An Empirical Study on Model Pruning and Quantization

Yuzhe Tian[1], Tom H. Luan[2], and Xi Zheng[1(✉)]

[1] School of Computing, Macquarie University, Macquarie Park, NSW 2109, Australia
yuzhe.tian@hdr.mq.edu.au, james.zheng@mq.edu.au
[2] School of Cyber Engineering, Xidian University, Xi'an 710126, Shaanxi, China
tom.luan@xidian.edu.cn

Abstract. In machine learning, model compression is vital for resource-constrained Internet of Things (IoT) devices, such as unmanned aerial vehicles (UAVs) and smart phones. Currently there are some state-of-the-art (SOTA) compression methods, but little study is conducted to evaluate these techniques across different models and datasets. In this paper, we present an in-depth study on two SOTA model compression methods, pruning and quantization. We apply these methods on AlexNet, ResNet18, VGG16BN and VGG19BN, with three well known datasets, *Fashion-MNIST*, *CIFAR-10*, and *UCI-HAR*. Through our study, we draw the conclusion that, applying pruning and retraining could keep the performance (average less than 0.5% degrade) while reducing the model size (at 10× compression rate) on spatial domain datasets (*e.g.* pictures); the performance on temporal domain datasets (*e.g.* motion sensors data) degrades more (average about 5.0% degrade); the performance of quantization is related with the pruning rate and the network architecture. We also compare different clustering methods and reveal the impact on model accuracy and quantization ratio. Finally, we provide some interesting directions for future research.

Keywords: Model compression · Deep neural network · Edge computing

1 Introduction

As the role of Internet of Things (IoT) and edge computing becomes more important, the amount of deployed IoT devices and edge devices keeps growing [39]. These devices are designed for a long period usage even under unattended environments. Thus long battery life becomes the first priority of design, as opposed to the computing abilities. On the other hand, these devices will produce real-time data. It is unrealistic to process these tremendous amount of data acquired by these devices over cloud as the data transmission would require stable internet connection and efficient bandwidth, and consume considerable amount of power.

W. Wang and J. Wu (Eds.): BROADNETS 2023, LNICST 511, pp. 111–125, 2023.
https://doi.org/10.1007/978-3-031-40467-2_7

This is against the real-time processing, power efficiency and network tolerance requirements of IoT devices. For instance, fatigue detection during driving needs to respond in real-time for live saving [27]. Unmanned aerial vehicles (UAVs) require power efficiency to conduct long term tasks without power recharge [14]. Human activities detection would require long term wearing without interrupt users' daily life [4,10,25,26]. Offshore oil platforms usually contain considerable IoT devices but the devices' connection to Internet is limited [30]. Industrial control systems require real-time response for monitoring and security [13,34].

Conversely, as the main focus of current research is targeted on increasing model accuracy, modern deep neural networks usually contain dozens to hundreds of layers, *e.g.* MobileNet [19] and Vision Transformer (ViT) [12]. The corresponding model parameters have also proportionally increased, which leads to a large model size. Though these models could perform well, they require a substantial computing resources to deploy. For instance, the size of commonly used pre-trained model VGG19 [32] is 550MB, while the onboard random access memory (RAM) of Raspberry Pico is only 264KB[1]. Not only the IoT devices, modern edge devices, *e.g.* micro controller units (MCUs) and mobile phones, are also limited by their battery capacity and computing resources. Table 1 shows a comparison of commonly used model sizes against mainstream edge devices memory capacity.

Table 1. Model size/hardware memory comparison.

Network	Model Size[d]	*B1*[a]	*B2*[a]	*B3*[a]	*SW1*[b]	*SW2*[b]	*M1*[c]	*M2*[c]
		Memory Capacity						
ResNet18	$\approx 45MB$	32 KB	32 KB	8 MB	1 GB	1.5 GB	6 GB	8GB
AlexNet	≈235 MB							
VGG11	≈500 MB							
VGG19	≈550 MB							

[a] *B1*: Arduino MKR1010, *B2*: Arduino Portenta H7, *B3*: Raspberry Pi Zero
[b] *SW1*: Apple Watch Series 8, *SW2*: Samsung Galaxy Watch 4
[c] *M1*: iPhone 13 Pro Max, *M2*: Samsung Galaxy 22 Ultra
[d] https://pytorch.org/docs/stable/hub.html

