CNN Architectures

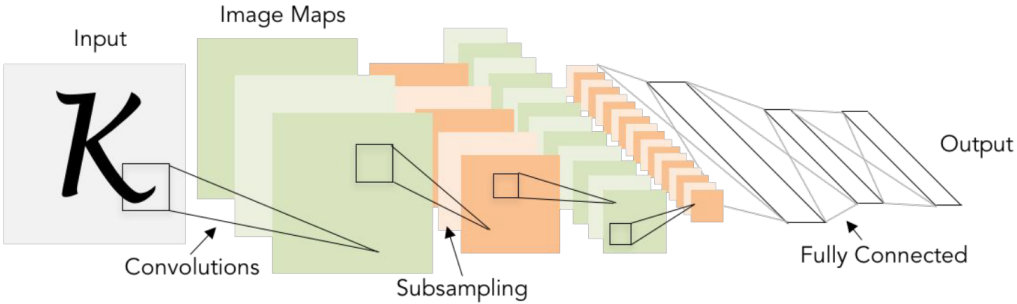
Praveen Kumar Chandaliya

Sardar Vallabhabhi National Institute of Technology, Surat

January 23, 2026

LeNet5

[LeCun et al., 1998]

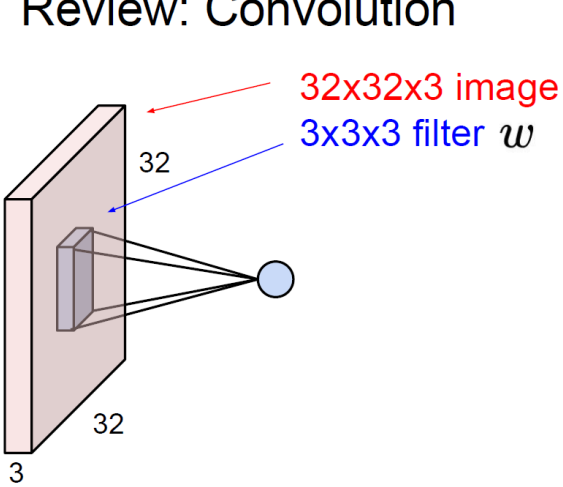


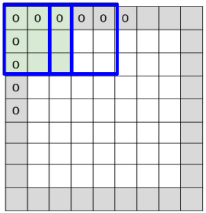
Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

1

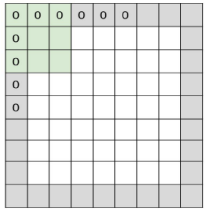
[1] Fei-Fei Li, Ranjay Krishna, Danfei Xu,CS213n Deep Learning for computer vision,Lecture 9

