Evolution of CNN Architectures

AI with Deep Learning
EE4016

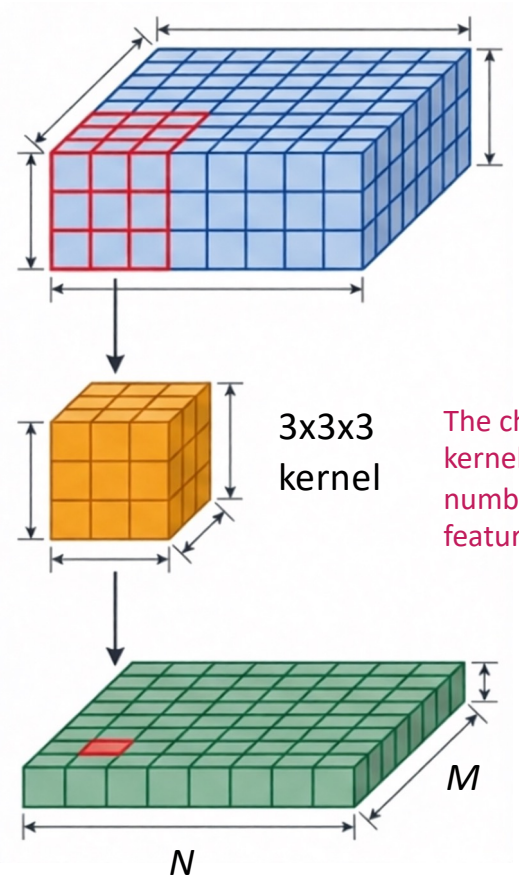
Prof. Lai-Man Po
Department of Electrical Engineering
City University of Hong Kong

Content

- **Variants of Convolutional Operation**
  - 1x1 Conv, Separable Conv, Transposed Conv, Dilated Conv, 3D Conv, Grouped Conv, etc.
- **Evolution of CNN Architectures**
  - LeNet, AlexNet, VGGNet, InceptionNet, ResNet, …
- **Transfer Learning**
  - Pre-training and fine-tuning
- **CNN-based Computer Vision Applications**

Variants of Convolutional Operation

The Baseline: Standard Convolution

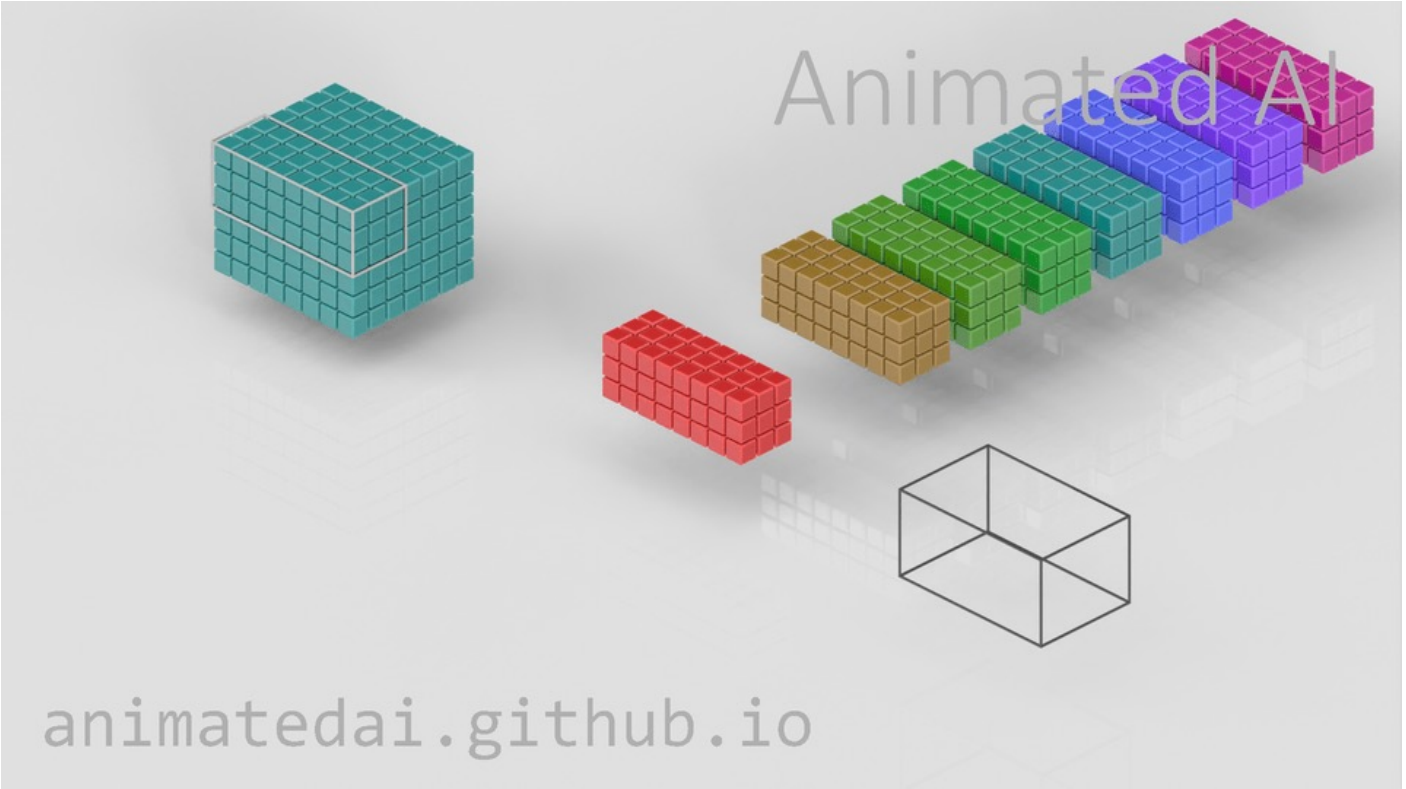


Mechanics:

- **Input**: 3 input channels processed simultaneously.
- **Kernel**: Spatial dimensions (3 × 3) extend through the full depth of the input volume (3 × 3 × 3).
- **Output**: One 2D feature map per filter.

Engineering Constraint:
Correlates spatial and channel-wise information in a single step. High computational cost ($N \times M \times K_1 \times K_2 \times K_3$)

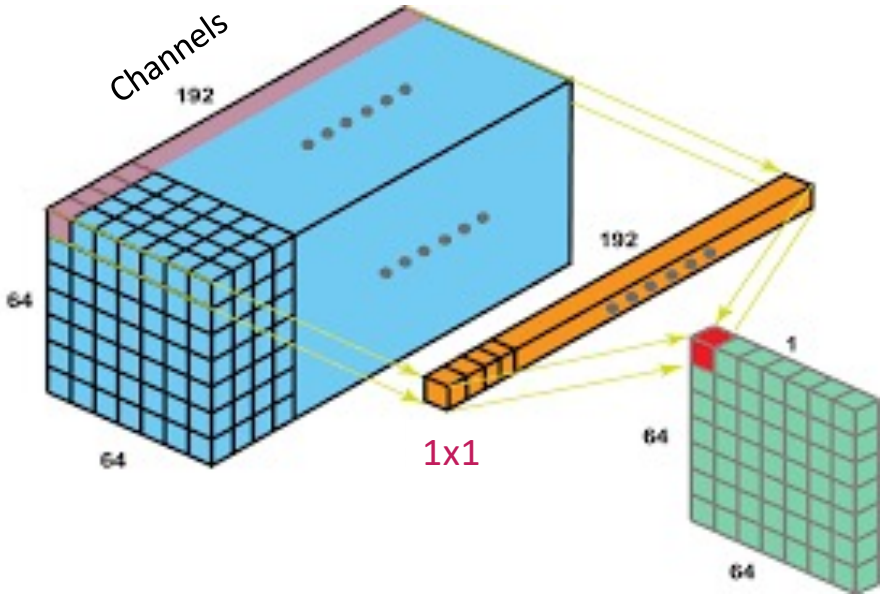
Standard Convolution Animation



Convolution Operation (Source)

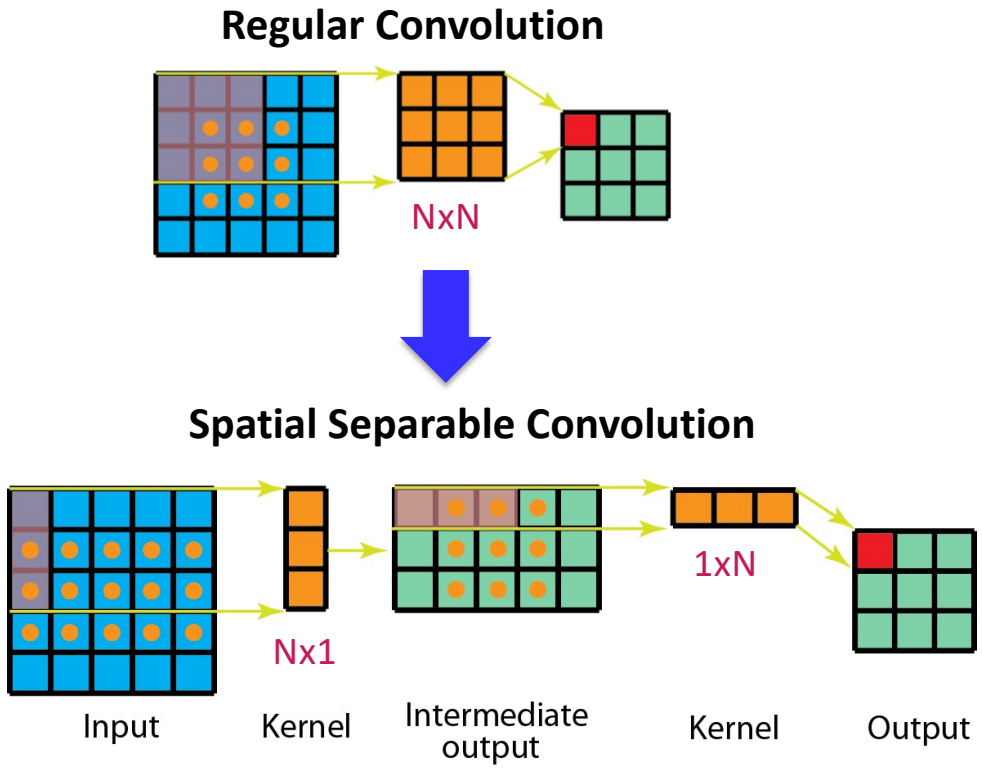
Pointwise Convolution: Controlling Dimensionality

- **Also Know As**: 1x1 Convolution.
- **The Engineering Win**: It plays a key role in reducing dimensionality, allowing for deeper networks without exploding parameter counts.
- **Real-World application**: Widely utilized in prominent architectures like GoogLeNet.
- **Capability**: Enables predictions at every spatial location (x, y) on feature maps, making it favored for semantic segmentation and object



Spatial Separable Convolution: Divide and Conquer

THE Concept: Decomposes a standard NxN kernel into two smaller kernels: Nx1 and 1xN.

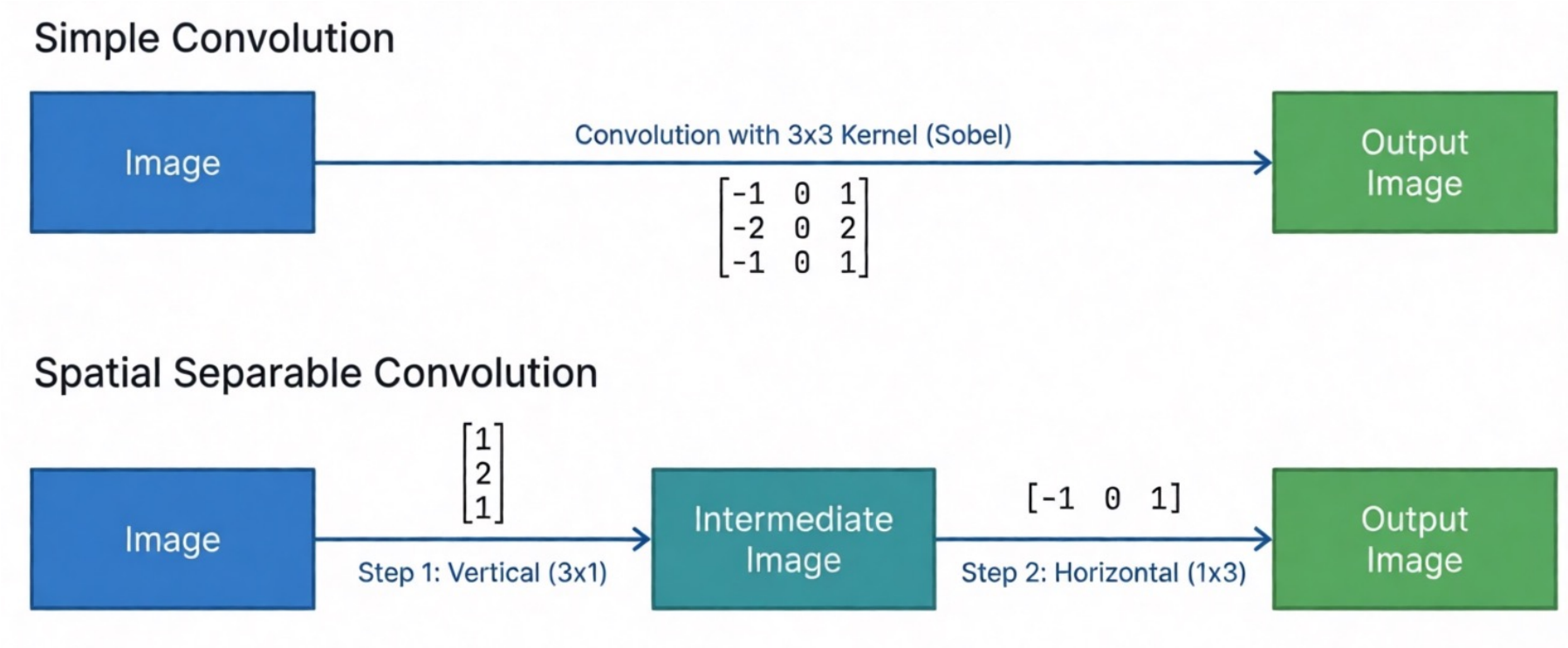


The Result: Significantly reduces computational complexity and enhances processing speed.

Example: The Sobel kernel in image processing is frequently split into 3x1 and 1x3 kernels for efficient computation

The constraint: This is not universally applicable; the kernel must have a Rank-1 matrix to be successfully separated.

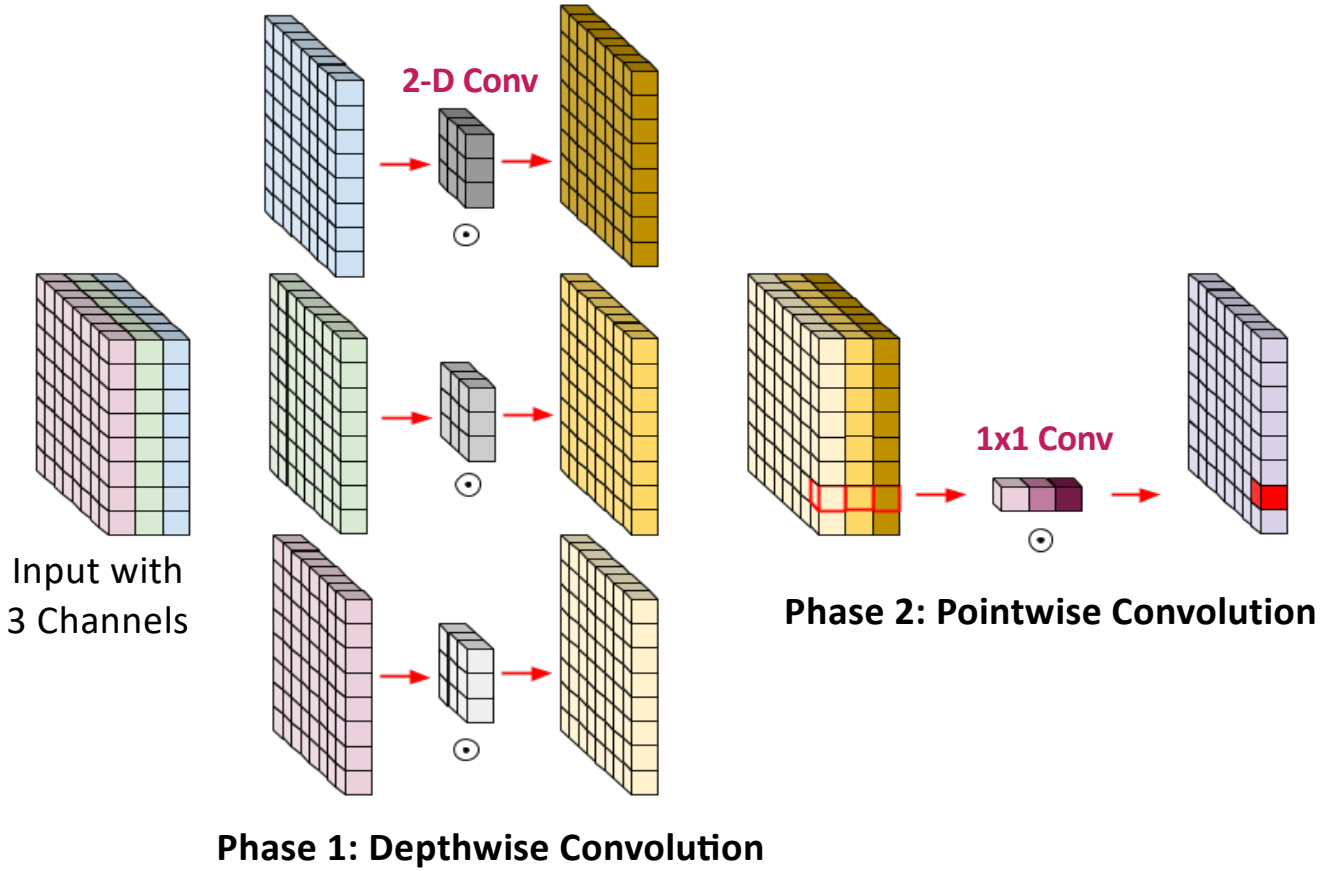
Spatial Separable Convolution in Practice



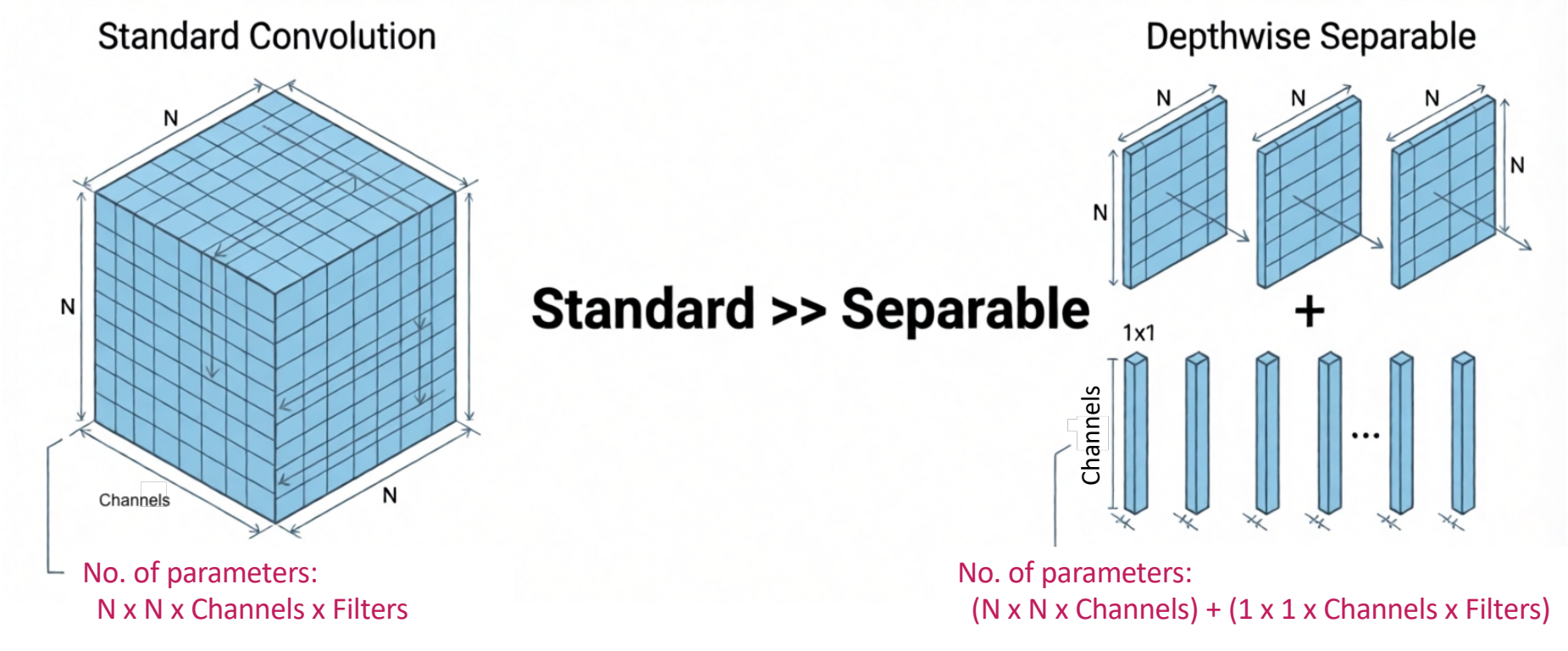
Both methods yield the identical mathematical result, but the Separable approach requires fewer floating-point operations.

Depthwise Separable Convolution

The Industry Standard for Efficiency (MobileNet, Xception)



Impact Analysis: Drastic Parameter Reduction

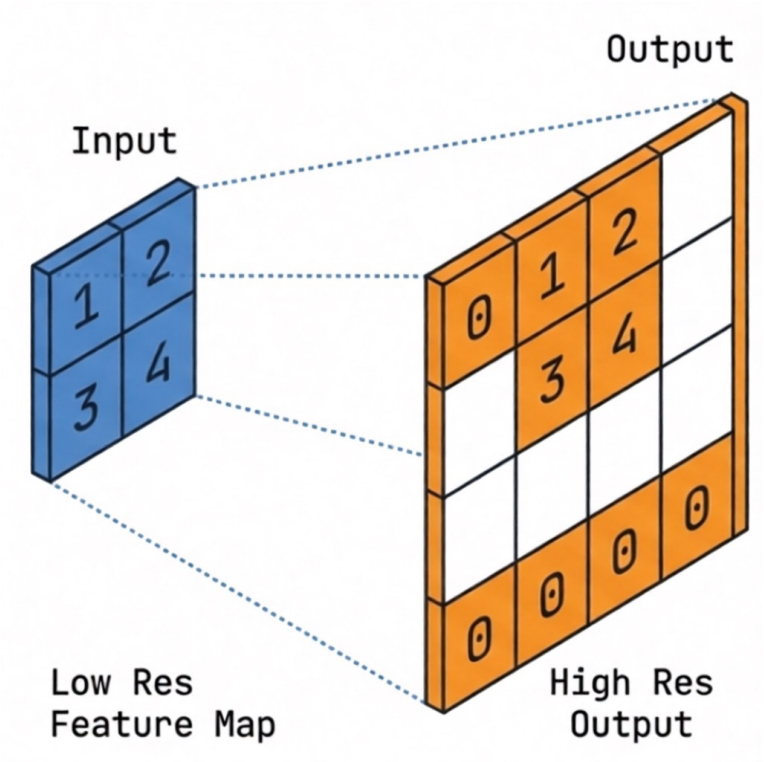


No. of parameters:
N x N x Channels x Filters

No. of parameters:
(N x N x Channels) + (1 x 1 x Channels x Filters)

The Trade-off: A marginal potential loss in accuracy for a massive reduction in model size and latency. Ideal for mobile and edge devices.

