Compositional Sparse Network for Semantic Segmentation of Salt Marsh Images

J. Parashar[1] S. M. Bhandarkar[1,2] B. M. Hopkinson[3]
[1]Institute for AI [2]Dept. of Computer Science [3]Dept. of Marine Sciences
University of Georgia
Athens, Georgia 30602, USA
{jayant.parashar, suchi,bmhopkin}@uga.edu

***Abstract*—We seek inspiration from the fields of neuroscience and continual learning to create networks that can compose meaningful features. We identify three critical principles lacking in modern neural networks that should be part of general-purpose networks: (a) interconnection richness, (b) sparsity, and (c) dynamic expansion. These principles form the basis of a general-purpose network architecture which we term as a *Compositional Sparse Network* (CSN). Since we dynamically expand the network and use learned weights from lower levels to train higher levels in the CSN, the CSN design process can be viewed as a pseudo Neural Architecture Search (NAS) procedure. The CSN is first tested on the CIFAR-10 image data set, and subsequently on the salt marsh image data set where the CSN is used as a backbone for the DeepLab-V3 architecture. We compare the performance of the CSN with a NAS-based approach called Auto-DeepLab which serves as a backbone for the DeepLab-V3 architecture. The proposed CSN approach is observed to perform worse than the NAS-based approach because the higher-level CSNs are seen to not fit the data distribution.**

***Index Terms*—convolutional neural networks, network topology, deep learning, semantic segmentation.**

I. Introduction

In deep learning, we observe that the search strategies for the optimal network architecture are typically differentiated based on the underlying problem formulation. We believe that such differentiation helps us create general-purpose models only if we can find a way to segregate features based on their underlying neuronal structures. Since neural networks are a black box, the problem-specific architecture search process is merely tantamount to solving a set of specific problems. A manual search of neural network architectures for different problems does not contribute to efficient out-of-distribution (O.O.D) generalization and general intelligence. We should aim to find deep learning architectures that, except for a few input and output layers, share the same structure. Neural architecture search (NAS) [10] is an efficient architecture search procedure used to determine an optimal neural architecture. Although NAS-based approaches are efficient in terms of the search procedure, they are usually limited to determining which stacking order of convolution filters is optimal. NAS-based approaches usually focus their search on the discovery of minor CNN enhancements that would potentially lead to an optimal architecture.

The architecture of a typical CNN follows a rigid three-level hierarchy comprising of: (a) CNN channels, (b) blocks, and (c) network architecture. Acting within the bounds of this hierarchy, the NAS procedure typically results in only incremental improvement in classification accuracy. We propose an alternative approach that tackles the above limitations. First, we recommend that deep learning research focus on the inter-connectivity of individual neurons, and how features are learned and composed [3]. Only based on this knowledge, can we then confidently construct general-purpose architectures that are capable of generalization on a wide array of problems. Based on a hypothesis of how neural networks generalize using interconnection richness, we propose a new paradigm for training neural networks. This paradigm is one of pseudo neural architecture search (NAS) that essentially constructs more complex neural architectures from simpler ones. The guiding vision behind this paradigm is to leverage rich interconnections within shallow sub-networks. Many intuitive ideas underlying CNN architecture design have followed this simple paradigm of feature composition such as gradual down-sampling of the image, dilated convolution [4], and residual network design [7]. Recent work in demystifying the reasons for the success of residual network has pointed towards their capacity to behave like ensembles of relatively shallow networks [12].

