**A ConvNet for the 2020s**

Zhuang Liu et al.

**Abstract**

**ViT models** quickly superseded ConvNets as the SOTA image classification in 2020s. A vanilla ViT, on the other hand, faces difficulties when applied to general computer vision tasks such as **object detection** and **semantic segmentation**.

It is the hierarchical Transformers (e.g., **Swin Transformers**) that reintroduced several ConvNet priors, making Transformers practically viable as **a generic vision backbone** and demonstrating remarkable performance on **a wide variety of vision tasks**.

However, the effectiveness of **such hybrid approaches** is still largely credited **to the intrinsic superiority of Transformers**, rather than **the inherent inductive biases of convolutions**.

**ConvNeXt** is “modernized” a standard **ResNet** toward the design of **a vision Transformer**, and several key components that contribute to the performance difference along the away are discovered. In terms of **accuracy** and **scalability**, ConvNeXts can be competed with Transformer, 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**.

**Introduction**

A “**sliding window**” strategy is intrinsic to visual processing, particularly when working with high-resolution images.ConvNets have several built-in **inductive biases** that make them well-suited to a wide variety of computer vision applications.

The most important one is **translation equivariance**, which is a desirable property for tasks like objection detection. ConvNets are also inherently efficient due to the fact that when used in a sliding-window manner, **the computations are shared**. Except for the initial “**patchify**” layer, which splits an image into a sequence of patches, ViT introduces **no image-specific inductive bias** and makes minimal changes to the original NLP Transformers.

One primary focus of ViT is on the scaling behavior: with the helpof **larger model** and **dataset sizes**, Transformers can outperform standard ResNets by a significant margin. Those results on image classification tasks are inspiring, **but computer vision is not limited to image classification**.

**Without the ConvNet inductive biases**, **a vanilla ViT** model faces many challenges in being adopted as a generic vision backbone. **The biggest challenge is ViT’s global attention design**, which has **a quadratic complexity** with respect to the input size. This might be acceptable for ImageNet classification, but quickly becomes intractable with higher-resolution inputs.

Hierarchical Transformers (e.g., **Swin-T**) employ a hybrid approach to bridge this gap. For example, the “**sliding window**” strategy (e.g. **attention within local windows**) was reintroduced to Transformers, allowing them to behave more similarly to ConvNets. The success of this model revealed one thing: the essence of convolution is not becoming irrelevant; rather, it remains much desired and has never faded.

A naïve implementation of **sliding window self-attention** can be **expensive**; with advanced approaches such as cyclic shifting, the speed can be optimized but **the system becomes more sophisticated in design**. On the other hand, it is almost ironic that **a ConvNet already satisfies many of those desired properties**. The only reason ConvNets appear to be losing steam is that (hierarchical) **Transformers surpass them in many vision tasks**, and the performance difference is usually attributed to **the superior scaling behavior of Transformers**, with multi-head self-attention being the key component.

ConvNets and hierarchical vision Transformers become different and similar at the same time: they are both equipped with **similar inductive biases**, but **differ significantly in the training procedure and macro/micro-level architecture design**.

The authors investigate **the architectural distinctions** between ConvNets and Transformers and try to identify **the confounding variables** when comparing the network performance. Their 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**.

To do this, They gradually modernize the architecture to the construction of a hierarchical vision Transformer (e.g. Swin-T). Their exploration is directed by a key question: **How do design decisions in Transformers impact ConvNets’ performance?**

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.

**Modernizing a ConvNet: a Roadmap**

Compare between two models in terms of FLOPs, one is the ResNet-50/Swin-T regime with FLOPs around

and the other being ResNet-200/Swin-T regime which has FLOPs around .

Chart, bar chart, funnel chart

Description automatically generated A series of design decisions which are summarized as “**Macro design**”, “**ResNeXt**”, “**Inverted bottleneck**”, “**Large kernel size**”, “**Various layer-wise micro designs**” trained and evaluated on ImageNet-1K.

**1. Training Techniques**

The first step of exploration is to train a baseline model with the vision Transformer training procedure, in this case, **ResNet-50/200**.

A set of modern training techniques can significantly enhance the performance of **a simple ResNet-50 model**.

A training recip that is close to **DeiT**’s and **Swin Transformer**’s.

**Epochs** : 300 (from the original 90 epochs for ResNets)

**Optimizer** : AdamW

**Data augmentation** : Mixup, Cutmix, RandAugment, Random Erasing, and regularization schemes including Stochastic Depth and Label Smoothing. And **hyper-parameters**.

This enhanced training recipe increased the performance of the ResNet-50 model from 76.1% to 78.8% (**+2.7%**), implying that a significant portion of the performance difference between traditional ConvNets and vision Transformers may be due to the **training techniques**.

