Dauphine | PSL
UNIVERSITÉ PARIS

Deep Learning for Image Analysis

Mehyar MLAWEH

🏫 Master BDIA - Dauphine Tunis
PhD student - Université PSL

👨‍💻 Generative AI Engineer - Wealthy Technology

Resources: https://mehyarmlaweh.github.io/mehyar-mlaweh/teaching

Module Overview

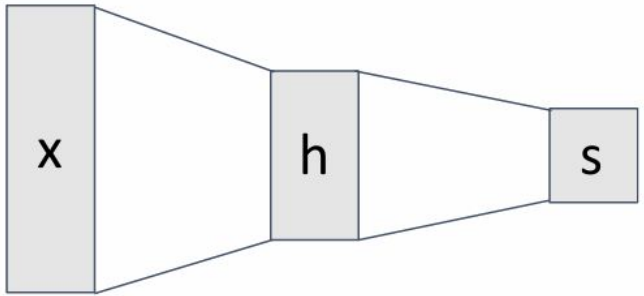
1. Computer Vision Foundations
2. CNNs
3. Transfer Learning
4. Vision Transformers
5. Object Detection
6. Segment Anything Model (SAM)
7. Final Project

Lecture 2 - Part 2:

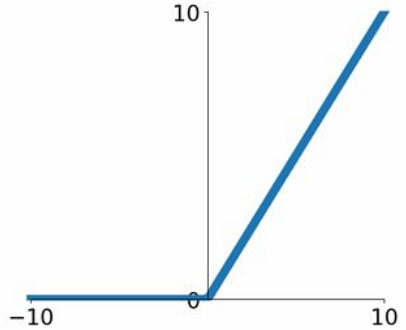
Convolutional neural networks

Last Time

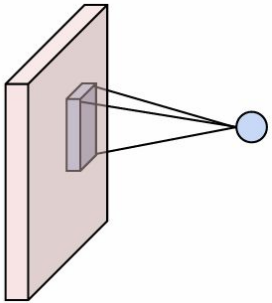
Fully-Connected Layers



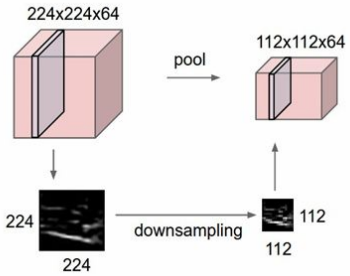
Activation Function



Convolution Layers



Pooling Layers

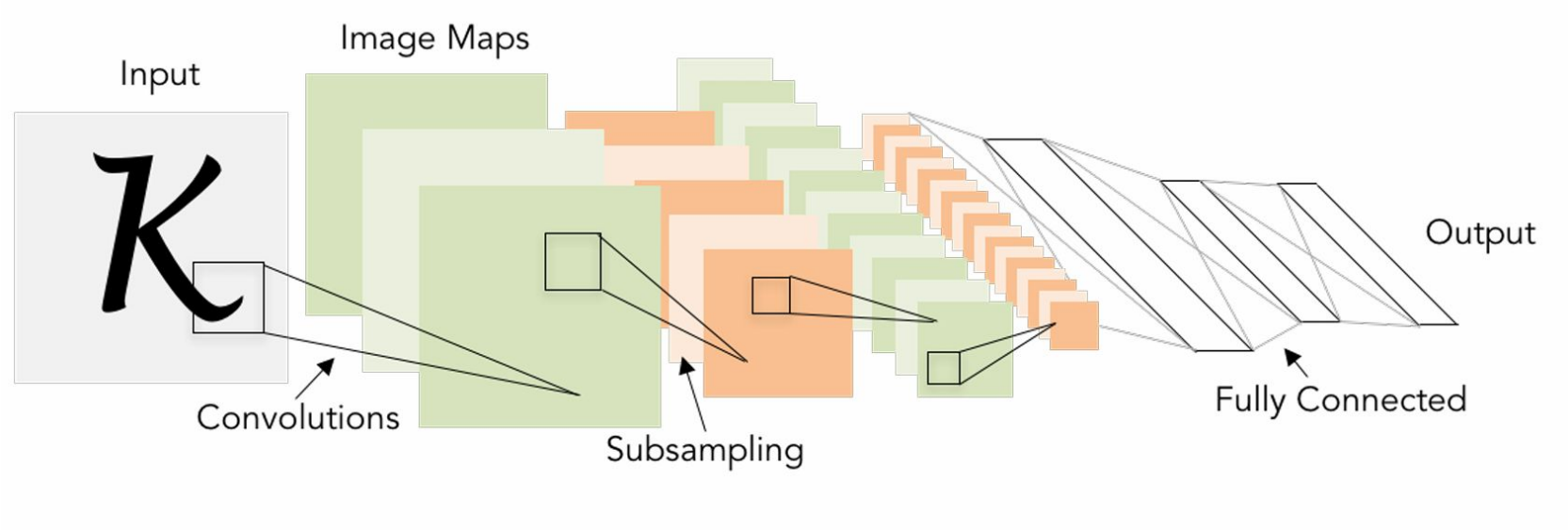


Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Last Time

Problem : **What is the right way to combine all these components ?**

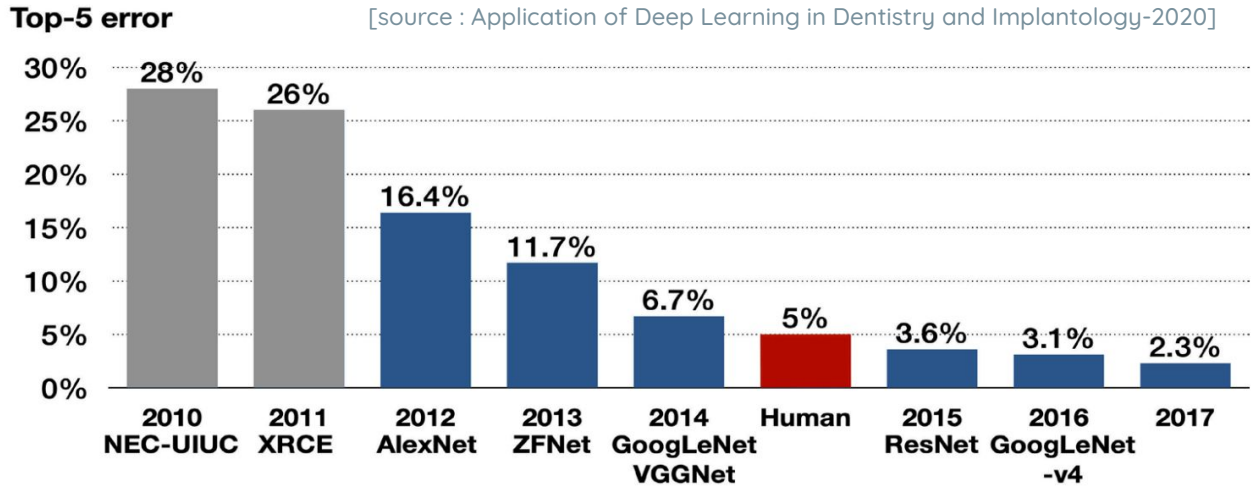


ImageNet Classification Challenge

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) uses a subset of the ImageNet dataset.

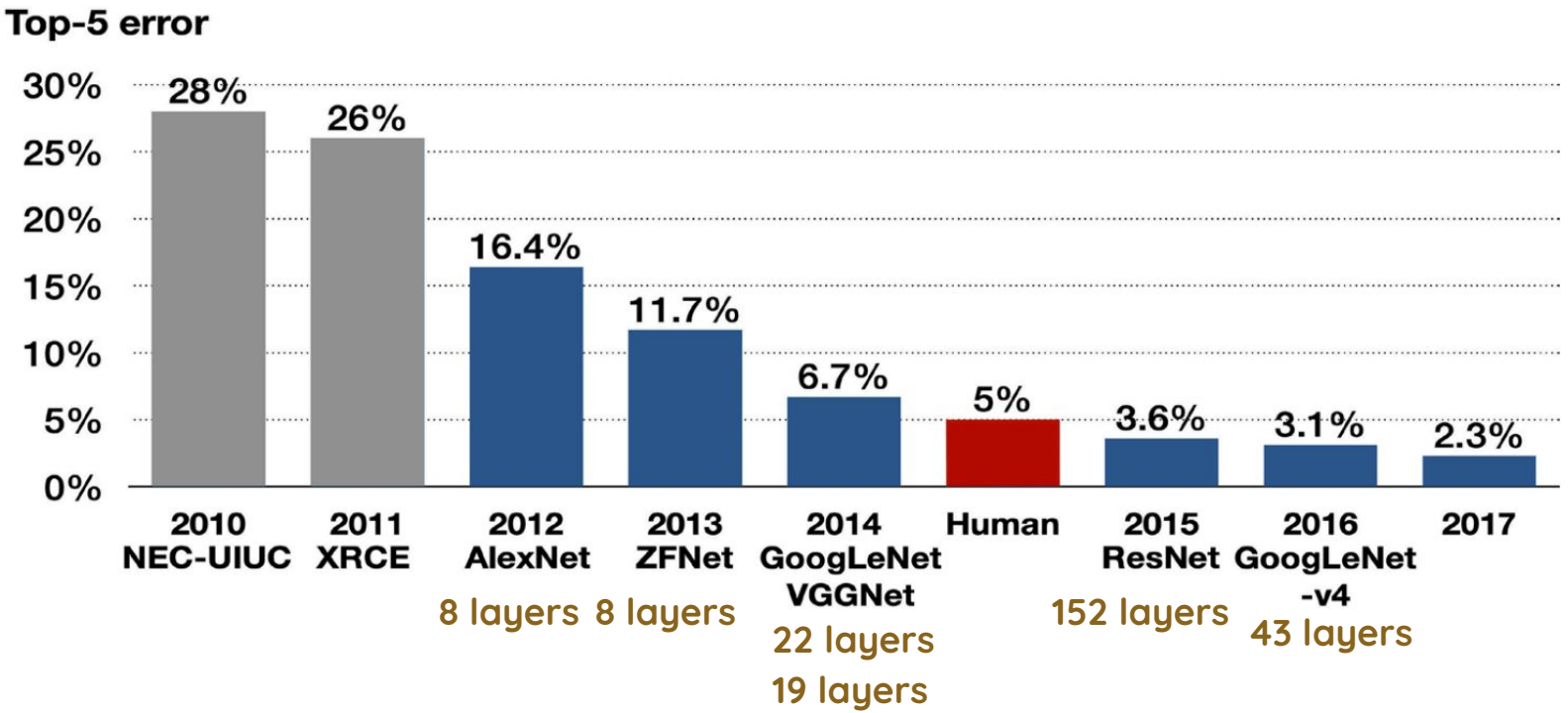
- ~1.2 million training images (from ~14 million total images in ImageNet)
- 1000 object classes (e.g., dog, airplane, keyboard)

ImageNet Classification Challenge



- Algorithms that won the ImageNet Challenge in 2010-2017.
- The top-5 error refers to the probability that all top-5 classifications proposed by the algorithm for the image are wrong.
- The algorithms with blue graph are convolutional neural network
- After 2017 the ImageNet competition stopped because deep networks had already surpassed human-level performance

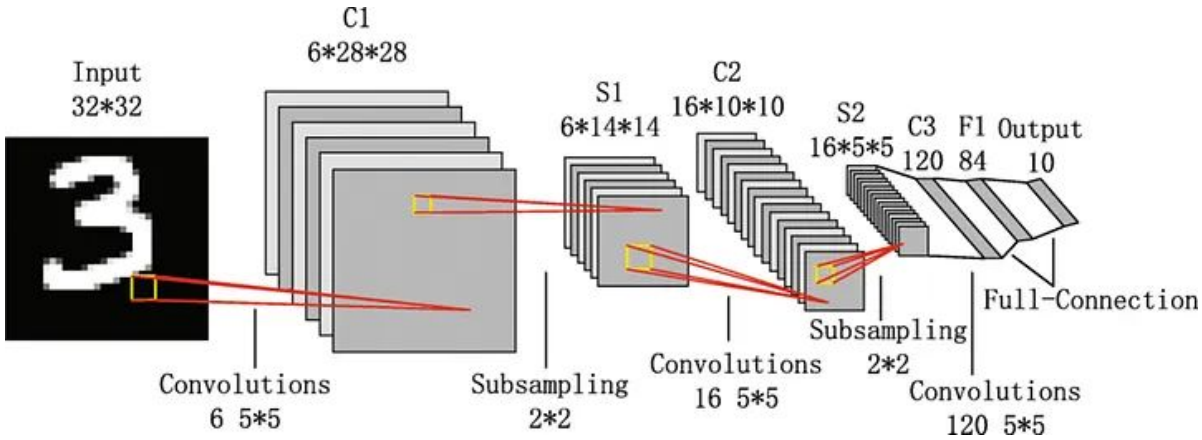
ImageNet Classification Challenge



[source : Application of Deep Learning in Dentistry and Implantology-2020]

LeNet-5 (Yann LeCun, 1998)

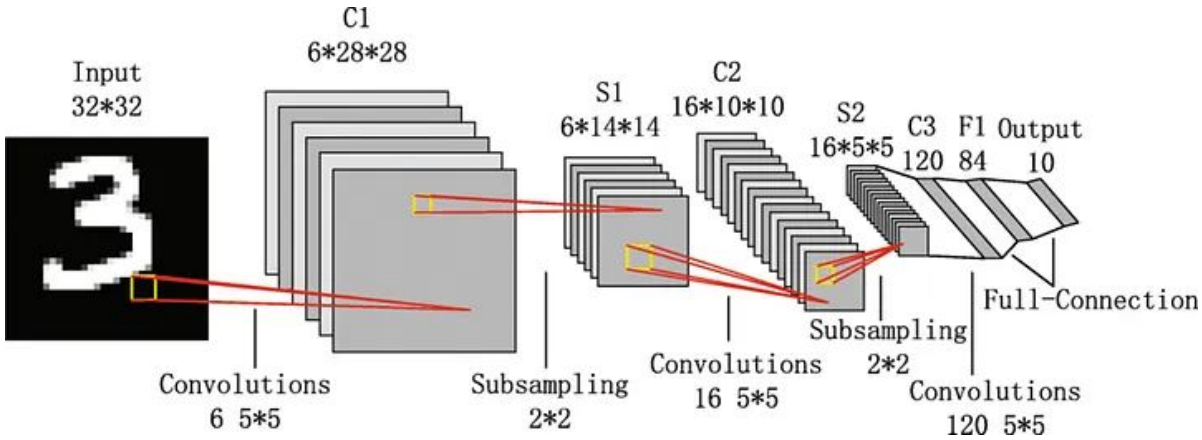
- First CNN architecture
- Designed for handwritten digit recognition
- Dataset: MNIST (28×28 grayscale digits)



LeNet-5 (Yann LeCun, 1998)

