MERGING WELL-TRAINED DEEP CNN MODELS FOR EFFICIENT INFERENCE

Cheng-En Wu*, Jia-Hong Lee*, Timmy S.T. Wan*, Yi-Ming Chan* and Chu-Song Chen*†‡
* Institute of Information Science, Academia Sinica, Taipei, Taiwan
E-mail: {chengen, honghenry.lee, timmywan, yiming, song}@iis.sinica.edu.tw
† MOST Joint Research Center for AI Technology and All Vista Healthcare, Taipei, Taiwan
‡ Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan

***Abstract*—In signal processing applications, more than one tasks often have to be integrated into a system. Deep learning models (such as convolutional neural networks) of multiple purposes have to be executed simultaneously. When deploying multiple well-trained models to an application system, running them simultaneously is inefficient due to the collective loads of computation. Hence, merging the models into a more compact one is often required, so that they can be executed more efficiently on resource-limited devices. When deploying two or more well-trained deep neural-network models in the inference stage, we introduce an approach that fuses the models into a condensed model. The proposed approach consists of three phases: Filter Alignment, Shared-weight Initialization, and Model Calibration. It can merge well-trained feed-forward neural networks of the same architecture into a single network to reduce online storage and inference time. Experimental results show that our approach can improve both the run-time memory compression ratio and increase the computational speed in the execution.**

***Index Terms*—Convolutional Neural Networks (CNNs), Multitask Learning, Co-compression of CNN Models, Fusing CNN Models, Merging CNN models**

I. INTRODUCTION

Deep neural networks play a main role in recent signal processing applications. To handle various tasks, deep models have been trained with particular datasets. Hence, they are often effective for individual tasks or specific purposes. Given different deep models (that have been well-trained with data), we hope to merge them into a single model. The merged neural network is capable of handling different tasks of the original networks simultaneously.

Deep model merging has great potential for applications in real-world problems. For example, in a face-related human-machine interaction system, various facial understanding tasks such as face recognition [1], [2], gender classification [3], [4], and facial expression prediction [5], [6], could be involved. In a multi-modal system, deep learning models based on heterogeneous signal sources (e.g., face image and speaker's voice) would need to be combined for human identity verification [7]. Merging well-trained deep models on the inference stage is beneficial to building a more efficient smart-system than running them separately.

To train a multitask model, a typical way is to construct a new architecture with multitasking outputs in the final layer at first, and then train this model with the union of training data of all tasks. However, such an approach requires a long-period trial-and-error procedure since the architecture selected could be inappropriate in the beginning, and multiple re-training processes are needed in every trial. Even if neural network architecture search (NAS) [8], [9], [10] techniques can find appropriate architectures, it is still demanding to conduct a satisfied deep-learning model for multiple tasks.

In this paper, we develop a general approach that can merge multiple feed-forward neural networks having the same architecture. The merging process is a gradual ensemble method that unifies the first several hidden layers of the networks incrementally, while a traditional ensemble method only uses the network's collective output. The merged network can then handle the original tasks simultaneously.

Note that when deep neural networks are learned, they often have much redundancy in the network weights. To exploit the redundancy, various works compress a neural network or design newly a condensed architecture [11], [12], [13] so that the network model can be executed more efficiently. In our work, the underlying principle adopted is that we remove the co-redundancy among the learned models for merging. A more compact model can thus be conducted by using our approach for efficient inference. When merging two networks, the models are aligned layer by layer in our approach. The layers are fused into a single layer with an error-minimized initialization. Then the merged model is fine-tuned and the performance can be recovered gradually.

The rest of this paper is organized as follows. Section II reviews related works and motives for our approach. Section III-A presents the details of our approach. Section IV gives the experimental results. Finally, Section V concludes this paper.

II. RELATED WORK AND METHOD OVERVIEW

We briefly review multitask learning. Then, we motivate the idea of our approach and give an overview of it.

A. Multitask Deep Learning

Multitask learning is widely used in multimedia signal processing [14], [15], which aims at making the model increase the generalization power through sharing similar representations from related tasks. To achieve multitask deep learning, a typical way is to build a network with numbers of task-specific output nodes. However, this naive strategy assumes that all tasks are relevant enough to prevent the negative transfer from deteriorating the performance of the multitask network. To relax the constraint, [15] designs a multitask network where lower layers are shared and upper layers are task-specific sub-networks, not just a task-specific node.

