Advanced Machine Learning
Lecture 15

Convolutional Neural Networks

11.01.2016

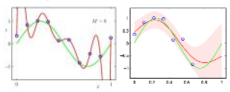
Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de/

leibe@vision.rwth-aachen.de

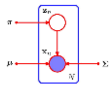
This Lecture: *Advanced Machine Learning*

$f : \mathcal{X} \rightarrow \mathbb{R}$

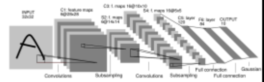
- Regression Approaches
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Gaussian Processes



- Learning with Latent Variables
  - Prob. Distributions & Approx. Inference
  - Mixture Models
  - EM and Generalizations



- Deep Learning
  - Linear Discriminants
  - Neural Networks
  - Backpropagation & Optimization
  - CNNs, RNNs, RBMs, etc.



Topics of This Lecture

- Tricks of the Trade
  - Recap
  - Initialization
  - Batch Normalization
  - Dropout
- Convolutional Neural Networks
  - Neural Networks for Computer Vision
  - Convolutional Layers
  - Pooling Layers
- CNN Architectures
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet

Recap: Data Augmentation

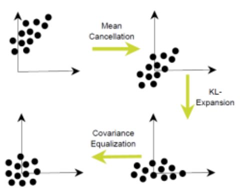
- Effect
  - Much larger training set
  - Robustness against expected variations
- During testing
  - When cropping was used during training, need to again apply crops to get same image size.
  - Beneficial to also apply flipping during test.
  - Applying several ColorPCA variations can bring another ~1% improvement, but at a significantly increased runtime.

Augmented training data
(from one original image)

Image source: Lucas Beyer

Recap: Normalizing the Inputs

- Convergence is fastest if
  - The mean of each input variable over the training set is zero.
  - The inputs are scaled such that all have the same covariance.
  - Input variables are uncorrelated if possible.



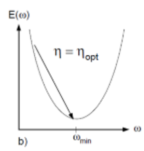
- Advisable normalization steps (for MLPs)
  - Normalize all inputs that an input unit sees to zero-mean, unit covariance.
  - If possible, try to decorrelate them using PCA (also known as Karhunen-Loeve expansion).

Image source: Yann LeCun et al., Efficient BackProp (1998)

Recap: Choosing the Right Learning Rate

- Convergence of Gradient Descent
  - Simple 1D example

$$W^{(\tau-1)} = W^{(\tau)} - \eta \frac{\mathrm{d}E(W)}{\mathrm{d}W}$$



  - What is the optimal learning rate η_{opt}?
  - If E is quadratic, the optimal learning rate is given by the inverse of the Hessian

$$\eta_{\text{opt}} = \left(\frac{\mathrm{d}^2 E(W^{(\tau)})}{\mathrm{d}W^2} \right)^{-1}$$

  - Advanced optimization techniques try to approximate the Hessian by a simplified form.
  - *If we exceed the optimal learning rate, bad things happen!*

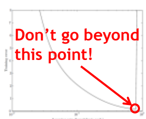
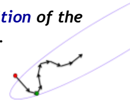


Image source: Yann LeCun et al., Efficient BackProp (1998)

Recap: Advanced Optimization Techniques

- Momentum
  - *Instead of using the gradient to change the position of the weight "particle", use it to change the velocity.*
  - Effect: dampen oscillations in directions of high curvature
  - Nesterov-Momentum: Small variation in the implementation



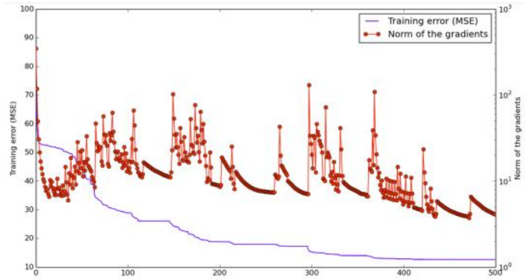
- RMS-Prop
  - *Separate learning rate for each weight: Divide the gradient by a running average of its recent magnitude.*
- AdaGrad
- AdaDelta
- Adam

Some more recent techniques, work better for some problems. Try them.

Image source: Geoff Hinton

Trick: Patience

- Saddle points dominate in high-dimensional spaces!

