FLAME: A NEW CNN MODULE FOR LARGE MODEL COMPRESSION

Ching-Yen Shih, Po-Hsiang Huang*, Sheng-Lung Chung*, Shang-Fu Chen*
Yu-Chiang Frank Wang, and Chiou-Shann Fuh

National Taiwan University
r06943182@ntu.edu.tw, pxhuang1994@gmail.com,
r06942052@ntu.edu.tw, sam145637890@gmail.com,
ycwang@ntu.edu.tw, fuh@csie.ntu.edu.tw

ABSTRACT

Nowadays, people desire to have a deep neural network operating on edge devices. However, it is hard to implement due to the hardware limitation of the device. In this paper, we introduce a new convolutional neural network filter module, **Flame**, which can substitute for standard convolutional filters to highly compress the CNN model. Our module combines the merits of both the Fire module from SqueezeNet [1] and depthwise separable convolutions from MobileNets [2], helping us to reach a better compression ratio but only suffer little performance drop. We verify our module on three different datasets. To elaborate, we can reduce 91.2% of parameters of our VGG11-based[3] model but have less than 1% accuracy drop. Furthermore, we also perform experiments of quantization of the model to observe the tradeoffs between accuracy and size of our module.

***Index Terms*—** Flame, Convolutional Neural Networks, Model Compression, Weight Quantization

1. INTRODUCTION

Recent years, convolutional neural network (CNN) have succeeded in dealing with various problems of computer vision, such as image classification, object detection, scene segmentation, just to name a few. However, many existing methods require rather deep architecture model and lots of parameters to achieve better performance, which results in large model size. For example, the size of DenseNet161 is 111 MB and the size of ResNet101 is 171 MB. It is hard for us to store and compute such large networks on edge devices such as cell phones, tablets and IoT devices. This limitation usually stop people from dealing daily problems with deep neural network models. Thus, compressing size of large models has become a hot topic since the emergence of deep neural network (DNN).

In the standard structure of DNN for computer vision task, most of the parameters are distributed at the CNN structure due to the large number of convolutional filters. For example, the convolutional layers occupy nearly 95% parameters of ResNet101, and 92% parameters of DenseNet161. As a result, convolutional layers account for most of the network size. In this paper, we focus on reducing the parameters of convolutional layers and show how to compress the model by replacing the convolutional layers with lighter structure. We combine the merits of exist methods [1] and [2] to further reach better performance. In addition, we use quantization to further compress our model. Finally, we present and analyze the classification performance of our model on MNIST[4], Fashion-MNIST[5] and CelebA[6] to show the capability of our module.

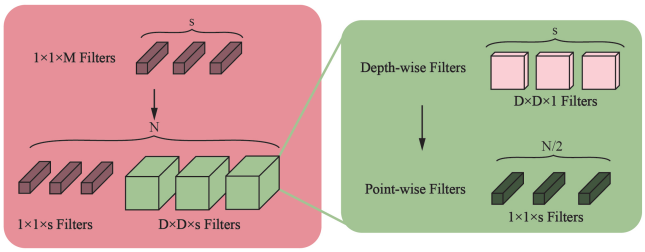
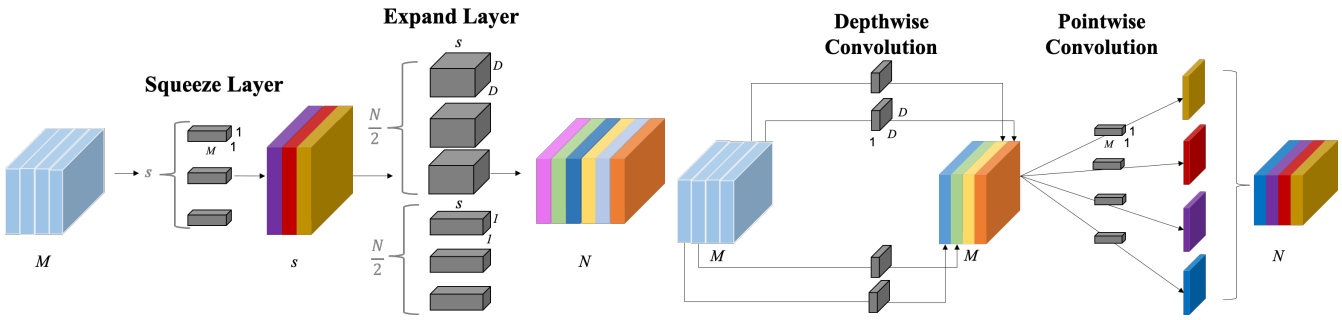


Fig. 1: Structure of Flame module. We replace the 3×3 convolution in the red block with the convolution operation in the green block. The red block is Fire module and the green part is the depthwise separable convolution.

