Doubly Nested Network for Resource-Efficient Inference

Anonymous review

Abstract

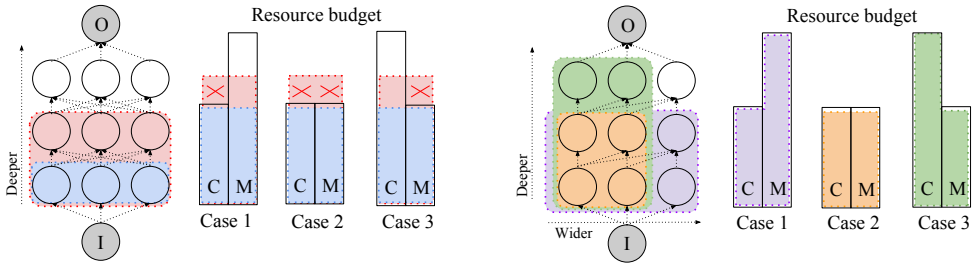
We propose a new anytime neural network which allows partial evaluation by sub-networks with different widths as well as depths. Compared to conventional anytime networks only with the depth controllability, the increased architectural diversity leads to higher resource utilization and consequent performance improvement under various and dynamic resource budgets. We highlight architectural features to make our scheme feasible as well as efficient, and show its effectiveness in image classification tasks.

1 Introduction

When we deploy deep neural network models on resource-constrained mobile devices or autonomous vehicles with a strict real-time latency requirement, it is essential to develop a model which makes the best of available resources. Although many network compaction techniques including distillation, pruning and quantization have been proposed [8, 13, 5, 4, 10, 7, 14], this goal is still challenging because (1) the resource availabilities are continuously changing over time while these resources are being shared with other program instances [2], and (2) multiple resources with different characteristics (e.g. computational capacity, memory usage) should be considered together.

Anytime machine learning algorithms have addressed the first issue, how to get the optimal performance under dynamic resource budgets, by allowing us to activate only a part of the model with graceful output quality degradation [15, 3]. Most anytime algorithms based on deep neural networks appear in the form of early termination of forward processing according to the current resource budget or the difficulty of the given task [9, 1, 11]. In other words, the conventional anytime networks are trained to embed many potential sub-networks with different effective depths so that the best one can be chosen according to the current budget.

In this work, we propose a new type of the anytime neural network, *doubly nested network*, to solve the other issue, more efficient utilization of multiple heterogeneous resources. The proposed network can be sliced along the width as well as the depth to generate more diverse sub-networks than the conventional anytime networks allowing the sub-network extraction only along the depth-wise direction. As depicted in Figure 1, the increased degree of freedom enables us to get higher resource utilization in the devices constrained by dynamically changing resource budget with multiple criteria.

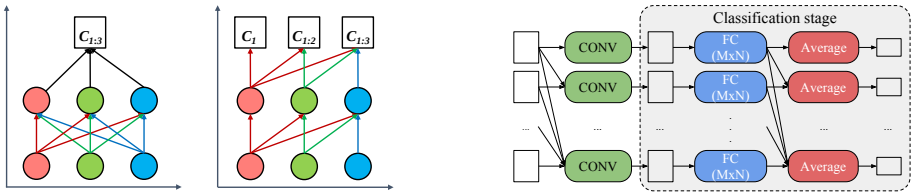


(a) Depth-controllable anytime network (b) Doubly nested network

Figure 1: Comparison of anytime networks in resource utilization (C: Computation, M: Memory)

Preprint. Work in progress.

2 Doubly Nested Network



(a) (left) Conventional convolution (right) Causal convolution.

