10703 Deep Reinforcement Learning and Control

Russ Salakhutdinov

Machine Learning Department
rsalakhu@cs.cmu.edu

Deep Learning II

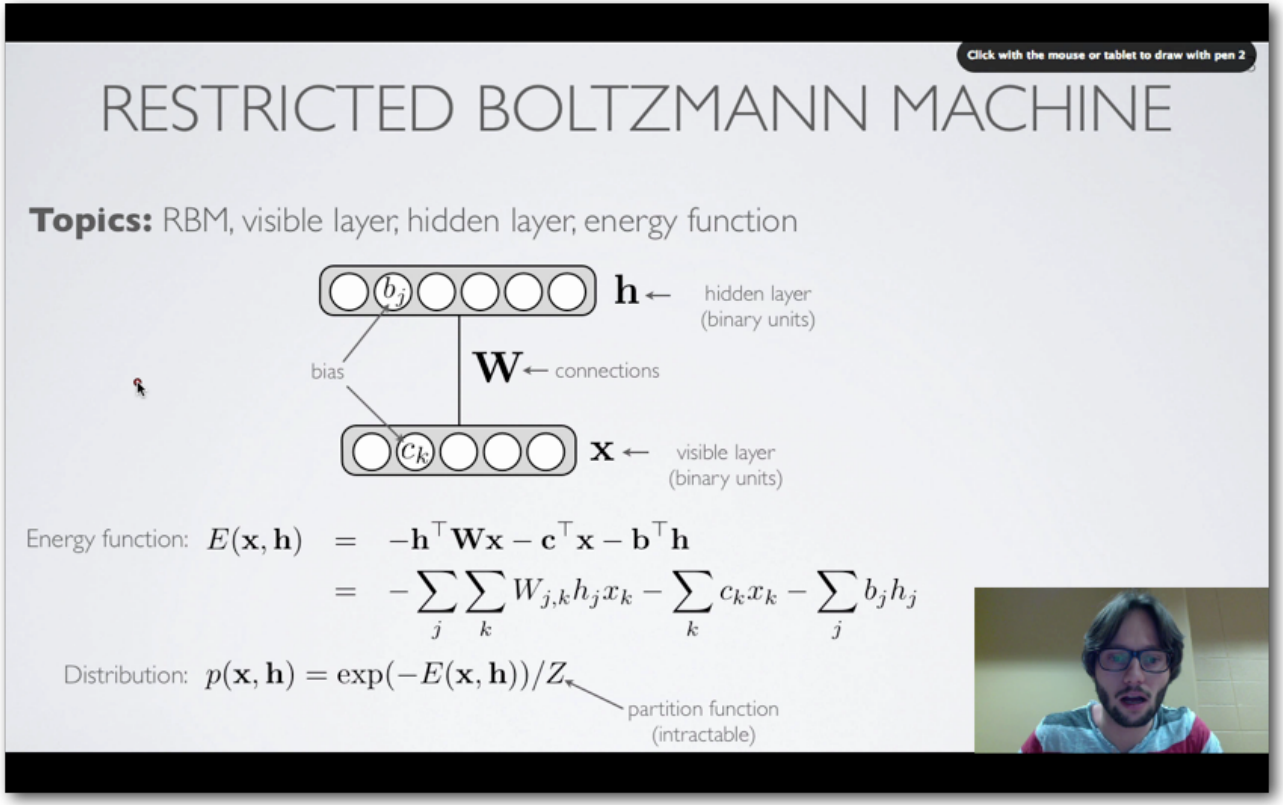
Neural Networks Online Course

• **Disclaimer**: Some of the material and slides for this lecture were borrowed from Hugo Larochelle's class on Neural Networks: https://sites.google.com/site/deeplearningsummerschool2016/

• Hugo's class covers many other topics: convolutional networks, neural language model, Boltzmann machines, autoencoders, sparse coding, etc.

• We will use his material for some of the other lectures.

