An Empirical Study on Model Pruning and Quantization

A thesis submitted in Fulfillment
of the Requirements for the Degree of
Master of Research

by

Yuzhe Tian

Supervisor: Dr. Xi Zheng

MACQUARIE
University

School of Computing
Faculty of Science and Engineering
Macquarie University, NSW 2109, Australia

Submitted November 2022

Declaration

I certify that the work in this thesis entitled **"An Empirical Study on Model Pruning and Quantization"** has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree to any other university or institution other than Macquarie University. I also certify that the thesis is an original piece of research and it has been written by me. Any help and assistance that I have received in my research work and the preparation of the thesis itself have been appropriately acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signed: ... .

Date: November 17, 2022.

Acknowledgements

First, and the most important, I would like to express my appreciation to my supervisor, Dr James Xi Zheng, who provided a lot of knowledge and direction to me. This study couldn't be done without his guidance. It is my honour to have him as my supervisor.

Meanwhile, I would like to thank Jianchao Lu, Yufei Wang, Yao Deng and Jiwei Guan. As current PhD candidates, they all inspired me in my experimentations and provided useful suggestions with their knowledge and experience.

And all the lecturers and tutors during my coursework period, their courses provided me with knowledge and inspiration, which helped me a lot during my research.

To my passed grandfather.
May him have eternal peace.

I write down this thesis with love to my motherland, the People's Republic of China. It is a great nation, constructed by great people, who are diligent and brave.

I would like to thank my family. The support they provide, both economic and mental, are the basis of not only my research but more importantly, my life. No matter where I am, they always accompany me in my heart.

As an international student, I appreciate the support from Siying Ren, Kexin Du, and Jiale Zhu. Their companionship grants me strength during my study, when I was on a foreign continent and far away from motherland. I would like to mention Ke Xu and Yufeng Xin as well. They are my undergraduate fellows and friends. May our friendship last.

It is worth noting that, the *LoveLive!* projects act as my spiritual support during my study. From *LoveLive!* project to *LoveLive! Superstar!!* project, from *μ's* group to *Liella!* group, the idols' passion and efforts did delivered to me through their songs and dances, which inspired me when I was down. Please allow me to express my thankfulness to them here, "今までありがとう！". In particular I would like to mention Miss Akina Homoto, the actress of character Lanzhu Zhong (鐘嵐珠), her persistence encouraged me to face the difficulties of life. Woof.

Last but not least, I thank Steve Jobs and Linus Torvalds. The great works they founded, the Apple Inc. and Linux Operating System, provided infinite possibility to the world. The marvellous devices created by Apple, and the Linux OS, greatly empowered me through my research time. All the direct outputs during this study are generated on Apple devices and Linux-enabled computers.

Abstract

In machine learning, model compression is vital for resource-constrained Internet of Things (IoT) devices, such as unmanned aerial vehicles (UAVs) and wearable devices. 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, network architecture, and clustering methods. We also conduct comparative experiments on knowledge distillation. The result indicates that more prerequisites need to be satisfied when using the knowledge distillation to achieve average performance.

Finally, we provide some interesting directions for future research.

Contents

List of Publications

- Samundra Deep, **Yuzhe Tian**, Jianchao Lu, Yipeng Zhou and Xi Zheng, *Leveraging Multi-view Learning for Human Anomaly Detection in Industrial Internet of Things*. 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), 533-537 (2020)
- Jianchao Lu, **Yuzhe Tian**, Shuang Wang, Michael Sheng and Xi Zheng, *PearNet: A Pearson Correlation-based Graph Attention Network for Sleep Stage Recognition*. IEEE International Conference on Data Science and Advanced Analytics, Accepted, (2022)
- **Yuzhe Tian**, Hao Luan and Xi Zheng, *An Empirical Study on Model Pruning and Quantization*. EAI International Conference on Broadband Communications, Networks, and Systems, Submitted, (2022)

List of Figures

List of Tables

Chapter 1

Introduction

"故　常無欲　以觀其妙　常有欲　以觀其徼"

Lao Tzu - Tao Te Ching

1.1 General Introduction

With the development of deep neural network (DNN), many success models are proposed, and DNNs have already became an important part of modern society, *e.g.* face recognition [37], electricity power grid management [77] and autonomous driving [75]. It is worth noting that, in order to improve the model accuracy, modern DNN usually contain dozens to hundreds of layers, *e.g.* MobileNet [36] and Vision Transformer (ViT) [20]. 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 a commonly used pre-trained model VGG19 [62] is 550MB. As a result, leveraging these models with high accuracy requires sufficient computing resources. Thus, most of them are deployed on edge computing (EC), fog computing (FC) and cloud computing (CC).

On the other hand, accompanied with the development of Internet of Things (IoT), the amount of deployed IoT devices and edge devices keeps growing [71]. These devices are commonly used around our daily lives. From smart homes to unmanned vehicles, from factory streamline to ocean oil platforms, IoT devices are deployed in every corner. This kind of usage has determined the design concept

of these devices, that they would require a long period of deployment within unattended environments. Directed by this concept, longer battery life becomes the first priority. Restricted by contemporary and foreseeable future technology and industrial process limitations, the only way to reach maximum battery life is to sacrifice the computing resources. Mainstream strategies to reduce power consumption is by:

- using single-level cell (SLC) flash memory for data storage instead of multi-level cell (MLC) or triple-level cell (TLC) to minimize electricity consumption during writing procedure [54];
- using small size random access memory (RAM) to decrease the energy cost during the memory refresh, which is the regular data maintaining procedure of RAM [26];
- using low central processing unit (CPU) clock speed to help reduce the power usage [7].

It is a last resort to leverage every aspect of requirements, but also leads to reduced computing abilities. Table A.1 shows a comparison of mainstream IoT MCU devices and their detailed specifications.

Meanwhile, the IoT devices will produce a large amount of raw and real-time data. According to IBM research[1], A large refinery is able to generate IoT sensor data for 1 TeraByte (TB) per day, while one offshore oil platform could generate several TBs. These data could only be handled well by DNN [9].

However, due to the resources requirement of conventional DNN, most of them are deployed on cloud or fog servers. Leveraging existed servers not only requires stable internet connection and sufficient bandwidth to transit the data, but also consumes a considerable amount of power. These requirements are not always

[1] https://www.ibm.com/blogs/internet-of-things/energy-industry/

possible to satisfy, and is against the real-time processing, power efficiency and network tolerance requirements of IoT devices. For instance, unmanned aerial vehicles (UAVs) require power efficiency to conduct long term tasks without power recharge [24]. Offshore oil platforms usually contain considerable IoT devices but the devices' connection to the Internet is limited, or require large investigation to use the internet, e.g. satellite service [53]. Industrial control systems require real-time response for monitoring and security [21, 65]. All these scenarios require the IoT data to be processed on the premises to minimize the power consumption, Internet usage and time delay. Not only the IoT devices, modern edge devices, *e.g.* micro controller units (MCUs) and smart wearable devices, are also limited by their battery capacity and computing resources. Table 1.1 shows a comparison of commonly used model sizes against mainstream edge devices memory capacity.

