DropCov: A Simple yet Effective Method for Improving Deep Architectures

Qilong Wang [1,3], Mingze Gao [1], Zhaolin Zhang [1], Jiangtao Xie [2], Peihua Li [2], Qinghua Hu [1,3]

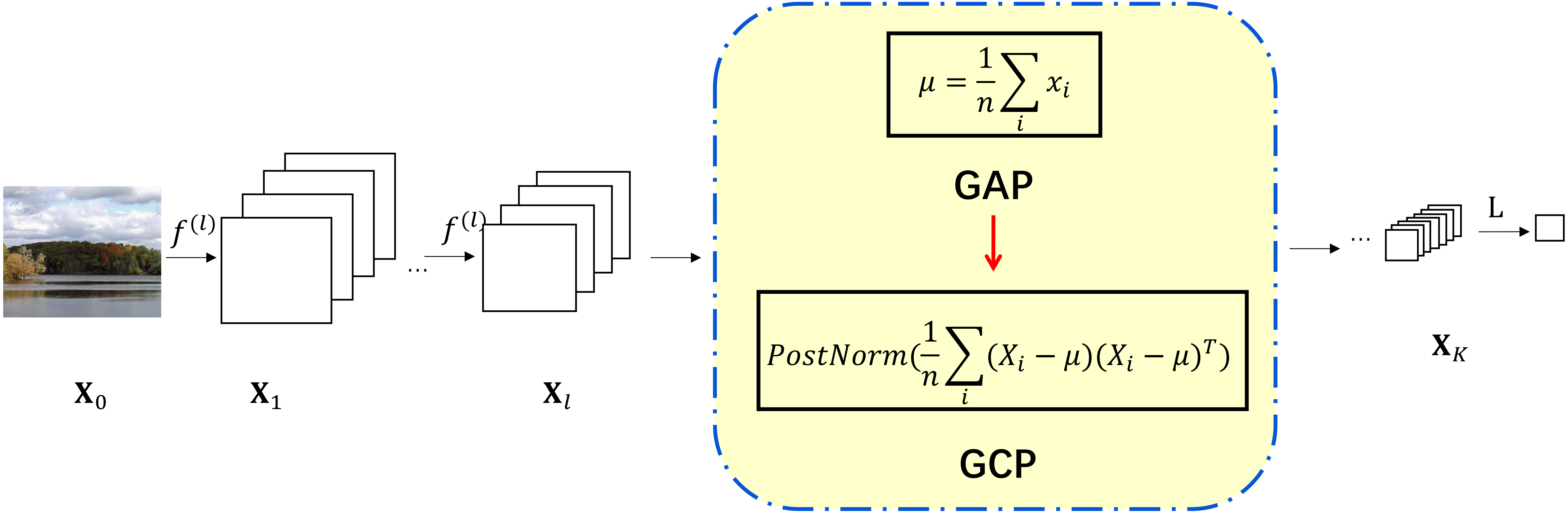
[1] Tianjin University, [2] Dalian University of Technology,
[3] Haihe Laboratory of Information Technology Application Innovation, Tianjin,China

https://github.com/mingzeG/DropCov

NeurIPS 2022

Motivation

Global covariance pooling (GCP) has shown remarkable potential to improve performance of deep architectures in a variety of tasks, especially visual recognition.



Motivation

- One of core differences among those deep GCP methods is post-normalization for covariance representations.
- Behaviors of existing approaches vary significantly, and there is a lack of an intuitive and unified interpretation.
- Another issue of existing post-normalization methods is high computational complexity（$O(d^3)$ or $O(d^2)$）.