We believe that the proposed pseudo NAS process, termed as *Compositional Sparse Network* (CSN) design should be performed on carefully chosen *base networks* such as *graph neural networks* (GNNs) [14] or *fully connected dense neural networks* (FCNNs). However, in our work, we choose a 3×3 kernel, the most basic CNN unit, as our base network. The primary motivation behind choosing such a minimalist base networks is to construct a modular architecture capable of composing features on its own, without the external help of image down-sampling or dilated convolution. Moreover, the current trend towards general-purpose network design is not motivated by the need to tackle real-world data sets, but by novel problem formulations such as meta-learning, few-shot learning, and continual learning[5]. These novel problem formulations usually exploit toy data sets and/or toy networks of small size. They have not been tested on larger real-world data sets and have been seen to not generalize well in many cases. This begs the question; why do we not implement our theories of general-purpose architecture design on common real-world problems such as semantic image segmentation? Since intuitively, we can expect a general-purpose network to significantly outperform a CNN that is custom designed for a specific problem, why do we not combine our hypotheses for out-of-distribution (O.O.D) generalization [8] arising from problem solving strategies such as meta-learning, few-shot learning, and continual learning and apply them to specific real-world problems such as semantic image segmentation to test their efficacy [2], [9]? With this in mind, we have formulated the concept of the CSN.

Deriving inspiration from the fields of Neuroscience [11] and Continual Learning [9], a CSN-based approach performs a pseudo NAS procedure that integrates our understanding of three basic principles required for incorporating general intelligence in neural networks: (a) interconnection richness hypothesis, (b) dynamic expansion and (c) sparsity. The interconnection richness hypothesis is based on the idea that the interconnections between local sub-networks and the corresponding interconnections between neurons in local sub-networks are critical to the performance of a neural architecture. The implementation of interconnection richness within a CSN framework is inspired by the structure of neocortex in the human brain [11]. In the CSN framework, to create a hierarchy of sub-networks, we facilitate dynamic expansion of repeating units. This idea is based on the intuition that a network must be able to increase its size based on the complexity of the problem. Traditional optimization-based dynamic expansion is short-sighted and leads to a cutoff depth that is much lower than the optimal depth required for a specific problem. Therefore, we propose and formulate a manual external dynamic expansion procedure coupled with network pruning to find general solutions [1]. While expanding a network dynamically, we should also be able to decrease its complexity to determine the most essential features to be composed, which is achieved via network pruning. Network pruning has been found to enhance the performance of a network despite deleting 60%-90% of its constituent neurons [6].

II. CSN Design Methodology

The design of a Compositional Sparse Network (CSN) is based on the unification of three principles: *interconnection richness, dynamic expansion* and *sparsity*. The CSN design starts with a base network, copies of which are combined using a cerebral cortex-like design in a *cerebral module* and inter-connected using an *interconnection module*. The cerebral module is a generic design that can work on a base network and itself. It represents a recursive structure which interconnects predefined modules with a predefined interconnection class. It is inspired by the structure of the neocortex in the human brain [11].

A. Base Network for CSN Design

For the base network in the CSN, we use three convolution layers of size 3×3 with 64 channels. We have one residual connection in the base network and one residual connection in each cerebral module. We experimented with multiple designs inspired by the early-age CNN architectures.

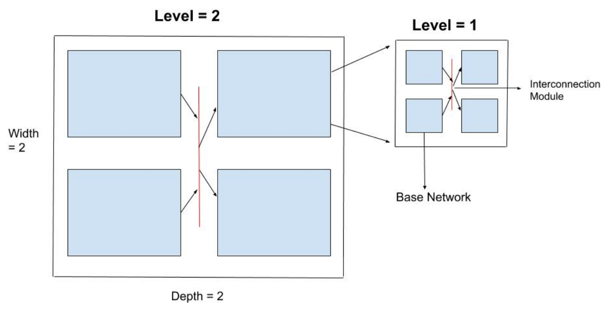

Fig. 1. An example of exponential expansion is shown. Weights from lower levels are used at higher levels.

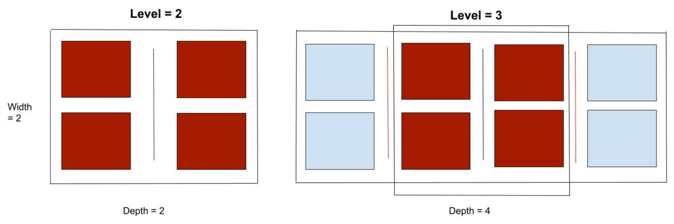

Fig. 2. An example of linear expansion is shown. Weights from lower levels are reused at higher levels as indicated by blocks and lines of the same color.

