INTERNATIONAL JOURNAL OF TECHNO-ENGINEERING
IJTE

ISSN: 2057-5688

Human Detection and Tracking for Autonomous Human-following Quadcopter

K SUNITHA [1], Dr. P VENKATA NAGANJANEYULU[2]

[1]Assistant professor, [2] professor & Principal
Department of Electronics and Communication Engineering

ECE Department, Sri *Mittapalli College* of *Engineering*, *Guntur*, *Andhra Pradesh*-522233

Abstract—In this project, a quadcopter which can autonomously detect and track a person using Deep Learning algorithm and correlation filter tracking technique on Raspberry Pi is presented. There are two major tasks for autonomous detection and tracking procedure executed by the quadcopter: First, a Deep Learning object detection CNN-based model named MobileNetv2-SSDLite was applied to detect a person. This model is proved to be efficient and fast for embedded system and mobile devices. The MobileNetv2-SSDLite model was originally trained on Common object in context dataset (COCO), we kept the weights in the Feature extractor layers (MobileNetsv2) and fine-tuned the detection layers (SSDLite). The purpose of this re-training task was to specialize the model on human detection and the evaluation after training were 98.6% for mAP@[0.5]IoU and 93.6% for mAP@[0.75]IoU. We used a vision-based tracking method for human tracking task named MOSSE to work along with the detection task. This technique uses the bounding box's coordinates collected from the detection task as the initial learning sample and consecutively updated new person's coordinate by online learning method. MOSSE tracking algorithm can also help with small occlusion due to the online-learning ability. Usually, running a Deep learning model like Object detection costs a lot of computational power, especially for an embedded computer like a Raspberry Pi. However, our work keeps the processing speed of the whole system fast enough for real-time implementation by combining an expensive Deep learning model with an inexpensive image-processing based tracking technique. Our proposed method can achieve 3 to 4 FPS on Raspberry Pi which is faster than 0.63 FPS of the detection algorithm. The mentioned results in this article were carried out by testing in a real-

time environment with a self – developed quadcopter model.

Keywords—autonomous quadcopter, deep learning, object detection, object tracking, embedded system.

I.INTRODUCTION

Object detection and tracking are classic problems in many computer vision applications. Many techniques have been developed for performing object detection and tracking without direct human intervention. Previous approaches are usually based on object's appearance [1] [2] [3], newer methods which proved to be very efficient like using Haar Cascade, Histogram of Oriented Gradient with Support Vector Machine [4] and especially Convolution Neural Networks (CNN). Methods like Faster R-CNN [5], SDD [6], YOLO [7] are the state-of-the-art deep learning CNN-based object detection algorithms. Quadcopter has been one of the highest developed technology recently. Thanks to the recent development of flight control algorithms, image processing power and especially machine learning, quadcopters now have the chance to be autonomous and adopted to provide services in a wide range of application scenario. Vision-assisted quadcopters play a vital role in military operations, urban surveillance or agriculture management. Object detection and tracking from quadcopters are important and interesting fields within many different applications of aerial vehicles and are required to have high accuracy and efficiency. However, due to the top-down camera angles and real-time limitations, there still isn't much attention in the intelligent vision application for quadcopters. Other problems like the amount of space and the weight of embedded hardware restraint drones to perform powerful intelligent computer algorithm. This project focused on implementing a computer vision algorithm on a quadcopter to detect and track a person using CNN-based model MobileNetv2-SSDLite [8] in conjunction with a correlation filter tracking technique named MOSSE [9]. In term of hardware, it is included a Raspberry Pi 3 Model B quad-core computer, a USB camera, an Arduino Uno R3 and