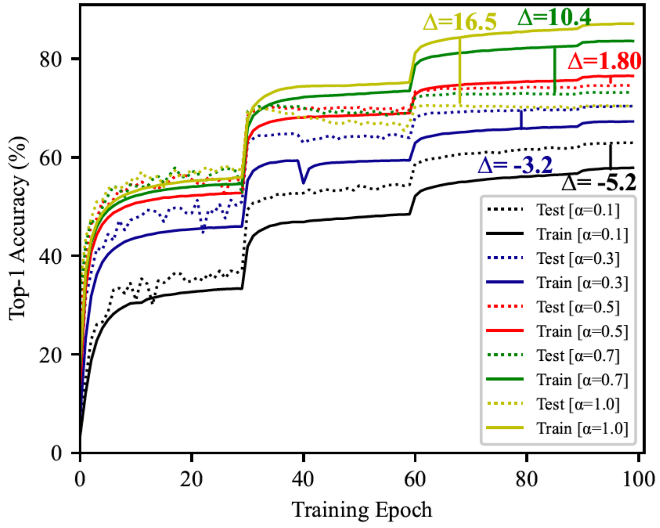
	Method	Formulation	Complexity	Train	Infer.
Element-wise Post-Norm.	B-CNN [31]	$\ell_2(sqrt(\mathrm{V}(\mathbf{X}^T\mathbf{X})))$	$O(d^2)$	✓	✓
	SigmE [25]	$2/(1+e^{-\beta\cdot\mathrm{V}(\mathbf{X}^T\mathbf{X})})-1$	$O(d^2)$	✓	✓
	LN [2]	$\boldsymbol{\beta}\odot(\mathrm{V}(\mathbf{X}^T\mathbf{X})-\mu)/\sigma)\oplus\boldsymbol{\gamma}$	$O(d^2)$	✓	✓
Structure-wise Post-Norm.	DeepO$_2$P [23]	$\mathrm{V}(\mathrm{LogM}(\mathbf{X}^T\mathbf{X}))$	$O(d^3)$	✓	✓
	MPN-COV [29]	$\mathrm{V}((\mathbf{X}^T\mathbf{X})^{\alpha})$	$O(d^3)$	✓	✓
	IB-CNN [30]	$\ell_2(sqrt(\mathrm{V}(\mathbf{X}^T\mathbf{X})^{1/2}))$	$O(d^3)$	✓	✓
	iSQRT-COV [28]	$\mathrm{V}(\approx(\mathbf{X}^T\mathbf{X})^{1/2})$	$O(d^3)$	✓	✓
	MaxExp [25]	$\mathbf{I}-(\mathbf{I}-\mathbf{X}^T\mathbf{X}/(\mathbf{X}^T\mathbf{X}+\varepsilon))^{\alpha}$	$O(d^3)$	✓	✓
Ours	DropCov	$\mathrm{V}(\delta_\rho(\mathbf{X})^T\delta_\rho(\mathbf{X}))$	$O(d)$	✓	×

※: Note β and ε are parameters. $\odot$ and $\oplus$ indicate element-wise multiplication and addition, respectively.

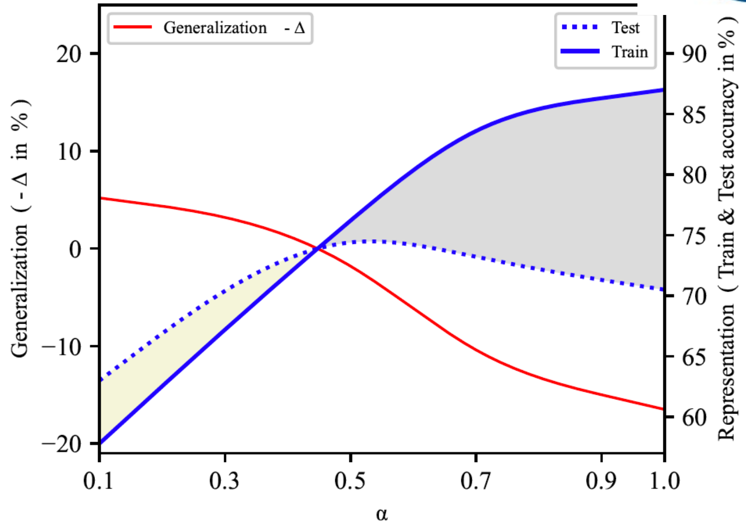
How Does Post-Normalization Impact Deep GCP ?

