# A Multistage Backward Differentiable Method for Constructing Light Convolutional Neural Networks

Fanghui Xue, Jack Xin
*University of California, Irvine*
Irvine, CA, USA.
{fanghuix, jack.xin}@uci.edu

Jiancheng Lyu, Shuai Zhang, Yingyong Qi
*Qualcomm AI Research**
San Diego, CA, USA.
{jianlyu, shuazhan, yingyong}@qti.qualcomm.com

***Abstract*—We propose a multistage differentiable method to select convolutional channels and construct light neural networks from a heavy network for inference on a subset of a big data set. The selection proceeds backward in layers and utilizes sparse penalty to diversify channel scores. The resulting light network gains sizable accuracy over the baseline heavy network.**

***Index Terms*—differentiable channel selection, light network.**

## I. Introduction

A major task of deep learning in computer vision is to identify objects and classify images as accurately as possible. Though researchers can afford a staggering amount of training time and an abundance of GPUs to search a high end deep neural network on large scale data sets, it is a challenge for industry to realize such a network in their products of limited energy budget. Moreover, the classes of objects most often encountered are on the order of dozens, far less than those in the big data sets for training heavy duty deep networks. In this paper, we initiate the study of *constructing a light weight network by performing structure selection from a high end (heavy) deep network*. Specifically, we study the removal of non-important channels in the convolution layers *when the inference is on a subset with fewer classes of objects selected from the big data set*. This is different from standard channel pruning [1] where the inference of the pruned network remains on the big data set associated with the heavy network.

Our approach is along the line of Neural Architecture Search (NAS) [2] that succeeded in automatically designing neural network models by machines. Among the NAS style training algorithms, Differentiable Architecture Search (DARTS) [3] is the most efficient. Let us consider the problem of extracting a light weight network to classify a dozen or so categories (classes) of images from a well-trained and heavy-duty baseline model which is designed to distinguish 1,000 classes. As the number of classes is reduced, it should be enough to use less channels to extract features since too detailed descriptions are redundant when there are not many candidates for classification. Hence channel selection is meaningful for model size reduction without degrading accuracy. We mention that compared to selecting layers of the network, channel selection is more stable and accuracy preserving. Instead of relying on the magnitude of weights in each channel, we opt to optimize on channel importance parameters (scores) as in [2], [3]. This makes the selection easy to interpret provided the scores are sufficiently distinct (away from being uniform to cause ambiguities). To this end, *our contribution is to employ a sparsity promoting penalty on channel scores, the difference of $\ell_1$ and $\ell_2$ norm [4] denoted by $\ell_{1-2}$, which is differentiable and sharper than $\ell_1$.*

The rest of the paper is organized as follows. In section II, we introduce the mathematical formulation of our approach and the training algorithm. In section III, we show experimental results where our method out-performs the state-of-the-art method (squeeze and excitation [5]) while gaining accuracy and reducing network complexity.

## II. Channel Selection Algorithm

As in [3], we introduce a list of channel scoring parameters. That is to multiply the feature map $F_{i,j}$ by a parameter $\alpha_{i,j}$, where $(i, j)$ are the layer and channel indices (see Fig. 1):

$$\tilde{F}_{i,j} = \alpha_{i,j} F_{i,j}.$$

Here $\tilde{F}_{i,j}$ is the new feature map, with the rest of the network unchanged. Our task is to learn the value of those parameters, and remove the channels whose scores are low. That means we can drop all the filters connected to those channels as well, which helps to reduce the model size, and saves time and power for computation.

The benefit of differentiable methods is that they can learn parameters continuously by gradient descent. One may directly adopt the DARTS algorithm [3]: $\alpha^{t+1} = \alpha^t - \eta^t_\alpha \nabla_\alpha f_{val}(w^t, \alpha^t)$, $w^{t+1} = w^t - \eta^t_w \nabla_w f_{train}(w^t, \alpha^{t+1})$, where $\eta$ is the learning rate, $w$ is the weight, $f_{train}(w, \alpha)$ and $f_{val}(w, \alpha)$ are the training and validation loss functions. However, this may cause some problems.

First, the values of the learned $\alpha$'s are likely to be very close since the initialization is random. As a result, it can be hard to determine which channels to drop without external force. To solve this problem, we impose penalty on the $\alpha$'s to make it sparse. In other words, a large proportion of the $\alpha$'s we obtain should be zero or very small. To this end, we consider the $\ell_{1-2}$ penalty (a better differentiable approximation to $\ell_0$ than $\ell_1$) [4], and so the *new penalized validation loss function* becomes:

$$L_{val}(w, \alpha) = f_{val}(w, \alpha) + \lambda \|\alpha\|_{\ell_{1-2}},$$

The work was partially supported by NSF grants IIS-1632935, DMS-1854434, Qualcomm Faculty Award, and Qualcomm AI Research.

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

978-1-7281-4087-2/19/$31.00 ©2019 IEEE
DOI 10.1109/AI4I.2019.00029

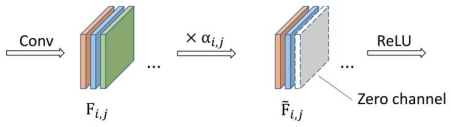


Fig. 1. Channel scoring parameters act on feature maps in the i-th layer.

where $\|\alpha\|_{\ell_{1-2}} = \|\alpha\|_1 - \|\alpha\|_2$, while the training loss function is still the same.