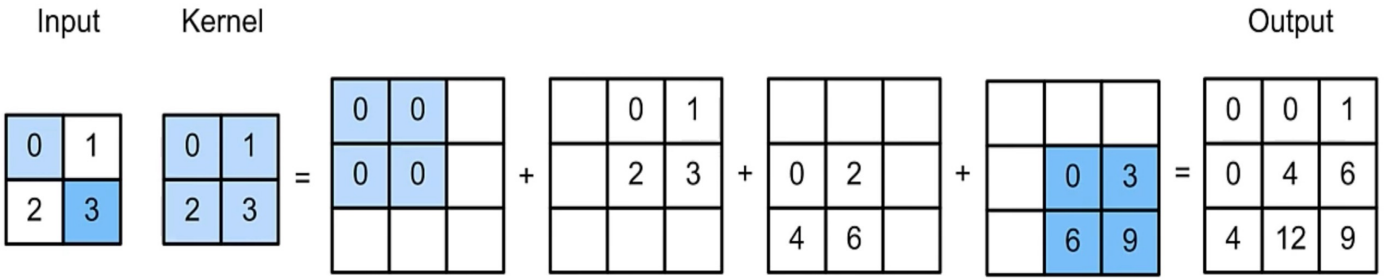
Transposed Convolution: Learnable Upsampling

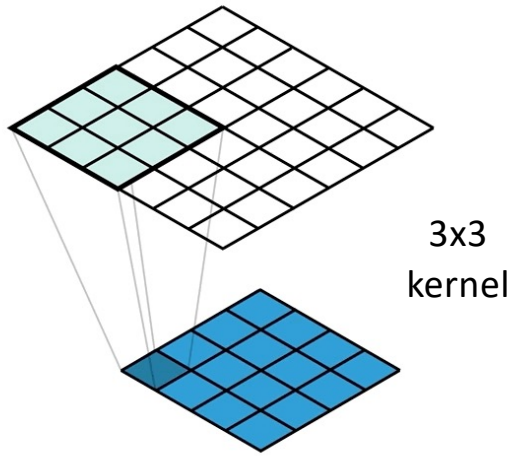


- **The Problem:** Standard convolution reduces spatial dimensions (pooling/striding). Traditional upsampling (Bilinear Interpolation) uses fixed, non-trainable rules.
- **The Solution:** Transposed Convolution (Deconvolution). Enables the network to learn the optimal way to upsample data to reconstruct images for segmentation or super-resolution.

How Transposed Convolution Works

It is not a reverse convolution. It is a standard convolution on a modified input



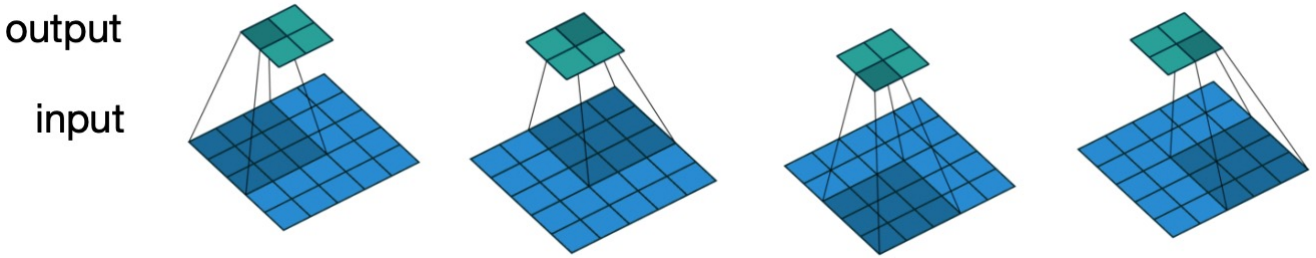


Also known as **Deconvolution**. Unlike fixed interpolation, this operation creates "Learnable Upsampling", allowing the network to discover the optimal way to expand spatial dimensions for generative tasks.

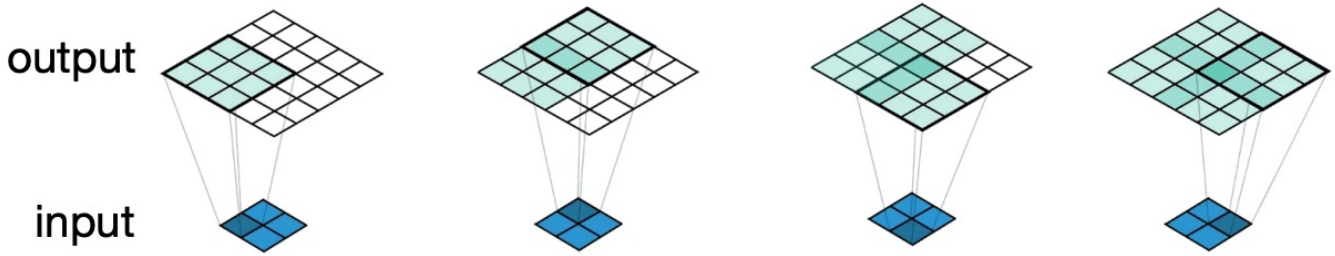
https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8

Regular vs Transposed Convolutions

- **Transpose convolution** allows to increase the size of the output feature map compared to the input feature map
- **Regular Convolution:** reduce the output size



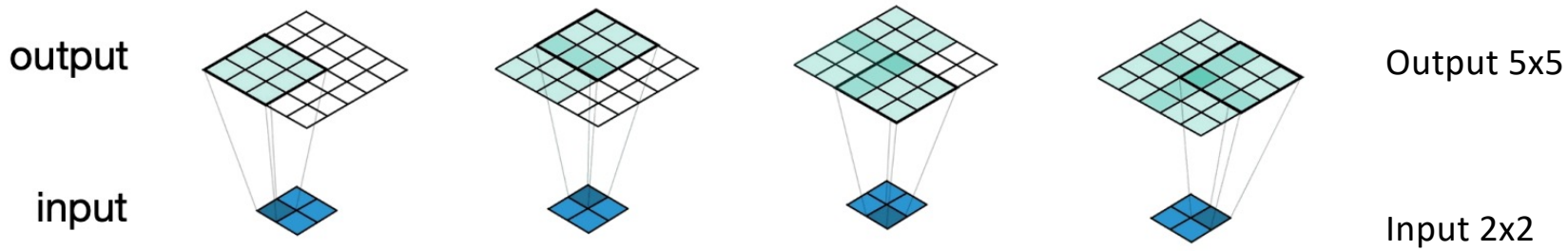
- **Transposed Convolution** (stride =2): increase the output size



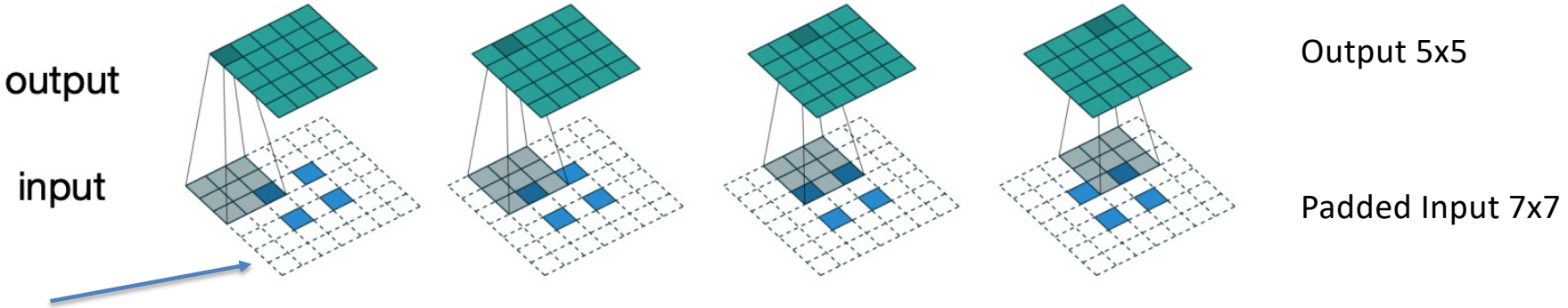
13

Transposed Convolution by Regular Convolution

- **Transposed Convolution** (3x3 kernel, **stride =2**)



- **Transposed Convolution that** emulated with regular convolution



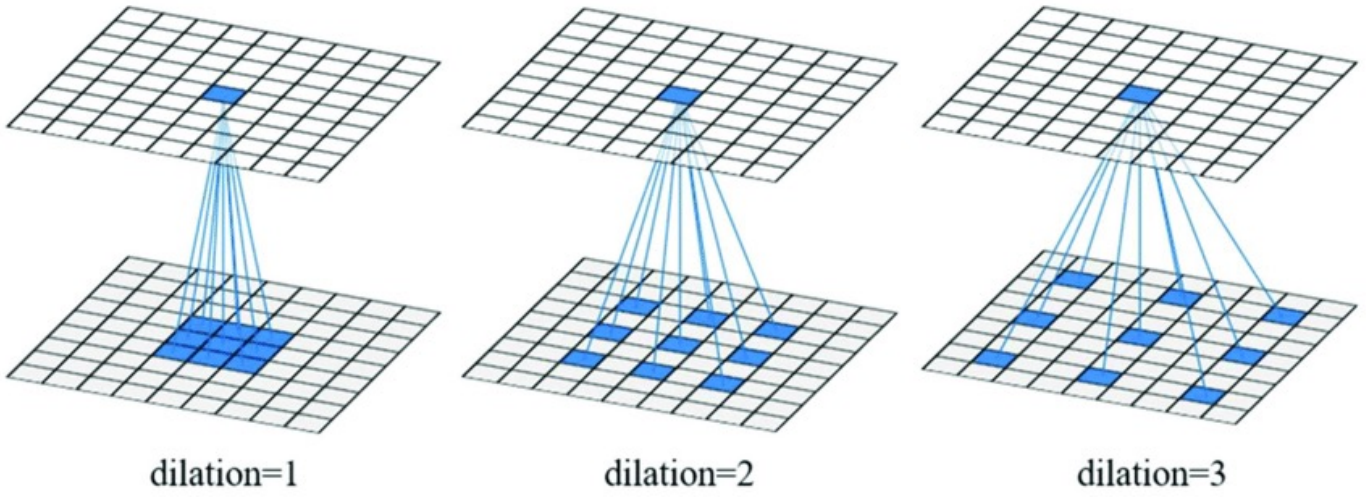
Putting paddings in the input feature maps to achieve transposed convolution using regular convolution,

14

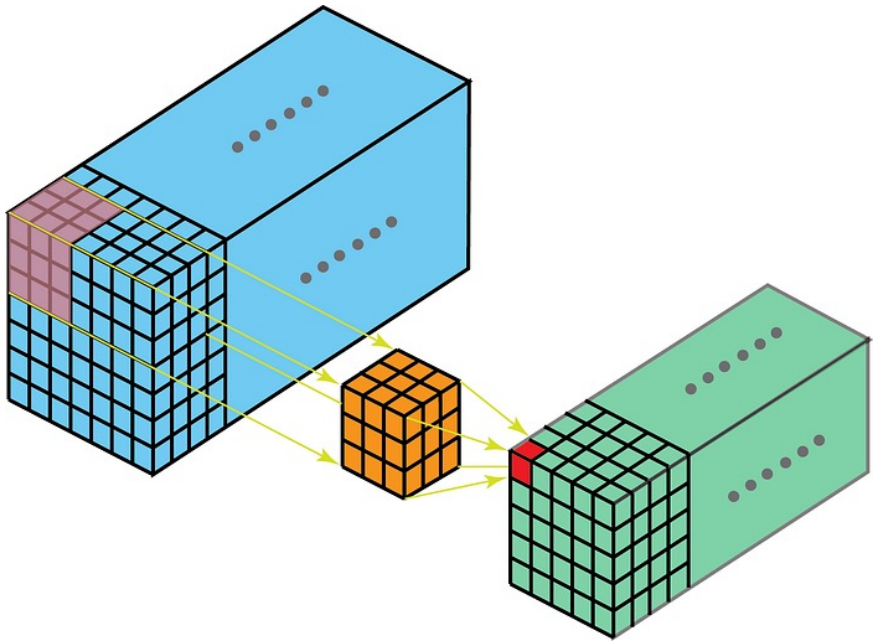
Dilated Convolution

Expanding the Receptive Field

- **The Innovation**: Introduces a "dilation rate" parameter that controls the spacing between values in a kernel.
- **The Effect**: Expands the receptive field (the area the network "sees") without increasing the number of parameters.

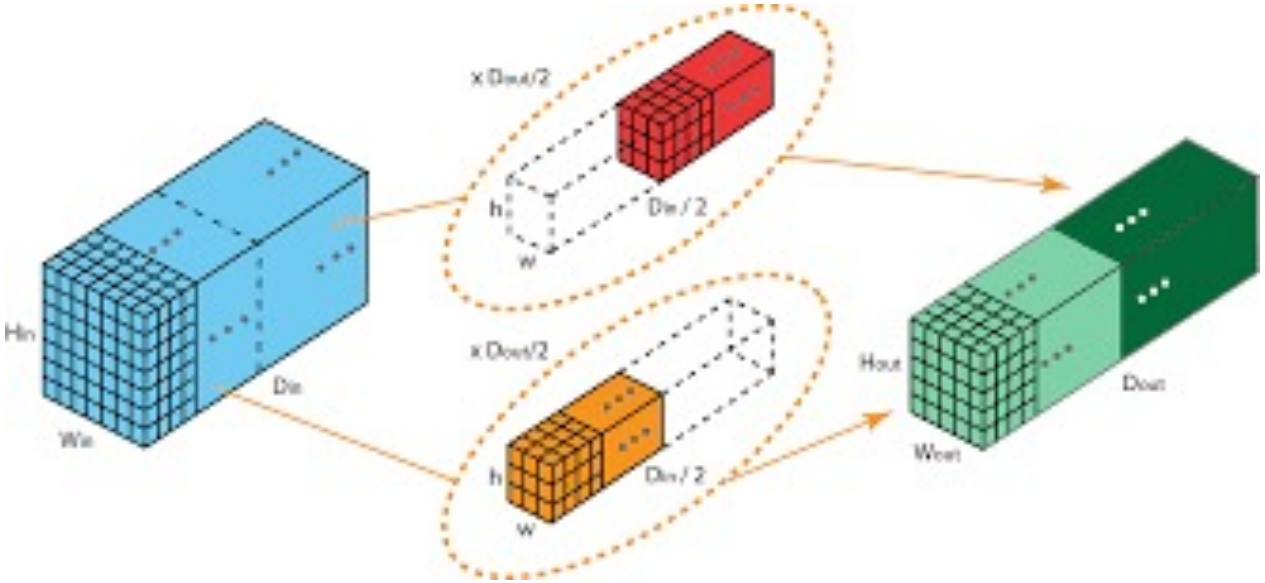


Processing Volumetric Data: 3D Convolution



Application: Video Analysis & Medical Imaging. The kernel slides through Height, Width, AND Depth, capturing temporal or volumetric relationships.

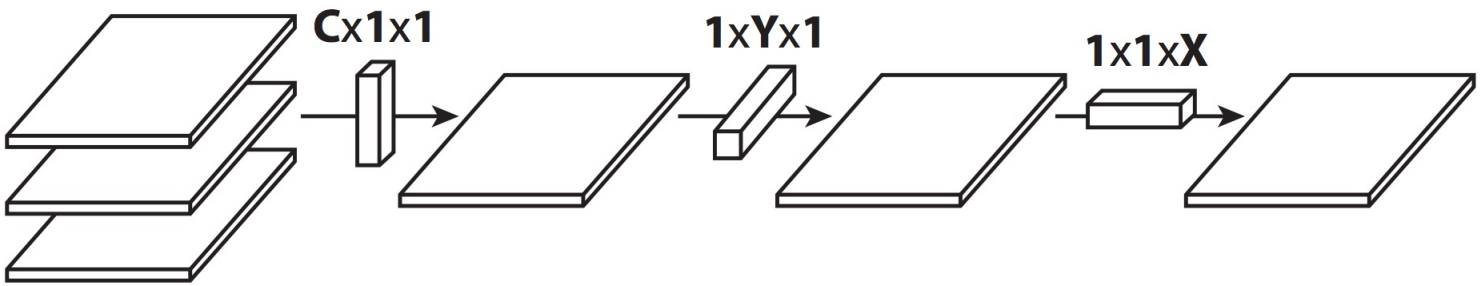
Parallel Pathways: Grouped Convolution



The input channels are split into distinct groups processed in parallel. Originally used for multi-GPU training (AlexNet), now used for efficient, structured architecture deign.

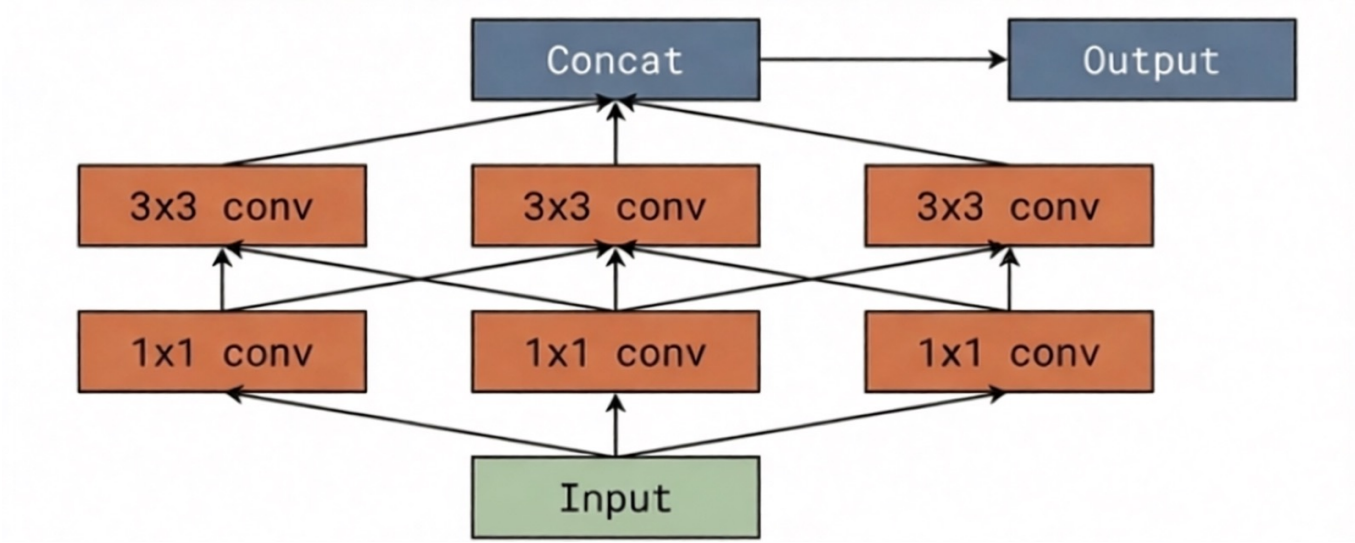
Flattened Convolution

- **The Architecture**: A pipeline that separates the task into three distinct stages:
  1. Lateral connections (Spatial)
  2. Vertical connections (Spatial)
  3. Cross-channel connections
- **The Outcome**: Achieves spatial and cross-channel convolution through a sequence of specialized 1D filters.



Cross-Channel Interaction

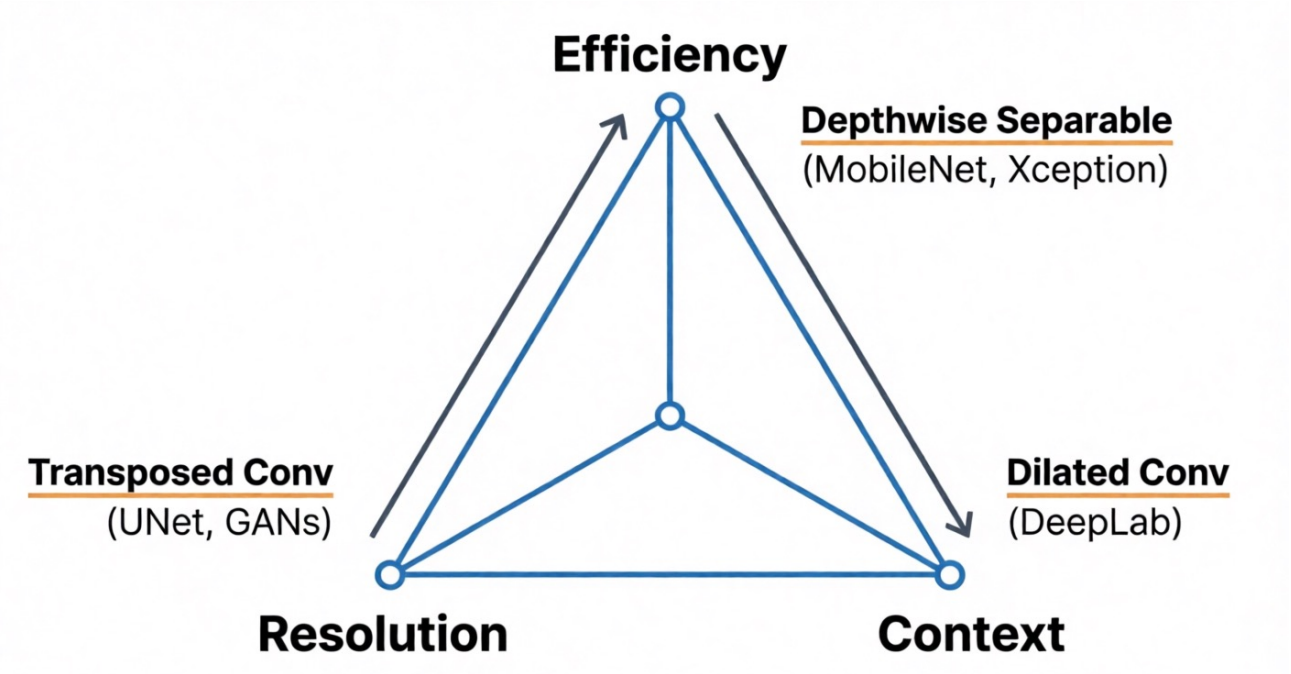
- Cross-Channel variants explicitly model correlations between feature maps.



The Convolutional Cheat Sheet

OPERATION	PRIMARY GOAL	KEY MECHANISM
Standard Conv	Feature Extraction	Sliding window over all channels (Baseline cost).
Pointwise (1x1)	Dimensionality Reduction	Projects channels to lower/higher depth. Keeps size.
Depthwise Separable	Efficiency (Mobile)	Separates spatial filtering from channel mixing.
Transposed Conv	Upsampling	Expands size via learned padding and convolution.
Dilated Conv	Context Aggregation	Uses kernel gaps to widen field of view.
3D Convolution	Volumetric/Video	Operates on Height, Width, and Depth/Time.

Choosing the Right Tool



Modern Neural Network architecture is the art of balancing these trade-offs. There is no single 'best convolution-only the right variant for the specific constraint.

