A ConvNet for the 2020s

Zhuang Liu[1,2*] Hanzi Mao[1] Chao-Yuan Wu[1] Christoph Feichtenhofer[1] Trevor Darrell[2] Saining Xie[1†]

[1]Facebook AI Research (FAIR) [2]UC Berkeley

2023.07.14

TAEWON KIM

0. Contents

1. Introduction

1. The Rise of ConvNets
2010년대에는 컨볼루션 신경망(ConvNet)이 물체 감지와 같은 작업에 적합한 고유한 설계 덕분에 컴퓨터 비전 분야를 주도했습니다. 주목할 만한 모델로는 AlexNet, VGGNet, ResNe(X)t 등이 있습니다.

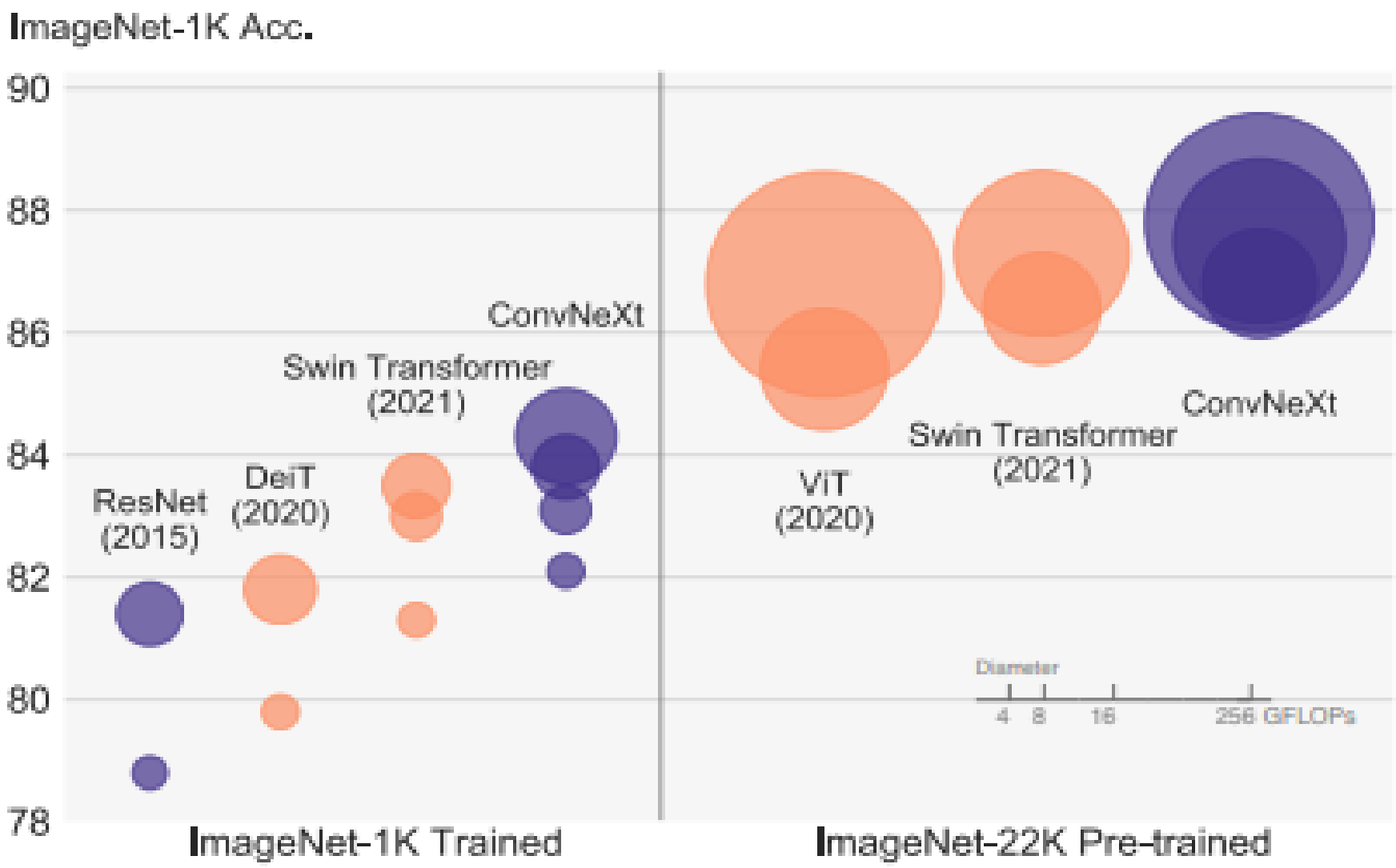
2. The Arrival of Vision Transformers
2020년, 비전 트랜스포머(ViT)가 컴퓨터 비전에 등장했습니다. 그러나 입력 크기에 따라 복잡성이 증가하기 때문에 고해상도 작업에서 사용하기에는 어려움이 있었습니다.

3. Hierarchical Transformers:
ViT의 문제를 해결하기 위해 Swin Transformer와 같은 계층적 트랜스포머는 ConvNets과 유사한 'sliding window' 전략을 재도입하여 다양한 컴퓨터 비전 작업에서 성공을 거뒀습니다.

4. The Case for ConvNets
트랜스포머의 발전에도 불구하고 복잡성과 효율성이라는 상충 관계에 놓여 있었습니다. 이는 컴퓨터 비전에서 ConvNets의 단순성과 효율성의 지속적인 중요성을 강조했습니다.

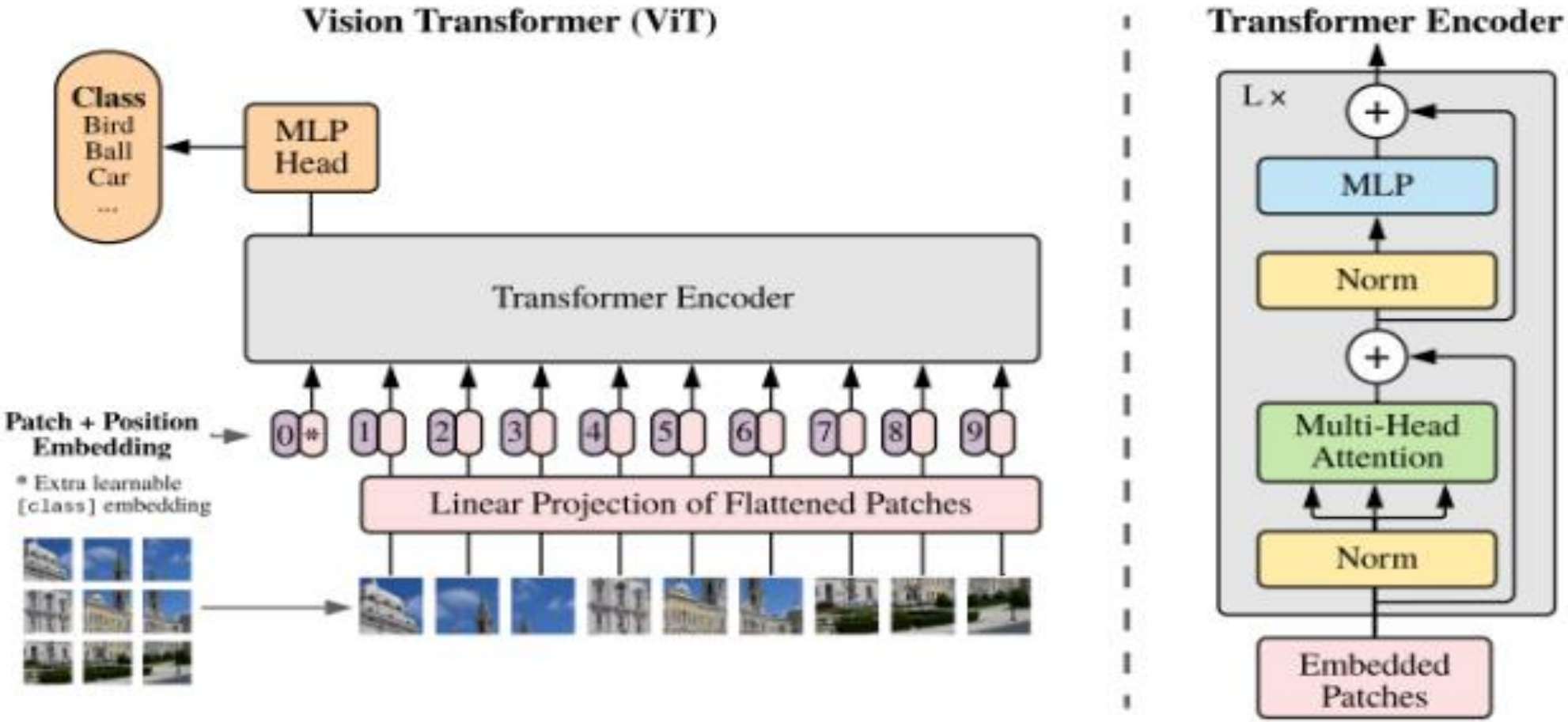
5. Introducing ConvNeXt
저자들은 트랜스포머와 경쟁할 수 있는 성능을 제공하는 현대화된 ConvNet인 ConvNeXt를 소개했습니다. 이 모델은 ConvNet의 효율성 및 단순성과 Transformers의 정확성 및 확장성을 결합한 모델입니다

1. Introduction



2. Background

Vision Transformer (VIT)



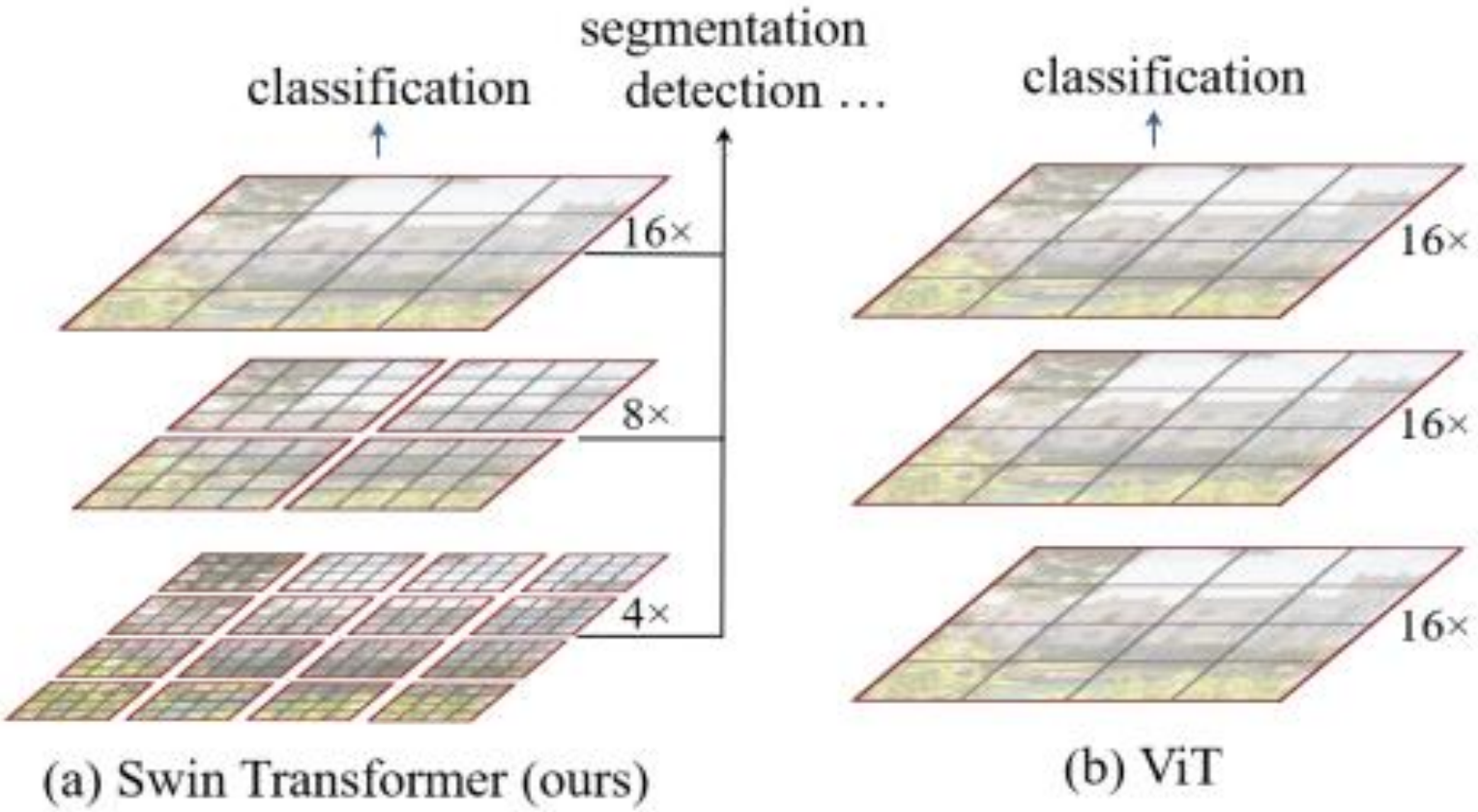
2. Background

Vision Transformer (VIT)