2. RELATED WORK

The ways of compressing deep neural network can be generally divided into three categories: network pruning, weight quantization, and light architecture. Network pruning is the process of deleting relatively unimportant connections of a network, because these connections contribute less to the output. Manessi et al. [7] proposed an automated pruning method, which can perform pruning during the backpropagation. Lee et al. [8] presented a method pruning the network before training. On the other hand, weight quantization is to

*-indicates equal contribution.



(a) Fire module (b) Depthwise separable convolution

Fig. 2: (a) Structure of Fire module. The filters are represented by dark gray cubes and feature maps are represented by other colors. Note that the number of feature map's channel is squeezed to 3 after passing through squeeze layer and then expand to 6 after passing through expand layer in this case. (b) Structure of depthwise separable convolution module. First, the depthwise filters are applied to the feature map to perform channel-wise convolution. Then, the pointwise filters combine the output of the previous stage to give the final output. The filters are also represented by dark gray cubes.

convert parameters from continuous set to a discrete set of value and then attain the lower storage. Many quantization methods had already been proposed. Gong et al. [9] used a k-means algorithm to cluster the parameters and achieved 20 times compression with only 1% loss of accuracy. Rastegari et al. [10] can quantize the network during the training phase, and finally turn the parameters into binary numbers. Moreover, Han et al. [11] even designed a framework including pruning, quantization, and Huffman encoding to minimize the model.

As the name suggests, to have a light structure means to decrease the amount of the parameters in the model with a more efficient way to do convolution. Although the network may be complex, we can reduce the size of a model greatly. Lots of researchers work on designing the smaller CNN architecture. [12] proposed Inception module, which can achieve dimensionality reduction by using 1×1 convolutions before expensive convolutions. Iandola et al. [1] introduced the Fire module as a new building block of CNN architectures. The Fire module contains a squeeze layer and an expand layer, consuming much less space than normal convolutional filters to store weight parameters but still get satisfying results. They used this module to build SqueezeNet which can reach AlexNet-level accuracy with fewer parameters. Howard et al. [2] built MobileNets using depthwise separable convolutions, which decompose the original convolution into a depthwise one and a pointwise one. The combination of depthwise filters and pointwise filters not only requires fewer parameters but also occupies less computation resource. In this paper, the method we propose also belongs to the light structure category.

3. CNN MODULE ARCHITECTURE

In this section, we introduce our CNN module architecture. We combine the Fire module from SqueezeNet and the depthwise separable convolutions from MobileNets to build a new module - Flame. First, we review the architecture of Fire module in section 3.1. Second, depthwise separable convolutions is described in section 3.2. And finally, we introduce the architecture of our new module - Flame in section 3.3.

3.1. Fire Module

A standard convolution layer can be parameterized as

$$D \times D \times M \times N \tag{1}$$

D is the kernel size, M is the number of the input channel, and N is the number of the output channel. SqueezeNet uses Fire module as a building block, whose purpose is to decrease the number of input channels to $D \times D$ convolution filters. Fire module consists of a squeeze layer and an expand layer as shown in the Fig. 2(a). By applying squeeze layer, which is composed by s filters with size $1 \times 1 \times M$, Fire module limits the number of input channels to $D \times D$ convolution filters from M to s, which is smaller than M. On the other hand, expand layer is composed by $e_{1\times1}$ filters with size $1 \times 1 \times s$ and $e_{D\times D}$ filters with size $D \times D \times s$, where $e_{1\times1} + e_{D\times D} = N$. After feeding features into expand layer, a desired number of output channels N can be obtained. In the following of this paper, we set $e_{1\times1} = e_{D\times D} = N/2$.