Nevertheless, despite such a partial sharing strategy works in some cases, it requires prior knowledge like task affinity and weighted loss for each corresponding task in advance. To further address the issue of relatedness among tasks, [16] proposes a task grouping mechanism to search a multitask network architecture adaptively through estimating affinity among tasks. To determine an appropriately weighted loss, some works [17], [14] propose different strategies to alter the loss-weight for each task dynamically during training.

To delve into the branches of multi-task learning, multi-modal learning is also discussed in some cases like [18], [19], [20], [21]. Unlike the aforementioned approaches using a single modality only, multi-modal multitask learning is to make the learner handle multiple tasks using multiple modalities. In emotion recognition, [22] proposes an approach to predict continuous emotion in terms of arousal, valence, and likability using acoustic, visual, and texture features from human simultaneously.

B. *Motivation of Our Approach*

To handle multiple tasks with a single model, a straightforward way is to use a single network and fork multiple outputs associated with the tasks.

When well-trained models are provided for the tasks, we could choose one task as the pre-trained model and fine-tune the output-branching network for all tasks to build a multitask model. An illustration of this baseline approach is given in Figure 1(b). The approach is easily realizable but implicitly assumes that the deepest feature representations are shared, which would work only when the tasks are highly related. Besides, the initial weights are set from one of the models, which could be biased for that task.

Better performance can likely be achieved by sharing part (instead of all) of the layers only. To address the problem, we propose a "zippering" approach that allows using part of the previous layers to form the common backbone network. Since the number of previous layers suitable to be shared is unknown in advance, we manage a progressive process (the zippering process) that merges the first k layers incrementally for $k = 1, \cdots, L$, where L is the number of total layers of the network. The zippering process can, therefore, find a suitable number of layers for sharing automatically.

III. Zippering Neural Networks

As mentioned, the models to be merged are assumed to be of the same architecture and have been well trained for individual goals. We set the same layer of the two models as a pair to be fused, where the weights (or filters) of the layers in a pair are different initially. In the zippering process, we

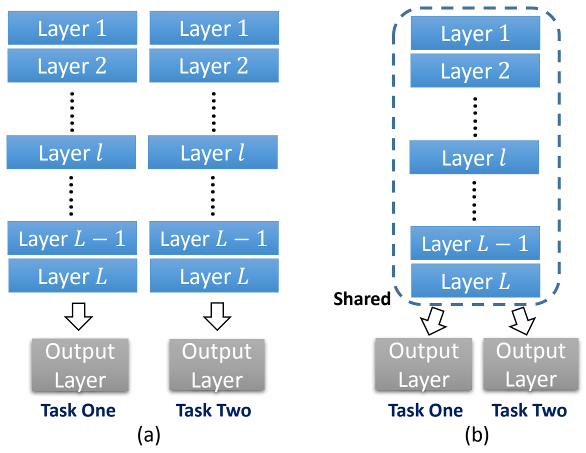

Fig. 1. (a) Two original models. (b) Baseline method: merged the models with the deepest features shared for the two tasks.

develop a filter permutation procedure to align the convolution kernels (filters), so that similar filters can be matched and coupled. Then, we fuse the two layers with the permuted filters and initialize the common filter weights of the fused layer. Finally, we fine-tune the merged model by minimizing the classification loss through fine-tuning. In the zippering process, the previous layers are fused and fine-tuned in the beginning, and then the later layers are fused sequentially. Hence, an incrementally merged model can be obtained. An overview of the zippering process is illustrated in Figure 2. In round l of the iteration, we merge the previous l layers. The merged layers serve as a common network for feature extraction of the subsequent branches of tasks, as illustrated in Figure 2(b). In each round, three operations are performed.

Filter Alignment: First, to minimize the error of the merged layer, the best order of filters is found.

Shared-weight initialization: Second, unified weights are initialized to be used concurrently for different networks.

Model Calibration: Third, the weights of the merged models are calibrated to reclaim the performance.

A. *Filter Alignment*

In convolutional neural networks (CNNs), a set of convolution kernels (filters) are applied to the input tensor in a layer. Assume that there are N such filters in a convolution layer. If we permute these N filters, the channels in the output tensor of this layer are permuted accordingly. Note that the output tensor of the current layer is the input tensor of the next layer in general (except that the current layer is the final layer). If we permute the filters in the current layer and shuffle the channels of the filters in the next layer with the same permutation order, the final output of the CNN remains invariant.

More specifically, let us consider a typical convolution layer in a feed-forward neural network (without loss of generality, assume that a padded convolution of stride 1 is used):

Input $x \in \mathrm{R}^{C \times W \times H}$: the tensor fed to this layer, with W, H the spatial size and C the depth (i.e. number of channels).