2. Background

Swin Transformer

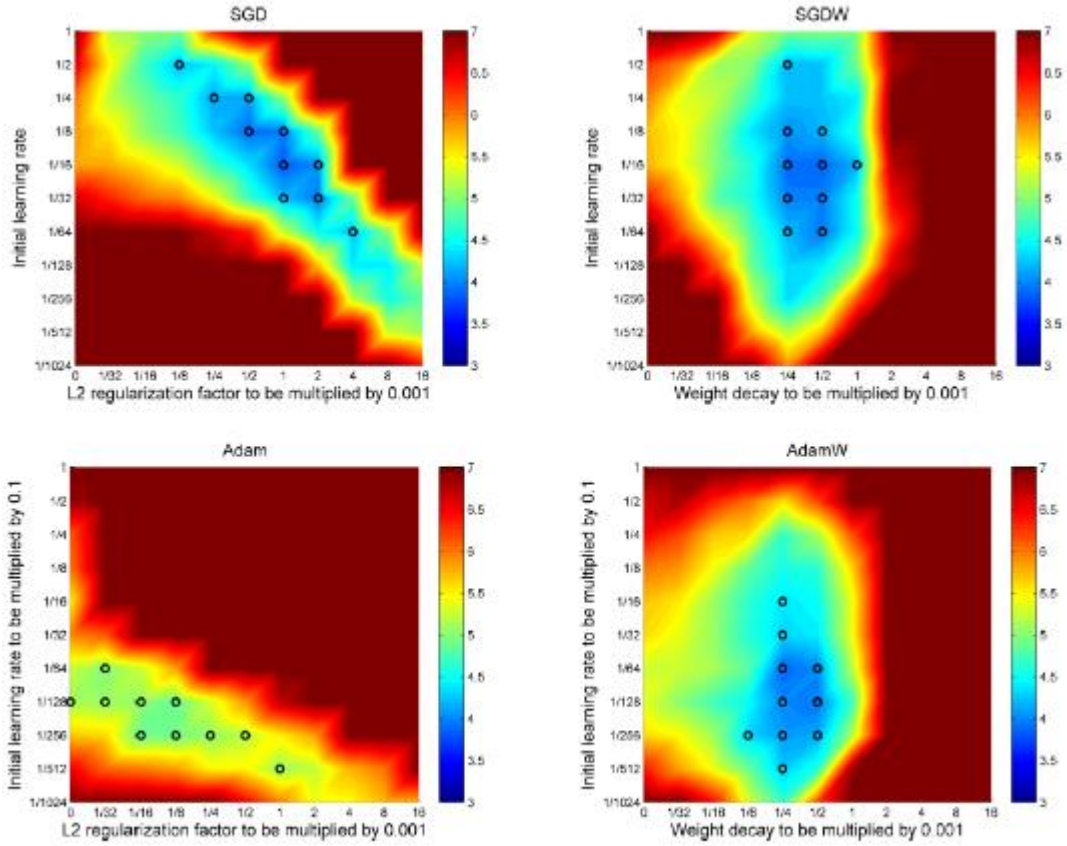
- ✓ VIT 모델은 classification 이외에 downstream task를 수행하기 적합하지 않은 형태
- ✓ Swin Transformer에서는 Window Self-Attention을 통해 inductive bias를 주입하고 block 별 down-sampling을 수행
- ✓ Global attentio과 local attention을 모두 활용한 구조



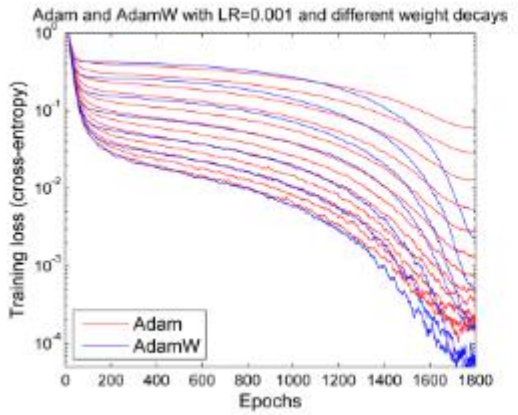
3. Modernizing ConvNet

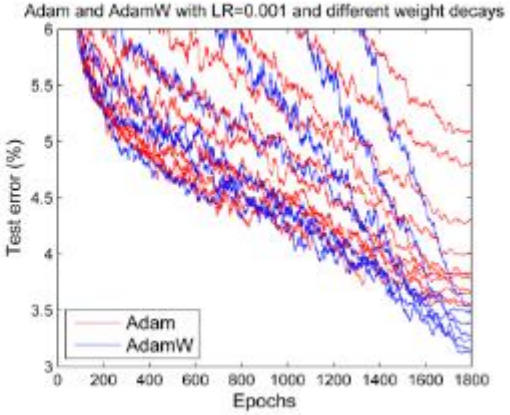
Training techniques

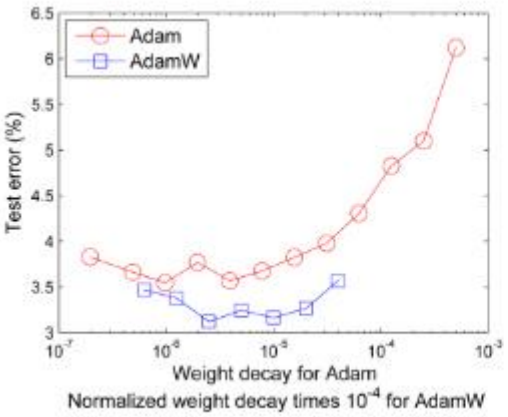
- ✓ Train은 AdamW로 수정하여 학습을 진행
- ✓ AdamW는 weight decay와 learning rate 간 다른 방법에 비해 독립적이며 일반화 성능이 우수

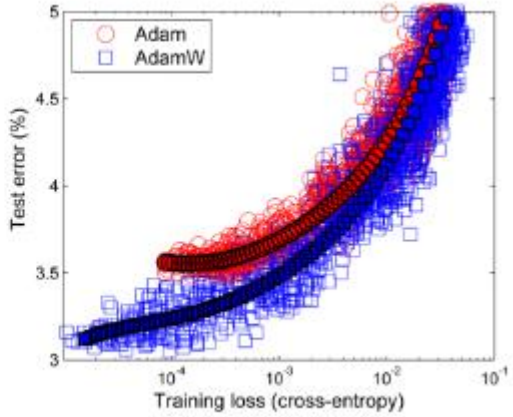


[서로 다른 initial decay rate와 learning rate에 따른 test error]



