Reference Detection for Off-road Self-Driving Vehicles Using Deep Learning

Marcelo Eduardo Pederiva

School of Electrical and
Computer Engineering
University of Campinas, Brazil
Email: marcelopederiva@gmail.com

Ely Carneiro de Paiva

School of Mechanical Engineering
University of Campinas, Brazil
Email: elypaiva@fem.unicamp.br

***Abstract*—This paper proposes the application of deep neural network models to detect references in off-road driving for autonomous vehicles. Due to the absence of traffic signs in non-urban areas, the work searched for a low-cost sensory-based solution for autonomous localization in this environment. Given the advancement of Machine Learning techniques, we used Object Detection algorithms to solve the localization problem. For this reason, we trained three existing object detection models (Fast YOLOv2, SSD300 and Faster R-CNN) to detect a reference at the road boundary. The project analyzed these three architectures performance after training with a small dataset (around 300 images), regarding the detection distance, the number of detection and image processing time. Through two experiments, one in the same environment as the training step and another with a different background, we evaluate the pros and cons of each model and the possible application scenario for each one in autonomous cars.**

Keywords–YOLO; Faster RCNN; SSD; Object Detection; Autonomous Vehicles.

I. INTRODUCTION

In the last years, autonomous driving in signposted roads is a research field that has received increasing interest, both from academy and industry. Different solutions and strategies have been proposed to provide the right information and actions to make a robot vehicle drive autonomously. The main objective is to use different sensors and algorithms to map and identify traffic signs, traffic lights, obstacles, pedestrians and cars on the street, in real-time. However, off-road environments still present challenges that need attention. In the absence of lane lines and traffic signs, uneven terrain and the presence of animals, off-road environments require careful driving with different approaches to extract the information of the road and make the right decisions.

Nowadays, the use of Deep Learning in autonomous vehicles is one of the most common solutions, becoming the state-of-the-art approach for a host of problems in perception area, such as image classification and semantic segmentation [1]. Based on how humans accelerate, brake, identify the signs and the limits of the road, the machine can learn and respond in the same fashion. One important research line focuses on the different ways to train the machine on how to detect lane lines [2] or the road itself [3][4]. Another investigation field related to the off-road driving used Semantic Segmentation to identify tracks in the middle of a forest [5]–[7]. The Semantic Segmentation method presents a perfect choice to train a robot how to drive on an off-road track because this method does not use any kind of line or reference on the road. The technique uses a mask as a reference to train the machine for identifying each place or object in the scene [8]. Nowadays, there are some techniques that can achieve more than 70% accuracy processing more than 70 frames per second [9][10]. However, it requires a powerful Graphics Processing Unit (GPU) to train the machine. Consequently, this is one of the main motivations for the work presented in this paper, where we use a different approach for reference detection.

Learning-based Object Detection is a fast method to train a robot to detect specific objects (car, dog, cat, person, etc.). Many models in the literature, with different architectures, compete to be the fastest and most accurate method [11]. The evolution, in the last decade, regarding fast processors and efficient object detection algorithms allows for the use of these models to train and identify specific objects, increasing their application to autonomous systems.

The concept of detecting patterns through images has been studied for decades and used in detecting faces, people and simple objects [12]. Nowadays, with the advantage of fast processing CPU's (Central Processing Unit) and GPU's, the complexity of artificial intelligence allows us to train a machine to detect and classify any pattern with a predetermined data set. Furthermore, new techniques, such as transfer learning, allow the use of a small amount of data to achieve good results [13].

The detector algorithms are usually composed of two parts, a backbone, that aims to identify the main characteristics of the image and, the head, which uses the backbone information to predict classes and bounding boxes of objects. The backbone component is usually represented by the VGG model [14], ResNet [15], MobileNet [16][17] or DarkNet [18]. On the other hand, the head component uses different approaches and, based on that, the detector algorithm can be categorized as a two-stage detector or an one stage detector.

