Received 21 June 2022, accepted 3 August 2022, date of publication 8 August 2022, date of current version 17 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3197608

RESEARCH ARTICLE

Analysis of Model Compression Using Knowledge Distillation

YU-WEI HONG, JENQ-SHIOU LEU, (Senior Member, IEEE), MUHAMAD FAISAL, AND SETYA WIDYAWAN PRAKOSA
Department of Electronic and Computer Engineering (ECE), National Taiwan University of Science and Technology, Taipei City 106, Taiwan

Corresponding author: Muhamad Faisal (d10802803@mail.ntust.edu.tw)

This work was supported by the National Science and Technology Council, Taiwan, under Grant 111-2221-E-011-070-.

ABSTRACT In the development of deep learning, several convolution neural network (CNN) models are designed to solve various tasks. However, these CNN models are complex and cumbersome to achieve state-of-the-art performance. The current CNN models remain to suffer from the problem of large models. Thus, model compression techniques are proposed to cope with the complex CNN models. Meanwhile, the selection of compressed model to suit the user requirement significantly contributes during the deployment process. This paper analyses two model compressions, namely the layerwise and the widthwise compression. The compression techniques are implemented in the MobileNetV1 model. Then, knowledge distillation is applied to compensate for the accuracy loss of the compressed model. We demonstrate the analysis of those compressed models from various perspectives and develop several suggestions on the trade-off between the performance and the compression rate. In addition, we also show that the feature that is learned by the compressed models using knowledge distillation has better representation compared to the vanilla model. Our experiment shows that the widthwise compression on MobileNetV1 achieves a compression rate of 42.27% and the layerwise compression achieves 32.42%, respectively. Furthermore, the improvement of the compressed models using knowledge distillation is notable for the widthwise compression with the increasing accuracy above 4.71%.

INDEX TERMS Deep learning, knowledge distillation, model compression.

I. INTRODUCTION

In the last decade, deep learning has been gaining many attentions in the application of computer vision [1], for example, plant leaf diseases [2], smart wearable health devices [3], and smart transportation system [4]. The various applications of deep learning was started by AlexNet [5] which utilized the stacked convolutional model. The proposed method achieved a large margin in terms of error rates in ImageNet LSCRC-2012 compared to the traditional method to deal with the complexity of the dataset. As the popularity of CNN increases, many CNN architectures have been designed, such as GoogLeNet [6], VGGNet [7], and ResNet [8]. To sum up, ResNet achieves an error rate of 3.57% higher than human-level performance in the ImageNet dataset. This dramatic CNN development has a trade-off that as the CNN accuracy increases, the computational complexity and massive storage are demanded.

The fact that achieving the state-of-the-art performance in the CNN models requires a deeper and wider convolutional layer remains an issue in the development of CNN. For example, ResNet [8] and Inception model [9] achieve higher accuracy than AlexNet and VGGNet. However, ResNet and Inception model require a deeper structure than AlexNet and VGGNet. This trade-off remains the challenge in real-time deployment on resource-limited devices. As a result, a scheme to compress the complex network to a small network with similar accuracy is necessary.

Recently, model compression has been proposed to obtain a small network with similar performance and to alleviate the challenge of deploying a model on the devices with limited resources. There are several types of model compressions:

The associate editor coordinating the review of this manuscript and approving it for publication was Chuan Li.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

the lightweight network, knowledge distillation, and pruning filter.

First, the lightweight CNN architectures such as MobileNet [10], EfficientNet [11], ShuffleNet [12], and Condense-Net [13] were proposed for mobile and embedded application. MobileNetV1 is composed of a depthwise separable convolution filter which factorizes the vanilla convolution filter into the depthwise convolution and 1 × 1 pointwise convolution. G Huang *et al.* proposed a novel architecture of lightweight network called CondenseNet [13]. The architecture is built upon the features of a dense connectivity layer and learned group convolution. In summary, the lightweight CNNs reduces the complexity of calculation without losing significant accuracy. However, training the reduced network to achieve high performance remains the issue.

Second, the idea of transferring knowledge from a cumbersome or large network to a compact and shallow network, namely knowledge distillation, is proposed by [14], [15], [16]. Knowledge distillation is a teacher-student approach where the teacher or the large network transfers the knowledge to the student or the shallow network so that the student network achieves a slightly different accuracy from the teacher network. Knowledge distillation has been studied and applied to many applications [17], [18]. Buciluă *et al.* [16] popularizes the knowledge distillation method by applying the concept to deep neural networks and introducing the new hyperparameter namely temperature. The deep complex network or the teacher model is trained from scratch using the full dataset. After the teacher network is obtained, the student network, a small and fast model, is designed to learn the knowledge from the teacher network continuously. The student network utilizes class probability by minimizing distillation loss using temperature loss produced by the teacher and student networks. However, selecting a stable student network architecture is a challenge.

Third, the study to prune the network filter, called the layerwise pruning, also offers promising research in model compression. The method is expected to reduce the number of filters, which has an insignificant influence on the whole structure. In early research, Hanson *et al.* [19] approached the pruning filter by studying biased weight decay. The proposed idea is examined in the simple counting problems and speech recognition. Several researchers [20], [21] studied the Hessian loss function to calculate the reduction of the number of connections. However, the calculation costs a lot at hardware implementation during memory calculation. Similarly, M. Courbariaux *et al.* [22] studied the model binarization by constraining weight at the neural network into 0 or 1 during the training process at forward and backward propagation while retaining the standard weight to calculate the gradient. A. Aghasi *et al.* [23] have proposed the net-trim method, a study to trim the LeNet network. This idea is extended by employing the hessian matrix to select the pruned filter [24]. The approach faces the issue of hardware usage, claiming the use of RAM 512 GB to fulfill the requirement.