Evolution of CNN Architectures

Evolution of CNN Architectures

- **LeNet-5** – First CNN for handwritten digit recognition (1989 and 1998)
- **AlexNet** – "ImageNet moment" (2012)
- **VGGNet** – Stacking 3x3 layers (2014)
- **Inceptions (GoogleNet)** – Parallel branches (2014)
- **ResNet** – Identity shortcuts (2015)
- **Wide ResNet** – Wide instead of depth (2016)
- **ResNeXt** – Grouped convolution (2016)
- **DenseNet** – Dense shortcuts (2016)
- **SENets** – Squeeze-and-excitation block (2017)
- **MobileNets** – Depthwise conv; inverted residuals (2017/18)
- **EfficientNet** – Model scaling (2019)
- **RegNet** – Design spaces (2020)
- **ConvNeXt** – (2022)

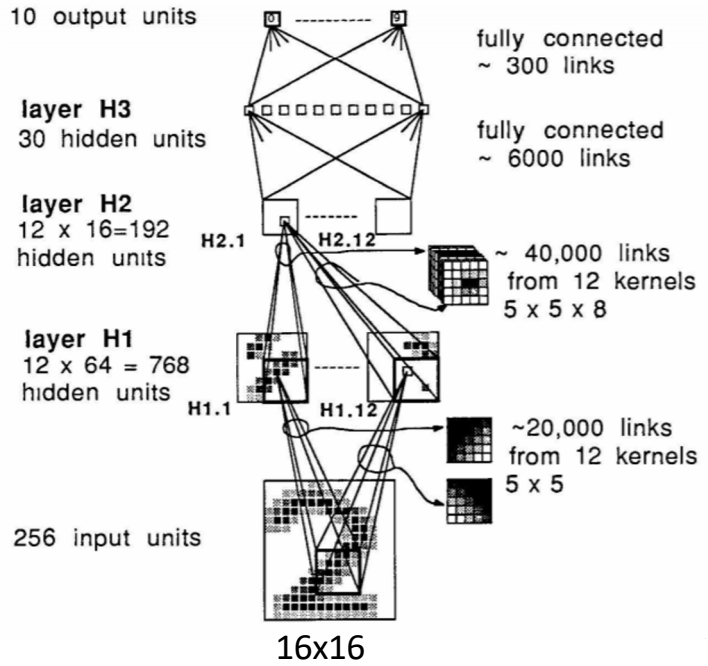
23

LeNet-5 (1989 -1998)

- Convolution operations are first introduced into Machine Learning by Yann LeCun at AT&T Laboratories (Y. LeCun et. al. 1989)

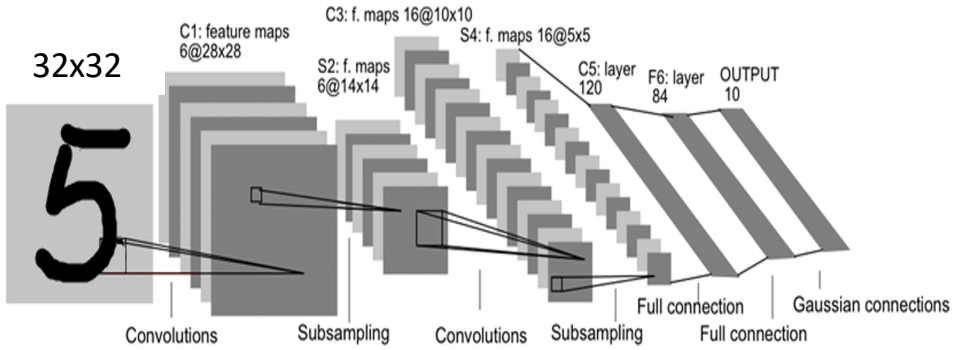
LeCun et al (1989)

Backpropagation first applied to CNN training



LeNet-5 Architecture (1998)

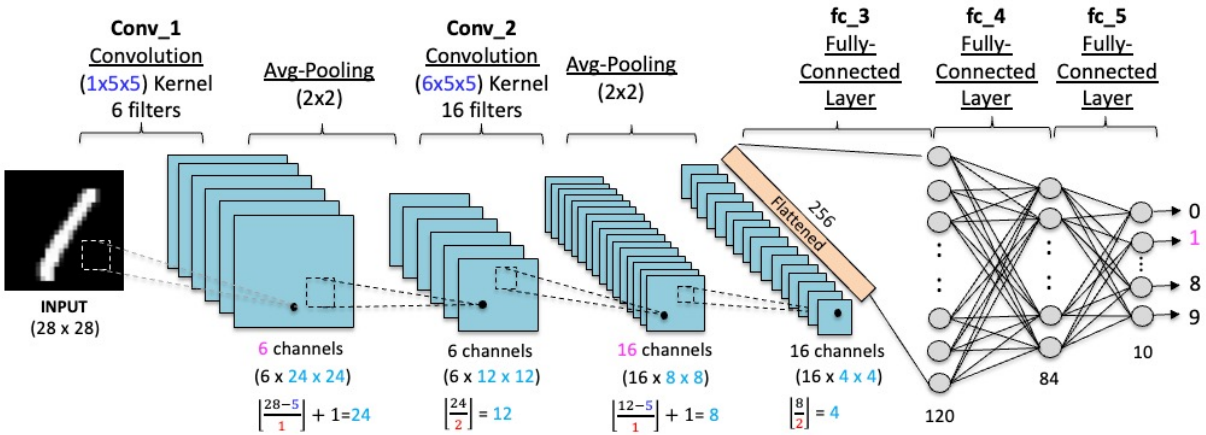
LeLeNet-5 achieved state-of-the-art handwritten digit recognition



24

LeNet-5 (1998)

- Conv filters were 5x5, applied at stride 1 and tanh activation function
- Subsampling (Average Pooling) layers were 2x2 applied at stride 2
- **Overall Architecture**: CONV-POOL-CONV-POOL-FC-FC-FC (Softmax)
  - **44.426K Parameters**
  - Paper: Gradient-based learning applied to document recognition




```
class LeNet5(nn.Module):
    def __init__(self):
        super().__init__()
        self.network = nn.Sequential(
            nn.Conv2d(1, 6, kernel_size=5, stride=1, padding=0),
            nn.Tanh(),
            nn.AvgPool2d(2, 2), # output: 6 x 14 x 14

            nn.Conv2d(6, 16, kernel_size=5, stride=1, padding=0),
            nn.Tanh(),
            nn.MaxPool2d(2, 2), # output: 16 x 5 x 5

            nn.Flatten(),
            nn.Linear(16*5*5, 120),
            nn.Tanh(),
            nn.Linear(120, 84),
            nn.Tanh(),
            nn.Linear(84, 10))

    def forward(self, xb):
        return self.network(xb)
```

```
Test Accuracy on the 10000 test images: 98.72 %
```

Colab Demo: https://colab.research.google.com/drive/1HVSreRywEAx0nn7Vwtr-mesJca67CoWQ?usp=sharing

25

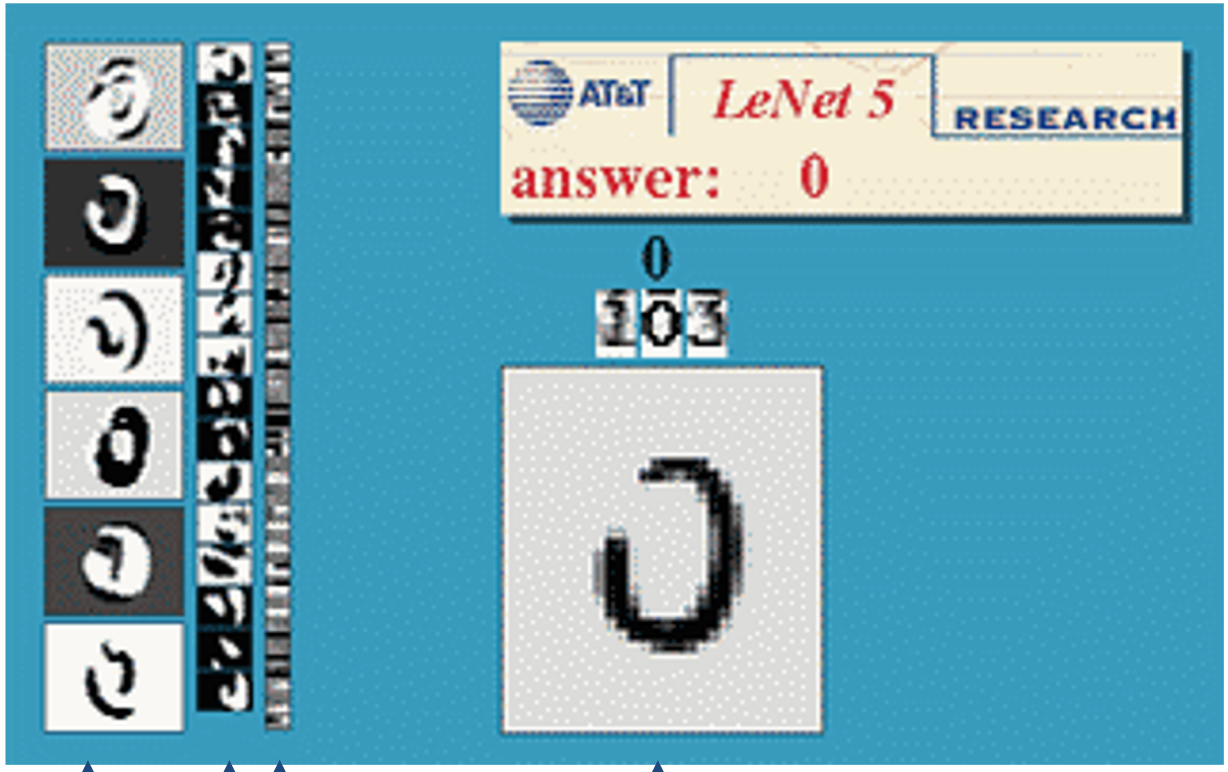
Key innovations of LeNet-5

- **Local Receptive Fields**: Each neuron in a feature map was connected to a small region (e.g., 5x5) of the input image, capturing local patterns.
- **Max Pooling**: LeNet-5 used max pooling with a 2x2 sliding window for downsampling, reducing the spatial dimensions of the input while retaining important information.
- **Average Pooling**: The final pooling layer used average pooling to further reduce the spatial dimensions before the fully connected layers.

LeNet-5 (1998)

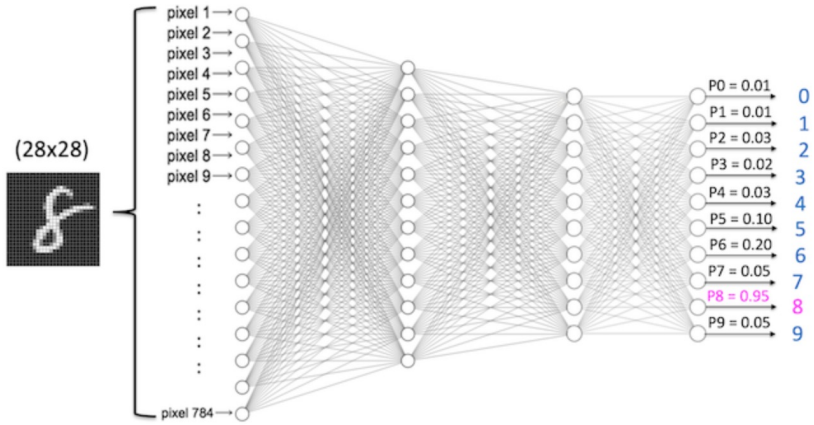
- Handwritten digit Recognition
- MNIST Dataset

https://yann.lecun.com/exdb/lenet/a35.html

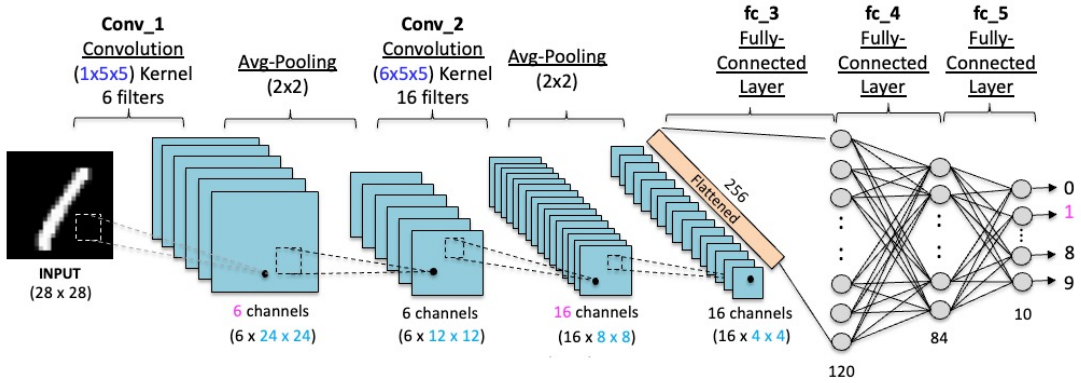


MLP vs CNN for NMIST Image Dataset

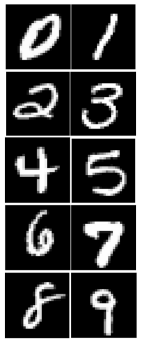
MLP



CNN: LeNet-5



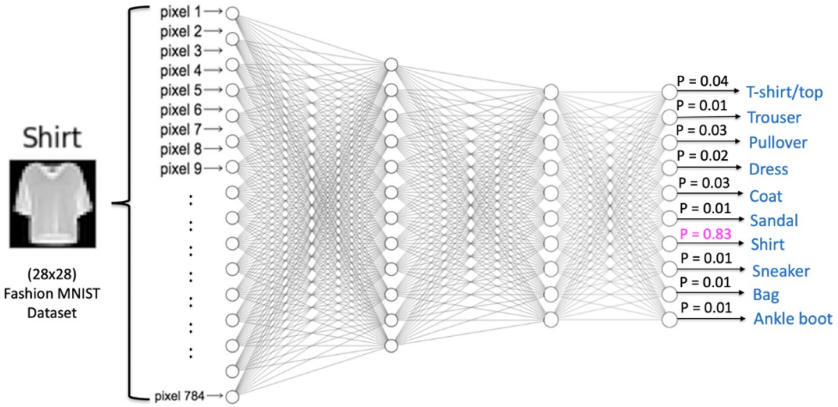
- **MLP**: MNIST Dataset
  - 70,000 28x28 Grayscale Images
  - 3-Layer MLP (784-128-64-10)
  - **Performance**: 98.3% Accuracy
  - **No. of Parameters**: 109.386K



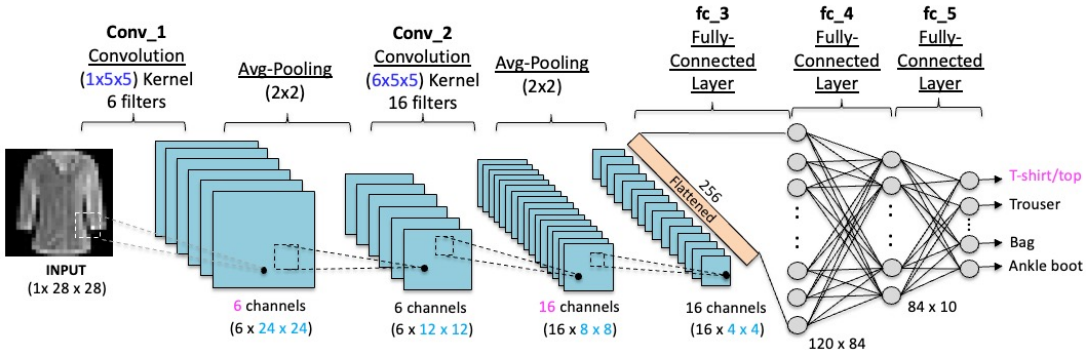
- **LeNet-5**: Fashion MNIST Dataset
  - 70,000 28x28 Grayscale Images
  - 5-Layer CNN: 5x5-5x5-120-84-10
  - **Performance**: 98.72% Accuracy
  - **No. of Parameters**: **61.706K**

MLP vs CNN for Fashion NMIST Image Dataset

MLP



CNN: LeNet-5

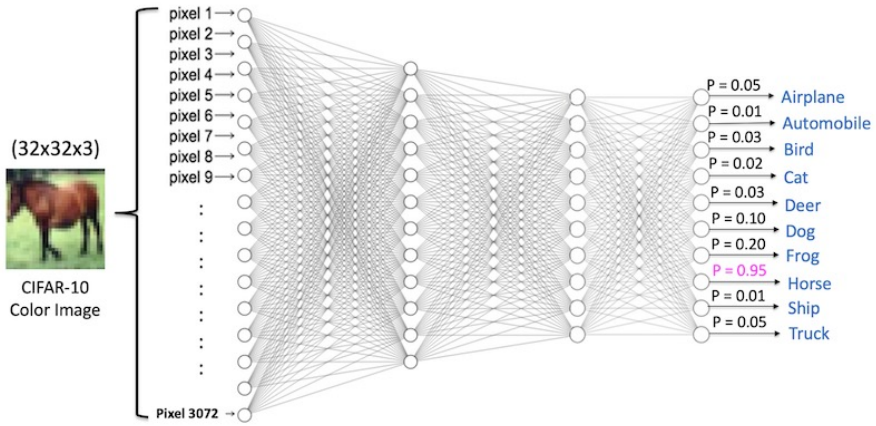


- **MLP**: Fashion MNIST Dataset
  - 70,000 28x28 Grayscale Images
  - 3-Layer MLP (784-128-64-10)
  - **Performance**: 84.18% Accuracy
  - **No. of Parameters**: 109.386K

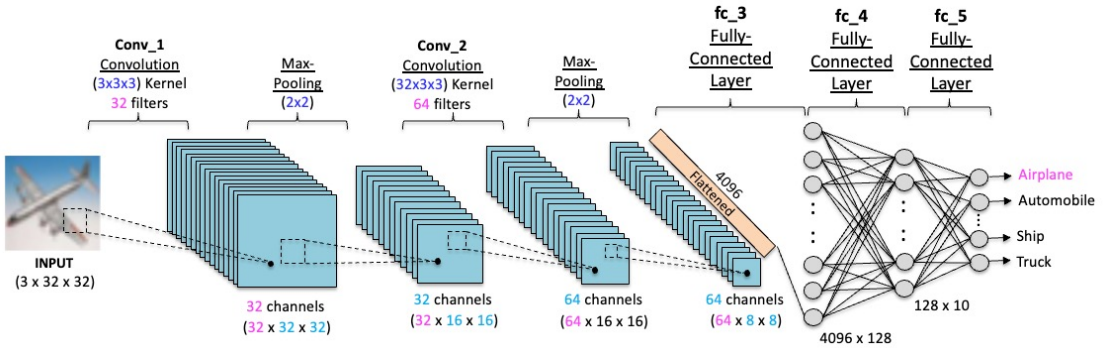
- **LeNet-5**: Fashion MNIST Dataset
  - 70,000 28x28 Grayscale Images
  - 5-Layer CNN: 5x5-5x5-120-84-10
  - **Performance**: 87.91% Accuracy
  - **No. of Parameters**: **44.426K**

MLP vs CNN for CIFAR-10 Color Image Dataset

MLP



CNN: LeNet-5



- **MLP**: CIFAR-10 Dataset
  - 60,000 32x32 RGB Color Images
  - 5-Layer MLP (3072-1536-768-384-128-10)
  - **Performance**: 54.9% Accuracy
  - **No. of Parameters**: 6,246.410K (about 6M)

https://colab.research.google.com/drive/1vbFi4_6gZ_-bPhBFEdkoSOc3syjshoXP

- **LeNet-5**: Fashion MNIST Dataset
  - 70,000 28x28 Grayscale Images
  - 5-Layer CNN: 5x5-5x5-120-84-10
  - **Performance**: 64.78% Accuracy
  - **No. of Parameters**: **62.006K**

https://colab.research.google.com/drive/1opQIrK6c2GDJP39PUJMMIu3cuznUggsU?usp=sharing#scrollTo=LVd86t5vVUk2

The Golden Era of CNNs (2010s)

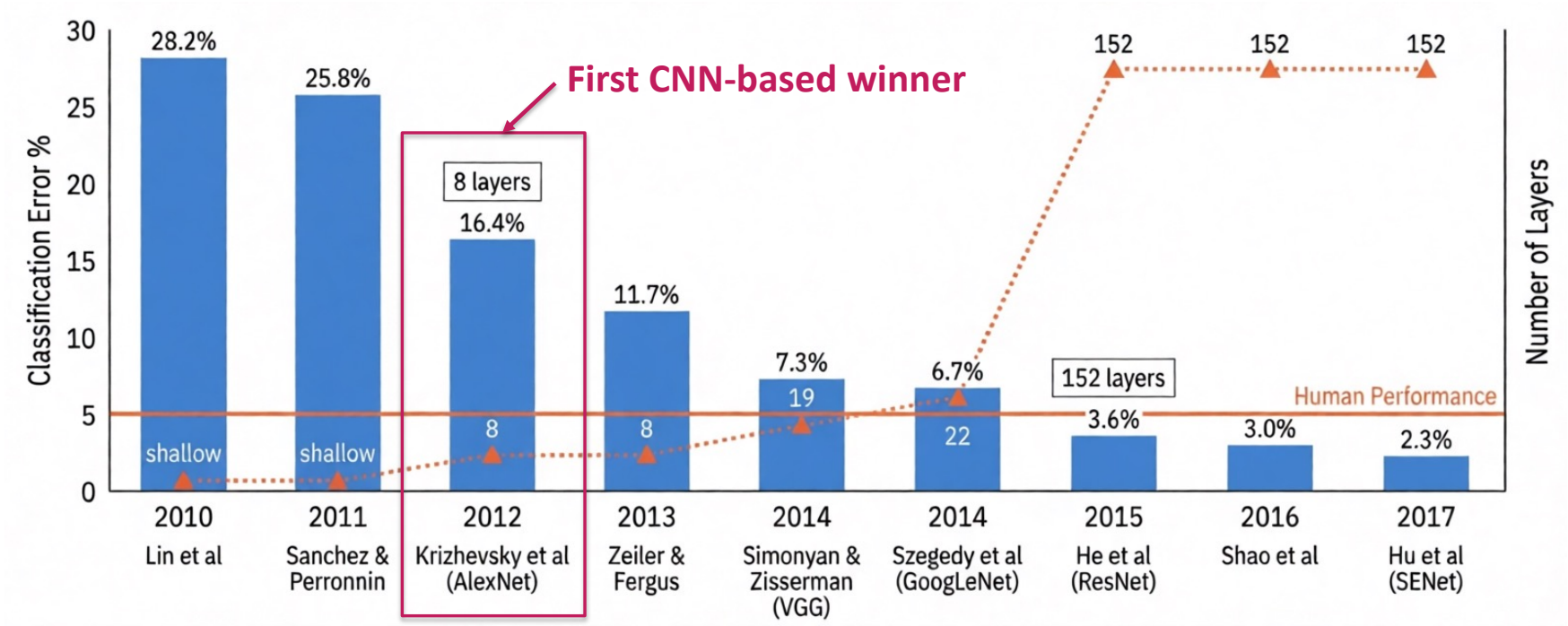
ImageNet Dataset and ILSVRC Challenge

- In the early 2000s, deep learning techniques like CNNs were less popular due to their high computational demands and **risk of overfitting to small datasets**.
- **ImageNet Dataset (2009)**
  - The introduction of the large-scale ImageNet dataset in 2009 changed this, providing over **14 million annotated images** across **1,000 object classes**.
  - Its size and variety enabled researchers to train deeper models, leading to significant advancements in computer vision.

- **Large Scale Visual Recognition Challenge (ILSVRC, 2010-2017)**
  - The ILSVRC challenge, using a subset of the ImageNet dataset, further accelerated innovation from 2010 to 2017, with notable models like AlexNet, VGG, and ResNet achieving state-of-the-art performance in image classification and driving progress in deep learning and computer vision.