B. Cerebral Module and Interconnection Module

The cerebral module stacks multiple children networks, for example the base network, across the given width and depth values of the neural architecture. The children networks are interconnected using an interconnection module that also has a depth value of 2 or 3. The output of each cerebral module also contains residuals from its input causing every module to output a residual value from its input to its output. The channel size of the interconnection module is based on number of modules it is connecting. The goal of the interconnection module is to sew together multiple sub-networks and ensure interconnection richness.

C. Network Expansion

The cerebral module recomposes itself when expanded. There are two ways in which the cerebral module can expand; the first is exponential expansion where both the network depth and width increase exponentially which increases the number of levels of hierarchy in the network (Fig. 1). The second is linear expansion, which adds multiple low-level CSNs across the network depth or network width or both (Fig. 2). Linear expansion does not increase the size of network exponentially but enables a slower linear network growth. Most of our experiments are done using linear expansion as exponential expansion turned out to be difficult to implement on a GPU. The manner in which the weights from lower-level CSNs are used for subsequent CSNs is illustrated in Fig. 1 and Fig. 2.

In the case of exponential expansion, all the CSN weights are reused multiple times whereas in the case of linear expansion, the weights from the lower-level CSNs are inserted in the middle of the network.

D. Network Pruning

We prune our Cerebral Modules using either structured or global pruning. Structured pruning is when percentage of pruning is evenly distributed on each layer. Whereas global pruning prunes a network by comparing weights to all other weights in the network. Structural integrity becomes a problem in global pruning, but not in structured pruning. We also randomize the pruned weights by taking a weighted mean of the pruned weights with random initialization of the same. The motivation behind randomizing weights is to induce instability that leads to better generalization[1].

E. CSN Algorithm

The CSN algorithm expands the base network using either exponential expansion or linear expansion as illustrated in Fig. 1 and Fig. 2 respectively. The base network model is the first trained model and is chosen before training the entire network. The base model is used to construct the CSN model at level i denoted by CSN_i by using CSN training algorithm with the expansion procedure but without the training and pruning procedures. The CSN algorithm is illustrated in Algorithm 1.

Algorithm 1: CSN Training Algorithm

```
$CSN_0 = BaseNetwork$
$BaseModel\_Level = i$
$Exponential\_Expansion\_Level = j$
$Linear\_Expansion\_Level = k$
$CSN_i = Expand(CSN_0, i)$
for num in range$(0, j)$ do
    $CSN_{i+num} =$
    $Expand(CSN_0, i + num, "Exponential")$
    Load Weights of $CSN_{i+num-1}$ into $CSN_{i+num}$
    Train($CSN_{i+num}$)
    Prune($CSN_{i+num}$)
    Save Weights($CSN_{i+num}$)
end for
for num in range$(0, k)$ do
    $CSN_{i+j+num} = Expand(CSN_0, i + num, "Linear")$
    Load Weights of $CSN_{i+j+num-1}$ into $CSN_{i+j+num}$
    Train($CSN_{i+j+num}$)
    Prune($CSN_{i+j+num}$)
    Save Weights($CSN_{i+j+num}$)
end for
```

III. Experimental Results

A. Experimental Setup

We experimented with base networks with a varying number of CNN filters. The first level of the proposed CSN architecture, termed as CSN_1. We settled on three $3 \times 3 \times 64$ convolution filters as the simplest unit and used a Tesla P100 GPU. For exponential expansions, we considered network width and depth values of 2 since higher depth and width values resulted in fewer expansions due to limited computational resources. In the case of linear expansion, we expanded the network depth by 4 units at each step whereas the network width was kept constant because of limited computational resources. The first trained model whose weights are used for subsequent CSNs is termed the base model. We typically start with a 1-2 level exponential expansion followed by 2-4 level linear expansion.

