Detect people on a device that fits in the palm of your hands

Luuk Willems
University of Twente
PO Box 217, 7500 AE Enschede
the Netherlands
l.j.t.willems@student.utwente.nl

ABSTRACT
This paper is about comparing different object detection models on a Raspberry Pi Zero 2 W using the TensorFlow Lite software. The models are filtered to only give the output of when it detects a person in the frame. Using different scenarios like walking into the room with a closed door, walking into the room with an open door, the object detection models got tested in different events. Each scenario was done once in light conditions and once in dark conditions to see if there is any difference in the models. All of the models that were chosen were models provider by TensorFlow themselves that were all trained with the COCO 2017 data set.

Keywords
Embedded Machine Learning, Raspberry Pi, CNN, TensorFlow lite.

1. INTRODUCTION
Embedded devices are getting used more and more in everyday life, for example for health monitoring, internet of things, home automation, military surveillance, and more [1].

Lately, there is a new trend where machine learning is merged with these embedded devices for an array of applications such as computer vision [2], translation and processing of languages [3], speech recognition [4], and more. Because of this new type of focus for machine learning with embedded devices, there has been a new theme called “Embedded Machine Learning” where machine learning models get to run in resource-constrained environments like a smartwatch [5]. This theme is where the focus of the research will be.

The main problem with Embedded Machine Learning is that the devices used, do not have a lot of resources and so everything has to run very efficiently. To solve this problem in recent years there has been a lot of research on both the algorithm and hardware levels. For both deep learning and classical machine learning algorithms, optimization techniques are being explored like pruning, quantization, reduced precision, hardware acceleration, and more to try to make them run as efficiently as possible on embedded devices [6] [7].

For my research, I will look into CNNs (Convolutional Neural Networks). CNN's are neural networks that are useful for computer vision or object recognition applications [8]. In a CNN the key features in an image get extracted and converted into a complex representation. This is done by using the pooling layer, and subsequently, the fully-connected layers to classify the image and identify the image appropriately. The architecture of the CNN is mainly made up of convolution layers, followed by a few fully connected layers. Convolution layers that perform kernel function, like vector-matrix-multiplications, make CNN's very computation-intensive, but less memory-hungry because of the few fully-connected layers. This makes it that optimizations for CNNs are more computer-centric directions for example by using hardware acceleration, quantization, processor technology, tiling and data reuse, reduced precision, etc [9].

36th Twente Student Conference on IT, Febr. 4nd, 2022, Enschede, The Netherlands.

The research is about comparing different models for person detection to each other and seeing what the difference is between them in terms of speed and accuracy. To make the implementation easier I will be using models that are supported by TensorFlow lite. The results will show how different models compare to each other in different scenarios and which one is the best choice for the intended purpose.

1.1. Research question
The research is done to find out of object detection works on a Raspberry Pi Zero 2 W and how usable it is. For this reason, there are two research questions.

- **RQ1**: Are TensorFlow lite models for object detection usable on a Raspberry Pi Zero 2 W?

When it is clear that this is possible it becomes important, what the difference is between the models. This is the reason for the second research question.

- **RQ2**: How well do different TensorFlow lite models for object detection perform on a Raspberry Pi Zero 2?

2. Review of literature
Person detection
Person detection solutions have been studied a lot in literature and were originally made using handcrafted features, combined with machine learning to be able to generate an abstract representation of the person [10] [11]. These approaches had promising results but are not that suited for challenging conditions. For example with illumination changes or dynamic backgrounds, there can be a high rate of false-positive detections.

CNN's for detecting people in images offer potentially a more robust solution [12] [13] [14]. They quickly pushed the traditional approaches to the background by automatically selecting the most discriminate feature set from a very large set of training data. For this, there are two approaches in literature. Multi-stage approaches work by having separate networks that first generate region proposals and after that classify the objects inside the proposal [15] [16] [17].

Another multi-stage approach that is promising and more recent is Trident-Net, which works by having a build-in scale-specific

feature map [18]. The other approach is single-shot. The single-shot approaches work by solving the detection task through classifying and proposing bounding boxes in a single feed-forward step through the network [13] [14] [12]. Because single-stage approaches usually have a faster and more compact architecture they are often preferred for lightweight embedded hardware solutions.

