Adaptive End-to-End Budgeted Network Learning via Inverse Scale Space

Zuyuan Zhong[1]
zyzhong19@fudan.edu.cn
Chen Liu[2]
cliudh@connect.ust.hk
Yanwei Fu*[1]
yanweifu@fudan.edu.cn

[1] School of Data Science, Fudan University, Shanghai, China

[2] The Hong Kong University of Science and Technology, Hong Kong

* Corresponding author

Abstract

This paper studies the task of budgeted network learning [35] that aims at discovering good convolutional network structures under parameters/FLOPs constraints. Particularly, we present a novel Adaptive End-to-End Network Learning (AdeNeL) approach that enables learning the structures and parameters of networks simultaneously within the budgeted cost in terms of computation and memory consumption. We keep the depth of networks fixed to ensure a fair comparison with the backbones of competitors. Our AdeNeL learns to optimize both the parameters and the number of filters in each layer. To achieve this goal, our AdeNeL utilizes an iterative sparse regularization path – Discretized Differential Inclusion of Inverse Scale Space (DI-ISS) to measure the capacity of the networks in the training process. Notably, the DI-ISS imports a group of augmented variables to explore the inverse scale space and this group of variables can be used to measure the redundancy of the network. According to the redundancy of the current network, our AdeNeL can choose appropriate operations for current network such as adding more filters. By this strategy, we can control the balance between the computational cost and the model performance in a dynamic way. Extensive experiments on several datasets including MNIST, CIFAR10/100, ImageNet with popular VGG and ResNet backbones validate the efficacy of our proposed method. In specific, on VGG16 backbone, our method on CIFAR10 achieves 92.71% test accuracy with only 0.58M (3.8%) parameters and 43M (13.7%) FLOPs, comparing to 92.90% test accuracy and 14.99 M parameters, 313M FLOPs of the full-sized VGG16 model.

1 Introduction

Deep Convolutional Neural Networks (CNNs) have made remarkable achievements in the Computer Vision community. In real-world applications, we are facing a much more severe problem of using the networks as discussed in [35]: it is very difficult to afford the tremendous computational cost and memory footprint to deploy the full CNNs models in many practical applications; so how to get the reasonable good generalization performance

with restricted computational resources? As a result, designing the light-weight network structures have become pretty prevailing. Typically, engineers and researchers will manually design the architectures [12, 16, 30, 33]. However, it requires expert-level efforts to tune the key network hyper-parameters, for example, the channel configuration of each layer and the number of layers.

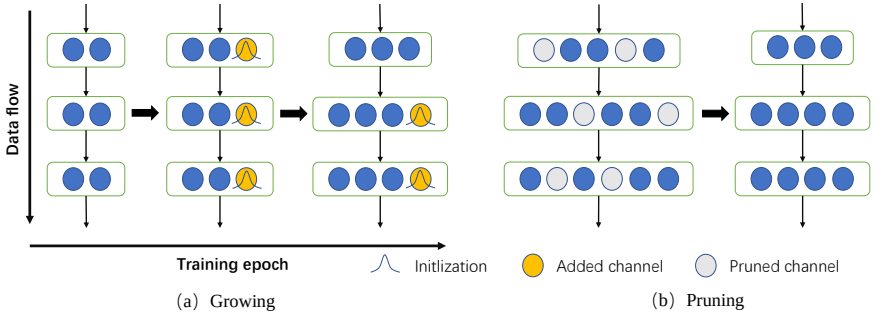

Figure 1: Comparisons between the pruning method and our growing strategy: pruning starts from full-sized network and remove unimportant filters while our method greedily increases filters from the small seed network.

To retain or achieve better predictive quality in image classification task under memory or computation constraints, practitioners have to find better network structures from the given specific task. Pruning techniques [19, 21] remove the unimportant weights from full-sized models to meet the parameters/FLOPs budgets. To maintain the accuracy of full-sized models, they have to fine-tune or train the pruned networks from scratch, which brings extra computational costs. Neural Architecture Search (NAS) techniques [6, 31, 41, 42] are also used to search the width configuration for budgeted network learning. These methods train a super-network and then search the best sub-network under the FLOPs/parameters budgets from all of possible sub-networks. Unfortunately, NAS methods are naturally much more computationally expensive than training a budgeted network from scratch.

