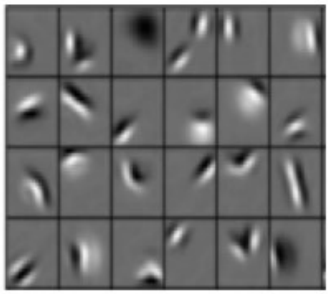
CNN Architectures

Learning Visual Features

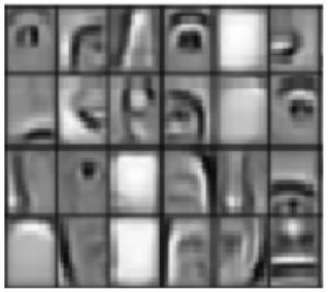
learn hierarchy of features
directly from the data
(rather than hand-engineering them)

Low level features



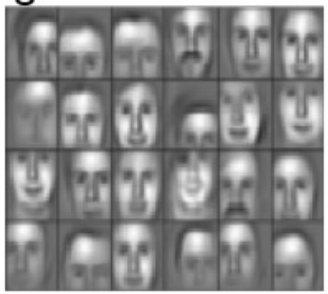
Edges, dark spots

Mid level features



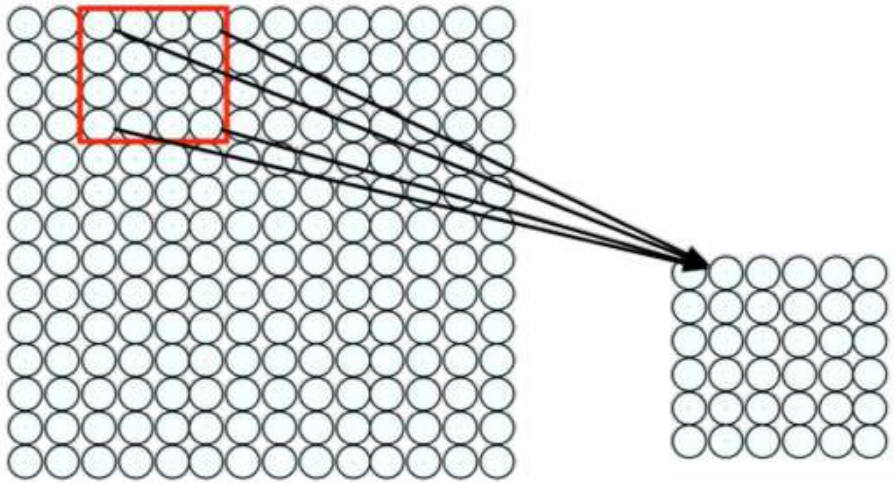
Eyes, ears,nose

High level features



Facial structure

Feature Extraction with Convolution



1) Apply a set of weights – a filter – to extract **local features**

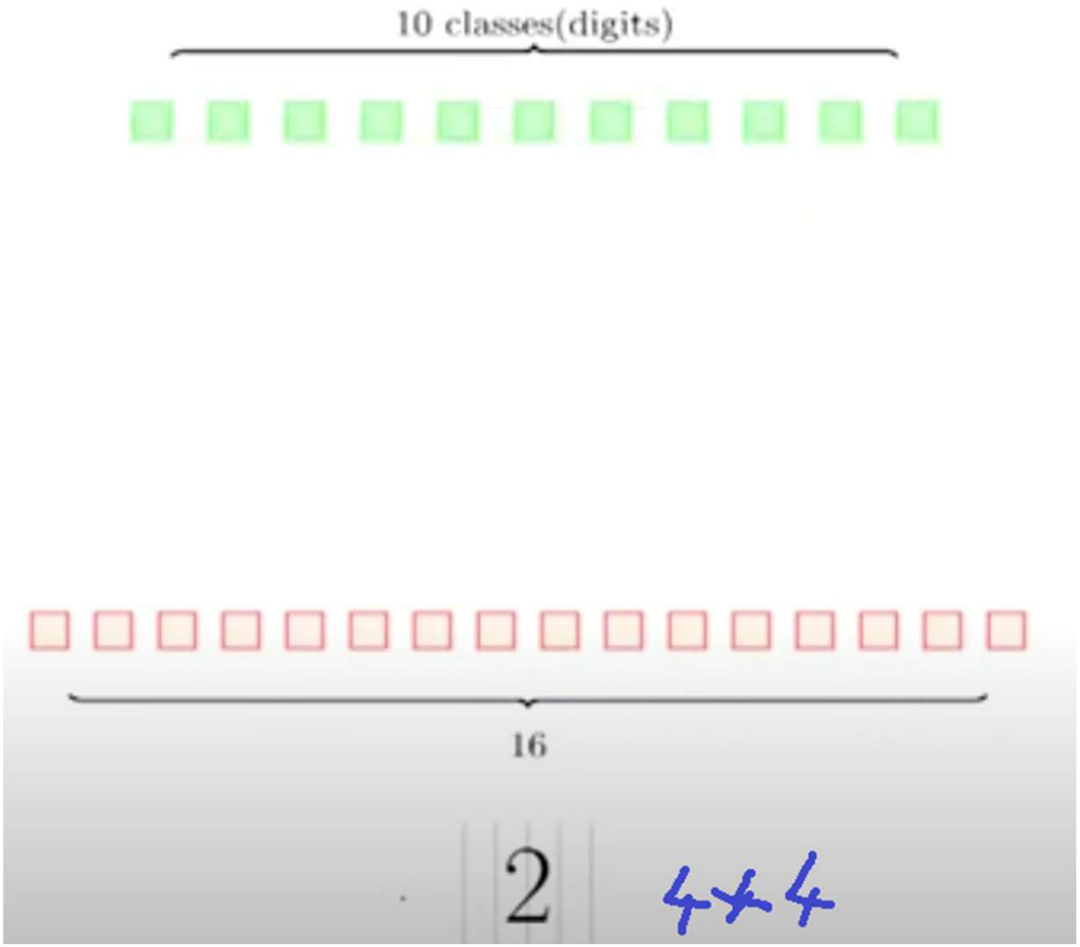
2) Use **multiple filters** to extract different features

3) Spatially share parameters of each filter

How CNN different from FFN?

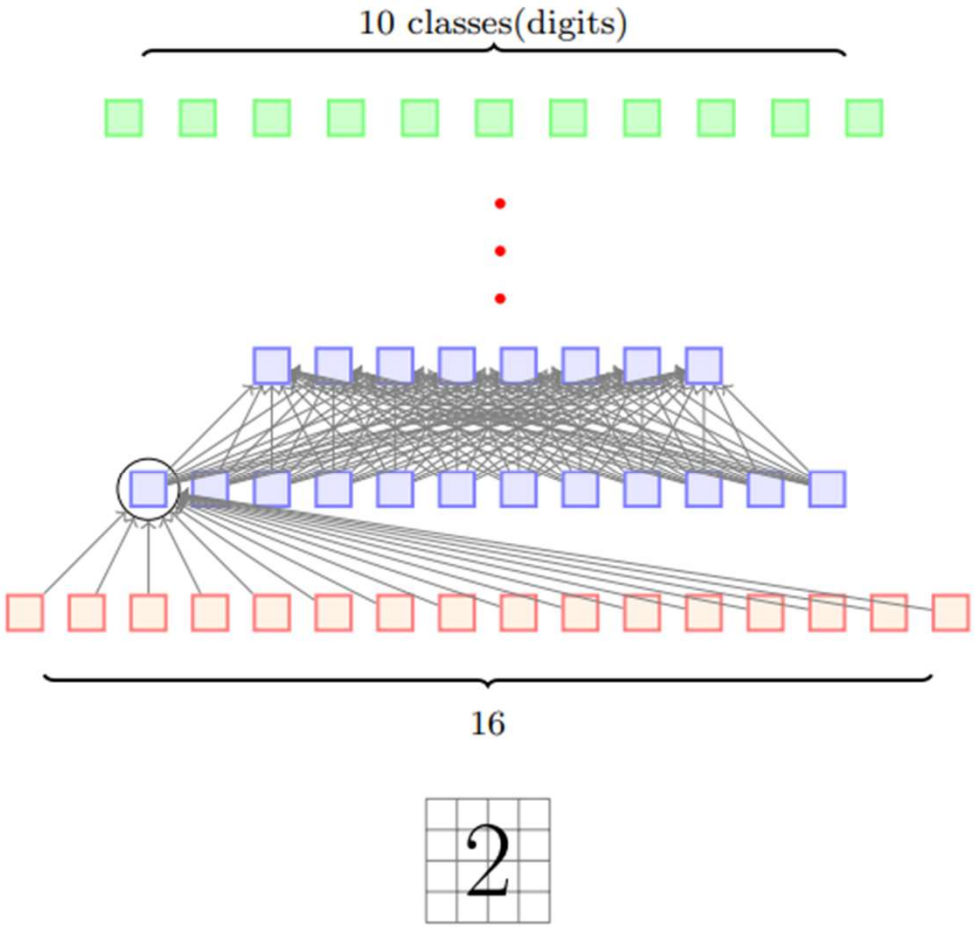
- Sparse Connections
- Weight Sharing

FFN for digit classification

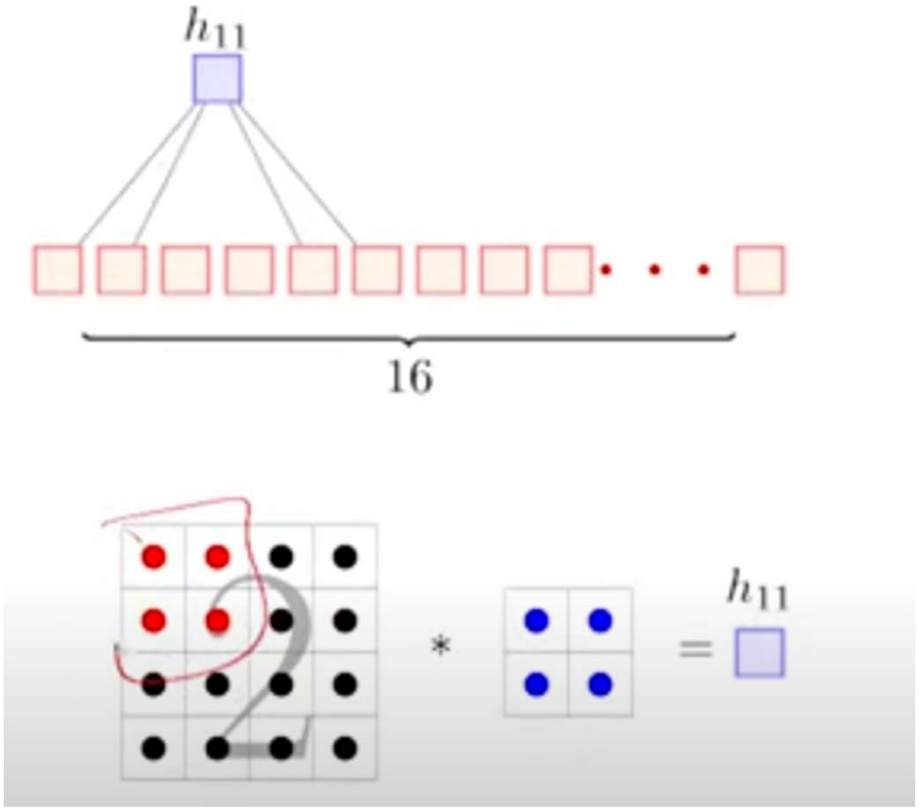


The MNIST dataset consists of **70,000 grayscale images** of handwritten digits,
Training Set: 60,000 images (digits 0–9)
•**Test Set**: 10,000 images (digits 0–9)
•**Classes**: 10 (one for each digit from 0 to 9)

FFN for digit classification

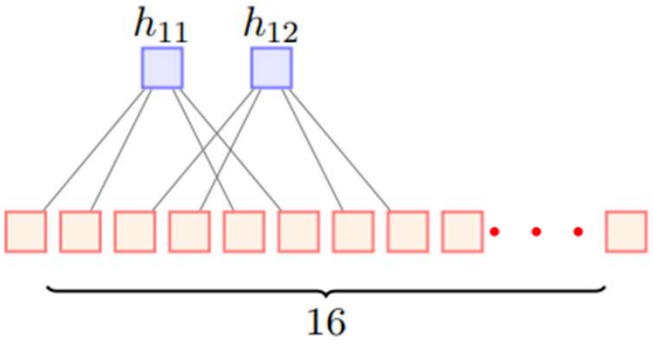


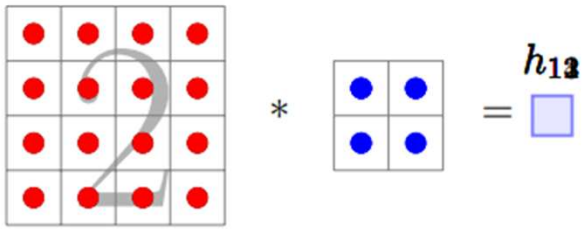
CNN



- Only a few local neurons participate in the computation of h_{11}
- For example, only pixels 1, 2, 5, 6 contribute to h_{11}
- The connections are much sparser

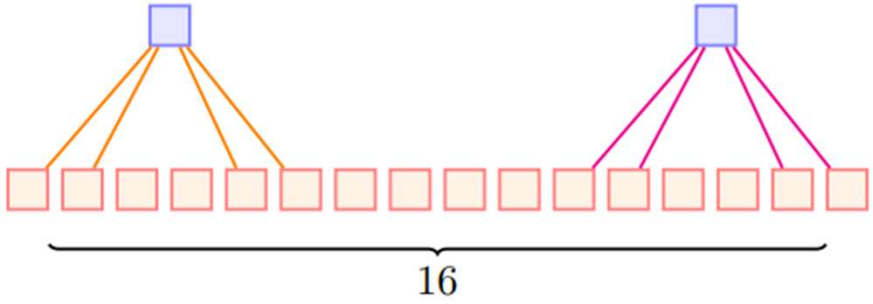
CNN – Sparse Connectivity

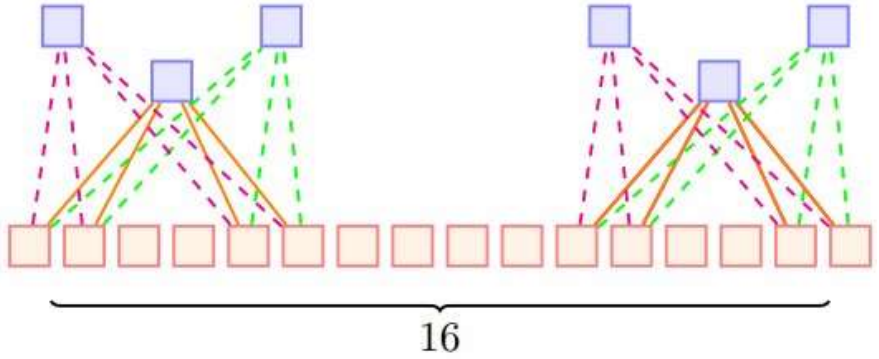


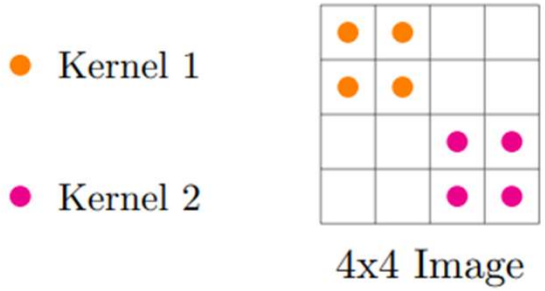


- Only a few local neurons participate in the computation of h_{11}
- For example, only pixels 1, 2, 5, 6 contribute to h_{11}
- The connections are much sparser
- We are taking advantage of the structure of the image(interactions between neighboring pixels are more interesting)
- This **sparse connectivity** reduces the number of parameters in the model

CNN – Weight Sharing



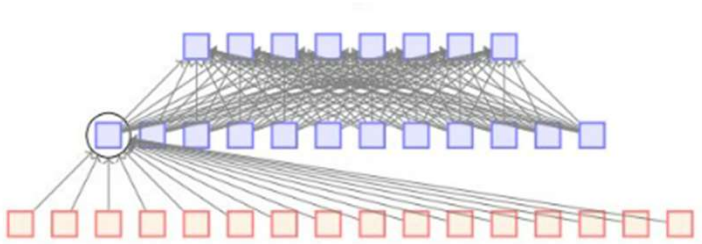


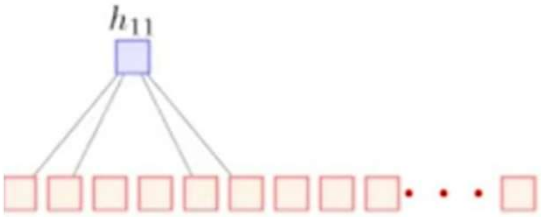


Each convolutional layer applies filters (or kernels) that slide over the input image to detect features, such as edges, textures, or shapes. The same set of weights is used across the entire spatial region of the input for each filter.

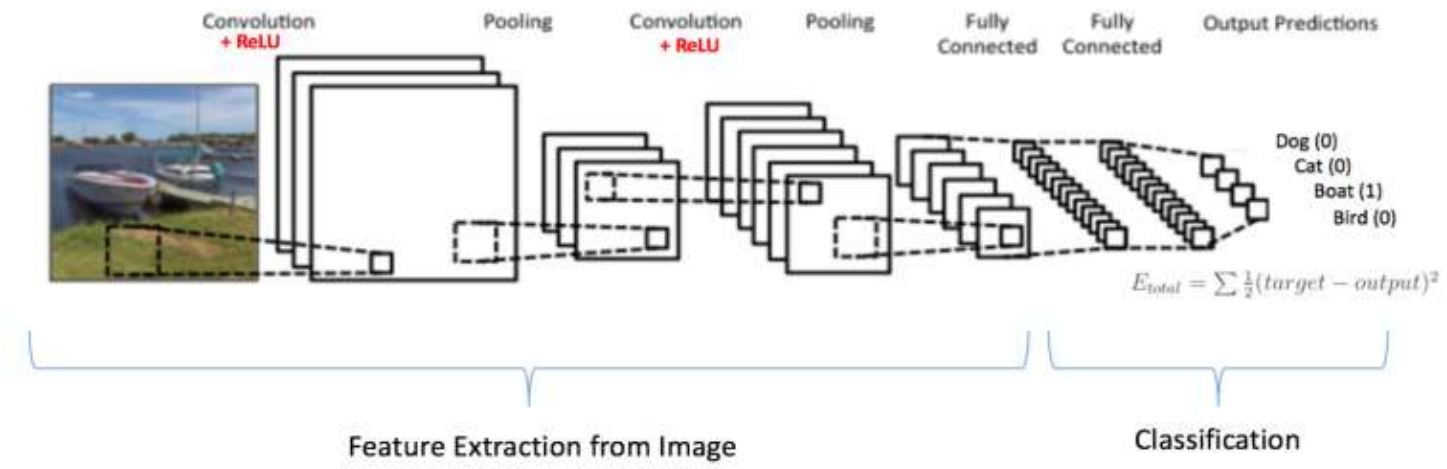
Weight Sharing Edge Detection Example

- In a fully connected layer:
  - Every neuron is connected to every input pixel in the image. This means there's a separate weight for each connection between the neuron and the pixel, with no local pattern focus.
- In a convolutional layer:
  - each filter has a small, fixed set of weights that it uses to look for a specific pattern—like an edge—in every part of the image.
- To detect an edge in a 32x32 pixel image, Fully Connected Layer: it would need 1,024 unique weights, one for each pixel.
- CNN with a 3x3 Filter: 3x3 edge-detection filter that is shared across all 1,024 possible 3x3 locations in the image. This requires only 9 weights,



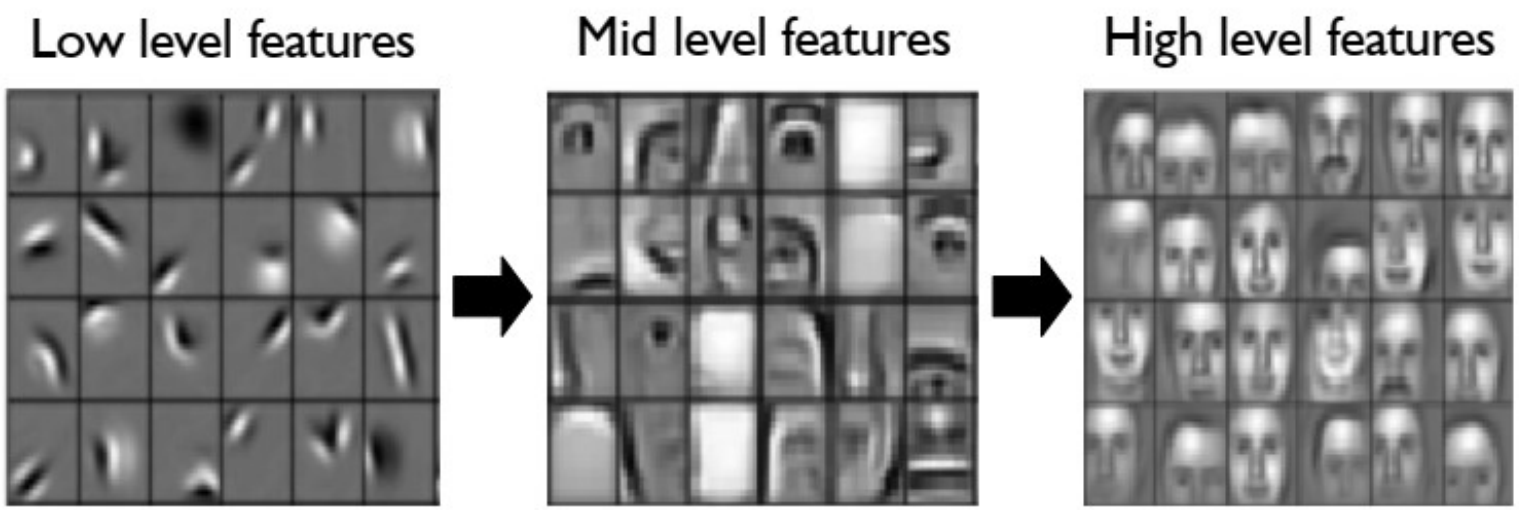


An image classification CNN

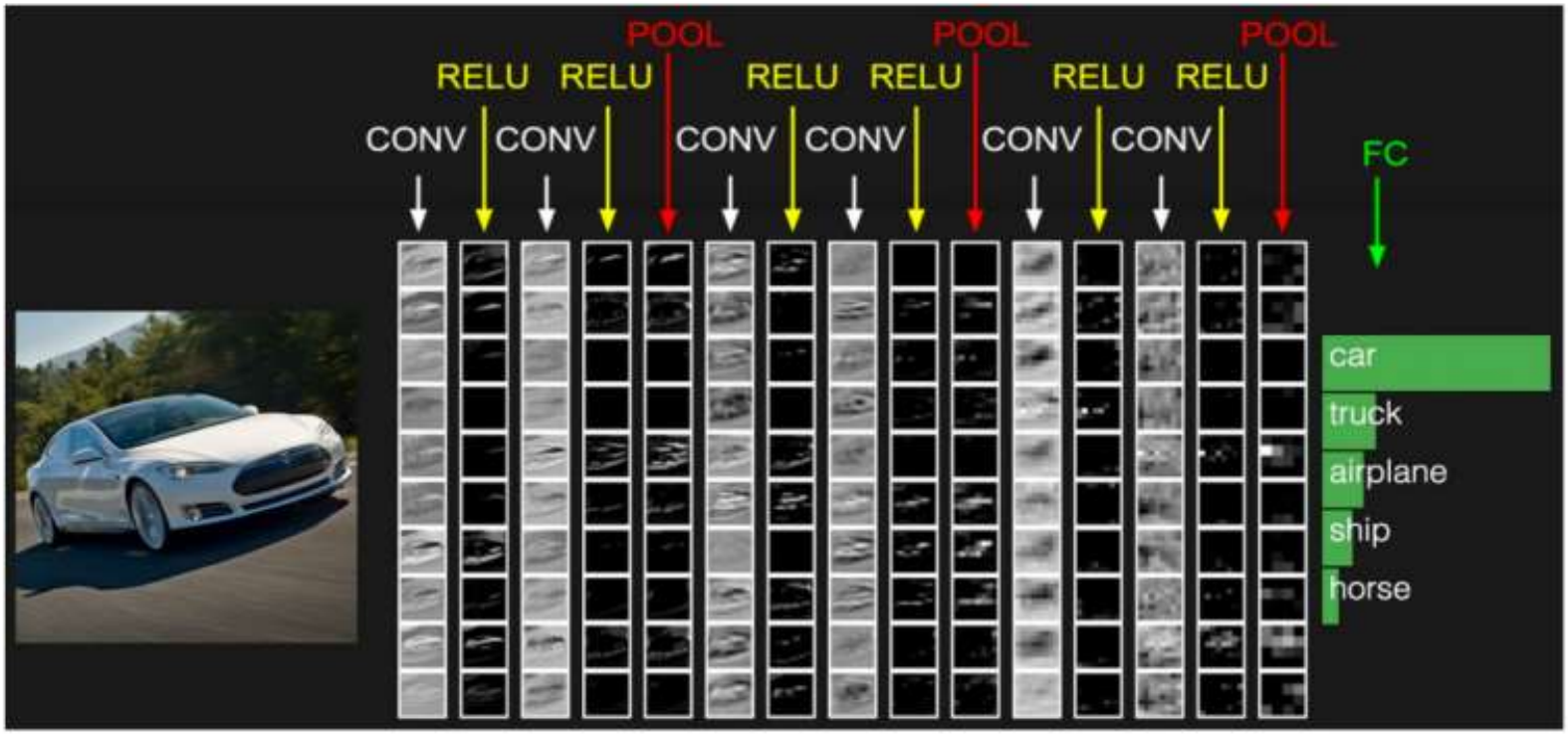


Input Layer: Receives the image data
Convolutional Layers: Extracts local features from the input image by applying convolutional filters (kernels) that detect patterns such as edges, textures, and shapes.
Pooling Layers : Reduces the spatial dimensions (height and width) of the feature maps while retaining important features, thus reducing computational load and adding translational invariance.
Flattening Layer: Converts the 2D feature maps produced by the final pooling layer into a 1D vector, making it compatible wi fully connected layers.
Fully Connected Layers: Combines the features extracted by convolutional layers to perform the final classification.

