Journal of Robotics, Networking and Artificial Life
Vol. **8**(**1**); June (2021), *pp.* 52–55
DOI: https://doi.org/10.2991/jrnal.k.210521.012; ISSN 2405-9021; eISSN 2352-6386
https://www.atlantis-press.com/journals/jrnal

Research Article

A Sub-Model Detachable Convolutional Neural Network

Ninnart Fuengfusin*, Hakaru Tamukoh

Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology, 2-4 Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka 808-0196, Japan

ARTICLE INFO

Article History

Received 25 November 2020
Accepted 20 April 2021

Keywords

Convolutional neural networks
supervised learning
model compression

ABSTRACT

In this research, we propose a Convolutional Network with sub-Networks (CNSN), i.e., a Convolutional Neural Network (CNN) or base-model that can be divided into sub-models on demand. The CNN architecture, entails that feature map shapes are varied throughout the model, therefore, the hidden layer within CNN may not directly process an input image without modification. To address this problem, we propose a *step-down convolutional layer,* which is a convolutional layer acting as an input layer for the sub-model. This *step-down convolutional layer* reshapes and processes an input image to a preferred representation to the sub-model. To train CNSN, we treat the base-model and sub-models as distinct models. Each model is forward- and back-propagated separately. Using *multi-model loss*, i.e., a linear combination of losses from base-model and sub-models, we thus update model parameters that can be utilized in both base-model and sub-models.

1. INTRODUCTION

In recent years, Convolutional Neural Networks (CNN) have been achieved state-of-the-art performance in varied tasks including image recognition [1] and object detection [2]. However, when comparing CNN with a Multilayer Perceptron (MLP) having the same number of weight parameters, CNN utilizes significantly more Multiply-Accumulate Operations (MAC) than MLP. This may lead to a bottleneck when deploying a CNN to a mobile device with low computational capacity. Addressing this problem, several research directions emerge from a structure pruning [3], network slimming [4], and early-exit network [5]. Structure pruning focuses to on reducing the number of filters in each layer of CNN by pruning filters with small magnitude. With a lower number of filters, this reduces the run time during inference in both the central processing unit and the field-programmable gate array. Network sliming focuses on creating a neural network model that can be detached into smaller models through inference. Therefore, the user can select trade-offs between latency and performance by selecting an inference with either a larger model or smaller model (detached model). In contrast, the early-exit network allows fast inference via predictions from the early layer of the neural network to its exit if the classifier on the early layer is sufficiently confident. Otherwise, feature maps from the early layer are passed and processed further in later layers, thereby producing more confident predictions.

In this research, we focus on network sliming. In our previous works, we purposed the Network with sub-Networks (NSN) [6,7] which introduced layer-wise detachment ability to MLP. Detaching weight layers on demand allows NSN to operate with lower MAC, therefore; it can decrease overall latency with a tradeoff in performance. However, it is not possible to directly apply the methods of NSN to CNN because of the following compatibility issues as follow. First, NSN requires a targeted neural network model to contain a same shape of a feature map throughout the model. This creates a major constraint to the CNN model; for example, the CNN cannot contain spatial reduction layers (pooling layers or convolutional layers without padding), without which, the CNN may operate with higher MAC than the layer detaching can reduce. The second issue is that NSN requires parallel training of $N + 1$ models (both base-model and sub-models), where N is the number of hidden layers. This indicates that NSN methods are not feasible with recent CNN models that contain more than 100 hidden layers. The third issue is that NSN required manual control of gradients across base-model and sub-models, causing high complexity during programing.

In this research, we propose Convolutional Network with sub-Networks (CNSN) to address the three aforementioned issues identified during the previous work. CNSN allows the use of different shapes of the feature map by attaching a *step-down convolutional layer* as the input layer of sub-models. This layer preprocesses and reshapes input images to a preferred representation to the sub-model. To allow CNSN to operate more depth CNN models, CNSN reduces a number of sub-models by allowing only few selected sub-models to be detached. Instead, of manually controlling the gradients, we introduce a *multi-model loss*, i.e., the combination of losses from a base-model and sub-models. This loss allows the base model to be detached into sub-models during inference and reduces its overall complexity in programing for the user making comparisons with NSN.