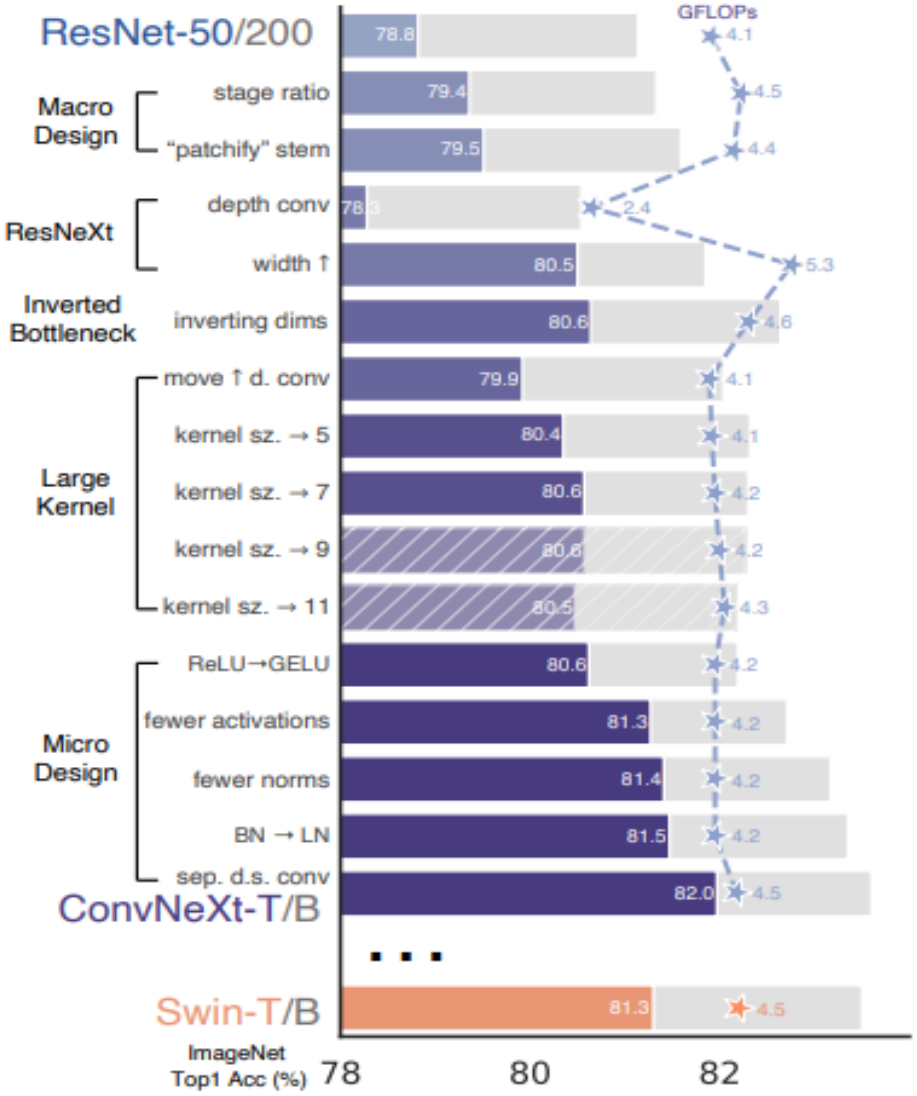






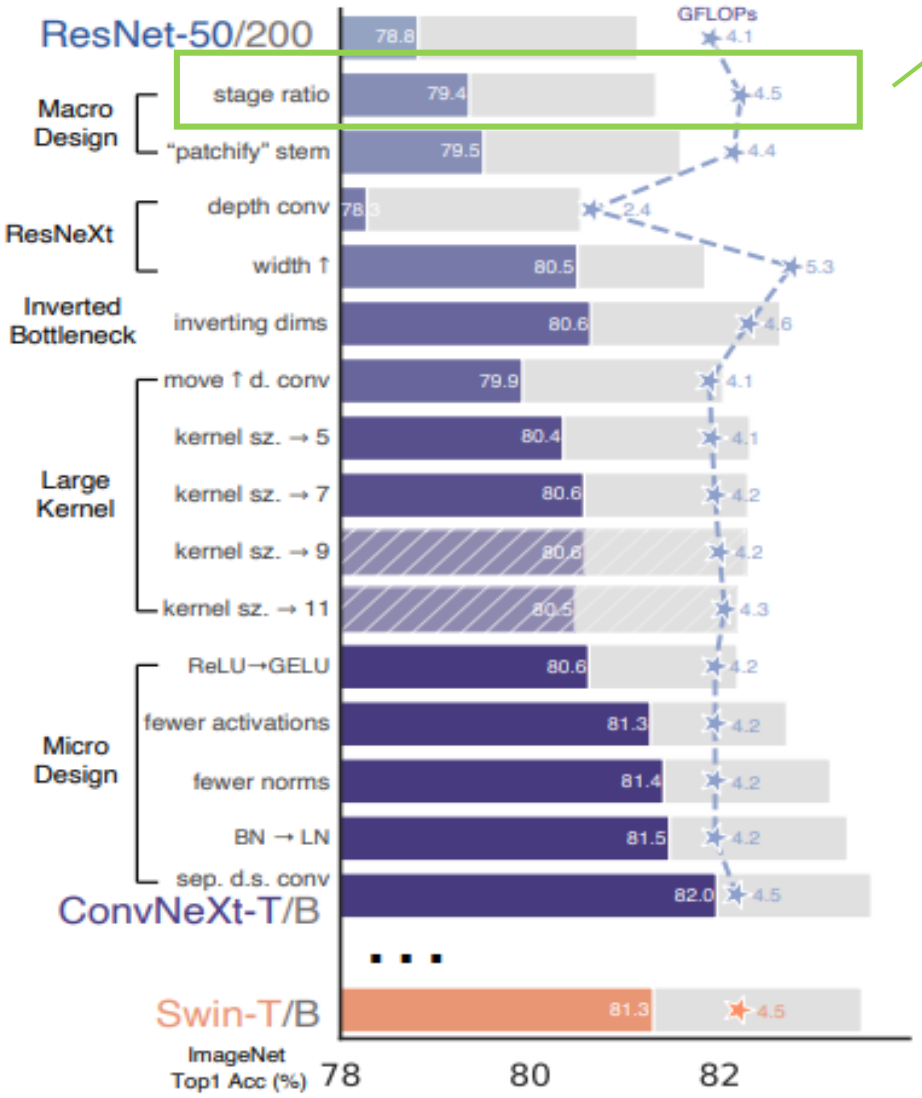
3. Modernizing a ConvNet

✓ Macro Desigin, ResNeXt, Inverted Bottleneck, Large Kernel, Micro Design



3. Modernizing a ConvNet(Macro Design)

✓ Stage ration

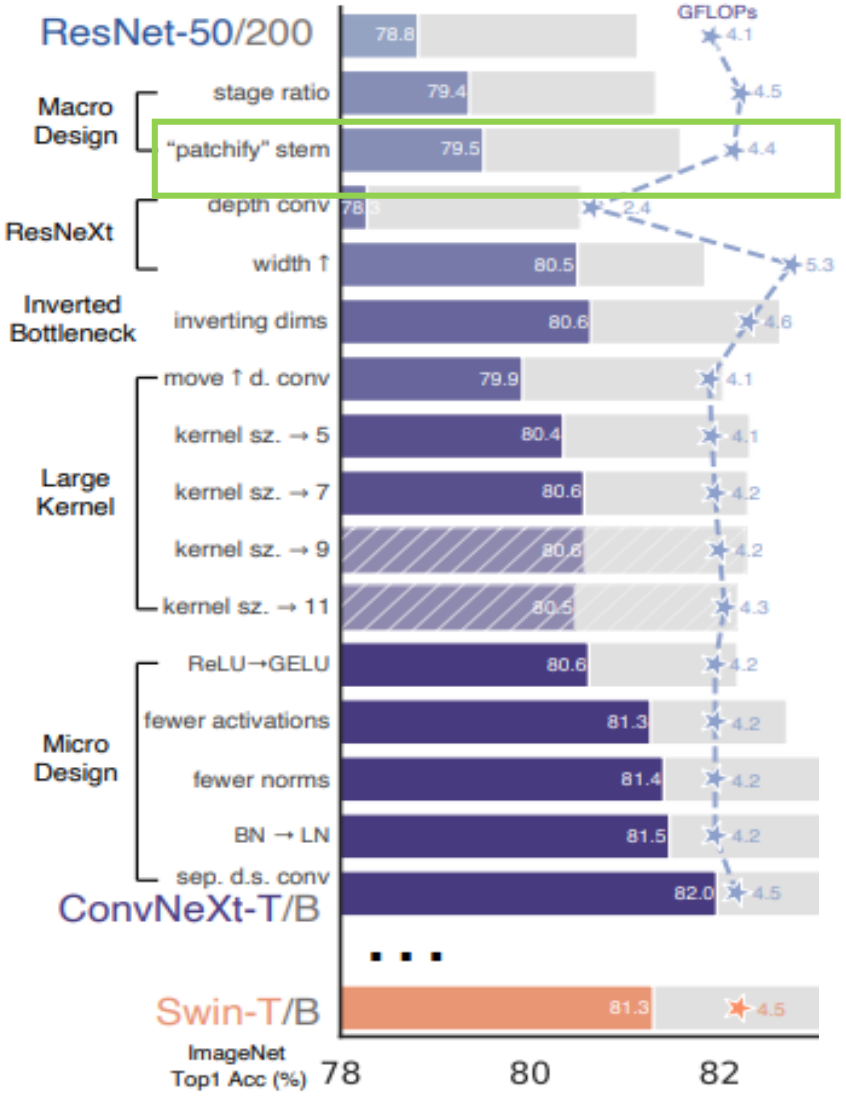


- 기존 stage 구성인 3,4,6,3을 Swin Transformer의 1:1:3:1 비율에 맞게 수정

	output size	• ResNet-50	• ConvNeXt-T	○ Swin-T
stem	56×56	7×7, 64, stride 2 3×3 max pool, stride 2	4×4, 96, stride 4	4×4, 96, stride 4
res2	56×56	[1×1, 64 3×3, 64 1×1, 256] × 3	[d7×7, 96 1×1, 384 1×1, 96] × 3	[1×1, 96×3 MSA, w7×7, H=3, rel. pos. 1×1, 96 [1×1, 384 1×1, 96] × 2
res3	28×28	[1×1, 128 3×3, 128 1×1, 512] × 4	[d7×7, 192 1×1, 768 1×1, 192] × 3	[1×1, 192×3 MSA, w7×7, H=6, rel. pos. 1×1, 192 [1×1, 768 1×1, 192] × 2
res4	14×14	[1×1, 256 3×3, 256 1×1, 1024] × 6	[d7×7, 384 1×1, 1536 1×1, 384] × 9	[1×1, 384×3 MSA, w7×7, H=12, rel. pos. 1×1, 384 [1×1, 1536 1×1, 384] × 6
res5	7×7	[1×1, 512 3×3, 512 1×1, 2048] × 3	[d7×7, 768 1×1, 3072 1×1, 768] × 3	[1×1, 768×3 MSA, w7×7, H=24, rel. pos. 1×1, 768 [1×1, 3072 1×1, 768] × 2
FLOPs		4.1×10^9	4.5×10^9	4.5×10^9
# params.		25.6×10^6	28.6×10^6	28.3×10^6

3. Modernizing a ConvNet(Macro Design)

✓ “Patchify”stem

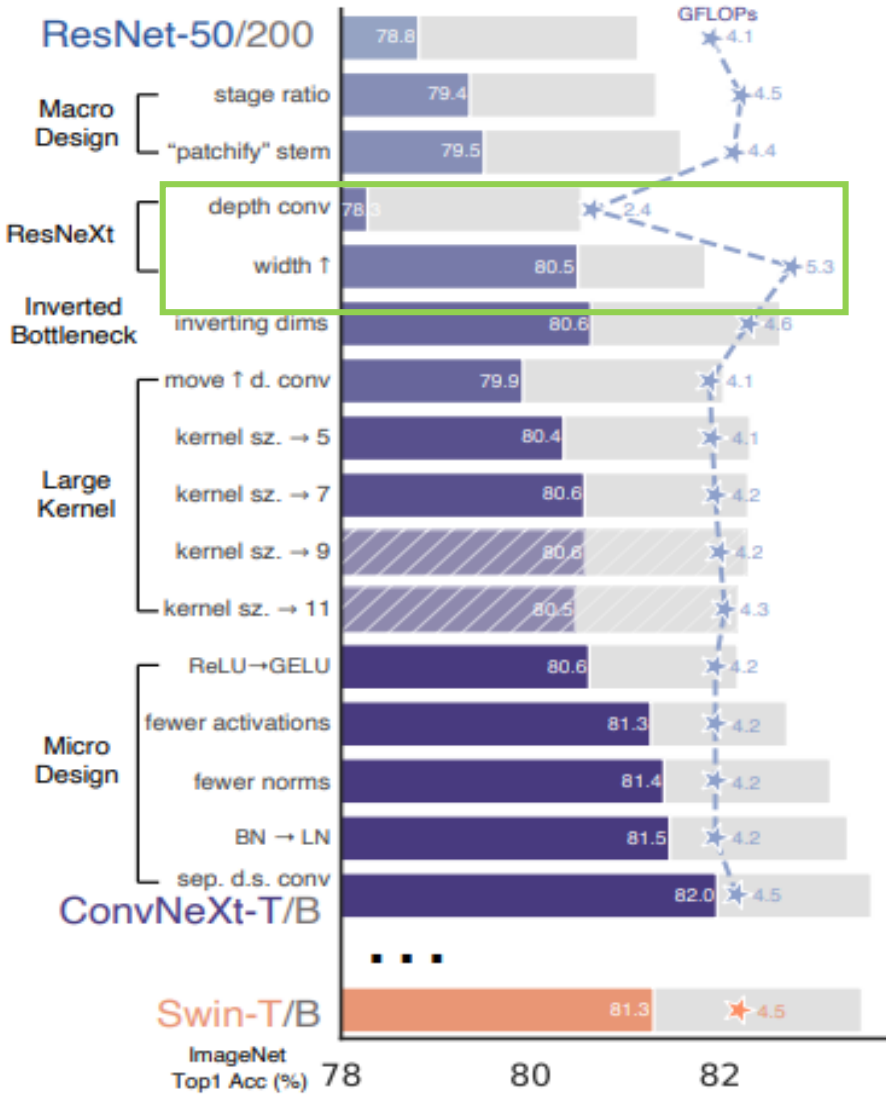


- Stam이란 신경망의 시작 부분을 의미하고 입력 이미지를 처리하고, 적절한 feature map으로 down sampling 합니다.
- Resnet에서의 stem은 7x7 Convolution 사용하고 strid가 2인 Maxpooling 레이어를 사용하여 입력 이미지는 4배 다운 샘플링 됩니다.
- 이 실험에서는, ResNet 스타일의 stem을 4x4 크기와 strid가 4인 “patchify” Convolution layer로 대체하였으며, 결과적으로는, 정확도는 79.4%에서 79.5%로 약간 변화하였는데, 이는 Resnet의 stem이 ViT 방식의 더 단순한 “patchify” layer로 대체될수 있으며, 비슷한 성능을 보여주는것을 의미

