Deployment of Compressed MobileNet V3 on iMX RT 1060

Kavyashree Prasad S P
IOT Collaboratory IUPUI
Department of Electrical and Computer Engineering
Purdue School of Engineering and Technology
Indianapolis, USA
kshalini@purdue.edu

Mohamed El -Sharkawy
IOT Collaboratory IUPUI Department of Electrical and Computer Engineering
Purdue School of Engineering and Technology
Indianapolis, USA
melshark@iupui.edu

Abstract—**Deep Neural Networks (DNN) are prominent in most applications today. From self-driving cars, sentiment analysis, surveillance systems, and robotics, they have been used extensively. Among DNNs, Convolutional Neural Networks (CNN) have achieved massive success in computer vision applications as the human visual system inspires their architecture. However, striving to achieve higher accuracies, CNN complexity, parameters, and layers were increased, which led to a drastic surge in their size, making their deployment challenging. Over the years, many researchers have proposed various techniques to alleviate this issue—one of them being Design Space Exploration (DSE) to minimize size and computation with little compromise to accuracy. MobileNet V3 is one such architecture designed to achieve good accuracy while being mindful of resources. It produces an accuracy of 88.93% on CIFAR-10 with a size of 15.3MB. This paper further reduces its size to 2.3MB while boosting its accuracy to 89.13% using DSE techniques. It is then deployed into NXP's i.MX RT1060 Advanced Driver Assistance System (ADAS) platform.**

Keywords—MobileNet V3, Convolution Neural Networks, Depthwise Pointwise Depthwise blocks, Compressed MobileNet V3, CIFAR-10, Design space exploration, TensorFlow, i.MX RT 1060.

I. Introduction

Convolutional neural networks have shown commendable performance in various computer vision tasks due to their compelling ability to use multiple feature description stages to grasp representations from images. They first came into the spotlight through the work of LeCun et al. 1989. Since the victory of AlexNet in the ImageNet challenge in 2012, they have gained high popularity [1]. The appealing factor of CNNs is their intelligence to extract spatial and temporal content from crude data. The design of CNN incorporates many convolutional layers, subsampling units, and nonlinear activation functions. Convolution operation guides extraction of valuable features from locally connected information points. Its output is then passed to nonlinear activations that produce diverse activations for various responses and encourage semantic contrasts in pictures. Subsampling is used in CNNs to make it invariant to the location of features. Subsequently, in this way, CNN learns pictures without the need for human involvement in feature extraction. The human visual cortex profoundly propels the building plan of CNNs. In the course of learning, CNN alters weights employing a backpropagation calculation. This ability to move towards the target is comparable to the brain's capacity to memorize based on responses. CNN's multi-layered structure helps gather low, mid, and high-level features, with high-level features being an aggregate of mid and low-level features. This dynamic learning ability of CNN emulates the neocortex in the human cerebrum and is responsible for its pervasiveness.

Over the years, there have been many advances in their architectures, activation functions, regularization, and parameter tuning. In a race to achieve higher accuracy, CNN model complexity and parameters have escalated [2] [3] [4] [5]. This has led to increased demand for resources needed for their storage and computation. Many small-sized architectures were proposed to ease this problem, such as MobileNet [6], SqueezeNet [7], ShuffleNet [8], and so on.

There are various advantages to using small architectures. Firstly, they are more suitable for embedded resource-constrained applications. Due to their size, computations can be performed in place for tasks such as image recognition, semantic segmentation, etc., rather than sending it to the cloud. It reduces latency and assures the privacy of data. In autonomous driving, companies make updates to the model and load it into customer vehicles from their servers [9]. Small-size CNNs make this update more convenient. Hence, to benefit from these advantages, many techniques were developed. Some of them include knowledge distillation [10] [11], pruning, and network quantization [12][13] , low rank and sparse decomposition [14], and developing new innovative architectures[8] [15][16].In knowledge distillation, a small model learns from a large model using a teacher-student approach. In pruning, weights that are insignificant to network performance are zeroed out based on a criterion [17] [18], and in network quantization, filter kernels and weights in fully connected layers are quantized. This quantization can be achieved by various methods such as k-means, Huffman coding, etc. Sparse decomposition and low-rank approximations achieve compression by reducing the parameter dimension of the network. This paper accomplishes a similar purpose by reducing MobileNet V3 small by making architectural modifications and changing the baseline model's activation functions.