Table 1.1: Model size / hardware memory comparison

Model		Memory Capacity								
Name	Size[2]	*B1*[a]	*B2*[a]	*B3*[a]	*SW1*[b]	*SW2*[b]	*G1*[c]	*G2*[c]	*M1*[d]	*M2*[d]
ResNet18	$\approx 45MB$									
ResNet50	$\approx 100MB$									
AlexNet	$\approx 235MB$	32KB	32KB	8MB	1GB	1.5GB	3GB	4GB	6GB	8GB
VGG11	$\approx 500MB$									
VGG19	$\approx 550MB$									

[a] *B1*: Arduino MKR1010, *B2*: Arduino Portenta H7, *B3*: Raspberry Pi Zero
[b] *SW1*: Apple Watch Series 8, *SW2*: Samsung Galaxy Watch 4
[c] *G1*: Google Glass Enterprise Edition 2, *G2*: Microsoft HoloLens 2
[d] *M1*: iPhone 13 Pro Max, *M2*: Samsung Galaxy 22 Ultra

Comparing with the raw model size, the RAM size of IoT MCUs (Arduino and

[2] https://pytorch.org/docs/stable/hub.html

Raspberry Serials) is far from enough for the deployment. For portable IoT devices (*i.e.* smart watches, smart glasses and smart phones), though the memory is enough for executing the network, occupying this considerable amount of memory is bound to lock memory space from other user-end softwares, which will lead to the degradation of user experience.

Thus, to deploy these accurate but sizable models onto IoT devices, it is a must to do pre-processing for both successful inference execution and promising user experience.

1.2 Recent Research

Recently, several related approaches were proposed, which attract our attention. Some of them are not directly designed for solving the conflict between strong computing requirements of intelligent application and low computing abilities, but provided a workaround solution which could help to deploy deep neural network (DNN) on IoT devices [14, 59, 73]. While some of them are aimed for deploying DNN on IoT devices and held this ideology though out the approaches. Among them the model compression (MC) and tiny maching learning (TinyML) are notice-worthy.

1.2.1 Internet of Things Enabled by Machine Learning

By connecting the astonishing amount of IoT objects, sensors and devices, IoT combined with machine learning (ML) has successfully been used not only in industrial area, but also been adapted by medical area and consumer grade area. For instance, Chun et al. [16] designed a hardware set, which could detect eating episodes by tracking the users' jawbone movement, and could provide more than 18 hours of continually working time. Jiang et al. [40] successfully achieved human walking monitoring with high accuracy and power efficiency in both in-door and out-door environments on smart watches. This level of accuracy could only achieved

by using cumbersome equipment in the past, which would require sufficient budget while affecting users' daily routine. The development of IoT and ML has successfully helped reaching the miniaturization and affordability.

The build-in high sensitivity attributes of IoT could also help managing manufacturing [61] and support the power distribution of electric power grid [69].

1.2.2 Model Compression

The approaches of model compression could be classified as four aspects, parameter pruning [30, 63], quantization [28, 57, 80], low-rank factorization [19, 51] and knowledge distillation [6].

Srinivas and Babu [63] proposed a systematic way for removing similar neurons in the network In 2015, which removed 35% for AlexNet, while still maintaining the performance of the network in the same level.

Gupta *et al.* [28] applied an parameter quantization method, which limited the data precision while representing and computing. Their results showed that DNN is endurable for quantization, which incur little or even no degradation for classification tasks.

Han *et al.* [30] proposed Deep Compression in 2015, which applied pruning, quantization, and Huffman coding on LeNet, AlexNet and VGG-16 networks. They successfully reached 35× compression rate (2.88% of the original size) for AlexNet at maximum, and 49× (2.04% of the original size) for VGG-16 at maximum, without impacting the model accuracy.

In 2014, Denton *et al.* [19] noted that, within a Convolution Neural Network (CNN), nearly 90% of the computing time are spent on convolution 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×.

Ba and Caruana [6] proposed knowledge distillation the same year. The main idea of this method is to train a shallow network by mimicking a deep network, which could reach the same level of accuracy. This method was based on one of the authors' previous work [10], which trained a small network to approximate a full network on pseudo data.

1.2.3 TinyML

Considering model compression, which act as a top-bottom strategy, is to modify and adjust current neural networks, which are trained on computers with sufficient computing resources, try to fit these networks inside devices with much less computing power.

Though using current networks that are been proved as success could help minimize network developing cost and cut down the fine tuning compression the models is not always possible, especially when the target model contains complex structures and blocks.

On the contrary, TinyML is a bottom-top strategy, which first consider the capacity of the target IoT devices and provide the well-adjusted framework. By this the framework will be suitable for devices with less computing resources, especially MCUs.

Unlike DNN, which has a long history and different significant contributions for the theory build-up, TinyML is a concept, which appears accompanied with the increasing urgent requirement of IoT environment machine learning deployment, and is still in its nascent stage [58].

The target IoT devices are typically enegry efficient with miliwatt level of power consumption. It is a challenge to deploy conventional machine learning approaches like Random Forest or support vector machine (SVM), not even mention about executing neural network models. By adopting TinyML, it is possible to provide

intelligence to every IoT devices, which could help decrease latency for and increase the reliability [22].

In the foreseeable future, the demands for pervasive IoT devices accompanied with edge computing IoT will keep increasing, and these demands could not be fulfilled without TinyML.

1.3 Research Gaps

However, these methods are limited within several aspects, *e.g.* on particular networks (*i.e.* LeNet [46], AlexNet [43]) or datasets (*i.e.MNIST* [45]). The generalization and compatibility is yet not revealed on modern deep networks, *e.g.* VGG network family [62] and ResNet networks family [34].

Meanwhile, with the advancement of neural network theory and design, modern networks contain complex architectures and layers to extract features from large datasets, *e.g.* residual block [34] which aimed for avoiding the vanishing gradient problem. Apply the compression methods directly to complex modern networks may lead to poor performance.

It is necessary, and urgently needed, to investigate the capabilities, as current researches lack the exploration of the applicability of these methods on modern models and datasets. Among them the compression ones are the most prominent, as compression could utilize current existing models, which have already been proved as success, and accelerate the deployment of DNN with IoT.

Thus, we present this study, in which we apply two state-of-the-art (SOTA) compression methods, *i.e.* pruning and quantization, on different models and datasets, provide an in-depth comparison of the method performance on different models along with different datasets. We also attempt the knowledge distillation method as a comparison baseline.

Within this study, we also show the understanding of both limitations and

possibilities of the model compression techniques, which indicates the research directions in the future.

1.4 Contributions of the Work

In order to address the research gaps we pointed out, we go through a comprehensive research which could help bridging the gap between previous research and the urgent increasing demand from IoT areas. We also point out the future research directions for model compression within IoT scenarios.

This study makes four major contributions as following:

- By comprehensive experiments of two SOTA model compression techniques, we propose a general guideline for applying compression methods on modern models. According to the best of our knowledge, this kind of guideline is one of the first been proposed.

- For the quantization experimentation within the empirical study, we apply the quantization progressively, probe the limitation and compare different clustering methods. By this exploration, we provide the boundaries for quantization and reveal the impact in great detail.

- We conduct knowledge distillation experiments and compare the result with pruning and quantization. From the comparison, we draw the conclusion that knowledge distillation has prerequisites while pruning and quantization are relatively universal.

- We open-sourced our implementation on GitHub[3]. We believe that by making our research results accessible to the public, the community could be benefited, not only from having solid experimentation as reference, but also from gaining a clearer vision for guiding the future researches.

[3] https://github.com/broadnet2022-compression/code

1.5 Roadmap of the Thesis

The remainder of this paper is structured as follows.

- In Chapter 2, we review the related work among the area of:
  - Internet of Things (IoT);
  - Model Compression, including *a)* Pruning, *b)* Low Rank Factorization, and *c)* Knowledge Distillation;
  - TinyML.
- In Chapter 3, we introduce the techniques we apply for the experiments and present the empirical study results with discussion.
- In Chapter 4, we draw the conclusion and point out our future research direction.
- In Appendix A, full comparison and experimentation results are presented as tables.

