A ConvNet for the 2020s: ConvNext

Jashwanthreddy Katamreddy, Chenqian Xu,

Outline

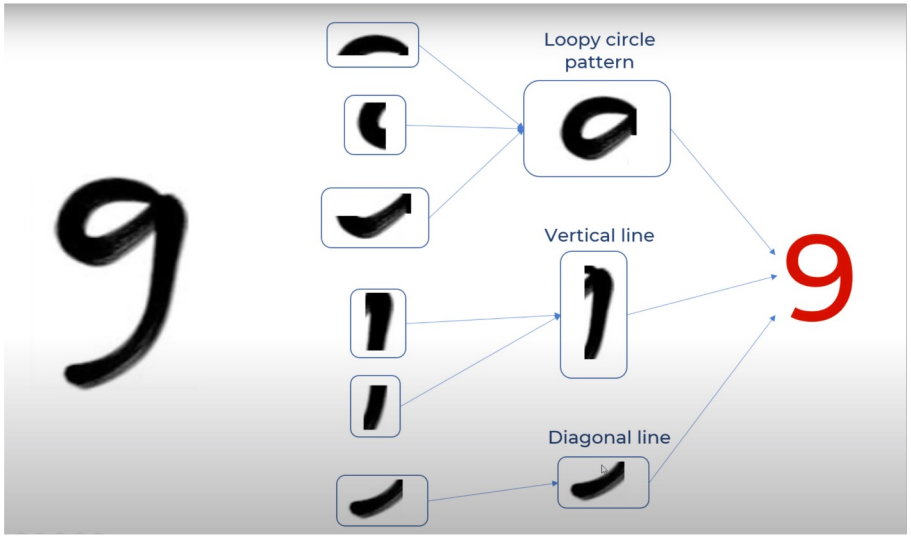
- Background
- Introduction
- Motivation
- Modernizing a ConvNet
- Empirical Evaluations on ImageNet
- Empirical Evaluations on Downstream Tasks
- Conclusions
- Future work

Background

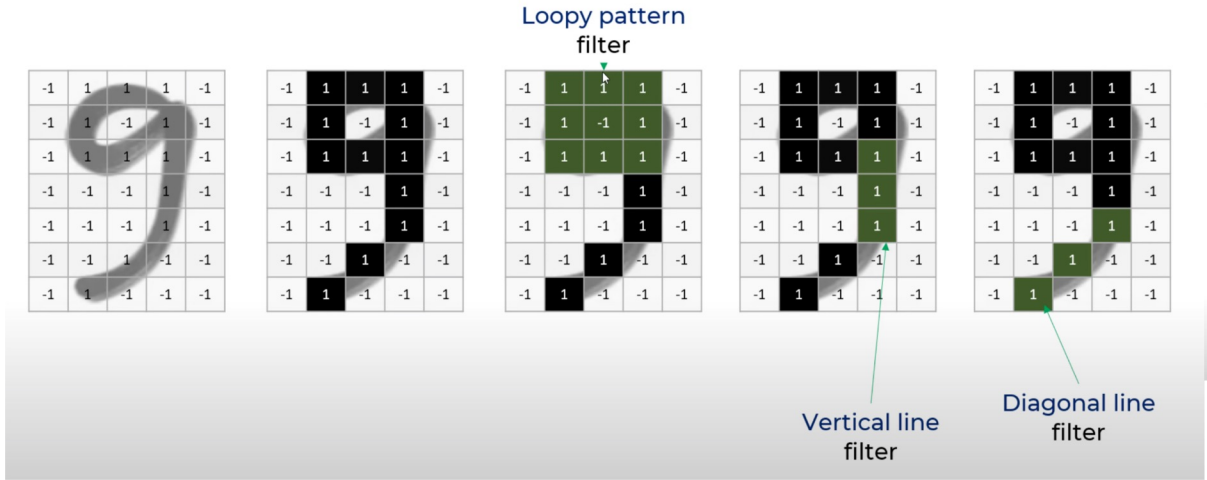
Convolutional Neural Network(ConvNet):

