**Title:**A ConvNrt for the 2020s

Code: https://github.com/facebookresearch/ConvNeXt

**Abstract:**

They gradually “modernize”a standard ResNet toward the design of a vision Transformer. ConvNeXts compete favorably with Transformers in terms of accuracy and scalability, achieving 87.8% ImageNet top-1 accuracy and outperforming Swin Transformers on COCO detection and ADE20K segmentation, while maintaining the simplicity and efficiency of standard ConvNets.

利用Swin Transformer类似的组建方法“现代化“了ConvNet，使其在目标检测和语义分割上领先了Swin Transformer，并保持了ConvNet的简单性和效率。

**Intro：**

BACKGROUND:

The full dominance of ConvNets in computer vision before Hierarchical Transformers: Swin Transformer. Swin Transformer [45] 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.

Swin Transformer 成为视觉任务的主流和骨干

PROBLEM:

These research is intended 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.

弥补Swin Transformer和ConvNet的差距，测试ConvNet的极限

APPROACH:

They discover several key components that contribute to the performance difference along the way. As a result, we propose a family of pure ConvNets dubbed ConvNeXt.

从Swin Transformer的构建发现了几个能提升性能的组件，并用于ConvNet，提出了模型族

RESULT：

ConvNeXts, constructed entirely from standard ConvNet modules, compete favorably with Transformers in terms of accuracy, scalability and robustness across all major benchmarks. ConvNeXt maintains the efficiency of standard ConvNets, and the fully-convolutional nature for both training and testing makes it extremely simple to implement。

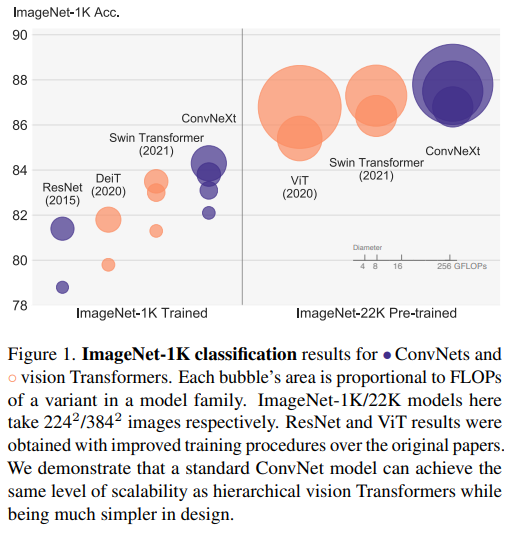
ConvNeXts和Swin Transformer在很多地方能竞争，并且保持了简洁和效率

**Conclusion：**

These observations are surprising while our ConvNeXt model itself is not completely new — many design choices have all been examined separately over the last decade, but not collectively. They 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.

在ConvNeXts上的这些改进不全是新的，之前就有提出，但是没有整合起来，他们希望他们的研究可以让人们重新思考卷积在CV的重要性。

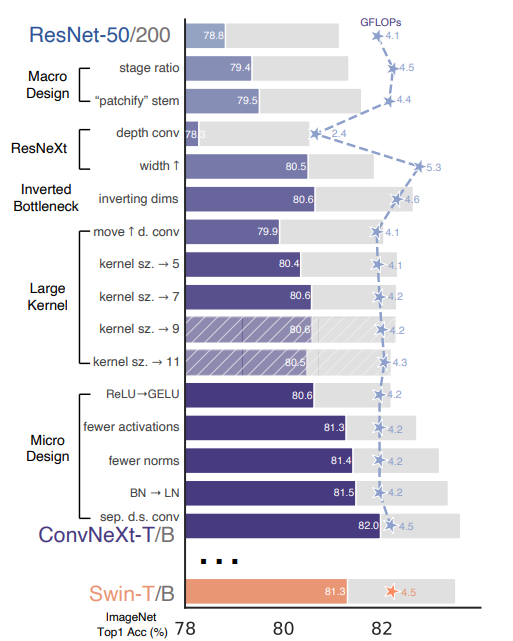
**Figure:**

****

这张图表示了ConvNeXt等几个网络的model family在ImageNet-1K上的准确率表现，和FLOPs大小

它显示了：

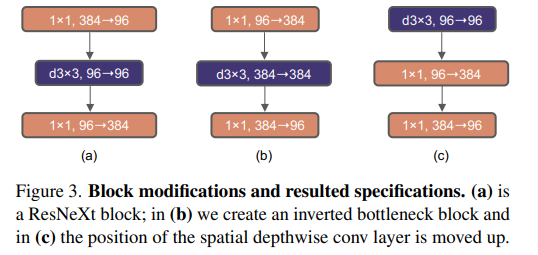
ConvNeXt可以达到同样规模的Swin Transformer的等级但是它更容易设计



