BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

– MIRPR report –

Team members
Ronald Galea, Mathematics Computer Science, 322, gallearonald@yahoo.com
Andrei Popovici, Mathematics Computer Science, 322, pasandrei98@gmail.com
Ionut Farauanu, Computer Science, 933, farauanuionut@gmail.com

2020

Abstract

Object detection is considered a difficult computer vision problem, with many real world applications, one example being automotive assistance. State of the art results have consistently been delivered by deep learning methods, however, most of these algorithms are extremely computationally heavy and are not feasible in practice. We aim to combine several different techniques in order to obtain decent performance to latency ratio, as well as make training easier on a smaller set of classes. Specifically, we use an SSD detection model [5], powered by a MobileNetV2 feature extractor (SSDLite [9]), trained with focal loss [3] to combat class imbalance.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 What? Why? How?

We are addressing the problem of object detection in the automotive context.

- What is the (scientific) problem?

  The problem consists of constructing bounding boxes for a predetermined set of classes of objects.

- Why is it important?

  Preventing and predicting possible accidents in order to make driving a less stressful experience.

- What is your basic approach?

  Our approach is based on Deep Learning methods.

1.2 Paper structure

The main contribution of this report is to present an intelligent algorithm for solving the problem of object detection on public roads. We aim to combine several methods in order to obtain a good performance to speed ratio.

The second chapter/section is a short introduction to the problem we try to solve and the method we plan to use.

The third chapter/section describes the current State of the Art Deep Learning models.

The chapter/section 4 details how we used the main ideas from 3 of the most important papers regarding object detection: the SSD feature map [5], MobileNetV2 [9] as a backbone and lastly, Focal Loss [3].

Chapter 2

Scientific Problem

2.1 Problem definition

Object Detection is a challenging Computer Vision problem which deals with identifying and locating object of certain classes in the image. Interpreting the object localisation can be done in various ways, including creating a bounding box around the object. State-of-the-art object detection algorithms are based on Deep Neural Networks.

Concretely, the input consists of a RGB video stream that will be split into frames which will then be analyzed. Afterwards, as output the aim is to cast a video to a screen located on the car's dashboard, with the corresponding bounding boxes for each object.

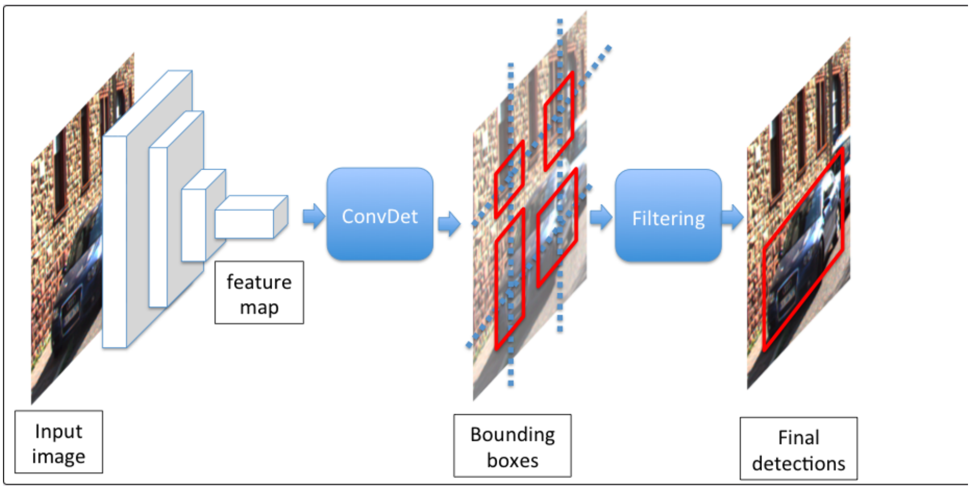


Figure 2.1: Object Detection Pipeline
Taken from https://mez.github.io/2017/04/21/squeezedet-deep-learning-for-object-detection

Chapter 3

State of art/Related work

In this Chapter we will present the current methods utilised to solve the object detection problem.

