CERBERUS: Simple and Effective All-In-One Automotive Perception Model with Multi Task Learning

Carmelo Scribano[1,2,*], Giorgia Franchini[1], Ignacio Sañudo Olmedo,[3] and Marko Bertogna[1,3]

***Abstract*— Perceiving the surrounding environment is essential for enabling autonomous or assisted driving functionalities. Common tasks in this domain include detecting road users, as well as determining lane boundaries and classifying driving conditions. Over the last few years, a large variety of powerful Deep Learning models have been proposed to address individual tasks of camera-based automotive perception with astonishing performances. However, the limited capabilities of in-vehicle embedded computing platforms cannot cope with the computational effort required to run a heavy model for each individual task. In this work, we present CERBERUS (CEnteR Based End-to-end peRception Using a Single model), a lightweight model that leverages a multitask-learning approach to enable the execution of multiple perception tasks at the cost of a single inference. The code will be made publicly available at `https://github.com/cscribano/CERBERUS`.**

I. INTRODUCTION

Multi Task Learning [1] is a branch of machine learning in which several learning tasks are solved at the same time, while exploiting common and differences between the tasks. Developing lightweight yet powerful **Multi Task** models that can solve several perception tasks in a single forward pass, while maximizing the parameter sharing among the individual tasks, is an enabling capability to bring deep-learning based perception to production vehicles with limited computing resources. The recent introduction of the freely available dataset Berkeley Deep Drive (BDD100K) [2] represents an important resource for the cause of multi-task perception models. Different Multi Task models [3], [4] have been lately proposed, showing encouraging results.

In this short paper we propose a Multi Task model that can simultaneously address the tasks of 1) road object detection (also classifying the object's occlusion), 2) lane estimation and 3) image classification for weather (sunny, rainy, cloudy etc..), driving scene (highway, city streets etc..) and day time (morning, night etc...). Taking inspiration from modern single stage object detectors [5]–[7], we decided to cast both the task of object detection and lane estimation as regression of heatmaps (to encode the likelihood of the presence of objects or lanes at any spatial location) and offsets (used to decode the individual bounding boxes or lane instances).

The advantages of this election are manifolds: 1) Employing the same representation among different tasks make the training process simple, allowing to even optimize for the same objective function. 2) The anchor free approach allow for simple and efficient decoding of both the detection bounding boxes and the road marking lanes. 3) The heatmap representation can be trivially extended to produce instance level prediction in a bottom-up fashion, as we demonstrate by incorporating the occlusion classification task in the object detection head. We design our model's architecture based on simple and well understood patterns, while focusing on a modular approach in order for the final model to be tailored accordingly to the deployment requirements. Leveraging a novel objective function for the heatmaps regression task, our model can be trained end-to-end while simultaneously optimizing for all the objectives. Finally, we tuned the model to reduce to the greatest extent possible the computational footprint, experimenting with efficient backbones [8], [9], and to ensure ease of deployment by refraining from leveraging exotic layers. The experiments presented suggest that the proposed approach represents a strong baseline and an important step forward towards a universal model for end to end perception.

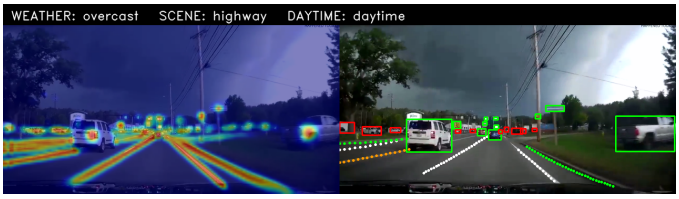

Fig. 1: Qualitative inference results of CERBERUS on publicly available footage. (left) superimposed heatmaps (right) output for object detection and lane estimation.

II. RELATED WORKS

A. Object Detection

Deep learning based object detectors can be classified in Multi-Stage and Single-Stage. The first kind of methods [10]–[13] include a first module to propose a large set of regions of the image that are likely to contain an object, the later stages then classify and refine the predictions. These methods have being known for being highly accurate, but computationally expensive. In opposition, single-stage detectors do not rely on a region proposal mechanism. In this sense, anchor-base detectors [14]–[16] output a set of *candidate* boxes at predefined locations in the form of offset with respect to a predefined set of anchor boxes, requiring an expensive process of non-maxima suppression to obtain the final detections. On the other hand, more recent anchor-

[1]Carmelo Scribano, Giorgia Franchini and Marko Bertogna are part of the Department of Physics, Informatics and Mathematics of the University of Modena and ReggioEmilia, 41125, Modena, Italy. {name}.{surname}@unimore.it

[2] Carmelo Scribano is part of the Department of Mathematical, Physical and Computer Sciences of the University of Parma, 43124, Parma, Italy.

