# A ConvNet for the 2020s - ConvNeXt

## Facebook AI Research, UC Berkeley - CVPR 2022

Members: **Momal, Sanjay, Kasun, Dane, Mehdi**
CAP-6412, Advanced Computer Vision
Spring 22, CECS, UCF.

# Contents

ConvNeXt - Reviving Convolutions in  a Transformers era.

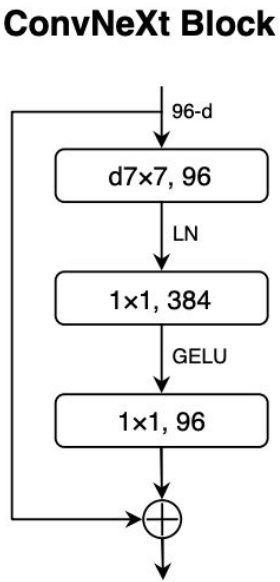

- Introduction
- Modernizing ConvNets
  - Macro Design
  - Micro Design
- Evaluating ConvNeXt
  - on ImageNet
  - on Downstream tasks
- Conclusion

# Introduction & Related Work

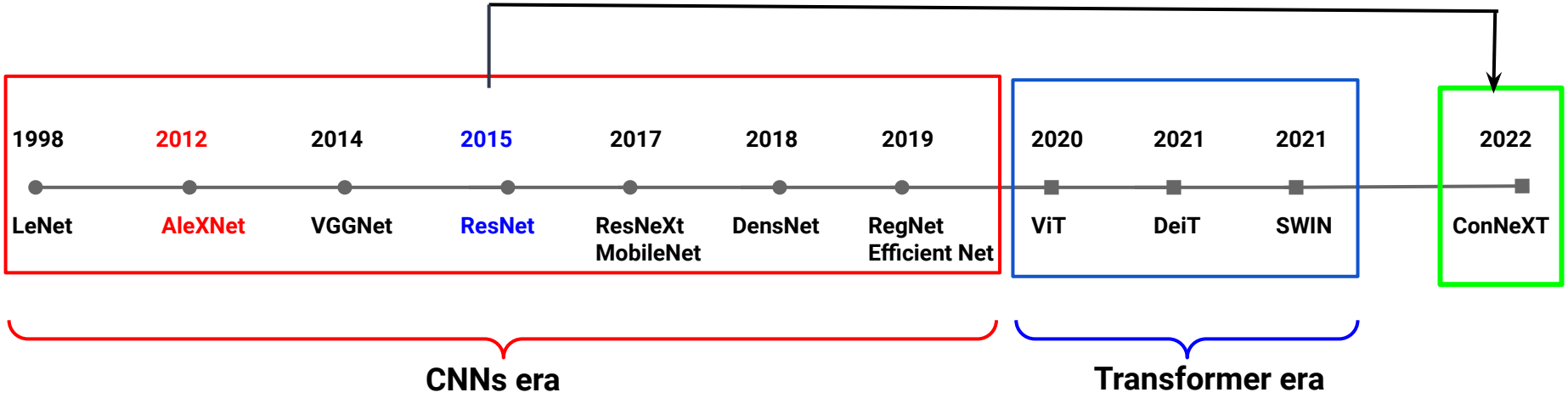
# Introduction

- CNNs can do as well as Transformers

- How?
  - Modernizing ConvNet using a SWIN as motivation
- Propose new architecture: ConvNeXt
  - Outperforms SWIN on detection, classification and segmentation.

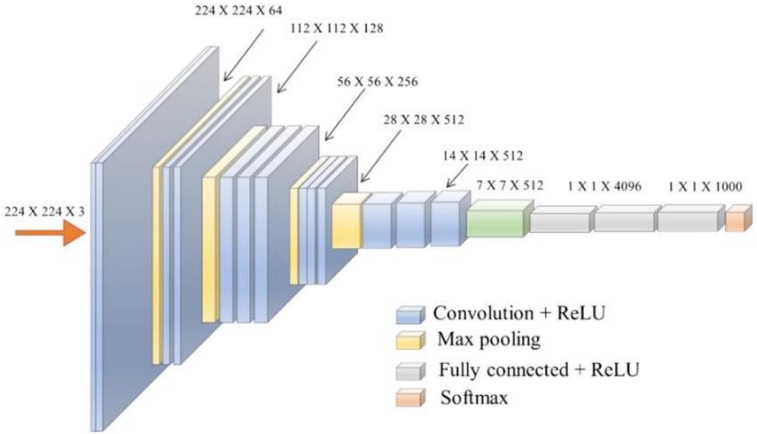


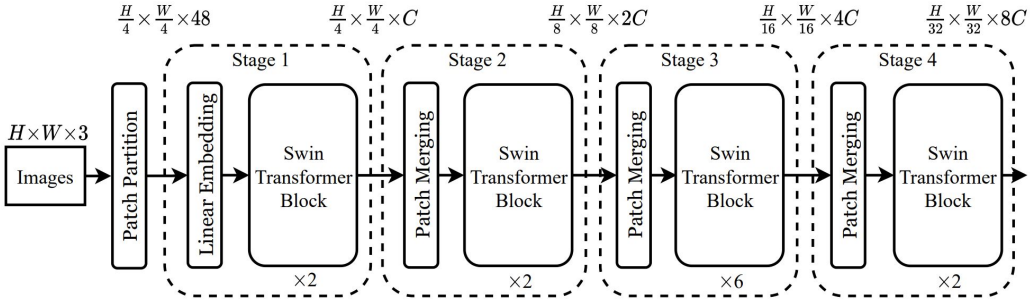
# Evolution of Vision Architectures



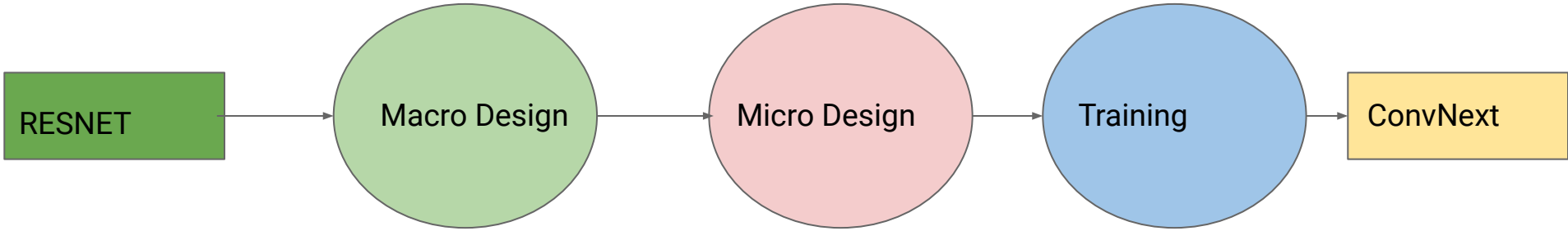
# Hierarchy of SWIN vs. CNNs

- **VIT failed** in common vision tasks
- Hierarchical transformer SWIN
  - introduced as **general purpose backbone**.
- Swin was credited for transformers