http://info.usherbrooke.ca/hlarochelle/neural_networks

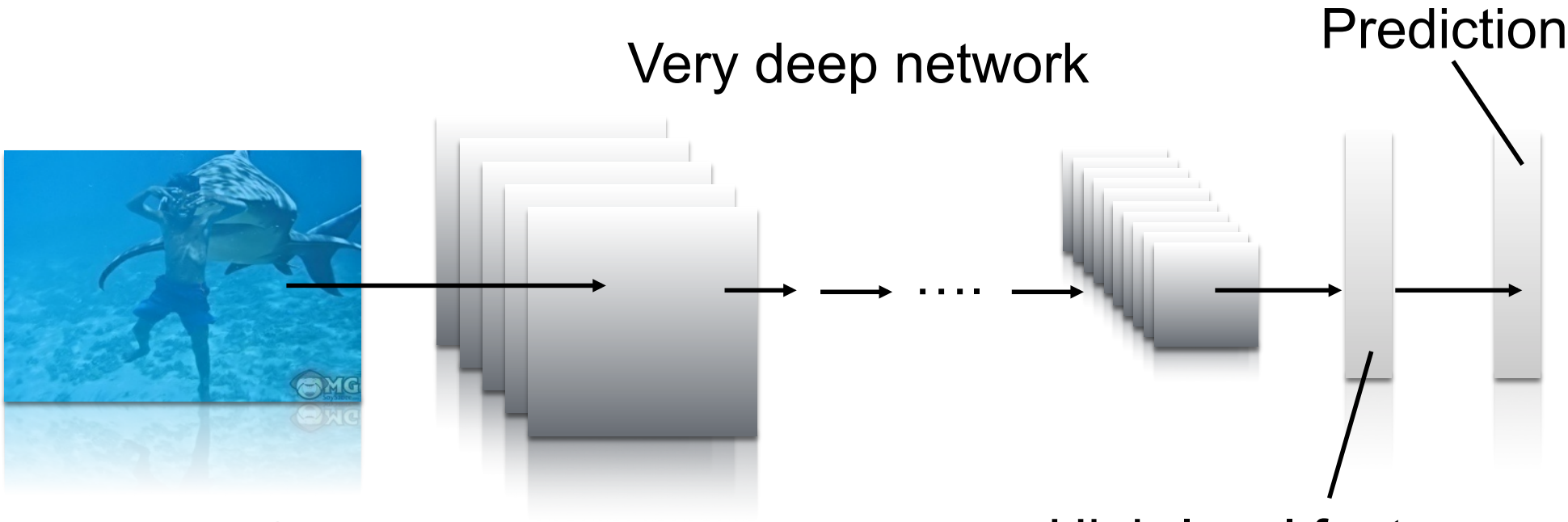


Computer Vision

- Design algorithms that can process visual data to accomplish a given task:

  - For example, object recognition: Given an input image, identify which object it contains

Deep Convolutional Nets



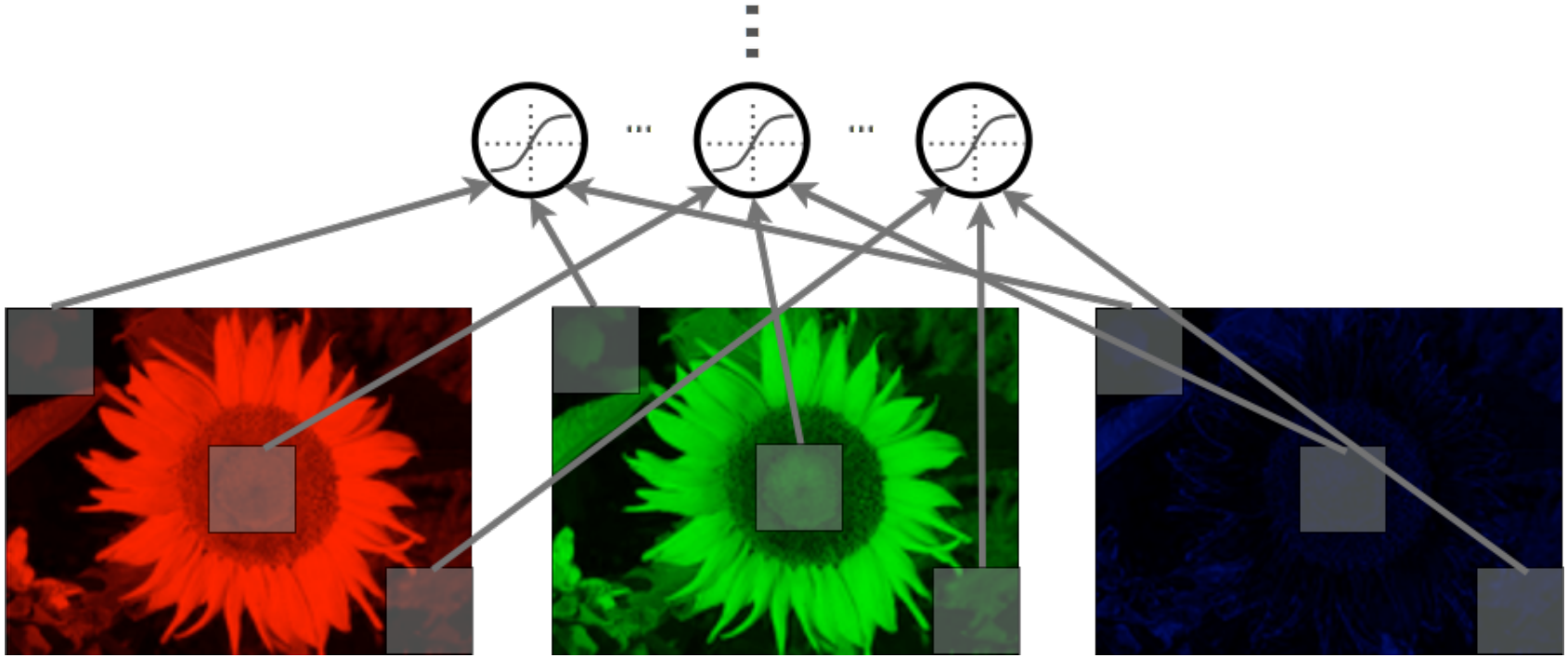
- Convolution
- Pooling
- Normalization
- Densely connected

Computer Vision

- Convolutional networks leverage these ideas
  - Local connectivity
  - Parameter sharing
  - Convolution
  - Pooling / subsampling hidden units