**2. Macro Design**

**Swin-T** follow ConvNets to use **a multi-stage design**, where each stage has a different feature map resolution. There are two interesting design considerations: **the stage compute ratio**, and **the “stem cell” structure**.

**Changing stage compute ratio.**

The original design of the computation distribution across stages in **ResNet** was largely **empirical**. The heavy “**res4**” stage was meant to be compatible with downstream tasks like **object detection**, where a detector head operates on the 14x14 feature plane. **Swin-T**, on the other hand, followed the same principle but with a slightly different stage ratio of **1:1:3:1**. For larger Swin-T, the ratio is **1:1:9:1**. Following the design, the number of blocks in each stage from (3,4,6,3) in **ResNet-50** to (**3,3,9,3**), which also aligns the FLOPs with Swin-T. This improves the model accuracy from 78.8% to 79.4% (**+0.6%**).

**Changing stem to “Patchify”.**

Due to **the redundancy inherent** in natural images, a common stem cell will aggressively **downsample the input images** to an appropriate feature map size in both standard ConvNets and vision Transformers.

In standard **ResNet**, the stem cell contains a **7x7 conv layer** with **stride 2**, followed by **max pool**, which results in a **4x downsampling** of the input images.

In **vision Transformers**, the stem cell contains a large kernel size (e.g. kernel size = **14** or **16**) and **non-overlapping** conv. (“**patchify**” strategy)

In **Swin Transformers**, the stem cell contains a smaller patch size of **4** to accommodate the architecture’s **multi-stage design**. (“**patchify**” strategy)

By replacing **the ResNet-style stem cell** with a patchify layer implemented using a **4x4** and **stride 4** convolutional layer(**non-overlapping conv**), the accuracy has changed from 79.4% to 79.5% (**+0.1%**). This suggests that the stem cell in a **ResNet** may be **substituted with a simpler “patchify” layer** a la **ViT** which will result in similar performance.

**3. ResNeXt-ify** (Depth-wise Separable Convolution[1])

**ResNeXt** which has **a better FLOPs/accuracy trade-off** than a vanilla ResNet. The core component is **grouped convolution**, where **the convolutional filters are** **separated into different groups**. At a high level, ResNeXt’s guiding principle is to “**use more groups, expand width**”. More precisely, ResNeXt employs grouped convolution **for the 3x3 conv layer in a bottleneck block**. As this significantly **reduces the FLOPs**, **the network width is expanded to compensate for the capacity loss**.

In this model, **the depthwise convolution** is used, a special case of grouped convolution where **the number of groups equals the number of channels**. **Depthwise convolution** is similar to **the weighted sum operation in self-attention**, which **operates on a per-channel basis**, i.e., only mixing information in the spatial dimension. **The combination of depthwise conv and 1x1 convs leads to a separation of spatial and channel mixing**, a property shared by vision Transformers, where **each operation either mixes information across spatial or channel dimension**, but not both. The use of depthwise convolution effectively **reduces the network FLOPs** from 4.4G to 2.4G (**-2.0G**)and **the accuracy** from 79.5% to 78.3% (**-1.2%**).

Following the strategy proposed in **ResNeXt**, ”**use more groups, expand width**”, 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% (**+2.2%**) with increased FLOPs (**5.3G**).

**Diagram

Description automatically generated4. Inverted Bottleneck** (Inverted Bottleneck -> Depthwise -> Inverted Bottleneck)

One important design in every Transformer block is that it creates **an inverted bottleneck**, i.e., **the hidden dimension of the MLP block** is **four times wider** than **the input dimension**. Interestingly, this Transformer design is connected to **the inverted bottleneck design** with **an expansion ratio of 4** used in ConvNets.

(a) is a ResNeXt block and (b) is what we create as an inverted bottleneck block. Despite the increased FLOPs for the depthwise convolution layer, this change reduces the whole network FLOPs from 5.3G to 4.6G (**-0.7G**), due to the significant FLOPs reduction in the **downsampling** **residual blocks’ shortcut 1x1 conv layer**. Interestingly, this results in slightly **improved performance** from 80.5% to 80.6% (**+0.1%**). In the ResNet-200 / Swin-B regime, this step brings even **more gain** from 81.9% to 82.6% (**+0.7%**) also **with reduced FLOPs**.

**5. Large Kernel Sizes**

One of the most distinguishing aspects of vision Transformers is their **non-local self-attention**, which enables each layer to have **a global receptive field**. While large kernel sizes have been used in the past with ConvNets, the gold standard is **to stack small kernel-sized** (**3x3**) conv layers, which have efficient hardware implementations on **modern GPUs**. Although **Swin-T** 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. Here we **revisit the use of large kernel-sized convolutions for ConvNets**.