# Modernizing ConvNets




Changes to whole architecture based on SWIN-Transformer

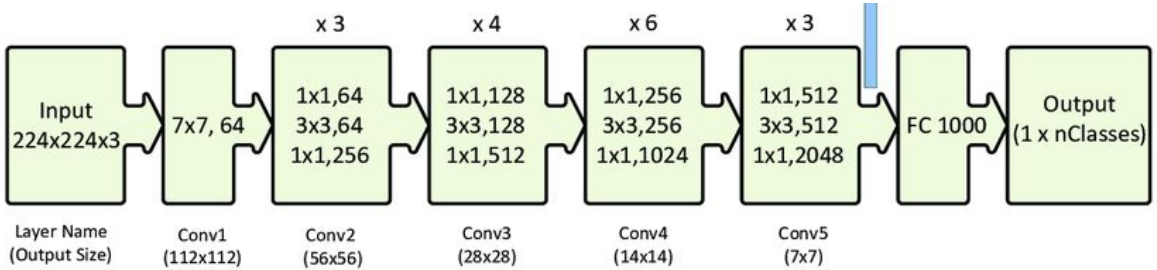
Network layer level changes

Changing Epochs
Changing optimization
New Augmentations
Regularization methods

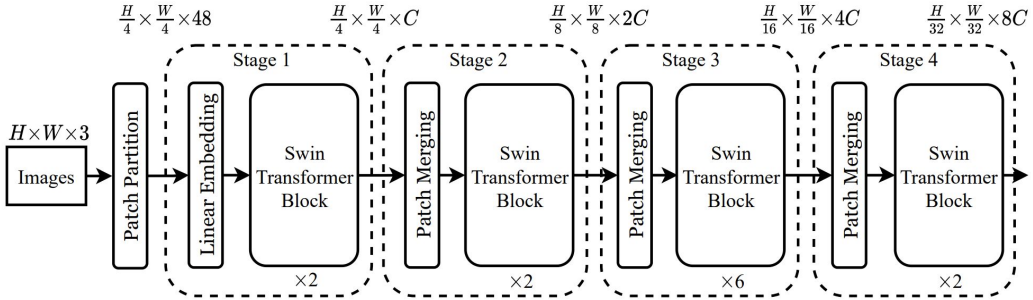
# Modernizing **ConvNets**

# Modernizing ResNet

BaseModel: **ResNet50**

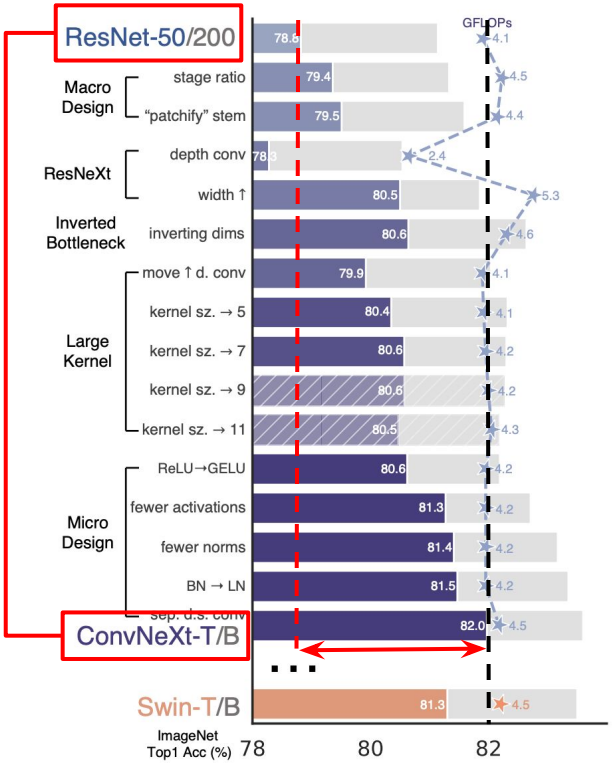



Motivation: **SWIN-Tiny**



# ResNet to ConvNeXt

- **Two** types of changes
  - **Macro** Design Changes
  - **Micro** Design Changes
- Accuracy increases by **4%**
- FLOPs remain under **4.5G**

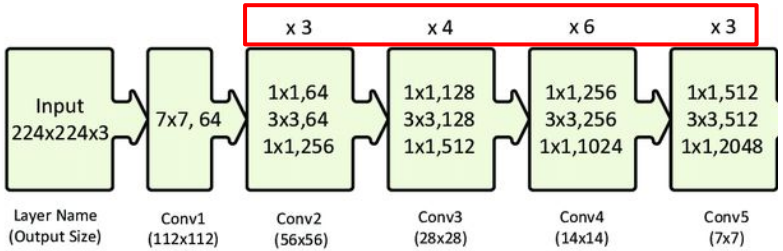


# Macro Design Changes

1. Change stage compute ratio
2. Change stem to Patch-ify
3. Depthwise convolution
4. Inverted bottleneck
5. Larger Kernel Size

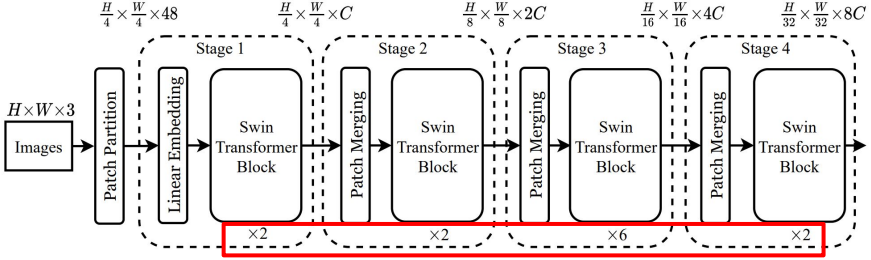
# MD1 : Changing stage compute ratio

**ResNet50**




**Stage compute ratio: 3:4:6:3**

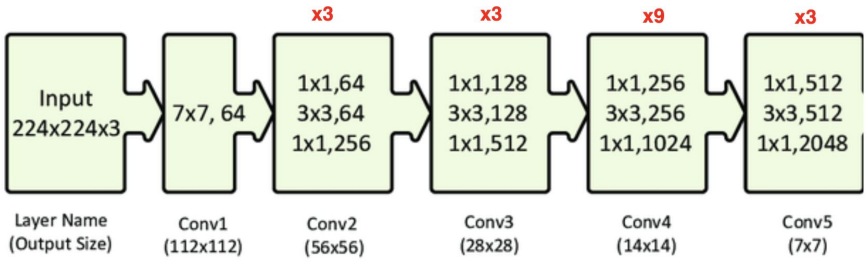
**SWIN Tiny**

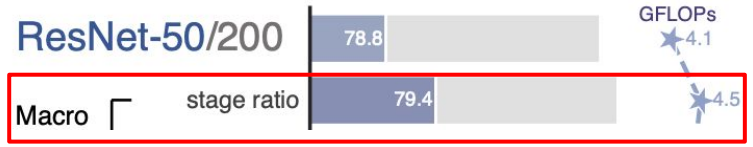



**Stage compute ratio:** 2:2:6:2 **= 1:1:3:1**

# MD1 : Improvements

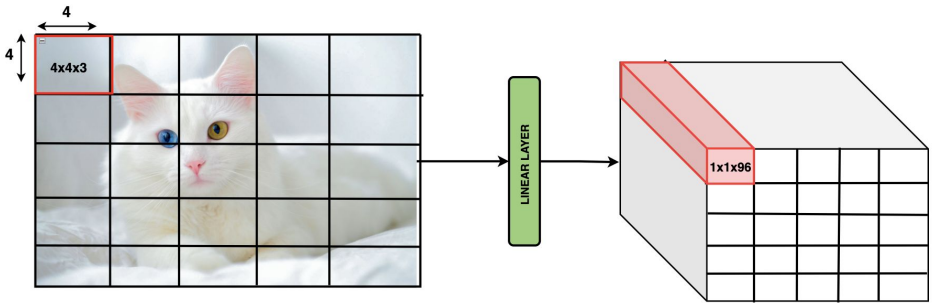
- Change stage compute ratio to **1:1:3:1 like SWIN-Tiny.**
- Accuracy for ResNet-50 increases by **0.6%.**



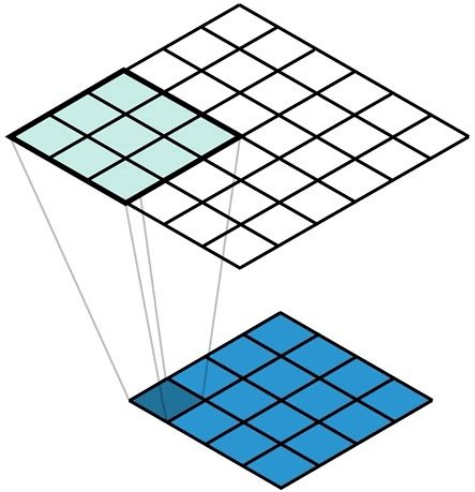



# MD2 : Changing stem to “Patch-ify”

- **ResNet 50**
  - Input stem: ***7x7 conv, stride 2,*** *64 filters*
- **SWIN-Tiny**
  - Input stem: ***4x4 patch, stride 4,*** *96 dim*



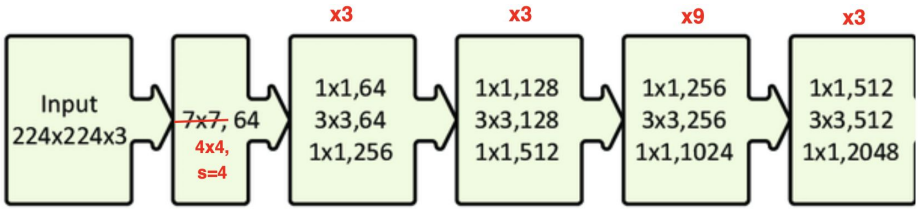

**SWIN-T Stem cell**

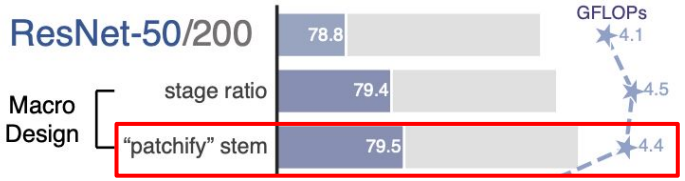


**ResNet-50 Stem cell**

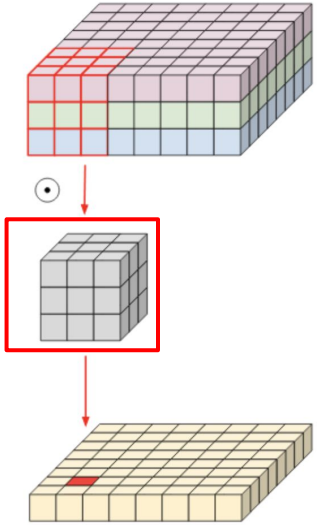
# MD2 : Improvements

- Change stage stem cell to **4x4 conv, with stride = 4.**
- Accuracy for ResNet-50 increases by **0.1%**