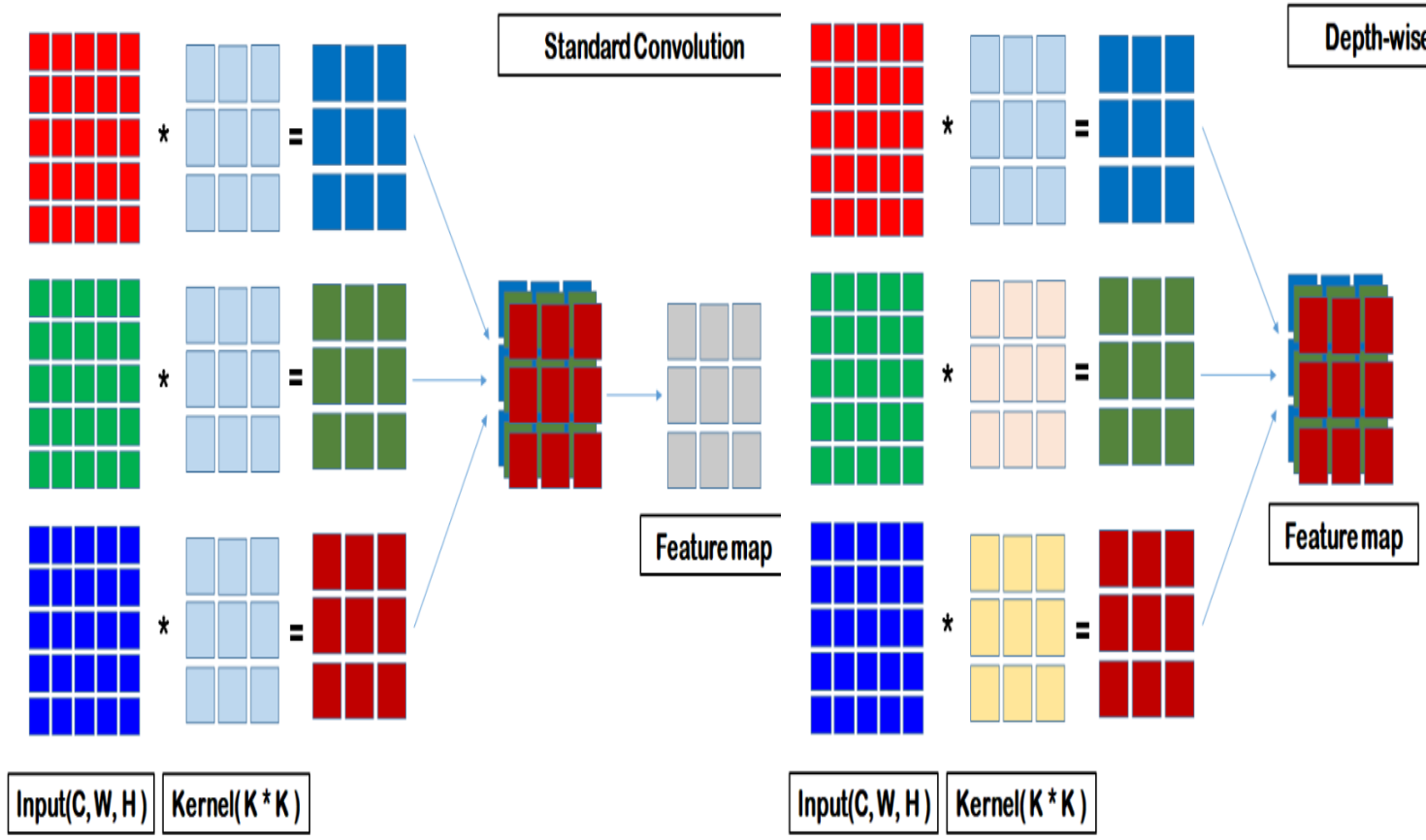
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

3. Modernizing a ConvNet(ResNeXt)

✓ ResNeXt

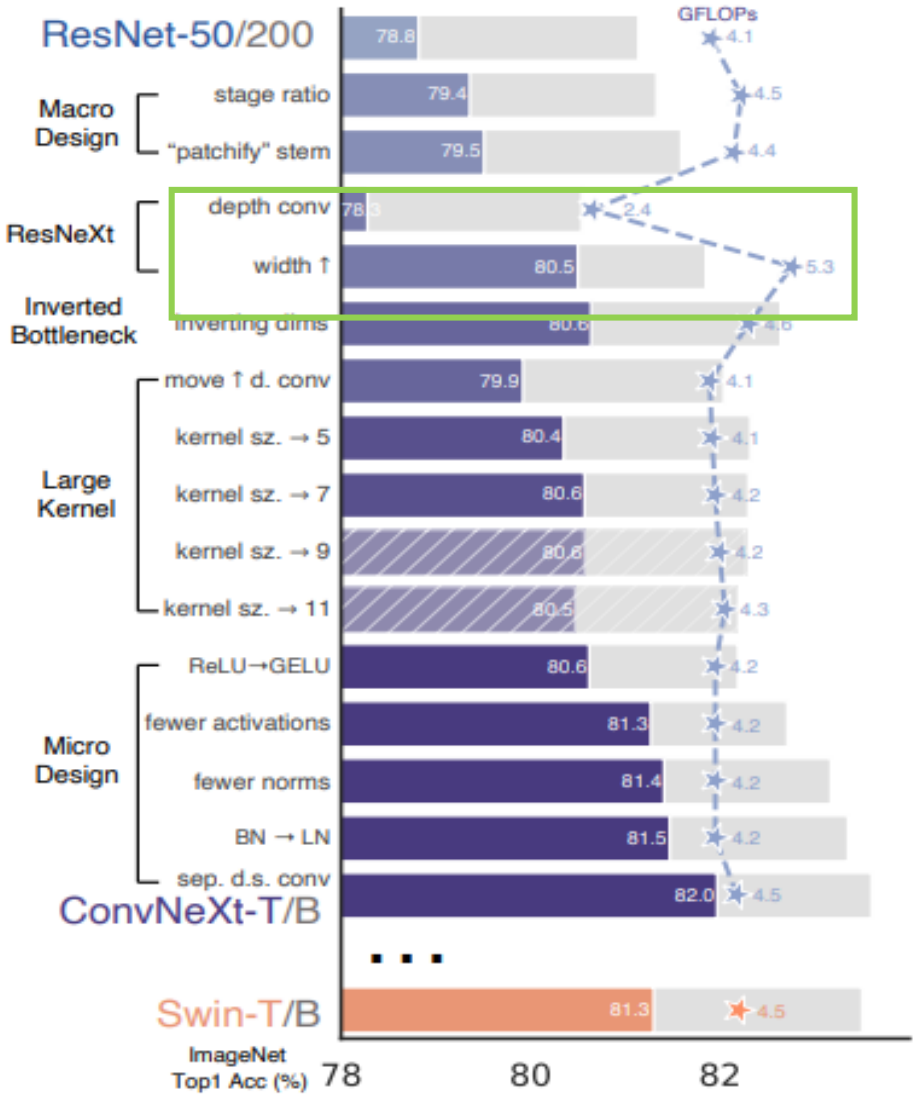


- ResNeXt에서 적용하는 depthwise seperable convolution 을 사용하여 연산량 (FLOPs)를 줄이고 capacity는 유지(Swin과 동일한 64 ->96)

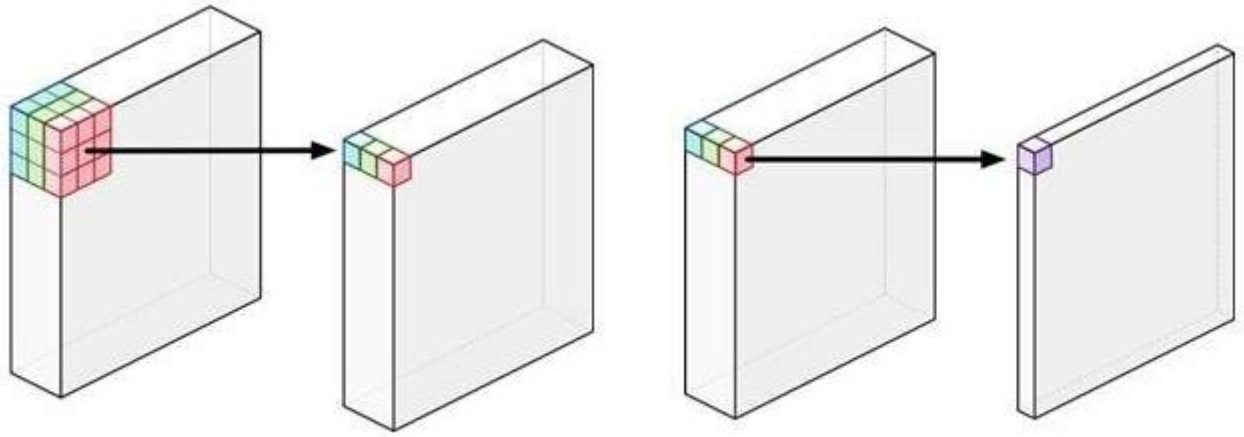


3. Modernizing a ConvNet(ResNeXt)

✓ ResNeXt



- ResNeXt에서 적용하는 depthwise seperable convolution 을 사용하여 연산량 (FLOPs)를 줄이고 capacity는 유지(Swin과 동일한 64 ->96)