In this paper, we propose a novel adaptive end-to-end budgeted network learning (AdeNeL) algorithm to learn budgeted networks during the training process. Contrary to pruning [19, 21] and width searching [31, 41, 42] which start from complex models and target for small compact models, we propose to solve it in a different way. In detail, we attempt to start with small seed networks and gradually increase their capacity during the training process. Both the weights and configurations of the models are optimized; and the key is to evaluate whether the model capacity is sufficient. To this end, we repurpose the well-known Differential Inclusion of Inverse Scale Space (DI-ISS) in applied mathematics [8, 15, 29]. The DI-ISS learns an *inverse scale space* (ISS) coupled for *primal parameter space*. While ISS is derived from LASSO or Group LASSO penalized in the primal space, the ISS can thus enable the implicit feature selection of the important parameters from the parameter space. Notably, the ISS utilizes a group of augmented variables to explore important structure in [8, 15] by learning from a predefined complex network to predict the final weights and important sub-structure simultaneously in training.

Comparably, our AdeNel starts from a simple seed network, and iteratively measures in ISS whether the model capacity should be increased . Essentially, we believe that the primary space should be slightly over-parameterized; and if most of parameters in primary space are taken as important ones by ISS, we should add the model capacity. Specifically, the model is initialized by a fixed depth and only a few filters for each layer and our AdeNeL learns to

increase the number of filters and optimizes the network parameters at the same time. During the training process, our AdeNeL simultaneously learns to optimize the network weights from the primary network parameter space, and constructs the support set of weights in the inverse scale space, respectively. The support set is utilized to measure the capacity of the current network structure. Accordingly, in the primary parameter space, AdeNeL will grow the network by dynamically adding more filters to the layers of not enough capacity.

Extensive experiments on several benchmark image classification datasets including MNIST, CIFAR10/100, ImageNet with popular VGG and ResNet validate the merits of our proposed method. For example, on VGG16 and CIFAR10, our method achieves 92.71% test accuracy with only 0.58M (3.8%) parameters and 43M (13.7%) FLOPs, comparing to 92.90% test accuracy and 14.99M parameters, 313M FLOPs of the full-sized VGG16 model.

Contributions. We summarize the contributions as follows.

1) We propose a novel budgeted network learning approach - Adaptive End-to-End Network Learning (AdeNeL): one starts from small-size models and then delves into proper-size model according to the capacity of current networks to the datasets in an end-to-end manner.

2) We re-purpose DI-ISS in Statistics to measure the capacity of network capacity. DI-ISS can efficiently help learn the support set of weights in the inverse scale space, from the primary network parameter space.

3) Empirically, we validate our AdeNeL on several benchmarks datasets and network backbones, and show our method enables the efficient budgeted network learning.

2 Related Work

Budgeted Network Learning. Previous works learn efficient networks in the budgets of memory cost or predictive quality. Dynamic networks [1, 2, 23, 26] improve the efficiency of networks by conditionally selecting the modules. For example, Bolukbasi *et al*. [2] studied a network evaluation scheme that adaptively selects the modules network to evaluate the inputs. This avoids the computational cost associated with full evaluation of the network. In contrast, our AdeNeL learns the balanced networks of both good prediction performance and model size, computation costand total training computational cost, in better favor of networks directly applied on the low-resource devices.

Neural Architecture Search (NAS) aims to design better architectures automatically. The early works [44, 45, 46] use reinforcement learning to search architectures on CIFAR10 dataset. Recently, NAS methods are used to search the width configuration of networks [31, 41] and get budgeted networks. They train large super-networks and search the best budgeted sub-networks within the super-networks. However, the NAS methods are naturally more computational expensive than training a budgeted network from scratch because they have an extra searching process. Our AdeNeL strategy enjoys the benefits of grow in training paradigm that start from small seed networks and growing during the training process.

Pruning network [9, 10, 19, 21, 24, 39] aims to reduce the parameters/FLOPs of networks while maintain the prediction quality by removing the weights/filters of full-sized networks. For example, l_1-pruning [19] removes filters with smaller l_1 norm to meet the requirement of parameters/FLOPs budgets. To maintain the performance, pruning methods usually need to fine-tune or train the found networks from scratch. Besides, the control of pruning rate in each layer is still manual. On the contrary, our AdeNeL starts from small seed models and growing filters during the training process until meeting parameters/FLOPs budgets in an automatic and end-to-end way.