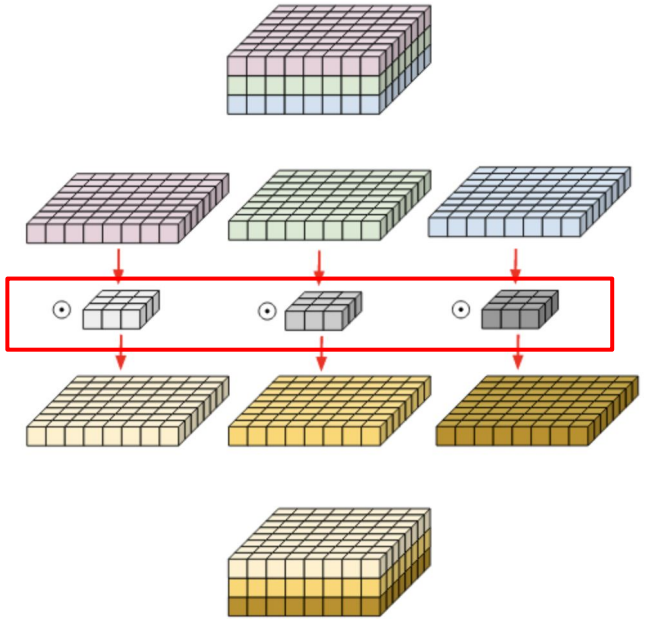



# Depthwise convolutions



Normal convolution

Channel wise filtering

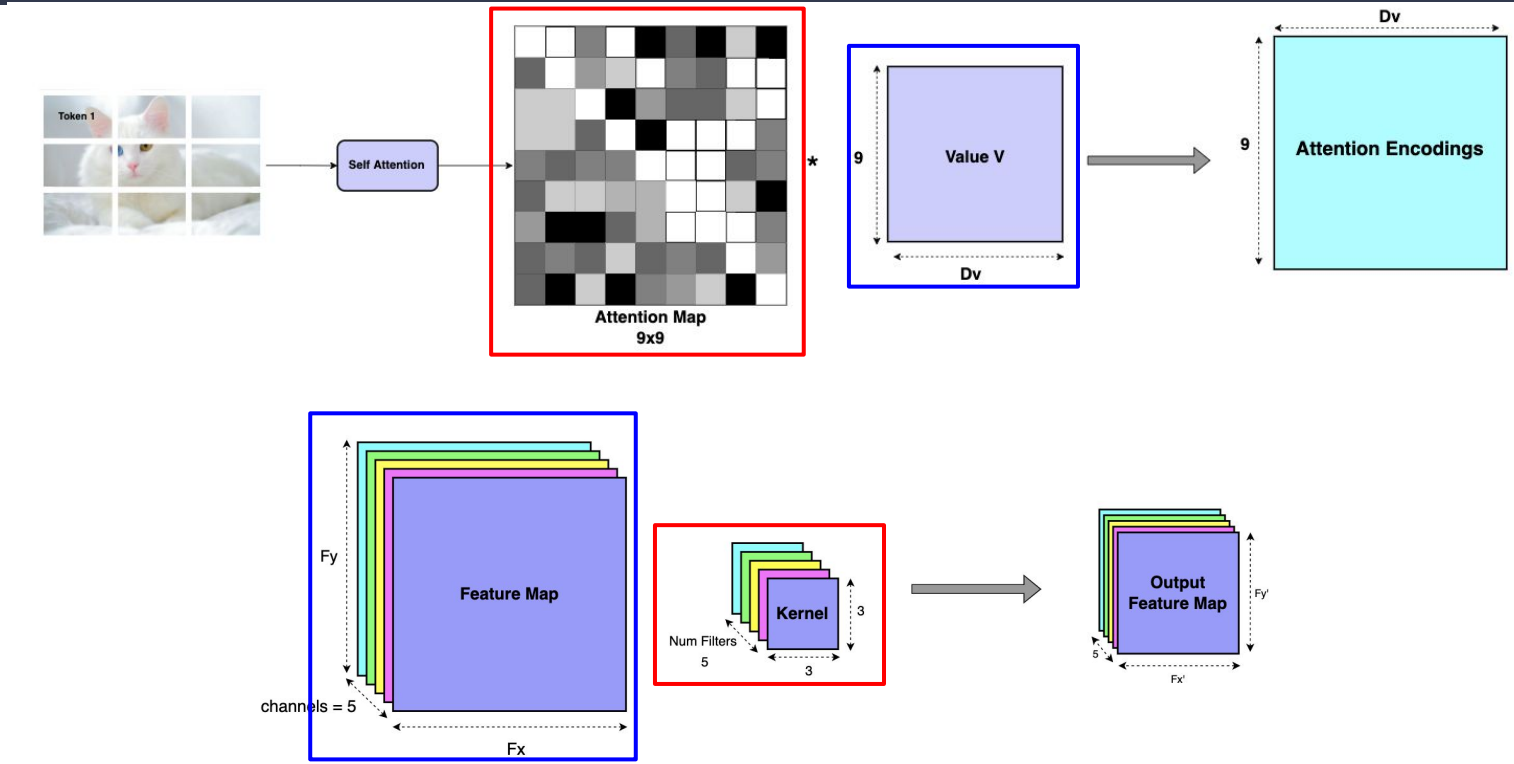


Depthwise convolution

# Why depthwise convolutions?

- **Depthwise convolutions** and **Self Attention** mechanism are similar
    - ***Channel wise Filter application*** similar to ***Weighted sum operations***

# Depthwise Conv. vs Self-Attention



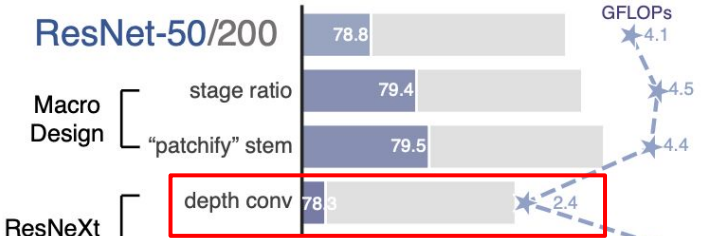
# MD3 : Improvements

- Change all normal conv to **depthwise conv.**

  ➔ **Accuracy drops** by 1.2%, **FLOPs drop** by 2G.

  ➔ Depthwise conv. → **less** parameters

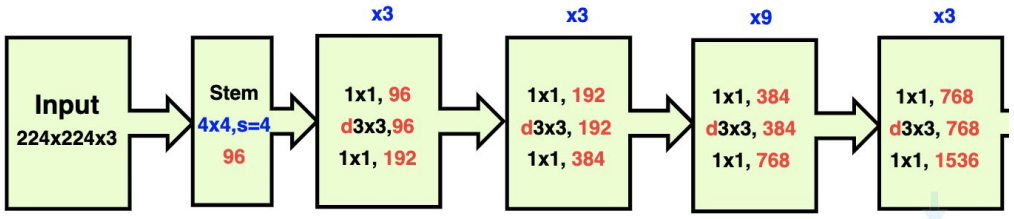


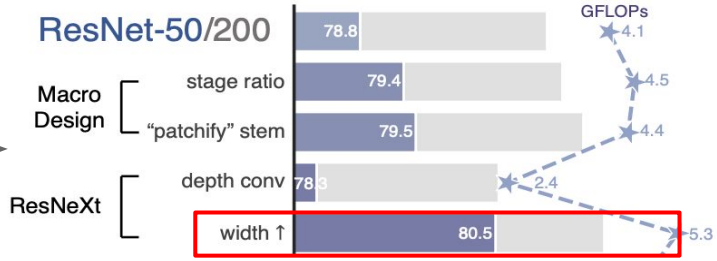



# MD3 : Improvements

- **Increased** Number of filter to **96(SWIN) at stem, channels inc by 2.**

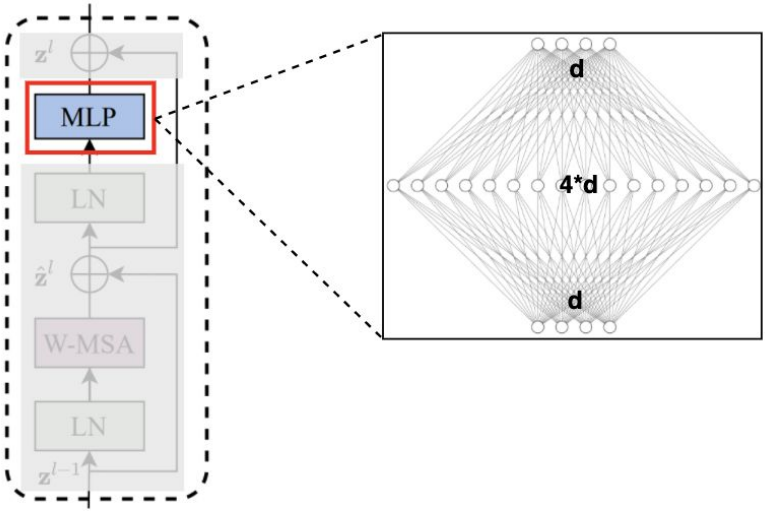
  ➔ **Accuracy inc** by 2.2%, **FLOPs inc** by 2.9G.