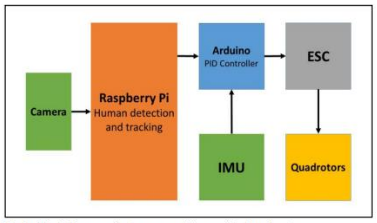

Fig.1. Block diagram of autonomous vision assisted quadcopter

INTERNATIONAL JOURNAL OF TECHNO-ENGINEERING
IJTE

ISSN: 2057-5688

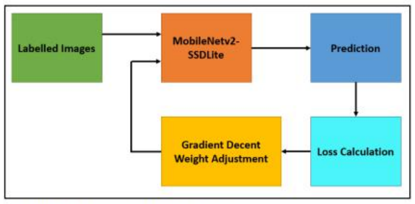

Fig. 2. Training process for human detection

an S500 Quadcopter frame. The human detection and tracking process was carried out using the images collected from a USB camera mounted on the front of the quadcopter. The tracking subject is the first person that the quadcopter detected and based on the reference position of the tracked person on the images, signals will be sent to the main flight controller to navigate the quadcopter. The process is shown in Figure 1. The MobileNetv2-SSDLite [8] model was trained on a 8750-images custom person dataset which specialized for detecting human from above. The training process (Figure 2) took 20 hours on a desktop computer which equipped with a CPU Core-i7, 2.4GHz, Ram 8GB. We also used some images from Penn-Fudan Pedestrian dataset and Indoor Multi-Camera Pedestrian Dataset of TU Graz to bring more varieties to our dataset. The model was then transferred to the Raspberry Pi and used the images from a USB-Camera as input and generates the detection and localization of human. The object detection model then worked along with a tracking technique in order to detect and keep track of a human target. The system is illustrated in Figure 3. Compared to tracking by detection technique, tracking by using detection along with a MOSSE tracker proved to be significantly faster (3.14 FPS compared to 0.63 FPS) and can also help with small occlusions. The rest of the paper proceeds as follow. Section II introduces the design of the hardware platform. Section III discusses the architecture of the system which includes MobileNetv2-SSDLite object detection model and the tracking technique MOSSE. The dataset, training and evaluating process is presented in Section IV. Section V concludes the paper and discusses future work.

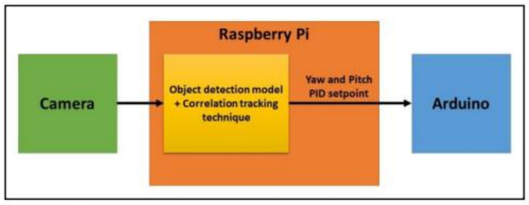
Raspberry Pi
Camera
Object detection model + Correlation tracking technique
Yaw and Pitch PID setpoint
Arduino

Fig. 2. Human detection and tracking on Raspberry Pi

III.LITERATURE SURVEY

A. Raspberry Pi 3 Model B

The Raspberry Pi is a small single-board computer which is equipped with a quad-core 64-bit Broadcom BCM2837 ARM Cortex-A53 SoC processor running at 1.2 GHz, 1 GB of RAM. To achieve high real-time performance, a few factors from the system need to be considered. Object detection models using convolution neural networks are computationally intensive, so the CPU's temperature rises considerably quick. And if the temperature reaches the threshold, the clock speed will be decreased to half of the maximum. Therefore, it is crucial to apply appropriate cooling solutions to sustain CPU performance.

B. Camera Logitech C270

It is a USB Webcam developed by Logitech and capable of 720p video streaming. The camera uses a high-speed USB 2.0 communication protocol which can provide 640x480 images at 30 frames per second or higher frame rate with lower image resolution.

C. Arduino Uno R3

The Arduino Uno is a single-board microcontroller based on the ATmega328. It has 14 digital input/output pins (6 of those can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator. In this work, the Arduino board was the main controller for the quadcopter. It was used to process the value from the IMUs (gyrometer, accelerometer, and barometer) in order to achieve a stable self-leveling quadcopter. At the same time, the Arduino also received the signal from the Raspberry Pi to control the quadcopter to follow the person

ig. 1. The hardware platform

III.PROPOSED SYSTEM

MobileNet-v2 MobileNet-v2 [8] was introduced as a refinement of its predecessor which proved to be more efficient and powerful. MobileNet-v2 innovated depthwise separable convolutions [10] structure into 3 layers instead of 2 as in MobileNet-v1 [11]. The last 2 layers are the same as in MobileNet-v1, which are a depthwise convolution layer and a pointwise convolution layer. The first layer is newly introduced as an expansion layer. Its purpose is to expand the input layer's data before going through the depthwise convolution layer. MobileNetv2 also introduces two new features: linear bottlenecks, and inverted residual connections. The MobileNets-v2 structure is shown in Figure 5. The bottlenecks block encodes the inputs and outputs while the inner layer encloses the model's ability to transform from lower-level concepts such as pixels to higher level descriptors such as image categories. First, the expansion layer expands the input layer to get more data, because not a lot of information is extracted by applying convolutional filters on a low-dimensional feature map. Next, depthwise separable convolutions are applied to the expanded layer