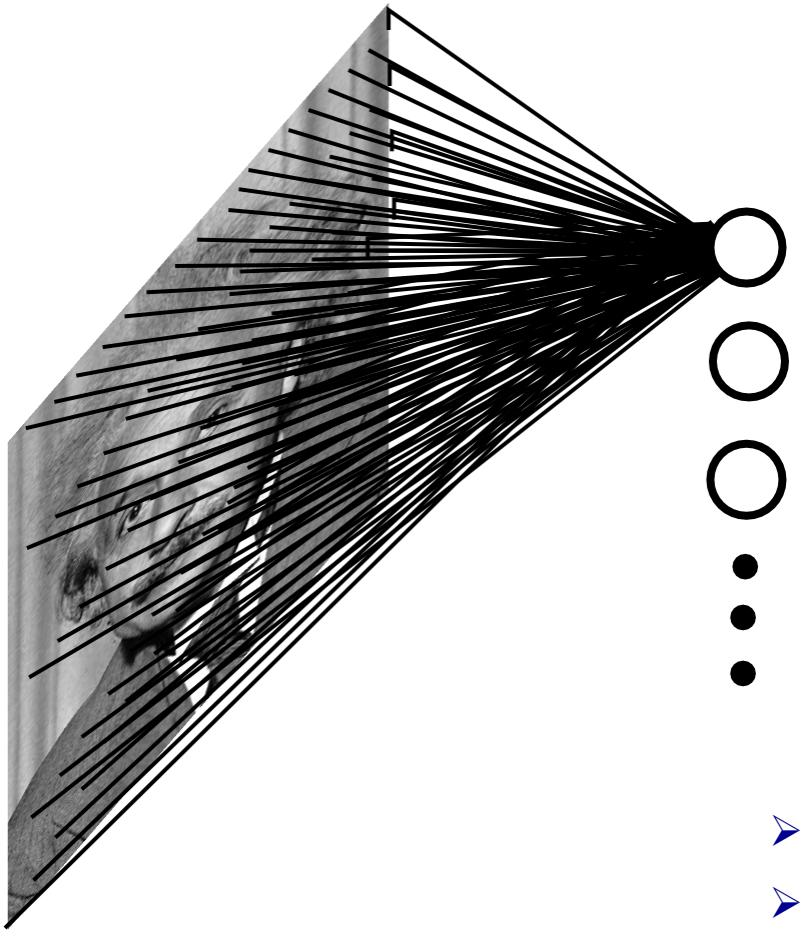
Local Connectivity

- Units are connected to all channels:
  - 1 channel if grayscale image,
  - 3 channels (R, G, B) if color image



Local Connectivity

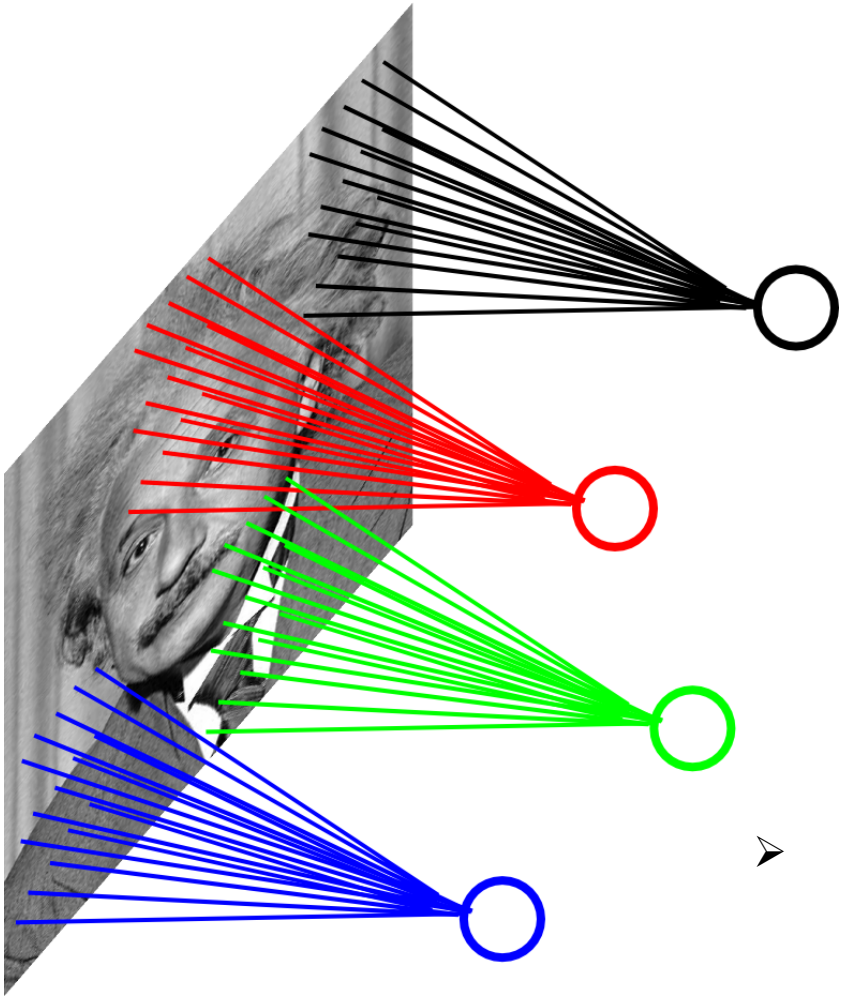
- Example: 200x200 image, 40K hidden units, ~2B parameters!