Review: Convolution



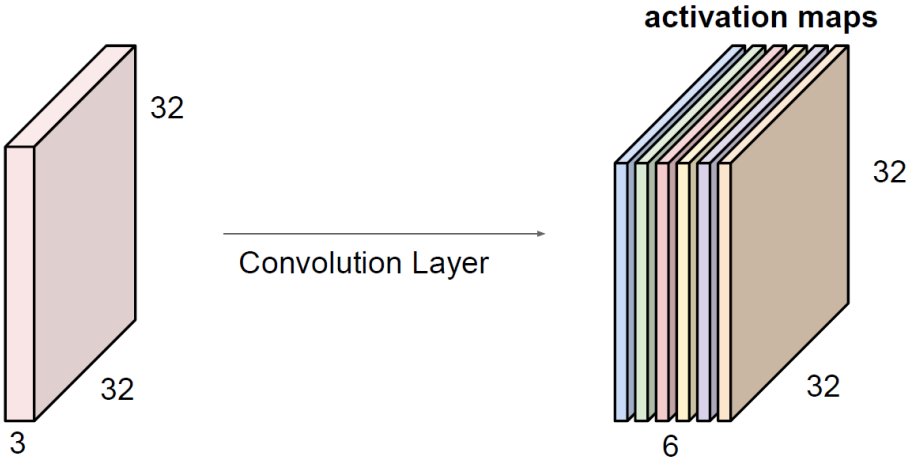


Stride: Downsample output activations



Padding: Preserve input spatial dimensions in output activations

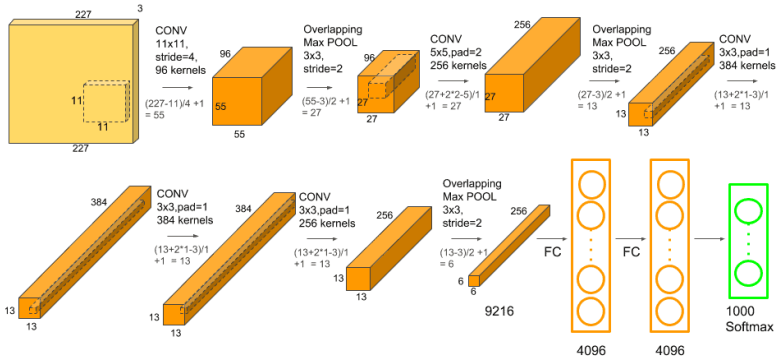
Review: Convolution



CNN Architectures

- AlexNet
- VGG
- GoogLeNet (Inception)
- ResNet
- DenseNet
- SENet
- Wide ResNet
- ResNeXt
- MobileNet
- NASNet
- EfficientNet

AlexNet



2

[2] https://learnopencv.com/understanding-alexnet/

AlexNet Parameters

Size / Operation	Filter	Depth	Stride	Padding	Number of Parameter
3* 227 * 227					
Conv1	11 * 11	96	4		(11*11*3 + 1) * 96=34944
96 * 55 * 55					
Max Pooling	3 * 3		2		
96 * 27 * 27					
Conv2	5 * 5	256	1	2	(5 * 5 * 96 + 1) * 256=614656
256 * 27 * 27					
Max Pooling	3 * 3		2		
256 * 13 * 13					
Conv3	3 * 3	384	1	1	(3 * 3 * 256 + 1) * 384=885120
384 * 13 * 13					
Conv4	3 * 3	384	1	1	(3 * 3 * 384 + 1) * 384=1327488
384 * 13 * 13					
Conv5	3 * 3	256	1	1	(3 * 3 * 384 + 1) * 256=884992
256 * 13 * 13					
Max Pooling	3 * 3		2		
256 * 6 * 6					
FC6					256 * 6 * 6 * 4096=37748736
4096					
FC7					4096 * 4096=16777216
4096					
FC8					4096 * 1000=4096000
1000 classes					
Overall					62369152=62.3 million
Conv VS FC	Conv:3.7million (6%) , FC: 58.6 million (94%)				

AlexNet

Key Idea

First deep CNN to win ImageNet, proving deep learning dominance.

- 8 layers: 5 convolution + 3 fully connected
- Introduced ReLU for faster training
- Used Norm layers (not common anymore)
- heavy data augmentation
- Used Dropout to reduce overfitting (0.5)
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- Trained on GPUs (Trained on GTX 580 GPU with only 3 GB of memory. Network spread across 2 GPUs, half the neurons (feature maps) on each GPU)

Limitation

Very large number of parameters ($\sim$60M)

What is ImageNet?

3

- **Core Concept:** Based on the **WordNet** hierarchy.
- **Scale:** Over 14 million manually annotated images.
- **Impact:** The primary benchmark for the "Deep Learning Revolution" since 2012.

Metric	Full ImageNet (21K)	ImageNet-1K
Total Images	∼14.2 Million	1,281,167 (Train)
Total Classes	21,841	1,000
Hierarchy	27 High-level categories	Flat 1,000 classes
Bounding Boxes	∼1.03 Million	∼150,000+

Table 1: Comparison of the Full vs. ILSVRC subsets.