Growing network. Network Morphism [3, 4, 36] accelerates training wider or deeper networks by morphing the smaller networks to the larger ones. Splitting filters [38], splits channels into several new channels based on designed loss function. Instead of exploring network width, AutoGrow [37] explores the effects of expanding network depth by greedily adding the layers during training process. Recent proposed work [42] combines pruning and NAS methods, which grows and prunes networks during training process in a continuous way. Although both [42] and our AdeNeL are growing networks from small to large, they are fundamentally different. [42] introduces indicating variables as the external representation of networks and update the network architecture through gradient descent. While our method introduces a group of variables which serve as internal representation, *i.e.*, support set, of network weights. And we use this support set to measure the capacity of the current networks and potentially add filters to the networks.

DI-ISS [29], can be related to non-smooth Mirror Descent Algorithm (MDA) in convex optimization [25]. Huang *et al*. [14, 15] learned high-dimensional sparse linear models and found applications in medical image classification [32] and computer vision [8, 43]. Its convergence has been studied mostly in the convex setting [27, 40], and recently in non-convex deep learning setting [8]. This work re-purposes DI-ISS through progressively learning an inverse scale space, in order to facilitate AdeNeL via a coupled inverse scale space.

3 Methodology of AdeNeL

Problem Setup. In supervised learning, convolutional networks with parameters W learn the mapping $\Phi_W : \mathcal{X} \to \mathcal{Y}$ from input space $\mathcal{X}$ to label space $\mathcal{Y}$, by the objective

$$W = \underset{W}{\operatorname{argmin}} \mathbb{E}_{(x,y)\sim\mathcal{P}_{data}} \left[\mathcal{L}\left(x,y;W\right)\right], \tag{1}$$

where $\mathcal{P}_{data}$ denotes the training data distribution and $\mathcal{L}(x,y;W)$ is loss function. Although the weights W are learned from data distribution, it is hard to define the capacity of W on representing data. In this paper, uniting the pruning methods [19, 21] which use the internal information of networks and network width searching method [42] which uses an external variable as an indicator, we use an auxiliary variable, Γ, as the internal approximation and indicator of weights W. Formally, the augmented loss function is

$$\bar{\mathcal{L}}\left(W,\Gamma\right) = \mathcal{L}\left(W\right) + \frac{1}{2\nu}\|W-\Gamma\|_2^2, \tag{2}$$

where $\nu > 0$ is a hyper-parameter.

3.1 Differential Inclusion of Inverse Scale Space

Directly solving problem 2 is difficult. Fortunately, we can take this problem into the Inverse Scale Space for solution. Specifically, we extend the neural network parameters W to (W,Γ), where Γ characterizes the support set of large primal parameter W, by the following inverse-scale-space dynamics introduced in [15, 29] for updating the parameter (W_t,Γ_t) at time t,

$$\frac{\mathrm{d}W_t}{\mathrm{d}t} = -\nabla_W \bar{\mathcal{L}}(W_t, \Gamma_t), \tag{3a}$$

$$\frac{\mathrm{d}Z_t}{\mathrm{d}t} = -\nabla_\Gamma \bar{\mathcal{L}}(W_t, \Gamma_t), \tag{3b}$$

$$Z_t \in \partial\bar{\Omega}(\Gamma_t), \tag{3c}$$

where Z is a sub-gradient of $\bar{\Omega}(\Gamma) := \Omega_1(\Gamma) + \|\Gamma\|^2$ for Lasso or Group Lasso penalties for $\Omega_1(\Gamma)$, $\|\cdot\|$ is denoted as the Frobenius norm. In general, we assume that $\bar{\mathcal{L}}(\cdot)$ is differentiable with respect to W; in practice, the gradient in Eq. (3a) is understood as sub-gradient and Eq. (3a) thus becomes an inclusion.

In the inverse scale space [28], important features/weights will be selected at a faster rate. During the exploration of inverse scale space, the augmented variables Γ evolve from sparse to dense. It can be viewed as a weight selection path from simple model to complex model. The augmented variables Γ are initialized as 0, which means that none of the corresponding weights are selected. Then during the training process, some of them become non-zero which indicates they are selected via exploring the inverse scale space. Meanwhile the model parameters W are also optimized with corresponding loss function to obtain prediction ability on specific task.

