

# Supplementary Material for Adaptive End-to-End Budgeted Network Learning via Inverse Scale Space

Zuyuan Zhong<sup>1</sup>  
zyzhong19@fudan.edu.cn

Chen Liu<sup>2</sup>  
cliudh@connect.ust.hk

Yanwei Fu<sup>\*1</sup>  
yanweifu@fudan.edu.cn

<sup>1</sup> School of Data Science,  
Fudan University,  
Shanghai, China

<sup>2</sup> The Hong Kong University of Science  
and Technology,  
Hong Kong

\* Corresponding author

## 1 Algorithm

The details of our AdeNeL algorithm is as described in Algorithm 1.

## 2 Experiment

### 2.1 Implement Details

For VGG-16 on CIFAR10 dataset, we adopt the following hyper-parameters:  $\mu = 200$ ,  $lr = 0.05$ ,  $batch\_size = 128$ ,  $J = 10$ ,  $FN = 4$ ,  $weight\_decay = 0.0005$ ,  $finetune\_epoch = 80$ ,  $\tau = 0.9$  for the first two stages and  $\tau = 0.3$  for other stages. The initialized number of channels is 4 for all convolutional layers. For ResNet-20 on CIFAR10 dataset, we adopt the following hyper-parameters:  $\mu = 200$ ,  $lr = 0.05$ ,  $batch\_size = 128$ ,  $J = 10$ ,  $FN = 1$ ,  $weight\_decay = 0.0005$ ,  $finetune\_epoch = 80$ ,  $\tau = 0.8$  for all convolutional layers. The initialized number of channels is 4 for all convolutional layers.

### 2.2 Results on MNIST

The visualization of growing on MNIST dataset of our method are shown in the main text. Here we present the details of the growing process and results. The seed network of LeNet-5 has 4 filters in each convolutional layer and 32/16 linear units in 2 fully-connect layers. When we find any layer is lack of capacity we add 2 filters/16 linear units to these layers. The full LeNet-5, the seed network and the final model train by AdeNeL can be seen in Table 1. One can see that our AdeNeL increase the number of first convolutional layer while reduce the units of fully-connected layers. This leads to the reduction of network parameters

**Algorithm 1** AdeNeL

**Input:**  $Net_{seed}$ : the seed backbone network of a specific network such as ResNet18;  $\tau$ : the threshold of capacity metric;  $J$ : the number of epochs in each growing round;  $r, n$ : the fix ratio and fix number for growing filters;  $s$ : growing indicator;  $P_b, F_b$ , the budgets for parameters and FLOPs.

**Output:** Trained proper-parameterized Model

```

1: for  $epoch = 0$  to  $\infty$  do
2:   train the model  $Net$ ;
3:   if  $epoch \bmod J$  then
4:     for  $l = 1$  to  $L$  do
5:       Calculate  $s^l$  of  $l$ -th convolutional layer;
6:       if  $C^l < 1 - \tau$  then
7:         add  $\max(n, \text{int}(r * |W^l|))$  randomly initialized filters to this layer;
8:       end if
9:     end for
10:  end if
11:  calculate parameters  $P_c$  and FLOPs  $F_c$  of current model;
12:  if  $P_c \geq P_b$  or  $F_c \geq F_b$  then
13:    Stop growing;
14:  end if
15:  Fine-tuning;
16: end for

```

	LeNet-5	Seed Network	AdeNeL
Conv1	6	4	12
Conv2	16	4	16
FC1	120	32	48
FC2	84	16	64
Params (K)	61.68	4.43	<b>46.13</b>
Acc. (%)	99.29	-	<b>99.32</b>

Table 1: Comparison between LeNet-5, the seed network and the learned network of AdeNeL using LeNet-5 on MNIST.

while achieve better performance due to the more features extracted by the first convolutional layer.

## 2.3 Effects of Growing Scale

This paper our growing strategy is adding a *fixed number* (FN) of new filters to the convolutional layers which are lack of capacity. For comparison, we propose several strategies: *fixed ratio* (FR) strategy adds new filters in a fixed proportion (typically 0.1) to the number of existing filters at each round and *mixed strategy* (MS) strategy add  $N_{MS} = \max(N_{FN}, N_{FR})$  filters at each growing round. We conduct experiments using VGG16 backbone on CIFAR10. The results are shown in Table 2, one can see that: (1) Small growing scale, either small FR, FN or MS, achieves good performance but results in larger growing rounds, which means more training time. (2) The FN strategy can achieve the best performance under different

Method	Ratio	Num.	Params (M)	FLOPs (G)	Rounds	Acc. (%)
FN	-	6	2.49	0.202	35	93.62
	-	8	2.47	0.204	25	93.81
	-	10	2.60	0.206	20	93.55
	-	12	2.57	0.205	17	93.76
FR	0.10	-	2.47	0.211	33	93.24
	0.15	-	2.27	0.214	24	93.27
	0.20	-	3.35	0.202	18	93.31
	0.25	-	5.14	0.214	15	93.08
MS	0.1	4	2.41	0.201	28	93.62
	0.1	6	2.66	0.201	25	93.36
	0.1	8	2.60	0.206	21	93.36

Table 2: Results of AdeNeL using VGG16 backbone on CIFAR10 dataset with different growing scale strategies: fixed ratio (FR), fixed number (FN) and mixed strategy (MS). We set the FLOPs budget as 0.2G. The rounds means the number of growing times during growing process.

growing scale. (3) With large growing scale (which means less growing rounds and more efficient), FN retains good performance. In summary, FN strategy can achieve better performance and more stable.

