On Merging MobileNets for Efficient Multitask Inference

Cheng-En Wu
Institute of Information Science, Academia Sinica; MOST Joint Research Center for AI Technology and All Vista Healthcare
chengen@iis.sinica.edu.tw

Yi-Ming Chan
Institute of Information Science, Academia Sinica; MOST Joint Research Center for AI Technology and All Vista Healthcare
yiming@sinica.edu.tw

Chu-Song Chen
Institute of Information Science, Academia Sinica; MOST Joint Research Center for AI Technology and All Vista Healthcare
song@sinica.edu.tw

Abstract

When deploying two or more well-trained deep-learning models on a system, we would hope to unify them into a single deep model for the execution in the inference stage, so that the computation time can be increased and the energy consumption can be saved. This paper presents an effective method to build a single deep neural network that can execute multiple tasks. Our approach can merge two well-trained feed-forward neural networks of the same architecture into a single one, where the required on-line storage is reduced and the inference speed is enhanced. We evaluate our approach by using MobileNets and show that our approach can improve both compression ratio and speedup. The experimental results demonstrate the satisfactory performance and verify the feasibility of the method.

Keywords Deep learning, CNNs, Model compression.

ACM Reference Format:
Cheng-En Wu, Yi-Ming Chan, and Chu-Song Chen. 2019. On Merging MobileNets for Efficient Multitask Inference. In *Proceedings of ACM SIGPLAN Conference on Programming Languages (EMC^2).* ACM, New York, NY, USA, 5 pages.

1 Introduction

In recent years, compact convolutional neural network (CNN) models have been introduced to improve the computational efficiency and storage required for the inference stage, such as MobileNet [7], Xception [4], and ShuffleNet [14]. With these condensed architectures, deep learning can be performed in a more light-weight and energy-efficient manner.

A deep-learning model is often trained for a specific purpose; hence, it is suitable for one task only. However, in real applications, we usually require to achieve not only a single functionality — Given two well-trained deep-learning models skillful for different purposes, we would need to perform the two tasks simultaneously in the inference stage.

In this paper, we study the problem: how to merge or unify well-trained models so that the two tasks can be performed more efficiently? Merging well-trained neural networks is potentially useful for real-world applications. For example, different types of visual classification tasks, such as clothing recognition and object classification, could be involved in an intelligent robot. Two models, one for clothing recognition and the other for object classification, may be available for the individual purposes. Merging them on the inference stage is helpful to build a smart system that is faster and more energy efficient than running them separately when deploying these models.

The goal of this study is to develop a general approach that can merge two feed-forward neural networks having the same architecture. In particular, we focus on merging two compact networks, MobileNets into a single one in this paper, and the principle can be applied to other types of deep learning models as well. The merged network can manage the original tasks simultaneously when deploying these models. When merging two MobileNets, our approach aligns the models layer by layer. The layers aligned are fused into a single layer, and the merged model can be fine-tuned for the performance recovering.

1.1 Motivation of Our Approach

To handle multiple tasks with a single model, a typical way is to use a single network (such as MobelNet), and establish multiple branches in the output layer of the network. The multiple tasks share the same previous layers that extract the feature (or embedding) representations, and the outputs of different tasks are predicted according to the same feature representations. When two well-trained models are provided for different tasks, we could use the weights of one task as the pre-trained initials, and fine-tune this output-branching network to build a multitask model. An illustration of this baseline approach is illustrated in Figure 1.

The above baseline approach is easy to be implemented but has some weaknesses. First, it implicitly assumes that the deepest feature representations are shared, which would be suitable only when the tasks are highly co-related. Second, the initial weights are from one of the models, which could be bias for that task. In this work, we propose a "zippering" approach that allows to use part of the previous layers to form the common backbone network for the feature representations sharing. The approach is motivated by

EMC^2, 2019, Washington D.C.,USA
2019.

