Deep Learning for Computer Vision

More and Recent CNN Architectures

Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

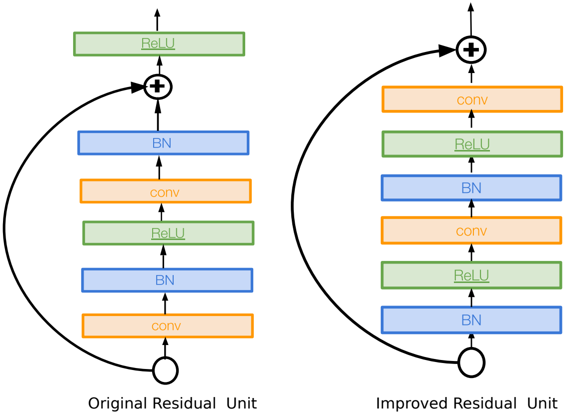
More and Recent CNN Architectures

We have already seen some deep convolutional architectures, including a very deep network that uses residual connections. Here we consider some other recent CNN architectures:

- ResNet Variants: Wide Residual Networks (WideResNet), ResNeXt
- Densely Connected Convolutional Networks (DenseNets)
- MobileNet, EfficientNet, SENet
- Recent architectures: ConvNext

Identity Mappings in Deep Residual Networks[1]

- Improved ResNet block design from creators of ResNet
- Switches up order of activations in the residual block
- Creates a more direct path for propagating information through the network
- Gives better performance

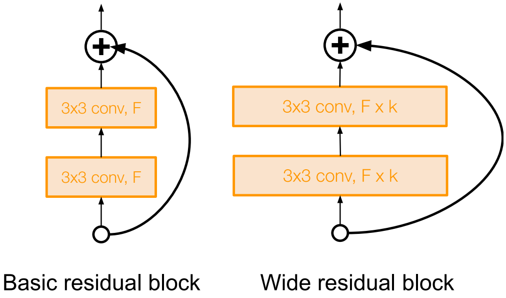


Credit: Fei-Fei Li, CS231n, Stanford Univ

[1] He et al, Identity Mappings in Deep Residual Networks, ECCV 2016

Wide Residual Networks[2]

- Builds on ResNets
- Argues that residuals are the important factor and not depth
- Uses wider residual blocks ($F \times k$ filters instead of F filters in each layer)
- 50-layer WideResNet outperforms 152-layer original ResNet
- Increasing width instead of depth computationally more efficient (parallelizable)

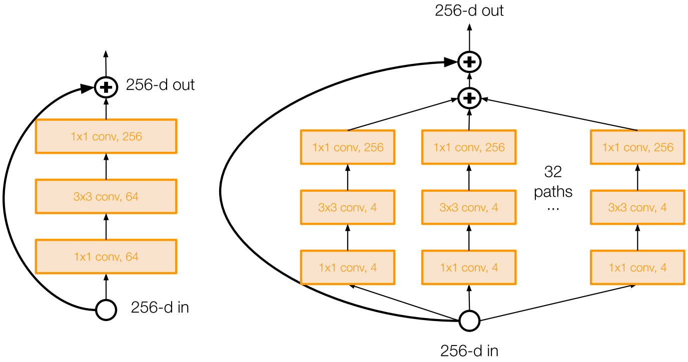


Credit: Fei-Fei Li, CS231n, Stanford Univ

[2]Zagoruyko and Komodakis, Wide Residual Networks, BMVC 2016

Aggregated Residual Transformations (ResNeXt)[3]

- Also from creators of ResNet
- Increases width of residual block through multiple parallel pathways (called **cardinality**)
- Parallel pathways similar in spirit to Inception module

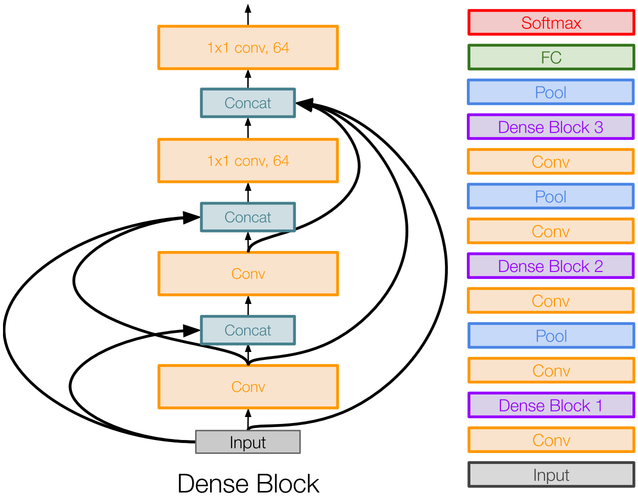


Credit: Fei-Fei Li, CS231n, Stanford Univ

[3] Xie et al, Aggregated Residual Transformations for Deep Neural Networks, CVPR 2017