Chapter 2

Literature Review

駆け抜けるシューティングスター
追いかけて星になる
止まらない　止まれないよ　まだちいさくても

KuuKaa/Liella! - Tiny Stars

This chapter goes through the past literatures, which helps us to build up the appropriate theoretical and methodological foundations. It is also crucial that only by reviewing the past researches we could have a better understanding of the underlying ideas and concepts. From within the research gap could been found, and form up into research questions for experimentation guidance.

2.1 Internet of Things Enabled by Machine Learning

The terminology of Internet of Things (IoT) has a rather long history. Accompanied by the development of semi-conductor manufacturing technology and the widely usage of cloud computing, IoT has successfully entered into every corners of the society, including industrial area, medical area, and consumer grade area.

The three layer structure of IoT is a widely accepted view, that IoT is constructed by perception layer, network layer and application layer, which complete the data collection, transmission, and presentation, respectively. This eventually became a consensus for both industrial and academic [79]. Fig.2.1 shows the technical architecture of *De facto* IoT framework.

By leveraging different cutting-edge transmission technologies, *e.g.* Zigbee, Wi-Fi, 5G and LoRa IoT has already been widely used in industrial areas and has

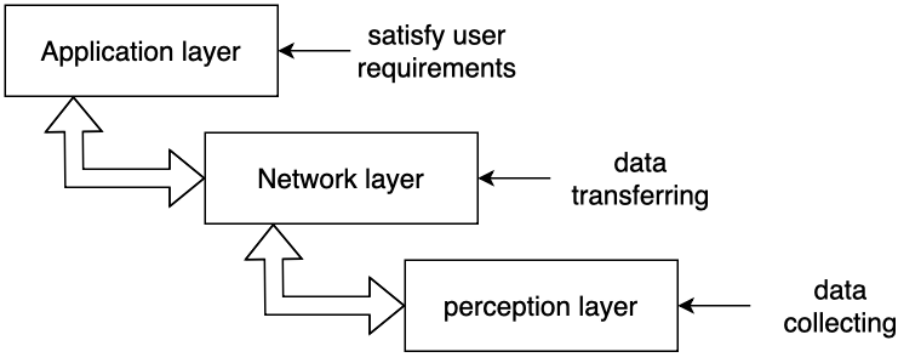

Fig. 2.1. The architecture of IoT from technical view

greatly increased the efficiency and performance[1].

The combination of IoT and machine learning (ML) has also achieved great success. For instance, in 2018, Chun et al. [16] designed a hardware set for detecting eating episodes by tracking jawbone movement. This set includes a proximity sensor, a Bluetooth LE module and a micro-controller, and could continually work for more than 18 hours. By leveraging the smartwatch and well designed algorithm, Jiang et al. [40] successfully achieved pedestrians' movements tracking in both indoor and outdoor scenarios, and reach high accuracy in 2019. In the past, these accurate tracking could only be done by expensive medical equipment, which usually presented as cumbersome machines and is hard to maintain tracking procedure while not affecting users' daily life. With the help of high accuracy sensors, transmission modules and edge devices, it is possible to approach the medical-level accuracy with consumer-level gadgets. These successful research approaches, *i.e.* miniaturised tracking system and monitoring framework, could only be done with the IoT and ML.

Meanwhile, unmanned vehicles and unmanned aerial vehicles (UAVs) are also benefited. The implementation of computing vision (CV) on on-board computer of cars and trucks could help achieving autonomous driving without using expensive

[1] https://www.ge.com/digital/blog/what-industrial-internet-things-iiot

LiDAR [25]. By leveraging ML, it is possible for the UAVs to distinguish rooftops, trees and open spaces, which could be used for automated landing [56].

IoT and ML could also provide more time-sensitive data and information than conventional methodologies, which could help large scale managing. For instance, they could empower manufacturing of the factory with visibility and flexibility, which helps reducing energy and material wastes and improving overall efficiency [61]. They could also help supervising electric power grid with automatic dispatching and fail-safe electric re-routing [69].

It is worth noting that, the base of IoT, *i.e.* the sensors and other real world objects, might be deployed in harsh environment and contains limited computing resources. For instance, Arduino's business grade IIoT controller, Nicla Sense ME[2], contains a 64MHz CPU with 64KB RAM and 512KB of storage. The environment of deployment decided that the hardware will focus on low power consumption and long battery life, instead of strong computing ability. Thus, the design of IoT-based framework should consider this situation as an important part. It is vital to reduce the power consumption, and the algorithm should be as lightweight as possible while keeping the accuracy.

In 2020, Lin *et al.* [50] proposed a framework MCUNet, which contains a neural architecture, TinyNAS, and an inference engine, TinyEngine. Instead of using interpreter-based inference libraries like TF-Lite Micro [1], TinyEngine will and will only compile the codes that will be executed. This not only reduced the execution time, but also cut the RAM requirement for the framework. MCUNet could reach 3x faster execution speed and about 3x smaller RAM usage. In 2021, Cox *et al.* [17] proposed MASA, which is a multi-DNN inference that could be de- ployed on edge. This research has proved the possibility of archiving low latency responsibility with tight computing resources.

Lin *et al.* from MIT-IBM Watson AI Lab proposed MCUNet [50] in 2020, which

[2] https://docs.arduino.cc/hardware/nicla-sense-me

is a framework build for Micro-controller Unit (MCU). By designing the compiler and runtime simultaneously, MCUNet used a compiled version of inference libraries instead of interpreter version like TF-Lite Micro. This could cut the runtime binary size, lower the memory usage, and accelerate the execution. MCUNet outperformed TF-Lite Micro [1], MicroTVM [13] and CMSIS-NN [44]. They also improved the MCUNet and released a second version in 2021[49].

In 2021, Cox *et al.* proposed *MASA* [17], a multi-DNN execution framework which could be deployed on small memory devices and cut the latency. Fig.2.2 shows the architecture of *MASA*.

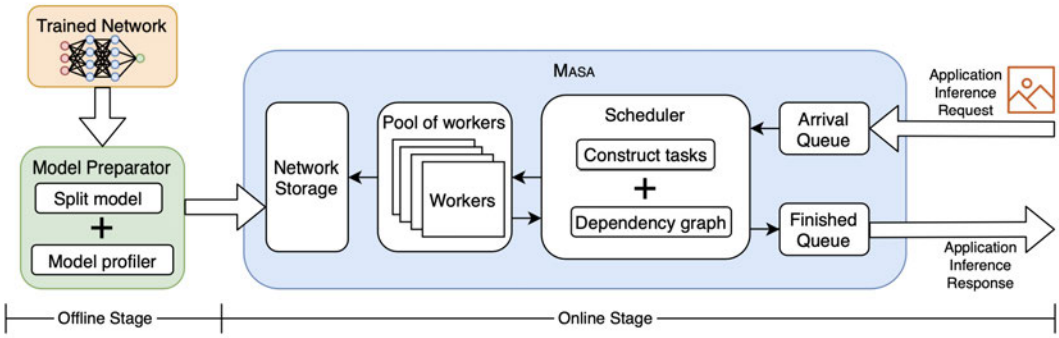

Fig. 2.2. Architecture of *MASA* [17].

By using a memory profiler mechanism, *MASA* could manage the memory usage of each mission and reach a minimum time latency. The task scheduler of *MASA* could split the task of DNN inference and leverage the CPU time for high responsibility. As the result, *MASA* showed high responsibility on small memory devices (*i.e.* 512MB, 1GB, 2GB) and reached the state-of-the-art.

There are also researches like ProxylessNAS [11], MnasNet [66], targeted on optimize hardware-related neural network and have fruitful results.

2.2 Model Compression

2.2.1 Pruning

To the best of our knowledge, the original idea of network pruning was proposed by Hanson and Pratt [31], which has been used to minimize the number of hidden units in the network. After then, optimal brain damage [47] and optimal brain surgeon [32] have been successively proposed. Both of them could purge unimportant weights within the network by leveraging Hessian matrix.