3.1 Faster R-CNN

In their paper [11], they present a new approach for the Fast R-CNN algorithm. Faster R-CNN consists of two components: a fully convolutional Region Proposal Network (RPN) for proposing candidate regions, followed by a downstream Fast R-CNN classifier and they say that despite its leading accuracy on several multi-category benchmarks, Faster R-CNN has not presented competitive results on popular pedestrian detection datasets (e.g., the Caltech set). Their new approach consisted of two components: a Region Proposal Network (RPN) that generates candidate boxes as well as convolutional feature maps, and a Boosted Forest that classifies these proposals using these convolutional features. The main downside of this approach is that 2 forward passes are required for a single prediction.

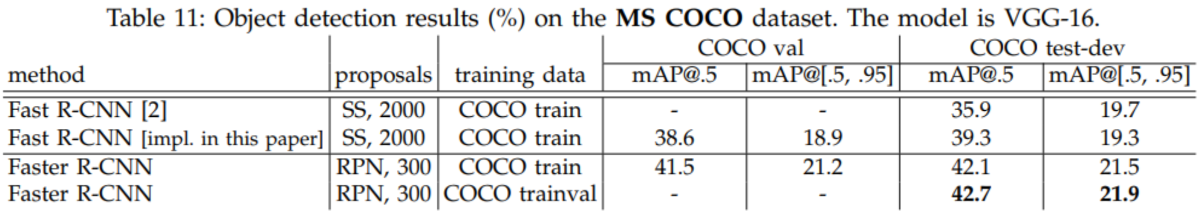


Table 11: Object detection results (%) on the **MS COCO** dataset. The model is VGG-16.

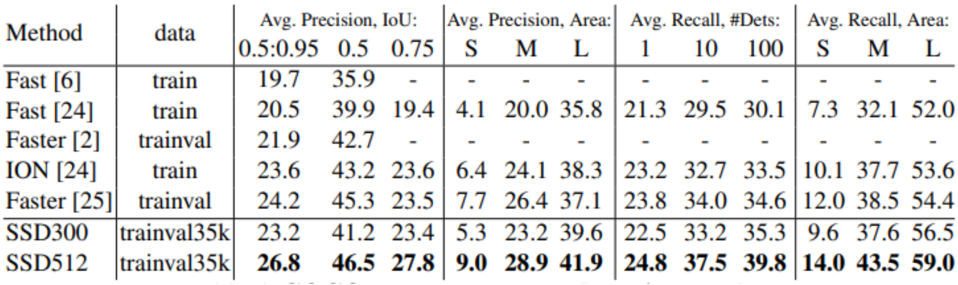
method	proposals	training data	COCO val		COCO test-dev	
			mAP@.5	mAP@[.5, .95]	mAP@.5	mAP@[.5, .95]
Fast R-CNN [2]	SS, 2000	COCO train	-	-	35.9	19.7
Fast R-CNN [impl. in this paper]	SS, 2000	COCO train	38.6	18.9	39.3	19.3
Faster R-CNN	RPN, 300	COCO train	41.5	21.2	42.1	21.5
Faster R-CNN	RPN, 300	COCO trainval	-	-	**42.7**	**21.9**

Table 3.1: Faster R-CNN mAP on COCO

3.2 SSDNet

W. Liu et. al [5] present a method for detecting objects in images using a single deep neural network. Their approach, named SSD, discretizes the output space of bounding boxes into a set of default boxes, which the network learns to alter so that they better fit the objects. This makes it a single-pass network resulting in it being faster than the Faster R-CNN. Training loss is composed of both localization and classification loss. One known weakness of single shot detectors is class imbalance. Unlike region proposal networks, single shot detectors rely on a dense set of anchors to cover the image 3.1, which in turn leads to an overwhelming majority of these predicting background most of the time. To combat this, hard negative mining is used, which samples positive to negative examples to a ratio of about 1 to 3, sampling those background examples on which the classification loss is highest

SSD300 with VGG16 as feature extractor achieves 23.2% mAP with 14.8M parameters, 1.25B Madds and 46 FPS on a Titan X