Densely Connected Convolutional Networks (DenseNets)[4]

- **Dense blocks** where each layer is connected to every other layer in feedforward fashion
- Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse
- Showed that shallow 50-layer network can outperform deeper 152-layer ResNet
- Quite popularly in use today for image classification problems



[4]Huang et al, Densely Connected Convolutional Networks, CVPR 2017

MobileNets: Efficient Convolutional Neural Networks for Mobile Applications[5]

- A class of efficient models for mobile and embedded vision applications
- What are desirable properties of a network for use in small devices?

[5] Howard et al, MobileNets: Efficient Convolutional Neural Networks for Mobile Applications, 2017

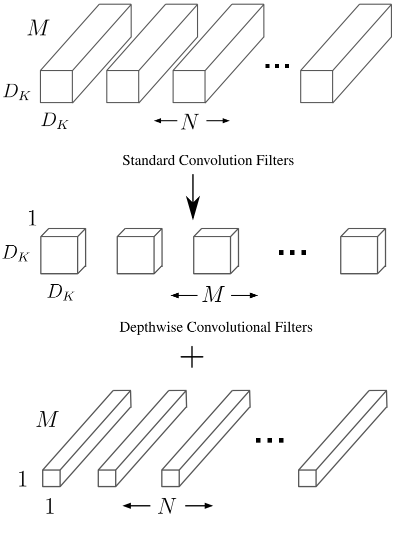
MobileNets: Efficient Convolutional Neural Networks for Mobile Applications[5]

- A class of efficient models for mobile and embedded vision applications
- What are desirable properties of a network for use in small devices?
  - Low latency
  - Low power consumption
  - Small model size (devices are low memory)
  - Sufficiently high accuracy
- **MobileNets** are small, low latency networks which are trained directly. A complementary approach to building efficient networks is compressing pre-trained networks.

[5] Howard et al, MobileNets: Efficient Convolutional Neural Networks for Mobile Applications, 2017

Key Ingredient: Depthwise Separable Convolutions

- MobileNets primarily built using **depthwise separable convolutions (DSC)**
- DSC replaces standard convolutions with depthwise convolution and 1×1 convolution
- DSC applies a single filter to each input channel; how does this help over normal convolution?



Depthwise Separable Convolutions

- Let input have size $D_f \times D_f \times M$ and output feature map (after passing input through conv layer) has $D_f \times D_f \times N$ size. Assume padded convolution. Let width of the square kernel in conv layer be k

Depthwise Separable Convolutions

- Let input have size $D_f \times D_f \times M$ and output feature map (after passing input through conv layer) has $D_f \times D_f \times N$ size. Assume padded convolution. Let width of the square kernel in conv layer be k
- A standard convolutional layer would have $k \times k \times M \times N$ parameters and a computational cost of $k \cdot k \cdot M \cdot N \cdot D_f \cdot D_f$

Depthwise Separable Convolutions

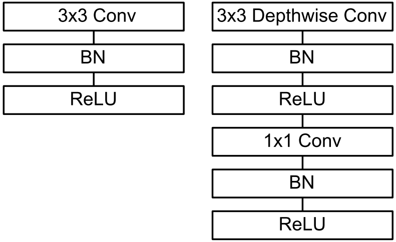
- Let input have size $D_f \times D_f \times M$ and output feature map (after passing input through conv layer) has $D_f \times D_f \times N$ size. Assume padded convolution. Let width of the square kernel in conv layer be k
- A standard convolutional layer would have $k \times k \times M \times N$ parameters and a computational cost of $k \cdot k \cdot M \cdot N \cdot D_f \cdot D_f$
- A depthwise separable conv layer factorizes the above into:
  - **Depthwise convolutions**, having $k \times k \times M$ parameters and a cost of $k \cdot k \cdot M \cdot D_f \cdot D_f$.
  - **Pointwise convolutions**, having $1 \times 1 \times M \times N$ parameters and cost of $M \cdot N \cdot D_f \cdot D_f$.
- By what fraction is computation reduced when DSC is used?

