SEDONA: Search for Decoupled Neural Networks toward Greedy Block-wise Learning

ICLR 2021

Myeongjang Pyeon, Jihwan Moon, Taeyoung Hahn, and Gunhee Kim

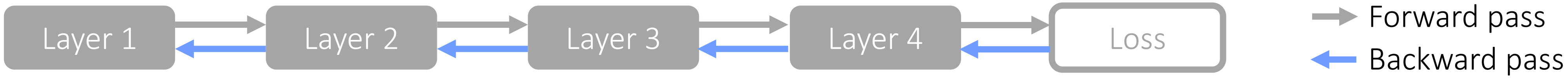
SEOUL NATIONAL UNIV.
VISION & LEARNING

I. Greedy Block-wise Learning

Inefficiency in Backpropagation

Limited concurrency due to the sequential nature

Each layer must wait for upper layers $\longrightarrow$ Backward locking & update locking[1]



[1] Jaderberg et al. 2017. Decoupled neural interfaces using synthetic gradients. *ICML*

Greedy Block-wise Learning

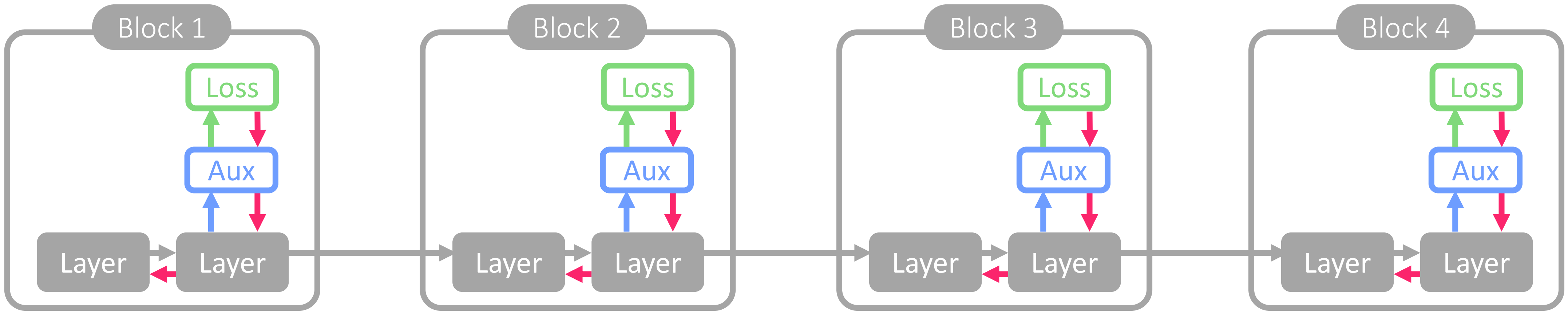
A Competitive Alternative

Layers are grouped into blocks

Each block has its lightweight auxiliary network to compute its own local loss

→ Enable asynchronous training by isolating gradients

→ Partially solve backward locking & update locking



Greedy Block-wise Learning

A Competitive Alternative

Known to be better than other backprop alternatives [1] but…

It still yields a worse performance than end-to-end backprop

(ImageNet error rates of ResNet-152 : SOTA greedy - 25.5%[2], Backprop - 21.7%[3])

Can we do better?

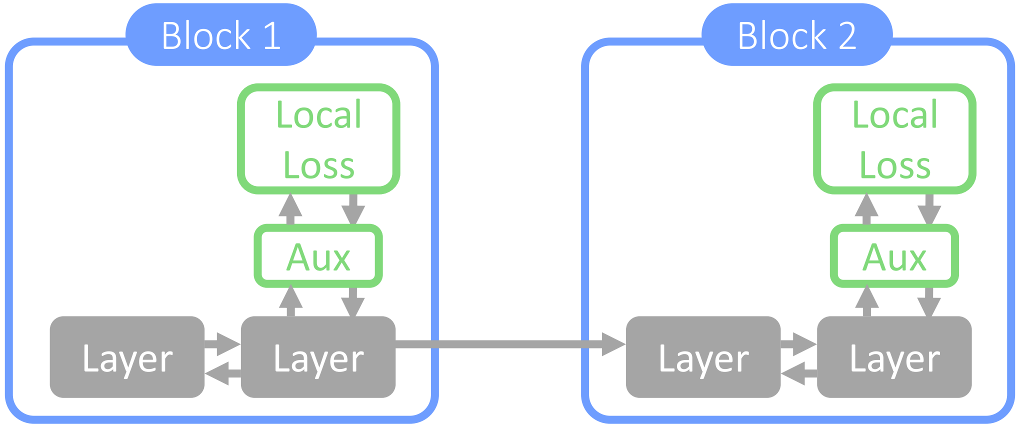
[1] Beliovsky et al. 2019. Greedy layerwise learning can scale to ImageNet. *ICML*