The following research attention are focused on removing redundant units from DNN models. In 2015, Srinivas and Babu [63] proposed a systematic way for removing similar neurons in the network, which successfully removed 35% for AlexNet, without affecting the performance of the network significantly. In the same year, Han *et al.* [30] proposed the prestigious method, Deep Compression, which applied pruning, quantization, and Huffman coding on LeNet, AlexNet and VGG-16 networks. By combining the three compression methods together, Han *et al.* successfully reached 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.

Pruning with pre-trained DNN models could use the networks that has been proved success, which significantly reduces the difficulty of redesigning the network and helps accelerating the research progress. However, network pruning could only help reducing the size of network, while requires manually setting for the hyperparameter (*i.e.* pruning sensitivity, which will be discussed in detail in Chapter 3).

It is also worth noting that, network pruning will not help reducing training time cost. As the pruning method is targeting on reducing model size by pruning trained model, the training of the full model is unchanged. Besides, the re-training of the pruned model still requires time for reaching the best performance. Meanwhile,the

reduction of time cost for inference are not quite noticeable, as the network structure is still maintained during the pruning procedure. Thus, it is important to combine pruning with other techniques for better performance, *e.g.* quantization.

2.2.2 Quantization

Gupta *et al.* [28] applied an parameter quantization method, which limited the data precision. Instead of using typical 32-bit floating-point value for data representing and computing, a fixed-point value has been used. It is defined as $\langle \alpha, \beta \rangle$, where α and β denote the bits length of the number's integer and fractional part, respectively. δ is defined as $(\alpha + \beta)$, which stands for the total length of the number. Their results showed that DNN is endurable for numerical quantization. Setting δ to 16 could incur little or even no degradation for classification tasks, comparing with normal 32-bit floating-point value.

In 2018, Zhou *et al.* [80], proposed an optimization framework using quantization. They first measured the error raise caused by quantization for each layer within the model, then adjusted the quantization bit-width for each layer accordingly. Using this layer-by-layer modification, they reached at least 15% smaller model size (40% at maximum).

2.2.3 Low Rank Approximation

In 2014, Denton *et al.* [19] noted that, within a Convolution Neural Network (CNN), nearly 90% of the computing time are spent on convolution layers, while the first several layers are the most time-consuming ones.

Based on the general statement of knowledge that the weight matrices within the network models usually contain high sparsity, the main idea of low rank approximation approach is to decompose the sparse weight matrices into several matrices with high density and low rank.

Fig.2.3 shows the core step of low rank approximation. For the sparse weight

matrix in one convolution layer, it will be analyzed and decomposed into dense matrices and help reduce the storage space usage. This process will go through the target network model.

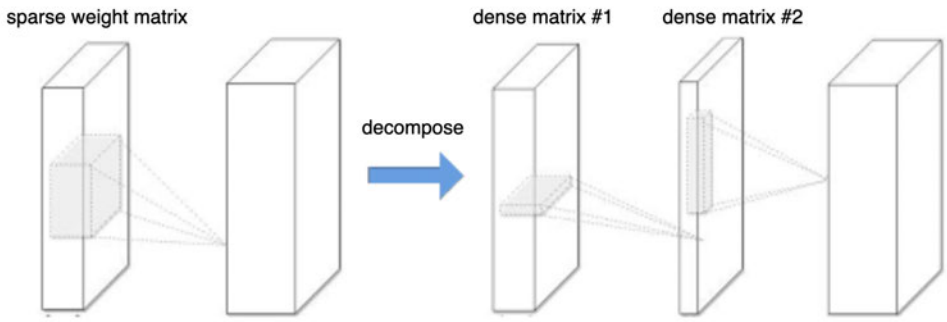

Fig. 2.3. Concept of low rank approximation [15]

Applying the low rank approximation methodology, Denton *et al.* decomposes the original parameter matrices with Singular Value Decomposition (SVD), the required space for matrices storage has been significantly decreased. Further more, the decomposed matrices could be clustered, and their approach reaches a final compression rate at 3.9×.

As the convolution layer plays an important part of CNN, the low rank approximation could decrease the model size notably. Nevertheless, the decomposing operation requires significant computing resources, and each layer in CNN model need to be processed. This will lead to remarkable time cost.

2.2.4 Knowledge Distillation

It is generally accepted that, Ba and Caruana are the first who proposed knowledge distillation [6]. They successfully trained a shallow network to mimic a deep network, without sacrificing accuracy back in 2014. This method was based on one of the authors' previous work [10], which trained a small network to approximate a full network on pseudo data.

Further more, Fu *et al.* [23] proposed the "Panel-Student" scheme. By introducing the *Panel*, which formed by the combination of teachers' representation, this approach could leverage multiple teacher networks and provide more comprehensive data representation to the student network. The student network could be supervised by the panel as *soft* target loss, along with the ground truth as *hard* target loss.

Comparing with previous approaches, knowledge distillation could impressively reduce the size of model by creating shallow student networks which learn from the original deep neural networks. However, this requires the student networks and teacher networks share similar network structures. If the difference between two structures are relatively large, this approach may not reach a satisfied situation. Meanwhile, comparing with other approaches, the accuracy performance of knowledge distillation is less competitive, as the student networks always under-perform the teacher, though the discrepancy may be tiny.

2.3 Tiny Machine Learning (TinyML)

Comparing with DNN, TinyML is a relative young concept [58], which appears with the development of IoT. As a bottom-top strategy, tinyML frameworks, which target at IoT and edge devices, have already taken the computing and energy capacity under consideration.

It is worth noting that TinyML is not a replacement, but a supplement for existing IoT architecture. While the cloud computing contains sufficient resources for heavy workloads, the latency of data transferring and the cost for stable network is remarkable. On the other hand, most of the IoT devices are designed with miliwatt level of power consumption, which is not possible to execute conventional machine learning nor deep learning. By empowering the edge devices with intelligence, the TinyML could help bridging the gap [22].

Currently, there are some pilot TinyML frameworks, *e.g.* TensorFlow Lite[3], Apple Core ML[4], and Pyorch Mobile[5]. These frameworks are designed for edge devices (*i.e.* smart portable devices and MCUs) and are widely adopted in several areas, including data collection, computer vision and medication [2, 3, 48].

[3] https://www.tensorflow.org/lite/
[4] https://developer.apple.com/documentation/coreml/
[5] https://pytorch.org/mobile/home/

Chapter 3

An Empirical Study on Model Pruning and Quantization[1]

> It is the nature of humankind to push it self towards the horizon.
> We test our limits, we face our fears, we rise to the challenge,
> and become something greater than ourselves, a civilization.
>
> Sid Meier's Civilization VI

3.1 Introduction

Recently, several approaches were proposed to perform model compression with different methdology, and reached state-of-the-art respectively. Deep Compression by Han *et al.* [30], which applied pruning, quantization, and Huffman coding on LeNet, AlexNet and VGG-16 networks; low rank approximation by Denton *et al.* [19], which decomposes the original parameter matrices with Singular Value Decomposition (SVD); Gong *et al.* [27] applied the vector quantization on a convolutional neural network, which contains 5 convolutional layers and 3 dense connected layers; knowledge distillation by Ba and Caruana [6], which trained a shallow network to mimic a deep network, without sacrificing accuracy.

These approaches provided promising outcomes and were widely accepted. However, it is worth noting that these methods are limited on particular networks or datasets, LeNet, AlexNet, *MNIST*, *etc.* The further investigation for other networks

[1] This chapter is based on the submitted paper, *An Empirical Study on Model Pruning and Quantization*. I am responsible for the solution, implementation, experimentation, and paper writing. The supervisor is involved with the discussion of motivation, design and results. The other co-authors are contributing to the writing of the paper.

and datasets are yet unrevealed.