The Explosion of Depth

Winners of ILSVR Challenge (2010 - 2017)



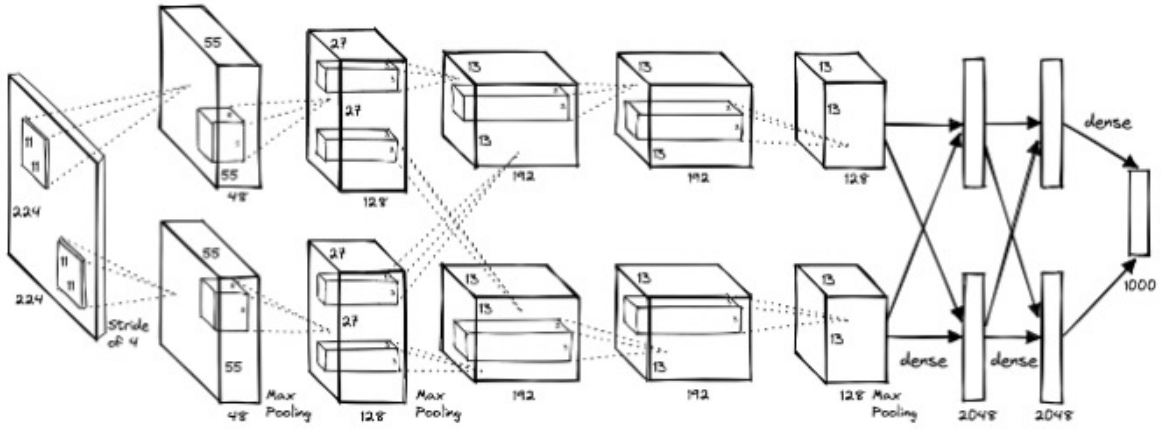
Following AlexNet, models became exponentially deeper to capture more complex features.
By 2015, ResNet (152 layers) surpassed human-level performance.

AlexNet (2012): The Deep Learning Big Bang

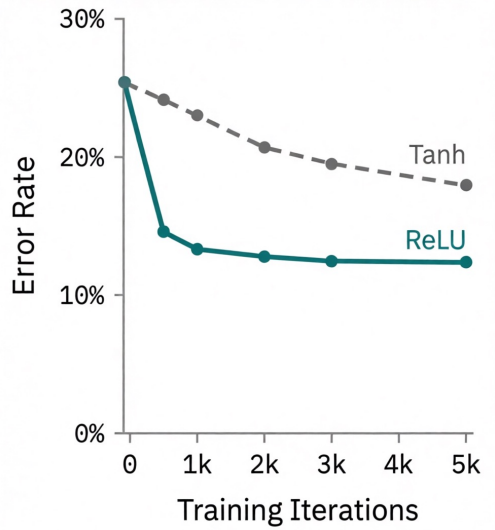
AlexNet won **ILSVRC 2012** with an 8-layer (5 Conv, 3 FC) —much deeper than LeNet. Key innovations:

- **ReLU** for faster training (6× speedup),
- **Dropout** (p=0.5) to reduce overfitting,
- **Data augmentation** via flips and color shifts,
- **Two-GPU training** splitting 60M parameters across GTX 580s (1 week).

- It achieved a 15.3% top-5 error—far ahead of the second-place 26.2%—igniting the deep learning era in vision.

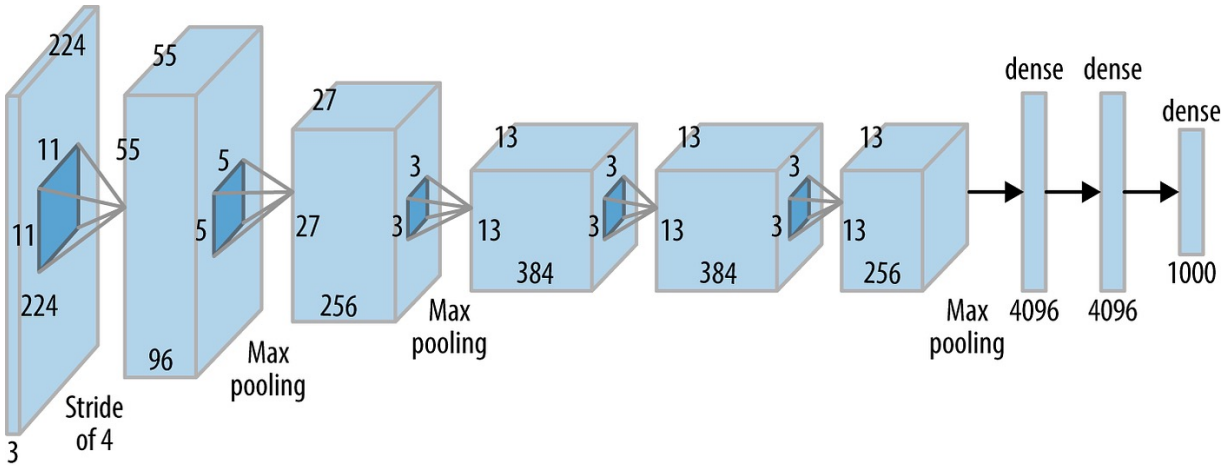


Implementation of Alex used 2 GPUs



AlexNet Architecture

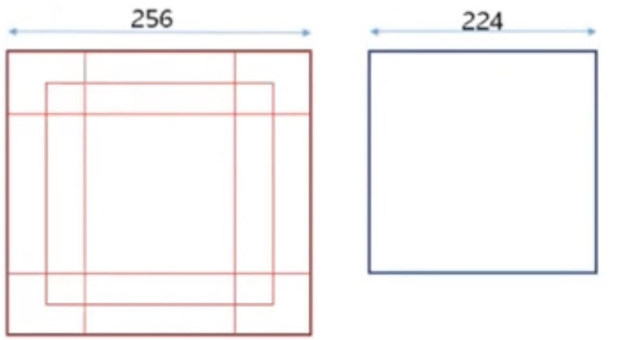
- **8 layers** (5 conv layers and 3 fully connected layers)
  - CONV1-CONV2-POOL-CONV3-POOL-CONV4-POOL-CONV5-POOL-FC-FC-FC
  - The ReLU non-linearity is applied to the output of every layers
  - Response normalization layer follow the first and second conv layers
  - Overlapped 3x3 Max Pooling and Dropout in FC layers
  - 60M Parameters
- Paper: ImageNet Classification with Deep Convolutional Neural Networks



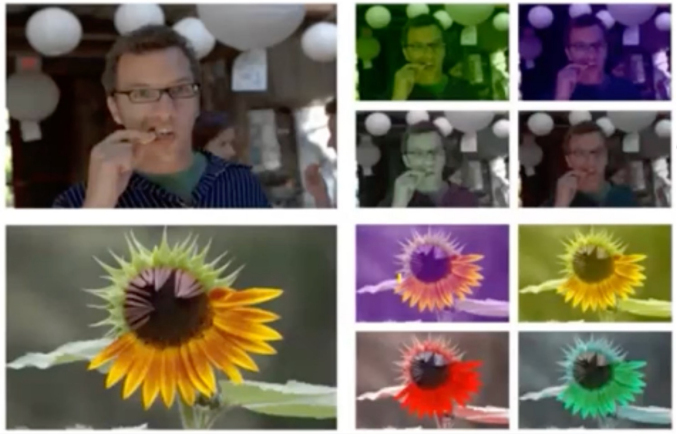
AlexNet: Reducing Overfitting Techniques

Data Augmentation

- Image Translation
- Horizontal Reflection



- Color Jittering
- Dropout in FC Layers

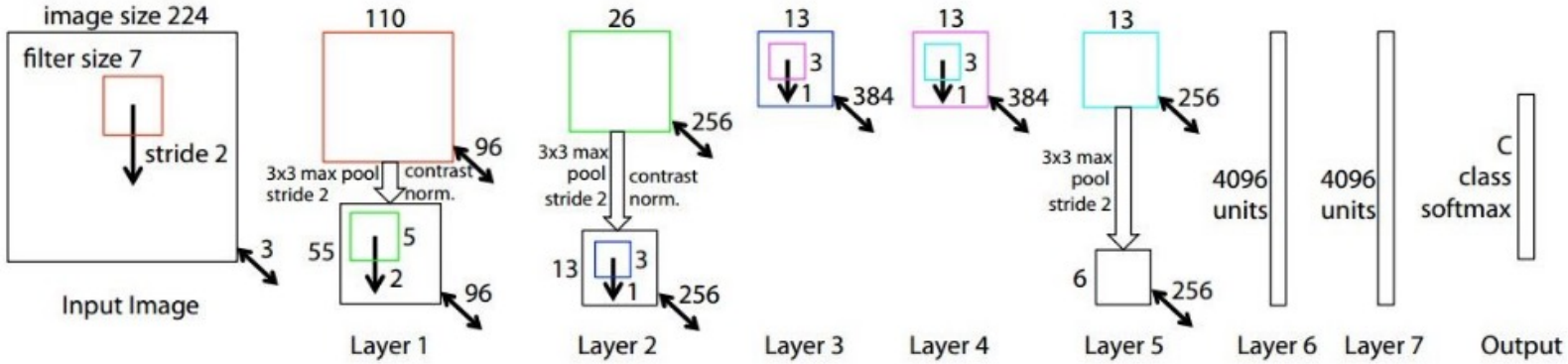


ZFNet (2013)

Showing reduced filter size and stride in initial layers.

Winner of ILSVRC-2013

- Similar to AlexNet architecture with **8 layers** but:
  - CONV1: change from (11x11 stride 4) to (7x7 stride 2) => **smaller filter's kernel**
  - CONV3, 4, 5: instead of 384, 384, 256 filters, use 512, 1024, 512 => **more filters (Wide AlexNet)**
  - Visualizing intermediate feature maps helped researchers understand what CNNs learn.
- **ImageNet top-5 error: 16.4% => 11.7%**

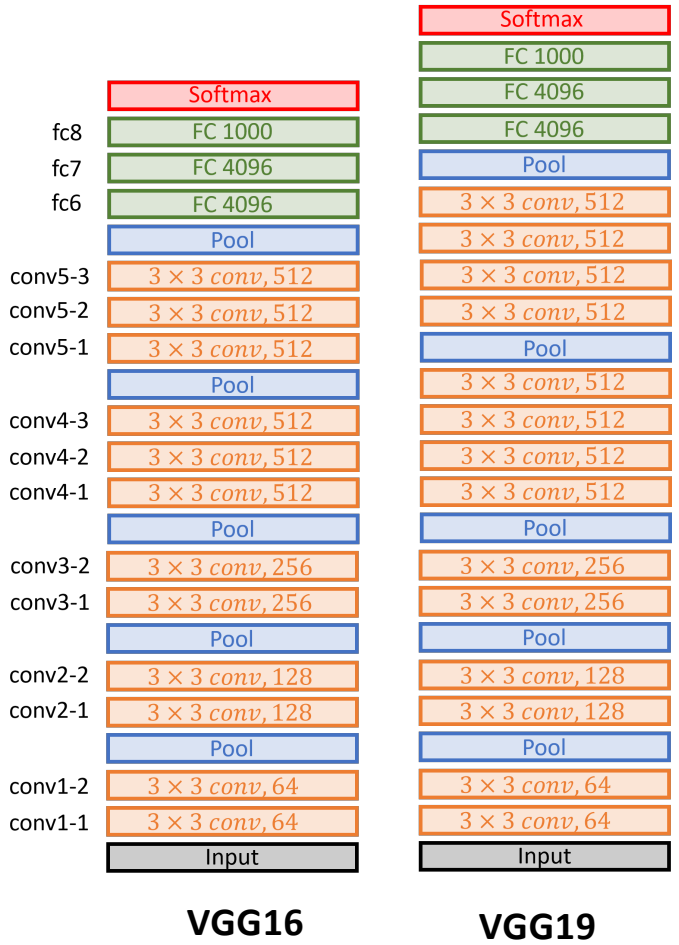


VGGNet (2014): The Philosophy of Depth

Runner-up in ILSVRC-2014

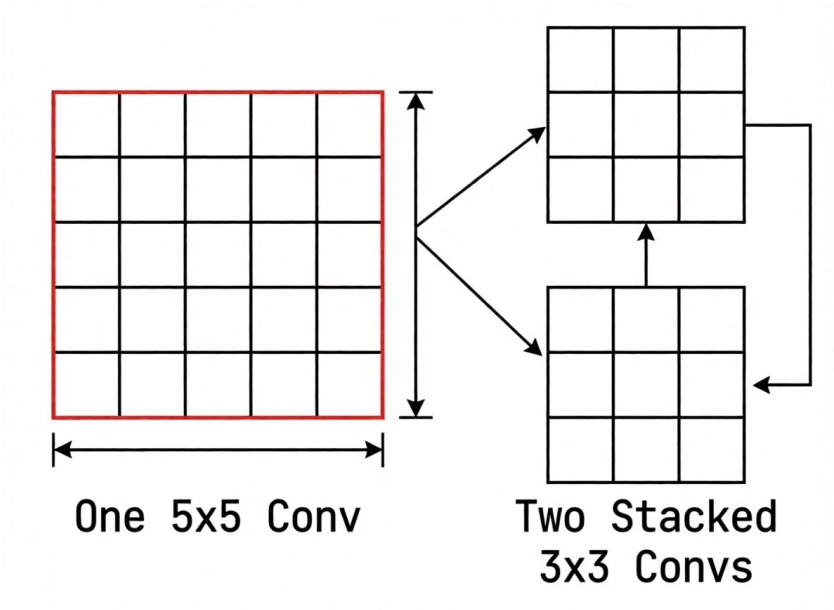
- VGG replaced the complex, variable filter sizes of AlexNet/ZFNet with a radically simple philosophy:
  - All Convs: 3x3, Stride 1
  - All Pools: 2x2
  - The Cost: 138 Million Parameters
  - The Gain: 7.3% Error Rate

The VGG Block:
3x3 Conv + 2x2 Max Pool



The Power of Small Filters

Why stacking 3x3 layers i s better than a single large layer



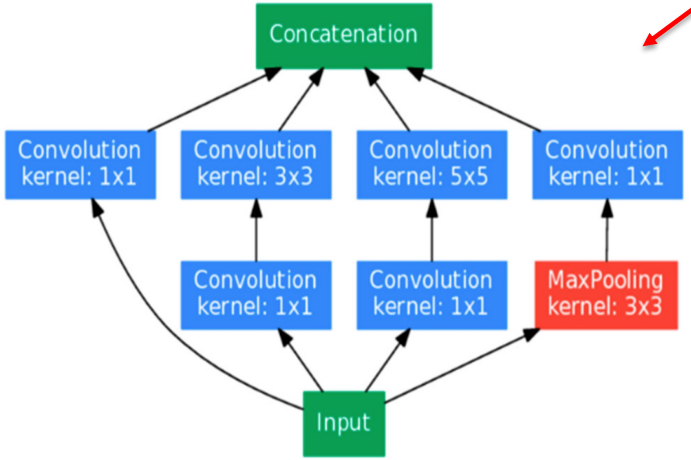
- Using stacked 3×3 convolutions reduces parameters while preserving receptive field:
  - Stacked two 3×3 ≈ one 5×5,
- For input and output with *C* channels, three 3×3 layers use $27C^2$ params vs. $49C^2$ for one 7×7—saving >40% parameters and speeding up computation.
- More non-linearity (two ReLU activations instead of one).

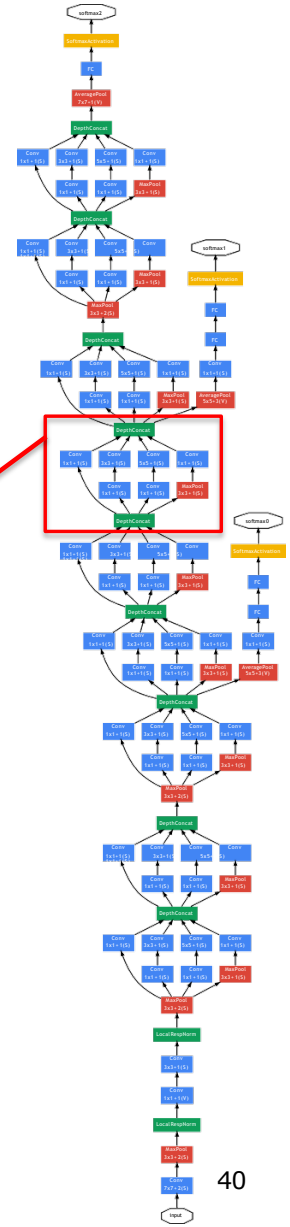
GoogLeNet/Inception V1 (2014)

Winner of ILSVRC-2014

- **6.7% top-5 error**

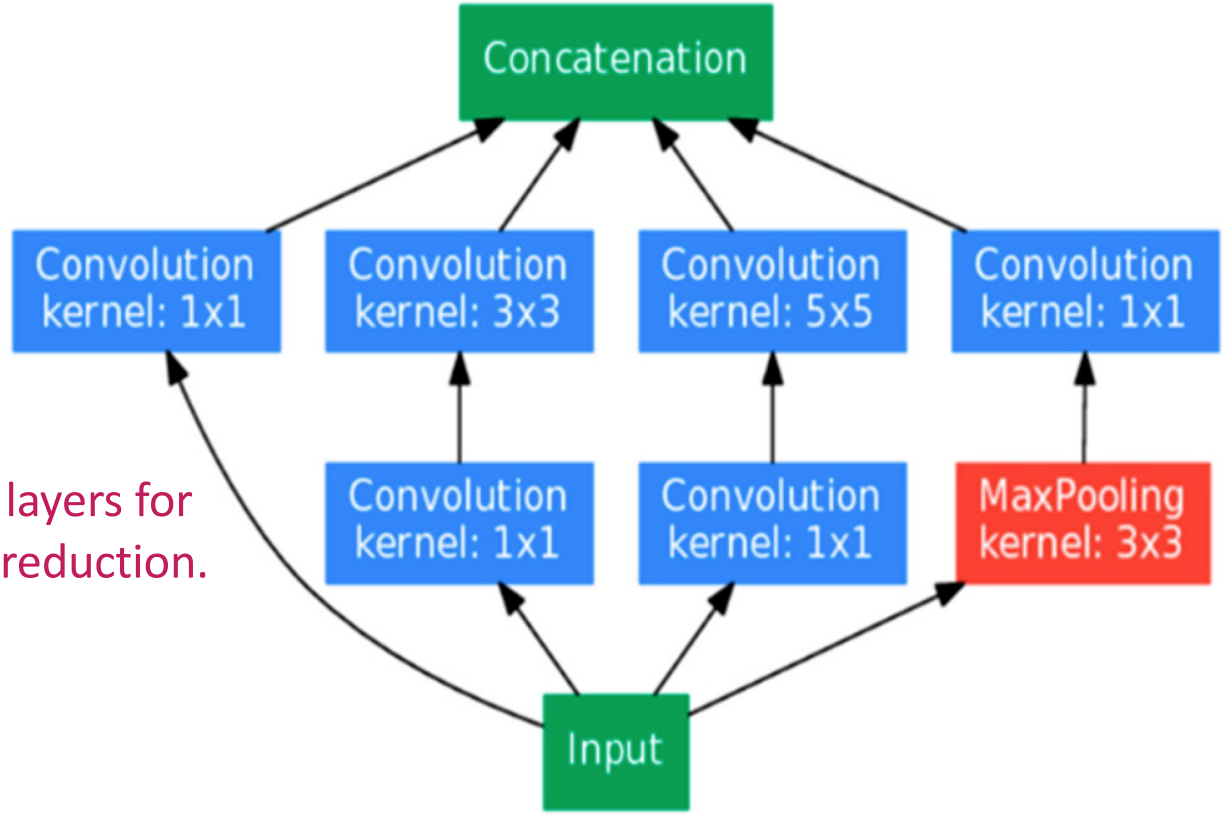
- **The Shock:** Only 5 Million Parameters.
- **Comparison:** 12x smaller than AlexNet. 27x smaller than VGG.
- **Design Philosophy:** Going Deeper (22 layers) and Wider
  - Inception Module





The Inception Module

A Network Within a Network

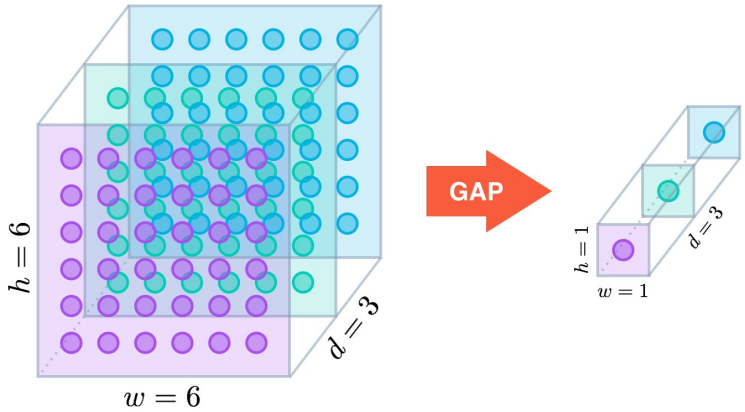


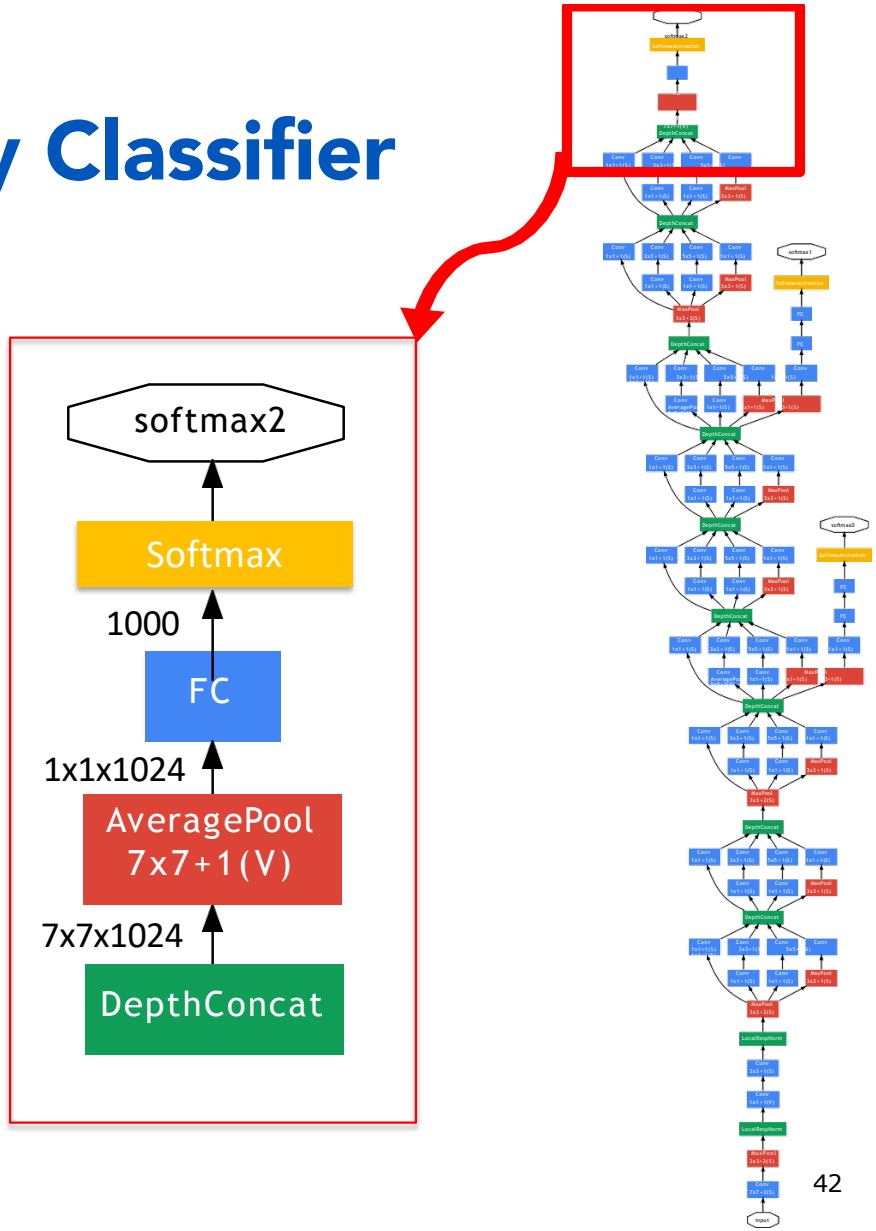
Captures features at multiple scales simultaneously.