Baseline architecture is demonstrated in section 2. Section 3 illustrates changes made to MobileNet V3 small to produce CMV3. Training setup is detailed in section 4. Section 5 has implementation details. Results and conclusion are mentioned in sections 6 and 7, respectively.

II. Prior Work

A. Baseline Architecture

MobileNet V3 is the latest variant of MobileNets. It was designed using a platform-aware network architecture search, and net adapt algorithm. MobileNet V3 small and MobileNet V3 large are two forms of this model developed to serve different resource constraints [15]. MobileNet V3 large has lesser latency than MobileNet V2 while being 3.2% more accurate on the ImageNet dataset [19].

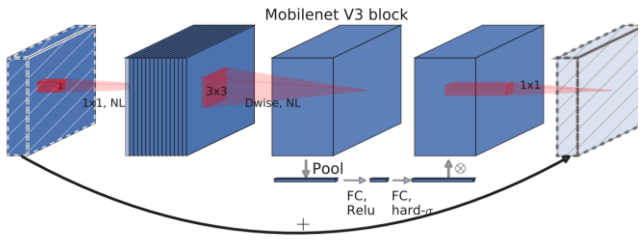


Fig. 1. MobileNet V3 Block [15].

MobileNet V3 encompasses the best practices from MobileNet V2 and Squeeze and excitation networks[20] .It is a combination of inverted residual bottlenecks and squeeze and excitation blocks. These SE blocks are added to improve networks' representational power by suppressing neurons that do not contribute to performance and enhancing those that do. The bottlenecks consist of an initial 1x1 pointwise expansion layer, a depthwise convolution layer (DWC) with a kernel of size 3x3 or 5x5, and a final 1x1 projection layer. The architecture of MobileNet V3 is shown in Fig.1. The model uses the H-swish activation function.

III. Modifications

Modifications made to MobileNet V3 are described below. Table I summarizes CMV3 architecture.

A. Convolution Layers

CNN with more Depthwise convolutions than pointwise convolutions have shown better performance [21]. This fact has been exploited to make architectural changes by emphasizing spatial information rather than aggregating channel information. Depthwise Pointwise Depthwise (DPD) blocks, as shown in Fig.2, were used for the model.These blocks comprise a 3×3 Depthwise convolution, with a stride s that expands the number of channels and performs down-sampling. They also consist of a 1×1 pointwise convolution that collects information along channels, merges them, and finally, a 3×3 Depthwise separable convolution layer. All DPD blocks were followed by batch normalization and RELU.As more Depthwise convolutions are used than pointwise, good compression is achieved. The ratio of number of parameters in Pointwise to Depthwise convolutions is displayed below:

$$\frac{W \times H \times C \times mC}{W \times H \times k \times k \times mC} = \frac{C}{k^2} \qquad (1)$$

(W × H) is the input dimension, C is the number of channels, m is the channel multiplier, and (k × k) is the filter size. Since the number of channels is much higher than the filter size, the ratio to be greater than 1.

B. Mish Activation Function

Mish possesses the self-regularizing capacity and lessens overfitting. It outruns other activation functions in performance. It also keeps negative gradients, has better generalization and eliminates saturation due to near-zero gradients [22]. The formula below can describe it:

$$f(x) = x \cdot \tanh(softplus(x)) \qquad (2)$$

$$softplus(x) = \ln(1 + e^x) \qquad (3)$$

The Mish activation function followed DPD blocks in the new architecture. It improved the accuracy from 88.14% to 89.13%. Fig.3 depicts the graph of Mish.

