State Of Art CNNs

ConvNeXt

Introduction

The 20s of visual recognition began with the introduction of Vision Transformers (ViTs), which quickly superseded ConvNets as the state-of-the-art image classification model. A vanilla ViT though faces difficulties when applied to general computer vision tasks such as object detection and semantic segmentation. It is the hierarchical Transformers like Swin-Transformers that reintroduced several ConvNet priors in a hybrid approach, 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. *We gradually modernise a standard ResNet toward the design of a vision Transformer, and discover several key components that contribute to the performance difference along the way. The outcome of this exploration is a family of pure ConvNet models dubbed ConvNeXt.*

The full dominance of ConvNets in computer vision was not a coincidence as in many application scenarios, 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

Around the same time, a new design for natural language processing (NLP) models took the lead as the backbone of the domain, as the Transformers replaced Recurrent Neural Networks (RNNs). Despite the disparity in the task of interest between language and vision domains, the two streams surprisingly converged in the year 2020, as the introduction of Vision Transformers (ViT) completely altered the landscape of network architecture design. 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.

Challenges

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 employ a hybrid approach to bridge the gap between Transformers and ConvNets, by using the sliding window strategy and using the attention mechanism in local windows allowing them to behave more similarly to ConvNets. Under this perspective, many of the advancements of Transformers for computer vision have been aimed at bringing back convolutions. 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 behaviour of Transformers, with multi-head self-attention being the key component.

Challenges

These attempts, however, come at a cost as a naive implementation of sliding window self-attention can be expensive, with advanced approaches such as cyclic shifting , the speed can be optimised but the system becomes more sophisticated in design.

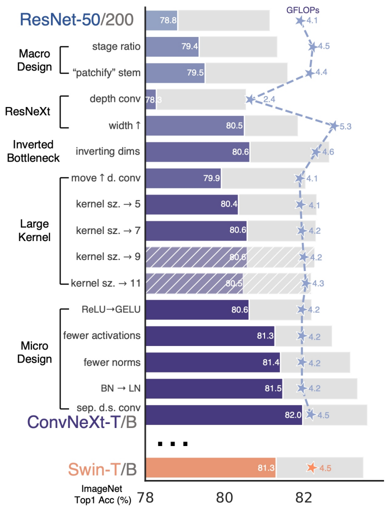
Unlike ConvNets, which have progressively improved over the last decade, the adoption of Vision Transformers was a step change. ConvNets and hierarchical vision Transformers become different and similar at the same time. They are both equipped with similar inductive biases, but differ significantly in the training procedure and macro/micro-level architecture design.

Challenges

Vision Transformers have shown rapid progress in computer vision tasks, achieving promising results on various benchmarks. However, due to the massive number of parameters and model design, *e.g.*, attention mechanism, ViT-based models are generally times slower than lightweight convolutional networks.

Modernising a ConvNet: a Roadmap

In this section, we provide a trajectory going from a ResNet to a ConvNet that bears a resemblance to Transformers. We consider two model sizes in terms of FLOPs, one is the ResNet-50 / Swin-T regime with FLOPs around 4.5×109 and the other being ResNet-200 / Swin-B regime which has FLOPs around 15.0 × 109. For simplicity, we will present the results with the ResNet-50 / Swin-T complexity models.



Training Techniques

Apart from the design of the network architecture, the training procedure also affects the ultimate performance. Recent studies demonstrate that a set of modern training techniques can significantly enhance the performance of a simple ResNet-50 model. In our study, we use a training recipe that is close to Swin Transformer's. The training is extended to 300 epochs from the original 90 epochs for ResNets.

We use the AdamW optimiser and data augmentation techniques such:

1. Mixup
2. Cutmix
3. RandAugment
4. Random Erasing
5. Regularisation schemes including Stochastic Depth and Label Smoothing

By itself, 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. Each reported accuracy on the ResNet-50 regime is an average obtained from training with three different random seeds

Macro Design

There are two interesting design considerations, the stage compute ratio, and the "stem cell" structure.

Changing stage compute ratio.