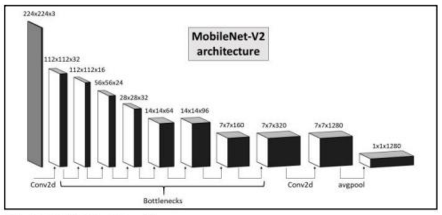

Fig. 4. MobileNet-V2 architecture

instead of normal convolution because of the calculation efficiency which

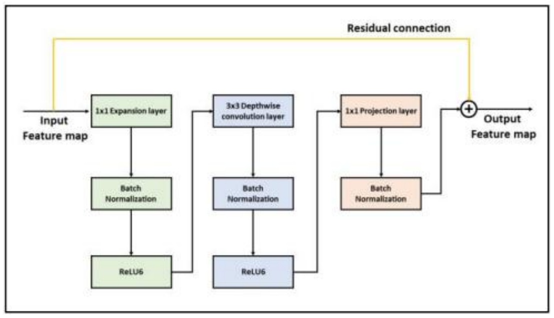

Fig. 3. Bottleneck convolution layer

can decrease the number of connection 8 to 9 times [11]. However, different from MobileNetv1, the pointwise convolution, in this case, will lower the channels from the previous layer to pack the data. This idea is called linear bottlenecks and it keeps the channels of the feature maps fairly small compared to other models and low-dimension layers will reduce the number of computational parameters. The structure is shown in Figure 6. Batch normalization [12] layer is added after each layer of the bottleneck convolution block to normalize the input data for the next layer, therefore it helps speed up the learning process. Residual connection [13] can be understood as the shortcut between bottleneck layers. These connections help with the flow of gradients through the network and will be used when the number of the input channel and the output channel is the same. B. SSDLite Unlike any expensive state-of-the-art object detection algorithm like R-CNN, Fast R-CNN or Faster R-CNN, which generate region proposal areas from the input images that may contain an object then make prediction on those regions and proved to achieve high accuracy but have low processing speed, Single shot detector techniques like YOLO [7] or SSD [6] require just a single pass through the neural network and predict all the bounding boxes in one cycle. SSDLite was introduced along with MobileNet-v2 [8] and was expected to be a friendlier SSD model to mobile devices than the original. All the regular convolutions in SSD prediction layers is replaced with the separable depthwise convolutions and the result is the reduction in computational cost and parameters count. The full architecture of the SSDLite-MobileNetv2 is shown in Figure 7. The SSDLite-MobileNetv2 takes a 300x300x3 dimension image as the input, uses MobileNetv2 to extract feature maps, then detects objects with multiple scales using 6 detection layers. The first detect layer is attached to the expansion layer of layer 15 of the base model. The rest of SSDLite layer is attached to the last feature map of MobileNetv2. The main purpose of the additional layers is to independently detect objects at multiple scales. The first detection layer attached to the expansion layer of the 15 bottleneck layer is expected to

