Evolution of Convolutional Neural Networks Architectures

By: Marwan Abdelatti

PhD Candidate (Mechanical, Systems, and Industrial Engineering)

Contents

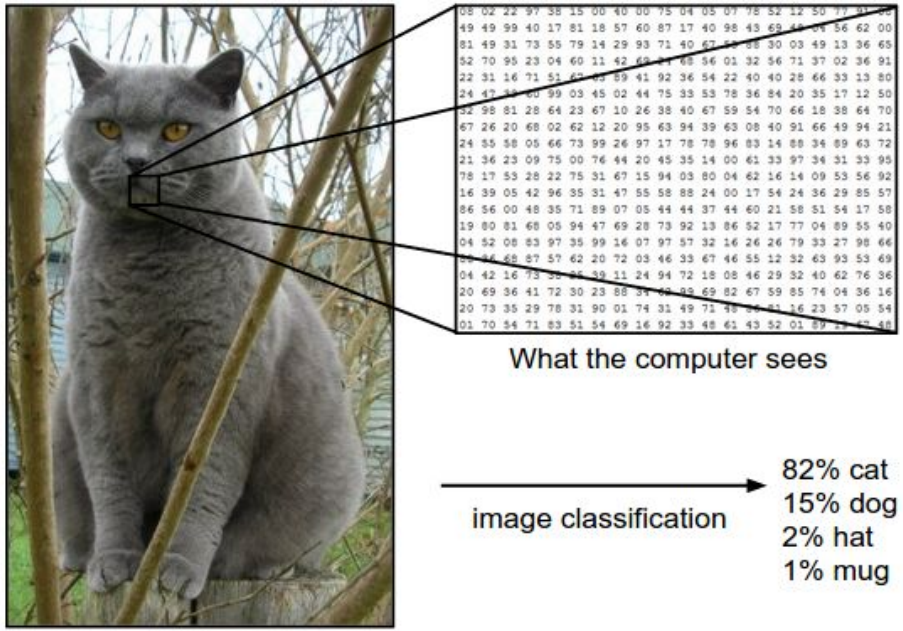
- Motivation
- Introduction
- AlexNet
- VGG
- GoogleNET
- ResNET
- Conclusion

Motivation

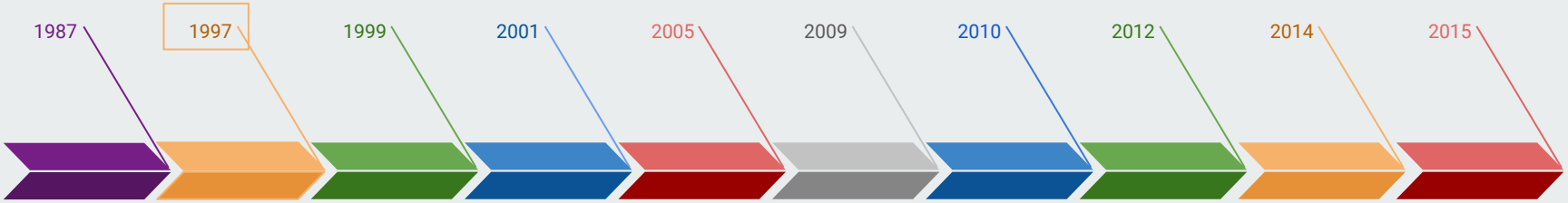
- To enable computers to "*see*" like humans

"Image classification is the task of taking an input image and outputting a class (cat, dog, etc.)"

- Computers "*see*" an image as an array of pixel values (N x N x 3 for RGB images).
- Researchers spent a plenty of time to make computers differentiate between all the images it's given.



Motivation



Segmentation and Aggregation

low-level vision techniques in which the object is distinguished from the background (e.g., edge and contour detection).

Normalized Cuts

By focusing on global features, image is divided into parts to be compared for similarities.

SIFT & Object Detection

Local features are extracted from reference images, saved in database. New images are compared to them.

Face Detection

An AdaBoost machine learning is used.

Quick and accurate!

HoG

Counts the occurrences of gradient orientation in localized portions of an image.

Deformable Part Model (DPM)

The best performance till that time.

Modeling of an object as a set of parts constrained in a spatial geometric arrangements. *

NEC-UIUC

IMAGENET

SIFT and LBP featuring and stochastic SVM.

AlexNet

The rise of Convolutional Neural Networks.

The beginning of modern machine vision.

GoogleNet

The beginning of inception modules.

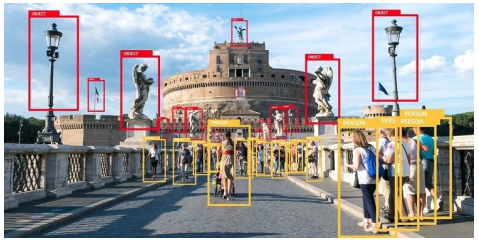
(Parallelism)

ResNet

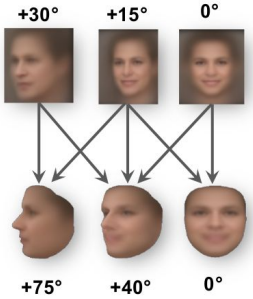
The beginning of residual blocks.

REALLY deep!

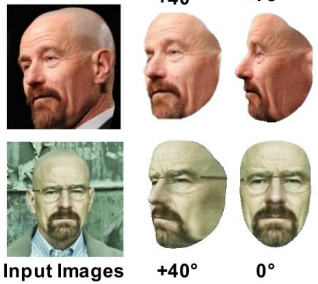
ConvNets are Ubiquitous: The Simple to The Complex



Object Recognition: Boxing Objects



3D Face-recognition



Segmentation - Richer than Boxing

- Need GPUs
- Used in Self-Driving Cars

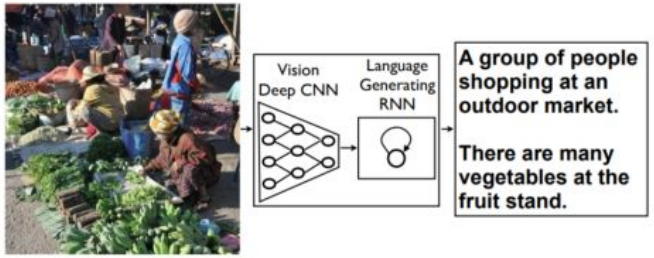
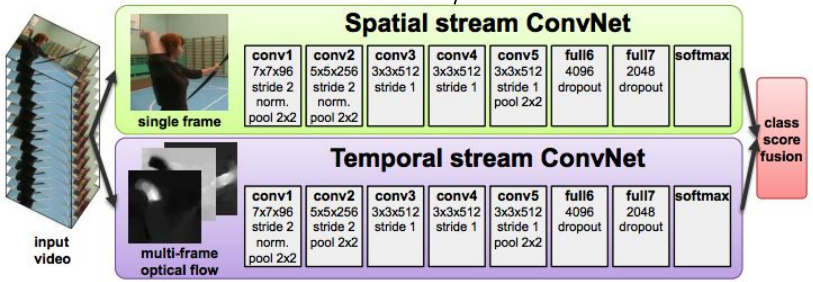


Image captioning

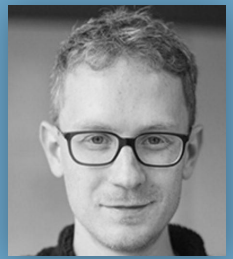


Classify Videos: Images as well as temporal Information

AlexNet -2012

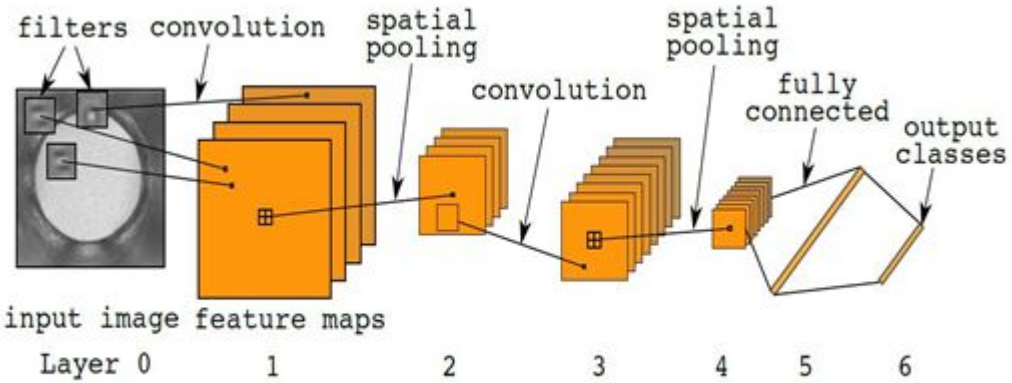
One of the most influential CNN architectures. As of 2018, the Alexnet paper has been cited over 37,000 times.

ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2012 winner (First CNN based winner)



Alex Krizhevsky

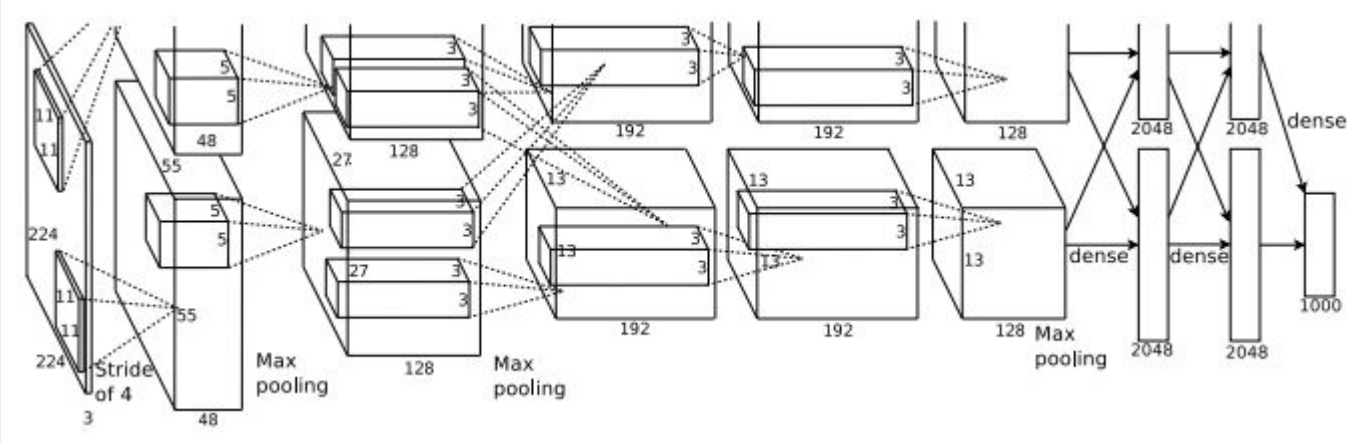
A quick glance back in time: Le-NET5 -1998



- Yann LeCun used Conv. Nets for digit recognition
- Widely used in postal services for zip code recognition
- Was not able to scale
- Limited set of applications
- Architecture is [CONV-POOL-CONV-POOL-FC-FC]

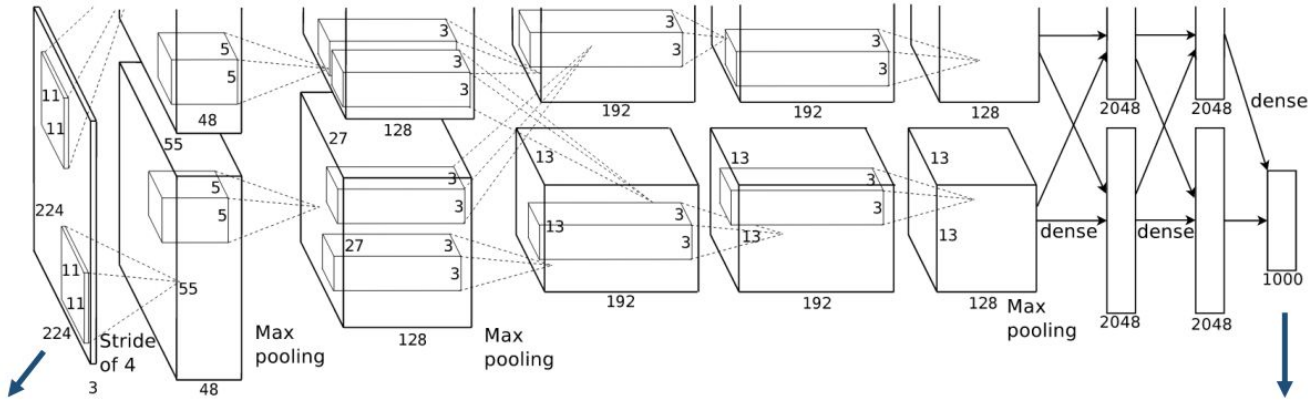
AlexNet

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



- Based on LeNET5 but deeper
- Started the gold rush of Conv. Nets
- Leveraged more digitally available databases (e.g. ImageNET) and GPU parallel processing
- Significantly outperformed the second runner-up (top 5 error of 16% compared to runner-up with 26% error
- First use of ReLU and Norm layers (not common now)
- 7 CNN ensembles: 18.2% to 15.4%

Architecture Description



*227 x 227 Input

1000 Softmax

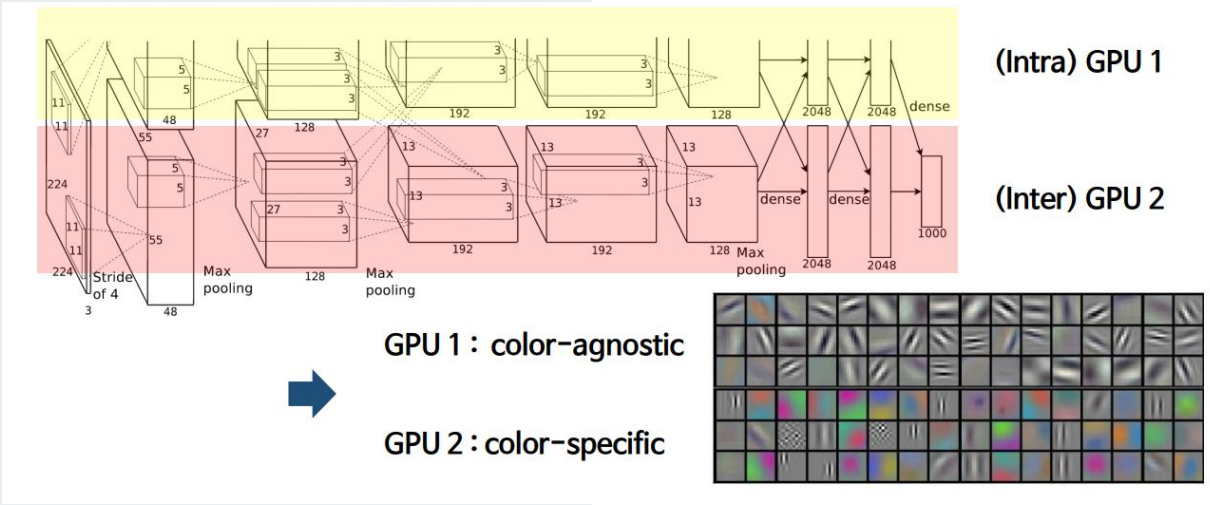
- First layer CONV1 output size: 96 11x11 filters applied at stride 4: 55x55x96
- Number of parameters: (11*11*3+1)*96 = 34944
- Second Layer POOL1 output volume: 3x3 filters applied at stride 2: 27x27x96
- Number of parameters - zero!

.....