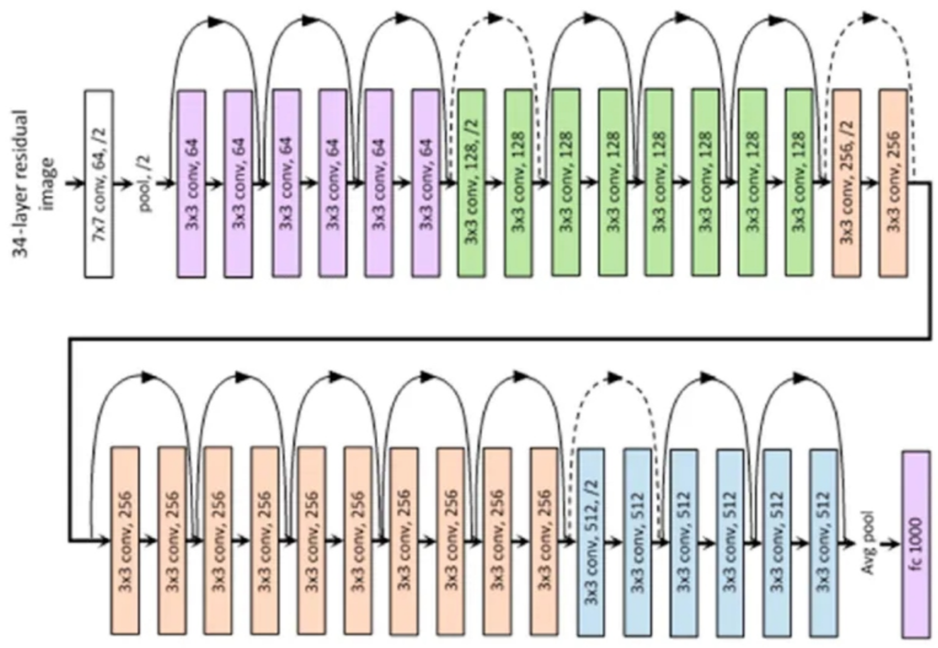
The original design of the computation distribution across stages in ResNet was largely empirical. The heavy "res4" stage was meant to be compatible with downstream tasks like object detection, where a detector head operates on the 14×14 feature plane.

[Swin Transformer Architecture]

Swin-T on the other hand, followed the same principle but with a slightly different stage compute ratio of 1:1:3:1. For larger Swin Transformers, the ratio is 1:1:9:1. Following the design, we adjust 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”.

Typically, the stem cell design is concerned with how the input images will be processed at the network’s beginning. 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.

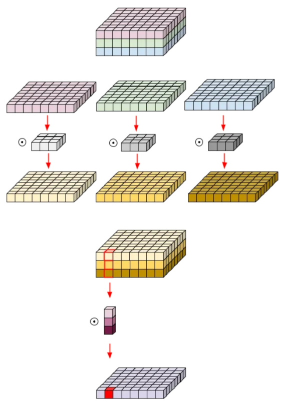


The stem cell in standard ResNet contains a 7×7 convolution layer with stride 2, followed by a max pool, which results in a 4× downsampling of the input images. In vision Transformers, a more aggressive patchify strategy is used as the stem cell, which corresponds to a large kernel size (kernel size = 14 or 16) and non-overlapping convolution. Swin Transformer uses a similar “patchify” layer, but with a smaller patch size of 4 to accommodate the architecture’s multi-stage design. We 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% (+0.1%). This suggests that the stem cell in a ResNet may be substituted with a simpler “patchify” layer à la ViT which will result in similar performance.

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. 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 3×3 convlayer in a bottleneck block. As this significantly reduces the FLOPs, the network width is expanded to compensate for the capacity loss.

In our case we use depthwise convolution, a special case of grouped convolution where the number of groups equals the number of channels. Depth-wise conv has been popularised by MobileNet and Xception. We note that depthwise convolution is similar to the weighted sum operation in self-attention, which operates on a per-channel basis.



The combination of depthwise conv and 1 × 1 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. This brings the network performance to 80.5% with increased FLOPs (5.3G).

Inverted Bottleneck

One important design in every Transformer block is that it creates an inverted bottleneck. Interestingly, this Transformer design is connected to the inverted bottleneck design with an expansion ratio of 4 used in ConvNets. The idea was popularised by MobileNetV2, and has subsequently gained traction in several advanced ConvNet architectures.