Compressed MobileNet V3 Architecture						
Input	***Operator***	***e***	***c***	***SE***	***NL***	***s***
32^2×3	Conv2d 3×3	-	16	-	HS	1
32^2×16	Bneck 3×3	48	32	✓	HS	1
32^2×32	DPD 3×3	88	40	-	MH	1
32^2×40	DPD 3×3	240	40	-	MH	1
32^2×40	Bneck 5×5	160	48	✓	HS	2
16^2×48	DPD 5×5	288	96	-	MH	1
16^2×96	DPD 5×5	592	128	-	MH	1
16^2×128	Conv2d 1×1	-	256	✓	HS	1
16^2×256	Pool 16×16	-	-	-	-	1
1^2×256	Conv2d 1×1	-	576	-	HS	1
1^2×576	Conv2d 1×1	-	k	-	-	1

TABLE I

WHERE E: EXPANSION FACTOR, C: NUMBER OF OUTPUT CHANNELS, SE: SQUEEZE AND EXCITE BLOCKS, NL: ACTIVATION, HS: H-SWISH, MH: MISH AND S: STRIDE

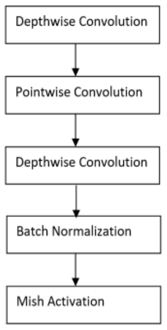


Fig. 2. DPD Blocks

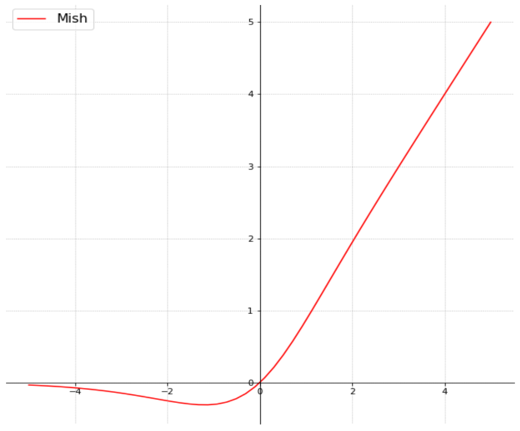


Fig. 3. Mish Activation Function [23].

C. Expansion filters

Mobilenet V3 uses expansion filters to extend to a high dimensional feature space to intensify non-linear transformation on channels [15]. This technique is used on CMV3 as well. Expansion filters in a few layers are increased. It boosted accuracy from 84.56% to 88.14%.

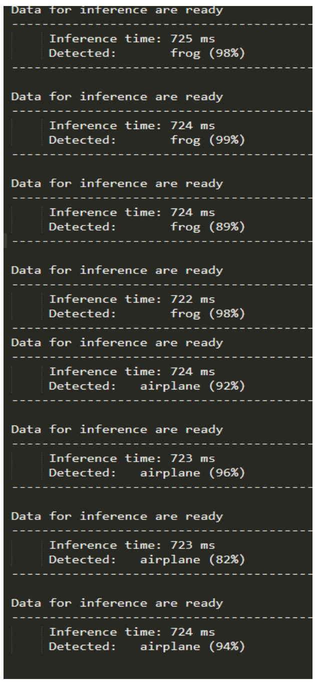
IV. Training Setup

The modified model was trained with Intel Xenon Gold 6126 processor with 32GB RAM and NVIDIA Tesla P100 GPU. An l2 weight decay of 1e-5 was used. A dropout of 0.8 and a cosine decay type scheduler were added. A width multiplier of 0.5 was selected to reduce overfitting and maintain a good trade-off between accuracy and size.

V. Deployment

NXP eIQ is a software platform comprising resources and tools to help machine learning deployment on NXP hardware. It has Neural Network (NN) compilers, libraries, inference engines, Hardware Abstraction Layers (HAL) to support TensorFlow lite (TFLite), ARM NN, glow, Cortex Microcontroller Software Interface Standard (CMSIS)-NN, and OpenCV [24].

The model used TFLite for deployment into iMX RT1060. It is available in both yocto and MCUXpresso environments. It is faster and consumes less memory than TensorFlow, making it suitable for use in low-resource devices. We used MCUXpresso IDE and built the SDK for iMX RT 1060 using eIQ middlewares. This middleware comes with a lot of demo examples. CIFAR-10 label image example was used. This example uses a DL model to classify images captured by the camera attached on board. CMV3 was then included in the header files, and many images were tested to decipher model accuracy and inference after deployment. Fig 4. shows the results observed on the console. Fig 5. shows the block diagram for eIQ inference procedure for TensorFlow Lite.



```
Data for inference are ready
----------------------------------------
     Inference time: 725 ms
     Detected:       frog (98%)
----------------------------------------

Data for inference are ready
----------------------------------------
     Inference time: 724 ms
     Detected:       frog (99%)
----------------------------------------

Data for inference are ready
----------------------------------------
     Inference time: 724 ms
     Detected:       frog (89%)
----------------------------------------

Data for inference are ready
----------------------------------------
     Inference time: 722 ms
     Detected:       frog (98%)
----------------------------------------
Data for inference are ready
----------------------------------------
     Inference time: 724 ms
     Detected:   airplane (92%)
----------------------------------------

Data for inference are ready
----------------------------------------
     Inference time: 723 ms
     Detected:   airplane (96%)
----------------------------------------

Data for inference are ready
----------------------------------------
     Inference time: 723 ms
     Detected:   airplane (82%)
----------------------------------------

Data for inference are ready
----------------------------------------
     Inference time: 724 ms
     Detected:   airplane (94%)
```

