Using Artificial Occlusion to Customize YOLO for Facial Occlusion Detection

Irfan Dwiki Bhaswara[1], Dan Zeng[2], Raymond Veldhuis[3], Asif Khan[4], Tauseef Ali[5]
[1,2,3]*Faculty of EEMCS*, *University of Twente*, Enschede, The Netherlands
[4,5]*20face*, Enschede, The Netherlands
[1]irfandwikibhaswara@student.utwente.nl
[2,3]{d.zeng, r.n.j.veldhuis}@utwente.nl
[4,5]{a.khan, t.ali}@20face.com

***Abstract*—Automatic detection of facial occlusion is one of the most challenging problems in a face recognition system. Once detected, a face recognition system can process the occluded faces differently to increase the accuracy of face recognition system. In this research, we are interested in knowing not only the occluded or occlusion-free faces but also the location and type of the occlusion. We treat facial occlusion detection as a special case of object detection and propose a solution that is based on deep neural network architecture and is inspired from Tiny YOLO. We generate a dataset of synthetic occlusions from occlusion free images of VGGFace2 for training. We also demonstrate some adjustments on our proposed architecture and provide quantitative evaluations on 90 real occluded images using *log-average miss rate*. Our method achieves a performance that is almost on par with Tiny YOLOv2 and is superior to VGG16 model.**

***Index Terms*—Face recognition; Occlusion Detection; Quality Check; YOLO**

I. Introduction

Accurate and efficient object detection in a wild scene is a fundamental computer vision problem that has been extensively studied for several decades. Most of the research presents solutions for the problem of detecting a general object such as, person, car or cat. Variations in lighting conditions and pose of the object make the problem challenging [1]. The detection of one or more specific objects in a specified background is a new problem that needs investigation. An example of such a problem is detecting objects like cup, mug, scarf and sunglasses when they are occluding a human face. Therefore, it is required to investigate on facial occlusion detection.

In this paper, we present three contributions for facial occlusion detection. The first contribution is the reduction in the input size from 416 to 224 of the regular Tiny YOLOv2. This contribution also includes decreasing the size of layer 7 and layer 8 from Tiny YOLOv2. We name this new proposed architecture as Custom Tiny Yolo (CTY). The second contribution is a method for artificially generating occluded human faces. The third contribution is the generation of a dataset having artificially occluded faces. We use this dataset to train and test our proposed CTY. The focus of the research is to detect the location (nose, mouth or eye) and type of the object (cup, ice cream or sunglasses) that is occluding a human face.

The rest of the paper is organized as follows. Section II describes the related work on facial occlusion detection. Section III explains the contributions of this research. Section IV describes the simulation setup and results of the research. Finally, Section V concludes the research work.

II. Related Work

The work in this paper has inspired by two aspects: facial occlusion detection and deep learning architectures for that purpose. Zhao [2] has proposed occlusion detection for face recognition on pixel level. He used Sparse Representation based Classification (SRC) to get the residual of each class. From this residual, he examined a pixel-level to estimate whether there is occluded or not in the image. Another approach for facial occlusion detection comes from Min et.al [3]. In their research, the authors proposed scarf occlusion detection by using Gabor wavelets, PCA, and SVM. The Gabor wavelet is used to extract the feature from the image, then the size of the features is reduced by PCA. Subsequently, SVM is used for classification. Chen et.al [4] propose a method to classify sunglasses and scarf occlusion using LBP [5].

On the other hand, all of those research mentioned above still refer to traditional object detection because it needs to find the best feature extractor and to tune hyper parameters of the classifier. In the last few years, Deep Neural Networks (DNN) have changed the paradigms of object detection. Basically, the idea comes from simple neural network modelling but introduces large-scale datasets for training and also introduces deep neural networks, which makes it easy to learn complex features [6]. Recently, DNNs for general purpose object detection are divided into two categories [7]. First, a DNN model that is based on two-stage detectors. The detector creates a region proposal that extracts the potential features of an object and used it to predict the classes. R-CNN [8], Fast R-CNN [9], Faster R-CNN [10], and Mask R-CNN [11] are some examples of two-stage detectors. Second, a DNN model which only uses one-stage detectors. The aim of one-stage detectors is to get the fast prediction in a real-time situation. It does not consider the selection of potential feature from the image, instead it selects all locations as potential features from the image, and it attempts to distinguish between the object and the background from image [7]. Some examples of one-

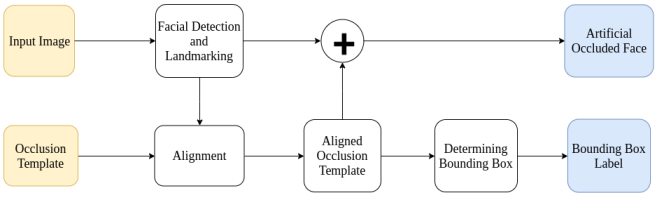

Figure 1: Artificial Occlusion and Automatic Labelling.