[2] Belilovsky et al. 2020. Decoupled greedy learning of CNNs. *ICML*

[3] ImageNet classification results with pretrained ResNets (1-crop). https://pytorch.org/hub/pytorch_vision_resnet

Problem Statement

For greedy-block-wise learning, one must make two design decisions:
1) How to group layers into blocks and 2) How to design auxiliary networks



The two design decisions are critical to the performance

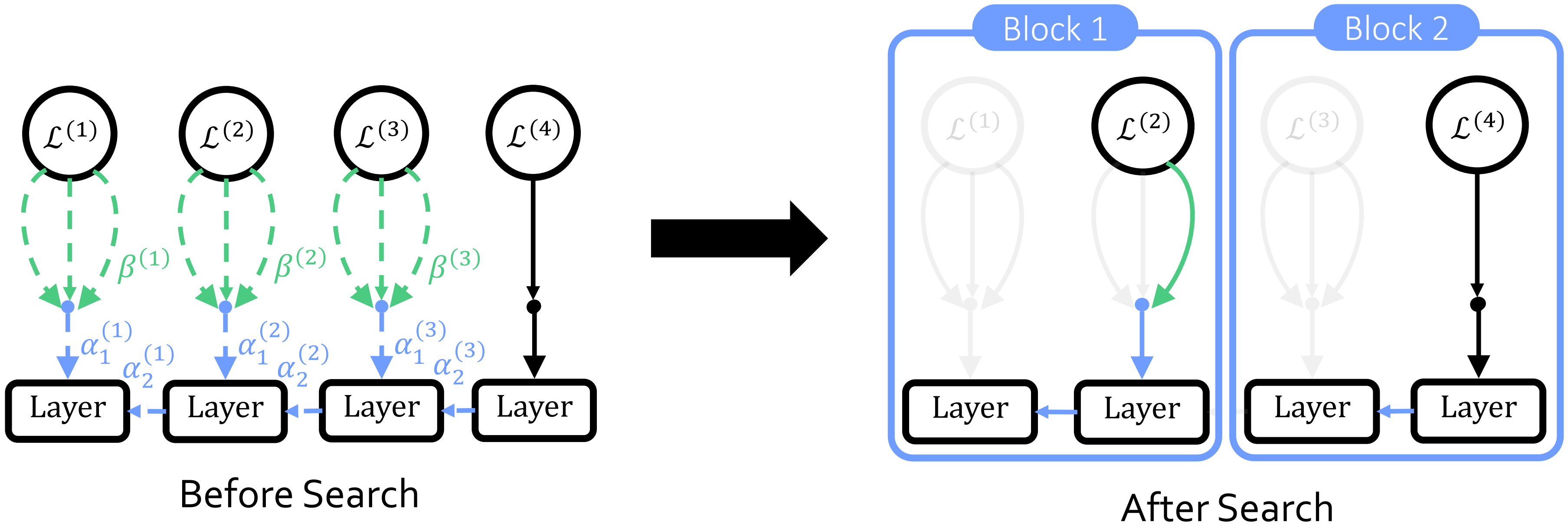
Our approach: Automating the discovery of the best configuration