TABLE 1. MobileNetV1 architecture.

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 3 × 32	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5× Conv / s1	1 × 1 × 512 × 512	14 × 14 × 256
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 x 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

Consequently, a comprehensive study on finding an effective model compression method could be conducted by combining the existing methods.

In this paper, we propose an analysis of model compression in MobileNetV1 using knowledge distillation. We employ MobileNetV1 architecture as the student network. MobileNetV1 network is compressed based on the layerwise and the widthwise compression. Knowledge distillation is utilized for training the compressed MobileNetV1. Furthermore, the deep Taylor decomposition [25] is applied to visualize the feature learned by the compressed MobileNetV1 using knowledge distillation. Our contributions in this paper include:

1. We analyse the compressed MobileNetV1 based on the layerwise and widthwise compression.

2. We utilize knowledge distillation to the compressed MobileNetV1 for both layerwise and widthwise compression to increase the accuracy. We investigate the compression performance in several aspects: model size, speed, and computational cost.

3. We further utilize deep Taylor decomposition to show visually that the feature learned by the compressed MobileNetV1 using knowledge distillation has better representation than the vanilla MobileNetV1.

TABLE 2. The layerwise compression of MobileNet architecture.

Models	MobileNetv1	MobileNet_cut13	MobileNet_cut12	MobileNet_cut11
Cut from	None	13th	12th	11th
Type / Stride	Output Shape			
Conv / s2	32×32×3			
Conv / s2	16×16×32	16×16×32	16×16×32	16×16×32
Conv dw / s1	16×16×32	16×16×32	16×16×32	16×16×32
Conv / s1	16×16×64	16×16×64	16×16×64	16×16×64
Conv dw / s2	8×8×64	8×8×64	8×8×64	8×8×64
Conv / s1	8×8×128	8×8×128	8×8×128	8×8×128
Conv dw / s1	8×8×128	8×8×128	8×8×128	8×8×128
Conv / s1	8×8×128	8×8×128	8×8×128	8×8×128
Conv dw / s2	4×4×256	4×4×256	4×4×256	4×4×256
Conv / s1	4×4×256	4×4×256	4×4×256	4×4×256
Conv dw / s1	4×4×256	4×4×256	4×4×256	4×4×256
Conv / s1	4×4×256	4×4×256	4×4×256	4×4×256
Conv dw / s2	2×2×256	2×2×256	2×2×256	2×2×256
Conv / s1	2×2×256	2×2×256	2×2×256	2×2×256
Conv dw / s1	2×2×512 (5×)	2×2×512 (5×)	2×2×512 (5×)	2×2×512 (4×)
Conv / s1	2×2×512 (5×)	2×2×512 (5×)	2×2×512 (5×)	2×2×512 (4×)
Conv dw / s2	1×1×512	1×1×512	N/A	N/A
Conv / s1	1×1×1024	1×1×1024	N/A	N/A
Conv dw / s2	1×1×1024	N/A	N/A	N/A
Conv / s1	1×1×1024	N/A	N/A	N/A
Avg Pool / s1	1×1024	1×1024	1×512	1×512
FC / s1	10			
Softmax / s1	10			

The rest of the paper is organized as follows. The analysis of model compression based on the layerwise and the widthwise compression using knowledge distillation is explained in section II. The experimental setup and result are presented in section III. In the end, section IV concludes the paper.

II. METHODOLOGY

The study analyses the layerwise and the widthwise compression in MobileNetV1. There are three investigations conducted in the experiment. First, we examine the effect of the compressed MobileNetV1 in terms of layerwise and widthwise perspectives. This experiment is expected to show how far the lightweight network can be reduced to achieve sufficient accuracy. Then, the training is conducted using the vanilla training step. Our preliminary study shows that the layerwise and the widthwise compression of MobileNetV1 decrease the accuracy significantly, particularly in training the network scracthly without any additional step. Second, knowledge distillation is utilized after obtaining the accuracy in the compressed MobileNetV1. The compressed MobileNetV1 network is then jointly trained in the knowledge distillation method with the teacher network. The investigation is studied by incorporating knowledge distillation to show that the expected result should increase as close as the teacher's accuracy. Third, the deep Taylor decomposition is then utilized to show visually that the feature learned by the compressed MobileNetV1 using knowledge distillation has better representation than the vanilla MobileNetV1.

A. FORMULATION

We utilize MobileNetV1 in the experiment. The structure of MobileNetV1 is composed of several layers of depthwise separable convolutional, as shown in Table 1. MobileNetV1 is analysed by compressing into the layerwise and the

TABLE 3. The widthwise compression of MobileNet architecture.