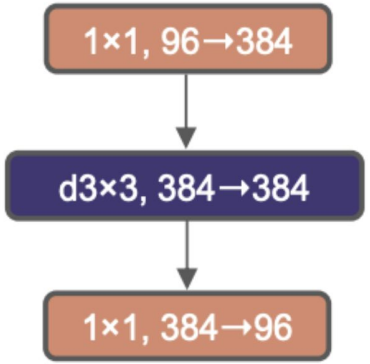


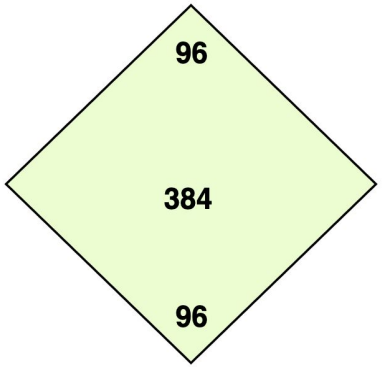
# MD4 : Inverted Bottleneck

- Change the **channel dim ratio to 4**
- **Invert** the **bottleneck** design




New ResNeXt Block

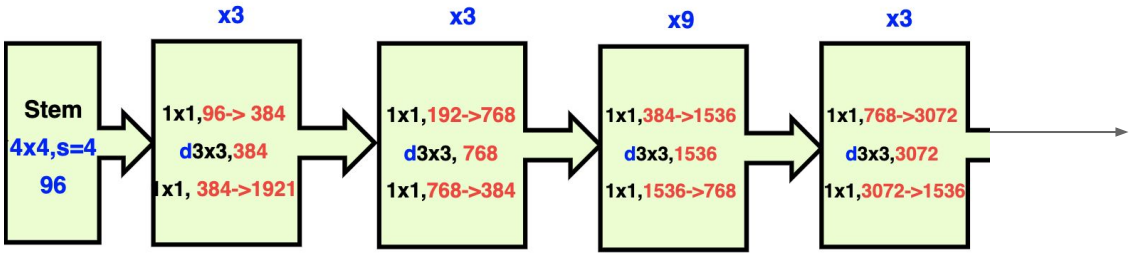


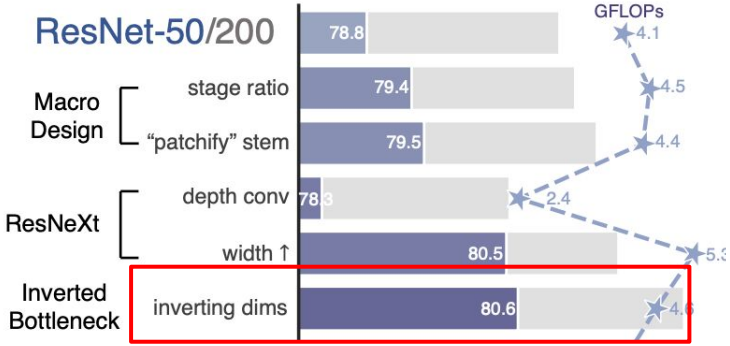

Inverted Bottleneck

# MD4 : Inverted Bottleneck

- Inverting **bottleneck**

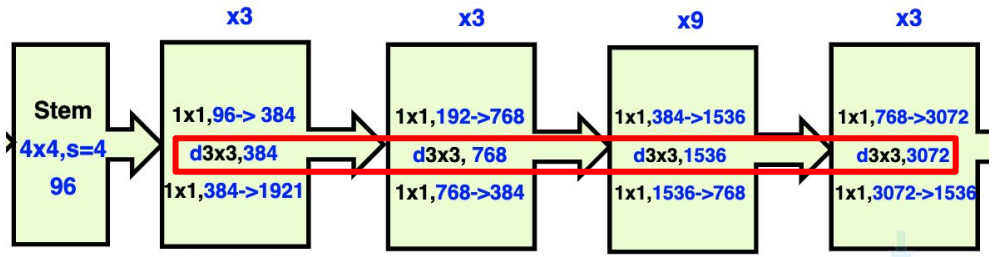
➔ Accuracy improves by **0.1%**



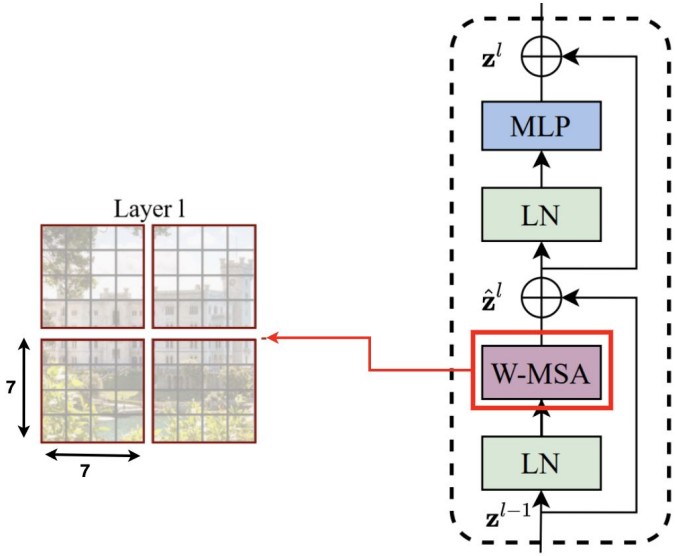
# MD5 : Larger Kernel Sizes

ResNet




Kernel size = 3x3

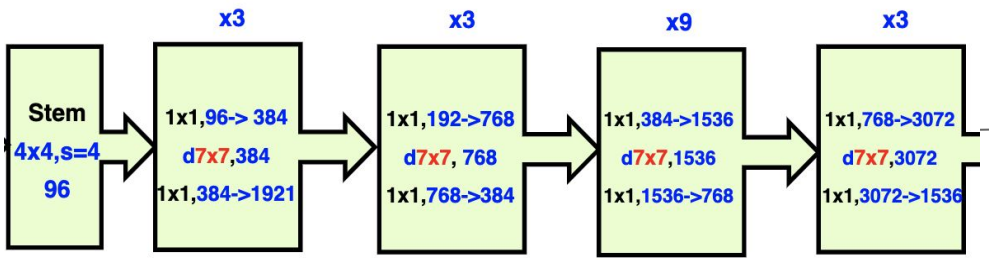
SWIN Tiny

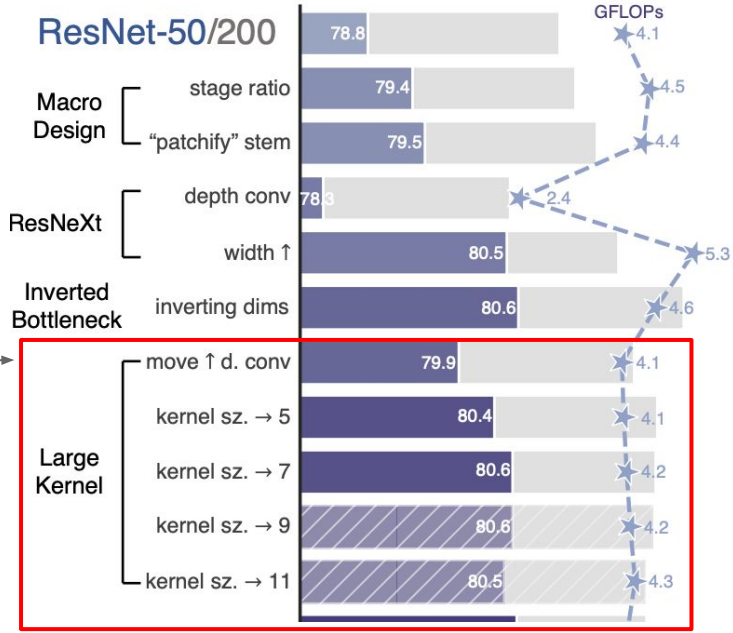



Local Attention window: 7x7

# MD5 : Larger Kernel Sizes

- Change kernel size to **7x7 for all blocks.**
  - ➔ Accuracy **does not improve**