Representation Learning in Deep CNNs

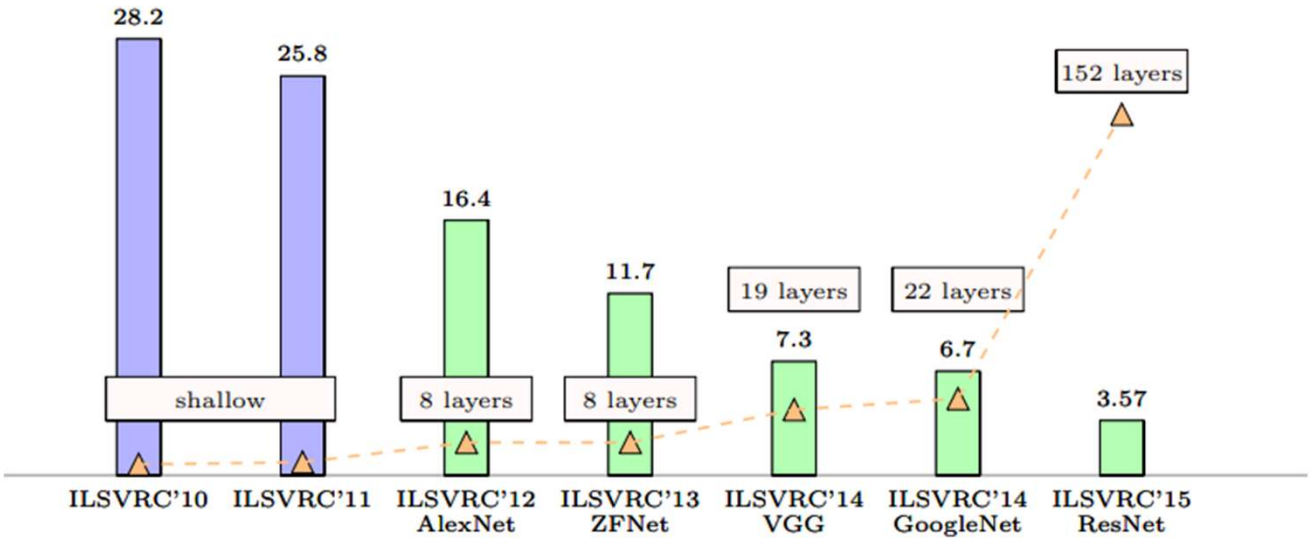


Example – Six convolutional layers



Successful Architectures

- LeNet-5
- AlexNet
- ZFNet
- VGGNet
- GoogLeNet
- ResNet



LeNet-5

- *Gradient Based Learning Applied To Document Recognition - Y. Lecun, L. Bottou, Y. Bengio, P. Haffner; 1998*
- Helped establish how we use CNNs today
- Replaced manual feature extraction

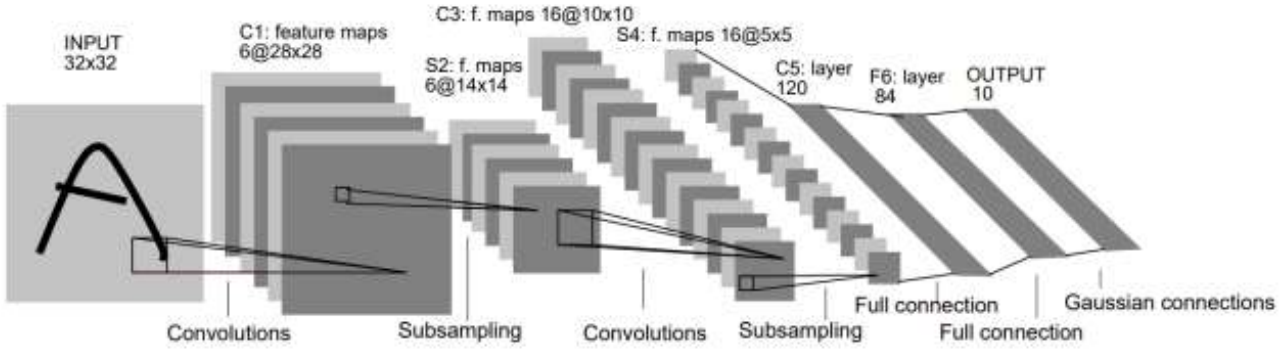
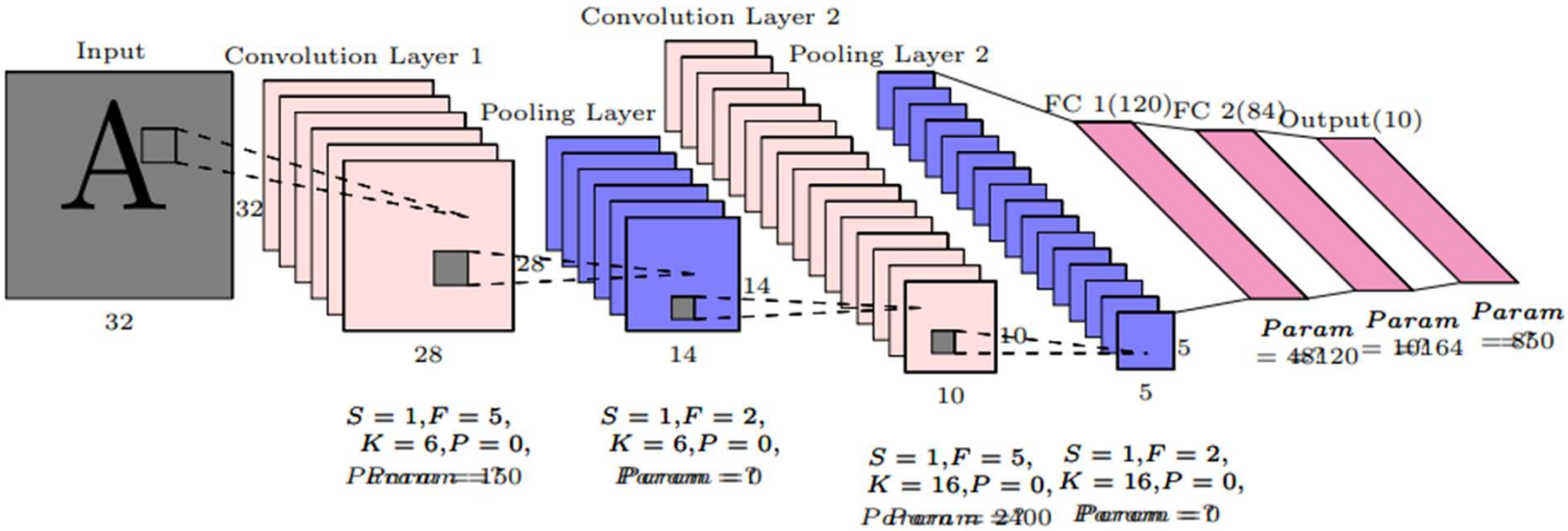


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

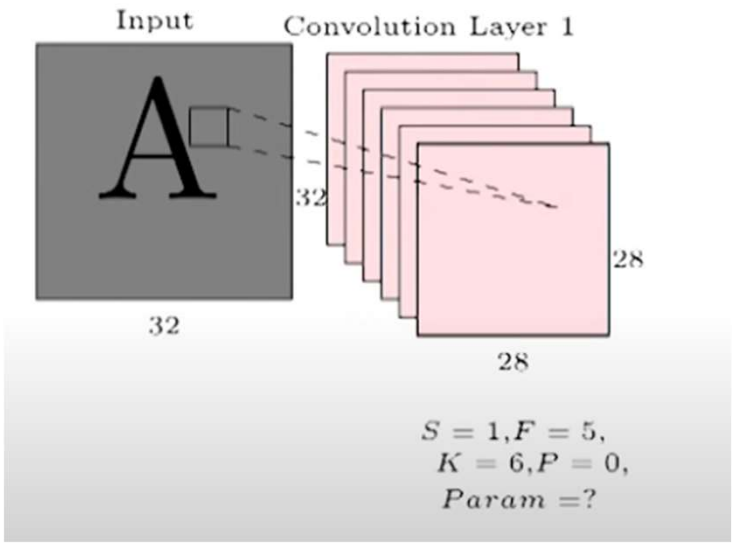
[LeCun et al., 1998]

LeNet-5



LeNet-5 consists of **seven layers**, with the layers responsible for learning hierarchical features of the input image

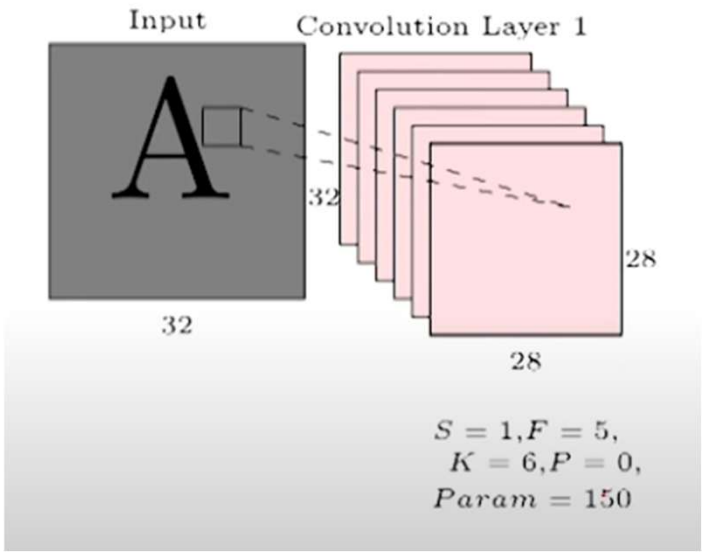
LeNet-5



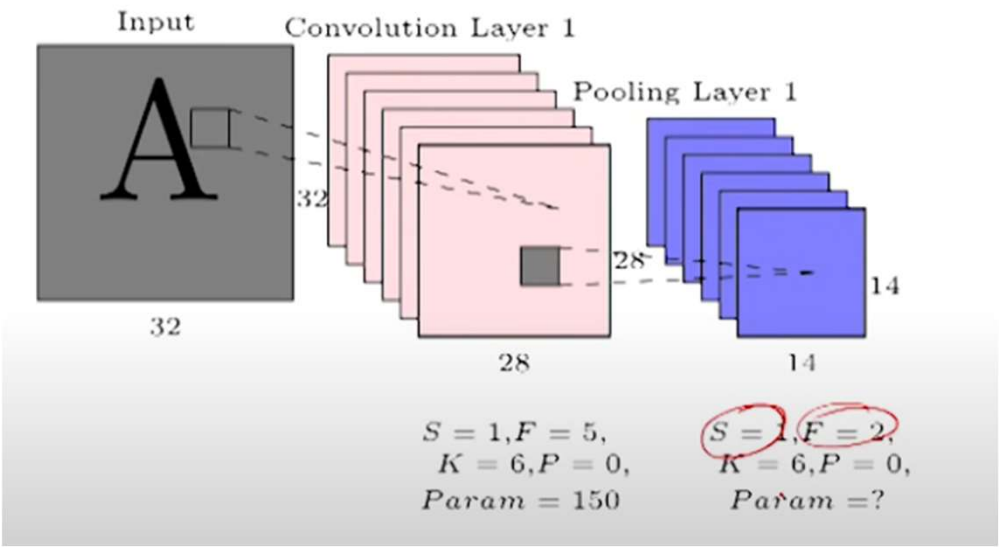
$$\frac{W_1 - F + 2P}{S} + 1$$

The number of filters K
The spatial extent F of each filter
S stride
P padding

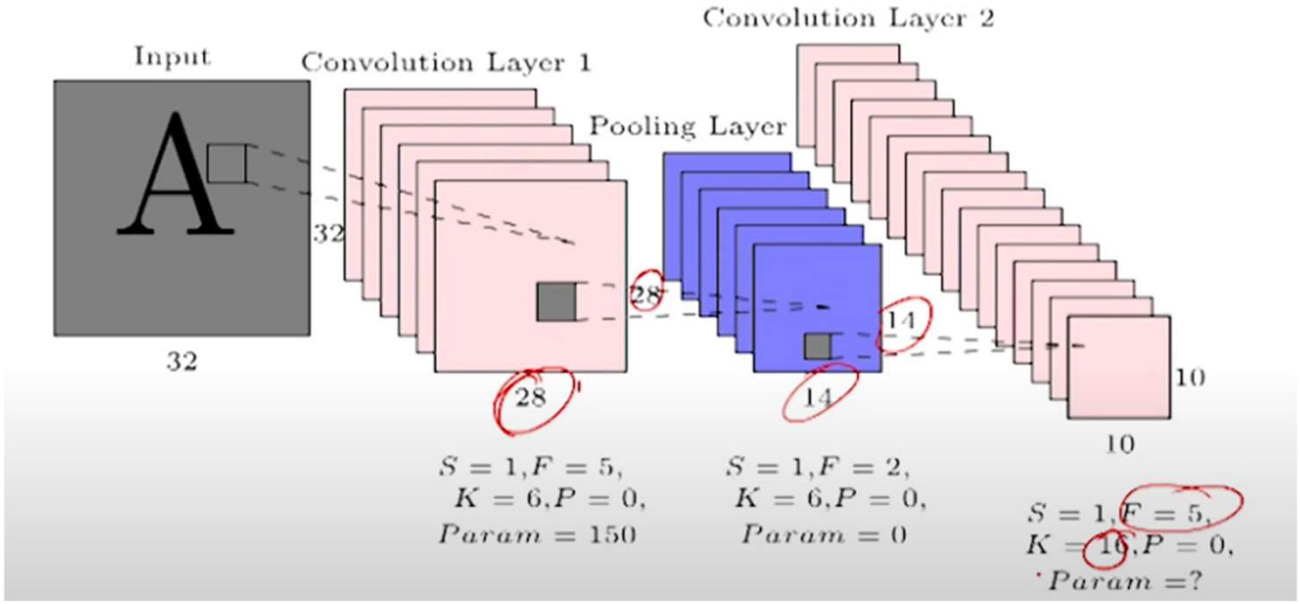
LeNet-5



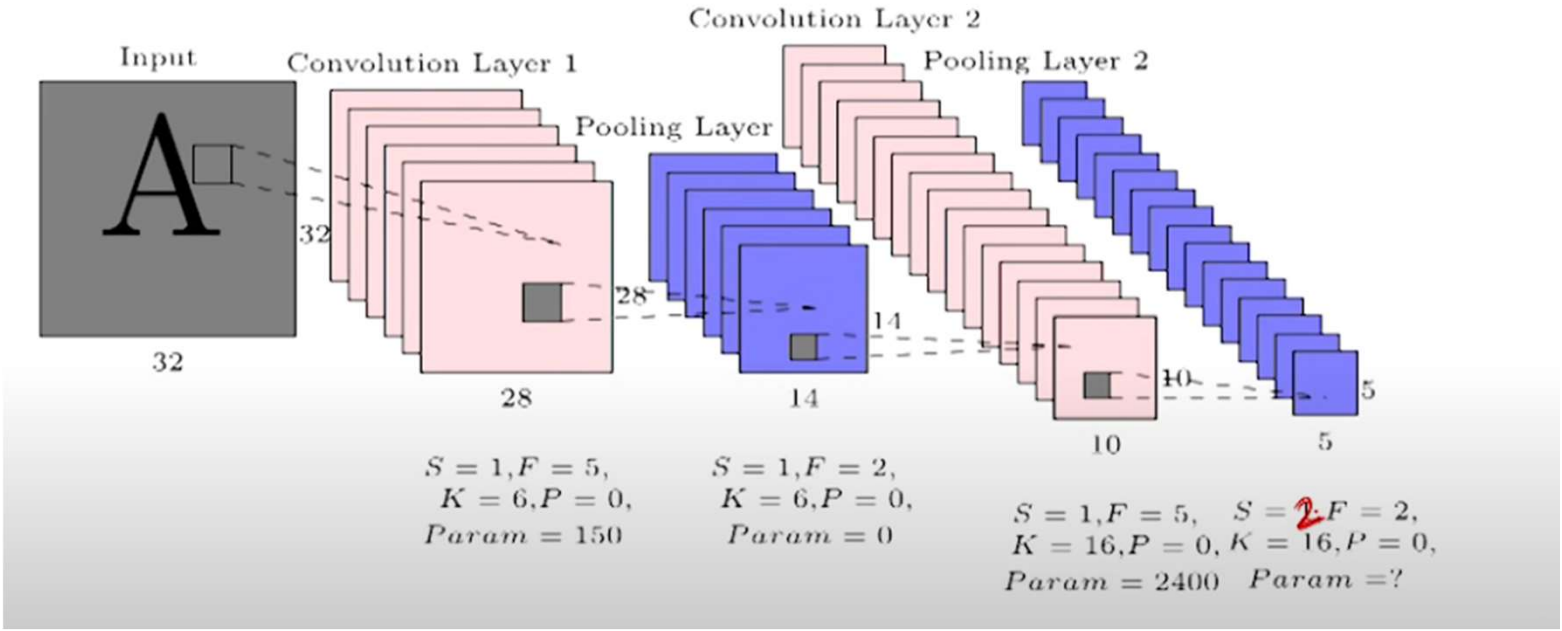
LeNet-5


Input
Convolution Layer 1
Pooling Layer 1
32
32
28
28
14
14
$S = 1, F = 5,$
$K = 6, P = 0,$
$Param = 150$
$S = 1, F = 2,$
$K = 6, P = 0,$
$Param = ?$

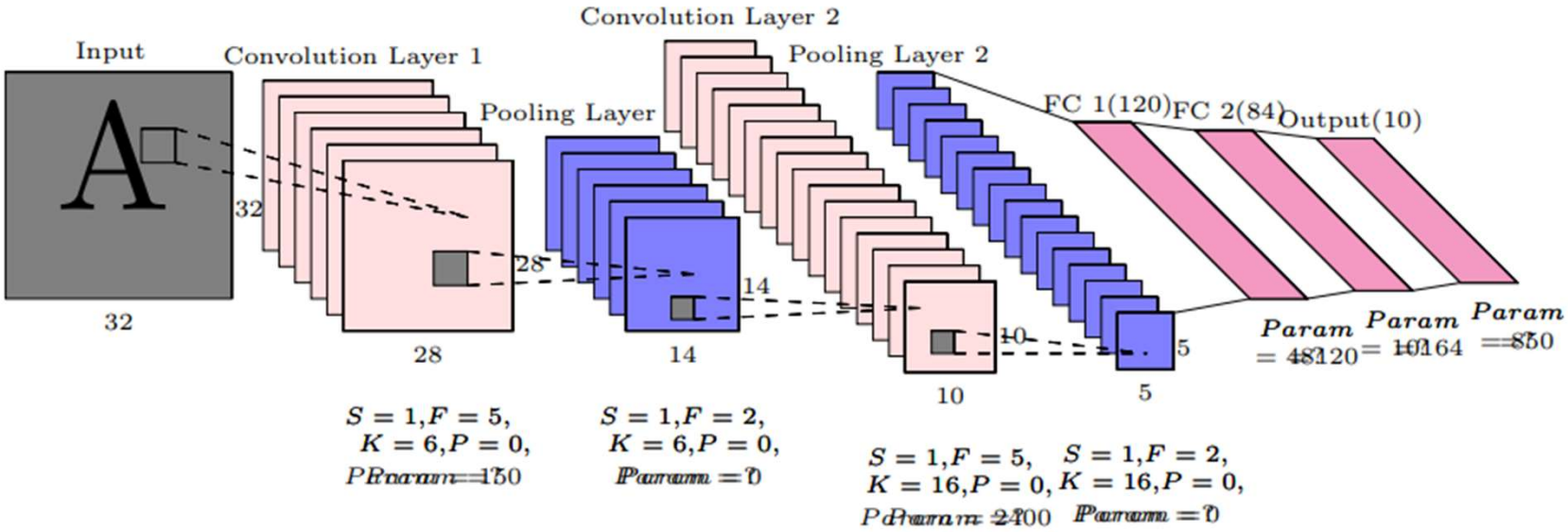
LeNet-5


Input
Convolution Layer 1
Pooling Layer
Convolution Layer 2
A
32
32
28
28
14
14
10
10
$S = 1, F = 5,$
$K = 6, P = 0,$
$Param = 150$
$S = 1, F = 2,$
$K = 6, P = 0,$
$Param = 0$
$S = 1, F = 5,$
$K = 16, P = 0,$
$Param = ?$

LeNet-5


Input
Convolution Layer 1
Pooling Layer
Convolution Layer 2
Pooling Layer 2
32
32
28
28
14
14
10
10
5
5
$S = 1, F = 5,$
$K = 6, P = 0,$
$Param = 150$
$S = 1, F = 2,$
$K = 6, P = 0,$
$Param = 0$
$S = 1, F = 5,$
$K = 16, P = 0,$
$Param = 2400$
$S = 2, F = 2,$
$K = 16, P = 0,$
$Param = ?$

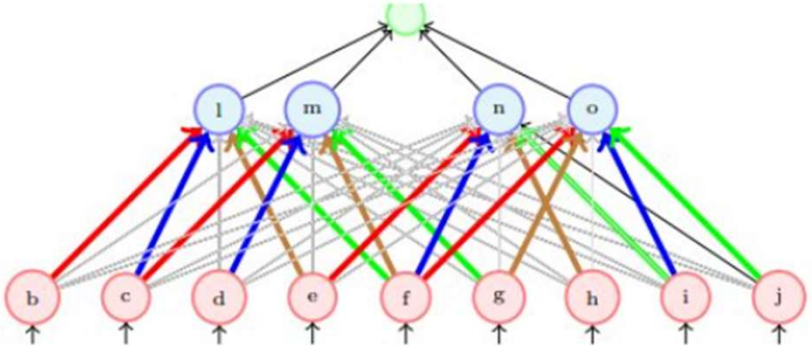
LeNet-5


Input
Convolution Layer 1
Pooling Layer
Convolution Layer 2
Pooling Layer 2
FC 1(120)
FC 2(84)
Output(10)
32
32
28
28
14
14
10
10
5
5
$S = 1, F = 5,$
$K = 6, P = 0,$
$S = 1, F = 2,$
$K = 6, P = 0,$
$S = 1, F = 5,$
$K = 16, P = 0,$
$S = 1, F = 2,$
$K = 16, P = 0,$
Param
Param
Param
Param
Param
Param

LeNet-5 Summary

Layer	Type	Number of Filters	Filter Size	Stride	Output Size	Parameters
Input	-	-	-	-	32×32	0
C1	Convolutional	6	5×5	1	$28 \times 28 \times 6$	$6 \times (5 \times 5) = 150$
S2	Average Pooling	-	2×2	2	$14 \times 14 \times 6$	0
C3	Convolutional	16	$5 \times 5 \times 6$	1	$10 \times 10 \times 16$	$16 \times (5 \times 5 \times 6) = 2,400$
S4	Average Pooling	-	2×2	2	$5 \times 5 \times 16$	0
C5	Fully Connected Conv.	120	$5 \times 5 \times 16$	1	$1 \times 1 \times 120$	$120 \times (5 \times 5 \times 16) = 48,000$
F6	Fully Connected	84	-	-	84	$84 \times 120 = 10,080$
Output	Fully Connected	10	-	-	10	$10 \times 84 = 840$

How do we train a CNN?

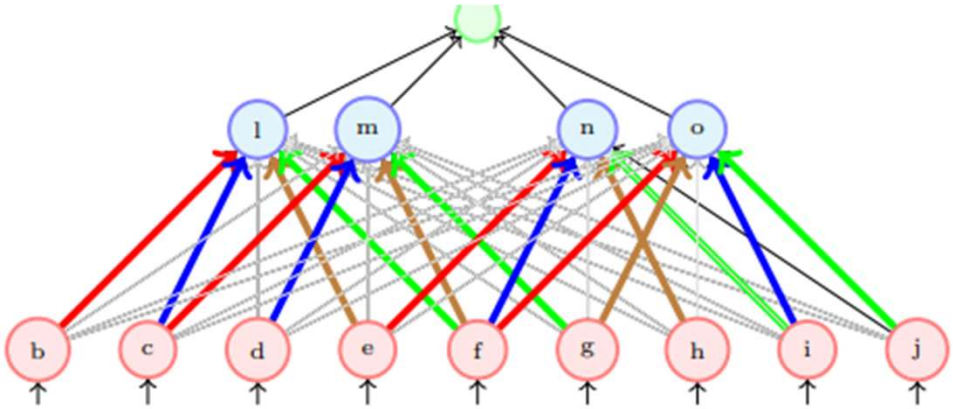


$$\begin{bmatrix} O_{11} & O_{12} \\ O_{21} & O_{22} \end{bmatrix} = \text{Convolution}\left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \right)$$

$$\begin{bmatrix} \partial E/\partial F_{11} & \partial E/\partial F_{12} \\ \partial E/\partial F_{21} & \partial E/\partial F_{22} \end{bmatrix} = \text{Convolution}\left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} \partial E/\partial O_{11} & \partial E/\partial O_{12} \\ \partial E/\partial O_{21} & \partial E/\partial O_{22} \end{bmatrix} \right)$$

Slide taken from Forward And Backpropagation in Convolutional Neural Network. - Medium

How do we train a CNN?



Backpropagation

- **In PyTorch**: The nn.Conv2d layer performs 2D convolutions.
- **In TensorFlow/Keras**: The Conv2D layer performs 2D convolutions.

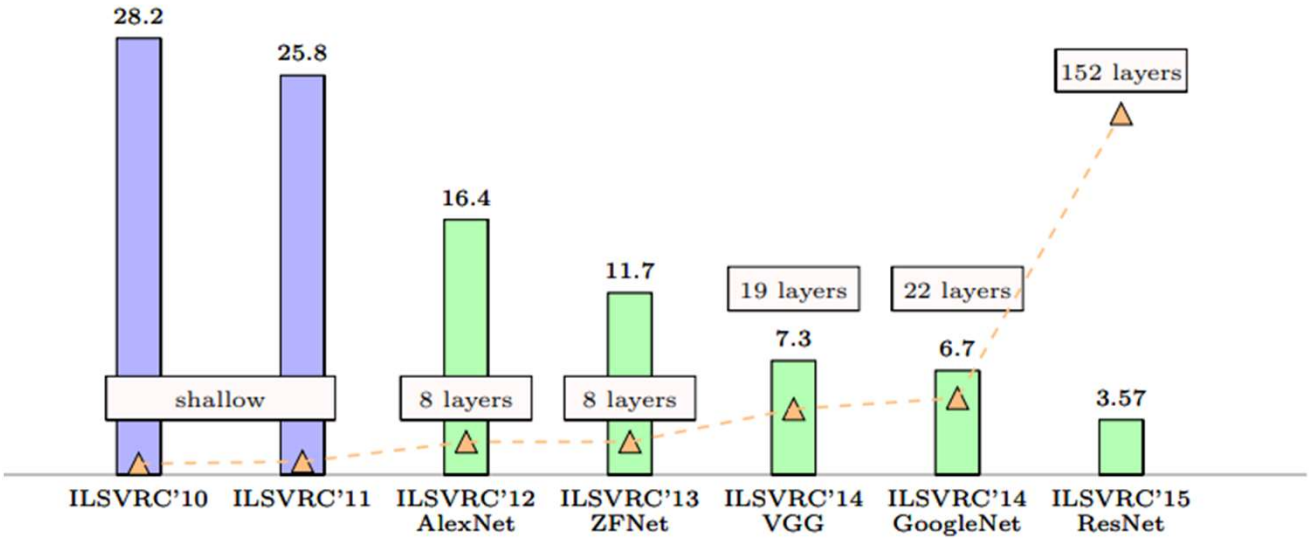
Packages like **PyTorch**, **TensorFlow**, and **Keras** simplify the process of convolution, gradient computation, and weight updates. They provide high-level abstractions, automatic differentiation, optimized performance, and scalability, allowing you to focus on model design and experimentation

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

- The annual “Olympics” of computer vision.
- Teams from across the world compete to see who has the best computer vision model for tasks such as classification, localization, detection, and more.
- **2012** marked **the first year where a CNN was used** to achieve a top 5 test error rate of 15.3%.
- The next best entry achieved an error of 26.2%.