Models	MobileNet	MobileNet_3/4	MobileNet_1/2	MobileNet_1/4
Width Multiplier	1	0.75	0.5	0.25
Type / Stride	Output Shape			
Conv / s2	32×32×3			
Conv / s2	16×16×32	16×16×24	16×16×16	16×16×8
Conv dw / s1	16×16×32	16×16×24	16×16×16	16×16×8
Conv / s1	16×16×64	16×16×48	16×16×32	16×16×16
Conv dw / s2	8×8×64	8×8×48	8×8×32	8×8×16
Conv / s1	8×8×128	8×8×96	8×8×64	8×8×32
Conv dw / s1	8×8×128	8×8×96	8×8×64	8×8×32
Conv / s1	8×8×128	8×8×96	8×8×64	8×8×32
Conv dw / s2	4×4×256	4×4×192	4×4×128	4×4×64
Conv / s1	4×4×256	4×4×192	4×4×128	4×4×64
Conv dw / s1	4×4×256	4×4×192	4×4×128	4×4×64
Conv / s1	4×4×256	4×4×192	4×4×128	4×4×64
Conv dw / s2	2×2×256	2×2×192	2×2×128	2×2×64
Conv / s1	2×2×256	2×2×192	2×2×128	2×2×64
5× Conv dw / s1	2×2×512	2×2×384	2×2×256	2×2×128
5× Conv / s1	2×2×512	2×2×384	2×2×256	2×2×128
Conv dw / s2	1×1×512	1×1×384	1×1×256	1×1×128
Conv / s1	1×1×1024	1×1×768	1×1×512	1×1×256
Conv dw / s2	1×1×1024	1×1×768	1×1×512	1×1×256
Conv / s1	1×1×1024	1×1×768	1×1×512	1×1×256
Avg Pool / s1	1×1024	1×768	1×1×512	1×256
FC / s1	10			
Softmax / s1	10			

widthwise compression. In the layerwise compression, the scheme is to design three compressed networks by eliminating the set of depthwise separable convolution from the 13th, 12th, and 11th layers. MobileNet_cut13 refers to the compressed network in which the 13th depthwise separable convolution is eliminated, while MobileNet_cut12 is the compressed network in which the 13th and 12th depthwise separable convolution layers are reduced. MobileNet_cut11 means the compressed network in which the 13th, 12th, and 11th depthwise separable convolution layers are eliminated. The whole comparison between the structure of MobileNetV1 and the other proposed three compressed networks is shown in Table 2. On another side, the widthwise compression, the scheme is to construct three different compressed networks using parameter tuning: width multiplier. The parameter will multiply the width of the input and the output channel, making the compressed network thinner, for example, MobileNet_3/4; this network refers to the compressed network using a width multiplier of 0.75. Furthermore, MobileNet_1/2 and MobileNet_1/4 mean the compressed network using width multipliers of 0.5 and 0.25. The structure of networks implemented in this scheme is shown in Table 3.

Knowledge distillation, popularized by Hinton *et al.* [15], is a model compression that transfers the knowledge from the large complex neural network model as the teacher network to a small and fast neural network as the student network without decreasing significant accuracy. Our formulation is based on Fig. 1, where the student network is learned from the teacher network by utilizing the soft logits by adding the new hyperparameter, temperature (T), in the distillation loss. To formalize mathematically, given the probability p_i of class

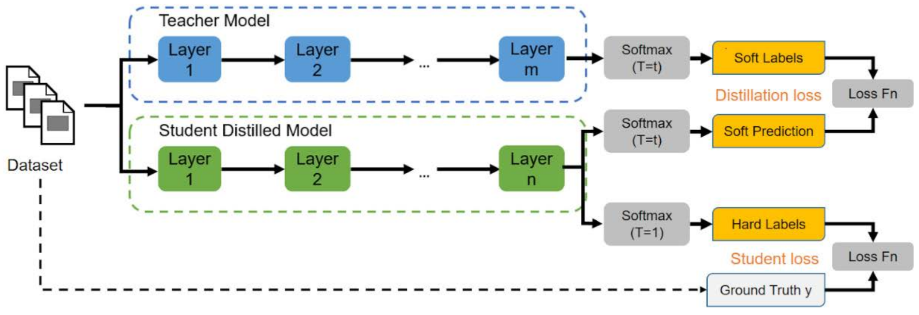

FIGURE 1. **The vanilla knowledge distillation process as stated in [14].**

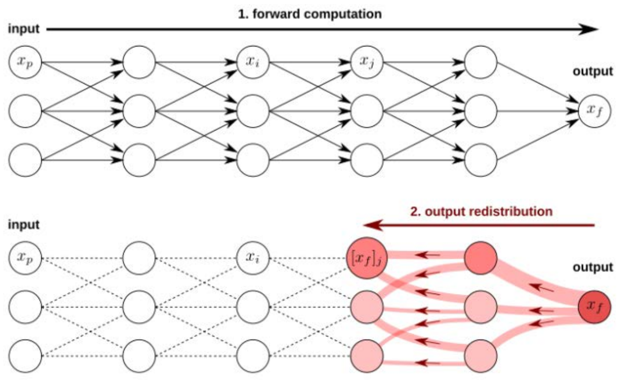

FIGURE 2. **Deep Taylor decomposition scheme in back-propagation pass (retrieved from [34]).**

i is calculated from logit z_i in softmax function as

$$p_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_i}{T}\right)} \tag{1}$$

where T is the temperature parameter. When $T = 1$, Equation 1 becomes the standard class probability of a vanilla neural network. As T grows, the softmax function probability distribution becomes softer and rich in information. This phenomenon is called dark knowledge and is transferred to the student network. The distillation loss is defined as the minimizing of softmax temperature produced by the teacher model called a soft label and the softmax temperature produced by the student model called soft prediction, formally written as

$$D_{loss} = \mathrm{H}\left(\sigma(z_t; T = \tau), \sigma(z_s; T = \tau)\right) \tag{2}$$