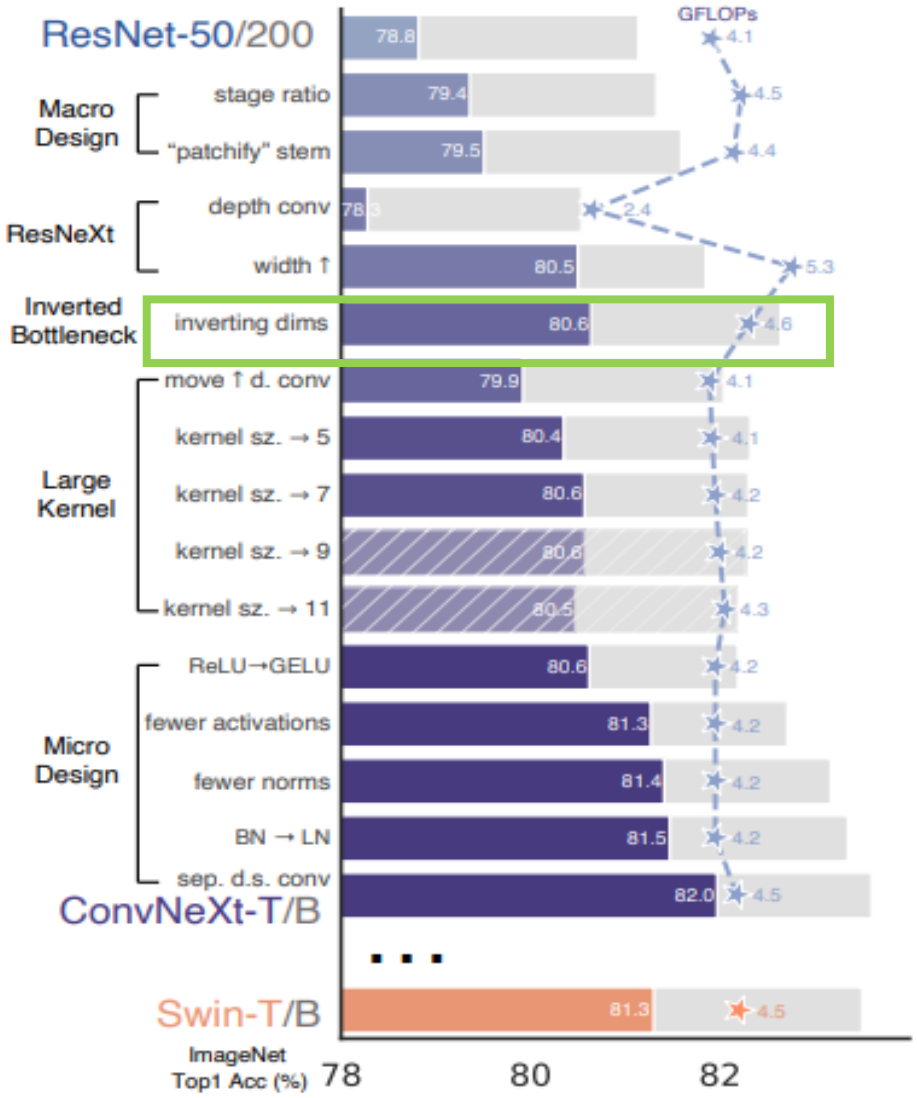


a) depthwise convolution

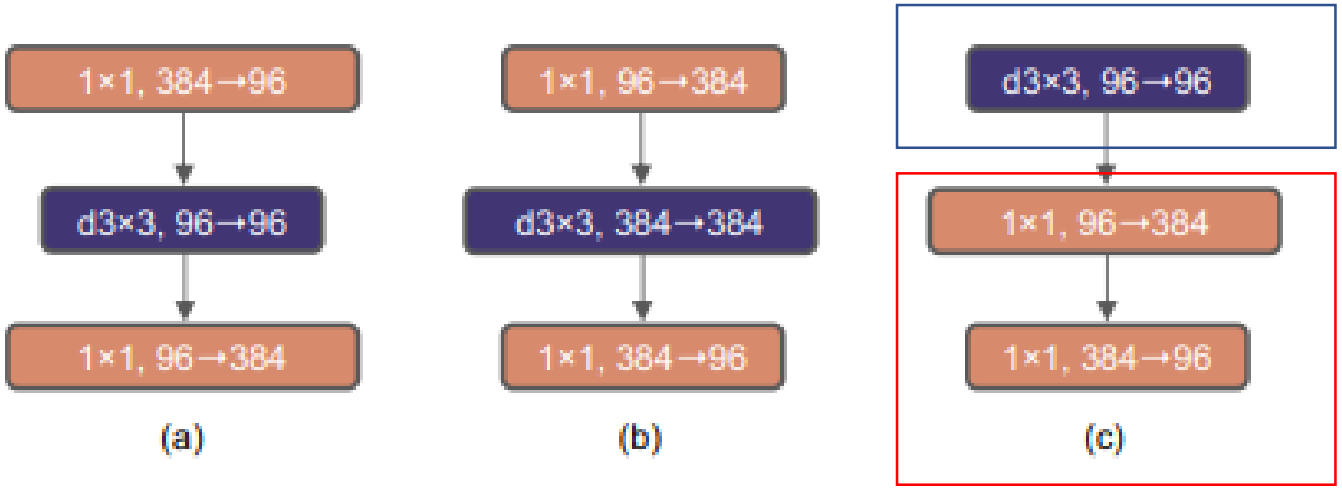
b) pointwise convolution

3. Modernizing a ConvNet (Inverted Bottleneck)

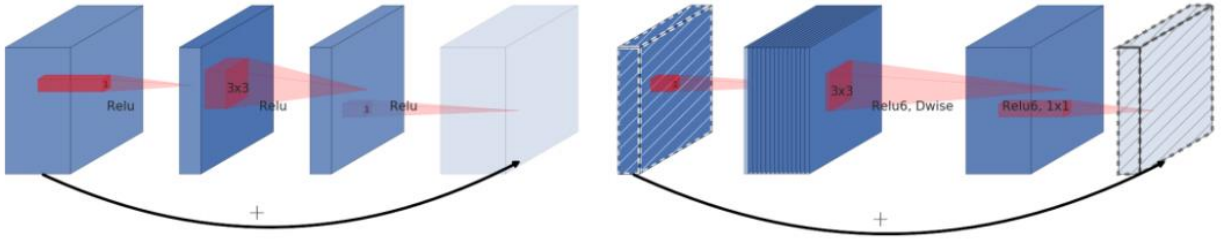
✓ Inverted Bottleneck



- MobileNetV2의 inverted residuals 방법을 사용 (메모리 효율적)
- Large kernel size를 위해서 위쪽으로 layer를 이동(



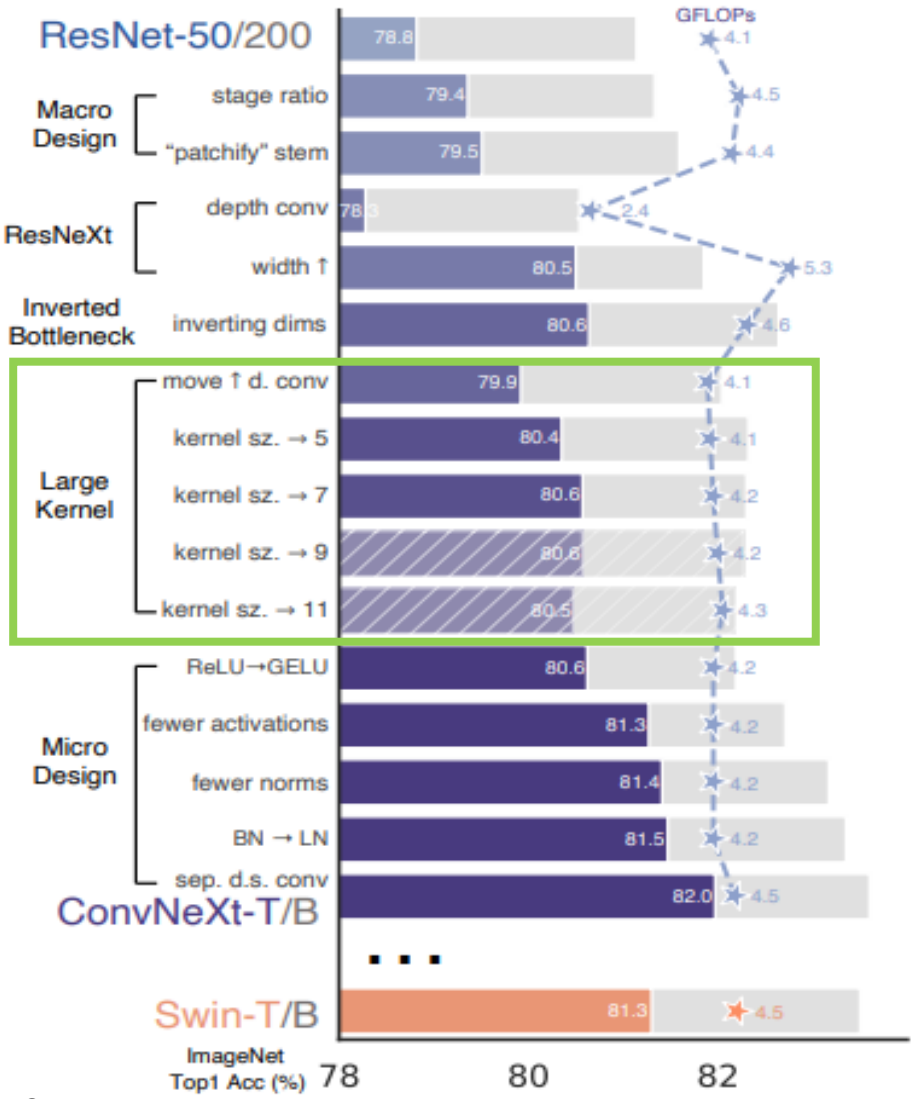
(a) Residual block (b) Inverted residual block