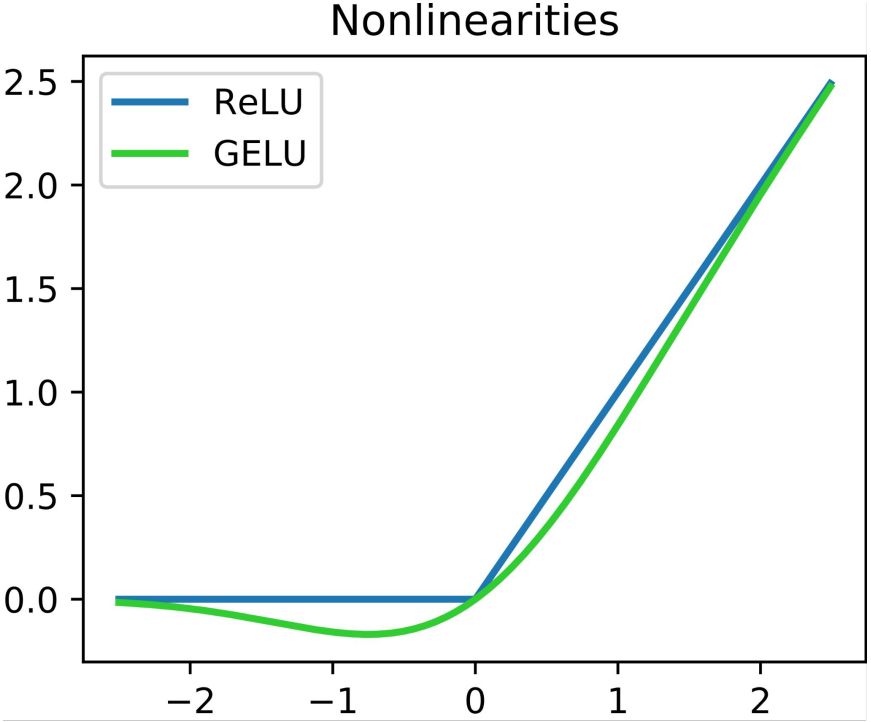


# Micro Designs (mD)

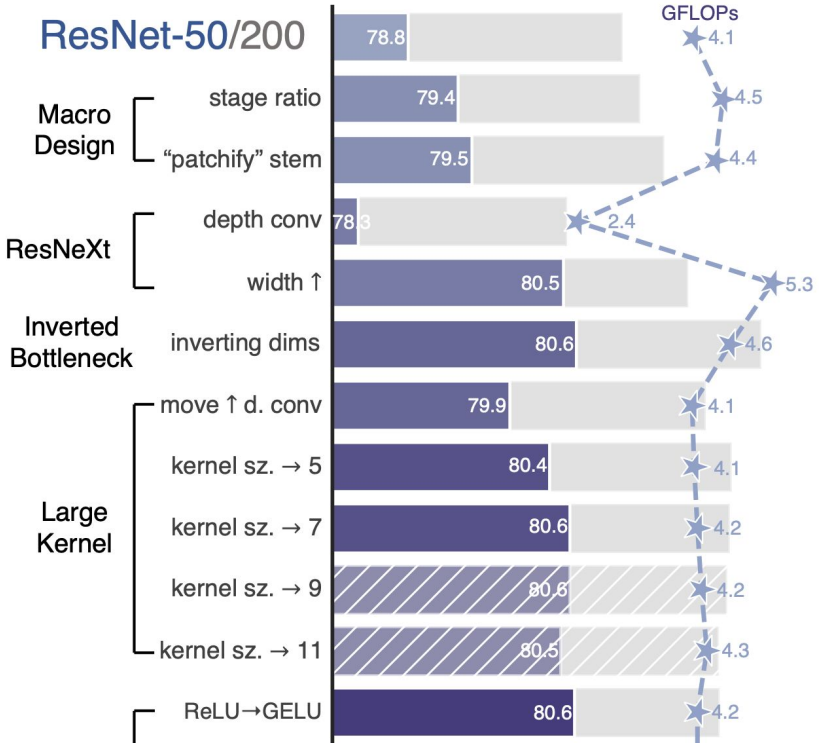
1. Replace RELU with GELU
2. Fewer Activation functions
3. Fewer normalization layers
4. Substituting BN with LN
5. Separate downsampling layers