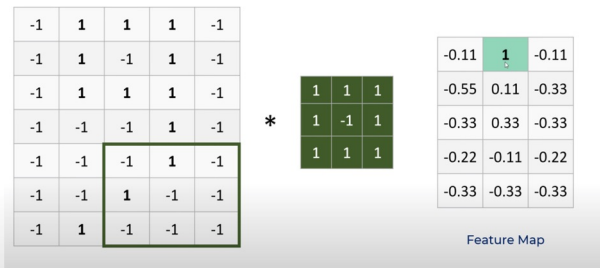
- CNN is a type of deep learning neural network designed to process pixel data and used in image recognition and processing.
- First introduced in the 1980s by Yann LeCun, a postdoctoral Computer Science researcher.
- The early version of CNNs, called LeNet (after LeCun), could recognize handwritten digits.

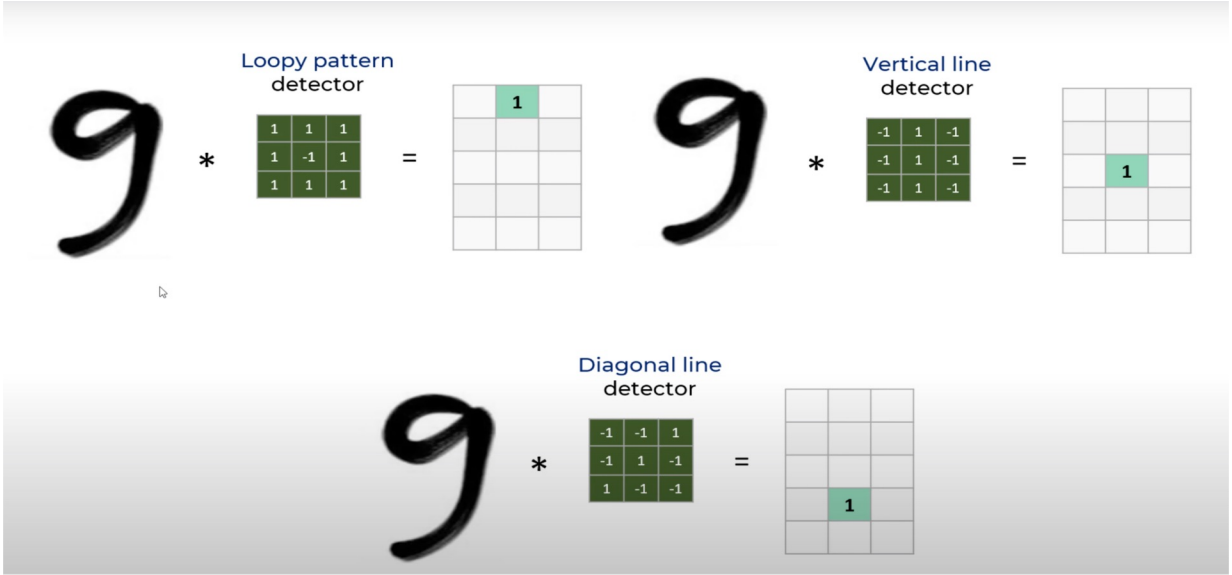
How do humans interpret image?