*Corresponding author. Email: *fuengfusin.ninnart553@mail.kyutech.jp*

To detach several convolutional layers on demand, CNSN promises a greater reduction in overall MAC than demonstrated in our previous work, which can detach only fully connected layers. CNSN also provides a select-ability between the amount of MAC and test accuracy to the user. By deploying CNSN to mobile devices that are diverse in specifications, the user can freely select a base model or sub-models that suits the user's mobile device and preferences. Our main contributions in this research are as follows:

- We propose CNSN, a CNN with the ability to detach a base-model into sub-models during inference.
- We introduce *multi-model loss*, a linear combination of losses from the base-model and sub-models.
- We introduce a *step-down convolutional layer* that operates as an input layer to sub-models.

2. CONVOLUTIONAL NETWORK WITH SUB-NETWORKS

Convolutional network with sub-networks consists of a base-model and several sub-models. The sub-model is a subset of the base-model with an exception, the *step-down convolutional layer.* Therefore, all parameters from sub-models the exception of *step-down convolutional layer,* are identical to those of the base-model in both learnable parameters and structure. In terms of the memory usage of CNSN, we only required the storage of the base-model and several *step-down convolutional layers.*

An overview of CNSN is illustrated in Figure 1 which shows a base-model represented at the top row and two sub-models are in the two bottom rows. Each base-model and sub-model directly receives an input image and independently produces predictions. With fewer number of parameters in the model, the sub-model contains less capacity compared with the base-model. However, it promises to perform inference faster than the base-model.

With both base-model and sub-models, the CNSN consists of the three components, making CNN detached to become sub-models. These three components are as follows: the *step-down convolutional layer*, *sub-models*, and *multi-losses*. We describe each component in the following sections.

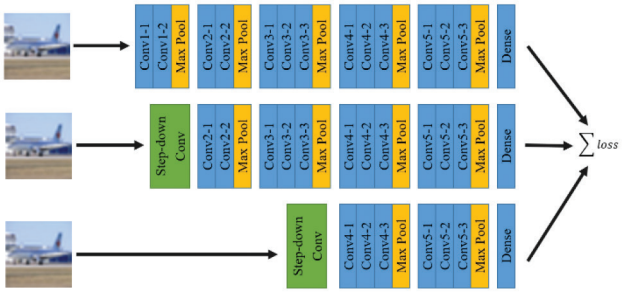

Figure 1 | Overview of CNSN. The top row represents a base model of CNSN whereas the two bottom rows represent sub-models. Each model can receive an image and produce a prediction. During training, losses from all models are accumulated as a *multi-model loss.*

2.1. Step-down Convolutional Layer

The *step-down convolutional layer* is a general convolutional layer designed to solve the spatial difference between an input and a hidden layer in the CNN. Solving this problem allows the CNSN to reuse a hidden convolutional layer of the base-model as a layer of the sub-model. However, there are some concerns, which are outlined as follows.

The first problem is the number of input filters of the hidden layer may not be the same as the number of channel of the input image. Consequently, the hidden layer cannot directly process the input image. To address this problem, we can insert a *step-down convolutional layer* with the same number of input filters to the number of channel of the input image. Another issue is that the expected width and height of the input feature is difference between the base-model to the sub-models. By adjusting the number of strides and size of the kernel of the *step-down convolutional layer*, we can adjust the shape of output feature maps as we desired and the processed feature maps can be further processed in the sub-model.

2.2. Sub-models

Instead of allowing every layer of the neural network to be similarly detachable as the NSN, CNSN only allows a few sub-models to be detached. In each training batch, both NSN and CNSN requires base-model and sub-models to be propagated in the same iteration. This causes a problem in that parameters from base- and sub-models may not be fitted in a single graphics processing unit if the base-model is large and contains many sub-models. Therefore, to train the CNSN for a large CNN, we reduced the detachability of the model to only a few sub-models.