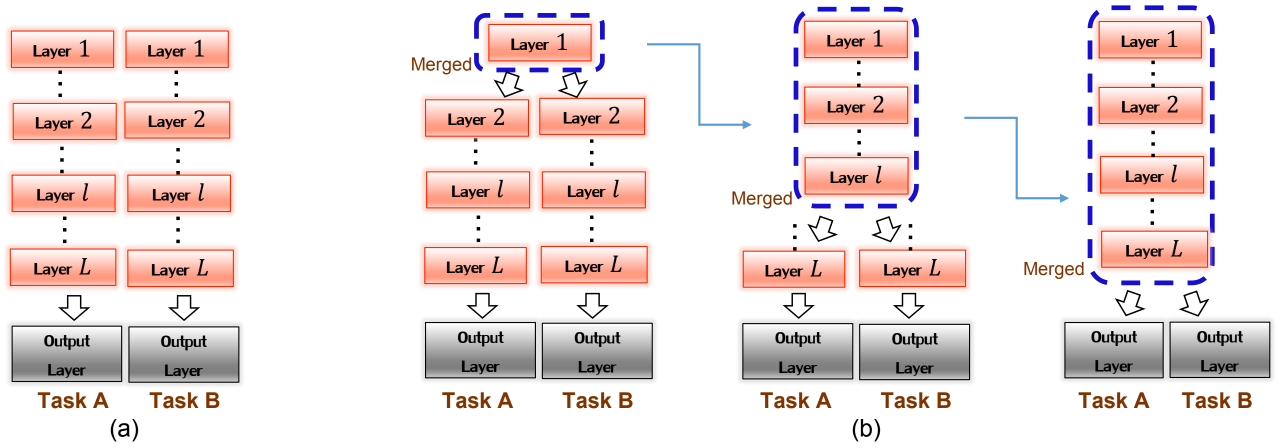

Fig. 2. The zippering process of merging two feed-forward networks. (a) Original neural networks. (b) The "Zippering" process. The merging is done sequentially from the first layer, and then iteratively fusing the later layers. At the beginning of the merging layers, the alignment of filters is done via the Hungarian algorithm with l_1 distance. The aligned filters are then initialized and calibrated with the training data to obtain the merged layer weights. Then the process is repeated for the subsequent layers until all layers are merged. A sequence of merged models is generated for selection; one of them can be chosen for the deployment according to the memory budget and accuracy requirement.

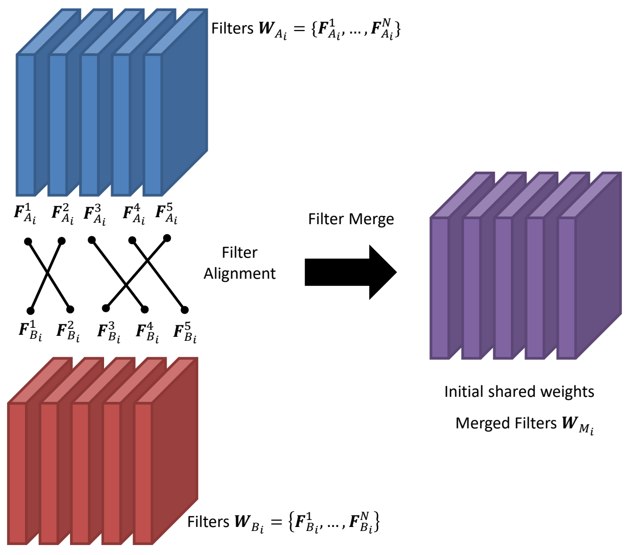
Filters $W_{A_i} = \{F^1_{A_i}, \ldots, F^N_{A_i}\}$
$F^1_{A_i}$ $F^2_{A_i}$ $F^3_{A_i}$ $F^4_{A_i}$ $F^5_{A_i}$
Filter Alignment
Filter Merge
$F^1_{B_i}$ $F^2_{B_i}$ $F^3_{B_i}$ $F^4_{B_i}$ $F^5_{B_i}$
Initial shared weights
Merged Filters W_{M_i}
Filters $W_{B_i} = \{F^1_{B_i}, \ldots, F^N_{B_i}\}$

Fig. 3. Filter alignment process. To minimize the difference of the merged model, the order of the merged filters may change in our approach. The best matching of the filters is modeled as an assignment problem; this can be done by the Hungarian algorithm using l_1 distance.