Thus, these accurate but sizeable models need to be compressed before deployed. Recent research proposed several approaches to perform model compression, *e.g.* parameter pruning and quantization [16], low-rank factorization [11,28] and knowledge distillation (KD) [3]. In 2015, Han *et al.* [16] proposed Deep Compression, which applied pruning, quantization, and Huffman coding on LeNet, AlexNet and VGG-16 networks, with the max compression rate 35× (2.88% of the original size) for AlexNet and 49× (2.04% of the original size) for VGG-16 without impacting the model accuracy. In 2014, Denton *et al.* [11]

[1] https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html.

noted that, within a Convolution Neural Network (CNN), nearly 90% of the computing time are spent on conv layers, while the first several layers are the most time-consuming ones. By using low rank approximation, which decomposes the original parameter matrices with Singular Value Decomposition (SVD), the required space for matrices storage could be significantly decreased. Moreover, the decomposed matrices could be clustered, and the method reaches a final compression rate at 3.9×. In 2014, Ba and Caruana [3] proposed knowledge distillation, which trained a shallow network to mimic a deep network, without sacrificing accuracy. This method was based on one of the authors' previous work [5], which trained a small network to approximate a full network on pseudo data.

However, these methods are limited on particular networks or datasets, *e.g.* LeNet, AlexNet, *MNIST*, or even on pseudo data. Meanwhile, with the advancement of neural network theory and design, modern networks contain complex architectures and layers to extract features from large datasets. Apply the compression methods directly to complex modern networks may lead to poor performance. It is necessary to investigate the capabilities, as current researches lack the exploration of the applicability of these compression methods on modern models and datasets. Thus, we present this work, in which we apply two state-of-the-art (SOTA) compression methods on different models and datasets, provide an in-depth comparison of the method performance on different models along with different datasets. We also show the understanding of both limitations and possibilities of the model compression techniques, which indicates the research directions in the future.

To summarize, this study makes the following major contributions:

- By comprehensive experiments of two SOTA model compression techniques, we propose a general guideline for applying compression methods on modern models;
- Within the empirical study, we apply one SOTA compression method (quantization) progressively, probe the limitation, compare different clustering methods, and reveal the impact in great detail.
- We open-source our implementation on GitHub[2], which could benefit the community for future researches.

The remainder of this paper is structured as follows. In Sect. 2, we introduce the techniques we apply for the experiments. In Sect. 3, we present the empirical study results and our discussion. In Sect. 4, we review the related work. In Sect. 5, we draw the conclusion and point out our future research direction.

[2] https://github.com/paul-tian/broadnets2022-compression.

2 Methodology

2.1 Data Augmentation

To increase the generalization of deep learning models, there are two major approaches, one on models architecture, and the other on training datasets [31]. Commonly used model-side techniques add specific functions to the models, *e.g.* dropout [33] and batch normalization [21], while commonly used data augmentation (images) methods are geometric transformations, flipping, color space alteration, cropping, rotation, and noise injection [31].

As our research target is to explore the compression effectiveness on specific models, we choose to augment the training datasets to help mitigate overfitting. In our experiments, the image datasets, *Fashion-MNIST* and *CIFAR-10*, are both augmented with the procedure shown in Fig. 1.

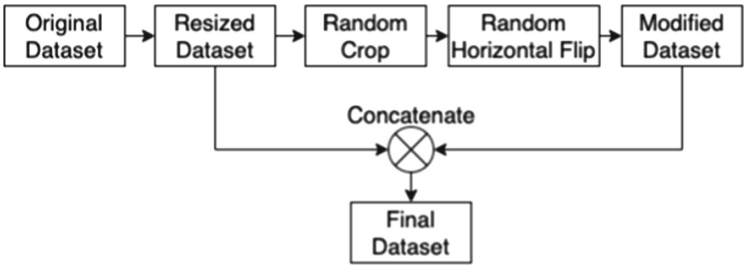

Fig. 1. Dataset augmentation procedure.

The augmentation step adopts from the practice in [18]. As our implementation of networks requires constant input shape, we first reshape the image to the required size, 63×63 for AlexNet [24], 33×33 for ResNet 18 [18], 32×32 for VGG-11 [32] with Batch Normalization [21] and VGG-19 [32] with Batch Normalization [21]. A random crop is sampled, follows by a random horizontal flip with 0.5 as the possibility. It is worth noting that the random crop is padding-enabled to maintain the required size.