(b) Output generation stage sharing the FC layers between sub-networks at the same depth

Figure 2: Architectural features of the doubly nested network

Causal convolution It is straightforward to form a sub-network along the depth by appending a separate output generation stage to the final convolution layer of the sub-network. Since one specific layer's output does not depend on the following (upper) layers' outputs, the jointly trained sub-network does not suffer from the performance degradation even after the extraction. However, this approach does not work along the width because of the interdependence between two nodes at different horizontal locations (e.g. different channels). To address this issue, we propose a channel-causal convolution where i-th channel group in one layer is calculated only with activation values from the channel groups from the first to i-th channel group in the previous layer as shown in the right of Figure 2a. The circle indicates the feature map while the square indicates the classifier. Color refers each channel. Our network based on the causal convolution allows us to extract the sub-network easily along any directions by making both horizontal and vertical data flow unidirectionally.

Output generation stage sharing fully-connected layers Rather than having a separate fully-connected (FC) layer for one sub-network to generate the final output (e.g. a predicted class given image input), our network is designed to have the FC layers each of which takes only a part of activations from the preceding convolution layers and produce the final output for one sub-network by averaging multiple FC layers' outputs as depicted in Figure 2b. Sharing the FC layers between the sub-networks at the same depth helps us to have similar computational and memory costs of the FC layers in the depth-controllable anytime network [9] even with much more possible output locations.

Joint loss function We can obtain a loss function for each sub-network:

$$\mathcal{L}_c^l = -\frac{1}{N}\sum_{i=1}^{N}[y_i \, log(\hat{y}_{i,c}^l)], \forall (l,c) \in \{1,2,...,L\} \times \{1,2,...,C\}, \tag{1}$$

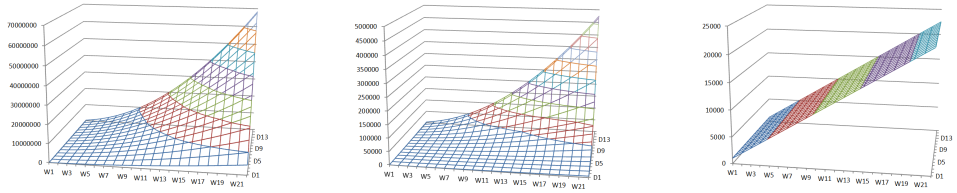
where L and C refer to the number of possible vertical and horizontal partitions and N is the number of classes. y_i is a target label of class i and $\hat{y}_{i,c}^l$ is a softmax output of $Z_{n,c}^l = \sum_{k=1}^{c} W_{n,k}^l \cdot f_k^l$. Finally, we can obtain a joint loss function by aggregating all loss functions coming from all possible partial models as:

$$\mathcal{L} = \frac{\sum_{l,c} \lambda(l,c)\mathcal{L}_c^l}{\sum_{l,c} \lambda(l,c)} \tag{2}$$

3 Experiments

Experimental setup We evaluated the proposed method on the CIFAR-10 and the SVHN datasets. Similarly to the ResNet-32 model [6], our full network architecture consists of one convolution layer fed by external input, the following 15 residual blocks and fully-connected layers for the final output generation. The network has 16 possible output locations along the depth from the first convolution layer and all residual blocks, and 22 locations along the width. Thus, we can extract 16×22 sub-networks with different widths and depths from the base network.

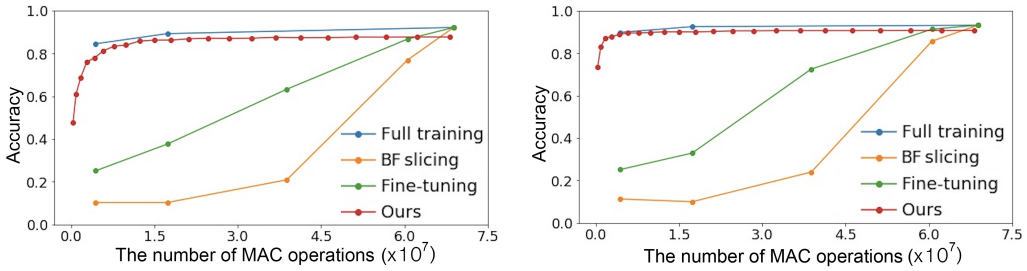
Resource usages of the sub-networks As the selected sub-network gets deeper or wider, all computational and memory requirements such as the number of MAC (multiply-accumulate) operations, the number of parameters and the size of the largest feature map increase. However, their rates of the increase are different from each other as shown in Figure 3. This means that our scheme can benefit from the larger diversity of resource usage compared to the conventional anytime methods.