Meanwhile, with the advancement of neural network theory and design, modern networks contain complex architectures, layers and blocks to extract features from large datasets. Apply the compression methods directly to complex modern networks may lead to performance degradation or even fatal corruption of the networks. 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.

In this chapter, we introduce the methodology that has been used to conduct a comprehensive experimentation and comparison among pruning and quantization. We also conduct an attempt on knowledge distillation. Finally, we explain the results to the best of our knowledge and draw the conclusion.

3.2 Methodology

3.2.1 Data Augmentation

To increase the generalization of deep learning models, there are two major approaches, one is on modifying models architecture, and the other one is on enlarging training datasets [60]. Commonly used model-side techniques will require adding specific functions and layers to the models, *e.g.* dropout layer [64] and batch normalization layer [39]. While commonly used data augmentation methods on images will perform geometric transformations, flipping, color space alteration, cropping, rotation, and noise injection [60].

As our research target is to explore the compression effectiveness on specific models, we choose to augment the training datasets, which could help mitigate model overfitting. The image datasets that we use in our experiments, *i.e. Fashion-MNIST* and *CIFAR-10*, are both augmented. The augmentation procedure that we follow is shown in Fig 3.1.

The augmentation step adopts from the practice in [34]. As our implementation

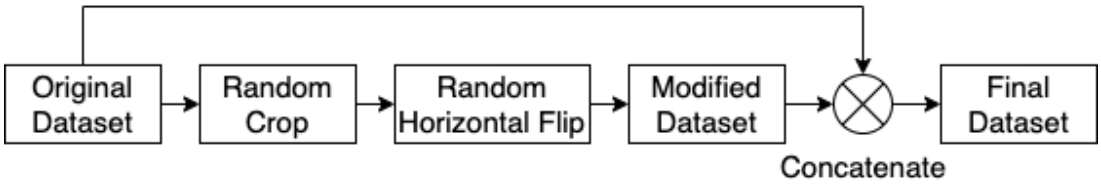

Fig. 3.1. Dataset augmentation procedure

of networks require constant input shape, we first reshape the image to the required size, 63×63 for AlexNet [43], 33×33 for ResNet 18 [34], 32×32 for VGG-11 [62] with Batch Normalization [39] and VGG-19 [62] with Batch Normalization [39]. A random crop is sampled, follows by a random horizontal flip with 0.5 as possibility.

It is worth noting that the random crop is padding-enabled to maintain the required size.

3.2.2 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* [30]. 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:

$$\mathcal{T} = STD(W_l) \times Sensitivity \tag{3.1}$$

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 3.2.

Fig. 3.2. Train, prune, retrain

3.2.3 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 3.3. 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.

Though there are several clustering centroid initialization methods (e.g. random [52], density-based [18], linear [12]), it has been proved that initializing the centroids with linear method could mitigate poor representation caused by the singular value [30]. 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:

$$Centroids = Cluster(2^{Q\_Bit}, W_l) \tag{3.2}$$

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:

$$Centroids = Cluster(W_l) \tag{3.3}$$

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.

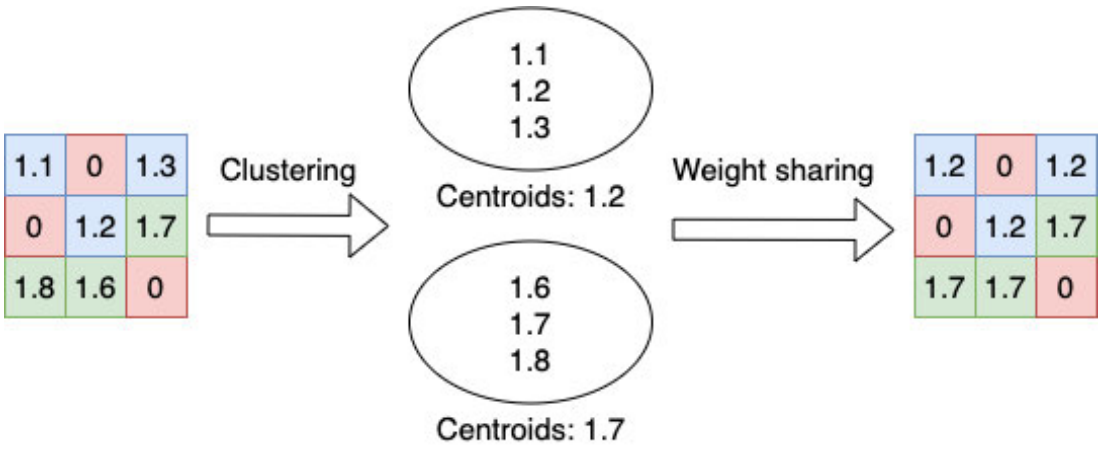

Fig. 3.3. An example of non-zero weight sharing

3.2.4 Knowledge Distillation, a Comparative Baseline

In most of the neural networks, the *cross-entropy* loss will be used to measure the diverges from predicted label to the actual label (ground truth), which is calculated as:

$$\mathscr{L} = -\sum_{i=1}^{K} t_i \log p_i \tag{3.4}$$

and the one-hot encoding will be used to represent the possibilities of the corresponding classes.

Meanwhile, it is worth noting that, for most of the real-life scenario, the similarity lies between different classes and within the same class is relatively important for the classification. For instance, in hand writing number recognition, some hand-written '2' may look similar to '3', while some may have higher similarity with '7'. For different '2', *cross-entropy* loss will provide different values for class '3' and class '7'. These different values together provide a comprehensive description for the data.

However, applying the one-hot encoding arbitrarily may cause loss of these kind of information. To help preserve the information, Hinton *et al.* [35] proposed an altered *softmax* function as:

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp\left(z_j/T\right)} \tag{3.5}$$

where the T stands for the distillation temperature, which is a hyperparameter. When $T = 1$, this function works the same as normal *softmax* function. When adjusting the value of T to greater than 1, this *softmax* function will be soften and help protecting the inner relationship within the data. The higher the value, the *softer* it is.

Using Eq.3.5 to replace the normal *softmax* function in the teacher model (*i.e.* the deep one), we could obtain the soft target rather than normal arbitrary one-hot target, which contains more information that could help training the student model (*i.e.* the shallow one).

For the student model, we could apply an altered *cross-entropy* loss function:

$$\mathscr{L}_{\text{soft}} = -\sum_{i=1}^{K} p_i \log q_i \tag{3.6}$$

where p_i stands for the soft target of the teacher network, and q_i stands for the soft target of the student network.

Meanwhile, to ensure the performance of distillation, the final loss function could be set as:

$$\mathscr{L} = w_1 \mathscr{L}_{\text{hard}} + w_2 \mathscr{L}_{\text{soft}} \tag{3.7}$$

where L_{hard} stands for the normal *cross-entropy* loss. Coefficients w_1 and w_2 are weights for the loss values, respectively. It is worth mentioning that, w_2 need to be multiplied by T^2 before actually been used. The reason is that the gradients been

calculated by using soft targets will be scaled as $1/T^2$.

3.3 Experiment Design

3.3.1 Experimental Setup

Dataset: The experiments are conducted based on three widely used datasets, *Fashion-MNIST*[2] [72], *CIFAR-10*[3] [42], and *UCI-HAR*[4] [4].

Fashion-MNIST 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) is 50Hz, 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 [34], AlexNet [43], VGG-11 [62] with Batch Normalization [39] (**VGG11BN**), and VGG-19 [62] with Batch Normalization [39]

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

Table 3.1: 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 [33]

[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*.