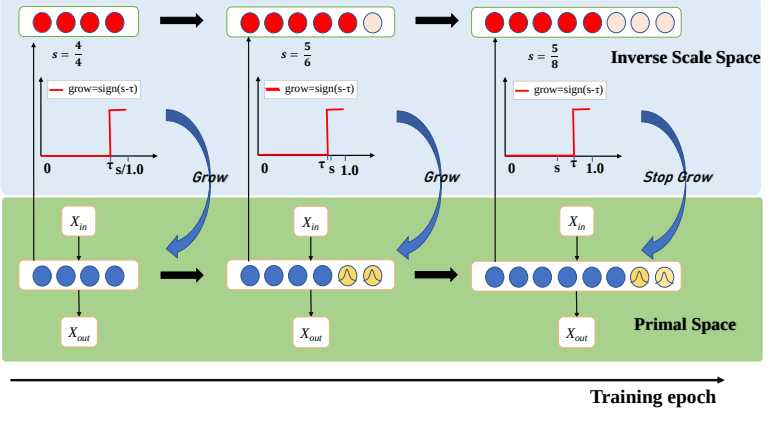

Figure 2: Illustration of our AdeNeL: initialized by a backbone with a small number of filters, AdeNeL learns both the structures and parameters from data, by periodically verifying the network capacity every J epochs. The capacity of each layer is verified by the support set learned in the inverse scale space.

Solving DI-ISS by LBI. The dynamics of Eqs. (3a) - (3a) can be rewritten as the standard Linearized Bregman Iteration (LBI), resulting in the problem:

$$P_{t+1} = \operatorname*{argmin}_P \left\{ \langle P - P_t, \alpha\nabla\bar{\mathcal{L}}(P_t)\rangle + B_\Psi^{p_t}(P, P_t) \right\}, \tag{4}$$

where $P := (W, \Gamma)$, $p_t \in \partial\Psi(P_t)$ and $\Psi(P) = \Omega_1(\Gamma) + \|W\|^2 + \|\Gamma\|^2$. We denote $B_\Psi^q(\cdot)$ as the Bregman divergence associated with convex function $\Psi(\cdot)$ and for some $q \in \partial\Psi(Q)$, we have $B_\Psi^q(P, Q) := \Psi(P) - \Psi(Q) - \langle q, P - Q\rangle$.

Solutions. W explores the parameters of models in primal space by gradient descent, while support set Γ explores important sub-network architectures in the inverse scale space, where those important parameters become nonzero faster than the others. A numerical solution (as Euler forward discretization) of DI-ISS can be given by

$$W_{t+1} = W_t - \alpha \nabla_W \bar{\mathcal{L}}(W_t, \Gamma_t), \tag{5}$$

$$Z_{t+1} = Z_t - \alpha \nabla_\Gamma \bar{\mathcal{L}}(W_t, \Gamma_t), \tag{6}$$

$$\Gamma_{t+1} = \mathrm{Prox}(Z_{t+1}), \tag{7}$$

where the initialization $Z_0 = \Gamma_0 = \mathbf{0}$; W_0 is initialized as [11] and α is the learning rate . $\bar{\mathcal{L}}(W_t, \Gamma_t) = \mathcal{L}_{task}(W_t) + \frac{1}{2\nu}\|W_t - \Gamma_t\|^2$ indicates the loss function at t, with task-specific loss $\mathcal{L}_{task}(W_t)$ (*e.g.*, cross-entropy loss). Γ is learned to approximate W here. $\mathrm{Prox}(\cdot)$ is the proximal mapping function with the following form,

$$\mathrm{Prox}(Z) = \min\left(0, \lambda - \frac{1}{\|Z\|_{1,2}}\right) Z \quad (\lambda \geq 0), \tag{8}$$

where $\|Z\|_{1,2}$ is a group Lasso (ℓ_1-ℓ_2) norm for convolutional filters or simply the Lasso (ℓ_1) norm for and λ is a constant. Note that, a pilot work [15] explored the sparsity of signal in linear transformation, *i.e.*, $\mathbf{y} = XW$.

One can see that Eq. (5) is essentially a gradient descent step over the primal parameter W_t. However, in Eqs. (6)-(7), DI-ISS extends the primal network parameters W, to a coupled parameter set, (W, Γ), where a sparse proximal gradient descent runs over the dual structural sparsity parameter Γ enforcing structural sparsity on network models.

3.2 Adaptive End-to-End Budgeted Network Learning

Algorithm Overview. Using the solution of DI-ISS by LBI, we propose the AdeNeL algorithm which grows the networks from small to proper size. As shown in Figure 2, AdeNeL has the following key steps:
(1) Seed network initialization. Given a network backbone, *e.g.*, VGG16, we set the number of filters in each convolutional layer as a small number (*e.g.*, 4). This slim network is the seed network in the later growing and training process.
(2) Growing in training. During training process, we periodically check the capacity of each convolutional layer using the measurement method in Section 3.2.1 and add new filters to those lack of capacity. The growing process stops when it meets parameters/FLOPs budgets.
(3) Fine-tuning. After the growing process stops, we adjust the learning rate to train the learned networks for extra epochs.