Depthwise Separable Convolutions

- Let input have size $D_f \times D_f \times M$ and output feature map (after passing input through conv layer) has $D_f \times D_f \times N$ size. Assume padded convolution. Let width of the square kernel in conv layer be k
- A standard convolutional layer would have $k \times k \times M \times N$ parameters and a computational cost of $k \cdot k \cdot M \cdot N \cdot D_f \cdot D_f$
- A depthwise separable conv layer factorizes the above into:
  - **Depthwise convolutions**, having $k \times k \times M$ parameters and a cost of $k \cdot k \cdot M \cdot D_f \cdot D_f$.
  - **Pointwise convolutions**, having $1 \times 1 \times M \times N$ parameters and cost of $M \cdot N \cdot D_f \cdot D_f$.
- By what fraction is computation reduced when DSC is used? Homework!
- Depthwise convolutions filter feature maps channelwise, and pointwise convolutions combine feature maps across channels; standard convolutions do these operations together

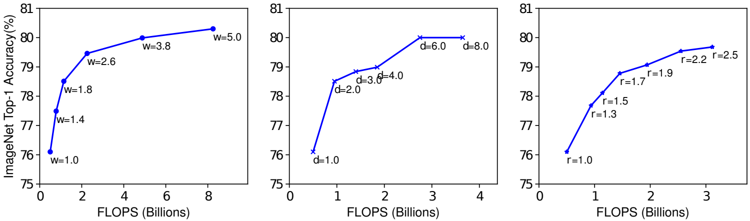
MobileNet: Architecture and Hyperparameters

- MobileNet built of many depthwise convolutions and pointwise convolutions, each of which is followed by BatchNorm and ReLU
- To obtain faster and smaller models, two more hyperparameters are considered:
  - **Width multiplier**, α, controls number of channels, making the number of input channels as αM and number of output channels as αN for all layers
  - **Resolution multiplier**, ρ, scales input image to a fraction of its size



Left: Standard conv layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks[6]



Scaling up a Baseline model with different network width (**w**), depth (**d**) and input resolution (**r**). Bigged networks with larger width, height and input resolution perform better but accuracy gain saturates.

- Conventional wisdom suggests that scaling up CNN architectures would lead to better accuracy i.e deeper and wider networks perform better in general
- Explores a principled way to scale up a CNN to obtain better accuracy and efficiency

[6] Tan and Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, ICML 2019

EfficientNet

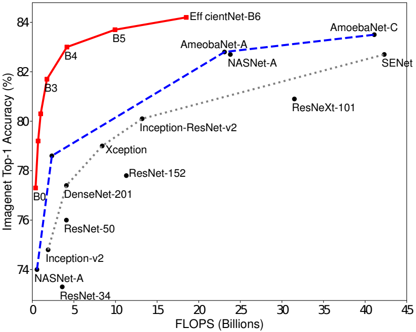
- Makes two observations:
  - Scaling up any dimension (w,d,r) independently improves accuracy, but return diminishes for bigger models
  - For better accuracy, critical to balance all dimensions during scaling; Intuitively, does it make sense to have deeper and wider models for larger input dimensions?

EfficientNet

- Makes two observations:
  - Scaling up any dimension (w,d,r) independently improves accuracy, but return diminishes for bigger models
  - For better accuracy, critical to balance all dimensions during scaling; Intuitively, does it make sense to have deeper and wider models for larger input dimensions?
- Based on these observations, a new **compound scaling method** is proposed
- A compound coefficient ϕ uniformly scales network width, depth and resolution

depth: $d = \alpha^{\phi}$
width: $w = \beta^{\phi}$
resolution: $r = \gamma^{\phi}$
s.t $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$
$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$
where α, β, γ are constants determined by a small grid search

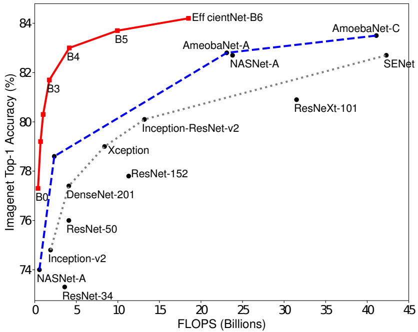
EfficientNet



FLOPS vs. ImageNet Accuracy

- For any new compound coefficient ϕ, total FLOPS will approximately increase by 2^{ϕ}. Why?