- Spatial correlation is local
- Too many parameters, will require a lot of training data!

Local Connectivity

• Example: 200x200 image, 40K hidden units, filter size 10x10, 4M parameters!



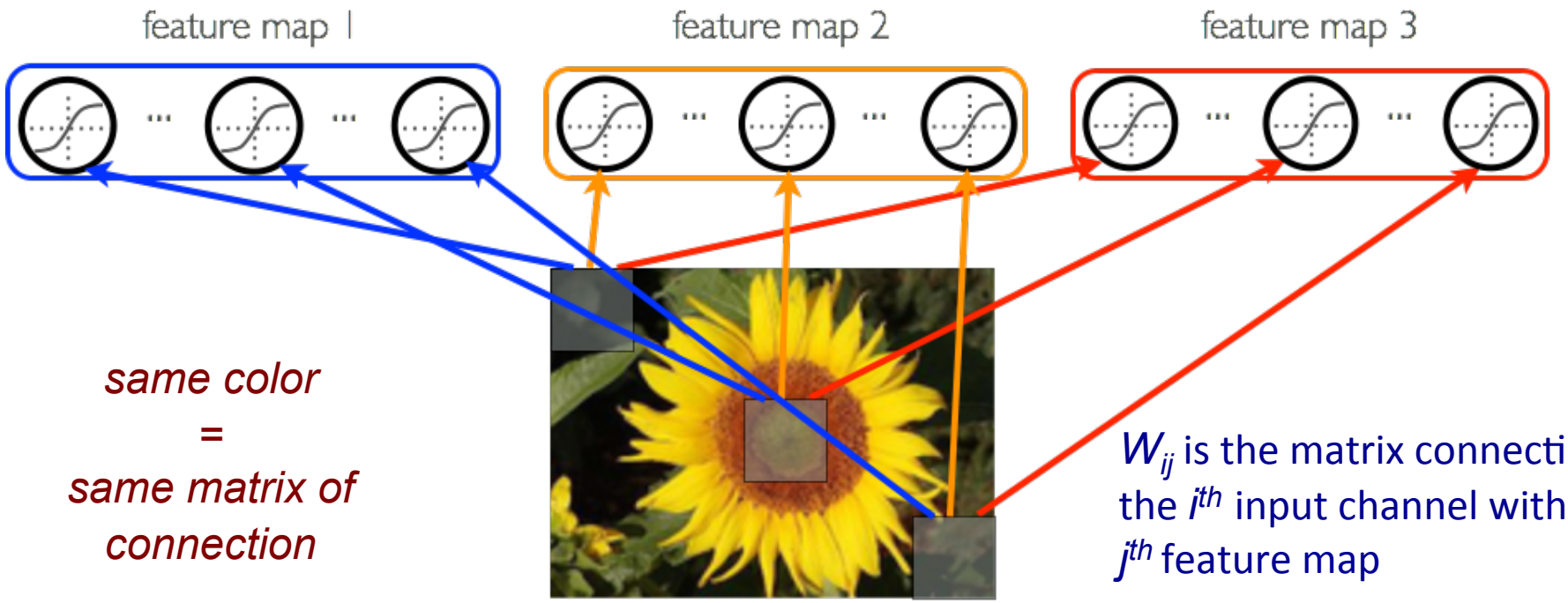
➢ This parameterization is good when input image is registered

Computer Vision

- Convolutional networks leverage these ideas
  - Local connectivity
  - Parameter sharing
  - Convolution
  - Pooling / subsampling hidden units