Using MPN as an example

$$\mathbf{Z} = (\mathbf{X}^T\mathbf{X})^{\alpha} = \mathbf{U}\mathbf{\Lambda}^{\alpha}\mathbf{U}^T$$



(a) Convergence curves of various α



(b) Generalization *vs.* representation ability

$$\begin{cases} \mathbf{Z} \longmapsto \mathbf{I}, \quad \text{if} \quad \alpha \longmapsto 0 & : \text{Representation Decorrelation \& Information Loss}, \\ \mathbf{Z} \longmapsto \mathbf{X}^T\mathbf{X}, \quad \text{if} \quad \alpha \longmapsto 1 & : \text{Information Preservation \& Strong Correlation}. \end{cases}$$

- When $\alpha \rightarrow 0$, Z will be decorrelated, which can help combat over-fitting, which, however, will lead to information loss for X^TX, hurting the representation ability of covariances.
- When $\alpha \rightarrow 1$, information of X^TX will be gradually preserved, maintaining the correlations as characterized in the covariance matrices, which makes training of neural networks difficult (e.g., over-fitting)

Our Finding

Corollary 1. *Effective post-normalization (e.g., matrix power normalization (MPN) with* $0 < \alpha \leq 1$*) can achieve a good trade-off between representation decorrelation and information preservation for GCP, which are crucial to alleviate over-fitting and increase representation ability of deep GCP networks, respectively. Particularly, MPN with* $\alpha = 0.5$ *achieves the best trade-off for* $\alpha \in (0, 1]$ *(without considering other factors), which is proved to be the widely used choice of* α *[29, 30].*

Q: Could this finding be extended to other existing post-normalization methods?

Extension of Existing Post-normalization Methods (Part-I)

From Matrix Power Normalization (MPN) to Adaptive Power Normalization (APN)

by considering effect of inputs :

$$\min_{\alpha}\Big[\underbrace{\alpha(\log(\lambda_{max})-\log(\max(C,\lambda_{min})))}_{\text{representation decorrelation}}\underbrace{-\tau\sum_{i=1}^{K}(\lambda_i^{\alpha}/\sum_{i=1}^{K}\lambda_i^{\alpha})\log(\lambda_i^{\alpha}/\sum_{i=1}^{K}\lambda_i^{\alpha})}_{\text{information preservation}}\Big]$$

Method	$d=64$	$d=128$	$d=256$
MPN ($\alpha=0.5$)	73.1	74.4	74.9
APN (Ours)	73.3	74.5	75.0

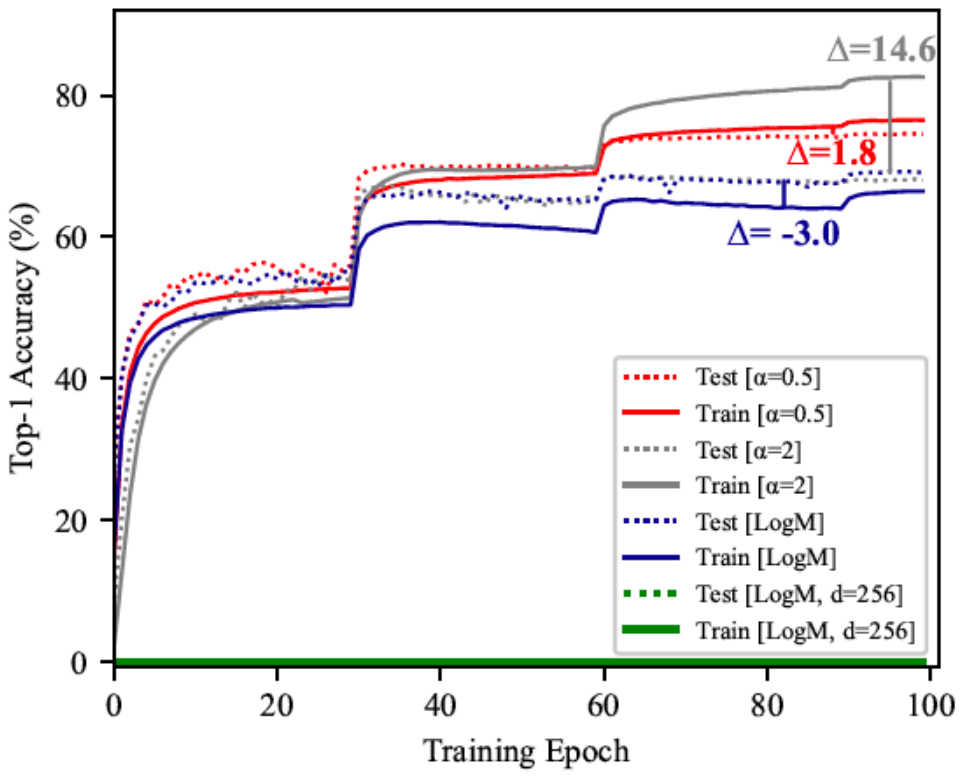
Comparisons (% in Top-1 accuracy) of adaptive normalization methods with the fixed ones using ResNet-18 on ImageNet-1K.