2.3. Multi-model Loss

To allow both the base-model and sub-models to operate as an individual model, we purpose a method termed *multi-model loss,* which is designed to balance between the loss from the base-model and sub-models. By assuming N sub-models in the base-model, *multi-model loss* or l_{all} is formulated as illustrated in Equation (1), where l_N is the loss from N numbered sub-models, and l_{base} is the loss from the base-model. To utilize this loss, all models must be forward propagated in the same batch training first, to find all l_N and l_{base}. Subsequently, we can utilize l_N and l_{base} to solve for l_{all} as follows.

$$l_{\text{all}} = \sum_{1}^{N} l_N + l_{\text{base}} \tag{1}$$

3. EXPERIMENTAL RESULTS AND DISCUSSION

We implemented our CNSN using a VGG-16 [8]-like model as the base-model. Because VGG-16 is designed for the ImageNet Large Scale Visual Recognition Challenge [9] 2014, some layers in VGG-16 are required to be modified to operate with CIFAR10 [10]. We removed first two fully-connected layers and modified

Table 1 | Results of CNSN models compared with a base-line model on the CIFAR10 dataset

	Test accuracy	MAC (G)	Weight decay
sub-model0	0.7738	**0.124**	6×10^{-4}
sub-model1	0.9115	0.275	
base-model	0.9267	0.314	
baseline model	**0.9316**	0.314	6×10^{-3}

the last fully-connected layer to 512 input and 10 output neurons. To stabilize the training process of CNSN, we also attached Batch Normalization (BN) [11] layers after each weight layer, except the last fully-connected layer. We utilized rectified linear units as an activation function except for the last fully-connected layer, which was instead assigned the log-softmax.

The overall setting of the reported experiment is illustrated in Figure 1. We assigned two sub-models within the base-model. Including neither an activation function, BN, nor a pooling layer, the first sub-model or *sub-model0* is a base-model following the removal of the first two convolutional layers. The second sub-model, *sub-model1,* is the base-model after removing the first seven convolutional layers. Each sub-model is attached to a *step-down convolutional layer* which acts as the input layer of a sub-model. The *step-down convolutional layer* for *sub-model0* is the convolutional layer with stride two and kernel size two. For *sub-model1*, the *step-down convolutional layer* is the convolutional layer with stride eight and kernel size eight. We defined the *baseline model* as a VGG-16-like model that was trained without modification.

We conducted a benchmark using CIFAR10 dataset. CIFAR10 consists of 10 different classes and 60,000 images. Each image is an RGB image with a size of 32×32, and each class consists of 5000 training and 1000 test images, we performed data augmentation by padding 4 pixels into the training image and randomly cropping back to the original size. The training images were further augmented by horizontal flipping and normalizing with a channel-wise mean and standard deviation of the CIFAR10 dataset. In this experiment, we trained all models using a stochastic gradient descent with a momentum of 0.9. We set an initial learning rate as 10^{-2} and it is step-downed to one-tenth after training for 50, 100, and 150 epochs. We warmed up the learning rate for one epoch and trained our models for 300 epochs using a training batch size of 32. We reported the best test accuracy identified during this training.

Experimental results are shown as Table 1, where we compared the CNSN base model with the *baseline model* having the same setting as CNSN except for weight decay. We found that the optimized weight decays differ between the *baseline model* and CNSN. Therefore, to ensure fair comparisons, we applied different weight decays to each model. Our *base-model* achieved a loss in test accuracy 0.0049 in exchange for an ability to detach into sub-models. The *sub-models1* was able to reduce more than half of MAC compared with the *baseline model*; however, this detachability was associated with a significant drop in terms of test accuracy relative to the *base-model*.

4. CONCLUSION

We propose CNSN, a CNN that can be detached into smaller CNNs or sub-models on demand. To enable CNN detachability, we propose the *multi-model loss* and *step-down convolutional layer. Multi-model loss* is used to introduce a model-wise detachability to the CNSN model. Although the *step-down convolutional layer* acts as a preprocessing or input layer to the sub-model, our sub-model can deliver a performance comparable with that of regular trained models, while sub-models significantly reduce the amount of MAC; however, this may be associated with a tradeoff in test accuracy.

