Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 192 (2021) 2249–2258

Procedia Computer Science

www.elsevier.com/locate/procedia

25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

MobileNet family tailored for Raspberry Pi

Wojciech Glegoła[a], Aleksandra Karpus[a,*], Adam Przybyłek[a]

[a]*Gdańsk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Narutowicza 11/12, 80-233 Gdańsk, Poland*

Abstract

With the advances in systems-on-a-chip technologies, there is a growing demand to deploy intelligent vision systems on low-cost microcomputers. To address this challenge, much of the recent research has focused on reducing the model size and computational complexity of contemporary convolutional neural networks (CNNs). The state-of-the-art lightweight CNN is MobileNetV3. However, it was designed to achieve a good trade-off between accuracy and latency on a single large core of a Google Pixel 1 smartphone. Accordingly, MobileNetV3 is not optimized for platforms with different hardware characteristics and its predecessors may perform better for a given target platform. The aim of this paper is twofold: 1) to analyze the performance of different compact CNNs on Raspberry Pi 4; 2) to manually adapted the most promising models to better utilize the Raspberry Pi 4 hardware. After exploring a number of modifications, we present a new CNN architecture, namely MobileNetV3-Small-Pi, which is 36% faster and slightly more accurate on ImageNet classification compared to the baseline MobileNetV3-Small.

Keywords: Convolutional Neural Network; MobileNets; optimization; Raspberry Pi

1. Introduction

Convolutional neural networks (CNNs) have been constantly progressing over the last decade, leading to significant advancements in a wide variety of computer vision tasks including image classification [9, 32, 29, 13, 50, 2], object detection [12, 14, 15, 28, 46], crowd counting [34, 35], semantic segmentation [16, 37, 52], among others [1]. A general trend of CNN design has been to find deeper and more powerful models to achieve super-human accuracy. Unfortunately, the high accuracy has been achieved at the cost of vastly increased computation time, memory usage, and energy consumption.

In the meantime, with the advancements in systems-on-a-chip [8, 31], there has been increasing demand to run high quality CNNs real-time on resource-constrained platforms in various applications such as robotics, self-driving cars [6, 33], and augmented reality [43, 27, 17]. Thereby, much recent research has focused on designing lightweight architectures, which have fewer parameters and require less computational power while maintaining reasonable accuracy.

* Corresponding author.
E-mail address: alekarpu@pg.edu.pl

1877-0509
10.1016/j.procs.2021.08.238

It has been found that there is a significant redundancy in most of the existing full-scale CNNs [48, 19]. As a result, SqueezeNet [25], ShuffleNet [51], and the MobileNet family [23, 42, 22] have been proposed, which significantly shrink the conventional CNNs without considerably decreasing accuracy. They utilize less expensive operations, such as 1×1 convolution [25], group convolution [51], or depthwise convolution [23].

MobileNetV3 is the state-of-the-art lightweight CNN. Nevertheless, it was designed to achieve a good trade-off between accuracy and latency on a single large core of a Google Pixel 1 smartphone. Accordingly, MobileNetV3 is not optimized for platforms with different hardware characteristics and its predecessors may perform better for a given target platform. In this paper, we manually adjust MobileNet architectures for Raspberry Pi, which is the most well-known single-board computer. Since the release of Raspberry Pi in 2012, researchers around the world have documented its use in a variety of science and engineering projects [5, 10, 18].

The aim of this paper is twofold: 1) to analyze the performance of different compact CNNs from the MobileNet family on Raspberry Pi 4, and 2) propose improvements to the MobileNet family to obtain faster and more accurate architecture for mobile applications on Raspberry Pi 4.

The rest of the paper is organized as follows. Related work is described in Section 2. Section 3 provides information about the dataset used and training settings. An analysis of MobileNets performance on the Raspberry Pi 4 is presented in Section 4. Section 5 introduces our modification of MobileNetV3-Small and briefly reviews our other attempts to optimize MobileNets for Raspberry Pi 4. Obtained results are discussed in Section 6. Conclusions and future work close the paper.

2. Related Work

One of the first parameter-efficient CNN architectures was **SqueezeNet** by Iandola et al. [25]. It has 50 times fewer parameters than AlexNet while preserving AlexNet accuracy. The reduction of parameters was achieved by employing three main strategies: (1) replacing the majority of 3×3 filters with 1×1 filters; (2) reducing the number of input channels to 3×3 filters; and (3) using delayed downsampling so that convolution layers have large activation maps. The basic building block of SqueezeNet consists of a squeeze layer (which has only 1×1 filters) followed by an expand layer that has a mix of 1×1 and 3×3 convolution filters. Besides, when the input and output of the block are of the same dimensions, they are connected by a shortcut.

MobileNetV1 by Howard et al. [23] has shifted the focus from reducing parameters to reducing the number of composite multiply-accumulate operations (Mult-Adds) per image. It exploits *depthwise separable convolutions* [26] as an efficient replacement for traditional convolution layers. A traditional convolution both filters and combines inputs in one step to produce a new feature. A depthwise separable convolution effectively factorizes traditional convolution by separating spatial filtering from the feature generation mechanism. It is defined by two separate layers: a depthwise convolution followed by a pointwise 1×1 convolution. The depthwise convolution applies a single filter per each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs of the depthwise convolution.