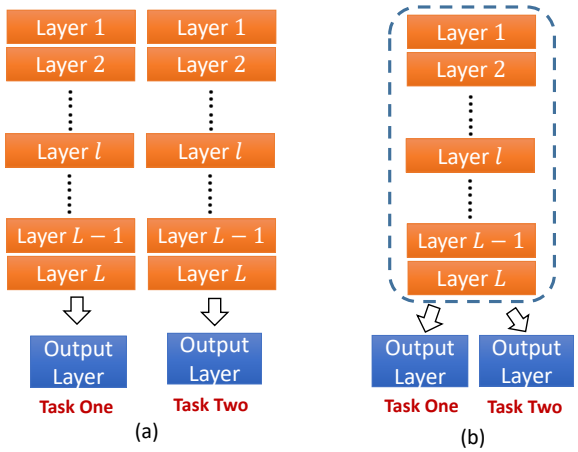

Figure 1. Baseline approach. (a) Two well-trained feed-forward neural networks. (b) Using one network as the feature-extraction backbone, and the extracted feature representation is shared by the two forked output layers.

the following reasons. First, the feature representations suitable to be shared for the two tasks would not be from the deepest layer, but some previous layer that extracts earlier-stage representations. Second, as we do not know how many previous layers are suitable to be shared beforehand, we conduct a progressive process (called the zippering process) that merges the first k layers incrementally for $k = 1, \cdots, L$, with L the number of total layers of the network. The zippering process can thus find an appropriate number of early layers for sharing automatically.

1.2 Overview of Our Approach

Because the architectures of the two deep-learning models are assumed to be the same in this work, we set the same layer of the two models as a pair. As the models are well-trained for different goals, the layer weights are different. For the pair of layers to be fused in the zippering process, We first merge them into a single layer with a common set of weights holding the smallest quantization error. Then, we fine-tune the entire model through minimizing both the classification loss and the output-preserving loss per each layer. In the zippering process, the first layers are merged and fine-tuned in the beginning, and then the other layers are merged and fine-tuned in turn sequentially. An overview of our approach is illustrated in Figure 2.

The rest of this paper is organized as follows. In Section 2, we give a review of related works. In Section 3, we introduce our zippering approach for merging two MobileNets. Experimental results are presented in Section 4. Finally, conclusions and discussions are given in Section 5.

2 Related Work

In this section, we briefly review two related topics of this study, multi-task learning and neural model merging.

2.1 Multi-task Deep Learning

A typical way to achieve various tasks via a single neural-net model is to train a new neural network with increased output nodes for multi-tasks. A joint facial age estimation and expression classification model is proposed in [10]. Learn-them-all approaches can solve multi-tasks across various domains with a universal model. In [8], MultiModel architecture is introduced to allow input data to be images, sound waves, and text of different dimensions, and then converts them into a unified representation. The convolution, attention, and sparsely-gated mixture-of-experts layers are incorporated to handle various problems. In [2], a deep CNN leverages massive synchronized data to learn an aligned representation. The aligned representation can be shared across modalities for multi-modal tasks. Applying the learn-them-all approaches, however, has to face heavy training effort and inference complexity.

2.2 Neural Network Merging

There are still few studies focusing on merging well-trained deep learning models into a single and more condensed model [3][5]. As also claimed in Bazrafkan and Corcoran [3], "there has been little research concerned with techniques for merging and optimizing a combination of existing deep neural network approaches into a more holistic solution." In the study of [3], a network merging example of sharing common nodes of multiple networks for a single task is discussed. They claim that this ensemble of different architecture can improve the performance while sharing layers with the same kernel configuration to reduce the complexity. Nevertheless, the work in [3] is designed only on a single task model, not for multi-task models.