•**ImageNet** is a database of millions of labeled images
• created by **Stanford's Vision Lab** under the leadership of **Fei-Fei Li** in 2009.
•ImageNet contains over 14 million labeled images.
•**Categories**: The dataset is organized into more than 20,000 categories .

ImageNet Solutions



- **Size**: Over 14 million images across more than 20,000 categories.
- **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)**: An annual competition that ran from 2010 to 2017, where researchers competed to create models that achieved the highest accuracy on the dataset. This challenge significantly advanced the development of deep learning architectures.

AlexNet

Architecture
CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

- Input: 227x227x3 images (224x224 before padding)
- First layer: 96 11x11 filters applied at stride 4

- **Output volume size?**

  (N-F)/s+1 = (227-11)/4+1 = 55 -> [55x55x96]

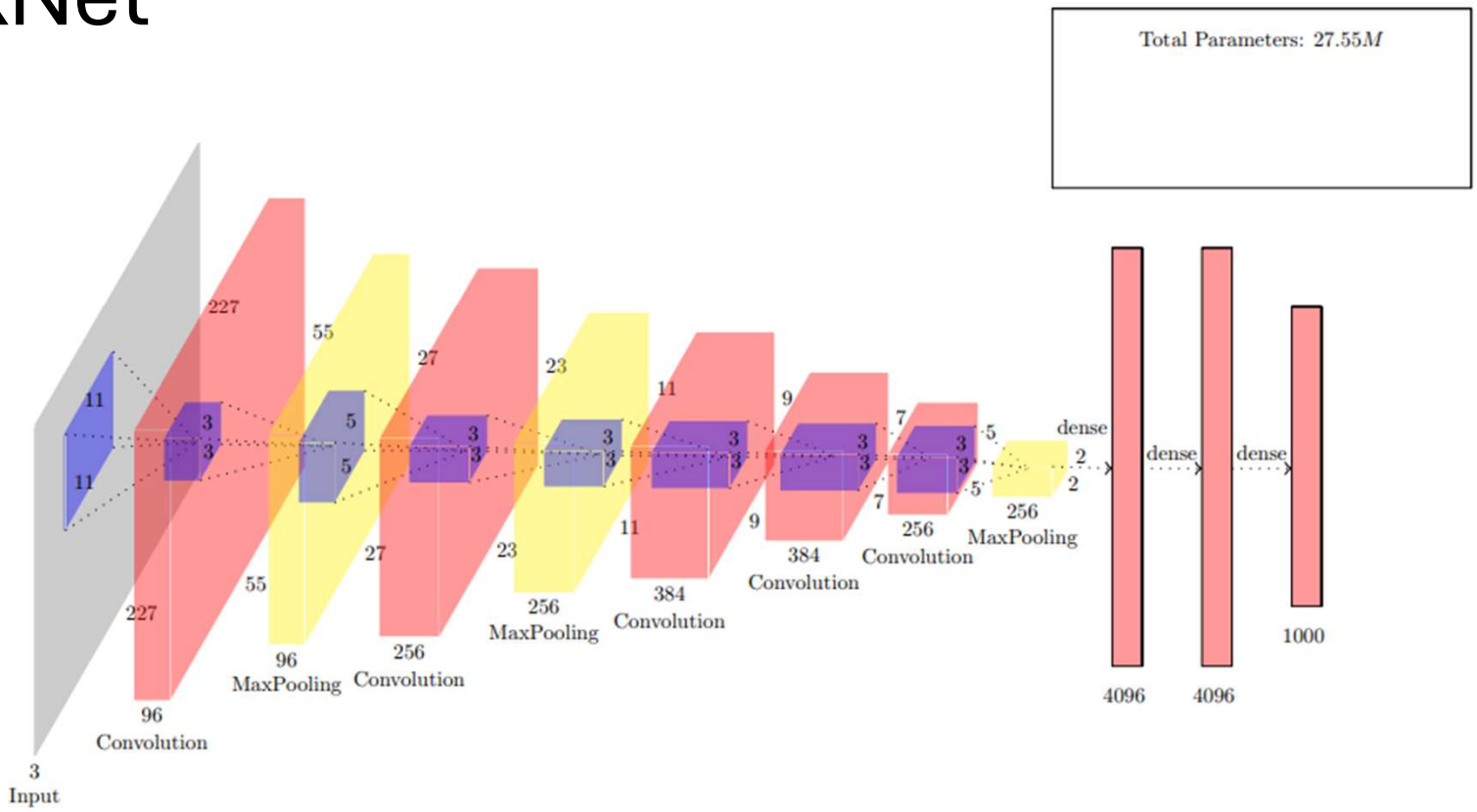
- **Number of parameters in this layer?**

  (11*11*3)*96 = 35K

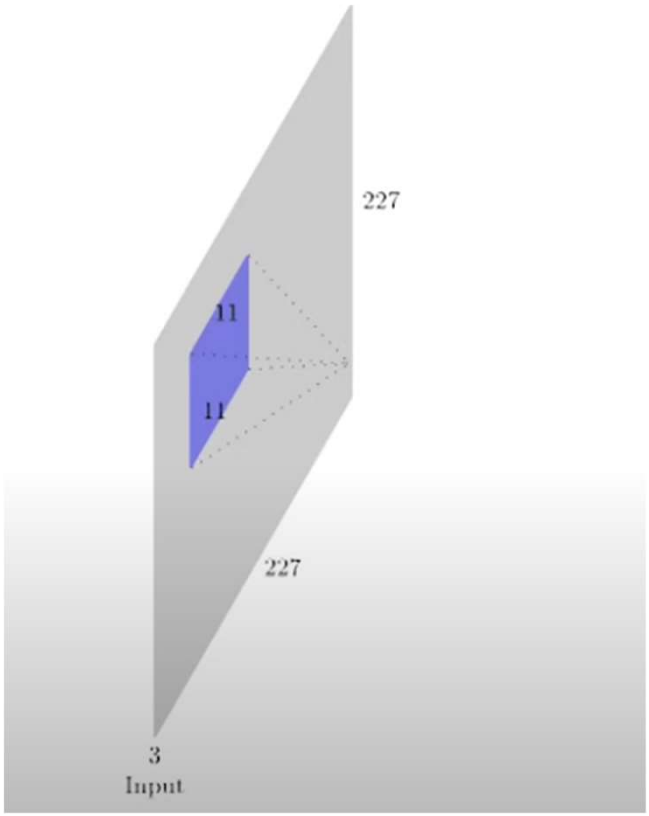
Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Krizhevsky et al., 2012]

AlexNet


Total Parameters: 27.55M
3
Input
227
11
11
96
Convolution
55
5
5
96
MaxPooling
27
3
3
256
Convolution
23
3
3
256
MaxPooling
11
3
3
384
Convolution
9
3
3
384
Convolution
7
3
3
256
Convolution
5
3
3
256
MaxPooling
dense
2
2
4096
dense
4096
dense
1000

AlexNet



Input: $227 \times 227 \times 3$

Conv1: $K = 96, F = 11$

$S = 4, P = 0$

Output: $W_2 = ?,\ H_2 = ?$

Parameters: ?

AlexNet

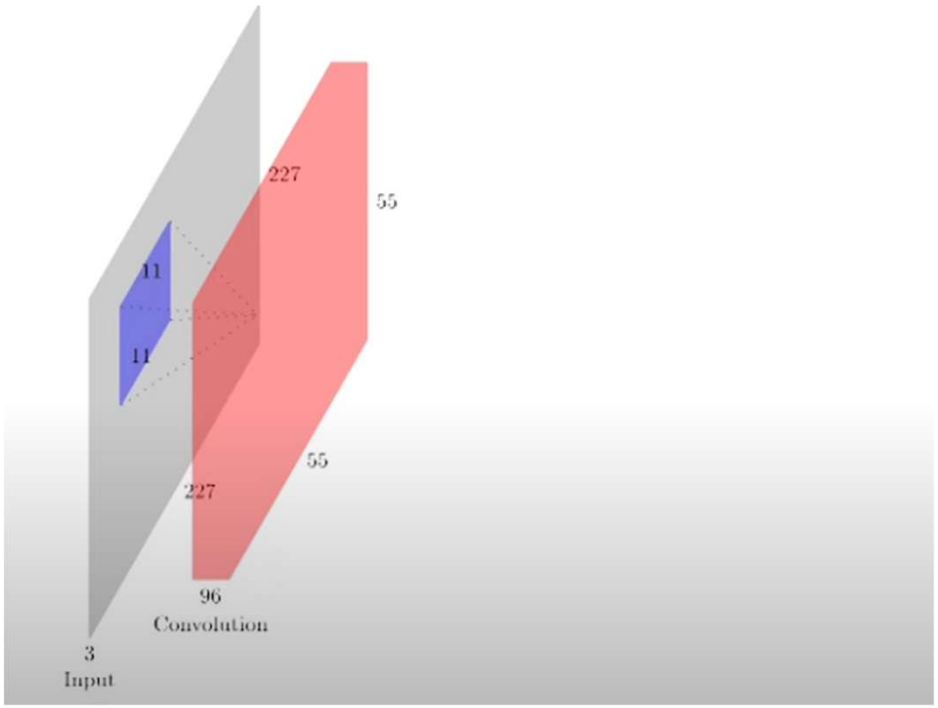
Input: $227 \times 227 \times 3$

Conv1: $K = 96, F = 11$

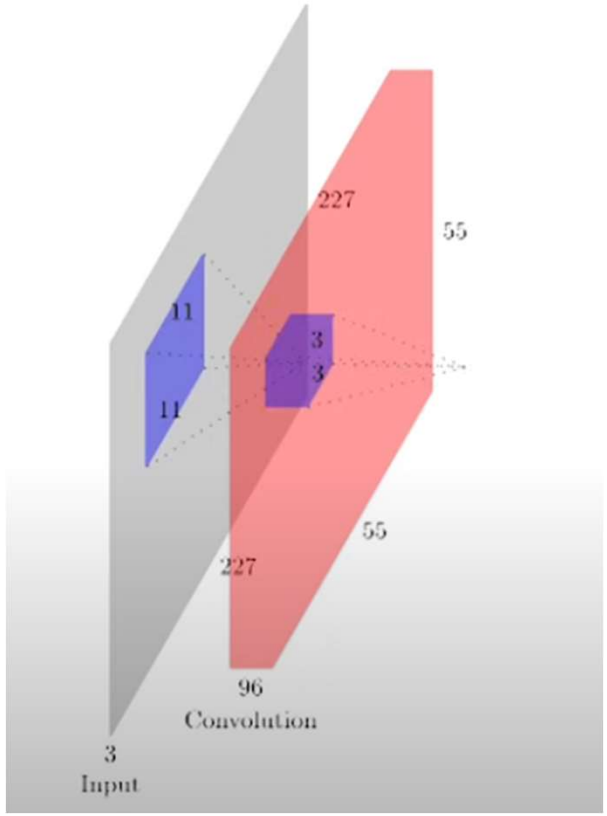
$S = 4, P = 0$

Output: $W_2 = 55,\ H_2 = 55$

Parameters: $(11 \times 11 \times 3) \times 96 = 34K$



AlexNet



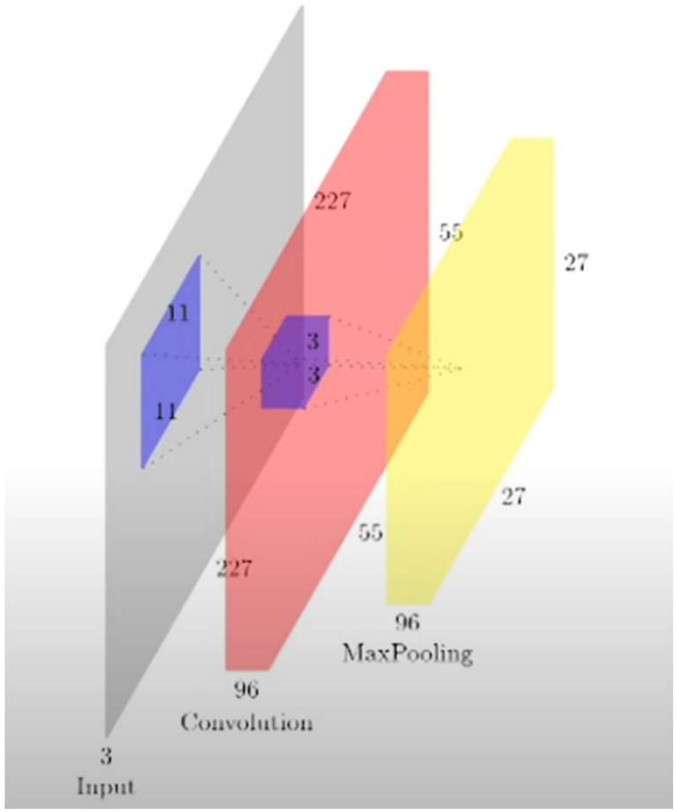
Max Pool Input: $55 \times 55 \times 96$

$F = 3, S = 2$

Output: W_2 −?, H_2 −?

Parameters: ?

AlexNet



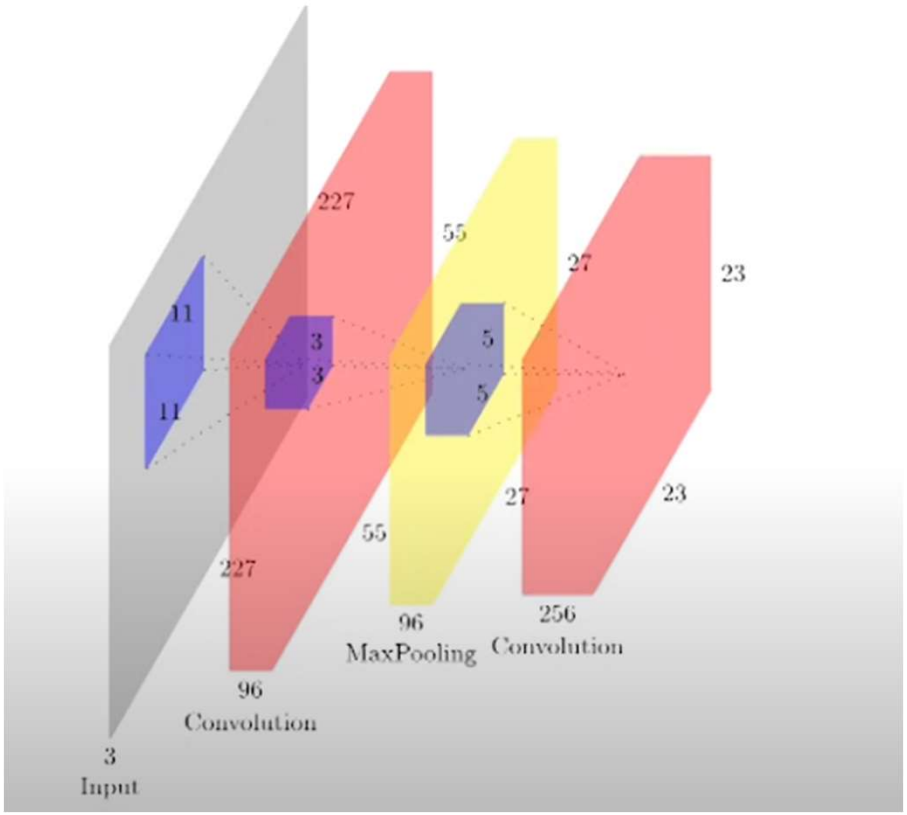
Max Pool Input: $55 \times 55 \times 96$

$F = 3, S = 2$

Output: $W_2 = 27,\ H_2 = 27$

Parameters: 0

AlexNet



Input: $27 \times 27 \times 96$

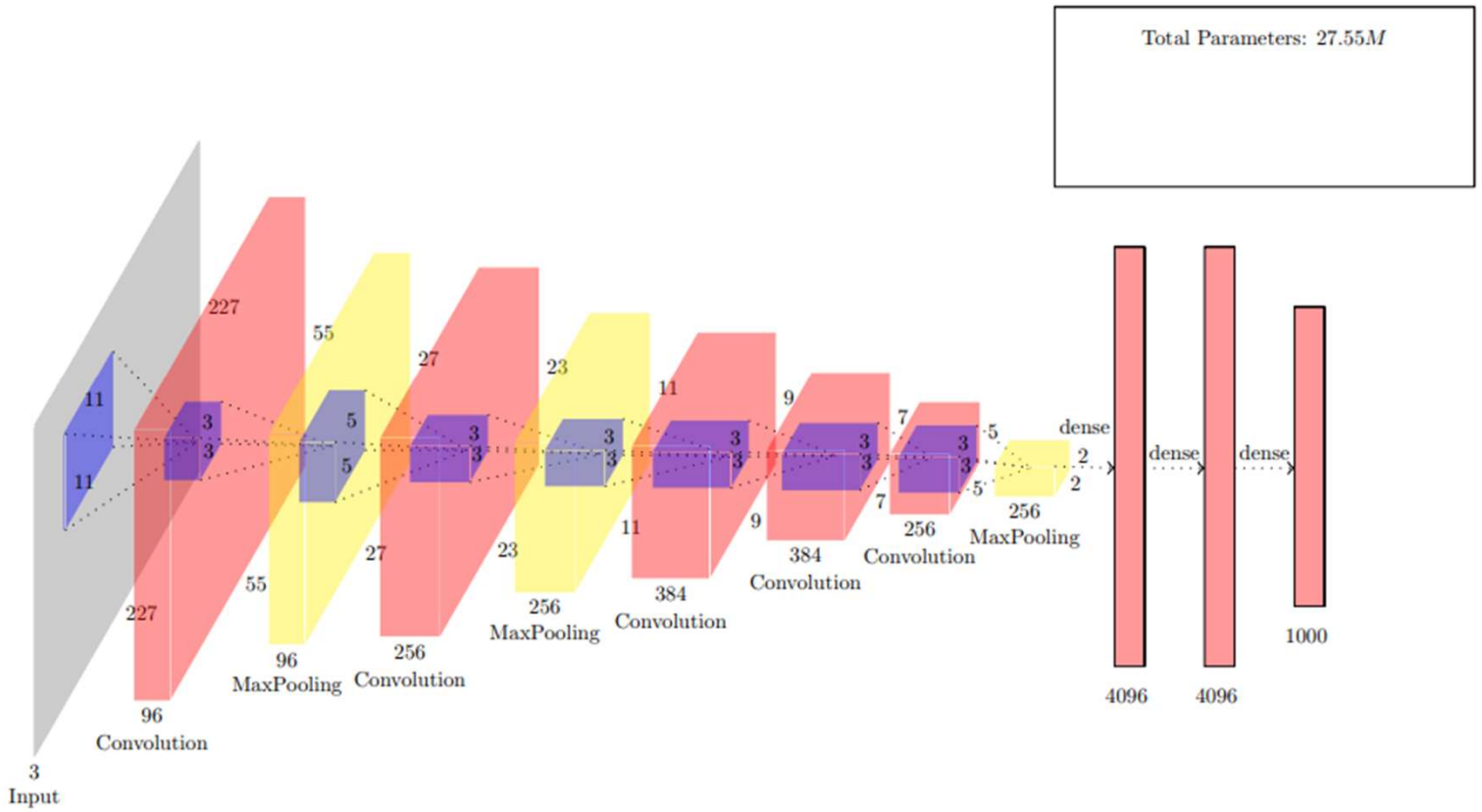
Conv1: $K = 256, F = 5$

$S = 1, P = 0$

Output: $W_2 = 23,\ H_2 = 23$

Parameters: $(5 \times 5 \times 96) \times 256 = 0.6M$

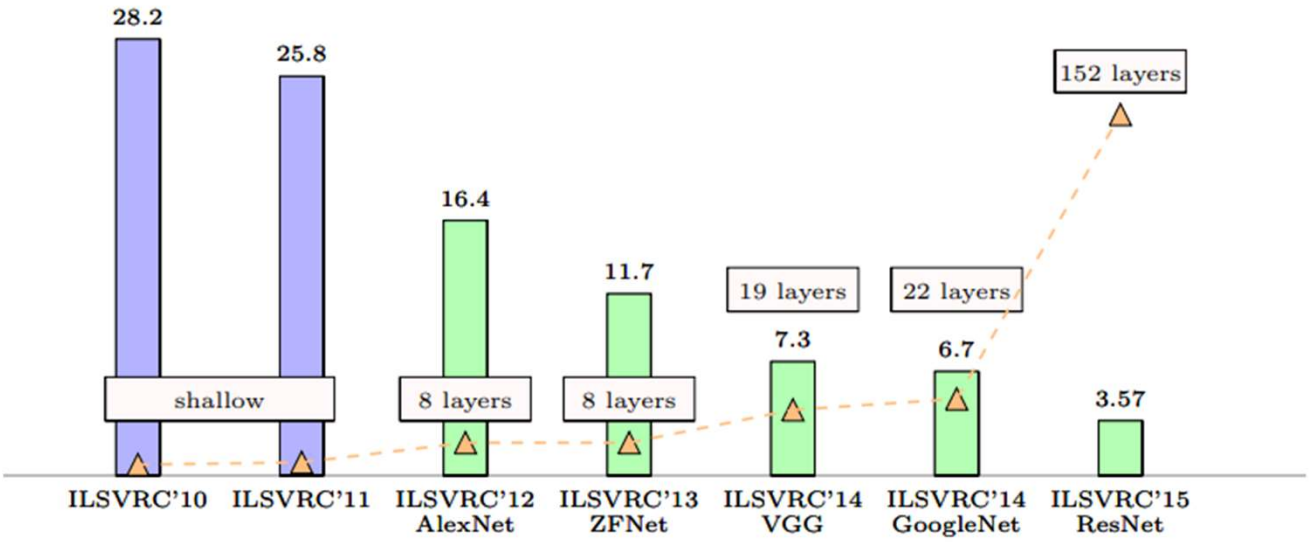
AlexNet


Total Parameters: 27.55M
3
Input
227
11
11
96
Convolution
55
5
5
96
MaxPooling
27
3
3
256
Convolution
23
3
3
256
MaxPooling
11
3
3
384
Convolution
9
3
3
384
Convolution
7
3
3
256
Convolution
5
3
3
256
MaxPooling
dense
2
2
4096
dense
4096
dense
1000

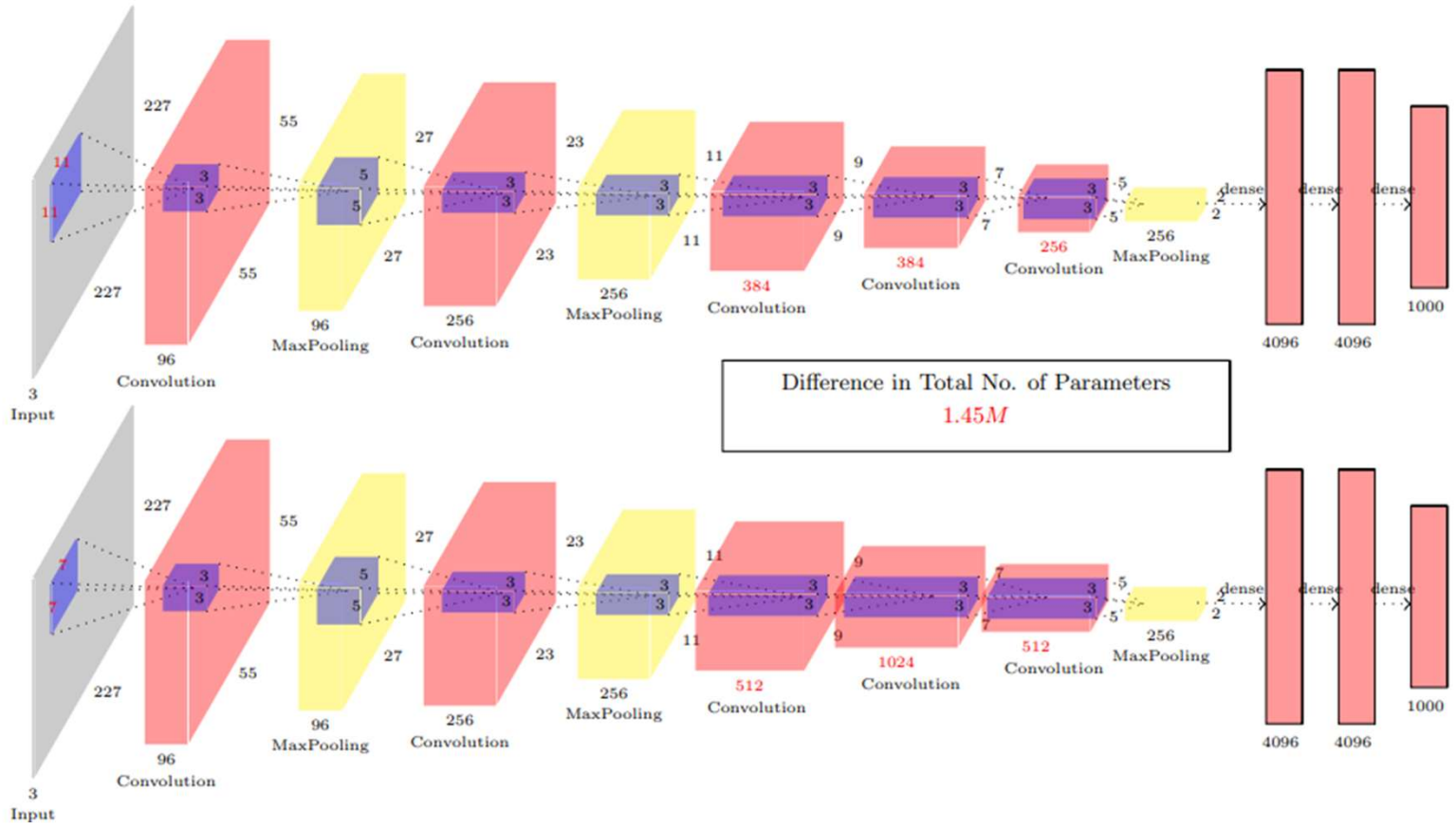
AlexNet Summary

Layer	Output Size	#Filters	F Size	Stride	Padding	Parameters
Input Layer	224 x 224 x 3	N/A	N/A	N/A	N/A	N/A
Conv1	55 x 55 x 96	96	11 x 11	4	0	34,944
Max Pool 1	27 x 27 x 96	N/A	3 x 3	2	0	0
Conv2	27 x 27 x 256	256	5 x 5	1	2	614,656
Max Pool 2	13 x 13 x 256	N/A	3 x 3	2	0	0
Conv3	13 x 13 x 384	384	3 x 3	1	1	884,736
Conv4	13 x 13 x 384	384	3 x 3	1	1	1,327,488
Conv5	13 x 13 x 256	256	3 x 3	1	1	884,992
Max Pool 3	6 x 6 x 256	N/A	3 x 3	2	0	0
Flatten	1 x 1 x 9216	N/A	N/A	N/A	N/A	0
FC1	1 x 1 x 4096	4096	N/A	N/A	N/A	37,752,832
FC2	1 x 1 x 4096	4096	N/A	N/A	N/A	16,781,312
FC3 (Output)	1 x 1 x 1000	1000	N/A	N/A	N/A	4,096,000