5. FUTURE WORKS

The main issue facing CNSN is that the performance of sub-models is decreased relative to that of the base-model. In future works, we plan to improve this problem by utilizing CNSN using knowledge distillation [12]. Generally, knowledge distillation requires a large teacher and small student model to distill knowledge from the teacher to the student model. In this case, our CNSN contains both models as base-model (teacher model) and sub-models (student models). However, we cannot directly utilize conventional knowledge distillation without modification. This requires one to many relations between a single teacher (base-model) and multi-student models (sub-models), which raises a research question regarding how to effectively distill knowledge from the teacher model to student models while each teacher and student model shares parameters. Furthermore, sub-models contain both strong and weak classifiers. Distillation between them is also possible. Currently, our CNSN can detach only in the depth or layer-wise directions. We are interested to expand its detachability to CSNS in the filter-wise direction. Our main goal is to discover detachable sub-models with an arbitrary number of filters and layers inside a base model. Each sub-model has objective functions that enable it to maximize performance and minimize the number of MACs.

CONFLICTS OF INTEREST

The authors declare they have no conflicts of interest.

ACKNOWLEDGMENTS

This research was supported by JSPS KAKENHI Grant Numbers 17K20010.

REFERENCES

[1] M. Tan, Q.V. Le, Efficientnet: rethinking model scaling for convolutional neural networks, arXiv preprint arXiv:1905.11946, 2019.
[2] M. Tan, R. Pang, Q.V. Le, EfficientDet: scalable and efficient object detection, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Seattle, WA, USA, 2020, pp. 10778–10787.
[3] L. Hao, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, Pruning filters for efficient ConvNets, arXiv preprint arXiv:1608.08710, 2016.
[4] Y. Jiahui, L. Yang, N. Xu, J. Yang, T. Huang, Slimmable neural networks, arXiv preprint arXiv:1812.08928, 2018.
[5] S. Teerapittayanon, B. McDanel, H.T. Kung, BranchyNet: fast inference via early exiting from deep neural networks, 2016 23rd

International Conference on Pattern Recognition (ICPR), IEEE, Cancun, Mexico, 2016, pp. 2464–2469.
[6] N. Fuengfusin, H. Tamukoh, Network with sub-networks, Aritf. Life Robot. 25 (2020), 191–194.
[7] N. Fuengfusin, H. Tamukoh, Network with sub-networks: layer-wise detachable neural network, J. Robot. Netw. Artif. Life 7 (2021), 240–244.
[8] K. Simonyan, A. Zisserman. Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2014.
[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, et al., ImageNet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (2015), 211–252.
[10] A. Krizhevsky, V. Nair, G. Hinton, The cifar-10 dataset [online], 2014. Available from: https://www.cs.toronto.edu/~kriz/cifar.html 55.
[11] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167, 2015.
[12] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv preprint arXiv:1503.02531, 2015.

AUTHORS INTRODUCTION

Mr. Ninnart Fuengfusin

He received his B.Eng. degree from King Mongkut's University of Technology Thonburi, Thailand in 2016. He received his M.Eng. degree from Kyushu Institute of Technology, Japan in 2018. He is currently a doctoral candidate at the Kyushu Institute of Technology, Japan. His research interests include deep learning, efficient neural network design and digital hardware design.

Assoc. Prof. Hakaru Tamukoh

He received the B.Eng. degree from Miyazaki University, Japan, in 2001. He received the M.Eng. and the PhD degree from Kyushu Institute of Technology, Japan, in 2003 and 2006, respectively. He was a postdoctoral research fellow of 21st century center of excellent program at Kyushu Institute of Technology, from 2006 to 2007. He was an Assistant Professor of Tokyo University of Agriculture and Technology, from 2007 to 2013. He is currently an Associate Professor in the Graduate School of Life Science and System Engineering, Kyushu Institute of Technology, Japan. His research interest includes hardware/software complex system, digital hardware design, neural networks, soft-computing and home service robots. He is a member of IEICE, SOFT, JNNS, IEEE, JSAI, and RSJ.