[3] https://image-net.org/

ImageNet-1K: The Research Standard

Training Split

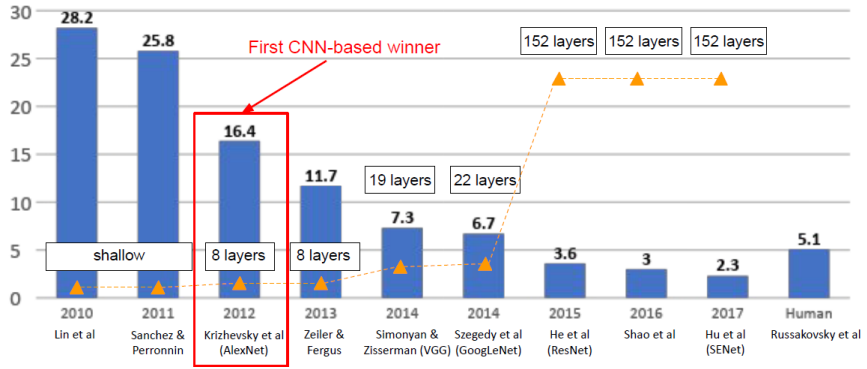
1,281,167 images used to train model weights.

Validation Split

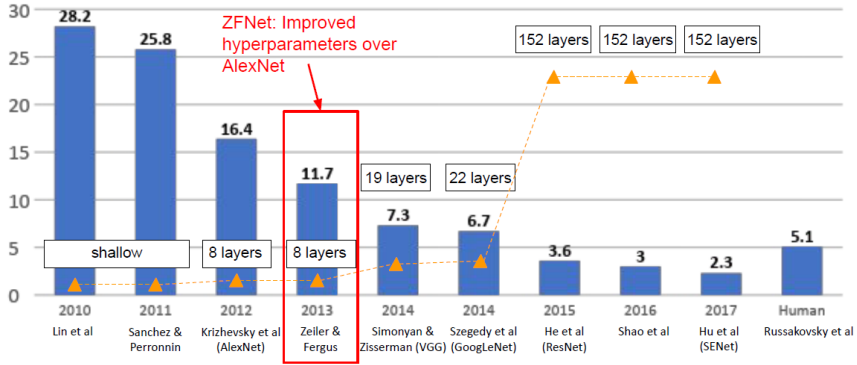
50,000 images (exactly 50 per class) used for hyperparameter tuning.

- Labels for the test set (100k images) remain private.
- Standard resolution: 224×224 or 256×256 pixels.

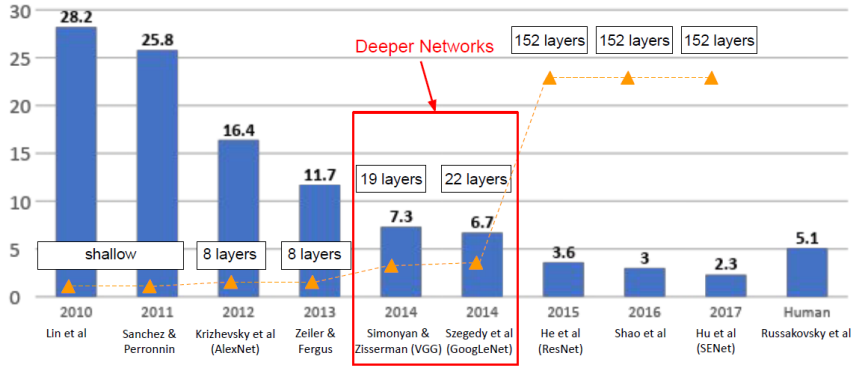
ImageNet Large Scale Visual Recognition Challenge



ImageNet Large Scale Visual Recognition Challenge



ImageNet Large Scale Visual Recognition Challenge



Performance Benchmarks: 2025

- **Top-1 Accuracy:** Elite models (ViT-G, MaX-ViT) reach **90–91%**.
- **Top-5 Accuracy:** Now frequently exceeds **98%**.
- **Pre-training Trend:** Modern models pre-train on ImageNet-21K and fine-tune on ImageNet-1K.