In Chou et al. [5], a systematic approach is proposed for merging and co-compressing multiple multile CNN models, where the architectures of the CNN models are allowed to be different. This approach decomposes the convolution kernels of different spatial sizes into common 1×1 convolutions of a shorter depth, and a shared codebook is conducted to build the 1×1 kernels used for different tasks. It can achieve a considerable speedup and a high ratio of model size compression on the CPU implementation in the inference stage, while the testing accuracy remains approximately the same. However, the on-line memory usage is increased because an extra codebook is used in this approach. Besides, the approach decomposes a convolution kernel into multiple spatially 1×1 kernels, and a shifting and summation formulation is applied to recover the original convolution result. However, such a shifting-adding mechanism is not supported in current deep-learning frameworks (such as TensorFlow, PyTorch, Caffe), and thus a low-level implementation should be done to support the merging process.

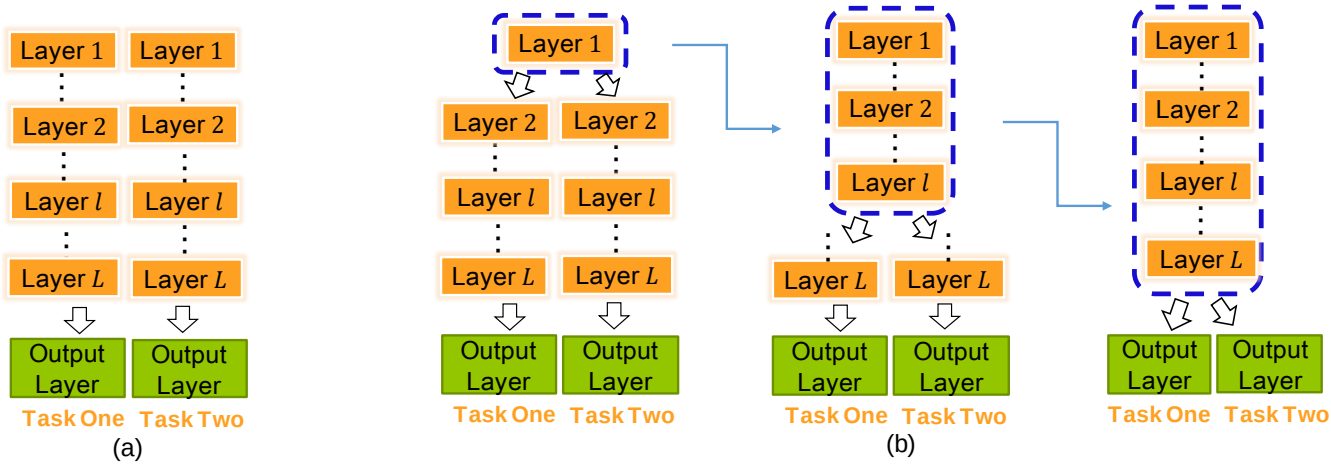

Figure 2. System overview of the zippering process of merging two networks. (a) Two original networks with the weights trained in advance. (b) Fuse the first layers of the two models at first, and fine-tune the weights of the entire model through calibration training (i.e. with both classification loss and per-layer output-preserving loss used). Then the two models are sequentially fused with more layers and fine-tuned in the same manner, until all the layers are fused. A sequence of merged models can thus be generated via the zippering process, and one of them can be chosen for the deployment according to the tradeoff between memory usage and accuracy drop.

In this paper, we propose a new approach for merging two feed-forward deep-learning models. Although the introduced approach requires that the models to be merged are of the same architecture, the merged model can be realized via existing deep-learning frameworks (where TensorFlow is used in this work) and achieves both speedup and the reduction of run-time memory usage.

3 Zippering MobileNets

We introduce an iterative method to merge MobileNets for efficient inference. In round l of the iteration, we merge the previous l layers. The merged layers serve as a common backbone network for feature extraction of the subsequent two branches of tasks, as illustrated in Figure 2(b). During each round, there are two stages of operations. **Shared-weight initialization**: First, the unified weights are initialized so that the weights of different neural networks can be used jointly. **Model Calibration**: Second, the weights of the merged models are calibrated with the training data to restore the performance. Following the concept of distilling[6], our method employs the original models to guide the calibration in this stage.