Filter $F_i \in \mathrm{R}^{N \times C \times w \times h}$ $(i = 1 \cdots N)$: N filters applied; each has C channels of spatial size $w \times h$.
Output $y \in \mathrm{R}^{N \times W \times H}$: a tensor consisting of N channels with each channel $y_i = F_i(x) \in \mathrm{R}^{W \times H}$, for $i = 1 \cdots N$.

If we reshuffle the filters of this layer as $F_{\sigma(i)}$ $(i = 1 \cdots N)$, with $\sigma(\cdot)$ a permutation on $\{1 \cdots N\}$, then the output remains the same but is permuted accordingly, $y_i = F_{\sigma(i)}(x), i = 1 \cdots N$. Full-connection layers hold a similar property. This property can be extended to the whole feed-forward network. Suppose we are given a network having L layers. If a permutation $\sigma_l(\cdot)$ is applied to the filters of the l-th layer, the output of this layer is then applied with the same permutation. Hence, the filters in the next layer (the $(l+1)$-th layer) should be pre-permuted via $\sigma_l(\cdot)$ in advance to ensure the invariance of the output. Iterating this idea yields that, if we apply the permutations $\sigma_1(\cdot)$, $\sigma_2(\cdot)$, $\cdots$, $\sigma_{L-1}(\cdot)$ to the filters of the respective layers and hope that the final output remains the same, then we have to apply actually $\sigma_1(\cdot)$ to the 2nd-layer filters, $\sigma_1 \circ \sigma_2(\cdot)$ to the 3rd layer filters, ..., and $\sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_{(L-1)}(\cdot)$ to the L-th layer filters, respectively, where $\circ$ denotes the function composition.

Leveraging this property, given two convolution layers as a pair to be fused, we propose to permute the filters so that similar filters are aligned to enforce a better initialization of model fusion. More specifically, consider two models A and B that have been well-trained in advance. We permute the filters in model A and then merge them with the filters of model B. For the l-th layer, we change the order of the filters in model A with the permutation $\sigma_l(\cdot)$, and hope that similar filters $F^A_{\sigma_l(i)}$ and F^B_i can be paired together. To achieve this, we calculate the score of each filter by summarizing the absolute values of weights. And then, we use a distance metric to measure the similarity between filters based on scores, where the l_1 distance between filters is adopted in this work. When the order of the filters changes, the error of merging may be reduced. Since this best order can be formulated as an assignment problem, the best matching of models can be found by the Hungarian algorithm [23]. The time complexity of the Hungarian algorithm is $O(N^3)$. Figure 3 shows the illustration of the alignment process. After the best order of filters is found, we adjust the channel order in the next layer accordingly as mentioned above. This will make the feed-forward neural network generate the same outputs as those that before the filter alignment.

B. Shared Weight Initialization

To effectively initialize the filter weights of fused layers, a promising way is to leverage the well-trained models given.

Consider a pair of convolution layers in model A and model B with $N \times C \times w \times h$ convolution kernels. We simply initialize the merged filter weights by averaging the original weights, which yields the *least quantization error* of the weights:

$$\mu_i = (F^A_{\hat{\sigma}(i)} + F^B_i)/2,\ i \in \{1 : C\}, \qquad (1)$$

where $\hat{\sigma}(\cdot)$ is the optimal permutation found by the Hungarian matching algorithm for this layer, and μ_i denotes the initial fusion weights, $i = 1 \cdots C$.

C. Weight Calibration

Once the fused weights are available, they are set as initial values to merge the models for multiple tasks. The entire model is fine-tuned with a portion (10% in our experiments) of randomly selected training data through end-to-end back-propagation learning. The original losses (such as cross-entropy losses) of both tasks are summed to fine-tune the fused model. Accordingly, the optimization problem is

$$\mathbf{w}^* = \min_{\mathbf{w}}[\cdot Loss_c A(\mathbf{w}) + \cdot Loss_c B(\mathbf{w}))] \qquad (2)$$

where $\mathbf{w}$ are the parameters of the convolutional filters, and $Loss_c$ are the individual losses. We refer this to as the model calibration procedure.

Shared-weight initialization and weight calibration are alternatively performed when the top l layers are fused in our zippering process, as shown in Figure 2(b). A sequence of merged models are obtained after finishing the zippering process. Users can flexibly choose one of them for the inference stage via the trade-off between resource/speed and accuracy.