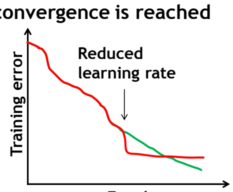


⇒ Learning often doesn't get stuck, you just may have to wait…

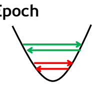
Image source: Yoshua Bengio

Reducing the Learning Rate

- Final improvement step after convergence is reached
  - Reduce learning rate by a factor of 10.
  - Continue training for a few epochs.
  - Do this 1-3 times, then stop training.



- Effect
  - Turning down the learning rate will reduce the random fluctuations in the error due to different gradients on different minibatches.



- *Be careful: Do not turn down the learning rate too soon!*
  - Further progress will be much slower after that.

Slide adapted from Geoff Hinton

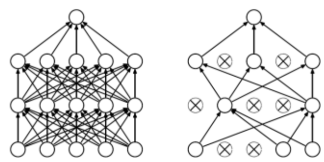
Topics of This Lecture

- Tricks of the Trade
  - Recap
  - Initialization
  - Batch Normalization
  - Dropout
- Convolutional Neural Networks
  - Neural Networks for Computer Vision
  - Convolutional Layers
  - Pooling Layers
- CNN Architectures
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet

Batch Normalization [Ioffe & Szegedy '14]

- Motivation
  - Optimization works best if all inputs of a layer are normalized.
- Idea
  - Introduce intermediate layer that centers the activations of the previous layer per minibatch.
  - I.e., perform transformations on all activations and undo those transformations when backpropagating gradients
- Effect
  - Much improved convergence

Dropout [Srivastava, Hinton '12]



(a) Standard Neural Net (b) After applying dropout.

- Idea
  - Randomly switch off units during training.
  - Change network architecture for each data point, effectively training many different variants of the network.
  - When applying the trained network, multiply activations with the probability that the unit was set to zero.

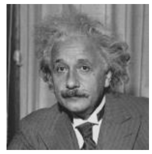
⇒ Greatly improved performance

Topics of This Lecture

- Tricks of the Trade
  - Recap
  - Initialization
  - Batch Normalization
  - Dropout
- Convolutional Neural Networks
  - Neural Networks for Computer Vision
  - Convolutional Layers
  - Pooling Layers
- CNN Architectures
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet

Neural Networks for Computer Vision

- How should we approach vision problems?



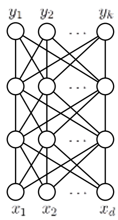
→ Face Y/N?

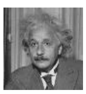
- Architectural considerations
  - Input is 2D ⇒ 2D layers of units
  - No pre-segmentation ⇒ Need robustness to misalignments
  - Vision is hierarchical ⇒ Hierarchical multi-layered structure
  - Vision is difficult ⇒ Network should be deep

Why Hierarchical Multi-Layered Models?

- Motivation 1: Visual scenes are hierarchically organized

Object	Face
↑	↑
Object parts	Eyes, nose, ...
↑	↑
Primitive features	Oriented edges
↑	↑
Input image	Face image



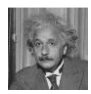


Slide adapted from Richard Turner

Why Hierarchical Multi-Layered Models?

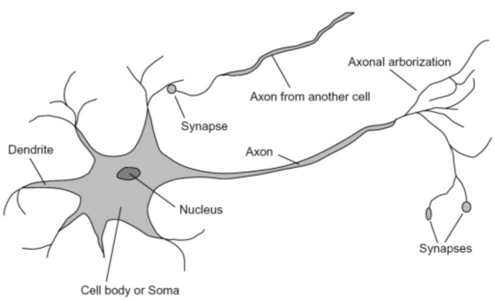
- Motivation 2: *Biological vision* is hierarchical, too

Object	Face	Inferotemporal cortex
↑	↑	
Object parts	Eyes, nose, ...	V4: different textures
↑	↑	
Primitive features	Oriented edges	V1: simple and complex cells
↑	↑	
Input image	Face image	Photoreceptors, retina



Slide adapted from Richard Turner

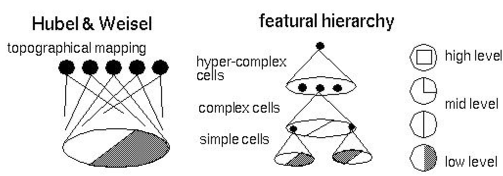
Inspiration: Neuron Cells



Slide credit: Svetlana Lazebnik, Rob Fergus

Hubel/Wiesel Architecture

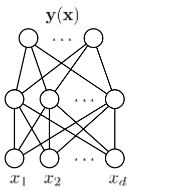
- D. Hubel, T. Wiesel (1959, 1962, Nobel Prize 1981)
  - Visual cortex consists of a hierarchy of *simple*, *complex*, and *hyper-complex* cells



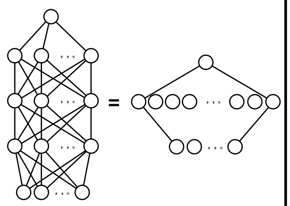
Slide credit: Svetlana Lazebnik, Rob Fergus

Why Hierarchical Multi-Layered Models?