3.1 Shared Weight Initialization

To effectively initialize the weights of the fused convolutional layers, the basic idea is to leverage the well-trained models given. Consider a convolutional layer in model A and model B. Assume that they have $k_{hA} \times k_{wA} \times N_A \times C_A$ and $k_{hB} \times k_{wB} \times N_B \times C_B$ convolutional filters, respectively. Here, k_h and k_w are the kernel size, N is the number of samples, and C is the number of output channels. Here, we simply initialize the weights by taking the average of the original weights as follows, which results in the smallest quantization error of the weights:

$$\mu_i = \frac{W_{A_i} + W_{B_i}}{2}, \; i \in \{1 : C\}, \tag{1}$$

where W_{A_i} and W_{B_i} are vectors of one of convolutional filter with respect to the i-th output channel of model A and model B, respectively, and μ_i is the mean of vector served as the fused weights.

Each layer of the MobileNet consists of two kinds of convolutional layers, depthwise-separable and point-wise convolutions. Note that the computational overhead of the depthwise-separable convolutions is considerably lower than that of the point-wise convolutions in MobileNet. Besides, the depthwise-separable convolutions serve as the main filters to extract the spatial information of the input tensor in each layer. Hence, we fuse only the point-wise convolutions, and keep the depthwise-separable convolutions as independent branches in the fused layer of the merged model.

3.2 Weight Calibration

Once the fused weights of are available, we set them as initial values to merge the models for multiple tasks and further fine-tune the entire model from all training data through end-to-end back-propagation learning. In this weight-calibration stage, we follow the concept of distilling dark knowledge of neural networks in [6]. The cross-entropy losses of both tasks, together with the L_1 loss that reflects the output difference of every point-wise convolution layers, are summed

to fine-tune the fused model. Accordingly, the optimization problem is equivalent to:

$$\mathbf{w}^* = \min_{\mathbf{w}}[L_c A(\mathbf{w}) + L_c B(\mathbf{w}) + L_d A(\mathbf{w}) + L_d B(\mathbf{w})] \quad (2)$$

where $\mathbf{w}$ are the parameters of the convolutional filters, L_c is individual cross-entropy loss, and L_d is the L_1-norm differene between the feature maps generated by the merged model and the well-trained individual models of each layer.

The above procedures, shared-weight initialization and weight calibration are alternatively performed when the top l layers are fused, with l increased by one at each round in our zippering process, as shown in Figure 2(b). Once the zippering process is finished, a sequence of merged models are obtained. They can then be flexibly chosen for the inference stage via the tradeoff of resource/speed and accuracy.

4 Experiment

The first experiment is to merge two models derived from the same MobileNet [7] architecture for image classification on four datasets:

ImageNet [9] contains 1,000 classes of objects with 1,281,144 training samples and 50,000 testing samples.

DeepFashion [11] has 50 classes of clothing categories with 289,222 training samples and 40,000 testing samples.

CUBS Birds [13] contains 196 classes of bird categories with 5,994 training samples and 5,794 testing samples.

Flowers [12] consisting of 102 classes of flower categories with 2,040 training samples and 6,149 testing samples.

We implement the proposed method with MobileNet as the backbone network using the TensorFlow [1] framework. MobileNet has 13 blocks including depthwise and point-wise layers. We merge each block from block 1 through block 10 incrementally to evaluate performance of the model, where only point-wise layers are merged as depicted in Section 3.1. Table 1 summarizes the experimental results of image classification on ImageNet and DeepFashion datasets.