B. Proof of Concept Implementation on the CIFAR-10 Data Set

The CSN training algorithm pipeline was fully implemented on the CIFAR-10 data set as a proof of concept. We used the Adam optimizer and found a learning rate of 5e-4 to perform the best. In case of exponential expansion, the width and depth values are kept perfectly symmetrical. We ran exponential expansion on CSN_1 and CSN_2 and, as a result, CSN_3 was our largest exponentially expanded model with 12 million parameters. We found that most of our models converged to around 70%-80% test accuracy with global and structured pruning with structured pruning performing slightly better than global pruning. As expected, higher-level CSNs took longer to train and were found to perform slightly better than lower-level CSNs.

We conducted experiments without pruning to see the effect of dynamic expansion. Without pruning, the larger-size networks were observed to always overfit. The generalization error increased rapidly as we increased the number of levels from 1 to 4 as shown in Table I. The results in Table I suggest that the higher-level CSNs without pruning are highly susceptible to fitting to the noise. It is possible that the re-using of weights led to convergence to the same local minima. When we pruned the weights and randomly varied them, we observed lower generalization error in the case of higher-level CSNs as shown in Table II. However, as seen in Table II, CSN_3 and CSN_4 are not able to completely learn the features produced by CSN_1. We surmise that this is because higher-level CSNs are merely trying to relearn what CSN_1 has already learned. The entire training process just reversing the instability we induced [1]. We had expected that, like ensembles, the different modules in the CSN would start to specialize in the formulation of distinct features. However, the experimental results ran quite contrary to that expectation. Also, despite the presence of residuals at multiple levels of the CSN hierarchy, we witnessed vanishing gradients above CSN_5.

TABLE I
Performance of CSN on CIFAR-10 without pruning

Model	*Train Accuracy*	*Test Accuracy*	*Optimizer*
CSN_1(Base)	84.03	74.46	Adam
CSN_2	96.53	75.32	Adam
CSN_3	92.26	69.33	Adam

TABLE II
PERFORMANCE OF CSN ON CIFAR-10 WITH PRUNING

Model	*Train Accuracy*	*Test Accuracy*	*Optimizer*
CSN_1(Base)	83.31	77.87	Adam
CSN_2	85.61	78.01	Adam
CSN_3	88.64	78.07	Adam
CSN_4	71.205	73.27	Adam

C. Semantic Segmentation on Salt Marsh Image Data Set

We tested the CSN training algorithm as a backbone for DeepLab-V3 [4] and compared it with a ResNet101 backbone for DeepLab-V3 and with Auto-DeepLab [10]. The performance of DeepLab-V3 was observed to be the best. The Auto-DeepLab search was performed on a smaller subset of the entire data set where the input images were resized to a low resolution for faster search. Auto-DeepLab's search produced a model that performed slightly worse than DeepLab-V3 as shown in Table III. The CSN was also tried and tested as a backbone for DeepLab-V3. The CSN stem for the backbone was created such that input was down-sampled to 1/4 before input to the core CSN and subsequently down-sampled to 1/4 after the core CSN resulting in an overall down-sampling rate of 1/16 for the backbone. This was done to test the efficacy of the structure of the CSN.

In our experiments, the CSN performed worse than both, Auto-DeepLab and DeepLab-V3. This is not a complete surprise given that testing on the CIFAR-10 data set also showed that the CSN design is not learning at par with modern CNNs. The degradation in mIoU values can be primarily attributed to the underrepresented classes. We observe that CSN_1 CSN_2 and CSN_3 exhibit similar performance despite the exponential variation in the number of parameters. We also see that the CSN_3 base model does not perform any worse than CSN_3. This indicates that the reuse of weights from CSN_1 and CSN_2 via dynamic expansion is not the cause for the lacklustre performance of CSN_3. However, it points towards a problem in the construction of either the base network or the cerebral module.