Concurrently with Howard et al. [23], Zhang et al. [51] proposed a new architecture called **ShuffleNet**, which integrates a bottleneck design, depthwise separable convolution, group convolution and channel shuffle. The ShuffleNet unit replaces 1×1 convolutions of the bottleneck module [21] by group convolutions and places a channel shuffle operation after the first convolution to mix features among different groups. *Grouped convolution*, which was originally introduced in AlexNet [30], divides its input and output into small groups, and calculates the outputs by using only input channels within the same group. Nevertheless, channel shuffle is difficult to implement efficiently since it requires moving feature maps to another memory space. Accordingly, ShuffleNet is slower than MobileNetV1 even though they have roughly the same number of operations.

CNNs use either pooling layers or convolutional layers with stride larger than one to decrease the spatial dimension of feature maps. In a slow downsampling strategy, downsampling is mainly performed in the later layers of the network, thus more layers have large spatial dimensions. On the contrary, in a fast downsampling strategy, downsampling is mainly performed at the beginning of the network. Qin et al. [38] applied the latter strategy to MobileNetV1 and achieved inference speedup under the same complexity. The proposed **FD-MobileNet** implements the faster downsampling by consecutively applying depthwise separable convolutions with large strides at the beginning of the network.

Sinha et al. proposed thinner versions of MobileNetV1 called **Thin MobileNet** [44] and **Ultra-thin MobileNet** [45]. They eliminated a few layers, replaced the standard non-linear activation function ReLU with either Drop Activation [44] or Swish [45], and introduced random erasing regularization technique in place of drop out. Random erasing is a kind of data augmentation method where we randomly select rectangular regions in an image, and erase the pixels of that region and substitute them with random values.

The work on MobileNet has been continued by Brzeski et al. [7] and Sandler et al. [42]. The former proposed a novel residual depth-separable convolution block, which is an improvement of the basic building block of MobileNet. They modified the original block by adding an identity shortcut connection (with zero-padding for increasing dimensions) from the input to the output. The latter improved the original architecture by utilizing a *bottleneck design* [21]. The main building block of **MobileNetV2**, which is named inverted residual with linear bottleneck, consists of three layers: a 1×1 pointwise convolution responsible for increasing dimensions, a 3×3 depth-wise convolution, and a 1×1 pointwise convolution responsible for reducing dimensions. Besides, shortcuts are used directly between the bottlenecks. This structure maintains a compact representation at the input and the output, while expanding to a higher-dimensional feature space internally to increase the expressiveness of nonlinear per-channel transformations [22].

Hu et al. [24] suggested a mechanism that allows the network to perform feature recalibration, through which it can learn to use global information to selectively emphasize informative features and suppress less useful ones. Their idea is implemented by a so-called **Squeeze-and-Excitation** (SE) block that explicitly models interdependencies between channels. The SE block can be used directly in existing state-of-the-art architectures by replacing components with their SE counterparts.

Alongside the hand-crafted networks described above, there are also efficient architectures discovered by *Neural Architecture Search* (NAS). NAS automates the design process by employing a reinforcement learning (RL) method that explores a predefined search space to find the network architectures with a good trade-off between accuracy and latency. Unlike the initial work [53], where latency was estimated by Mult-Adds, later research has directly measured real-world latency by executing the model on actual hardware platforms. For instance, **MnasNet** [47], which built upon the MobileNetV2 architecture and the SE block, utilizes latency on the Google Pixel 1 smartphone as the reward to perform RL search. Compared to MobileNetV2, the baseline MnasNet model improves the classification accuracy on ImageNet by three percentage points with similar latency.

Howard et al. [22] extended the MnasNet approach through a combination of complementary search techniques as well as novel architecture advances. The revised framework first uses platform-aware NAS to search for the global network structures and then uses the NetAdapt algorithm [49] to determine the number of filters in each layer. In addition to network search, Howard et al. [22] also introduced several network improvements to further enhance the final model. They redesigned the computationally-expensive layers at the beginning and the end of the network. Besides, they introduced a new nonlinearity, h-swish, a modified version of the swish nonlinearity, which is practical for the mobile setting. Lastly, in contrast with Hu et al. [24], they utilized the SE block inside the residual layer after the depthwise filters in the expansion in order to apply attention on the largest representation. As a result, two **MobileNetV3** models were released: MobileNetV3-Large and MobileNetV3-Small which are targeted for high and low resource use cases respectively.

3. Experimental Setup

In this section we briefly describe a dataset used for our experiments, a training setup with hyperparameter values as well as our methodology for evaluation. We also mention issues that occurred during the training of existing proposals of MobileNet modifications.