where H is the cross-entropy loss function, σ is the softmax function, Z_s and Z_p is the logit of teacher and student networks. The student loss using the standard softmax function called the hard prediction is defined as

$$S_{loss} = \mathrm{H}\left(y, \sigma(z_s; T = 1)\right) \tag{3}$$

where H is the cross-entropy loss function, σ is the softmax function, Z_s is the logit of the student model. The complete loss function of the knowledge distillation process is the weighted average between the distillation loss in Equation 2 and the student loss in Equation 3, defined as

$$\Phi_{KD}(x; W) = \alpha * D_{loss} + \beta * S_{loss} \tag{4a}$$

$$\Phi_{KD}(x; W) = \alpha * \mathrm{H}\left(y, \sigma(z_s; T = 1)\right) + \beta * \mathrm{H}\left(\sigma(z_t; T = \tau), \sigma(z_s; T = \tau)\right) \tag{4b}$$

where α and β are the new hyperparameters, which is $\beta = 1 - \alpha$. In general, Hinton *et al.* introduced three new hyperparameters in the knowledge distillation: T as temperature, α as distillation loss coefficient, and β as student loss coefficient, which is defined as $(1 - \alpha)$.

Moreover, we then train the teacher and the student networks in the layerwise and the widthwise compression using knowledge distillation. A visualization tool is helpful to understand better the classifier and the difference in the impact of learning using knowledge distillation. We utilize the deep Taylor decomposition proposed by Montavon *et al.* [25] as the visualization tool. Deep Taylor Decomposition operates in the backward pass when a network updates weights and propagates its error. This backpropagation operation aims to dissociate the total prediction into sets of localized neuron computations layer by layer and recombine these computations appropriately (chained rule for derivatives is used in the case of error backpropagation). The decomposition must satisfy the conservation property as denoted in Equation 5a and Equation 5b.

$$\sum_p [f(x)]_p = x_f \tag{5a}$$

$$\forall p : [x_f]_p \geq 0 \tag{5b}$$

where x_f is the output neuron, and $(x_p)_p$ denotes the input neuron. The target is to produce a decomposition $([x_f]_p)_p$ of the output in terms of input variables, where $[x_f]_p$ distributes x_f to a particular input variable x_p. The workflow of deep Taylor decomposition is depicted in Fig. 2. In this scenario, x_p is denoted as neuron input, and x_f is neuron output. In the backward pass, $[x_f]_j$ denotes the volume from output x_f relevant to node x_j, and it is going to be redistributed to each neuron x_i in the previous layer (the collection represented as $(x_i)_i$).

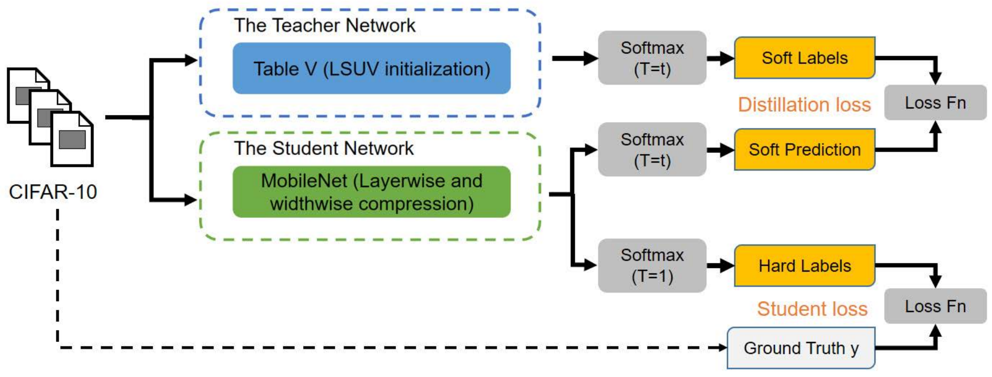

FIGURE 3. The training procedure of MobileNet in the layerwise and widthwise compression.

B. TRAINING PROCEDURE

The learning procedure is comprised of two stages of training. In the first stage, we train the teacher network to minimize the standard softmax function in Equation 1 using T = 1. This initialization is expected to achieve higher accuracy. In the second stage, we train each student network in the layerwise and the widthwise compression using the pre-trained teacher simultaneously, minimizing Equation 4, as shown in Fig. 3.

Our proposed scheme is to analyse the student networks, in this case, the compressed networks in Table 2 and Table 3, based on the layerwise and the widthwise compression trained using the knowledge distillation method. It is crucial to show that knowledge distillation can improve the accuracy and the feature of the compressed MobileNetV1. Note that our scheme to train the student network is directly derived from [15]. We demonstrate that the compressed MobileNetV1 in terms of the layerwise and the widthwise compression improved the performance in Section III. Furthermore, we attempt to visually investigate whether the compressed network could mimic the teacher network behaviour even with a thin structure.

III. EXPERIMENT

This section verifies the effectiveness of the compressed MobileNetV1 and investigates the validity visually using deep Taylor decomposition. Both compressed MobileNetV1 in terms of the layerwise and widthwise compression is trained using knowledge distillation. The result then is compared to the vanilla training. Please note that several hyperparameters in the experiment are introduced consistently for knowledge distillation.