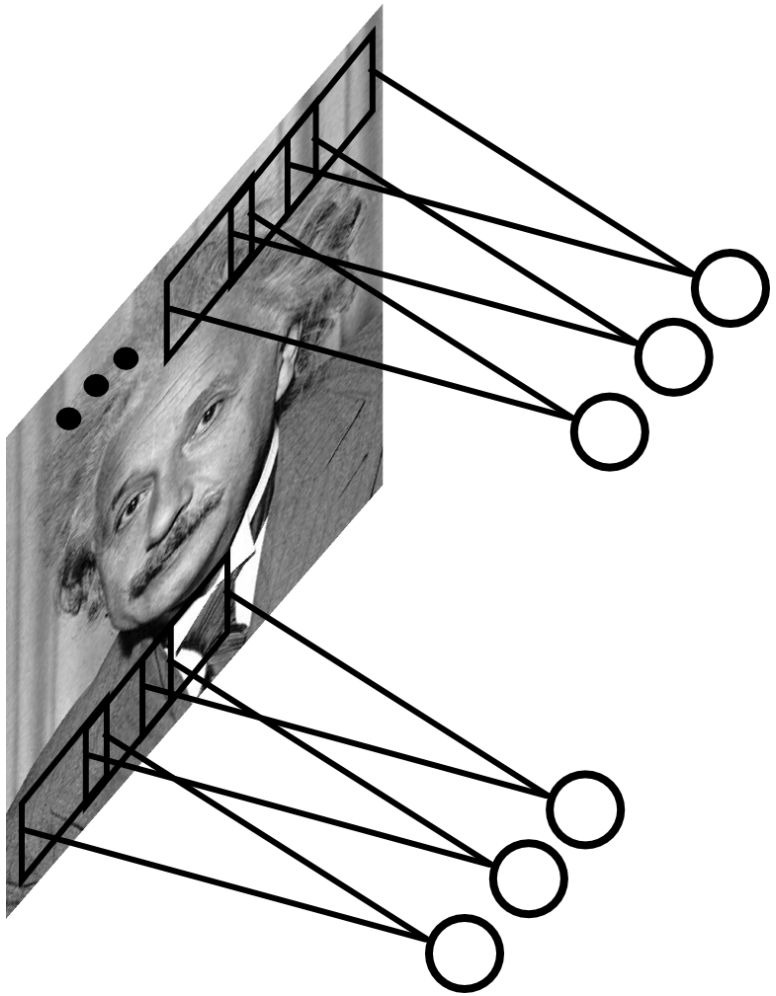
Parameter Sharing

- Share matrix of parameters across some units
  - Units that are organized into the 'feature map" share parameters
  - Hidden units within a feature map cover different positions in the image



Parameter Sharing

- Share matrix of parameters across certain units



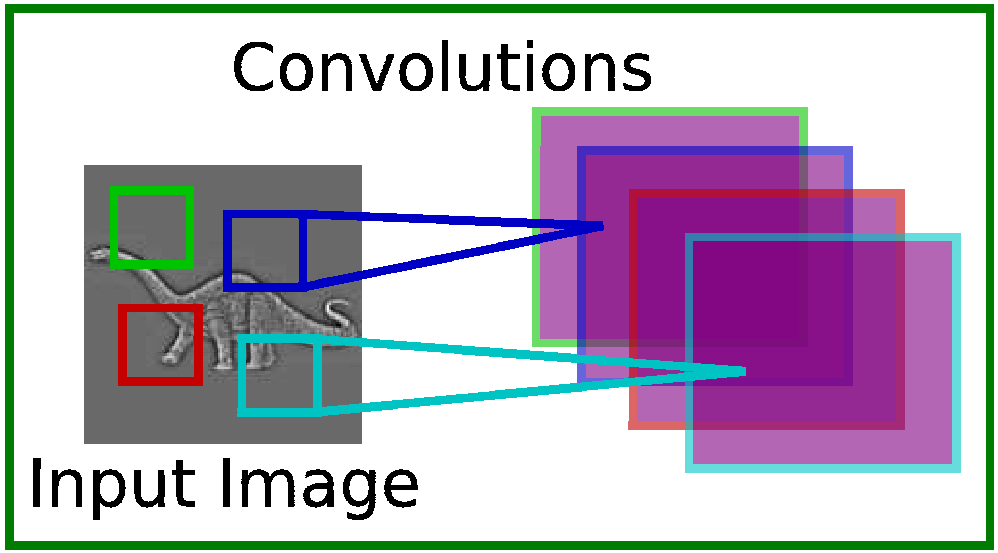
- Convolutions with certain kernels

Computer Vision

- Convolutional networks leverage these ideas
  - Local connectivity
  - Parameter sharing
  - Convolution
  - Pooling / subsampling hidden units

Parameter Sharing

- Each feature map forms a 2D grid of features
  - can be computed with a discrete convolution ($*$) of a kernel matrix k_{ij} which is the hidden weights matrix W_{ij} with its rows and columns flipped