[3]Ignacio Sañudo Olmedo and Marko Bertogna are part of HIPERT s.r.l, 41122, Modena, Italy. {name}{surname}@hipert.it

*Corresponding Author

free detectors [5]–[7], [17]–[19] output predictions only at object locations and thus require little post-processing. Those methods are often based on detection of keypoints, centerpoints or even on transformer decoders.

B. Lane Estimation

Lane estimation methods are divided in segmentation-based and detection-based: the first category [20], [21] cast the lane estimation as a pixel-wise classification problem (either the pixel belongs to a lane or not), with the addition of a mechanism to associate the lane masks to separate marking instances. Detection-based methods instead have a lot in common with object detectors, as such some methods regress the lane instance using an anchor mechanism [22], [23]. Finally, recent works cast the lane estimation as a regression of keypoints and embeddings [24] or offset [25] to aggregate the keypoints in distinct lanes.

C. Multi-Task Approaches

To the best of our knowledge, the closest works to our are the very recent YoloP [3] and HybridNets [4]. In our understanding, a main limitation of such models is the author's choice of relying on *heterogeneous* representations for the individual tasks (e.g, combining a Yolo [15] style object detection head with a segmentation head for the lane estimation). Such a choice makes the model inherently convoluted and the optimization phase harder because it relies on completely different objective functions.

III. Method

A. Problem Formulation

As mentioned in the introduction, the proposed model is able to address multiple distinct tasks required for vision-based perception in a single inference step. In particular, given as input a single RGB image $I \in \mathbb{R}^{(w \times h \times 3)}$ we address the tasks of:

a) Road Object Detection: predicting a bounding box and an associated class label for each object of among 10 distinct object classes. Additionally, for each object, we provide a binary label which indicates whether the object is fully visible or partially occluded.
b) Lane Markings Estimation: regressing a polynomial curve and an associated class label for each lane marking visible.
c) Image Tagging: providing three distinct multi-class classification labels associated to the whole frame, weather (7 classes), scene (7 classes) and time of day (4 classes).

We train and evaluate our model on BDD100K. This publicly available dataset consists of 70K train images and 10K validation images sampled at 10Hz from a large set of 100K driving videos (around 40s each). Videos are captured in a wide range of scene types and conditions, allowing to train robust models able to generalize to real driving conditions.

B. Object detection

The desired output for the object detection task is a set of N_O detections in the form $(x_1, y_1, x_2, y_2, c, o)$ where x_1, y_1 (resp. x_2, y_2) define the image space coordinate of the top-left (resp. bottom-right) corner of the bounding box, c characterizes the object class and o is the binary label for the object's occlusion state. Our approach is heavily based on the anchor-free detector described in [5] which is based on keypoint estimation.

For each object class $k \in C_O$ with $|C_O| = 10$, we output a heatmap $\overline{H}_D^k \in \left(\frac{w}{S} \times \frac{h}{S}\right)$ (with S being output stride) that encodes the center points of all the objects in I belonging to class k. Given a set of detection ground truths, first the target keypoints in output space $c^i = \left(\frac{x_2^i - x_1^i}{2S}, \frac{y_2^i - y_1^i}{2S}\right)$ for $i \in \{0, \ldots, N_O^k\}$ are computed as the geometrical center point of each box and rescaled with the output stride (rounding to the nearest integer), as in fig. 2 (left). Then the target heatmap H_D^k for the k-th object class is obtained by computing for each keypoint a Gaussian centered at the keypoint's location and taking the element-wise maximum:

$$H_D^k(x,y) = \max_j \left\{ \exp\left(-\frac{(x - c_x^j)^2 + (y - c_y^j)^2}{\sigma^2} \right) \right\} \quad \text{for } j \in \{0, \ldots, N_O^k\} \tag{1}$$

with σ being a hyperparameter.

