Combination of Computer Vision Detection and Segmentation for Autonomous Driving

Yu-Ho Tseng, Shau-Shiun Jan
Department of Aeronautics and Astronautics
National Cheng Kung University
Tainan, Taiwan

***Abstract*—Most existing deep learning networks for computer vision attempt to improve the performance of either semantic segmentation or object detection. This study develops a unified network architecture that uses both semantic segmentation and object detection to detect people, cars, and roads simultaneously. To achieve this goal, we create an environment in the Unity engine as our dataset. We train our proposed unified network that combines segmentation and detection approaches with the simulation dataset. The proposed network can perform end-to-end prediction and performs well on the tested dataset. The proposed approach is also efficient, processing each image in about 30 ms on an NVIDIA GTX 1070.**

Keywords: computer vision, detection, segmentation, autonomous driving.

I. Introduction

Many research institutes and companies have developed computer vision systems for self-driving cars [1] [2]. Computer vision has thus received extensive research attention [3] [4]. The detection of the shape of roads, cars, and people can provide reliable information for computer vision systems in self-driving cars to know where to go and what to avoid. Deep learning networks for computer vision can be divided into two main types, namely those based on semantic segmentation [5] [6] and those based on object detection [7] [8]. Existing advanced deep learning networks for autonomous cars are based on semantic segmentation, in which objects are detected based on edge detection. However, such detection can be inaccurate. Since classification for semantic segmentation is conducted pixel by pixel, detected edges are often rough and blurry. Inaccurate segmentation may lead to accidents. In object detection, the shape of an object is represented as a rectangular box, providing a margin of safety. However, object detection is unsuitable for curved items, which produce excessive classification area error. Therefore, a reliable computer vision system for autonomous cars requires the integration of object detection and semantic segmentation. This study integrates semantic segmentation and object detection and designs a unified network to take advantage of each approach. A segmentation network is first trained, and then its weights are used to fine-tune the additional detection detector layer. The well-trained network can perform end-to-end prediction.

II. Related work

This section reviews semantic segmentation and object detection. We mainly focus on the suitability of integrating these approaches.

A. Semantic Segmentation

Early approaches used sliding windows as a small detector on an image and used convolutional neural networks (CNNs) as a classifier to label each small detector [10]. This is very time-consuming and accuracy is unsatisfactory. Fully convolutional networks (FCNs) [5] were proposed to process semantic segmentation, allowing an end-to-end deep learning pipeline. FCNs adopt max pooling for effective classification and use transposed convolution [11] in upsampling to preserve image size. The proposed network uses the concept of an end-to-end pipeline in the segmentation decoder.

B. Object Detection

Conventional object detection approaches used two steps: regions are extracted using a sliding window or selective search and then scored using a CNN classifier to detect objects [12]. Therefore, traditional approaches have two ways to improve performance: (i) develop a better method for extracting regions; (ii) improve the classification model. Methods that use a single deep learning network have been proposed, including the You Only Look Once (YOLO) algorithm [8]. YOLO shows that CNNs can make use of different areas of visual perception as good proposed regions and that CNNs are a great classifier. Single deep learning network can significantly reduce computation time and improve inference accuracy. This is most suitable for real-time detection applications. The Single Shot Multibox Detector (SSD) [9] uses some middle layers, called feature maps, in the network to detect objects of different sizes instead of predicting using the last layer, as done in traditional networks. This makes the prediction size-invariant and more accurate. Therefore, we use the concept of feature maps to train our detection decoder.