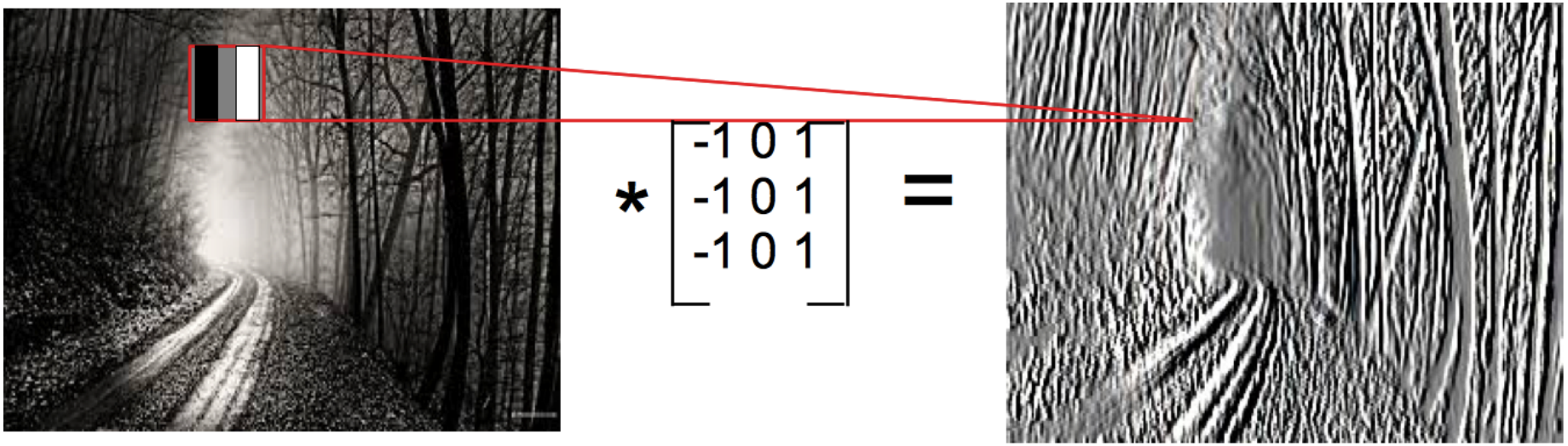
Jarret et al. 2009

$$y_j = g_j \tanh(\sum_i k_{ij} * x_i)$$

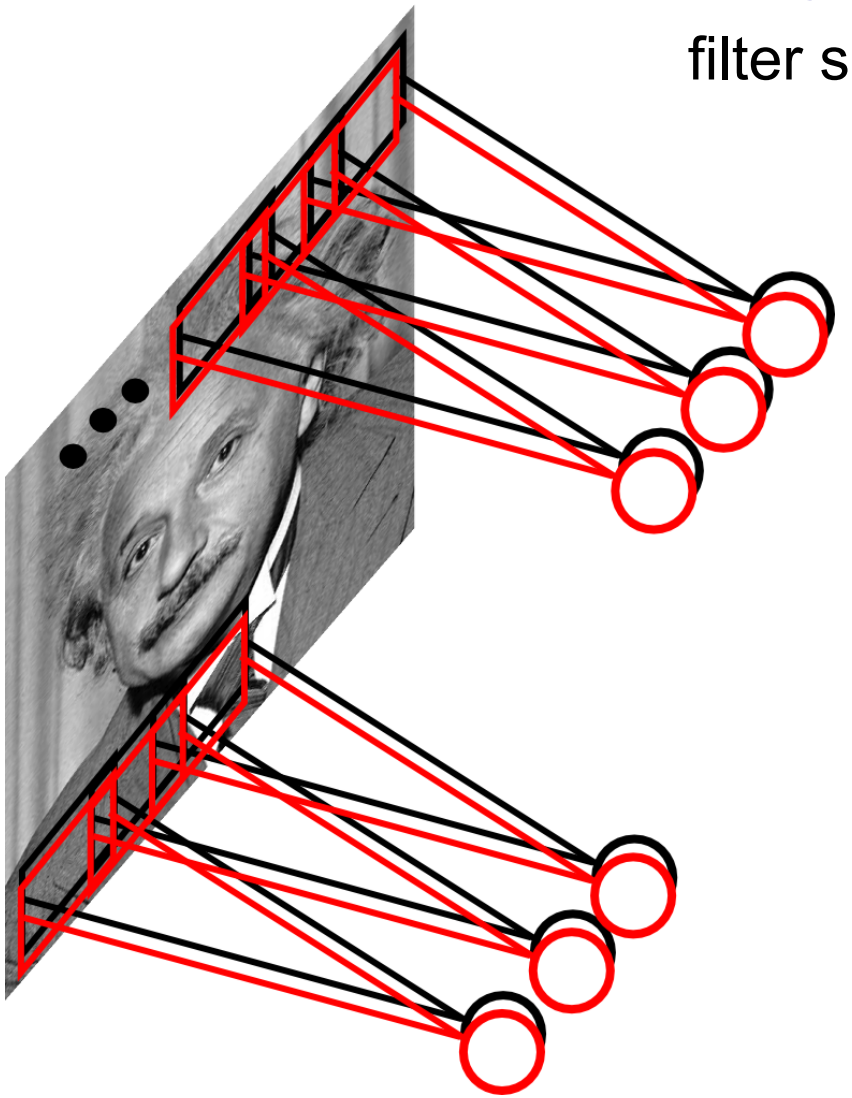
- x_i is the i[th] channel of input
- k_{ij} is the convolution kernel
- g_j is a learned scaling factor
- g_j is the hidden layer

can add bias

Example



Multiple Feature Maps



- Example: 200x200 image, 100 filters, filter size 10x10, 10K parameters

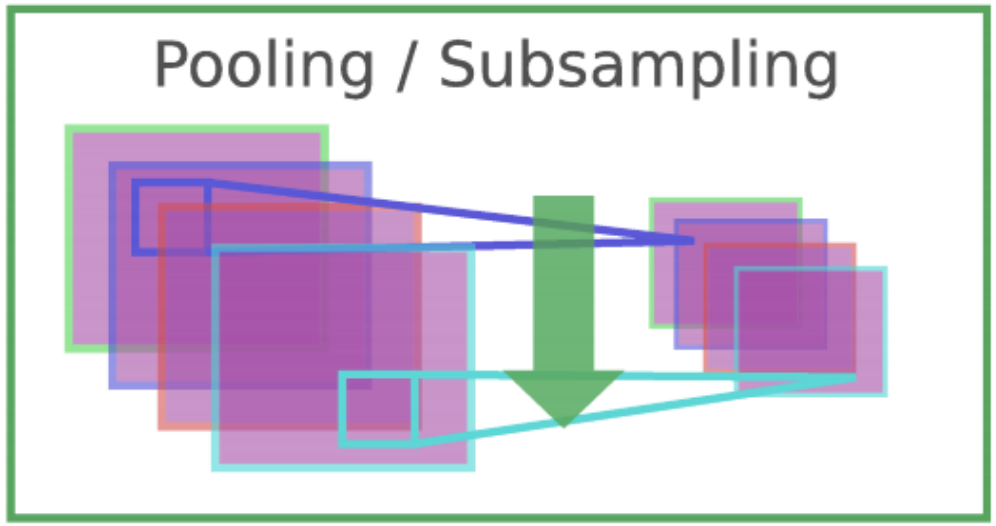
Computer Vision

- Convolutional networks leverage these ideas
  - Local connectivity
  - Parameter sharing
  - Convolution
  - Pooling / subsampling hidden units

