A ConvNet for the 2020s

^o^ ConvNeXt

Abstract

20년대를 아우르는 시각적 인식은 Convolution을 압도하는 ViT기반이다. 그러나 순수 ViT는 오브젝트 디텍션이나 시멘틱 세그먼테이션을 다룰 수는 없다. 계층적 트랜스포머인 Swin Transformer의 경우 ConvNet을 다시 제시하면서 트랜스포머가 실질적으로 시각 백본으로 사용할 수 있게 하며 여러 비젼 테스크에서의 성능을 입증했습니다. 그러나 이러한 하이브리드 접근은 컨볼루션의 유도 편향 보다는 트랜스포머의 우월성에 기인합니다. 이 논문에서 우리는 Convolution을 새롭게 제작하여 컨볼루션이 어느 정도의 성능을 가질 수 있는지를 실험합니다. 우리는 차츰 표준 ResNet을 현대화시킵니다. 그리고 몇가지의 핵심적인 구성 요소를 찾아냅니다. 이러한 최종적인 모델은 ConvNeXt입니다. ConvNeXt는 Convolution기반의 모델로 ViT계열에 성능을 뛰어넘게 됩니다.

Introduction

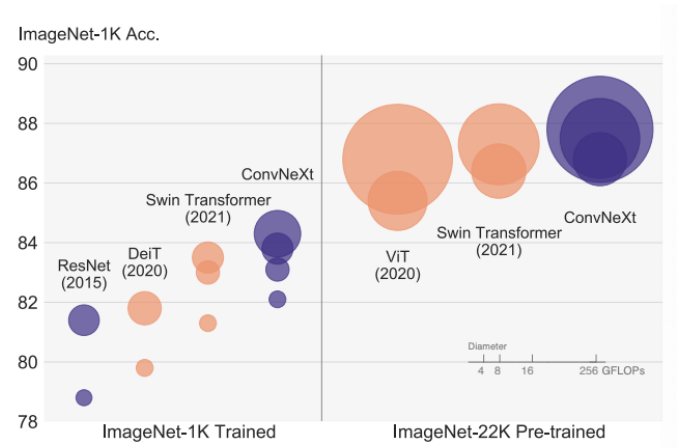

Figure 1. **ImageNet-1K classification** results for • ConvNets and ○ vision Transformers. Each bubble's area is proportional to FLOPs of a variant in a model family. ImageNet-1K/22K models here take 224^2/384^2 images respectively. ResNet and ViT results were obtained with improved training procedures over the original papers. We demonstrate that a standard ConvNet model can achieve the same level of scalability as hierarchical vision Transformers while being much simpler in design.

2010년대를 돌아보면 딥러닝에서는 많은 발전이 있었고, 특히 Convolution이 많이 사용되었습니다. 기존의 피쳐를 만드는 방식에서 ConvNet model을 만드는 방식으로 왔습니다. 백프로파게이션의 경우 1980년대에 생겨났지만 그 효과는 2012년 이후에 입증되었습니다. 그 이후 AlexNet, VGGNet, Inceptions, ResNe(X)t, DenseNet, MobileNet, EfficientNet, RegNet이 생겨났고 이는 정확도 효율성 확장성에 집중하였고 많은 디자인의 원리가 되었습니다.

Computer Vision에서 ConvNet이 좋은 성능을 나타내는 것은 몇몇 이유가 있습니다. sliding window 방식의 경우 큰 해상도 이미지를 처리가 가능했습니다. 그리고 ConvNets 다양한 컴퓨터 비젼 테스크를 처리할 수 있는 inductive bias를 가지고 있습니다. 가장 중요한 것은

translation equivariance입니다. 이는 위치에 상관 없이 지역 별 포착을 해낼 수 있다는 것입니다. 이것은 object detection에 유용했습니다. 그리고 ConvNets은 sliding window의 가중치를 공유함으로써 상당히 효율적입니다. 많은 시간이 흐르고 이러한 것이 ConvNet의 표준이 되었습니다. 그리고 여러 블록을 통해서 detection도 가능해졌습니다.