1. Input Layer

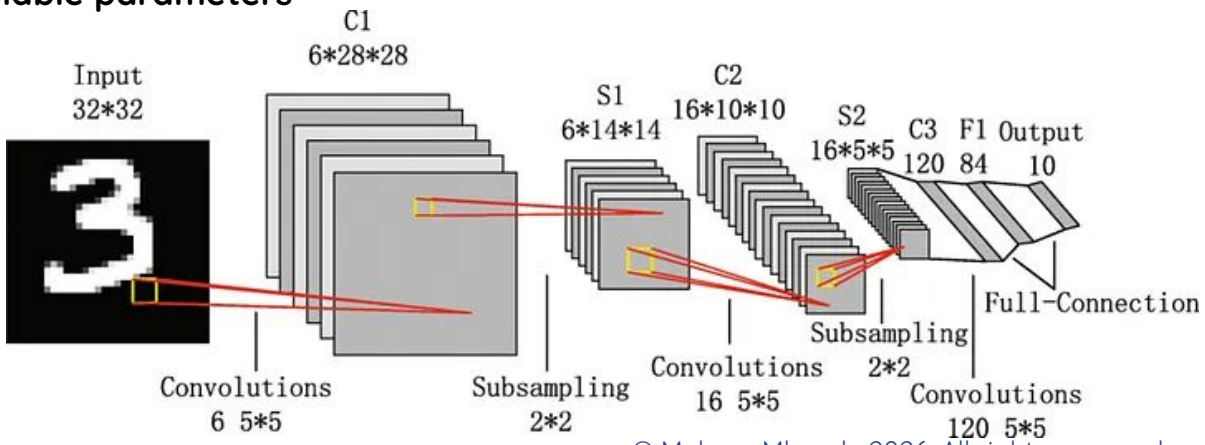
- Input Size: 32x32 pixels.
- MNIST digits are about 20×20, centered in 28×28 images
- Padding to 32×32 ensures important features (corners, stroke ends)



LeNet-5 (Yann LeCun, 1998)

2. Layer C1

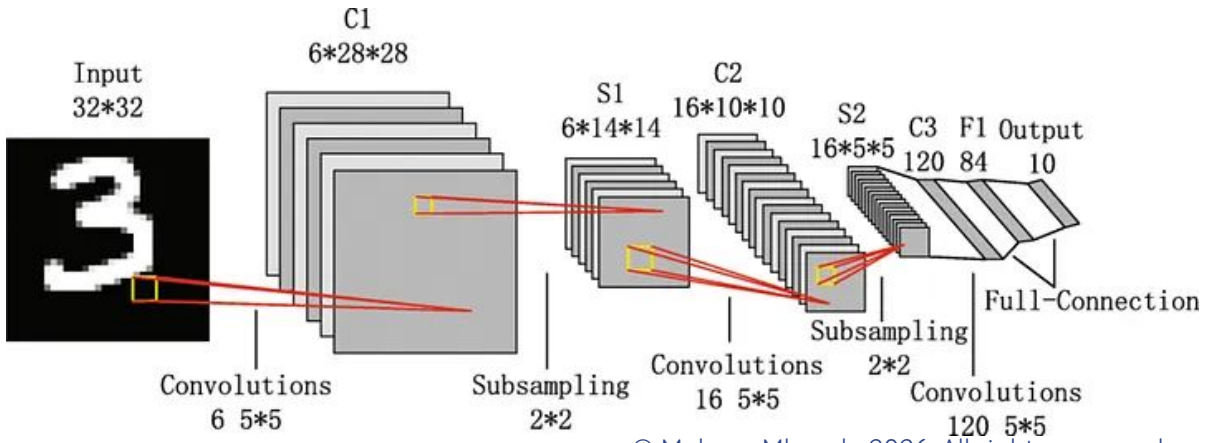
- Feature Maps: 6 feature maps.
- Number of filters : 6
- Filter Size : 5 x 5
- Padding : 0
- Stride : 1
- 5 x 5 x 1 x 6 + 6 = 156 trainable parameters



LeNet-5 (Yann LeCun, 1998)

3. Layer S1 (Pooling)

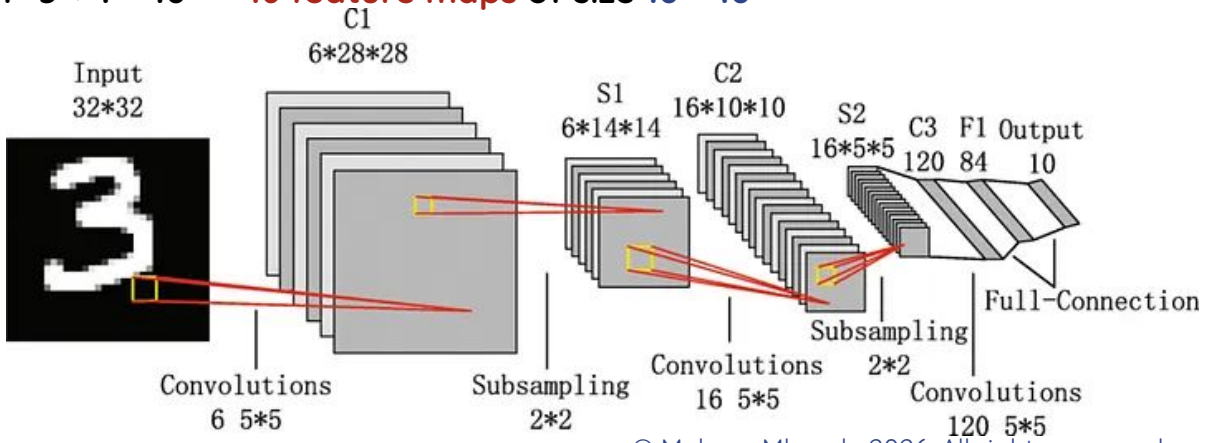
- 2×2 pooling
- Reduces the spatial size of feature maps
- Keeps the most important information
- output : 6 feature maps of size 14×14



LeNet-5 (Yann LeCun, 1998)

4. Layer C2

- Input From S1 : 6 feature maps of size 14 × 14
- Number of filters : 16
- Filter Size : 5 x 5
- Padding : 0
- Stride : 1
- Output Size : W−K + 1 = 14 -5 + 1 = 10 -> 16 feature maps of size 10 × 10

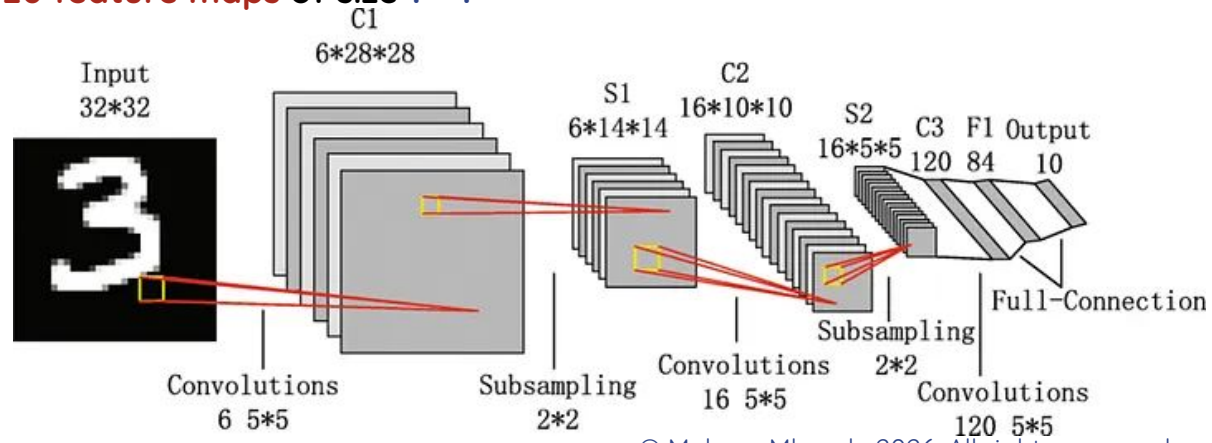


LeNet-5 (Yann LeCun, 1998)

5. Layer S2 : 16 X 5 X 5

6. Layer C3 :

- 120 filters
- kernel size = 5 × 5
- stride = 1
- padding = 0
- Output : W - K + 1 = 1 -> 120 feature maps of size 1 × 1



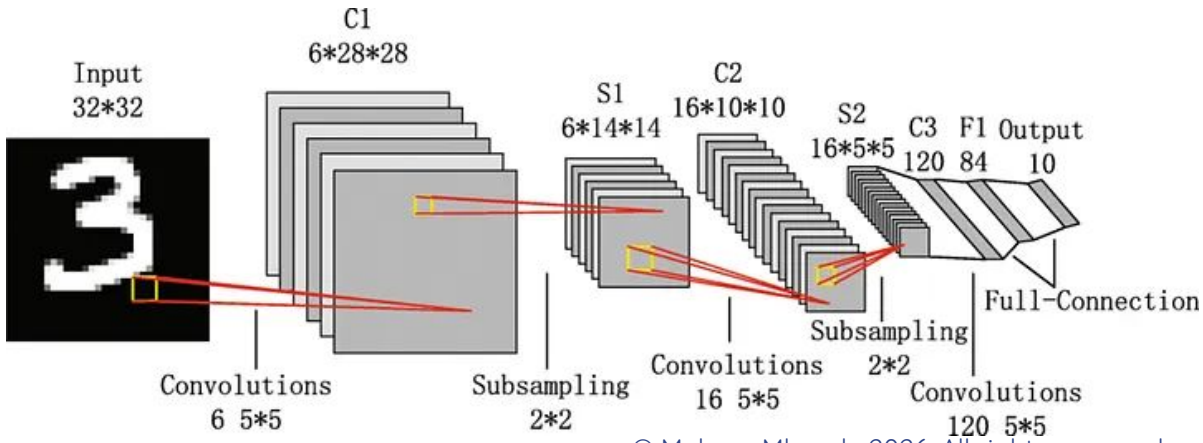
LeNet-5 (Yann LeCun, 1998)

7. Flatten:

- 120 × 1 × 1 -> 120

8. Layer F1:

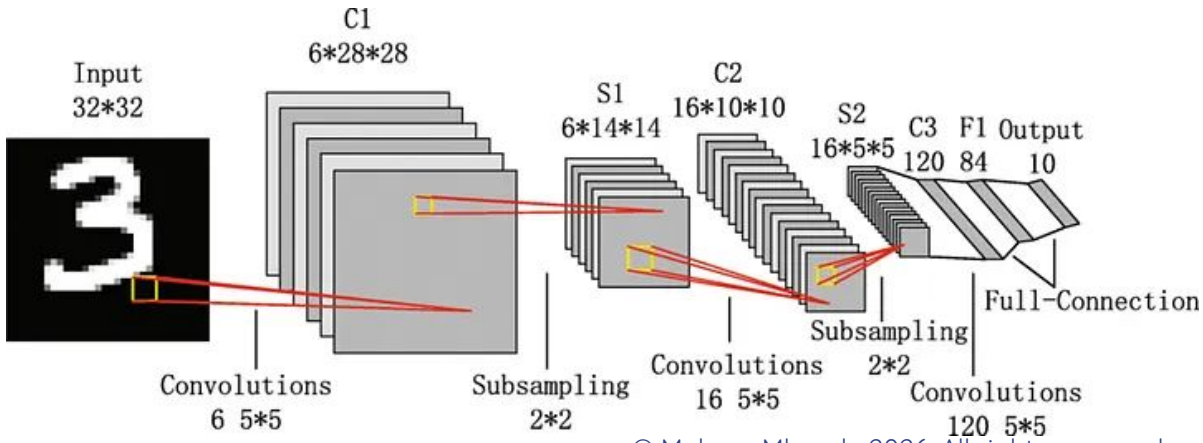
- Fully connected layer
- 84 neurons
- Trainable parameters : 120 x 84 + 84 = 10,164
- Output : 84



LeNet-5 (Yann LeCun, 1998)

9. Output Layer:

- Fully connected layer
- 10 neurons (digits 0–9)
- Trainable parameters : 84 x 10+ 10 = 850
- Output : 84



LeNet-5 (Yann LeCun, 1998)

Dauphine | PSL UNIVERSITÉ PARIS

Complete Architecture

Layer	#Filters Neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-		32x32x1	
Conv 1	6	5x5	1	28x28x6	tanh
Avg. pooling 1	-	2x2	2	14x14x6	
Conv 2	16	5x5	1	10x10x16	tanh
Avg. pooling 2	-	2x2	2	5x5x16	
Conv 3	120	5x5	1	120	tanh
FC1	-			84	tanh
FC2	-			10	softmax

LeNet-5 (Yann LeCun, 1998)

Why wouldn’t LeNet work for ImageNet?

LeNet-5 (Yann LeCun, 1998)

Why wouldn't LeNet work for ImageNet?

