Connectivity Learning by Gradient Descent

Karim Ahmed, Lorenzo Torresani
Department of Computer Science, Dartmouth College.

Although deep networks have recently emerged as the model of choice for many computer vision problems, in order to yield good results they often require time-consuming architecture search. To combat the complexity of design choices, prior work has adopted a modularized design which defines the network in as a composition of similar modules. This reduces architecture search to the problem of determining the number of modules to compose and how to connect such modules. Previous approaches have relied on simple rules of connectivity, e.g., connecting each module to only the preceding module or to all of the previous ones. Such simple connectivity rules are unlikely to yield the optimal architecture. In this work we remove these predefined choices and propose an algorithm to learn the connections between modules. Instead of being chosen a priori by the human designer, the connectivity is learned simultaneously with the network weights by optimizing the loss function of the end task using a modified version of gradient descent. We demonstrate our connectivity learning method on the problem of *multi-class image classification* using two architectures: ResNet [3] and ResNeXt [4]. Experiments on different datasets show that connectivity learning using our approach yields consistently higher accuracy compared to relying on traditional predefined rules of connectivity. Furthermore, in certain settings it leads to significant savings in number of parameters.

Modular architecture: Given a general modular architecture, we denote with $\mathbf{x}_j$ the input to the j-th module. When using ResNet the modules will be residual blocks, while for ResNeXt each module will consist of multiple parallel branches. We assume that the module implements a function $\mathcal{G}(.)$ parameterized by learnable weights θ_j. Thus, the output $\mathbf{y}_j$ computed by the j-th module is given by $\mathbf{y}_j = \mathcal{G}(\mathbf{x}_j;\theta_j)$. While this makes network design straightforward, it limits the topology of architectures.

Masked architecture: We introduce learnable *masks* defining the connectivity between modules. Each module j takes input from one or more of the preceding modules $k = 1,\ldots,j-1$. We define a binary mask vector for each module that are learned jointly with the weights, and controls the input pathway of that module. Let $\mathbf{m}_j = [m_{j,1}, m_{j,2}, \ldots, m_{j,j-1}]^\top \in \{0,1\}^{j-1}$ be the binary mask vector of the j-th module. If $m_{j,k} = 1$, the activation volume produced by the k-th module is fed as input to the j-th module. If $m_{j,k} = 0$, the output from the k-th module is ignored. Let $\mathbf{y}_k$ be the output activation tensor computed by the k-th module, then the input $\mathbf{x}_j$ to the j-th module is given by: $\mathbf{x}_j = \sum_{k=1}^{j-1} m_{j,k} \cdot \mathbf{y}_k$. We note that under this model we no longer have predefined connectivity among modules. Instead, the mask $\mathbf{m}_j$ determines *selectively* for each module which outputs from the previous modules will be aggregated to form the input to the next block.

MaskConnect: learning to connect We refer to our learning algorithm as MaskConnect. It performs joint optimization of a given learning objective ℓ with respect to both the weights of the network (θ) as well as the masks ($\mathbf{m}$). The weights have real values, while the masks have binary values. To learn these binary parameters, we adopt a modified version of backpropagation. During training we store and update a real-valued version $\tilde{\mathbf{m}}_j \in [0,1]^{j-1}$ of masks. We stochastically binarize the real-valued masks into binary-valued vectors $\mathbf{m}_j \in \{0,1\}^{j-1}$ which are used for the forward and backward propagation steps. During the parameters update step, we update the real-valued masks. We enforce a constrain that there can be only K active entries in each binary mask, where $\sum_{k=1}^{j-1} m_{j,k} = K$. We refer to this hyperparameter K as the *fan-in* of a module. In Figure 1 we show the application of MaskConnect to ResNet and ResNeXt architectures. For ResNet, we change the input to block $j+1$ to be $\mathbf{x}_j \leftarrow \sum_{k=1}^{j-1} m_{j,k}[\mathcal{F}(\mathbf{x}_k;\theta_k) + \mathbf{x}_k]$, where the binary parameters $m_{j+1,k}$ are learned by our approach. For a ResNeXt model of C branches, the modified input to block $j+1$ in the i-th branch is given by: $\mathbf{x}_j^{(i)} \leftarrow \sum_{k=1}^{C} m_{j,k}^{(i)} \left[\mathcal{F}(\mathbf{x}_k^{(i-1)};\theta_k^{(i-1)}) + \mathbf{x}_k^{(i-1)}\right]$.

Experiments: We tested our approach on the task of *image categorization* using ResNet and ResNeXt architectures. In Table 1 and Table 2, we report the results achieved on CIFAR-100 based on ResNet and ResNeXt, respectively. Table 3 shows the results on ImageNet for ResNeXt architecture.