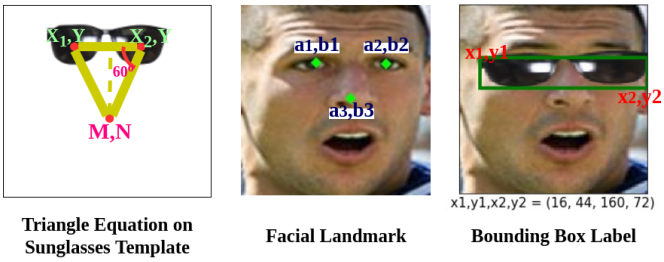

Figure 2: Triangle Equation on Sunglasses Template.

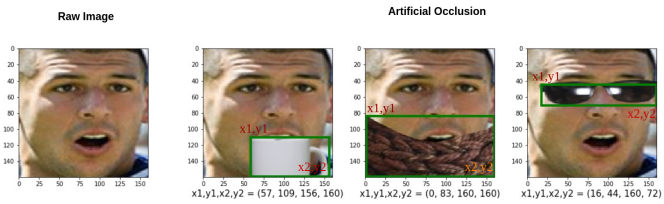

Figure 3: Artificial occlusion generation and labelling.

stage detectors are YOLOv1, YOLOv2, YOLOv3 [12, 13, 14], Single-Shot Multibox Detector (SSD) [15], and RetinaNet [16].

Currently, YOLO is widely used in many research because of its speed and accuracy in detection. A lot of improvement in YOLO also is made to increase the performance. YOLOv2 improves YOLOv1 model by adding an anchor box inspired from SSD model. The anchor box that is presented in YOLOv2 is better than the SSD model because of the use of k-means clustering from training data, which can generate the anchor box value automatically. The YOLOv2 architecture is able to reach 40 FPS. While the smaller version, Tiny YOLOv2, reaches 244 FPS tested on COCO dataset [17]. Another architecture that also refers to Tiny YOLOv2 model was proposed by Pedoeem et.al [18], called YOLO-LITE. The model cuts the layers of Tiny YOLOv2 into 7 layers and change the last 3 layers to smaller dimensions of convolution. YOLO-LITE is meant to be used in non-GPU systems. Although the performance of all YOLO version is satisfactory, but it is only tested on common objects, such as car, person, cat, dog, and so on. None of them is tested in the specific object, for example facial occlusion detection. We are interested to solve this problem by applying YOLO, as facial occlusion detection and to analyse its performance in such scenario.

III. Methodology

In order to train a deep neural network for detecting the location and type of the object that is occluding the face, we need a huge data-set of occluded faces. The unavailability of such a dataset motivated us to create our own dataset. We present a method to generate artificially occluded faces by blending real face images and real object images. We also explain the CTY network as well as its loss function.

A. Occlusion Generation and Labelling

Generation of artificially occluded faces involves three steps. First, the detection of human face and at least three landmarks on the detected human face. Second, the synthetic occlusion template with three landmarks. Third, the alignment of occlusion template over the detected human face. Figure 1 graphically shows this three step method.