It was designed for:

- 28×28 grayscale digits
- 10 classes
- small dataset (MNIST)

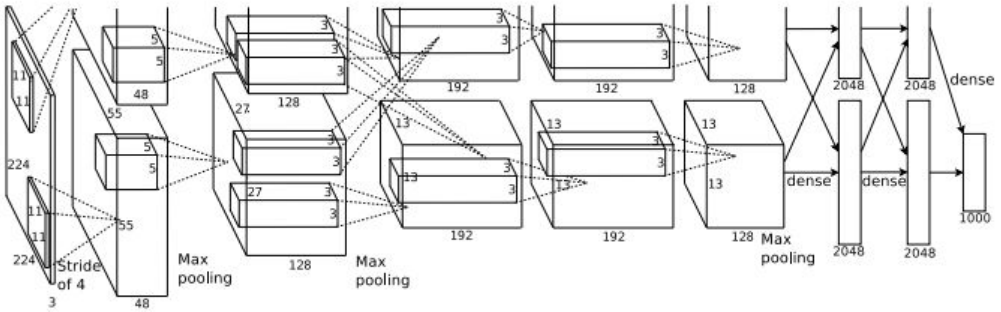
But ImageNet has:

- 224×224 RGB images
- 1000 classes
- 1.2 million images

GPUs were not available

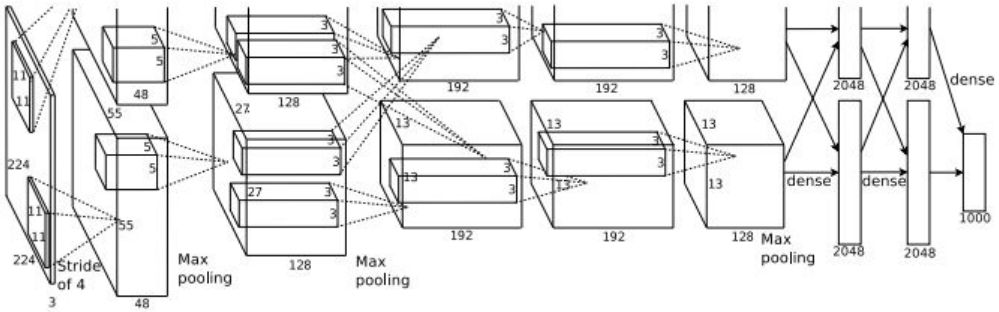
AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

- The architecture that popularized CNN
- Similar to LeNet but more deeper
- Winner of imageNet large-scale visual recognition challenge (ILSVRC) in 2012



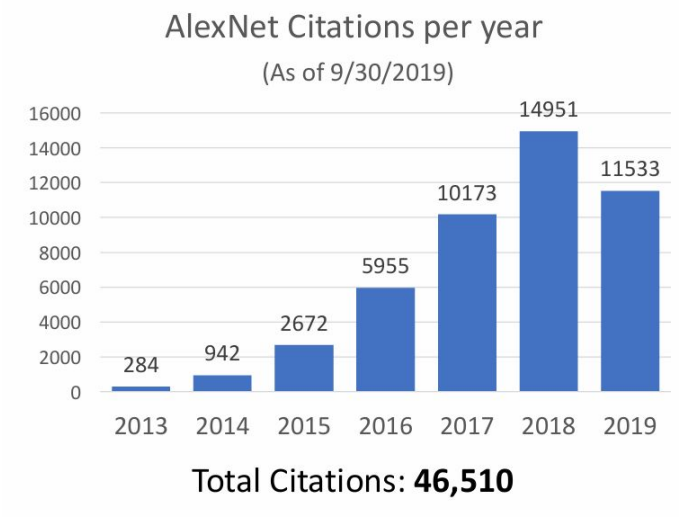
AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

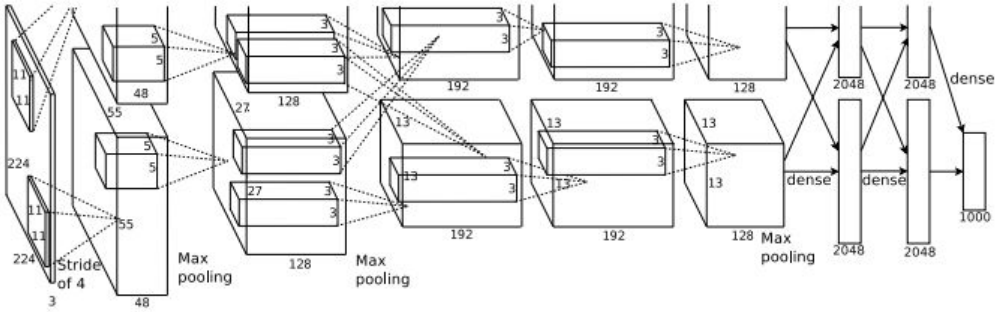
- 227 x 227 inputs
- 5 Convolutional layers
- Max pooling
- 3 fully-connected layers
- ReLU non-linearities



AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

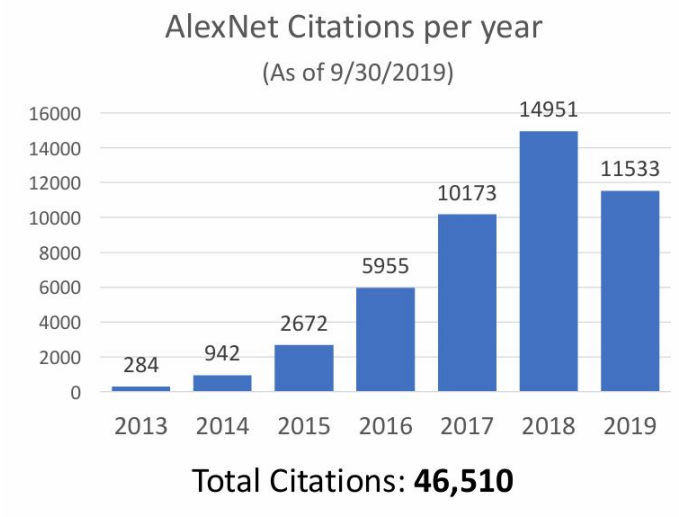
- 227 x 227 inputs
- 5 Convolutional layers
- Max pooling
- 3 fully-connected layers
- ReLU non-linearities

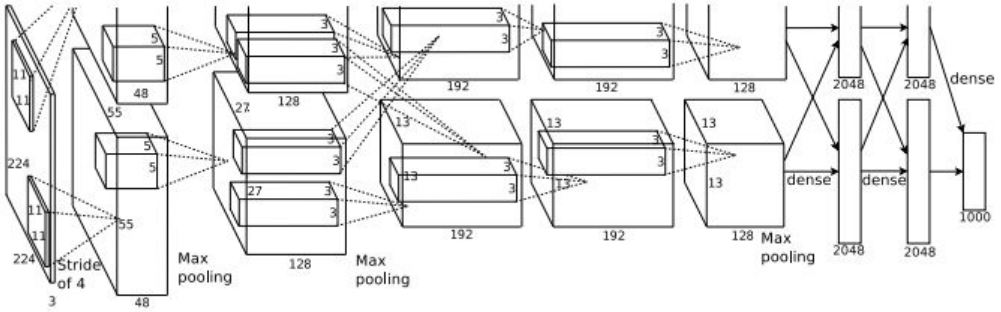




AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

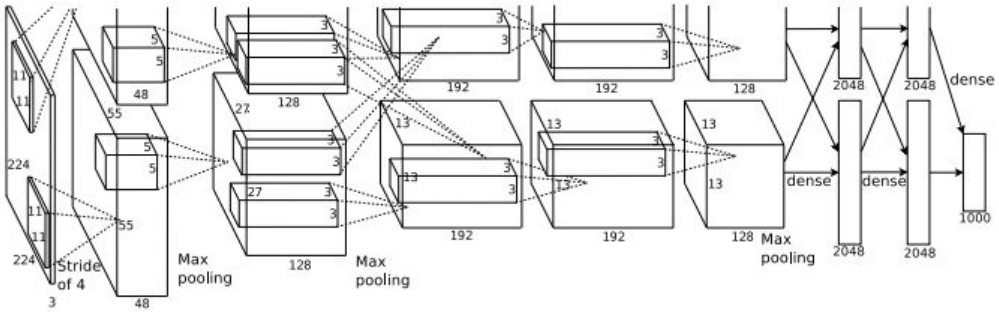
- 227 x 227 inputs
- 5 Convolutional layers
- Max pooling
- 3 fully-connected layers
- ReLU non-linearities





AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

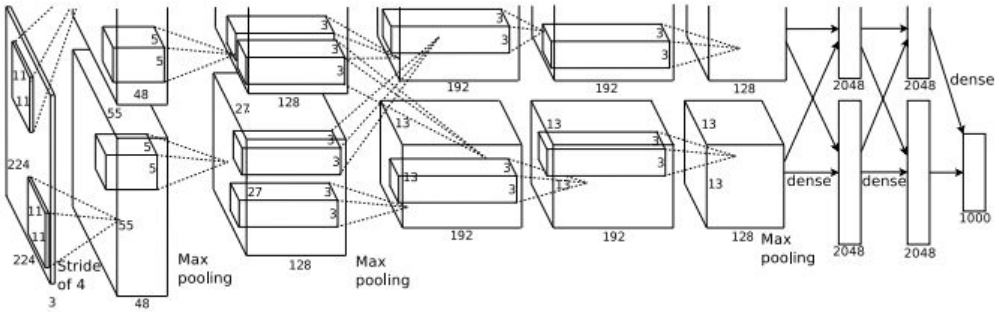
	Input size		Layer				Output size	
Layer	C	H / W	filters	kernel	stride	pad	C	H / W
conv1	3	227	64	11	4	2	?	



AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

	Input size		Layer				Output size	
Layer	C	H / W	filters	kernel	stride	pad	C	H / W
conv1	3	227	64	11	4	2	?	

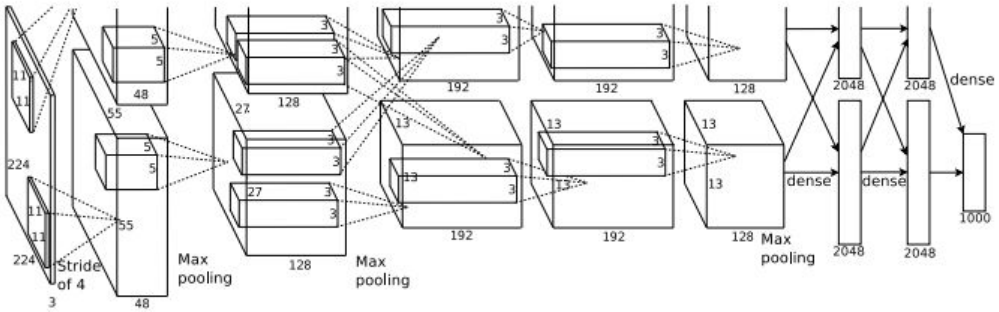
Recall: Output channels = number of filters



AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

	Input size		Layer				Output size	
Layer	C	H / W	filters	kernel	stride	pad	C	H / W
conv1	3	227	64	11	4	2	?	

Recall: W' = (W −K + 2P) / S + 1



AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

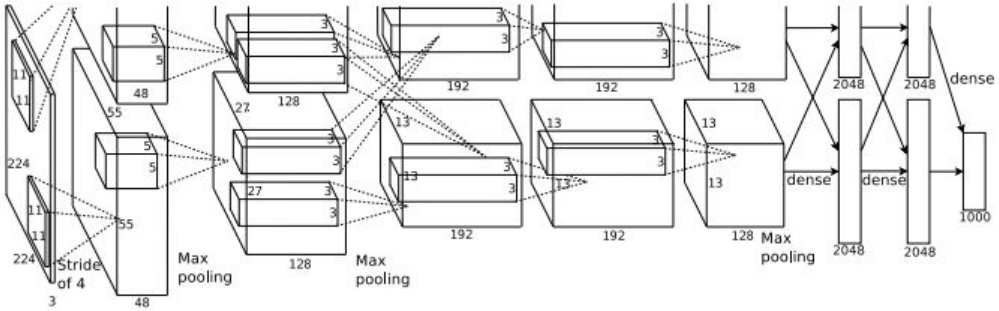
	Input size		Layer				Output size	
Layer	C	H / W	filters	kernel	stride	pad	C	H / W
conv1	3	227	64	11	4	2	?	

Recall: W' = (W −K + 2P) / S + 1

= 227 − 11 + 2*2) / 4 + 1

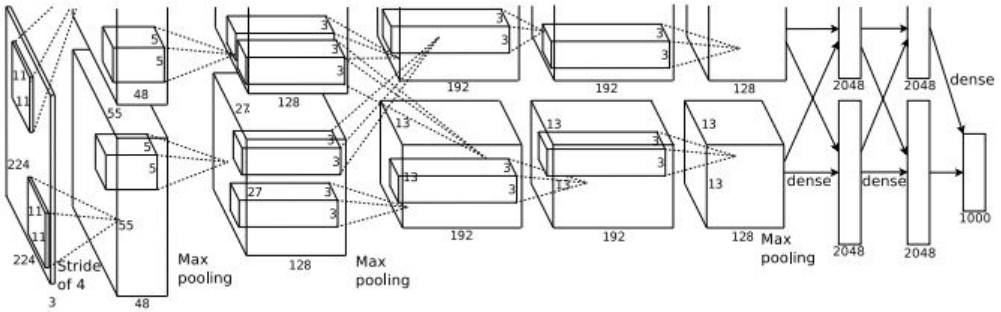
= 220/4 + 1

= 56



AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

	Input size		Layer				Output size		
Layer	C	H / W	filters	kernel	stride	pad	C	H / W	params (k)
conv1	3	227	64	11	4	2	64	56	?



AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

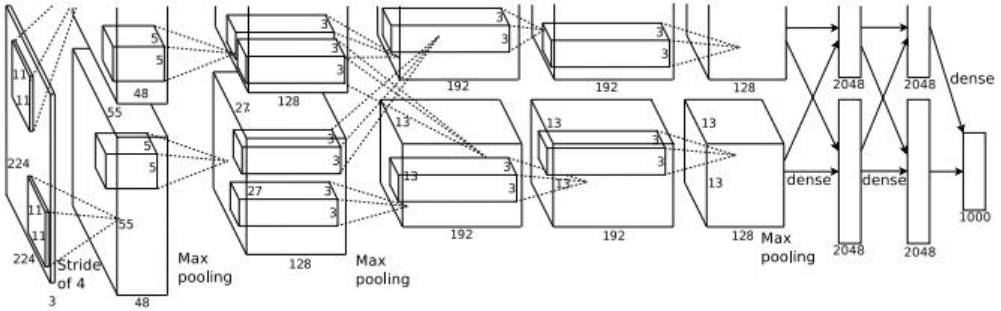
	Input size		Layer				Output size		
Layer	C	H / W	filters	kernel	stride	pad	C	H / W	params (k)
conv1	3	227	64	11	4	2	64	56	?