3.1. Dataset

We evaluated different neural networks from the MobileNet family on a dataset released for ImageNet Large Scale Visual Recognition Challenge 2012 (IamgeNet 2012) [41]. The train set of IamgeNet 2012 contains 1,200,000 labeled images depicting 1000 object categories. The validation set has 50,000 images with labels from 1000 categories.

3.2. Training Settings

For training our models we used a computer with an AMD Ryzen Threadripper 2920X CPU and GeForce RTX 2080 Ti GPU. The performance tests were conducted on a Raspberry Pi 4B computer with an ARM Broadcom BCM2711 processor, 4 GB of RAM memory and 64-bit Ubuntu 20 LTS installed.

Both training and testing were performed using MxNet library. The reason behind this choice is that MxNet provides many common architectures along with the training scripts. Thus, it allows for reproducing the reported accuracy of these neural nets.

We trained all models with a batch size of 256 using the Nesterov's Accelerated Gradient optimizer with momentum value 0.9. The initial learning rate was set to 0.1 with a tenfold decay in the 40th and 60th epochs. We also used gradual warmup for first 5 epochs. We initialized weights with MSRA function [20] and set the weight decay value to 0.0001. We used label smoothing with the α set to 0.1.

For data augmentation we used mixup training with the λ randomly sampled for each batch from the beta distribution ($\alpha = 0.2$).

3.3. Evaluation method

In order to evaluate each architecture, we wanted to know its accuracy as well as how fast it classifies a single image. To achieve this goal, we used profiler from MxNet library and we measured the time that is needed to classify 50 thousand pictures from the IamgeNet 2012 validation dataset. First, the test set was divided into 50 equal parts. Then, we ran a classification of one picture (batch size set to 1) in a loop for each part of the test data. We repeated this step three times to minimize the risk that other system processes could affect the measurement. Finally, we computed an average from all times obtained from all loops for all test data parts.

3.4. Training issues

We were unable to finish training for two architectures - Squeeze-and-excite MobileNet and FD-MobileNet. This was because of very slow data processing. In both cases, the reason was the SE building block.

We also had to slightly modify two architectures - Thin MobileNet and Ultra-Thin Mobile Net. Both models were designed to be trained on the CIFAR-10 dataset, which consists of 32x32 images. The ImageNet 2012 dataset consists of photos with a higher resolution, thus we had to increase the size of layers.

4. MobileNets performance on Raspberry Pi 4

In this section we analyze accuracy and inference time obtained for MobileNets on Rasperry Pi 4. Table 1 presents performance of different architectures from the MobileNet family (upper part of the table) and their existing modifications (lower part of the table). The second and third columns present accuracy values: reported by inventors of a certain architecture and reproduced by the authors of this paper, respectively. Note, that Thin MobileNet and Ultra-Thin Mobile Net were originally optimized on the CIFER-10 dataset.

Each architecture from the MobileNet family was originally implemented in TensorFlow Lite library and optimized for a Google Pixel 1 mobile phone. Originally, each subsequent version of MobileNet (upper part of Table 1) had better performance that its predecessor by means of classification accuracy as well as latency (not reported here). An exception was made for MobileNetV3-Small, which by assumptions had to be fast and compact with acceptable accuracy.

For basic MobileNet family architectures we managed to reproduce accuracy values with MxNet library. Furthermore, for MobileNetV1 we achieved higher accuracy value than those for MobileNetV2. This is due to better values of hyperparameters during the training process.

Since we did not manage to finish training for Squeeze-and-Excite MobileNet and Fast downsampling strategy, we decided to use accuracy values reported by inventors of these two models in the further analysis.

The fourth column in Table 1 shows the inference time measured according to the procedure described in Subsection 3.3. It should be noticed that we did not obtain decreasing times for first four architectures, as could be expected.

Table 1. Lightweight CNNs performance on Raspberry Pi 4.

Architecture	Accuracy [%]		Time [ms]
	Reported by inventors	Our implementation	
MobileNetV1	70.60	72.90	305.05
MobileNetV2	72.00	72.70	370.46
MobileNetV3-Large	75.20	75.32	397.20
MobileNetV3-Small	67.40	67.72	200.57
Squeeze-and-Excite MobileNet	74.70	-	389.34
Ultra-thin MobileNet	-	61.58	153.88
Thin Mobilenet	-	63.96	126.01
FD-MobileNet	65.30	-	100.98

Both MobileNetV2 and MobileNetV3 Large are slower than MobileNetV1. A similar observation was reported by Bianco et al. [4]. They compared MobileNetV1 and MobileNetV2 on two different computer architectures, a workstation equipped with a NVIDIA Titan X Pascal, and an embedded system based on a NVIDIA Jetson TX1 board. In both cases, MobileNetV1 worked faster than MobileNetV2.

We performed time measurements using MxNet profiler, which allows developers to check performance for each layer in a network. In this way we found out that two layers are responsible for the main difference in times of classification for the MobileNet family, i.e. batch normalization and convolution layers. However, these layers are functionally connected with each other. Exact times are reported in Table 2. Note, that in MobileNetV2 and MobileNetV3 the fully connected layer was replaced with a functionally equivalent pointwise convolution layer. Thus, convolution time reported for MobileNetV1 is the sum of times for convolution and fully connected layers measured with the profiler.