We use MTCNN [19] to detect a human face in an input image. In addition to the bounding box, MTCNN also provides five landmarks: two for eyes, one for nose tip and two for mouth corners. Occlusion template is a transparent image of an object that can occlude a human face. Some examples of these objects (object types) are cup, mug, scarf, microphone, ice cream, and hand. We use one template per object type. We manually select two out of three landmarks on the occlusion template. The selection of third landmark is based on using the triangle equation, as shown in Equation (1) and Figure 2. These landmarks depend on the object (template) that is occluding the face. Figure 2 shows an example of sunglasses template with manually selected (X_1, Y) and (X_2, Y). Two manually selected landmarks (X_1, Y) and (X_2, Y) correspond to two eyes and the third landmark (M, N) correspond to nose-tip on the human face.

$$\begin{aligned} M &= \frac{X_1 + X_2}{2} \\ N &= Y + \frac{X_2 - X_1}{2} \tan(60°) \end{aligned} \tag{1}$$

We apply Affine transform that uses the three landmarks from face and three landmarks from the occlusion template to align the occlusion template over the human face. The Affine transform can slightly change the shape of the template to adjust the location and appearance of the template on the face. The method then blends the human face with the result of Affine transform to generate the final result of occluded face. This method can generate occluded faces that look like realistic occluded faces.

To create the annotation label, we detect the shape of aligned occlusion template. From the detection, we can draw bounding box around the shape and extract the coordinate location of the bounding box. Figure 3 shows an example result of artificial occlusion where the green box indicate the bounding box location of our occlusion template.

B. Custom Tiny YOLO

Custom Tiny YOLO (CTY) has a design that looks similar to regular Tiny YOLOv2. Figure 4 shows the CTY architecture. The CTY consists of 9 convolutional layers with

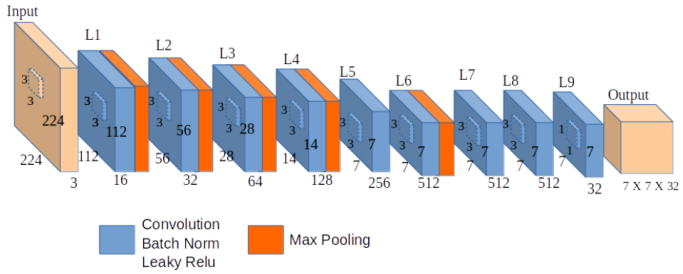

Figure 4: Our proposed architecture (Custom Tiny YOLO).

same padding. Each of the convolution layers is followed by batch normalization [20] and leaky ReLU [21] except layer 9 (L9). To reduce the size of convolution for each layer, we use max pooling with the size of 2×2 with valid padding, except L6 which uses same padding to preserve its output. The max pooling also selects the important features from the convolutional layer.

The input in CTY is reduced from the original Tiny YOLOv2. This reduction is intended to fit the input network with the size of face image, that normally has small resolution. So, it is required to shrink the input from 416 into 224. Hence, the network will be able to capture the most prominent occlusion features that occur on a face.

We change the size of L7 and L8 from the original Tiny YOLOv2, that is 1024, to 512. The idea here is to make our network computationally efficient while maintaining the same performance as Tiny YOLOv2. At L9, we use filter size of 32. This filter number is based on the number of anchor box that we used (4 anchor boxes) and the number of classes (3 classes). This L9 will create $S \times S$ grid which contains B boundary boxes. From this, we can calculate the loss function of YOLO corresponding to the grid cell i and anchor box j. The total loss are the summation of three losses: classification loss, localization loss, and confidence loss.

The classification loss L_{class} is calculated by measuring the difference between probability $p_{i,j}(c)$ of true class distribution and probability $\hat{p_{i,j}}(c)$ of estimated class distribution. Or, simply we apply cross-entropy on $p_{i,j}(c)$ and $\hat{p_{i,j}}(c)$. Equation 2 shows how to calculate the loss. The variable $1_{i,j}^{obj}$ has a value 1 if there is an object in cell i and box j, else it is 0. The variable λ_{class} is a scalar to weight the loss function.

$$L_{class} = -\lambda_{class} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{obj} \sum_{c \in classes} p_{i,j}(c) \log \hat{p_{i,j}}(c) \tag{2}$$

The localization loss L_{loc} calculates the error between the predicted bounding box location and the ground truth of the box. The loss is only calculated for the box that detects an object and based on mean-squared error. Equation 3 points this loss function, where $(x_i; y_i)$ are the ground truth center of the box, $(\hat{x_i}; \hat{y_i})$ are the prediction center of the box, and λ_{coord} is a scalar to weight the loss function. Moreover, $(w_i; h_i)$ denote the width and height of the ground truth box and $(\hat{w_i}; \hat{h_i})$ are the prediction of the width and height of the box. The square root on the width and height is to penalize small bounding box, so it makes sure smaller objects get better prediction than bigger objects [12]. The value of $1_{i,j}^{obj}$ is same with the previous explanation.