- Last layers FC6 (4096 neurons), FC7 (4096 neurons), FC8 (1000 neurons giving class scores)

- 8 layers: 5 convolutional layers and 3 fully connected layers
- ReLu is applied after every convolutional and fully connected layer
- Dropout is applied before the first and the second fully connected layer

Training on Multiple GPUs



- AlexNET was trained on GTX 580 GPUs with only 3 GB memory - couldn't fit the network on 1 GPU and used 2 GPUs
- Half of the neurons or feature maps assigned to each GPU
- CONV1, CONV2, CONV4, CONV5 connected with feature maps on same GPU
- Only a few layers have inter-GPU communication
- CONV3, FC6, FC7, FC8 connected with all feature maps in preceding layer

VGGNet -2014

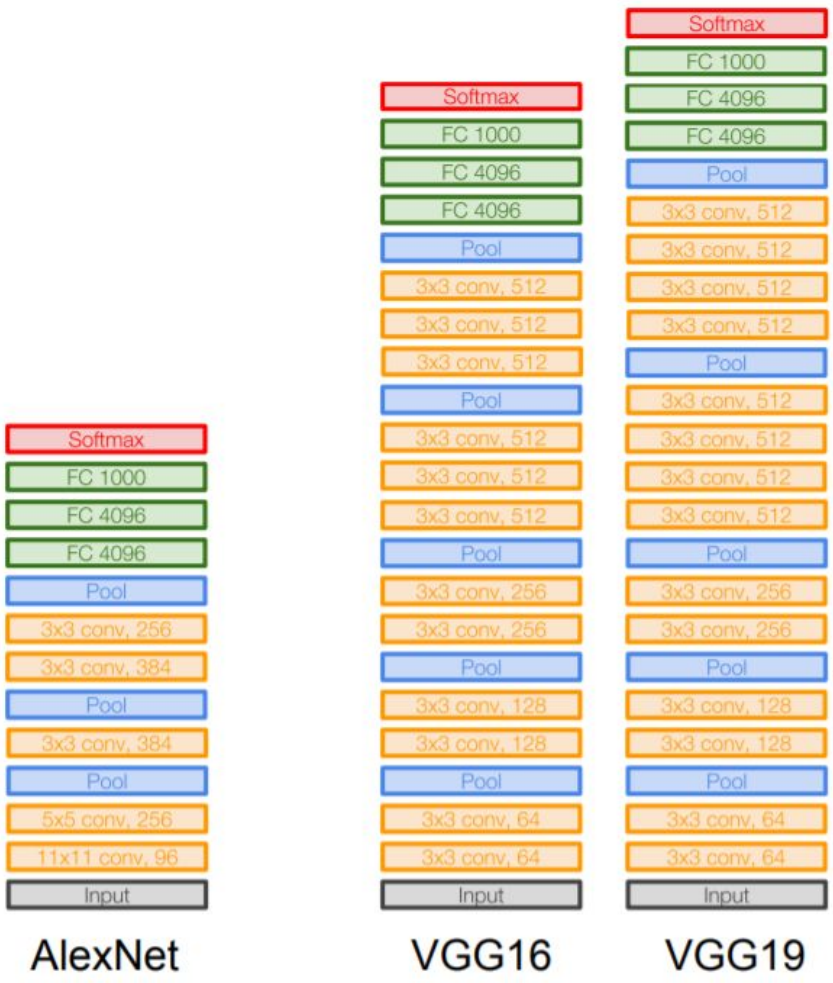
Visual Geometry Group network, University of Oxford

2014 ILSVRC close runner up to GoogLeNet - Beat GoogLeNet on localization

(Karen Simonyan and Andrew Zisserman)

VGG

- Significantly deeper than AlexNet.
- 140 million parameters.
- **Small filters** - Only 3x3 CONV stride 1, pad 1 and 2x2 MAX POOL stride 2
- 7.3% top 5 error



VGGNet

- **Why use smaller filters? (3x3 conv)**

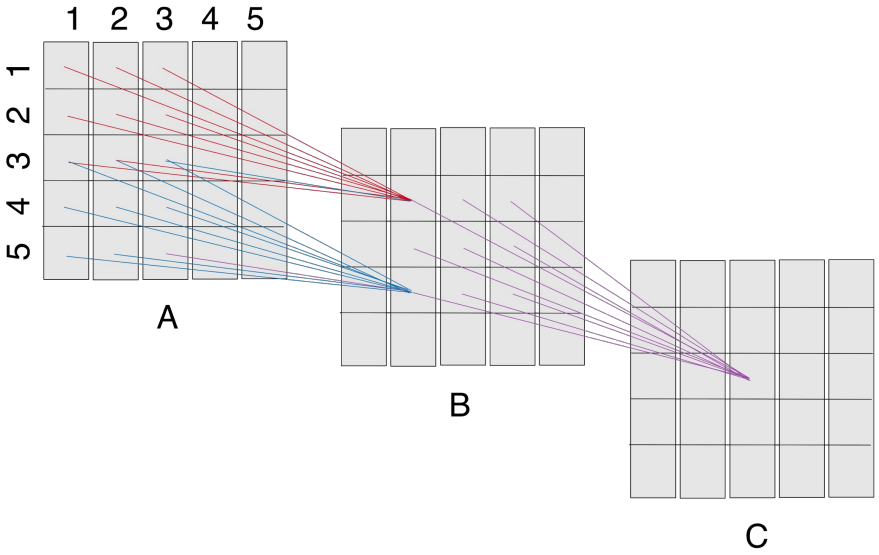
Stack of three 3x3 conv (stride 1) layers has the same effective receptive field as one 7x7 conv layer.

- **What is the effective receptive field of three 3x3 conv (stride 1) layers?**

7x7

But deeper, more non-linearities

And fewer parameters: 3 * (3^2C^2) vs. 7^2C^2 for C channels per layer



- The “receptive field” of a “neuron” in a layer would be the cross section of the previous layer from which neurons provide inputs.
- So, the receptive field of B(2,2) is the square A(1:3,1:3), that of B(4,2) is the square A(3:5,1:3), that of C(3,3) is B(2:4,2:4), and so on..
- Now the receptive field of C(3,3) is B(2:4,2:4), which itself receives inputs from A(1:5,1:5)

(not counting biases)

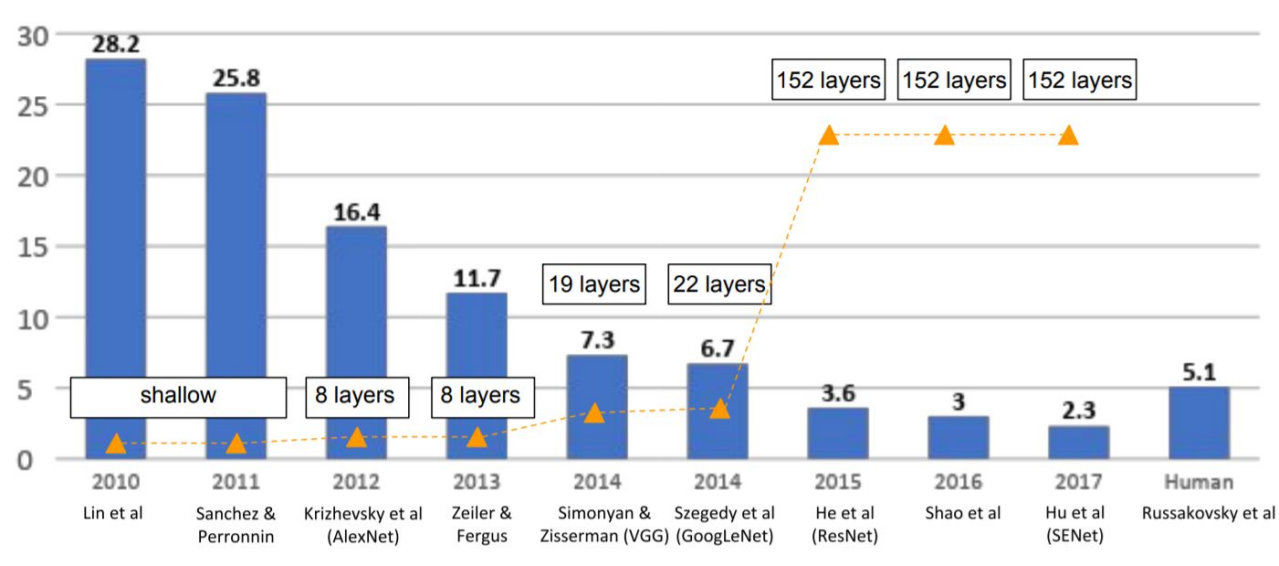
INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M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

TOTAL memory: 24M * 4 bytes ~= 96MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