II. SEDONA: SEarching for DecOupled Neural Architectures

Key Ideas

1. Formulating greedy block-wise learning with decision variables
2. Optimizing decision variables by gradient descent

   for efficiency & reusability



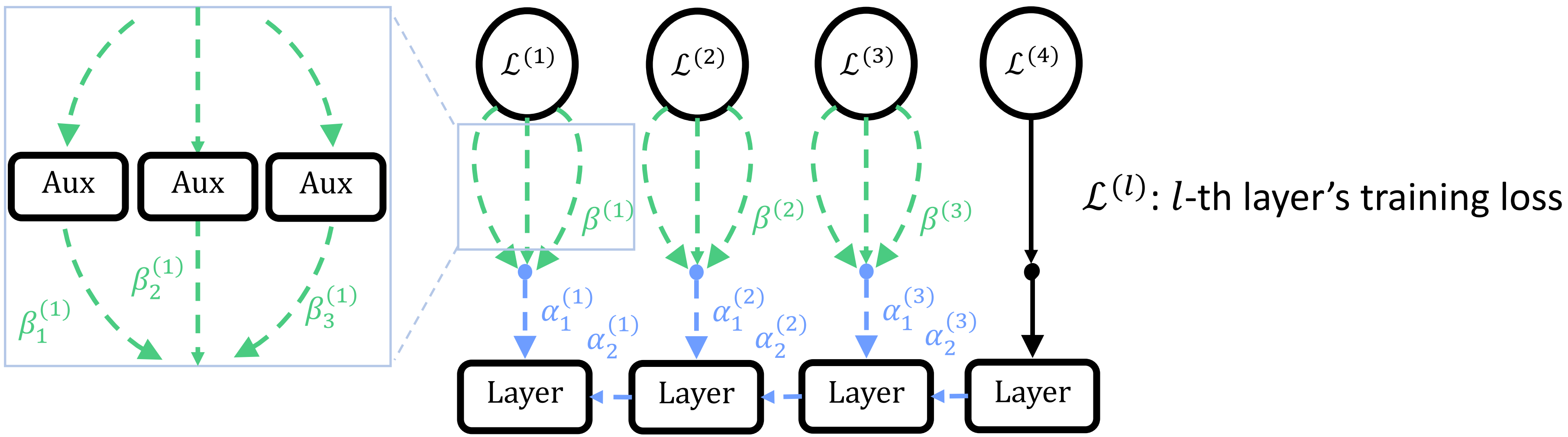
$\mathcal{L}^{(l)}$: l-th layer's training loss

Decision Variables

How to group layers into blocks & How to design auxiliary networks

Equivalent to making two types of decisions *at every (non-last) layer*

1. Whether to use local or backpropagated gradients
   $\longrightarrow$ Signal variable $\alpha^{(l)} \in \{0,1\}^2$
2. Which auxiliary network to use among M candidates
   $\longrightarrow$ Auxiliary type variable $\beta^{(l)} \in \{0,1\}^M$



Decision Variables

How to group layers into blocks & How to design auxiliary networks

Equivalent to making two types of decisions *at every (non-last) layer*

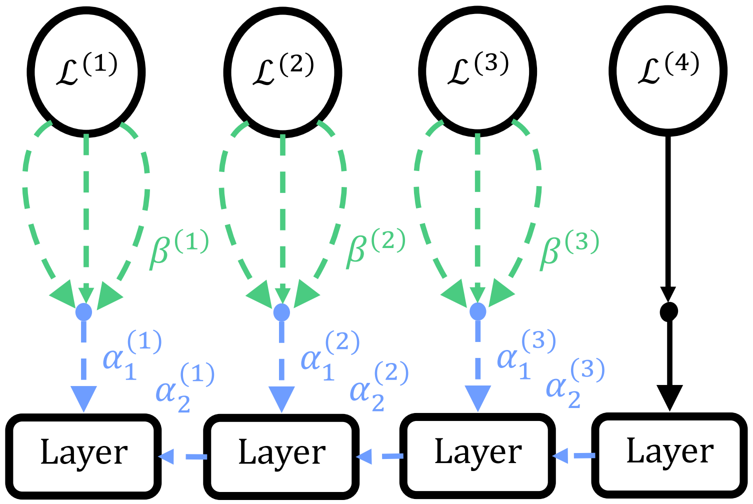
1. Whether to use local or backpropagated gradients
   $\rightarrow$ Signal variable $\alpha^{(l)} \in \{0,1\}^2$
2. Which auxiliary network to use
   $\rightarrow$ Auxiliary type variable $\beta^{(l)} \in \{0,1\}^M$ (M aux. candidates)