Embedded hardware optimization

The transition to using person detection on embedded devices has gained a lot of traction in recent years and is an active research topic. An approach that is used commonly on embedded platforms is using more compact models like Tiny-YOLOv3 [19]. These compacter models can run with decent speeds on embedded hardware like a Raspberry Pi, they do often lose some percentage in accuracy compared to their full-size counterparts.

There also exist other approaches like Fast YOLO [20], ShuffleNet [21], or ShuffleNet V2 [22] they optimize the architecture by looking at the platform characteristics, address efficient memory access, and the indirect computation complexity. Recently there have also been some algorithms that use optimized filter solutions to improve the performance on embedded devices, like depth-wise separable convulsions by MobileNets [23] and inverted residuals by MobileNetV2 [24].

3. Methodology

The research was done by measuring the speed and accuracy of the different models on a Raspberry Pi Zero 2 W. The results will provide results for quantitative research. For the research, I used models that are supported by TensorFlow lite. TensorFlow Lite is an open-source deep learning framework that helps developers to run TensorFlow models on IoT and embedded devices such as a Raspberry Pi.

The data was collected by first running the program with the efficientdet_lite0 model. This model was chosen because it is the default model that is provided by TensorFlow in the example program. The program was adapted to make sure that when started it would wait 5 seconds before starting to capture the webcam. This was done to make sure position could be taken at the right place without being detected yet. When the program started capturing it would save all the pictures it took in a separate folder to be used later for the other models. The program would keep going until it collected 15 pictures in total. While the program ran it would collect the time it took to detect everything on the image. This time would only be the time of the detection and not the rest of the program because this is negligible compared to the detection time. When it detected a person it would save this information with the accuracy to an excel file and also save the picture with the bounding boxes to be able to see later what it detected and to see if it is not a false positive.

In the research, only one person was present in the frame and a controlled environment. The results can be different with multiple people in the frame or with real-life effects, like ambient conditions.

To test the other models the pictures collected by the first model were used to make sure all other models have the same pictures to run detection on. This does affect the fps that each model has because each model takes a different amount of time between pictures to run the detection. For this reason, some of the models will be less sufficient for running on a live feed and are more appropriate for other goals, where the pictures do not have to be scanned in real-time but afterwards.

The used equipment:

- Raspberry Pi Zero 2 W
- Logitech C310 webcam
- Micro SD card 16 GB

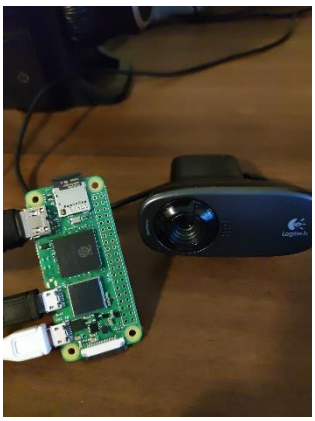
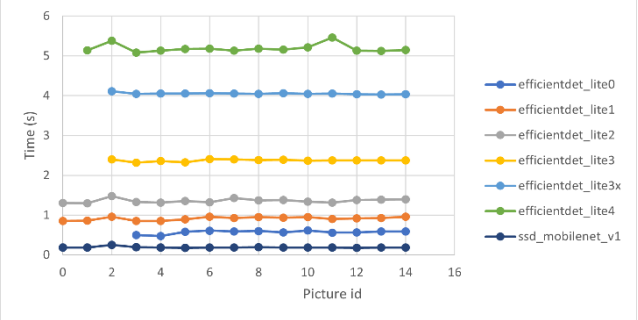


Figure 1: Setup

4. Results

The scenarios that were tested. All scenarios were tested in light conditions and dark conditions. The scenarios were all chosen because they were easy to do and easy to replicate if needed. Because all of the times of the models are almost the same there is only one table provided with these results to reduce redundant information.

Table 1: Time each model took



Backwards

In this scenario, I would walk backwards into the room towards the desk where the webcam was positioned.

Figure 2: Walking backwards into the room.