Weight shape = C_{out} x C_{in} x K x K

= 64 x 3 x 11 x 11

Bias Shape = C_{out} = 64

Number of parameters = 64 x 3 x 11 x 11 + 64 = 23,296



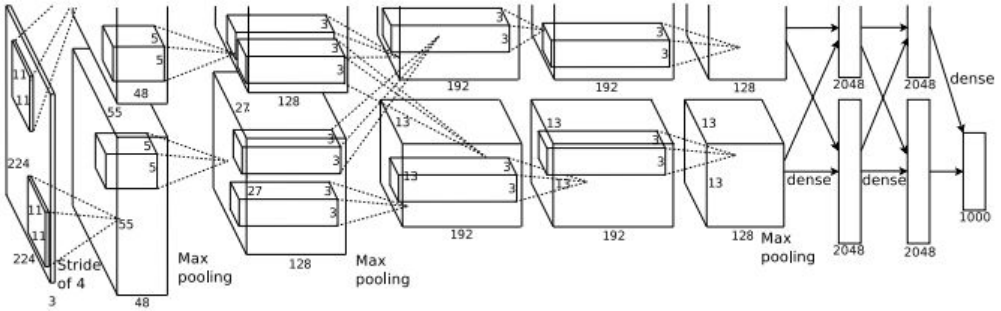
AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

	Input size		Layer				Output size		
Layer	C	H / W	filters	kernel	stride	pad	C	H / W	params (k)
conv1	3	227	64	11	4	2	64	56	23
pool1	64	56		3	2	0	64	27	0
conv2	64	27	192	5	1	2	192	27	307
pool2	192	27		3	2	0	192	13	0
conv3	192	13	384	3	1	1	384	13	664
conv4	384	13	256	3	1	1	256	13	885
conv5	256	13	256	3	1	1	256	13	590
pool5	256	13		3	2	0	256	6	0
flatten	256	6					9216		0

Flatten output size = C_{in} x H x W

= 256 x 6 x 6

= 9216



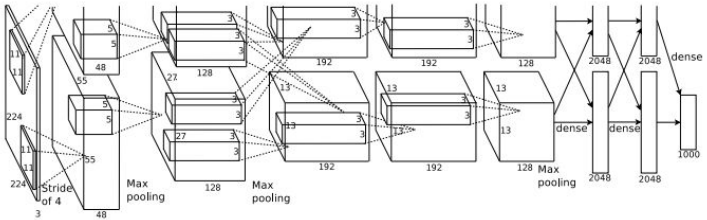
AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

	Input size		Layer				Output size		
Layer	C	H / W	filters	kernel	stride	pad	C	H / W	params (k)
conv1	3	227	64	11	4	2	64	56	23
pool1	64	56		3	2	0	64	27	0
conv2	64	27	192	5	1	2	192	27	307
pool2	192	27		3	2	0	192	13	0
conv3	192	13	384	3	1	1	384	13	664
conv4	384	13	256	3	1	1	256	13	885
conv5	256	13	256	3	1	1	256	13	590
pool5	256	13		3	2	0	256	6	0
flatten	256	6					9216		0
fc6	9216		4096				4096		37,749

FC params = $C_{in} \times C_{out} + C_{out}$

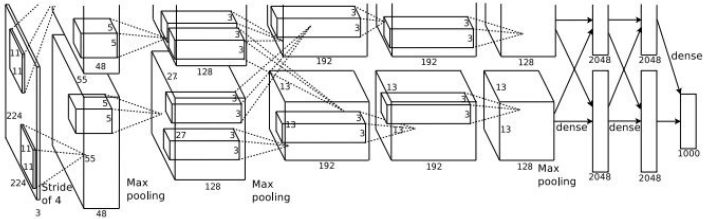
= 9216 x 4096 + 4096

= 37,748,832



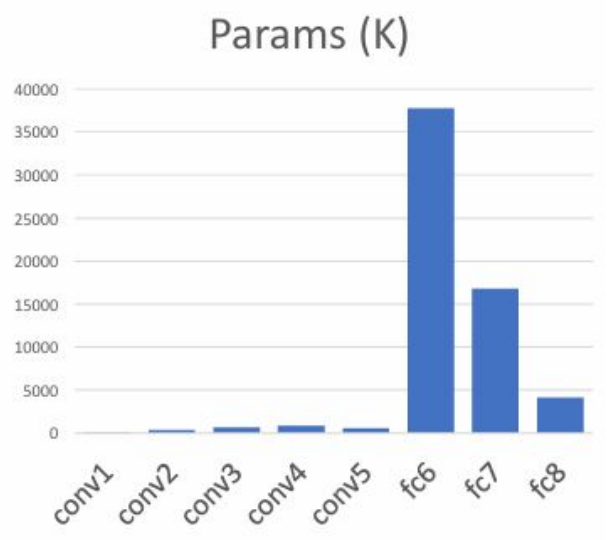
AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

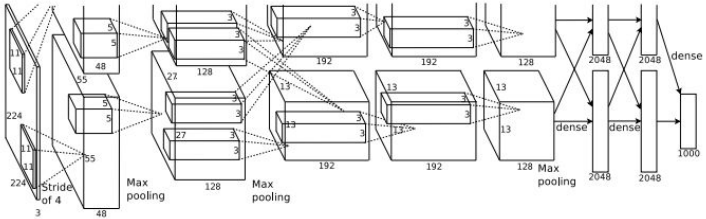
	Input size		Layer				Output size		
Layer	C	H / W	filters	kernel	stride	pad	C	H / W	params (k)
conv1	3	227	64	11	4	2	64	56	23
pool1	64	56		3	2	0	64	27	0
conv2	64	27	192	5	1	2	192	27	307
pool2	192	27		3	2	0	192	13	0
conv3	192	13	384	3	1	1	384	13	664
conv4	384	13	256	3	1	1	256	13	885
conv5	256	13	256	3	1	1	256	13	590
pool5	256	13		3	2	0	256	6	0
flatten	256	6					9216		0
fc6	9216		4096				4096		37,749
fc7	4096		4096				4096		16,777
fc8	4096		1000				1000		4,096



AlexNet-5 (Krizhevsky, Sutskever, Hinton, 2012)

Nearly all parameters are in the fully-connected layers





ZFNet: A bigger AlexNet (2013)

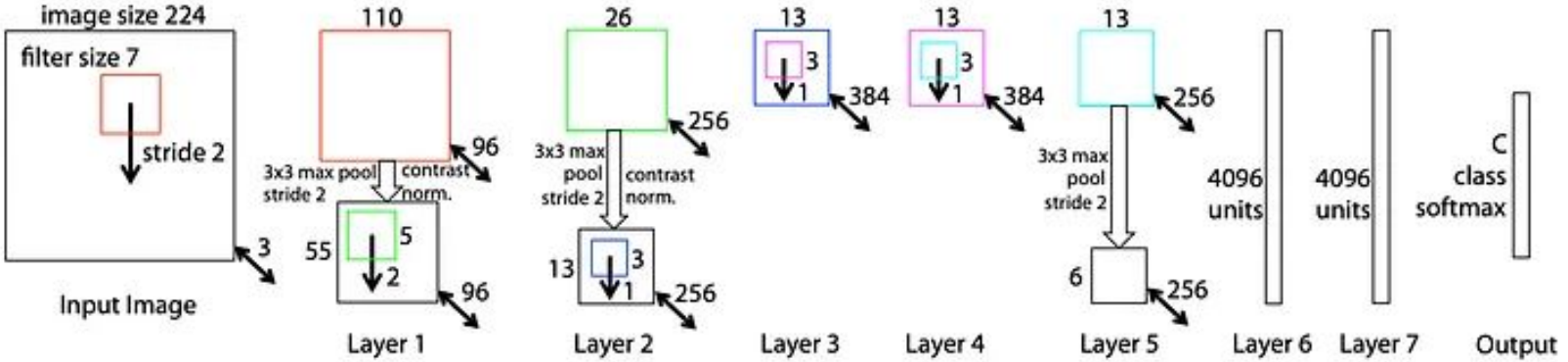
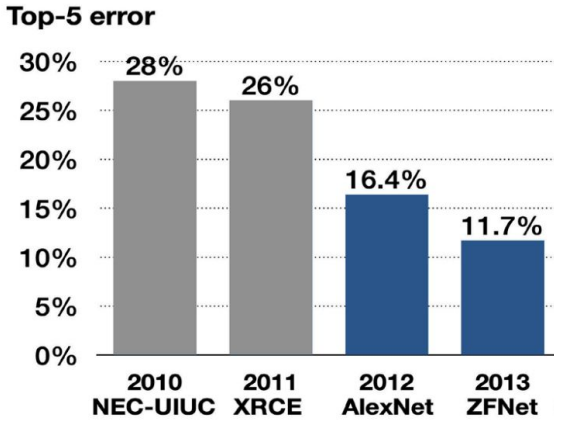


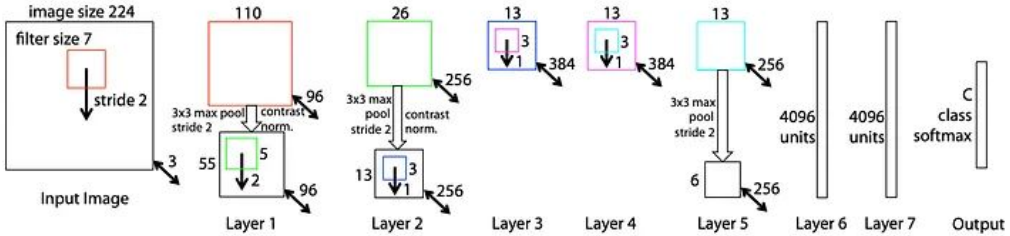
Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

ZFNet: A bigger AlexNet (2013)

AlexNET but :

- CONV 1 : change from (11x11 stride 4) to (7x7 stride 2)
- CONV 3,4,5 : instead of 384,384,256 filters use 512, 1024, 512

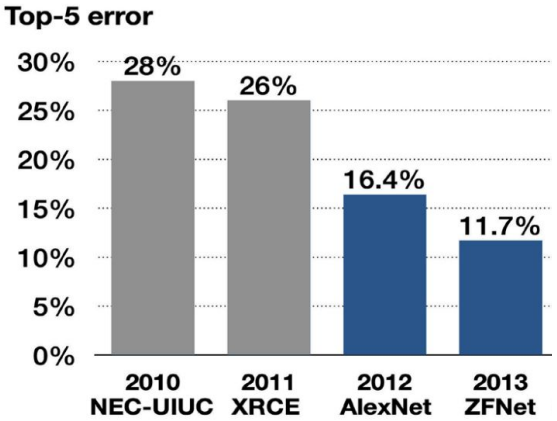




ZFNet: A bigger AlexNet (2013)

AlexNET but :

- CONV 1 : change from (11x11 stride 4) to (7x7 stride 2)
- CONV 3,4,5 : instead of 384,384,256 filters use 512, 1024, 512

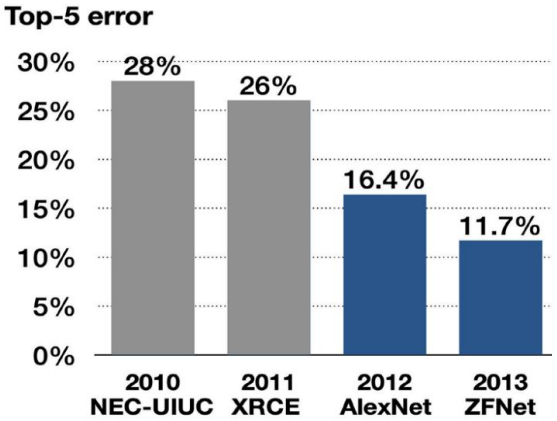


Why do you think these changes improved ImageNet accuracy?

ZFNet: A bigger AlexNet (2013)

AlexNET but :

- CONV 1 : change from (11x11 stride 4) to (7x7 stride 2)
- CONV 3,4,5 : instead of 384,384,256 filters use 512, 1024, 512

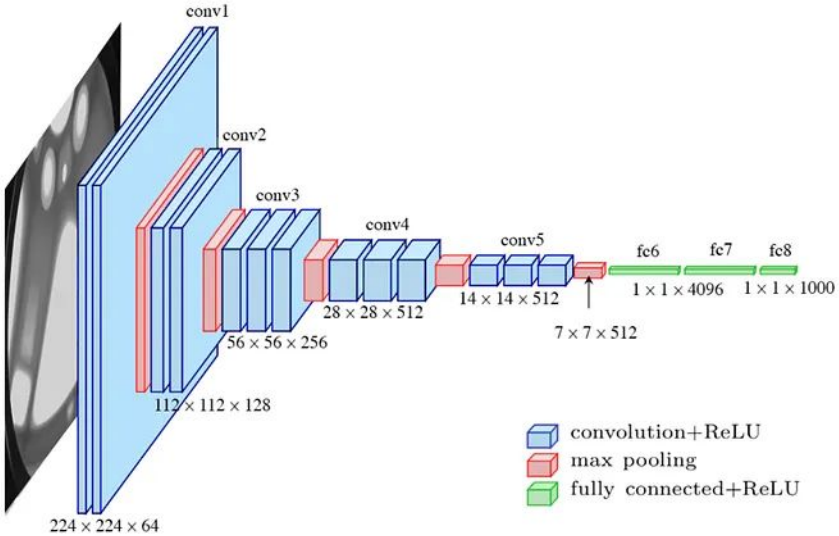


Why do you think these changes improved ImageNet accuracy?

- Smaller stride (2 instead of 4) preserves more spatial information in early layers.
- 7×7 kernels capture more visual details than 11×11.
- More filters → higher model capacity allows learning richer and more diverse features.

VGGNet (Karen Simonyan, Andrew Zisserman,2014)

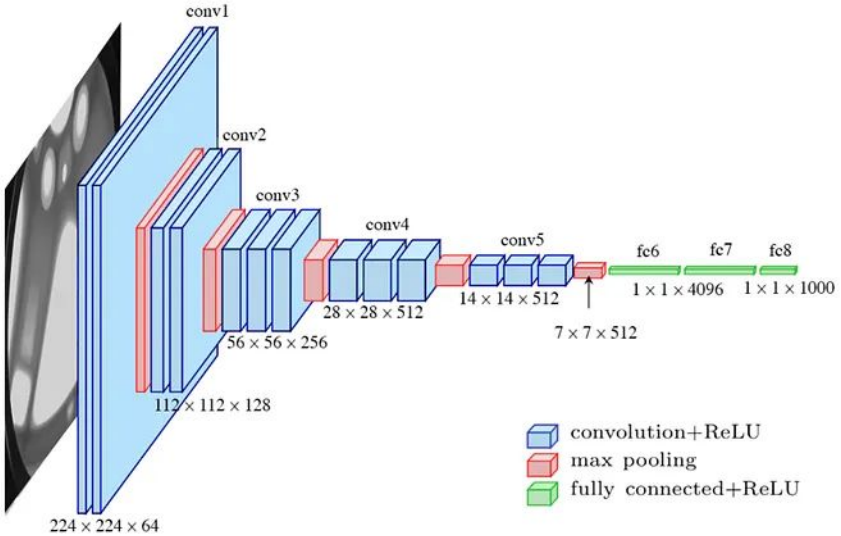
After AlexNet and ZFNet, researchers observed that deeper networks often achieve better performance on complex datasets like ImageNet.



VGGNet (Karen Simonyan, Andrew Zisserman,2014)

After AlexNet and ZFNet, researchers observed that deeper networks often achieve better performance on complex datasets like ImageNet.

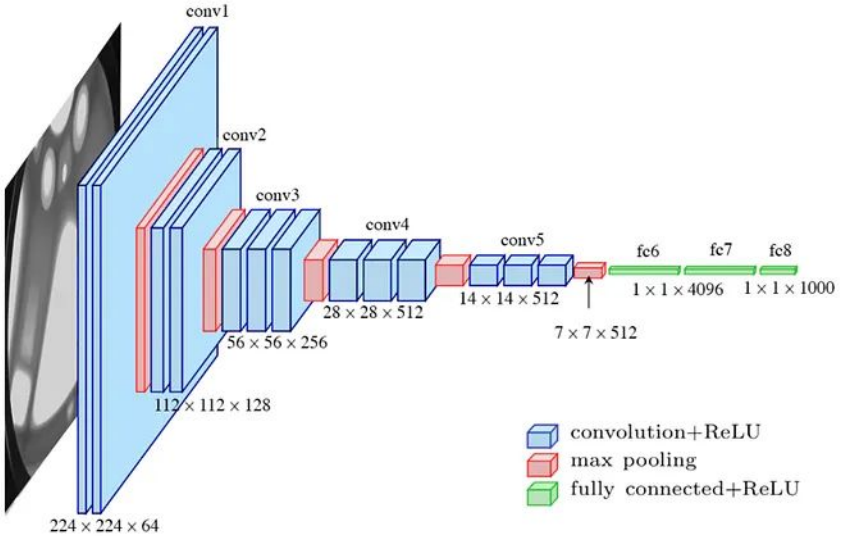
Can we significantly increase the depth of CNNs while keeping the architecture simple and effective?