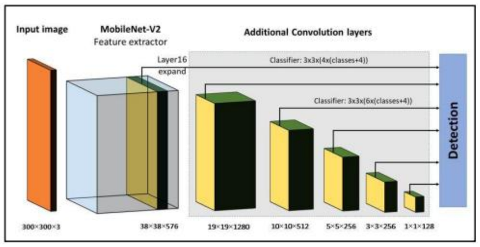

Fig. 5. MobileNetv2-SSDLite

detect the small objects. As the feature maps get smaller due to stride = 2, the resolution is lower and the model has more details about bigger objects. The SSD model starts the predictions based on the Priorboxes (also called default boxes) and use Gradient decent to optimize the model in training phase. For each cell in the detection feature map, 6 default boxes are initialized with 5 different aspect ratio. SSDLite computes both the location and the class score from 6 detection layers using small convolution filters. The 3x3 filters are applied for k prior boxes on each cell of the detection layers and outputs (c+5)k predictions for c classes, one class for background, and four offset value for the predicted bounding box. After prediction, non-max suppression technique is applied to reduce the number of predictions in a frame to the actual number of ground-truth objects. For every bounding boxes which have high IoU value with ground-truth, non-max suppression keeps the boxes with the highest confidence and eliminates lower confidence ones. C. MOSSE correlation tracking technique MOSSE [9] tracker is based on correlation filter technique. The tracker is initialized using a selected bounding box centered on the object in the first frame. The target is tracked by correlating the filter over a search window in the next frame; the location corresponding to the maximum value in the correlation output indicates the new position of the target. The online update is then performed based on that new location. To make the process faster, correlation is computed in the Fourier domain. Correlation task becomes an element-wise multiplication in the Fourier domain based on The Convolution Theorem. First, the 2D Fourier transform of the input image: X = F(x), and of the filter: F = F(f) are computed using Fast Fourier Transform. The correlation has the form: $* = Y\ X\ F$ (1) The symbol indicates element-wise multiplication and the * symbol represents the complex conjugate. Reverse FFT is then applied to convert the output Y back to the spatial domain. However, these methods for creating filters by cropping a region from an image and produce strong peaks for the target are not robust to variations in target appearance and fail on

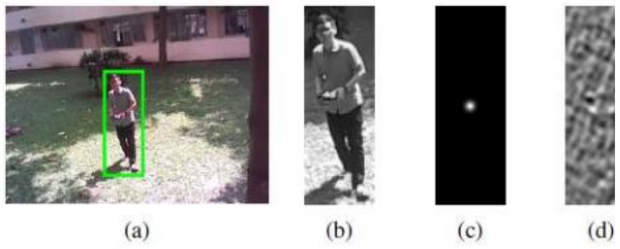
Fig. 6. MOSSE tracker: (a) the camera's image, (b) the cropped image a detection, (c) the desire output, (d) the filter initially calculated

challenging tracking problems. Average of Synthetic Exact Filters (ASEF) [14] and Minimum Output Sum of Squared Error (MOSSE) [9] produce filters that are more robust to appearance changes and are better at discriminating between targets and background. The MOSSE [9] method is originally based on ASEF [14] method but requires fewer training images and uses online training. At first, a set of training images and training output is required. The desired output i y is created to have a 2D Gaussian (2.0= σ) shaped peak centered on the target in training image (Figure 8c). The training task is implemented in the Fourier domain due to the simplicity of the element-wise multiplication of the input and the output.

The relationship between the output and input is showed as: * Y X F i i = (2) MOSSE technique is expected to find a filter F that minimizes the sum square error between the actual output and the desired one. This task takes the form: $\min_{F^*} \sum_i |X_i \odot F^* - Y_i|^2$ In order to find F, we take the derivative of the function with respect to F* and set it equal to zero: $\frac{\partial}{\partial F^*} \sum_i |X_i \odot F^* - Y_i|^2 = 0$ Solve for F*, the expression of MOSSE is found $F^* = \frac{\sum_i Y_i \odot X_i^*}{\sum_i X_i \odot X_i^*}$ As shown in Figure 8, the filter is generated from the cropped image and the desire output. Throughout the tracking task, the tracker has to quickly adapt to appearance change, rotation, pose, various lighting condition. Therefore, the average technique is applied to solve this purpose. The online learning of MOSSE from frame i is computed as: $F^* = \frac{A_i}{B_i} = \frac{\eta Y_i \odot X_i^* + (1-\eta) A_{i-1}}{\eta X_i \odot X_i^* + (1-\eta) B_{i-1}}$ The learning rate η retains data from recent frames and lets the effect of previous frames decay over time. With =η 0.15, the filter to quickly adapt to tracking challenges while still maintaining a robust filter. In the next frame, i 1 + y is calculate using equation (2), the offset values for the bounding box are found by taking the mean value of all the pixels which have the max value from i 1 + y . The tracking filter is updated using

ISSN: 2057-5688

equation (6) with the new bounding box coordinates. Strong occlusion is detected using Peak to Sidelobe Ratio (psr) measurement and it is calculated as: $psr = \frac{y_{peak} - \mu_{sl}}{\sigma_{sl}}$ where peak y is the peak values, slμ and slσ are the mean and standard deviation of the sidelobe. When MOSSE is under good tracking condition, psr ranges between 20.0 to 60.0. If psr drops under 8.0, that indicates the target is occluded or tracking task has failed.