Table 2: Backwards in dark conditions

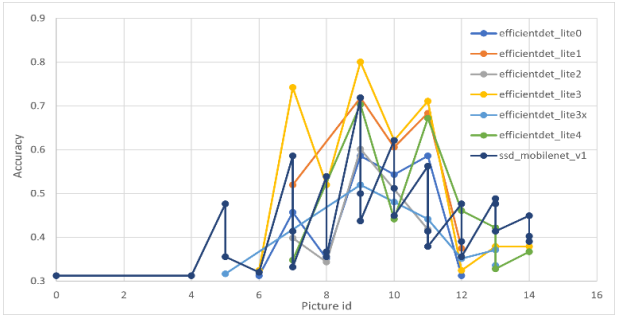
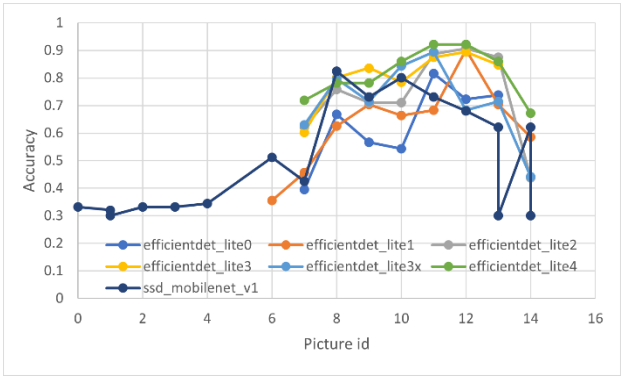


Table 3: Backwards in light conditions



From the results, it becomes clear that the ssd_mobilenet_v1 model most often registers a person. However, a lot of these are false-positives. Efficientdet_lite4 seems to have the highest accuracy in this scenario, however, in the dark, the efficientdet_lite3 model seems to detect the most and with the best accuracy. These detections are also all true-positives. That might be surprising because the efficientdet_lite4 model is built for higher accuracy but with taking more time. Looking at the light conditions it is also visible that the efficientdet_lite0 model has quite a small difference in accuracy while being a lot faster than the efficientdet_lite3 and efficientdet_lite4 models.

Door closed

In this scenario, I would open the door and walk towards the desk.

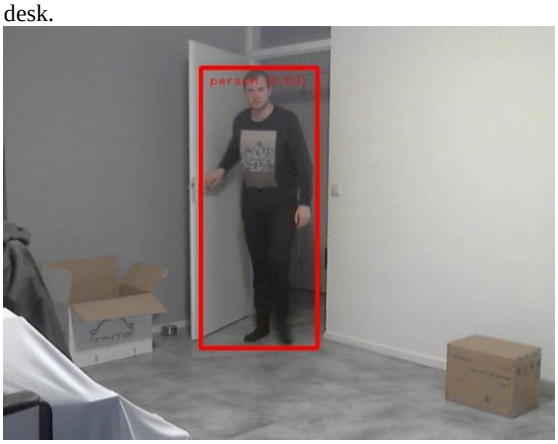


Figure 3: Walking into the room with door first closed

Table 4: Door closed in dark conditions

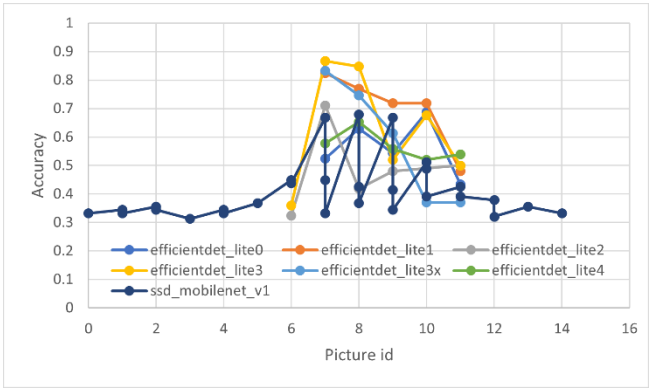
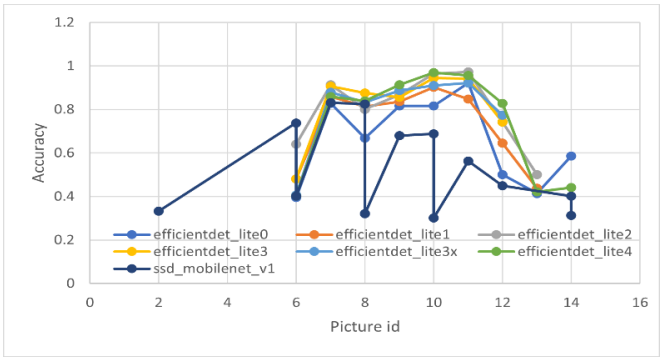