이와 같은 시기에 NLP는 transformer라는 다른 길을 걸었습니다. CV와 NLP는 다른 감이 있지만 ViT로 인해 두 테스크는 2020년에 들어 합쳐지게 되었습니다. patch를 구분하는 layer를 제외하고 ViT는 기존 NLP에서 사용하는 방식과 크게 다를 점이 없습니다. 그리고 inductive bias 또한 도입하지 않습니다. ViT는 기존의 ResNet을 큰 격차로 성능을 이기게 됩니다. 그러나, ViT는 다양한 Vision task에 적용하기 어려움이 있었고 전체 이미지의 관계를 봐야했기 때문에 큰 해상도의 이미지에 대해서 큰 용량이 필요했습니다.

계층적 Transformer → swin transformer의 경우 이러한 문제를 해결했습니다. sliding window방식을 재도입하여 Convnet과 유사해졌습니다. 이를 통해 Swin Transformer는 다양한 Task의 사용할 수 있게 되었지만, 여기서 중요한 점은 우리는 Convolution 방식은 본질적이며 사라지지 않을 것임을 알 수 있습니다.

이러한 관점에서 Transformer의 많은 발전은 Convolution의 방식을 차용합니다. 그러나 sliding window방식은 많은 cost를 사용할 수 있게 하고 따라서 다양한 sliding방식을 사용할 수 있지만 이는 복잡한 방식입니다.

위와 같은 내용을 들으면 Convolution 방식이 transformer에 밀린다고 생각이 듭니다. 실제로 transformer의 성능은 많은 분야에서 뛰어나고 이는 transformer의 뛰어난 scaling방식과 multi-head-self-attention에 기인합니다.

ConvNets과 Swin Transformer는 비슷하면서 다른 점이 있습니다. 예를 들어 유사한 inductive bias를 가지고 있음과 동시 다른 구조를 가집니다. 여기서 우리는 두 아키텍쳐의 구조에 대해 성능을 확인하며 살펴봅니다. 그리고 ViT와 ConvNet의 격차를 해소시키고 순수 ConvNet의 최대 성능을 확인합니다.

이렇게 하기 위해서 표준의 ResNet을 개선된 방식을 통해 학습시킵니다. 우리는 발전시키며 몇 가지 핵심 요소들을 찾았습니다. 그리고 최종적으로 ConvNeXt를 제시합니다. 우리는 이미지 분류, 디텍션, 세그먼테이션에서 이를 평가합니다. 놀랍게도 ConvNeXt는 transformer의 성능과 경쟁하며 Convolution으로만 구성되어 있어 쉽게 학습과 실행이 가능합니다.

우리는 이러한 새로운 관측이 사람들이 CV에서 Convolution의 중요성을 다시금 생각하기를 바랍니다.

Modernizing a ConvNet : a Roadmap

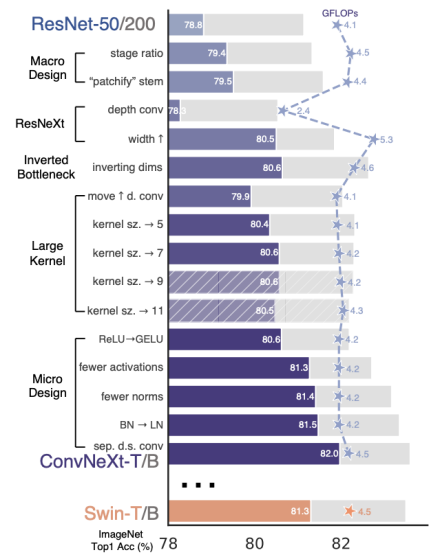

Figure 2. We modernize a standard ConvNet (ResNet) towards the design of a hierarchical vision Transformer (Swin), without introducing any attention-based modules. The foreground bars are model accuracies in the ResNet-50/Swin-T FLOP regime; results for the ResNet-200/Swin-B regime are shown with the gray bars. A hatched bar means the modification is not adopted. Detailed results for both regimes are in the appendix. Many Transformer architectural choices can be incorporated in a ConvNet, and they lead to increasingly better performance. In the end, our pure ConvNet model, named ConvNeXt, can outperform the Swin Transformer.