- Motivation 3: Shallow architectures are inefficient at representing complex functions



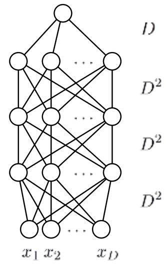
An MLP with 1 hidden layer can implement *any* function (universal approximator)



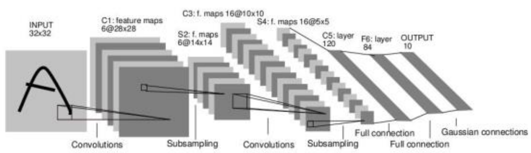
However, if the function is deep, a very large hidden layer may be required.

What's Wrong With Standard Neural Networks?

- Complexity analysis
  - How many parameters does this network have?
    $|\theta| = 3D^2 + D$
  - For a small 32×32 image
    $|\theta| = 3 \cdot 32^4 + 32^2 \approx 3 \cdot 10^6$
- Consequences
  - Hard to train
  - Need to initialize carefully
  - *Convolutional nets reduce the number of parameters!*



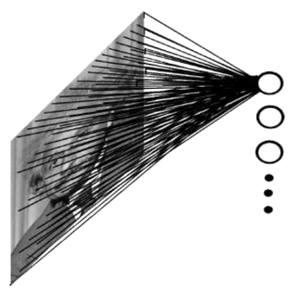
Convolutional Neural Networks (CNN, ConvNet)



- Neural network with specialized connectivity structure
  - Stack multiple stages of feature extractors
  - Higher stages compute more global, more invariant features
  - Classification layer at the end

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

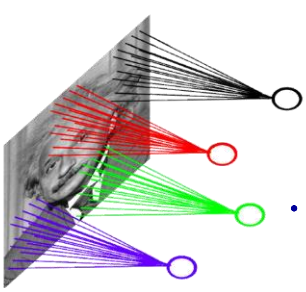
Convolutional Networks: Intuition



- Fully connected network
  - E.g. 1000×1000 image
    1M hidden units

  ⇒ 1T parameters!
- Ideas to improve this
  - Spatial correlation is local

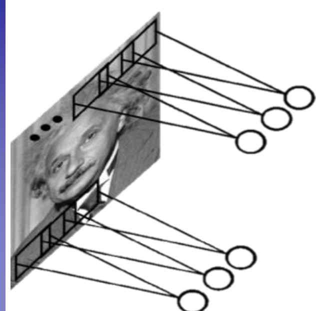
Convolutional Networks: Intuition



- Locally connected net
  - E.g. 1000×1000 image
    1M hidden units
    10×10 receptive fields

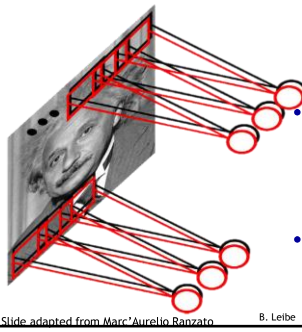
  ⇒ 100M parameters!
- Ideas to improve this
  - Spatial correlation is local
  - Want translation invariance

Convolutional Networks: Intuition



- Convolutional net
  - Share the same parameters across different locations
  - Convolutions with learned kernels

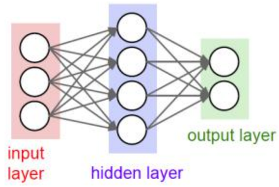
Convolutional Networks: Intuition



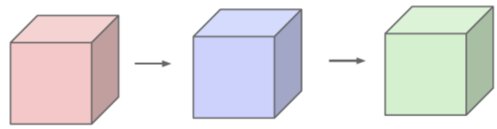
- Convolutional net
  - Share the same parameters across different locations
  - Convolutions with learned kernels
- Learn *multiple* filters
  - E.g. 1000×1000 image
    100 filters
    10×10 filter size
  - ⇒ 10k parameters
- Result: Response map
  - size: 1000×1000×100
  - Only memory, not params!

