. Effect of backbone network

Due to its reduced number of parameters when compared to VGG or ResNet models, MobileNetV2 is one of the fastest CNNs. We evaluated the performance using more complex backbones (Table III).

Our results show that performance drops when more complex models are used, which seems counter intuitive. We conjecture that, for this task, the reduced number of parameters helps preventing overfitting. For this reason, we used MobileNetV2 as a backbone for the subsequent experiments.

E. Multi-task model with class information

We evaluated the effect of adding the vehicle type information to the network structure (Table IV). Adding the class information as input feature outperforms the class specific output approach.

Overall, the class information yields minor improvements on the test set in terms of total accuracy, while it increases considerably the average accuracy across classes. This discrepancy is likely due to the high class imbalance in the dataset (Table I). Specifically, we can observe that the accuracy per class on *bike* and *motorbike* (the minority classes) improves for each discretization level.

F. Multi-task model with siamese approach

We evaluated different λ values for the Equation 3. Since the distance metrics we used have different scale, we evaluated two different search ranges:

- $\lambda \in \{10^k\}, k \in \{-2, -1, 0, 1\}$ for the angular distance.
- $\lambda \in \{10^k\}, k \in \{-4, -3, -2\}$ for the euclidean distance.

The models which provided the best performance on the validation set for each discretization level were selected for

TABLE V

MULTI-TASK MODEL & SIAMESE NETWORK RESULTS.

Bins	Metric	λ	Bike(%)	Bus(%)	Car(%)	Motorbike(%)	Truck(%)	Avg(%)	Total(%)
4	angular	10^{-1}	65.66	86.20	92.26	73.36	83.65	80.23	89.40
8	angular	10^{-1}	54.75	80.61	85.46	62.40	75.24	71.69	82.15
16	L2 norm	10^{-3}	39.95	71.23	76.17	44.52	65.76	59.53	72.52
24	L2 norm	10^{-4}	29.19	63.57	70.11	35.45	58.09	51.28	65.92

TABLE VI

RESULTS SUMMARY NUSCENES.

Bins	Architecture	Avg(%)	Total(%)
4	No Class	78.89	89.15
	Class specific output	80.22	89.10
	Class input feature	**80.35**	89.29
	Siamese	80.23	**89.40**
8	No Class	70.42	81.50
	Class specific output	69.92	81.47
	Class input feature	71.44	81.82
	Siamese	**71.69**	**82.15**
16	No Class	57.97	71.86
	Class specific output	58.14	71.81
	Class input feature	59.19	71.98
	Siamese	**59.53**	**72.52**
24	No Class	50.53	65.62
	Class specific output	50.05	65.68
	Class input feature	**51.30**	65.66
	Siamese	51.28	**65.92**

performance evaluation on the test set. In Table V the results on the test set are provided. It is worth noting that, in our tests, we verified that the training procedure is robust and that one can choose a set of parameters such that the model performs best on average.

G. Summary of nuScenes Results

In Table VI the best results for each discretization level obtained are reported. Overall, the siamese approach provides the best results in terms of the average accuracy per class and the total accuracy.

H. Pascal3D+ results

The official metric of Pascal3D+ is the Average Viewpoint Precision(AVP), which is meant to evaluate jointly the detector and viewpoint model performances. Since we considered the 2D Object Detection Task as solved, we evaluated the models performance on Pascal3D+ test with the same metric used for nuScenes.

To compare our results with the ones in the literature, we used the pre-trained model provided by Su et al. [18] and we run it on the Pascal3D+ test vehicles.

In Table VII, we report the performance on the Pascal3D+ test set (it should be noted that accuracy on class truck is not applicable because the truck label is not present in the original Pascal3D+ dataset). We can observe that the siamese approach which provides better results on nuScenes, provides the best performance on Pascal3D+ in almost every case.

Furthermore, our model outperforms the one from [18] at all discretization levels and in terms of both total accuracy and average accuracy across classes. From a memory

TABLE VII

PASCAL3D+ RESULTS.