우리는 Transformer의 성능을 갖게 해주는 ResNet의 발전 방식을 보여줍니다. 우리는 FLOPs를 통하여 두 가지 모델의 사이즈를 확인합니다. ResNet-50 / Swin-T는 4.5*10^9, ResNet-200 / Swin-B는 15.0 × 10^9입니다. 그리고 우리는 ConvNet의 표준을 사용하며 발전시켜나갑니다. 우리는 먼저 기존과 유사하게 original ResNet-50과 transformer를 학습 시킵니다. transformer는 resnet보다 좋은 성능을 보입니다. 이게 우리의 출발 지점입니다. 우리가 발전시킬 방식을 보여주자면

1) macro design 2) ResNeXt 3) inverted bottleneck

4) large kernel size 5) various layer-wise micro designs

입니다. 모델의 정확성은 모델의 복잡성과 관련이 있기 때문에 우리는 FLOPs를 통하여 조절합니다. 그리고 우리의 모든 모델은 ImageNet-1K에서 평가합니다.

Training Techniques

모델의 구조 뿐만이 아니라 학습 과정도 성능에 영향을 미칩니다. Vision Transformer는 모델의 형식 뿐만이 아니라 학습 방식에도 변화가 있었고 이는 모델의 성능에 영향을 미칠 수 있습니다. (예를 들어 AdamW) 그래서 우리의 첫 번째 할 일은 ViT와 학습 방식을 맞추는 것입니다. 이는 ResNet에 적용할 경우 성능이 오르는 것이 증명된 사례입니다. 그리고 학습 epochs를 기존 90에서 300가지 증가시킵니다. data augmentation으로 Mixup, Cutmix, RandAugument, Random Erasing, regularization, Label Smoothing을 사용합니다. 이러한 사용은 2.7%포인트의 성능 향상을 가져다 주었습니다. 모든 성능은 3개의 랜덤 시드의 평균값으로 측정합니다.

Macro Design

우리는 이제 Swin Transformer의 네트워크 디자인을 봅시다. ST는 Convnet의 각각 다른 해상도를 가지는 멀티 스테이지 전략을 따릅니다. 중요한 점은 스테이지 비율 계산, 그리고 이어지는 구조라는 것입니다. 원본 모델의 스테이지 계산 방식은 대개 경험적입니다. res4 레이어는 오브젝트 디텍션과 호환 가능한 무거운 14*14 해상도의 스테이지입니다. 반면, Swin-T는 유사한 원리를 따르지만 비율이 1:1:9:1입니다. 우리는 block을 3, 4, 6, 3에서 3, 3, 9, 3으로 변형했습니다. 이는 0.6%p의 성능 향상을 이뤄냈습니다.

sliding window를 보면 중복적으로 이루어져 이는 인풋 이미지를 적절한 크기로 다운 샘플을 유도하게 됩니다. convolution은 7*7 레이어 가 2stride로 max pooling과 이루어집니다. 이는 4분의 1로 다운샘플링하게 됩니다. 그러나 vision transformer에서는 패치 단위의 전략을 통해서 중복 convolution이 이루어지지 않습니다. 따라서 우리도 4*4 conv를 4stride를 사용해 중복을 없애고 이는 0.1%p의 성능 향상을 기록합니다.

ResNeXt-ify

여기서 우리는 FLOPs와 accuracy간 최적의 trade-off를 사용하는 ResNeXt의 아이디어를 사용합니다. 핵심 요소는 convolution filter가 다른 그룹으로 들어가는 grouped conv입니다. ResNeXt의 아이디어는 더 많은 그룹을 만들어 너비를 확장시키는 것입니다. 더 자세히 보면 3*3conv를 가운데에 적용시킵니다. 우리의 경우 depthwise convolution을 사용하는데, 그룹을 채널의 수만큼 키웁니다. 우리는 이러한 방식이 self-attention의 sum aggregation과 유사한 것으로 봅니다. depthwise convolution과 1*1 convolution의 조합은 공간과 채널의 혼합을 분리를 이끄는데 이는 트랜스포머와 유사합니다. 이러한 방식은 네트워크의 FLOPs를 감소시키지만 성능은 그렇지 않습니다. 그리고 우리는 visionTransformer와 채널을 맞춰주기 위해 channel을 64 → 96으로 증가시켰습니다. 이는 1%p의 성능 향상과 FLOPs의 증가를 얻습니다.