Interestingly, this results in slightly improved performance (80.5% to 80.6%), also with reduced FLOPs.

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 [3×3] conv layers. Although Swin Transformers reintroduced the local window to the self-attention block, the window size is at least [7×7], significantly larger than the ResNe(X)t kernel size of [3×3].

Moving up depth wise conv layer.

To explore large kernels, one prerequisite is to move up the position of the depth-wise conv layer. That is a design decision also evident in Transformers: the MSA block is placed prior to the MLP layers. As we have an inverted bottleneck block, this is a natural design choice — the complex/inefficient modules will have fewer channels, while the efficient, dense 1×1 layers will do the heavy lifting. This intermediate step reduces the FLOPs to 4.1G, resulting in a temporary performance degradation to 79.9%.

Increasing the kernel size.

With all of these preparations, the benefit of adopting larger kernel-sized convolutions is significant. We experimented with several kernel sizes, including 3, 5, 7, 9, and 11. The network's performance increases from kernel size[3×3] with accuracy 79.9% to 80.6% and a kernel size of [7×7].

Additionally, we observe that the benefit of larger kernel sizes reaches a saturation point at 7×7.

Micro Design

In micro design we are focusing on specific choices of activation functions and normalisation layers.

Replacing ReLU with GELU

One discrepancy between NLP and vision architectures is the specifics of which activation functions to use. Numerous activation functions have been developed over time, but 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 utilised in the most advanced Transformers, including Google's BERT and OpenAI's GPT-2, and, most recently, ViTs. We find that ReLU can be substituted with GELU, although the accuracy stays unchanged (80.6%).

Fewer activation functions.

One minor distinction between a Transformer and a ResNet block is that Transformers have fewer activation functions. Consider 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. In comparison, it is common practice to append an activation function to each convolutional layer, including the 1 × 1 convs. This process improves the result by +0.7% to 81.3%, practically matching the performance of Swin-T.

Fewer normalisation layers.

Transformer blocks usually have fewer normalisation layers as well. 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% (+0.1%), already surpassing Swin-T's result.

Note that we have even fewer normalisation 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 BatchNorm with LayerNorm.

BatchNorm is an essential component in ConvNets as it improves the convergence and reduces overfitting, but also has many intricacies that can have a detrimental effect on the model's performance. On the other hand, the simpler Layer Normalisation 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, here we revisit the impact of using LN in place of BN, where the performance is slightly better, obtaining an accuracy of 81.5% (+0.1%).

Separate downsampling layers.

In ResNet, the spatial downsampling is achieved by the residual block at the start of each stage, using 3×3 conv with stride 2. In Swin Transformers, a separate downsampling layer is added between stages. We explore a similar strategy in which we use 2×2 conv layers with stride 2 for spatial downsampling.

These include several LN layers also used in Swin-Transformers: one before each downsampling layer, one after the stem, and one after the final global average pooling. We can improve the
accuracy to
82.0%, significantly exceeding Swin-T's 81.3%.

EfficientFormer

Introduction

Vision Transformers have shown rapid progress in computer vision tasks, achieving promising results on various benchmarks.