2.2 Pruning

Model pruning will remove certain weights from the network, which could help reduce the amount of parameters, while keep the performance degradation in an acceptable level. The pruning method we apply to the chosen models is adopted from Han *et al.* [16]. We train the chosen models from scratch, prune these models with different thresholds, and evaluate the consequence of different compression rates.

The pruning threshold is calculated as:

$$\mathcal{T} = STD(W_l) \times Sensitivity \tag{1}$$

where $\mathcal{T}$ stands for the pruning threshold, STD stands for standard deviation, and W_l stands for weights per layer.

Sensitivity is a hyperparameter, by adjusting it, we could tune the pruning ratio and explore through different compression rates.

The pruning procedure is shown as Fig. 2.

Fig. 2. Train, prune, retrain.

2.3 Quantization

After the pruning, the weight matrices could still be compressed through quantization. We choose to perform weight sharing as a method of quantization, which will cluster weight values and use the value of centroids as the representative value for the weights in the same clustering. An example of weight sharing is shown in Fig. 3. Though there are several clustering centroid initialization methods (e.g. random [29], density-based [9], linear [7]), it has been proved that initializing the centroids with linear method could mitigate poor representation caused by the singular value [16]. However, the limitation of the linear centroids lacks discovery. Thus, we investigate the boundary of the linear centroid method in terms of compression ratio and accuracy. We also investigate modern non-linear method and conduct comparison.

The linear centroids of weights clustering can be calculated as:

$$Centroids = Cluster(2^{Q\_Bit}, W_l) \tag{2}$$

where the Q_Bit stands for the quantization bits, which is a hyperparameter, $Cluster$ stands for the clustering algorithm, and W_l stands for weights per layer.

The non-linear centroids of weights clustering can be calculated as:

$$Centroids = Cluster(W_l) \tag{3}$$

where W_l stands for weights per layer. As the centroids initialization in this non-linear method is value-based, no centroid value is required for input.

3 Experiments

3.1 Experimental Setup

Dataset: The experiments are conducted based on three widely used datasets, *Fashion-MNIST*[3] [40], *CIFAR-10*[4] [23], and *UCI-HAR*[5] [1]. ***Fashion-MNIST***

[3] https://github.com/zalandoresearch/fashion-mnist.
[4] https://www.cs.toronto.edu/~kriz/cifar.html.
[5] https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones.

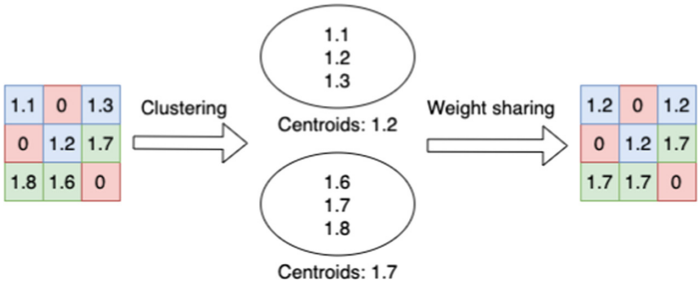

Fig. 3. An example of non-zero weight sharing. There are 6 non-zero weights in this 3×3 weight matrix. $1.1, 1.2, 1.3$ are clustered to 1.2, and $1.6, 1.7, 1.8$ are clustered to 1.7. The new weight matrix is then generated by replacing the original number with the clustered one.

dataset is an image dataset of Zalando's article images, with a training set of $60,000$ examples and a test set of $10,000$ examples. Each example is a 28×28 grayscale image, associated with a label from 10 different classes. ***CIFAR-10*** dataset consists of $60,000$ colour images, with a training set of $50,000$ examples and a test set of $10,000$ examples. Each example is a 32×32 grayscale image, associated with a label from 10 classes. ***UCI-HAR*** dataset is a human activities dataset. It is built from the recordings of 30 study participants, aged from 19 to 48, performing activities of daily living (ADL). The data are collected by a waist-mounted smartphone (Samsung Galaxy S II) with embedded inertial sensors. This dataset contains six ADL, *i.e.* WALKING, WALKINGUPSTAIRS, WALKINGDOWNSTAIRS, SITTING, STANDING and LAYING. The sampling rate of the inertial sensors (*i.e.* accelerometer and gyroscope) 50 Hz, and the labels are tagged manually with respect to the video recording. The sensor signals are pre-processed with noise-filtering and sampled through fixed-width (128 readings per window) sliding windows.