The two-stage detectors are composed of two parts: the first part uses the input image to propose a set of regions of interest, with select search or Region Proposal Network (RPN) and the second part performs the classification of the candidate regions. The models that use this approach are the Regions with Convolutional Neural Network (R-CNN) features [19], Fast R-CNN [20] and Faster R-CNN [21].

On the other hand, one-stage detectors do not use the region proposal step, but rather go straight to the detection of a limited number of predefined bounding boxes. This method makes the

 ISBN: 978-1-61208-787-0

model processing faster, however, it decreases the accuracy. The most known models that use this one-stage detector are You Only Look Once (YOLO) [18][22][23], Single Shot Detector (SSD) [24] and RetinaNet [25].

This paper is organized as follows. Section II presents the background related to the object detection models that were used in this work. Section III proposes the training method and two experiments to evaluate the model's performance. Finally, Section IV presents the conclusion about the experiments and discusses improvements to the project in the future.

II. BACKGROUND

In this section, we will present the main Learning-based Detecion algorithms used for image recognizing applications.

A. *Faster R-CNN*

In 2014, Ross Girshick proposed a simple and scalable detection algorithm, an approach that combines the high-capacity of convolutional neural networks with proposed regions to localize and segment objects [19]. The model, called Regions with CNNs features (R-CNN), receives the input image and extracts around 2000 region proposals. Each region is warped to a fixed-size and computed by a large CNN, where it is classified by label probabilities (Figure 1).

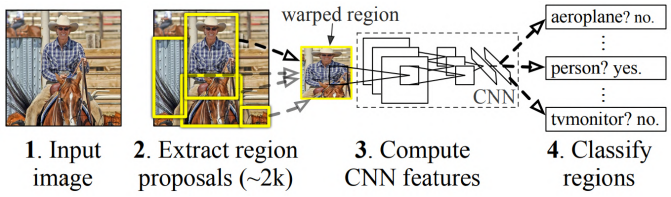


Figure 1. R-CNN architecture [19].

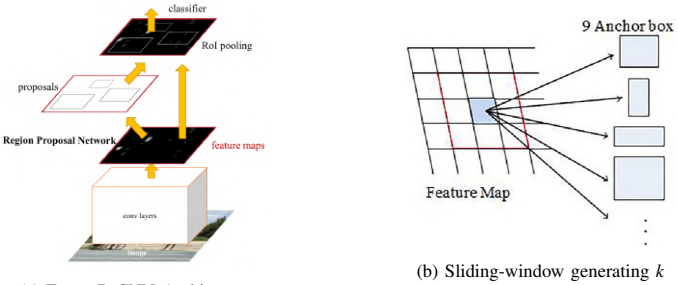
However, the R-CNN method presents some problems in real-time implementation. It needs a huge amount of time to train the network by classifying 2000 region proposals per image. Thus, in 2015, Girshick presented Fast R-CNN, a new model evolved from the previous one and intended to be faster and more accurate [20].

The approach of Fast R-CNN is similar to the R-CNN algorithm, however, instead of feeding the region proposals to the CNN, it sends the image into the CNN to generate a convolutional feature map. From this map, the regions proposals are identified and warped into bounding boxes. Using a Region of Interest (RoI) pooling layer, the regions are reshaped into a fixed size to be fed into a sequence of fully connected layers, each one with two outputs. The first output is a softmax classification layer, where it decides which object class was found in the prediction. The second output is the Bounding Box Regressor (BBox Regressor), a popular technique to refine or predict localization boxes in recent object detection approaches. This technique approximates the nearby bounding boxes to the region proposals (or *Anchors*). In other words, the BBox Regressor outputs the bounding box coordinates for each object class [20].

Both algorithms (R-CNN and Fast R-CNN) use Selective Search. This involves sliding a window over the image to generate region proposals where objects could possibly be found [26]. However, this method is a slow and time-consuming process that affects the performance of the network. For this reason, in 2017 Shaoqing Ren et al. proposed a different object detection design, called Faster R-CNN, that eliminates the selective search algorithm and makes a network learn the region proposals [21].