Bottleneck layers for dimension reduction.

GoogLeNet: Output Auxiliary Classifier

- **Auxiliary Classifier**
  - In this classifier, Global Average Pool (GAP) is used instead of Flatten, which can reduce the number of parameters



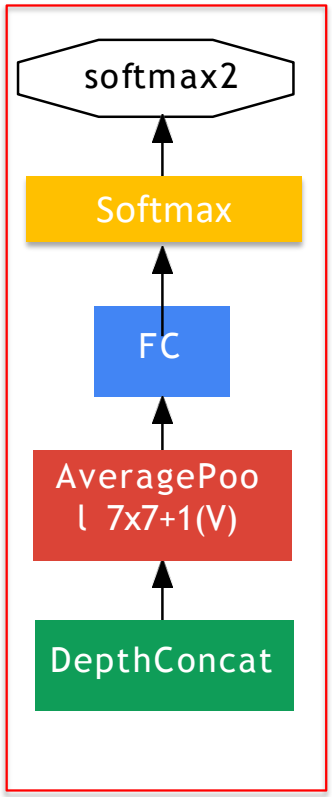


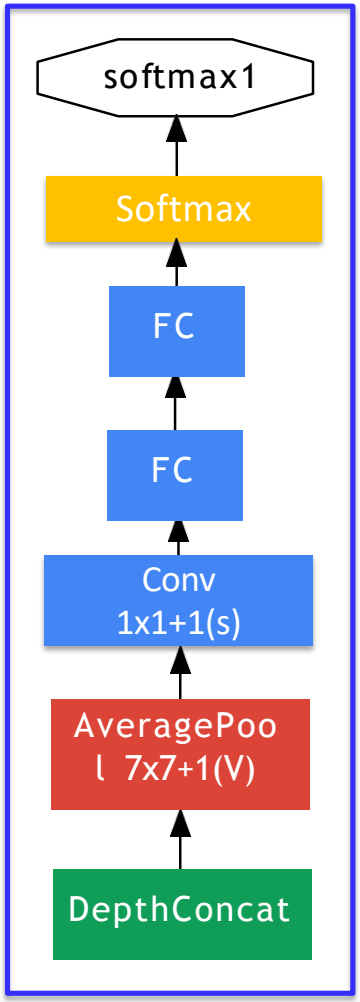
GoogLeNet: Auxiliary Classifiers

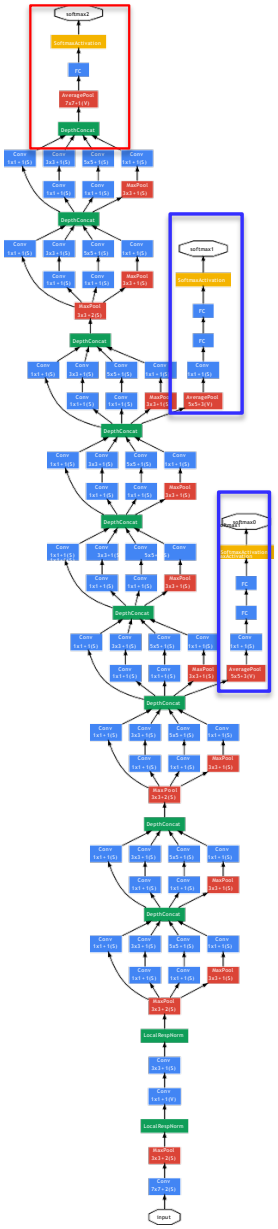
There are 3 Auxiliary Classifiers

- **During Training:**
  - **Inject additional gradient at lower layers.**
  - Make the optimization easier.
- **During Test:**
  - Disable the auxiliary outputs.

Auxiliary classification outputs to inject additional gradient at lower layers (GlobalAvgPool-1x1Conv-FC-FC-Softmaxe)

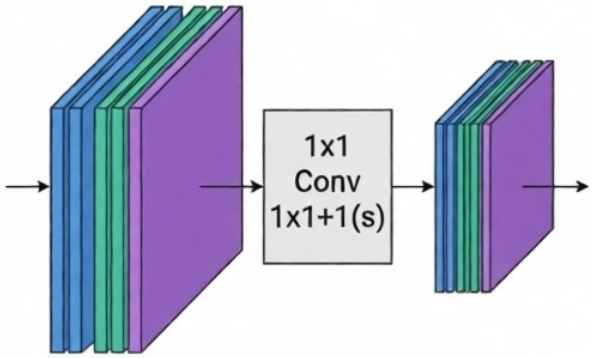






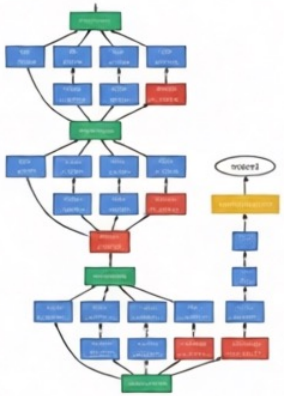
Optimization Innovations in GoogLeNet

The Bottleneck



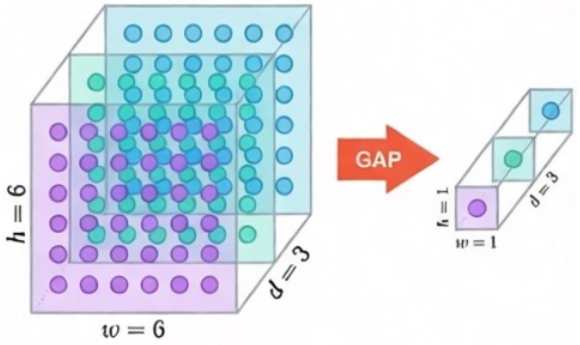
1x1 Convs reduce depth before expensive operations.

Auxiliary Classifiers



Injects gradients early to to stop vanishing gradient problem.

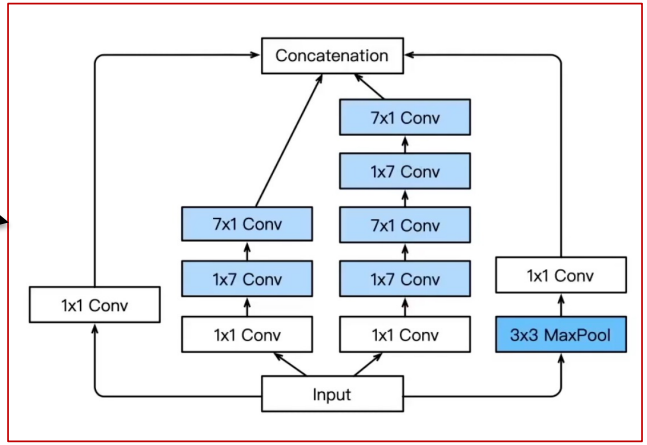
Global Avg Pooling



Replaces massive FC layers. Saves millions of parameters.

Variants of Inception V1 (2015-2016)

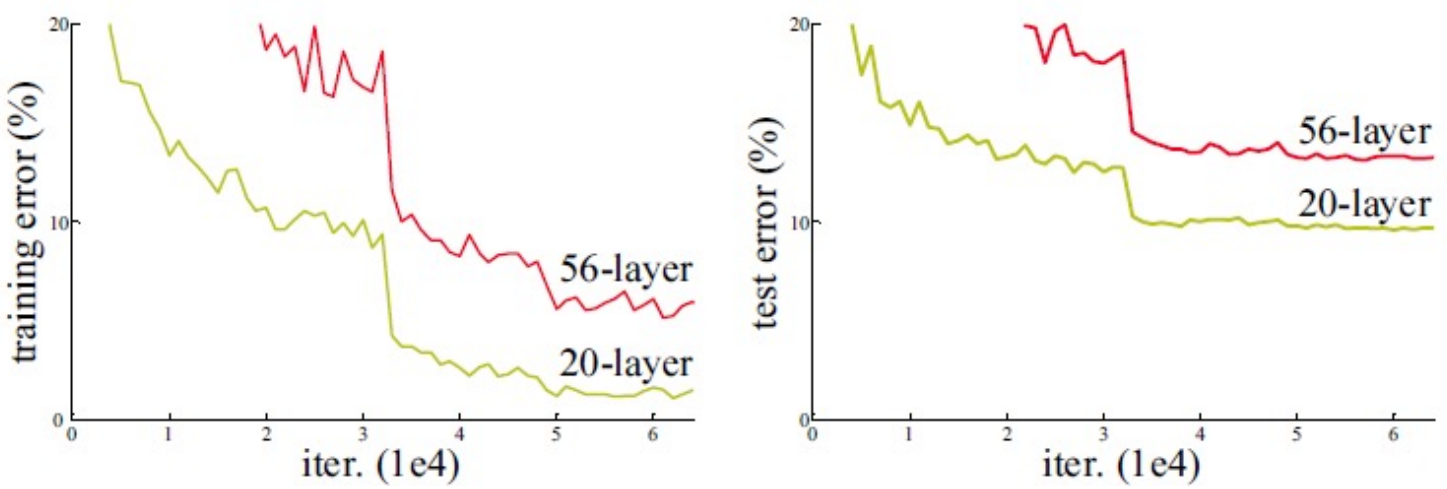
- **Inception-BN (V2)** (2015): Use **Batch Normalization (BatchNorm)**
- **Inception V3** (2016): Uses modified Inception modules
  - Replace 5x5 by multiple 3x3 conv layers
  - Replace 5x5 by 1x7 and 7x1 conv layers
  - Replace 3x3 by 1x3 and 3x1 conv layers
  - Deeper
- **Inception V4** (2016): Use residual connections
  - This is the winner of ImageNet LSVRC-2016



The Optimization Wall

The Degradation Paradox

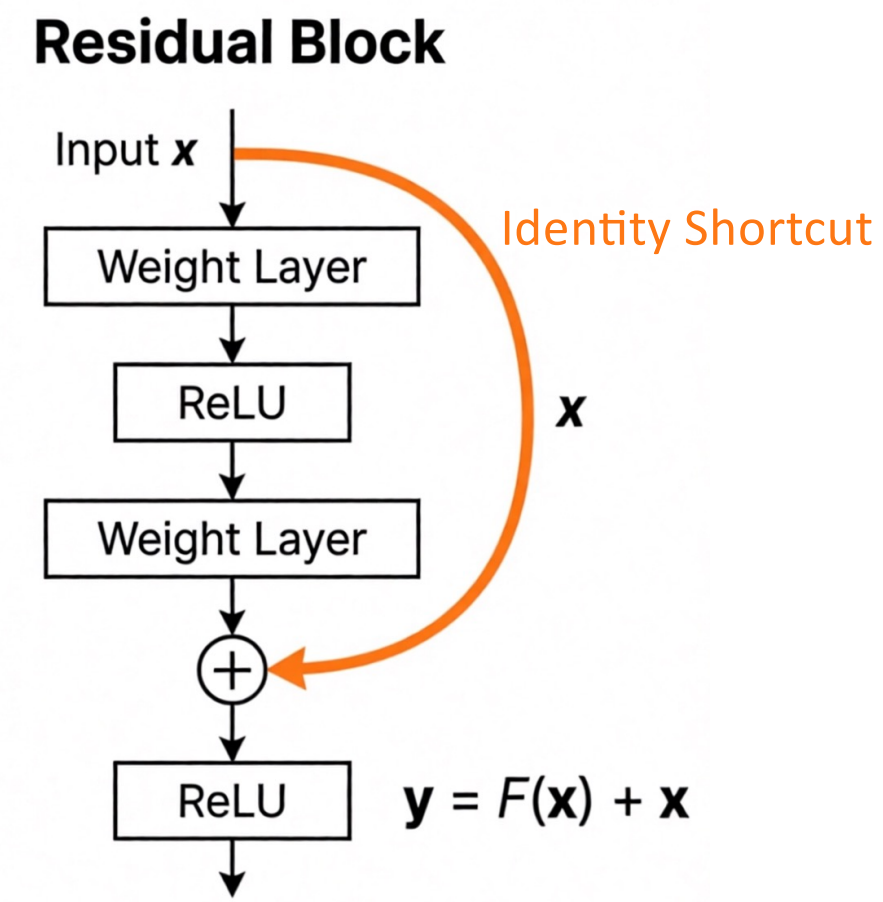
- What happens when we continue stacking deeper layers on a “plain” convolutional neural network for CIFAR-10 Dataset?



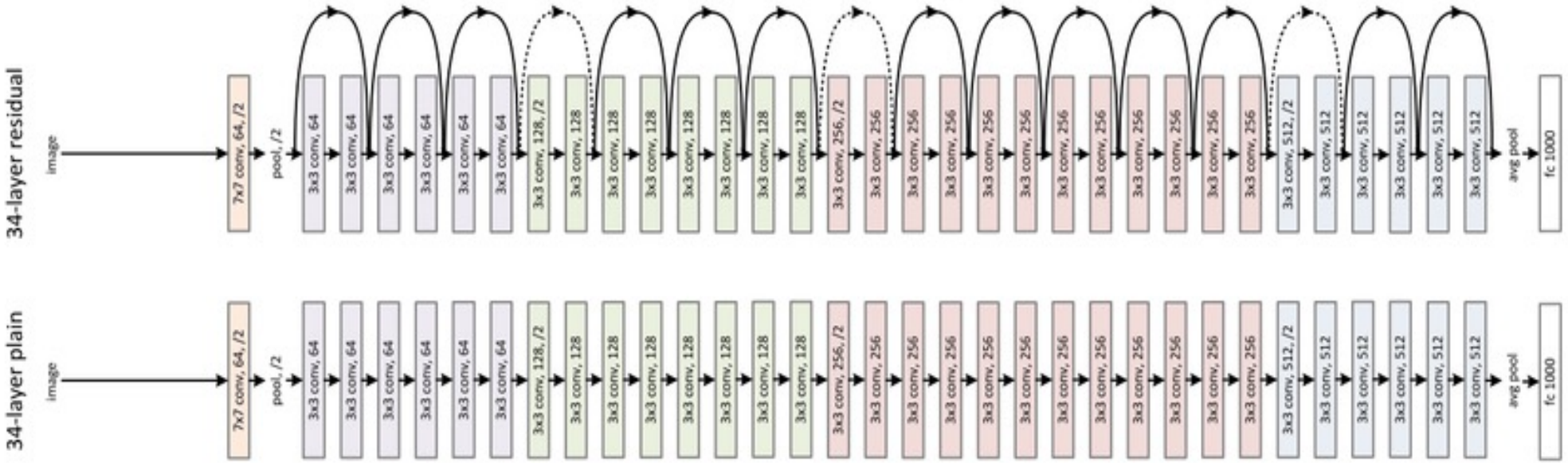
Not overfitting. The deep network struggles to optimize even the training data.
The signal vanishes in the depth.

Enter ResNet (2015): The Power of the Shortcut

- **Innovation**:
  - Learning the residual (difference) is easier than learning the mapping.
  - If the layers learn nothing, the identity connection (x) ensures the signal passes through unchanged.
- **Result:**
  - 152 Layers.
  - 3.57% Error (**ILSVRC 2015 Winner**).



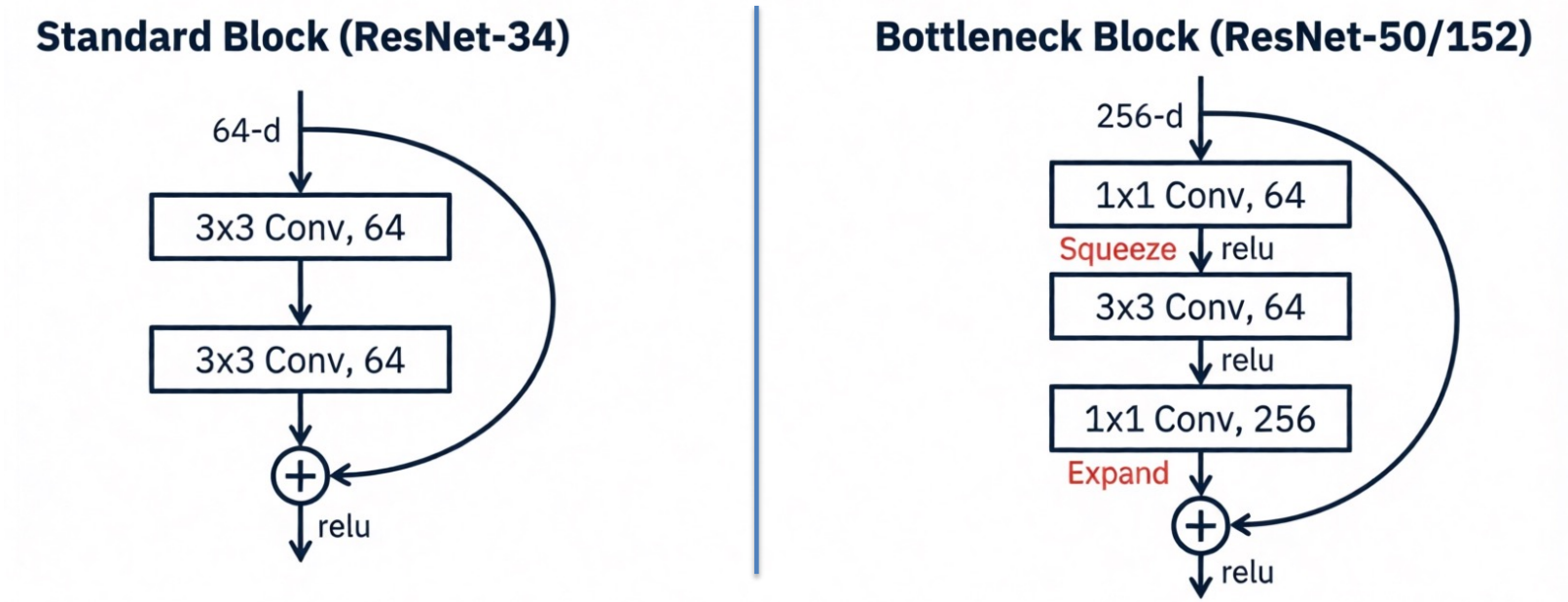
34-Layer Plain vs 34-Layer Residual



https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8

Engineering for Depth: The Bottleneck Design

- How to stack 152 layers without exploding computational cost.



Efficiency Metric: ResNet-152 (11.3B FLOPs) is computationally cheaper than VGG-19 (19.6B FLOPs) despite being ~8x deeper.

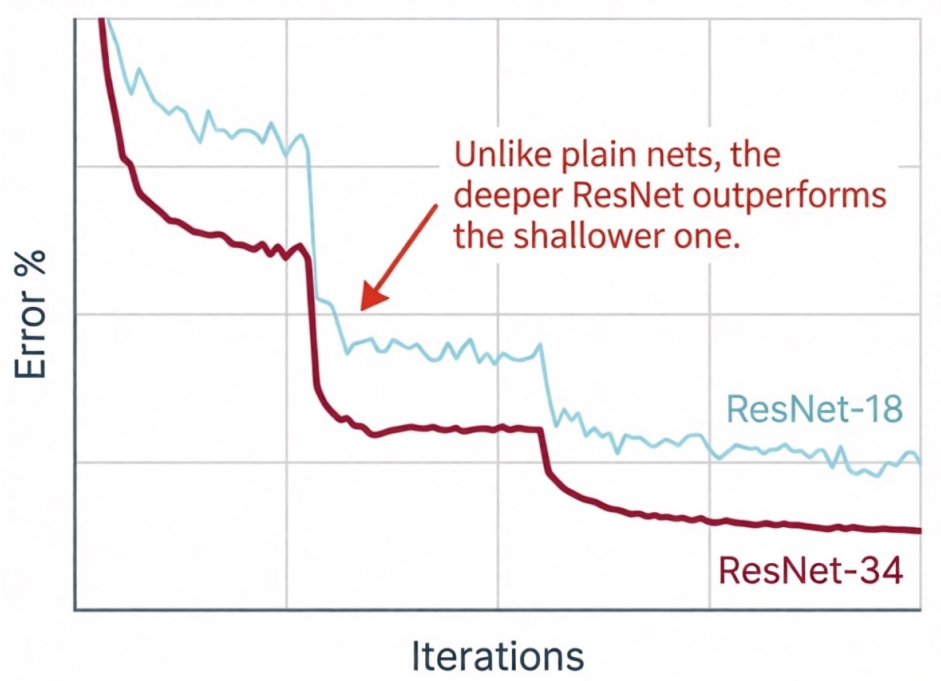
Total Dominance: ILSVRC & COCO 2015

- The hypothesis was validated. ResNet swept every major category in the 2015 competitions.

ImageNet Classification:
1st Place (3.57% error)

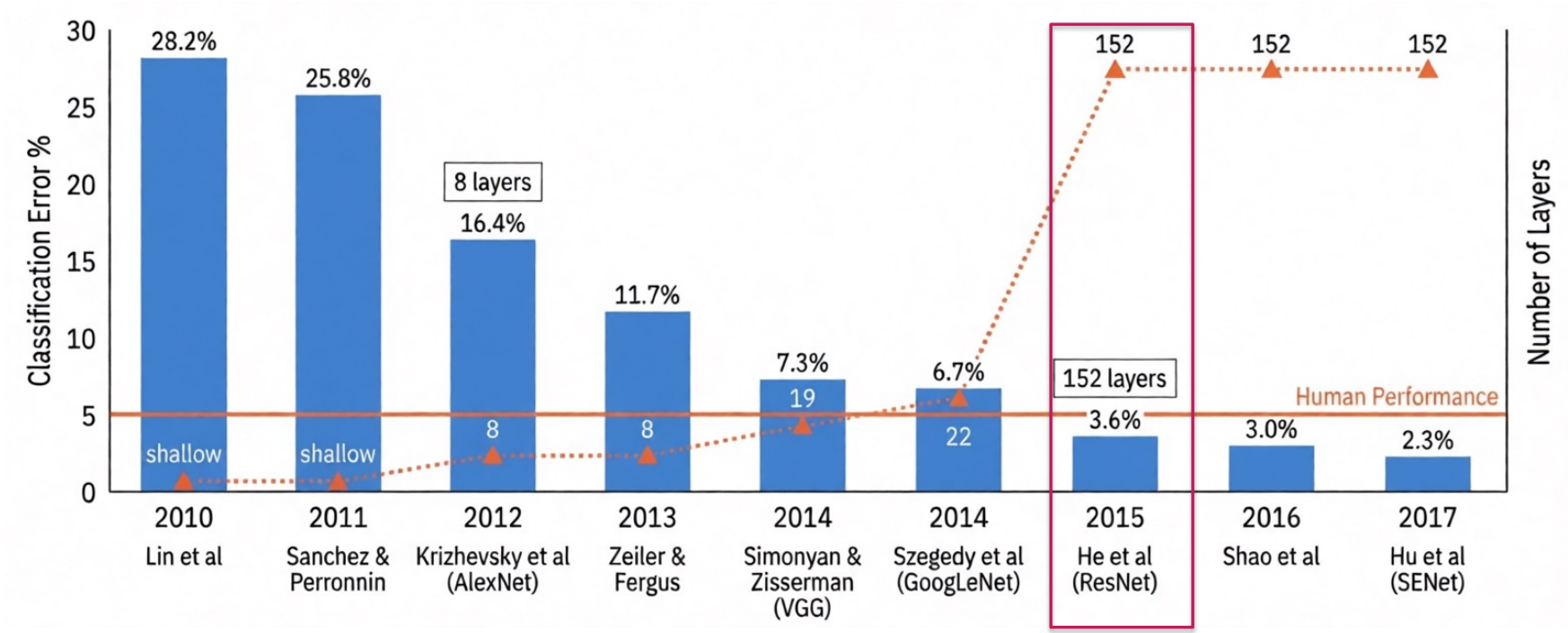
ImageNet Detection:
1st Place (+16% improvement)

ImageNet Localization:
1st Place (+27% improvement)



The 152-Layer Revolution

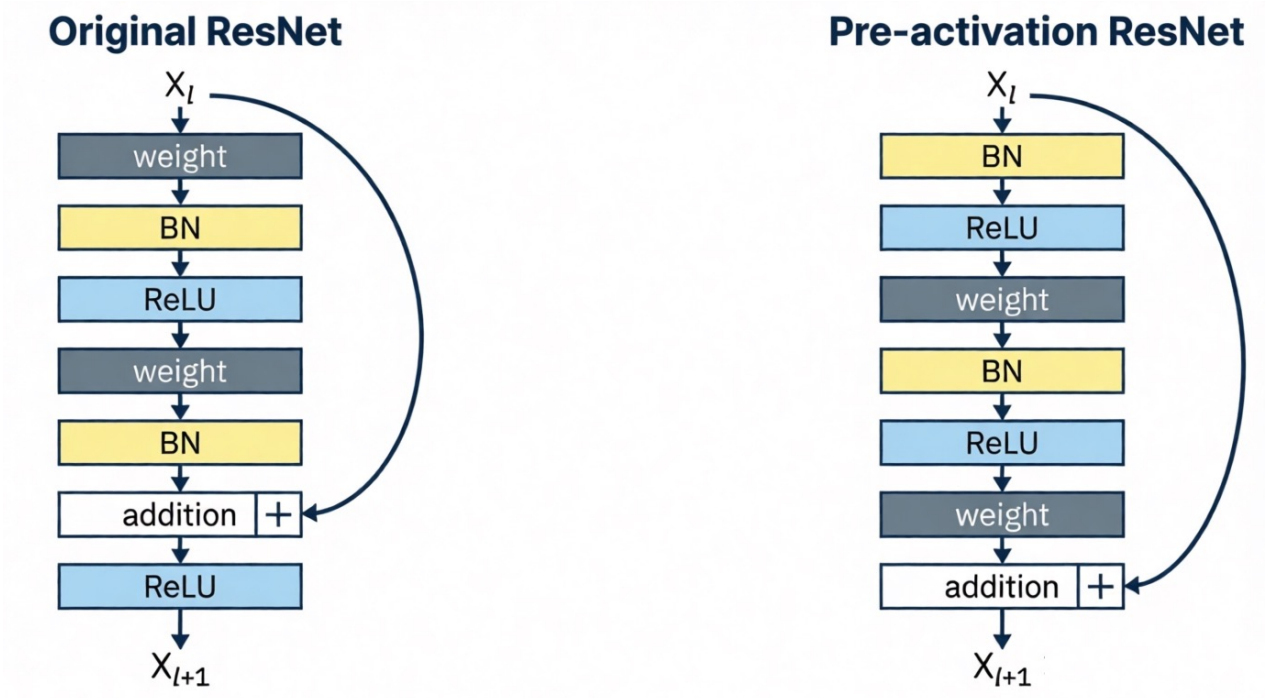
ILSVRC 2015 Results



Efficiency Paradox: ResNet-152 (11.3B FLOPS) is deeper but lighter than VGG-19 (19.6B.