IV. Experiments

We evaluate our method on three scenarios with different benchmarks. First, we use the facial images as input for three tasks: face recognition, gender identification, and expression classification. Second, we examine the effectiveness under different modalities, including face recognition (image) and speaker verification (voice). Finally, we show the performance of merging object and clothing detection models.

A. Face Verification, Gender and Expression

To verify the effectiveness of our method on merging multiple facial-informatic tasks, three CNN models respectively trained for face recognition, gender identification and expression classification are merged. The accuracy of these well-trained models are 99.4% (face recognition), 91.5% (gender identification) and 57.6% (expression classification), respectively. In this experiment, four benchmarks are used:

VGGFace2 dataset [25] contains approximately 3.3 millions facial images, including 8,631 identities spanning a wide range of ethnicities, accents, professions, and ages.

LFW dataset [26] contains more than 13,000 facial images and approximately includes 1.680 identities.

FotW dataset [27] is used for gender identification in Chalearn big challenge [27]. It contains 9,258 facial images; 6,171 facial images for training and 3,087 for validation.

AffectNet dataset [28] contains 287,401 facial images; 283,901 for training and 3,500 for validation. Its labels are seven primary expressions, Neutral, Happy, Sad, Surprise, Fear, Disgust and Anger; each has 500 images for validation.

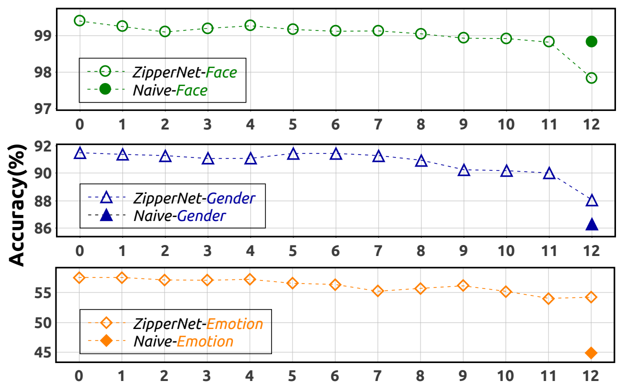

Fig. 4. Evaluation of face recognition, gender identification, and emotion classification, respectively. ZipperNet is our proposed method while the naive network is a baseline illustrated in Figure 1, where the face-recognition model is used as the pre-trained model. The backbone network are CNN-20 [24]. The values at the 0-th merged block are generated by the individual well-trained model for the three tasks.

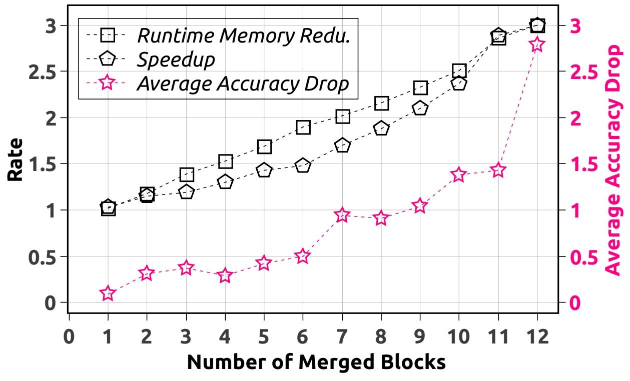

Fig. 5. The left vertical axis is the rate of speedup and run-time memory reduction, while the right vertical axis is the average accuracy drop of ZipperNet, which handling face verification, gender identification and expression classification tasks simultaneously

CNN-20 architecture is employed in this experiment as it achieves high performance (higher than 99%) on face recognition [24] and is verified by a wide variety of data.

We merge the three individual models and call the merged network *ZipperNet* that can achieve the three tasks with a single model. When applying our zippering procedure, we group the 21 layers of CNN-20 into 12 blocks (according to the shortcut locations) to simplify the merging process. The 12 blocks correspond to the 1st, 2nd-3rd, 4th, 5th-6th, 7th-8th, 9th, 10th-11th, 12th-13th, 14th-15th, 16th-17th, 18th-19th and 20th-21st layers in CNN-20. Then we fuse the blocks, from the first to the 12th, incrementally.