A. EXPERIMENT SETUP

The experiment set up is conducted using a personal computer workstation under the specification of Ubuntu 18.04 LTS operating system, Intel(r)Xeon(R) CPU E5-262 v4 2.10Ghz, 64 GB RAM, NVIDIA GeForce RTX 2080Ti. It utilizes Python 3.6.7 and the framework of Tensorflow-GPU 1.12.2m with Keras 2.2.4 with Tensorflow backend. The learning policy of experiment is set using an SGD optimizer with an initial learning rate of 0.1 and a momentum of 0.9. The loss function uses categorical cross-entropy for the vanilla training and Equation 4 for knowledge distillation. The step decay is selected by the decay rate of 0.9 every ten epochs of 300 epochs.

TABLE 4. The variation of a and T.

Hyperparameters	Selected
Alfa α	0.3; 0.5; 1
Temperature T	1; 3; 5; 10

We utilize CIFAR-10 as the primary dataset in the experiment. CIFAR-10 is composed of 60000 32 x 32 color images in 10 classes, with 6000 images per class. There are 50000 images divided into the training set, while the rest are 10000 images divided into the testing set. These ten categories include airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

As stated in loss function Equation 4b, knowledge distillation requires three new hyperparameters. In our experiment, the new hyperparameters are selected in Table 4. We vary α of 0.3, 0.5, and 1 while the temperature T of 1, 3, 5, and 10. To follow the standard procedure of vanilla knowledge distillation, the teacher network and the student network are required to initialize. As stated, the teacher network should have accuracy as big as possible so that the teacher network can transfer the knowledge effectively.

B. PRELIMINARY STUDY

Our preliminary study shows that MobileNetV1 trained on CIFAR-10 only achieves an accuracy of 51.25%. Due to a lack of accuracy performance, it is not possible to apply MobileNetV1 as the teacher network. We utilized a simple model as the teacher network, as shown in Table 5. The architecture is trained on CIFAR-10 using layer-sequential unit variance (LSUV) initialization, achieving almost 90% accuracy. Layer-sequential unit-variance (LSUV) initialization is a simple method to initialize weights in a neural

TABLE 5. The teacher architecture with LSUV initialization.

Layer	Type / Stride	Output Channel	Layer	Type / Stride	Output Channel
1	Conv 2D	32 × 32 × 32	22	relu	16 × 16 × 80
2	relu	32 × 32 × 32	23	Max Pooling	8 × 8 × 80
3	Conv 2D	32 × 32 × 32	24	Dropout	8 × 8 × 80
4	relu	32 × 32 × 32	25	Conv 2D	8 × 8 × 128
5	Conv 2D	32 × 32 × 32	26	relu	8 × 8 × 128
6	relu	32 × 32 × 32	27	Conv 2D	8 × 8 × 128
7	Conv 2D	32 × 32 × 48	28	relu	8 × 8 × 128
8	relu	32 × 32 × 48	29	Conv 2D	8 × 8 × 128
9	Conv 2D	32 × 32 × 48	30	relu	8 × 8 × 128
10	relu	32 × 32 × 48	31	Conv 2D	8 × 8 × 128
11	Max Pooling	16 × 16 × 48	32	relu	8 × 8 × 128
12	Dropout	16 × 16 × 48	33	Conv 2D	8 × 8 × 128
13	Conv 2D	16 × 16 × 80	34	relu	8 × 8 × 128
14	relu	16 × 16 × 80	35	Max Pooling	128
15	Conv 2D	16 × 16 × 80	36	Dropout	128
16	relu	16 × 16 × 80	37	Dense	500
17	Conv 2D	16 × 16 × 80	38	relu	500
18	relu	16 × 16 × 80	39	Dropout	500
19	Conv 2D	16 × 16 × 80	40	Dense	10
20	relu	16 × 16 × 80	41	Softmax	10
21	Conv 2D	16 × 16 × 80	-	-	-

TABLE 6. The accuracy comparison between the teacher network and the MobileNetV1.

Network	Accuracy
Teacher model	89.56%
MobileNet	51.25%

network. In work[26], they proposed two steps to achieve better performance: (1) re-initialize weights in convolution or inner product layers with orthonormal matrices, (2) normalize the variance of the output of every layer to be equal to one. Therefore, a simple neural network proposed in [26] with LSUV initialization plays a role as a teacher network in our scheme. The accuracy comparison of MobileNet and the teacher model is shown in Table 6. It shows that there is a significant difference between the two models.

C. RESULTS

To compare the structural difference and its effects in the layerwise and the widthwise compression, we measure the model size, Floating Point Operations (FLOPs), and inference time. As shown in Table 7, the layerwise compression helps to shorten inference time. It also reduces limited computational costs. In contrast, model size and FLOPs decrease significantly as the width multiplier gets closer to zero, as shown in Table 8. However, it does not speed up inference time at all. To summarize the above observations, the following suggestions are provided for different situations depending on the user requirements. First, one may consider compressing models widthwise if the goal is to reduce the model size and save computational costs. For example, they implement models on the edge computing platform. Second, if the user wants to improve the speed of a model, reducing the number of layers could be considered.

In addition to the compression rate or speed, the accuracy of the compressed models is also essential to consider. After all, it makes no sense if sacrificing performance too much to achieve a specific compression rate. Table 9-10 shows the total combination of both hyperparameters (T and α) but different structures. In this table, the accuracy labeled as red is lower than the same model using the vanilla training (without knowledge distillation). The accuracy highlighted as the yellow background is the highest one compared to the other hyperparameters.

TABLE 7. The comparison of different structure of MobileNetV1.