Challenges

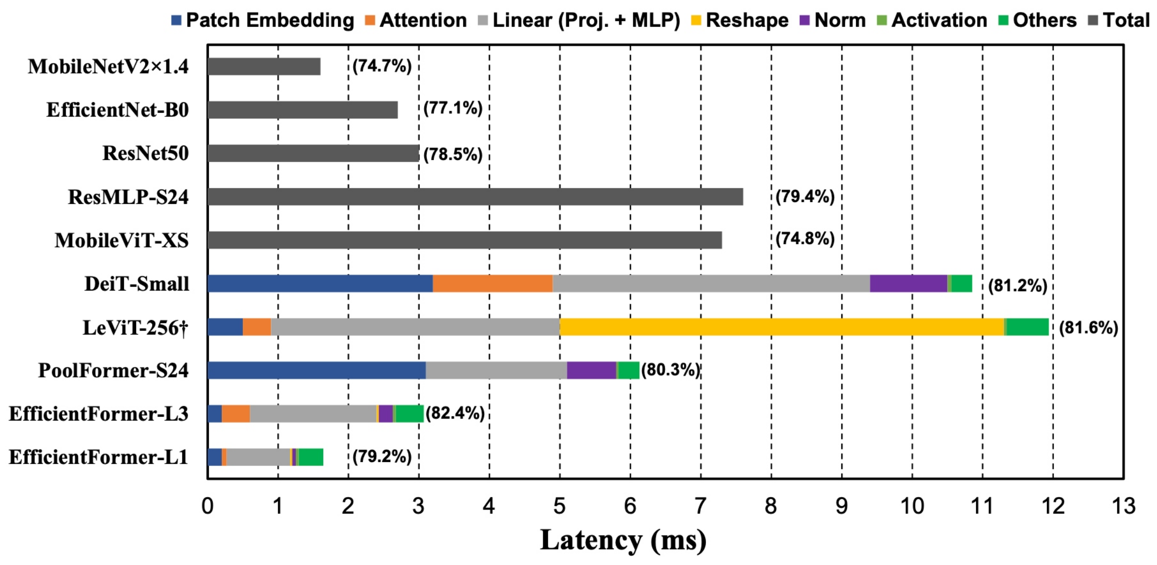
However, due to the massive number of parameters and model design, ViT-based models are generally times slower than lightweight convolutional networks. Recent efforts try to reduce the computation complexity of ViT through network architecture search or hybrid design with MobileNet block, yet the inference speed is still unsatisfactory.

The transformer architecture, initially designed for Natural Language Processing tasks, introduces the Multi-Head Self Attention-MHSA mechanism that allows the network to model long-term dependencies and is easy to parallelise. In this context, Dosovitskiy adapted the attention mechanism to 2D images and propose Vision Transformer (ViT), where the input image
is divided into non-overlapping patches, and the inter-patch representations are learned through
MHSA without inductive bias

We perform latency-driven slimming to get a series of final models dubbed EfficientFormer. Extensive experiments show the superiority of EfficientFormer in performance and speed on mobile devices. Our fastest model, EfficientFormer-L1, achieves 79.2% top-1 accuracy on ImageNet-1K with only 1.6 ms inference latency on iPhone 12, which runs as fast as MobileNetV2×1.4. These promising results demonstrate that latency is no longer an obstacle for the widespread adoption of vision transformers.

On-Device Latency Analysis of Vision Transformers

To have a clear understanding of which operations and design choices slow down the inference of ViTs on edge devices, we perform a comprehensive latency analysis over a number of models and operations as shown below:



Latency per Operation

Observation 1

Patch embedding with large kernel and stride is a speed bottleneck on mobile devices.

Patch embedding is often implemented with a non-overlapping convolution layer that has large kernel size and stride. A common belief is that the computation cost of the patch embedding layer in a transformer network is unremarkable or negligible. However, our comparison in the figure above between models with large kernel and stride for patch embedding, like DeiT-S and PoolFormer-S24, and the models without it LeViT-256 and EfficientFormer, shows that patch embedding is instead a speed bottleneck on mobile devices.

Large-kernel convolutions are not well supported by most compilers and cannot be accelerated
through existing algorithms like Winograd. Alternatively, the non-overlapping patch

embedding can be replaced by a convolution stem with fast downsampling that consists of several hardware-efficient 3 × 3 convolutions.

Observation 2:

Consistent feature dimension is important for the choice of token mixer. MHSA is not necessarily a speed bottleneck.

Selecting a token mixer is an essential design and the options are many, but we narrow the comparison to the two token mixers, **Pooling** and **MHSA**, where we choose the former for its simplicity and efficiency, while the latter for better performance. More complicated token mixers like shifted window are currently not supported by most public mobile compilers and we leave them outside our scope.

To understand the latency of the two token mixers, we perform the following two comparisons:

- First pooling naturally suits the 4D tensor when the network primarily consists of CONV-based implementations, *e.g.*, CONV 1 × 1 as MLP implementation and CONV stem for downsampling. As a result, PoolFormer exhibits faster inference speed.
- Second, by comparing DeiT-Small [3] and LeViT-256 [23], we find that MHSA does not bring
significant overhead on mobiles if the feature dimensions are consistent and Reshape is not
required. Though much more computation intensive, DeiT-Small with a consistent 3D feature can achieve comparable speed to the new ViT variant, *i.e.*, LeViT-256.