With Fire module, the parameters of convolution layer can be reduced to

$$(1\times1\times M\times s)+(1\times1\times s\times N/2+D\times D\times s\times N/2) \tag{2}$$

SqueezeNet also can balance its performance and model size by tuning the squeeze ratio (SR), which is the ratio of

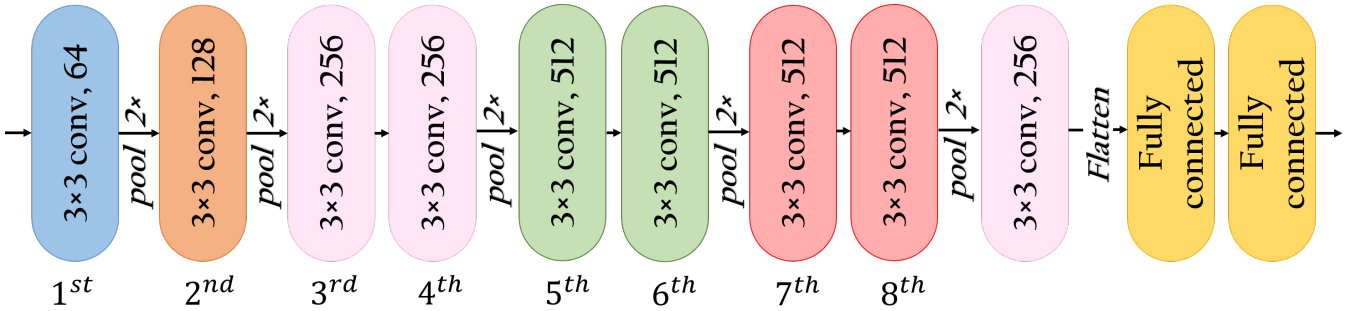


Fig. 3: Network architecture. We use feature extractor of the VGG11 with batch normalization following with two fully connected layers as our basic model. We halve the number of filters in the first 9 layers as the shrink version of our model, and we remove the 3rd, 5th and 7th layer as the shallow version of our model.

the number of filters between a squeeze layer and a expand layer, that is, s/N. Therefore, one can make the best use of Fire module by tuning this hyperparameter according to how much computational power a device has. We inherit this concept when designing our module which is able to adapt to different scenarios.

3.2. Depthwise Filters and Pointwise Filters

Depthwise separable convolution is a two-step convolution, depthwise and pointwise convolutions, as shown in Fig. 2(b). A standard convolution layer has $D \times D \times M \times N$ parameters in total, mentioned in section 3.1. The depthwise filters, which are M filters with size $D \times D \times 1$, are applied to each channel (depth) of input feature maps. The pointwise filters, which are N filters with size $1 \times 1 \times M$, linearly combine the output of depthwise convolution and generate output feature with N channels. After splitting the original filters into smaller filters, the parameters of convolution layer can be reduced to

$$(D \times D \times 1 \times M) + (1 \times 1 \times M \times N) \tag{3}$$

3.3. Flame Module

Although SqueezeNet and MobileNets proposed effective and light structure networks to perform on different tasks, they didn't verify the effect of their modules on compressing existing models. Since machine learning based methods for different tasks rely heavily on model architectures, we argue that a compressing method that only compresses the size of a model but not changes macro-architecture, like the number of hidden layers, is very important. Therefore, we replace standard convolutional layers of the existing model with both of the modules mentioned above to observe the size compression. Also, we merge the characteristics of both modules and propose a new module named Flame module, which is a further compact CNN module.

As Fire module still contains $N/2$ filters with size $D \times D \times s$ in the expand layer, which still account for most of the computation, we apply depthwise separable convolutions to save more parameters. That is, we replace filters mentioned above with s depthwise filters with size $D \times D \times 1$ and $N/2$ pointwise filters with size $1 \times 1 \times s$. Finally, we reduce parameters of convolution layer to

$$(1 \times 1 \times M \times s) + (D \times D \times 1 \times s) + (1 \times 1 \times s \times N/2) \tag{4}$$

We compare number of parameters of a single convolutional layer between our module and other structures on a standard setting with 3×3 kernel size, 256 input channels, and 512 output channels as shown in Table 1.

We can reduce 95.78% parameters by using our module, which is much more than the effect of Fire module (84.72%, when SR = 0.125) and depthwise separable convolutions (88.69%). Furthermore, we can tune the squeeze ratio depending on how small the model we need and how much the acceptable accuracy drop is. The higher squeeze ratio, which means larger model size, can achieve better accuracy, and vice versa.

CNN Structure	Parameters (1 thousand)	Compression ratio
Standard CNN	1179.648	–
Fire module	180.224	84.72%
Depthwise separable	133.376	88.69%
Our module	**49.728**	**95.78%**

Table 1: The comparison of the number of parameters with different CNN modules. Our module reach the best compression ratio and contain least parameters compare to previous CNN structure.