The Faster R-CNN head part is composed of two modules. First, there is a deep fully convolutional neural network that proposes regions, the RPN and, second, a network that uses these proposals of RPN to detect objects (Figure 2a). The second module works with the same detector used in Fast R-CNN.

The RPN takes an image of any size and, with a Convolutional Neural Network (VGG-16 was used in [21]), it proposes a set of region boxes (Anchors) and gives the probability of those region boxes being an object class or a background. To generate these regions, an $n \times n$ window slides over the feature map. Each sliding-window predicts multiple region proposals (*Anchors*), where the maximum number of possible regions is denoted by k ($k = 9$ [21]) (Figure 2b).



(a) Faster R-CNN Architecture [21].

(b) Sliding-window generating k ($k = 9$) *Anchors*.

Figure 2. Faster R-CNN.

As a final step, the model unifies the RPN with the Fast R-CNN detector. The algorithm applies a RoI to reduce all *Anchors* to the same size and, for each region proposal, the model flattens the input, passing it through two fully-connected layers with Rectified Linear Unit (ReLU) activation. Finally, these fully-connected layers generate the prediction of the class and the box of each object.

B. *YOLO*

The YOLO architecture, differently from Faster R-CNN, has no RPN. It uses a single feed-forward convolutional network to predict classes and bounding boxes.

The YOLO algorithm divides the input image into a $S \times S$ grid, where each grid cell is responsible for detecting the object in its area. Each one predicts B bounding boxes and it scores the confidence to be an object or not. The confidence score reflects the probability of the predicted box to contain an object $P_r(Object)$, as well as how accurate is the predicted box by evaluating its Intersection over Union value (IoU^{truth}_{pred}). In this sense, the confidence score becomes:

$$\text{Confidence Score} = P_r(Obj) * IoU^{truth}_{pred} \tag{1}$$

$$P_r = \begin{cases} 1 & \text{If object exists} \\ 0 & \text{Otherwise} \end{cases} \tag{2}$$

Each bounding box consists of 5 values, 4 representing its coordinates (*x,y,w,h*) and one representing the confidence score.

Regarding the grid cell, each one also predicts the number of C Conditional Class Probabilities, where C represents the number of classes and the Conditional Class Probabilities

 ISBN: 978-1-61208-787-0

represent the chance of an object to belong to class i (Equation 3).

$$\text{Conditional Class Probabilities} = P_r(Class_i|Object) \quad (3)$$

At the test time, the model multiplies the Conditional Class Probabilities and the individual box Confidence score:

$$P_r(Class_i|Object) * P_r(Obj) * IoU^{truth}_{pred} = P_r(Class_i) * IoU^{truth}_{pred} \quad (4)$$

Consequently, the above equation yields the score of the probability of the class appearing in the box and how the predicted box fits the object (Figure 3)[22].

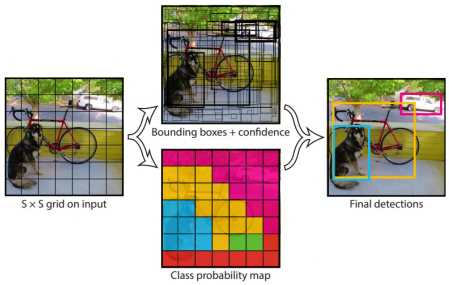


Figure 3. The representation of the two YOLO steps, identifying the objects and defining the probabilities of the classes in each grid [22].

The predictions results into a $S \times S \times (B * 5 + C)$ tensor. For example, if it takes $S = 7$, $B = 2$ and $C = 20$, we have a $(7, 7, 2 * 5 + 20) \rightarrow (7, 7, 30)$ tensor. This tensor provides the information about the Bounding Boxes and each class probabilities.

There are two types of YOLO algorithms: Regular YOLO and Fast (Tiny) YOLO. The Regular model consists of 24 convolutional layers followed by 2 fully connected layers (Figure 4). On the other hand, the Tiny model has 11 layers, 9 convolutional and 2 fully connected. As this architecture is much smaller compared to other detecting methods, it presents a fast response in object predictions, allowing the detection in real-time speed.