Important Conceptual Shift

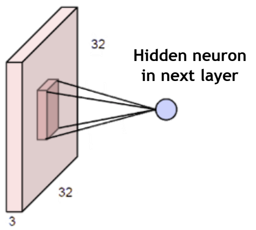
- Before



- Now:



Convolution Layers



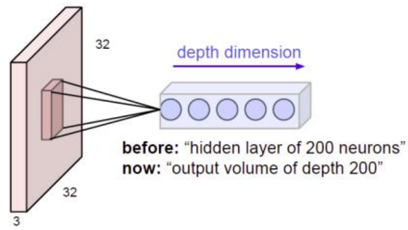
Example
image: 32×32×3 volume

Before: Full connectivity
32×32×3 weights

Now: Local connectivity
One neuron connects to, e.g., 5×5×3 region.
⇒ Only 5×5×3 shared weights.

- Note: Connectivity is
  - Local in space (5×5 inside 32×32)
  - But full in depth (all 3 depth channels)

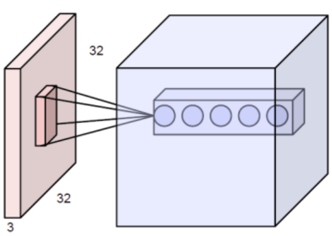
Convolution Layers



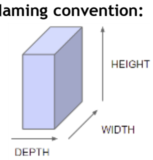
before: "hidden layer of 200 neurons"
now: "output volume of depth 200"

- All Neural Net activations arranged in 3 dimensions
  - Multiple neurons all looking at the same input region, stacked in depth

Convolution Layers

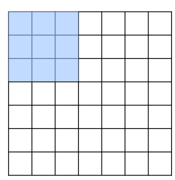


Naming convention:



- All Neural Net activations arranged in 3 dimensions
  - Multiple neurons all looking at the same input region, stacked in depth
  - Form a single [1×1×depth] depth column in output volume.

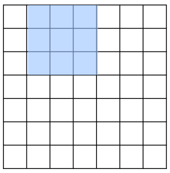
Convolution Layers



Example:
7×7 input
assume 3×3 connectivity
stride 1

- Replicate this column of hidden neurons across space, with some stride.

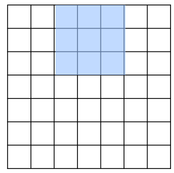
Convolution Layers



Example:
7×7 input
assume 3×3 connectivity
stride 1

- Replicate this column of hidden neurons across space, with some stride.

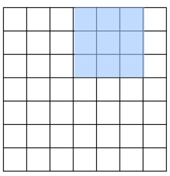
Convolution Layers



Example:
7×7 input
assume 3×3 connectivity
stride 1

- Replicate this column of hidden neurons across space, with some stride.

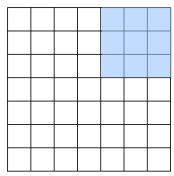
Convolution Layers



Example:
7×7 input
assume 3×3 connectivity
stride 1

- Replicate this column of hidden neurons across space, with some stride.

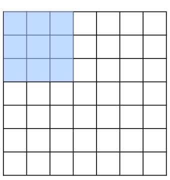
Convolution Layers



Example:
7×7 input
assume 3×3 connectivity
stride 1
⇒ 5×5 output

- Replicate this column of hidden neurons across space, with some stride.

Convolution Layers

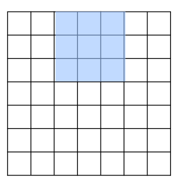


Example:
7×7 input
assume 3×3 connectivity
stride 1
⇒ 5×5 output

What about stride 2?

- Replicate this column of hidden neurons across space, with some stride.

Convolution Layers

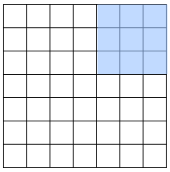


Example:
7×7 input
assume 3×3 connectivity
stride 1
⇒ 5×5 output

What about stride 2?

- Replicate this column of hidden neurons across space, with some stride.

Convolution Layers



Example:
7×7 input
assume 3×3 connectivity
stride 1
⇒ 5×5 output

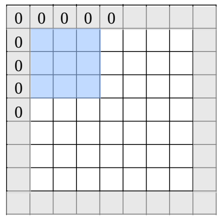
What about stride 2?
⇒ 3×3 output

- Replicate this column of hidden neurons across space, with some stride.