# mD1: Replace RELU with GELU

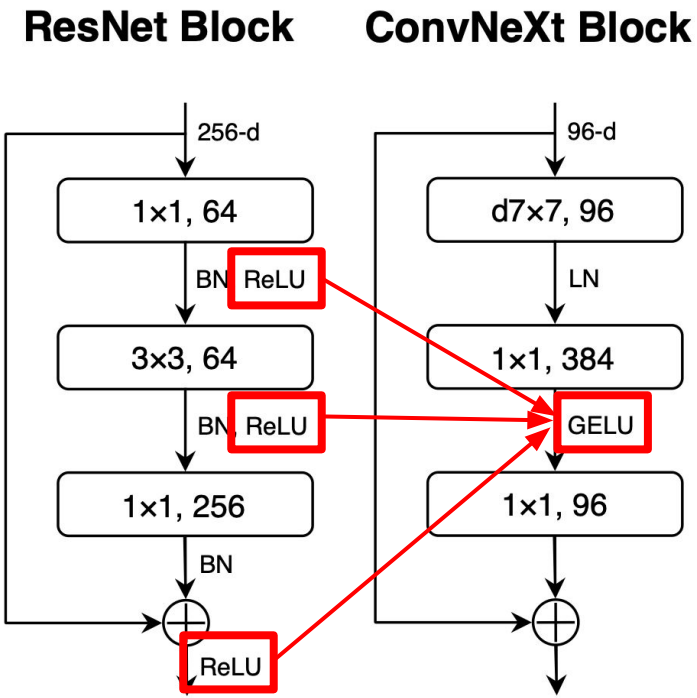



1. BERT
2. GPT-2
3. ViTs

# mD1- Improvement



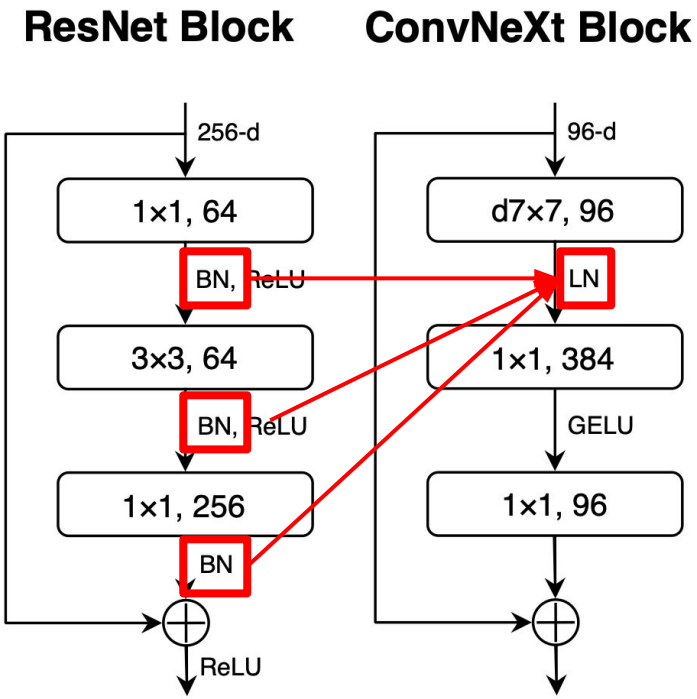
# mD2: Fewer Activation functions



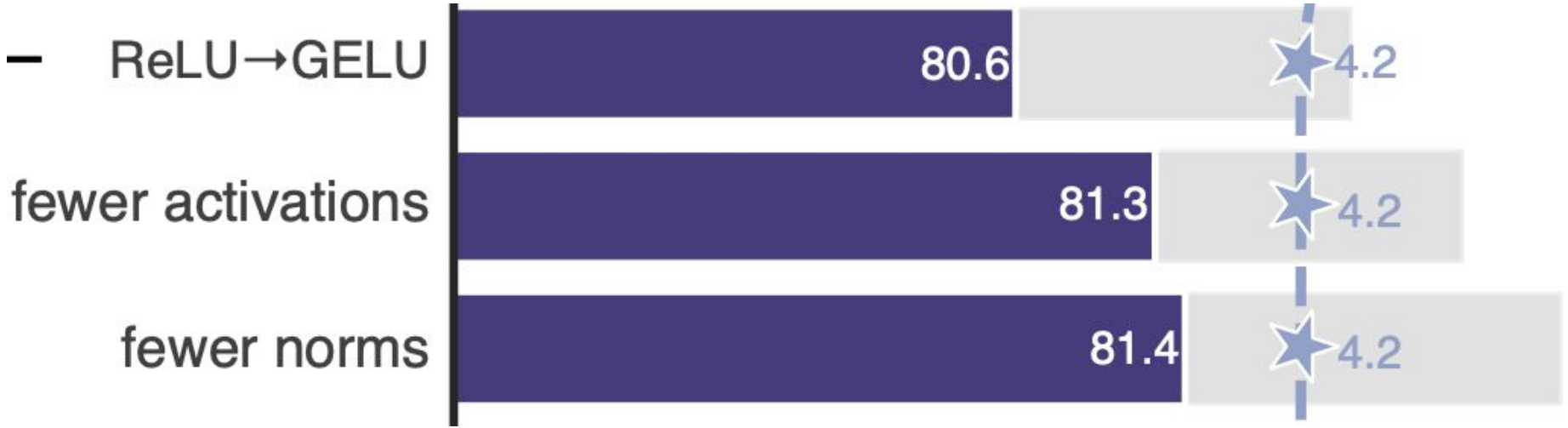
# mD2 - Improvement



# mD3: Fewer Normalization Layers



# mD3- Improvement



# mD4: Substituting BN with LN

**Input:** Values of $x$ over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: $\gamma$, $\beta$
**Output:** $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

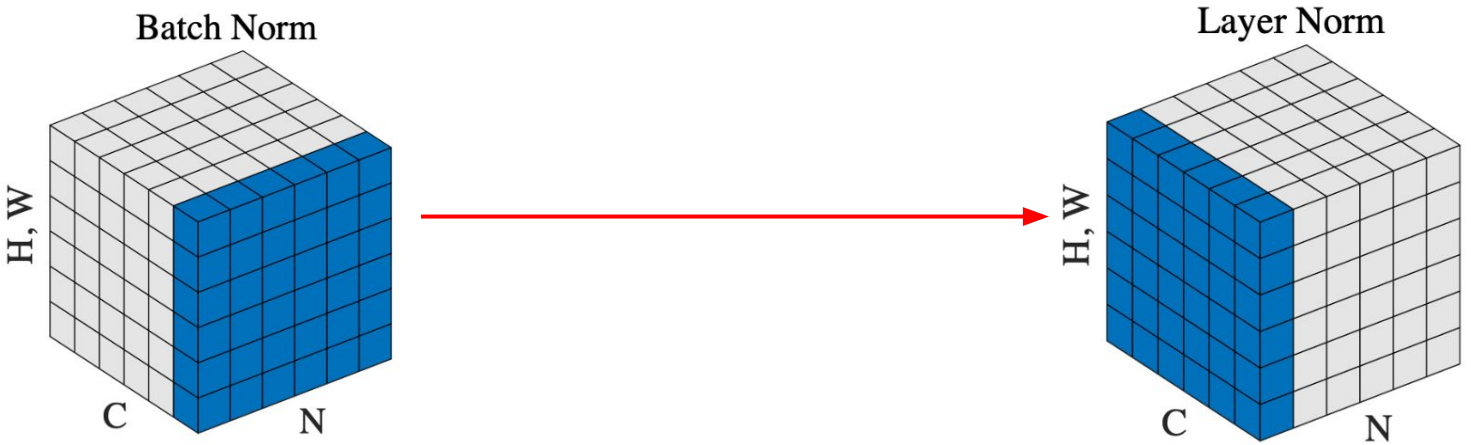
Batch Norm

- $\mu_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij}$
- $\sigma_i^2 = \frac{1}{m}\sum_{j=1}^{m}\left(x_{ij} - \mu_i\right)^2$
- $\hat{x}_{ij} = \frac{x_{ij} - \mu_{ij}}{\sqrt{\sigma_i^2 + \epsilon}}$