"ImageNet-1K accuracy is now used more for measuring efficiency rather than just raw capacity."

VGG

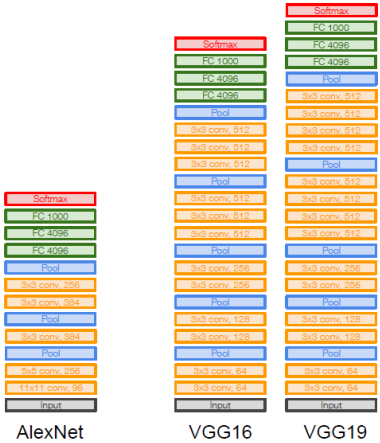
Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13 (ZFNet)
-> 7.3% top 5 error in ILSVRC'14

[Simonyan and Zisserman, 2014]

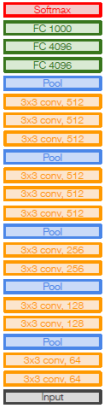


VGG FLOPS and Parameters

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 96MB / image (for a forward pass)
TOTAL params: 138M parameters



VGG16

VGG

Key Idea

Depth through simplicity: small filters, deep networks.

- Uses only 3×3 convolution filters
- VGG-16 and VGG-19 are popular variants
- Uniform architecture throughout
- ILSVRC'14 2nd in classification, 1st in localization
- Similar training procedure as Krizhevsky 2012 (AlexNet)
- Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
- Use ensembles for best results
- FC7 features generalize well to other tasks

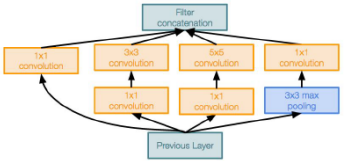
Limitation

Very high memory and computation cost

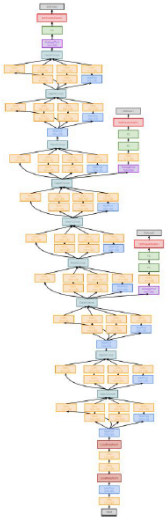
GoogLeNet (Inception):2014

Deeper networks, with computational efficiency

- ILSVRC'14 classification winner (6.7% top 5 error)
- 22 layers
- Only 5 million parameters!
  12x less than AlexNet
  27x less than VGG-16
- Efficient "Inception" module
- No FC layers



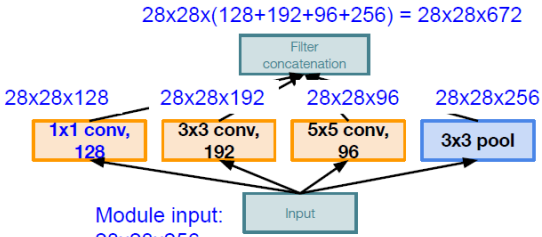
Inception module



[Szegedy et al., 2014]

GoogLeNet (Inception)

What is output size after filter concatenation?



Naive Inception module

[Szegedy et al., 2014]

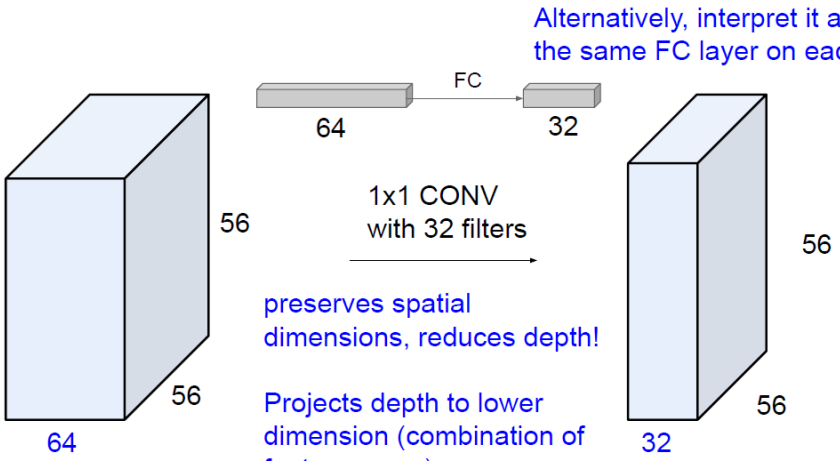
What is the problem with this?
[Hint: Computational complexity]

Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x**192x3x3x256**
[5x5 conv, 96] 28x28x**96x5x5x256**
Total: 854M ops

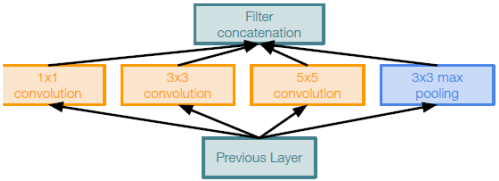
Very expensive compute

Pooling layer also preserves feature depth, which means total depth after concatenation can only grow at every layer!

1x1 Convolution

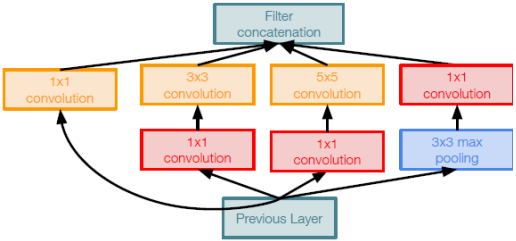


Naive Inception to dimension reduction



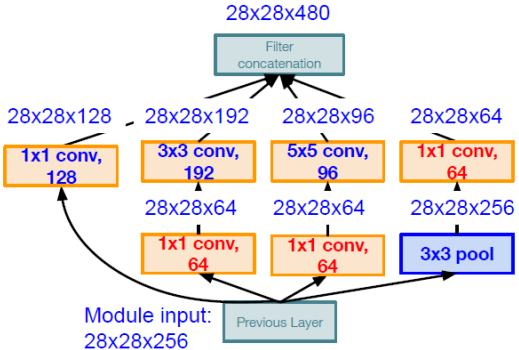
Naive Inception module

1x1 conv “bottleneck” layers



Inception module with dimension reduction

GoogLeNet



Inception module with dimension reduction

[Szegedy et al., 2014]

Using same parallel layers as naive example, and adding "1x1 conv, 64 filter" bottlenecks:

Conv Ops:

[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256

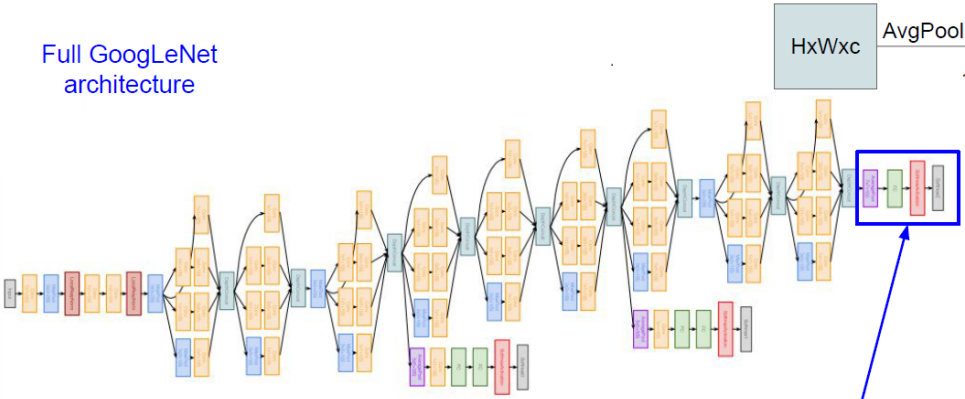
Total: 358M ops

Compared to 854M ops for naive version

Bottleneck can also reduce depth after pooling layer

GoogLeNet

Full GoogLeNet architecture



Note: after the last convolutional layer, a global average pooling layer is used that spatially averages across each feature map, before final FC layer. No longer multiple expensive FC layers!