Bins	Architecture	Bike(%)	Bus(%)	Car(%)	Motorbike(%)	Truck(%)	Avg(%)	Total(%)
4	Render for CNN ([18])	68.5	81.5	68.4	73.2	N/A	72.9	74.3
	No Class	68.2	81.5	78.1	72.3	74.4	74.9	75.6
	Class specific output	68.5	81.5	79.6	74.9	79.1	76.7	76.9
	Class input feature	70.1	81.9	79.0	72.9	77.9	76.3	76.6
	Siamese	71.6	84.0	80.0	74.9	76.7	**77.5**	**78.0**
8	Render For CNN ([18])	58.6	70.5	58.3	62.5	N/A	62.5	63.2
	No Class	60.6	68.0	69.6	60.8	65.1	64.8	65.9
	Class specific output	61.4	69.8	68.4	61.1	69.8	66.1	66.1
	Class input feature	58.3	73.3	69.7	62.0	67.4	66.2	66.7
	Siamese	60.3	71.2	70.2	66.4	68.6	**67.3**	**67.8**
16	Render For CNN ([18])	39.4	63.3	46.5	45.1	N/A	48.6	49.7
	No Class	38.9	65.8	57.1	41.0	48.8	51.8	51.8
	Class specific output	38.9	68.0	60.2	42.8	52.3	52.4	**53.9**
	Class input feature	37.5	69.0	58.8	44.0	53.5	**52.5**	53.4
	Siamese	42.5	68.0	57.9	42.8	50.0	52.2	53.5
24	Render For CNN ([18])	34.9	55.9	40.8	36.0	N/A	41.9	42.6
	No Class	30.4	54.8	47.3	31.3	39.5	40.7	42.1
	Class specific output	31.6	58.0	47.0	30.7	33.7	40.2	42.3
	Class input feature	29.9	58.0	49.7	31.9	37.2	41.3	43.5
	Siamese	32.7	56.2	49.3	36.3	41.9	**43.3**	**44.6**

footprint point of view, having about $2.7M$ parameters our model is much more lightweight then the model in [18], with $110M$ trainable parameters. That makes the model backward pass faster in training phase and the model itself suitable for edge computing applications.

I. Error Analysis

Pascal3D+ vs nuScenes. We used the average accuracy per class to compare the performance on nuScenes and Pascal3D+. The average accuracy per class allows us to perform a fair comparison between the performance on the two datasets, because it is less affected by the different class balance, compared to the overall test accuracy.

Considering Tables VI and VII, our model performs better on nuScenes than on Pascal3D+. In fact we gain on average over different discretization levels about 5.6% on the average accuracy per class. These results suggest that the dataset context, described in Section III, influences the performance.

Bounding Box Size. Since for many applications it is not of interest to detect objects that are very far, we investigated how the performance changed removing the smaller bounding boxes from the nuScenes test set.

We restricted our analysis to the *car* class, the majority class in nuScenes (Table I). We considered different thresholds, by removing the bottom 20%, 30%, 40%, 50%, 60% of the boxes sorted by area. For each threshold, we evaluated the accuracy on class car at different levels of discretization (Figure 3). In Figure 4, we provide a sample from the nuScenes dataset for visualization.

VI. CONCLUSIONS

In this paper we presented a lightweight deep learning model to predict the viewpoint of vehicles from dashcam images.

In our work, we adapted methods for general object viewpoint estimation to our context. Furthermore we introduced a new training loss, which enforces consistency between angle estimates at different levels of scale. Finally, we introduced a model improving the performance on the vehicle viewpoint estimation task.

Experimental results on the nuScenes dataset show that a small convolutional backbone, like MobileNetV2 [24], is

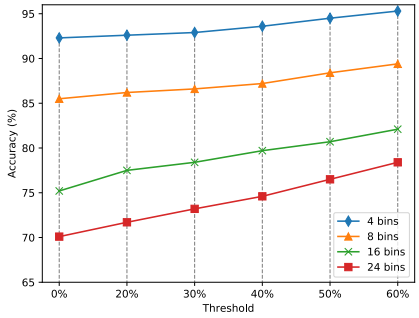

Fig. 3. Accuracy on class *car* removing the bottom 20%, 30%, 40%, 50%, 60% of the boxes sorted by area.