Refining the Formula: Identity Mappings (2016)

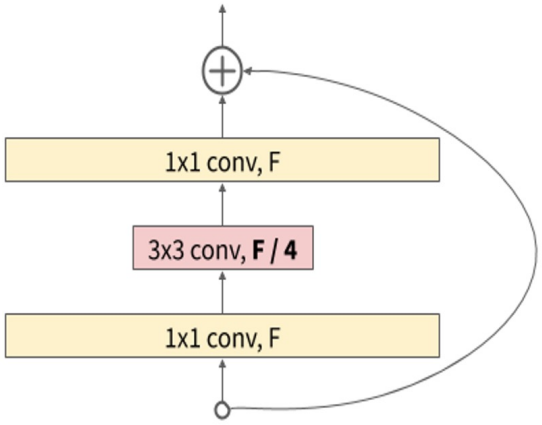
Moving activation functions (ReLU) to the "**Pre-activation**" position clears the path for signal propagation, enabling training of 1000+ layer networks.



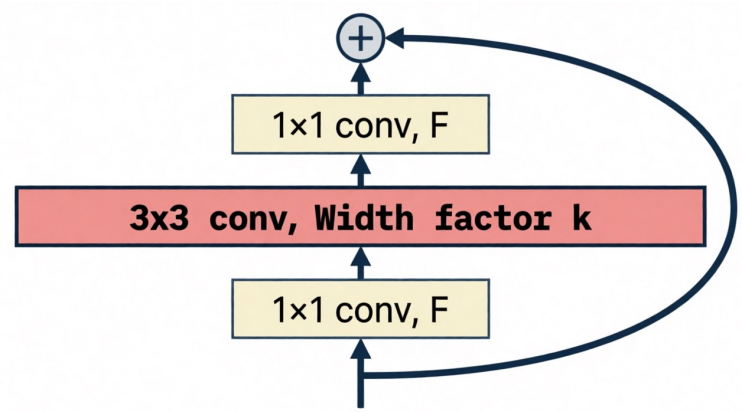
Moving Activation before Convolution clears the propagation path.

Challenging Depth: Wide ResNets (2016)

- Is depth the only answer? Increasing the width (channels) proved more parallelizable and faster to train.



ResNet bottleneck

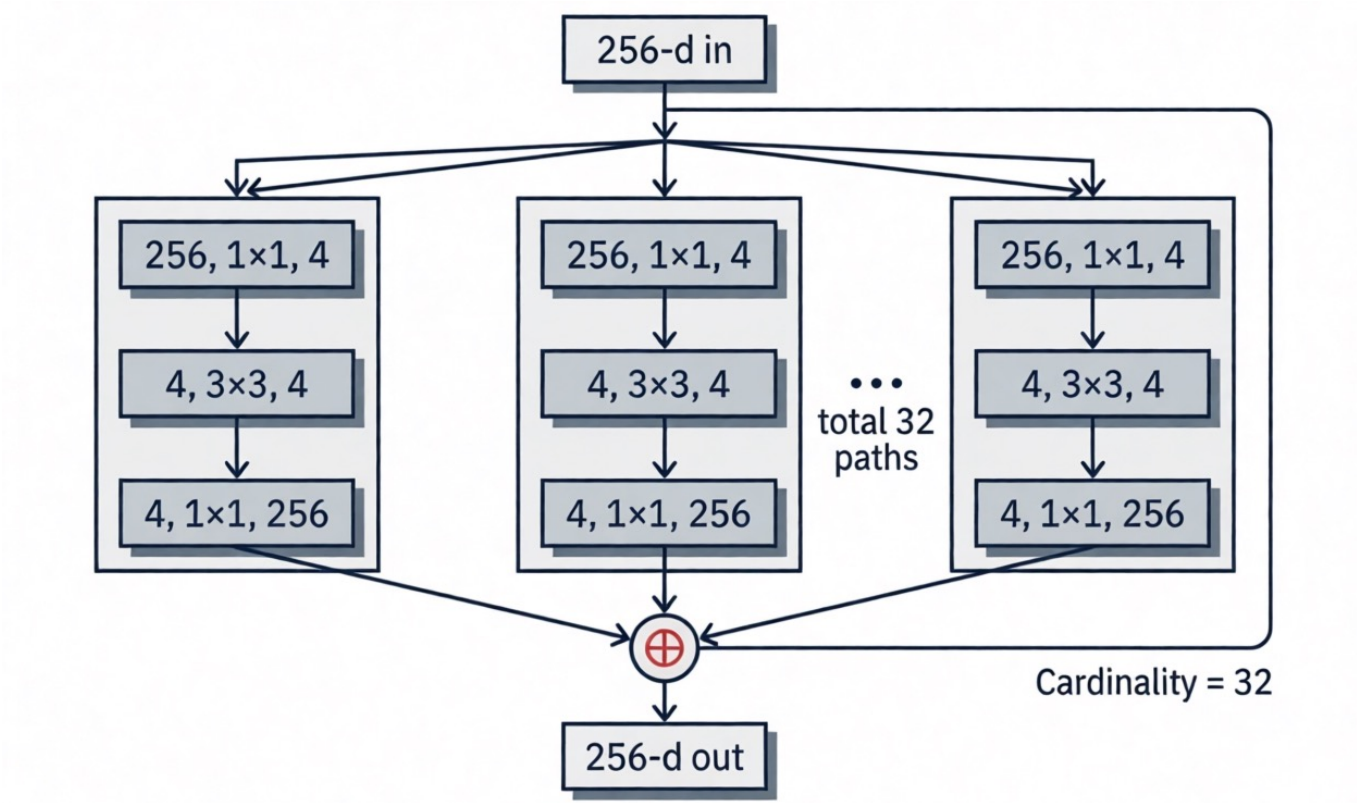


Wide ResNet bottleneck

Result: A 50-layer Wide ResNet outperforms a 152-layer standard ResNet.

Increasing Connectivity: ResNeXt (2017)

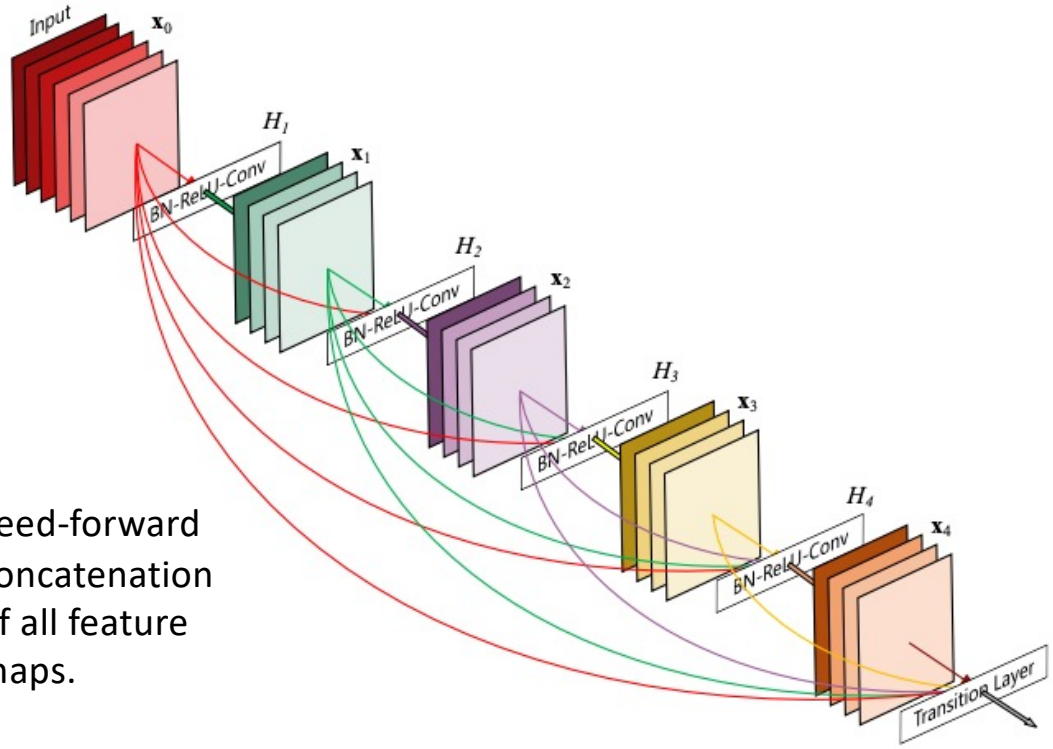
The 'Split-Transform-Merge' Strategy.



'ResNeXt bottleneck: High cardinality (C=32) allows for more parallel transformations, enhancing representational power without significantly increasing parameters.' (Inter, Deep Navy)

Radical Connectivity: DenseNet (2017)

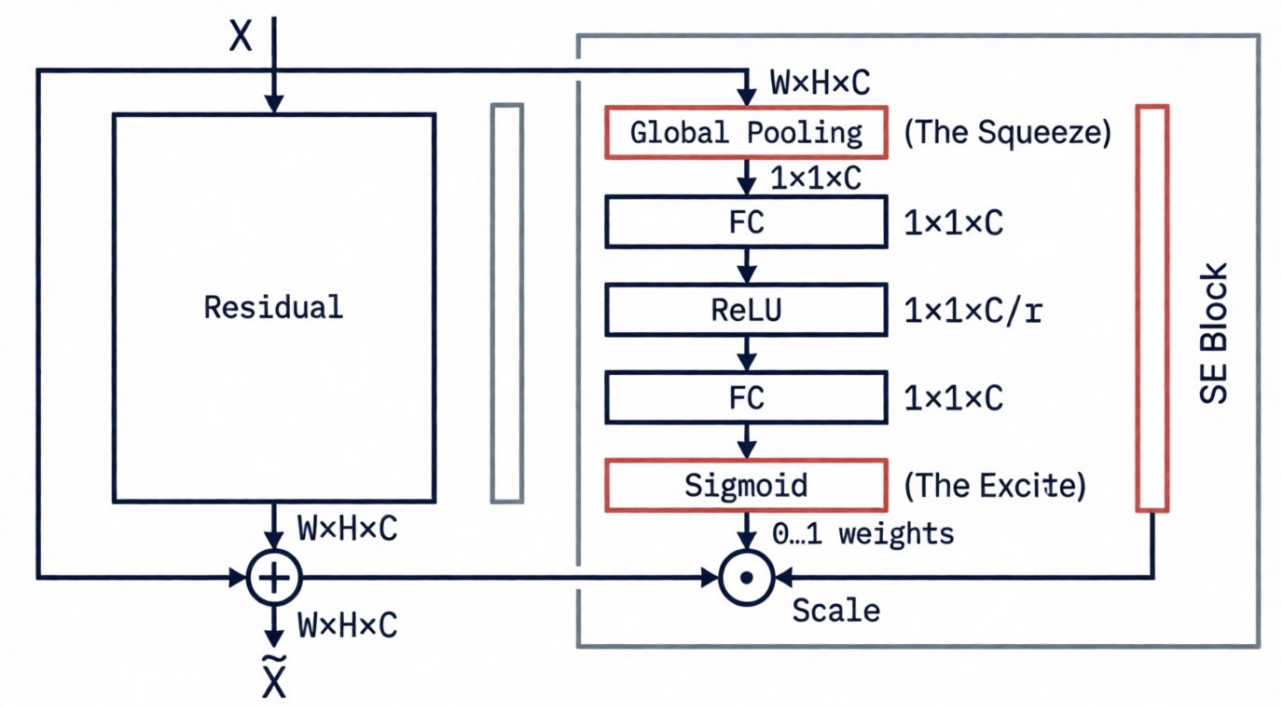
- Connecting every layer to every subsequent layer maximizes feature reuse and signal flow.



Feed-forward concatenation of all feature maps.

Smarter Features: Squeeze-and-Excitation (SENet)

- ILSVRC 2017 Winner. A volume knob for feature channels.

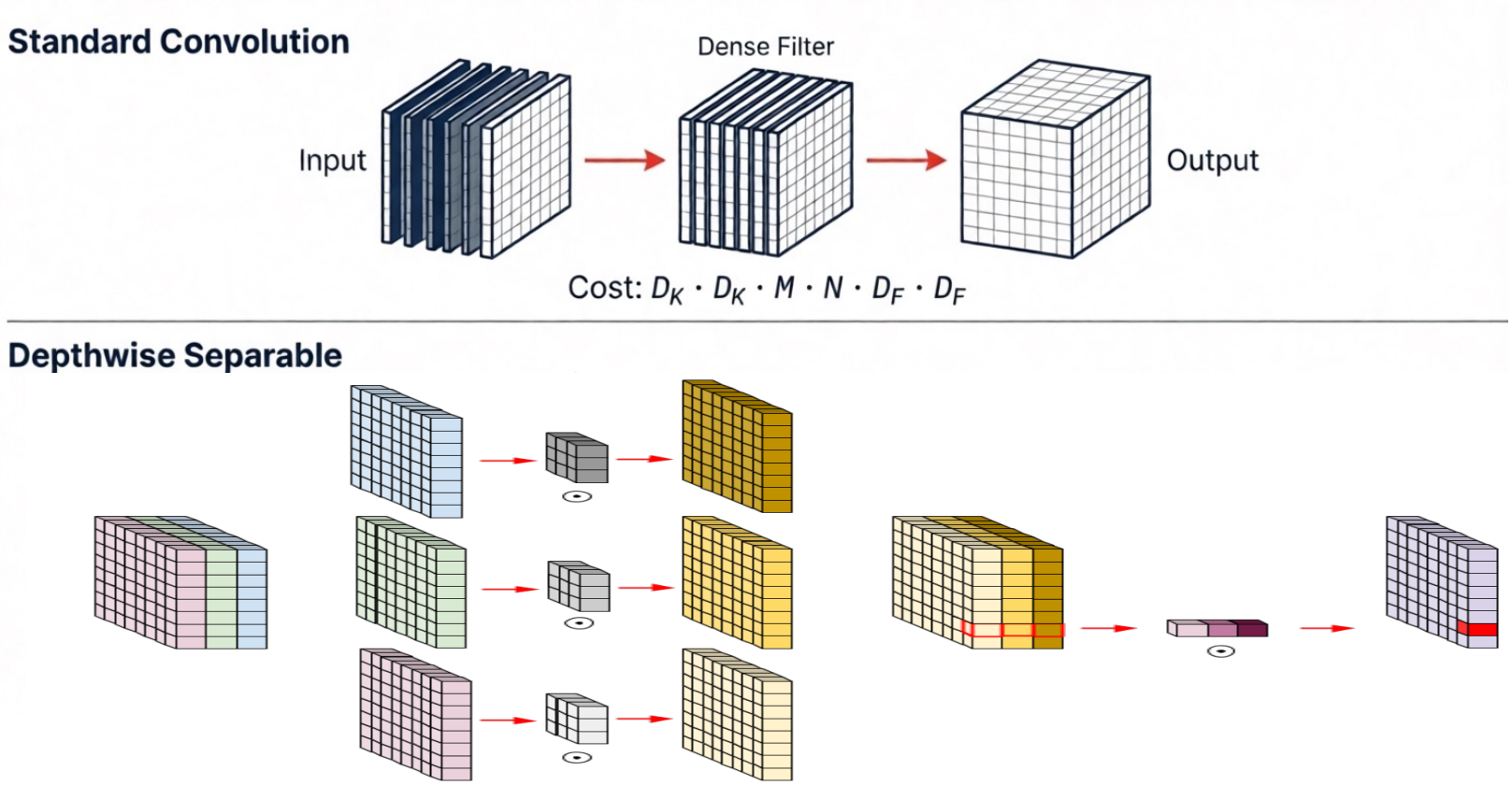


SE-ResNet Module: The SE block explicitly models channel interdependencies to recalibrate feature maps.

56

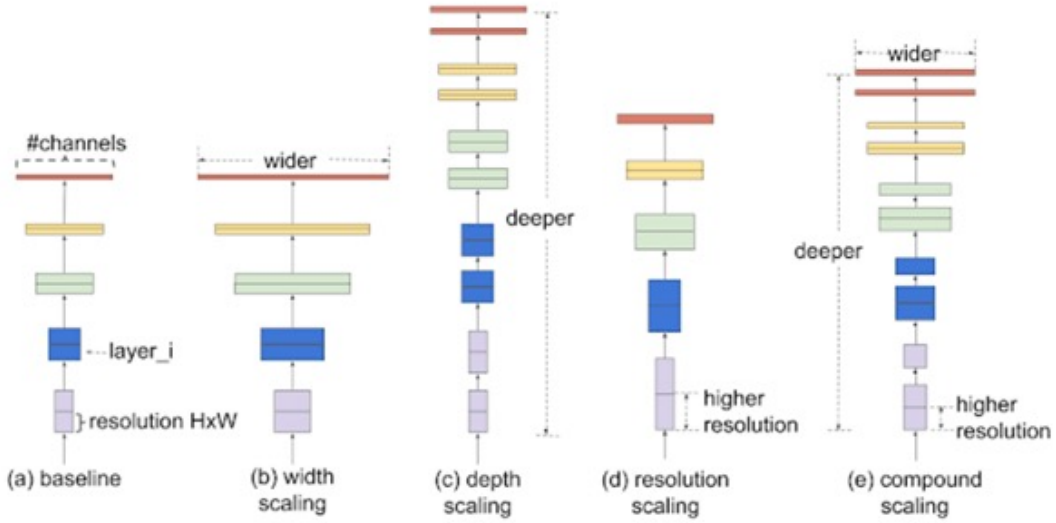
MobileNet Mechanics: Depthwise Separable Convolutions

- Factorizing the convolution to slash computation costs.



EfficientNet (2019)

- 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

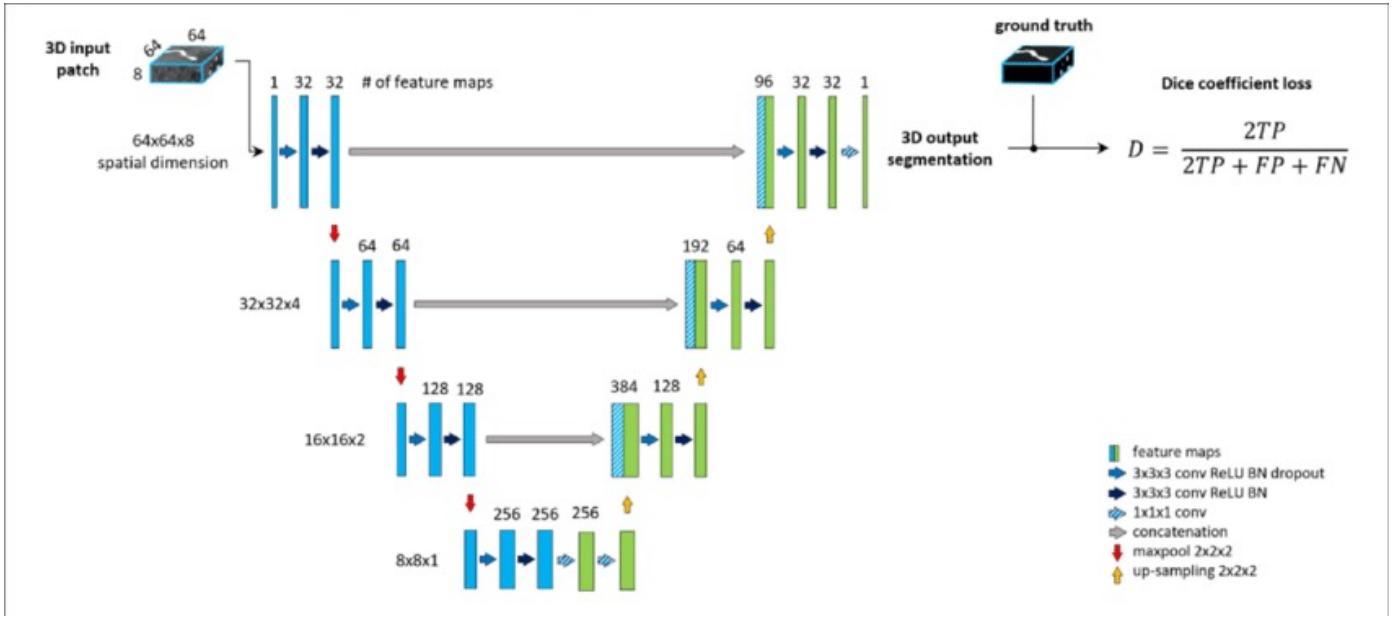


Comparison of different scaling methods. Unlike conventional scaling methods (b)-(d) that arbitrary scale a single dimension of the network, our compound scaling method uniformly scales up all dimensions in a principled way.