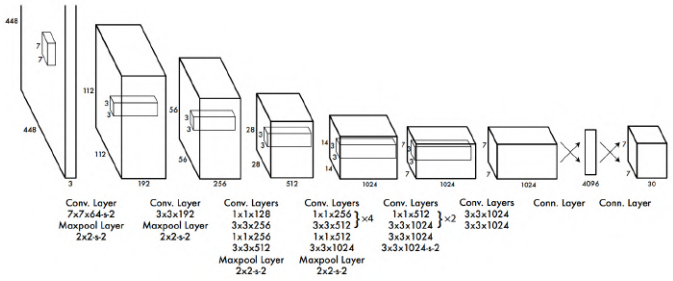


Figure 4. YOLO architecture [22].

The YOLOv2 represents the second version of YOLO where the objective is to improve the accuracy while making it faster. To achieve these features, the second model adds new features to improve the previous model performance. Among them, there are the adding of Batch Normalization in convolution layers, the high resolution classifier, convolutional with *Anchors* boxes, dimension clusters, direct location prediction, fine-grained features and a multi-scale training.

C. SSD

Single Shot Detector (SSD), like YOLO, takes only one shot to detect multiple objects present in an image using multibox. Furthermore, similar to Faster R-CNN, SSD model is built on a network architecture (backbone), e.g., the VGG-16 architecture [14], a high quality image classification model, without the final classification layers (fully connected layers). Instead, a set of convolutional layers are added, decreasing the size of the input to each subsequent layer, enabling to extract features at multiple scales. Thus, predictions for bounding boxes and confidence for different objects are done by multiple feature maps of different sizes (Figure 5).

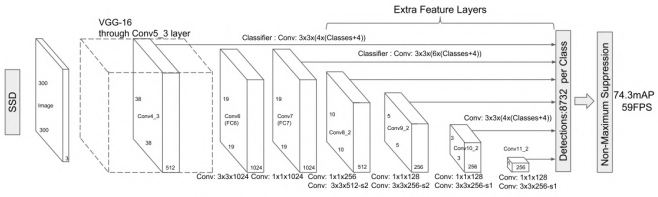


Figure 5. SSD architecture [24].

During the training process of SSD model, each added feature layer produces a detection prediction using convolutional filters (the "Extra Feature Layers" in Figure 5). For a layer of size $m \times n$ (number of locations) with p channels, the predicting map is a $3 \times 3 \times p$ kernel that produces a score for a category. For each location, the predicting map generating k bounding boxes. These boxes have different sizes (e.g., a 4 size box as in Figure 6). Furthermore, for each bounding box, there are computed c class score and 4 offsets relative to the original default box shape. At the training time, the default boxes are matched to the ground truth boxes, where the maching cases are treated as positives and the remaining as negatives. In this way, it results in a $(c+4)k$ filters that are applied around each location in feature map, resulting in a $(c+4)kmn$ outputs for an $m \times n$ map. Using the Figure 5 as an example, the result will be an 8732 bounding boxes.

During the training process, it is necessary to determine which boxes correspond to a ground truth detection. With this purpose, SSD model matches each ground truth box to the default box by their IoU ratio and, the boxes with an IoU value over than 0.5 are selected, simplifying the learning problem and allowing the network to predict high scores for multiple overlapping boxes.

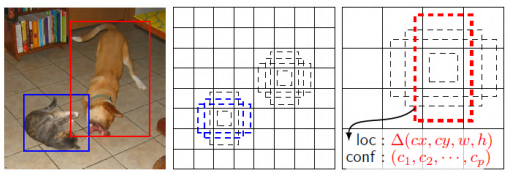


Figure 6. SSD Framework [24].