978-1-5386-1647-5/18/$31.00 ©2018 IEEE

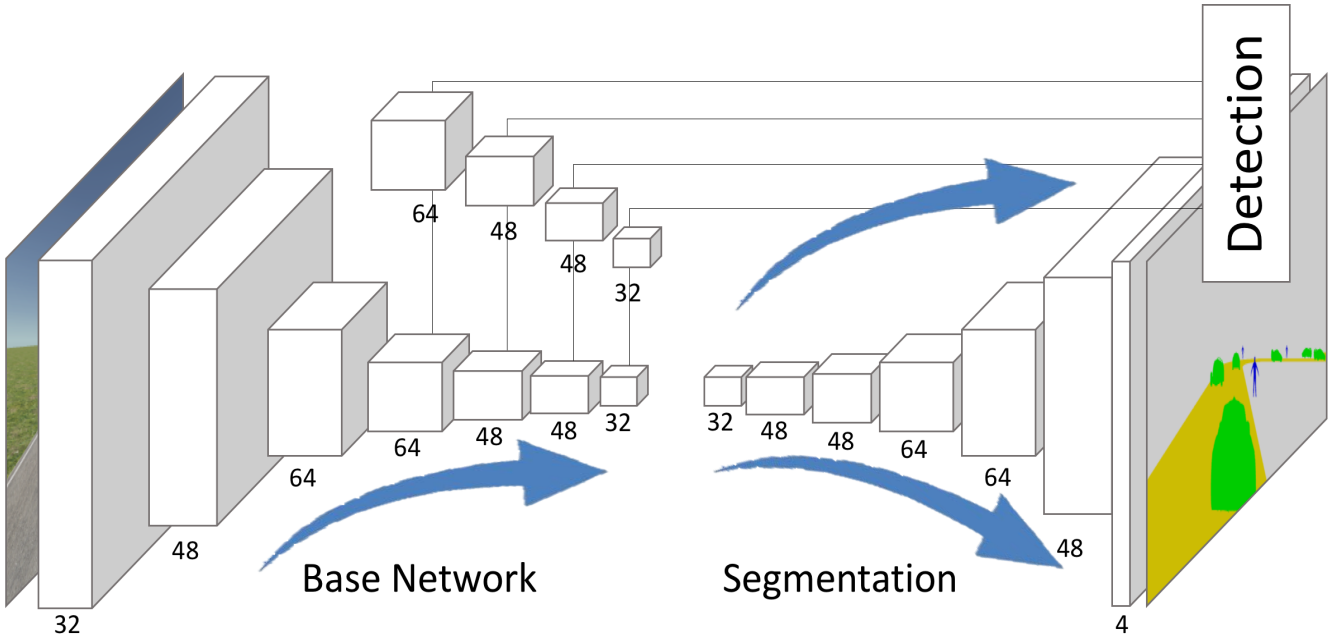

Figure 1: Illustration of proposed network architecture. The depth of each filter is shown under the specific layer.

III. Proposed Network

This study proposes a unified network that combines semantic segmentation and object detection approaches. Semantic segmentation is used for detecting roads and object detection is used for detecting cars and people. Figure 1 shows the proposed architecture. FCNs [5] and SSD [9] are used to build our model. Our network uses SSD7 [13] as a reference. SSD7 (proposed by user pierluigiferrari on Github) detects cars and pedestrians with Udacity, a dataset for self-driving cars. SSD7 was chosen because the size of the network is small and the performance is adequate, as shown in Figure 2, with our simulation data. In Figure 2, the blue rectangle is the prediction for cars and the green rectangle is the prediction for people. The proposed network architecture has three parts: base network, segmentation decoder, and detection decoder.

A. Base Network Architecture

SSD7 has 7 layers. The input image has dimensions of 256 (height) x 256 (width) x 3 (number of channels). Dimensions of 256 x 256 were chosen to facilitate segmentation and upsampling without changing output size. As shown in Figure 1, each layer is composed of convolution, batch normalization, and max pooling, except for the last layer, which lacks max pooling. In the Figure 1, the depth of each filter is shown under each layer. The filter size of each layers is 3 x 3, except that for the first convolution filter, whose size is 5 x 5 (visual perception in the first layer is very small so a larger size is required to improve vision).

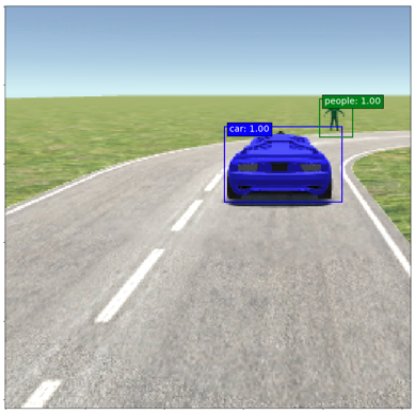

Figure 2: Prediction results for SSD7 with simulation data.