Layer Norm

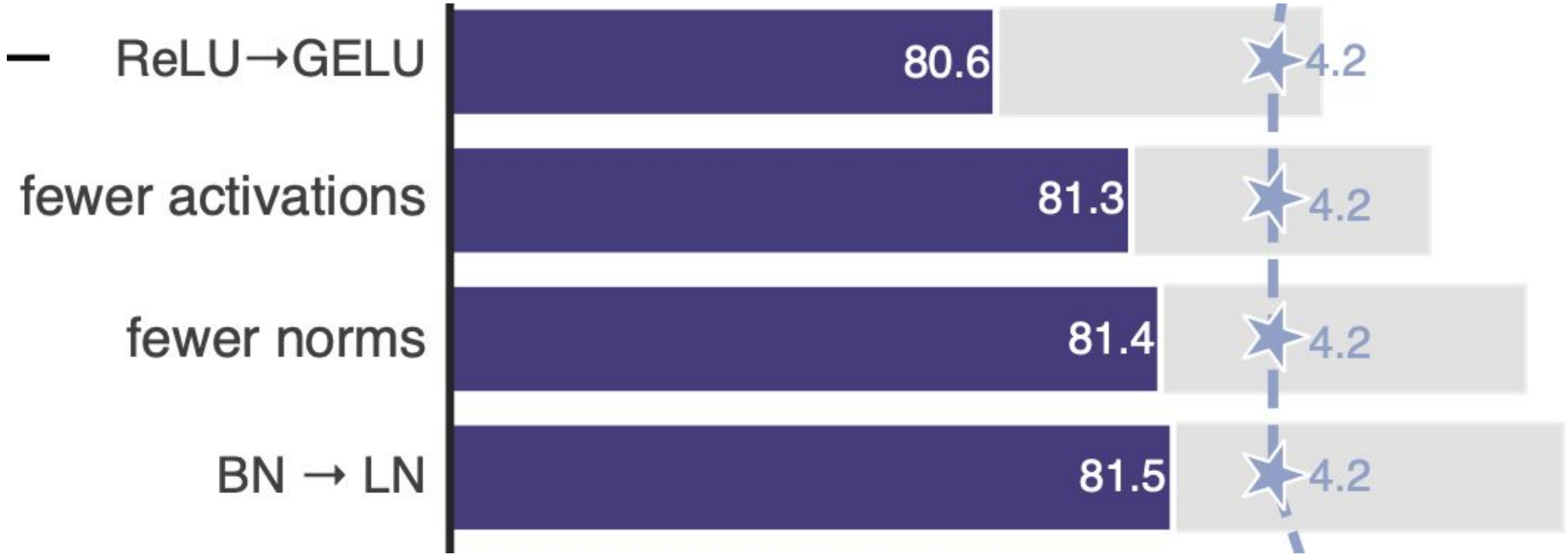
# mD4 - Visualization

Feature Map

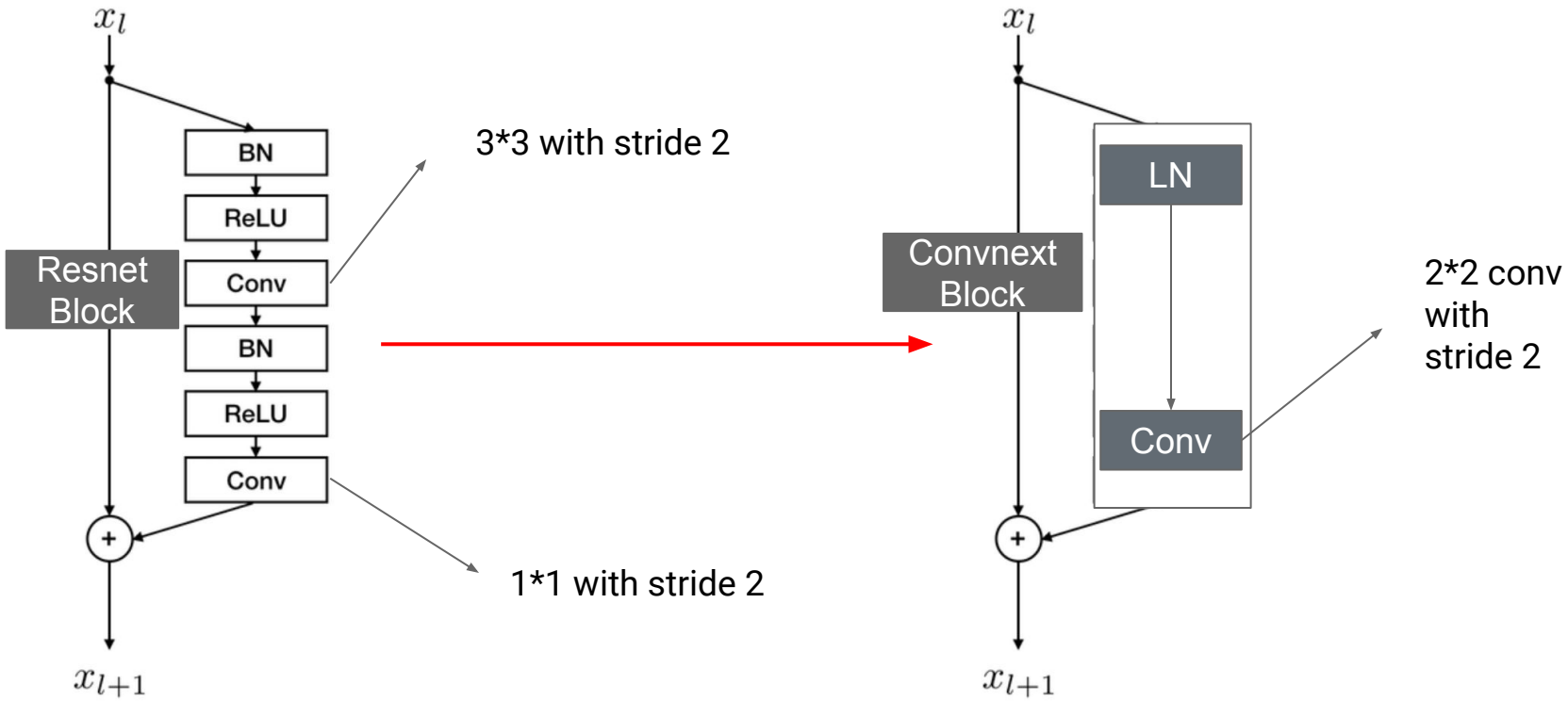



C: Channels
N: Batch Dimension
H, W: Height and Width of feature map

# mD4- Improvement



# mD5: Separate Downsampling Layer

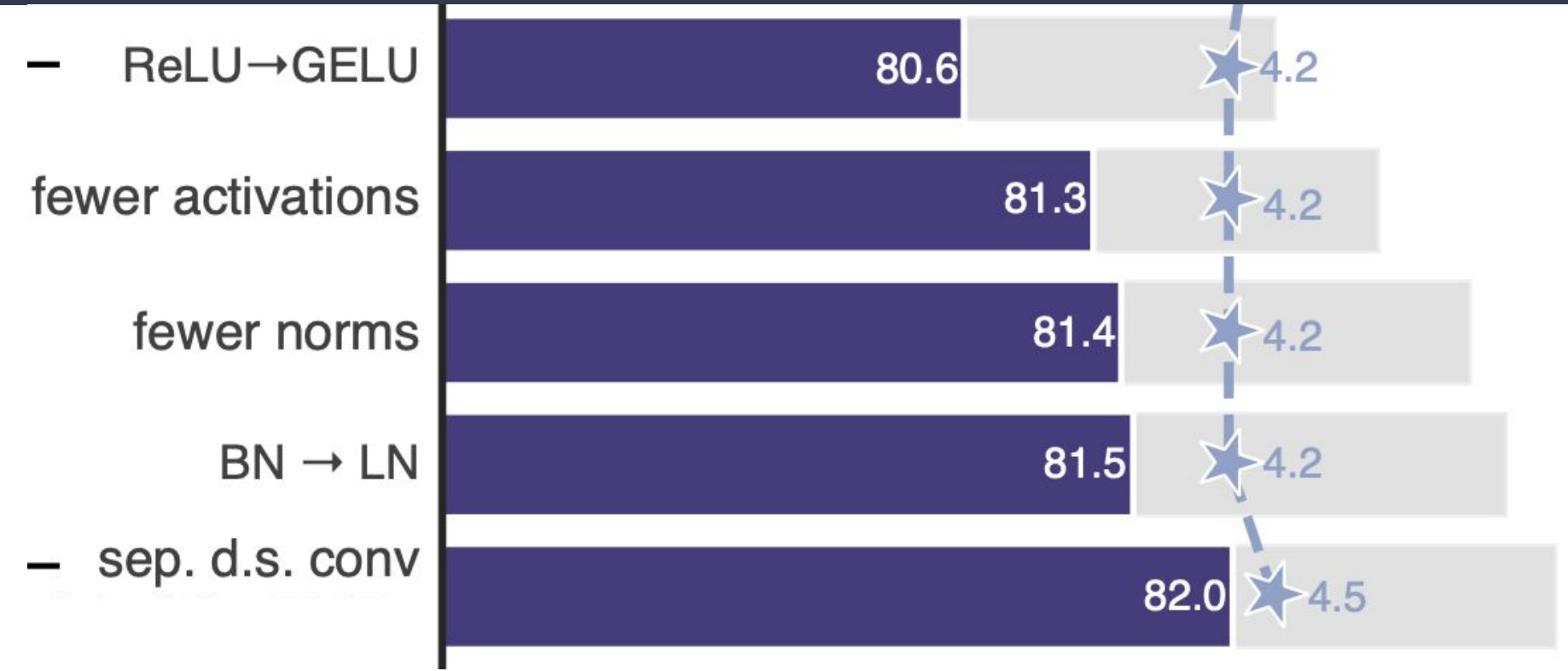


**This modification surprisingly leads to diverged training!**



adding normalization layers wherever spatial resolution is changed can help stabilize training

# mD5 - Improvement

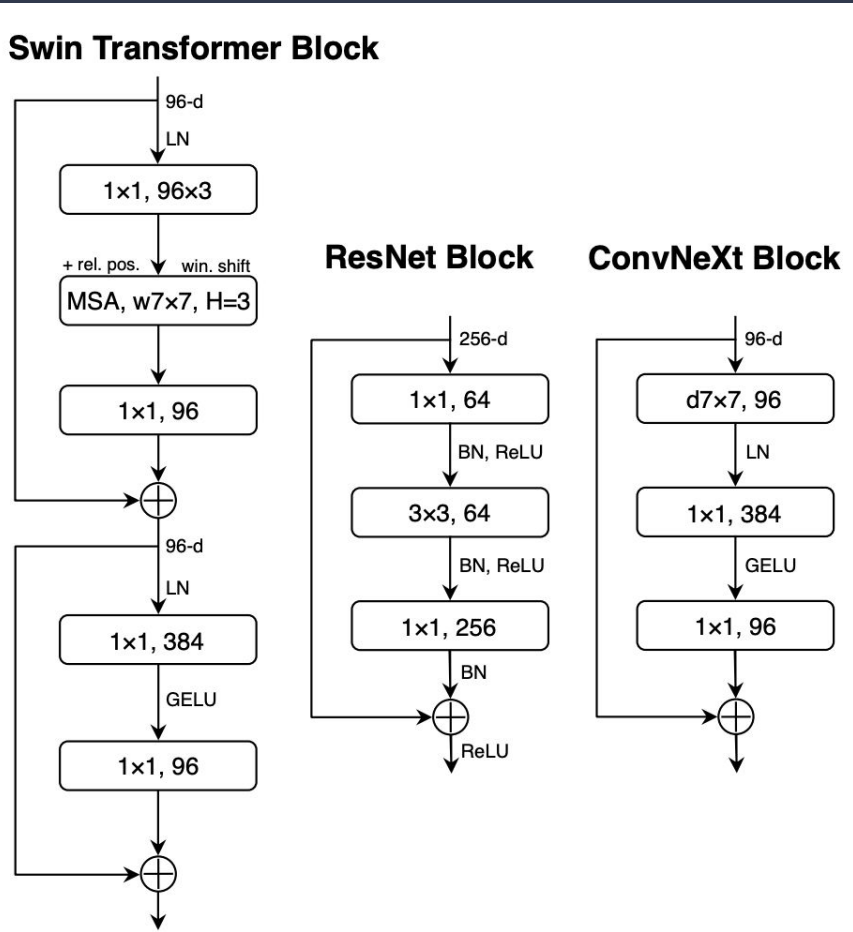


# Final ConvNeXt block

- Macro Design Changes
- Micro Design Changes
- ConNeXt has similar FLOPs as SWIN.
- No specialized modules unlike SWIN
  - Shifted window attentions
  - Relative position bias

*all design choices discussed so far are adapted from vision Transformers.*

*these designs are not novel even in the ConvNet literature they have all been researched separately, but not collectively, over the last decade.*



| 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 |

Base Line

Macro Changes

Micro Changes

# Evaluation on ImageNet

# Networks for Evaluation

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 = (256,512,1024,20148) , B = (3,3,27,3)