VGGNet (Karen Simonyan, Andrew Zisserman,2014)

After AlexNet and ZFNet, researchers observed that deeper networks often achieve better performance on complex datasets like ImageNet.

Can we significantly increase the depth of CNNs while keeping the architecture simple and effective?

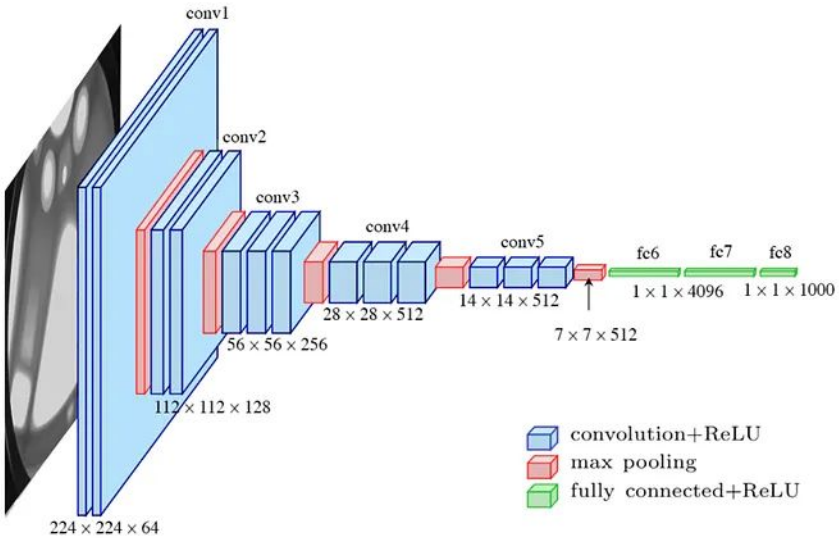


VGGNet (Karen Simonyan, Andrew Zisserman,2014)

After AlexNet and ZFNet, researchers observed that deeper networks often achieve better performance on complex datasets like ImageNet.

Can we significantly increase the depth of CNNs while keeping the architecture simple and effective?

VGGNet answered this question by proposing a very deep convolutional network built almost entirely from small 3×3 convolution filters.

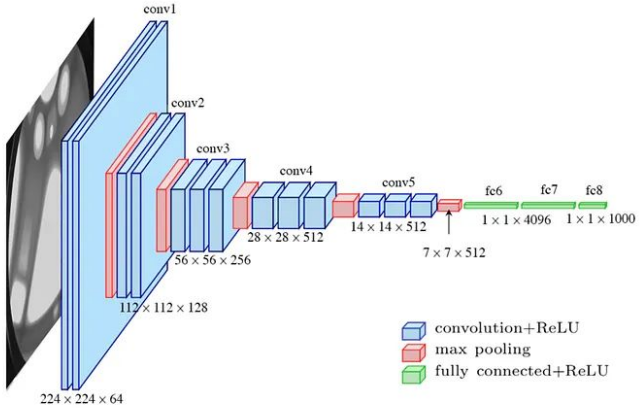


VGGNet (Karen Simonyan, Andrew Zisserman,2014)

Dauphine | PSL UNIVERSITÉ PARIS

Design Rules:

- All convolution layers: 3×3 kernel, stride = 1, padding = 1
- All Pooling Layers : MaxPool 2x2, stride = 2
- After pool : #channels doubles



VGGNet (Karen Simonyan, Andrew Zisserman,2014)

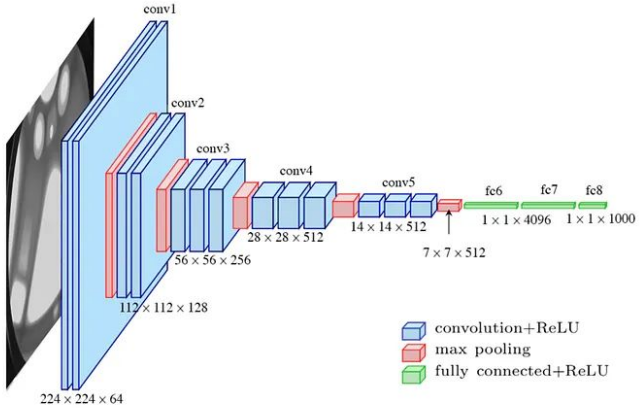
Design Rules:

- All convolution layers: 3×3 kernel, stride = 1, padding = 1
- All Pooling Layers : MaxPool 2x2, stride = 2
- After pool : #channels doubles

Pooling reduces spatial information, which means Increasing channels lets the network learn more feature types.

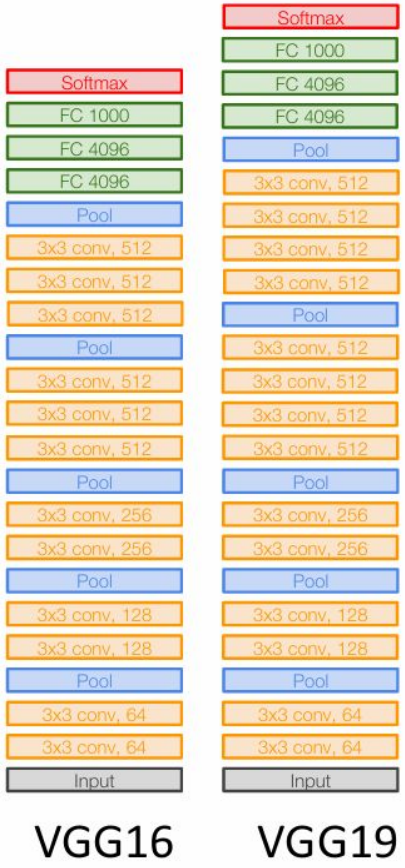
Example :

- Input : 224 × 224 × 3
- After the first conv block: 224 x 224 x 64
- After first max pool : 112 x 112 x 64
- After the first conv block: 112 x 112 x 128



VGGNet (Karen Simonyan, Andrew Zisserman,2014)

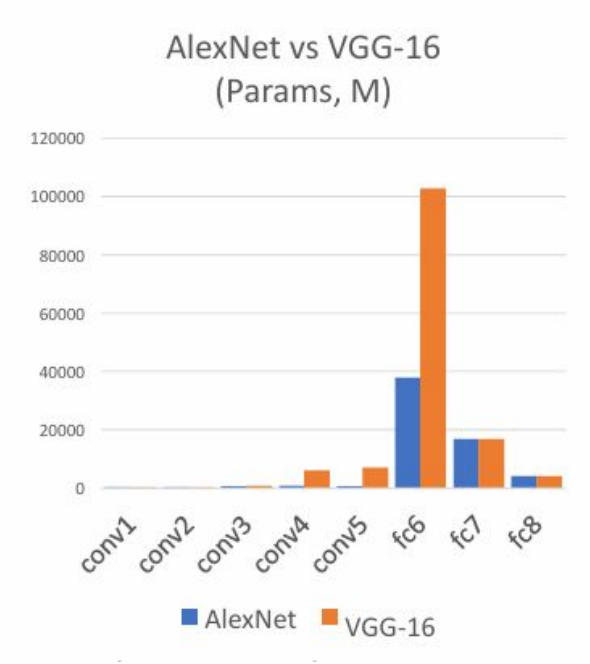
Model	# Layers
VGG-16	13 conv + 3 FC
VGG-19	16 conv + 3 FC

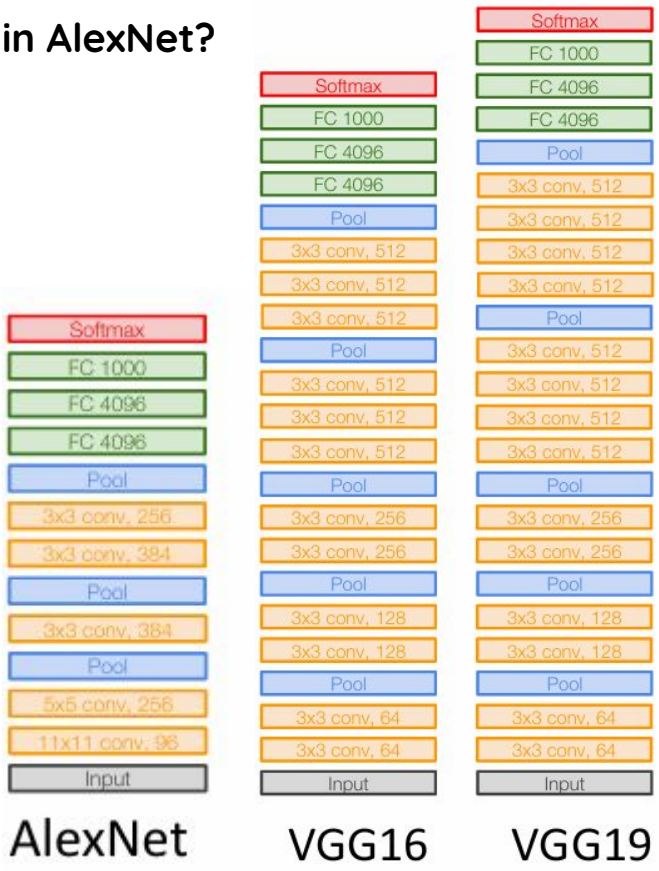


VGGNet (Karen Simonyan, Andrew Zisserman,2014)

Dauphine | PSL UNIVERSITÉ PARIS

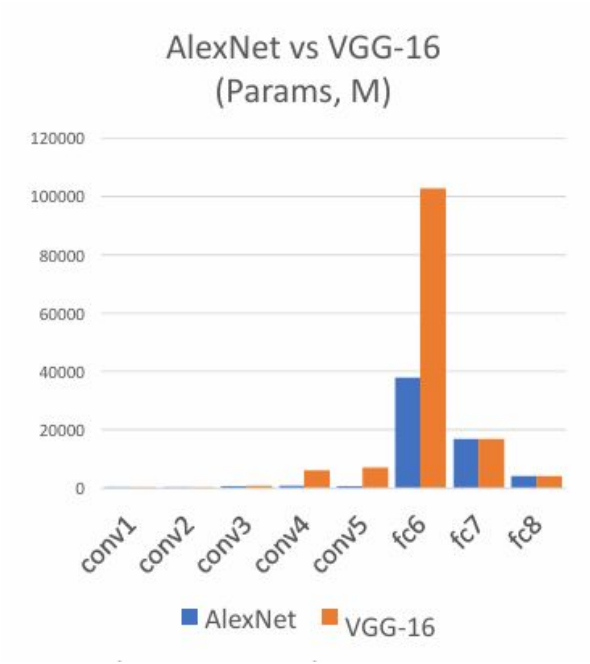
why does FC6 in VGG have many more parameters than in AlexNet?

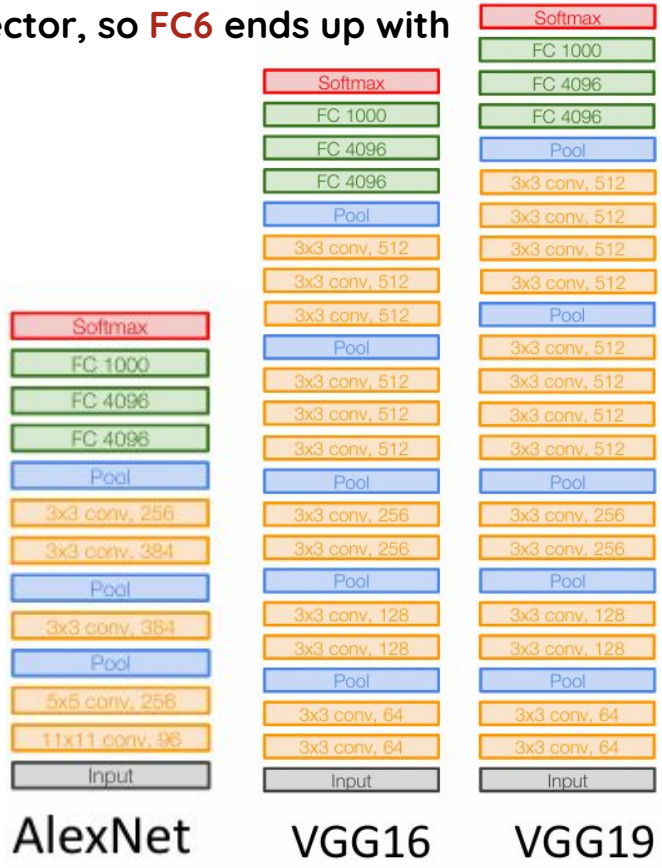




VGGNet (Karen Simonyan, Andrew Zisserman,2014)

Doubling channels in VGG increases the flattened feature vector, so **FC6** ends up with many more parameters





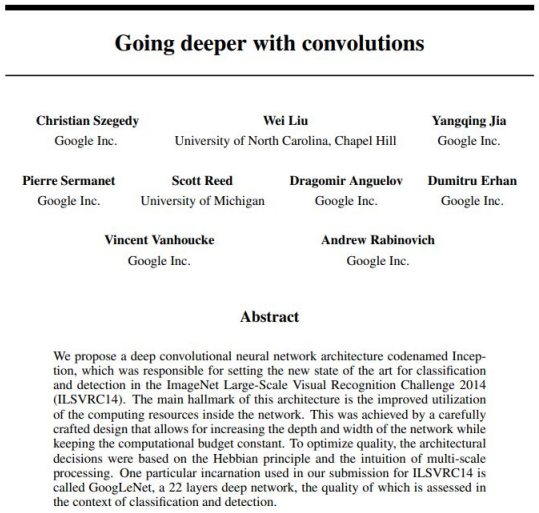
GoogLeNet(Szegedy et al, 2014)

While VGG improved accuracy by increasing depth, it had a major drawback:

- Very large number of parameters (~138M)
- High computational cost
- Risk of overfitting

Solution:

GoogLeNet ➡ Introduce the Inception module to build deeper but computationally efficient networks.



Going deeper with convolutions

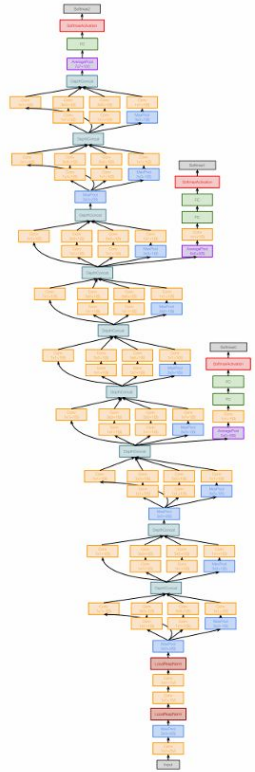
Christian Szegedy Google Inc.
Wei Liu University of North Carolina, Chapel Hill
Yangqing Jia Google Inc.
Pierre Sermanet Google Inc.
Scott Reed University of Michigan
Dragomir Anguelov Google Inc.
Dumitru Erhan Google Inc.
Vincent Vanhoucke Google Inc.
Andrew Rabinovich Google Inc.