4. EXPERIMENTS

Our baseline model comprises feature extractor of VGG11[3] and two fully connected layers as shown in Fig. 3. Fig. 4 shows the flow chart of our steps to compress our baseline model. We replace the standard convolutional layers of our

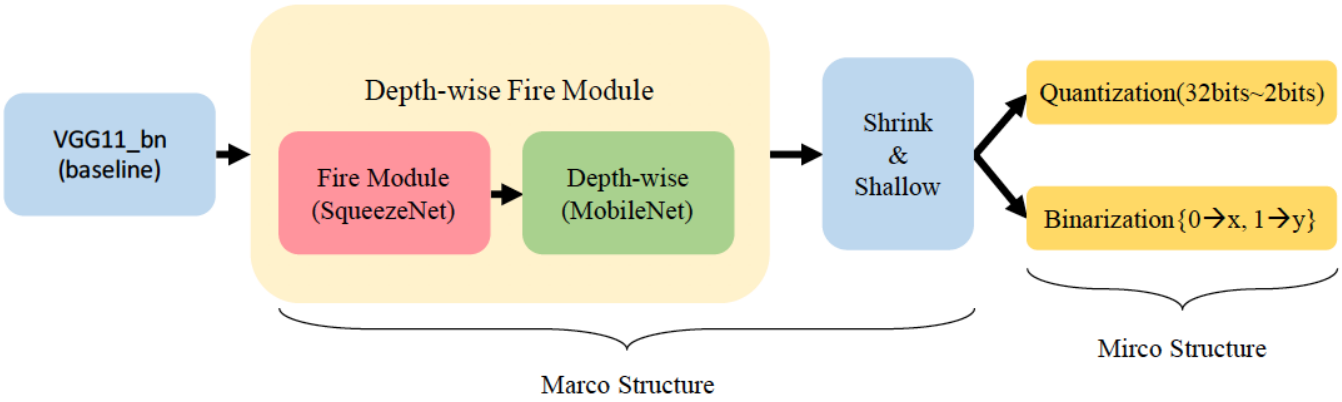


Fig. 4: The flow chart of our methods to compress a models, which contains three steps. First, replace the standard convolutional layers with our Flame module. Second, shrink or shallow the model by halving the filters or remove several layers. Finally, perform weight quantization or weight binarization to further compress the model.

baseline model with three different kinds of filter, namely Fire, depthwise separable convolution, and Flame. To verify the performance of each kind of filter, we perform the experiments on two tasks: MNIST-related classification (digit recognition and product recognition) and human face recognition (CelebA). We compare the accuracy and compression ratio of three different convolution methods. We also compare the computation efficency (FLOPs) of different modules. Also, we adjust the squeeze ratio of our model to compare its influence on the performance of the model. Besides, we perform weight quantization to compress the model to explore the limit of different modules.

Structure	**MNIST**	**Fashion-MNIST**	**Size (MB)**
Standard CNN	99.56%	93.48%	36.67
Fire module	99.35%	92.48%	4.87
Depthwise separable	99.26%	91.26%	5.57
Flame module (SR=0.750)	99.53%	93.21%	8.48
Flame module (SR=0.250)	99.41%	92.67%	3.79
Flame module (SR=0.125)	99.36%	91.71%	**2.61**

Table 2: Experiment results of the baseline model with different CNN modules on the MNIST-related datasets. SR means the squeeze rate of the fire module. Our module still retain compatible performance even the size of model decrease from 36.67 MB to 2.61 MB

4.1. MNIST and Fashion-MNIST classification

We first compare the performance of each module on the simple datasets: MNIST and Fashion-MNIST. MNIST is a large dataset of handwritten digits containing numbers from 0 to 9. Fashion-MNIST is a dataset of dressing images from Zalando's article which can also be divided into 10 classes. The experiment results are shown in Table 2. After replacing convolutional layers with our module, the model size reduces from 36.67 MB to 2.02 MB (with SR = 0.125). As MNIST dataset has now been tackled well, the accuracy of the baseline model is nearly 100%. However, with model size extremely reduction on the model with our module, the accuracy hardly drop. On Fashion-MNIST, the accuracy only drops 0.2% after model size is reduced from 36.67 MB to 8.48 MB (SR = 0.75).