Method	data	Avg. Precision, IoU:			Avg. Precision, Area:			Avg. Recall, #Dets:			Avg. Recall, Area:		
		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast [6]	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast [24]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster [2]	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [24]	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
Faster [25]	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD300	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD512	trainval35k	**26.8**	**46.5**	**27.8**	**9.0**	**28.9**	**41.9**	**24.8**	**37.5**	**39.8**	**14.0**	**43.5**	**59.0**

Table 3.2: SSDNet mAP on COCO - detailed

3.3 MobileNetV2: Inverted Residuals and Linear Bottlenecks

MobileNetV2 [9] is an efficient mobile architecture that uses two novel techniques: Inverted Residuals and Linear Bottlenecks on top of Depthwise Separable Convolutions from the original MobileNet [2]. In the MobileNetV2 paper, M. Sandler et. show that it can be used as a feature extractor for SSD, resulting in the creation of SSDLite.

This new model achieves 22.1% mAP with 4.3M parameters, 0.8 Madds and 5 FPS on a mobile phone.

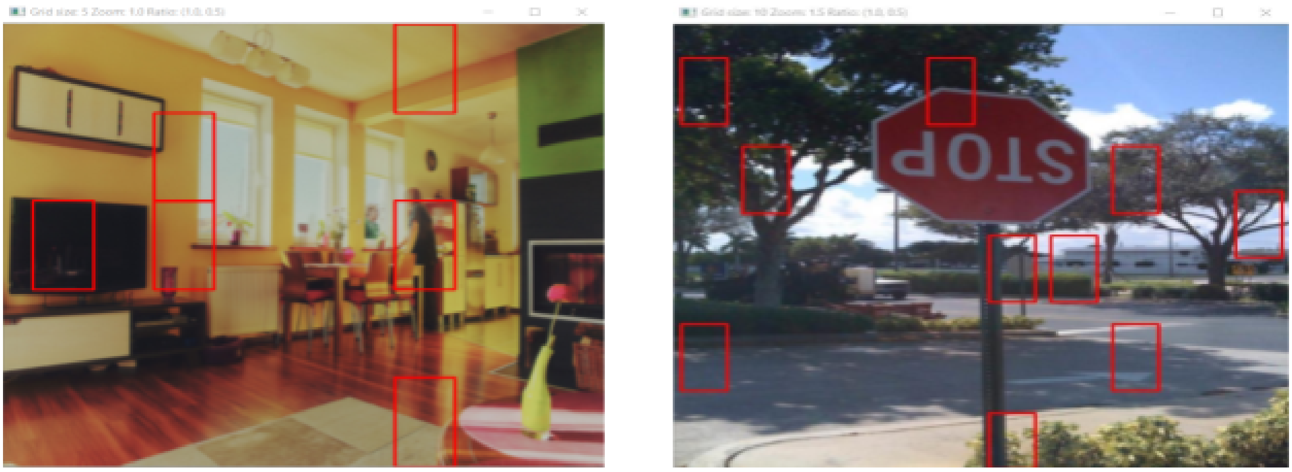
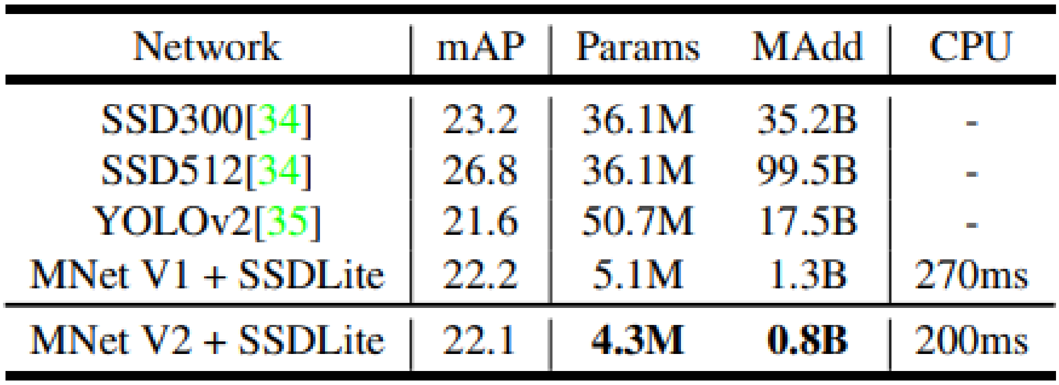