58

U-Net (2015): A Different Architecture

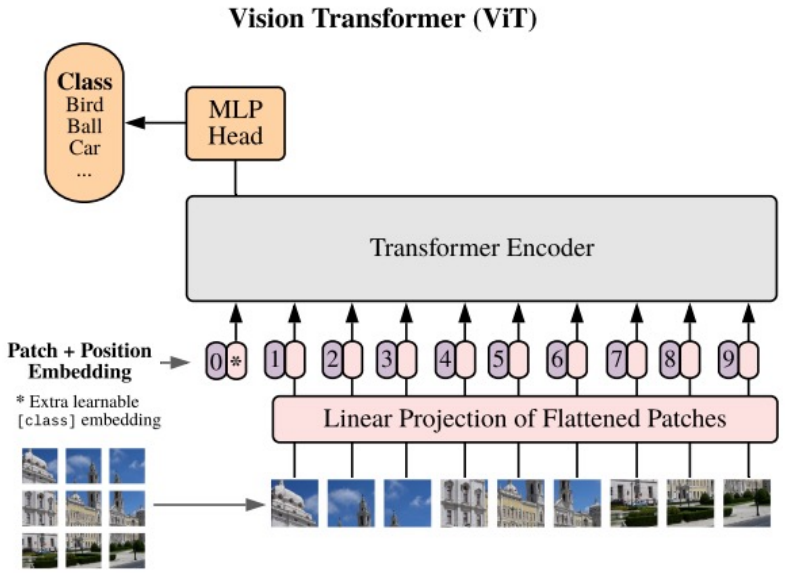
- **U-Net revolutionized biomedical image segmentation** with its **U-shaped encoder-decoder architecture**: downsampling for context, upsampling for localization, and skip connections to preserve spatial detail. Heavy data augmentation overcame limited medical datasets, making it the field's standard.



The Transformer Challenge (2020-2022)

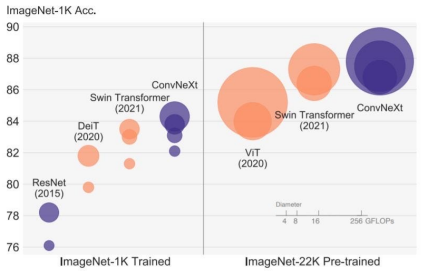
- In 2020, **Vision Transformers (ViT) challenged CNN dominance**. ViT divided images into patches, embedded them as tokens, and processed them with self-attention—modeling relationships between all patches simultaneously.

- With sufficient data and compute, ViT matched or exceeded CNN performance, capturing global context that convolutions struggled with.



ConvNeXt (2022): The CNN Response

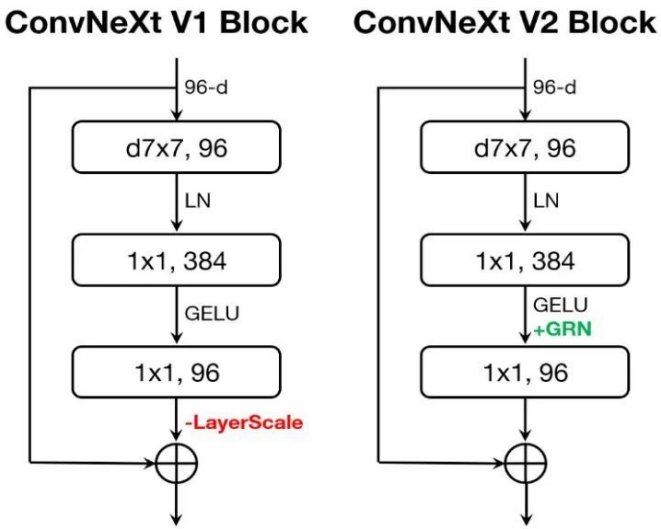
- Rather than concede defeat, ConvNeXt modernized CNNs by adopting Transformer innovations:
  - **Larger patches** (4×4 convolutions with stride 4) mimicked ViT's patching
  - **Depthwise separable convolutions** for efficiency
  - **Large 7×7 kernels** approximated self-attention's global context
  - **Inverted bottlenecks** from MobileNetV2
  - **GELU activation** instead of ReLU
  - **LayerNorm** replacing BatchNorm



- ConvNeXt demonstrated that convolution-based architectures, when properly designed, still compete with Transformers. The best approach often depends on your specific task, data, and computational constraints.

ConvNeXTv2 (2023): Pushing Further

- ConvNeXt V2 (2023) advanced CNNs with **Global Response Normalization (GRN)**—a new layer that boosts inter-channel diversity by normalizing features using global spatial statistics, reducing feature collapse.
- It also featured a simplified architecture, stronger scaling from tiny to huge models, improved masked image modeling for self-supervised learning, and better transfer performance on tasks like detection and segmentation.
- The result: **CNNs that match or surpass Transformers in accuracy while remaining more efficient and interpretable.**



https://medium.com/the-techlife/convnextv2-overview-tutorial-2023-afebce7315e9

Hybrid Architectures and the Path Forward (2024)

- By 2024, CNNs and Transformers converged into **hybrid architectures** that leveraged their complementary strengths.
  - ConvFormer models used convolutional stems for efficient early processing and attention layers for long-range modeling, achieving superior accuracy-efficiency trade-offs.
  - Depthwise attention (e.g., in MaxViT, FasterViT) reduced self-attention's cost by applying it locally, matching global performance with far lower complexity.
  - Dynamic sparse architectures allocated compute adaptively—focusing resources on informative regions—ideal for high-res images and video.
  - The field shifted toward task-specific design: medical imaging adopted attention-enhanced U-Nets, autonomous driving used edge-optimized hybrids, and satellite analysis balanced global context with detail.
  - Hardware co-design became essential, with models tailored to tensor cores, NPUs, and custom chips—not just FLOPs or parameters.

Transfer Learning and Fine-Tuning

The High Cost of Training from Scratch

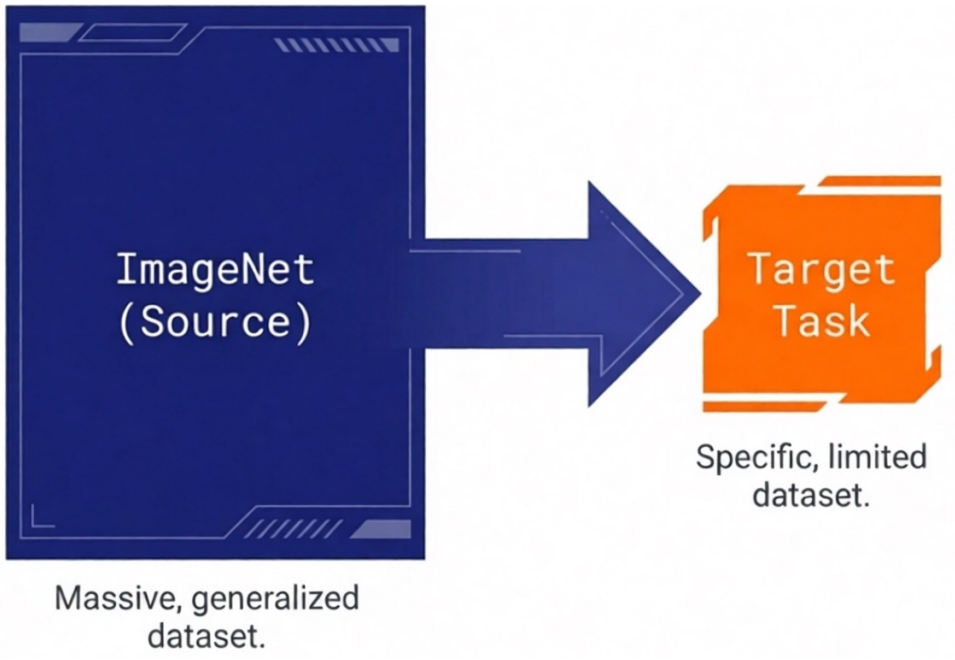
PARAMETER COMPLEXITY	
AlexNet:	60m
VGG:	138m
Inception v3:	23m
ResNet 50:	25m

TRAINING BENCHMARKS	
AlexNet:	5-6 Days (2x GTX 580)
VGG:	2-3 Weeks (4x Titan Black)

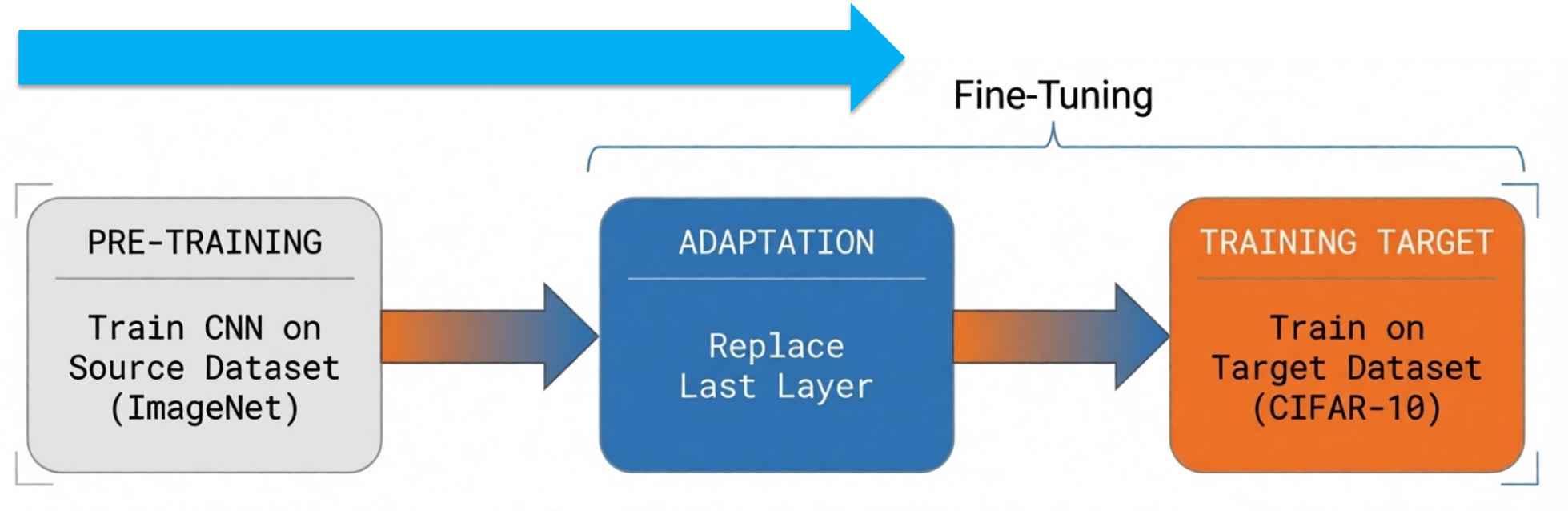
Challenges: High Parameter Complexity • Data Hunger • Extreme Hardware Intensity

Standing on the Shoulders of Giants

- **The Core Concept:** Rather than initializing with random weights, we leverage models pre-trained on massive datasets.
- **The Advantage:** Bypasses the need for colossal data gathering and reduces computational complexity.
- **Key Mechanisms:** Transfer Learning or Fine-Tuning

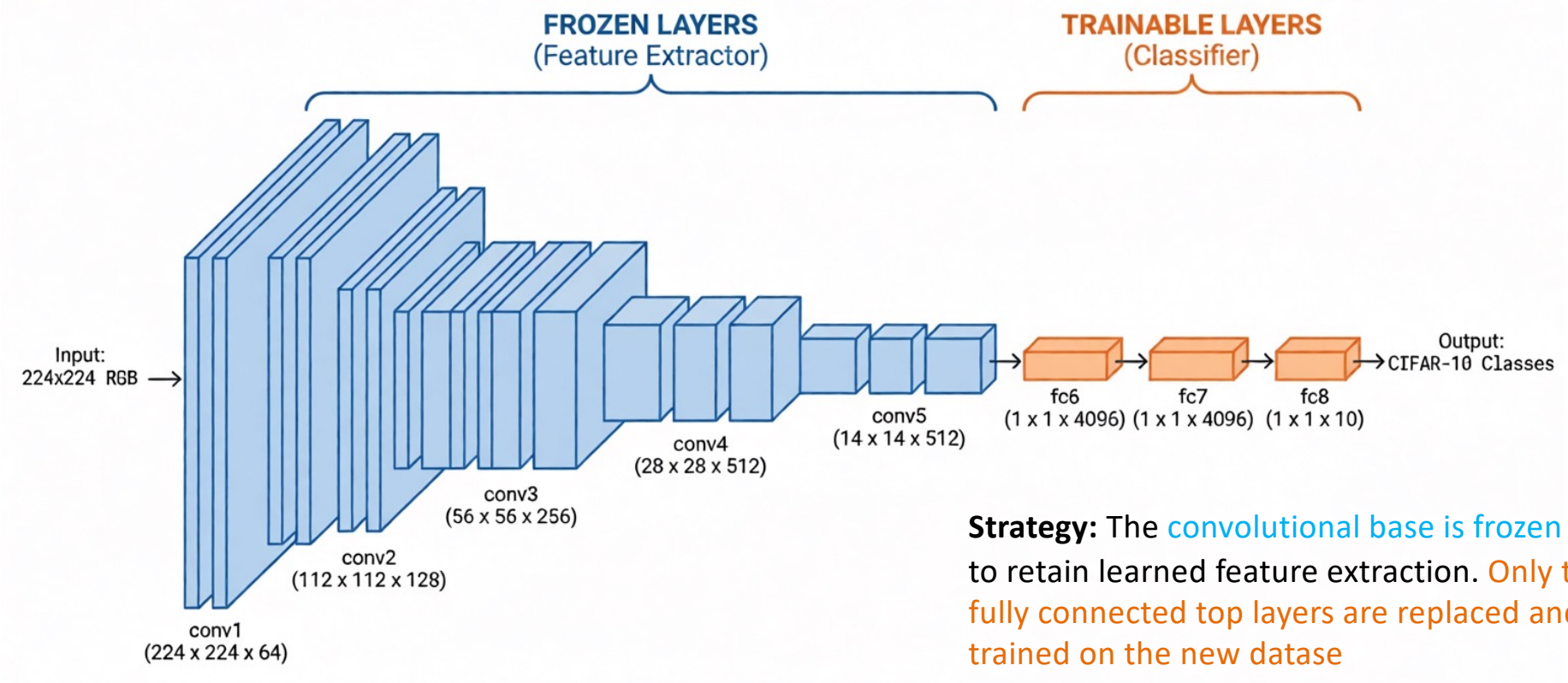


Transfer Learning (or Fine-Tuning)



Implementation Strategy: Adapting VGG16

Case Study: ImageNet to CIFAR-10



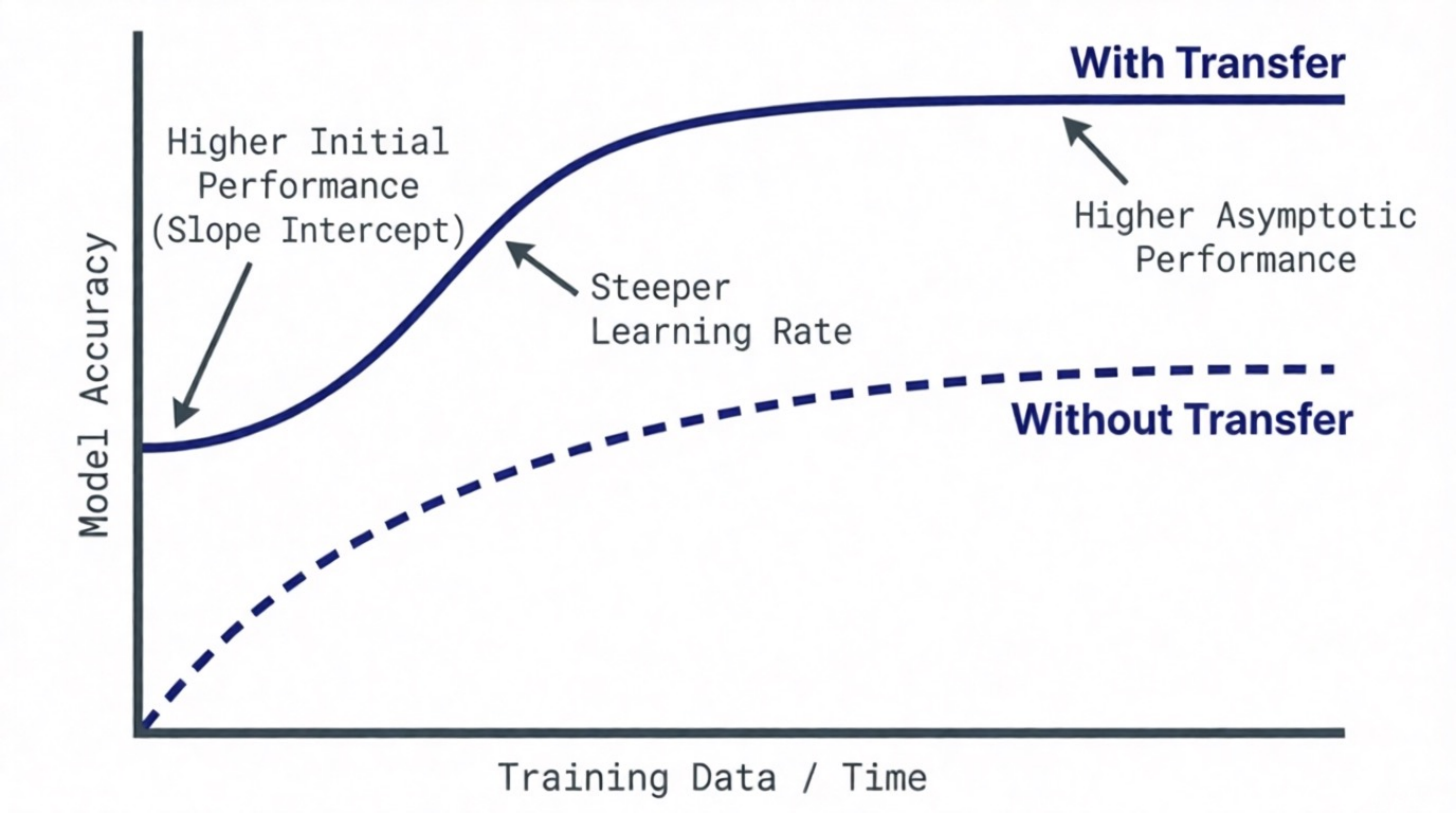
Strategy: The convolutional base is frozen to retain learned feature extraction. Only the fully connected top layers are replaced and trained on the new datase

https://colab.research.google.com/drive/1tRl-AEbV33q6VSkoqgItb15uq2y5NkHG

Case Study: VGG16 on CIFAR-10

> **Result: Transforms a general image recognizer into a specialized classifier without weeks of compute time.**

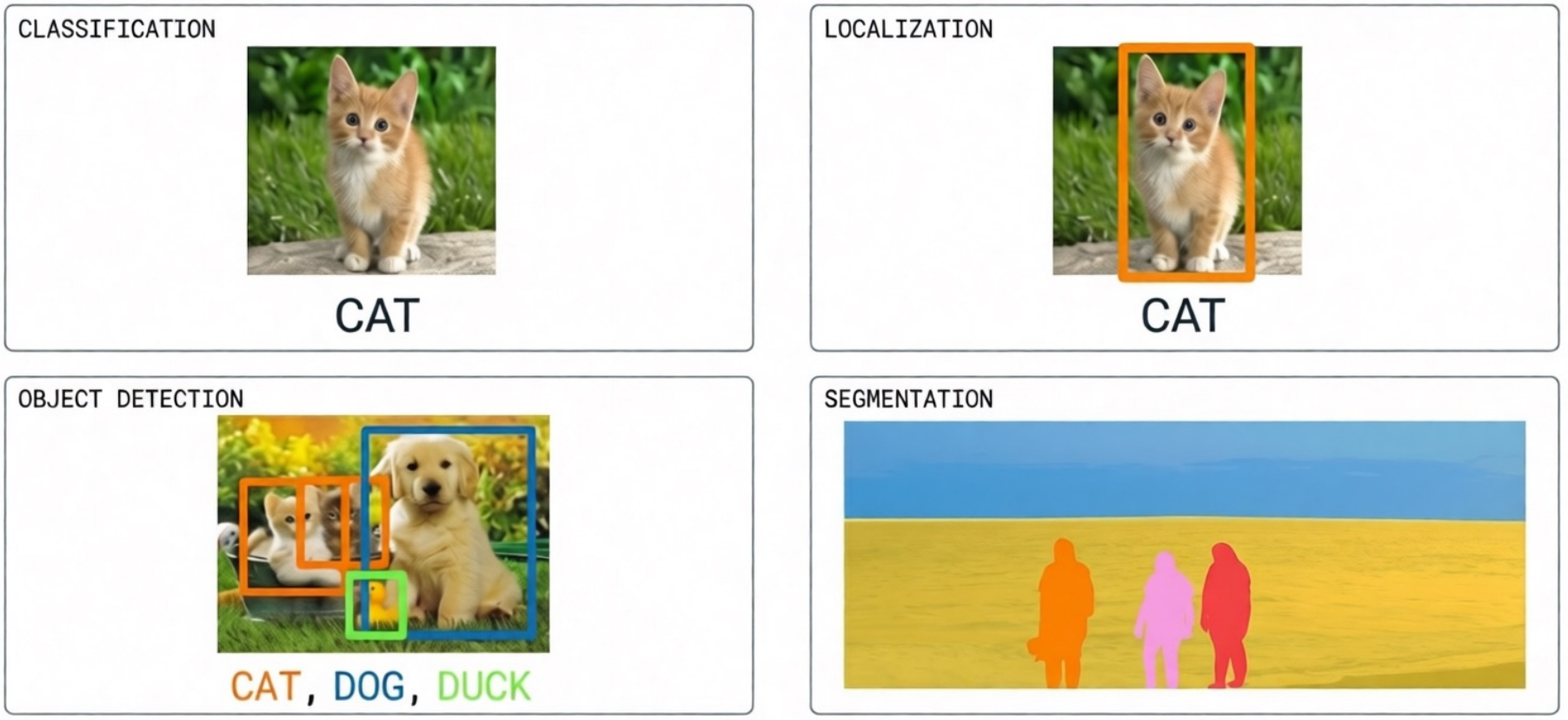
Quantifying the Advantage



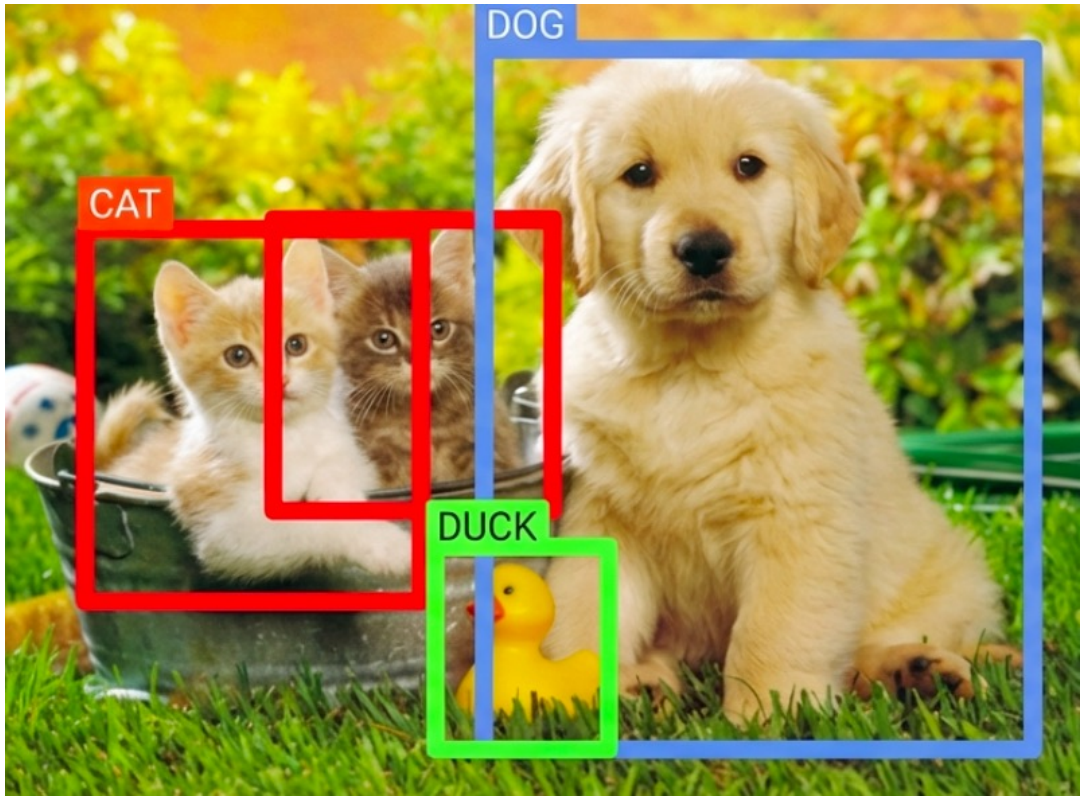
CNN-based Computer Vision Applications

Beyond Simple Classification

CNN capabilities have evolved from single labels to complex scene parsing



Object Detection: Finding the 'Where' and 'What'

