3. ConvNeXt, SLaK (+ RepLKNet) (2022.07.29) 박희찬

Title Paper

- A ConvNet for the 20202s (2022, March)
- More ConvNets in the 20202s: Scaling up Kernels Beyond 51 x 51 using Sparsity (2022, July)

Ref

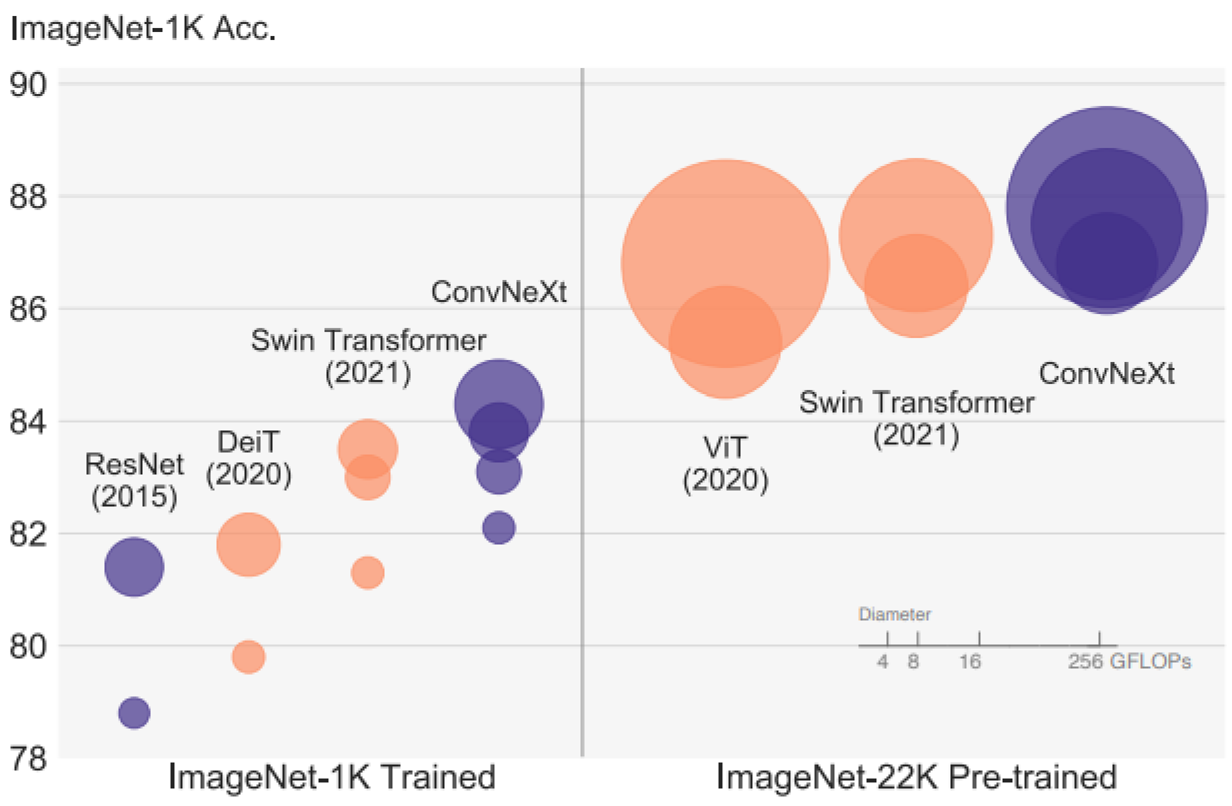
- Hierarchical Vision Transformer using Shifted Windows (Swin-Transformer) (2022)
- Scaling Up Your Kernels to 31x31: Revisiting Large Kernel Design in CNNs (RepNeXt)(2022 April)
- Rigging the Lottery: Making All Tickets Winners (Sparse Neural Network)