Grid size: 5 Zoom: 1.0 Ratio: (1.0, 0.5)
Grid size: 10 Zoom: 1.5 Ratio: (1.0, 0.5)
STOP

Figure 0: Sparsely plotted anchors

Figure 3.1: Anchors



Network	mAP	Params	MAdd	CPU
SSD300[34]	23.2	36.1M	35.2B	-
SSD512[34]	26.8	36.1M	99.5B	-
YOLOv2[35]	21.6	50.7M	17.5B	-
MNet V1 + SSDLite	22.2	5.1M	1.3B	270ms
MNet V2 + SSDLite	22.1	**4.3M**	**0.8B**	200ms

Figure 3.2: SSDLite mAP on COCO and speed

Chapter 4

Proposed approach

4.1 Main idea

The main drawback of large neural networks is the immense computational requirement that comes along. As an example, the number on spot on the COCO detection [4] leaderboard is occupied by MegDet [6], who train their network using 128 GPUs on with a large batch size. Recently, attention has drifted more towards lightweight networks [2] [9].

Our approach is based on SSD and MobileNetV2 techniques. Given that we attempt to detect a small number of classes we expect to achieve decent performance with a much smaller model as the original SSDLite was used on 80 classes from the COCO dataset.

We aim to use COCO dataset for pretraining only considering images with vehicles and people and afterwards we will do inference on JAAD which is specific for pedestrian detection.

We closely follow the method described in [5], which includes predefining a set of anchors (priors) to cover (hopefully) most of the possible areas objects may appear in images. For the model to be able to detect objects at multiple scales, we make predictions using the 10x10, 5x5, 3x3, 2x2 and respectively the 1x1 feature map cells. Concretely, we compute confidence scores and offsets from our anchors. Detections are made with a single forward pass through the network.

However, the original SSD (which uses VGG [10] as backbone) is still quite a heavy model, and thus we seek something more feasible computationally. SSDLite [9] is a model that fits our criteria, as it is well optimized for efficient inference, while also achieving good performance, using MobileNetV2 as feature extractor.

Finally, since we only aim to train on a few classes, we expect the problem of class imbalance (foreground vs background predictions) to be quite severe. In order to alleviate this issue, we train using focal loss [3], replacing hard negative mining.

It should be noted that this work does not bring any novel methods, nor does it aim to advance the SOTA. We simply experiment using the above three main ideas.

4.2 The hyperparameters that we used

Parameter	**Value**
Initial Learning rate	0.01
Batch size	16
Optimizer	adam
Resolution	320x320
Focal Loss gamma	2
Focal Loss alpha	0.25
IoU mapping threshold	0.5
prediction confidence threshold	0.25
NMS suppression threshold	0.46
weight decay	0.0001
number of epochs	40
first decay	10
second decay	20
third decay	35
decay rate	0.1

Table 4.1: List of Hyperparameters

4.3 Detailed description

4.3.1 Transfer Learning

Given the difficulty of training DCNNs, it is almost always a good idea to make use of transfer learning. PyTorch provides numerous pretrained models on ImageNet [1], and we chose MobileNetV2 as our backbone.

4.3.2 Learning Rate and Handling a pretrained model

For the first 3 epochs, the backbone is completely frozen. Starting with the fourth epoch, we gradually unfreeze 3 layers every epoch. Additionally the learning rate for the backbone is, initially multiplied by 0.2. We divided the learning rate by 10 at epochs 10, 20 and 35.

4.3.3 Loss

The loss is computed as the sum of two components: the localization loss and the classification loss. The localization loss is simply the L1 difference between the ground truth bounding boxes and those predicted by the model, whereas the classification loss consists of a Binary Cross Entropy taken between the ground truth values and the confidence scores given by the model. It is worth noting that only the feature map cells that have mapped to ground truth instances have the localization loss accounted for. However, for classification, all feature map cells need to correctly return the scores, because in the case of no object there is still background.