$$\begin{aligned} L_{loc} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_{i,j} - \hat{x_{i,j}})^2 + (y_{i,j} - \hat{y_{i,j}})^2] \\ + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_{i,j}} - \sqrt{\hat{w_{i,j}}})^2 + (\sqrt{h_{i,j}} - \sqrt{\hat{h_{i,j}}})^2] \end{aligned} \tag{3}$$

To make sure if there is an object appears inside the cell and box, the confidence loss L_{conf} is applied here. This loss is calculated using mean-squared error between the IOU of prediction and ground truth $C_{i,j}$ with the prediction confidence score of a box $\hat{C_{i,j}}$. The equation for this loss is written in Equation 4 with λ_{obj} and λ_{noobj} as a constant number. The variable of 1_{ij}^{noobj} is to penalize the network if there is no object in the cell and box with a value of 1, otherwise is 0.

$$\begin{aligned} L_{conf} = \lambda_{obj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_{i,j} - \hat{C_{i,j}})^2 \\ + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_{i,j} - \hat{C_{i,j}})^2 \end{aligned} \tag{4}$$

IV. Results and Discussion

All training and testing were done using Ubuntu 16.04 with Intel i7 CPU and Nvidia GTX 1050 graphic card. The frameworks used are Keras [22], Tensorflow [23], and OpenCV [24]. The experiments conducted are finding the optimal threshold and comparison between each models.

A. Dataset

We used occlusion-free images from VGGFace2 dataset [25] to create the synthetic images. This dataset contain 3.31 million of images from 9131 subjects. For this research purpose, we only select 2250 occlusion-free images and we focus on 3 occlusion types: sunglasses, mug, and scarf. Those samples are divided into training (1000 images for sunglasses, 1000 images for mug, 2000 images for scarf) and validation (250 images for sunglasses, 250 images for mug, 250 images for scarf). In addition to this, we collect 90 real occluded images from the internet for setting a threshold parameter for our architecture. We also test CTY with another 90 different images from the internet to get the performance of our model as well as for a comparison with other models. Figure 5 shows some examples of real occluded image.

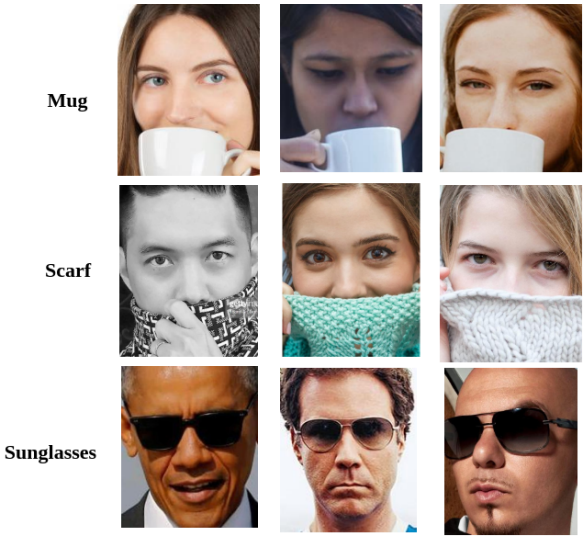

Figure 5: Some real occlusion images for finding threshold and testing the performance.

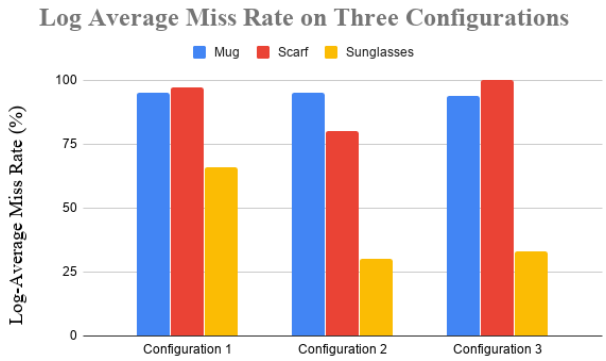

Figure 6: *Log-average miss rate* on three configurations of CTY on real occluded images.