$\alpha^{(1)} = (0, 1)$ → Use backproped gradients at layer 1 & 3

$\alpha^{(2)} = (1, 0)$ → Use local gradients at layer 2

$\alpha^{(3)} = (0, 1)$

$\beta^{(2)} = (0, 0, 1)$ → Use 3rd aux net at layer 2



Decision Variables

How to group layers into blocks & How to design auxiliary networks

Equivalent to making two types of decisions *at every (non-last) layer*

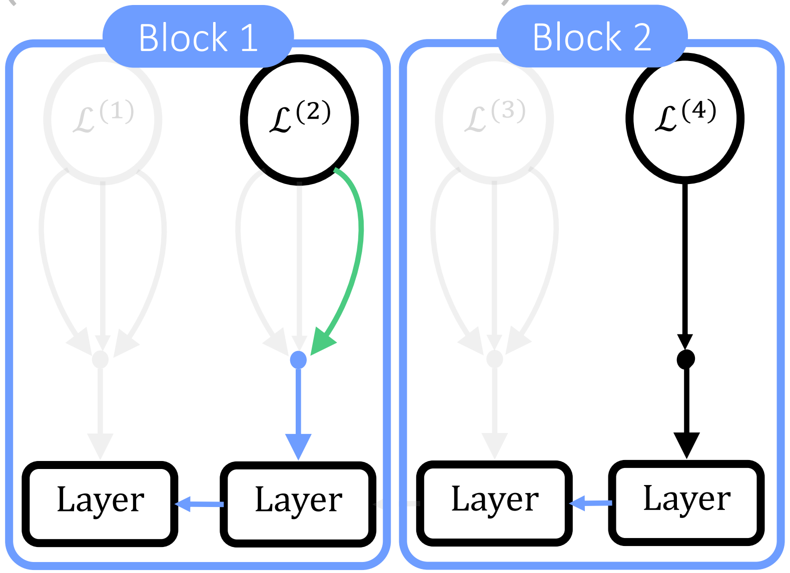
1. Whether to use local or backpropagated gradients
   $\longrightarrow$ Signal variable $\alpha^{(l)} \in \{0,1\}^2$
2. Which auxiliary network to use
   $\longrightarrow$ Auxiliary type variable $\beta^{(l)} \in \{0,1\}^M$ (M aux. candidates)