Advanced Machine Learning Winter'15

Slide credit: FeiFei Li, Andrej Karpathy B. Leibe 38

Convolution Layers



Example:
7×7 input
assume 3×3 connectivity
stride 1
⇒ 5×5 output

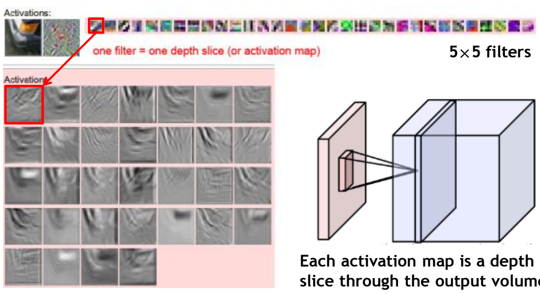
What about stride 2?
⇒ 3×3 output

- Replicate this column of hidden neurons across space, with some stride.
- In practice, common to zero-pad the border.
  - Preserves the size of the input spatially.

Advanced Machine Learning Winter'15

Slide credit: FeiFei Li, Andrej Karpathy B. Leibe 39

Activation Maps of Convolutional Filters



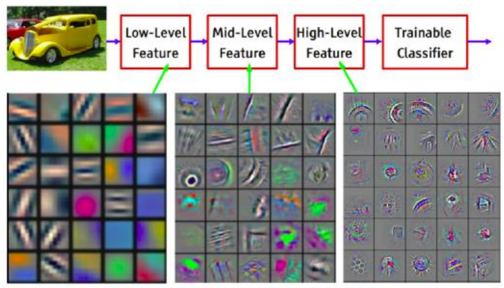
Activation maps

Each activation map is a depth slice through the output volume.

Advanced Machine Learning Winter'15

Slide adapted from FeiFei Li, Andrej Karpathy B. Leibe 40

Effect of Multiple Convolution Layers

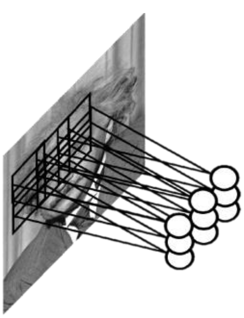


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Advanced Machine Learning Winter'15

Slide credit: Yann LeCun B. Leibe 41

Convolutional Networks: Intuition

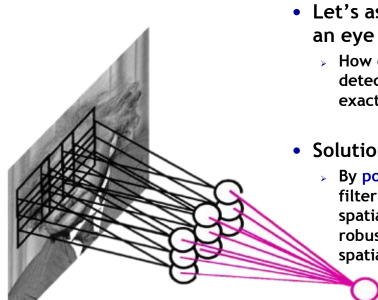


- Let's assume the filter is an eye detector
  - How can we make the detection robust to the exact location of the eye?

Advanced Machine Learning Winter'15

Slide adapted from Marc'Aurelio Ranzato B. Leibe 42 Image source: Yann LeCun

Convolutional Networks: Intuition



- Let's assume the filter is an eye detector
  - How can we make the detection robust to the exact location of the eye?
- Solution:
  - By pooling (e.g., max or avg) filter responses at different spatial locations, we gain robustness to the exact spatial location of features.

Advanced Machine Learning Winter'15

Slide adapted from Marc'Aurelio Ranzato B. Leibe 43 Image source: Yann LeCun

Max Pooling

Single depth slice

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

x, y

max pool with 2x2 filters and stride 2

6	8
3	4

- **Effect:**
  - Make the representation smaller without losing too much information
  - Achieve robustness to translations

Max Pooling

Single depth slice

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

x, y

max pool with 2x2 filters and stride 2

6	8
3	4

- **Note**
  - Pooling happens independently across each slice, preserving the number of slices.

CNNs: Implication for Back-Propagation

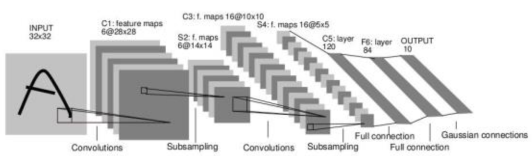
- Convolutional layers
  - Filter weights are shared between locations

  ⇒ Gradients are added for each filter location.