Table 2. Times needed to perform computations for selected layers in the MobileNet family. The time reported for MobileNetV1 is the sum of times for convolution and fully connected layers.

Layer	Time [ms]			
	MobileNetV1	MobileNetV2	MobileNetV3 Large	MobileNetV3 Small
BatchNorm	23.36	41.62	40.83	33.65
Convolution	257.34	298.39	300.45	134.27

Figure 1 presents the overall performance of the aforementioned architectures. Note, that some architectures are dominated by others by means of both time and accuracy values. This is the case for Thin MobileNet and Ultra-Thin MobileNet as well as MobileNetV2, which does not work well on the Raspberry Pi with the MxNet library. Nevertheless, other architectures from the MobileNet family present a good compromise between accuracy and speed.

Depending on an application, one can choose from the Pareto efficient architectures. When time is crucial and accuracy can be reasonable, the best choice is Fast downsampling strategy. On the other hand, when accuracy is crucial and time is secondary, one should choose MobileNetV3 Large or Sqeeze-and-Excite MobielNet which has comparable performance.

The relative difference between inference time of specific CNNs depends on the environment in which the network is run. This confirms the legitimacy of searching for improvements in the MobileNet architectures.

5. Proposed modifications

In this section we describe our modification of MobileNetV3-Small. We decided to modify MobileNetV3-Small because it is relatively fast and has reasonable accuracy. Ahmed et al. [3] noticed that when the filter size is 3×3 pixels, the accuracy is better than when applying filter size of 5×5 pixels or 7×7 pixels. Therefore, we replaced 5x5 filters with filters of a 3×3 size. Besides, we increased the number of channels especially in the later layers.

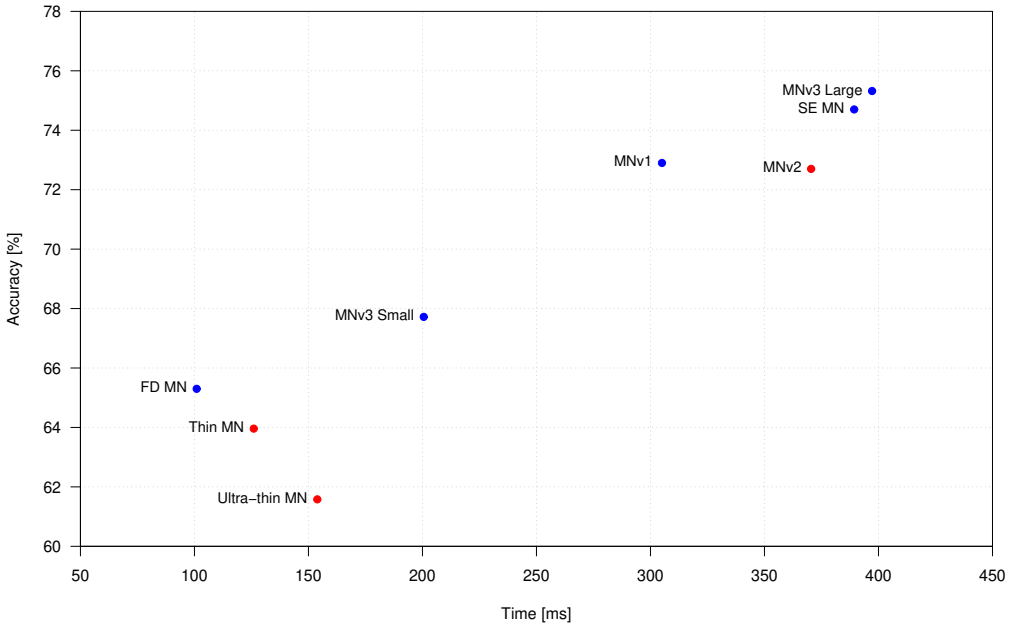

Fig. 1. Comparison of the performance of MobileNet architectures and their modifications. The Pareto efficient architectures are marked in blue, while the dominated ones are marked in red. Abbreviations: MN - MobileNet, SE - Squeeze-and-Excite, FD - Fast Downsampling.

Although the selection of an activation function has a minor influence on the computational costs, it has a significant impact on the network accuracy. Ramachandran et al. [40] showed that changing activation function from ReLU to Swish improves classification accuracy. While this nonlinearity improves accuracy, it comes with non-zero cost in embedded environments, as the sigmoid function is much more expensive to compute on mobile devices [22]. To deal with this problem, Howard et al. [22] proposed a *hard-Swish* activation function. Moreover, they found out that most of the *hard-Swish* benefits are realized in the deeper layers. Nevertheless, *hard-Swish* still introduces some latency cost. Considering all arguments presented above as well as our experimental findings which are presented in Section 6.2, we decided to replace *hard-Swish* with ReLU in our modification of MobileNetV3-Small. Table 3 shows our final architecture, which we name as MobileNetV3-Small-Pi.