B. Segmentation Decoder

For the segmentation decoder, we follow MLND-Capstone [14] by user mvirgo on Github, which is based on a deconvolution network for semantic segmentation [15]. The network of MLND-Capstone is composed of 7 convolution layers, 3 max pooling layers, 3 upsampling layers, and 7 transposed convolution layers. We modified the MLND-Capstone network to fit our base network architecture, giving it 7 upsampling layers and 7 transposed convolution layers (see Figure 1, segmentation part). In the last layer, the depth of the transposed convolution layer is 4; there are thus 4 classes, namely background, people, cars, and roads. The prediction obtained using the modified segmentation part is shown in Figure 3 (yellow: road; green: car; red: person).

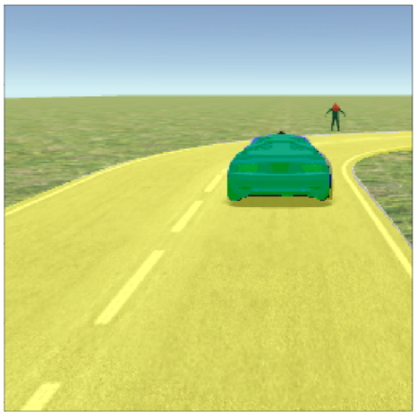

Figure 3: Prediction results for segmentation with simulation data. (yellow: road; green: car; red: person).

C. Detection Decoder

For the detection decoder, we follow the architecture of SSD7. As shown in Figure 1, there are 4 extra feature layers connected to the last 4 layers. These feature layers can provide different areas of visual perception and more detectors for training and prediction. Because of the different areas of visual perception, the network can deal with a range of sizes. We also added 1 extra convolution layer after each feature layer to increase learning depth for targets. We set the extra convolution filter depth to 3 (background, people, and cars).

IV. Training Details

In this section, we describe the loss functions and provide details of the integration of segmentation and detection.

A. Loss Functions

We define loss functions for segmentation and object detection. In segmentation, we take the MLND-Capstone [14] loss function as a reference and employ mean square error as our loss function, which is sufficient to detect roads. This function is defined below, where $\hat{Y}_i$ is the prediction classification of the i-th pixel, Y_i is the ground truth of the relative pixel classification, and n is the total number of pixels of the input image.

$$mean\ square\ error = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \tag{1}$$

The detection loss function is more complicated. We use the loss function for SSD [9], which comprises confidence and location loss:

$$Loss(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{2}$$

Where N is the number of matched default boxes. If N = 0, loss equals 0. x_{ij}^p is an indicator of matching the i-th default box to the j-th ground truth box of category p. The localization loss is a Smooth L1 loss used to calculate the relation between the predicted box (l) and the ground truth box (g):

$$smooth_{L1} = \begin{cases} 0.5x^2 & if\|x\| < 1 \\ \|x\| - 0.5 & otherwise \end{cases} \tag{3}$$

Localization is the loss between the prediction and the ground truth. For SSD, we regress to the offsets for the center (cx,cy) of the bounding box (d) in the feature map and for its width (w) and height (h):

$$L_{loc}(x, l, g)) = \sum_{i\in Pos}^{N}\sum_{m\in\{cx,cy,w,h\}} x_{ij}^p smooth_{L1}(l_i^m - \hat{g}_j^m) \tag{4}$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \tag{5}$$

$$\hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h \tag{6}$$

$$\hat{g}_j^w = (\frac{g_j^w}{d_i^w}) \tag{7}$$

$$\hat{g}_j^h = (\frac{g_j^h}{d_i^h}) \tag{8}$$

The confidence loss is the softmax loss over multiple class confidences (c):

$$L_{conf}(x, c)) = -\sum_{i\in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i\in\{Neg\}} log(\hat{c}_i^0) \tag{9}$$

where $\hat{c}_i^p = \frac{exp(x_i^p)}{\sum_p exp(c_i^p)}$, and in our case we set α to 1.