APN brings 0.1% - 0.2% gains over MPN with = 0.5 by considering effect of inputs.

Extension of Existing Post-normalization Methods (Part-II)

MPN with α > 1

If α of MPN is larger than one would violate the principle of representation decorrelation and disrupts equilibrium between eigenvalues.



Matrix Logarithm Normalization (LogM)

$$\log(\lambda_i) \longrightarrow \epsilon \log((\lambda_i + \epsilon)/\epsilon$$

DeepO$_2$p

I-LogM*

Element-wise Normalization (EwN)

$$\ell_2(sqrt(\mathrm{V}(\mathbf{X}^T\mathbf{X}))) \longrightarrow \boldsymbol{\beta} \odot \ell_2(sqrt(\mathrm{V}(\mathbf{X}^T\mathbf{X}))) \oplus \boldsymbol{\gamma}$$

B-CNN

B-CNN+LT*

Extension of Existing Post-normalization Methods (Part-III)

Method	$d = 64$			$d = 256$		
	Top-1 Acc. (%)	Train (μs)	Test (μs)	Top-1 Acc. (%)	Train (μs)	Test (μs)
Plain GCP	71.1	3.45	1.04	70.0	35.08	11.89
B-CNN [31]	38.3	4.31	1.23	41.1	44.97	14.38
B-CNN + LT*	68.3	4.45	1.28	73.2	47.37	15.24
LN [2]	71.7	4.10	1.13	70.2	40.79	13.59
$DeepO_2P$ [23]	70.1	922.50	910.95	*Not Converge*	9731.71	9638.71
I-LogM*	71.2	922.53	910.98	72.0	9732.46	9639.65
MPN-COV [29]	73.1	925.16	913.87	74.9	9735.11	9642.43
iSQRT-COV [28]	73.4	12.35	4.94	75.2	193.65	77.46
IB-CNN [30]	*Not Converge*	13.93	5.16	36.1	202.38	80.55
IB-CNN + LT*	70.0	14.16	5.21	72.8	207.48	82.98

I - LogM * and B - CNN + LT* bring 1.1% and 30% gains for DeepO2P and B-CNN.

Above results clearly verify our finding in Corollary 1.

Proposed DropCov

$$\mathbf{Z} = (\mathbf{X}^T\mathbf{X})^\alpha = \mathbf{U}\mathbf{\Lambda}^\alpha\mathbf{U}^T$$

$$\begin{cases} \mathbf{Z} \longmapsto \mathbf{I}, \quad \text{if} \quad \alpha \longmapsto 0 & : \text{Representation Decorrelation \& Information Loss,} \\ \mathbf{Z} \longmapsto \mathbf{X}^T\mathbf{X}, \quad \text{if} \quad \alpha \longmapsto 1 & : \text{Information Preservation \& Strong Correlation.} \end{cases}$$