The results are shown in Figures 4 and 5. The well-trained model for each individual task is expressed as the 0-th merged block (i.e., the initial model before merging). In Figure 4, the performance with respect to the number of merged blocks

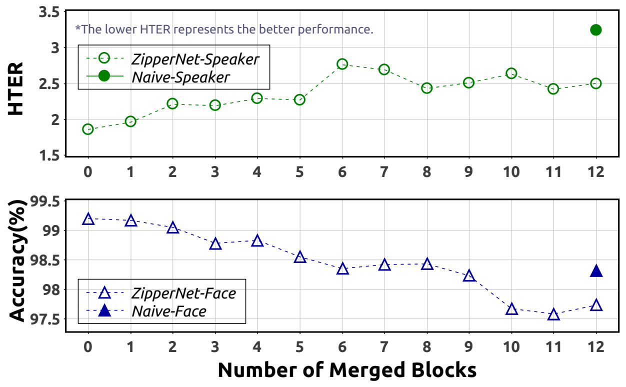

Fig. 6. Evaluation of speaker and face verification. The upper-left axis is the Half Total Error Rate (HTER) for speaker verification, while the lower-left axis is the LFW accuracy for face verification. The backbone network of ZipperNet and the naive network is CNN-20 [24].

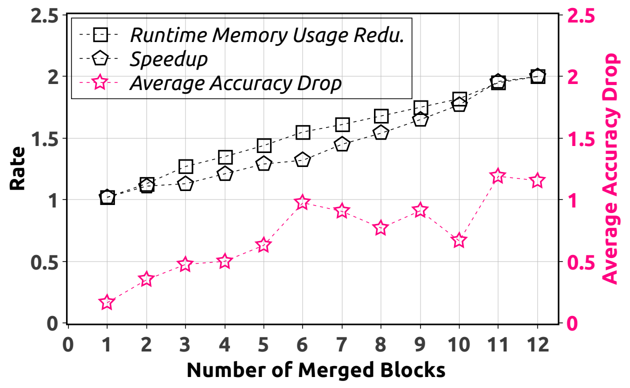

Fig. 7. The accuracy, speedup, and compression versus the number of merged blocks of the merged CNN (ZipperNet) for face and speaker recognition. The left axis is the rate of speedup and run-time memory reduction, while the right axis is the average accuracy drop of ZipperNet.

are shown via hollow circles, triangles, and diamonds for the tasks of face, gender and emotion, respectively. In Figure 5, compared to that of 0-th merged block, the average accuracy drop of the three tasks are plotted by stars, and the runtime memory-reduction and speedup rates are drawn with squares and pentagons, respectively. The performance of the naive model (obtained from the baseline method in Figure 1) are shown via solid circle, triangle, and diamond in Figure 4 for the three respective tasks.

As can be seen, when merging the models gradually, the accuracy is getting decreased slowly, but the runtime memory required is reduced and the inference speed is increased. ZipperNet thus provides a flexibility of trade-off between the model complexity and the inference speed for the model deployment. It yields a sequence of getting-condensed multitask models via removing the joint redundancy between the merged layers. In Figure 5, we observe that ZipperNet can achieve up to three times speedup and run-time memory reduction when merging all of the 12 blocks, while the average accuracy drop of tasks is less than 3%. The optimal choice for an application can be dependent to the resource budget and accuracy requirements for end platforms (such as mobile or embedding devices) to be deployed.

As no similar approach has been done on publicly available platform (we use PyTorch [29]) before, we compare our approach to the naive baseline (illustrated in Figure 1) that uses one model as the pre-trained and fine-tunes it to the three-tasks outputs. In this experiment, the face-recognition model weights serve as the pre-trained initials for the naive (i.e. baseline) approach. As shown in the figure, when merging only the blocks from 1 to 4, the accuracy is 99.27 (ZipperNet-Face), 91.09 (ZipperNet-Gender), and 57.28 (ZipperNet-Emotion), respectively, which remain very close to that of the well-trained individual models and outperforms the naive baseline model. When merging the blocks from 1 to 11, ZipperNet still performs more favorably than Naive-Face, Naive-Gender and Naive-Emotion. After merging all 12 blocks, despite the accuracy of Naive-Face is slightly better, ZipperNet outperforms Naive-Gender by 1.7% and Naive-Emotion by 9.4%, and has superior average accuracy. Generally, ZipperNet achieves more satisfiable performance and provides better flexibility for practical use. As shown in the results, the naive baseline roughly maintains the accuracy of the task from which it is pre-trained (Face Verification), but suffers from serious accuracy degradation on the others (Gender and Expression classification). The performance are thus quite imbalance. This reveals the niche of our zippering approach.

Besides, since the training starts from the shared weight initialization in the zippering process, we find that the convergence is fast due to the good initialization. Thus, to realize the zippering approach, the merged model is fine-tuned with only one epoch per one block during merging in our implementation. In this way, the total epochs required (from the individual well-trained models) depends only on the number of blocks merged. Hence, calibration training of the ZipperNet is efficient although multi-stages merging is needed.