Abstract

We propose a deep convolutional neural network architecture codenamed Inception, which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. This was achieved by a carefully crafted design that allows for increasing the depth and width of the network while keeping the computational budget constant. To optimize quality, the architectural decisions were based on the Hebbian principle and the intuition of multi-scale processing. One particular incarnation used in our submission for ILSVRC14 is called GoogLeNet, a 22 layers deep network, the quality of which is assessed in the context of classification and detection.

GoogLeNet- InceptionV1 (2014)

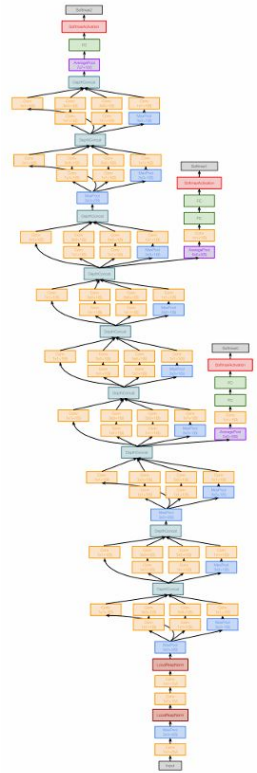
- Winner of ImageNet 2014
- 22 layers deep
- Introduced the Inception module
- Uses 1×1 convolutions for dimensionality reduction
- No large fully connected layers
- Much fewer parameters (~5M) vs VGG (~138M)



GoogLeNet- Inception Modules

Small details → small filters
Large objects → large filters

Instead of deciding whether to use 1×1, 3×3, or 5×5 filters, the Inception module uses all of them in parallel.

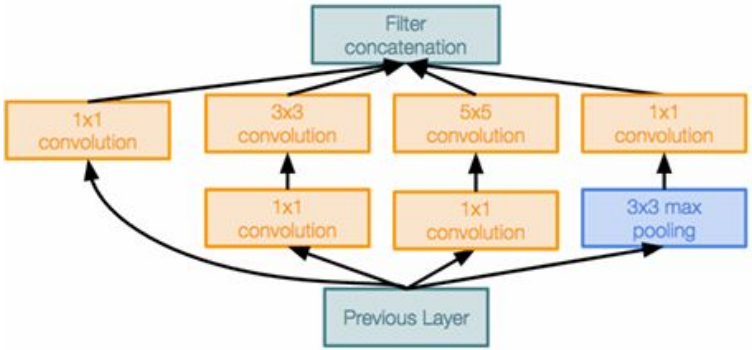


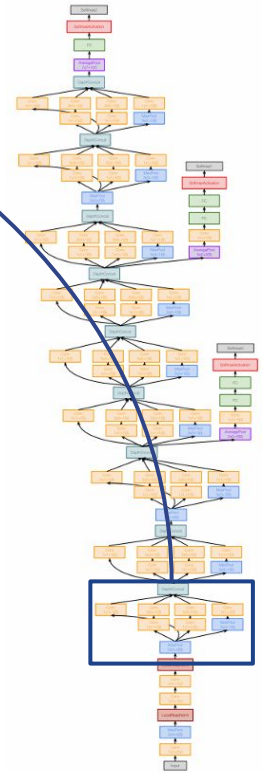
GoogLeNet- Inception Modules

Small details → small filters
Large objects → large filters

Structure of Inception module:

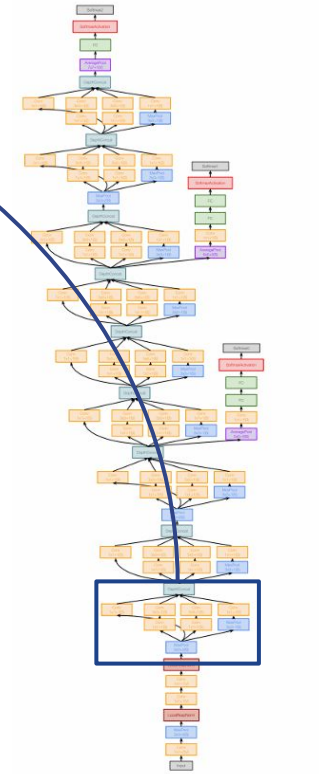
- Instead of deciding whether to use 1×1, 3×3, or 5×5 filters, the Inception uses all of them in parallel.





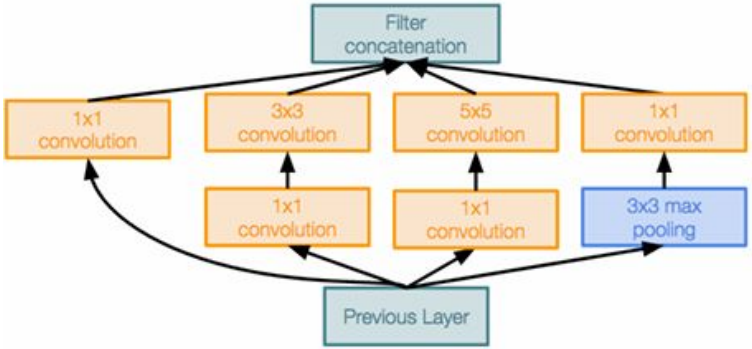
GoogLeNet- Inception Modules

Small details → small filters
Large objects → large filters



Structure of Inception module:

- Instead of deciding whether to use 1×1, 3×3, or 5×5 filters, the Inception uses all of them in parallel.
- Uses 1x1 'Bottleneck' layers to reduce channel dimensions before expensive conv



GoogLeNet- Inception Modules

Exemple :

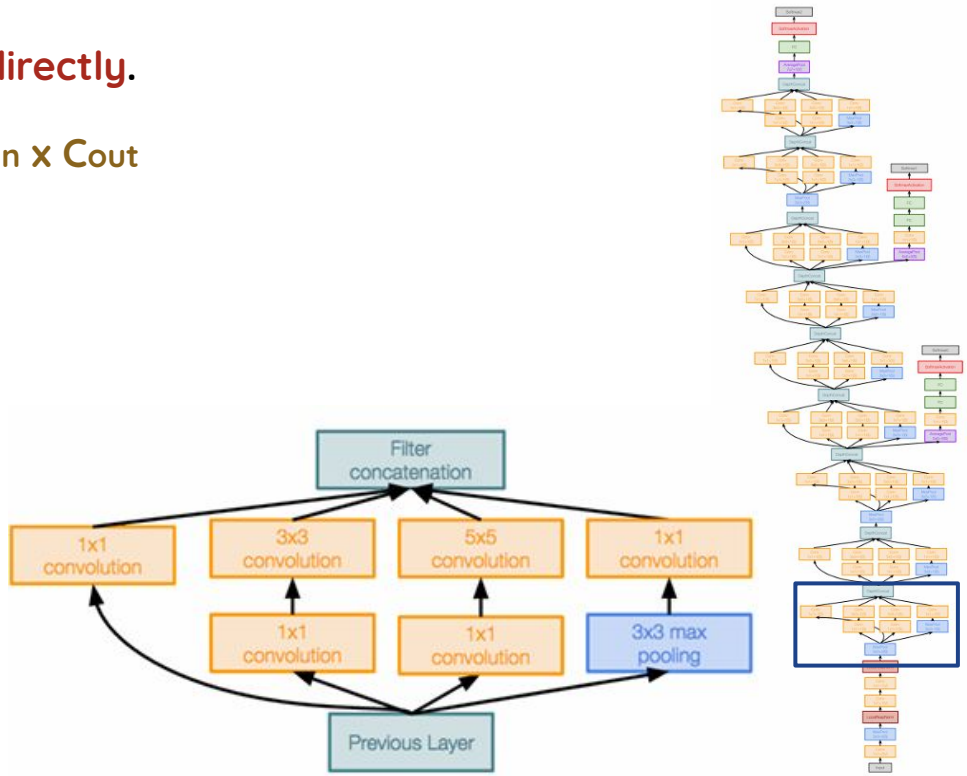
Suppose we apply a 5×5 convolution directly.

Cost formula : parameters = K x H x Cin x Cout

Input channels = 192
Output filters = 32

5 × 5 × 192 × 32 = 153,600 parameters

Very expensive.



GoogLeNet- Inception Modules

Exemple :

With 1x1 Bottleneck

Cost formula : parameters = K x H x Cin x Cout

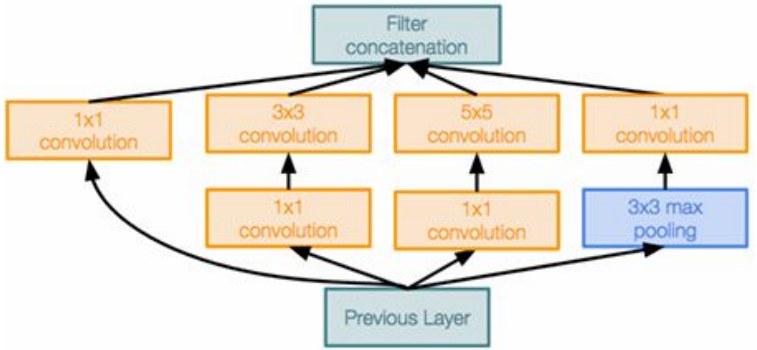
Input channels = 192
Output filters = 32

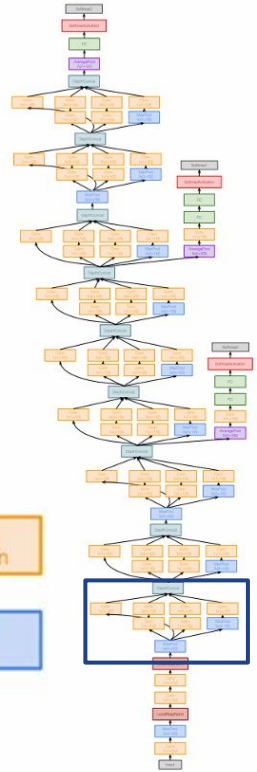
Step 1 : Reduce channels : 1×1 conv : 192 → 32

Step 2 : Apply 5×5 conv

Cost : 5 × 5 × 32 × 32 = 25,600 params

~5× cheaper

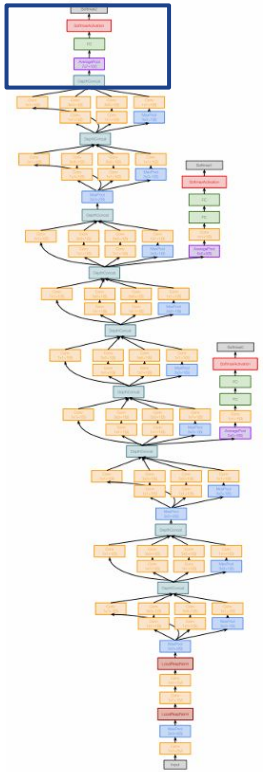




GoogLeNet- Global Average Pooling

Instead of large FC layers:

- Global Average Pooling : Each feature map becomes one number.
- Example : 7×7×1024 → 1×1×1024
- One Linear Layer : Final classification layer.
- Exemple 1024 → 1000 classes



GoogLeNet- Global Average Pooling

Instead of large FC layers:

- Global Average Pooling : Each feature map becomes one number.
- One Linear Layer : Final classification layer.
- Recall VGG-16: Most parameters were in the FC layers!

	Input size		Layer				Output size		
Layer	C	H/W	filters	kernel	stride	pad	C	H/W	params (k)
avg-pool	1024	7		7	1	0	1024	1	0
fc	1024		1000				1000		1025

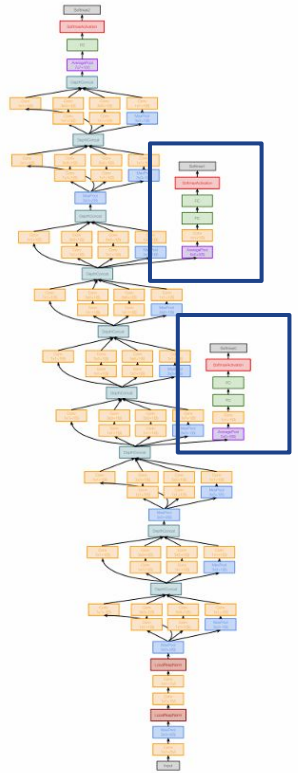
Compare with VGG-16

Layer	C	H/W	filters	kernel	stride	pad	C	H/W	params (K)
flatten	512	7					25088		
fc6	25088			4096			4096		102760
fc7	4096			4096			4096		16777
fc8	4096			1000			1000		4096

GoogLeNet- Auxiliary Classifiers

To address the vanishing gradient problem during training, GoogLeNet introduces auxiliary classifiers.

- Intermediate branches that act as smaller classifiers
- They are active only during training and help regularize the network.
- GoogLeNet was before batch normalization! With BatchNorm no longer need to use this trick



Batch Normalization (Ioffe & Szegedy, 2015)

When networks become deeper, training becomes unstable.

Problems observed:

- Slow convergence
- Unstable gradients
- Activation distributions change during training

Batch Normalization (Ioffe & Szegedy, 2015)

When networks become deeper, training becomes unstable.

Problems observed:

- Slow convergence
- Unstable gradients
- Activation distributions change during training

This phenomenon is called: Internal Covariate Shift

Batch Normalization (Ioffe & Szegedy, 2015)

This phenomenon is called: Internal Covariate Shift

Solution : Batch Normalization

Batch Normalization (Ioffe & Szegedy, 2015)

This phenomenon is called: Internal Covariate Shift

Solution : Batch Normalization

Idea:

Normalize activations during training:

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

Then apply learnable parameters:

$$y = \gamma \hat{x} + \beta$$

Batch Normalization (Ioffe & Szegedy, 2015)

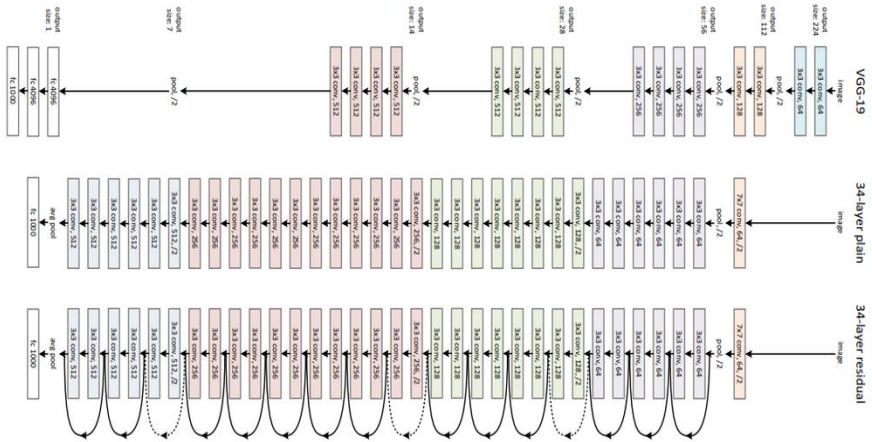
Once Batch Normalization made deeper networks possible, researchers tried very deep architectures.