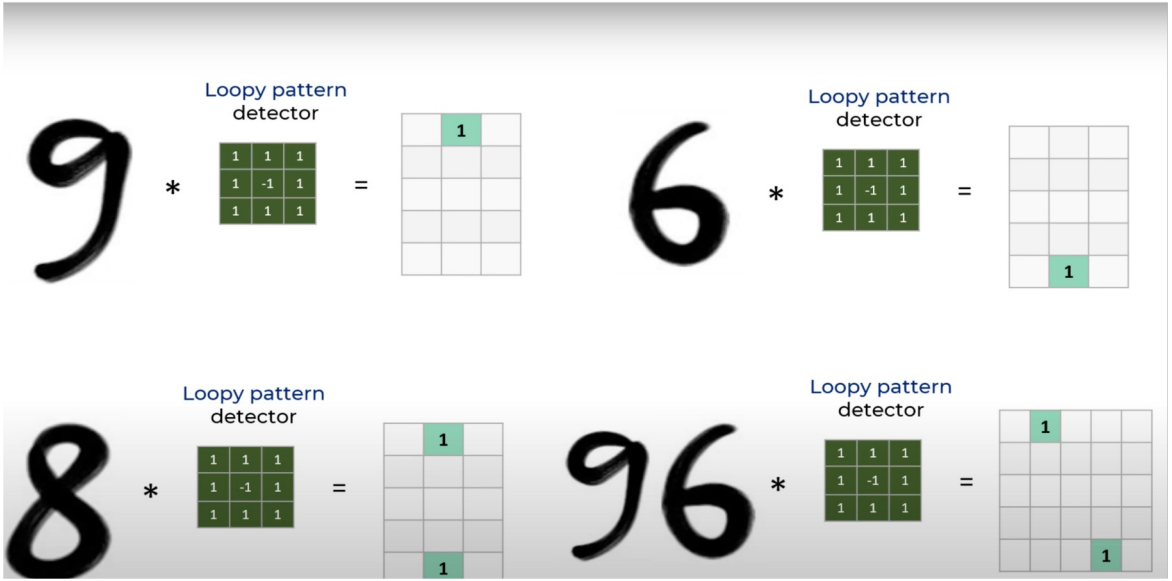


How do we make Computers understand the image?






Loopy pattern
detector
1 1 1
1 -1 1
1 1 1
1
Vertical line
detector
-1 1 -1
-1 1 -1
-1 1 -1
1
Diagonal line
detector
-1 -1 1
-1 1 -1
1 -1 -1
1


Loopy pattern detector
1 1 1
1 -1 1
1 1 1
Loopy pattern detector
1 1 1
1 -1 1
1 1 1
Loopy pattern detector
1 1 1
1 -1 1
1 1 1
Loopy pattern detector
1 1 1
1 -1 1
1 1 1

Background

- In 2012, **AlexNet** which uses multi-layered neural networks precipitated the “ImageNet moment” in a new era of computer vision.
- Many ConvNets focused on different aspects of accuracy, efficiency and scalability, and popularized many useful design principles.
- New ConvNets have several built-in inductive biases that make them well- suited to a wide variety of CV applications and also efficient when used in a sliding-window manner(computations are shared).
- This has been the default use of ConvNets, generally on limited object categories such as digits, faces and pedestrians.

Introduction

- Around the same time, natural language processing (NLP) took a very different path, as the Transformers replaced Recurrent Neural Networks(RNNs) to become the dominant backbone architecture.
- The two streams converged in the year 2020, as the introduction of **Vision Transformers (ViT)**.
- One primary focus of ViT is on the scaling behavior.
- With the help of larger model and dataset sizes, Transformers can outperform standard ResNets by a significant margin.

Introduction

- The biggest challenge is ViT's global attention design, which has a quadratic complexity with respect to the input size.
- To overcome that, the sliding window strategy was reintroduced to Transformers, allowing them to behave more similarly to ConvNets.
- **S**hifted **Win**dow **Transformer** is a milestone work in this direction, demonstrating for the first time that Transformers can be adopted as a generic vision backbone and achieve state-of-the-art performance across a range of computer vision tasks beyond image classification.

Motivation

- To bridge the gap between the pre-ViT and post-ViT eras for ConvNets, as well as to test the limits of what a pure ConvNet can achieve, the authors propose a family of pure ConvNets dubbed ConvNeXt.
- How to improve ConvNet in modernized way to get closer or exceed Transformer model?

Roadmap to modernize ConvNet

Two models are considered:
- ResNet-50 / Swin-T regime with FLOPs around **4.5 x 10^9**
- ResNet-200 / Swin-B regime which has FLOPs around **15.0 x 10^9**

Network Modernization:

1. Applying similar training techniques used to train ViT and obtain much improved results compared to the original ResNet-50.
2. Macro design
3. ResNeXt-ify
4. Inverted bottleneck
5. Large Kernel Size
6. Micro design

Training Techniques

- **Training epochs**: 300 epochs (from the original **90 epochs** for ResNets)
- **Optimizer**: AdamW (Weight Decay)
- **Data augmentation**
  Mixup
  Cutmix
  Rand Augment

- **Regularization**
  Stochastic Depth
  Label Smoothing

- This **increased the performance** of the ResNet-50 model from 76.1% to 78.8%**(+2.7%)**

Macro Design

Changing Stage compute ratio:

- Swin-T's computation ratio of each stage is 1:1:3:1, and for larger Swin Transformers, the ratio is 1:1:9:1.
- The number of blocks in each stage from (3, 4, 6, 3) in ResNet-50 is changed to (3, 3, 9, 3).
- This improved the model accuracy from 78.8% to 79.4%.