Pooling

- Pool hidden units in same neighborhood
  - pooling is performed in non-overlapping neighborhoods (subsampling)

$$y_{ijk} = \max_{p,q} x_{i,j+p,k+q}$$

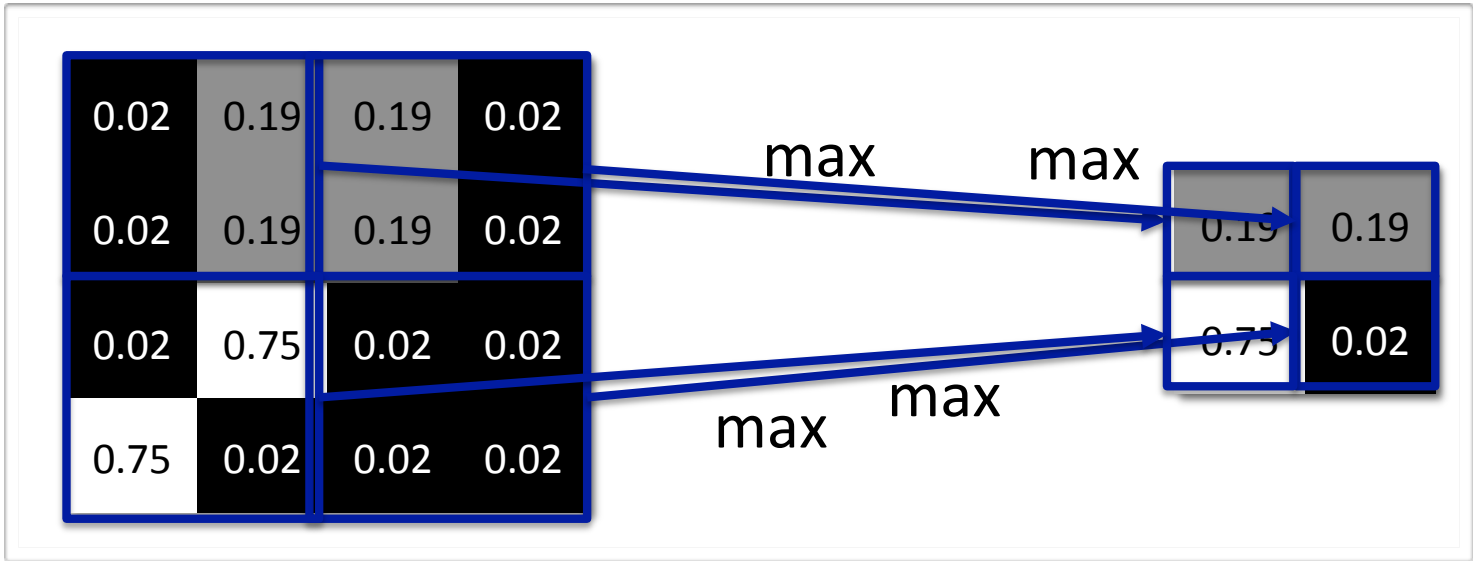


Jarret et al. 2009

- x_i is the i^{th} channel of input
- $x_{i,j,k}$ is value of the i^{th} feature map at position j,k
- p is vertical index in local neighborhood
- q is horizontal index in local neighborhood
- y_{ijk} is pooled / subsampled layer