Evaluated Neural Networks: The aim of this experiment is to investigate the feasibility of the network compression method in different SOTA neural networks. The chosen neural networks are ResNet 18 [18], AlexNet [24], VGG-11 [32] with Batch Normalization [21] (**VGG11BN**), and VGG-19 [32] with Batch Normalization [21] (**VGG19BN**), which have 11.2M, 20.3M, 28.1M, 40.0M parameters, respectively.

Training: All experiments are implemented using Pytorch (version 1.11.0) with CUDA (version 11) on a single NVIDIA RTX 2080Ti GPU. The experimental hyperparameter settings are illustrated in Table 2.

Table 2. Experimental hyperparameter setting.

Hyperparameter description	Values
Number of training epochs	100
Batch size	64
Spatial learning rate[a]	0.01 with a decay rate = 0.1 per 20 epochs
Temporal learning rate[b]	0.001 with a decay rate = 0.1 per 30 epochs
Random seed	42
Optimizer	SGD with default setting
Weight initialization	Kaiming He Initialization [17]

[a] Learning rate for spatial domain datasets, *i.e. Fashion-MNIST* and *CIFAR-10*.
[b] Learning rate for temporal domain datasets, *i.e. UCI-HAR*.

3.2 Research Questions

- **RQ1: How the weight-based pruning affects the models' performance?**
  We first train the full networks from plain, then apply the weight-based pruning on the backbone networks with different pruning rates $\{-10\%, -50\%, -90\%\}$. The corresponding compression rates are $\{1.1\times, 2\times, 10\times\}$, respectively.
- **RQ2: How the various compression rates affect the models' performance if retraining the model?**
  We retrain the various pruned networks with the same training settings as the full ones.
- **RQ3: How the various quantization storage bits affect the models' performance, and what is the modern non-linear centroids initialization performance?**
  We quantize the non-zero weights with different storage bits settings (hyperparameter Q_Bit). The corresponding centroid numbers for different Q_Bit $\{4, 3, 2\}$ are $\{2^4, 2^3, 2^2\}$, respectively. We also apply a modern non-linear clustering centroid initialization (K-Means++ [2]) along with the linear clustering centroid initialization for comparison.

3.3 Experiment Results

In order to explore the impact and suitability of model compression on different networks, we perform extensive experiments on the selected neural networks. The pruning results for the three datasets, *Fashion-MNIST*, *CIFAR-10* and *UCI-HAR*, are shown in Table 3, 4, 5, respectively. The quantization results are shown in Table 6.

For **RQ1**, comparing with the full models, the performance of compressed ones degrade by 2.33% on *Fashion-MNIST* dataset, 4.25% on *CIFAR-10*, 41.57% on *UCI-HAR*, on average, with 2× models. While 73.19% on *Fashion-MNIST* dataset, 62.88% on *CIFAR-10*, 67.29% on *UCI-HAR*, on average, with 10× models. The 1.1× models show no significant difference. This phenomenon shows that pure pruning could still keep the model performance until 2×, but models for temporal domain datasets are more vulnerable to pruning.

Table 3. Pruning result comparison, *Fashion-MNIST*.

Network	*Sensitivity*[a]	#Params	Comp. Rate[b]	Acc-AP[c]	Acc[d]
ResNet18	–	11.2M	1×	–	93.27
	0.111	10.1M	1.1×	93.26	93.27
	0.609	5.6M	2×	90.81	93.10
	1.590	1.1M	10×	29.04	93.05
AlexNet	–	20.3M	1×	–	92.91
	0.127	18.3M	1.1×	92.89	92.90
	0.671	10.2M	2×	89.01	92.78
	1.510	2.0M	10×	21.61	92.45
VGG11BN	–	28.1M	1×	–	94.39
	0.095	25.3M	1.1×	94.31	94.26
	0.528	14.1M	2×	92.00	94.23
	1.499	2.8M	10×	17.92	94.22
VGG19BN	–	40.0M	1×	–	93.98
	0.102	35.0M	1.1×	93.97	93.98
	0.557	19.5M	2×	93.40	93.85
	1.510	3.9M	10×	13.22	93.61