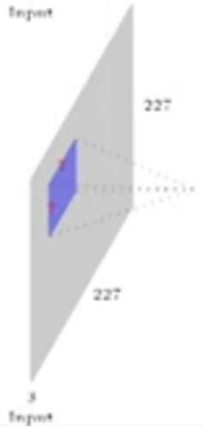
ImageNet Solutions



ZFNet


3
Input
227
11
11
227
96
Convolution
55
55
96
MaxPooling
27
27
256
Convolution
23
23
256
MaxPooling
11
11
384
Convolution
9
9
384
Convolution
7
7
256
Convolution
5
5
256
MaxPooling
2
2
dense
dense
dense
4096
4096
1000
Difference in Total No. of Parameters
1.45M
3
Input
227
7
7
227
96
Convolution
55
55
96
MaxPooling
27
27
256
Convolution
23
23
256
MaxPooling
11
11
512
Convolution
9
9
1024
Convolution
7
7
512
Convolution
5
5
256
MaxPooling
2
2
dense
dense
dense
4096
4096
1000

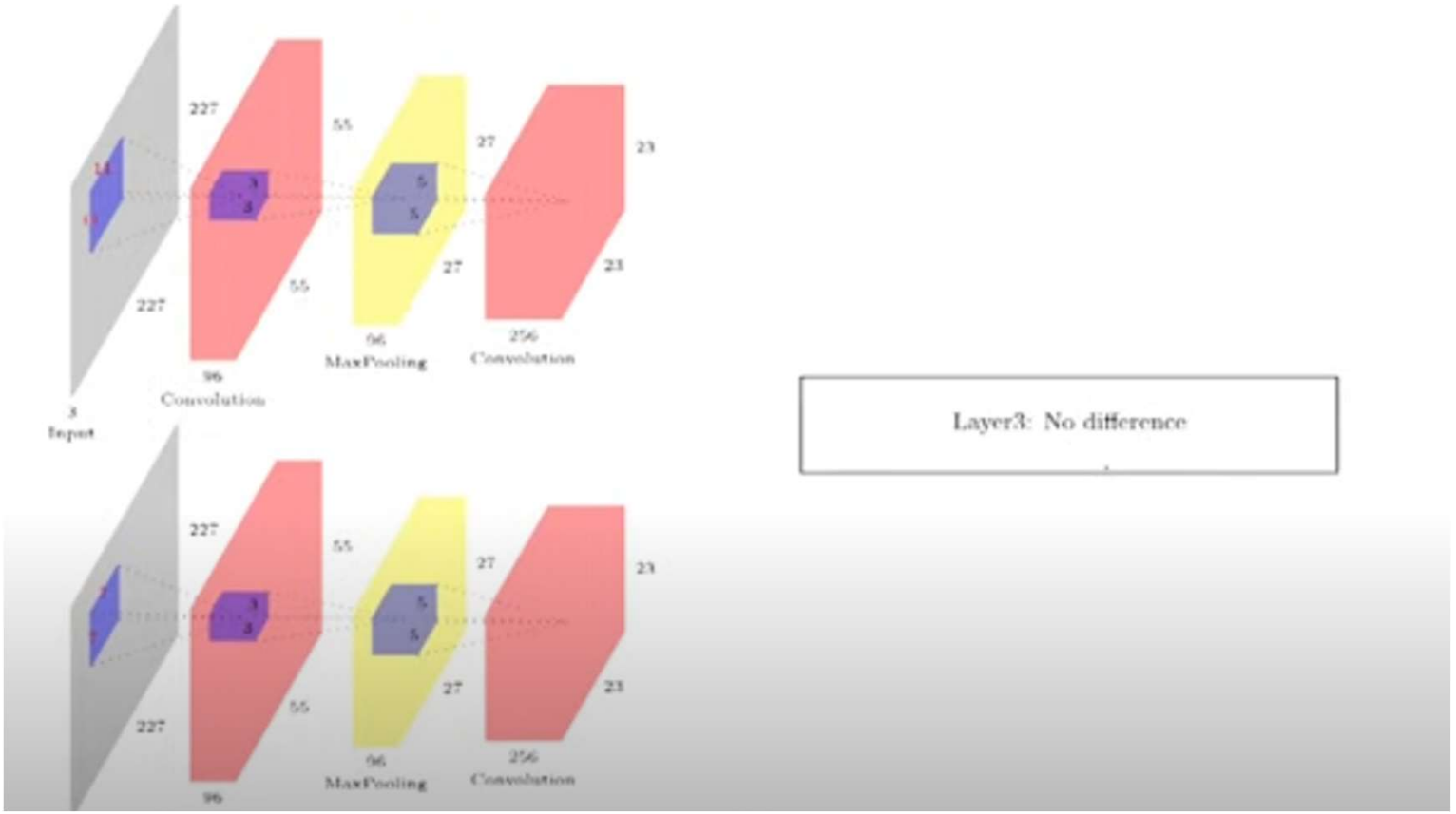
ZFNET vs AlexNet



Layer1: $F = 11 \rightarrow 7$
Difference in Parameters
$((11^2 - 7^2) \times 3) \times 96 = 20.7K$

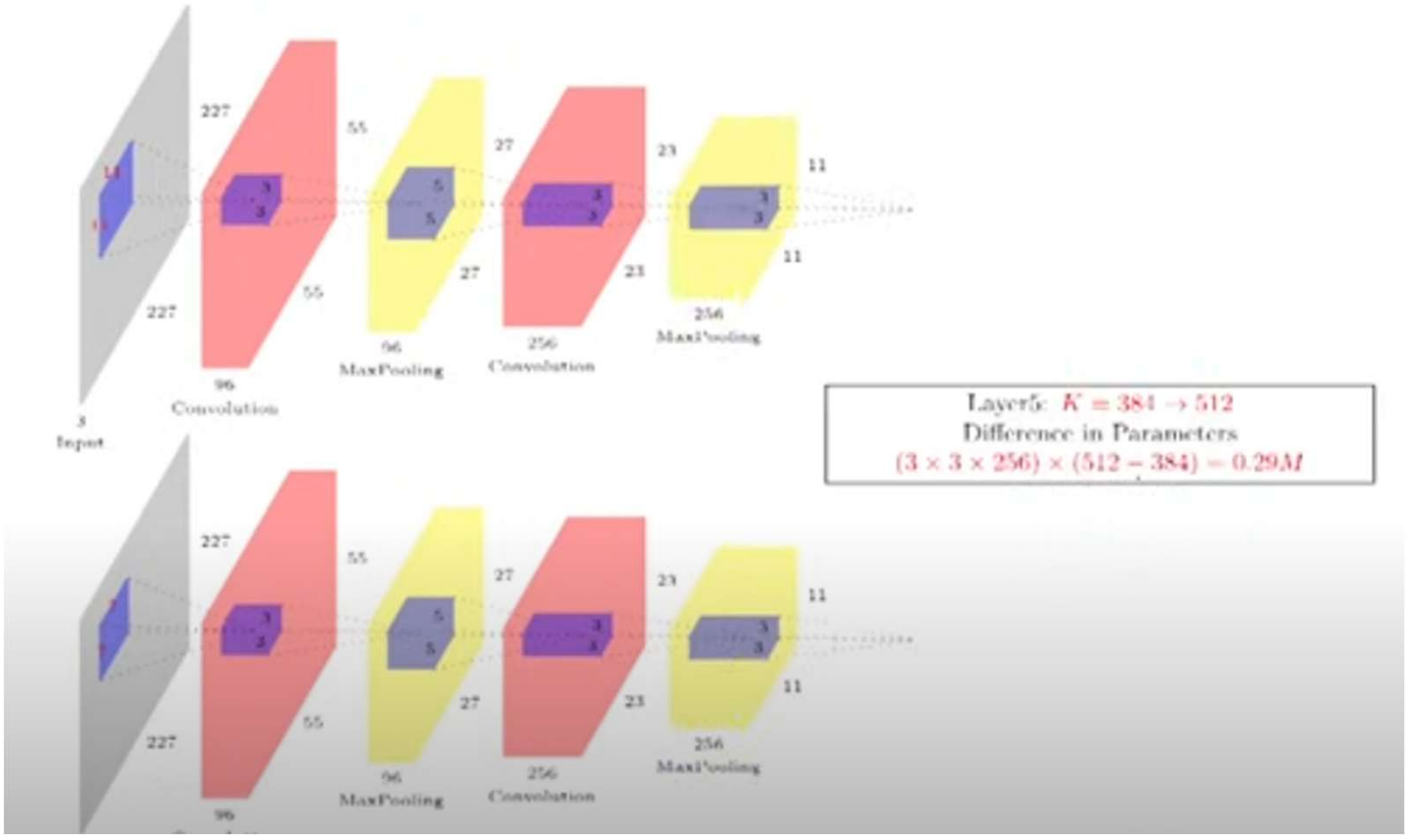
ZFNET vs AlexNet

ZFNET vs AlexNet

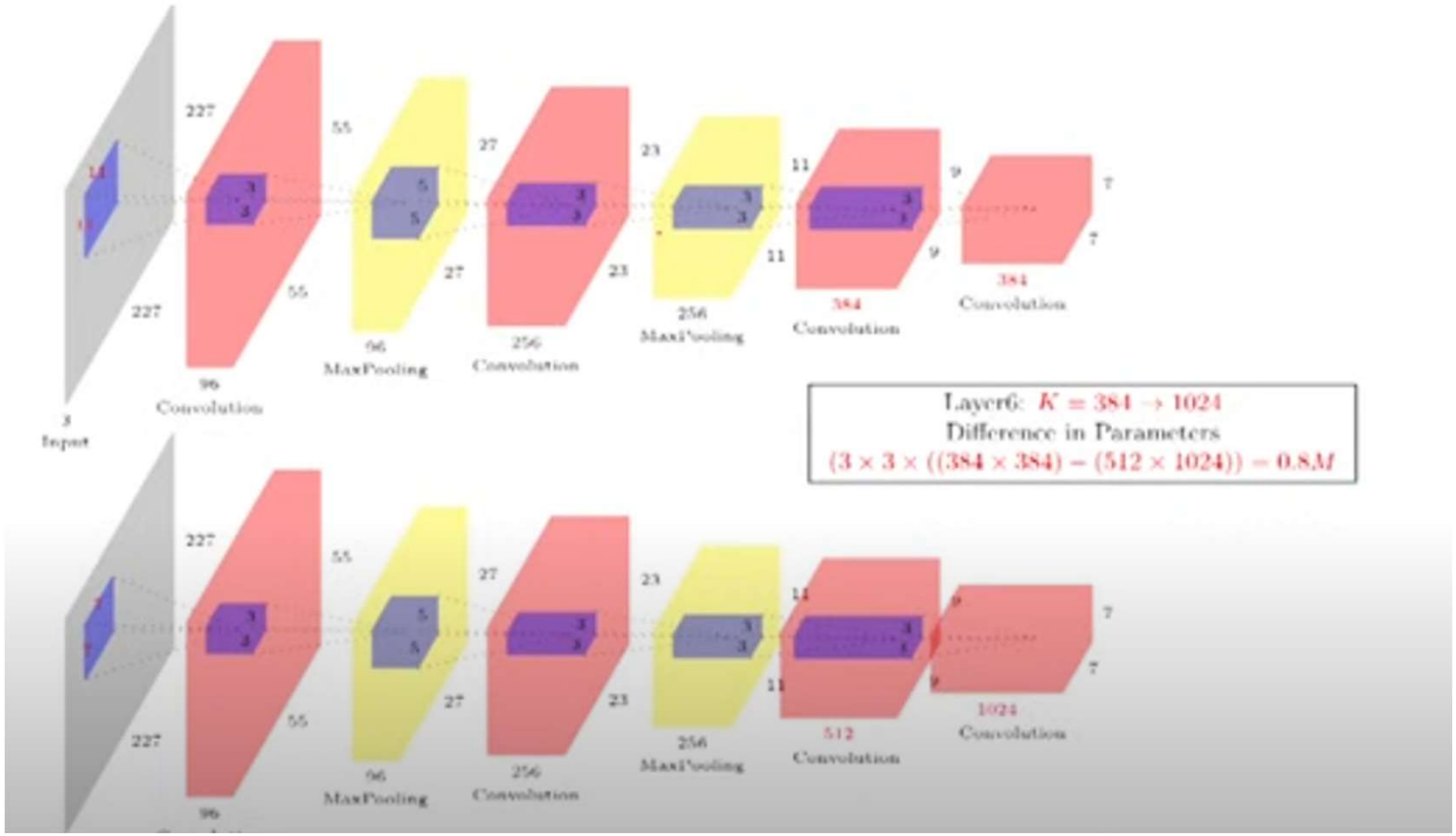


ZFNET vs AlexNet

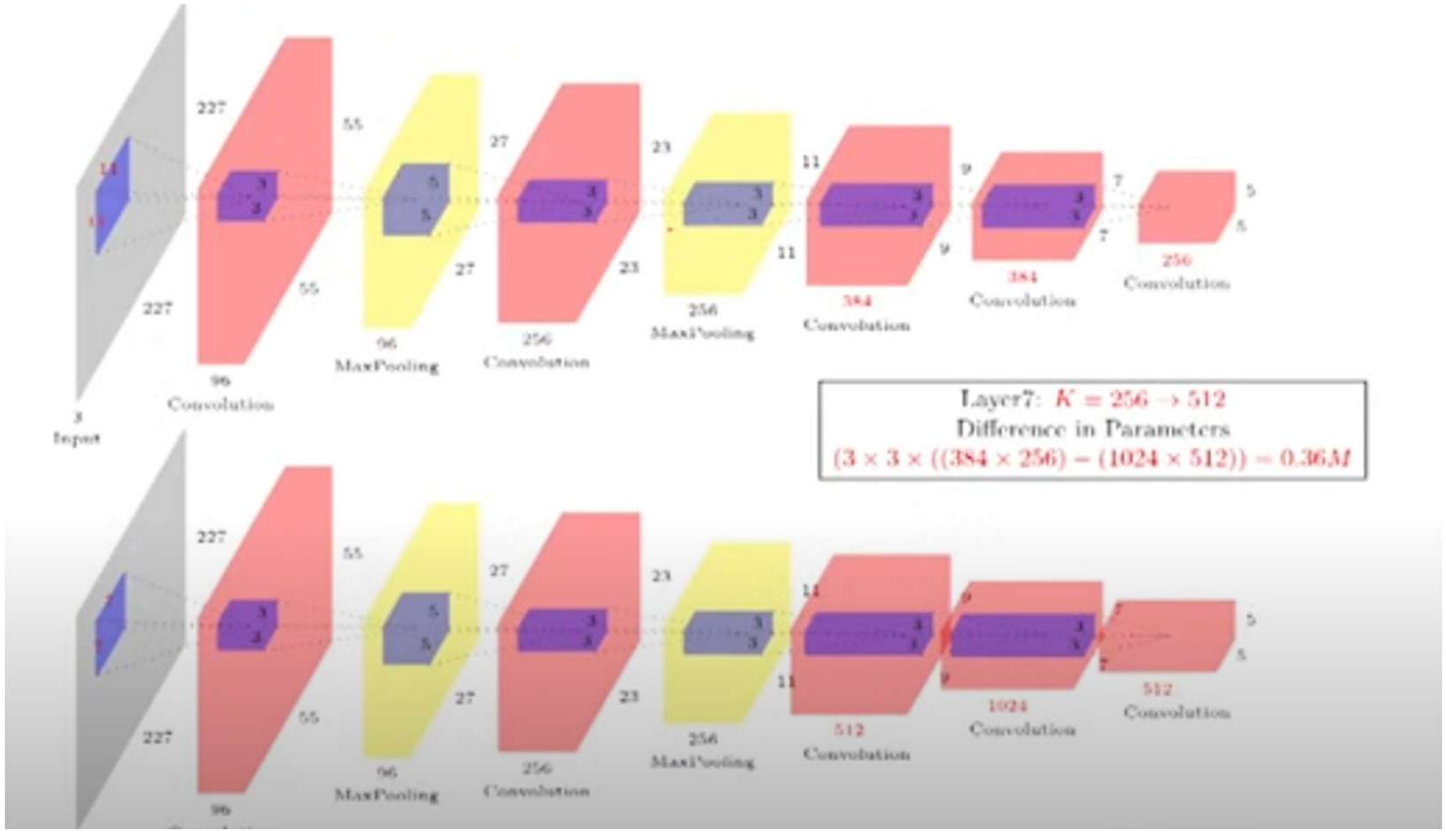
ZFNET vs AlexNet



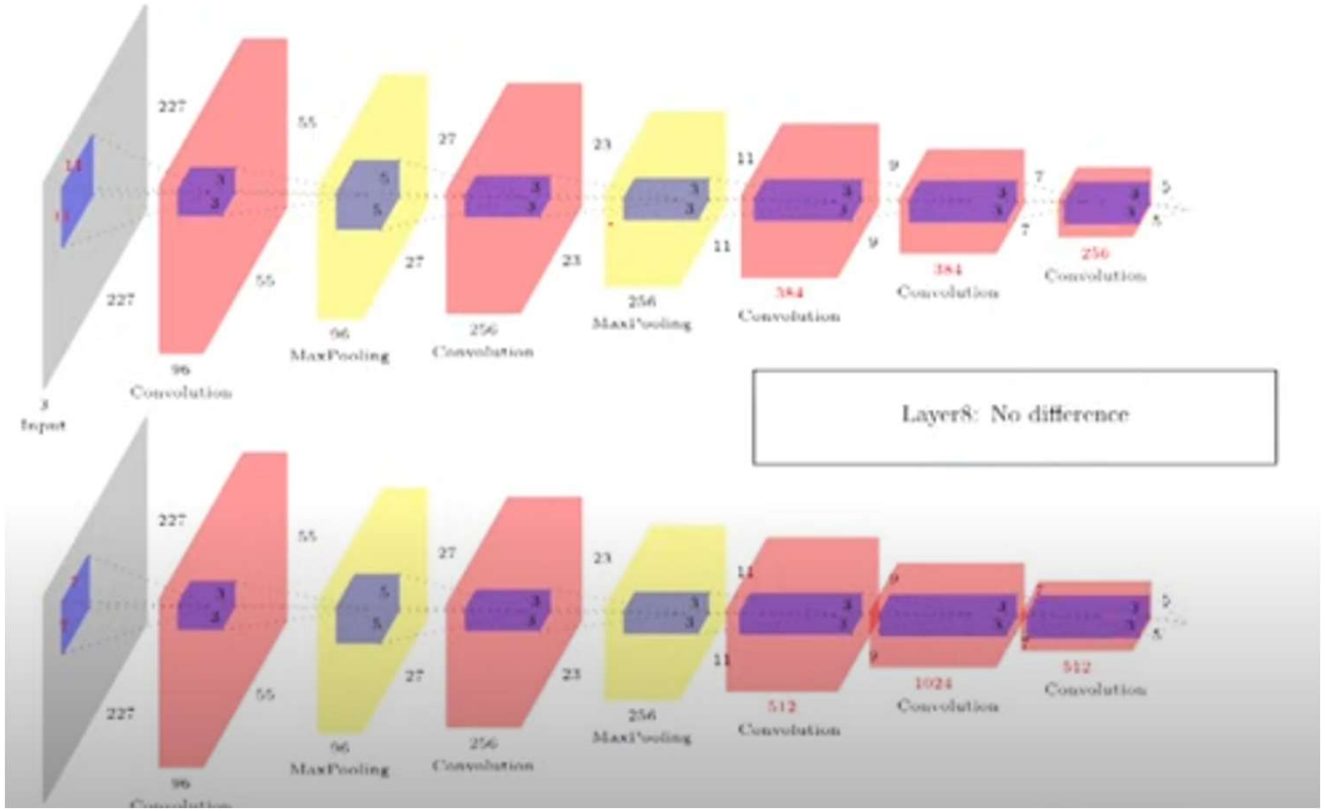
ZFNET vs AlexNet



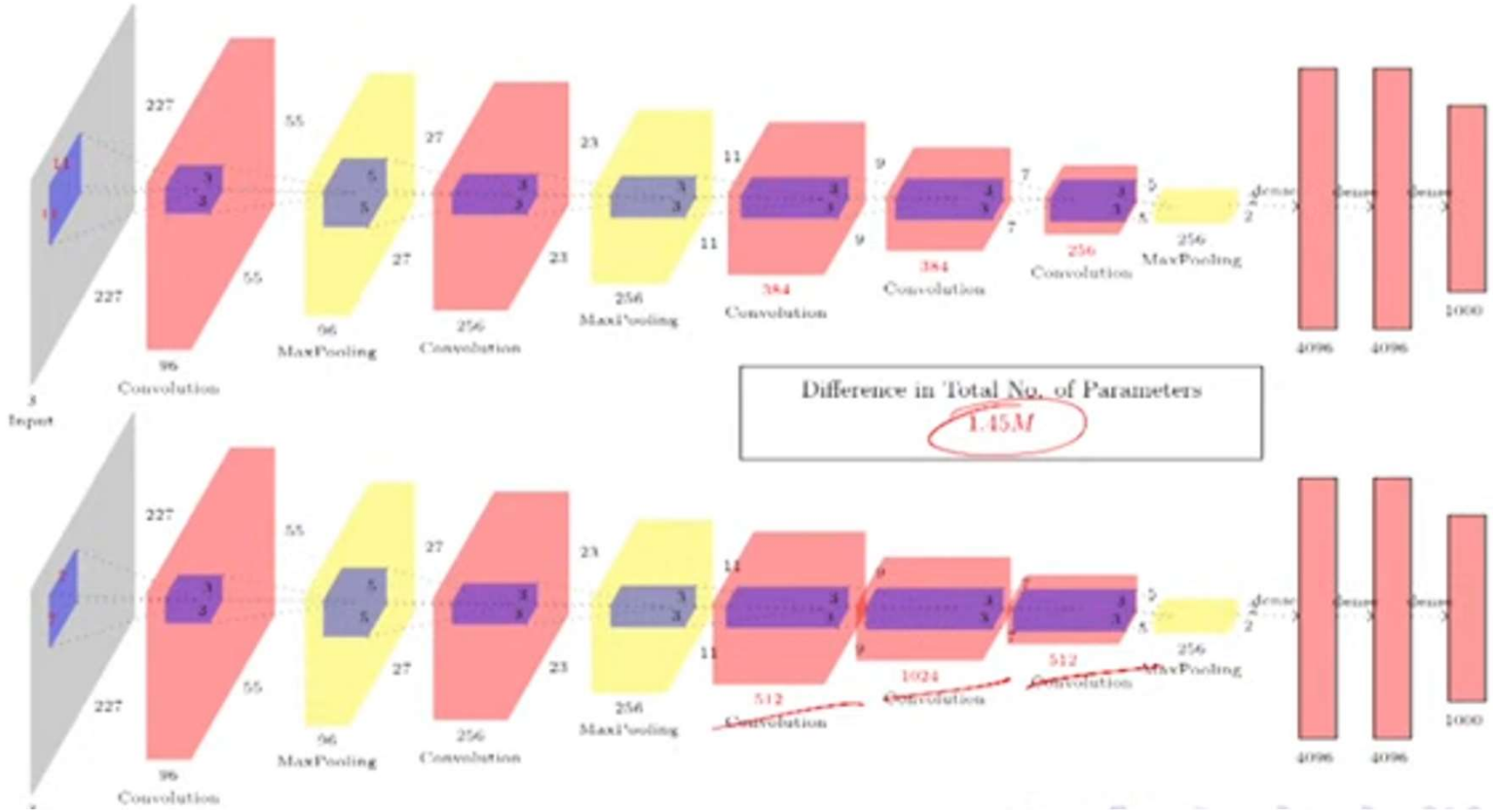
ZFNET vs AlexNet



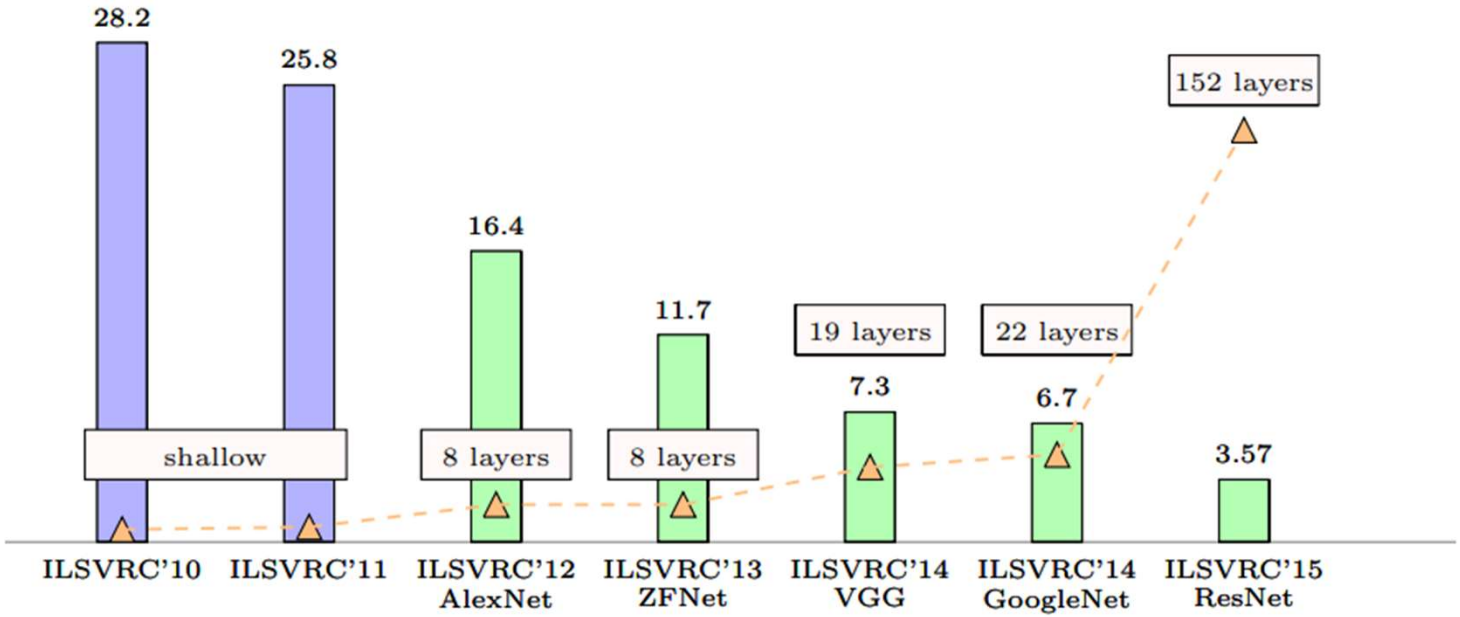
ZFNET vs AlexNet



ZFNET vs AlexNet

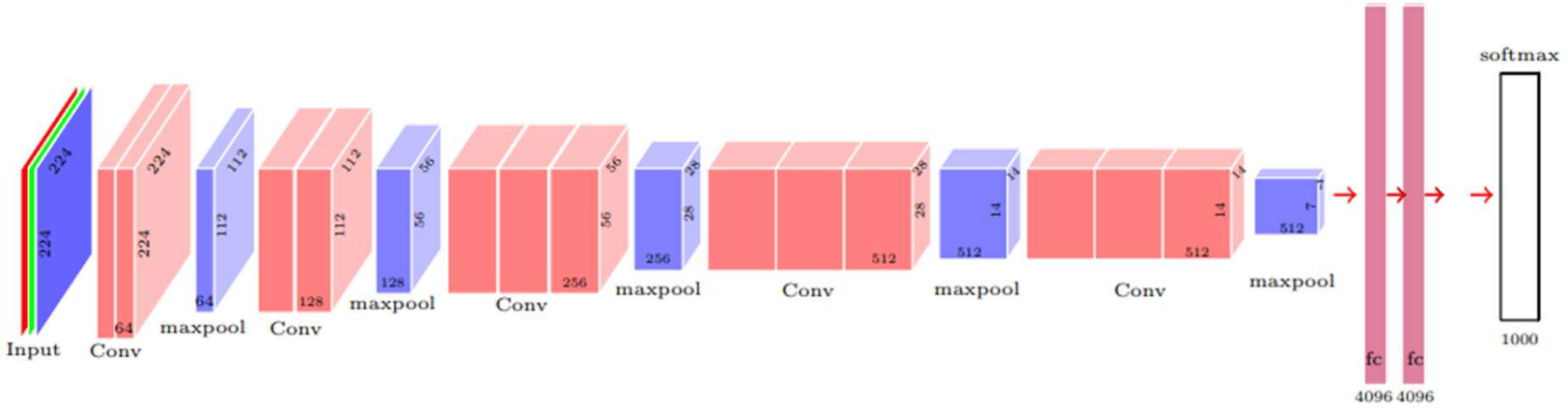


ImageNet Solutions



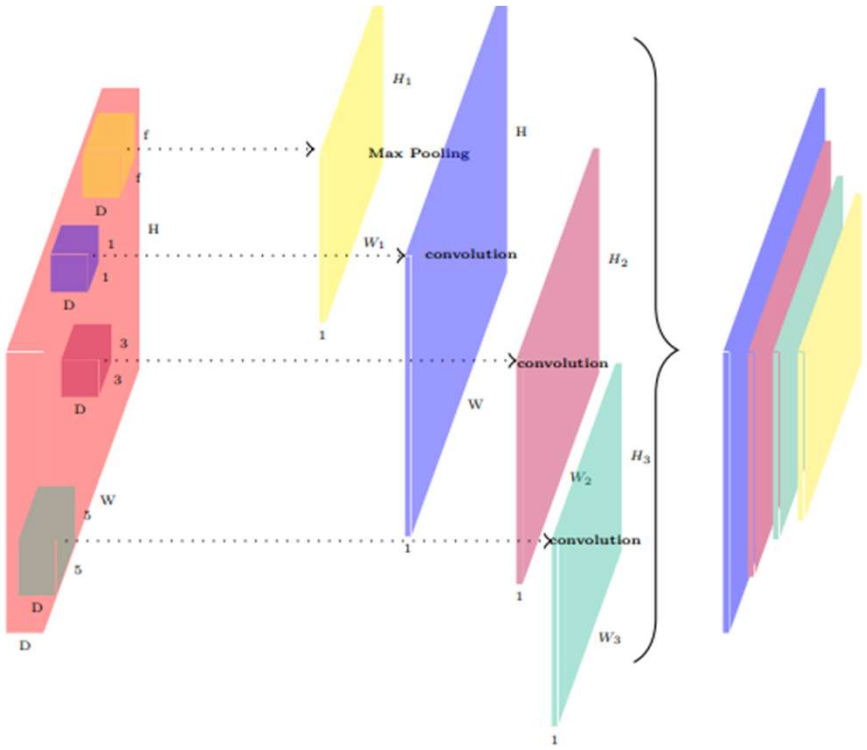
VGGNet

Visual Geometry Group (VGG) at the University of Oxford
16 Layers: VGG-16 has 16 weighted layers (13 convolutional layers and 3 fully connected layers), hence the name "VGG-16.