Changing stem to "Patchify" :

- The stem cell in standard ResNet contains a 7x7 convolution layer with stride 2, followed by a max pool, which result in a 4x downsampling.
- ConvNeXt replaces the ResNet-style stem cell with a patchify layer implemented using a 4x4, stride 4 convolution layer.
- The accuracy has changed from 79.4% to 79.5%.

ResNeXt-ify

- In this part, we attempt to adopt the idea of ResNeXt, which has a better FLOPs/accuracy trade-off than a vanilla ResNet.
- ResNeXt-ify has grouped convolution as the core which is similar to sum operation in self-attention.
- Following the strategy proposed in ResNeXt, the network width has been increased to the same number of channels as Swin-T's (from 64 to 96).
- This brings the network performance to 80.5% with increased FLOPs (5.3G).

Inverted bottleneck

- MLP blocks 4 times wider than input dimension
- Larger kernel size: moving up depthwise conv layer. MSA is put prior to MLP layers.

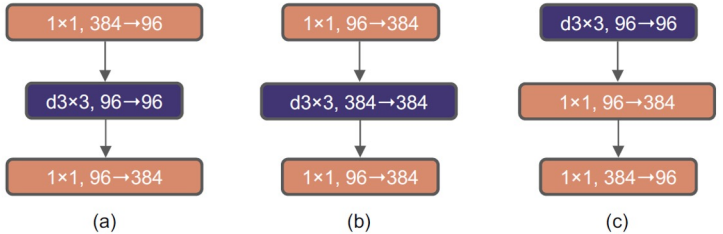


Figure 3. **Block modifications and resulted specifications. (a)** is a ResNeXt block; in **(b)** we create an inverted bottleneck block and in **(c)** the position of the spatial depthwise conv layer is moved up.