The SSD model is distinguished by two categories SSD512 and SSD300, working with images 512×512 and 300×300 respectively. SSD512 provides the best accuracy by detecting objects in large images, however the SSD300 model process the image almost twice as faster as SSD512 (Figure 7) [24].

To conclude, the SSD uses VGG-16 network as a backbone to extract features of images. However, different bases

Method	mAP	FPS	# Boxes	Input resolution
SSD300	74.3	46	8732	300×300
SSD512	76.8	19	24564	512×512

Figure 7. Comparison of SSD models [24].

(GoogleNet, MobileNet, AlexNet, Inception, etc.) can be used for better performance.

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Figure 8. Performance on ImageNet, comparison for different networks [17]. MAdds represents the counting of total number of Multiply-Adds.

In 2017, Howard presented the first MobileNet, a network that performs faster and nearly as accurate as VGG-16 network [16]. And in 2018, the second version of MobileNet has shown to be faster and more accurate than the last network [17] (Figure 8).

III. Proposal Work and Experimental Results

In this work, we seek for high speed and accurate detection of landmarks on the off-road track. To obtain the best result, we chose three models (Faster R-CNN, Fast YOLOv2 and the MobileNetv2 SSD300) and analyzed the result of each one for a real application.

This section presents the Training Stage, where all methods are trained considering the same conditions, as wall as the experiments comparing the different object detection models. Furthermore, the training and detection process were done with a GPU Nvidia Geforce 1060 3GB and CPU Intel i5 8500u on Windows 10 operational system.

A. Training Stage

To start the training process, it was chosen a white cone as the object for the reference detection to represent the limits of the road (Figure 9a). The training of this object was presented in Dhall work (June 2019). A monocular camera was used to detect and estimate the localization of a traffic cone in 3D world coordinates [27]. As the work aims to observe the performance of the models by detecting an object at the road boundary, the models were trained and tested to recognize the reference in this area.

The training was made using a set of different images with a single cone in different backgrounds and off-road tracks with cones spaced by 3 meters, which makes up most of the dataset (Figure 9).

To provide a fair comparison between the accuracy of the models, all object detection methods were adapted to train with the same dataset, 296 images (608×608 pixels).

The models were trained until the convergence of the Localization Loss value of each model. The Loss represents the quantitative measure of how much the predictions differ from the actual output (label). As an evaluation method, the localization loss value shows the difference of each model accuracy.

The loss of each model is represented by different equations [21][22][24]. To compare the predicted box localization

(a) Object reference. (b) Training example image.

Figure 9. Training process.

accuracy, it will be used only the *Localization Loss* which shows the errors of the predicted box localization when compared to the ground truth (Table I).

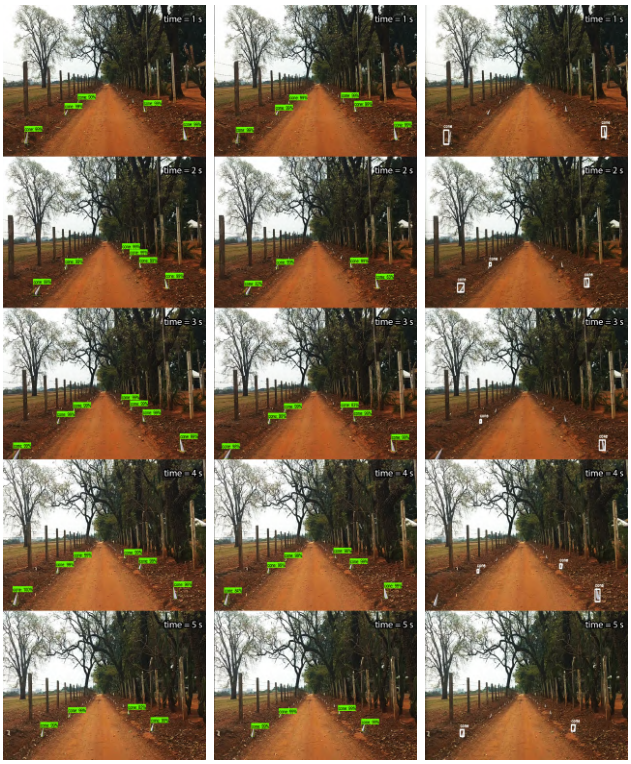
TABLE I. Model Loss comparative

Model	Loss	Training images
Faster R-CNN	0.017	296
MobileNetv2 SSD300	0.258	296
Fast YOLOv2	1.500	296