6. Results

6.1. Comparison with the MobileNet family and their modifications

Figure 2 depicts comparison of the performance of MobileNetV3-Small-Pi and networks from the MobileNet family. We do not show Pareto dominated architectures here. The fastest architecture is Fast downsampling strategy with accuracy that enables one to use it in practice, especially in the case of when time is crucial.

The accuracy of MobileNetV3-Small-Pi is 68.32%, which is an improvement over the baseline by 0.92% and 0.6% over the replicated result. Besides, our architecture is 36% faster, since we shortened the inference time from 201 ms to 148 ms (Table 4).

Table 4 also presents a comparison of times needed to perform computations for selected layers by MobileNetV3-Small and MobileNetV3-Small-Pi. These two layers were responsible for the slowdown of MobileNetV2 on Raspberry Pi with usage of MxNet library. We observed speedup for the two layers for MobileNetV3-Small-Pi in comparison with the original architecture. This is confirmed by the total execution time of the two architectures.

Table 3. Architecture of MobileNetV3-Small-Pi

Layer	Filter Size	Channels	Stride	Expansion	Activation function
Convolution	3x3	16	2	-	**ReLU**
Bottleneck; SE	3x3	16	2	16	ReLU
Bottleneck	3x3	**16**	2	72	ReLU
Bottleneck	3x3	**32**	1	88	ReLU
Bottleneck; SE	**3x3**	**32**	2	96	**ReLU**
Bottleneck; SE	**3x3**	**64**	1	240	**ReLU**
Bottleneck; SE	**3x3**	**64**	1	240	**ReLU**
Bottleneck; SE	**3x3**	**64**	1	120	**ReLU**
Bottleneck; SE	**3x3**	**256**	1	144	**ReLU**
Bottleneck; SE	**3x3**	**256**	2	288	**ReLU**
Bottleneck; SE	**3x3**	**512**	1	576	**ReLU**
Bottleneck; SE	**3x3**	**512**	1	576	**ReLU**
Convolution	**3x3**	576	1	-	**ReLU**
Global Avg Pooling	7x7	-	-	-	-
Convolution; no batch norm	1x1	**1024**	1	-	**ReLU**
Convolution	1x1	1000	1	-	-

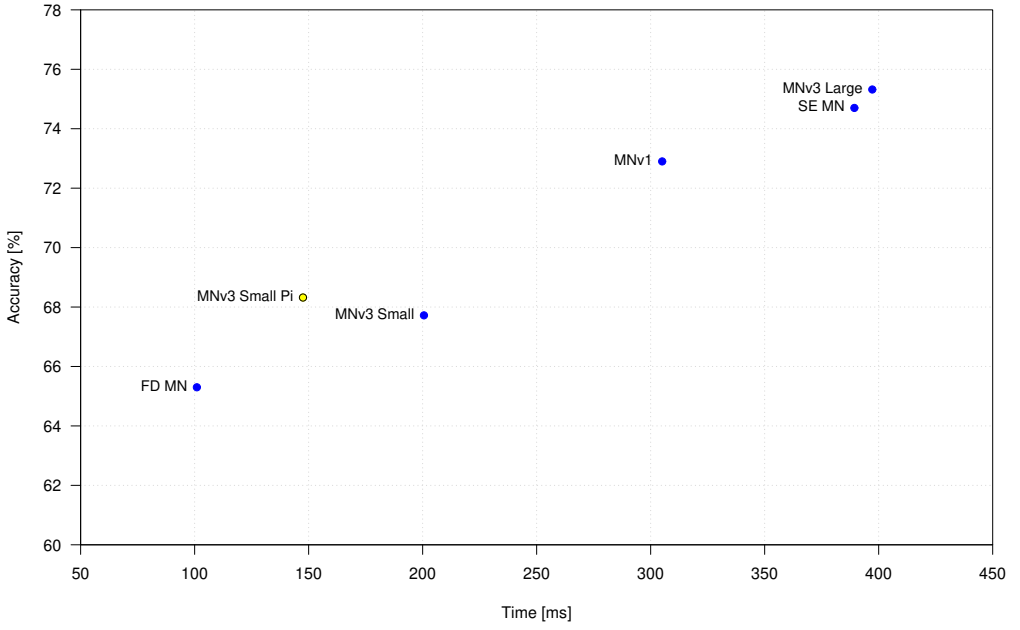

Fig. 2. Comparison of the performance of networks from the MobileNet family and their modifications. The Pareto efficient existing architectures are marked in blue. Our architecture is marked in yellow. Abbreviations: MN - MobileNet, SE - Squeeze-and-Excite, FD - Fast Downsampling.

Table 4. Comparison of times needed to perform computations for selected layers by MobileNetV3-Small and MobileNetV3-Small-Pi.