$\alpha^{(1)} = (0, 1)$ → Use backproped gradients at layer 1 & 3

$\alpha^{(2)} = (1, 0)$ → Use local gradients at layer 2

$\alpha^{(3)} = (0, 1)$

$\beta^{(2)} = (0, 0, 1)$ → Use 3rd aux net at layer 2



Optimizing Decision Variables

3 steps

1. Continuous relaxation

   Discrete domain → continuous domain
2. Bilevel optimization

   Optimize relaxed decision variables
3. Discretization

   Continuous domain → discrete domain

Optimizing Decision Variables

3 steps

1. Continuous relaxation

   Discrete domain ⟶ continuous domain

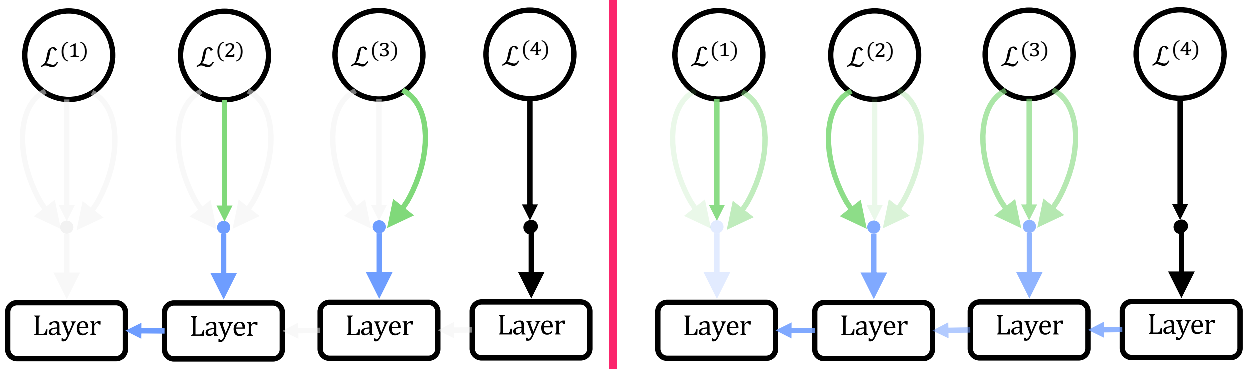
2. Bilevel optimization

   Optimize relaxed decision variables

3. Discretization

   Continuous domain ⟶ discrete domain

Discrete domain	Continuous domain



Non-differentiable	Differentiable ⟶ We can optimize α, β by gradient descent

Optimizing Decision Variables

3 steps

1. Continuous relaxation

   Discrete domain ⟶ continuous domain

2. Bilevel optimization

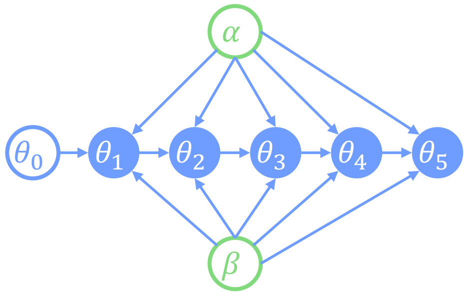
   Optimize relaxed decision variables

3. Discretization

   Continuous domain ⟶ discrete domain

- Inner-level optimization

  Perform gradient steps on layer weights θ



- Outer-level optimization

Optimizing Decision Variables

3 steps

1. Continuous relaxation

   Discrete domain ⟶ continuous domain

2. Bilevel optimization

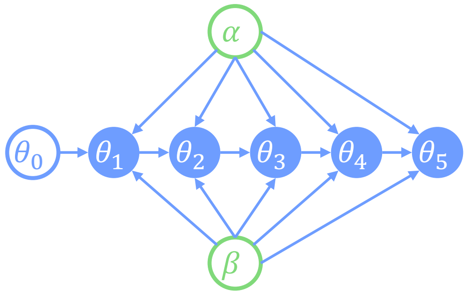
   Optimize relaxed decision variables

3. Discretization

   Continuous domain ⟶ discrete domain

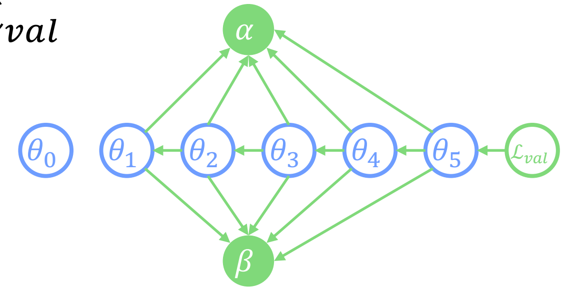
- Inner-level optimization

  Perform gradient steps on layer weights θ



- Outer-level optimization

  Update decision variables α, β by descending $\nabla_{(\alpha,\beta)}\mathcal{L}_{val}$



Optimizing Decision Variables

3 steps

1. Continuous relaxation

   Discrete domain → continuous domain

2. Bilevel optimization

   Optimize relaxed decision variables

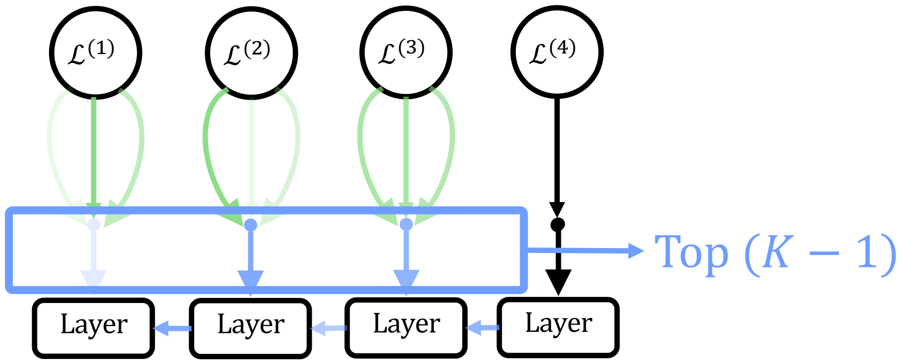
3. Discretization

   Continuous domain → discrete domain

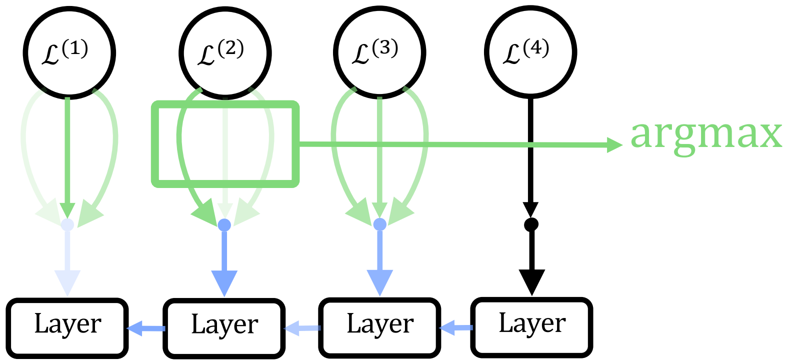
α & β are reusable for any desired K

K: # blocks (desired level of parallelism)