Fig. 4. Classification on i.MX RT 1060 as displayed using semi hosting [24]

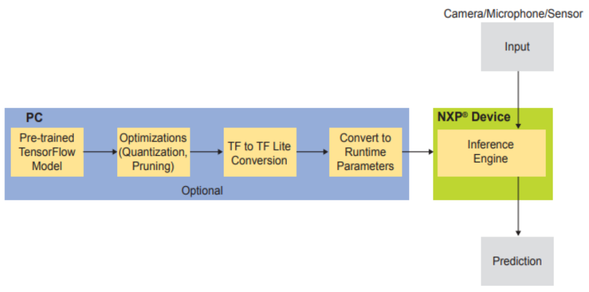


Fig. 5. eIQ inference procedure for TFLite models [24]

VI. Results

MobileNet V3 small was modified to give rise to CMV3 with no compromise to accuracy. The revised model has a size of 2.3 MB with an accuracy of 89.13%. Its parameter count has been reduced from 1,846,930 in baseline to 171,946 after compression. It was then deployed onto iMX RT 1060 for inference. It gave an average inference time of 720ms. A plot of proposed model accuracy vs the number of epochs using the Tensorboard visualization tool is shown in Fig. 6.

Various Scaling factors for CMV3.		
Width Multiplier	*Model Accuracy*	*Model size*
1 5	91.39%	10.5 MB
1.0	90.64%	5.2 MB
0.75	90.10%	3.6 MB
0 5	89.13%	2.3 MB
0 35	87.36	1.9 MB

TABLE II

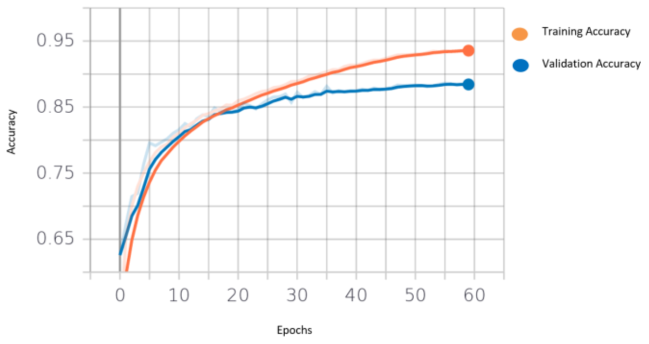


Fig. 6. Compressed MobileNet V3

VII. Conclusion

In this paper, using DPD blocks, mish activation function, and increase in expansion filters, an architecture that is 84.96% smaller in size and 0.2% more accurate than baseline is accomplished. It can be successfully used in various embedded vision platforms. Table II shows different width scaling factors that can be used with the model. Based on the application, a suitable configuration can be used to achieve optimal trade-off.

Acknowledgment

The authors would like to thank Lilly Endowment Inc for their support to Indiana University Pervasive Technology Institute.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *arXiv:1512.03385 [cs]*, Dec. 2015, Accessed: Mar. 09, 2021. [Online]. Available: http://arxiv.org/abs/1512.03385.

[3] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” *arXiv:1602.07261 [cs]*, Aug. 2016, Accessed: Mar.

09, 2021. [Online]. Available: http://arxiv.org/abs/1602.07261.

[4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *arXiv:1512.00567 [cs]*, Dec. 2015, Accessed: Mar. 09, 2021. [Online]. Available: http://arxiv.org/abs/1512.00567.