Topics of This Lecture

- Tricks of the Trade
  - Recap
  - Initialization
  - Batch Normalization
  - Dropout
- Convolutional Neural Networks
  - Neural Networks for Computer Vision
  - Convolutional Layers
  - Pooling Layers
- **CNN Architectures**
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet

CNN Architectures: LeNet (1998)



- Early convolutional architecture
  - 2 Convolutional layers, 2 pooling layers
  - Fully-connected NN layers for classification
  - Successfully used for handwritten digit recognition (MNIST)

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

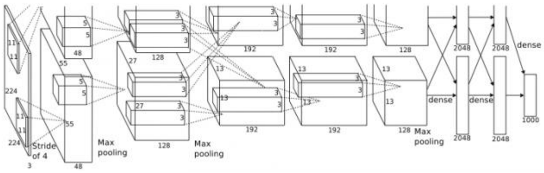
ImageNet Challenge 2012

- ImageNet
  - ~14M labeled internet images
  - 20k classes
  - Human labels via Amazon Mechanical Turk

[Deng et al., CVPR'09]

- Challenge (ILSVRC)
  - 1.2 million training images
  - 1000 classes
  - Goal: Predict ground-truth class within top-5 responses
  - Currently one of the top benchmarks in Computer Vision

CNN Architectures: AlexNet (2012)

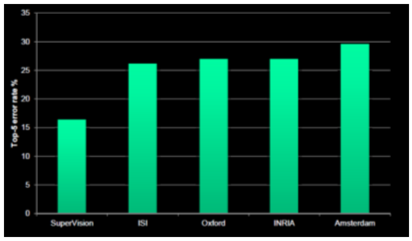


- Similar framework as LeNet, but
  - Bigger model (7 hidden layers, 650k units, 60M parameters)
  - More data (10^6 images instead of 10^3)
  - GPU implementation
  - Better regularization and up-to-date tricks for training (Dropout)

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012.

Image source: A. Krizhevsky, I. Sutskever and G.E. Hinton, NIPS 2012

ILSVRC 2012 Results



- AlexNet almost halved the error rate
  - 16.4% error (top-5) vs. 26.2% for the next best approach

⇒ A revolution in Computer Vision

  - Acquired by Google in Jan '13, deployed in Google+ in May '13

AlexNet Results

Image source: A. Krizhevsky, I. Sutskever and G.E. Hinton, NIPS 2012

AlexNet Results

Test image Retrieved images

Image source: A. Krizhevsky, I. Sutskever and G.E. Hinton, NIPS 2012

CNN Architectures: VGGNet (2015)

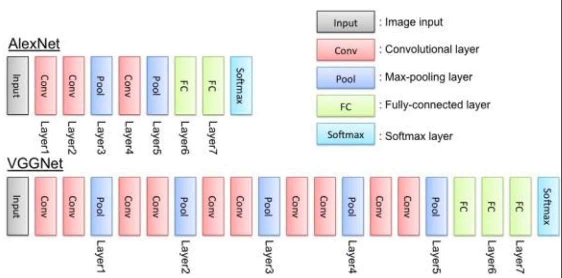
- Main ideas
  - Deeper network
  - Stacked convolutional layers with smaller filters (+ nonlinearity)
  - Detailed evaluation of all components

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Mainly used

Image source: Simonyan & Zisserman

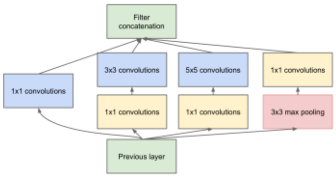
Comparison to AlexNet



K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

Image source: Hirokatsu Kataoka

CNN Architectures: GoogLeNet (2014)

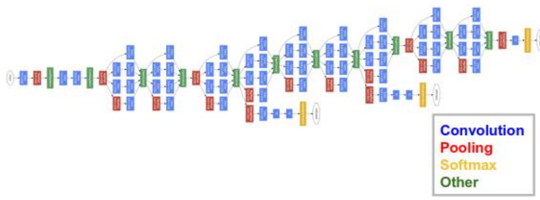


(b) Inception module with dimension reductions

- **Main ideas**
  - "Inception" module as modular component
  - Learns filters at several scales within each module

C. Szegedy, W. Liu, Y. Jia, et al, Going Deeper with Convolutions, arXiv:1409.4842, 2014.

GoogLeNet Visualization



Results on ILSVRC

Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	**23.7**	**6.8**	**6.8**
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	**6.7**	
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-

References and Further Reading

- **LeNet**
  - Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.
- **AlexNet**
  - A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012.
- **VGGNet**
  - K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015
- **GoogLeNet**
  - C. Szegedy, W. Liu, Y. Jia, et al, Going Deeper with Convolutions, arXiv:1409.4842, 2014.