In Table 1, we denote Merged Blocks as the interval of blocks to be merged, Comp. as compression ratio, Speedup as improvement of inference time, and Acc. as top-1 accuracy of image classification. Top-1 accuracy of an well-trained network for individual task on ImageNet and DeepFashion is 71.02% and 66.21%, respectively. In the case of merged block from 1 through 4 incrementally, proposed model achieves accuracy drop less than 2% compared to the individual networks per task. Moreover, there is the substantial increment in speedup about from 1.08× to 1.24× and the reduction in the usage of runtime memory about from 1.09× to 1.23×. The results reveal that our approach can exploit redundancy and demonstrate satisfactory compression ratio and speedup with low accuracy drop. In Table 1, the details of accuracy on different task are shown to guide the user for the tradeoff between speed and accuracy.

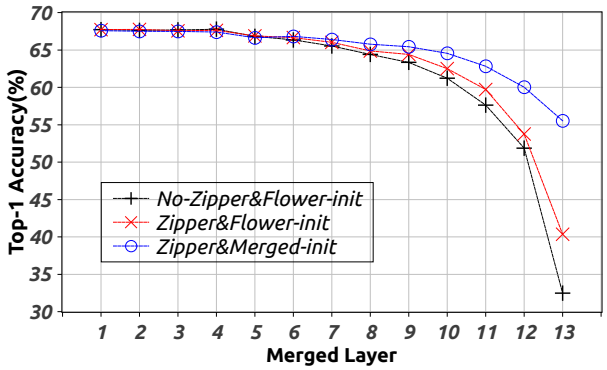

Figure 3. Top-1 classification accuracy in CUBS Birds dataset. Zipper&Merged-init is our method. No-Zipper&Flower-init's initial weights come from the Flower model and uses traditional fine-tune, while Zipper&Flower-init uses zippering process.

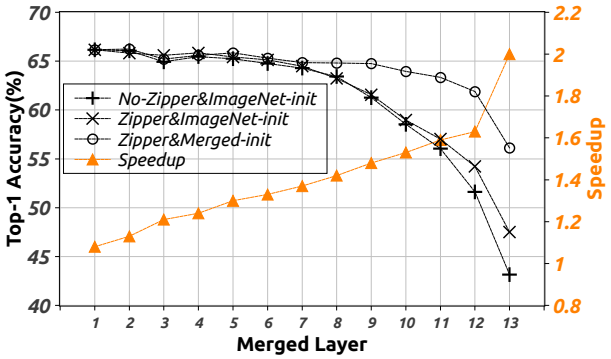

Figure 4. The left axis is the top-1 classification accuracy in DeepFashion dataset, while the right axis is the speedup ratio. The description of three black lines as Figure 3. The orange line is speedup of the merged model.

Figure 3 shows the effect of different initial weights and merging strategies when fusing different layers of MobileNet of Flowers and CUBS task. If the merged network is initialized with Flower task, the performance drop suddenly when merging ten more layers. When using our "zippering" strategy, it can be improved more than 8% when two networks merged. It can even perform better when the initial weights are chosen according to equation (1). The Fig. 4 shows the effect of merging MobileNets from ImageNet and DeepFashion. It reveals that weight trained from ImageNet is not the best choice for all tasks. Our strategy can outperform ImageNet initial weights with more than 13% in DeepFashion dataset. The speedup ratios are also shown in the right hand side of the Fig. 4. It is generally linear respect to the number of merged layers.

To show the importance of initial weights, the loss in DeepFashion training is shown in Fig. 5. Our approach converges

Table 1. Merge performance on different tasks, and the compression and speedup ratio of different merged layers.