Vs

ConvNet Architectures

1. RegNets
2. EfficientNet and EfficientNet V2

Transformer Architectures

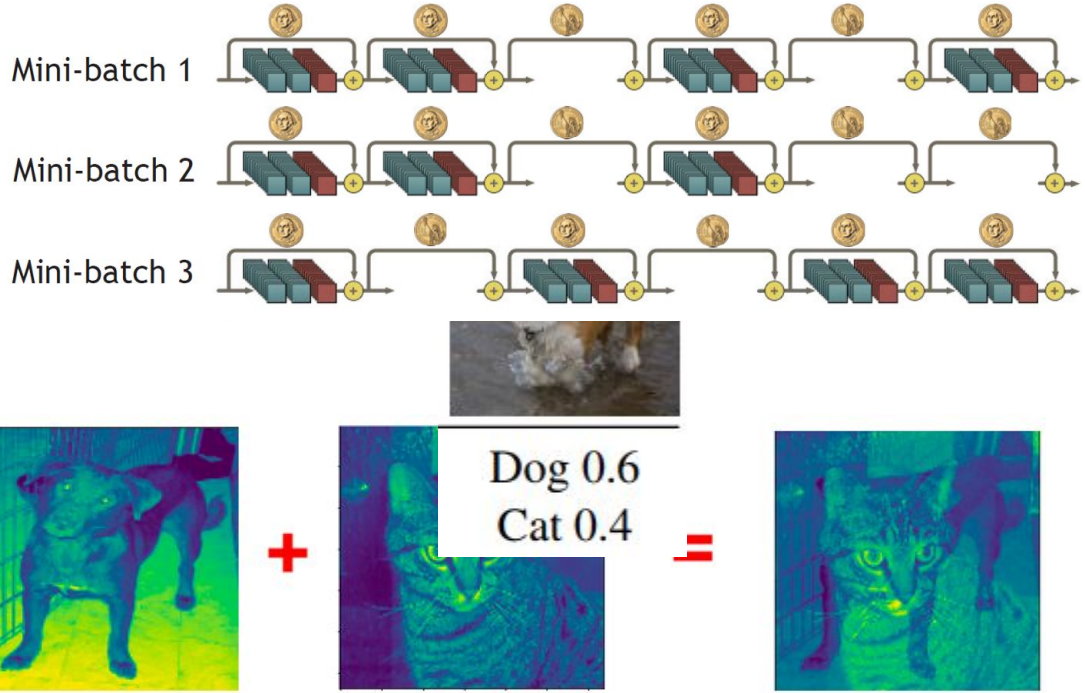
1. DeiT
2. SWIN

# Training Settings

**ImageNet-1K - Training from Scratch**

- AdamW Optimizer
- 300 Epochs → Warmup 20 epoch
- Regularizers →Stochaistic depth, label smoothing
- Data Augmentation → Mixup, RandAug , CutMix etc
- EMA and Layer Scale

**Pre-training on ImageNet-22K**



# Results on ImageNet - 1K

- Outperformed SWIN in-terms of Throughput and Accuracy
- ConvNext still lags behind ConvNet i.e., EffNet V2

| model | image size | #params | FLOPs | throughput (image / s) | IN-1K top-1 acc. |
|---|---|---|---|---|---|
| • 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 |
| ○ Swin-S | $224^2$ | 50M | 8.7G | 436.7 | 83.0 |
| • ConvNeXt-L | $224^2$ | 198M | 34.4G | 146.8 | **84.3** |
| • ConvNeXt-L | $384^2$ | 198M | 101.0G | 50.4 | **85.5** |
| ○ Swin-T | $224^2$ | 28M | 4.5G | 757.9 | 81.3 |
| • ConvNeXt-T | $224^2$ | 29M | 4.5G | 774.7 | **82.1** |

0.3 %

0.1 %

0.8 %

# Results on ImageNet - 22K

- Observed the scaling property of transformers in convnext's

| model | image size | #param. | FLOPs | throughput (image / s) | IN-1K top-1 acc. |
|---|---|---|---|---|---|
| • ConvNeXt-T | $224^2$ | 29M | 4.5G | 774.7 | **82.9** |
| • ConvNeXt-S | $224^2$ | 50M | 8.7G | 447.1 | **84.6** |
| • 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 |
| • 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** |

0.8%

1.5%

# Machine Performance Comparison

| 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** |

# Takeaways

- Observed the scaling behaviour in ConvNexts
- Outperformed the ConvNets in almost every aspect
- ConvNets attained higher Inference throughputs

# Training

Accuracy Improvement → +2.7%
76.1 % to 78.8%

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

# Stem Change

- Input Layer in the processing layers of the Network

| Architecture | Conv Kernel Size | Stride |
|---|---|---|
| ResNet | 7x7 | 2 |
| Vision Transformer | 16x16 / 14x14 | 16 / 14 |
| SWIN Transformer | 4x4 | 4 |

| | | |
|---|---|---|
| ConvNext | 4x4 | 4 |

# IsoTropic ConvNeXt - ImageNet-1K

| 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 |

# Evaluation on Downstream tasks

# COCO - Object Detection

| backbone | FLOPs | FPS | $AP^{box}$ | $AP^{box}_{50}$ | $AP^{box}_{75}$ | $AP^{mask}$ | $AP^{mask}_{50}$ | $AP^{mask}_{75}$ |
|---|---|---|---|---|---|---|---|---|
| Mas[illegible]-RCNN [illegible]× schedule | | | | | | | | |
| ○ Swin-T | 267G | 23.[illegible] | 46.0 | 68.1 | 50.3 | 41.6 | 65.1 | 44.9 |
| ● ConvNeXt-T | 262G | 25.[illegible] | **46.2** | 67.9 | 50.8 | **41.7** | 65.0 | 44.9 |
| Cascade Mask-RC[illegible]N 3× schedu[illegible]le | | | | | | | | |
| ● ResNet-50 | 739G | 16.2 | 46.3 | 64.3 | 50.5 | 40.1 | 61.7 | 43.4 |
| ● X101-32 | 819G | 13.8 | 48.1 | 66.5 | 52.4 | 41.6 | 63.9 | 45.2 |
| ● X101-64 | 972G | 12.[illegible] | 48.3 | 66.4 | 52.3 | 41.7 | 64.0 | 45.1 |
| ○ Swin-T | 745G | 12.2 | 50.4 | 69.2 | 54.7 | 43.7 | 66.6 | 47.3 |
| ● ConvNeXt-T | 741G | 13.[illegible] | **50.4** | 69.1 | 54.8 | **43.7** | 66.5 | 47.3 |
| ○ Swin-S | 838G | 11.[illegible] | 51.9 | 70.7 | 56.3 | 45.0 | 68.2 | 48.8 |
| ● ConvNeXt-S | 827G | 12.[illegible] | **51.9** | 70.8 | 56.5 | **45.0** | 68.4 | 49.1 |
| ○ Swin-B | 982G | 10.[illegible] | 51.9 | 70.5 | 56.4 | 45.0 | 68.1 | 48.9 |
| ● ConvNeXt-B | 964G | 11.[illegible] | **52.7** | 71.3 | 57.2 | **45.6** | 68.9 | 49.5 |
| ○ Swin-B‡ | 982G | 10.[illegible] | 53.0 | 71.8 | 57.5 | 45.8 | 69.4 | 49.7 |
| ● ConvNeXt-B‡ | 964G | 11.[illegible] | **54.0** | 73.1 | 58.8 | **46.9** | 70.6 | 51.3 |
| ○ Swin-L‡ | 1382G | 9.2 | 53.9 | 72.4 | 58.8 | 46.7 | 70.1 | 50.8 |
| ● ConvNeXt-L‡ | 1354G | 10.[illegible] | **54.8** | 73.8 | 59.8 | **47.6** | 71.3 | 51.7 |
| ● ConvNeXt-XL‡ | 1898G | 8.6 | **55.2** | 74.2 | 59.9 | **47.7** | 71.6 | 52.2 |

# ADE20K - Segmentation

| backbone | input crop. | mIoU | #param. | FLOPs |
|---|---|---|---|---|
| | ImageNet-1K pre-trained | | | |
| ○ Swin-T | $512^2$ | 45.8 | 60M | 945G |
| ● ConvNeXt-T | $512^2$ | **46.7** | 60M | 939G |
| ○ Swin-S | $512^2$ | 49.5 | 81M | 1038G |
| ● ConvNeXt-S | $512^2$ | **49.6** | 82M | 1027G |
| ○ Swin-B | $512^2$ | 49.7 | 121M | 1188G |
| ● ConvNeXt-B | $512^2$ | **49.9** | 122M | 1170G |
| | ImageNet-22K pre-trained | | | |
| ○ Swin-B‡ | $640^2$ | 51.7 | 121M | 1841G |
| ● ConvNeXt-B‡ | $640^2$ | **53.1** | 122M | 1828G |
| ○ Swin-L‡ | $640^2$ | 53.5 | 234M | 2468G |
| ● ConvNeXt-L‡ | $640^2$ | **53.7** | 235M | 2458G |
| ● ConvNeXt-XL‡ | $640^2$ | **54.0** | 391M | 3335G |

# Conclusion

# Conclusion

- ConvNeXts proposed as a ConvNet model that competes with Transformers
- Authors hope the importance of convolutions is re-examined in modern vision architectures
- Optimization is important, as well as architecture

# Deficiencies in the work