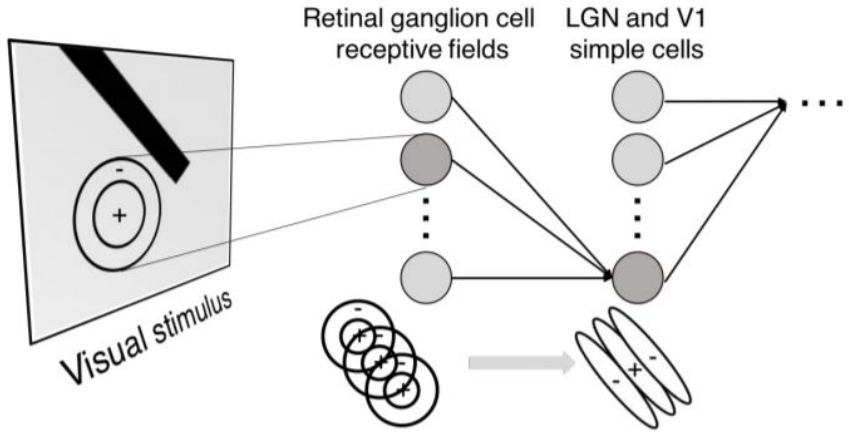
Note:

Most memory is in early CONV

Most params are in late FC



VGGNet reinforced the notion that convolutional neural networks need to have deep layers in order to represent the hierarchical representation of visual data, as demonstrated by Hubel and Wiesel 1959



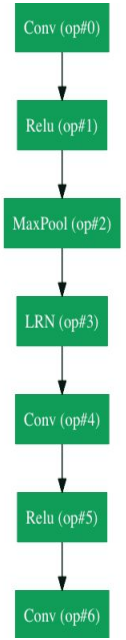
GoogLeNet-2014

Inception module . . .

Winner of ILSVRC 2014 with a top 5 error rate of 6.7%.

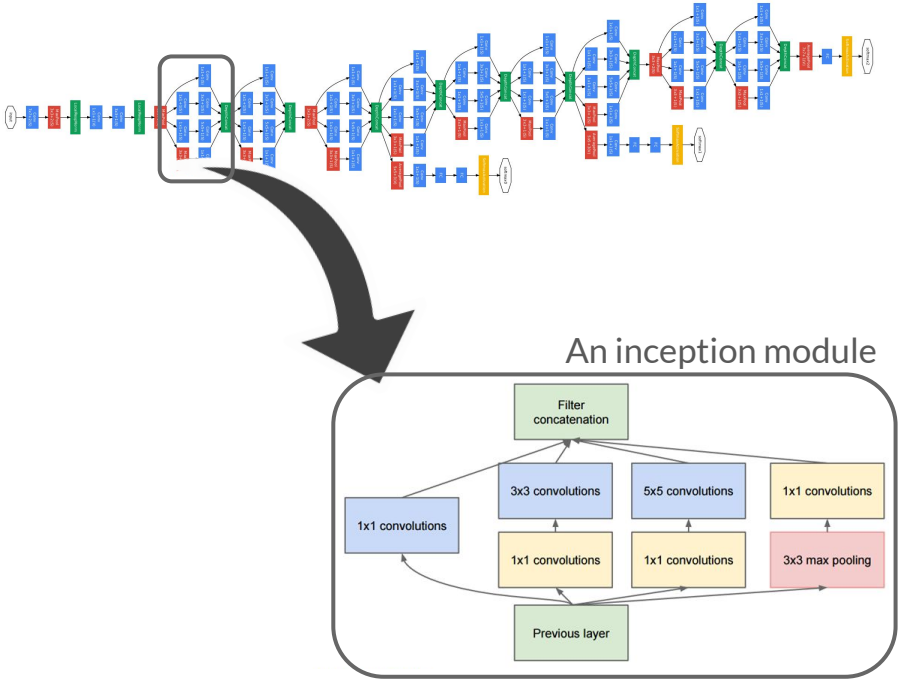
GoogleNet

- Introduced the inception module concept.
- Used 9 Inception modules, with over 100 layers in total.
- Has 12x fewer parameters than AlexNet.

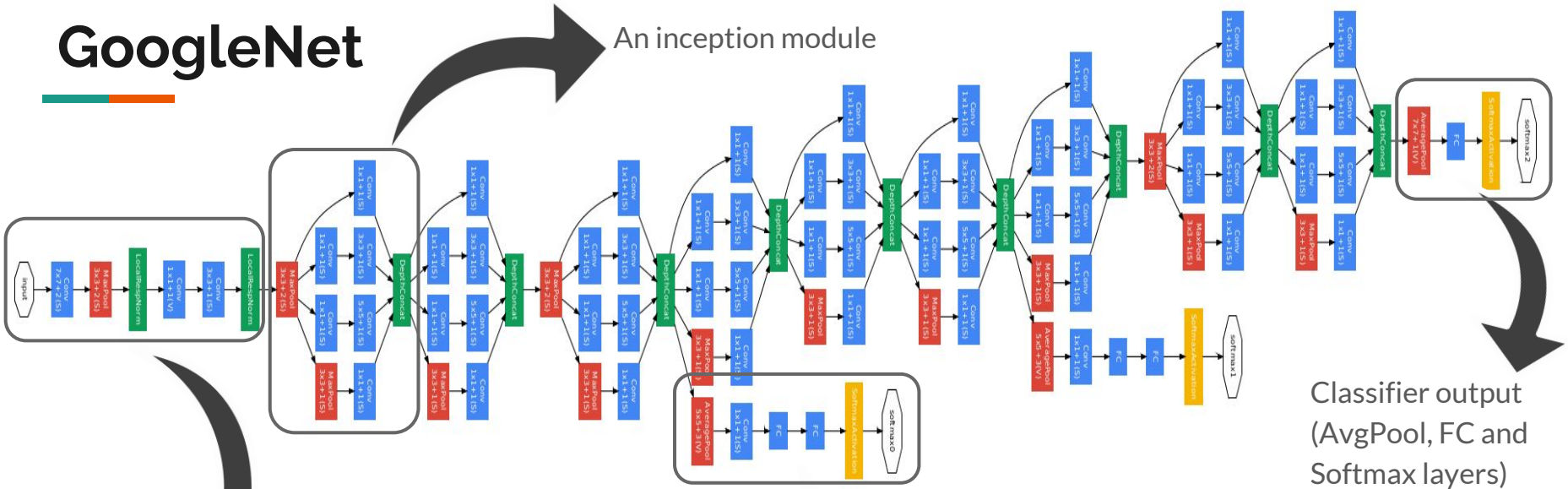


GoogleNet

- Not everything is happening sequentially.
- Perform operations in parallel (e.g., Conv, Pooling, filter size...etc).
- The 1x1 conv provides depth reduction.

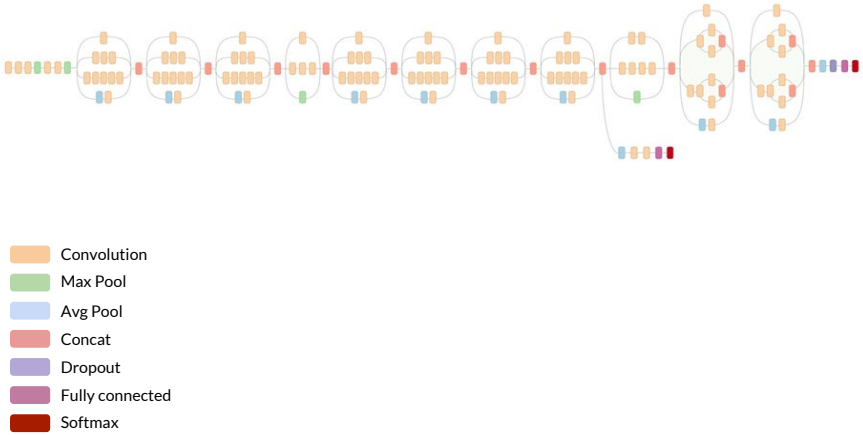


GoogleNet


An inception module
Stem layer: (Conv, Max Pool, and Local response normalization)
Auxiliary classification outputs to inject additional gradient at lower layers:
(AvgPool, 1x1 Conv, FC, and Softmax)
Classifier output (AvgPool, FC and Softmax layers)

GoogleNet

- The network is able to perform different operations:
  - It extracts fine details by the 1x1 conv filter.
  - Extracts large receptive field of the input information by the 5x5 filter.
  - Pooling operation that helps to reduce spatial sizes and address overfitting.
  - ReLUs helps improve the nonlinearity of the network.