It is possible to observe from Table I that at the end of the training stage, the Fast Yolov2 presents a bigger error between the predictions and the real label with 1.5 of loss value. Furthermore, the Faster R-CNN and MobileNetv2 SSD300 showed small loss values, representing a good precision from these two methods.

B. Experiment 1

The first experiment was done by post-processing of an off-road driving video with similar characteristics of the dataset training. To compare the results of each model, it was chosen video frames of the off-road ride (Figure 10).



(a) Faster R-CNN Model. (b) SSD300 Model. (c) Fast YOLOv2 Model.

Figure 10. Comparative of three Detection Methods.

 ISBN: 978-1-61208-787-0

As can be observed from Figure 10, the Faster R-CNN model and the SSD300 presented comparable detection. The Faster R-CNN presents an advantage in accuracy, detecting the reference up to 12 meters ahead of the car (Figure 10a), while the SSD300 keeps the identification around 9 meters from the car (Figure 10b).

On the other hand, the Fast YOLOv2, which presented the biggest *Localization loss* value, showed the worst detector accuracy among the three models. The model resulted in a maximum detection of ~ 4 meters ahead from the car during all trajectory, failing sometimes to detect near references (Figure 10c).

Looking at the accuracy of each model, the Faster R-CNN showed the best choice for an autonomous driving application in off-road environments. However, besides a high precision in its detection, it is necessary a fast response of the reference identification. To estimate this, we tested 10 images, using different angles of the reference on the off-road street and, then the Mean Time Process (MTP) of each method was calculated (Table II).

TABLE II. COMPARATIVE OF MODELS. MTP: MEAN TIME PROCESSING; FPS: FRAMES PER SECOND

Model	Loss	Training images	MTP (seconds)	FPS
Faster R-CNN	0.017	296	3.14	00.3
MobileNetv2 SSD300	0.258	296	1.41	00.7
Fast YOLOv2	1.500	296	0.07	14.3

Through these results, it is possible to observe that the Faster R-CNN proved to be the slowest detector. It takes around 3 seconds to process each image, resulting in an unfeasible application for autonomous driving. On the other hand, the Fast YOLOv2, which showed a low precision in the detection, identified the references in less than 0.1 seconds. For this reason, it is expected to be a good detector for prevention moments, detecting nearby warnings, like animals or holes, or even tight curves.

Finally, the MobileNetv2 SSD300 showed an intermediary MTP. It detects all references in an image at around 1.4 seconds. The advantage of this model is that its accuracy was near to that of the Faster R-CNN, though SSD processing the image twice faster. For autonomous driving application, 1.4 seconds to detect the limits of a road in a curve presumably would result in a car off the track. On the contrary, in a straight road, such a fast steer correction may not be necessary and, this model can be useful.