- Grouping layers into K blocks



- Selecting auxiliary networks



Evaluation Results

Better generalization than backprop & other greedy-learning methods

- K: # blocks

Error Rates (%) on **CIFAR-10** (1st, 2nd)

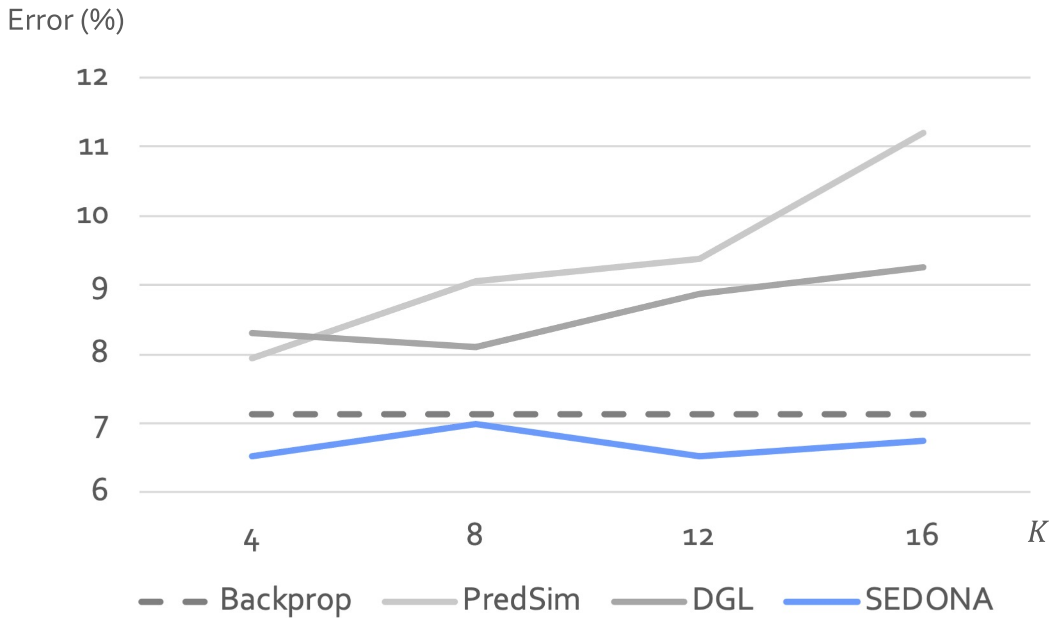
Architecture	Backprop	PredSim[1] ($K = 4$)	DGL[2] ($K = 4$)	SEDONA ($K = 4$)
VGG-19	12.31	13.87	12.19	11.58
ResNet-50	7.99	8.93	8.27	7.53
ResNet-101	7.14	7.93	8.30	6.59
ResNet-152	6.35	7.41	7.39	6.13

[1] Nøkland & Eidnes. 2019. Training neural networks with local error signals. *ICML*
[2] Belilovsky et al. 2020. Decoupled greedy learning of CNNs. *ICML*

Evaluation Results

Maintain the good performance even when # blocks increases

- K: # blocks



ResNet-101, **CIFAR-10**

[1] Nøkland & Eidnes. 2019. Training neural networks with local error signals. *ICML*
[2] Belilovsky et al. 2020. Decoupled greedy learning of CNNs. *ICML*

Evaluation Results

Decouplings found on CIFAR-10 can be transferable to ImageNet

- K: # blocks

ResNet-101, **ImageNet** (1st, 2nd)

Method	K	**Top-1 Err (%)**	**Top-1 Err (%)**	**Training Speedup**
Backprop	1	21.34	5.86	1
DGL[1]	2	21.53	5.84	1.42
	4	23.13	6.82	1.92
SEDONA	2	20.72	5.39	1.67
	4	21.32	5.83	2.01

ResNet-152, **ImageNet** (1st, 2nd)