[a] The pruning sensitivity.
[b] The compression rate.
[c] The accuracy after pruning without retrain. (**RQ1**)
[d] The accuracy after training (**full network**)/retraining (**pruned network**). (**RQ2**)

It is also noticeable that, though pruning will lead to model performance degradation, different pruning rates will lead to opposite performance. On spatial domain datasets (*Fashion-MNIST* and *CIFAR-10*), VGG19BN, the biggest network among the 4 chosen networks, shows the least degrade (less than 1%) for 2× compression, while the largest degrade (more than 78%) occurs for 10× compression. In contrast, ResNet18, the smallest network, shows the largest degradation (more than 4%, the same as AlexNet) for 2× compression, while the least degradation (less than 63%) for 10× compression. We believe that for large networks with considerable feature extraction layers, the stability could maintain when pruning. But when certain portions of the weights are removed, the performance will drastically drop. For residual networks, the feature extraction layers are shallow, but the structure makes them more restrainable when a large number of weights are pruned (*i.e.* 90%). In other words, large deep networks are insensitive but vulnerable to pruning. Meanwhile, residual networks are sensitive but robust for pruning. On the temporal domain dataset (*UCI-HAR*), though AlexNet shows the least degradation among all chosen networks, all four networks show obvious degradation starting from 2× compression. We believe that models trained for temporal domain datasets are more fragile and susceptible to pruning.

For **RQ2**, from Table 3, 4, 5, after retraining the pruned networks, the performance of all compressed networks (*i.e.* $1.1\times, 2\times, 10\times$) shows no noticeable

Table 4. Pruning result comparison, *CIFAR-10*.

Network	*Sensitivity*[a]	#Params	Comp. Rate[b]	Acc-AP[c]	Acc[d]
ResNet18	–	11.2M	1×	–	83.88
	0.111	10.1M	1.1×	83.17	83.81
	0.633	5.6M	2×	77.55	83.51
	1.437	1.1M	10×	42.37	83.21
AlexNet	–	20.3M	1×	–	82.39
	0.137	18.3M	1.1×	82.16	82.39
	0.718	10.2M	2×	76.49	82.38
	1.520	2.0M	10×	18.43	81.98
VGG11BN	–	28.1M	1×	–	88.76
	0.100	25.3M	1.1×	88.46	88.73
	0.552	14.1M	2×	85.29	88.67
	1.510	2.8M	10×	19.94	88.14
VGG19BN	–	40.0M	1×	–	90.33
	0.107	35.0M	1.1×	89.80	90.23
	0.582	19.5M	2×	89.01	90.22
	1.535	3.9M	10×	13.11	89.75

[a] The pruning sensitivity.
[b] The compression rate.
[c] The accuracy after pruning without retrain. (**RQ1**)
[d] The accuracy after training (**full network**)/retraining (**pruned network**). (**RQ2**)

Table 5. Pruning result comparison, *UCI-HAR*.

Network	*Sensitivity*[a]	#Params	Comp. Rate[b]	Acc-AP[c]	Acc[d]
ResNet18	–	11.2M	1×	–	86.97
	0.126	10.1M	1.1×	85.44	85.51
	0.684	5.6M	2×	51.68	84.43
	1.709	1.1M	10×	16.66	82.80
AlexNet	–	20.3M	1×	–	83.17
	0.175	18.3M	1.1×	80.69	83.10
	0.888	10.2M	2×	68.88	82.15
	1.601	2.0M	10×	23.82	71.71
VGG11BN	–	28.1M	1×	–	85.44
	0.126	25.3M	1.1×	84.70	84.80
	0.678	14.1M	2×	32.85	84.36
	1.668	2.8M	10×	14.25	84.05
VGG19BN	–	40.0M	1×	–	84.29
	0.126	35.0M	1.1×	84.19	84.19
	0.678	19.5M	2×	20.20	83.37
	1.663	3.9M	10×	15.98	81.68

[a] The pruning sensitivity.
[b] The compression rate.
[c] The accuracy after pruning without retrain. (**RQ1**)
[d] The accuracy after training (**full network**)/retraining (**pruned network**). (**RQ2**)

or tolerable degradation (average degrade is less than 0.7% on spatial domain datasets and 4.9% on temporal domain datasets, maximum 0.62% and 11.46%, respectively). This reflects that retraining is essential for high compression rates (*e.g.* 10×), and applying retraining step could keep the overall accuracy.