Models		Model Size	Compression Rate	Number of Parameter	FLOPs	Inference Time
MobileNetV1		13.213 MB	-	3,239,114	6,533,228	12.254 ms
Widthwise compresision	3/4	7.628 MB	42.27%	1,840,666	3,722,652	12.641 ms
	1/2	3.594 MB	72.80%	834,666	1,696,972	12.336 ms
	1/4	1.138 MB	91.38%	221,114	456,188	12.336 ms
Layerwise compression	cut 13	8.930 MB	32.42%	2,173,130	4,390,998	11.848 ms
	cut 12	6.751 MB	48.91%	1,632,970	3,302,976	10.993 ms
	cut 11	5.651 MB	57.23%	1,362,122	2,756,138	9.980 ms

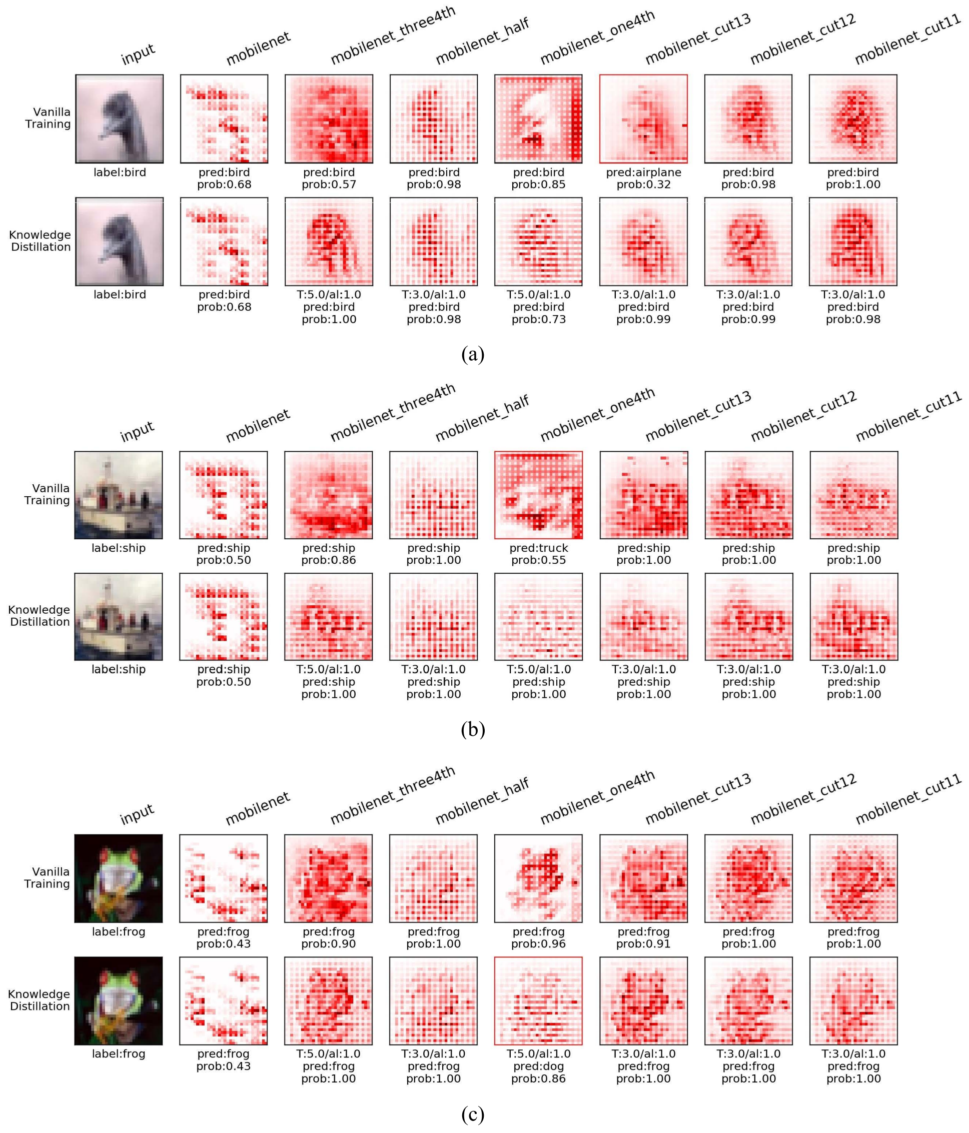

FIGURE 4. Heatmap of different sample labels (a) bird, (b) Ship, (c) frog.

From the viewpoint of the structure of models, we could state two conclusions. First, the performance of MobileNetV1 on CIFAR-10 is restricted to the complexity of structure. One could say that this dataset does not need models with too

TABLE 8. The detail of FLOPs.

Models		Mul	Add	Sub	AssignSub	RealDiv	Assign Add	Power
MobileNet		3.23 M	3.20 M	43.94 K	43.78 K	21.91 K	54	54
Widthwise compresision	3/4	1.83 M	1.81 M	33.00 K	32.83 K	16.44 K	54	54
	1/2	829.18 K	812.77 K	22.05 K	21.89 K	10.97 K	54	54
	1/4	218.37 K	210.16 K	11.11 K	10.94 K	5.50 K	54	54
Layerwise compression	cut 13	2.16 M	2.14 M	35.74 K	35.58 K	17.82 K	50	50
	cut 12	1.63 M	1.60 M	29.58 K	29.44 K	14.74 K	46	46
	cut 11	1.36 M	1.34 M	25.47 K	25.34 K	12.69 K	42	42

many layers. Therefore, the accuracy increased as the sets of depthwise separable convolution is cut more and trained these models with original Vanilla Training. Second, knowledge distillation helps boost accuracy, but it has constrained influence models with the layerwise method. It has limited improvements, mostly from the fluctuation, when launching experiments several times with identical environmental settings and hyperparameters.