TABLE III
PERFORMANCE COMPARISON BETWEEN CSN AND CNNS ON SALT MARSH IMAGE DATA SET WITH NETWORK PRUNING

Model	*mIOU*	*Pixel Accuracy*	*Optimizer*
DeeplabV3	*0.451*	86.54	SGD
AutodeepLab	*0.3706*	85.049	SGD
CSN_1(Base)	*0.2483*	81.6	Adam
CSN_2	*0.2411*	81.6	Adam
CSN_3	*0.2361*	81.76	Adam
CSN_3(Base)	*0.2379*	81.04	Adam

IV. CONCLUSION

The consequence of the increased computational cost of exponential expansion and linear expansion is that the interconnection hypothesis is not implemented at a scale we had hoped. Since we are only able to reach hierarchies of up to 3 levels in the CSN, this is merely 1 level above that of modern CNN's. Another cause for concern is that the CSN base models at higher levels are prone to not learning anything new. The challenge of training extremely deep and wide networks is evident in our work. We believe that the solution to this problem will be pivotal in moving the field forward. A lot more work will be needed to find general-purpose neural network architectures that work and scale well on real-world problems. We believe that the reward for this effort will be in the form of solutions that are extremely unique and useful. The reason that our interconnection design has under-performed is likely due to the fact that the cerebral module or base network design needs to be revamped. We believe our interconnection design may not have met our criterion of interconnection richness due to the lack of a memory system [13]. Most modern CNNs are organized to compose features hierarchically before degradation of the input is observed. Unfortunately, in our efforts to find a novel CSN-based approach, we negated many architectural advancements that have contributed to the success of modern CNNs.

ACKNOWLEDGMENT

This research was supported in part by the US National Science Foundation GCE-LTER grant (OCE-1832178) to Dr. Bhandarkar and Dr.Hopkinson.

BIBLIOGRAPHY

[1] B. Bartoldson, A. S. Morcos, A. Barbu, and G. Erlebacher, "The generalization-stability tradeoff in neural network pruning," *ArXiv*, vol. abs/1906.03728, 2019.

[2] S. Beaulieu, L. Frati, T. Miconi, J. Lehman, K. Stanley, J. Clune, and N. Cheney, "Learning to continually learn," *arXiv*, 2020.

[3] N. Cammarata, S. Carter, G. Goh, C. Olah, M. Petrov, and L. Schubert, "Thread: Circuits," *Distill*, 2020, https://distill.pub/2020/circuits. DOI: 10.23915/distill.00024.

[4] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv*, 2017.

[5] J. Clune, "AI-GAs: AI-generating algorithms, an alternate paradigm for producing general artificial intelligence," *arXiv e-prints*, arXiv:1905.10985, arXiv:1905.10985, May 2019. arXiv: 1905 . 10985 [cs.AI].

[6] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv: Learning*, 2019.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Multimedia Tools and Applications*, 2015.

[8] D. A. Krueger, E. Caballero, J. Jacobsen, A. Zhang, J. Binas, R. L. Priol, and A. C. Courville, "Out-of-distribution generalization via risk extrapolation (rex)," *ArXiv*, vol. abs/2003.00688, 2020.

[9] J. Lee, J. Yoon, E. Yang, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," *ArXiv*, vol. abs/1708.01547, 2018.

[10] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," *CVPR*, 2019.

[11] A. Rockel, R. Hiorns, and T. Powell, "The basic uniformity in structure of the neocortex," *Brain : a journal of neurology*, vol. 103, no. 2, pp. 221–244, Jun. 1980, ISSN: 0006-8950. DOI: 10.1093/brain/103.2.221. [Online]. Available: https://doi.org/10.1093/brain/103.2.221.

[12] A. Veit, M. Wilber, and S. Belongie, *Residual networks behave like ensembles of relatively shallow networks*, 2016. arXiv: 1605.06431 [cs.CV].

[13] "What learning systems do intelligent agents need? complementary learning systems theory updated," *Trends in Cognitive Sciences*, vol. 20, no. 7, pp. 512–534, 2016, ISSN: 1364-6613. DOI: https://doi.org/10.1016/j.tics.2016.05.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1364661316300432.

[14] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, "Graph neural networks: A review of methods and applications," *ArXiv*, vol. abs/1812.08434, 2018.