Merged Layer	Comp.	Speedup	Runtime memory usage Comp.	Acc.(%) (ImageNet[1])	Acc.(%) (DeepFashion[1])	Acc.(%) (CUBS[2])	Acc.(%) (Flowers[2])
0	1.000	1.00	1.00	71.02	66.21	68.01	91.70
1(First)	1.000	1.08	1.09	70.55	66.24	67.58	91.63
1~2	1.002	1.13	1.14	70.22	66.17	67.50	91.33
1~3	1.004	1.21	1.19	69.65	65.95	67.48	90.82
1~4	1.009	1.24	1.23	69.46	65.59	67.40	91.02
1~5	1.020	1.30	1.27	68.94	65.85	66.63	91.18
1~6	1.042	1.33	1.28	68.53	65.31	66.82	90.75
1~7	1.089	1.37	1.30	68.25	64.85	66.40	90.53
1~8	1.140	1.42	1.32	67.75	64.81	65.77	90.16
1~9	1.196	1.48	1.34	67.34	64.75	65.43	89.74
1~10	1.258	1.53	1.37	66.66	63.95	64.55	88.40
1~11	1.258	1.53	1.39	65.70	63.34	62.82	86.66
1~12	1.258	1.53	1.40	64.35	61.89	60.02	84.00
1~13(All)	1.258	1.53	1.41	61.63	56.12	55.53	80.99

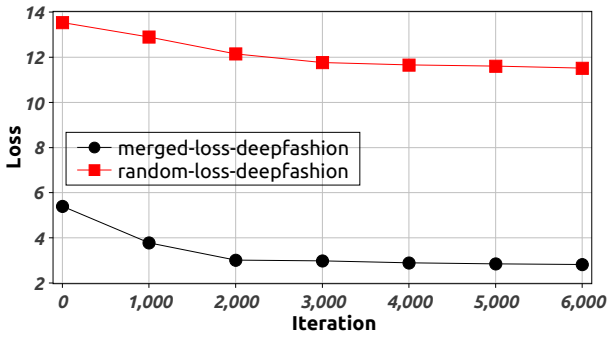

Figure 5. Convergence of different initial methods. The loss convergence of random or "merged" initial methods are compared in DeepFashion dataset. "merged" is initialized using our approach.

at about 2000 iterations, while random initialization struggle at a higher loss with a slower convergent speed.

5 Conclusion

We present a zippering process to unify two well-trained feed-forward networks with the same architecture into a single one that can perform multiple tasks. Compared to the two individual networks, this single network can reduce the run-time memory required and boost the speed for the inference. We take MobileNets as an example to evaluate our approach and demonstrate the improvements in the model size, run-time memory usage, and speedup. The experimental results reveal that our method can provide a flexibility choice for model merging and achieves satisfiable performance.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv* (2016).

[2] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. 2017. See, Hear, and Read: Deep Aligned Representations. *CoRR* abs/1706.00932 (2017). http://arxiv.org/abs/1706.00932

[3] S. Bazrafkan and P. M. Corcoran. 2018. Pushing the AI Envelope: Merging Deep Networks to Accelerate Edge Artificial Intelligence in Consumer Electronics Devices and Systems. *IEEE Consumer Electronics Magazine* 7, 2 (March 2018), 55–61. https://doi.org/10.1109/MCE.2017.2775245

[4] François Chollet. 2017. Xception: Deep Learning with Depthwise Separable Convolutions. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 1800–1807.

[5] Yi-Min Chou, Yi-Ming Chan, Chih-Yi Chiu, and Chu-Song Chen. 2018. Unifying and Merging Well-trained Deep Neural Networks for Inference Stage. In *International Joint Conference on Artificial Intelligence*.

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. *NIPS Workshops* (2014).

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

[8] Lukasz Kaiser, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One Model To Learn Them All. *CoRR* abs/1706.05137 (2017). http://arxiv.org/abs/1706.05137

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[10] Gil Levi and Tal Hassner. 2015. Age and gender classification using convolutional neural networks. In *CVPR Workshops*.

[11] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. 2016. DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[12] M-E. Nilsback and A. Zisserman. 2008. Automated Flower Classification over a Large Number of Classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*.

[13] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. 2011. *The Caltech-UCSD Birds-200-2011 Dataset.* Technical Report CNS-TR-2011-001. California Institute of Technology.

[14] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.