- Kernel size is 3×3 throughout
- Total parameters in non FC layers $= \sim 16M$
- Total Parameters in FC layers $= (512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122M$
- Most parameters are in the first FC layer ($\sim$ 102M)

GoogLeNet - **Inception v1**



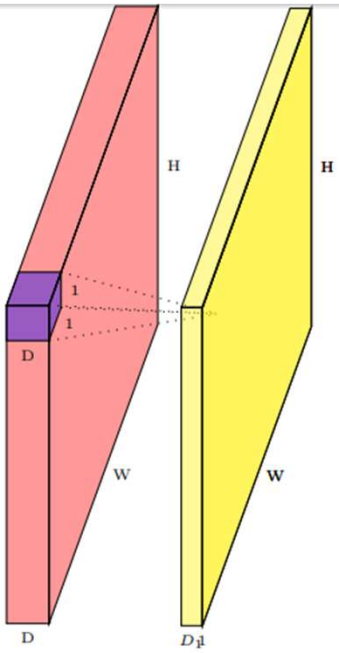
- Consider the output at a certain layer of a convolutional neural network
- After this layer we could apply a max-pooling layer
- Or a 1×1 convolution
- Or a 3×3 convolution
- Or a 5×5 convolution
- **Question:** Why choose between these options (convolution, maxpooling, filter sizes)?
- **Idea:** Why not apply all of them at the same time and then concatenate the feature maps?

GoogLeNet

introduced a novel module called the **Inception module** to improve computational efficiency and reduce the model's number of parameters without compromising performance.

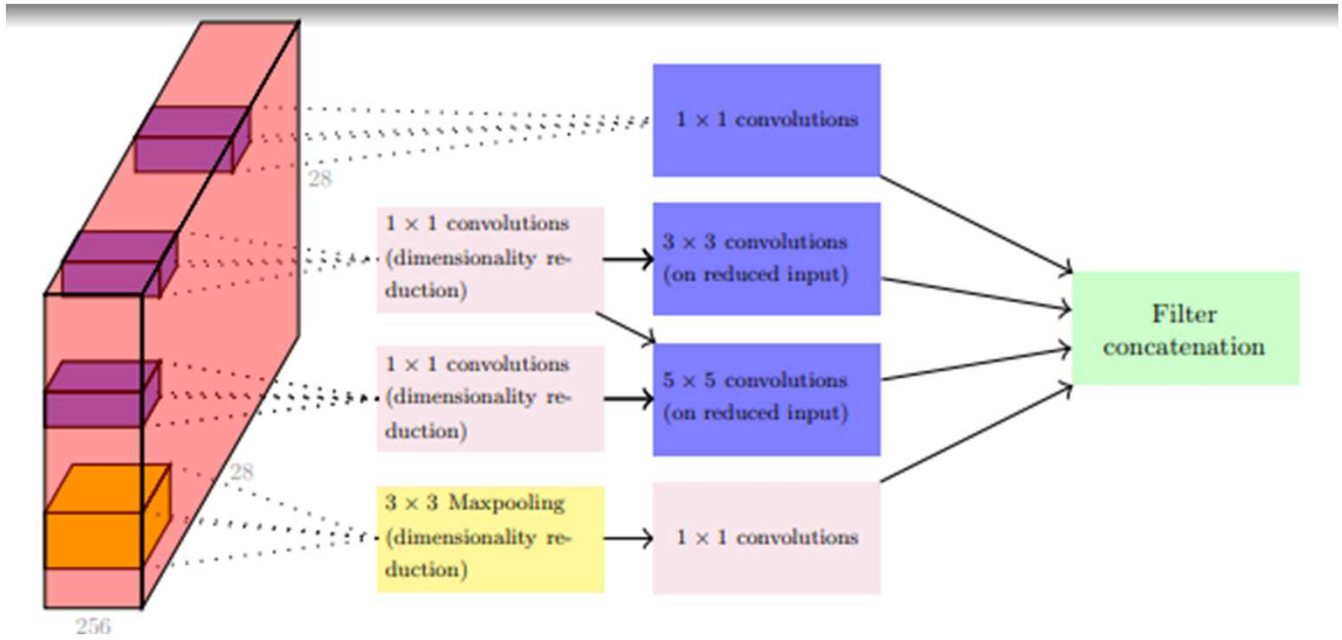
- **Inception Modules**:
- The core innovation in GoogLeNet is the Inception module, which processes input at multiple scales (1x1, 3x3, and 5x5 convolutions) and combines the outputs.
- Each module performs a mix of 1x1, 3x3, and 5x5 convolutions, along with 3x3 max-pooling, which are concatenated to capture both local and global features effectively.
- Using 1x1 convolutions reduces dimensionality and computational cost, making the network deeper without increasing the number of parameters excessively.
- **Depth and Architecture**:
- GoogLeNet has **22 layers** (excluding pooling and softmax layers) with only **5 million parameters**, which is much smaller than VGG-16's 138 million parameters, despite being deeper.

GoogLeNet

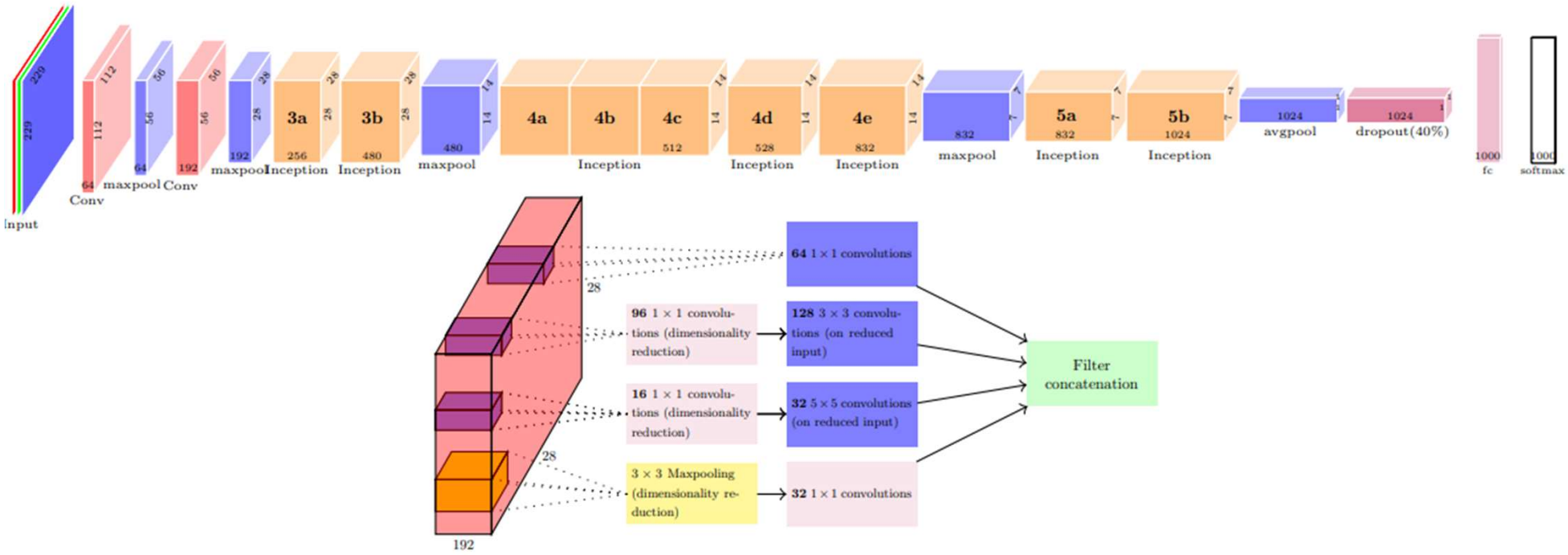


- Yes, by using 1×1 convolutions
- Huh?? What does a 1×1 convolution do ?
- It aggregates along the depth
- So convolving a $D \times W \times H$ input with D_1 1×1 ($D_1 < D$) filters will result in a $D_1 \times W \times H$ output ($S = 1, P = 0$)
- If $D_1 < D$ then this effectively reduces the dimension of the input and hence the computations
- Specifically instead of $\mathrm{O}(F \times F \times D)$ we will need $\mathrm{O}(F \times F \times D_1)$ computations
- We could then apply subsequent 3×3, 5×5 filter on this reduced output

GoogLeNet

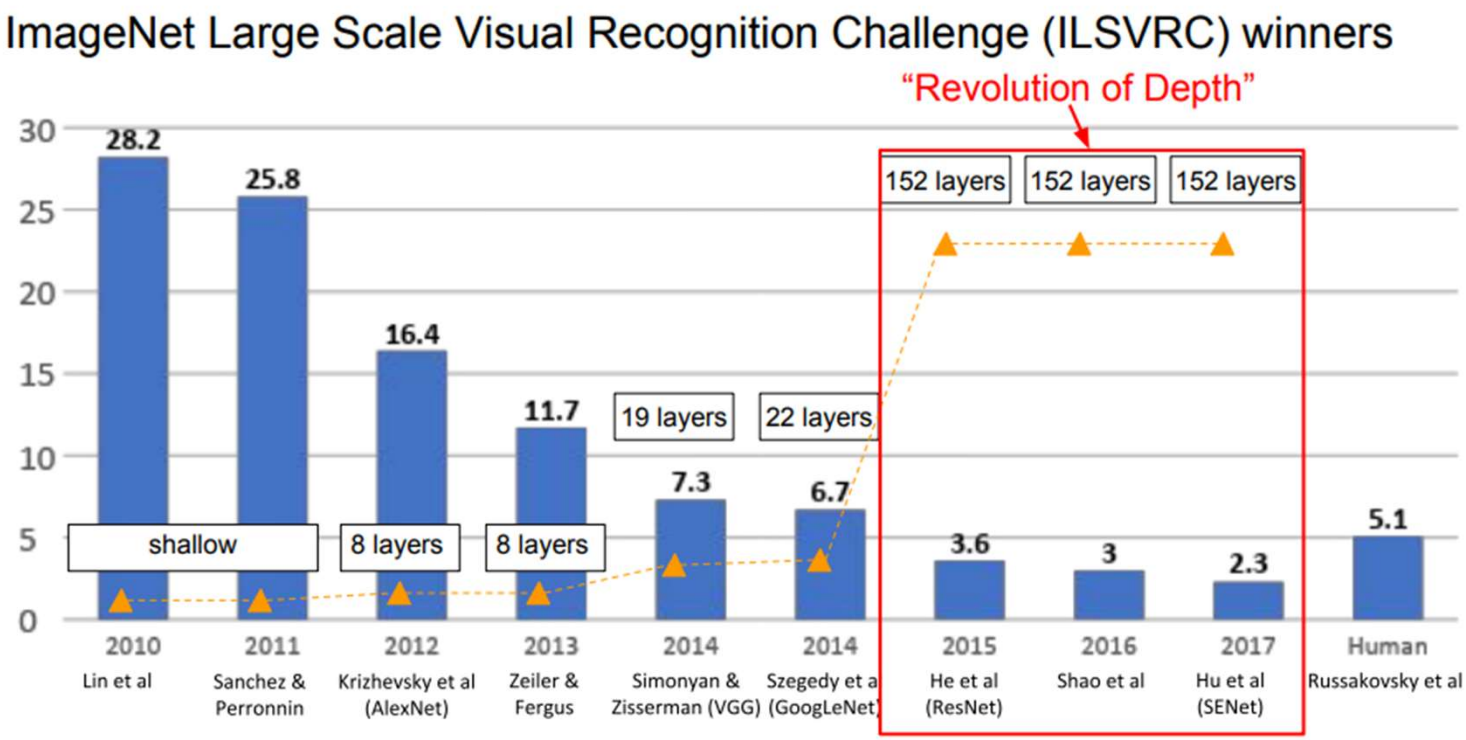

28
28
256
1 × 1 convolutions
1 × 1 convolutions (dimensionality reduction)
3 × 3 convolutions (on reduced input)
1 × 1 convolutions (dimensionality reduction)
5 × 5 convolutions (on reduced input)
3 × 3 Maxpooling (dimensionality reduction)
1 × 1 convolutions
Filter concatenation

GoogLeNet


input
Conv
maxpool
Conv
maxpool
Inception
Inception
maxpool
Inception
Inception
Inception
maxpool
Inception
Inception
avgpool
dropout(40%)
fc
softmax
3a
3b
4a
4b
4c
4d
4e
5a
5b
64 1 × 1 convolutions
96 1 × 1 convolutions (dimensionality reduction)
128 3 × 3 convolutions (on reduced input)
16 1 × 1 convolutions (dimensionality reduction)
32 5 × 5 convolutions (on reduced input)
3 × 3 Maxpooling (dimensionality reduction)
32 1 × 1 convolutions
Filter concatenation
28
28
192

GoogLeNet

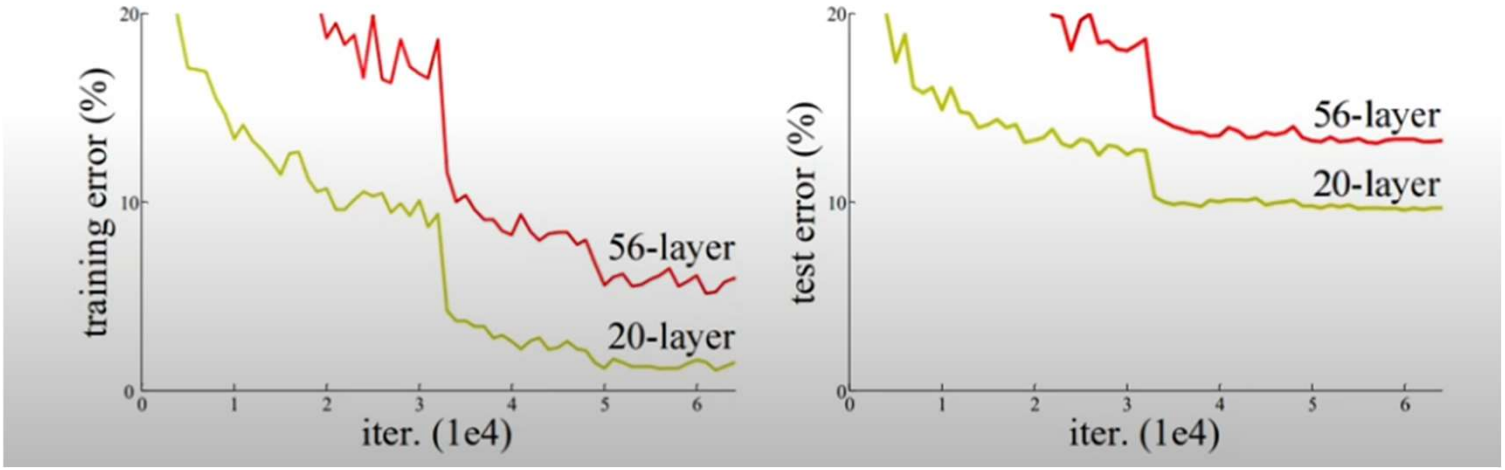
Successors
Following the success of GoogLeNet, Google continued to improve the Inception architecture with the development of:
•**Inception v2**: Introduced factorization techniques like reducing 5x5 convolutions into two smaller 3x3 convolutions.
•**Inception v3**: Introduced optimizations like the use of batch normalization, improved factorization strategies, and the RMSProp optimizer.
•**Inception v4 and Inception-ResNet**: Further improvements in both accuracy and efficiency, with the inclusion of residual connections.


ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners
"Revolution of Depth"
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.6
3
2.3
5.1
shallow
8 layers
8 layers
19 layers
22 layers
152 layers
152 layers
152 layers
2010
Lin et al
2011
Sanchez & Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler & Fergus
2014
Simonyan & Zisserman (VGG)
2014
Szegedy et a (GoogLeNet)
2015
He et al (ResNet)
2016
Shao et al
2017
Hu et al (SENet)
Human
Russakovsky et al

ResNet

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com



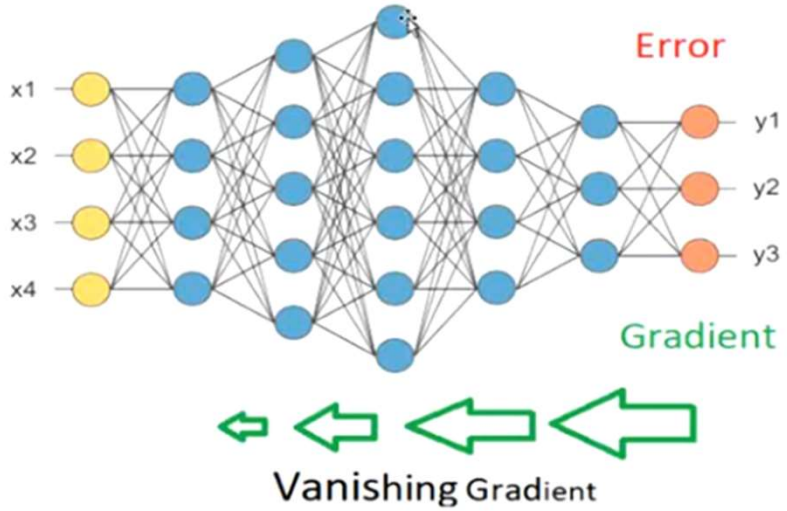
56-layer model performs worse on both training and test error -> The deeper model performs worse, but it's not caused by overfitting!

Hypothesis: the problem is an optimization problem, deeper models are harder to optimize

Is learning better networks as easy as stacking more layers?

- vanishing/exploding gradients
  - normalized initialization and intermediate normalization layers
- degradation problem : with the network depth increasing, accuracy gets saturated and then degrades rapidly.
- not caused by overfitting
- Addressed by deep residual learning

Vanishing Gradient Problem



The Degradation Problem

- When the network depth increases, accuracy gets saturated, and then **degrades rapidly**
- Adding more layers leads to higher training error

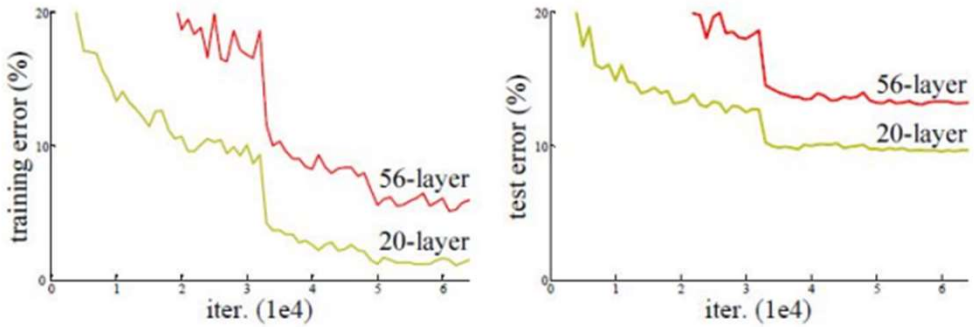
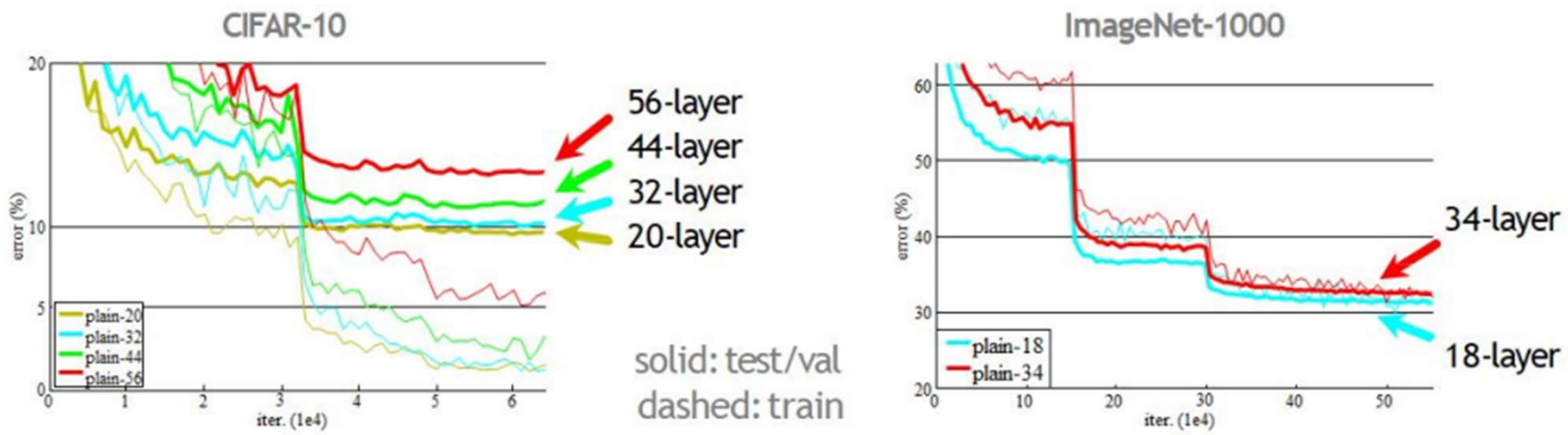


Figure 1: Training Error (left) and Test error (right) on CIFAR-10 with “Plain” networks

The Degradation Problem



- Overly deep plain networks have higher training error
- A general phenomenon, observed in many datasets

Deep Residual Learning

1. **Traditional Mapping:** A deep neural network tries to learn a function $F(x)$, which maps the input x to the desired output.
2. **Residual Mapping:** Instead of learning $F(x)$, ResNet learns the **residual**:

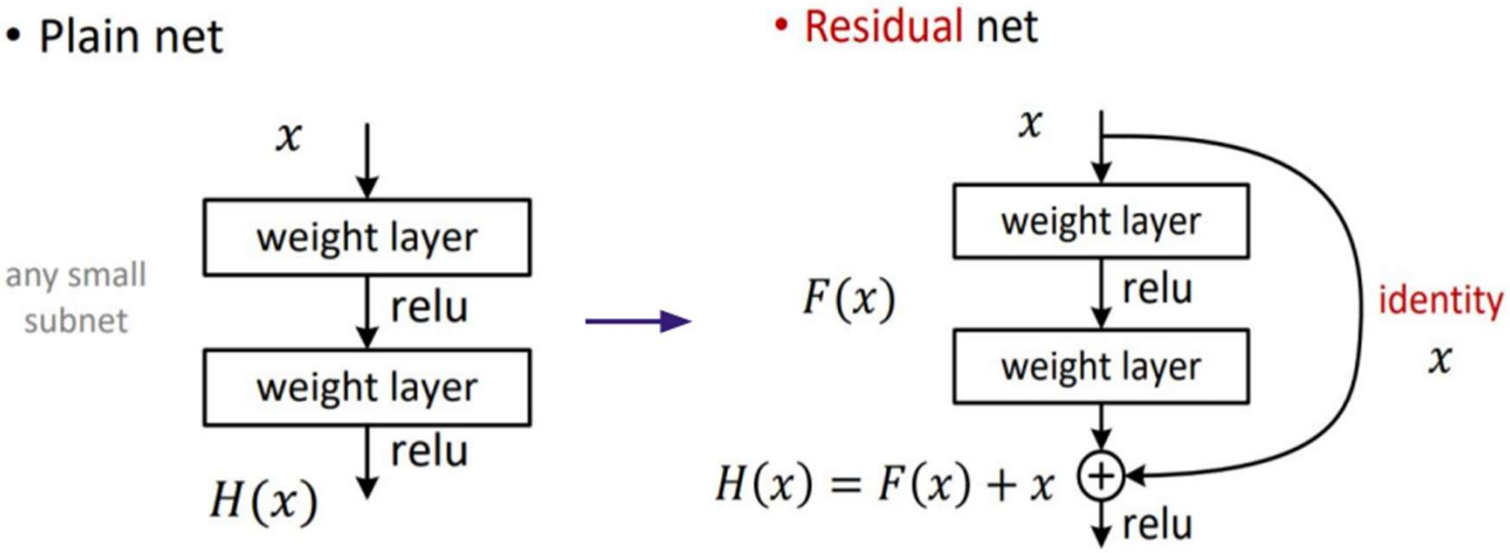
   $$H(x) = F(x) + x$$

   where:

   - $F(x)$ is the residual function to be learned by the network.
   - x is the input to the layer (or block).
   - $H(x)$ is the final output after adding $F(x)$ and x.