CNN의 Inductive Bias 요소들을 Transformer 에서도 강해지도록 연구

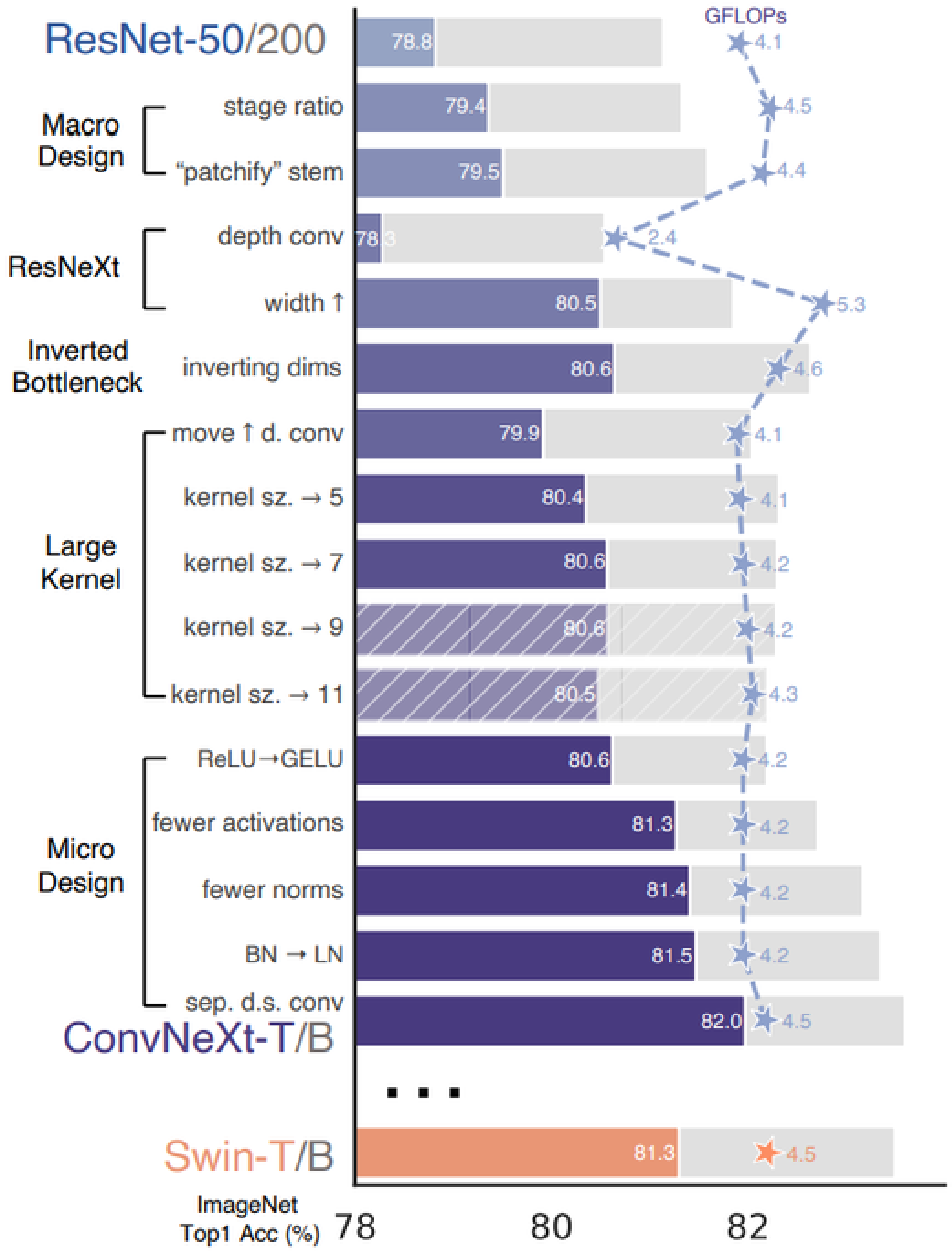
- Vision Transformer의 pachify -> Locality
- Swin Transformer의 Shifted Window -> Receptive Field

→

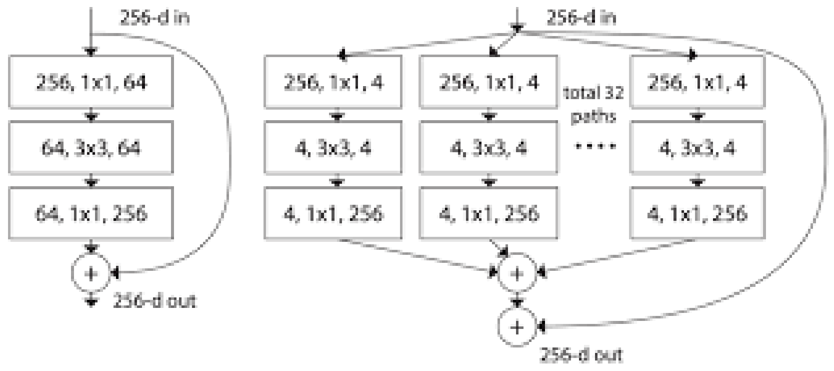
1. A ConvNet for the 2020s



- NLP 분야에서 처음 나온 Transformer가 Vision 분야에서도 활용되며 대표적으로 Vision Trtansformer (ViT) 를 기반으로 수많은 Transformer 모델들이 쏟아져 나오기 시작.
- 수많은 Transformer 모델들은 계속해서 기존의 잘 알려진 CNN 모델들과 비교하면서 SOTA를 달성하는 중에 저자들은 비교 대상인 CNN 모델들이 오래되었다고 생각하며 CNN 연구가 등한시 되어가는 것 같다고 주장. 대표적인 Transformer 모델인 ViT, Swin Transformer 의 아이디어와 여러가지 기법들을 CNN에 맞게 적용하여 좀 더 "Modernize" 한 CNN 모델 (**ConvNeXt**)로 Transformer 를 뛰어 넘을 수 있을 것 이라고 주장



1. stage ratio : swin transformer block 비율로 ResNet 블록 개수를 조정 (3464 → 3393)
2. "patchify" stem : 입력 부분에서 4x4 convolution, 4 stride. Swin Transformer의 패치를 나누는 것과 동일한 효과
3. ResNeXt (depth conv, width up)