3.2.1 Capacity in Layer Level

Along the training path of Eqs. (5)-(8), the Γ will gradually becomes non-zeros and finally converges. Due to the Eq. 8, the Γ will converge to a sparse approximation of parameter set W. And if the Γ is too sparse, it means only small parts of Γ can represent the information of the network weights W. This indicates that the weights W is redundant and the network capacity is currently sufficient. On the contrary, if the Γ is compact, the network is currently lack of capacity and may need to add new weights to improve its performance. Formally, we use s^l as the redundancy of the l-th layer of a network: $s^l = |\Gamma^l|/|W^l|$, where $|\Gamma^l|$ is the

non-zero item of Γ in the l-th layer and $|W^l|$ is the number of filters in the l-th layer. We introduce a threshold τ to control sign of the growing and if

$$s^l > \tau, \tag{9}$$

it means the l-th layer is lack of capacity and we can add some new filters into this layer. Otherwise this layer has sufficient capacity and is not need to be grown.

3.2.2 Growing Network in Training

	Methods	Acc.(%)	Params(M)	FLOPs(G)	Train-Cost
VGG-16	Baseline	92.90	14.99 (100.0%)	0.313 (100.0%)	100.0%
	$l1$-Pruning [19]	91.80	2.98 (19.9%)	0.062 (19.9%)	25.0%
	SoftNet [13]	92.10	5.40 (36.0%)	0.113 (36.1%)	62.5%
	ThiNet [23]	90.80	5.40 (36.0%)	0.113 (36.1%)	62.5%
	Provable [20]	92.40	0.85 (5.0%)	0.047 (15.0%)	28.6%
	Grow&Prune [42]	92.50	0.75 (5.0%)	0.042 (13.5%)	20.2%
	Ours	**92.71**	**0.58 (4.1%)**	**0.042 (13.5%)**	**10.2%**
ResNet-20	Baseline	91.30	0.27 (100.0%)	0.041 (100.0%)	100.0%
	l_1-prune [19]	90.90	0.15 (55.6%)	0.023 (55.4%)	90.9%
	SoftNet [13]	90.80	0.14 (53.6%)	0.021 (50.6%)	83.3%
	ThiNet [23]	89.20	0.18 (67.1%)	0.028 (67.3%)	90.0%
	Provable [20]	90.80	0.10 (37.3%)	0.022 (54.5%)	58.8%
	Grow&Prune [42]	90.90	**0.096 (35.8%)**	0.021 (50.2%)	**41.6%**
	Ours	**91.21**	0.12 (44.0%)	**0.021 (50.0%)**	60.0%

Table 1: Results of network slimming of AdeNeL and other methods on CIFAR10. We evaluate the prediction quality by Top-1 accuracy (%) (↑), memory cost by parameters (M) (↓), computational cost by FLOPs (G) (↓) and efficiency by train-cost (%) (↓).

Methods	Acc@Top-1(%)	Acc@Top-5	Params(M)	FLOPs(G)	Train-Cost
Baseline	69.15	88.87	11.80	1.80	100.0%
l_1-prune [19]	67.57	87.50	10.33	1.67	100.0%
MIL [7]	66.33	86.94	-	1.18	-
Ours	68.70	88.65	10.44	1.57	78.7%

Table 2: Results of network slimming of AdeNeL and other methods on ImageNet. We evaluate the prediction quality by Top-1 and Top-5 accuracy (%) (↑), memory cost by parameters (M) (↓), computational cost by FLOPs (G) ↓ and efficiency by train-cost (%) (↓).

Growing filters in training. When the seed network starts training, we periodically verify the growing indicator s after training every J epochs. If Eq. (9) holds for some convolutional layers, our algorithm preserves the filters learned before and adds randomly initialized new filters to these layers and continues the training.

Growing scale. For simplicity, we add the fixed number of new filters to the current models at each growing round. Indeed, there can be other strategies and we will discuss the effectiveness of this strategy in the supplement materials.

Stop criterion. Our AdeNeL is stopped once the network has enough capacity for the dataset. As no filters added in growing round or meet the parameters/FLOPs budgets, the

AdeNeL will stop growing process and train extra few epochs with a smaller learning rate to finish the training process.