**Moving up depthwise conv layer.**

One prerequisite is **to move up the position of the depthwise conv layer**, like (b) to (c). That is a design decision also evident in Transformers: **the MSA block** (Depth-wise conv layer in ConvNets) **is placed prior to the MLP layers**. As we have an inverted bottleneck block, this is a natural design choice – the complex/inefficient modules (**MSA**, large-kernel conv) will have **fewer channels**, while the efficient, dense 1x1 layers will do the heavy lifting. This intermediate step **reduces the FLOPs** from 4.6G to 4.1G (**-0.5G**), resulting in a temporary performance degradation from 80.6% to 79.9% (**-0.7%**).

**Incresing the kernel size.**

When we experimented with several kernel sizes, including 3,5,7,9 and 11, the network’s performance increases from 79.9% to 80.6% (**+0.7%**) in **kernel-sized (7x7)**, while the network’s **FLOPs stay roughly the same**. Intriguingly, a significant portion of the design choices taken in a **vision Transformer** may be **mapped** to ConvNet instantiations.

**6. Micro Design**

Most of the explorations here are done **at the layer level**, focusing on specific choices of **activation functions** and **normalization layers**.

**Diagram

Description automatically generated Replacing ReLu with GELU**

**One discrepancy** between **NLP** and **vision architectures** is the specifics of **which activation functions** to use.

**The Rectified Linear Unit** (ReLU) is still extensively used in **ConvNets** due to its simplicity and efficiency. ReLU is also used as an activation function **in the original Transformer paper**.

**The Gaussian Error Linear Unit** (GELU), which can be thought of as a smoother variant of ReLU, is utilized in the most advanced Transformers, including Google’s **BERT** and OpenAI’s **GPT-2**, and, most recently, **ViTs**.

**ReLU** can be substituted with **GELU** in ConvNeXt, although **the accuracy stay unchanged** (80.6%)

**Fewer activation functions.**

A Transformer block with key/query/value linear embedding layers, the projection layer, and two linear layers in an MLP block. There is only **one activation function** present in the **MLP block**. By eliminating all GELU layers from the residual block **except for one between two 1x1 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.

**Fewer normalization layers.**

By removing two BatchNorm (BN) layers, **leaving only one BN layer before the conv 1x1 layers**, this further boosts the performance from 81.3% to 81.4% (**+0.1%**), already **surpassing Swin-T’s result**. Note that we have even **fewer normalization layers per block** than Transformers, as empirically we find that **adding one additional BN layer** at the beginning of the block does **not improve the performance**.

**Substituting BN with LN.**

**BatchNorm** is an essential component in ConvNets as it **improves the convergence** and **reduces overfitting**. However, BN also has many intricacies that can have **a detrimental effect** on the model’s performance.

The simpler **Layer Normalization** (LN) has been used in Transformers, resulting in good performance across different application scenarios.

**Directly** substituting LN for BN in the original ResNet will result in **suboptimal performance**. **With all the modifications** in network architecture and training techniques, ConvNeXt model does not have any difficulties training with LN; in fact, **the performance is slightly better**, obtaining an accuracy of 81.5%.

**Separate downsampling layers.**

In **ResNet**, the spatial downsampling is achieved by **the residual block** at the start of each stage, using **3x3 conv** with **stride 2** (and **1x1 conv** with **stride 2** at the **shortcut connection**).

In **Swin-T**, a **separate downsampling layer** is added between stages.

In **ConvNeXt**, we explore **a similar strategy** in which we use **2x2 conv layers** with **stride 2** for spatial downsampling.

This modification leads to **diverged training**, and **adding normalization layers** wherever spatial resolution is changed can help **stabilize training**. These include **several LN layers** also used in **Swin-T**: one **before each downsampling layer**, one **after the stem**, and one **after the final global average pooling**. We can improve the accuracy from 81.5% to 82.0% (**+0.5%**), significantly exceeding Swin-T’s 81.3%.

We will use separate downsampling layers. This brings us to our final model, which we have dubbed ConvNeXt.

**Closing remarks.**

It is worth noting that all design choices discussed so far are adapted from vision Transformers. In addition, these designs are **not novel** even in the ConvNet literature – **they have all been researched separately, but not collectively, over the last decade**. **ConvNeXt** model has approximately the same **FLOPs**, **#params.**, **throughput**, and **memory use** as the Swin Transformer, but does **not require specialized modules** such as shifted window attention or relative position biases. But there has some questions of whether a ConvNet can compete with Swin-T on **downstream tasks** such as **object detection** and **semantic segmentation** is a central concern for computer vision practitioners.

**Empirical Evaluations on ImageNet**

Text

Description automatically generated

Constructing different ConvNeXt variants,

ConvNeXt-T/S/B/L is the end product of the “**modernizing**” procedure on ResNet-50/200 regime, respectively.