ISSUE: Although structure-wise post-norm methods have a satisfying performance, they suffer from high computational complexity.

$$\mathbf{z} = \mathrm{V}(\mathbf{Y}^T\mathbf{Y}), \ \mathbf{Y} = \delta_\rho(\mathbf{X})$$

ρ: dropout probability

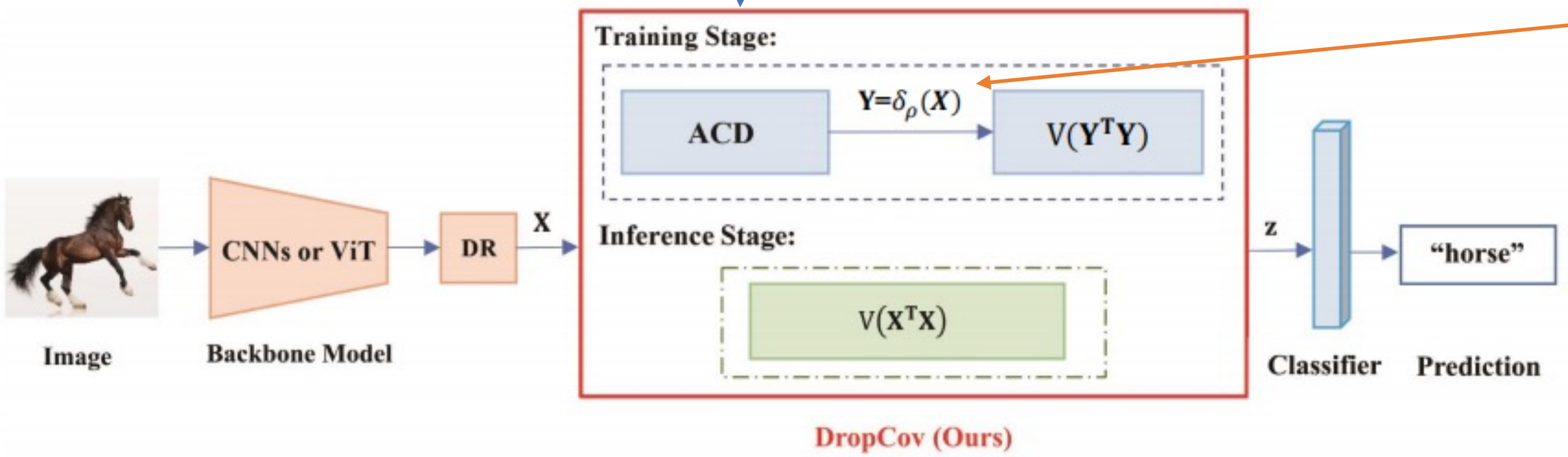
$$\begin{cases} \text{If} \quad \rho \longmapsto 0 & : \text{Preserves more information while performing less decorrelation.} \\ \text{If} \quad \rho \longmapsto 1 & : \text{Sparser, less correlated features but more information loss.} \end{cases}$$

Q: Could we use efficient dropout to perform representation decorrelation for GCP?

Proposed DropCov

Q: How to perform dropout ? & How to choose probability of dropout?

A pre-normalization performs adaptive channel dropout of probability $\boldsymbol{\rho}$ on features X before GCP: (1) keep the structure of GCP and (2) a linear computational complexity of O(d).



$$\rho = 1 - \frac{D}{\log(d)}\left(\frac{\boldsymbol{\omega}^T\boldsymbol{\pi}}{\|\boldsymbol{\omega}\|\|\boldsymbol{\pi}\|}\right)$$

Feature importance :

$$\boldsymbol{\omega} = \sigma(\mathrm{C1D}_3(\mathrm{GAP}(\mathbf{X})))$$

Feature correlation :

$$\boldsymbol{\pi} = \mathrm{SUM}_{\mathrm{row}}(\mathbf{X}^T\mathbf{X})$$

Since feature importance $\boldsymbol{\omega}$ and feature correlation $\boldsymbol{\pi}$ are closely related to representation decorrelation and information preservation, we adaptively decide probability of channel dropout for reaching a good trade-off between representation decorrelation and information preservation.