From the viewpoint of hyperparameters, several conclusions could be made. First, most of the red-labeled accuracy is distributed in T of 1, where the unsoftened class probabilities are used in the training procedure. It is evident since when T of 1 is selected, the knowledge distillation loss function in Equation 4 becomes the standard categorical cross-entropy. Second, most of the yellow-background accuracy is distributed in T of 3 and 5. None of them situates in T of 10. This implies that it would not yield better performance even if we set a higher temperature. Third, all of the yellow-background accuracies lie in the row of $\alpha = 1$, where the loss relies on totally on soft loss in Equation 4.

The accuracy tables can show overall model performance, but it cannot tell specific advantages or disadvantages of them. As a consequence, deep Taylor decomposition is utilized to show their specific and unique function. In Fig. 5, the first-row models use Vanilla Training, while those in the second-row use Knowledge Distillation. Each column in the second raw loads model's checkpoint file is the yellow-background model in Table 9-10, which is that the model performs the best compared with all the other models with identical structure but different hyperparameters. The first column is the input image, and the rest columns show how these classifiers recognize the object. If the frame of a heatmap is red, then this model is mispredicted.

According to the visualization of the heatmap, several phenomena are observed. First, from the perspective of structural differences, we can analyse using the layerwise and widthwise compression. The layerwise compression will not decrease the classifier's resolution, but it will increase/decrease the number of features that the classifier

TABLE 9. The layerwise compression accuracy.

Student Model (MobileNet)			Layerwise Compression		
			cut13	cut12	cut11
Vanilla training			73.96%	83.97%	84.68%
Knowledge Distillation	T = 1	α = 1	68.83%	84.31%	84.09%
		α = 0.5	76.00%	84.19%	84.42%
		α = 0.3	74.56%	84.58%	84.80%
	T = 3	α = 1	**84.38%**	**85.31%**	**85.32%**
		α = 0.5	82.40%	84.81%	84.78%
		α = 0.3	77.51%	84.68%	84.99%
	T = 5	α = 1	83.81%	84.79%	83.87%
		α = 0.5	82.05%	84.16%	85.15%
		α = 0.3	80.37%	84.36%	85.06%
	T = 10	α = 1	83.03%	83.36%	82.92%
		α = 0.5	82.04%	84.55%	85.15%
		α = 0.3	79.76%	84.33%	84.55%

can detect. For example, the classifier with fewer layers identifies more features in Fig. 4a-b. However, it will focus on fewer features of the object in Fig. 4c. The layerwise compression releases the classifier from overwhelming of too many unnecessary features. On the other side, the widthwise compression responds that the lower the width of the output channel, the lower the classifier's resolution is. In other words, as the width of the output channel decreases, we can find out that the heatmap becomes blurry, and the magnitude of noise gets close to that of the object itself. This situation means that cutting too much output channel width will confuse the classifier with the foreground (the object aims to detect) and background. Second, we analyse from

TABLE 10. The widthwise compression accuracy.

Student Model (MobileNet)			Widthwise Compression		
			3/4	1/2	1/4
Vanilla training			72.86%	67.04%	70.52%
Knowledge Distillation	T = 1	α = 1	62.43%	66.42%	69.24%
		α = 0.5	51.31%	68.94%	68.77%
		α = 0.3	70.71%	33.06%	67.50%
	T = 3	α = 1	82.30%	**80.77%**	73.09%
		α = 0.5	78.60%	74.19%	70.87%
		α = 0.3	74.08%	76.85%	68.78%
	T = 5	α = 1	**82.72%**	80.49%	**75.23%**
		α = 0.5	79.47%	79.43%	70.67%
		α = 0.3	61.81%	76.46%	71.36%
	T = 10	α = 1	81.12%	79.42%	73.43%
		α = 0.5	80.35%	77.97%	69.48%
		α = 0.3	78.05%	73.92%	68.25%

the perspective of using vanilla training and knowledge distillation training. Knowledge distillation enables classifiers to eliminate more noise. Therefore, it helps them focus on the object themselves, as MobileNet three4th and MobileNet one4th in Fig. 4a-b. It also makes classifiers concentrate on more features of objects, as in Fig. 4a-b. Please note that not all the influence of knowledge distillation is good. It sometimes distracts the network with irrelevant and unimportant features, as MobileNet one4th in Fig. 4c.

IV. CONCLUSION

In this work, we focus on analyzing compression techniques in multiple aspects and suggesting how to compress models according to constumer demands. In this scheme, two experiments are demonstrated: model compression and analysis of results. First, we select two kinds of compression to compress MobileNetV1, the layerwise and widthwise compression. Then, Knowledge distillation is deployed after compressing MobileNetV1 in order to improve their performance. Second, we analyse the compression performance in different aspects: model size, speed, computational cost, and use deep Taylor decomposition to identify how networks learn the feature of the image. After the analysis, we conclude that the layerwise compression is suggested for those who intend to speed up models. On the other side, the widthwise compression is recommended for applications on edge devices to save memory or computational cost. Furthermore, if knowledge distillation is considered to be deployed in the scheme of model compression, it would be more helpful if the widthwise compression is selected as the compressed model.