[5] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv:1409.1556 [cs]*, Apr. 2015, Accessed: Mar. 09, 2021. [Online]. Available: http://arxiv.org/abs/1409.1556.

[6] 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,” *arXiv:1704.04861 [cs]*, Apr. 2017, Accessed: Mar. 12, 2021. [Online]. Available: http://arxiv.org/abs/1704.04861.

[7] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size,” *arXiv:1602.07360 [cs]*, Nov. 2016, Accessed: Mar. 12, 2021. [Online]. Available: http://arxiv.org/abs/1602.07360.

[8] X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, Jun. 2018, pp. 6848–6856, doi: 10.1109/CVPR.2018.00716.

[9] F. Iandola and K. Keutzer, “Keynote’ESWEEK’2017:’Small’Neural’Nets’Are’Beautiful:’ Enabling’Embedded’Systems’with’Small’DeepBNeuralB Network’Architectures’,” *arXiv:1710.02759 [cs]*, p. 10, Sep. 2017.

[10] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, “Learning Efficient Object Detection Models with Knowledge Distillation,” p. 10.

[11] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network,” *arXiv:1503.02531 [cs, stat]*, Mar. 2015, Accessed: Mar. 09, 2021. [Online]. Available: http://arxiv.org/abs/1503.02531.

[12] Y. Gong, L. Liu, M. Yang, and L. Bourdev, “Compressing Deep Convolutional Networks using Vector Quantization,” *arXiv:1412.6115 [cs]*, Dec. 2014, Accessed: Mar. 09, 2021. [Online]. Available: http://arxiv.org/abs/1412.6115.

[13] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, “Quantized Convolutional Neural Networks for Mobile Devices,” *arXiv:1512.06473 [cs]*, May 2016, Accessed: Mar. 09, 2021. [Online]. Available: http://arxiv.org/abs/1512.06473.

[14] X. Yu, T. Liu, X. Wang, and D. Tao, “On Compressing Deep Models by Low Rank and Sparse Decomposition,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, Jul. 2017, pp. 67–76, doi: 10.1109/CVPR.2017.15.

[15] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, “Searching for MobileNetV3,” *arXiv:1905.02244 [cs]*, Nov. 2019, Accessed: Mar. 09, 2021. [Online]. Available: http://arxiv.org/abs/1905.02244.

[16] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, “MnasNet: Platform-Aware Neural Architecture Search for Mobile,” *arXiv:1807.11626 [cs]*, May 2019, Accessed: Mar. 12, 2021. [Online]. Available: http://arxiv.org/abs/1807.11626.

[17] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning Convolutional Neural Networks for Resource Efficient Inference,” *arXiv:1611.06440 [cs, stat]*, Jun. 2017, Accessed: Dec. 23, 2020. [Online]. Available: http://arxiv.org/abs/1611.06440.

[18] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning Filters for Efficient ConvNets,” *arXiv:1608.08710 [cs]*, Mar. 2017, Accessed: Dec. 23, 2020. [Online]. Available: http://arxiv.org/abs/1608.08710.

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *arXiv:1409.0575 [cs]*, Jan. 2015, Accessed: Mar. 09, 2021. [Online]. Available: http://arxiv.org/abs/1409.0575.

[20] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, “Squeeze-and-Excitation Networks,” *arXiv:1709.01507 [cs]*, May 2019, Accessed: Mar. 10, 2021. [Online]. Available: http://arxiv.org/abs/1709.01507.

[21] G. Li, M. Zhang, Q. Zhang, Z. Chen, W. Liu, J. Li, X. Shen, J. Li, Z. Zhu, and C. Yuen, “PSDNet and DPDNet: Efficient Channel Expansion,” p. 15.

[22] D. Misra, “Mish: A Self Regularized Non-Monotonic Activation Function,” *arXiv:1908.08681 [cs, stat]*, Aug. 2020, Accessed: Mar. 10, 2021. [Online]. Available: http://arxiv.org/abs/1908.08681.

[23] D. Misra, “Mish: A Self Regularized Non-Monotonic Neural Activation Function,” p. 13.

[24] NXP, “EIQ-FS.pdf.” Aug. 05, 2020, Accessed: Mar. 12, 2021. [Online]. Available: https://www.nxp.com/docs/en/fact-sheet/EIQ-FS.pdf.