Versions of the inception module

GoogleNet

Conv Operations:

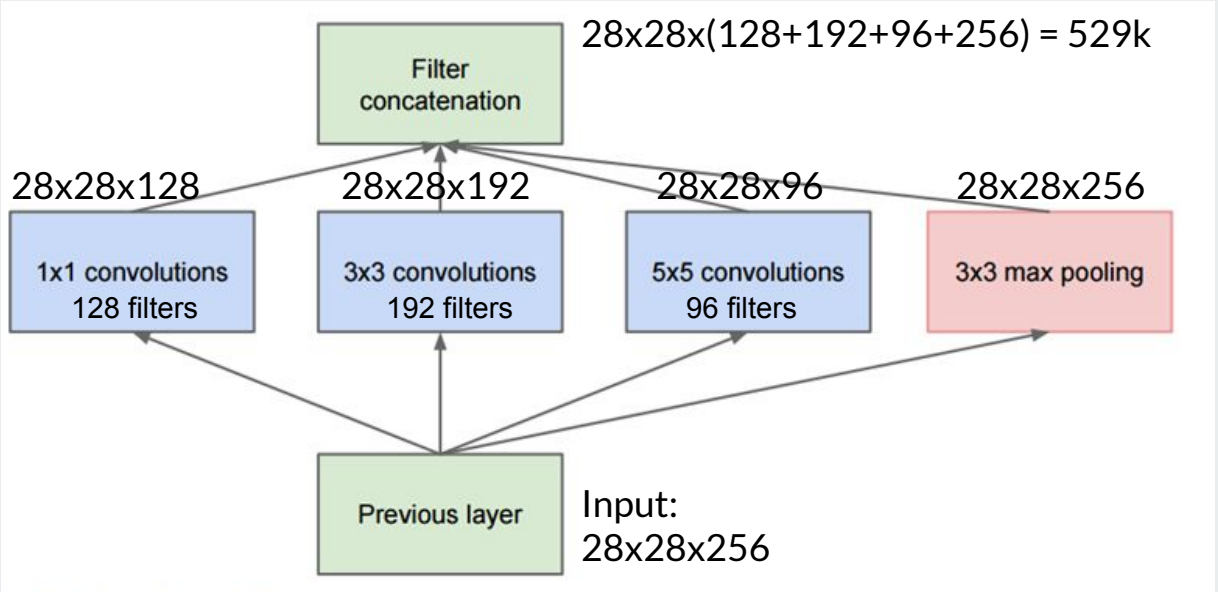
[1x1 conv, 128] 28x28x256x1x1x128

[3x3 conv, 192] 28x28x256x3x3x192

[5x5 conv, 96] 28x28x256x5x5x96

Total: 854M operations

Naive inception module

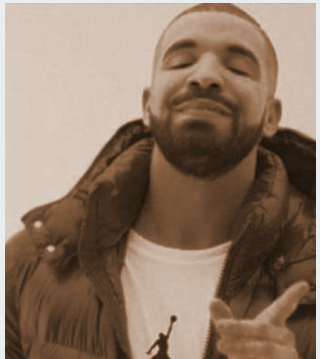


GoogleNet

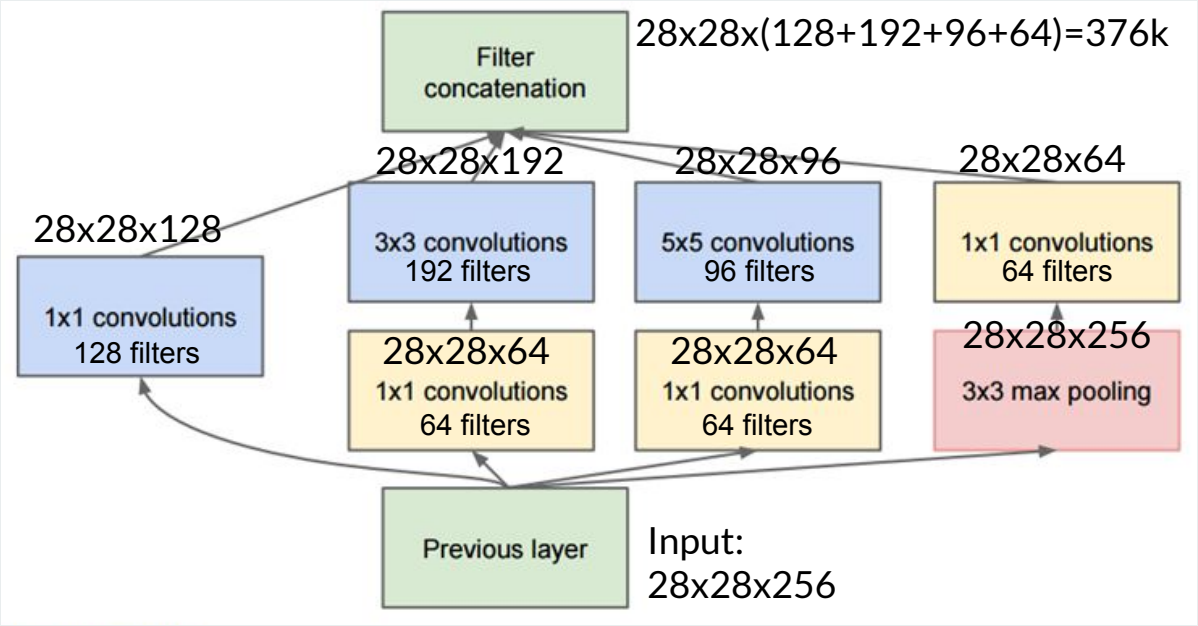
Conv Operations:

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

Total: 358M operations ~ 58% less

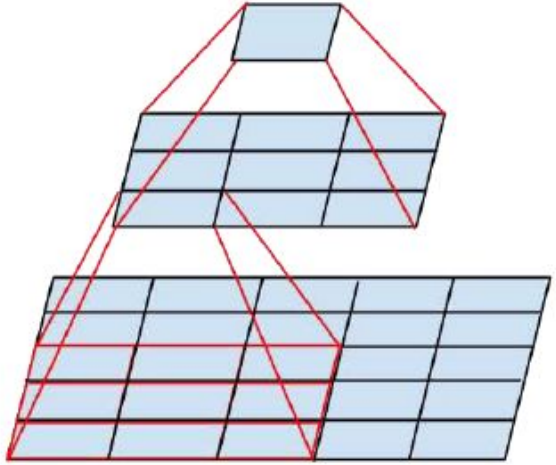


Full inception module



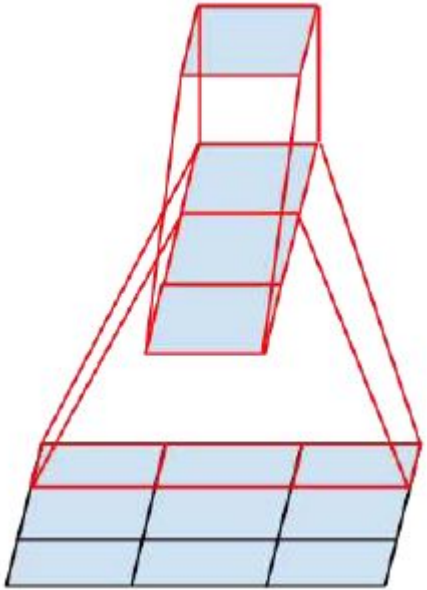
GoogleNet Inception V2

- Factorize 5x5 conv to two 3x3 conv to improve computational speed.
- By using 1 layer of 5×5 filter, number of parameters = 5×5=25
- By using 2 layers of 3×3 filters, number of parameters = 3×3+3×3=18
- Number of parameters is reduced by 28%.