For **RQ3**, from Table 6, the performance degradation for 4 quantization bits (Q_Bit {4}) is not noticeable (average less than 1%, minimum 0.04% and maximum 1.92%). For Q_Bit {2}, all the evaluated neural networks show fatal performance degradation, which indicates that quantization with {2} (*i.e.* 2^2, or 4 unique numbers as clustering centroids) cannot maintain a usable status for model quantization. It is worth noting that, for deep networks (*i.e.* ResNet18 and VGG19BN), the performance of quantization correlates to the pruning rate. The higher the pruning rate, the higher the quantization performance. We believe the reason is that, small-value weights will become the noise for clustering and degrade the performance worse. The increasing pruning rate can reduce the noise and help mitigate the performance degradation. On the contrary, for shallow networks (*i.e.* AlexNet and VGG11BN), the performance of quantization is related not only to the pruning rate, but also to the Q_Bit value. When Q_Bit is lower than {4}, the quantization performance is in inverse proportion to the pruning rate. When Q_Bit equals {4}, it shows a similar characteristic as deep networks. We believe the reason is that, shallow networks contain fewer layers and the distinctiveness of parameters is the key to maintaining the performance. Applying quantization with fewer clustering centroids will destroy the distinctiveness and degrade the performance. While deep networks contain enough layers, which is more robust for the distinctiveness loss after the quantization. It is also worth noting that, ResNet18 performs the best among all four networks. As ResNet18 contains residual block (RESBLOCK), which might be the key part for maintaining the accuracy.

4 Related Work

Since AlexNet [24] won the 2012 ImageNet competition, Convolutional Networks have become more accurate by growing larger. However, deep Convolutional Neural Networks are often overparameterized, which has led to researchers attempting to reduce model size by trading accuracy for efficiency.

Handcraft Efficient Mobile-Size Convolutional Networks: It is a common solution to handcraft efficient mobile-size Convolution Networks, such as SqueezeNets [20], and ShuffleNets [42]. One of the most famous models is MobileNet [19], which proposes depthwise separable convolutions to build light weight deep neural networks. This work introduces two simple global hyperparameters that efficiently trade off between latency and accuracy. These hyperparameters allow the model builder to choose the right sized model for their application based on the constraints of the problem. Comparing with model compression, this solution is also possible to deploy a high-accurate neural network on IoT devices, which could be an efficient way for solving particular tasks. However, to design a network from scratch requires considerable time and com-

Table 6. Quantization result comparison.

AlexNet				ResNet18			
P-Rate[a]	*Q_Bit*[b]	Deg-L[c](%)	Deg-NL[d](%)	P-Rate[a]	*Q_Bit*[b]	Deg-L[c](%)	Deg-NL[d](%)
1.1×	2	23.74	20.90	1.1×	2	62.70	68.43
	3	3.48	5.34		3	8.19	9.68
	4	1.04	0.77		4	0.17	0.50
2×	2	23.07	36.62	2×	2	47.61	58.43
	3	3.51	4.45		3	2.29	3.28
	4	0.28	0.65		4	0.67	0.68
10×	2	32.63	42.07	10×	2	43.79	39.12
	3	6.42	7.45		3	1.92	3.02
	4	0.30	0.09		4	0.19	0.66

VGG11BN				VGG19BN			
P-Rate[a]	*Q_Bit*[b]	Deg-L[c](%)	Deg-NL[d](%)	P-Rate[a]	*Q_Bit*[b]	Deg-L[c](%)	Deg-NL[d](%)
1.1×	2	62.41	73.75	1.1×	2	57.22	77.55
	3	18.69	16.72		3	34.84	35.55
	4	0.94	0.55		4	0.12	1.32
2×	2	48.37	75.19	2×	2	77.33	77.33
	3	2.66	2.08		3	12.21	16.32
	4	0.24	0.04		4	0.42	0.29
10×	2	52.80	67.86	10×	2	71.38	77.09
	3	2.94	3.44		3	3.46	5.18
	4	0.31	0.50		4	0.06	0.04

[a] The model of corresponding compression rate from pruning.
[b] The quantization bit for storing the shared weight.
[c] The average accuracy degrade for linear clustering centroid initialization.
[d] The average accuracy degrade for non-linear clustering centroid initialization

puting resources, while model compression could directly leverage existing, high-maturity neural networks with small efforts.