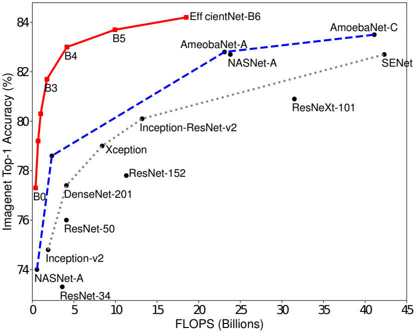
EfficientNet



FLOPS vs. ImageNet Accuracy

- For any new compound coefficient ϕ, total FLOPS will approximately increase by 2^{ϕ}. Why? Homework!
- Fixing $\phi = 1$ and assuming double the amount of resources, a grid search is performed on α, β, γ for chosen baseline network
- For every available computational budget, ϕ is calculated and model is scaled accordingly

EfficientNet



FLOPS vs. ImageNet Accuracy

- For any new compound coefficient ϕ, total FLOPS will approximately increase by 2^{ϕ}. Why? Homework!
- Fixing $\phi = 1$ and assuming double the amount of resources, a grid search is performed on α, β, γ for chosen baseline network
- For every available computational budget, ϕ is calculated and model is scaled accordingly
- Baseline model is obtained by performing Neural Architecture Search (an advanced topic we will see later in this course); scaling up this baseline leads to a family of models called EfficientNets

Squeeze-and-Excitation Networks (SENet)[7]

- **Motivation:** Improve quality of representations produced by network by explicitly modeling interdependencies between channels of its convolutional features
- Proposes a novel architectural unit termed **Squeeze-and-Excitation (SE) block**:
  - Squeeze operation embeds global information
  - Excitation operation re-calibrates feature maps channel-wise

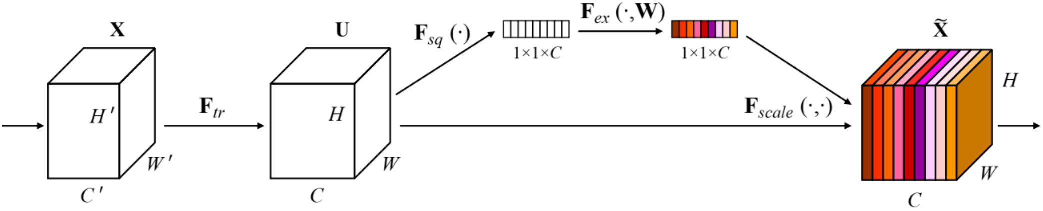
[7] Hu et al, Squeeze-and-Excitation Networks, CVPR 2018

Squeeze-and-Excitation Networks (SENet)[7]

- **Motivation:** Improve quality of representations produced by network by explicitly modeling interdependencies between channels of its convolutional features
- Proposes a novel architectural unit termed **Squeeze-and-Excitation (SE) block**:
  - Squeeze operation embeds global information
  - Excitation operation re-calibrates feature maps channel-wise
- If F_{tr} is a transformation mapping input $X \in \mathbb{R}^{H' \times W' \times C'}$ to output feature maps $U \in \mathbb{R}^{H \times W \times C}$, e.g. a convolution, then SE block squeezes and recalibrates U

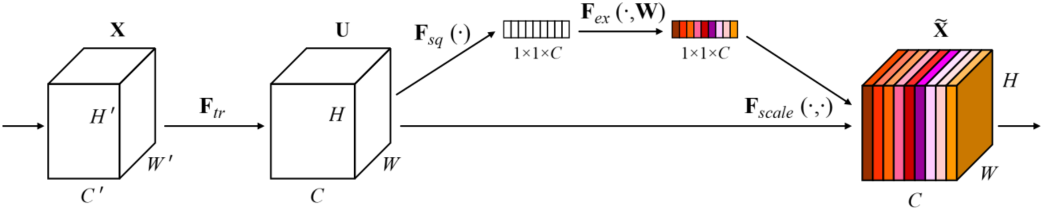
[7] Hu et al, Squeeze-and-Excitation Networks, CVPR 2018