	Methods	CIFAR10	CIFAR100	Params(M)	FLOPs(G)	Train-Cost
Width	ResNet-32-3x [34]	94.81	76.30	4.16	0.619	100.0%
	Ours	**95.07**	-	2.49	0.569	55.8%
	Ours	-	**77.33**	4.39	0.460	66.7%
	ResNet-32-4x [34]	95.40	77.80	7.39	1.099	100.0%
	Ours	**95.50**	-	5.67	0.861	66.4%
	Ours	-	**77.95**	7.44	0.607	90.0%
	ResNet-32-5x [34]	95.20	78.34	11.54	1.716	100.0%
	Ours	**95.50**	-	4.47	1.197	49.4%
	Ours	-	**78.97**	6.10	0.916	46.6%
Depth	Baseline	94.30	-	4.06	0.597	100.0%
	AutoGrow [37]	94.27	-	4.06	0.597	142.3%
	Ours	**95.07**	-	2.49	0.569	57.8%
	Baseline	-	75.67	5.14	0.759	100.0%
	AutoGrow [37]	-	74.72	5.14	0.759	159.3%
	Ours	-	**77.33**	4.39	0.460	54.6%

Table 3: Comparison between AdeNeL and network width expansion [34] and depth growing methods [37, 42] on CIFAR10/100. We evaluate the prediction quality by Top-1 accuracy (%) (↑), computational cost by FLOPs (G) (↓) and efficiency by train-cost (%) (↓).

4 Experiments

Datasets and Backbones. We conduct experiments on several benchmark classification datasets, including MNIST [18], CIFAR10/ CIFAR100 [17] and ImageNet-2012 [5]. The standard backbones are utilized for each dataset, including, LeNet-5 for MNIST, ResNet and VGG family for CIFAR10/CIFAR100, ImageNet-2012 respectively.

Evaluation Metrics. Several evaluation metrics are adopted here. We have Top-1 (Acc@Top-1) and Top-5 (Acc@Top-5) classification accuracy. The total number of network Parameters, i.e., Params. (M), reflecting the network size. FLOPs (G), the computational complexity of a network. Train-Cost (%) is the accumulation of FLOPs along the training epochs. Here we report the percentage of the train-cost of each method comparing to baseline model, showing the training efficiency of each method.

Experimental Setup. The filter number is initially set as 4/8/16 (according to settings of different experiments) for each convolutional layer of each model with the weight initialization [11]. Our AdeNeL experiments use the optimizer we implement for DI-ISS and added new filters are initialized as in [11]. For comparison, the baselines are trained by the settings from the good practices, including SGD optimizer, mini-batch size (128), initial learning rate (0.1), learning rate decay (0.1 at every 1/3 total epochs), number of epochs (300 for CIFAR10/100 and 90 for ImageNet) and weight decay (0.0005).

4.1 Results

Results of Network Slimming. On CIFAR10, our AdeNeL is compared against baselines models, several state-of-the-art pruning methods [13, 19, 20, 21, 22] and growing & pruning

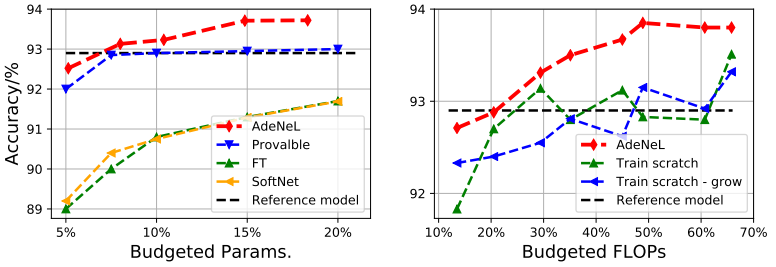

Figure 3: The accuracy of the models learned by AdeNeL under various FLOPs/parameters budgets. The experiments are conducted on CIFAR10 using VGG16 backbone. Left: Comparison between AdeNeL and other pruning methods. Right: The results of AdeNeL and the networks found by AdeNeL trained from scratch.

Model	Methods	Acc. (%)	Params. (M)	FLOPs (G)	Train-Cost
VGG-16	Train Scratch - Base	91.83	0.589	0.042	100.0%
	Train Scratch - Grow	92.33	0.589	0.042	54.5%
	Ours	**92.71**	0.589	0.042	54.5%
ResNet-32	Train Scratch - Base	94.64	4.47	1.197	100.0%
	Train Scratch - Grow	94.87	4.47	1.197	70.6%
	Ours	**95.50**	4.47	1.197	70.6%

Table 4: Ablation study: we train the network learned by AdeNeL from scratch with ordinary and growing manner with SGD optimizer.

method [42]. Table 1 and Table 2 summarize the results of these methods and our AdeNeL. One can see that: (1) The prediction quality of AdeNeL is the best of all, and even comparable to full-sized baseline models. (2) The parameters and FLOPs of models learned by our method both small, suggesting that our AdeNeL indeed learns better channel configurations. (3) The train costs of our AdeNeL are still comparable or even smaller than competitors, and saves a lot comparing to training baselines, showing the efficiency of our method. (4) On large-scale dataset ImageNet-2012, our method still achieve better performance and using significantly fewer train costs. In conclusion, These results shows that our AdeNeL can learn good prediction quality model under the extreme small parameters/FLOPs budgets.