	Time [ms]	
Layer	MobileNetV3-Small	MobileNetV3-Small-Pi
BatchNorm	33.65	27.64
Convolution	134.27	99.72
Total	200.57	147.52

6.2. Other attempts to improve the MobileNet family

Initially, we tried to improve the MobileNetV1 architecture. Firstly, we investigated different activation functions. When replacing ReLU with Swish, we achieved classification accuracy of about 73.4%, which is 0.5% better than the original. However, computations of Swish function last 100 ms in comparison to 26.8 ms for ReLU. The total run time of this modification is about 436 ms which is the worst time obtained by us during experimentation. Next, we replaced ReLU with Hard-swish, which resulted in an accuracy increase from 72.9% to 74.4%. However, the inference time of this architecture is 391 ms, which is only 6 ms less than that of MobileNetV3 Large. We also experimented with ReLU6. Replacing ReLU with ReLU6 decreases the inference time by 6 ms, but deteriorates the accuracy by 0.7% .

Secondly, we checked the influence of batch normalization reduction on MobileNet performance, we removed batch normalization operations occurring after pointwise convolution. This modification achieves classification accuracy 70.22% and total run time of about 283 ms. Time for computing batch normalization and convolution layers decreased by approx 11.58 ms and 9.36 ms, respectively.

The last modification was the removal of a fully connected layer and decreasing the number of channels from 1024 to 1000. This modification did not have any impact on the inference time. We observed a decrease in accuracy by 1.14%, which is in line with findings by Qian, et al. [36].

7. Conclusions and Further Work

In this paper we compared the performance of all baseline architectures from the MobileNet family on Raspberry Pi 4 with the usage of MxNet library. Moreover, we proposed a new compact CNN architecture, namely MobileNetV3-Small-Pi, which is 36% faster and slightly more accurate on ImageNet classification compared to the baseline MobileNetV3-Small. Our results are useful for practitioners, who are provided with a new state-of-the-art architecture for Raspberry Pi, as well as for researchers, who have a complete view of what solutions have been explored so far and which research directions are worth exploring in the future. As for the latter, we suggest the investigation of a new activation function, i.e. a Fast Exponentially Linear Unit (FELU) [39], which could bring further improvements in the MobileNet architectures. Besides, we encourage other researchers to exploit the platform-aware architecture search approach [11] that could help to improve MobileNets performance on a specific platform. For those who want to continue our research we made our architectures publicly available at https://gitlab.com/WGlegola/faster-mobilenets.

References

[1] Abayomi-Alli, A., Abayomi-Alli, O., Vipperman, J., Odusami, M., Misra, S., 2019. Multi-class classification of impulse and non-impulse sounds using deep convolutional neural network (dcnn), in: Misra, S., Gervasi, O., Murgante, B., Stankova, E., Korkhov, V., Torre, C., Rocha, A.M.A., Taniar, D., Apduhan, B.O., Tarantino, E. (Eds.), Computational Science and Its Applications – ICCSA 2019, Springer International Publishing, Cham. pp. 359–371.

[2] Abayomi-Alli, O.O., Damaševičius, R., Maskeliūnas, R., Misra, S., 2021. Few-shot learning with a novel voronoi tessellation-based image augmentation method for facial palsy detection. Electronics 10. URL: https://www.mdpi.com/2079-9292/10/8/978, doi:10.3390/electronics10080978.

[3] Ahmed, W.S., a. A. Karim, A., 2020. The impact of filter size and number of filters on classification accuracy in cnn, in: 2020 International Conference on Computer Science and Software Engineering (CSASE), pp. 88–93. doi:10.1109/CSASE48920.2020.9142089.

[4] Bianco, S., Cadene, R., Celona, L., Napoletano, P., 2018. Benchmark analysis of representative deep neural network architectures. IEEE Access 6, 64270–64277. doi:10.1109/ACCESS.2018.2877890.

[5] Blinowski, G., Pieczerak, T., 2014. Układ Raspberry Pi jako uniwersalna platforma mikro-appliance do zastosowań sieciowych i bezpieczeństwa. Przegląd Telekomunikacyjny + Wiadomości Telekomunikacyjne nr 7, 697–701.

[6] Brzeski, A., Grinholc, K., Nowodworski, K., Przybyłek, A., 2019a. Evaluating performance and accuracy improvements for attention-ocr, in: Saeed, K., Chaki, R., Janev, V. (Eds.), Computer Information Systems and Industrial Management, Springer International Publishing, Cham. pp. 3–11.

[7] Brzeski, A., Grinholc, K., Nowodworski, K., Przybylek, A., 2019b. Residual mobilenets, in: Welzer, T., Eder, J., Podgorelec, V., Wrembel, R., Ivanović, M., Gamper, J., Morzy, M., Tzouramanis, T., Darmont, J., Kamišalić Latifić, A. (Eds.), New Trends in Databases and Information Systems, Springer International Publishing, Cham. pp. 315–324.

[8] Butt, S.A., Jamal, T., Azad, M.A., Ali, A., Safa, N.S., . A multivariant secure framework for smart mobile health application. Transactions on Emerging Telecommunications Technologies n/a, e3684. doi:https://doi.org/10.1002/ett.3684.