(a) Number of MAC operations (b) Number of parameters (c) Size of the largest feature map

Figure 3: Resource usages of the 16×22 sub-networks (horizontal axis: width, vertical axis: height)

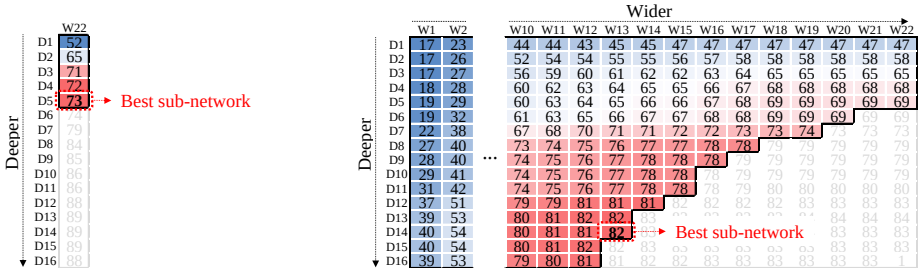
Comparison with other methods One of the key advantages of the proposed architecture is non-trivial nesting of sub-networks along the width direction. Figure 4 shows that our scheme outperforms two straightforward vertical slicing schemes (*Brute-force slicing*, *Fine-tuning*) that can generate sub-networks with different widths to a large extent without significant performance degradation compared to the upper bound (*Full training*).



(a) CIFAR-10 (b) SVHN

Figure 4: Comparison with other schemes to generate sub-networks with different widths. *Full training*: Training separate ResNet-32 models with various numbers of channels from scratch, *Brute-Force (BF) slicing*: Training a full-sized ResNet-32, then, extracting sub-networks by slicing vertically, *Fine-tuning*: BF slicing, then, only the classifier is re-trained while the network parameters are fixed.

Figure 5 shows a comparison between (a) a conventional anytime network trained to support only the slicing along the depth and (b) ours. If we are allowed to spend 24 million or less MAC operations with sufficient memory space, our network can get 9% higher accuracy by choosing the narrower but deeper sub-network (D14 & W12) than the best one (D5) in the conventional scheme.



(a) Depth-controllable anytime network (b) Doubly nested network

Figure 5: CIFAR-10 accuracies (%) of all feasible sub-networks under the budget of 24 million MACs

4 Discussion

We revealed that resource-constrained devices could benefit from the architectural diversity enriched by our anytime prediction scheme. Our future works include adding adaptive conditioning [12] which modulates intermediate activations or weight parameters depending on the current sub-network configuration to improve the performance only with a small increase of conditioning parameters.

References

[1] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama. Adaptive neural networks for efficient inference. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 527–536, 2017.

[2] Computer Vision Machine Learning Team. An on-device deep neural network for face detection. *Apple Machine Learning Journal*, Vol. 1, Issue 7, 2017 (Retrieved from https://machinelearning.apple.com/2017/11/16/face-detection.html).

[3] A. Grubb and D. Bagnell. Speedboost: Anytime prediction with uniform near-optimality. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, Spain, April 21-23, 2012*, pages 458–466, 2012.

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

[5] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016.

[7] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1398–1406, 2017.

[8] G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[9] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger. Multi-scale dense networks for resource efficient image classification. *CoRR*, abs/1703.09844, 2017.

[10] J. Lin, Y. Rao, J. Lu, and J. Zhou. Runtime neural pruning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2178–2188, 2017.

[11] M. McGill and P. Perona. Deciding how to decide: Dynamic routing in artificial neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2363–2372, 2017.

[12] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.

[13] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2014.

[14] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2074–2082, 2016.

[15] S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.