We also conduct experiments under different FLOPs/parameters targets, the results are in Figure 3. One can see that only using 20% FLOPs and 7.5% parameters of the full model, AdeNeL can retain the accuracy of the full model. And given more budgets, AdeNeL can achieve remarkable better performance than full model, showing the superiority of the proposed method. We also compare AdeNeL with other pruning methods (left in Figure 3). One can see that AdeNeL achieves better performance under different parameter budgets.

AdeNeL vs Network Width Expansion & Depth Growing. Width and depth are two important dimensions of networks and it is verified that expanding width/depth is a simple but efficient way to improve the performance of networks. Thus, in this part we also compare our AdeNeL with network width expansion [34] and depth growing [37, 42] methods. Table 3 summarizes the results. Comparing to directly expanding network width by times, our growing strategy can learn models with better prediction quality, fewer parameters and FLOPs and training costs. This suggests that our AdeNeL learns better channel configurations. Table 3 also shows that under the parameters and FLOPs budgets of models grown by

depth, our width growing method achieves better performance.

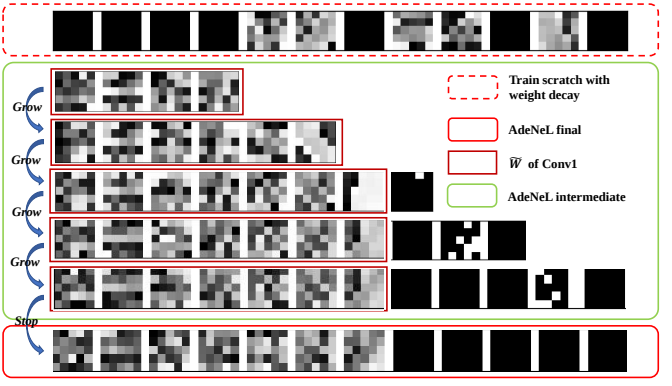

Figure 4: Visualization of filters of the first convolutional layer of model trained from scratch and model during the growing process of AdeNeL.

4.2 Analysis

Ablation Study. To verify the effectiveness of our growing strategy, we train the learned networks from scratch, *i.e.*, by ordinary training process using SGD optimizer (Train-Scratch-Base) and growing in the training process with SGD optimizer (Train-Scratch-Grow). The results in Figure 3 (right) and Table 4 show that models trained from scratch or repeated the growing process with SGD optimizer perform significantly worse than the model trained by our method. This confirms the superiority and effectiveness of our proposed method.
Visualization of AdeNeL. We also visualize the training process of AdeNeL. Specifically, we use AdeNeL to train a seed LeNet-5 model with 4 filters in each convolutional layer and we add 2 filters to the layers need to grow at each round. Filters learn by first convolutional layer at each growing round in Figure 4. Rows in the green box are filters of the first convolutional layers of the model in each growing round. The filters in the red box are $\widetilde{W}$, which indicates that AdeNeL can indeed correctly select important filters from W.

5 Conclusion

This paper studies the novel task of adaptive end-to-end budgeted network learning and the proposed algorithm which simultaneously grows and trains filters. Experiments on VGG and ResNet show that the proposed algorithms can efficiently grow networks with fixed depth and a small number of filters in each layer. Our algorithms achieve classification accuracy on par with those of big network models, yet with remarkable economic training cost. Note that our work is that currently we only focus on exploring network width configuration; and a potential future work is to explore both width and depth configuration of the network.

6 Acknowledgements

This work was in part supported by Science and Technology Commission of Shanghai Municipality Project (#19511120700), and Major Project (No.2021SHZDZX0103).

References

[1] Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.

[2] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for fast test-time prediction. *arXiv preprint arXiv:1702.07811*, 2017.

[3] Han Cai, Jiacheng Yang, Weinan Zhang, Song Han, and Yong Yu. Path-level network transformation for efficient architecture search. In *International Conference on Machine Learning*, pages 678–687. PMLR, 2018.