Another problem is that, if we select the channels in all the layers simultaneously, the accuracy will be affected adversely. This is because the penalty forces the $\alpha$'s to converge too quickly before the weights are trained well. Our strategy is to fix all but one layer, select the channels in the fixed layer, and then repeat for the subsequent layers. As the first few layers are more sensitive, there is not much room to select channels in those layers. Hence we start from the last layer and go backward, then terminate at some optimal intermediate layer. In each stage, we select channels in one layer at a time. Let $N$ and $m$ (in our experiments $m = 10$) be the indices of the last layer and the intermediate layer we choose to end selection, we summarize the above discussion in the following:

**Algorithm 1:** Multistage Backward Differentiable Method (MBDM)

Input $N, m$. Initialize $w^0$ and $\alpha^0$.
**for** *i = N, N-1, ..., m* **do**
  **while** *not converged* **do**
    $\alpha_i^{t+1} \leftarrow \alpha_i^t - \eta_\alpha^t \nabla_{\alpha_i} L_{val}(w^t, \alpha^t)$
    $\alpha_i^{t+1} \leftarrow \sigma(\alpha_i^{t+1}) / \|\sigma(\alpha_i^{t+1})\|_1$
    $w^{t+1} \leftarrow w^t - \eta_w^t \nabla_w L_{train}(w^t, \alpha^{t+1})$
  **end**
**end**

In Algorithm 1, we apply ReLU function $\sigma(\cdot)$ to zero out all the negative $\alpha$'s and take $\ell_1$ norm so that all these scoring parameters are non-zero and sum to 1 in each layer.

## III. Experiments

Given the 1,000-class ImageNet data [6] and ResNet-18 [7] as the baseline model, we extract a light model on a subset of 10 or 20 classes of data via channel selection. Depending on the number of sub-classes (10 or 20), the training data set contains 13,000 or 26,000 images, respectively. To test the capability of the algorithm, the 10 classes we select are more random and distinguishable, while the 20 classes are divided into 5 categories and the classes within each category are similar (Table I).

The whole training procedure consists of two steps. First, we run the MBDM algorithm to learn the values of the $\alpha$'s and select a slim model. Second, we train the full model for ease of programming with the $\alpha$'s fixed (a slim model in essence as many $\alpha$'s are zero), so that the weights in each channel can be fine-tuned. As seen in Table II, our algorithm out-performs considerably the baseline ResNet-18 model in distinguishing 10 or 20 classes. Though the gap is closer, our algorithm also beats the SE-ResNet [5], a recent model that handles channel-wise information flow by enhancing their relations. What is more, our model is much lighter than the baseline model as many of the channels are not activated, which is supported by Table III.

TABLE I
THE 10 & 20 CLASSES.

| | | | | | |
|---|---|---|---|---|---|
| 10 Classes | Shark, Ant, Panda, Train, Castle, Piano, Umbrella, Broccoli, Lemon, Volcano | | | | |
| 20 Classes | *Cats* | *Dogs* | *Vehicles* | *Architectures* | *Landscapes* |
| | Egyptian | Sheepdog | Bike | Bridge | Valley |
| | Persian | Bulldog | Sports car | Dam | Sandbar |
| | Tiger | Mountain | Scooter | Castle | Cliff |
| | Siamese | Maltese | Cab | Fence | Volcano |

TABLE II
COMPARISON OF RESNET, SE-RESNET, MBDM (AVERAGE TEST ACCURACY OF 5 RUNS), ALL TRAINED FROM SCRATCH.

| Number of classes | ResNet | SE-ResNet | MBDM |
|---|---|---|---|
| 20 | 83.8 | 86.8 | **87.9** |
| 10 | 87.0 | 91.4 | **93.7** |

TABLE III
SPARSITY (PENCENTAGE OF ZERO CHANNELS AMONG ALL CHANNELS) IN THE LAST 8 CONV LAYERS FOR THE EXPERIMENT OF 20 CLASSES.

| Layer | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|
| Zero chs | 210 | 60 | 221 | 58 | 246 | 261 | 427 | 206 |
| All chs | 256 | 256 | 256 | 256 | 512 | 512 | 512 | 512 |
| Sparsity | 82.0 | 23.4 | 86.3 | 22.7 | 48.0 | 51.0 | 83.4 | 40.2 |

## IV. Conclusions and Discussions

We developed a novel differentiable method on channel selection for a subset inference problem based on a heavy network on a big data set. We showed that the ResNet-18 model with channel selection performs far better than the full model on a 10-class or 20-class data set (a subset of ImageNet). Our method generalizes to other components of network architecture on similar tasks. In future work, we plan to study other models such as MobileNet and on a variety of sub-classes.

## References


[1] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *ICLR*, 2019.

[2] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[3] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055; ICLR*, 2019.

[4] E. Esser, Y. Lou, and J. Xin, "A method for finding structured sparse solutions to non-negative least squares problems with applications," *SIAM J. Imaging Sciences*, vol. 6, pp. 2010–2046, 2013.

[5] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.