In addition, we regress a double offset map (shared across all the object classes) $\overline{O}_D \in (w/S \times h/S \times 4)$. The values at the coordinates of each object center $c = (c_x, c_y)$ correspond to the box-corners offset vectors from the object center to the top-left and bottom-right corners of the target bounding box, as in fig. 2 (right).

$$O_D(c_x, c_y) = (c_x - x_1/S, c_y - y_1/S, c_x - x_2/S, c_y - y_2/S)$$

We also regress the occlusion state for each detected object: this is simply achieved by predicting an occlusion map $V_D = \left(\frac{w}{S} \times \frac{h}{S} \times 1\right)$ where the value at a center point's coordinates (c_x, c_y) is meant to be 0 if the object is fully visible and 1 otherwise.

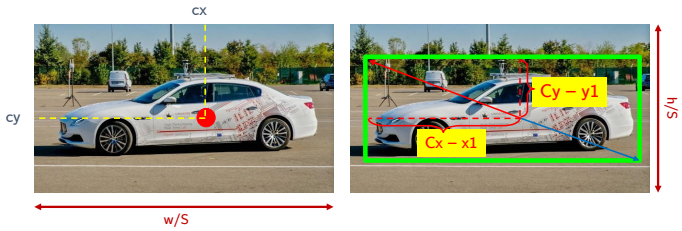

Fig. 2: Definition of the object detection's targets: (left) object center-point (right) box-corners offsets: top-left (red) and bottom-right (blue).

At inference time the predicted object detections for each class k are retrieved by first taking the local maxima (the peaks of the Gaussians) from the keypoint heatmaps $\overline{P}^k = [(\bar{c}_x^1, \bar{c}_y^1)^k, ..., (\bar{c}_x^{N_O^k}, \bar{c}_y^{N_O^k})^k]$, then the bounding boxes are reconstructed by summing the predicted box-corners offset vector for each center keypoint taken from $\overline{O}_D$ at the center point's coordinates. The occlusion classification is retrieved from $\overline{V}_D$ in the same way.

C. Lane Estimation

The output of the lane estimation task is a set of lane instances for each lane class l among the $N_l = 8$ possible ones. The generic i-th lane instance $L_i^l = [(x_1, y_1), ..., (x_{n_i^l}, y_{n_i^l})]$ is a list of keypoint coordinates of arbitrary length n_i^l (different for each lane instance). All the keypoints in L_i^l should belong to a single lane and can be easily used to fit a polynomial curve representing the lane line boundaries in image space coordinates. As for the object detection we employ a keypoint-based approach, in particular our method is derived from [25] and [24]. As shown in fig. 3 this method allow to simultaneously recover all the keypoints belonging to any lane visible for the l-th lane class and regressing an offset vector for each keypoint that is used to effectively cluster the keypoints in distinct lane instances.

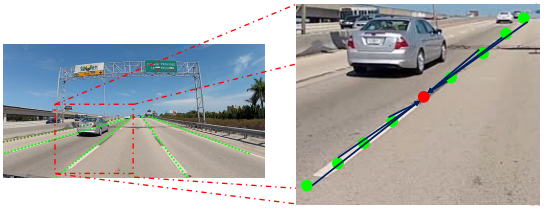

Fig. 3: Representation of the lane estimation head used: each keypoint belonging to a line cast a vote offset for the mid-point of the lane that it belongs to.

Like for the object detection we produce a heatmap H_L^l for each lane class l, the target heatmap is obtained by first rescaling the keypoints coordinates to the output space and then fitting a Gaussian at each lane keypoint as from Eq. 1, disregarding of the lane instance. Additionally, we produce a single offset map $O_L \in (\frac{w}{S} \times \frac{h}{S} \times 2)$ where at each j-th keypoint location $p_j^i = (x_j^i, y_j^i)$ of the i-th lane instance we regress an offset vector to the mid-point $L_i[|L_i|/2] = (xm_i, ym_i)$ of the lane instance (eq. (2)).

$$O_L(x_j^i, y_j^i) = (xm_i - x_j^i, ym_i - y_j^i) \tag{2}$$

At inference time the candidate lane keypoints $[(x_1, y_1), ..., (x_{N_l}, y_{N_l})]$ are first retrieved, for each lane class l, by selecting the Gaussian peaks from the predicted heatmap $\overline{H}_L^l$. Then for each candidate keypoint the corresponding lane center point $(\overline{xm}_i, \overline{ym}_i)_l$ is computed. Finally, the predicted lane midpoints are used to cluster the keypoints in individual lane instances using an agglomerative clustering algorithm [26] with a fixed distance threshold.