However, they discovered, as depth increases:

- Training error increases
- Networks become harder to optimize

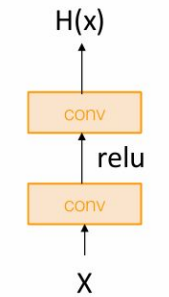
ResNet(He et al., 2015)

- In traditional neural networks, each layer feeds into the next layer.
- However in a network with residual blocks, each layer feeds into the next layer and directly into the layers about 2–3 hops away.

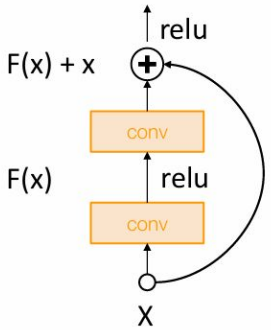


ResNet(He et al., 2015)

- In traditional neural networks, each layer feeds into the next layer.
- However in a network with residual blocks, each layer feeds into the next layer and directly into the layers about 2–3 hops away.

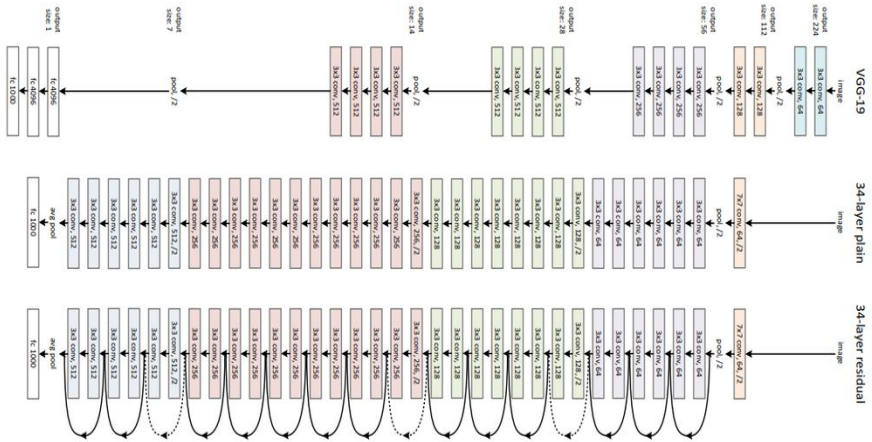


"Plain" block



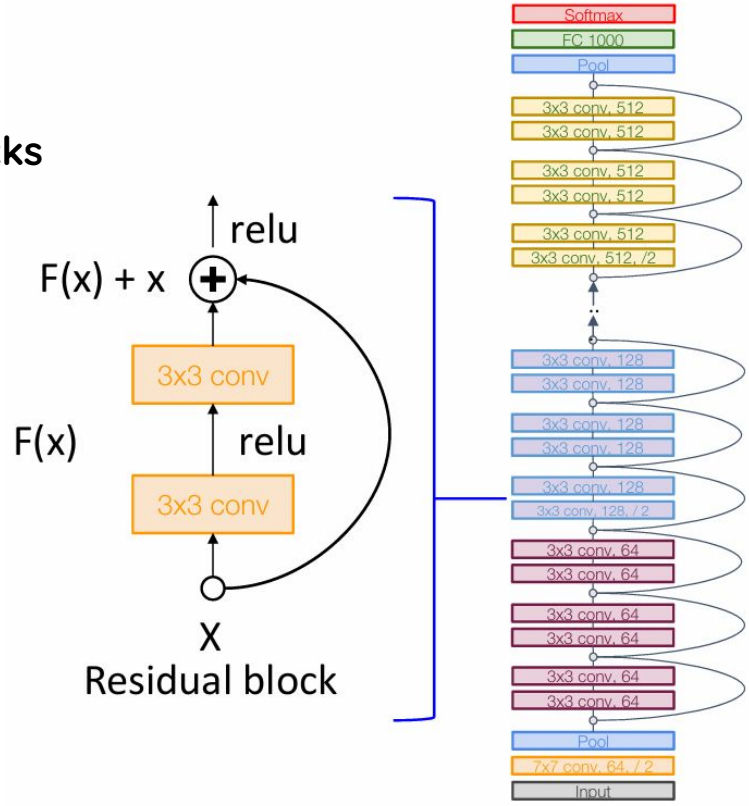
Additive "shortcut"

Residual Block



Residual Networks

A residual network is a stack of many residual blocks

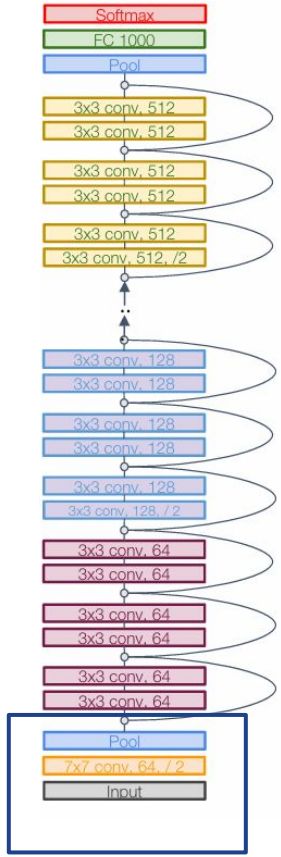


ResNet(He et al., 2015)

Residual Networks

The same stem as GoogLeNet to downsample the input 4x before applying residual blocks

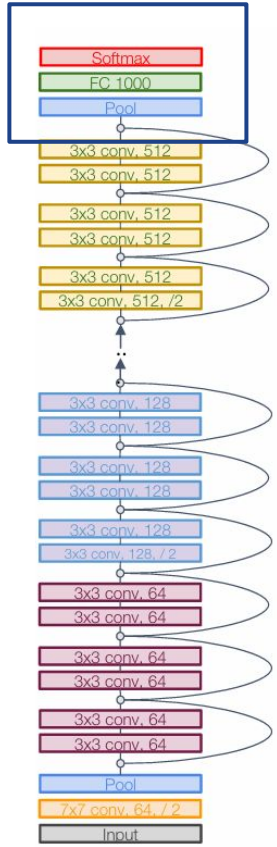
	Input size		Layer				Output size		
Layer	C	H/W	filters	kernel	stride	pad	C	H/W	params (k)
conv	3	224	64	7	2	3	64	112	9
max-pool	64	112		3	2	1	64	56	0



ResNet(He et al., 2015)

Residual Networks

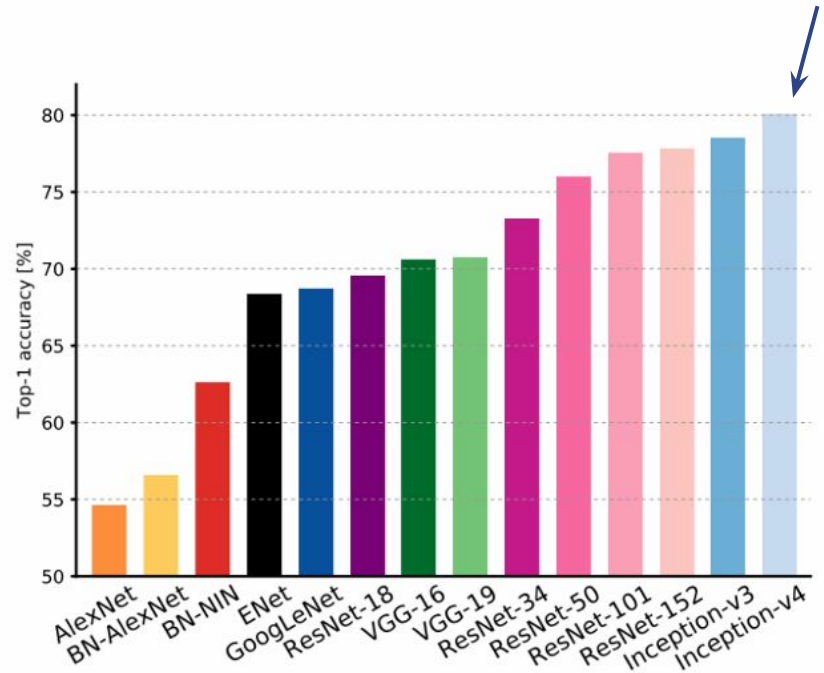
Like GoogLeNet, no big fully-connected-layers: instead use global average pooling and a single linear layer at the end