4. Inverted Bottleneck

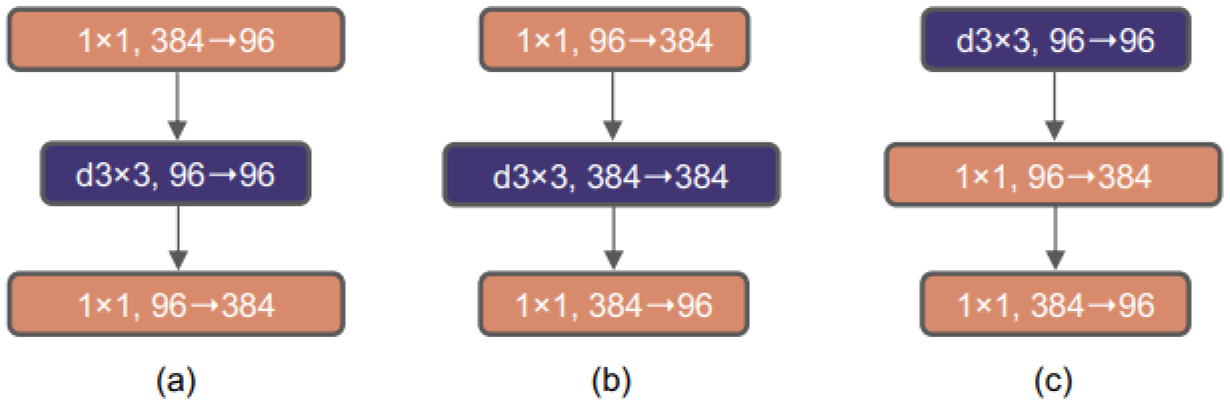
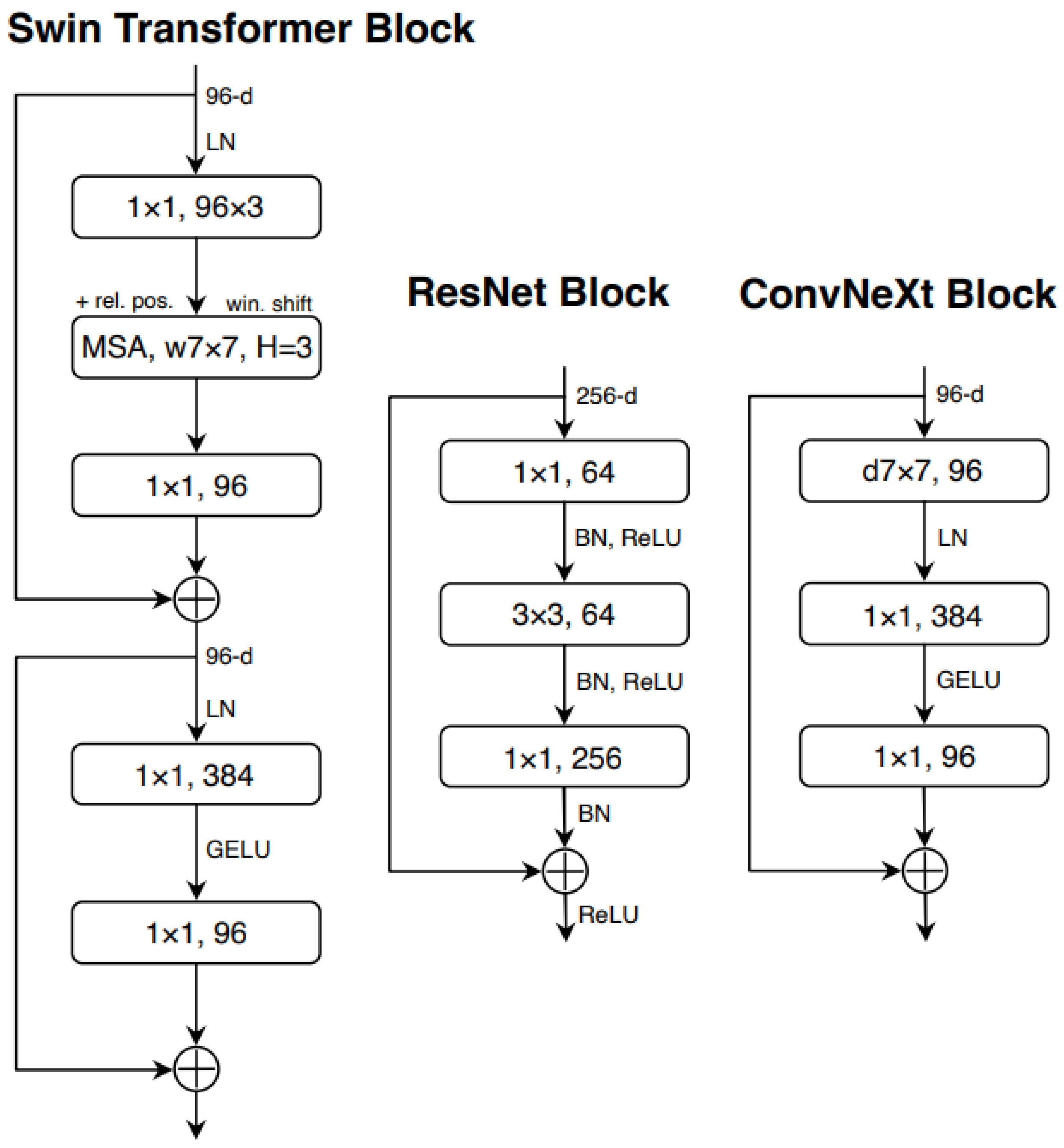


Figure 3. **Block modifications and resulted specifications. (a)** is a ResNeXt block; in **(b)** we create an inverted bottleneck block and in **(c)** the position of the spatial depthwise conv layer is moved up.