Model	**Size(MB)/ CR(%)**	**FLOPs (M)**	**Accuracy (%)**
Baseline	42 / -	2120	77.33
Fire module	6.1 / 85	480	79.13
Depthwise separable	10.4 / 75.2	260	79.66
Flame	3.7 / 91.2	110	76.42
+Shrink	**3.2 / 92.4**	**31.7**	**78.02**
+Shallow	3.2 / 92.3	71.7	74.8
+Shrink & Shallow	2.7 / 93.5	21.7	63.29

Table 3: Experiment results of the CNN with different structure on the CelebA. The shrink version of our model achieve great performance both in model size and accuracy.

4.2. Human Face Recognition

We perform the experiment of human face recognition on CelebA. Table. 3 shows the compression ratio, floating-point

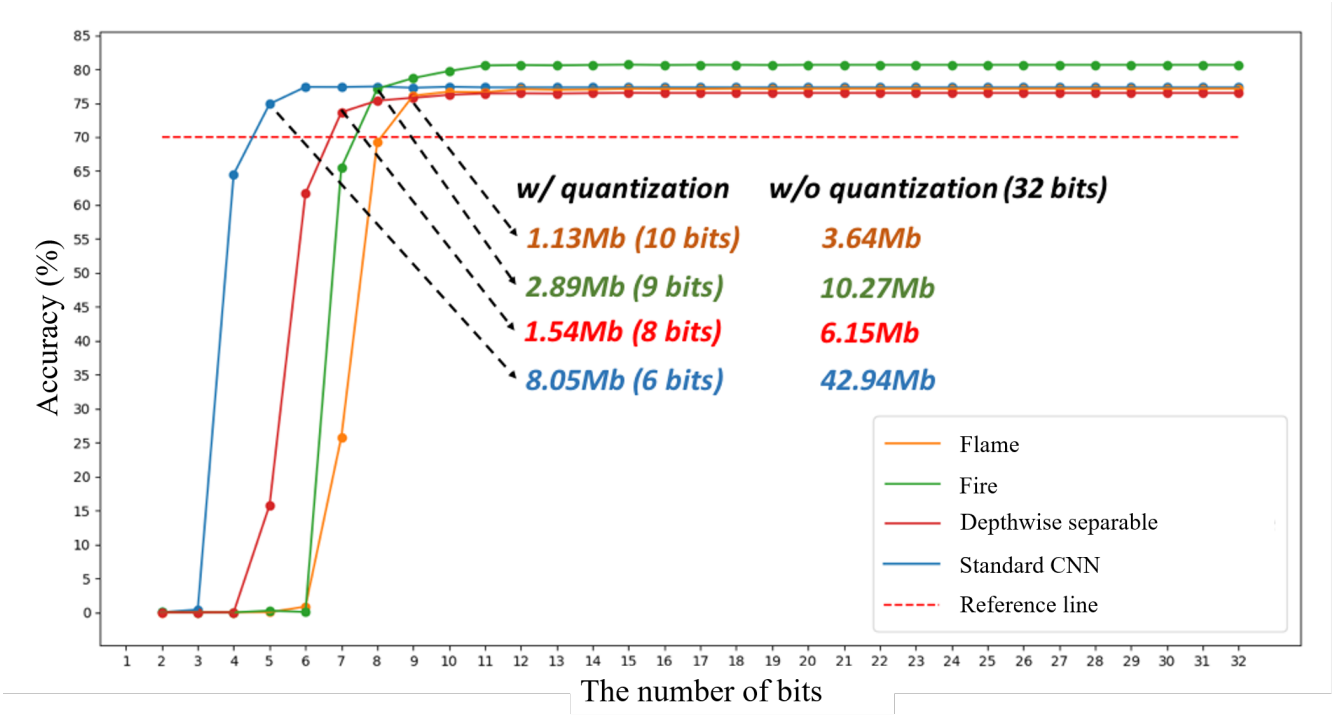


Fig. 5: The results of human face recognition from different module to different quantization bits. The red dashed line is set to be our reference threshold (70%) to measure the performance drop of models.

operations (FLOPs) and accuracy of our experimental results. After replacing convolutional layers in baseline model with our module, model size is compressed from 42 MB to 3.7MB and FLOPs are reduced from 2120 M to 110 M. The proposed module realizes 91.2% compression ratio with less than 1% accuracy drop, surpassing Fire module and depthwise separable convolutions.