B. *Face and Speaker Verification*

We also utilize our ZipperNet to achieve face and speaker verification tasks simultaneously. The well-trained model (CNN-20) for face verification and speaker verification achieve the accuracy of 99.2% (higher is better) and Half Total Error Rate (HTER) of 1.86% (lower is better), respectively. We follow the same evaluation metric for face recognition as before. For speaker verification, we conduct an experiment on the chosen speakers from VoxForge (//www.voxforge.org/), an open source speech corpus collected transcribed speech data recorded at 16 bit, 16kHz from volunteer speakers. Following the settings of [30], 300 speakers with at least 20 utterances are chosen and further split into three subsets: training, development, and evaluation.

For speaker verification, the HTER values obtained from ZipperNet (merging from the first to the 12th blocks) and the naive baseline are shown on top of Figure 6. For face recognition, the results are shown on bottom of the figure. As can be seen, all HTER values (lower is better) of Zipper-Speaker when merging from the first to the 12th blocks are lower than that of Naive-Speaker. Besides, the accuracy (higher is better) of Zipper-Face during merging from the first to the 8th are better than that of Naive-Face, but when merging from the 9th to 12th, Naive-Face is slightly better. We owe that the signal sources are heterogeneous (images and voice), and thus merging the intermediate CNN layers of them are more demanding. However, it still reveals that ZipperNet provides a flexibility of trade-off between the compression rate and accuracy in the zippering process.

C. Object and Clothes Detection

Finally, we apply our approach to object- and cloths-detection models (using RFBNet300 [31]) on PascalVOC 2007 and 2012 [32] and DeepFashion2 datasets [33], respectively. The accuracy of the two well-trained models are $mAP_{0.5} = 79.97\%$ and $mAP_{0.5:0.95} = 60.6\%$, respectively, which are fined-tuned from the ImageNet pre-trained weights.

Table I shows the accuracy of each task and the ratios of model size (parameter) compression, memory reduction and speedup when the layers are merged from 1 to 10 incrementally. Compared to the individual models (layer-0), we observe that the accuracy can even be better when merging lower layers, and is gradually decreased and becomes worse when merging deeper layers. We conjecture that lower-layer features of these two tasks are common and thus can be fused more easily to yield a unified feature representations; deeper features are learned with higher semantic information of the tasks, and thus will be exclusive to each other. Besides, the ratios of memory reduction and speedup on inference are increased slowly and smoothly when merging more layers.

TABLE I
THE DETECTION RESULTS OF MERGING THE FIRST LAYER TO THE 10TH LAYER INCREMENTALLY ON PASCAL VOC (2007 AND 2012) AND DEEPFASHION2 DATASETS, WHICH SHOW MAP (MEAN AVERAGE PRECISION), COMPRESSION OF THE MODEL PARAMETERS RATIO, REDUCTION OF THE RUN-TIME MEMORY USAGE RATIO, AND SPEEDUP RATIO.

# of merged layers	$mAP_{0.5}$ (VOC 2007)	$mAP_{0.5:0.95}$ (DF2)	Mem. Redu.	Speedup
0 (Individual)	79.97	60.60	1.00	1.00
1	80.53	61.10	1.18	1.09
2	80.39	61.50	1.20	1.20
3	80.50	61.50	1.23	1.26
4	80.30	61.50	1.26	1.33
5	80.55	61.30	1.27	1.37
6	80.53	62.00	1.27	1.41
7	80.12	60.00	1.29	1.45
8	79.20	60.70	1.31	1.47
9	77.61	58.50	1.34	1.50
10	76.07	57.30	1.36	1.52

V. CONCLUSION

We introduced a new approach, ZipperNet, to merge well-trained deep models. ZipperNet removes the co-redundancy among the models in a gradual manner. The merged model can reduce the inference time with little accuracy drop compare to original models for multitasks. It generatess a sequence of merged models via trade-off between speedup and accuracy-drop. ZipperNet therefore provides a flexible choice for the model deployment on resource-limited edge devices which meets memory budget and accuracy requirement. We verify our method using different network architectures on various source of inputs. Experimental results on a plenty of bench-marks demonstrate its effectiveness.

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support from the Ministry of Science and Technology, Taiwan (MOST 109-2634-F-001-009) and National Center for High-performance Computing (NCHC) of National Applied Research Laboratories (NARLabs) in Taiwan for providing computational and storage resources.