By reformulating the problem this way, the network focuses on learning the difference (residual) between the input and output, which is often easier than learning the entire transformation.

Deep Residual Learning



Learn the residual mapping F(x) rather than unreferenced H(x)

Skip Connection: bypasses the convolutional layers and adds the input directly to the output.

Types of Skip Connections

- **Identity Connection:** The input x is directly added to the output without any transformation.
- **Projection Connection:** When dimensions do not match (e.g., due to different feature map sizes), a linear transformation (e.g., 1×1 convolution) is applied to x.

1. Identity Connection

- **Definition:**
  - The input (x) is directly added to the output of the residual block ($F(x)$) without any transformation.
  - Used when the input and output have the same dimensions.
- **Mathematical Representation:**

$$y = F(x) + x$$

  - x: Input to the residual block.
  - $F(x)$: Output of the block's transformations (e.g., convolution, ReLU, etc.).
  - y: Final output after addition.
- **When to Use:**
  - When the input (x) and the residual block output ($F(x)$) have the same spatial dimensions and the same number of channels.
  - No additional computation is needed for alignment.

2. Projection Connection

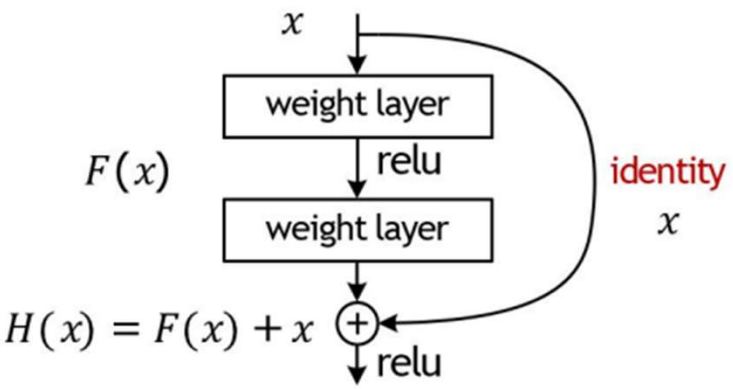
- **Definition:**
  - The input (x) is transformed (via a convolutional layer or other operations) before being added to the output ($F(x)$).
  - Used when the dimensions of the input and output differ.
- **Mathematical Representation:**

$$y = F(x) + W_s \cdot x$$

  - W_s: A projection layer (e.g., a 1×1 convolution) that transforms x to match the dimensions of $F(x)$.
  - $F(x)$: Output of the residual block.
  - y: Final output after addition.
- **When to Use:**
  - When the spatial size of the input and output differs (e.g., due to downsampling).
  - When the number of channels in x and $F(x)$ do not match.

Identity shortcut

- $F(x)$ is a residual mapping w.r.t. identity



- If optimal mapping is closer to identity, easier to capture small fluctuations
- If identity is optimal, easy to set weights as 0

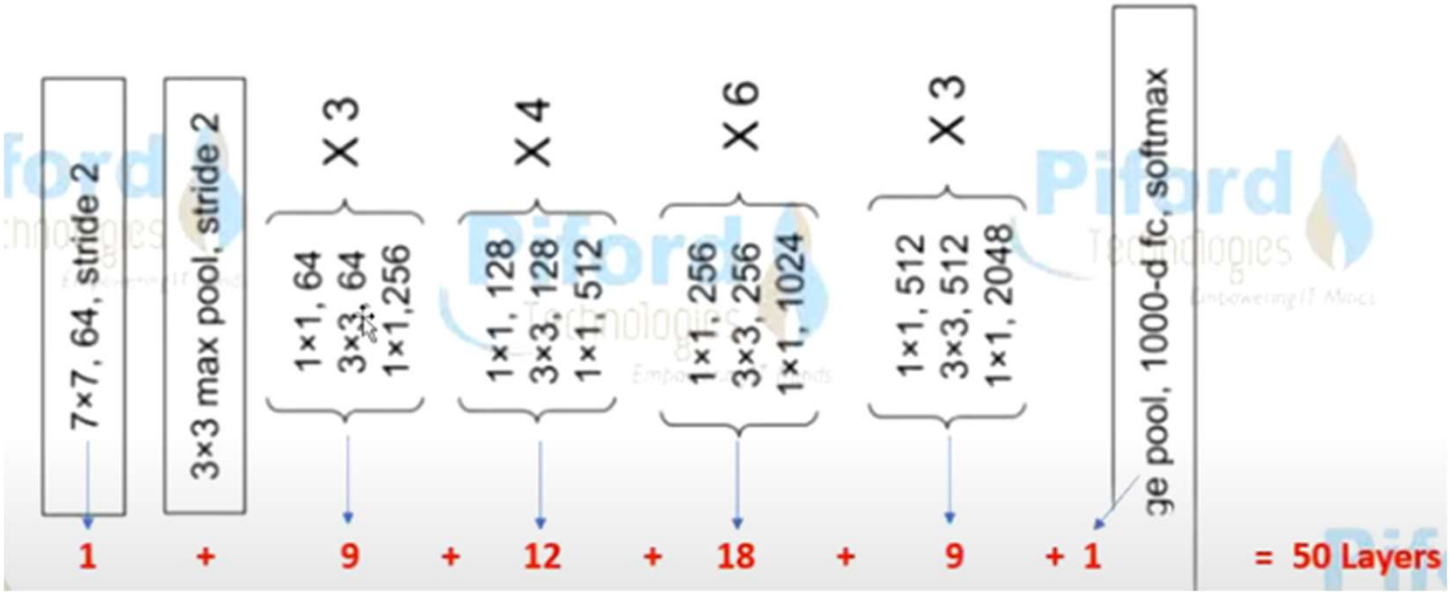
- Residual learning reformulates the target function $H(x)$ to:

$$H(x) = F(x) + x$$

  where $F(x)$ is the residual function, and x is the input to the layer.

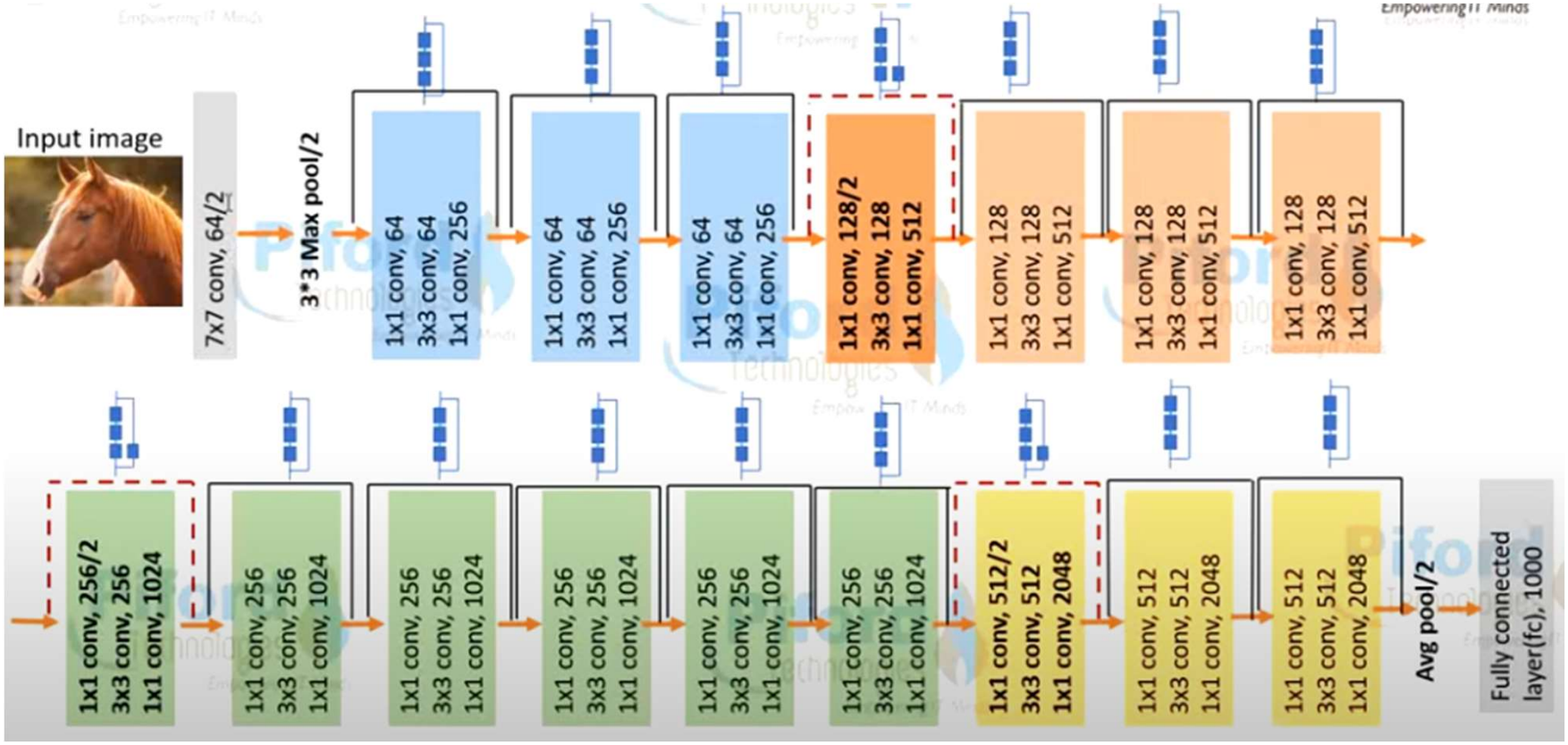
- If the desired function $H(x)$ is close to x (the identity mapping), the residual $F(x)$ will be small. Thus, the network only needs to learn these small fluctuations $F(x) = H(x) - x$.
- Learning small residuals $F(x)$ is simpler and faster because:
  - The gradients during backpropagation focus on fine-tuning the small differences.
  - Small residuals are less likely to cause numerical instability during training.

ResNet 50

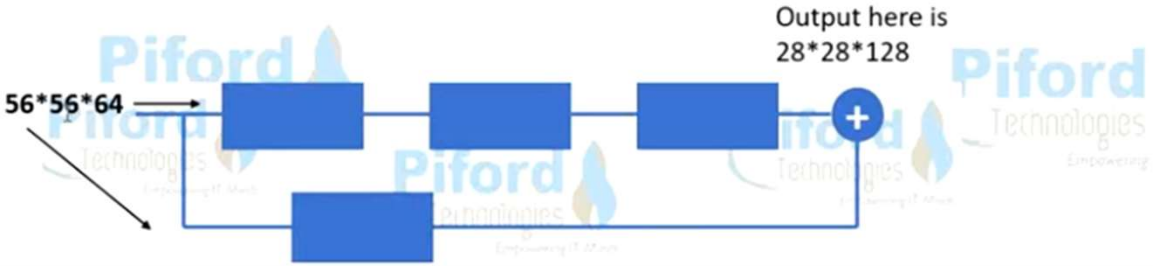


ResNet-18
ResNet-34
ResNet-50,
ResNet-101,
ResNet-152

ResNet 50


Input image
7x7 conv, 64/2
3*3 Max pool/2
1x1 conv, 64
3x3 conv, 64
1x1 conv, 256
1x1 conv, 128/2
3x3 conv, 128
1x1 conv, 512
1x1 conv, 128
3x3 conv, 128
1x1 conv, 512
1x1 conv, 256/2
3x3 conv, 256
1x1 conv, 1024
1x1 conv, 512/2
3x3 conv, 512
1x1 conv, 2048
1x1 conv, 512
3x3 conv, 512
1x1 conv, 2048
Avg pool/2
Fully connected layer(fc), 1000

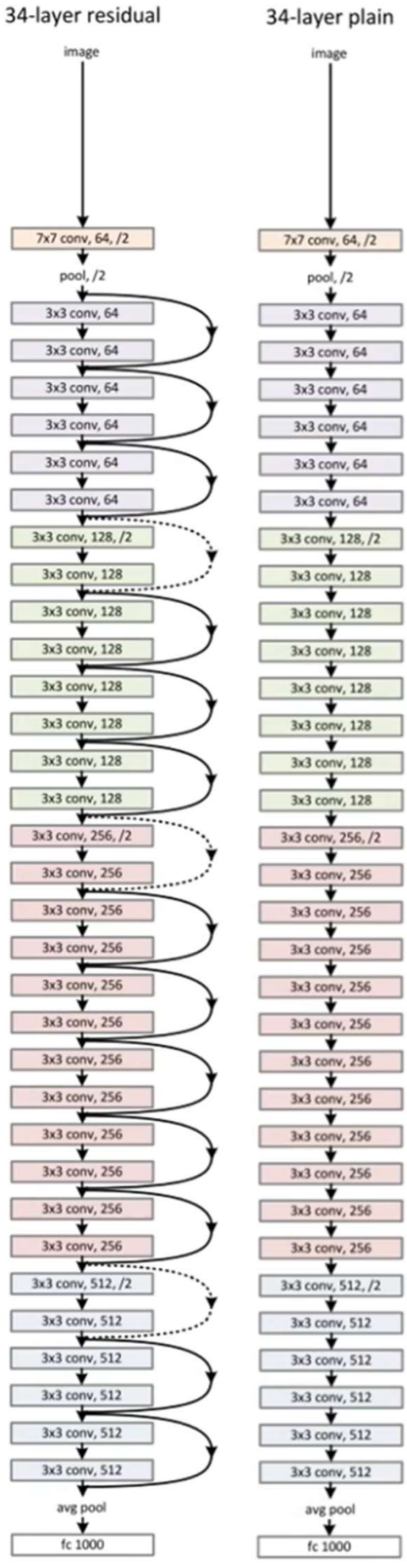
1 x 1 Convolutions



$$\frac{n+2P-f}{s}+1 \quad * \quad \frac{n+2P-f}{s}+1$$

$$\frac{56-1}{2}+1 = 28 \quad * \quad \frac{56-1}{2}+1 = 28$$

$$28 * 28$$


34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
34-layer plain
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000

Summary

- **Key Concept**:
  - Learns residual mappings ($F(x) = H(x) - x$) instead of direct mappings ($H(x)$).
  - Introduced skip connections to alleviate **vanishing gradient problem**.
- **Architecture**:
  - **Residual Block**: $y = F(x) + x$ (Identity Connection).
  - **Bottleneck Block**: 1×1 → 3×3 → 1×1 convolutions for deeper networks.
- **Variants**:
  - **ResNet-18/34**: Basic residual blocks.
  - **ResNet-50/101/152**: Bottleneck blocks for efficiency.
- **Advantages**:
  - Enables training very deep networks (up to 152 layers).
  - Backbone for modern architectures like Faster R-CNN and Mask R-CNN.
- **Applications**:
  - Image Classification, Object Detection, Medical Imaging, etc.

Complexity Comparison

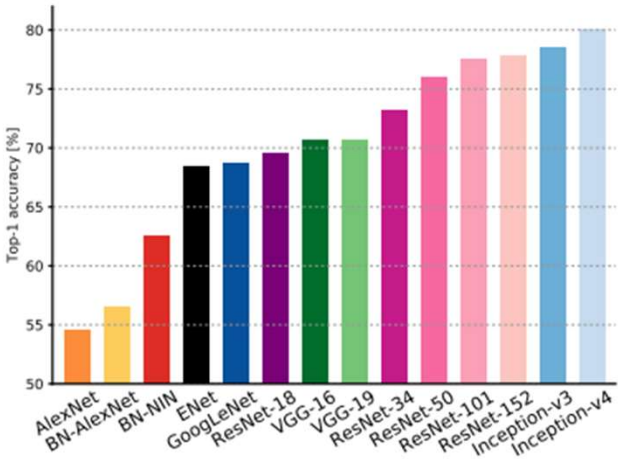


Figure 1: **Top1 *vs.* network.** Single-crop top-1 validation accuracies for top scoring single-model architectures. We introduce with this chart our choice of colour scheme, which will be used throughout this publication to distinguish effectively different architectures and their correspondent authors. Notice that networks of the same group share the same hue, for example ResNet are all variations of pink.

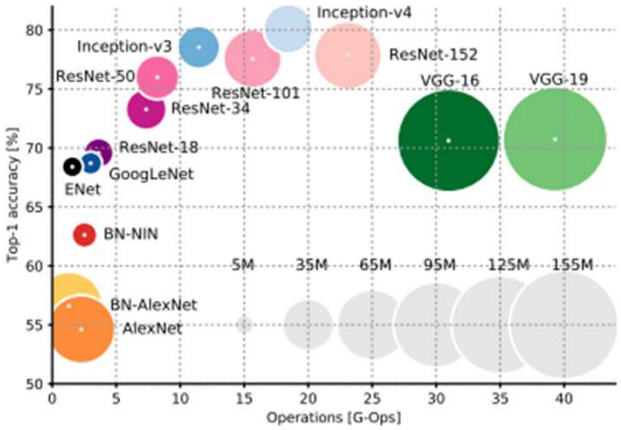
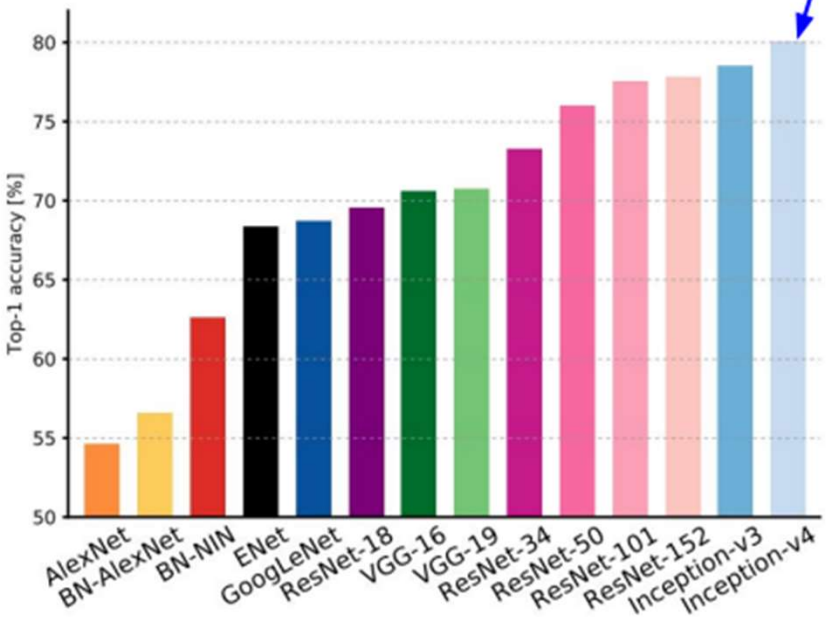


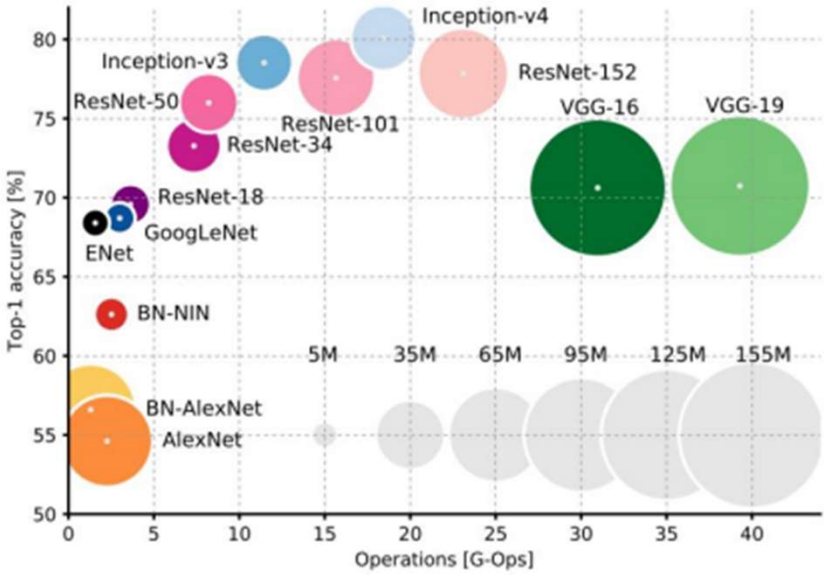
Figure 2: **Top1 *vs.* operations, size $\propto$ parameters.** Top-1 one-crop accuracy versus amount of operations required for a single forward pass. The size of the blobs is proportional to the number of network parameters; a legend is reported in the bottom right corner, spanning from 5×10^6 to 155×10^6 params. Both these figures share the same y-axis, and the grey dots highlight the centre of the blobs.

Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." *arXiv preprint arXiv:1605.07678* (2016).

Comparing complexity...

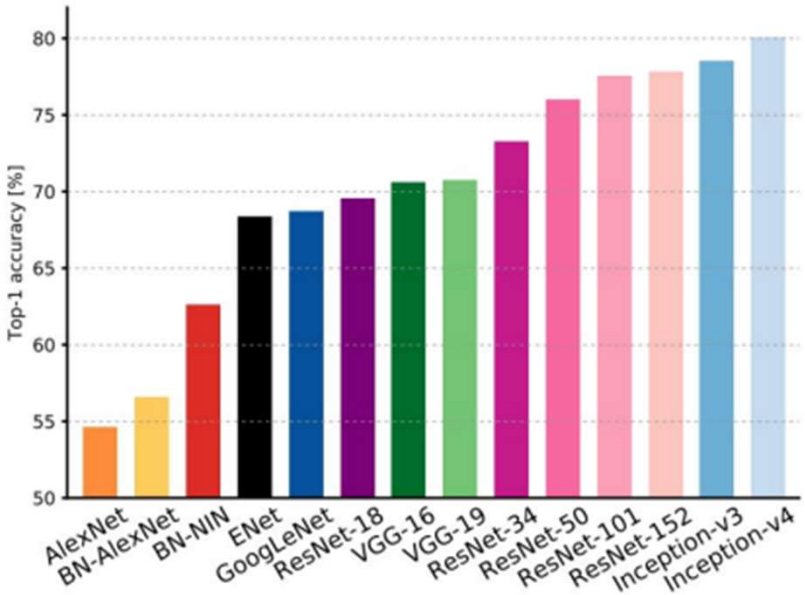
Inception-v4: Resnet + Inception!

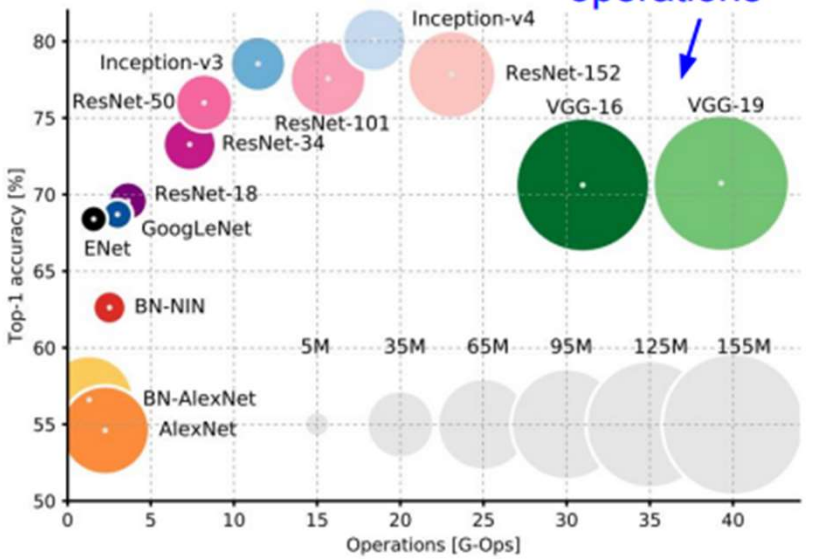




An Analysis of Deep Neural Network Models for Practical Applications, 2017.

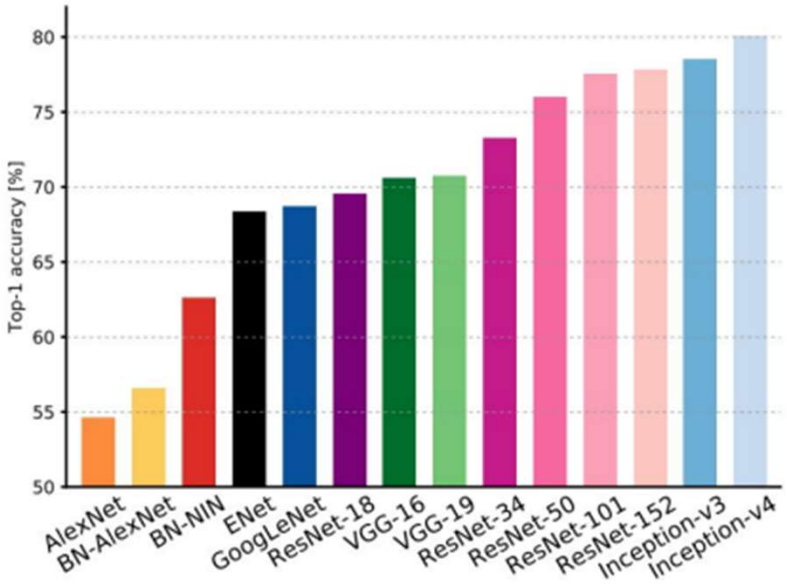
Comparing complexity...

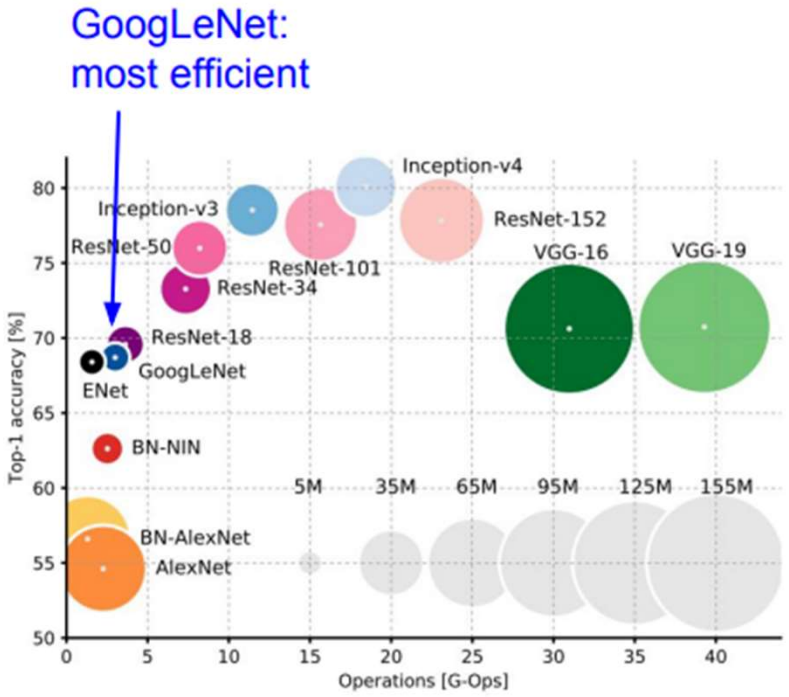




An Analysis of Deep Neural Network Models for Practical Applications, 2017.