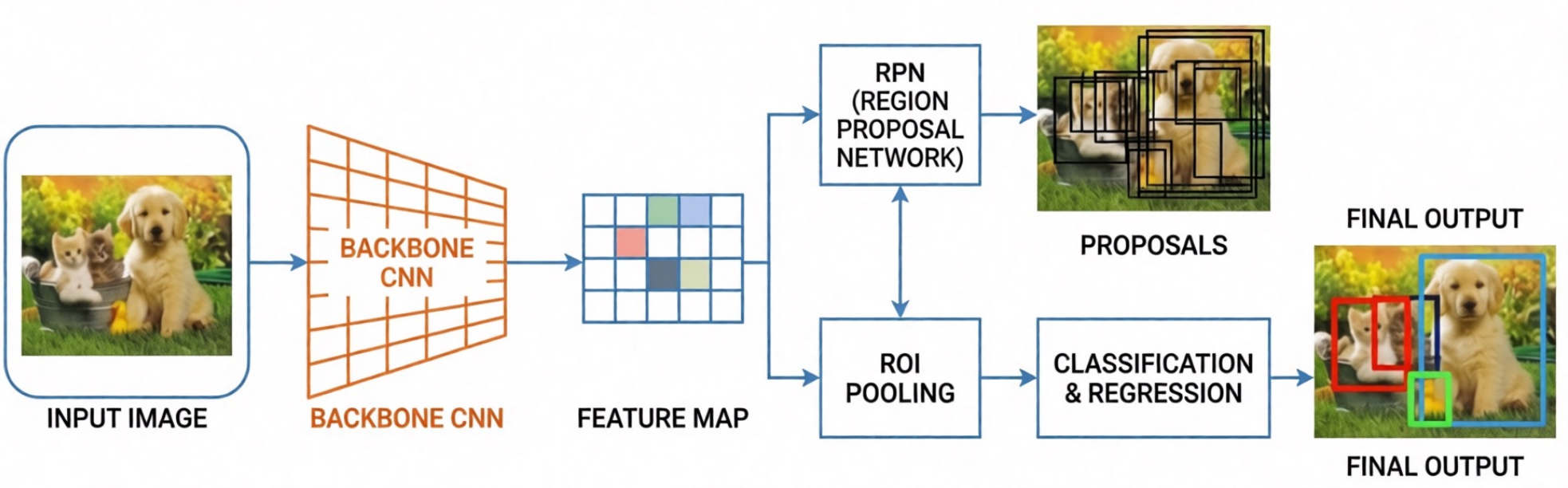


PROBLEM SPACE

- Moving beyond single-label classification.
- Task 1: Identify multiple classes (Classification).
- Task 2: Localize each instance with coordinates (Regression)

PRECISION ARCHITECTURES: THE R-CNN FAMILY

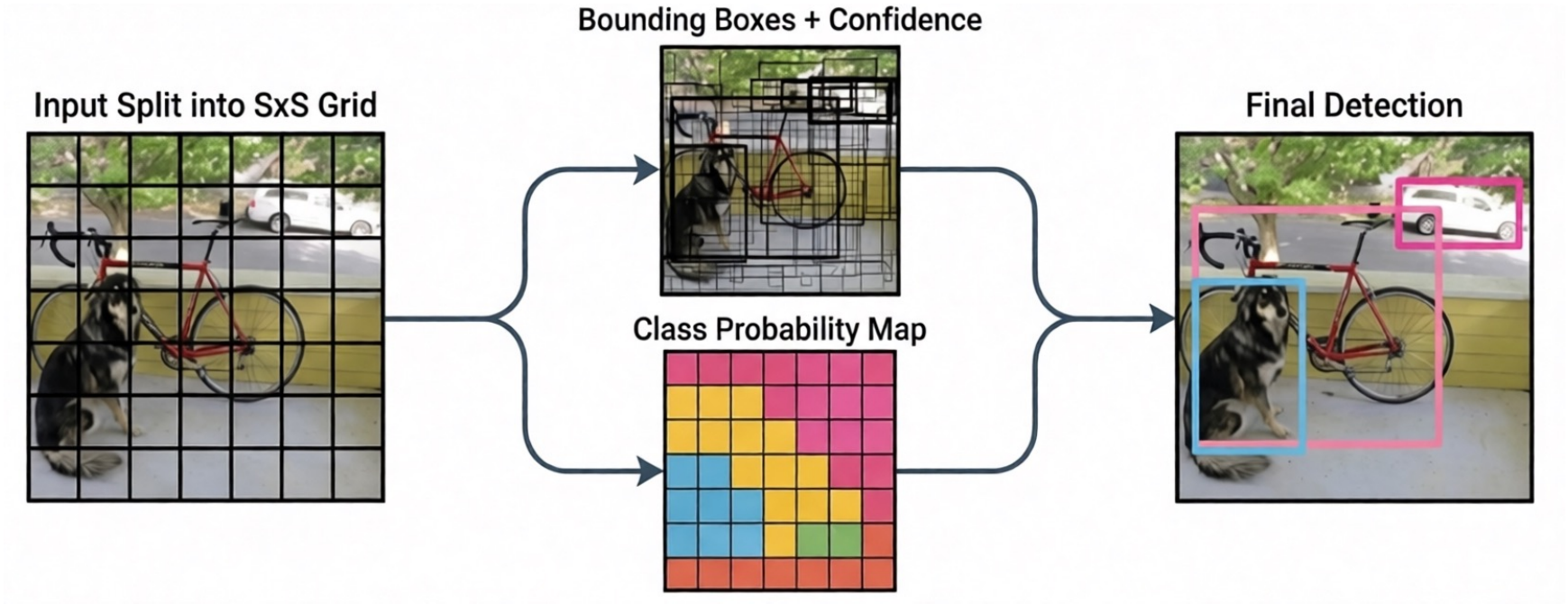
Focus: High Accuracy | Cost: Computational Intensity



Region Proposal Networks (RPN) scan the image for potential objects, which are then normalized and classified. Highly accurate, but computationally expensive.

REAL-TIME SPEED: YOLO (You Only Look Once)

Focus: Speed & Real-Time Inference | Cost: Slight Accuracy Trade-off



Mechanism: Unlike R-CNN, YOLO treats detection as a single regression problem, predicting tensors directly from the image in one pass.

Image Segmentation: Understanding Every Pixel

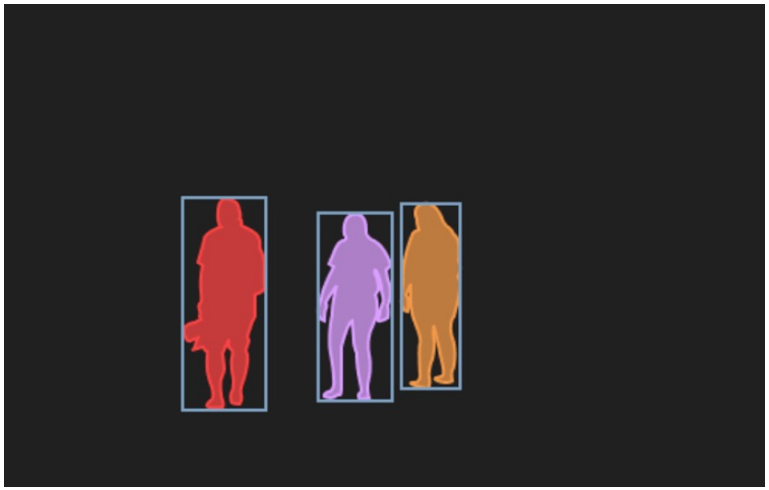
SEMANTIC SEGMENTATION

- Classifies pixels by category (e.g., 'Person', 'Sea'). No distinction between individuals.

Land Sea Sky Person

INSTANCE SEGMENTATION

- Distinguishes between individual objects of the same class (e.g., 'Person 1' vs. 'Person 2').



Person 1, Person 2, Person 3

Advanced Tasks: Human Pose Estimation

Topology Mapping:

- Predicting the coordinates of key joints to model human movement.

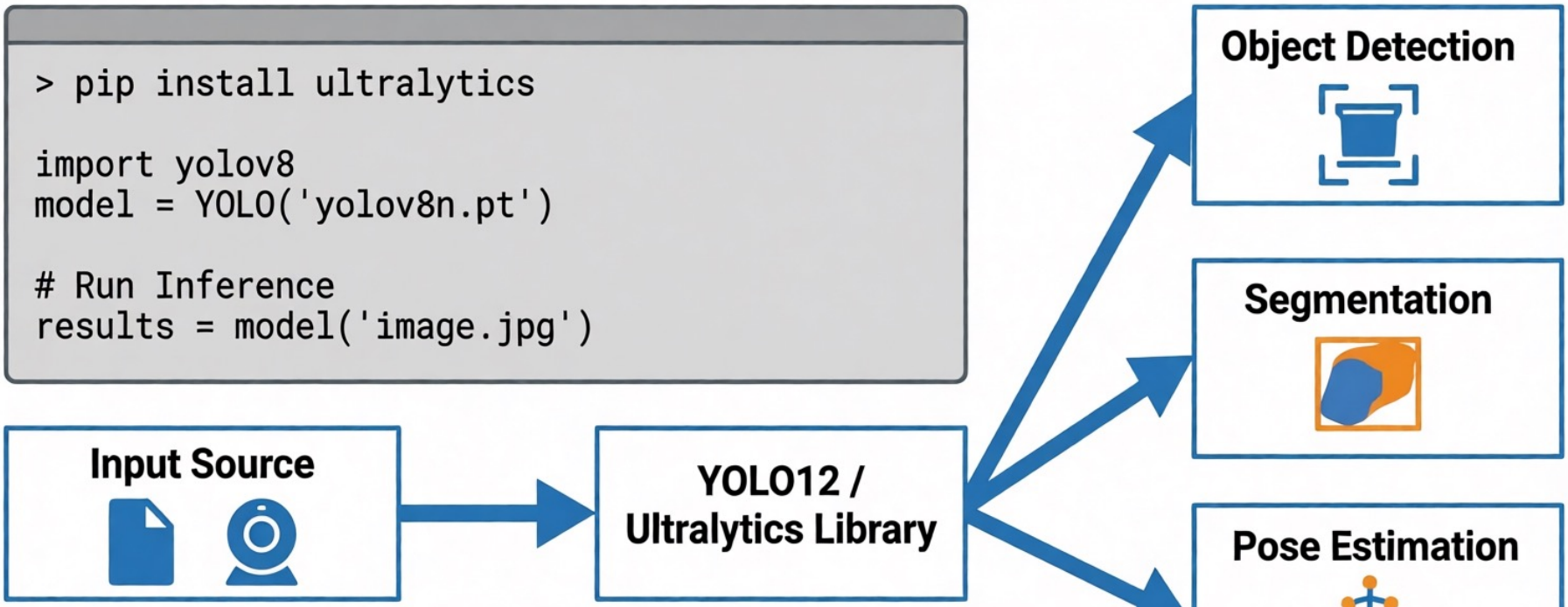
Applications:

- Sports Analytics
- Healthcare/Rehab
- AR/VR Gaming

Skeletal structure overlay showing predicted joint positions.

Practical Deployment: The Modern Workflow

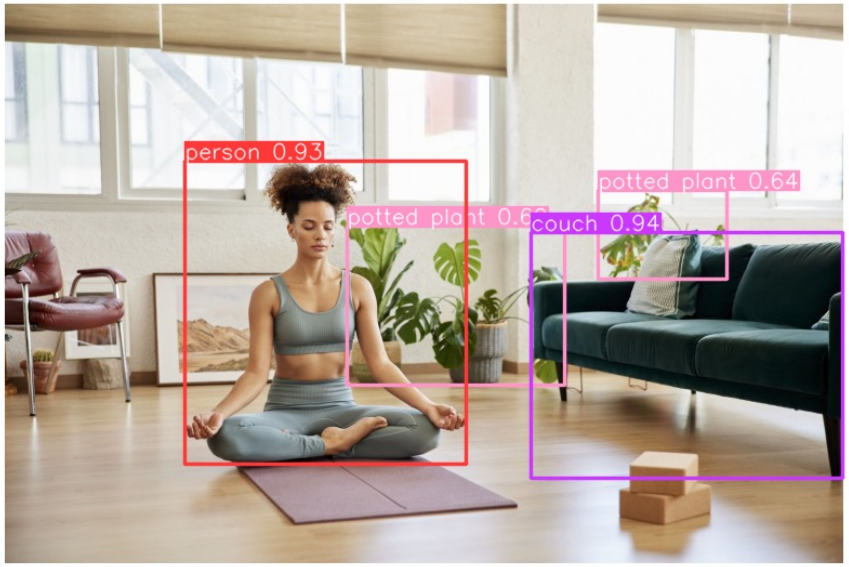
```
> pip install ultralytics

import yolov8
model = YOLO('yolov8n.pt')

# Run Inference
results = model('image.jpg')
```



Accessible via Python libraries for rapid prototyping.

Colab: YOLO12 Object Detection Demo

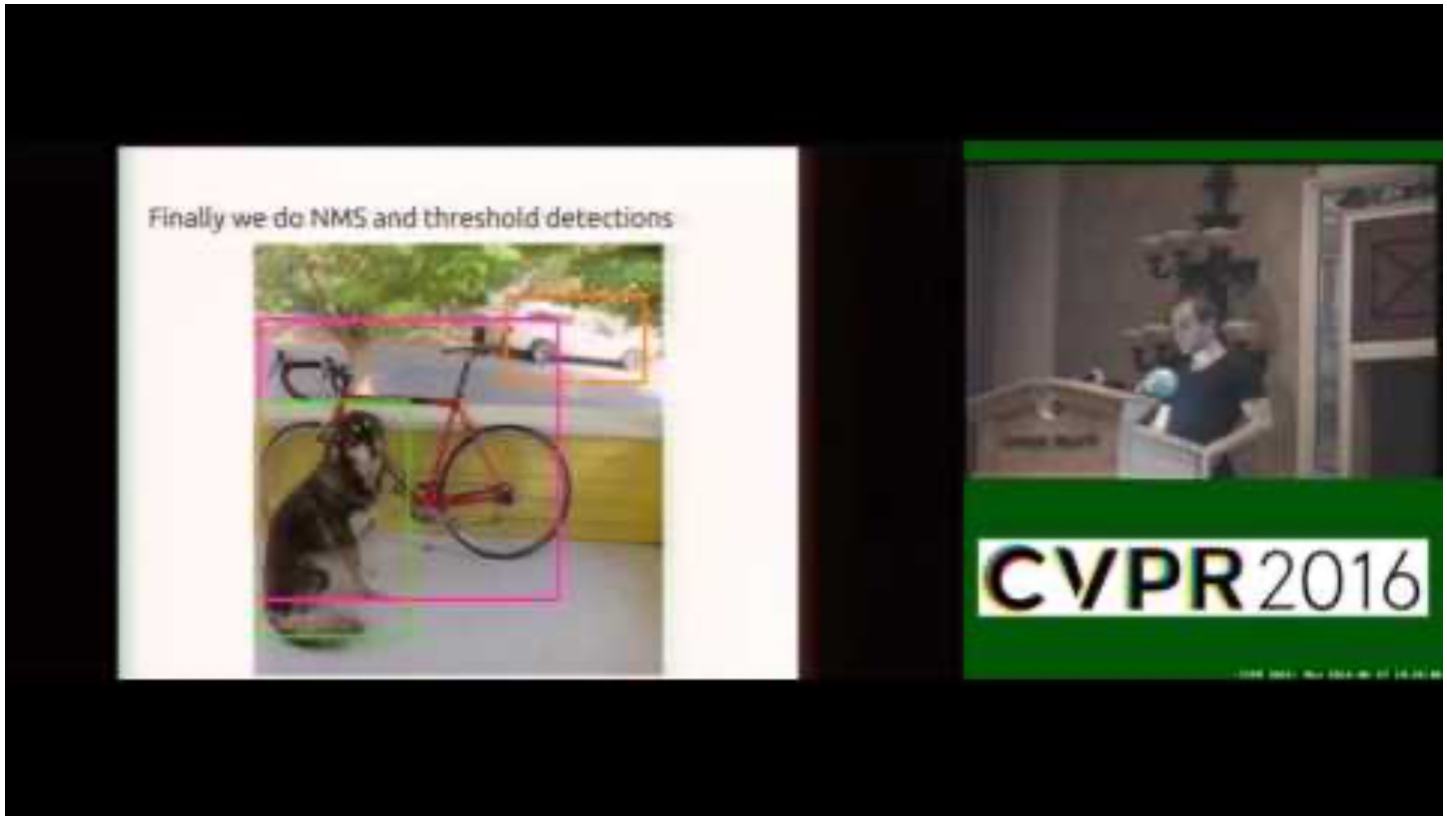
- **We will be implementing the following steps:**
  1. Install YOLO12 from Ultralytics
  2. Running inference on single **image**.
  3. YOLO12 Image Segmentation and Human Pose Estimation
  4. Download sample **video** and run YOLO12 inference on it.
  5. YOLO12 Webcam demo.



https://colab.research.google.com/drive/1XPgQuos-FwMh8pvvmftADSv6XcRG0YKI?usp=sharing

79

YOLO CVPR16 Oral Presentation

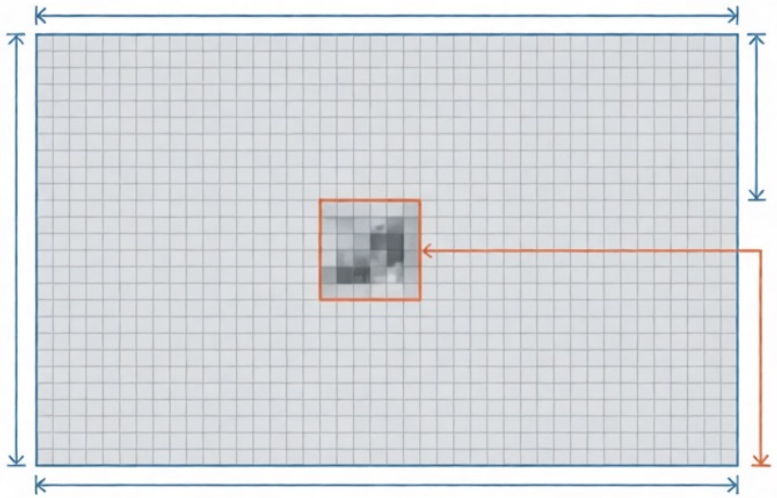


https://www.youtube.com/watch?v=NM6lrxy0bxs

80

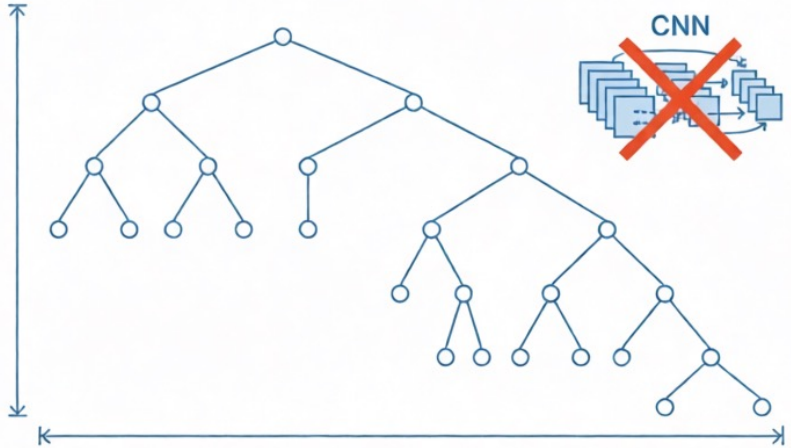
The Structural Limits of CNNs

The Receptive Field Problem



Fixed Receptive Field: CNNs focus on local features (edges, textures) but struggle to connect distant dependencies.

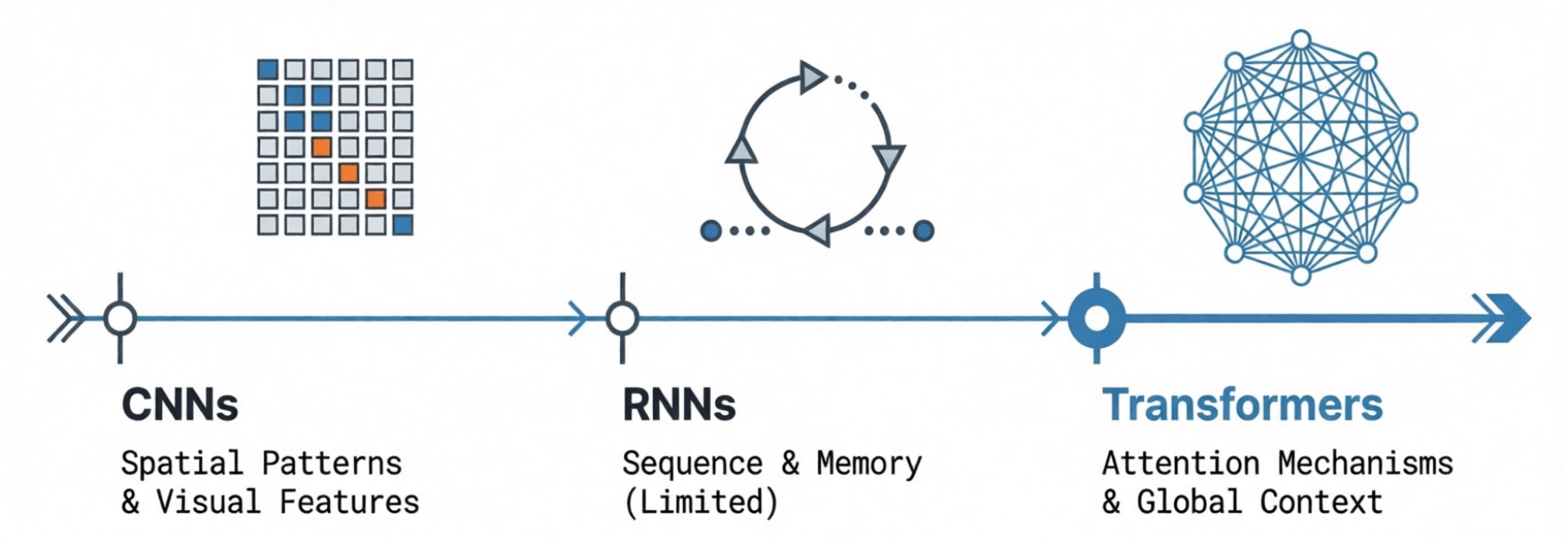
The Hierarchy Problem



Hierarchical Context: Difficulty modeling the nested, sequential structure of natural language or complex logic.

CNNs excel at local features (edges, textures) but struggle to capture long-distance dependencies essential for complex context or natural language.

The Next Horizon: Beyond Convolution



The future lies in architectures that solve the "long-distance dependency" problem using **Attention**.