서울아산병원
San Medical Center

3. Modernizing a ConvNet(Large Kernel)

✓ Large Kernel

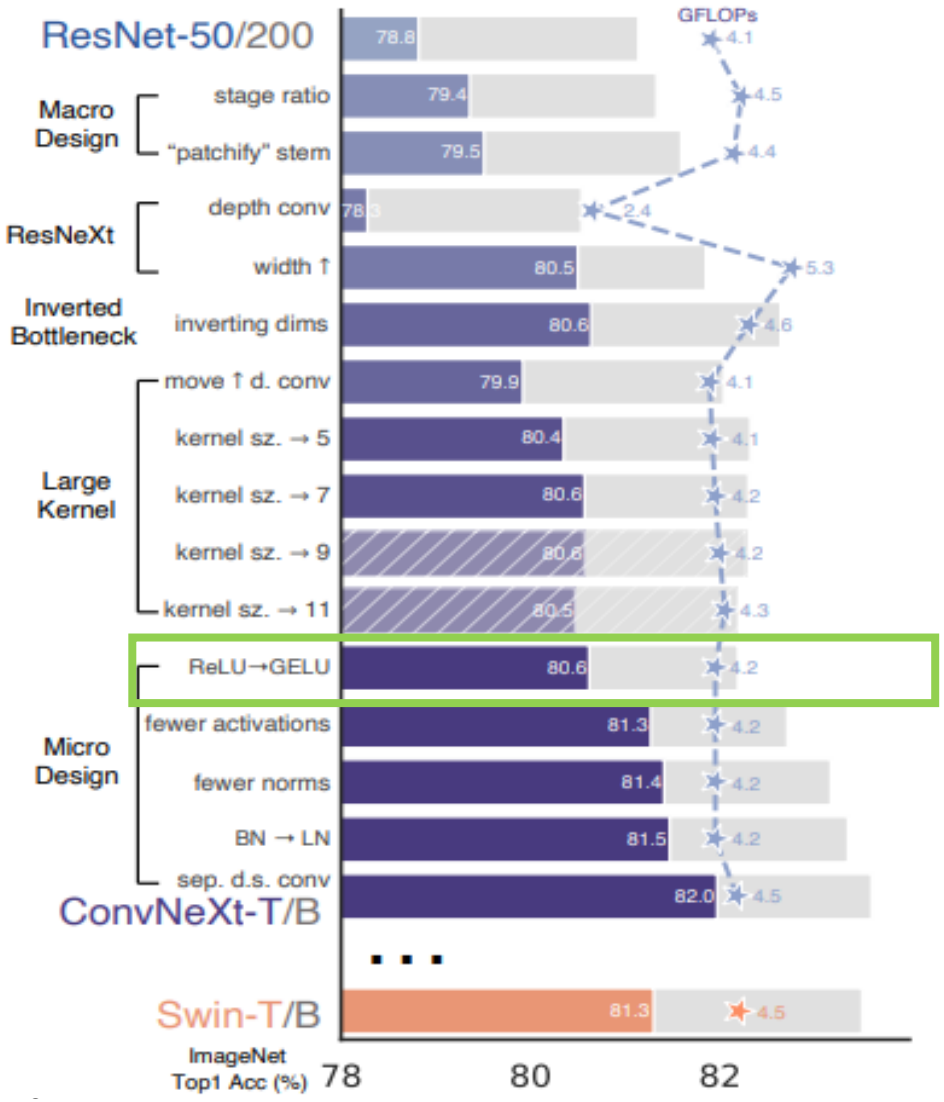


- VGG 부터 3x3 kernel size가 일반적으로 사용되었지만 Swin Transformer 에 맞게 7x7로 수정

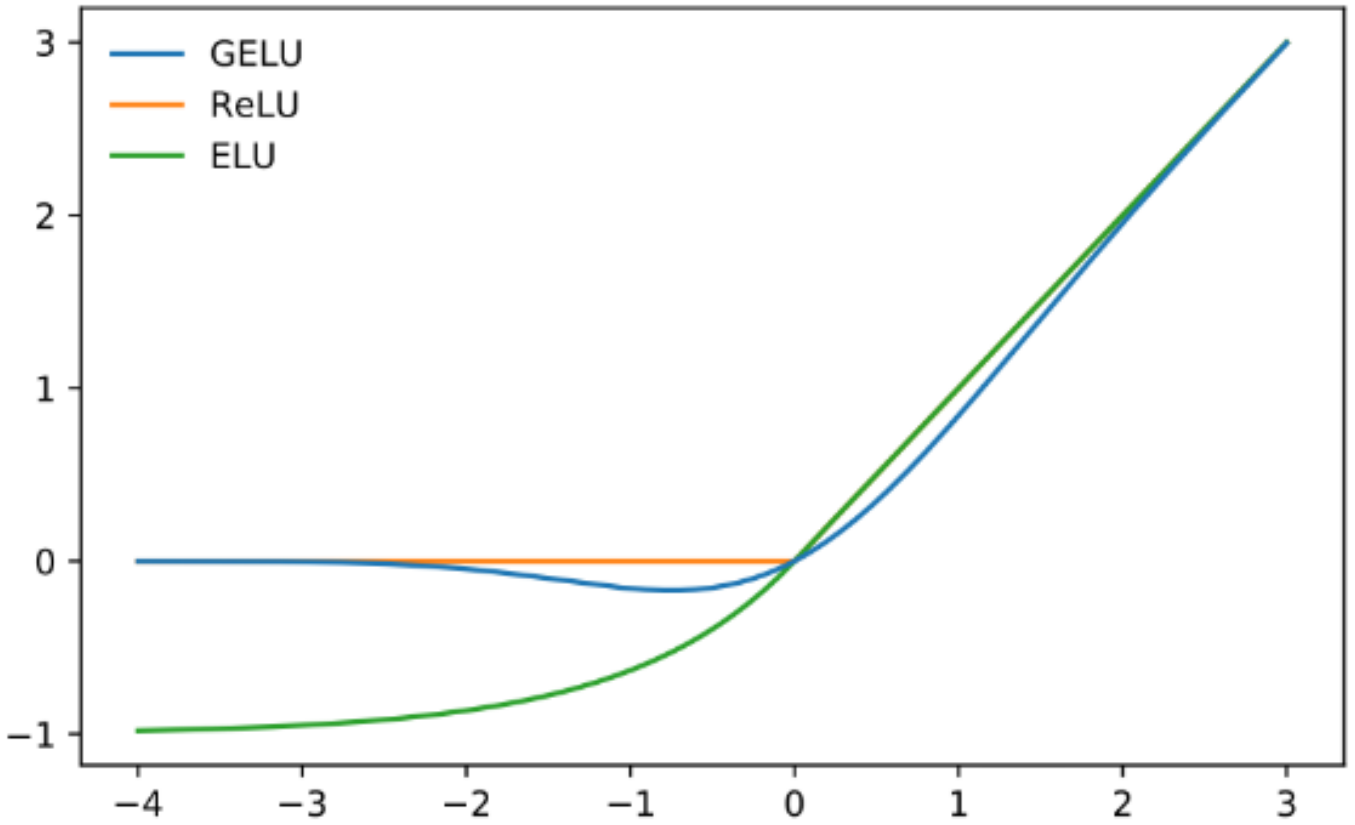
	output size	• ResNet-50	• ConvNeXt-T	○ Swin-T
stem	56×56	7×7, 64, stride 2 3×3 max pool, stride 2	4×4, 96, stride 4	4×4, 96, stride 4
res2	56×56	[1×1, 64 3×3, 64 1×1, 256] × 3	[d7×7, 96 1×1, 384 1×1, 96] × 3	[1×1, 96×3 MSA, w7×7, H=3, rel. pos. 1×1, 96 1×1, 384 1×1, 96] × 2
res3	28×28	[1×1, 128 3×3, 128 1×1, 512] × 4	[d7×7, 192 1×1, 768 1×1, 192] × 3	[1×1, 192×3 MSA, w7×7, H=6, rel. pos. 1×1, 192 1×1, 768 1×1, 192] × 2
res4	14×14	[1×1, 256 3×3, 256 1×1, 1024] × 6	[d7×7, 384 1×1, 1536 1×1, 384] × 9	[1×1, 384×3 MSA, w7×7, H=12, rel. pos. 1×1, 384 1×1, 1536 1×1, 384] × 6
res5	7×7	[1×1, 512 3×3, 512 1×1, 2048] × 3	[d7×7, 768 1×1, 3072 1×1, 768] × 3	[1×1, 768×3 MSA, w7×7, H=24, rel. pos. 1×1, 768 1×1, 3072 1×1, 768] × 2
FLOPs		4.1×10^9	4.5×10^9	4.5×10^9
# params.		25.6×10^6	28.6×10^6	28.3×10^6

3. Modernizing a ConvNet(Micro Design)

✓ Micro Design



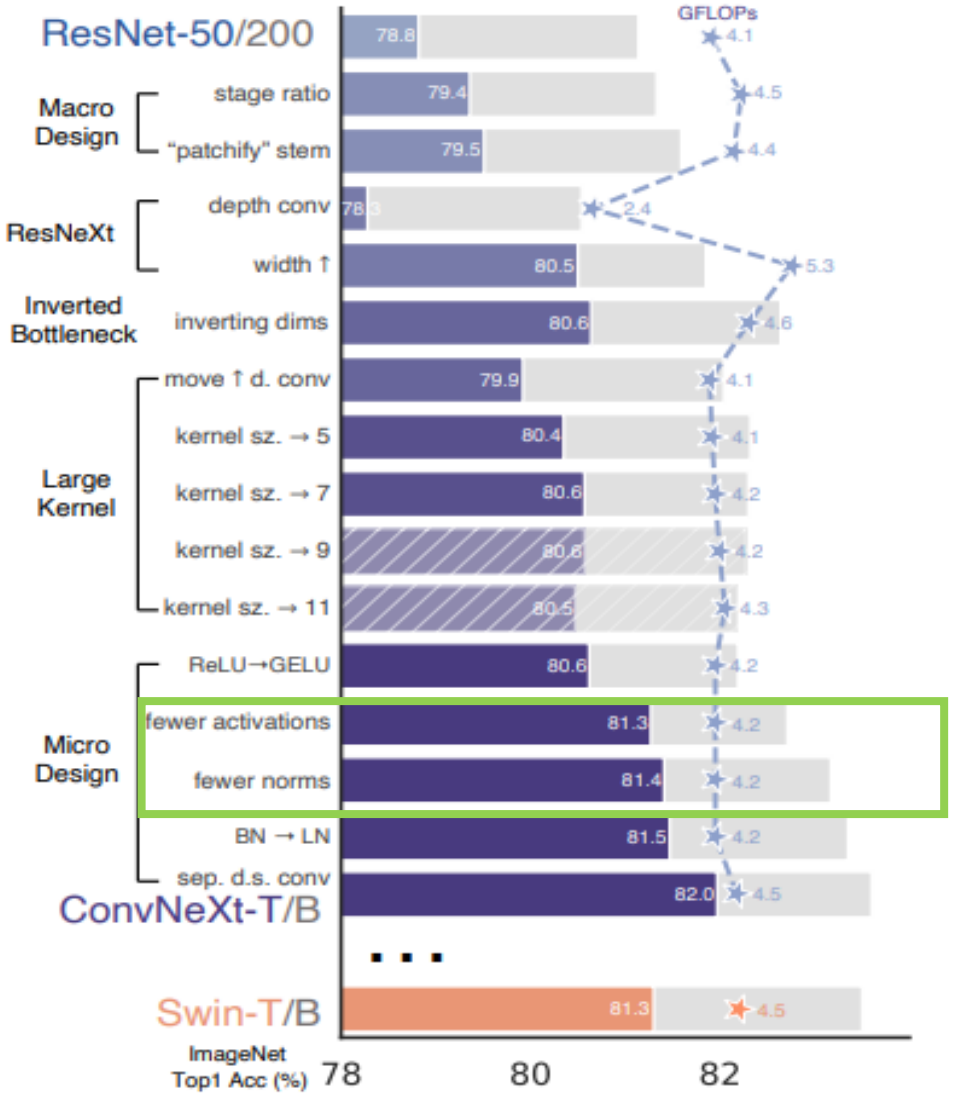
- ReLU -> GELU (Gaussian Error Linear Unit)
- 기존 Transformer에도 ReLU가 사용되었지만 BERT 이후로 GELU 로 모두 대체



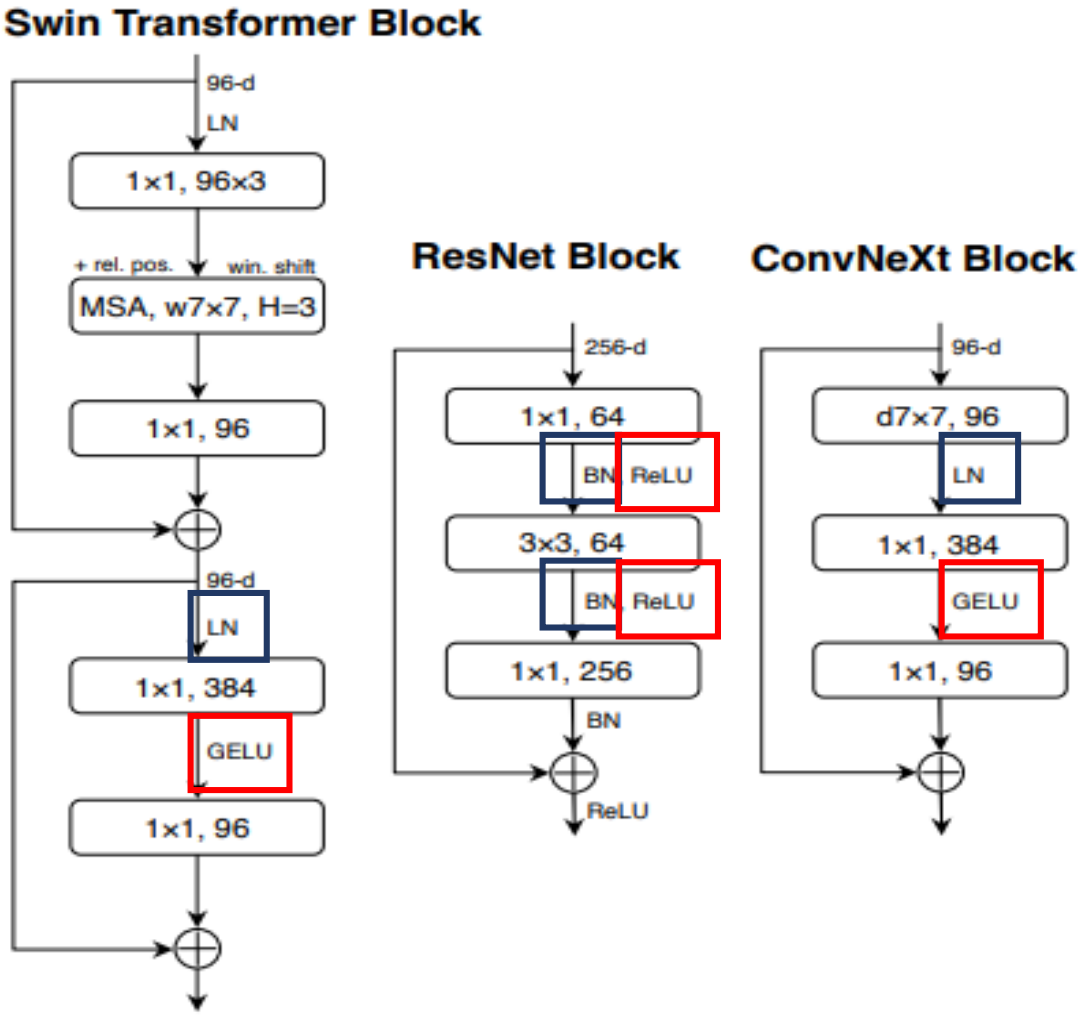
울산대학교 UNIVERSITY OF ULSAN 서울아산병원 Asan Medical Center