[9] Byra, M., Sznajder, T., Korzinek, D., Piotrzkowska-Wroblewska, H., Dobruch-Sobczak, K., Nowicki, A., Marasek, K., 2019. Impact of ultrasound image reconstruction method on breast lesion classification with deep learning, in: Morales, A., Fierrez, J., Sánchez, J.S., Ribeiro, B. (Eds.), Pattern Recognition and Image Analysis, Springer International Publishing, Cham. pp. 41–52.
[10] Chechliński, Ł., Siemiątkowska, B., Majewski, M., 2019. A system for weeds and crops identification—reaching over 10 fps on raspberry pi with the usage of mobilenets, densenet and custom modifications. Sensors 19, 3787.
[11] Cheng, A.C., Lin, C., Juan, D.C., Wei, W., Sun, M., 2020. Instanas: Instance-aware neural architecture search. Proceedings of the AAAI Conference on Artificial Intelligence 34, 3577–3584. doi:10.1609/aaai.v34i04.5764.
[12] Cychnerski, J., Brzeski, A., Boguszewski, A., Marmolowski, M., Trojanowicz, M., 2017. Clothes detection and classification using convolutional neural networks, in: 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–8. doi:10.1109/ETFA.2017.8247638.
[13] Dai, Y., Xu, B., Yan, S., Xu, J., 2020. Study of cardiac arrhythmia classification based on convolutional neural network. Computer Science and Information Systems , 11–11.
[14] Dembski, J., Szymański, J., 2019. Bees detection on images: Study of different color models for neural networks, in: Fahrnberger, G., Gopinathan, S., Parida, L. (Eds.), Distributed Computing and Internet Technology, Springer International Publishing, Cham. pp. 295–308.
[15] Dembski, J., Szymański, J., 2020. Weighted clustering for bees detection on video images, in: Krzhizhanovskaya, V.V., Závodszky, G., Lees, M.H., Dongarra, J.J., Sloot, P.M.A., Brissos, S., Teixeira, J. (Eds.), Computational Science – ICCS 2020, Springer International Publishing, Cham. pp. 453–466.
[16] Dziubich, T., Białas, P., Znaniecki, Ł., Halman, J., Brzeziński, J., 2020. Abdominal Aortic Aneurysm Segmentation from Contrast-Enhanced Computed Tomography Angiography Using Deep Convolutional Networks. Communications in Computer and Information Science 1260 CCIS, 158–168. doi:10.1007/978-3-030-55814-7_13.
[17] Dziubich, T., Szymański, J., Brzeski, A.M., Cychnerski, J., Korłub, W.M., 2016. Depth images filtering in distributed streaming. Polish Maritime Research nr 2, 91–98. doi:10.1515/pomr-2016-0025.
[18] Gabriel, P.E., Butt, S.A., Francisco, E.O., Alejandro, C.P., 2021. Performance assessment of static and dynamic routings using 6lowpan on small scenarios applied to monitoring of barranquilla flash flood .
[19] Gholami, A., Kwon, K., Wu, B., Tai, Z., Yue, X., Jin, P., Zhao, S., Keutzer, K., 2018. Squeezenext: Hardware-aware neural network design, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1719–1728. doi:10.1109/CVPRW.2018.00215.
[20] He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1026–1034. doi:10.1109/ICCV.2015.123.
[21] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. doi:10.1109/CVPR.2016.90.
[22] Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., Adam, H., 2019. Searching for MobileNetV3. arXiv:1905.02244.
[23] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861.
[24] Hu, J., Shen, L., Sun, G., 2018. Squeeze-and-excitation networks, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7132–7141. doi:10.1109/CVPR.2018.00745.
[25] Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K., 2016. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. arXiv:1602.07360.
[26] Jaderberg, M., Vedaldi, A., Zisserman, A., 2014. Speeding up convolutional neural networks with low rank expansions, in: Proceedings of the British Machine Vision Conference, BMVA Press. doi:10.5244/C.28.88.
[27] Kim, Y.G., Wu, C.J., 2020. Autoscale: Energy efficiency optimization for stochastic edge inference using reinforcement learning, in: 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 1082–1096. doi:10.1109/MICRO50266.2020.00090.
[28] Ko, J., Cheoi, K.J., 2021. A novel distant target region detection method using hybrid saliency-based attention model under complex textures. Computer Science and Information Systems , 1–1.
[29] Koguciuk, D., Chechliński, Ł., El-Gaaly, T., 2019. 3d object recognition with ensemble learning—a study of point cloud-based deep learning models, in: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Ushizima, D., Chai, S., Sueda, S., Lin, X., Lu, A., Thalmann, D., Wang, C., Xu, P. (Eds.), Advances in Visual Computing, Springer International Publishing, Cham. pp. 100–114.
[30] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2017. Imagenet classification with deep convolutional neural networks. Commun. ACM 60, 84–90. doi:10.1145/3065386.
[31] Piñeres-Espitia, G., Butt, S.A., Cañate-Masson, M., Alvarez-Navarro, A., Hassan, S.A., Gochhait, S., 2021. Gas sensing system using an unmanned aerial vehicle, in: 2021 6th International Conference for Convergence in Technology (I2CT), pp. 1–7. doi:10.1109/I2CT51068.2021.9418000.
[32] Podlodowski, L., Roziewski, S., Nurzynski, M., 2018. An ensemble of deep convolutional neural networks for marking hair follicles on microscopic images, in: FedCSIS (Position Papers), pp. 23–28.
[33] Poth, A., Meyer, B., Schlicht, P., Riel, A., 2020. Quality assurance for machine learning – an approach to function and system safeguarding, in: 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS), pp. 22–29. doi:10.1109/QRS51102.2020.00016.
[34] Przybylek, K., Shkroba, I., 2019. Crowd counting á la bourdieu, in: Welzer, T., Eder, J., Podgorelec, V., Wrembel, R., Ivanović, M., Gamper, J., Morzy, M., Tzouramanis, T., Darmont, J., Kamišalić Latifić, A. (Eds.), New Trends in Databases and Information Systems, Springer International Publishing, Cham. pp. 295–305.
[35] Przybylek, K., Shkroba, I., 2020. Crowd counting á la bourdieu: Automated estimation of the number of people. Computer Science and