D. Image Tagging

The image tagging task simply consists of three multi-class classification problems: the model is tasked to produce three labels (C_w, C_s, C_t) corresponding to the frame-level label for weather, scene, and time of day respectively.

IV. Model Architecture

Our model, depicted in fig. 4, follows a simple and robust approach common to several single-stage detector: first the convolutional part of an off-the-shelf classification model is used to extract a set of n_F multiscale feature maps:

$$\mathcal{F}_i \in (C_i, w/S_i, h/S_i) \text{ for } i \in \{0, \dots, n_F\}$$

With C_i denoting the number of channels (increasing) and S_i denoting the output stride (also increasing). For all of our models, we always select $n_F = 4$ output feature maps at fixed output strides $(4, 8, 16, 32)$, while the number of output channels C_i is specific to the chosen backbone.
Subsequently, a *neck* block further process the feature maps, increasing the output resolution and (optionally) aggregating the multiscale features. The neck provide as output a small resolution feature map ($S = 32$), to be used exclusively for the image tagging task and a high resolution feature map ($S = 4$) to be used for both the object and lane detection tasks. Finally, a distinct *head* for each individual task apply a final convolutional layer in order to produce the task-specific output.

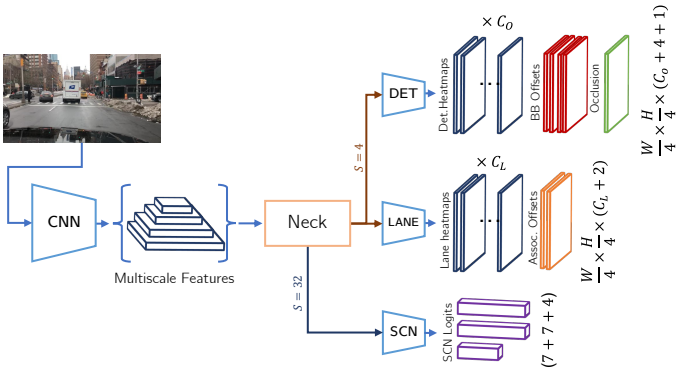

Fig. 4: Overview of the proposed model. The outputs are produced by three distinct heads for object detection (DET), lane estimation (LANE) and scene classification (SCN).

A. Backbone

In order to evaluate the flexibility of our approach we experimented with three different model families as convolutional backbones: 1) Resnet [27], with its simple and well understood architecture, represents the most conservative option 2) Mobilenet [9] instead represent a well established example of convolutional model with reduced memory and computational footprint 3) finally EfficientNet [8] is a more recent example of a model with low inference cost, leveraging modern solutions.
We purposely avoided to tinker with backbones either too computationally intensive to be effective in our intended deployment target or built with uncommon layers and architectural choices [28] that would make the conversion to the inference framework harder and potentially inaccurate. All the tested backbones are pretrained on the ImageNet classification dataset [29].

B. Necks

To process the feature maps obtained by the convolutional backbone, we experiment with two different neck blocks.

The **simple** neck only takes the last feature map (stride 32) and gradually increase the spatial resolution up to a stride of 4, required for object and lane detection, with alternating layers of regular 2D Convolution and Transposed convolution with stride of 2 and kernel size of 4 (implying a 2-time upsampling factor for each layer). The featuremap returned for the image tagging task is the output from the first convolutional layer. Furthermore, all the Transposed convolutional layers are initialized as bilinear interpolation kernels.

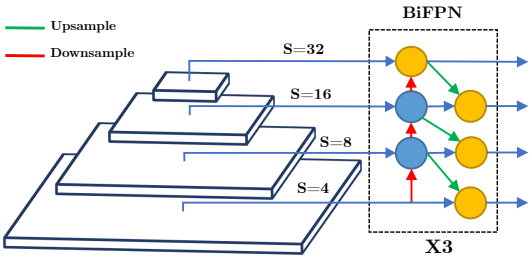

Fig. 5: Overview of the BiFPN based head. In all the experiments, all the up sample operations are implemented as nearest neighbor interpolation and the down sample operations as MaxPooling layers.