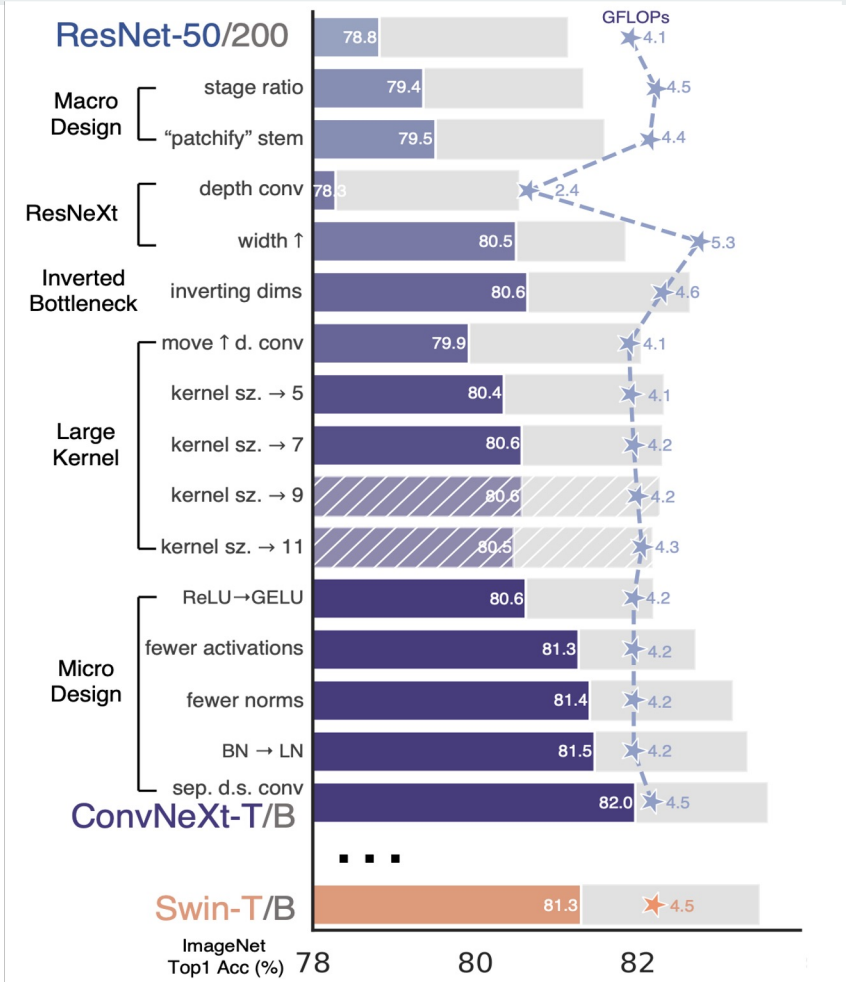
Larger kernel sizes

- Although Swin Transformers reintroduced the local window to the self-attention block, the window size is at least 7x7, significantly larger than the ResNe(X)t kernel size of 3x3.
- The authors experimented with several kernel sizes, including 3, 5, 7, 9, and 11. The network's performance increases from 79.9% (3x3) to 80.6% (7x7), while the network's FLOPs stay roughly the same.
- A ResNet-200 regime model does not exhibit further gain when increasing the kernel size beyond 7x7.

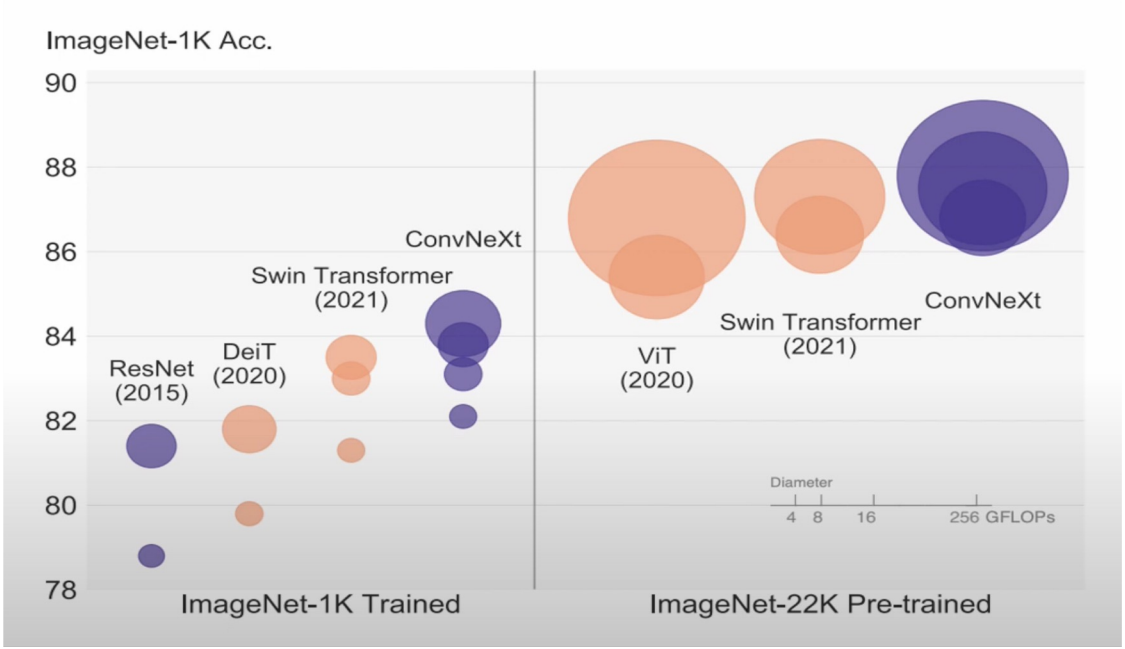
Micro Design

- GELU replace ReLU: smoother variant
- Fewer activation function: consider a block with key/query/value linear embedding layer
- Fewer normalization layer
- LN substitute BN
- Separate downsampling layer: residual block