5. Large Kernel : Swin Transformer 의 커널 사이즈를 늘려준다.

6. Micro Design


Swin Transformer Block
96-d
LN
1×1, 96×3
+ rel. pos.
win. shift
MSA, w7×7, H=3
1×1, 96
96-d
LN
1×1, 384
GELU
1×1, 96
ResNet Block
256-d
1×1, 64
BN, ReLU
3×3, 64
BN, ReLU
1×1, 256
BN
ReLU
ConvNeXt Block
96-d
d7×7, 96
LN
1×1, 384
GELU
1×1, 96

model	IN-1K acc.	GFLOPs
ResNet-50 (PyTorch [1])	76.13	4.09
ResNet-50 (enhanced recipe)	78.82 ± 0.07	4.09
stage ratio	79.36 ± 0.07	4.53
"patchify" stem	79.51 ± 0.18	4.42
depthwise conv	78.28 ± 0.08	2.35
increase width	80.50 ± 0.02	5.27
inverting dimensions	80.64 ± 0.03	4.64
move up depthwise conv	79.92 ± 0.08	4.07
kernel size → 5	80.35 ± 0.08	4.10
kernel size → 7	80.57 ± 0.14	4.15
kernel size → 9	80.57 ± 0.06	4.21
kernel size → 11	80.47 ± 0.11	4.29
ReLU → GELU	80.62 ± 0.14	4.15
fewer activations	81.27 ± 0.06	4.15
fewer norms	81.41 ± 0.09	4.15
BN → LN	81.47 ± 0.09	4.46
separate d.s. conv (ConvNeXt-T)	81.97 ± 0.06	4.49
Swin-T [45]	81.30	4.50

model	IN-1K acc.	GFLOPs
ResNet-200 [29]	78.20	15.01
ResNet-200 (enhanced recipe)	81.14	15.01
stage ratio and increase width	81.33	14.52
"patchify" stem	81.59	14.38
depthwise conv	80.54	7.23
increase width	81.85	16.76
inverting dimensions	82.64	15.68
move up depthwise conv	82.04	14.63
kernel size → 5	82.32	14.70
kernel size → 7	82.30	14.81
kernel size → 9	82.27	14.95
kernel size → 11	82.18	15.13
ReLU → GELU	82.19	14.81
fewer activations	82.71	14.81
fewer norms	83.17	14.81
BN → LN	83.35	14.81
separate d.s. conv (ConvNeXt-B)	83.60	15.35
Swin-B [45]	83.50	15.43

model	image size	FLOPs	throughput (image / s)	IN-1K / 22K trained, 1K acc.
○ Swin-T	224^2	4.5G	1325.6	81.3 / –
● ConvNeXt-T	224^2	4.5G	**1943.5** (+47%)	**82.1 /** –
○ Swin-S	224^2	8.7G	857.3	83.0 / –
● ConvNeXt-S	224^2	8.7G	**1275.3** (+49%)	**83.1 /** –
○ Swin-B	224^2	15.4G	662.8	83.5 / 85.2
● ConvNeXt-B	224^2	15.4G	**969.0** (+46%)	**83.8 / 85.8**
○ Swin-B	384^2	47.1G	242.5	84.5 / 86.4
● ConvNeXt-B	384^2	45.0G	**336.6** (+39%)	**85.1 / 86.8**
○ Swin-L	224^2	34.5G	435.9	– / 86.3
● ConvNeXt-L	224^2	34.4G	**611.5** (+40%)	**84.3 / 86.6**
○ Swin-L	384^2	103.9G	157.9	– / 87.3
● ConvNeXt-L	384^2	101.0G	**211.4** (+34%)	85.5 / **87.5**
● ConvNeXt-XL	224^2	60.9G	**424.4**	– **/ 87.0**
● ConvNeXt-XL	384^2	179.0G	**147.4**	– **/ 87.8**

Table 12. **Inference throughput comparisons on an A100 GPU.** Using TF32 data format and "channel last" memory layout, ConvNeXt enjoys up to ~49% higher throughput compared with a Swin Transformer with similar FLOPs.

결론: (1) 최신화된 테크닉을 CNN에 적용하여 Swin Transformer 보다 높은 성능을 보여주었다.

(2) 실제로 Transformer 의 성능이 CNN과 비교하여 MHSA의 효과가 때문인지 검증해야 한다는 여지를 남김

Transformer > CNN ?