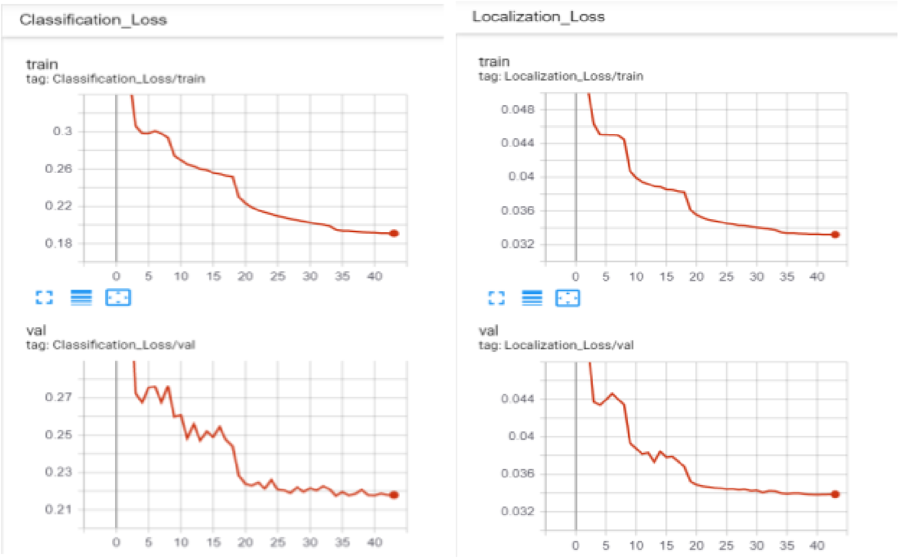


Figure 5: Localization and classification losses on the train and validation sets.

Figure 4.1: Loss function graphs over time

4.3.4 Post-processing

We have to keep in mind that more than one anchor may map to a single ground truth object, and in that case the model may output multiple bounding boxes for a single instance, which is undesirable. Thus, we use the non maximum suppression algorithm to only keep the best matching and highest confidence score bounding box for each object, see 4.2. Two hyperparameters are important when it comes to NMS, the confidence score for which the prediction is considered an object, and the suppression threshold specifying what IoU two bounding boxes have to have to be considered as predicting the same object. Both of these hyperparameters can vary quite a bit, so we tested out multiple values and picked the best performing on the validation set.

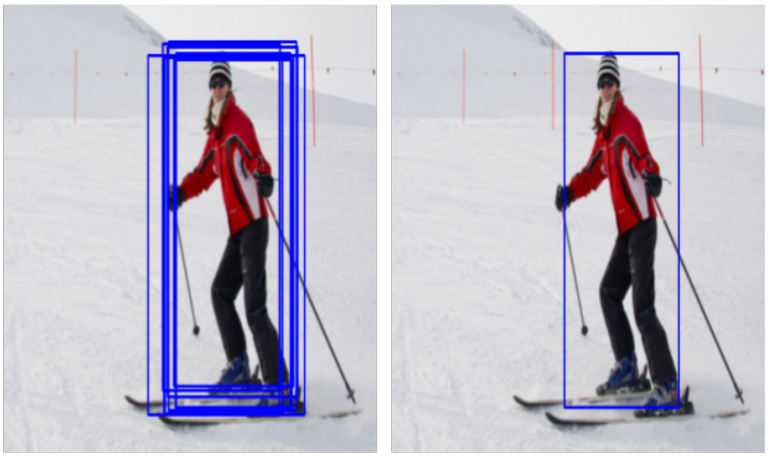


Figure 5: Before and after NMS

Figure 4.2: Before and after Non Maximum Supression

Algorithm 1 NMS Pseudocode

```
procedure NMS($B$,$c$)
    $B_{nms} \leftarrow \emptyset$
    for $b_i \in B$ do
        $discard \leftarrow$ False
        for $b_j \in B$ do
            if same$(b_i, b_j) > \boldsymbol{\lambda_{nms}}$ then
                if score$(c, b_j) >$ score$(c, b_i)$ then
                    $discard \leftarrow$ True
        if not $discard$ then
            $B_{nms} \leftarrow B_{nms} \cup b_i$
    return $B_{nms}$
```

Taken from https://towardsdatascience.com

Chapter 5

Application (numerical validation)

5.1 Methodology

- We trained our Neural Network on a subset of COCO 2017 train-val split. We kept only the annotations which meets the following criteria: it is a person or a car and it is considered a Medium or Large object (it has area $>$ 32*32). From the original data set we removed all the images which do not contain any annotations with the previous property.