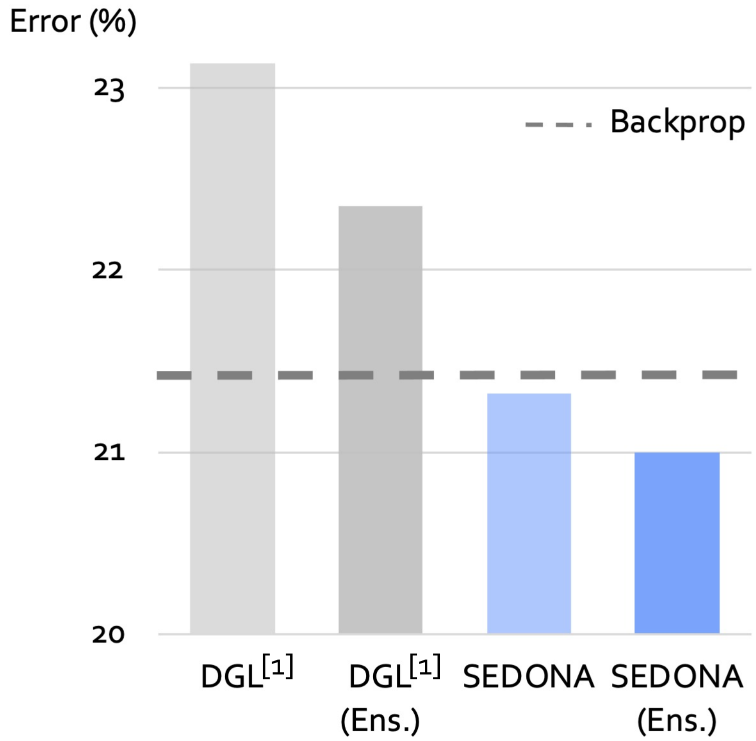
Method	K	**Top-1 Err (%)**	**Top-1 Err (%)**	**Training Speedup**
Backprop	1	21.22	5.79	1
DGL[1]	2	21.45	5.86	1.51
	4	22.89	6.80	2.23
SEDONA	2	20.69	5.58	1.61
	4	21.09	5.13	2.02

[1] Belilovsky et al. 2020. Decoupled greedy learning of CNNs. *ICML*

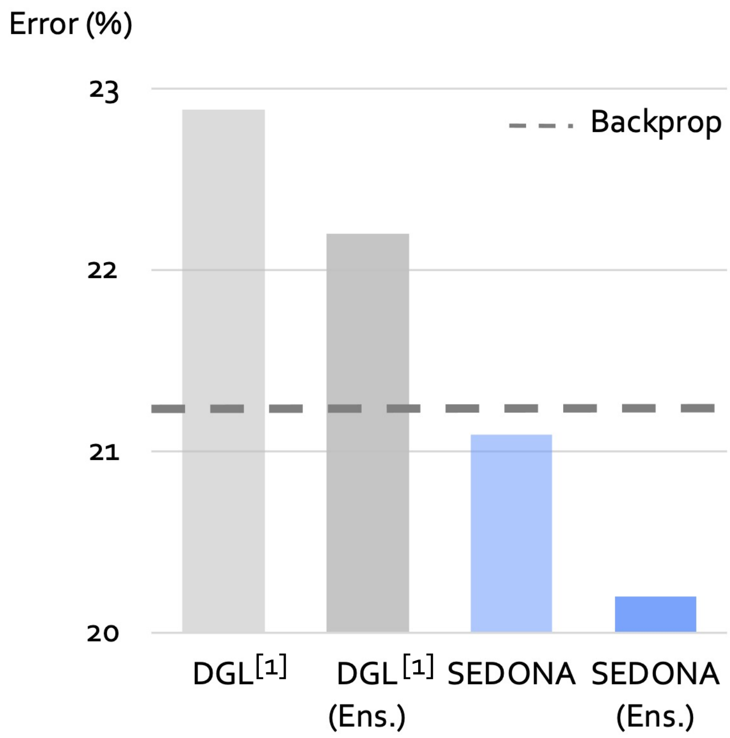
Evaluation Results

Further improvement is possible by ensembling found aux nets

- K: # blocks



ResNet-101, **ImageNet**
$(K = 4)$



ResNet-152, **ImageNet**
$(K = 4)$

[1] Belilovsky et al. 2020. Decoupled greedy learning of CNNs. *ICML*

Discussions

Grouping layers is more important than selecting auxiliary networks

Especially, lower blocks seem to be a key to success of greedy learning

Concluding Remarks

Greedy block-wise learning has several benefits in terms of efficiency

Well-configured greedy learning can outperform end-to-end backprop

SEDONA automatically finds such good configurations

Thank you

Code https://vision.snu.ac.kr/projects/sedona

Paper https://openreview.net/forum?id=XLfdzwNKzch

Contact mjpyeon@vision.snu.ac.kr