Table 5: Door closed in light conditions



In this scenario, the efficientdet_lite3 and efficientdet_lite4 model are very close in accuracy in light conditions while efficientdet_lite3 again seems to be giving the best performance in dark conditions. Ssd_mobilenet_v2 detects a lot more but often these are false-positives. Ssd_mobilenet_v2 also detect multiple people in the same frame when only one is present. This happens in both light and dark conditions and is something the efficientdet_lite models seem to handle a lot better.

Door open

This scenario is almost the same as the one with the door closed, but in this scenario, the door is already open.

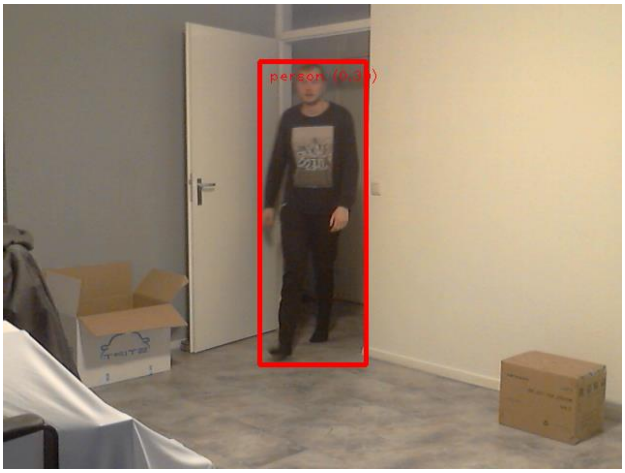


Figure 4: Walking into the room with the door open

Table 6: Door open in dark conditions

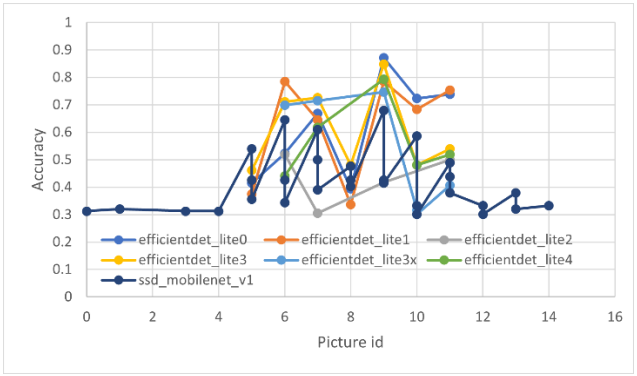
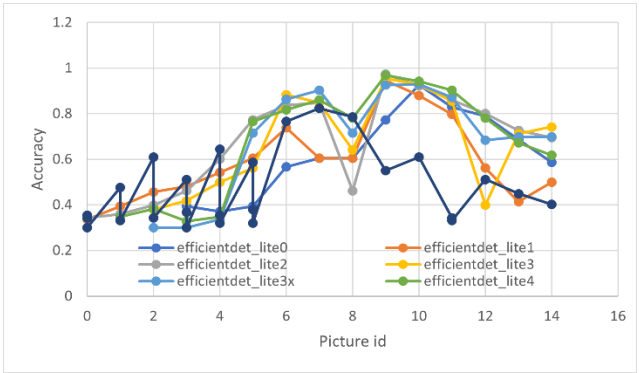


Table 7: Door open in light conditions



The results from this scenario in dark conditions give less of a clear winner in terms of accuracy. It seems that in each frame another model seems to perform better. In light conditions, the results look more like the results of the other scenarios. Ssd_mobilenet_v2 also here detects multiple people in the same frames when only one is present. The efficientdet_lite0 model seems to have trouble with the first part of this scenario and scores a low accuracy compared to the other efficientdet_lite models, from frame 10 it is performing almost the same.

Pickup

In this scenario, I will walk into the frame to a table pick something up and walk out of frame again.