IV.RESULT

Training process The human detection dataset was created using two different sources. 85% of the dataset consisted of self-taken images from a camera and 15% of our dataset was from PennFudan Pedestrian dataset and Indoor Multi-Camera Pedestrian Datasets of TU Graz. All the image from the dataset was handlabeled as shown in Figure 9. Due to the lack of powerful training platform, we used transfer learning method by taking a pre-trained model of Mobilenetv2-SSLite on COCO dataset and trained on our own dataset. The labeled data was stored on a desktop computer, which is equipped with a Core-i7, 2.4GHz, Ram 8GB. The dataset was split into three subsets: a training set (85%) contained 7500 images, a validation set (15%) contained 1250 images. We found that using the original frame resolution 640x480 from the camera was expensive for the Raspberry Pi, so all of the images from the dataset were resized to 320x240 to reduce the computation cost to 4 times. The code was written in

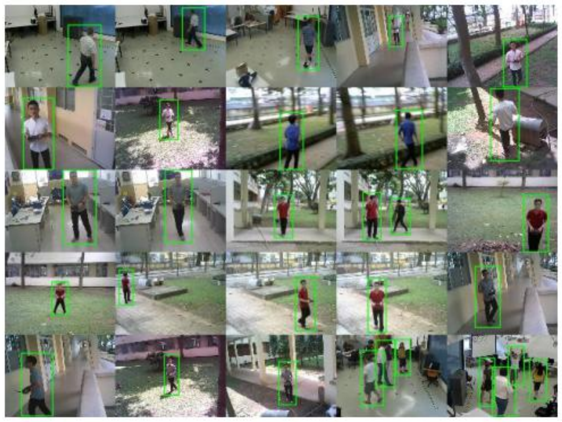
Fig. 8. Labeled detection dataset

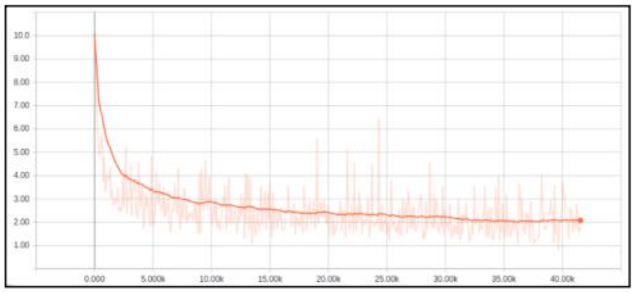

Fig. 7. Training total loss

Tensorflow Object Detection API. We trained the model with batch size of 8 and the learning rate was 0.0003. Weighted_sigmoid and Weighted_smooth_l1 were the techniques used to calculate the loss of classification and localization. The model was trained about 20 hours and the total loss (sum of classification and localization loss) was approximately 2.00 as shown in Figure 10. Mean Average Precision