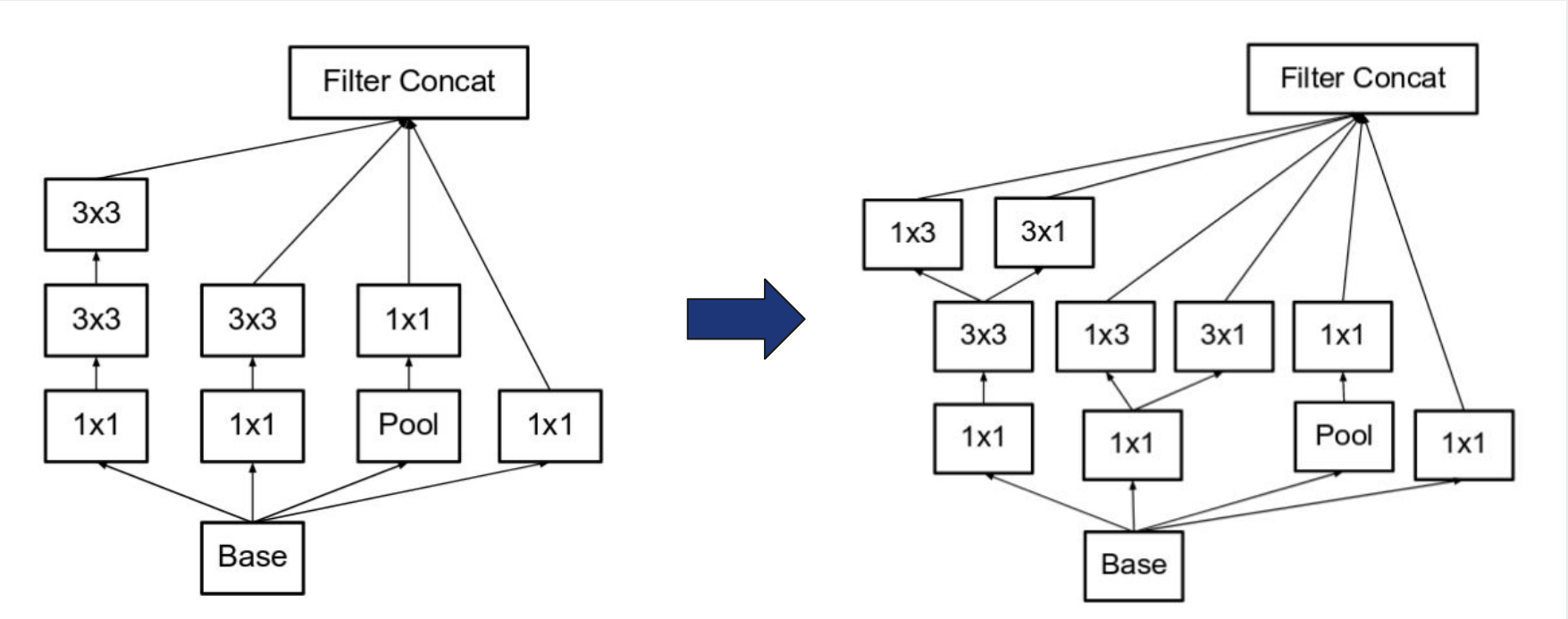


GoogleNet Inception V2

- Factorize conv of filter size 3x3 to a combination of 1x3 and 3x1 conv.
- By using 3×3 filter, number of parameters = 3×3=9
- By using 3×1 and 1×3 filters, number of parameters = 3×1+1×3=6
- Number of parameters is reduced by 33%.



GoogleNet Inception V2

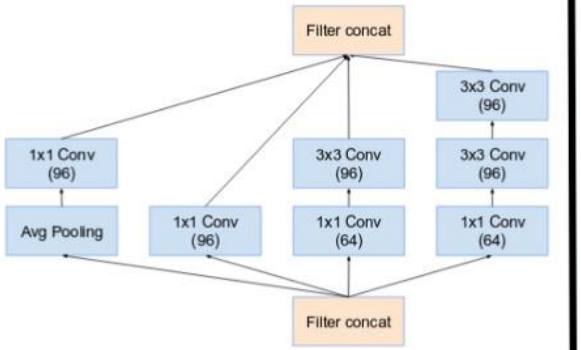


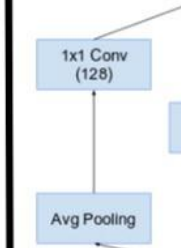
GoogleNet Inception V3

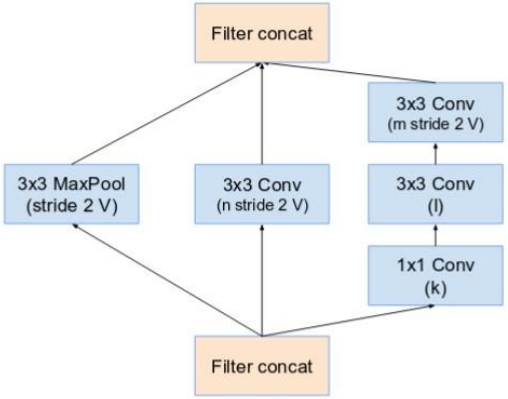
- Mainly to improve in the auxiliary classifiers:
  - Employed RMSProp Optimizer.
  - BatchNorm in the Auxiliary Classifiers.
  - Label Smoothing (to prevent overfitting).
- Factorized the 7x7 conv in the stem network.

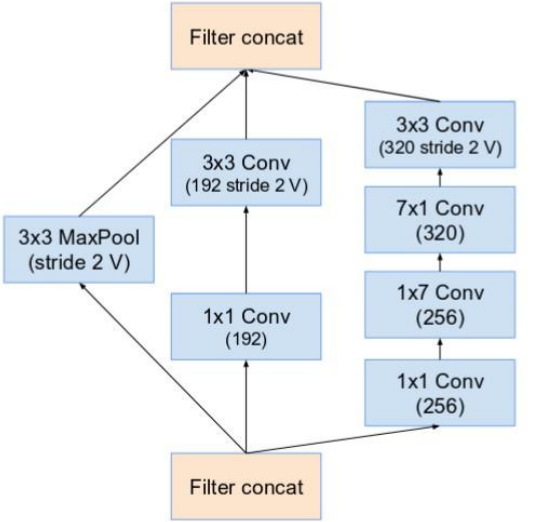
GoogleNet Inception V4

- Further modifications on the stem network (mainly factorization).
- Three different inception modules.
- Reduction blocks introduced:
  - Reduction in lengths and widths



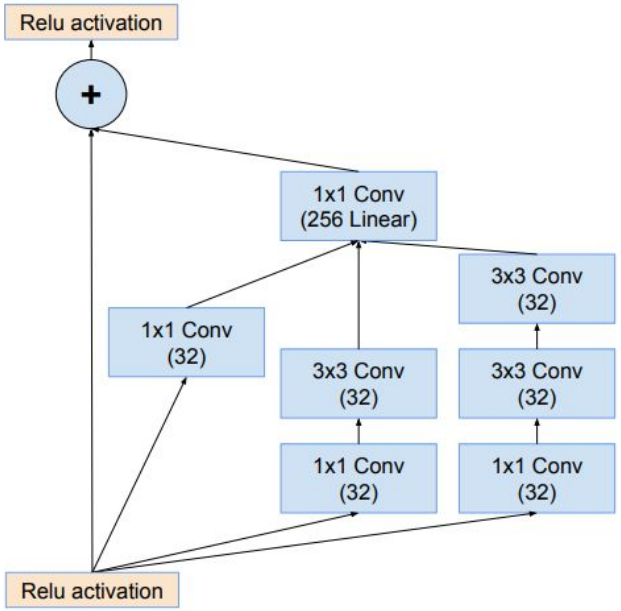






GoogleNet

- A newer version (2016) called inception—ResNet inspired from ResNet and uses residual connection.
  - Training with residual connections accelerates the training of Inception networks significantly.
  - Higher performance (slightly!).