B. Training Strategy

There are two steps in the training process. First, we train the base network and segmentation decoder without the detection decoder. Then, the weights of the well-trained base network and segmentation are used to fine-tune the detection decoder. The segmentation decoder is trained first because segmentation is better than edge detection for roads. This is why we set the road ground truth in segmentation rather than in detection. Although detection is used for people and cars, the ground truth for people and cars is given in segmentation. We train the segmentation classifier with people and cars so that the network can learn their features. This improves detection training since people and car features are used in fine-tuning weights. In the training process, the segmentation network is first trained to learn road features. Figure 4 shows the network prediction with segmentation for people, cars, and roads, and with detection for people and cars (in rectangles).

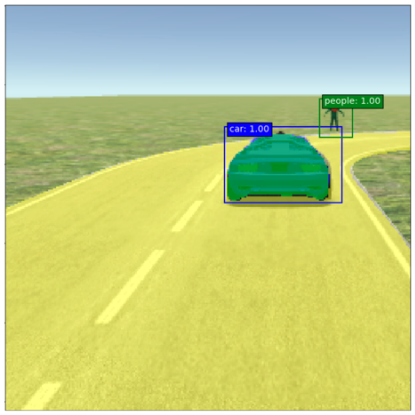

Figure 4: Segmentation (person: red; car: green; road: yellow) and detection (person: green box; car: blue box) results with cars and people.

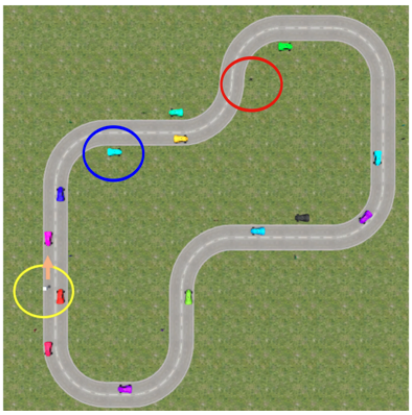

Figure 5: Unity simulation environment (blue circle: car; red circle: person; yellow circle: driver).

V. Experimental Results

This section describes the simulation environment. We analyzed performance in three situations: segmentation alone, detection alone, and combined approach. Performance was evaluated in terms of accuracy, prediction time (data only), and total time (data and images on screen). All tests were run on an NVIDIA GTX 1070 graphics card. Networks with segmentation alone and detection alone were trained without fine-tuning. For segmentation alone, a branch from the proposed network composed of the base network and segmentation was used (see Figure 1). For detection alone, a branch from the proposed network composed of the base network and detection was used.

A. Dataset

A Unity engine simulation environment was used as our dataset (see Figure 5). The simulation environment contained 5 types of things: people, cars, roads, grass, and the sky. Figure 6 shows the driver's view of the scene (two cars and a person on a road). The Unity engine was used because it allows the semantic segmentation ground truth to be easily labeled. All objects in Unity are modular; for example, a car in the simulation environment is not just a three-dimensional model, it is an independent object. This can be helpful because we can set the texture of each object by writing scripts. We can set the simulation environment by changing the object's texture to get the segmentation ground truth without a manual process, which reduces the time required for labeling the ground truth. The left image in Figure 7 shows the semantic segmentation ground truth of Figure 6. Two cars in green and one person in blue with a clear shape can be seen in the image. However, the object detection ground truth needs to be manually labeled. We used the Matlab imageLabeler tool to label objects. The right image in Figure 7 shows the object detection ground truth.

Figure 6: Driver's view in simulation environment.

B. Performance Evaluation of Segmentation Alone

The following segmentation accuracy formula was used [16]:

$$Seg.accuracy = \frac{true\ pos.}{true\ pos. + false\ pos. + false\ neg.} \quad (10)$$

Where "true pos." is a prediction corresponding to the ground truth, "false pos" is a prediction that is a false alarm, and "false neg" is a prediction that did not detect the target although it was present.