Inverted Bottleneck

Transformer block의 중요한 디자인은 bottleneck입니다 MLP block의 차원은 4번 증가시키며 다시 감소시킵니다. 따라서 우리는 기존의 사용했던 A의 방식에서 B로 변경합니다. 이는 FLOPs를 감소시켰으며 놀랍게도 성능이 0.1%p가 증가했고 ResNet-200/ Swin-B에서는 0.7%p의 성능이 증가했습니다.

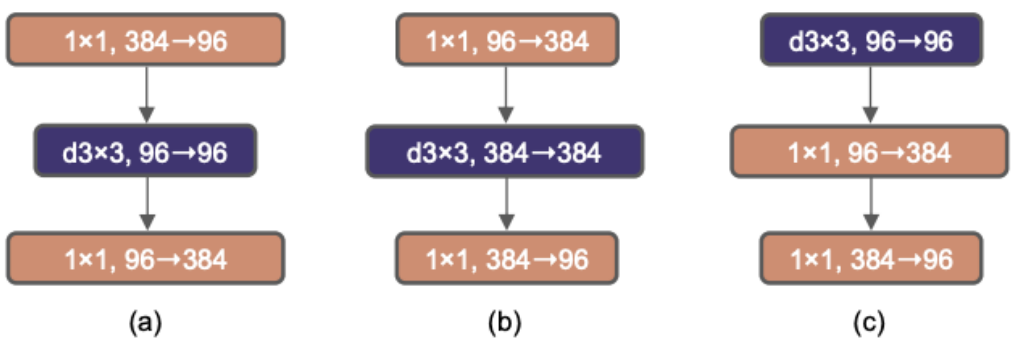

Figure 3. **Block modifications and resulted specifications. (a)** is a ResNeXt block; in **(b)** we create an inverted bottleneck block and in **(c)** the position of the spatial depthwise conv layer is moved up.

Large Kernel Sizes

1*1 convolution의 경우, MLP와 비슷한 역할을 하게 됩니다. 이렇게 생각했을 경우 transformer는 self-attention이라는 공간적 정보를 해결한 다음 MLP block이 오게 되는데 이는 위의 C와 같습니다. 이를 통해 FLOPs는 감소했으나 성능은 감소했습니다.

Transformer는 큰 사이즈의 커널 패치를 이용합니다. 적어도 7*7사이즈의 패치를 그러나, Convolution의 경우 과거에는 큰 커널을 많이 사용했으나 VGG의 3*3 kernel이 깊게 쌓였을 때 좋은 효과를 나타내는 것을 보고 3*3의 kernel이 많이 사용됐습니다. 우리는 3, 5, 7, 9, 11의 kernerl size 실험을 진행했습니다. 결과 3*3일 때는 79.9% → 7*7일 때는 80.6%의 성능 증가가 있었습니다. FLOPs는 거의 유사했습니다. 그리고 큰 커널의 장점은 7까지였음을 확인합니다.

Micro Design

- ReLU를 GELU로 교체합니다.

GELU는 RELU보다 더 부드러운 변화를 일으키며 최신의 transformer에서 사용됩니다. 성능은 같았으나 GELU를 사용합니다.

- 활성화 함수 감소

Transformer의 경우 활성화 함수는 두개의 layer가 있는 MLP blocks에서 한번만 사용됩니다. 일반적으로 Conv에서는 모든 layer에서 사용하지만 우리는 1*1layers를 제외하고 나머지 GELU 활성화 함수를 제거했습니다. 이는 0.7%p의 성능 향상을 가져다 주었습니다.

- 정규화 레이어 감소

Transformer는 적은 정규화 레이어를 가집니다. 그래서 우리는 1*1 layer 이전에 한개의 레이어만 남기고 제거했습니다. 이는 0.1%p의 성능 향상을 가져다 주었습니다.