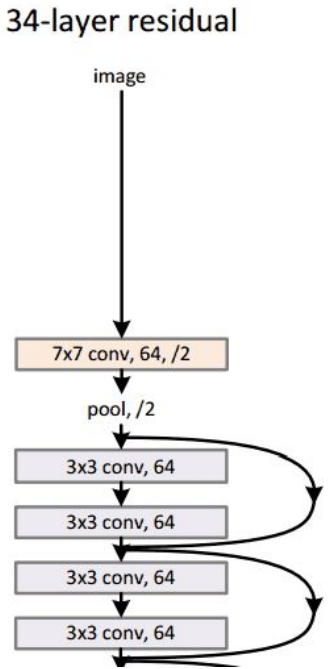
ResNet-2015

Revolution in depth

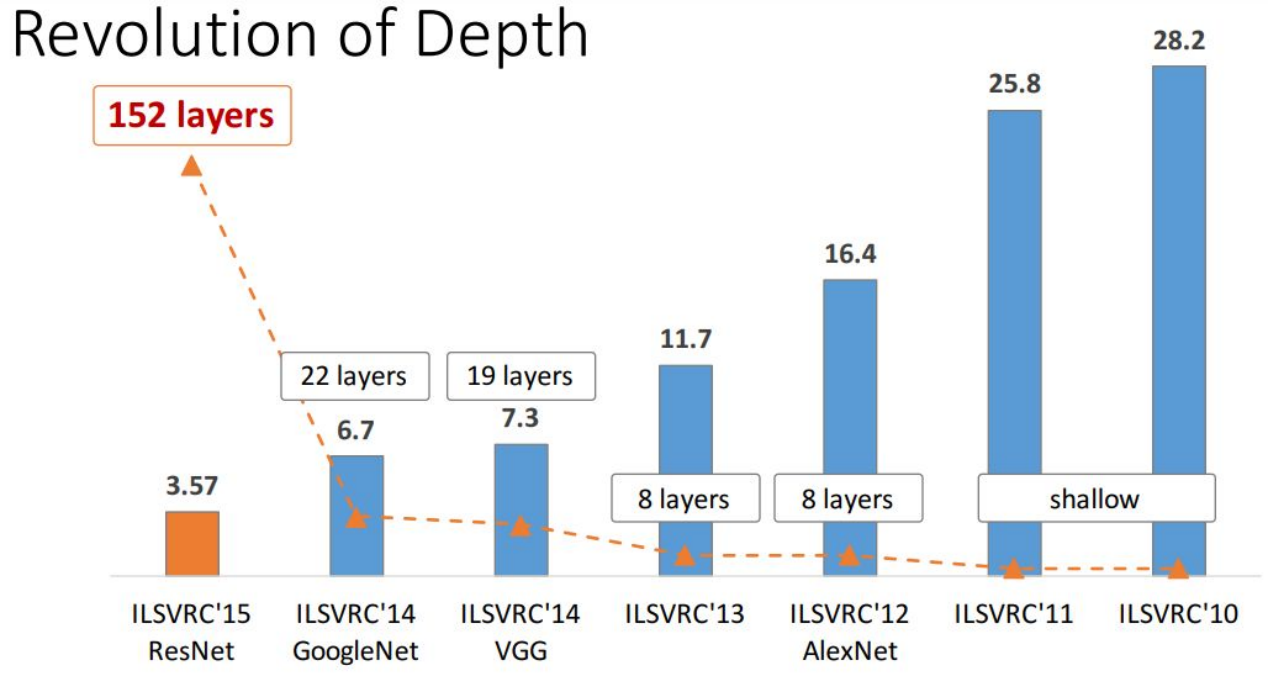
Winner of ILSVRC 2015 with a top 5 error rate of 3.57%.

ResNet

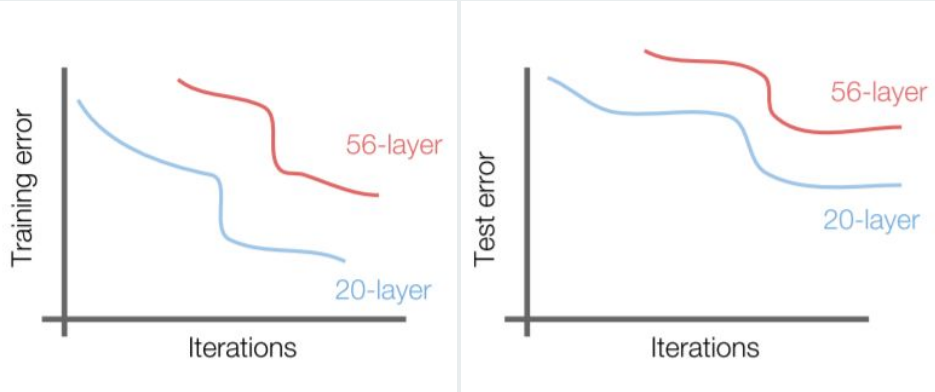
- Very deep networks using residual connections
- Consists of 152 layers in total (up to >1000 !!).
- Residual blocks are introduced.

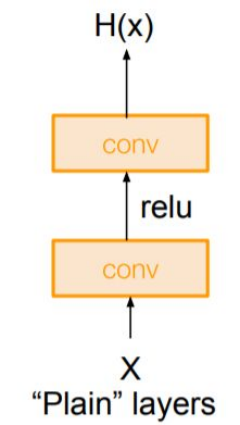


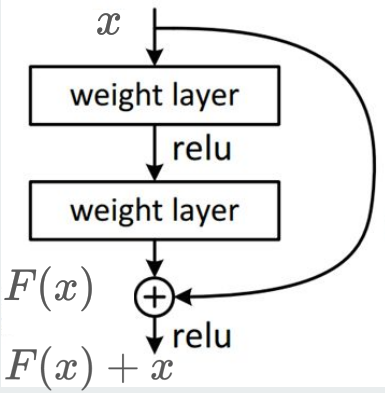
ResNet


Revolution of Depth
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10

Does stacking old blocks Help?







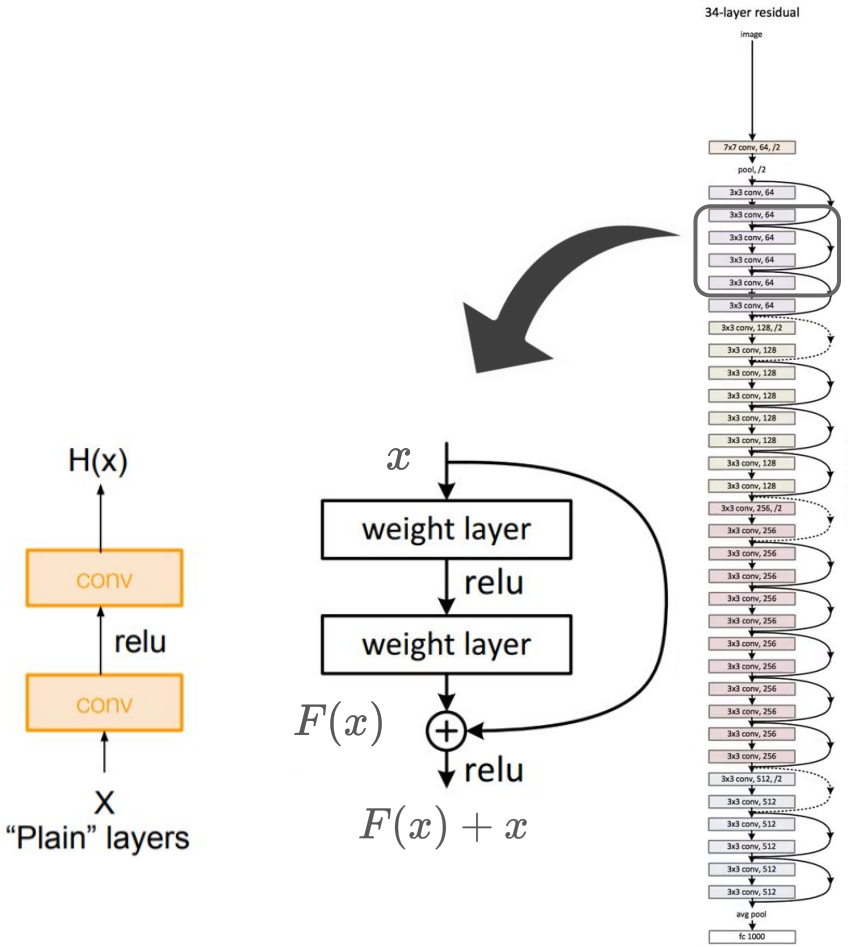
- 56-layer model performs worse on both training and test error - not caused by over-fitting
- Hypothesis: This is an optimization problem, deeper modules are harder to optimize
- Deeper modules should be able to perform at least as well as the shallower model
- Solution by construction: Copy the learned layers from shallower model and set additional layers to identity mapping

ResNet

- Input x goes through conv-relu-conv series giving $F(x)$.
- The resulted $F(x)$ is then added to the original input x giving:

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

- This helps preserving the information of the original input through all the layers.
- Which will simplify the optimization process.