Neural Architecture Search for Efficient Mobile-Size Convolutional Networks: Recently, neural architecture search has become increasingly popular in the design of efficient mobile-sized Convolutional Networks [6,35], and can achieve even better performances than hand-crafted mobile Convolutional Networks by carefully tuning the width, depth, size, and type of convolution kernels. EfficientNet [36] is one of the typical models which scales the networks to small their size. EfficientNet uses fixed scaling coefficients to uniformly scale width, depth, and resolution of networks. It is noteworthy that their scaling method can also be applied to MobileNet and ResNet. Compared with other single-dimension scaling methods [18,19,41], their compound scaling method performs better on

all of these models. It is also possible to combine this method with model compression for even smaller computing requirement and higher efficiency (*e.g.* we apply the compression on ResNet in the experimentation).

Others Researches on Model Compression: In 2020, Yu et al. [8] reviewed recent model compression techniques, and classified them as **Pruning**, **Quantization**, **Low-rank Approximation**, and **Knowledge Distillation**. Low rank approximation could decompose the original parameter matrices with SVD and decrease the required matrices storage space, while knowledge distillation researches the target by training a shallow network to mimic a deep one, instead of altering the original network itself.

Other Researches on Splitting Models: As edge devices play an vital role in IoT, one possible methodology of deploying the deep neural networks on them are model splitting. Inference models are spitted by layers, and the execution framework will schedule the layers execution and transfer the layer output through internet connection [15,22,38]. By leveraging the strong cloud computing abilities and internet connection (*e.g.* Wi-Fi, LTE and 5G), model splitting could significantly improve the network efficiency, comparing with pure local computing or cloud computing.

5 Conclusion

It this work, we practice the pruning and quantization compression empirical study towards a popular SOTA method. We reveal the advantages of the SOTA method, *e.g.* pruning 10× with no noticeable performance degrade (less than 0.7%) and quantization with 2^3 with acceptable degrade (less than 5.0%) on spatial domain datasets (*i.e. Fashion-MNIST* and *CIFAR-10*); while the performance temporal domain datasets (*i.e. UCI-HAR*) degrades more (maximum 15.55%). Meanwhile, we compare different clustering algorithms and improve the performance (maximum 0.5%, on AlexNet 10×). We have open-sourced these experimentation codes on GitHub[6]. Based on the well performance of graph neural networks on non-euclidean distance datasets, we plan on further explore the compatible compression methods towards graph neural networks. It is worth noting that, in recent years, other methods for deploying neural networks on IoT devices appear, from both compression aspect (*e.g.* knowledge distillation, low-rank approximation and transfer learning) and execution aspect (*e.g.* model splitting). These new approaches provide us a horizon of researching. Meanwhile, as transformer networks (i.e. [12,37]) also contain residual blocks as ResNet18, it is possible to apply model compression on transformer networks with high P-Rate (*e.g.* 10×) and $Q\_Bit\{4\}$ to reach a high compression rate while maintaining the performance. We also plan on further explore the boundaries of these methods.

[6] https://github.com/paul-tian/broadnets2022-compression.

Acknowledgements. This work is in part supported by an Australian Research Council (ARC) Discovery Project (DP210102447), an ARC Linkage Project (LP190100676), and a DATA61 project (Data61 CRP C020996).