**Settings**

ImageNet-1K (consists of 1000 object classes with 1.2M training images) top-1 accuracy on the validation set.

Conduct pre-training on ImageNet-22K, a larger dataset of 21841 classes with ~14M images for pre-training, and then fin-tune the pre-trained model on ImageNet-1L for evaluation.

**Training on ImageNet-1K.**

Training ConvNeXts for **300 epochs** using **AdamW** with **a learning rate of 4e-3**.

There is a **20-epoch linear warmup** and **a cosine decaying schedule afterward**.

Using **a batch size of 4096** and **a weight decay of 0.05**.

**Adopting** common schemes including **Mixup**, **Cutmix**, **RandAugment**, and **Random Erasing** for data augmentations and **regularizing** the networks with **Stochastic Depth** and **Label Smoothing**.

**Layer Scale of initial value 1e-6** is applied.

Using **Exponential Moving Average** (EMA) **to** **alleviate larger models’ overfitting**.

**Training on ImageNet-22K.**

**Pre-training** ConvNeXts on ImageNet-22K for **90 epochs** with **a warmup of 5 epochs** and **doing not use EMA**.

**Fine-tuning on ImageNet-1K.**

**Fine-tuning** ImageNet-22K pre-trained models on ImageNet-1K for **30 epochs**. Using **AdamW**, **a learning rate of** **5e-5**, **cosine learning rate schedule**, **layer-wise learning rate decay**, **no warmup**, **a batch size of** **512**, and **weight decay of** **1e-8**. The default **pre-training**, **fine-tuning**, and **testing resolution** is . Fine-tuning at a **larger** resolution of , for both ImageNet-22K and ImageNet-1K pre-trained models.

Compared with ViTs/Swin Transformers, **ConvNeXts are simpler to fine-tune at different resolutions**, as the network is **fully-convolutional** and there is no need to adjust the input patch size of interpolate absolute/relative position biases.

**ResultsTable

Description automatically generated**Table

Description automatically generated

**Empirical Evaluation on Downstream Tasks**

**Object detection and segmentation on COCO.**

**Semantic segmentation on ADE20K.**

**Remarks on model efficiency.**

Under **similar FLOPs**, models with **depthwise convolutions** are known to be **slower** and consume **more memory** than ConvNets with only dense convolutions. As demonstrated throughout the paper, the inference throughputs of **ConvNeXts** are **comparable** to or **exceed** that of **Swin Transformers**. This is true for both **classification** and **other tasks requiring higher-resolution inputs**. Training ConvNeXts requires **less memory** than training Swin-T.

In comparison to **vanilla ViT**, both **ConvNeXt** and **Swin Transformer** exhibit a more favorable accuracy-FLOPs trade-off due to **the local computations**. It is worth noting that **this improved efficiency** is a result of **the ConvNet** **inductive bias**, and is not directly related to the **self-attention mechanism** in **vision Transformers**.

**Related Work**

**Hybrid models.**

In both the pre- and post-ViT eras, **the hybrid model** combining **convolutions and self-attentions** has been actively studied. **Prior to ViT**, the focus was **on augmenting a ConvNet with self-attention/non-local modules** to capture long-range dependencies. The original ViT first studied **a hybrid configuration**, and **a large body** of follow-up works focused on **reintroducing convolutional** priors to ViT, either in an explicit or implicit fashion.

Recent convolution-based approaches.

Local Transformer attention is equivalent to inhomogeneous dynamic depthwise conv. The MSA block in Swin-T is then replaced with a dynamic or regular depthwise conv, achieving comparable performance to Swin-T.

A concurrent work **ConvMixer** demonstrates that, **in small-scale settings**, **depthwise convolution** can be used **as a promising mixing strategy**. ConvMixer uses **a smaller patch size** to achieve the best results, making the **throughput much lower** than other baselines.

**GFNet** adopts **Fast Fourier Transform** (FFT) for token mixing. FFT is also a form of convolution, but with a global kernel size and circular padding.

**Conclusions**

We propose ConvNeXts, **a pure ConvNet model** that 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**.

**ConvNeXt** model itself is **not completely new** – many design choices have all **been examined separately** over the last decade, **but not collectively**.

We 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**.

**Mentioned Models in this Paper**

**Transformers**: vanilla ViT, Swin Transformers

**ConvNeXt**: AlexNet, VGGNet, Inceptions, ResNe(X)t, DenseNet, MobileNet, Xception, EfficientNet, RegNet

**Semantic segmentation** : SegNet (by Badrinarayanan et. a.), U-Net (by Ronnerberger et. al.),

Atrous Convolution (by DeepLab)

**Reference**

[1] Depth-wise Convolution (<https://coding-yoon.tistory.com/122>)