MLP-Mixer, ConvNeXt, EfficientFormer 등

CNN 혹은 MLP로 다시 돌아가는 논문들이 발표되는 중

Focus Kernel Size

Trade-off

- Large Receptive Field
- more parameter

***RepLKNet (RepVGG 저자가 발표한 논문)**

- Scaling Up Your Kernels to 31x31: Revisiting Large Kernel Design in CNNs (2022 April)

1. (1)ViT 에서 7x7 패치를 이용하여 (2)ConvNeXt 에서의 Large Kernel Size로 기존 CNN 3x3 Stack 보다 더 넓은 Receptive Field를 가져 높은 성능을 보여준 것에 주목해 31x31 까지 늘려서 실험

2. FLOPs 문제는 ConvNeXt 에서 소개한 것 처럼 Depth-wise convolution 으로 어느 정도 낮추는 효과

(Depth-wise + Large Kernel을 사용할 경우, GPU 메모리 재사용성 고려하여 Latency가 크게 증가하지 않음)

3. RepVGG 와 같이 Re-Parameterization

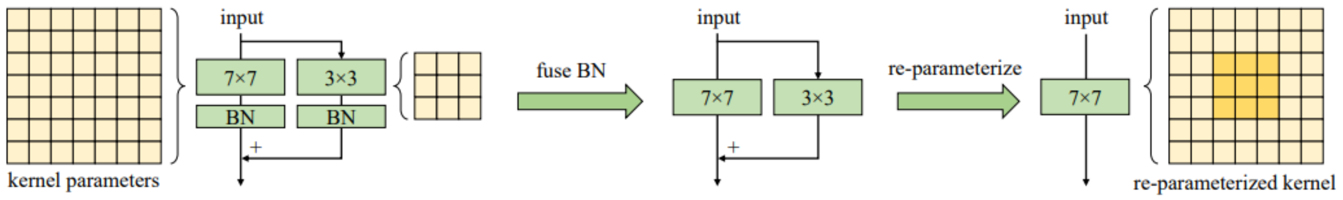


Figure 2. An example of re-parameterizing a small kernel (*e.g.*, 3×3) into a large one (*e.g.*, 7×7). See [28, 31] for details.

---

2. More ConvNets in the 20202s: Scaling up Kernels Beyond 51 x 51 using Sparsity

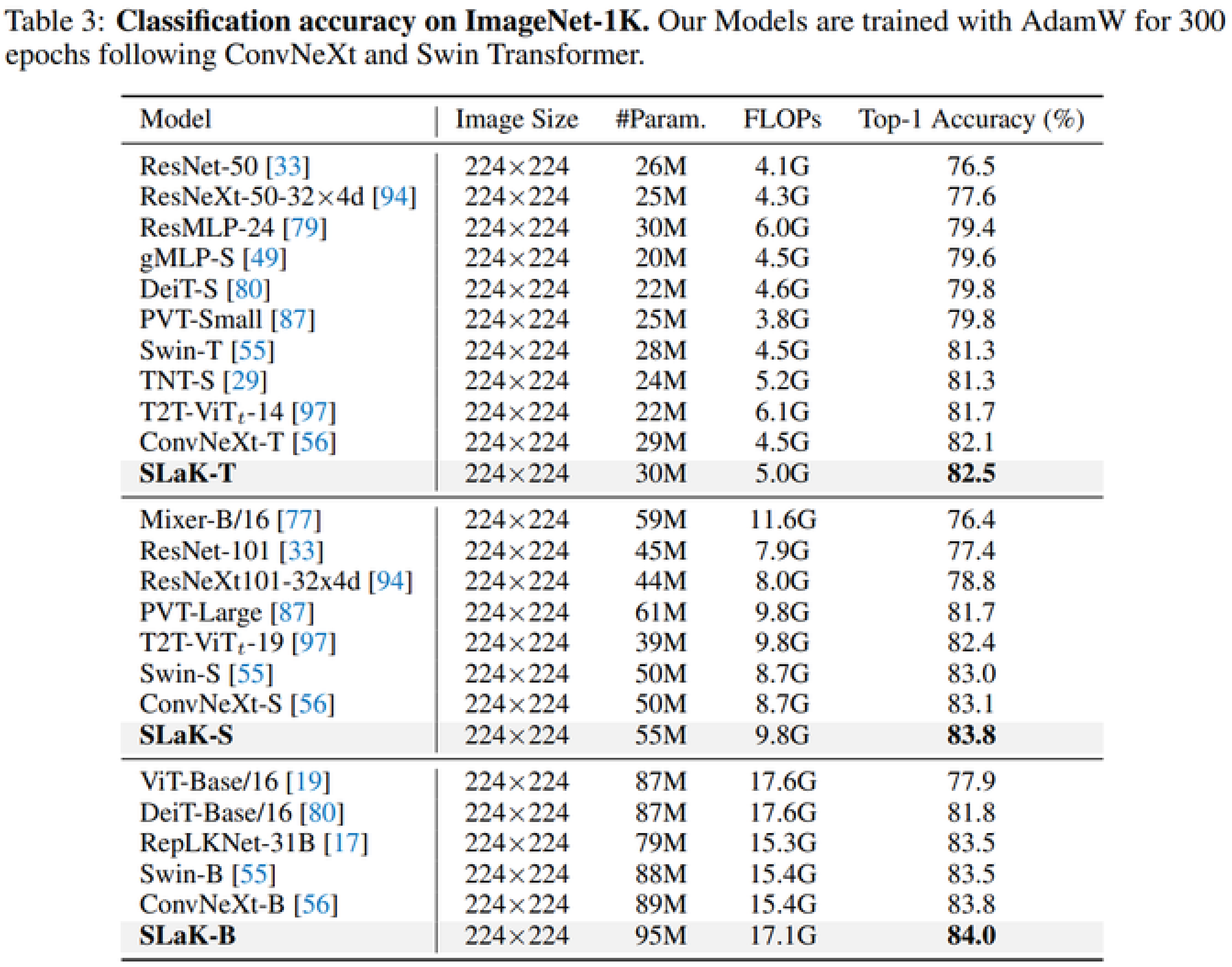
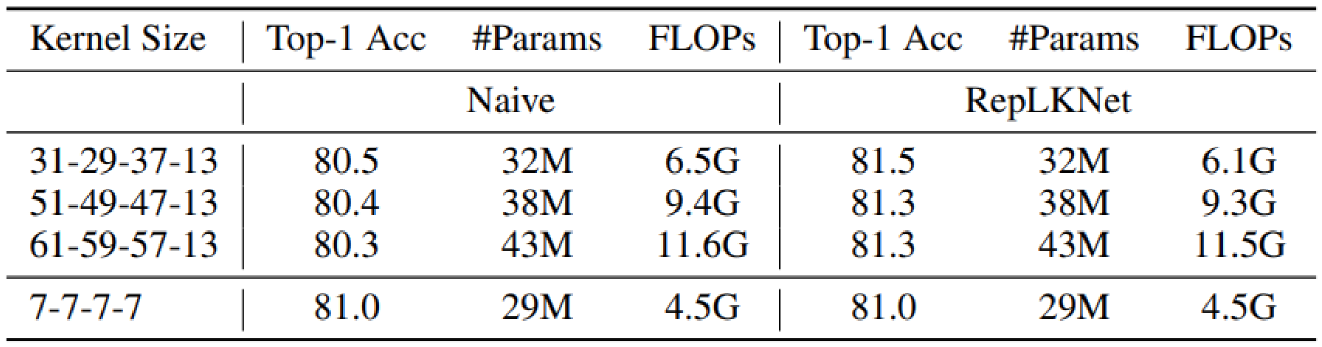