B. Exploration on Custom Tiny Yolo

Some further research of CTY need to discuss. Those discussions are related to the proposed model threshold. The threshold is to adjust our proposed method so it can detect all facial occlusion with low false positive (FP) and false negative (FN). Our proposed method has a threshold value of 0.1. This threshold value is found by adjusting the threshold for the confidence score based on the FP and FN of the detection. Table I shows the FP and FN of each threshold obtained from real occlusion images. From the table, increasing the threshold value will decrease the FP, yet it will increase the FN. Choosing the threshold value where it has low FP and FN but high TP for all occlusion is needed to find the optimal value. The table shows threshold of 0.1 is the optimal number for our proposed architecture since at threshold 0.11, the true positive of mug starts to decreasing again .

Another study is related to the effect of changing the input layer and the last layer from Tiny YOLOv2. To get the understanding of CTY architecture, we also make other trial configurations by changing the middle layer and last layer of CTY network. Table II shows the configurations. The configurations are tested on real occlusion images and we measure the performance using *log-average miss rate* [26]. The *log-average miss rate* is calculated by averaging the miss rate at every 9 false positive per image (FPPI) points evenly spaced in log-space from 10^{-2} to 10^{0}. The result is shown in Figure 6. Interpreting the result, reducing the middle layer as in Configuration 1, will make the model suffers with large miss rate. This is because some important features from previous layer is not well extracted. While keeping the middle filter as same as Tiny YOLOv2 will preserve its feature during fine-tuned as in Configuration 2 and 3. In addition to this, changing the L7 and L8 configuration also has an effect in the performance. We can see that by maintaining the L7 and L8 with small filter size will slightly increase the performance on the model. This is also same with increasing the filter size in L8 into 1024 as in Configuration 3, which result in good performance.

C. Result on Real Occlusion and Occlusion Free Images

The aim in this section is to see the performance of our proposed architecture with the existing model: Tiny YOLOv2 and VGG16 [27] on real occluded images. The existing model is re-trained with the dataset that mentioned in section **Dataset** and tested on real occluded images. The result is then compared using *log-average miss rate* as seen in Figure 7. The figure shows that our proposed model is almost on par with the performance of Tiny YOLOv2. The result of Tiny YOLOv2 is slightly better because Tiny YOLOv2 preserves layer L7 and L8 into 1024 which make the model capture all most important feature at the end. On the other hand, VGG16 is not suitable for facial occlusion detection although it has same input size with our proposed model. The reason is because the convolutional layer of VGG16 is to deep for our small training data, hence it makes the performance worst. In Figure 7, all of the model suffer in detection of mug and scarf. This is caused by the variation style of the mug and scarf, whereas our occlusion template for that is only limited.

Beside testing on the real occlusion image, we want to see if our proposed model is tested on occlusion free image. The result is shown at Table III. Our proposed model is able to detect 108 images from 110 images. False positive that occurs on scarf is caused by illumination in the image, while on sunglasses is due to the person wears eye glasses as depicted in Figure 8.

V. Conclusion

In this paper, we have presented our proposed model for detecting facial occlusion of sunglasses, mug, and scarf by customizing Tiny YOLO, based on artificial occlusion training data. The results show that by preserving the model's layer similar to Tiny YOLOv2, it can increase the performance and the detection accuracy. In addition, keeping the last layer of the

Table I: Effect on varying threshold parameter for all facial occlusions

Threshold	Mug (30)			Scarf (30)			Sunglasses (30)		
	True Positive	False Positive	False Negative	True Positive	False Positive	False Negative	True Positive	False Positive	False Negative
0.01	26	5	4	27	53	3	28	14	2
0.02	26	2	4	27	37	3	28	4	2
0.03	26	2	4	27	33	3	28	1	2
0.04	26	2	4	27	32	3	28	1	2
0.05	26	2	4	27	29	3	28	1	2
0.06	26	2	4	27	29	3	28	1	2
0.07	25	2	5	27	27	3	28	1	2
0.08	25	2	5	27	27	3	28	1	2
0.09	25	1	5	27	26	3	28	1	2
0.1	**25**	**1**	**5**	**27**	**24**	**3**	**28**	**1**	**2**
0.11	23	1	7	27	24	3	28	1	2
0.12	23	1	7	27	23	3	28	1	2

Table II: Some configurations of our proposed architecture

Model	Depth	Description
Configuration 1	8	We change L5 of our model into 128 and delete L8
Configuration 2	9	This is our proposed network
Configuration 3	9	We move the max pooling in L6 into L7 and change L8 into 1024