References

1. Anguita, D., Ghio, A., Oneto, L., Parra Perez, X., Reyes Ortiz, J.L.: A public domain dataset for human activity recognition using smartphones. In: Proceedings of the 21th International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, pp. 437–442 (2013)
2. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. Technical report, Stanford (2006)
3. Ba, J., Caruana, R.: Do deep nets really need to be deep? Adv. Neural Inf. Process. Syst. **27** (2014)
4. Bhandari, B., Lu, J., Zheng, X., Rajasegarar, S., Karmakar, C.: Non-invasive sensor based automated smoking activity detection. In: 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 845–848. IEEE (2017)
5. Buciluǎ, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 535–541 (2006)
6. Cai, H., Zhu, L., Han, S.: ProxylessNAS: direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332 (2018)
7. Celebi, M.E., Kingravi, H.A.: Linear, deterministic, and order-invariant initialization methods for the K-means clustering algorithm. In: Celebi, M.E. (ed.) Partitional Clustering Algorithms, pp. 79–98. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-09259-1_3
8. Cheng, Y., Wang, D., Zhou, P., Zhang, T.: A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282 (2017)
9. Dalhatu, K., Sim, A.T.H.: Density base k-mean's cluster centroid initialization algorithm. Int. J. Comput. Appl. **137**(11) (2016)
10. Deep, S., Tian, Y., Lu, J., Zhou, Y., Zheng, X.: Leveraging multi-view learning for human anomaly detection in industrial internet of things. In: 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), pp. 533–537. IEEE (2020)
11. Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. Adv. Neural Inf. Process. Syst. **27** (2014)
12. Dosovitskiy, A., et al.: An image is worth 16×16 words: transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
13. Drias, Z., Serhrouchni, A., Vogel, O.: Analysis of cyber security for industrial control systems. In: 2015 International Conference on Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC), pp. 1–8. IEEE (2015)
14. Fujii, K., Higuchi, K., Rekimoto, J.: Endless flyer: a continuous flying drone with automatic battery replacement. In: 2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing, pp. 216–223. IEEE (2013)

15. Han, S., Shen, H., Philipose, M., Agarwal, S., Wolman, A., Krishnamurthy, A.: MCDNN: an approximation-based execution framework for deep stream processing under resource constraints. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, pp. 123–136 (2016)
16. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv preprint arXiv:1510.00149 (2015)
17. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
19. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
20. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016)
21. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456. PMLR (2015)
22. Kang, Y., Hauswald, J., Gao, C., Rovinski, A., Mudge, T., Mars, J., Tang, L.: Neurosurgeon: collaborative intelligence between the cloud and mobile edge. ACM SIGARCH Comput. Archit. News **45**(1), 615–629 (2017)
23. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009)
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. **25** (2012)
25. Lu, J., Wang, J., Zheng, X., Karmakar, C., Rajasegarar, S.: Detection of smoking events from confounding activities of daily living. In: Proceedings of the Australasian Computer Science Week Multiconference, pp. 1–9 (2019)
26. Lu, J., Zheng, X., Sheng, M., Jin, J., Yu, S.: Efficient human activity recognition using a single wearable sensor. IEEE Internet Things J. **7**(11), 11137–11146 (2020)
27. Lu, J., et al.: Can steering wheel detect your driving fatigue? IEEE Trans. Veh. Technol. **70**(6), 5537–5550 (2021)
28. Lu, Y., Kumar, A., Zhai, S., Cheng, Y., Javidi, T., Feris, R.: Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5334–5343 (2017)
29. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Oakland, CA, USA, vol. 1, pp. 281–297 (1967)
30. Mammadova, M., Jabrayilova, Z.: Conceptual approaches to IoT-based personnel health management in offshore oil and gas industry. Control Optim. Industr. Appl. 257 (2020)
31. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. J. Big Data **6**(1), 1–48 (2019)
32. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
33. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)

34. Sung, W.T., Hsu, Y.C.: Designing an industrial real-time measurement and monitoring system based on embedded system and ZigBee. Expert Syst. Appl. **38**(4), 4522–4529 (2011)
35. Tan, M., et al.: MnasNet: platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2820–2828 (2019)
36. Tan, M., Le, Q.: EfficientNet: rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning, pp. 6105–6114. PMLR (2019)
37. Vaswani, A., et al.: Attention is all you need. Adv. Neural Inf. Process. Syst. **30** (2017)
38. Wang, S., Zhang, X., Uchiyama, H., Matsuda, H.: Hivemind: towards cellular native machine learning model splitting. IEEE J. Sel. Areas Commun. **40**(2), 626–640 (2021)
39. Wang, T., et al.: Mobile edge-enabled trust evaluation for the internet of things. Inf. Fusion **75**, 90–100 (2021)
40. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms (2017)
41. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
42. Zhang, X., Zhou, X., Lin, M., Sun, J.: ShuffleNet: an extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6848–6856 (2018)