Hyperparameters are fixed in the knowledge distillation of our experiments. There may be some difference to set those hyperparameters dynamic adjusted as the training steps increase, just as the same idea of modifying the learning rate to help models converge better and easier. Moreover, these hyperparameters can be adjusted and decided according to do preliminary experiments to plot soft loss and hard loss and observe how they contribute to the total loss.

REFERENCES

[1] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, pp. 5455–5516, Apr. 2020, doi: 10.1007/S10462-020-09825-6.

[2] Z. Jiang, Z. Dong, W. Jiang, and Y. Yang, "Recognition of Rice leaf diseases and wheat leaf diseases based on multi-task deep transfer learning," *Comput. Electron. Agricult.*, vol. 186, Jul. 2021, Art. no. 106184, doi: 10.1016/j.compag.2021.106184.

[3] X. Zhang, "Application of human motion recognition utilizing deep learning and smart wearable device in sports," *Int. J. Syst. Assurance Eng. Manage.*, vol. 12, no. 4, pp. 835–843, Aug. 2021, doi: 10.1007/S13198-021-01118-7.

[4] G. Jeon and A. Chehri, "Security analysis using deep learning in IoT and intelligent transport system," in *Smart Transportation Systems 2021* (Smart Innovation, Systems and Technologies), vol. 231, X. Qu *et al.*, Eds. Singapore: Springer Nature, 2021, pp. 9–19, doi: 10.1007/978-981-16-2324-0_2.

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

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[9] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, Feb. 2016, pp. 1–7.

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[11] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, May 2019, pp. 1–10.

[12] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 6848–6856, doi: 10.1109/CVPR.2018.00716.

[13] G. Huang, S. Liu, L. V. D. Maaten, and K. Q. Weinberger, "CondenseNet: An efficient DenseNet using learned group convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2752–2761, doi: 10.1109/CVPR.2018.00291.

[14] L. J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2013, pp. 1–9.

[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[16] C. Buciluă, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2006, p. 535, doi: 10.1145/1150402.1150464.

[17] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, pp. 1789–1819, Mar. 2021, doi: 10.1007/S11263-021-01453-Z.

[18] L. Wang and K.-J. Yoon, "Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 6, pp. 3048–3068, Jun. 2022, doi: 10.1109/TPAMI.2021.3055564.

[19] S. Hanson and L. Pratt, "Comparing biases for minimal network construction with back-propagation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 1, 1988, pp. 1–9. [Online]. Available: https://proceedings.neurips.cc/paper/1988/file/1c9ac0159c94d8d0cbedc973445af2da-Paper.pdf

[20] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 2, 1989, pp. 1–8, [Online]. Available: https://proceedings.neurips.cc/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf
[21] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *Proc. IEEE Int. Conf. Neural Netw.*, Mar./Apr. 1993, pp. 293–299, doi: 10.1109/ICNN.1993.298572.
[22] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, Nov. 2015, pp. 1–9.
[23] A. Aghasi, A. Abdi, N. Nguyen, and J. Romberg, "Net-Trim: Convex pruning of deep neural networks with performance guarantee," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3180–3189.
[24] X. Dong, S. Chen, and S. J. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," in *Proc. Adv. Neural Inf. Process. Syst.*, May 2017, pp. 1–11.
[25] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep Taylor decomposition," *Pattern Recognit.*, vol. 65, pp. 211–222, May 2017, doi: 10.1016/j.patcog.2016.11.008.
[26] D. Mishkin and J. Matas, "All you need is a good Init," 2015, *arXiv:1511.06422*.

YU-WEI HONG received the master's degree from the Department of Electronic Engineering and Computer Science, National Taiwan University of Science and Technology. After graduation, she has joined the industry. She is currently working as a Software/Firmware Engineer. Her research interests include model compression and object detection.

JENQ-SHIOU LEU (Senior Member, IEEE) received the B.S. degree in mathematics and the M.S. degree in computer science and information engineering from the National Taiwan University, Taipei, Taiwan, in 1991 and 1993, respectively, and the Ph.D. degree on a part-time basis in computer science from the National Tsing Hua University, Hsinchu, Taiwan, in 2006. He was a Research and Development Engineer with Rising Star Technology, Taiwan, from 1995 to 1997. He was an Assistant Manager with Mobitai Communications and Taiwan Mobile, from 1997 to 2007. In 2007, he joined the Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taipei, as an Assistant Professor. From February 2011 to January 2014, he was an Associate Professor. Since February 2014, he has been a Professor. His research interests include heterogeneous networks and mobile services over heterogeneous networks.

MUHAMAD FAISAL received the B.E. degree in electrical engineering from the Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia, and the M.S. degree from the Graduate Institute of Automation and Control, National Taiwan University of Science and Technology, Taipei, Taiwan. He is currently pursuing the Ph.D. degree with the Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology. From 2016 to 2019, he was a Project Engineer at Berca Schindler Lifts, Indonesia. His current research interests include the intelligent control systems, robust control systems, machine learning, deep learning, the Internet of Things, and natural language processing. He received the NTUST Scholarship Award.

SETYA WIDYAWAN PRAKOSA received the B.Eng. degree *(cum laude)* from Jember University, Jember, Indonesia, in 2013, and the M.S. degree from the Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, in 2018. He is currently pursuing the Ph.D. degree in electronic and computer engineering with the National Taiwan University of Science and Technology. His research interests include pattern analysis on agricultural and medical data, machine learning, and deep learning.