Information Systems 17, 959–982.
[36] Qian, Z., Hayes, T.L., Kafle, K., Kanan, C., 2020. Do we need fully connected output layers in convolutional networks? arXiv preprint arXiv:2004.13587 .
[37] Qin, P., Chen, J., Zhang, K., Chai, R., 2018a. Convolutional neural networks and hash learning for feature extraction and of fast retrieval of pulmonary nodules. Computer Science and Information Systems 15, 517–531.
[38] Qin, Z., Zhang, Z., Chen, X., Wang, C., Peng, Y., 2018b. Fd-mobilenet: Improved mobilenet with a fast downsampling strategy, in: 25th IEEE International Conference on Image Processing (ICIP), pp. 1363–1367. doi:10.1109/ICIP.2018.8451355.
[39] Qiumei, Z., Dan, T., Fenghua, W., 2019. Improved convolutional neural network based on fast exponentially linear unit activation function. IEEE Access 7, 151359–151367. doi:10.1109/ACCESS.2019.2948112.
[40] Ramachandran, P., Zoph, B., Le, Q., 2018. Searching for activation functions, in: 2018 Sixth International Conference on Learning Representations (ICLR). URL: https://arxiv.org/pdf/1710.05941.pdf.
[41] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115, 211–252. doi:10.1007/s11263-015-0816-y.
[42] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C., 2019. MobileNetV2: Inverted residuals and linear bottlenecks. arXiv:1801.04381.
[43] Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S., Jagersand, M., Zhang, H., 2018. A comparative study of real-time semantic segmentation for autonomous driving, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 700–710. doi:10.1109/CVPRW.2018.00101.
[44] Sinha, D., El-Sharkawy, M., 2019. Thin mobilenet: An enhanced mobilenet architecture, in: 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference, pp. 0280–0285. doi:10.1109/UEMCON47517.2019.8993089.
[45] Sinha, D., El-Sharkawy, M., 2020. Ultra-thin mobilenet, in: 10th Annual Computing and Communication Workshop and Conference, pp. 0234–0240. doi:10.1109/CCWC47524.2020.9031228.
[46] Sun, Y., Yan, Z., 2021. Image target detection algorithm compression and pruning based on neural network. Computer Science and Information Systems , 7–7.
[47] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V., 2019. Mnasnet: Platform-aware neural architecture search for mobile. arXiv:1807.11626.
[48] Wang, Y., Xu, C., Xu, C., Tao, D., 2017. Beyond filters: Compact feature map for portable deep model, in: Precup, D., Teh, Y.W. (Eds.), Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia. pp. 3703–3711. URL: http://proceedings.mlr.press/v70/wang17m.html.
[49] Yang, T.J., Howard, A., Chen, B., Zhang, X., Go, A., Sandler, M., Sze, V., Adam, H., 2018. Netadapt: Platform-aware neural network adaptation for mobile applications, in: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (Eds.), Computer Vision – ECCV 2018, Springer International Publishing. pp. 289–304.
[50] Zhang, T., Hou, M., Zhou, T., Liu, Z., Cheng, W., Cheng, Y., 2020. Land-use classification via ensemble dropout information discriminative extreme learning machine based on deep convolution feature. Computer Science and Information Systems , 10–10.
[51] Zhang, X., Zhou, X., Lin, M., Sun, J., 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6848–6856. doi:10.1109/CVPR.2018.00716.
[52] Zhao, Z., Ren, C., Sun, H., Fan, Z., Gao, Z., 2018. Research on automatic identification technique of ct image in lung. Computer Science and Information Systems 15, 501–515.
[53] Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V., 2018. Learning transferable architectures for scalable image recognition. arXiv:1707.07012.