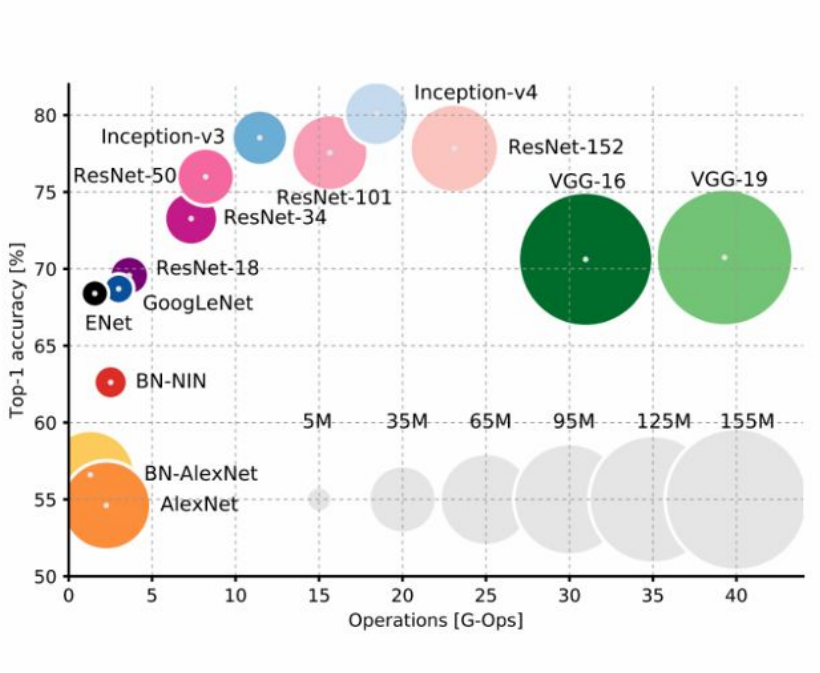


Comparing Complexity

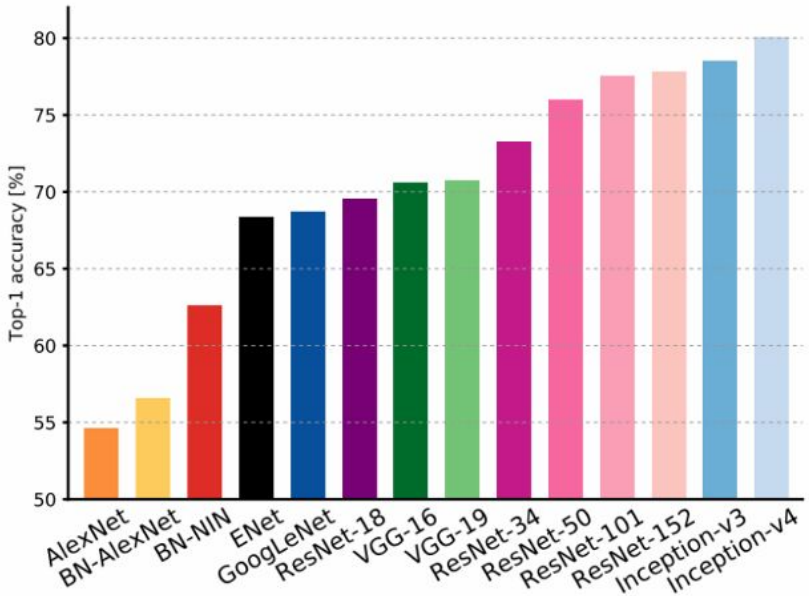
Dauphine | PSL
UNIVERSITÉ PARIS

Inception-v4: Resnet + Inception

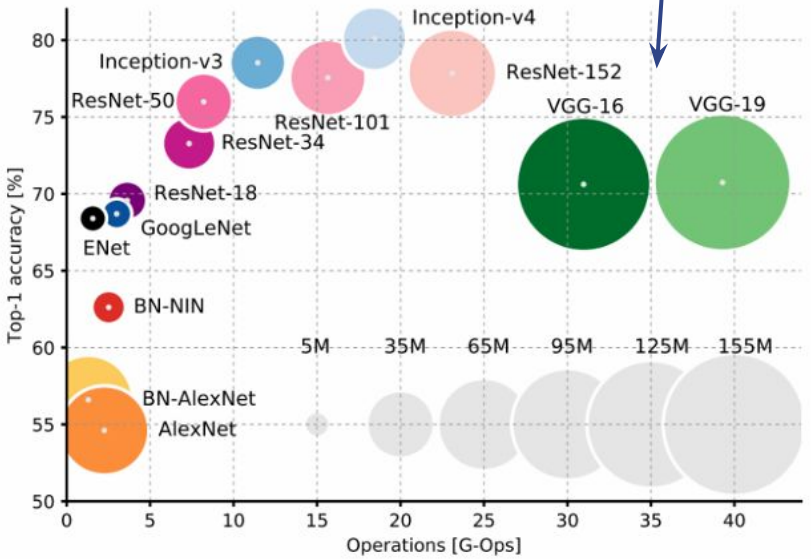




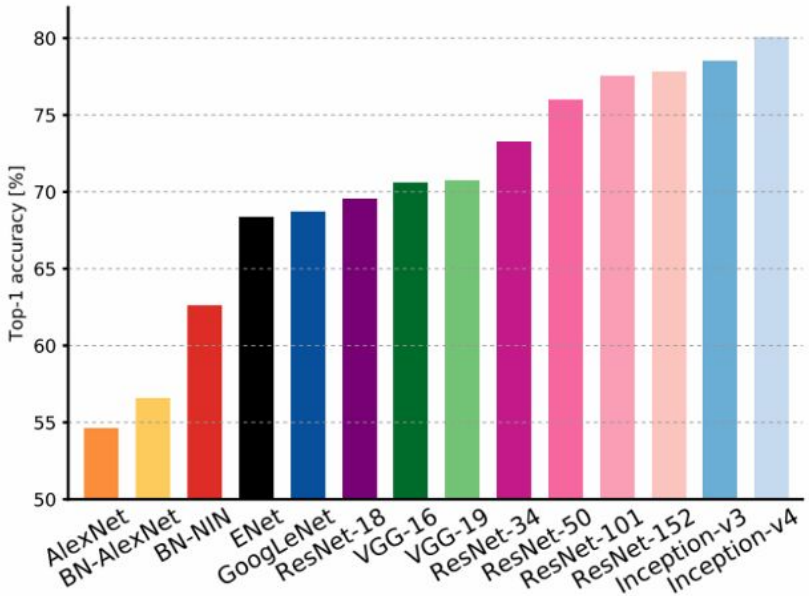
Comparing Complexity



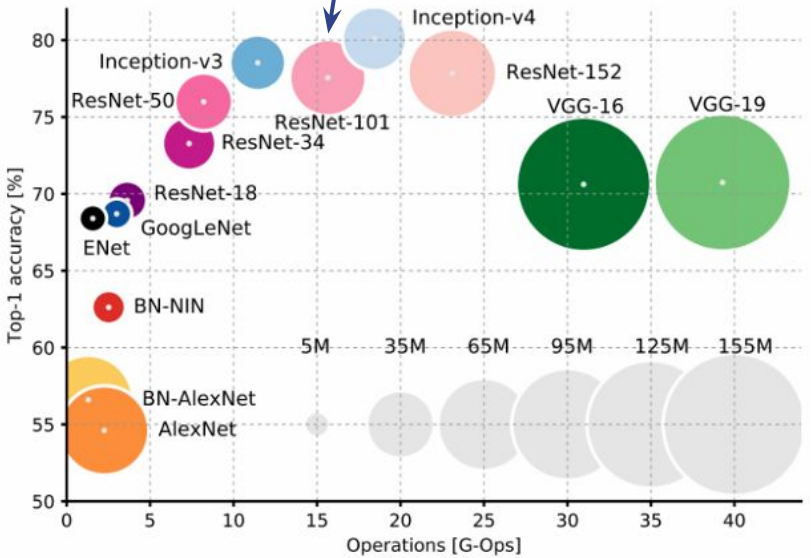
VGG: highest memory most operations



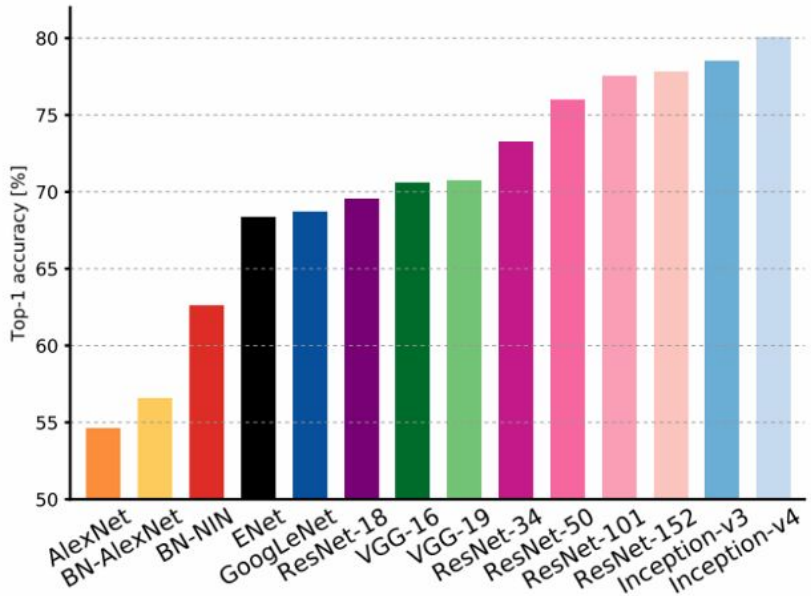
Comparing Complexity

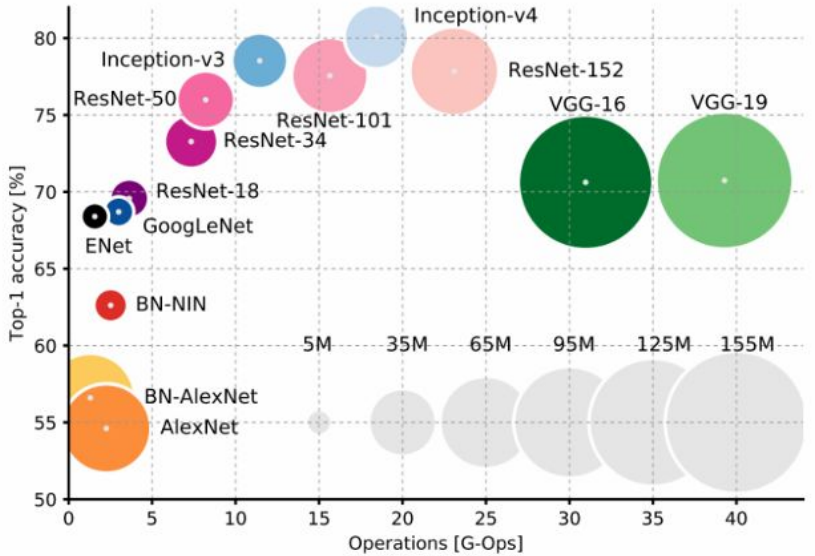


ResNet: moderate efficiency, high accuracy



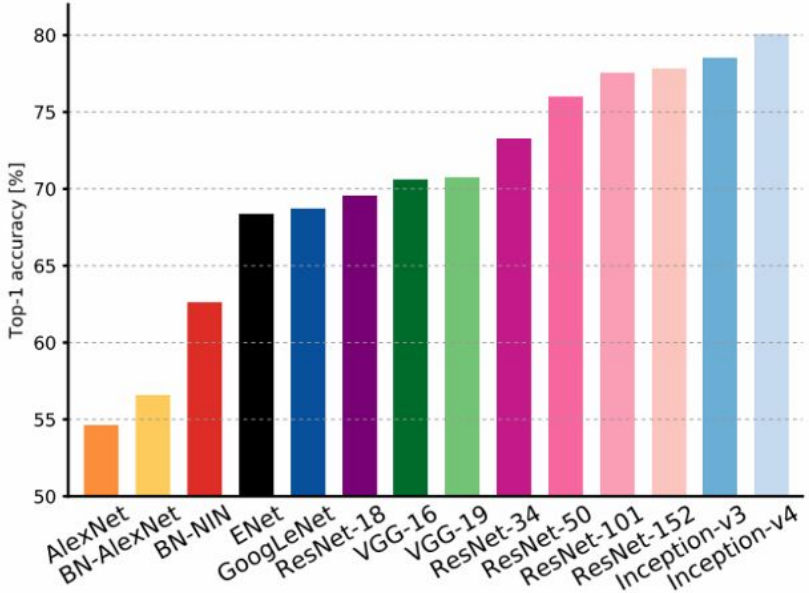
Comparing Complexity



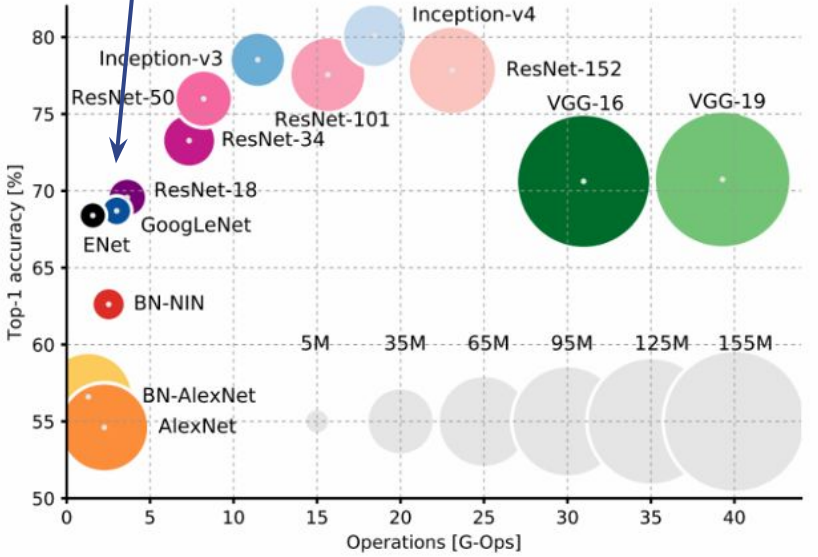


1 G-Op = 1 billion operations

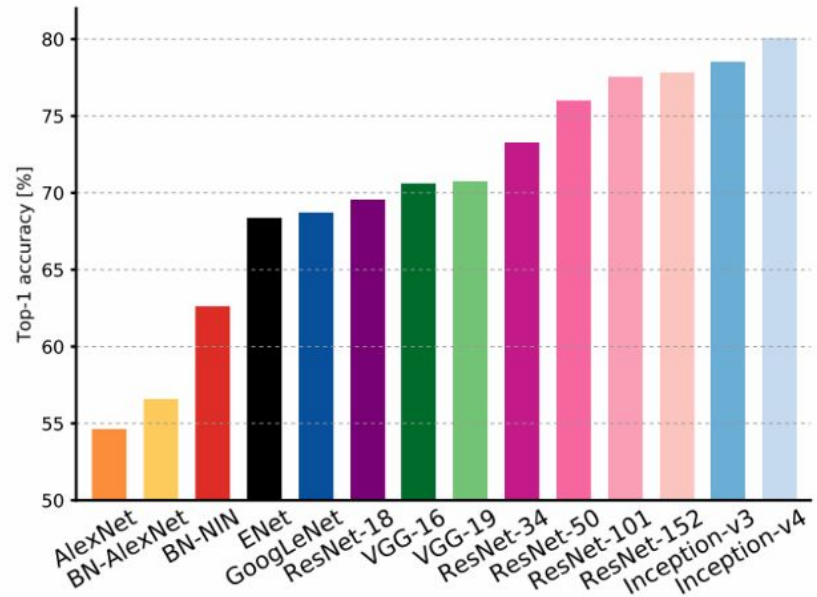
Comparing Complexity



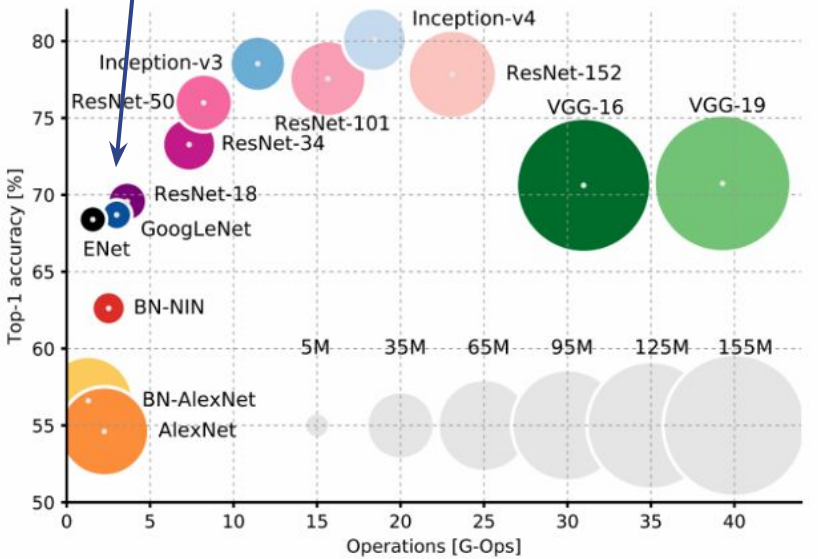
GoogLeNet: very efficient



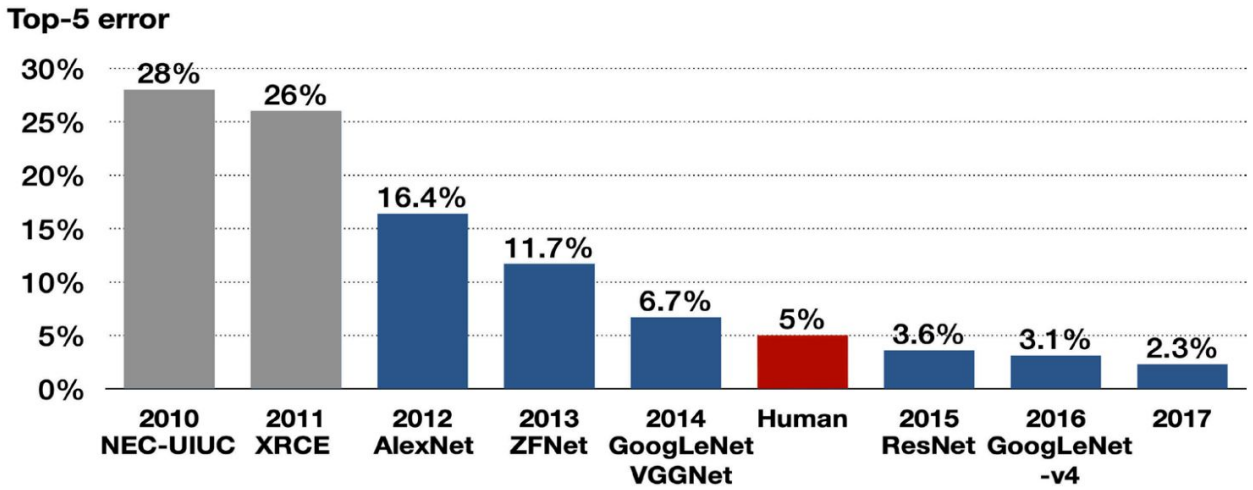
Comparing Complexity



GoogLeNet: very efficient



ImageNet Challenge



Annual ImagNet Competition held after 2017, now moved to Kaggle

Transfer Learning

References

Books

• Computer Vision: Models, Learning, and Inference (2012). https://amzn.to/2rxrdOF
• Programming Computer Vision with Python (2012). https://amzn.to/2QKTaAL
• Multiple View Geometry in Computer Vision (2004). https://amzn.to/2LfHLE8
• Computer Vision: Algorithms and Applications (2010). https://amzn.to/2LcIt4J
• Computer Vision: A Modern Approach (2002). https://amzn.to/2rv7AqJ
• Deep Learning for Computer Vision : Image Classification, Object Detection and Face Recognition in Python(2019). https://machinelearningmastery.com/deep-learning-for-computer-vision/
• Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow — Aurélien Géron (2019) https://amzn.to/3uZbN6Z

Articles & Online Resources

• Krizhevsky, Sutskever, Hinton (2012) — ImageNet Classification with Deep CNNs (AlexNet — deep learning breakthrough)
• Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis, 2017
• "Computer vision," Wikipedia. https://en.wikipedia.org/wiki/Computer_vision
• "Machine vision," Wikipedia. https://en.wikipedia.org/wiki/Machine_vision
• "Digital image processing," Wikipedia. https://en.wikipedia.org/wiki/Digital_image_processing
• "Training Convolutional Neural Networks: What Is Machine Learning" Analog Devices. https://www.analog.com/en/resources/analog-dialogue/articles/training-convolutional-neural-networks-what-is-machine-learning-part-2.html