Experiments Results

Method	$d = 64$			$d = 256$		
	Top-1 Acc. (%)	Train (µs)	Test (µs)	Top-1 Acc. (%)	Train (µs)	Test (µs)
Plain GCP	71.1	3.45	1.04	70.0	35.08	11.89
B-CNN [31]	38.3	4.31	1.23	41.1	44.97	14.38
B-CNN + LT*	68.3	4.45	1.28	73.2	47.37	15.24
LN [2]	71.7	4.10	1.13	70.2	40.79	13.59
DeepO$_2$P [23]	70.1	922.50	910.95	*Not Converge*	9731.71	9638.71
I-LogM*	71.2	922.53	910.98	72.0	9732.46	9639.65
MPN-COV [29]	73.1	925.16	913.87	74.9	9735.11	9642.43
iSQRT-COV [28]	73.4	12.35	4.94	75.2	193.65	77.46
IB-CNN [30]	*Not Converge*	13.93	5.16	36.1	202.38	80.55
IB-CNN + LT*	70.0	14.16	5.21	72.8	207.48	82.98
DropCov (Ours)	**73.5**	**3.54**	**1.04**	**75.2**	**36.20**	**11.89**

◊: Note we compute running speed (µs) of single GCP module with post-normalization on a 2080Ti GPU.
♯: The original ResNet-18 with GAP achieves 70.2% in Top-1 accuracy.

Our DropCov performs better or on par with the counterparts in terms of efficiency and effectiveness, providing a very promising normalization method for deep GCP networks.

Experiment Results

Method	Params.	FLOPs	IN-1K (↑)	IN-C (↓)	IN-A (↑)	Sty.-IN (↑)
ResNet-34 [17]	21.8 M	3.66 G	74.19	77.9	1.63	7.59
ResNet-50 [17]	25.6 M	3.86 G	76.02	76.7	2.47	7.15
ResNet-101 [17]	44.6 M	7.57 G	77.67	70.3	4.15	9.51
ResNet-152 [17]	60.2 M	11.28 G	78.13	69.3	5.98	10.09
ResNet-34+DropCov (Ours)	29.6 M	5.56 G	76.81 (2.62)	71.1 (6.8)	3.45 (1.82)	11.16 (3.57)
ResNet-50+DropCov (Ours)	32.0 M	6.19 G	78.19 (2.17)	69.8 (6.9)	5.08 (2.61)	9.90 (2.75)
ResNet-101+DropCov (Ours)	51.0 M	9.90 G	**79.51** (1.84)	**65.8** (4.5)	**7.54** (3.39)	**11.41** (1.90)
DeiT-S [43]	22.1 M	4.6 G	79.8	54.6	18.9	14.91
Swin-T [32]	28.3 M	4.5 G	81.2	62.0	21.6	13.40
T2T-ViT-14 [49]	21.5 M	5.2 G	81.5	53.2	23.9	15.80
DeiT-B [43]	86.6 M	17.6 G	82.0	48.5	27.4	17.94
ConViT-B [6]	86.5 M	17.7 G	82.4	**46.9**	29.0	**19.67**
DeiT-S+DropCov (Ours)	25.6 M	5.5 G	82.4 (2.6)	52.6 (2.0)	31.2 (12.3)	17.10 (2.19)
Swin-T+DropCov (Ours)	31.6 M	6.0 G	82.5 (1.3)	54.8 (7.2)	33.1 (11.5)	14.13 (0.73)
T2T-ViT-14+DropCov (Ours)	24.9 M	5.4 G	**82.7** (1.2)	52.1 (1.1)	**31.7** (7.8)	18.81 (3.01)

Our DropCov provides a simple yet effective method to improve deep architectures. CNNs and ViTs with DropCov achieve better trade-offs between accuracy and model complexity.

Thanks！

Source code is available at ：

https://github.com/mingzeG/DropCov