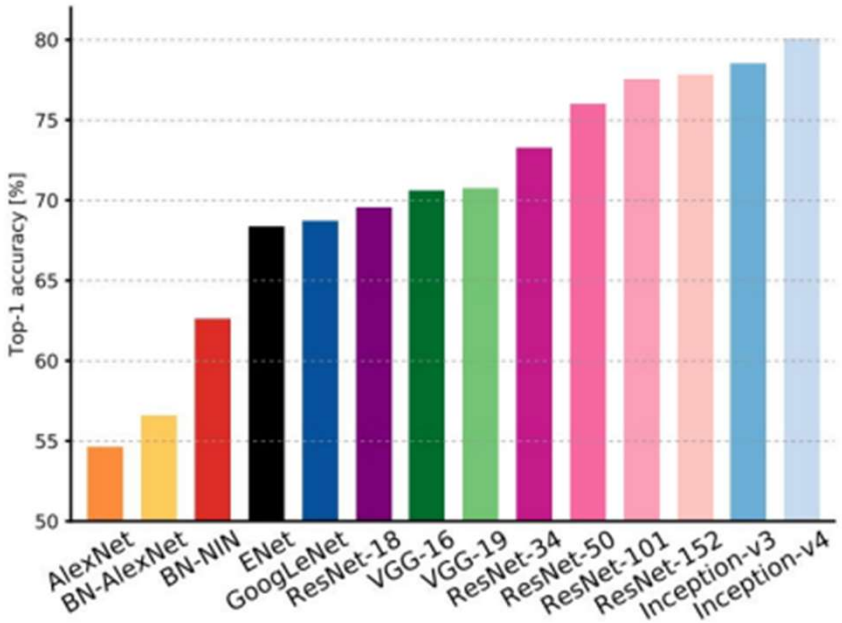
Comparing complexity...



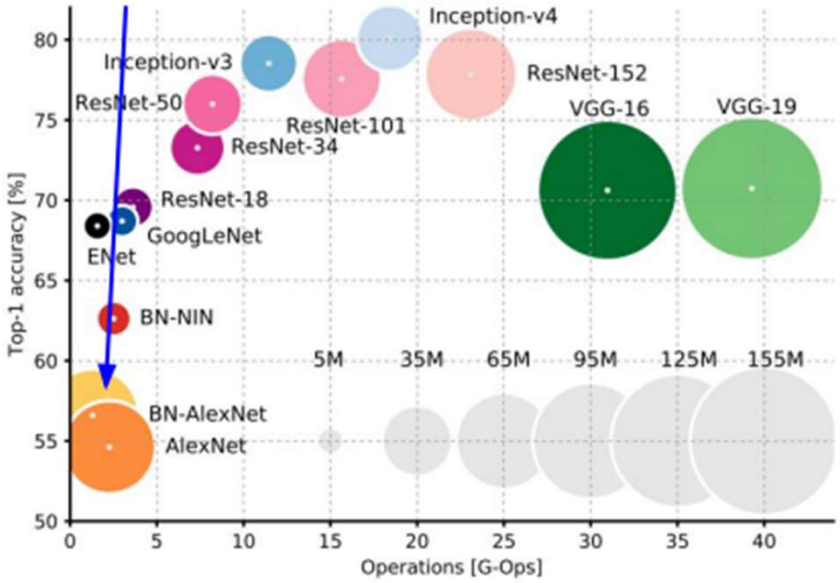


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparing complexity...

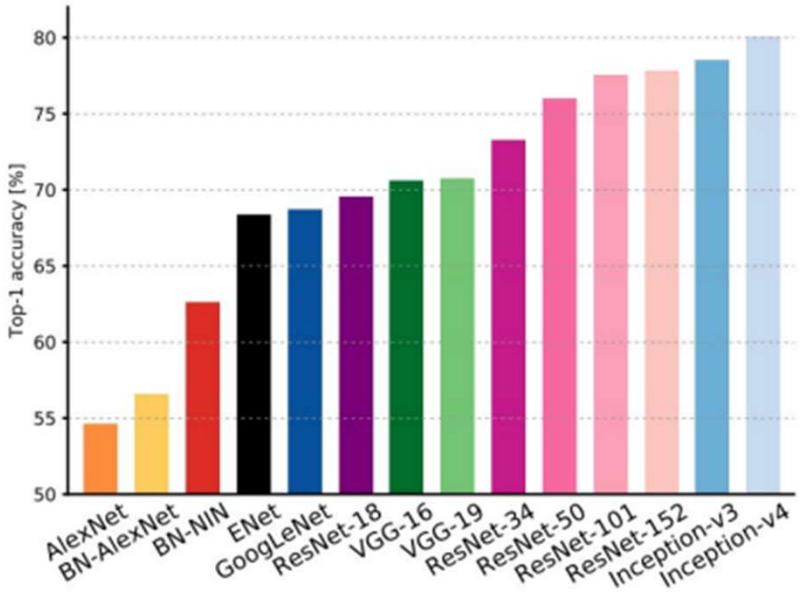


AlexNet:
Smaller compute, still memory heavy, lower accuracy

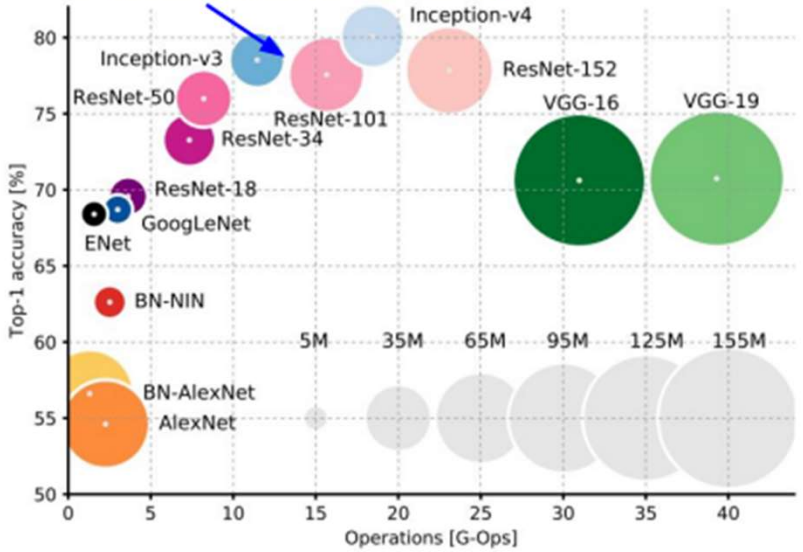


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparing complexity...



ResNet:
Moderate efficiency depending on model, highest accuracy



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

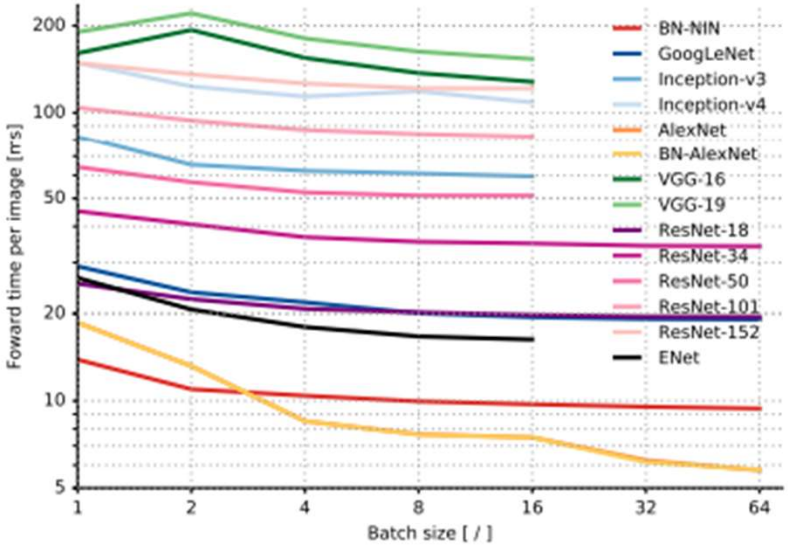


Figure 3: **Inference time *vs.* batch size.** This chart show inference time across different batch sizes with a logarithmic ordinate and logarithmic abscissa. Missing data points are due to lack of enough system memory required to process larger batches. A speed up of $3\times$ is achieved by AlexNet due to better optimisation of its fully connected layers for larger batches.

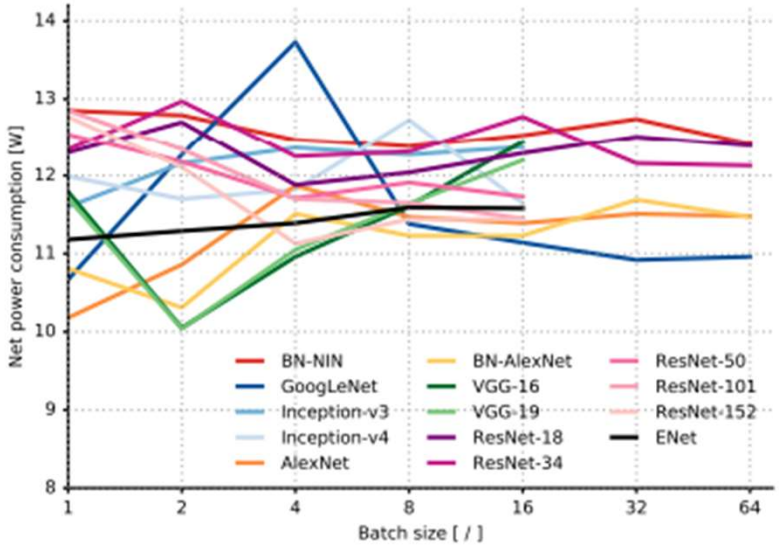
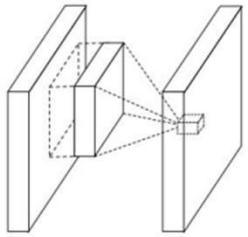


Figure 4: **Power *vs.* batch size.** Net power consumption (due only to the forward processing of several DNNs) for different batch sizes. The idle power of the TX1 board, with no HDMI screen connected, was 1.30 W on average. The max frequency component of power supply current was 1.4 kHz, corresponding to a Nyquist sampling frequency of 2.8 kHz.

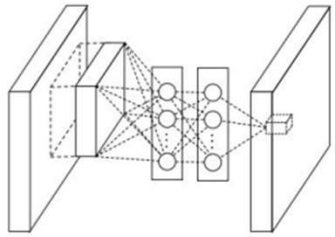
Network in Network (NiN)

[Lin et al. 2014]

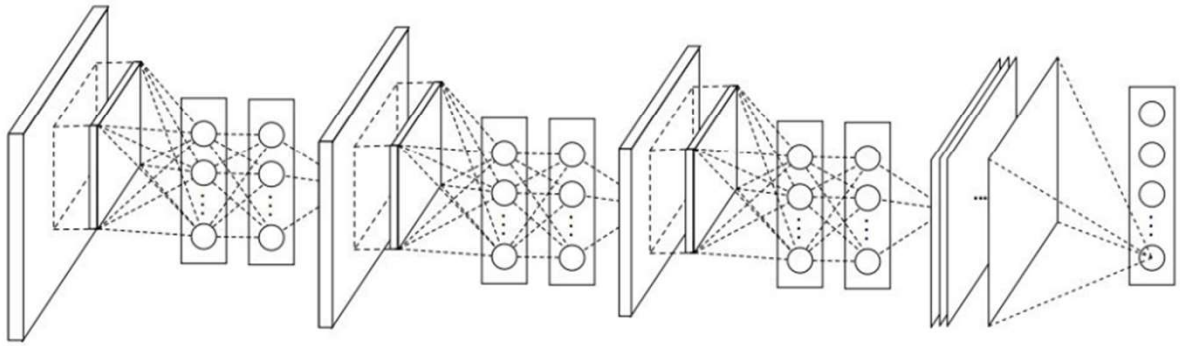
- Mlpconv layer with "micronetwork" within each conv layer to compute more abstract features for local patches
- Micronetwork uses multilayer perceptron (FC, i.e. 1x1 conv layers)
- Precursor to GoogLeNet and ResNet "bottleneck" layers
- Philosophical inspiration for GoogLeNet



(a) Linear convolution layer



(b) Mlpconv layer

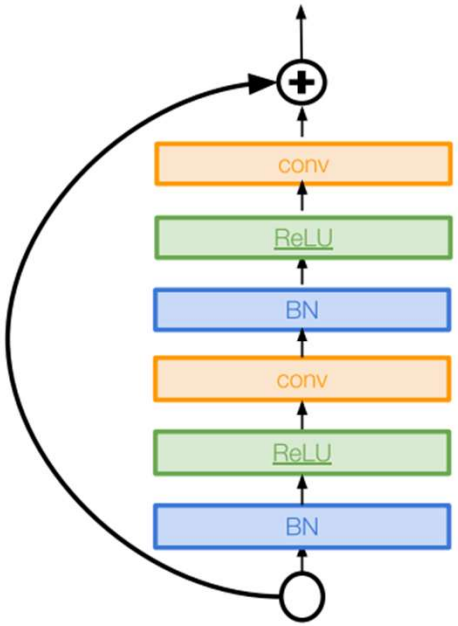


Improving ResNets...

Identity Mappings in Deep Residual Networks

[He et al. 2016]

- Improved ResNet block design from creators of ResNet
- Creates a more direct path for propagating information throughout network (moves activation to residual mapping pathway)
- Gives better performance

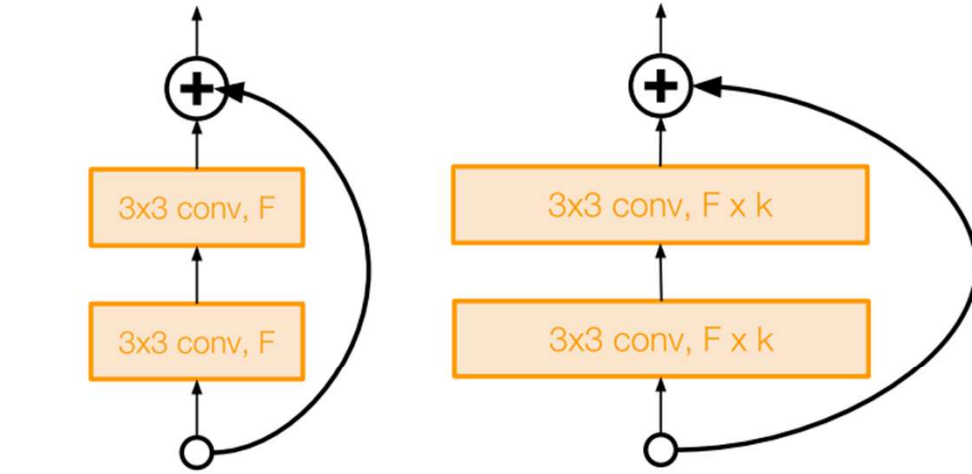


Improving ResNets...

Wide Residual Networks

[Zagoruyko et al. 2016]

- Argues that residuals are the important factor, not depth
- User wider residual blocks (F x k filters instead of F filters in each layer)
- 50-layer wide ResNet outperforms 152-layer original ResNet
- Increasing width instead of depth more computationally efficient (parallelizable)



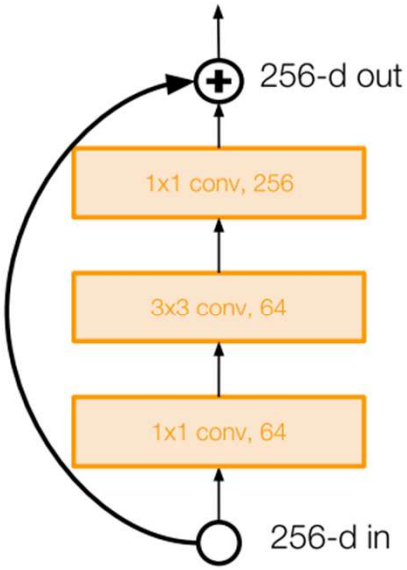
Basic residual block Wide residual block

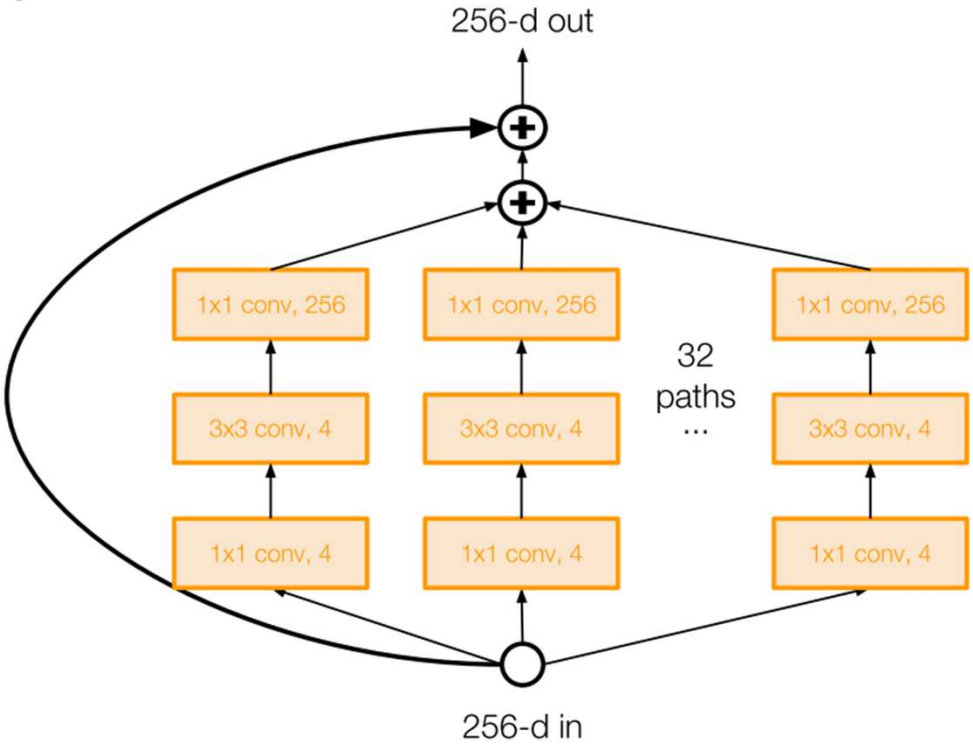
Improving ResNets...

Aggregated Residual Transformations for Deep Neural Networks (ResNeXt)

[Xie et al. 2016]

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



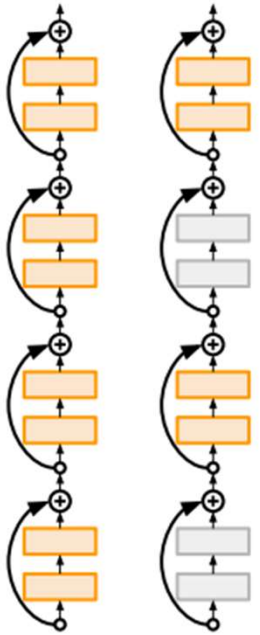


Improving ResNets...

Deep Networks with Stochastic Depth

[Huang et al. 2016]

- Motivation: reduce vanishing gradients and training time through short networks during training
- Randomly drop a subset of layers during each training pass
- Bypass with identity function
- Use full deep network at test time

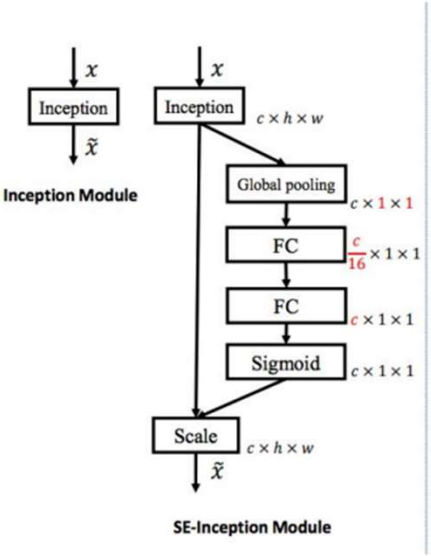


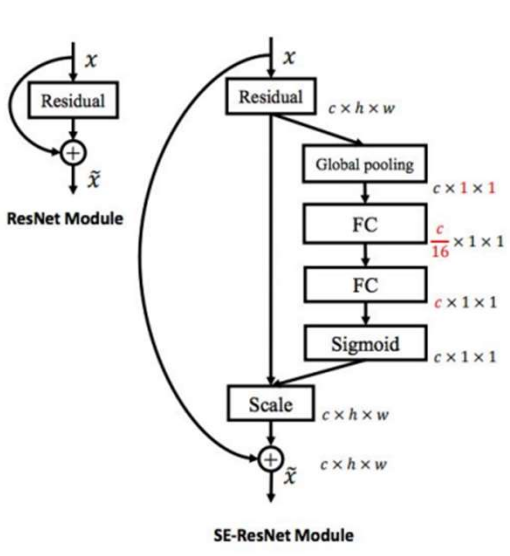
Improving ResNets...

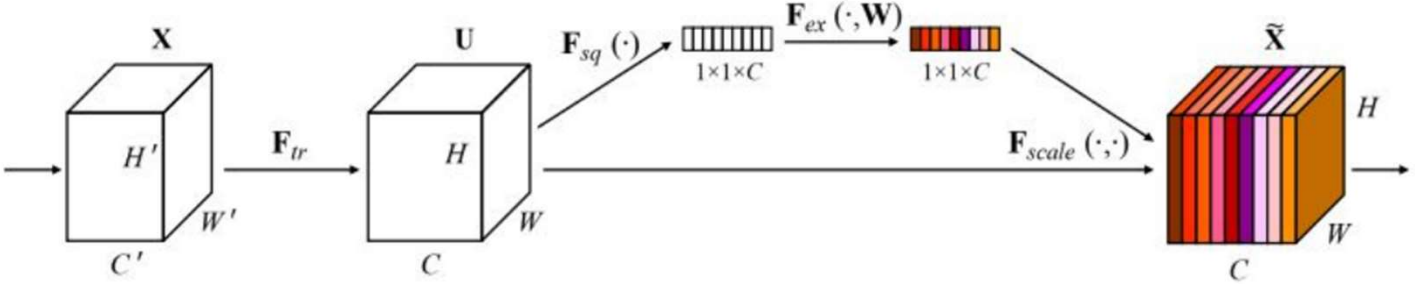
Squeeze-and-Excitation Networks (SENet)

[Hu et al. 2017]

- Add a “feature recalibration” module that learns to adaptively reweight feature maps
- Global information (global avg. pooling layer) + 2 FC layers used to determine feature map weights
- ILSVRC’17 classification winner (using ResNeXt-152 as a base architecture)





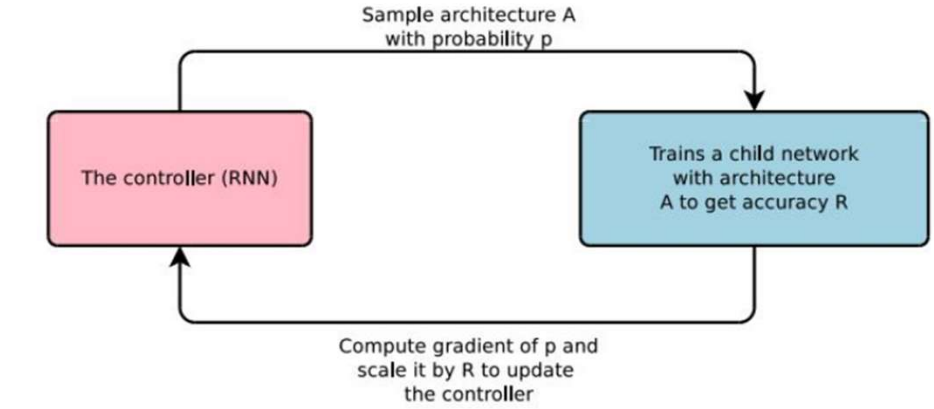


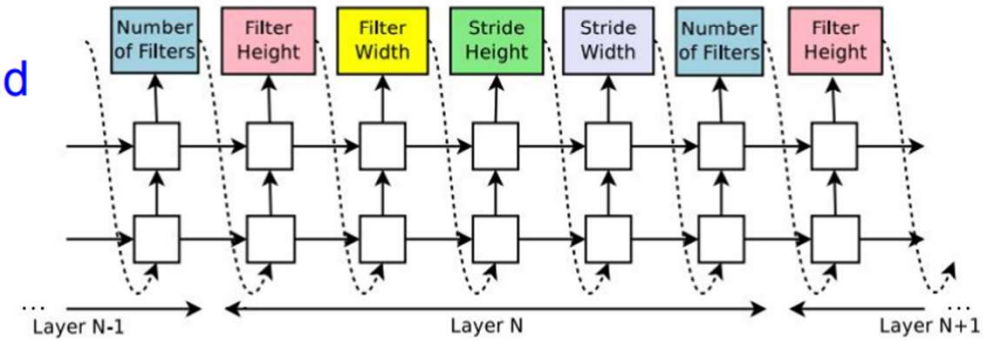
Meta-learning: Learning to learn network architectures...

Neural Architecture Search with Reinforcement Learning (NAS)

[Zoph et al. 2016]

- "Controller" network that learns to design a good network architecture (output a string corresponding to network design)
- Iterate:
  1) Sample an architecture from search space
  2) Train the architecture to get a "reward" R corresponding to accuracy
  3) Compute gradient of sample probability, and scale by R to perform controller parameter update (i.e. increase likelihood of good architecture being sampled, decrease likelihood of bad architecture)



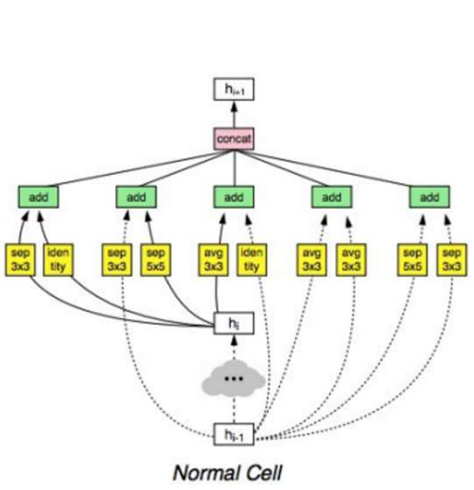


Meta-learning: Learning to learn network architectures...