The more sophisticated, yet lighter, neck implementation is based on the Bi-directional Feature Pyramid Network (**BiFPN**) introduced in [30]. This block is an evolution of the Feature Pyramid Layer (Fpn, [31]), it allows fusing feature maps ad different scales leveraging both top-down (up sampling) and a bottop-up (down sampling) paths to allow the information to flow in both directions, as depicted in fig. 5. Given a set of multiscale feature maps $\mathcal{F}_S$ with S denoting the stride, $(\mathcal{F}_4, \mathcal{F}_8, \mathcal{F}_{16}, \mathcal{F}_{32})$ the intermediate output for the top-down path is:

$$\mathcal{T}_S = \begin{cases} \mathcal{F}_S & \text{if } S = \min_S \\ Conv_{2D}(\mathcal{F}_S + up(\mathcal{F}_{S/2}) * w_{\mathcal{T}}^S) & \text{otherwise} \end{cases}$$

Then the bottop-up path is defined as follows:

$$\mathcal{B}_S = \begin{cases} \mathcal{T}_S & \text{if } S = \max_S \\ Conv_{2D}(\mathcal{T}_S + down(\mathcal{T}_{2S}) * w_{\mathcal{B}}^S) & \text{otherwise} \end{cases}$$

The trained weights $w_{\{\mathcal{B}|\mathcal{T}\}}^S$ are added to the feature fusion to allow the network to model the importance of each feature. We take $\mathcal{B}_4$ as input for the object and lane detection tasks, and $\mathcal{B}_{32}$ for the task of image tagging, respectively.

C. Heads

The heads follow a minimalistic design aimed at keeping the complexity overhead of each individual task to a minimum. The head for the lane estimation and object detection head share also the same input features and an identical structure: each of the desired outputs (i.e, heatmaps, offsets etc...) is obtained by applying a dedicated 3×3 convolutional layer to the feature map, followed by a ReLU activation, followed by a final 1×1 convolution to produce the desired number of output channels. A sigmoid activation is applied only to the heatmap's output layer, for training stability we found important to initialize the bias value of the last convolution to 4.6 ($\sigma(4.6) = 0.99$).
The image tagging head is as simple as it gets, a single 3×3 convolution is applied to the smallest feature map produced by the neck block, followed by activation and max-pooling, finally three fully-connected layer are applied to obtain the image level classification outputs.

V. EVALUATION

A. Experimental Setup

The annotation provided by BDD100K for the object detection task already comes in the desired format, where each bounding box is represented in image space as the coordinates of top-left and bottom-right corners plus an object label class. Hence the heatmap and offset targets can be easily computed as detailed in section III-B, with the value of σ computed as in [7]. Other competitors ([3], [4]) select only specific classes and squash them in a single "vehicle" super class. Instead, we preserve the class as provided by the dataset. For the lane estimation task the provided lane markings are annotated by labeling the two edges of each lane line as two distinct lines, while for our method we would rather predict a single line in the middle of each road marking. We are first required to associate and merge the two road marking annotation in a single one, then we can obtain a keypoint array for each lane instance by sampling the Bézier curves with a fixed sampling pace, finally the targets can be computed as in section III-C, with σ fixed to 2

To keep the computational demands low, the RGB images, with an original resolution of 1280×720 pixels, are first downscaled with a factor of 2 and then cropped to a final resolution of 640×320. The output stride $S = 4$ for all the experiments. During training several data augmentation strategies are adopted in order to improve generalization against changes in appearance and illumination (random noise, RGB shift, random brightness and contrast...) and camera positioning (random affine transforms, random crop, random horizontal flips).

B. Objective Functions

Multitask deep learning models are often non-trivial to train due to instability and some tasks dominating the gradient of the loss. Hence, careful weighting of the individual tasks and often elaborate multi-stage training strategies are commonly required. Instead, our model can be trained end-to-end mostly relying on basic objective function, except for the loss function used for the heatmaps regression. In total, we use three different loss functions:

1) Heatmap Regression Loss: It is the same for the two tasks, commonly used objective function for heatmaps regression include the standard $L1$ and MSE ($L2$) distance functions, other loss function explicitly engineered to boost heatmap regression include the Wing Loss [32], [33]. Lately the Focal Loss introduced in [16] has been found to be useful