For evaluating segmentation performance, only road accuracy was analyzed. Table I shows the segmentation accuracy,

TABLE I: Performance of Segmentation Alone

	Segmentation
Accuracy	99.70%
Prediction Time	0.005 s
Total Time	0.031 s
Processing Speed	32.26 fps

total time, and prediction time. Since an image is not shown, prediction time is much lower than total time.

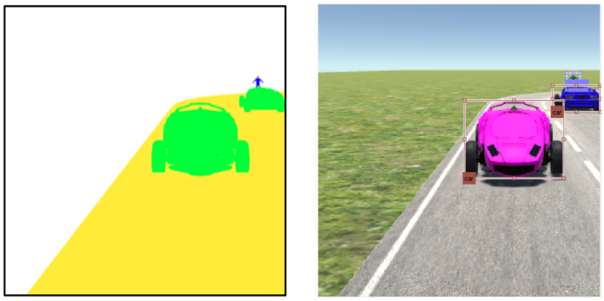

Figure 7: Ground truth for semantic segmentation (left) and object detection (right).

C. Performance Evaluation of Detection Alone

We used mean average precision (mAP) to evaluate the performance of detection alone. We set the intersection over union (IoU) threshold to 0.5. If the classification of prediction matches ground truth and IoU meets the threshold, the detection is a correct prediction.

For this detection evaluation, we trained the reference network SSD7 [13] with our dataset. The performance results are shown in Table II.

TABLE II: **Performance of Object Detection Alone**

	Object Detection
Accuracy	99.51%
Prediction Time	0.004 s
Total Time	0.031 s
Processing Speed	32.26 fps

D. Proposed Approach

Figure 8 shows that our segmentation for roads and detection for cars and people work adequately. Roads are marked yellow by semantic segmentation. People and cars are marked in green and blue rectangular boxes, respectively, by object detection. The prediction results are shown in Figure 8. There are four groups of images. In each group, the left image is the driver's view and the right image is the prediction by our network. Table III shows the performance of our network.

The accuracy values of segmentation alone and the proposed segmentation are the same (99.70%). The accuracies are the same because our network is trained at segmentation first, with weights used to fine-tune our detection. However, the accuracy values of detection alone (SSD7) and the proposed detection are different, with that of the latter being lower. This is because our detection is fine-tuned by the weights from the base network segmentation (see Figure 1), which may restrict the network's flexibility. Nevertheless, our detection accuracy is 0.05% less than that of SSD7 and is beyond 99% for our dataset. The computation times of the proposed approach, segmentation alone, and detection alone are shown in Table III. The proposed approach is about 1-2 ms slower than the other approaches.

TABLE III: **Performance Comparison Among Segmentation, Detection and Proposed Approach**

	Segmentation	Detection	Proposed
Accuracy	99.70%	99.51%	99.46%
Prediction Time	0.005 s	0.004 s	0.006 s
Total Time	0.031 s	0.031 s	0.0033 s
Processing Speed	32.26 fps	32.26 fps	30.30 fps

To sum up, the accuracy of the proposed approach is similar to that of SSD7 for our dataset. Although the processing time of our approach is higher than those of segmentation alone and detection alone, our approach still achieves 30 fps. Our approach is 0.05% less accurate and 1-2 ms slower while doing both segmentation and detection for various purposes, such as tracking or detecting. If prediction alone without showing image on screen, the computation time is only 6 ms ($\sim$166 fps) with only one GTX 1070.

VI. Conclusion

This study developed a unified deep learning architecture that combines object detection and semantic segmentation. The performance of the proposed network was evaluated using a simulation environment. Our approach can perform end-to-end prediction. It has 99.70% accuracy for the segmentation decoder and 99.45% accuracy for the detection decoder for our simulation environment data. Our approach is efficient as well; with an NVIDIA GTX 1070, it can perform all tasks in only 6 ms and show an image on the screen in about 30 ms. In the future, we plan to implement our model in a more complicate environment and train our network end-to-end to make the network more powerful.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316, 2016.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316, 2016.