(**VGG19BN**), which have 11.2M, 20.3M, 28.1M, 40.0M parameters, respectively.

Training: All experiments are implemented on **Ubuntu 20.04**. The machine learning framework is **Pytorch** (version 1.11.0) with **CUDA** (version 11). The GPU we use for computing acceleration is one **NVIDIA RTX 2080Ti**. The experimental hyperparameter settings we use during the model training stages are illustrated in Table 3.1.

3.3.2 Research Questions

- **RQ1: How the weight-based pruning affects the models' performance?**
  To explore the impact of network pruning, we first train the full networks from the scratch, 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?**
  To inspect the retraining output, we adopt the pruned networks with different compression rates from the previous research question. The training settings we use here are set the same as the full ones in **RQ1**, which could prevent

other variates from affecting the results.

- **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 (adjust by hyperparameter Q_Bit). The corresponding centroid numbers for different Q_Bit $\{4, 3, 2\}$ are $\{2^4, 2^3, 2^2\}$, respectively. It is worth mentioning that, We apply a modern non-linear clustering centroid initialization (*i.e.* K-Means++ [5]), along with the conventional linear clustering centroid initialization, for further comparison.

- **RQ4: How the knowledge distillation perform with the selected networks?**

  We implement the knowledge distillation from scratch for the selected backbone networks and compare the performance. We use VGG-11 and VGG-19 as teacher models while AlexNet as the student model.

3.4 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 A.2, A.3, A.4, respectively. The quantization results are shown in Table A.5. The comparison with knowledge distillation are shown in Table A.6.

For **RQ1**, comparing with the full models, the performance of compressed ones degrade 2.33% on *Fashion-MNIST* dataset, 4.25% on *CIFAR-10*, 41.57% on *UCI-HAR*, at average, with 2× models. While 73.19% on *Fashion-MNIST* dataset, 62.88% on *CIFAR-10*, 67.29% on *UCI-HAR*, at average, with 10× models. The 1.1× models

shows no significant difference. This phenomenon shows that pure pruning could still keep the model performance until 2×, but models for temporal domain dataset are more vulnerable for pruning.

It is also noticeable that, though pruning will lead to model performance degrade, 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. On the contrast, ResNet18, the smallest network, shows the largest degrade (more than 4%, the same as AlexNet) for 2× compression, while the least degrade (less than 63%) for 10× compression.

We believe that for large network with considerable feature extraction layers, the stability could maintain when pruning. But when certain portion of the weights been removed, the performance will drastically drop. For residual networks, the feature extraction layers are shallow, but the structure makes them more restrainable when large amount of weights are pruned (*i.e.* 90%). In other words, large deep networks are insensitive but vulnerable for pruning. Meanwhile, residual networks are sensitive but robust for pruning.

On temporal domain dataset (*UCI-HAR*), though AlexNet shows the least degrade among all chosen networks, all four networks shows obvious degrade start from 2× compression. We believe that models trained for temporal domain datasets are more fragile and susceptible for pruning.

For **RQ2**, from Table A.2, A.3, A.4, after retraining the pruned networks, the performance of all compressed networks (*i.e.* $1.1\times, 2\times, 10\times$) show no noticeable or tolerable degrade (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). Which reflects that retraining is essential for high compression rate (*e.g.* 10×), applying retraining step could keep the overall accuracy.

For **RQ3**, from Table A.5, the performance degrade for 4 quantization bits (Q_Bit {4}) is not noticeable, as the average degrade is less than 1%, while minimum degrade is 0.04% and maximum degrade is 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 with the pruning rate, but also 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 the similar characteristic as deep networks.

We believe the reason is that, shallow networks contains less layers and the distinctiveness of parameters are the key for maintain the performance. Applying quantization with less 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. We believe the reason is that, shallow networks contains less layers and the distinctiveness of parameters are the key for maintain the performance. Applying quantization with less 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.

Among all four selected backbone networks, ResNet18 reaches the best performance. As ResNet18 contains residual block (RESBLOCK), which might be the key part for maintaining the accuracy.

For **RQ4**, our experimentation could not provide a satisfied results among the networks we've chosen, as shown in Table A.6. Comparing with the original VGG11BN and VGG19BN, the corresponding AlexNets trained through distillation show perceivable degradation of performance (more than 7%). This degradation even exists when comparing with original AlexNet (more than 5%).

Compare with pruning and quantization, for instance, VGG11BN been processed by $\langle 10\times, Q\_Bit\{3\}\rangle$ setting could still reach 90% accuracy on *Fashion-MNIST* and 85% accuracy on *CIFAR-10*, distilled AlexNet only reaches 85% accuracy on *Fashion-MNIST* and 74% accuracy on *CIFAR-10*. Considering the parameters in distilled AlexNet is 7× as VGG11BN $\langle 10\times, Q\_Bit\{3\}\rangle$, the performance is not as good as pruning and quantization.

We believe the reason is that, during our experiments, the teacher models we choose from the backbone networks are relatively small, which could not provide enough representation of the feature to the student model. Nevertheless, we have drawn some conclusions for future usage. The student model should share a similar structure with the teacher model to guarantee the feature representation could be successfully adapted. Meanwhile, the hyperparameter T within the training procedure is inversely proportional to the size ratio between teacher and student models, and need to be adjust accordingly. However, if the student model is relatively too small for the teacher model and the dataset, it would not be possible to learn the feature representation no matter how the T changes.

3.5 Related Work

Since AlexNet [43] 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 [38], and ShuffleNets [78]. One of the most famous models is MobileNet [36], 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 hyper-parameters 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 computing 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 [11, 66], 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 [67] 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 [34, 36, 76], 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. [15] 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 [29, 41, 70].

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.

Other researches on distilling models: Data gathered from the real life world usually contains several dimensions and semi-labeled [74], which lacks the trainability for conventional neural networks. One of the solution is leveraging multi-view learning (*i.e.* co-training) to bootstrap the training process and achieve maximum mutual agreement [8, 55]. By adopting the co-training concept, Fu *et al.* proposed the "Panel-Student" learning scheme [23]. The core idea is to combining three teachers networks' soft target loss as a training panel, rather than just using one network's output. Under the panel's supervision, student model could leverage the

knowledge from all three teacher models and reach high accuracy. The proposed approach has been tested in real-world scenarios under a 6 months study, where it outperformed previous state-of-the-art methods.

3.6 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 degradation of performance (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 (average 6.5%, maximum 15.55%). Meanwhile, we compare different clustering algorithms and improve the performance (maximum 0.5%, on AlexNet 10×). we conduct knowledge distillation attempt as well. The experiment shows that for relatively small networks, knowledge distillation might not be an ideal option for compression.

We have open-sourced these experimentation codes on GitHub[5]. 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. [20, 68]) 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. It is also worth mentioning that, the temperature T for

[5] https://github.com/broadnet2022-compression/code

knowledge distillation is a hyperparameter which needs to be adjusted manually. It would be a significant progress if the learning scheme could be built to automatically train the T as a parameter.

Through our experimentation, we draw the conclusion that a small pruning rate is more suitable for large deep networks, while a large pruning rate could be deployed on residual networks. For quantization, clustering with $Q\_Bit\{4\}$ could be used as a general compression tool, as it shows no noticeable performance degradation while greatly reducing the size for weight storage. As for knowledge distillation, this method requires more prerequisites, some of which might not be able to satisfy, thus leading to decreased practicality. Based on these conclusions, we plan to further explore the boundaries and possibilities of these methods.

Chapter 4

Conclusion

Salimmo sù, el primo e io secondo, tanto ch'i' vidi de le cose belle che porta 'l ciel, per un pertugio tondo. E quindi uscimmo a riveder le stelle.