Classifier output

[Szegedy et al., 2014]

GoogLeNet

Key Idea

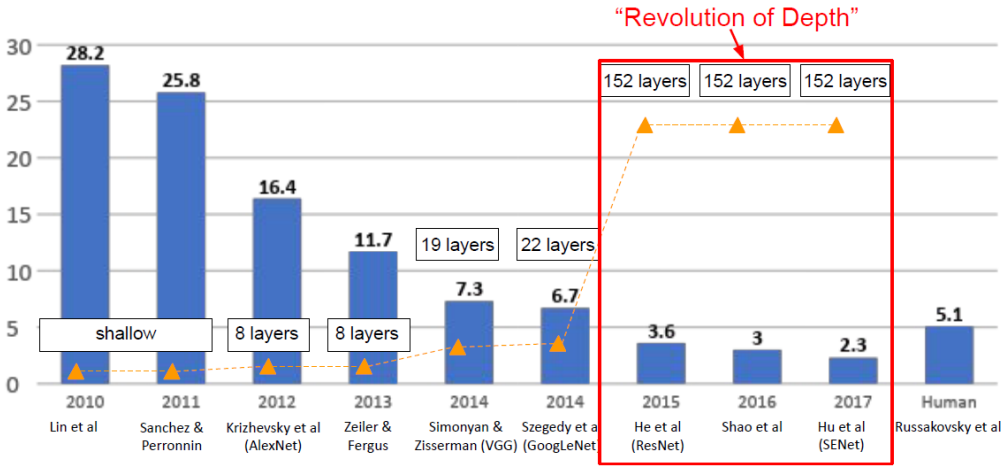
Multi-scale feature extraction in the same layer.

- Apply parallel filter operations on the input from previous layer (Inception module): Multiple receptive field sizes for convolution 1×1, 3×3, 5×5 convolutions
- Pooling operation (3×3)
- Concatenate all filter outputs together channel-wise
- Uses 1×1 convolution for dimensionality reduction
- "Inception module": design a good local network topology (network within a network) and then stack these modules on top of each other
- Much fewer parameters than VGG

Innovation

Efficient depth without computational explosion

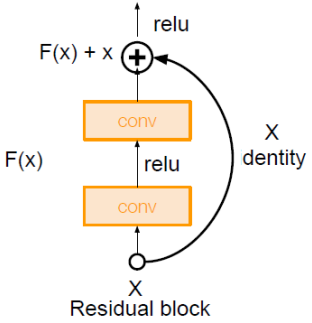
Revolution of Depth



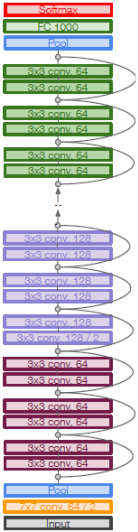
ResNet

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!



[He et al., 2015]



ResNet

Key Idea

Learning residuals instead of direct mappings.

- Skip connections: $y = F(x) + x$
- Enables very deep networks (up to 152 layers)
- Solves vanishing gradient problem