SENet: Squeeze-and-Excitation Block



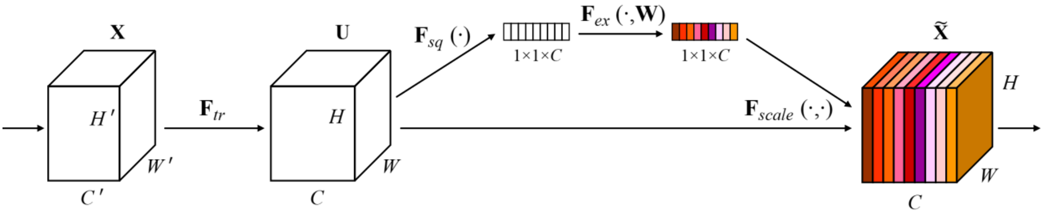
- Learns to reweigh feature maps (using global information) in a way that emphasises informative features and inhibits less useful ones.
- F_{sq}, the **squeeze function**, is channel-wise **global average pooling** - globally aggregate feature maps spatially

SENet: Squeeze-and-Excitation Block



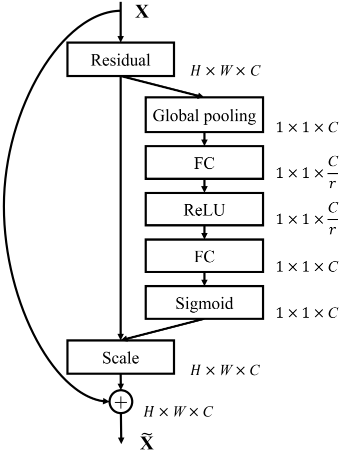
- Learns to reweigh feature maps (using global information) in a way that emphasises informative features and inhibits less useful ones.
- F_{sq}, the **squeeze function**, is channel-wise **global average pooling** - globally aggregate feature maps spatially
- F_{ex}, the **excitation function**, learns the relationships between channels, and outputs channelwise activations

SENet: Squeeze-and-Excitation Block



- Learns to reweigh feature maps (using global information) in a way that emphasises informative features and inhibits less useful ones.
- F_{sq}, the **squeeze function**, is channel-wise **global average pooling** - globally aggregate feature maps spatially
- F_{ex}, the **excitation function**, learns the relationships between channels, and outputs channelwise activations
- F_{scale} performs channelwise multiplication of feature maps U with learned activations

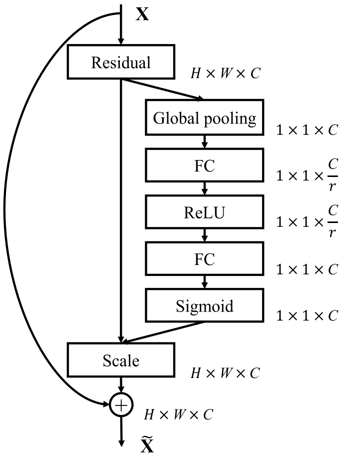
Squeeze-and-Excitation Block in ResNet



SE-ResNet Module

- r is a hyperparameter that controls size of hidden layer

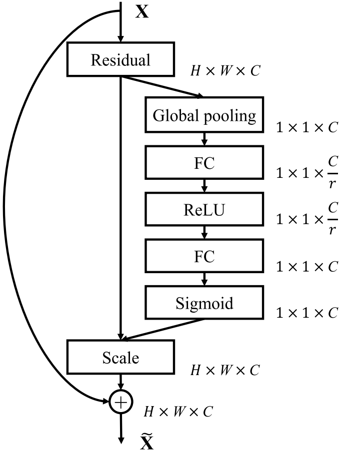
Squeeze-and-Excitation Block in ResNet



SE-ResNet Module

- r is a hyperparameter that controls size of hidden layer
- Output of F_{ex} is a set of C numbers between $(0, 1)$, each detailing how much attention each channel receives

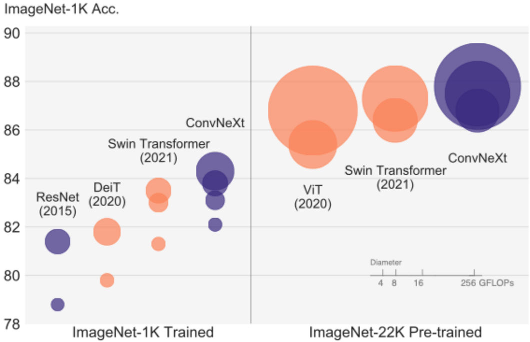
Squeeze-and-Excitation Block in ResNet



SE-ResNet Module

- r is a hyperparameter that controls size of hidden layer
- Output of F_{ex} is a set of C numbers between $(0, 1)$, each detailing how much attention each channel receives
- SE block is a cheap way to increase model depth
- Can be added to a wide variety of conv architectures, not just ResNet - to bring improvements to performance at minor additional computation cost