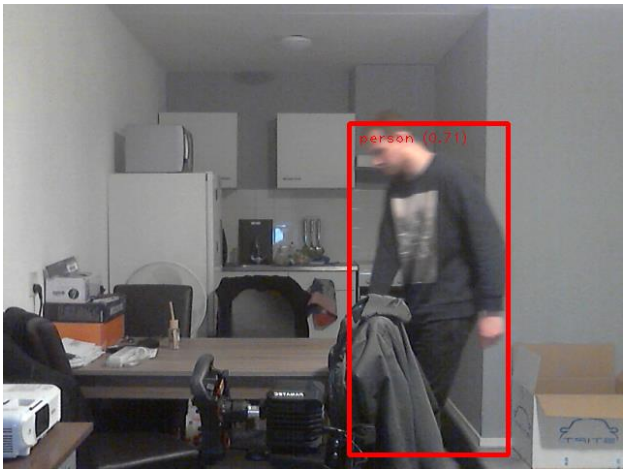


Figure 5: Picking up something from the table

Table 8: Picking up something in dark conditions

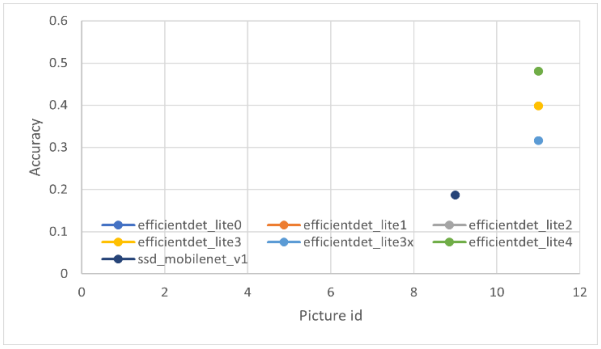
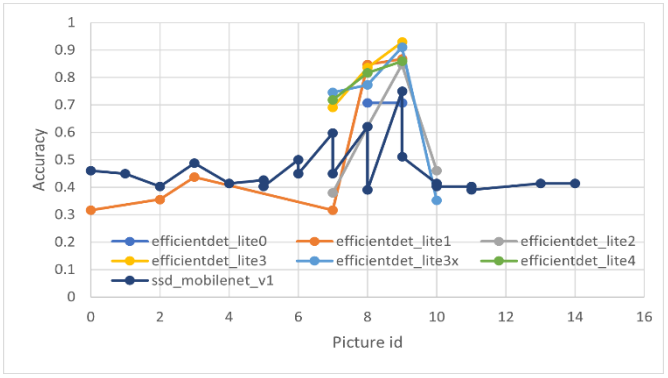


Table 9: Picking up something in light conditions



From the results, it can be concluded that this scenario was very hard in dark conditions and not even all models were able to detect something. It does show how the efficientdet_lite models should compare to each other in terms of accuracy. The ssd_mobilenet_v2 model does detect a person in a frame where no other model does and even though it is low accuracy it is a true-positive. In light conditions, it is noticeable that efficientdet_lite1 detects more than the other efficientdet_lite models however all those low accuracy detections are false-positives like the ssd_mobilenet_v2 model also has. Here the efficientdet_lite3 model seems to be performing best

Run

In this scenario, I would run into the room.