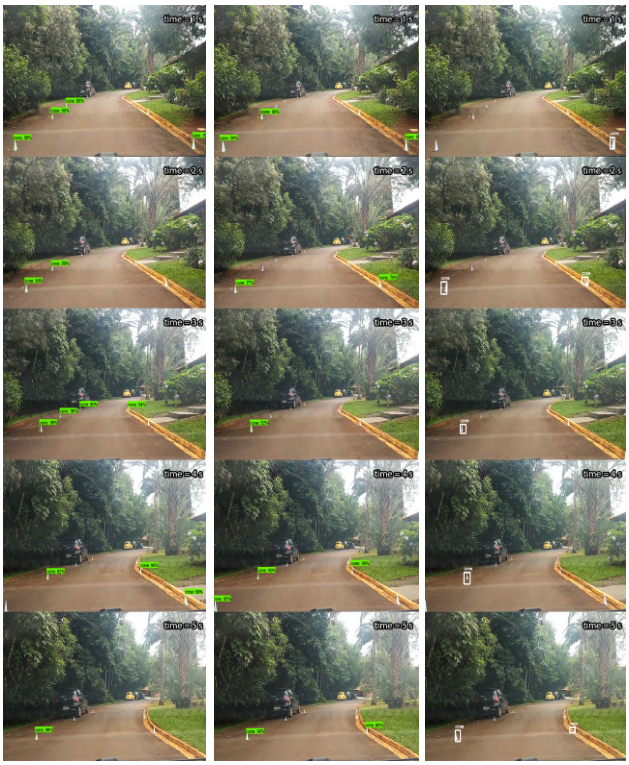
C. Experiment 2

For the purpose of observing eventually overfitting in the models, in the second experiment, the detection architectures were tested to identify the same reference in a different background from those they were trained.

The experiment was done by post-processing video of a car driving in a different environment from the training step (Figure 11).

The experiment presented a decrease in the detection of the Faster R-CNN and SSD300, where the models presented some failures in near detection. However, even with a small number of training images (296) from a different background, the methods did not present many false positives detection.

The Fast YOLOv2 had the smallest performance decrease. As in the first experiment, the model could only detect close references, failing sometimes to recognize them.



(a) Faster R-CNN Model. (b) SSD300 Model. (c) Fast YOLOv2 Model.

Figure 11. Comparative of three Detection Methods in a different environment.

Ten images were used in the new background scenario to observe the change in time detection. However, the models provided the same time process observed in the first experiment.

IV. CONCLUSION AND FUTURE WORKS

This work showed a Deep Learning application to identify reference marks on a road. The technique was implemented by training three existing Object Detection models with transfer learning, to identify a new object as a reference and comparing their results, Faster R-CNN, Fast YOLOv2 and MobileNetv2 SSD300.

While Faster R-CNN and SSD300 showed similar accuracy, detecting references in close and far ahead distances, Fast YOLOv2 model did not detect references when it was more than 4 meters ahead. Despite the precision of Fast YOLOv2, it processes the detection in less than 0.1 seconds, providing the faster detection among the three models.

While the Fast YOLOv2 presented a fast detection, the Faster R-CNN showed the slowest detector. It spent more than 3 seconds to detect the references in each image. On the other hand, the SSD300 with an accuracy similar to that of the Faster R-CNN, processes the image twice faster, detecting objects in about 1.5 seconds.

Each model presented a particularity in accuracy and speed that is important for autonomous cars to drive safely on a road without line lanes. The Faster R-CNN presented the best accuracy of all, though it is considerably slow for an autonomous drive application. The SSD300 with a similar precision was proven to be a good model to identify the limits on straight roads. With a low process time to detect references,

 ISBN: 978-1-61208-787-0

the model could only analyzes and determine the action to keep the vehicle in the middle of the road at every 1.5 seconds.

In addition, the Fast YOLOv2 only detected close references which can be useful for fast detection, e.g., tight curves or emergency situations, allowing the machine to make fast decisions on a drive.

For an autonomous driving application, that uses a powerful GPU and CPU, the implementation of SSD and Fast YOLOv2 should result in a safe drive. The first one for detecting the main limits of the road and the second one to work as a warning system for fast actions.

Future works in this project are aimed to explore alternative architectures, including the recent versions of the models presented at this work. Additionally, the work pretends to label a test dataset to provide a quantitative evaluation of the models or to conduct statistical tests known from the literature [28][29].

Searching for generalizing the ideas of this project, the work will extend the detection from the reference object described in the text to real objects that can be used to represent the road boundary in an off-road environment.

ACKNOWLEDGMENT

The authors acknowledge the funding received from: Project INCT-SAC - Aut. Collaborative Systems - (CNPq 465755/2014-3, FAPESP 2014/50851-0) and Fapesp AutoVERDE (p. 2018/04905-1).