“it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping”

- Kaiming He, the main developer of ResNet

ResNet

- Very smooth forward propagation:

$$x_{l+1} = F(x_l) + x_l$$

$$\downarrow$$

$$x_{l+2} = F(x_{l+1}) + x_{l+1}$$

$$\vdots$$

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

- Any x_L is an additive outcome in contrast to multiplicative $x_L = \prod_{i=l}^{L-1} W_i x_l$

ResNet

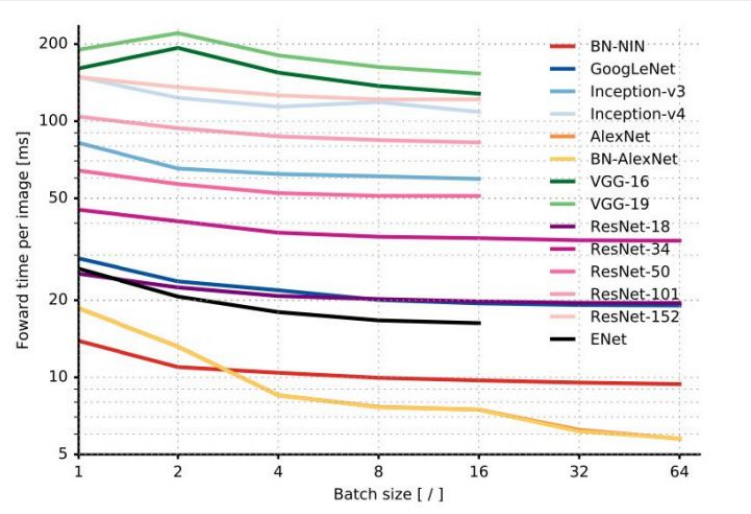
- Very smooth backward propagation:

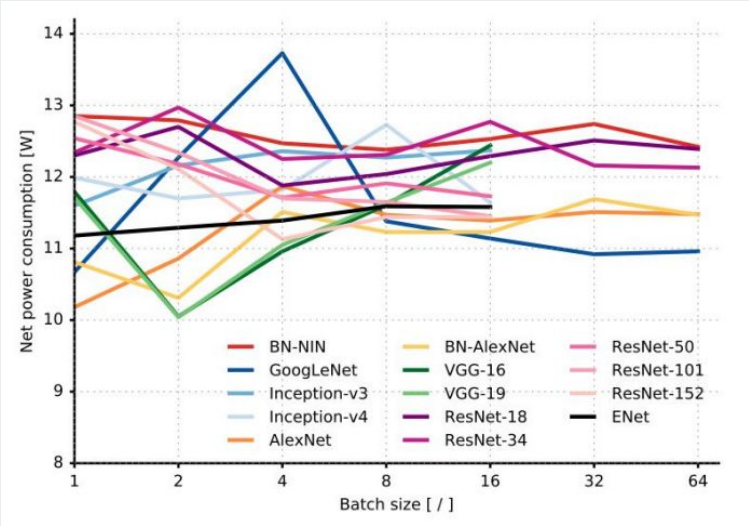
$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

$$\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial E}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i)\right)$$

- Any $\frac{\partial E}{\partial x_L}$ is directly back-propagated to any $\frac{\partial E}{\partial x_l}$ plus residual (i.e., $F(x_i)$).
- Any $\frac{\partial E}{\partial x_l}$ is additive (i.e., unlikely to vanish) in contrast to $\frac{\partial E}{\partial x_l} = \prod_{i=1}^{L-1} W_i \frac{\partial E}{\partial x_L}$

ResNet

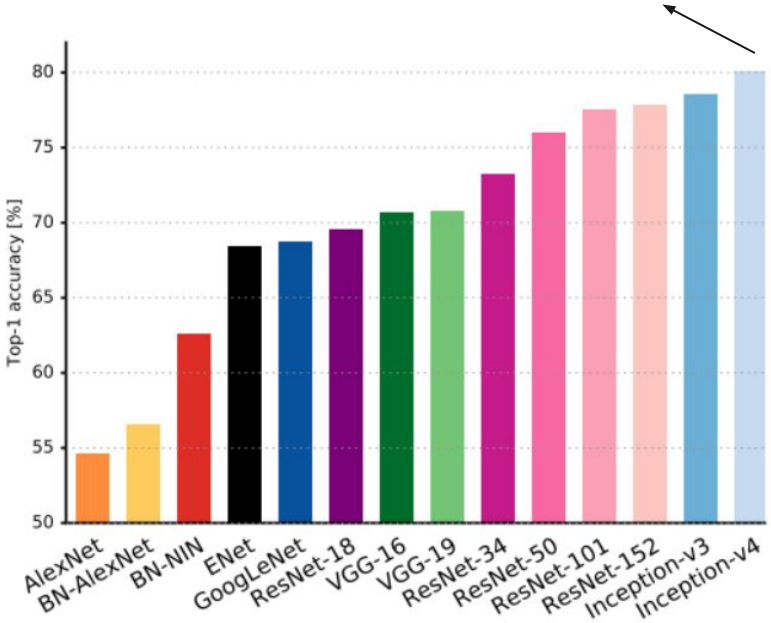




Forward pass time and power consumption

Conclusions

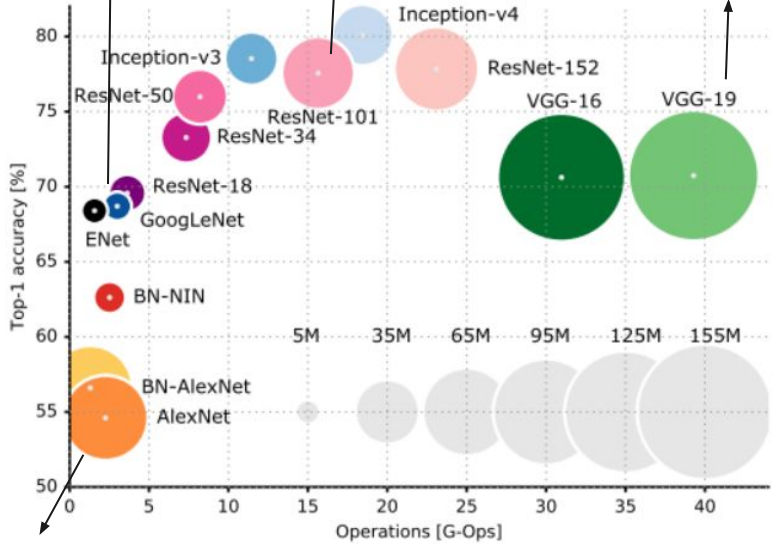
Inception-v4: resnet + Inception



GoogLeNet: most efficient

ResNet: Moderate efficiency, highest accuracy

VGG: Highest memory, most operations



AlexNet: Smaller compute, lower accuracy

Recent updates....

Recent Updates

- In 2015, Baidu was banned for a year for using different accounts to greatly exceed the specified limit of two submissions per week.
- CUImage was the winner of ILSVRC 2016:
    - Graph convolutional network (GCN) is proposed.
- In 2017, 29 of 38 competing teams had greater than 95% accuracy.
- ImageNet stated it would roll out a new, much more difficult, challenge in 2018 that involves classifying 3D objects using natural language.

Conclusion

- Deep Convolutional Neural Networks represent current state-of-the-art techniques in image classification, object detection and localization.
- Powerful CNN models are like AlexNet, InceptionNet, Deep Residual Networks.
- Large number of training data and annotations are needed.
- Cheating doesn’t help :-D

References

1. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In Advances in neural information processing systems, pp. 1097-1105. 2012.
2. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
3. Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1-9. 2015.
4. Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826. 2016.
5. Szegedy, Christian, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. "Inception-v4, inception-resnet and the impact of residual connections on learning." In Thirty-First AAAI Conference on Artificial Intelligence. 2017.
6. He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.
7. Peng, Zhanglin, Lingyun Wu, Jiamin Ren, Ruimao Zhang, and Ping Luo. "CUImage: A Neverending Learning Platform on a Convolutional Knowledge Graph of Billion Web Images." In 2018 IEEE International Conference on Big Data (Big Data), pp. 1787-1796. IEEE, 2018.

Questions?


What
WHEN
When
WHY
When
Who
HOW
WHY
What
WHAT
Where
Who
WHO
WHAT
Where
Where
HOW
WHEN
How
Why
WHEN
When
What
How
HOW
What
WHEN
Why
Why
Why
HOW
Why
How
Who
WHERE
WHO
How
Who
What
HOW
Where
HOW
Who
How
WHERE
WHO
Where
Who
Who