Model Accuracy



Empirical evaluation on ImageNet



Empirical evaluation on ImageNet

model	image size	#param.	FLOPs	throughput (image / s)	IN-1K top-1 acc.
ImageNet-1K trained models					
• RegNetY-4G [51]	224^2	21M	4.0G	1156.7	80.0
• RegNetY-8G [51]	224^2	39M	8.0G	591.6	81.7
• RegNetY-16G [51]	224^2	84M	16.0G	334.7	82.9
• EffNet-B3 [67]	300^2	12M	1.8G	732.1	81.6
• EffNet-B4 [67]	380^2	19M	4.2G	349.4	82.9
• EffNet-B5 [67]	456^2	30M	9.9G	169.1	83.6
• EffNet-B6 [67]	528^2	43M	19.0G	96.9	84.0
• EffNet-B7 [67]	600^2	66M	37.0G	55.1	84.3
○ DeiT-S [68]	224^2	22M	4.6G	978.5	79.8
○ DeiT-B [68]	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**

ImageNet-22K pre-trained models					
• R-101x3 [36]	384^2	388M	204.6G	-	84.4
• R-152x4 [36]	480^2	937M	840.5G	-	85.4
○ ViT-B/16 [18]	384^2	87M	55.5G	93.1	84.0
○ ViT-L/16 [18]	384^2	305M	191.1G	28.5	85.2
○ 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**

Empirical Evaluation on COCO: object detection and segmentation

backbone	FLOPs	FPS	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
Mask-RCNN 3× schedule								
○ Swin-T	267G	23.1	46.0	68.1	50.3	41.6	65.1	44.9
● ConvNeXt-T	262G	25.6	**46.2**	67.9	50.8	**41.7**	65.0	44.9
Cascade Mask-RCNN 3× schedule								
● ResNet-50	739G	11.4	46.3	64.3	50.5	40.1	61.7	43.4
● X101-32	819G	9.2	48.1	66.5	52.4	41.6	63.9	45.2
● X101-64	972G	7.1	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	13.5	**50.4**	69.1	54.8	**43.7**	66.5	47.3
○ Swin-S	838G	11.4	51.9	70.7	56.3	45.0	68.2	48.8
● ConvNeXt-S	827G	12.0	**51.9**	70.8	56.5	**45.0**	68.4	49.1
○ Swin-B	982G	10.7	51.9	70.5	56.4	45.0	68.1	48.9
● ConvNeXt-B	964G	11.4	**52.7**	71.3	57.2	**45.6**	68.9	49.5
○ Swin-B‡	982G	10.7	53.0	71.8	57.5	45.8	69.4	49.7
● ConvNeXt-B‡	964G	11.5	**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	10.0	**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

Table 3. **COCO object detection and segmentation results** using Mask-RCNN and Cascade Mask-RCNN. ‡ indicates that the model is pre-trained on ImageNet-22K. ImageNet-1K pre-trained Swin results are from their Github repository [3]. AP numbers of the ResNet-50 and X101 models are from [42]. We measure FPS on an A100 GPU. FLOPs are calculated with image size (1280, 800).

Empirical Evaluation on ADE20K: Semantic 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

Table 4. **ADE20K validation results** using UperNet [80]. ‡ indicates IN-22K pre-training. Swins' results are from its GitHub repository [2]. Following Swin, we report mIoU results with multi-scale testing. FLOPs are based on input sizes of (2048, 512) and (2560, 640) for IN-1K and IN-22K pre-trained models, respectively.

Conclusion

- In the 2020s, vision Transformers, particularly hierarchical ones such as Swin Transformers, began to overtake ConvNets as the favored choice for generic vision backbones.
- The widely held belief is that vision Transformers are more accurate, efficient, and scalable than ConvNets.
- ConvNeXts, a pure ConvNet model can compete favorably with state-of- the-art hierarchical vision Transformers across multiple computer vision benchmarks, while retaining the simplicity and efficiency of standard ConvNets.

Future work

- The authors hope that the new results reported in this study will challenge several widely held views and prompt people to rethink the importance of convolution in computer vision.

Thank you!!!!