- 배치 정규화를 레이어 정규화로 교체

배치 정규화는 Convnet에서 일반적으로 과대적합을 감소시키기 위해 사용합니다. 그러나 배치 정규화는 많은 외부 요인에 의해 영향을 받습니다. 그리고 Transformer에서는 레이어 정규화가 사용됩니다. 따라서 이것 또한 교체하였고 0.1%p의 성능 향상을 가져다 줍니다.

- downsampling layer 분리

ResNet에서, 각 스테이지 시작 전 residual block에 의한 다운샘플링은 성공했습니다. Swin Trnasformer에서는 스테이지 사이에 다운샘플링 레이어를 더해줍니다. 따라서 우리 또한 이와 유사한 방식을 사용하며 이는 0.5%p의 성능 향상을 시켰으며 이는 Swin-Transformer의 성능을 이기는 수치입니다. 아래는 우리의 최종적인 모델입니다.

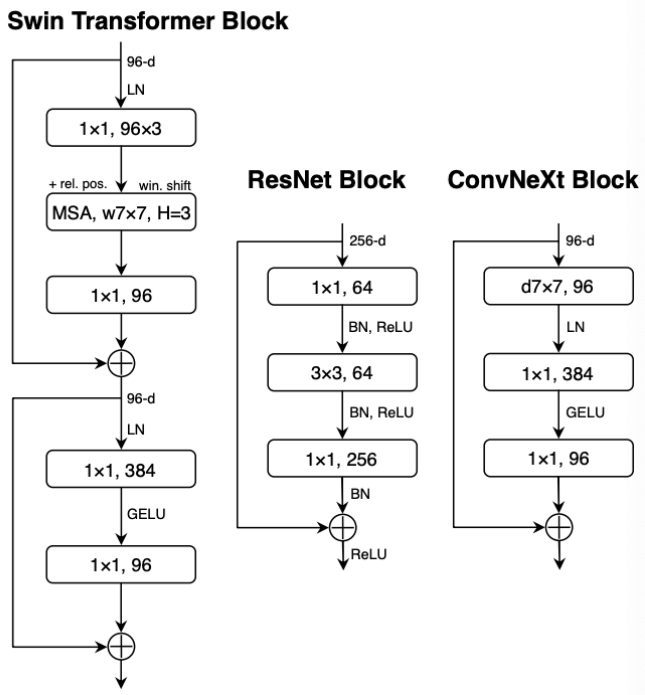

Figure 4. **Block designs** for a ResNet, a Swin Transformer, and a ConvNeXt. Swin Transformer's block is more sophisticated due to the presence of multiple specialized modules and two residual connections. For simplicity, we note the linear layers in Transformer MLP blocks also as "1×1 convs" since they are equivalent.

지금까지 우리는 Swin-Transformer에서 사용한 방식을 차용하여 ResNet을 변경하였습니다. Convolution도 많은 발전이 있었으나 이는 전부 개별적으로 실행되었습니다. 따라서 우리는 이러한 발전을 합쳤고, Swin-Transformer에 방식을 차용했습니다. 우리는 특별한 모듈을 사용하지 않고 Convolution이 Transformer를 이긴 것을 확인할 수 있습니다.

Empirical Evaluations on ImageNet

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**
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**

우리는 최종적으로 Swin-Transformer를 넘어서는 성능의 Convolution network인 ConvNeXt를 제시하며 여러 데이터 셋과 Task에서 높은 성능을 보였습니다.

Conclusions

2020년대에 들어, transformer 계열이 backbone model로 선호되기 시작했습니다. 그 이유는 더 정확하며, 효율적이며, 확장 가능했기 때문입니다. 우리는 ConvNeXt를 제안하여 sota의 transformer와 다양한 benchmarks에서 경쟁합니다. Convolution은 많은 발전이 있었지만 이러한 발전이 종합적으로 나타나지는 않았습니다. 이렇게 종합적으로 하였을 경우 Convolution은 상당한 성능을 보이며 CV분야에서의 Convolution에 대해서 사람들이 다시 생각해보았으면 합니다.