[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

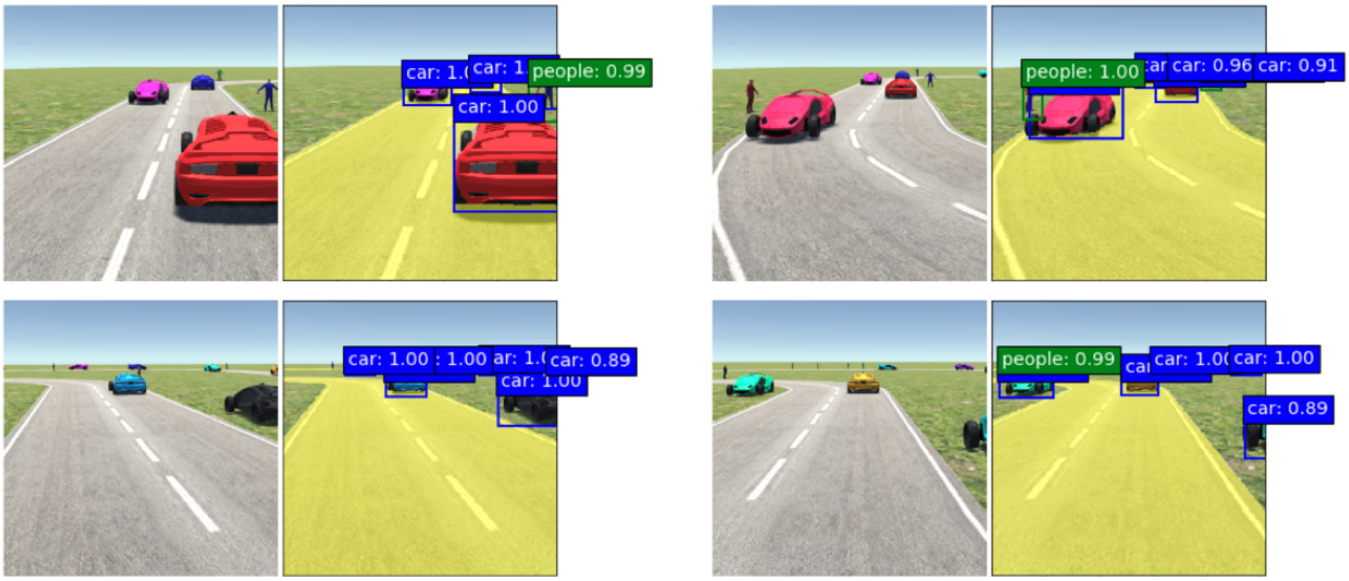

Figure 8: **Prediction results for proposed approach. Input image (left) and prediction (right).**

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classifcation with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097∼1105. Curran Associates, Inc., 2012.

[5] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. CVPR (to appear), Nov. 2015.

[6] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. CoRR, abs/1502.03240, 2015.

[7] R. B. Girshick. Fast R-CNN. CoRR, abs/1504.08083, 2015.

[8] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unifed, real-time object detection. CoRR, abs/1506.02640, 2015.

[9] R. B. Girshick. Fast R-CNN. CoRR, abs/1504.08083, 2015.

[10] H. Li, R. Zhao, and X. Wang. Highly effcient forward and backward propagation of convolutional neural networks for pixelwise classification.

[11] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 2528-2535. IEEE, 2010.

[12] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1-8. IEEE, 2008.

[13] Pierluigiferrari, ssd_keras. https://github.com/pierluigiferrari/ssd_keras. 2017.

[14] Michael Virgo, MLND-Capstone. https://github.com/mvirgo/MLND-Capstone, 2017.

[15] Noh, Hyeonwoo, Hong, Seunghoon, and Han, Bohyung. Learning deconvolution network for semantic segmentation. In ICCV, 2015.

[16] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C., Winn, J., and Zisserman, A. The Pascal visual object classes challenge: A retrospective. IJCV, 111(1):98-136, 2015.