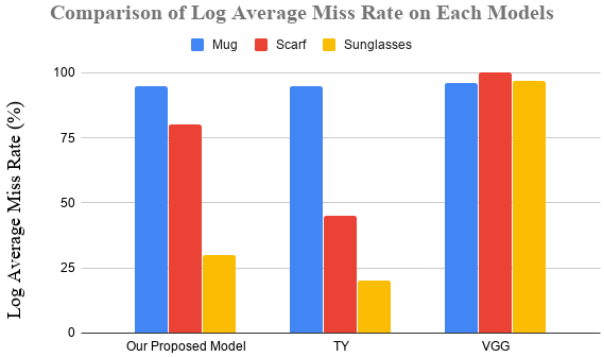

Figure 7: *Log-average miss rate* on our proposed model, Tiny YOLOv2, and VGG16 on real occluded images.

model with high filter size will also escalate the performance. Further, the experiments indicate that mug and scarf occlusion is the most difficult occlusion to detect than sunglasses. This is because the lack of occlusion template on the mug and scarf as well as its inconsistency on the shape of scarf. As a future work, adding some scarf and mug variations may increase the performance of the model. In addition, another refinement could be implemented by updating YOLOv2 version into YOLOv3, where it has an improvement on predicting across multiple scales, which potentially help to reduce the false detection in the system.

VI. Acknowledgement

This research was done under an internship task on 20Face Company. A special thanks to all employees in 20Face Company for their help and support.

Table III: Testing occlusion-free images on our proposed model, Tiny YOLOv2, and VGG16

Model	Sunglasses	Mug	Scarf	Correct
Tiny YOLOv2	2	-	4	104
VGG16	1	-	59	50
Our Proposed Model	1	-	1	108

FP as Scarf

FP as Sunglasses

Figure 8: False positive on occlusion-free images tested on our proposed network.

References

[1] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object Detection with Deep Learning: A Review. *arXiv e-prints*, page arXiv:1807.05511, Jul 2018.

[2] Shuhuan Zhao. Pixel-level occlusion detection based on sparse representation for face recognition. *Optik*, 168, 05 2018.

[3] Rui Min, Angela Angelo, and Jean-Luc Dugelay. Efficient scarf detection prior to face recognition. *European Signal Processing Conference*, 08 2010.

[4] Zhaohua Chen, Tingrong Xu, and Zhiyuan Han. Occluded face recognition based on the improved svm and block weighted lbp. 10 2011.

[5] Sheryl Brahnam, Lakhmi C. Jain, Loris Nanni, and Alessandra Lumini. *Local Binary Patterns: New Variants and Applications*. Springer Publishing Company, Incorporated, 2013.

[6] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13,

pages 2553–2561, USA, 2013. Curran Associates Inc.

[7] Xiongwei Wu, Doyen Sahoo, and Steven C. H. Hoi. Recent Advances in Deep Learning for Object Detection. *arXiv e-prints*, page arXiv:1908.03673, Aug 2019.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv e-prints*, page arXiv:1311.2524, Nov 2013.

[9] Ross Girshick. Fast r-cnn. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1440–1448, Washington, DC, USA, 2015. IEEE Computer Society.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 91–99, Cambridge, MA, USA, 2015. MIT Press.

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Oct 2017.

[12] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *arXiv e-prints*, page arXiv:1506.02640, Jun 2015.

[13] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. *arXiv e-prints*, page arXiv:1612.08242, Dec 2016.

[14] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv e-prints*, page arXiv:1804.02767, Apr 2018.

[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. *arXiv e-prints*, page arXiv:1512.02325, Dec 2015.

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. *arXiv e-prints*, page arXiv:1708.02002, Aug 2017.

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *arXiv e-prints*, page arXiv:1405.0312, May 2014.

[18] Jonathan Pedoeem and Rachel Huang. YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers. *arXiv e-prints*, page arXiv:1811.05588, Nov 2018.

[19] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, Oct 2016.

[20] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv e-prints*, page arXiv:1502.03167, Feb 2015.

[21] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv e-prints*, page arXiv:1505.00853, May 2015.

[22] François Chollet et al. Keras. https://keras.io, 2015.

[23] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[24] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[25] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *International Conference on Automatic Face and Gesture Recognition*, 2018.

[26] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34:743–61, 07 2011.

[27] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv e-prints*, page arXiv:1409.1556, Sep 2014.