3. Modernizing a ConvNet(Micro Design)

✓ Micro Design

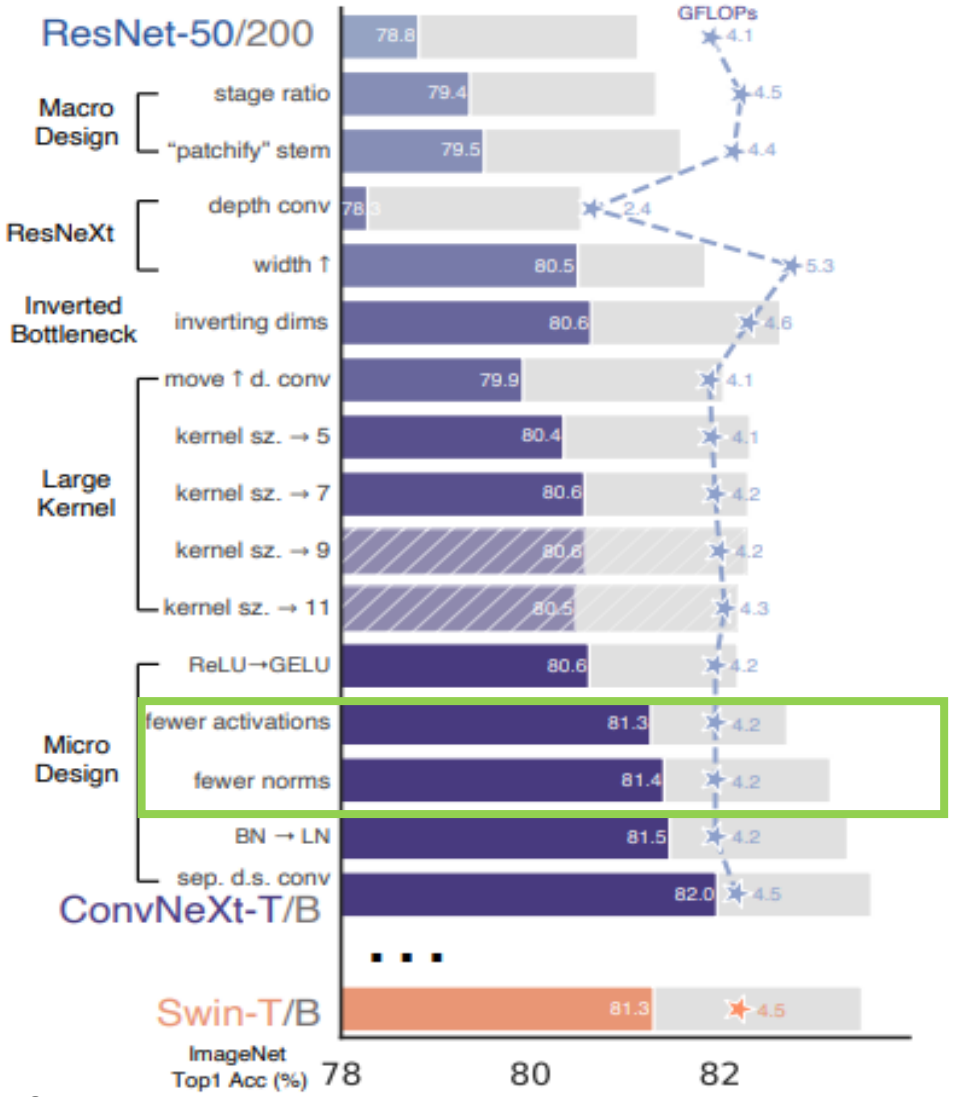


- Activation / Normalization 을 layer 마다 한번만 적용

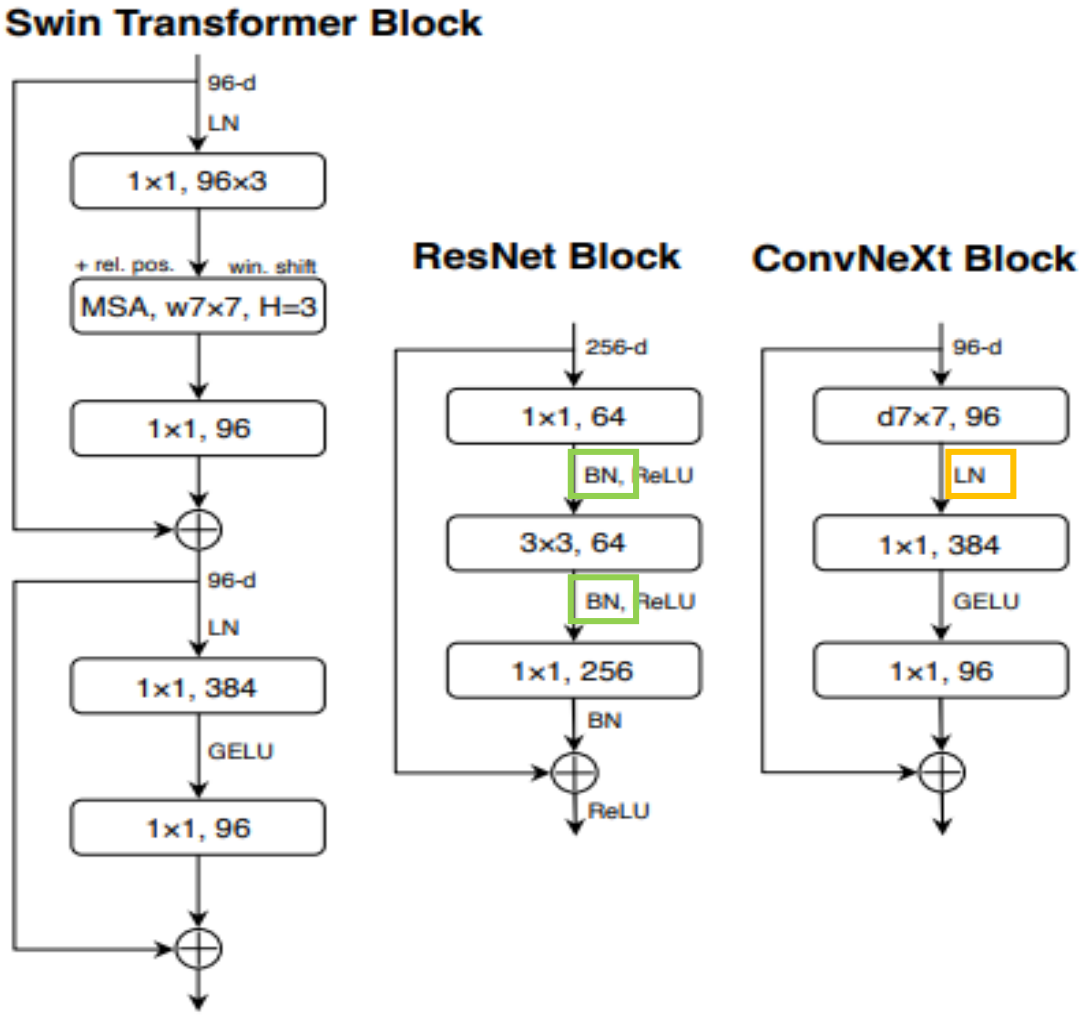


3. Modernizing a ConvNet(Micro Design)

✓ Micro Design

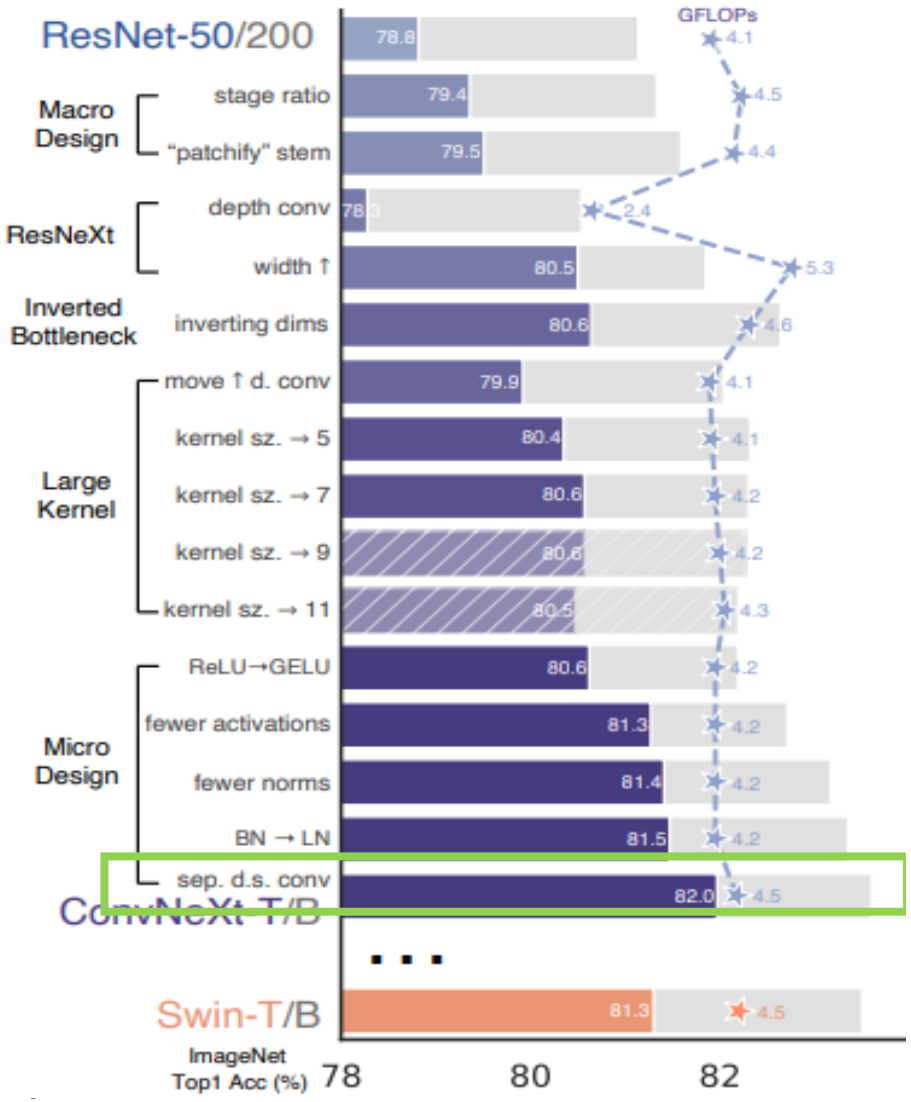


- Batch Normalization -> Layer Normalization으로 변경



3. Modernizing a ConvNet(Micro Design)

✓ Micro Design



- 매번 Convolution Layer 이후 down-sampling 하는 방식에서 Stage 마다 down-Sampling 하는 방식으로 변경

	output size	• ResNet-50	• ConvNeXt-T	○ Swin-T
stem	56×56	7×7, 64, stride 2 3×3 max pool, stride 2	4×4, 96, stride 4	4×4, 96, stride 4
res2	56×56	[1×1, 64 3×3, 64 1×1, 256] × 3	[d7×7, 96 1×1, 384 1×1, 96] × 3	[1×1, 96×3 MSA, w7×7, H=3, rel. pos. 1×1, 96 1×1, 384 1×1, 96] × 2
res3	28×28	[1×1, 128 3×3, 128 1×1, 512] × 4	[d7×7, 192 1×1, 768 1×1, 192] × 3	[1×1, 192×3 MSA, w7×7, H=6, rel. pos. 1×1, 192 1×1, 768 1×1, 192] × 2
res4	14×14	[1×1, 256 3×3, 256 1×1, 1024] × 6	[d7×7, 384 1×1, 1536 1×1, 384] × 9	[1×1, 384×3 MSA, w7×7, H=12, rel. pos. 1×1, 384 1×1, 1536 1×1, 384] × 6
res5	7×7	[1×1, 512 3×3, 512 1×1, 2048] × 3	[d7×7, 768 1×1, 3072 1×1, 768] × 3	[1×1, 768×3 MSA, w7×7, H=24, rel. pos. 1×1, 768 1×1, 3072 1×1, 768] × 2
FLOPs		4.1×10^9	4.5×10^9	4.5×10^9
# params.		25.6×10^6	28.6×10^6	28.3×10^6