REFERENCES

[1] C.-P. Chen and C.-S. Chen, "Lighting normalization with generic intrinsic illumination subspace for face recognition," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2. IEEE, 2005, pp. 1089–1096.

[2] H.-R. Chou, J.-H. Lee, Y.-M. Chan, and C.-S. Chen, "Data-specific adaptive threshold for face recognition and authentication," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2019, pp. 153–156.

[3] W.-S. Chu, C.-R. Huang, and C.-S. Chen, "Identifying gender from unaligned facial images by set classification," in *2010 20th International Conference on Pattern Recognition*. IEEE, 2010, pp. 2636–2639.

[4] J.-H. Lee, Y.-M. Chan, T.-Y. Chen, and C.-S. Chen, "Joint estimation of age and gender from unconstrained face images using lightweight multi-task cnn for mobile applications," in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2018, pp. 162–165.

[5] K.-Y. Chang, C.-S. Chen, and Y.-P. Hung, "Intensity rank estimation of facial expressions based on a single image," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2013, pp. 3157–3162.

[6] H.-F. Yang, B.-Y. Lin, K.-Y. Chang, and C.-S. Chen, "Joint estimation of age and expression by combining scattering and convolutional networks," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 1, pp. 1–18, 2018.

[7] T. S. Wan, J.-H. Lee, Y.-M. Chan, and C.-S. Chen, "Co-compressing and unifying deep cnn models for efficient human face and speaker recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[8] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *International Conference on Machine Learning*, 2018, pp. 550–559.

[9] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *ICLR*, 2019.

[10] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *ICLR*, 2019.

[11] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *International Conference on Learning Representations*, 2016.

[12] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.

[13] 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," *arXiv preprint arXiv:1704.04861*, 2017.

[14] K. He, Z. Wang, Y. Fu, R. Feng, Y.-G. Jiang, and X. Xue, "Adaptively weighted multi-task deep network for person attribute classification," in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 1636–1644.

[15] R. Ranjan, S. Sankaranarayanan, C. D. Castillo, and R. Chellappa, "An all-in-one convolutional neural network for face analysis," in *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*. IEEE, 2017, pp. 17–24.

[16] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. 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*, 2017, pp. 5334–5343.

[17] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7482–7491.

[18] K. Thung, P.-T. Yap, and D. Shen, "Multi-stage diagnosis of alzheimer's disease with incomplete multimodal data via multi-task deep learning," *MICCAI Workshop*, 2017.

[19] Y. Ning, J. Jia, Z. Wu, R. Li, Y. An, Y. Wang, and H. M. Meng, "Multi-task deep learning for user intention understanding in speech interaction systems," in *AAAI*, 2017.

[20] C. Cadena, A. R. Dick, and I. D. Reid, "Multi-modal auto-encoders as joint estimators for robotics scene understanding," in *RSS*, 2016.

[21] Q. Liu, Y. Zhang, Z. Liu, Y. Yuan, L. C. Cheng, and R. Zimmermann, "Multi-modal multi-task learning for automatic dietary assessment," in *AAAI*, 2018.

[22] S. Chen, Q. Jin, J. Zhao, and S. Wang, "Multimodal multi-task learning for dimensional and continuous emotion recognition," in *ACM MM Workshop*, 2017.

[23] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[24] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 212–220.

[25] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE, 2018, pp. 67–74.

[26] E. Learned-Miller, G. B. Huang, A. RoyChowdhury, H. Li, and G. Hua, "Labeled faces in the wild: A survey," in *Advances in face detection and facial image analysis*. Springer, 2016.

[27] S. Escalera, M. Torres Torres, B. Martinez, X. Baró, H. Jair Escalante, I. Guyon, G. Tzimiropoulos, C. Corneou, M. Oliu, M. Ali Bagheri *et al.*, "Chalearn looking at people and faces of the world: Face analysis workshop and challenge 2016," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 1–8.

[28] A. Mollahosseini, B. Hasani, and M. H. Mahoor, "Affectnet: A database for facial expression, valence, and arousal computing in the wild," *IEEE Transactions on Affective Computing*, vol. 10, no. 1, pp. 18–31, 2017.

[29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

[30] H. Muckenhirn, M. M. Doss, and S. Marcell, "Towards directly modeling raw speech signal for speaker verification using cnns," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4884–4888.

[31] S. Liu, D. Huang *et al.*, "Receptive field block net for accurate and fast object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 385–400.

[32] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[33] Y. Ge, R. Zhang, X. Wang, X. Tang, and P. Luo, "Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5337–5345.