Impact

Foundation for most modern CNNs

DenseNet

Key Idea

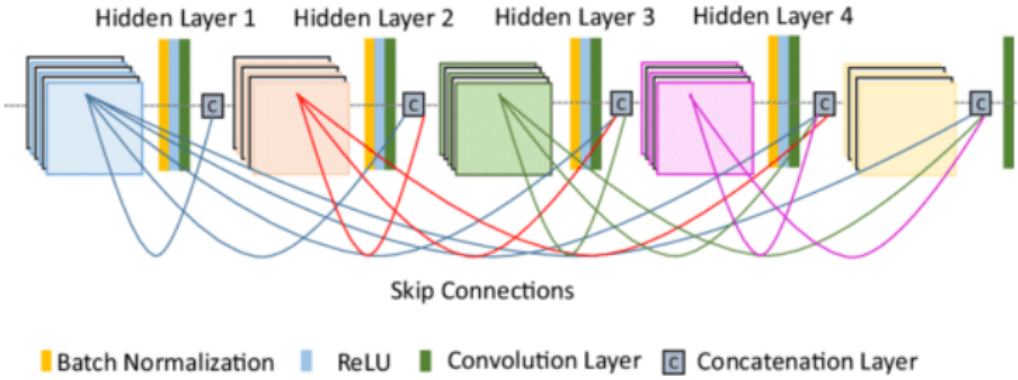
DenseNet, short for Densely Connected Convolutional Network. It revolutionized convolutional neural networks (CNNs) by introducing dense connectivity, where each layer is connected to every other layer in a feed-forward manner. This design improves gradient flow, feature reuse, and parameter efficiency, making DenseNet highly effective for tasks like image classification, object detection, and semantic segmentation. Each layer connects to all previous layers.

- Feature reuse improves efficiency
- Strong gradient flow
- Fewer parameters than ResNet

Trade-off

High memory usage due to dense connections

DenseNet121



SENet

Key Idea

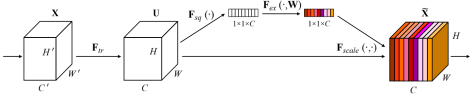
Channel-wise attention mechanism.

- Learns importance of feature channels
- Squeeze: global average pooling
- Excitation: channel re-weighting

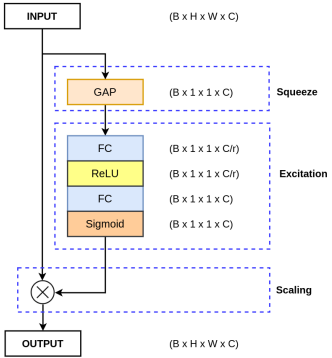
Advantage

Improves performance with minimal overhead

Squeeze-and-Excitation Networks



(a) SE Block – Squeeze Operation



(b) SE Block – Excitation Operation

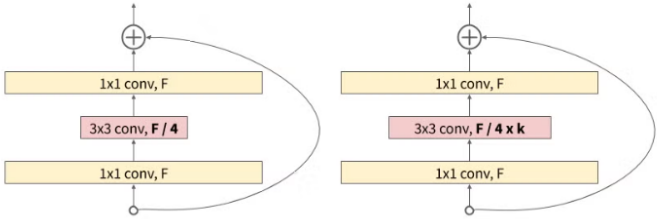
Figure 1: Architecture of the Squeeze-and-Excitation (SE) Block

Wide ResNet

Key Idea

Wider layers instead of deeper networks.

- Fewer layers, more feature maps
- Faster training than deep ResNet
- Better performance in practice



ResNet bottleneck

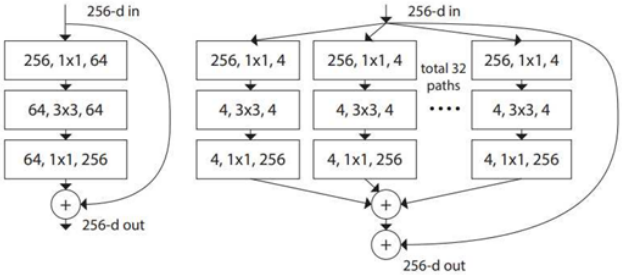
Wide ResNet bottleneck

ResNeXt

Key Idea

ResNeXt improves ResNet by introducing cardinality multiple parallel transformations aggregated together.