Model	Backbone	Neck	Object Detection		Lane Estimation (IoU)	Scene Classification (F1)		
			mAP50 (Boxes)	Accuracy (Occl)		Weather	Scene	Time of Day
RN34-sim	Resnet34	Simple	0.210	0.913	0.640	0.815	0.790	0.932
RN34-bifpn	Resnet34	BiFPN	0.265	0.910	0.644	0.817	0.791	0.932
RN50-bifpn	Resnet50	BiFPN	0.274	0.911	0.648	0.822	0.794	0.935
RN101-bifpn	Resnet101	BiFPN	0.281	0.912	0.65	0.825	0.794	0.934
EnB2-bifpn	Efficientnet b2	BiFPN	0.272	0.907	0.647	0.821	0.790	0.937
Mobbv2-bifpn	MobilenetV2	BiFPN	0.230	0.9	0.633	0.813	0.790	0.934

TABLE I: Results of various configurations for all the tasks evaluated on BDD100K

for this matter [5]–[7], the focal loss add a weighting term to the Cross-Entropy Loss (usually used in binary classification problems) to avoid the contribution from the large amount easy to classify values (i.e, background pixels) to dominate the gradient magnitude. In our setup, we experience severe instability issue with the Focal Loss (as well as with Generalized Focal Loss [34]). Driven by a similar principle we instead propose to use a weighted version of $L2$ distance reported in eq. (3)

$$\mathcal{L}_H(H, \overline{H}) = \frac{1}{N_K} \sum max\{(1+H)^{\alpha}, (1+\overline{H})^{\beta}\}(H-\overline{H})^2 \quad (3)$$

Recalling that both H and $\overline{H}$ are tensors of shape $(w/S \times h/S)$, the elementwise weighting factor will give an exponentially larger relevance to locations where either the target heatmap or the predicted heatmap are closer to the maximum value of 1. In this way the vast heatmap background (filled with zeros) has a low relevance, the target weight with exponential α allow focusing on hard values where a peak is present, the prediction weighting term with exponential β is instead responsive of penalizing false positives. For all our experiments, we fix $\alpha = 4$ and $\beta = 2$.

2) Offsets Regression Loss: The objective function for the offsets required by the formulation of both tasks is computed only at the corresponding keypoint location (kx, ky) as a simple $L1$ distance. Other works suggest that a more sophisticated function as IoU [35] or DIoU loss [36] could boost the performance of the object detection task, we did not further investigate these alternatives. While practically those objective functions could be leveraged in our formulation, we opted for the simplest loss function to avoid incurring in hard to track down convergence issues.

3) Classification Loss: Finally, we use the standard Cross-Entropy Loss function both for the image tagging task and for the (binary) object level occlusion classification. The predicted object occlusion logit is readout at the ground-truth boxes center points values from the occlusion map, $\overline{O}_D$ and the loss value is then computed for each object and averaged out.

C. Experiments and Evaluation

All the experiments reported in table I are run with the same set of hyperparameters: in particular we train for 75 epochs with a batch size of 64 (for a total of over $160K$ steps) using the optimizer Adam, the learning rate is linearly increased from zero to 2.5^{-4} during 3500 steps, then is halved after $100K$ steps. For comparison in table I we report the results of six different architectures with different backbones: Resnet (34, 50 and 101), MobilenetV2 and Efficientnet-b2. For the neck (section IV-B) we report a single experiment using the simple version, due to the obvious superiority of the BiFPN Neck.

Model	Params (M)	MACCs (G)	FP16 Infer (ms)	
			Xavier	Nano
RN34-sim	29.1	61.67	14.34	205.64
RN34-bifpn	22.2	19.77	8.85	103.98
RN50-bifpn	24.6	21,8	11.09	130.06
EnB2-bifpn	8,7	7.45	14.11	135.30
Mobbv2-bifpn	2.7	5.9	5.86	68.37

TABLE II: Inference results: inference times reported account only for the forward pass computation, without accounting for the decoding phase.

1) Performance evalutation: We evaluate the performance for the tasks of Object detection using the popular mAP@0.5 metric [37], considering all the classes of the dataset, we extract from the predictions all the bounding boxes whose confidence score (value at the keypoint location) is greater or equal to 0.25. The object occlusion sub-task is evaluated by first matching the ground truth bounding boxes to predicted ones, solving a minimum weight matching problem, then Accuracy metric is computed only for the successfully matched boxes.
To evaluate the lane estimation, we generate binary masks for both the true and the estimated lane instances (disregarding the class label). This choice introduces in the final error the noise added by the decoding and polynomial fitting process. In a future iteration we will likely be focusing on a different metric less dependent on the post-processing stage. Finally, for the image tagging task we individually evaluate the three multi-class classification tasks computing the $F1$ score across all classes.