ISSN: 2057-5688

was used evaluate the model's quality. Mean Average Precision is the mean of every class's Average Precision. AP computes the average precision value for recall value over 0 to 1. Precision is the percentage of correct positive predictions, recall is the percentage of true positive detected among all relevant ground truths. Predictions are considered true positive if the IoU value (Intersection over Union) is higher than a given threshold. IoU is calculated as: $IoU(\mathrm{A},\mathrm{B}) = \frac{A \cap B}{A \cup B}$ AP is defined as the mean precision at a set of eleven equally spaced recall levels [0,0.1, . . . ,1]: $AP = \frac{1}{11} \sum_{r \in [0,0.1,\ldots,1]} \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r})$ where pr( ) is the measured precision at recall r . The model was evaluated on the validation dataset and achieved mAP@[0.5]IoU = 98.6%, mAP@[0.75]IoU = 93.6% mAP@[0.5:0.95]IoU = 77.9%. B. Experiment Once trained on the desktop computer, the model was transferred to the Raspberry Pi to perform the detection task on the quadcopter. We made an assumption in this project that there is only one person to be detected and tracked, the target will be the person who has the highest detection score. 1) Detection: Multiple test results are shown in Figure 11. The number on top of the bounding box is the confident value of the detection. The model performed well on images which are similar or slightly different from the training dataset, but worse on images taken from different environment. The model failed when implemented on very challenging scenarios where the contrast between the human and the surrounding environment is not obvious. 2) Tracking: Tracking algorithm can help with small occlusion because of the ability to learn online. Tracking task was implemented after the target has been successfully detected. Target's position was updated online in every frame along with psr value. If psr was smaller than 8.0, it was indicated that the tracker has failed or encountered high occlusion and object detection will be called in this situation.

V.CONCLUSION

In this paper, a human detection and tracking system running real-time on Raspberry Pi for a vision-assisted quadcopter is presented. This work addresses a software architecture that combines the accurate but slow detection algorithm and the fast but less accurate tracking algorithm to achieve a fast and accurate real-time human detection and tracking quadcopter system. One of the crucial parts of this project is the dataset. Without a dataset which contains a lot of images with considerable diversity, we would not have been able to build a robust model. The encouraging result shows that it is possible to use convolution neural networks along with a correlation tracker to detect and track human objects on a quadcopter platform in real-time. The slowness caused by the complexity of the object detection model is compensated by the lessexpensive tracking algorithm and the whole system is efficiently processed by the low-power embedded platform Raspberry Pi 3. The average processing speed is 3 to 4 FPS and fast enough to perform detection and tracking on-board the quadcopter. After the detection phase, the target's offset position are calculated to send commands to the quadcopter's main controller board. An issue we encountered while testing the algorithm on the quadcopter was the unstable images collected by the camera due to the oscillating characteristic of a flying vehicle platform. Another issue was the camera latency, which was the amount of time before the computer received digital images from the sensor. This latency can sometimes be high, depends on the performance of the Raspberry Pi, which is about 320 milliseconds. This could lead to a decrease of the performance of the tracking phase and therefore causes an unstable flight. There are still many areas we can explore to push this research further and achieve more convincing results. In the future, we continue to improve the object detection model to get higher processing speed without losing much accuracy and also try to improve the performance of the quadcopter platform, especially related to the self-leveling control in order to get a better input for the detection and tracking algorithm.

REFERENCES

[1] A. G. Kendall, N. N. Salvapantula and Karl A. Stol, “On-board object tracking control of a quadcopter with monocular vision,” 2014 International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, FL, USA, 2014.

[2] I. N. Thiang, Dr. LuMaw, and H. M. Tun, “Vision-based object tracking algorithm with ar.drone,” International Journal of Scientific and Technology Research, 2016.

[3] M. A. Alsaedi, B. M. Albaker, H. A. Dawood, H. abd aoun, and Z. kamel tayeh, “Quadcopter based object detection and localization,” Iraqi Journal for Computers and Informatics (IJCI), 2017.

[4] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” 2005 IEEE Computer Society Conference on

Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005.

[5] S. Ren, K. He, R. B. Girshick, and Jian Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," CoRR, vol. abs/1506.01497, 2015.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," CoRR, vol. abs/1512.02325, 2015.

[7] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detector," CoRR, vol. abs/1506.02640, 2015.

[8] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: mobile networks for classification, detection and segmentation," CoRR, vol. abs/1801.04381, 2018.

[9] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, 2010.

[10] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," CoRR, vol. abs/1610.02357, 2016.

[11] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreeto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," CoRR, vol. abs/1704.04861, 2017

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," CoRR, vol. abs/1502.03167, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CoRR, vol. abs/1512.03385, 2015.

[14] D. S. Bolme, B. A. Draper, and J. R. Beveridge, "Average of synthetic exact filters," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, 2009.

[15] M. Everingham, L. V. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," International Journal of Computer Vision, 2010.