- Split-transform-merge strategy
- Depth → number of layers, Width → number of channels, Cardinality → number of parallel paths
- Improves accuracy without increasing complexity



MobileNet

Key Idea

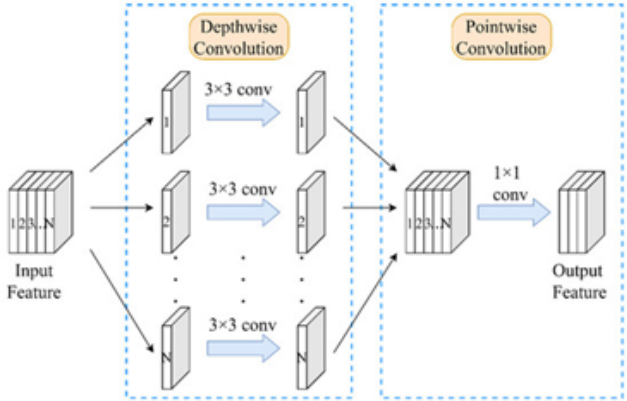
Efficient CNNs for mobile and edge devices.

- Depthwise separable convolutions
- Reduces computation by $\sim 9\times$
- Used in real-time applications
- MobileNetV1 (Depthwise separable convolution), MobileNetV2 (Inverted residuals + linear bottleneck), MobileNetV3 (NAS + SE attention + h-swish)

Use Case

Mobile, IoT, embedded AI

MobileNetV1



EfficientNet

Key Idea

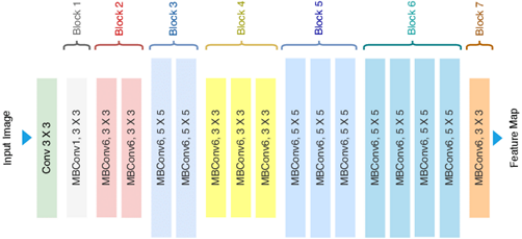
Compound scaling of depth, width, and resolution.

- Balanced scaling strategy
- EfficientNet-B0 to B7
- Excellent accuracy–efficiency tradeoff

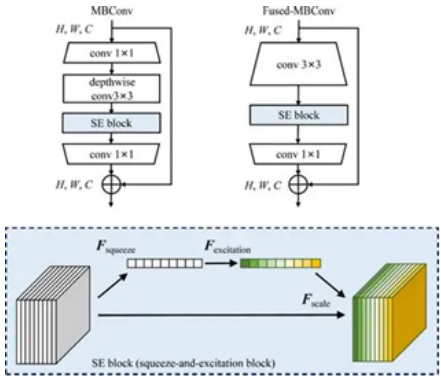
State-of-the-Art

Best performance per FLOP

EfficientNet



(a) EfficientNet



(b) Mobile Inverted Bottleneck blocks + Squeeze-and-Excitation

Figure 2: EfficientNet

CNN Architecture Comparison

- **AlexNet**: First deep CNN
- **VGG**: Depth via simplicity
- **GoogLeNet**: Multi-scale features
- **ResNet**: Skip connections
- **DenseNet**: Feature reuse
- **SENet**: Channel attention
- **MobileNet**: Lightweight models
- **EfficientNet**: Optimal scaling

CNN Architecture Comparison

Model	Params (M)	FLOPs (G)	Top-1 Acc. (%)
AlexNet	60	0.7	57.2
VGG-16	138	15.5	71.5
GoogLeNet	6.8	1.6	74.8
ResNet-50	25.6	4.1	76.0
DenseNet-121	8.0	2.9	74.9
SENet-154	115	20.7	81.3
Wide ResNet	68	11.4	78.5
ResNeXt-50	25	4.2	77.8
MobileNetV2	3.4	0.3	71.8
NASNet-A	88.9	23.8	82.7
EfficientNet-B7	66	37.0	84.4

Table 2: Comparison on ImageNet

CNN Architecture Comparison

- **AlexNet**: First deep CNN
- **VGG**: Depth via simplicity
- **GoogLeNet**: Multi-scale features
- **ResNet**: Skip connections
- **DenseNet**: Feature reuse
- **SENet**: Channel attention
- **MobileNet**: Lightweight models
- **EfficientNet**: Optimal scaling

Thank You