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[2] S. Lee et al., "Vpgnet: Vanishing point guided network for lane and road marking detection and recognition," in 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 1965–1973.

[3] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Los Alamitos, CA, USA, jun 2016, pp. 3234–3243.

[4] J. M. Alvarez, Y. LeCun, T. Gevers, and A. M. Lopez, "Semantic road segmentation via multi-scale ensembles of learned features," in Computer Vision – ECCV 2012. Workshops and Demonstrations. Springer Berlin Heidelberg, 2012, pp. 586–595.

[5] T. P. Breckon, "From On-Road to Off : Transfer Learning within a Deep Convolutional Neural Network for Segmentation and Classification of Off-Road Scenes," pp. 1–14.

[6] S. Adhikari, C. Yang, K. Slot, and H. Kim, "Accurate natural trail detection using a combination of a deep neural network and dynamic programming," Sensors, vol. 18, no. 2, Jan. 2018, p. 178.

[7] C. Caraffi, S. Cattani, and P. Grisleri, "Off-road path and obstacle detection using decision networks and stereo vision," IEEE Transactions on Intelligent Transportation Systems, vol. 8, no. 4, 2007, pp. 607–618.

[8] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2016-Decem, no. 600388, 2016, pp. 3234–3243.

[9] P. Chao, C.-Y. Kao, Y.-S. Ruan, C.-H. Huang, and Y. Lin, "Hardnet: A low memory traffic network," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 3551–3560.

[10] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang, "Fasterseg: Searching for faster real-time semantic segmentation," ArXiv, vol. abs/1912.10917, 2020, accessed on 31.08.2020.

[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," International Journal of Computer Vision, vol. 88, no. 2, 2010, pp. 303–338.

[12] C. Papageorgiou and T. Poggio, "Trainable system for object detection," International Journal of Computer Vision, vol. 38, no. 1, 2000, pp. 15–33.

[13] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," CoRR, vol. abs/1808.01974, 2018, accessed on 31.08.2020. [Online]. Available: http://arxiv.org/abs/1808.01974

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR, vol. abs/1409.1556, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, accessed on 31.08.2020. [Online]. Available: http://arxiv.org/abs/1512.03385

[16] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," vol. abs/1704.04861, 2017, accessed on 31.08.2020.

[17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.

[18] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, vol. 2017-January, 2017, pp. 6517–6525.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.

[20] R. Girshick, "Fast R-CNN," Proceedings of the IEEE International Conference on Computer Vision, vol. 2015 Inter, 2015, pp. 1440–1448.

[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, 2017, pp. 1137–1149.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," accessed on 31.08.2020. [Online]. Available: http://arxiv.org/abs/1506.02640

[23] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," accessed on 31.08.2020. [Online]. Available: http://arxiv.org/abs/1804.02767

[24] W. Liu et al., "SSD: Single shot multibox detector," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9905 LNCS, 2016, pp. 21–37.

[25] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2999–3007.

[26] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," International journal of computer vision, vol. 104, no. 2, 2013, pp. 154–171.

[27] A. Dhall, D. Dai, and L. Van Gool, "Real-time 3D Traffic Cone Detection for Autonomous Driving," 2019, pp. 494–501.

[28] G.-F. Fan, L.-L. Peng, W.-C. Hong, and F. Sun, "Electric load forecasting by the SVR model with differential empirical mode decomposition and auto regression," Neurocomputing, vol. 173, Jan. 2016, pp. 958–970. [Online]. Available: https://doi.org/10.1016/j.neucom.2015.08.051

[29] Z. Zhang, S. Ding, and Y. Sun, "A support vector regression model hybridized with chaotic krill herd algorithm and empirical mode decomposition for regression task," Neurocomputing, vol. 410, Oct. 2020, pp. 185–201. [Online]. Available: https://doi.org/10.1016/j.neucom.2020.05.075

 ISBN: 978-1-61208-787-0