Table 3: **Classification accuracy on ImageNet-1K.** Our Models are trained with AdamW for 300 epochs following ConvNeXt and Swin Transformer.

Model	Image Size	#Param.	FLOPs	Top-1 Accuracy (%)
ResNet-50 [33]	224×224	26M	4.1G	76.5
ResNeXt-50-32×4d [94]	224×224	25M	4.3G	77.6
ResMLP-24 [79]	224×224	30M	6.0G	79.4
gMLP-S [49]	224×224	20M	4.5G	79.6
DeiT-S [80]	224×224	22M	4.6G	79.8
PVT-Small [87]	224×224	25M	3.8G	79.8
Swin-T [55]	224×224	28M	4.5G	81.3
TNT-S [29]	224×224	24M	5.2G	81.3
T2T-ViT$_t$-14 [97]	224×224	22M	6.1G	81.7
ConvNeXt-T [56]	224×224	29M	4.5G	82.1
SLaK-T	224×224	30M	5.0G	**82.5**
Mixer-B/16 [77]	224×224	59M	11.6G	76.4
ResNet-101 [33]	224×224	45M	7.9G	77.4
ResNeXt101-32x4d [94]	224×224	44M	8.0G	78.8
PVT-Large [87]	224×224	61M	9.8G	81.7
T2T-ViT$_t$-19 [97]	224×224	39M	9.8G	82.4
Swin-S [55]	224×224	50M	8.7G	83.0
ConvNeXt-S [56]	224×224	50M	8.7G	83.1
SLaK-S	224×224	55M	9.8G	**83.8**
ViT-Base/16 [19]	224×224	87M	17.6G	77.9
DeiT-Base/16 [80]	224×224	87M	17.6G	81.8
RepLKNet-31B [17]	224×224	79M	15.3G	83.5
Swin-B [55]	224×224	88M	15.4G	83.5
ConvNeXt-B [56]	224×224	89M	15.4G	83.8
SLaK-B	224×224	95M	17.1G	**84.0**

- Kernel Size를 극단적으로 넓혀서 (Extreme Kernel) 51x51, 61x61까지 실험
- 기존의 Large Kernel 을 적용한 논문들은 Extreme Kernel 에서 성능이 좋지 않았음



Kernel Size	Top-1 Acc	#Params	FLOPs	Top-1 Acc	#Params	FLOPs
	Naive			RepLKNet		
31-29-37-13	80.5	32M	6.5G	81.5	32M	6.1G
51-49-47-13	80.4	38M	9.4G	81.3	38M	9.3G
61-59-57-13	80.3	43M	11.6G	81.3	43M	11.5G
7-7-7-7	81.0	29M	4.5G	81.0	29M	4.5G

- M x M 사이즈의 Large Kernel 을 M x N, N x M (M > N) 2 사각형으로 Decompose 하여 실험

Kernel Size	Top-1 Acc	#Params	FLOPs	Top-1 Acc	#Params	FLOPs	Top-1 Acc	#Params	FLOPs
	Decomposed			Sparse Decomposed			Sparse Decomposed + 1.3× width		
31-29-37-13	81.3	30M	5.0G	80.4	18M	2.9G	81.5	30M	4.8G
51-49-47-13	81.5	31M	5.4G	80.5	18M	3.1G	81.6	30M	5.0G
61-59-57-13	81.4	31M	5.6G	80.4	19M	3.2G	81.5	31M	5.2G
7-7-7-7	81.0	29M	4.5G	81.0	29M	4.5G	81.0	29M	4.5G