2) Profiling: To evaluate the efficiency of our model we report in table II measures of number of parameters and the total number of multiply-accumulate operations. More important, we collect inference times on real deployment scenarios by running the models of two popular embedded platforms of the Nvidia Tegra family: the AGX Xavier represents the high-end segment, being capable of up to 32 TOPs with a power consumption of up to 20W. Meanwhile, the Jetson Nano represents the lowest end of the spectrum, with only 0.5 TOPs and 5W of power consumption. For both devices we replicate the same setup by running the

last version (4.6.1 at the time of writing of the Jetpack SDK and executing the models in 16-bits floating point precision using the proprietary inference framework TensorRT (version 8.2.1).

VI. CONCLUSIONS

Due to the preliminary nature of this short paper we cannot draw firm conclusions. It is utterly clear that more experiments and in particular detailed comparison with other methods will be required. However, the experimental results are enough to conclude that our methodology achieve the proposed goal of addressing all the considered perception tasks at a low inference cost, while exhibiting satisfactory performance and being easily and effectively deployable on embedded hardware.

ACKNOWLEDGMENT

This work has been partially supported by the project "CEMP" funded by Region Lombardia, Italy (POR-FESR 2014-2020 - Call HUB Research and Innovation) and by the INdAM research group GNCS.

REFERENCES

[1] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, "Multi-task learning for dense prediction tasks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[2] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2636–2645.

[3] D. Wu, M. Liao, W. Zhang, and X. Wang, "Yolop: You only look once for panoptic driving perception," *arXiv preprint arXiv:2108.11250*, 2021.

[4] D. Vu, B. Ngo, and H. Phan, "Hybridnets: End-to-end perception network," *arXiv preprint arXiv:2203.09035*, 2022.

[5] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[6] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6569–6578.

[7] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 734–750.

[8] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[9] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[11] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[13] Z. Cai and N. Vasconcelos, "Cascade r-cnn: high quality object detection and instance segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 5, pp. 1483–1498, 2019.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[15] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[17] X. Zhou, J. Zhuo, and P. Krähenbühl, "Bottom-up object detection by grouping extreme and center points," in *CVPR*, 2019.

[18] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9627–9636.

[19] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[20] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: an instance segmentation approach," in *2018 IEEE intelligent vehicles symposium (IV)*. IEEE, 2018, pp. 286–291.

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

[22] Z. Chen, Q. Liu, and C. Lian, "Pointlanenet: Efficient end-to-end cnns for accurate real-time lane detection," in *2019 IEEE intelligent vehicles symposium (IV)*. IEEE, 2019, pp. 2563–2568.

[23] H. Xu, S. Wang, X. Cai, W. Zhang, X. Liang, and Z. Li, "Curvelanenas: Unifying lane-sensitive architecture search and adaptive point blending," in *European Conference on Computer Vision*. Springer, 2020, pp. 689–704.

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

[25] J. Wang, Y. Ma, S. Huang, T. Hui, F. Wang, C. Qian, and T. Zhang, "A keypoint-based global association network for lane detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1392–1401.

[26] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236–244, 1963.

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

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

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[30] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[31] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[32] Z.-H. Feng, J. Kittler, M. Awais, P. Huber, and X.-J. Wu, "Wing loss for robust facial landmark localisation with convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2235–2245.

[33] Z.-H. Feng, J. Kittler, M. Awais, and X.-J. Wu, "Rectified wing loss for efficient and robust facial landmark localisation with convolutional neural networks," *International Journal of Computer Vision*, vol. 128, no. 8, pp. 2126–2145, 2020.

[34] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 002–21 012, 2020.

[35] D. Zhou, J. Fang, X. Song, C. Guan, J. Yin, Y. Dai, and R. Yang, "Iou loss for 2d/3d object detection," in *2019 International Conference on 3D Vision (3DV)*. IEEE, 2019, pp. 85–94.

[36] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 12 993–13 000.

[37] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.