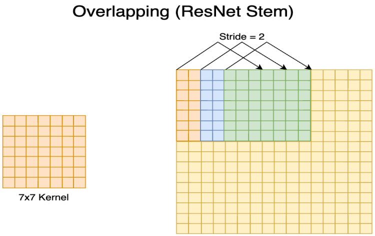
ConvNeXt: Recent CNN Architecture[8]

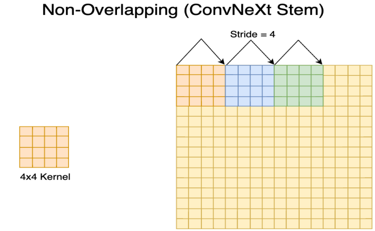


- **Introduction:** ConvNeXt is a modernized convolutional neural network (CNN) architecture developed recently, inspired by design choices from transformers.
- **Origin:** Developed by Facebook AI Research through a series of ablation studies on ResNet architectures.
- **Goal:** Enhance the performance of CNNs by integrating key ideas from transformers, data augmentation, and improved regularization techniques.

[8]Liu et al, A ConvNet for the 2020s, 2022

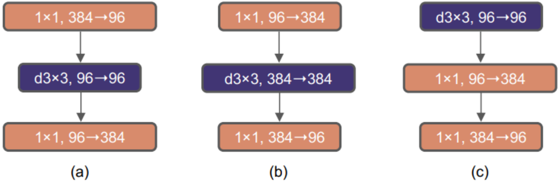
Patchify Stem in ConvNeXt





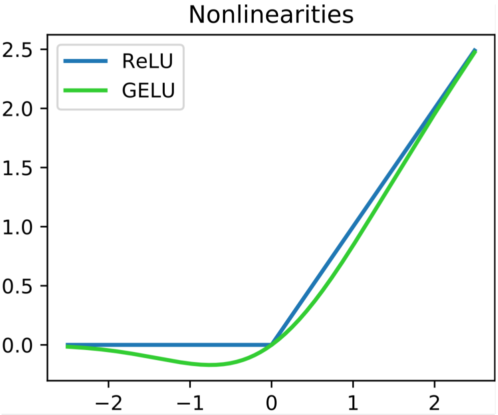
- **Patchify Stem:** Inspired by Vision Transformers (ViTs – to be covered later), ConvNeXt replaces traditional stem of early-stage convolutions with a "*patchify*" approach.
- **Non-overlapping Convolutions:** Convolution operations have kernel sizes equal to stride, ensuring that each patch is independent with no shared information between them.

Inverted Bottlenecks in ConvNeXt



- **Inverted Bottlenecks:** ConvNeXt uses inverted bottlenecks to first expand and then reduce dimensionality
- **ResNeXt Block:** (a) is based on ResNeXt, leveraging grouped convolutions
- **Inverted Bottleneck Block:** Modification (b) introduces an inverted bottleneck
- **Depthwise Convolution Shift:** Modification (c) repositions the depthwise convolution layer

Fewer & Different Activation Functions in ConvNeXt

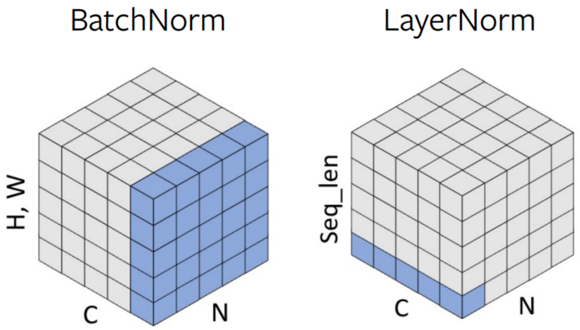


- **Reduced Activation Functions:** ConvNeXt reduces number of activation functions used throughout the network, streamlining the architecture
- **GELU Over ReLU:** ConvNeXt adopts the GELU (Gaussian Error Linear Unit) activation function instead of ReLU
- **GELU Activation Function:** Given by:

$$\text{GELU}(x) = x \cdot \Phi(x) = x \cdot \frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right]$$

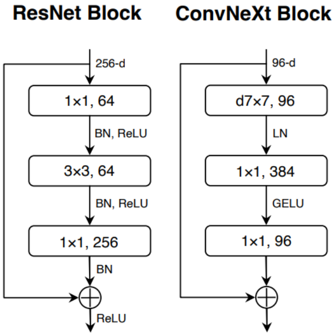
where $\text{erf}(x)$ is the Gaussian error function.