we show the procedure and the results we are able

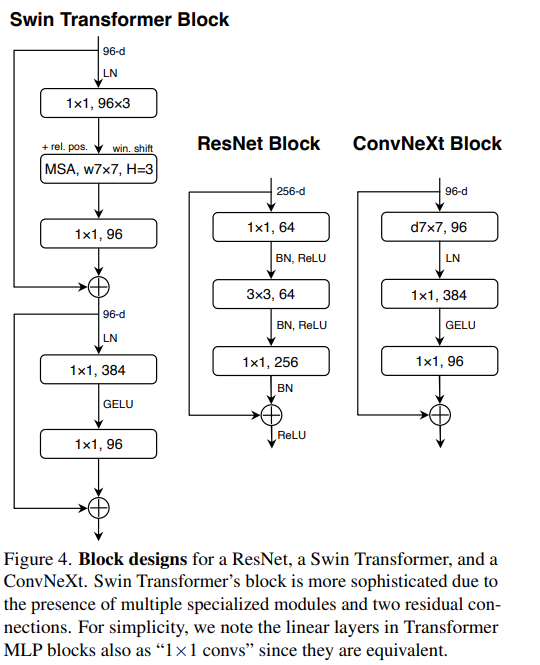
to achieve with each step of the “network modernization”.

每一步“现代化“所展现出的性能变化，和Swin Transformer的对比。



. Figure 3(a) to (b) illustrate the configurations.

讲述了瓶颈块的配置，使性能提高



这张图对比了三个模型的构造，并且解释了ConvNeXt为什么要这样构造：实验表明这样性能更好。卷积层变为7\*7，BN变为LN，激活函数变成GELU等等，具体的性能变化可以在Figure2 上看到

**Modernizing a ConvNet: a Roadmap**

starting point : ResNet-50 model

**Training Techniques**

We first train it with similar training techniques used to train vision Transformers and obtain much improved results compared to the original ResNet-50.

90-300 epochs , use the AdamW optimizer, data augmentation techniques, Random Erasing, and regularization schemes including Stochastic Depth and Label Smoothing.

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

使用训练ViT的方法训练ResNet-50，获得了较大的提升，这是baseline

方法有：轮次从90提升到300、使用AdamW优化器、数据增广技术，随机擦除，正则化方案包括随机深度和标签平滑

**Macro Design（宏观设计）**

1.Changing stage compute ratio

They adjust the number of blocks in each stage from(3, 4, 6, 3) in ResNet-50 to (3, 3, 9, 3).

This improves the model accuracy from 78.8% to 79.4%

块数改变

2.Changing stem to “Patchify”

They replace the ResNet-style stem cell with a patchify layer implemented using a 4×4, stride 4 convolutional layer

The accuracy has changed from 79.4% to 79.5%

使用4×4 步长为4 的卷积层代替了ResNet风格的stem cell，提升了性能（79.4% to 79.5%）

**ResNeXt-ify**

Following the strategy proposed in ResNeXt, we increase the network width 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).

拓宽了网络（64-96），提升了性能但是增加了FLOPs

**Inverted Bottleneck**

Figure3, (b)

This change reduces the whole network FLOPs to 4.6G, due to the significant FLOPs reduction in the downsampling residual blocks’ shortcut 1×1 conv layer.

这项举措减少了FLOPs，略微提升了性能(80.5% to 80.6%)

**Large Kernel Sizes**

1.Moving up depthwise conv layer

Figure 3 , (b) to (c)

This intermediate step reduces the FLOPs to 4.1G, resulting in a temporary performance degradation to 79.9%.

2.Increasing the kernel size.

They will use 7×7 depthwise conv in each block.

The network’s performance increases from 79.9% (3×3) to 80.6% (7×7)

**Micro Design**

1. Replacing ReLU with GELU

GELU, which can be thought of as a smoother variant of ReLU is utilized in the most advanced Transformers.

the accuracy stays unchanged (80.6%).

使用激活函数GELU代替ReLU，GELU是ReLU更平滑的变体，精确度保持不变

2. Fewer activation functions.

They eliminate all GELU layers from the residual block except for one between two 1 × 1 layers, replicating the style of a Transformer block. This process improves the result by 0.7% to 81.3%, practically matching the performance of Swin-T.

两个1\*1卷积层中保留激活函数，去除了其余所有激活函数，性能提升了0.7% 至 81.3%。

3.Fewer normalization layers

Here we remove two BatchNorm (BN) layers, leaving only one BN layer before the conv 1 × 1 layers. This further boosts the performance to 81.4%

在1\*1卷积层前保留一层BN层，移除其余BN层，性能提升至81.4%

4.Substituting BN with LN

Directly substituting LN for BN in the original ResNet will result in suboptimal performance, but in fact, the performance is slightly better in ConvNeXt, obtaining an accuracy of 81.5%.

直接在ResNet上使用LN代替BN会导致次优性能，但在这个模型上会使性能稍好，达到81.5%

5.Separate downsampling layers.

We will use separate downsampling layers. We can improve the accuracy to 82.0%, significantly exceeding Swin-T’s 81.3%

使用单独的下采样层，使性能提升到了82.0%，形成了最终的模型。

**QUESTION:**

-What did authors try to accomplish?

A batter ConvNet:(ConvNeXt), and evaluated this net so that can compete with Swin Transformer in CV task.

-What the key elements of the approach?

They found several key components to increase the ACC like kernel size equal to 7 and activated foundation GELU and so on. They collected these approach which can improve performance and applied to they ConvNet.

-What can you use yourself?

I think the background introduction and some experiment approaches can be use by myself.

-What other reference do you want to follow?

Swin Transformers