## 2.4 Filter Configuration Learned by AdeNeL

Table 2.4 compares the filter configuration of ResNet32, ResNet32-3x [10] and model learned by our AdeNeL. Our model is learned by setting the parameters and FLOPs of ResNet32-3x as budgets. The largest difference between global network width expansion [10] and AdeNeL is that the former expands the network width in network level, *i.e.*, expands all convolutional layers with same times, while AdeNeL expands network width in layer level which allows more delicate expansion. Specifically, AdeNeL can spend more filters in some layers while less filter in other layers according their capacity. This can be seen in Table 2.4 that the model leaned by AdeNeL has more filters in the shallower layers whose output size is larger while less in layers with smaller output size. This make the models learned by AdeNeL enjoy performance improvement and remarkably reduction of parameterz/FLOPs.

## 2.5 Adaptiveness of AdeNeL

Another important merit of our AdeNeL is that the AdeNeL can be adaptively learned by varying the number of training instances. Specifically, we randomly sample 50/100/500/1000 images in each class in CIFAR10 and get 4 subsets. We use the backbone of ResNet8, which only preserves the first BasicBlock in each block of ResNet20, with only 4 filters in each convolutional layer, as the seed network. We set the hyper-parameters as:  $\mu = 100$ , initial learning rate is 0.1. Considering the iteration in each epoch of different dataset, we set mini-batch size as 16/16/64/128/128 and  $J = 50/50/25/25/5$  for dataset with 50/100/500/1000/5000 images in each class. We using the FN growing strategy which growing 4 filters to layers in each growing round. Figure 1 shows the results of these experiments. One can see that along the increasing of the dataset scale, the size and performance of models learned by AdeNeL

		width=3x	AdeNeL
ResNet32	output size	Maps	Maps
Conv_1	$32 \times 32$	48	28
Layer1.1	$32 \times 32$	$48 \times 2$	$62 \times 2$
Layer1.2	$32 \times 32$	$48 \times 2$	$68 \times 2$
Layer1.3	$32 \times 32$	$48 \times 2$	$57 \times 2$
Layer1.4	$32 \times 32$	$48 \times 2$	$36 \times 2$
Layer1.5	$32 \times 32$	$48 \times 2$	$36 \times 2$
Layer2.1	$16 \times 16$	$96 \times 2$	$167 \times 2$
Layer2.2	$16 \times 16$	$96 \times 2$	$106 \times 2$
Layer2.3	$16 \times 16$	$96 \times 2$	$81 \times 2$
Layer2.4	$16 \times 16$	$96 \times 2$	$74 \times 2$
Layer2.5	$16 \times 16$	$96 \times 2$	$62 \times 2$
Layer3.1	$8 \times 8$	$192 \times 2$	$127 \times 2$
Layer3.2	$8 \times 8$	$192 \times 2$	$106 \times 2$
Layer3.3	$8 \times 8$	$192 \times 2$	$201 \times 2$
Layer3.4	$8 \times 8$	$192 \times 2$	$106 \times 2$
Layer3.5	$8 \times 8$	$192 \times 2$	$48 \times 2$
FC	-	10	10
Params	-	4.16 M	<b>2.49 M</b>
FLOPs	-	619 M	<b>569 M</b>
Acc.	-	94.81%	<b>95.07%</b>

Table 3: Filter configuration of ResNet32 (width=3x) and ResNet32 learned by AdeNeL.

increase also increase. This means for different size of dataset, AdeNeL can learn reasonable size of networks within the budget.

For comparison, we also train the ResNet-8 and its expanded versions from scratch with SGD with momentum (0.9) and weight decay (0.0001), the batch size are the same as corresponding AdeNeL experiments. The results are in Table 4. We show the baselines with similar params and FLOPs as models trained by AdeNeL. One can see that with similar complexity, models trained by AdeNeL perform better, which indicates that our AdeNeL can learn better networks in scenes of different dataset scale.

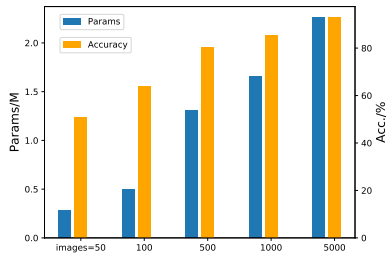


Figure 1: Classification accuracy with different skip connection weights on (a): clean images and (b): adversarial examples generated by FGSM-10 attack, along the reconstruction iterations.

N/C	model	Params	FLOPs	Acc.(%).
50	ResNet8-2x	298.03 k	48 M	45.59
	AdeNeL	<b>284.29 k</b>	<b>55 M</b>	<b>50.73</b>
100	ResNet8-3x	668.22 k	108 M	60.81
	AdeNeL	<b>495.35 k</b>	<b>116 M</b>	<b>63.91</b>
500	ResNet8-4x	1.19 M	191 M	79.35
	AdeNeL	<b>1.31 M</b>	<b>227 M</b>	<b>80.38</b>
1000	Baseline 5x	1.85 M	298 M	84.75
	ResNet8-filter	<b>1.66 M</b>	<b>351 M</b>	<b>85.21</b>
5000	Baseline 5x	1.85 M	298 M	92.40
	Baseline 6x	2.66 M	428 M	93.05
	AdeNeL	<b>2.26 M</b>	<b>315 M</b>	<b>92.87</b>

Table 4: Comparison between ResNet8 and its expanded versions with AdeNeL on different scale of datasets. N/C means the number of image in each class.

References

[1] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.