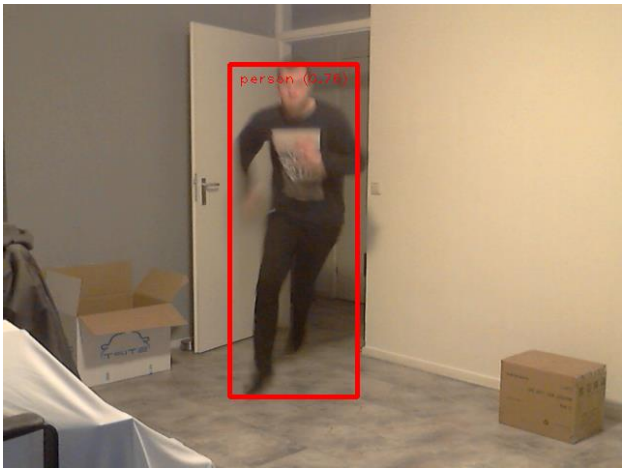


Figure 6: Running into the room

Table 10: Running into the room in dark conditions

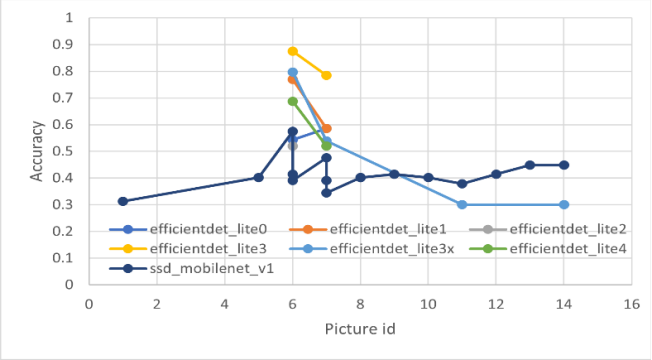
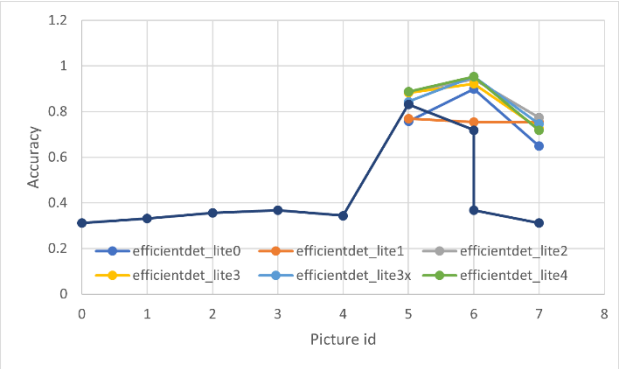


Table 11: Running into the room in light conditions



The dark conditions of this scenario show that the efficientdet_lite3 model was able to achieve the highest accuracy while efficientdet_lite3x had 2 false-positives. Ssd_mobilnet_v2 detected in the frames where there was a person that there were multiple persons while there were not. Also detected it a person in most frames while there was none present. The light conditions show the same for the ssd_mobilenet_v2 but without the multiple detections in the same frame. Here efficientdet_lite4 has the highest accuracy with efficientdet_lite3 being very close.

Table

In this scenario, I would walk into the room and go sit down at the table.

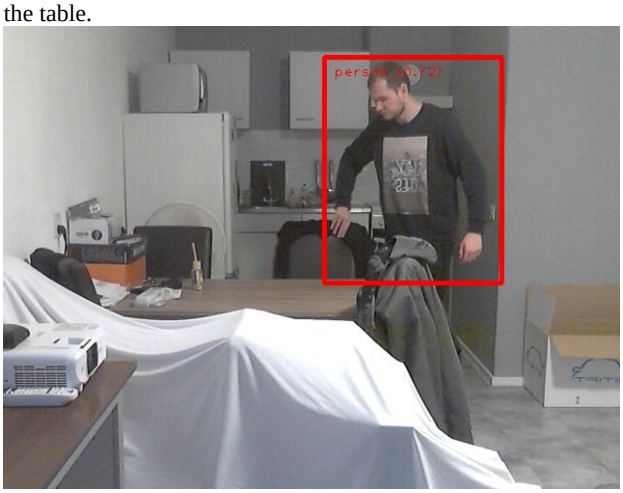


Figure 7: Walking into the room and sitting down

Table 12: Sitting down at the table in dark conditions

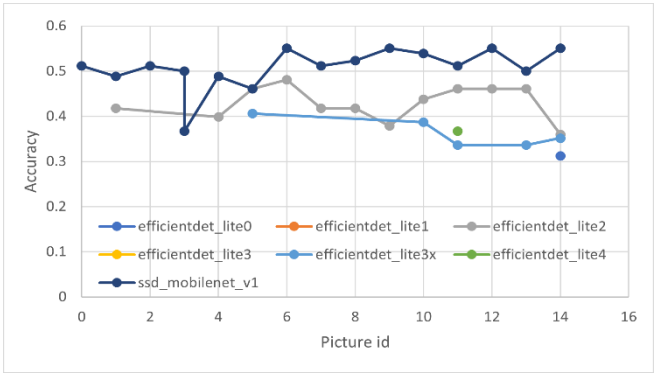
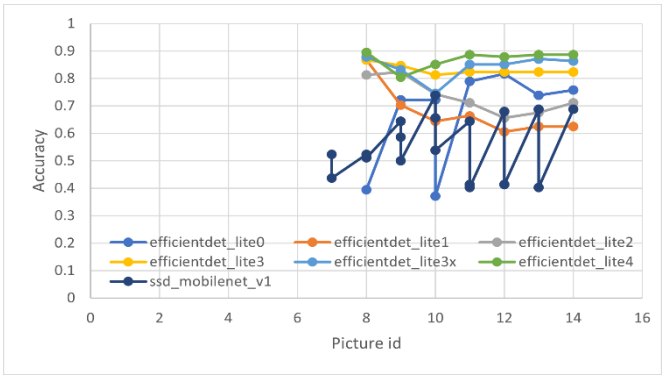