In this work, we propose a dimension-consistent network (Sec. 4.1) with both 4D feature implementation and 3D MHSA, but the inefficient frequent Reshape operations are eliminated.

Observation 3

CONV-BN is more latency-favourable than LN (GN)-Linear and the accuracy drawback is generally acceptable

Usually, one of the two options is selected:

1. Layer Normalisation with 3D linear projection
2. CONV 1 × 1 with Batch Normalisation

CONV-BN is more latency favourable because BN can be folded into the preceding convolution for inference speedup, while dynamic normalisations, such as LN and GN, still collects running statistics at the inference phase, thus contributing to latency. From the analysis of DeiT-Small and PoolFormer-S24 in Fig. 2 and previous work, the latency introduced by LN constitutes around 10%−20% latency of the whole network.

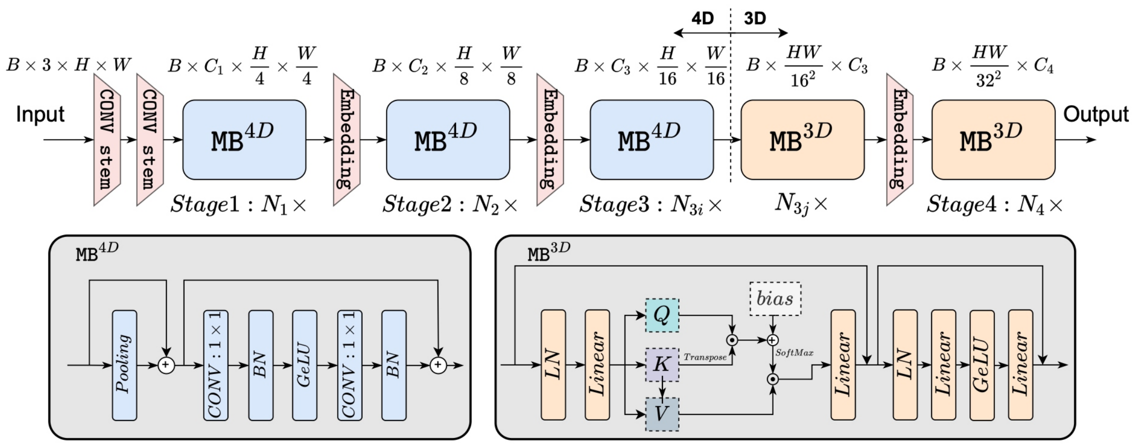
Observation 4

The latency of nonlinearity is hardware and compiler dependent.

Lastly, we study nonlinearity, including GeLU, ReLU, and HardSwish. Previous work suggests
GeLU is not efficient on hardware and slows down inference. However, we observe GeLU is well
supported by iPhone 12 and hardly slower than its counterpart, ReLU. On the contrary, HardSwish is surprisingly slow in our experiments and may not be well supported by the compiler.

Given all the observations above and the decisions made the architecture of the model is the

following:



Mobile-Former

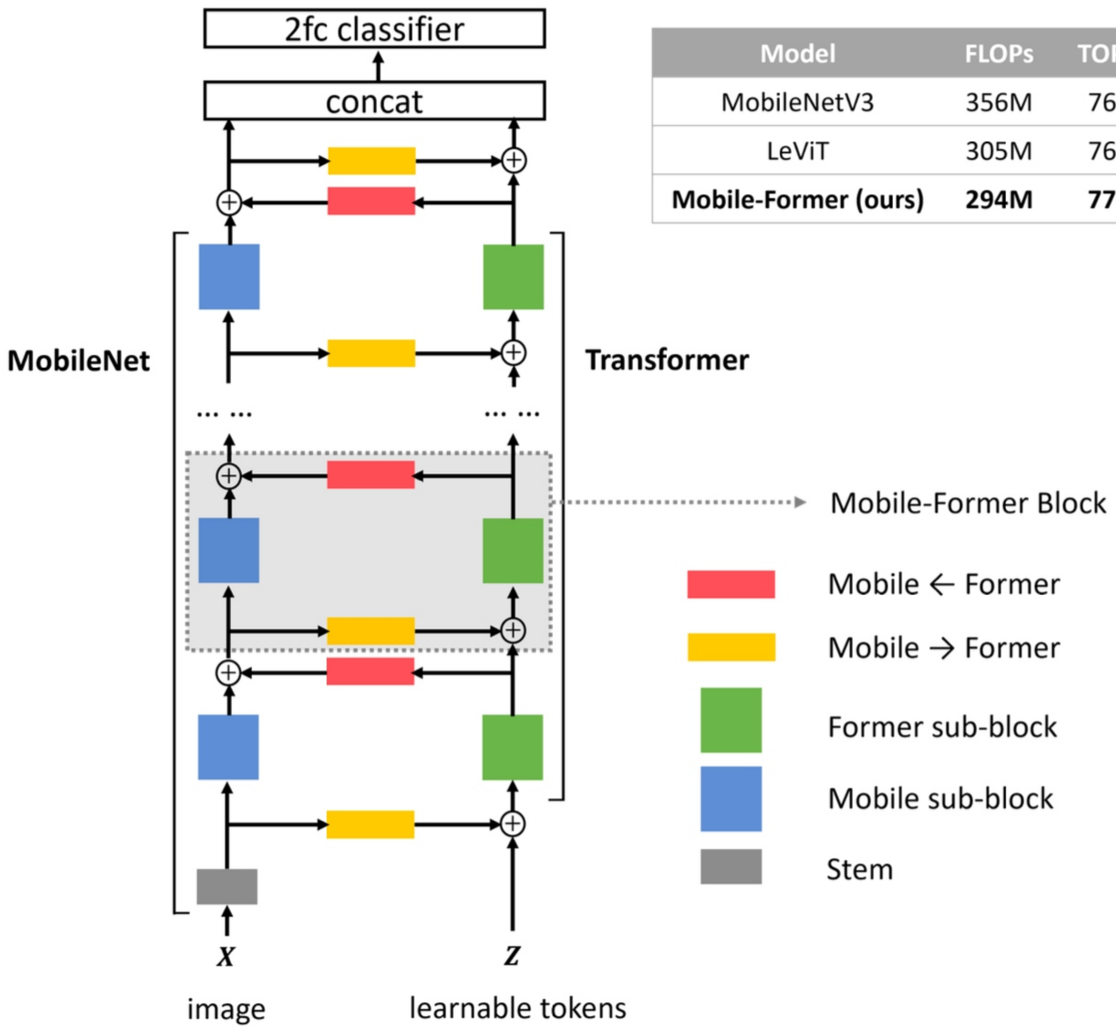
We present Mobile-Former, a parallel design of MobileNet and transformer with a two-way bridge in between. This structure leverages the advantages of MobileNet at local processing and transformer at global interaction. And the bridge enables bidirectional fusion of local and global features. Different from recent works on vision transformer, the transformer in Mobile-Former contains very few tokens that are randomly initialised to learn global priors, resulting in low computational cost. Combining with the proposed light-weight cross attention to model the bridge, Mobile-Former is not only computationally efficient, but also has more representation power.

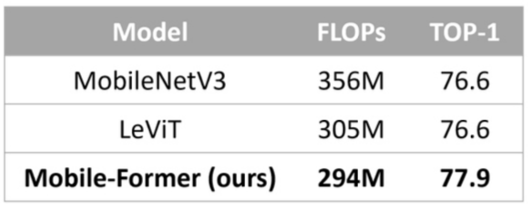
Recently, vision transformer ViT demonstrates the advantage of global processing and achieves significant performance boost over CNNs. However, when constraining the computational budget within 1G FLOPs, the gain of ViT diminishes.