[4] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[5] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[6] Xuanyi Dong and Yi Yang. Network pruning via transformable architecture search. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 760–771, 2019.

[7] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5840–5848, 2017.

[8] Yanwei Fu, Chen Liu, Donghao Li, Xinwei Sun, Jinshan Zeng, and Yuan Yao. Dessilbi: Exploring structural sparsity of deep networks via differential inclusion paths. In *International Conference on Machine Learning*, pages 3315–3326. PMLR, 2020.

[9] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[10] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2234–2240, 2018.

[14] Chendi Huang, Xinwei Sun, Jiechao Xiong, and Yuan Yao. Split lbi: An iterative regularization path with structural sparsity. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3377–3385, 2016.

[15] Chendi Huang, Xinwei Sun, Jiechao Xiong, and Yuan Yao. Boosting with structural sparsity: A differential inclusion approach. *Applied and Computational Harmonic Analysis*, 48(1):1–45, 2020.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2016.

[20] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. In *International Conference on Learning Representations*, 2019.

[21] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017.

[22] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[23] Mason McGill and Pietro Perona. Deciding how to decide: Dynamic routing in artificial neural networks. In *International Conference on Machine Learning*, pages 2363–2372. PMLR, 2017.

[24] P Molchanov, S Tyree, T Karras, T Aila, and J Kautz. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings*, 2019.

[25] Arkadij Semenovič Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

[26] Augustus Odena, Dieterich Lawson, and Christopher Olah. Changing model behavior at test-time using reinforcement learning. *arXiv preprint arXiv:1702.07780*, 2017.

[27] Stanley Osher, Martin Burger, Donald Goldfarb, Jinjun Xu, and Wotao Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 4(2):460–489, 2005.

[28] Stanley Osher, Feng Ruan, Jiechao Xiong, Yuan Yao, and Wotao Yin. Sparse recovery via differential inclusions. *Applied and Computational Harmonic Analysis*, 41(2):436–469, 2016.

[29] Stanley Osher, Feng Ruan, Jiechao Xiong, Yuan Yao, and Wotao Yin. Sparse recovery via differential inclusions. *Applied and Computational Harmonic Analysis*, 41(2):436–469, 2016.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[31] Xiu Su, Shan You, Tao Huang, Fei Wang, Chen Qian, Changshui Zhang, and Chang Xu. Locally free weight sharing for network width search. In *International Conference on Learning Representations*, 2020.

[32] Xinwei Sun, Lingjing Hu, Yuan Yao, and Yizhou Wang. Gsplit lbi: Taming the procedural bias in neuroimaging for disease prediction. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 107–115. Springer, 2017.

[33] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

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

[35] Tom Veniat and Ludovic Denoyer. Learning time/memory-efficient deep architectures with budgeted super networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3492–3500, 2018.

[36] Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *International Conference on Machine Learning*, pages 564–572, 2016.

[37] Wei Wen, Feng Yan, Yiran Chen, and Hai Li. Autogrow: Automatic layer growing in deep convolutional networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 833–841, 2020.

[38] Lemeng Wu, Dilin Wang, and Qiang Liu. Splitting steepest descent for growing neural architectures. In *Advances in Neural Information Processing Systems*, pages 10656–10666, 2019.

[39] Mao Ye, Chengyue Gong, Lizhen Nie, Denny Zhou, Adam Klivans, and Qiang Liu. Good subnetworks provably exist: Pruning via greedy forward selection. In *International Conference on Machine Learning*, pages 10820–10830. PMLR, 2020.

[40] Wotao Yin, Stanley Osher, Donald Goldfarb, and Jerome Darbon. Bregman iterative algorithms for \ell_1-minimization with applications to compressed sensing. *SIAM Journal on Imaging sciences*, 1(1):143–168, 2008.

[41] Jiahui Yu and Thomas Huang. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 2019.

[42] Xin Yuan, Pedro Henrique Pamplona Savarese, and Michael Maire. Growing efficient deep networks by structured continuous sparsification. In *International Conference on Learning Representations*, 2020.

[43] Bo Zhao, Xinwei Sun, Yanwei Fu, Yuan Yao, and Yizhou Wang. Msplit lbi: Realizing feature selection and dense estimation simultaneously in few-shot and zero-shot learning. In *International Conference on Machine Learning*, pages 5912–5921. PMLR, 2018.

[44] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2423–2432, 2018.

[45] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2016.

[46] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.