- Extreme Kernel 로 성능을 향상 시킬 수 있는 Sparse Large Kernel (SLaK) 제안
- SLaK은 ConvNeXt 기반의 RepLKNet 의 Reparametric Method 를 Dynamic Sparsity 라는 Pruning 기법을 적용
- 커널의 일부를 Sparsity Weight 로 대체

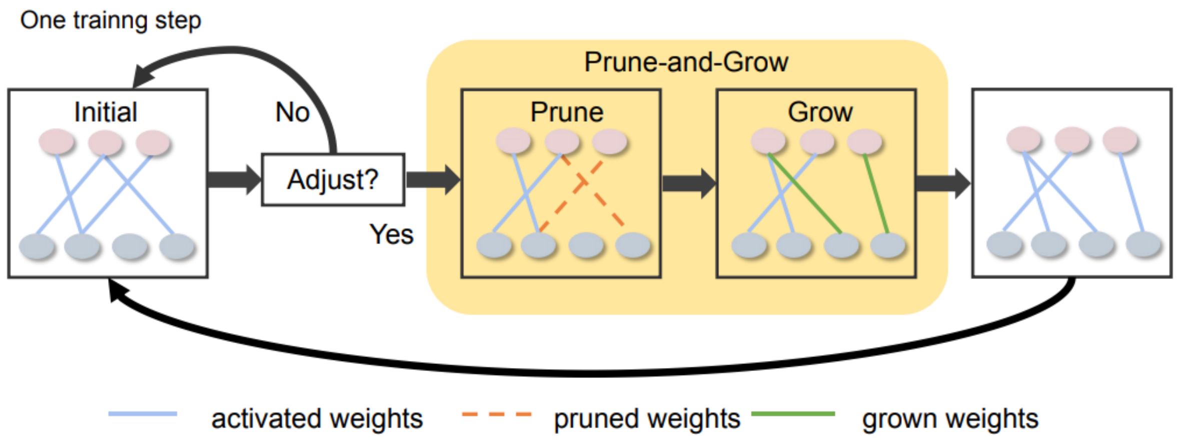


Figure 2: **Dynamic sparsity.** Dynamic sparsity allows us to construct and train initially sparse neural networks (sparse kernels) from scratch. During training, it dynamically adjusts the sparse weights by pruning the least important weights and adding new. Such dynamic procedure gradually optimizes the sparse kernels to a good pattern and hence encourages a more elaborate capture of local features.

Table 3: **Classification accuracy on ImageNet-1K.** Our Models are trained with AdamW for 300 epochs following ConvNeXt and Swin Transformer.

Model	Image Size	#Param.	FLOPs	Top-1 Accuracy (%)
ResNet-50 [33]	224×224	26M	4.1G	76.5
ResNeXt-50-32×4d [94]	224×224	25M	4.3G	77.6
ResMLP-24 [79]	224×224	30M	6.0G	79.4
gMLP-S [49]	224×224	20M	4.5G	79.6
DeiT-S [80]	224×224	22M	4.6G	79.8
PVT-Small [87]	224×224	25M	3.8G	79.8
Swin-T [55]	224×224	28M	4.5G	81.3
TNT-S [29]	224×224	24M	5.2G	81.3
T2T-ViT$_t$-14 [97]	224×224	22M	6.1G	81.7
ConvNeXt-T [56]	224×224	29M	4.5G	82.1
SLaK-T	224×224	30M	5.0G	**82.5**
Mixer-B/16 [77]	224×224	59M	11.6G	76.4
ResNet-101 [33]	224×224	45M	7.9G	77.4
ResNeXt101-32x4d [94]	224×224	44M	8.0G	78.8
PVT-Large [87]	224×224	61M	9.8G	81.7
T2T-ViT$_t$-19 [97]	224×224	39M	9.8G	82.4
Swin-S [55]	224×224	50M	8.7G	83.0
ConvNeXt-S [56]	224×224	50M	8.7G	83.1
SLaK-S	224×224	55M	9.8G	**83.8**
ViT-Base/16 [19]	224×224	87M	17.6G	77.9
DeiT-Base/16 [80]	224×224	87M	17.6G	81.8
RepLKNet-31B [17]	224×224	79M	15.3G	83.5
Swin-B [55]	224×224	88M	15.4G	83.5
ConvNeXt-B [56]	224×224	89M	15.4G	83.8
SLaK-B	224×224	95M	17.1G	**84.0**

- ConvNeXt-T: C = (96, 192, 384, 768), B = (3, 3, 9, 3) • ConvNeXt-S: C = (96, 192, 384, 768), B = (3, 3, 27, 3) • ConvNeXt-B: C = (128, 256, 512, 1024), B = (3, 3, 27, 3)