Dante - Divine Comedy

In this study, we go through the previous research, extract the fundemental knowledge, implement the experimentation, and provide our views and explanations to the results. 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, and show the understanding of both limitations and possibilities of the model compression techniques.

As a result, we propose the guideline for applying compression methods, reveal the impact of quantization, explore the knowledge distillation, indicate the future research directions, and open our source code.

It is worth noting that there still exists research gaps that lie between model compression and IoT deployment, including but not limited to paralleling DNN on distributed MCU networks and scalability of DNN for universal deployment. Meanwhile, accompanied by IoT and machine learning development, tinyML shows a bright future for implementing neural networks on MCUs. These uncharted areas and methodologies are worth exploring, and we plan on continuing the research in the future.

Appendix

> We choose to go to the Moon in this decade and do the other things, not because they are easy, but because they are hard; because that goal will serve to organize and measure the best of our energies and skills, because that challenge is one that we are willing to accept, one we are unwilling to postpone, and one we intend to win, and the others, too.
>
> John Fitzgerald Kennedy, the 35th President of the United States

A.1 Experimentation Results in Details

During the study, we review voluminous publications and materials, from where we conduct the comparison among IoT devices. Meanwhile, In order to explore the impact and suitability of model compression methods, we perform extensive experiments on selected neural networks (*i.e.* ResNet18, AlexNet, VGG11BN and VGG19BN) and datasets (*i.e.* *Fashion-MNIST*, *CIFAR-10* and *UCI-HAR*). The full results of the experimentation we conduct during this study are listed here.

Table A.1 shows the comparison of mainstream IoT MCU devices and corresponding specifications in details.

Table A.2 shows the pruning results on *Fashion-MNIST* dataset.

Table A.3 shows the pruning results on *CIFAR-10* dataset.

Table A.4 shows the pruning results on *UCI-HAR*, dataset.

Table A.5 shows the quantization results.

Table A.1: Mainstream IoT MCU Devices Specification

Brand Name	*Chip*	*Clock Speed*	*On Chip RAM/ROM size*	*On Board RAM/ROM size*
Arduino Pro Portenta H7	Dual Cortex-M7+M4	M7: 480 MHz M4: 240 MHz	1MB RAM 2MB Flash	8MB RAM 16MB Flash
Arduino MKR 1000 Arduino MKR 1010	Cortex-M0+	48 MHz	32KB RAM 256KB Flash	N/A
Arduino Nicla Sense ME	Cortex-M4F	64MHz	64KB RAM 512KB Flash	2MB Flash
Raspberry Pi Pico	Dual Cortex-M0+	133MHz	264KB RAM	2MB Flash

The specifications are gathered from the official Arduino website[a] and official Raspberry website[b], respectively.

[a] https://www.arduino.cc

[b] https://www.raspberrypi.org

Table A.2: Pruning result comparison, *Fashion-MNIST*

Network	$Sensitivity$[a]	#Params	Compression Rate	Accuracy-AP[b]	Accuracy[c]
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 accuracy after pruning without retrain. (**RQ1**)
[c] The accuracy after training (**full network**) / retraining (**pruned network**). (**RQ2**)

Table A.3: Pruning result comparison, *CIFAR-10*

Network	$Sensitivity$[a]	#Params	Compression Rate	Accuracy-AP[b]	Accuracy[c]
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 accuracy after pruning without retrain. (**RQ1**)
[c] The accuracy after training (**full network**) / retraining (**pruned network**). (**RQ2**)

Table A.4: Pruning result comparison, *UCI-HAR*

Network	$Sensitivity$[a]	#Params	Compression Rate	Accuracy-AP[b]	Accuracy[c]
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 accuracy after pruning without retrain. (**RQ1**)
[c] The accuracy after training (**full network**) / retraining (**pruned network**). (**RQ2**)

Table A.5: Quantization result comparison

AlexNet

P-Rate[a]	*Q_Bit*[b]	Degrade-L[c](%)	Degrade-NL[d](%)
1.1×	2	23.74	20.90
	3	3.48	5.34
	4	1.04	0.77
2×	2	23.07	36.62
	3	3.51	4.45
	4	0.28	0.65
10×	2	32.63	42.07
	3	6.42	7.45
	4	0.30	0.09

ResNet18

P-Rate[a]	*Q_Bit*[b]	Degrade-L[c](%)	Degrade-NL[d](%)
1.1×	2	62.70	68.43
	3	8.19	9.68
	4	0.17	0.50
2×	2	47.61	58.43
	3	2.29	3.28
	4	0.67	0.68
10×	2	43.79	39.12
	3	1.92	3.02
	4	0.19	0.66

VGG11BN

P-Rate[a]	*Q_Bit*[b]	Degrade-L[c](%)	Degrade-NL[d](%)
1.1×	2	62.41	73.75
	3	18.69	16.72
	4	0.94	0.55
2×	2	48.37	75.19
	3	2.66	2.08
	4	0.24	0.04
10×	2	52.80	67.86
	3	2.94	3.44
	4	0.31	0.50

VGG19BN

P-Rate[a]	*Q_Bit*[b]	Degrade-L[c](%)	Degrade-NL[d](%)
1.1×	2	57.22	77.55
	3	34.84	35.55
	4	0.12	1.32
2×	2	77.33	77.33
	3	12.21	16.32
	4	0.42	0.29
10×	2	71.38	77.09
	3	3.46	5.18
	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.

Table A.6: Knowledge distillation comparison

VGG11BN → AlexNet, *Fashion-MNIST*

Temperature[a]	Degrade-V[b](%)	Degrade-A[c](%)
1.1	7.34	5.86
1.2	9.33	7.34
1.5	10.58	8.97

VGG19BN → AlexNet, *Fashion-MNIST*

Temperature[a]	Degrade-V[b](%)	Degrade-A[c](%)
1.1	8.74	6.86
1.2	11.39	8.25
1.5	12.47	9.78

VGG11BN → AlexNet, *CIFAR-10*

Temperature[a]	Degrade-V[b](%)	Degrade-A[c](%)
1.1	11.76	8.91
1.2	12.30	10.04
1.5	14.12	11.86

VGG19BN → AlexNet, *CIFAR-10*

Temperature[a]	Degrade-V[b](%)	Degrade-A[c](%)
1.1	13.32	10.77
1.2	14.94	11.48
1.5	17.32	14.13

[a] The hyperparameter T in distillation.
[b] The accuracy degrade comparing with VGG.
[c] The accuracy degrade comparing with AlexNet.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[2] Shamim Ahmed and Marc Bons. Edge computed nilm: a phone-based implementation using mobilenet compressed by tensorflow lite. In *Proceedings of the 5th International Workshop on Non-intrusive Load Monitoring*, pages 44–48, 2020.

[3] Oscar Alsing. Mobile object detection using tensorflow lite and transfer learning, 2018.

[4] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra Perez, and Jorge Luis Reyes Ortiz. 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*, pages 437–442, 2013.

[5] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[6] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? *Advances in neural information processing systems*, 27, 2014.

[7] Javier Balladini, Remo Suppi, Dolores Rexachs, and Emilio Luque. Impact of parallel programming models and cpus clock frequency on energy consumption of hpc systems. In *2011 9th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, pages 16–21. IEEE, 2011.

[8] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[9] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. *Advances in neural information processing systems*, 20, 2007.

[10] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[11] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[12] M Emre Celebi and Hassan A Kingravi. Linear, deterministic, and order-invariant initialization methods for the k-means clustering algorithm. In *Partitional clustering algorithms*, pages 79–98. Springer, 2015.

[13] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. {TVM}: An automated {End-to-End} optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 578–594, 2018.