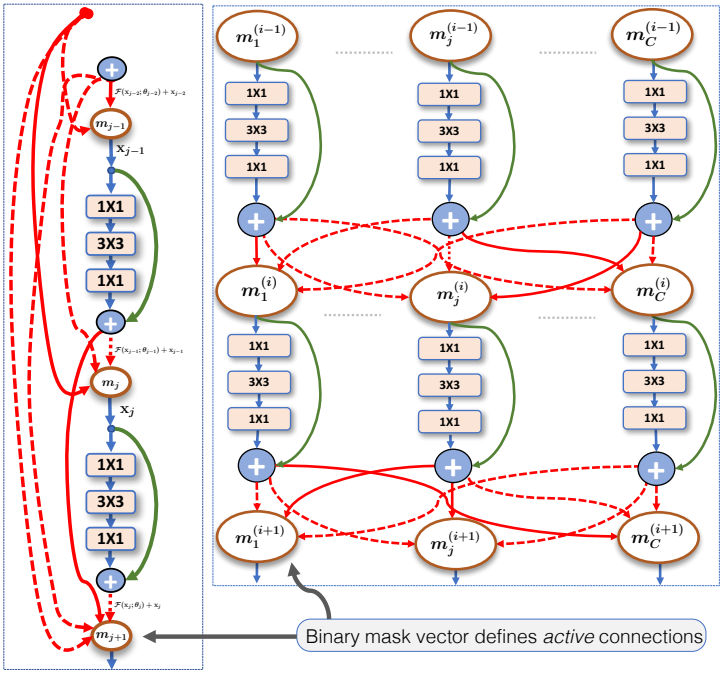

Figure 1: Application of MaskConnect to two forms of modular network: **(a)** (left) ResNet and **(b)** (right) multi-branch ResNeXt

Table 1: CIFAR-100 accuracies achieved by **ResNet** trained using full connectivity (Fixed-Prev), and the connectivity learned by MaskConnect (Learned)

Model	Connectivity	Accuracy (%)
ResNet-38	Fixed-Prev, K=1 [3]	68.54
	Learned, K=10	**70.40**
ResNet-74	Fixed-Prev, K=1 [3]	70.64
	Learned, K=15	**72.81**
ResNet-110	Fixed-Prev, K=1 [3]	71.21
	Learned, K=20	**73.15**

Table 2: CIFAR-100 accuracies achieved by **ResNeXt** trained using full connectivity (Fixed-Full), a fixed random connectivity (Fixed-Random), and the connectivity learned by MaskConnect (Learned).

Architecture	Connectivity	Params		Accuracy (%)
{Depth, Bottleneck width, Cardinality}		*Train*	*Test*	*best (mean±std)*
{29,8,8}	Fixed-Full, K=8 [4]	0.86M	0.86M	73.52 (73.37±0.13)
	Learned, K=1	0.86M	0.65M	**73.91** (73.76±0.14)
	Learned, K=4	0.86M	0.81M	**75.89** (75.77±0.12)
	Fixed-Random, K=4	0.86M	0.85M	72.85 (72.66±0.24)
{29,64,8}	Fixed-Full, K=8 [4]	34.4M	34.4M	82.23 (82.12±0.12)
	Learned, K=1	34.4M	20.5M	**82.31** (82.15±0.15)
	Learned, K=4	34.4M	32.1M	**84.05** (83.94±0.11)
	Fixed-Random, K=4	34.4M	34.3M	81.96 (81.73±0.20)

Table 3: ImageNet accuracies achieved by ResNeXt using full connectivity (Fixed-Full) and the connectivity learned by MaskConnect (Learned)

Architecture	Connectivity	Accuracy	
{Depth, Bottleneck width, Cardinality}		*Top-1*	*Top-5*
{50,4,32}	Fixed-Full, K=32 [4]	77.8	93.3
	Learned, K=16	**79.1**	**94.1**
{101,4,32}	Fixed-Full, K=32 [4]	78.8	94.1
	Learned, K=16	**79.5**	**94.5**
{101,4,64}	Fixed-Full, K=64 [4]	79.6	94.7
	Learned, K=32	**79.8**	**94.8**

The full article was published at ECCV 2018 [1, 2].
Available at https://arxiv.org/abs/1807.11473.

[1] Karim Ahmed and Lorenzo Torresani. Connectivity learning in multi-branch networks. *arXiv preprint arXiv:1709.09582*, 2017.

[2] Karim Ahmed and Lorenzo Torresani. Maskconnect: Connectivity learning by gradient descent. In *European Conference on Computer Vision (ECCV)*, 2018.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, 2016.

[4] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017.