Table 13: Sitting down at the table in light conditions



What is curious about this scenario is that in the dark all the detections that were done where false-positives were often the whole frame would be detected as a person according to the borders. This can be because the lighting in this scenario might have just been a bit too low compared to the others. In light conditions, the results are as can be expected with efficientdet_lite3 and efficientdet_lite4 performing the best. In this scenario, ssd_mobilenet_v2 detects multiple persons in the same frame when there is only one.

Walk away

In this scenario, I would walk away from the webcam out of the room.

Figure 8: Walking out of room

Table 14: Walking out of room in dark conditions

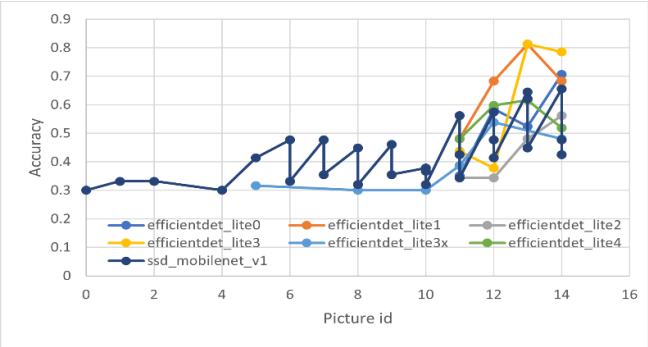
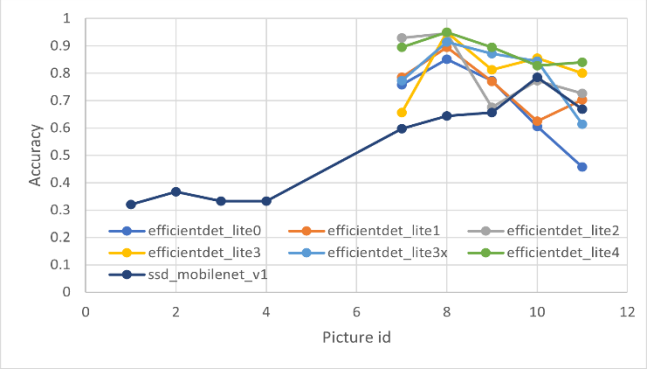


Table 15: Walking out of room in light conditions



The efficientdet_lite1 scenario seems to perform well in this scenario by being able to detect a person sooner than efficientdet_lite3 and with higher accuracy than the efficientdet_lite4 model. The efficientdet_lite3x model detects even sooner but those are all false-positives. Ssd_mobilenet_v2 is performing again like expected by having a lot of false-positives and detecting multiple people in the frame when there is only one or sometimes even when there is none. In light conditions, efficientdet_lite4 seems to be performing the best overall here too has ssd_mobilenet_v2 a lot of false positives.

5. Conclusion

Looking at the results from the research it shows that for speed the ssd_mobilenet_v1 model seems to be the fastest and seems to be detecting the most, however, a lot of these results are false-positives, this can of course be accounted for my only using the results with high accuracy, but you will also have to account for the often detection of multiple objects when there is only one. So for applications where speed is very important, like a live feed for higher fps ssd_mobilenet_v1 seems to be a good choice, but will require tuning and more checking.

For applications where higher accuracy is requested and almost no false-positives, the efficientdet_lite3 and efficientdet_lite4 models seem to get high accuracy in almost all scenarios. Looking at the speed of those three and the difference in accuracy between them efficientdet_lite3 seems to be the best choice, because of the big time advantage compared to the other two.