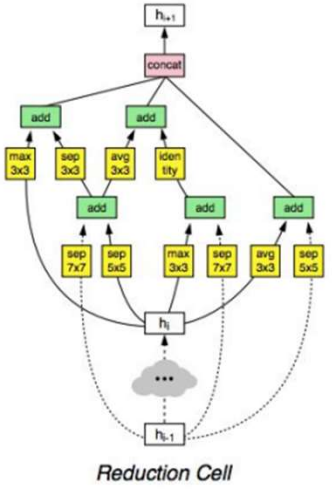
Learning Transferable Architectures for Scalable Image Recognition

[Zoph et al. 2017]

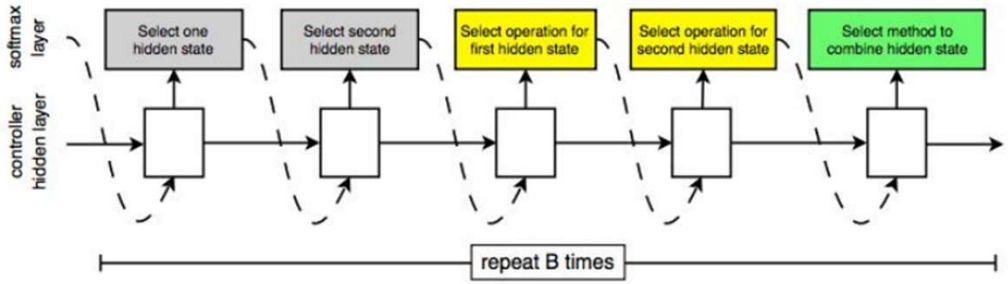
- Applying neural architecture search (NAS) to a large dataset like ImageNet is expensive
- Design a search space of building blocks ("cells") that can be flexibly stacked
- NASNet: Use NAS to find best cell structure on smaller CIFAR-10 dataset, then transfer architecture to ImageNet

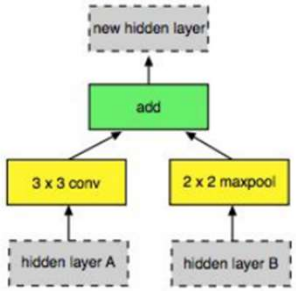


Normal Cell



Reduction Cell





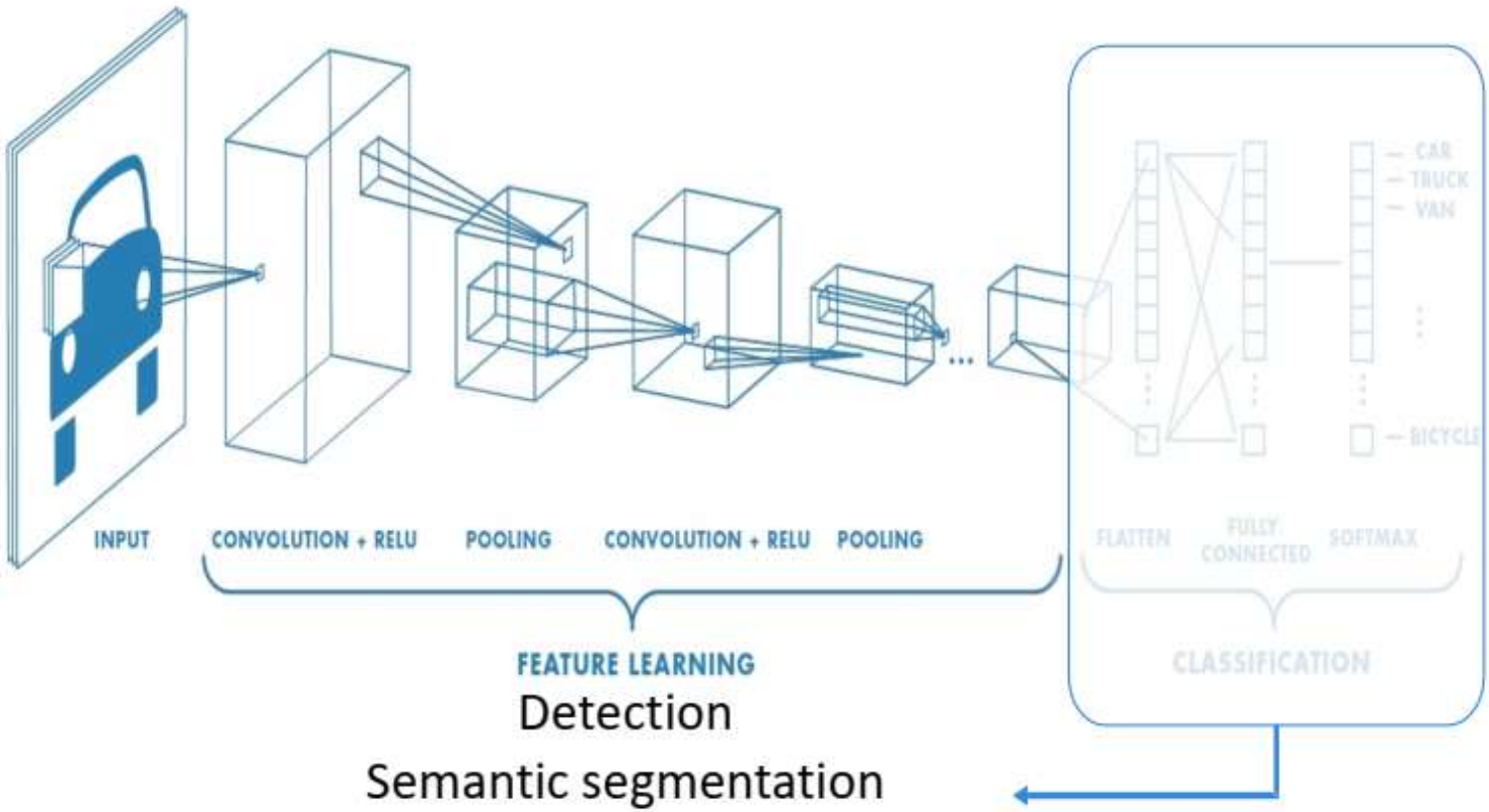
Summary: CNN Architectures

- VGG, GoogLeNet, ResNet all in wide use, available in model zoos
- ResNet current best default, also consider SENet when available
- Trend towards extremely deep networks
- Significant research centers around design of layer / skip connections and improving gradient flow
- Efforts to investigate necessity of depth vs. width and residual connections
- Even more recent trend towards meta-learning

Applications

https://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture09.pdf

An Architecture for Many Applications



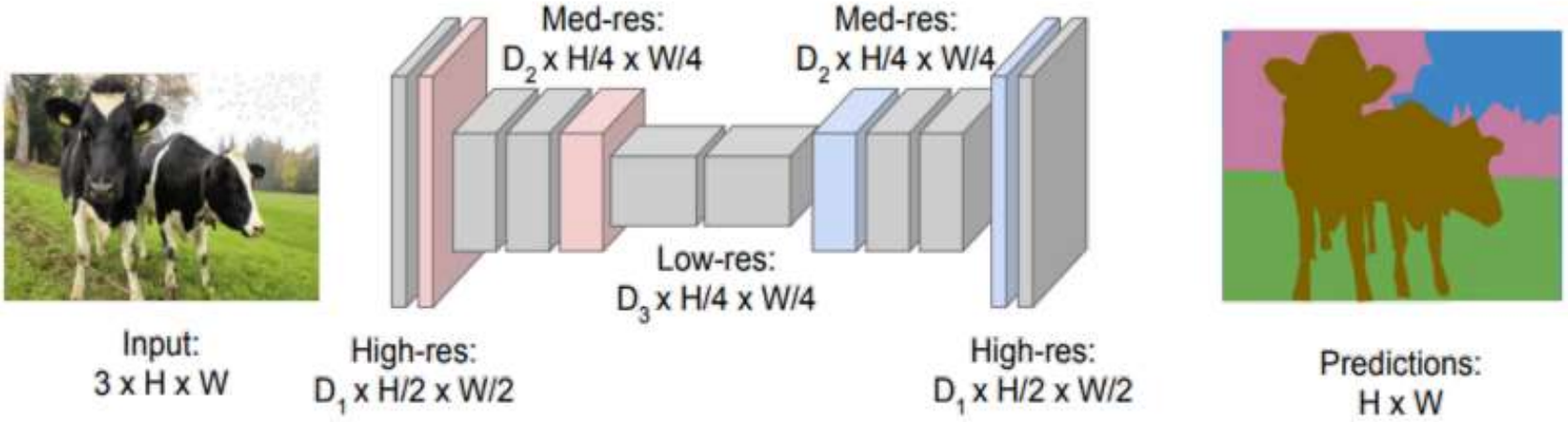
Detection

Semantic segmentation

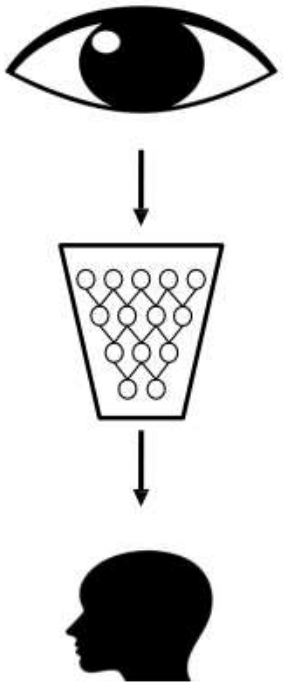
End-to-end robotic control

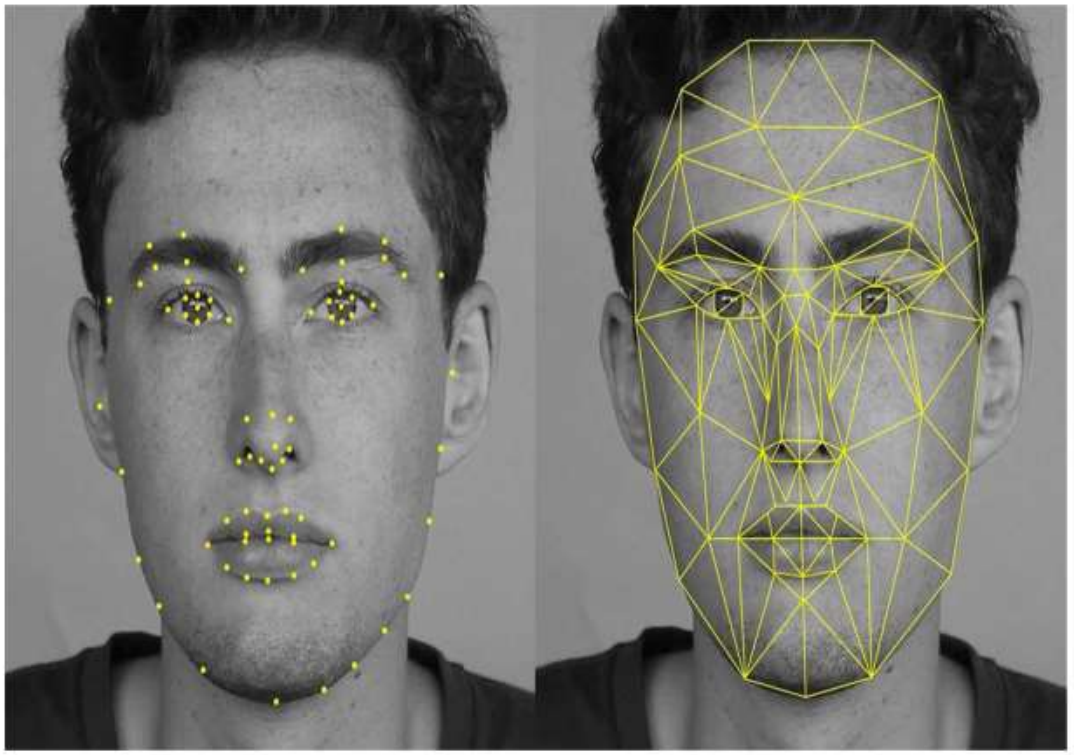
Semantic Segmentation: Fully Convolutional Networks

FCN: Fully Convolutional Network.
Network designed with all convolutional layers, with **downsampling** and **upsampling** operations

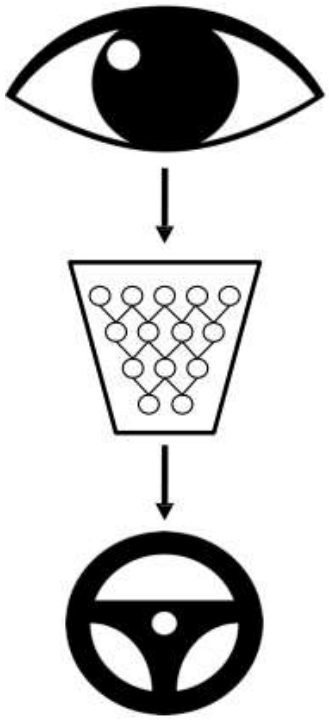


Facial Detection & Recognition

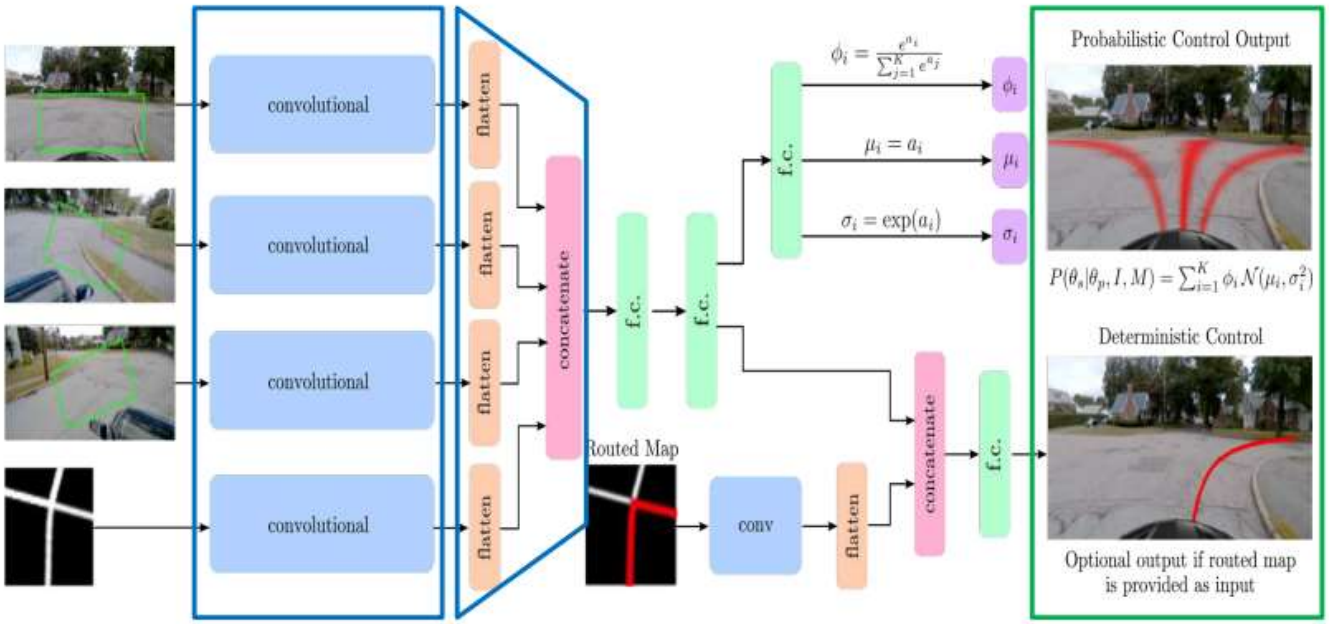




Self-Driving Cars



End-to-End Framework for Autonomous Navigation



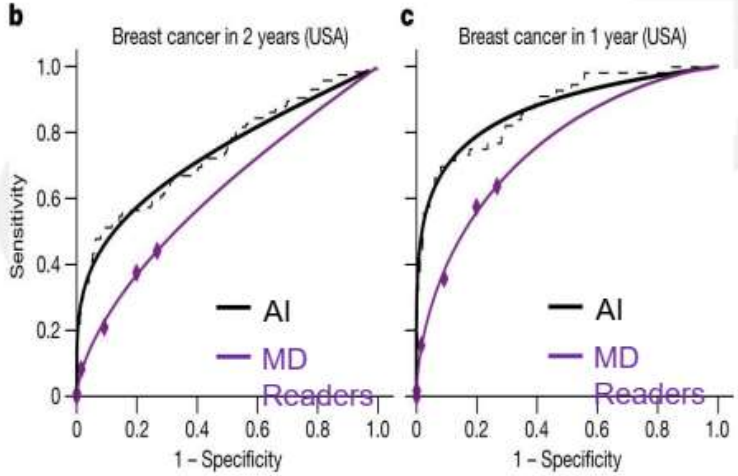
Amini+ *ICRA* 2019

Entire model trained end-to-end

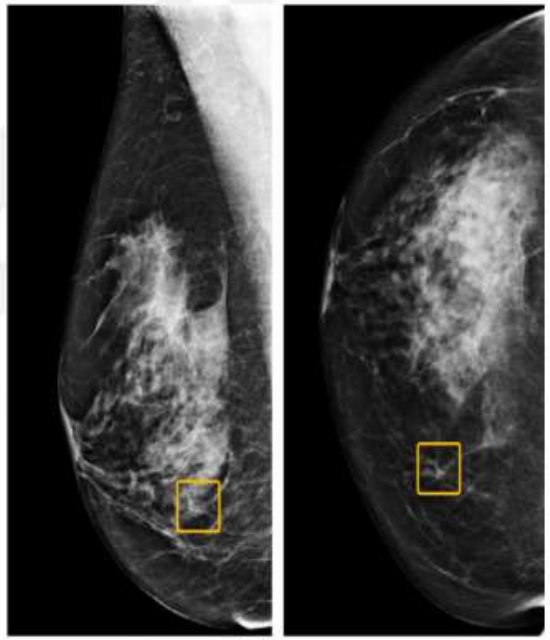
without any human labelling or annotations

Breast Cancer Screening

International evaluation of an AI system for breast cancer screening nature



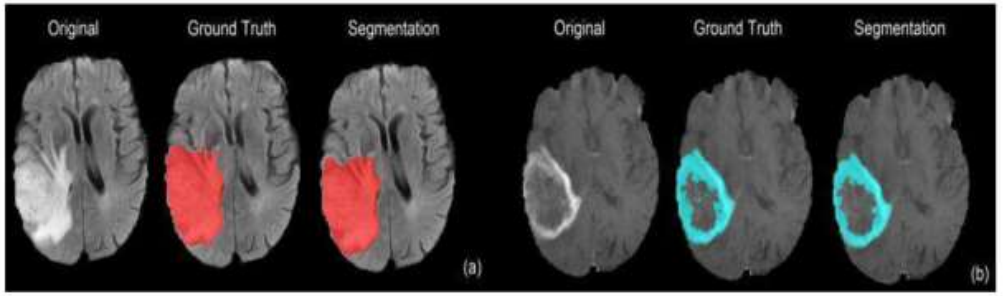
CNN-based system outperformed expert radiologists at detecting breast cancer from mammograms



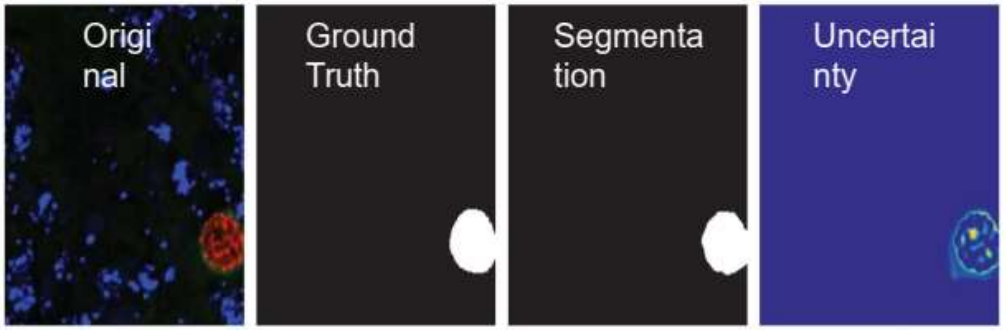
Breast cancer case missed by radiologist but detected by AI

Semantic Segmentation: Biomedical Image Analysis

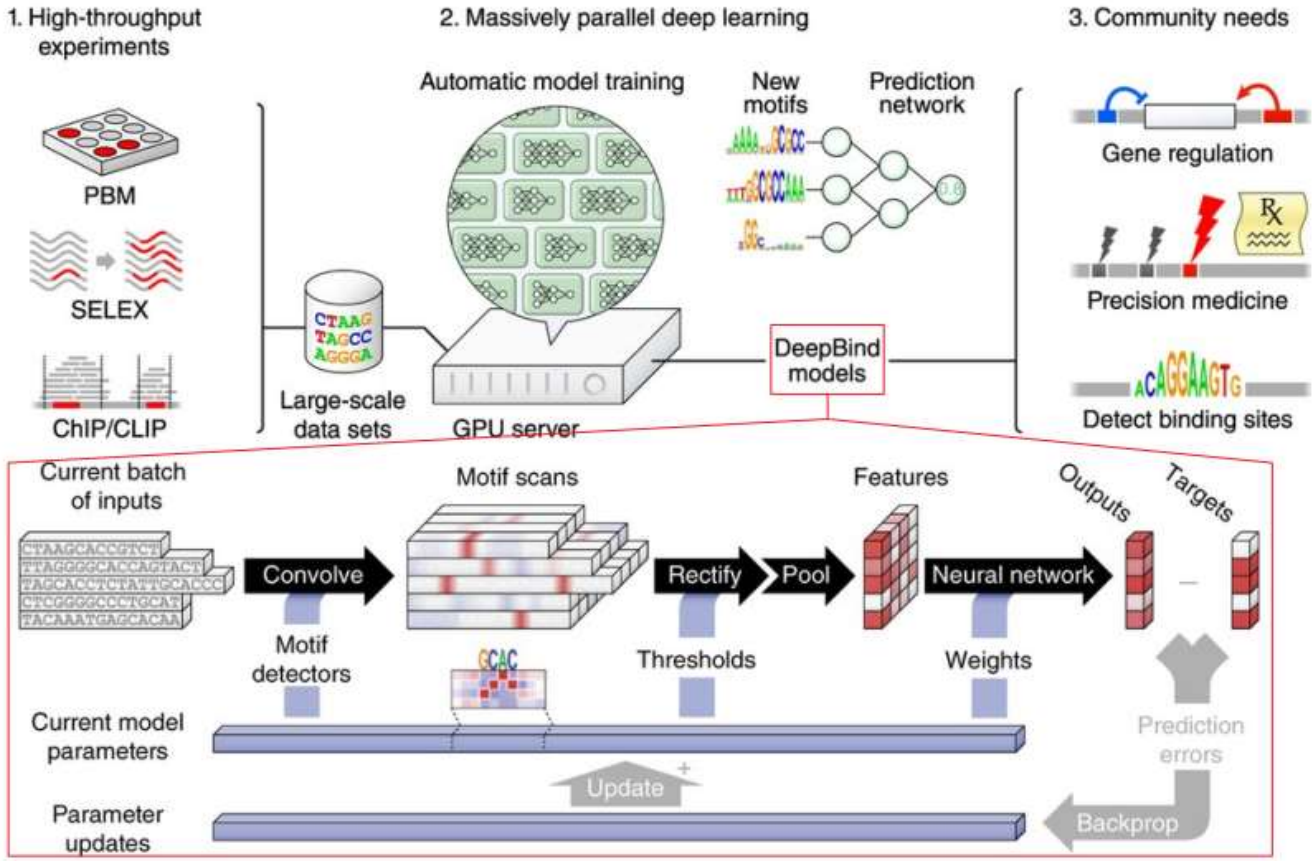
Brain Tumors
Dong+ *MIUA* 2017.



Malaria Infection
Soleimany+ *arXiv* 2019.

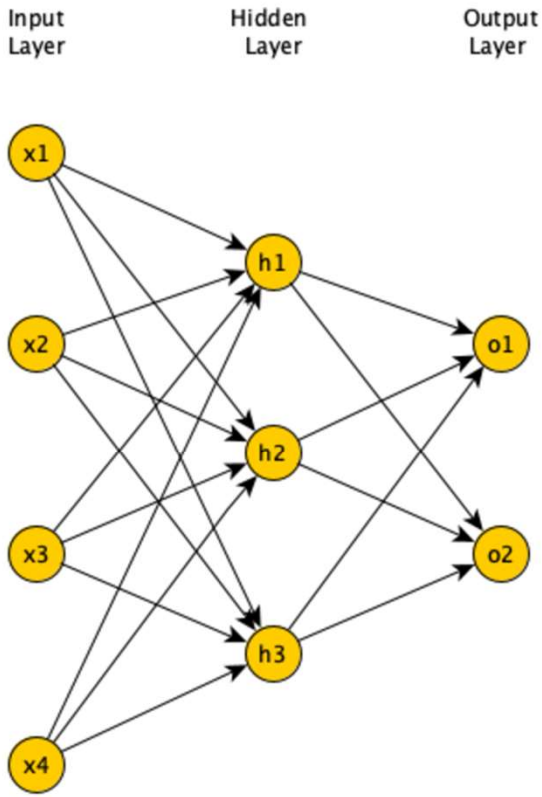


DeepBind


1. High-throughput experiments
2. Massively parallel deep learning
3. Community needs
PBM
SELEX
ChIP/CLIP
Large-scale data sets
Automatic model training
GPU server
New motifs
Prediction network
DeepBind models
Gene regulation
Precision medicine
Detect binding sites
Current batch of inputs
Convolve
Motif scans
Rectify
Pool
Features
Neural network
Outputs
Targets
Motif detectors
Thresholds
Weights
Current model parameters
Prediction errors
Update
Parameter updates
Backprop

Feedforward neural networks (FNN)

- In FNN all nodes in a layer connected to all nodes in next layer
  - Each connection has a weight, must be learned by learning algorithm



Limitations of FNN

- If input is 64 pixel by 64 pixel grayscale image
  - Each grayscale pixel represented by 1 value, usually between 0 to 255
  - Where 0 is black, 255 is white, and values in between are shades of gray
- Since each grayscale pixel represented by 1 value, we say *channel* size is 1
- Image represented by 64 x 64 x 1 = 4,096 values (rows x columns x channels)
  - Hence, input layer of FNN has 4096 nodes
- Lets assume next layer has 500 nodes
  - Since FNN fully connected, we have 4,096 x 500 = 2,048,000 weights

Limitations of FNN

- For complex problems, we need multiple hidden layers in our FNN
  - Compunds the problem of having many weights
- Having too many weights
  - Makes learning more difficult as dimension of search space is increased
  - Makes training more time/resource consuming
  - Increases the likelihood of overfitting
- Problem is further compunded for color images
  - Each pixel in color image represented by 3 values (RGB color mode)
  - Since each pixel represented by 3 values, we say *channel* size is 3
  - Image represented by 64 x 64 x 3 = 12,288 values (rows x columns x channels)
  - Number of weights is now 12,288 x 500 = 6,144,000

Inspiration for CNN

- In 1959 Hubel & Wiesel did an experiment to understand how visual cortex of brain processes visual info
  - Recorded activity of neurons in visual cortex of a cat
  - While moving a bright line in front of the cat
- Some cells fired when bright line is shown at a particular angle/location
  - Called these *simple* cells
- Other cells fired when bright line was shown regardless of angle/location
  - Seemed to detect movement
  - Called these *complex* cells
- Seemed complex cells receive inputs from multiple simple cells
  - Have an hierarchical structure
- Hubel and Wiesel won Noble prize in 1981

Inspiration for CNN

- Inspired by complex/simple cells, Fukushima proposed *Neocognitron* (1980)
  - Hierarchical neural network used for handwritten Japanese character recognition
  - First CNN, had its own training algorithm
- In 1989, LeCun proposed CNN that was trained by backpropagation
- CNN got popular when outperformed other models at ImageNet Challenge
  - Competition in object classification/detection
  - On hundreds of object categories and millions of images
  - Run annually from 2010 to present
- Notable CNN architectures that won ImageNet challenge
  - AlexNet (2012), ZFNet (2013), GoogLeNet & VGG (2014), ResNet (2015)