Challenge

*How to design **efficient** networks to **effectively** encode both local processing and global interaction?*

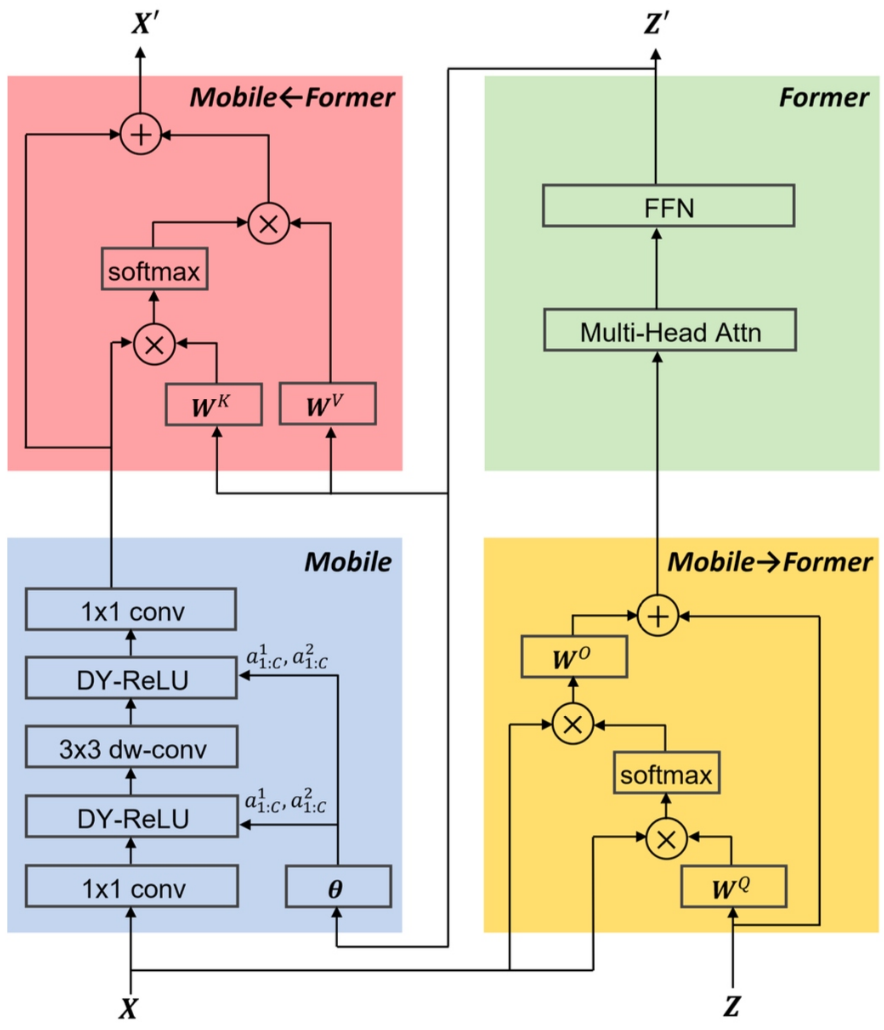
Mobile-Former takes an image as input and stacks mobile (or inverted bottleneck) blocks. It leverages the efficient depth-wise and point-wise convolution to extract local features. *Former* takes a few learnable tokens as input and stacks multi-head attention and feed-forward networks (FFN). These tokens are used to encode global features of the image.





Model	FLOPs	TOP-1
MobileNetV3	356M	76.6
LeViT	305M	76.6
Mobile-Former (ours)	**294M**	**77.9**

Mobile and *Former* communicate through a two-way bridge to fuse local and global features. This is crucial since it feeds local features to *Former's* tokens as well as introduces global views to every pixel of feature map in *Mobile*. We propose a light-weight cross attention to model this bidirectional bridge by performing the cross attention at the bottleneck of *Mobile* where the number of channels is low, and removing projections on query, key and value (W Q , WK, WV) from *Mobile* side.



This parallel structure with a bidirectional bridge leverages the advantages of both MobileNet and transformer. Decoupling of local and global features in parallel leverages MobileNet's efficiency in extracting local features as well as transformer's power in modeling global interaction. More importantly, this is achieved in an efficient way via a thin transformer with very few tokens and a light-weight bridge to exchange local and global features between *Mobile* and

Former. The bridge and *Former* consume less than 20% of the total computational cost, but significantly improve the representation capability. This showcases an efficient and
effective implementation of part-whole hierarchy.

Glossary

1. Vision Transformers → ViTs
2. AdamW → Adaptive Moment Estimation Weight Decay
3. ReLu → Rectified Linear Unit
4. GeLu → Gaussian Error Linear Unit
5. MHSA → Multi-Head Self Attention