Down sampling

Down sampling

Down sampling

서울아산병원 Asan Medical Center

4. Experiment

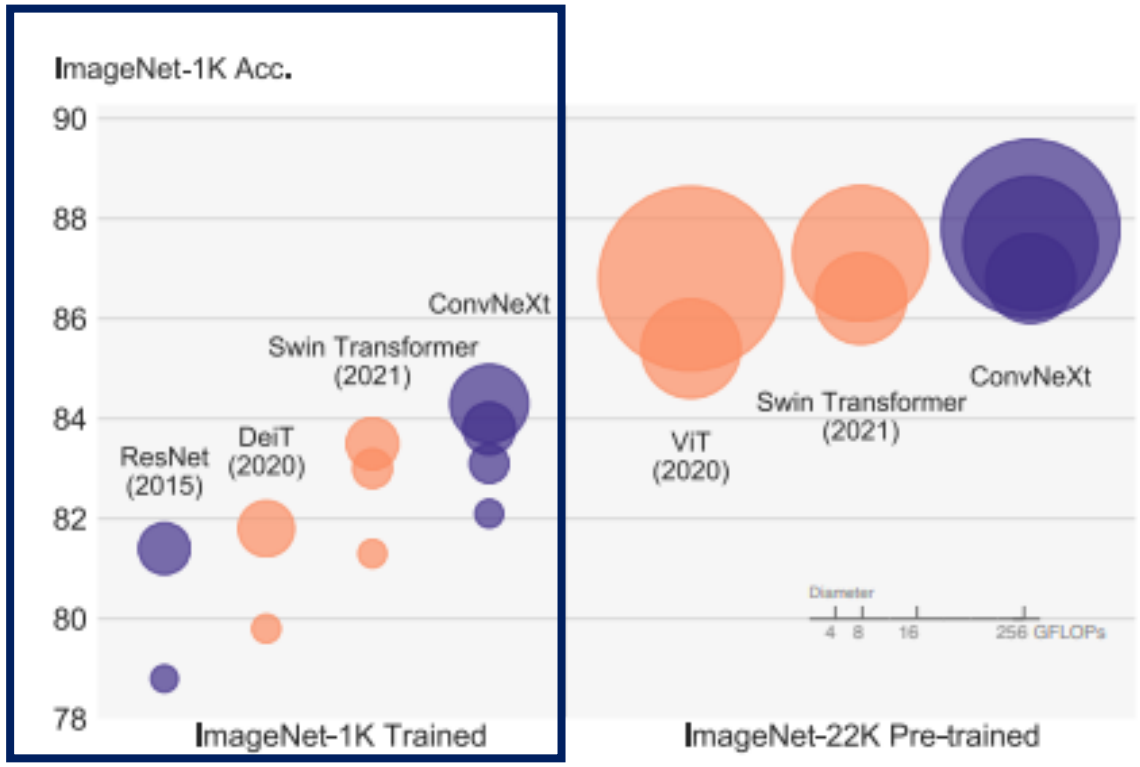
ConvNet

- ViT 기반 모델과 동일하게 dimension & block 구성에 따라 모델을 구성

- 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: $C = (128, 256, 512, 1024), B = (3, 3, 27, 3)$
- ConvNeXt-L: $C = (192, 384, 768, 1536), B = (3, 3, 27, 3)$
- ConvNeXt-XL: $C = (256, 512, 1024, 2048), B = (3, 3, 27, 3)$

4. Experiment

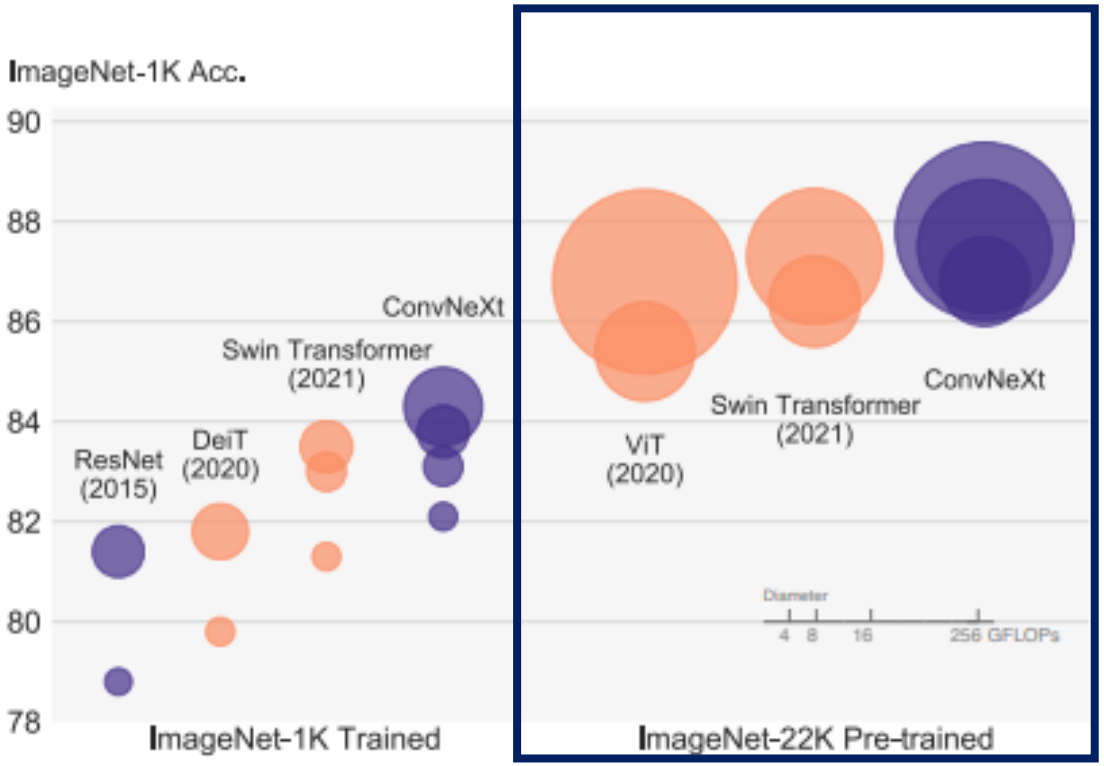
ImageNet-1K



model	image size	#param.	FLOPs	throughput (image / s)	IN-1K top-1 acc.
ImageNet-1K trained models					
• RegNetY-16G [54]	224^2	84M	16.0G	334.7	82.9
• EffNet-B7 [71]	600^2	66M	37.0G	55.1	84.3
• EffNetV2-L [72]	480^2	120M	53.0G	83.7	85.7
○ DeiT-S [73]	224^2	22M	4.6G	978.5	79.8
○ DeiT-B [73]	224^2	87M	17.6G	302.1	81.8
○ Swin-T	224^2	28M	4.5G	757.9	81.3
• ConvNeXt-T	224^2	29M	4.5G	774.7	**82.1**
○ Swin-S	224^2	50M	8.7G	436.7	83.0
• ConvNeXt-S	224^2	50M	8.7G	447.1	**83.1**
○ Swin-B	224^2	88M	15.4G	286.6	83.5
• ConvNeXt-B	224^2	89M	15.4G	292.1	**83.8**
○ Swin-B	384^2	88M	47.1G	85.1	84.5
• ConvNeXt-B	384^2	89M	45.0G	95.7	**85.1**
• ConvNeXt-L	224^2	198M	34.4G	146.8	**84.3**
• ConvNeXt-L	384^2	198M	101.0G	50.4	**85.5**

4. Experiment

ImageNet-22K



ImageNet-22K pre-trained models					
• R-101x3 [39]	384^2	388M	204.6G	-	84.4
• R-152x4 [39]	480^2	937M	840.5G	-	85.4
• EffNetV2-L [72]	480^2	120M	53.0G	83.7	86.8
• EffNetV2-XL [72]	480^2	208M	94.0G	56.5	87.3
○ ViT-B/16 (☎) [67]	384^2	87M	55.5G	93.1	85.4
○ ViT-L/16 (☎) [67]	384^2	305M	191.1G	28.5	86.8
• ConvNeXt-T	224^2	29M	4.5G	774.7	**82.9**
• ConvNeXt-T	384^2	29M	13.1G	282.8	**84.1**
• ConvNeXt-S	224^2	50M	8.7G	447.1	**84.6**
• ConvNeXt-S	384^2	50M	25.5G	163.5	**85.8**
○ Swin-B	224^2	88M	15.4G	286.6	85.2
• ConvNeXt-B	224^2	89M	15.4G	292.1	**85.8**
○ Swin-B	384^2	88M	47.0G	85.1	86.4
• ConvNeXt-B	384^2	89M	45.1G	95.7	**86.8**
○ Swin-L	224^2	197M	34.5G	145.0	86.3
• ConvNeXt-L	224^2	198M	34.4G	146.8	**86.6**
○ Swin-L	384^2	197M	103.9G	46.0	87.3
• ConvNeXt-L	384^2	198M	101.0G	50.4	**87.5**
• ConvNeXt-XL	224^2	350M	60.9G	89.3	**87.0**
• ConvNeXt-XL	384^2	350M	179.0G	30.2	**87.8**

4. Experiment

Isotropic ConvNeXt vs ViT

- ViT와 동일하게 down-sampling 없이 비교 시 거의 동일한 성능 대비 적은 연산량(FLOPs)을 보여줌

model	#param.	FLOPs	throughput (image / s)	training mem. (GB)	IN-1K acc.
○ ViT-S	22M	4.6G	978.5	4.9	79.8
● ConvNeXt-S (*iso.*)	22M	4.3G	1038.7	4.2	79.7
○ ViT-B	87M	17.6G	302.1	9.1	81.8
● ConvNeXt-B (*iso.*)	87M	16.9G	320.1	7.7	82.0
○ ViT-L	304M	61.6G	93.1	22.5	82.6
● ConvNeXt-L (*iso.*)	306M	59.7G	94.4	20.4	82.6

4. Experiment

Robustness

- 여러 ImageNet-C, A, R, SK의 Robustness 평가에 대해 강건함 보임
- ImageNet C에 대해서는 mCE(mean Corruption Error)로 평가

Model	Data/Size	FLOPs / Params	Clean	C (↓)	$\bar{C}$ (↓)	A	R	SK
ResNet-50	$1K/224^2$	4.1 / 25.6	76.1	76.7	57.7	0.0	36.1	24.1
Swin-T [45]	$1K/224^2$	4.5 / 28.3	81.2	62.0	-	21.6	41.3	29.1
RVT-S* [47]	$1K/224^2$	4.7 / 23.3	81.9	49.4	37.5	25.7	47.7	34.7
ConvNeXt-T	$1K/224^2$	4.5 / 28.6	82.1	53.2	40.0	24.2	47.2	33.8
Swin-B [45]	$1K/224^2$	15.4 / 87.8	83.4	54.4	-	35.8	46.6	32.4
RVT-B* [47]	$1K/224^2$	17.7 / 91.8	82.6	46.8	**30.8**	28.5	48.7	36.0
ConvNeXt-B	$1K/224^2$	15.4 / 88.6	**83.8**	**46.8**	34.4	**36.7**	**51.3**	**38.2**
ConvNeXt-B	$22K/384^2$	45.1 / 88.6	86.8	43.1	30.7	62.3	64.9	51.6
ConvNeXt-L	$22K/384^2$	101.0 / 197.8	87.5	40.2	29.9	65.5	66.7	52.8
ConvNeXt-XL	$22K/384^2$	179.0 / 350.2	**87.8**	**38.8**	**27.1**	**69.3**	**68.2**	**55.0**

Table 8. **Robustness evaluation of ConvNeXt.** We do not make use of any specialized modules or additional fine-tuning procedures.

4. Experiment

Inference

- Swin Transformer에 비해 추론 속도에서 더 빠른 처리 속도를 보임

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.

감사합니다.