[14] Xing Chen, Ming Li, Hao Zhong, Yun Ma, and Ching-Hsien Hsu. Dnnoff: offloading dnn-based intelligent iot applications in mobile edge computing. *IEEE transactions on industrial informatics*, 18(4):2820–2829, 2021.

[15] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[16] Keum San Chun, Sarnab Bhattacharya, and Edison Thomaz. Detecting eating episodes by tracking jawbone movements with a non-contact wearable sensor. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies*, 2(1):1–21, 2018.

[17] Bart Cox, Jeroen Galjaard, Amirmasoud Ghiassi, Robert Birke, and Lydia Y Chen. Masa: Responsive multi-dnn inference on the edge. In *2021 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE, 2021.

[18] Kabiru Dalhatu and Alex Tie Hiang Sim. Density base k-mean's cluster centroid initialization algorithm. *International Journal of Computer Applications*, 137(11), 2016.

[19] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014.

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[21] Zakarya Drias, Ahmed Serhrouchni, and Olivier Vogel. Analysis of cyber security for industrial control systems. In *2015 international conference on cyber security of smart cities, industrial control system and communications (ssic)*, pages 1–8. IEEE, 2015.

[22] Lachit Dutta and Swapna Bharali. Tinyml meets iot: A comprehensive survey. *Internet of Things*, 16:100461, 2021.

[23] Min Fu, Jiwei Guan, Xi Zheng, Jie Zhou, Jianchao Lu, Tianyi Zhang, Shoujie Zhuo, Lijun Zhan, and Jian Yang. Ics-assist: Intelligent customer inquiry resolution recommendation in online customer service for large e-commerce businesses. In *International Conference on Service-Oriented Computing*, pages 370–385. Springer, 2020.

[24] Katsuya Fujii, Keita Higuchi, and Jun Rekimoto. 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*, pages 216–223. IEEE, 2013.

[25] Hironobu Fujiyoshi, Tsubasa Hirakawa, and Takayoshi Yamashita. Deep learning-based image recognition for autonomous driving. *IATSS research*, 43(4):244–252, 2019.

[26] Jack Ganssle, Tammy Noergaard, Fred Eady, Lewin Edwards, David J Katz, Rick Gentile, Ken Arnold, Kamal Hyder, and Bob Perrin. *Embedded Hardware: Know It All*. Newnes, 2007.

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

[28] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pages 1737–1746. PMLR, 2015.

[29] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. 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*, pages 123–136, 2016.

[30] 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.

[31] Stephen Hanson and Lorien Pratt. Comparing biases for minimal network construction with back-propagation. *Advances in neural information processing systems*, 1, 1988.

[32] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[35] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[36] 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.

[37] Guosheng Hu, Yongxin Yang, Dong Yi, Josef Kittler, William Christmas, Stan Z Li, and Timothy Hospedales. When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 142–150, 2015.

[38] 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.

[39] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[40] Yonghang Jiang, Zhenjiang Li, and Jianping Wang. Ptrack: Enhancing the applicability of pedestrian tracking with wearables. *IEEE Transactions on Mobile Computing*, 18(2):431–443, 2018.

[41] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1):615–629, 2017.

[42] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[44] Liangzhen Lai, Naveen Suda, and Vikas Chandra. Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus. *arXiv preprint arXiv:1801.06601*, 2018.

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

[46] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[47] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[48] Hongjia Li, Geng Yuan, Wei Niu, Yuxuan Cai, Mengshu Sun, Zhengang Li, Bin Ren, Xue Lin, and Yanzhi Wang. Real-time mobile acceleration of dnns: From computer vision to medical applications. In *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 581–586. IEEE, 2021.

[49] Ji Lin, Wei-Ming Chen, Han Cai, Chuang Gan, and Song Han. Memory-efficient patch-based inference for tiny deep learning. *Advances in Neural Information Processing Systems*, 34:2346–2358, 2021.

[50] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, Song Han, et al. Mcunet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems*, 33:11711–11722, 2020.

[51] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. 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*, pages 5334–5343, 2017.

[52] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[53] MH MAMMADOVA and ZG JABRAYILOVA. Conceptual approaches to iot-based personnel health management in offshore oil and gas industry. *CONTROL AND OPTIMIZATION WITH INDUSTRIAL APPLICATIONS*, page 257, 2020.

[54] Vidyabhushan Mohan, Trevor Bunker, Laura Grupp, Sudhanva Gurumurthi, Mircea R Stan, and Steven Swanson. Modeling power consumption of nand flash memories using flashpower. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(7):1031–1044, 2013.

[55] Xin Ning, Xinran Wang, Shaohui Xu, Weiwei Cai, Liping Zhang, Lina Yu, and Wenfa Li. A review of research on co-training. *Concurrency and computation: practice and experience*, page e6276, 2021.

[56] Lucas Prado Osco, José Marcato Junior, Ana Paula Marques Ramos, Lúcio André de Castro Jorge, Sarah Narges Fatholahi, Jonathan de Andrade Silva, Edson Takashi Matsubara, Hemerson Pistori, Wesley Nunes Gonçalves, and Jonathan Li. A review on deep learning in uav remote sensing. *International Journal of Applied Earth Observation and Geoinformation*, 102:102456, 2021.

[57] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[58] Partha Pratim Ray. A review on tinyml: State-of-the-art and prospects. *Journal of King Saud University-Computer and Information Sciences*, 2021.

[59] Enrico Russo, Maurizio Palesi, Salvatore Monteleone, Davide Patti, Andrea Mineo, Giuseppe Ascia, and Vincenzo Catania. Dnn model compression for

iot domain-specific hardware accelerators. *IEEE Internet of Things Journal*, 9(9):6650–6662, 2021.

[60] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[61] Fadi Shrouf, Joaquin Ordieres, and Giovanni Miragliotta. Smart factories in industry 4.0: A review of the concept and of energy management approached in production based on the internet of things paradigm. In *2014 IEEE international conference on industrial engineering and engineering management*, pages 697–701. IEEE, 2014.

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

[63] Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*, 2015.

[64] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[65] Wen-Tsai Sung and Yao-Chi Hsu. Designing an industrial real-time measurement and monitoring system based on embedded system and zigbee. *Expert Systems with Applications*, 38(4):4522–4529, 2011.

[66] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[67] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[69] Chunxin Wang, Xiangzhen Li, Yakun Liu, and Huanjuan Wang. The research on development direction and points in iot in china power grid. In *2014 International Conference on Information Science, Electronics and Electrical Engineering*, volume 1, pages 245–248. IEEE, 2014.

[70] Song Wang, Xinyu Zhang, Hiromasa Uchiyama, and Hiroki Matsuda. Hivemind: Towards cellular native machine learning model splitting. *IEEE Journal on Selected Areas in Communications*, 40(2):626–640, 2021.

[71] Tian Wang, Pan Wang, Shaobin Cai, Xi Zheng, Ying Ma, Weijia Jia, and Guojun Wang. Mobile edge-enabled trust evaluation for the internet of things. *Information Fusion*, 75:90–100, 2021.

[72] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[73] Xiufeng Xie and Kyu-Han Kim. Source compression with bounded dnn perception loss for iot edge computer vision. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.

[74] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013.

[75] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.

[76] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[77] Dongxia Zhang, Xiaoqing Han, and Chunyu Deng. Review on the research and practice of deep learning and reinforcement learning in smart grids. *CSEE Journal of Power and Energy Systems*, 4(3):362–370, 2018.

[78] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[79] Chang-Le Zhong, Zhen Zhu, and Ren-Gen Huang. Study on the iot architecture and gateway technology. In *2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 196–199. IEEE, 2015.

[80] Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. Adaptive quantization for deep neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.