Normalization Layers in ConvNeXt



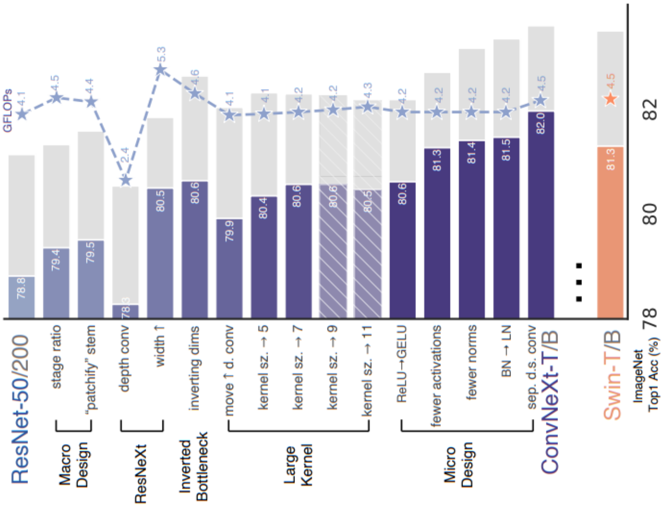
- **LayerNorm over BatchNorm:** ConvNeXt replaces BatchNorm with LayerNorm, inspired by transformer architectures. LayerNorm normalizes across the feature dimension, enhancing stability.
- **Reduction in Normalization Layers:** Network reduces the total number of normalization layers, simplifying the architecture while retaining performance.

Downsampling Layers in ConvNeXt



- **ResNet:** Downsampling is integrated within residual blocks at the start of each stage using 3×3 convolutions with stride 2 (and 1×1 convolutions with stride 2 for shortcuts)
- **ConvNeXt:** Uses 2×2 convolutions with stride 2 for downsampling, combined with normalization layers to stabilize training (inspired by Swin transformers – to be discussed later)

Resnet → ConvNeXt



Homework

Readings

- Lecture 9 of CS231n, Stanford Univ
- Google AI Blog on MobileNet
- Kungfu AI blog on ConvNext
- (Optional) ConvNext paper
- (Optional) Lecture 4 of Svetlana Lazebnik CS598 course, UIUC

Exercises

- By what fraction is computation reduced when DSC is used over standard convolution? (Slide 10)
- For a compound coefficient ϕ, total FLOPS will approximately increase by 2^{ϕ}. Why? (Slide 14)

References I

[1] Kaiming He et al. "Identity Mappings in Deep Residual Networks". In: *ArXiv* abs/1603.05027 (2016).

[2] Gao Huang et al. "Deep Networks with Stochastic Depth". In: *ECCV*. 2016.

[3] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks". In: *ArXiv* abs/1605.07146 (2016).

[4] A. Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *ArXiv* abs/1704.04861 (2017).

[5] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 2261–2269.

[6] Saining Xie et al. "Aggregated Residual Transformations for Deep Neural Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 5987–5995.

[7] Barret Zoph and Quoc V. Le. "Neural Architecture Search with Reinforcement Learning". In: *ArXiv* abs/1611.01578 (2017).

[8] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 4510–4520.

References II

[9] Barret Zoph et al. "Learning Transferable Architectures for Scalable Image Recognition". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 8697–8710.

[10] M. Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *ArXiv* abs/1905.11946 (2019).

[11] Jie Hu et al. "Squeeze-and-Excitation Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (2020), pp. 2011–2023.

[12] Lilian Weng. "Neural Architecture Search". In: *lilianweng.github.io/lil-log* (2020).

[13] Zhuang Liu et al. "A convnet for the 2020s". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11976–11986.

[14] Kungfu.ai. "ConvNeXt: A Transformer-Inspired CNN Architecture". In: *https://www.kungfu.ai/blog-post/convnext-a-transformer-inspired-cnn-architecture* (2023).