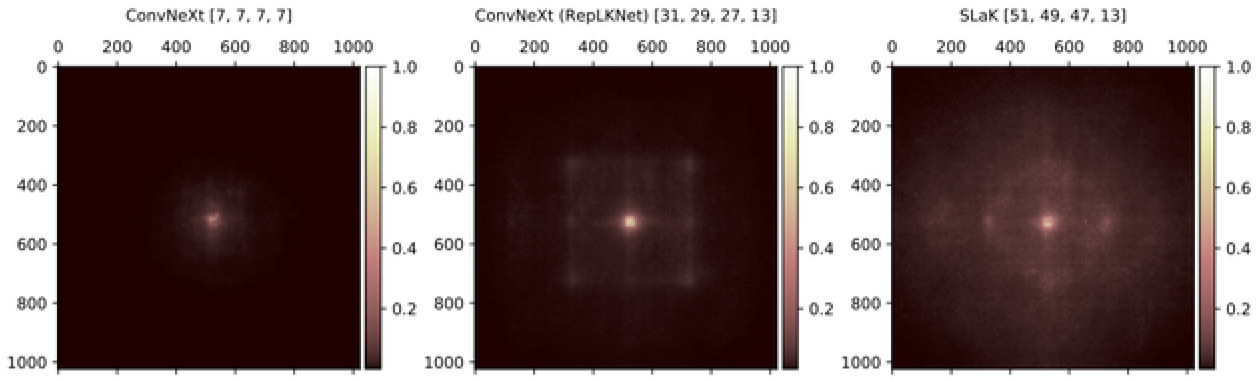


Figure 3: **Effective receptive field (ERF). Left:** ERF of the original ConvNeXt-T constructed with kernel sizes of [7, 7, 7, 7]. **Middle:** ERF of ConvNeXt-T constructed with kernel sizes of [31, 29, 27, 13] trained with the structural re-parameterization used in RepLKNet [17]. **Right:** ERF of SLaK-T constructed with kernel size [51, 49, 47, 13]. SLaK is not only able to capture long-range dependence but also the local context features.

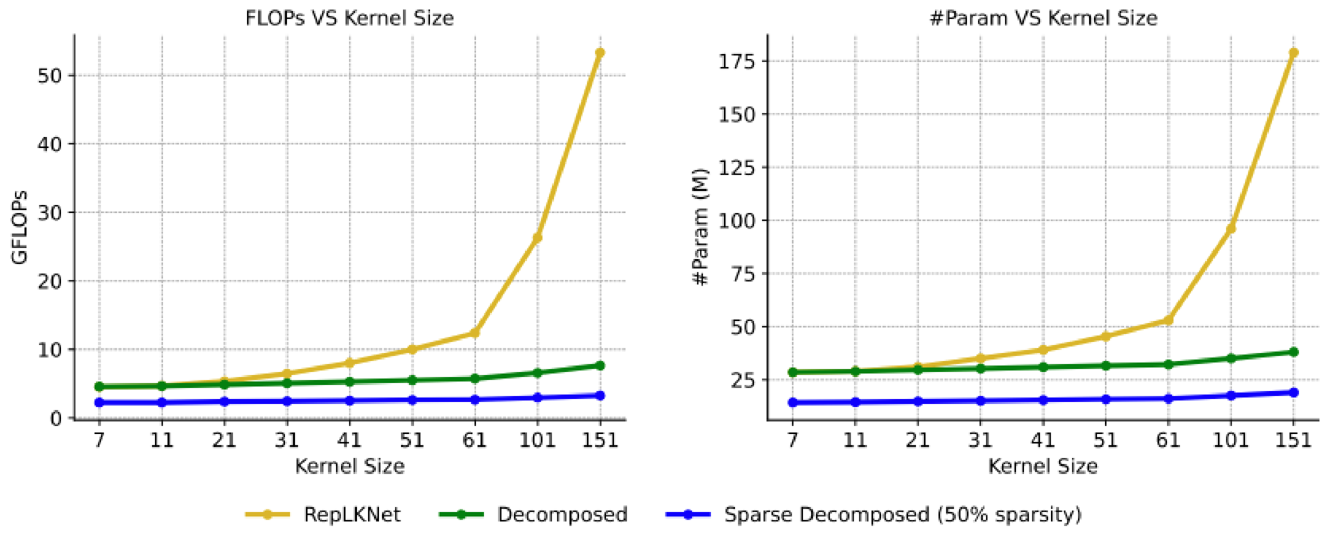


Table 7: **Trade-off between sparsity and width.** The experiments are conducted on SLaK-T trained for 120 epoch on ImageNet-1K.

Model	(Sparsity, Width)	#Param	FLOPs	Top-1 Accuracy (%)
SLaK-S	(0.20, 1.1×)	29M	5.0G	81.3
	(0.40, 1.3×)	30M	5.0G	81.6
	(0.55, 1.5×)	30M	4.9G	81.7
	(0.70, 1.9×)	32M	5.0G	81.6
	(0.82, 2.5×)	32M	5.1G	81.2

결론

- Sparsity Decomposing + Re-Parametric 커널 사이즈 대비 연산량을 극적으로 낮춤

개인 총평.

1.CNN은 아직도 Transformer 와 경쟁 가능

2.Large Kernel 을 늘려서 실험한다는 발상은 오래전부터 있었을 것 같은데...

3.CNN에 여러가지 기법을 엄청 넣어서 Transformer 성능을 따라 잡긴 했는데..