- We evaluated our algorithm on COCO mAP[1] (also known as AP@[.5:.95]) metric which aims to balance the algorithm's precision, recall and the accuracy of the bounding boxes. This metric is computed as the mean of the Average Precision at multiple IoU[2] thresholds (e.g. for a prediction to be considered at AP75 it has to overlap a ground truth with an IoU that's at least 0.75). Now the predictions are sorted by their decreasing confidences and they are matched in that order with the (yet) unmatched ground truth bounding box that has the highest IoU. The unmatched predictions are considered False Positives and unmatched ground truths are considered False Negatives. Now a Precision vs Recall graph can be plotted and the AP@IoU is the area under the 101-point interpolation.

- This dataset was used because it shows object in multiple common contexts. Even though the images are mostly not from the automotive domain, we believe that a Deep Neural Network can learn general features about humans and cars so that it will generalise well in another context.

[1] mAP = mean Average Precision

[2] IoU = Intersection over Union

5.2 Data

We use the Microsoft COCO [4] dataset for training. COCO is a large-scale object detection, segmentation, and captioning dataset with 330K images and 1.5 million object instances. In our case, for detection, each training image has associated with it a list of bounding boxes and class IDs for the object instances it contains.

We use data from the Joint Attention in Autonomous Driving (JAAD) dataset for inference [7] [8]. It is composed of 346 high-resolution video clips, most being 5-10 seconds long and extracted from approximately 240 hours of driving videos filmed in several locations in North America and Eastern Europe. These videos were further split in 88000 frames.

5.3 Results

We currently achieve 24.6% mAP and 56% AP50 on the person and car classes on the validation set, on medium and large instances. Although per category results are not usually reported, upon consulting the COCO leaderboard, models with similar mAP as SSDLite (22.1% reported in the paper) achieve 56% AP50 on people and 34% on cars respectively on the COCO test set. However, this also includes small objects, which considerably decrease the overall score. So we currently fall short of reproducing the results reported in the paper.

5.4 Discussion

- Training on a subset of classes only, is it really efficient?

  Upside: the model can focus only on targeted classes, allowing possibly for even a lighter model to achieve good performance, but we do not have enough evidence to confirm this.

  Downside: class imbalance is much worse, and also an entire image is processed for a much lower number of positive examples, actually slowing down learning.

- Low accuracy issues.

  We tried many different setups and hyperparameter adjustments aiming to boost our performance. In particular, we found that adjusting anchors to better represent the target classes helps increase accuracy, while even light data augmentation techniques help the model generalize better. We also tried training with hard negative mining as suggested in [5], but it did not yield better results.

Chapter 6

Conclusion and future work

Currently, the main strength of our approach is speed. The model is extremely lightweight only requiring about 3M parameters for two classes. On the other hand, accuracy is not yet where it needs to be. On one side, picking the very best hyperparameters is out of our reach given our limited resources. However, our big mistake was directly implementing SSDLite and training with focal loss, instead of starting with the standard SSD model and gradually making changes towards our final goal: switching backbones (VGG to MobileNetV2), changing regular convolutions in the SSD head with depth wise separable ones, and finally adding focal loss.

To remedy this, we identify the following steps to follow in the future:

- Implementing everything described above
- Much heavier data augmentation
- Resize images without changing the aspect ratio too much

Bibliography

[1] 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.

[2] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[3] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[4] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[6] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. *CoRR*, abs/1711.07240, 2017.

[7] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. Are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–213, 2017.

[8] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. Itâs not all about size: On the role of data properties in pedestrian detection. In *ECCVW*, 2018.

[9] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[11] Liliang Zhang, Liang Lin, Xiaodan Liang, and Kaiming He. Is faster r-cnn doing well for pedestrian detection? *Lecture Notes in Computer Science*, page 443â457, 2016.