If the purpose of the application is to use a live webcam feed with getting instant results, so no afterwards computing, speed is the most important. This is because with a high speed you will be able to achieve a higher fps. If we have a low fps there is a possibility that important events get missed between two frames. As mentioned before ssd_mobilnet_v1 is a good choice for this because of the high speed, but because of the number of false positives it gets, efficientdet_lite0 seems to be a good balance between having a good speed and almost no false positives in testing.

The dark conditions often produce results similar to the light conditions. For this reason, this does not have to be taken into account when choosing. In some of the scenarios, only a couple of the models got a result while others got nothing. In these scenarios, it shows that the efficientdet_lite3, efficientdet_lite3x, and efficientdet_lite4 models did get a true positive while the ssd_mobilenet_v1 did get some true-positive results but with a lot of false positives with around the same accuracy. For this reason, the efficientdet_lite model seems to be the better choice.

Overall it can be concluded that all of the models worked well on a Raspberry Pi Zero 2 W depending on the purpose is of your application. It gives a good overview of how far technology has come with how such a small device can be used for so many applications.

6. REFERENCES

[1] W. Wolf, High-Performance Embedded Computing: Architectures, Applications, and Methodologies, San Francisco, CA: Morgan Kaupmann Publishers, 2007.

[2] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 2016.

[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, “Natural Language Processing (Almost) from Scratch,” Journal of Machine Learning Research , Princeton, NJ, 2011.

[4] W. Chan, N. Jaitly, Q. Le and O. Vinyals, “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” IEEE, Shanghai, China, 2016.

[5] J. Dean, “The Deep Learning Revolution and Its Implications for Computer Architecture and Chip Design,” IEEE, San Francisco, CA, USA, 2020.

[6] D. Kim, J. Ahn and S. Yoo, “A novel zero weight/activation-aware hardware architecture of convolutional neural network,” IEEE, Lausanne, Switzerland, 2017.

[7] Y. Deng, “Deep learning on mobile devices: a review,” Proceedings Volume 10993, Mobile Multimedia/Image Processing, Security, and Applications 2019, Baltimore, Maryland, United States, 2019.

[8] K. O'Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” Lancaster University, Lancashire; Aberystwyth University, 2015.

[9] T. S. Ajani, A. L. Imoize and A. A. Atayero, “An Overview of Machine Learning within Embedded and Mobile Devices–Optimizations and Applications,” MDPI, Lagos State, Nigeria; Bochum, Germany; Ogun State, Nigeria, 2021.

[10] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” IEEE, San Diego, CA, USA, 2005.

[11] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” IEEE, Kauai, HI, USA, 2001.

[12] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” IEEE, Honolulu, HI, USA, 2017.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, “SSD: Single Shot MultiBox Detector,” ECCV, Amsterdam, The Netherlands, 2016.

[14] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” IEEE, Las Vegas, NV, USA, 2016.

[15] R. Girshick, “Fast R-CNN,” IEEE, Santiago, Chile, 2015.

[16] R. Girshick, J. Donahue, T. Darrell and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” IEEE, Columbus, OH, USA, 2014.

[17] K. He, G. Gkioxari, P. Dollár and R. Girshick, “Mask R-CNN,” IEEE, Venice, Italy, 2017.

[18] Y. Li, Y. Chen, N. Wang and Z.-X. Zhang, “Scale-Aware Trident Networks for Object Detection,” IEEE, Seoul, Korea (South), 2019.

[19] R. Gopal, S. Kuinthodu, M. Balamurugan and M. Atique, “Tiny Object Detection:Comparative Study using Single Stage CNN Object Detectors,” IEEE, Bangalore, India, 2019.

[20] M. J. Shafiee, B. Chywl, F. Li and A. Wong, “Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video,” arXiv, ON, Canada, 2017.

[21] X. Zhang, X. Zhou, M. Lin and J. Sun, “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices,” IEEE, Salt Lake City, UT, USA, 2018.

[22] N. Ma, X. Zhang, H.-T. Zheng and J. Sun, “ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design,” ECCV, Munich, Germany, 2018.

[23] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” 2017.

[24] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” IEEE, Salt Lake City, UT, USA, 2018.