Except simply change the convolutional structure of the model, we further perform experiments on shrink and shallow version of our model. The shrink version means that we halve the number of filters except last two CNN layers of the original network to reduce parameters. The shallow version means that we remove the third, fifth and seventh layer of the model. The results are shown in the last three row of Table 3. Although the size of shrink and shallow version is almost equal, the accuracy is about 3.2% difference, meaning that the depth of the network is more important than the number of filters. However, if we combine both strategy of shrink and shallow - a shrink shallow version of our model, we observe an obvious accuracy drop.

To sum up, the best model is the shrink version with our proposed module, which is only 3.2 MB in model size and 31.7M in FLOPs with 78.02% recognition accuracy on CelebA dataset.

min max.png

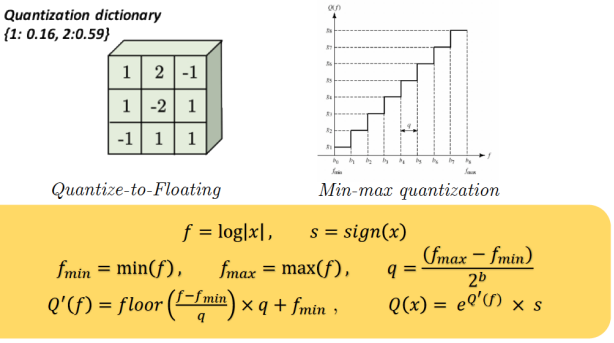


Fig. 6: The procedure of our quantization method - log min-max quantization.

4.3. Weight Quantization

After replacing the convolutional layers of baseline model with different modules, we apply parameter quantization on the model to observe the tradeoffs between weight precision and model performance. As shown in Fig. 6, we create a quantization dictionary containing floating numbers based on log min-max quantization method. Log min-max quantization divide the interval of weight space into 2^b value, and each value is represented by b-bit (b is the number quantization bit). When loading the network to memory for inference, we first look up the quantization dictionary to find the

correspond floating numbers to the b-bit representation. This method helps us to store the model with b/32 times space than the original 32-bit floating points.

Fig. 5 shows the human face recognition accuracy result of different quantization bits. We can observe that every model has a limit of quantization called quantization-threshold, where recognition accuracy will severely drop after quantization lower than this limit. As Table 4 shows, models with larger size can endure more bits loss but retain the accuracy. We test weight quantization on the shrink version of Flame mentioned in the previous section. Although our model has the highest quantization-threshold, it still gets the smallest size after quantization due to its original compression ratio. Finally, using our Flame module, our model can reach 76.11% recognition accuracy on CelebA dataset, which is a competitive performance, while the size of the model is only 1.13MB, meaning 97.37% compression ratio compared to the baseline model.

Methods	**# of bits / Size(MB)**	**CR**	**Accuracy**
Baseline model	- / 42.94	-	77.33%
Quantized Baseline	6 / 8.05	81.25%	74.89%
Fire module	8 / 1.54	96.41%	77.12%
Depthwise separable	9 / 2.89	93.27%	73.67%
Flame module	10 / **1.13**	**97.37**%	76.11%

Table 4: The result of the models applied quantization method. The first row shows the attributes of our baseline model without quantization. Our Flame module reach the best performance on compression rate and only a slightly accuracy drop on image classification.

5. CONCLUSION

In this paper, we introduce the Flame module, which combines advantages of Fire module and depthwise separable convolution, to serve as an alternative of standard convolutional layers and compress existing models. We verify our module by comparing experiment results of human face recognition and MNIST-related classification with baseline model and models substituted with other modules. Results show that our module can not only achieve the best reduction of the model size but also save the computational operation. Also, it can still retain the accuracy of the baseline model. In addition, we get the 97.37% of compression ration by halving the number of filters of convolutional layers and quantizing parameters to 10 bits with only about 1% accuracy drop.

6. REFERENCES

[1] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[2] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[3] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] Yann LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[5] Han Xiao, Kashif Rasul, and Roland Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[6] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang, "From facial parts responses to face detection: A deep learning approach," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3676–3684.

[7] Franco Manessi, Alessandro Rozza, Simone Bianco, Paolo Napoletano, and Raimondo Schettini, "Automated pruning for deep neural network compression," *arXiv preprint arXiv:1712.01721*, 2017.

[8] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr, "Snip: Single-shot network pruning based on connection sensitivity," *arXiv preprint arXiv:1810.02340*, 2018.

[9] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[10] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[11] Song Han, Huizi Mao, and William J Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.