Fig. 4. Bounding boxes of size 61×61 (on the left) and 134×134 (on the right), which represent the 20-th and 60-th percentiles of the areas of the bounding boxes in the dataset.

able to obtain results which are comparable to or even better than the ones obtained by much more complex backbones. Thanks to the speed and the size of the backbone, our model is suitable to be deployed on edge devices.

Furthermore, we showed that adding the object class as network input and training the network with a Siamese approach results in further improvement both in total accuracy as well as in accuracy per class, especially on the minority classes (bike and motorbike)

Comparing the performance between Pascal3D+ and nuScenes, we showed the superiority of our model on nuScenes. It is worth noting that the fixed road point of view of the camera improves vehicle viewpoint estimation.

Finally, our results indicate that model accuracy increases when detected vehicles are closer to the camera. This is particularly useful for applications which do not require the viewpoint estimation of far away vehicles.

Possible directions for future research involve the usage of videos. We strongly believe that video temporal information can noticeably improve the performance in estimating vehicle viewpoint, by imposing strong constraints on the rigid object motion through time.

REFERENCES

[1] J.-R. Xue, J.-W. Fang, and P. Zhang, "A survey of scene understanding by event reasoning in autonomous driving," *International Journal of Automation and Computing*, vol. 15, 04 2018.

[2] Q. Li, H. Lu, X. Liu, X. Huang, C. Song, S. Huang, and J. Huang, "Optimized 3d street scene reconstruction from driving recorder images," *Remote Sensing*, vol. 7, pp. 9091–9121, 07 2015.

[3] N. Cornelis, B. Leibe, K. Cornelis, and L. V. Gool, "3D Urban Scene Modeling Integrating Recognition and Reconstruction," *International Journal of Computer Vision*, vol. 78, pp. 121–141, 2007.

[4] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, "Learning lightweight lane detection cnns by self attention distillation," 08 2019.

[5] T. Suleymanov, P. Amayo, and P. Newman, "Inferring road boundaries through and despite traffic," *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 409–416, 2018.

[6] L. Taccari, F. Sambo, L. Bravi, S. Salti, L. Sarti, M. Simoncini, and A. Lori, "Classification of crash and near-crash events from dashcam videos and telematics," *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2460–2465, 2018.

[7] M. Naseer, S. Khan, and F. Porikli, "Indoor scene understanding in 2.5/3d for autonomous agents: A survey," *IEEE Access*, vol. 7, pp. 1859–1887, 2019.

[8] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[9] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, "Class-balanced grouping and sampling for point cloud 3d object detection," *ArXiv*, vol. abs/1908.09492, 2019.

[10] A. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," 12 2018.

[11] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *ArXiv*, vol. abs/1804.02767, 2018.

[12] D. Oñoro, R. López-Sastre, C. Redondo Cabrera, and P. Gil-Jiménez, "The challenge of simultaneous object detection and pose estimation: A comparative study," *Image and Vision Computing*, vol. 79, 01 2018.

[13] G. Divon and A. Tal, "Viewpoint estimation—insights and model," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 265–281.

[14] F. Massa, M. Aubry, and R. Marlet, "Convolutional neural networks for joint object detection and pose estimation: A comparative study," 12 2014.

[15] F. Massa, R. Marlet, and M. Aubry, "Crafting a multi-task CNN for viewpoint estimation," 09 2016.

[16] A. Ghodrati, M. Pedersoli, and T. Tuytelaars, "Is 2D Information Enough For Viewpoint Estimation?" 01 2014, pp. 19.1–19.12.

[17] S. Tulsiani and J. Malik, "Viewpoints and keypoints," 06 2015, pp. 1510–1519.

[18] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[19] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond PASCAL: A benchmark for 3D object detection in the wild," in *2014 IEEE Winter Conference on Applications of Computer Vision (WACV)*, vol. 00, March 2014, pp. 75–82.

[20] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NeurIPS Autodiff Workshop*, 2017.

[23] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[24] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.