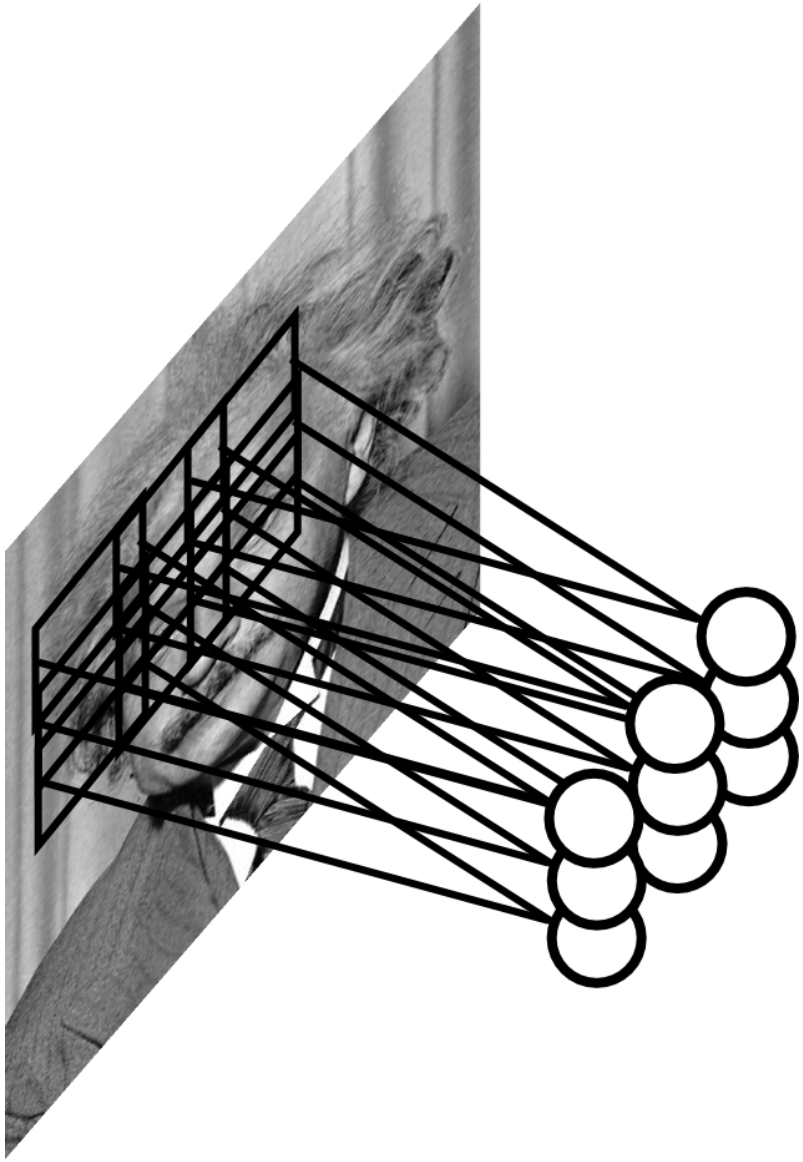
Example: Pooling

- Illustration of pooling/subsampling operation



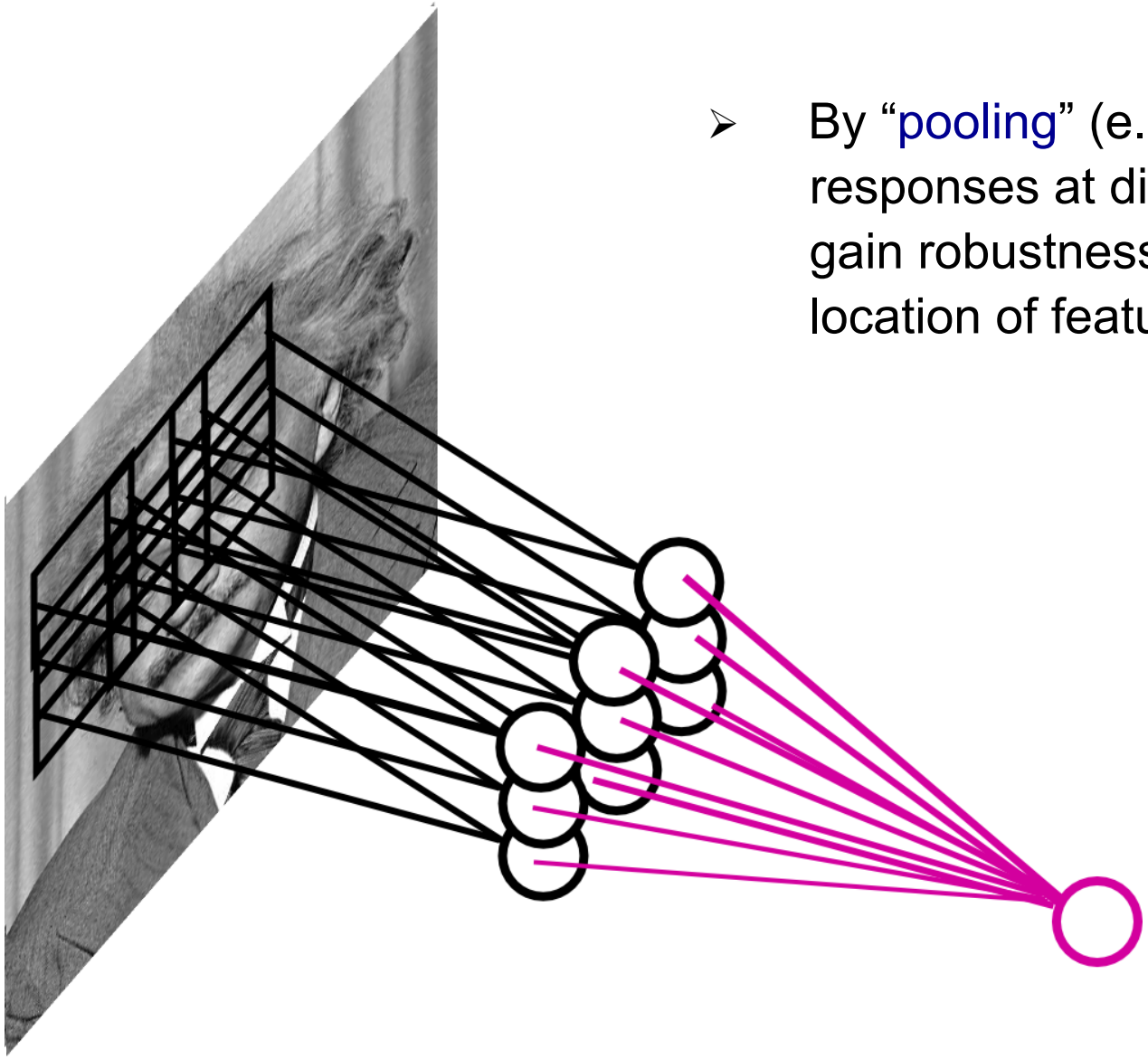
- Why pooling?
  - Introduces invariance to local translations
  - Reduces the number of hidden units in hidden layer

Example: Pooling



- can we make the detection robust to the exact location of the eye?

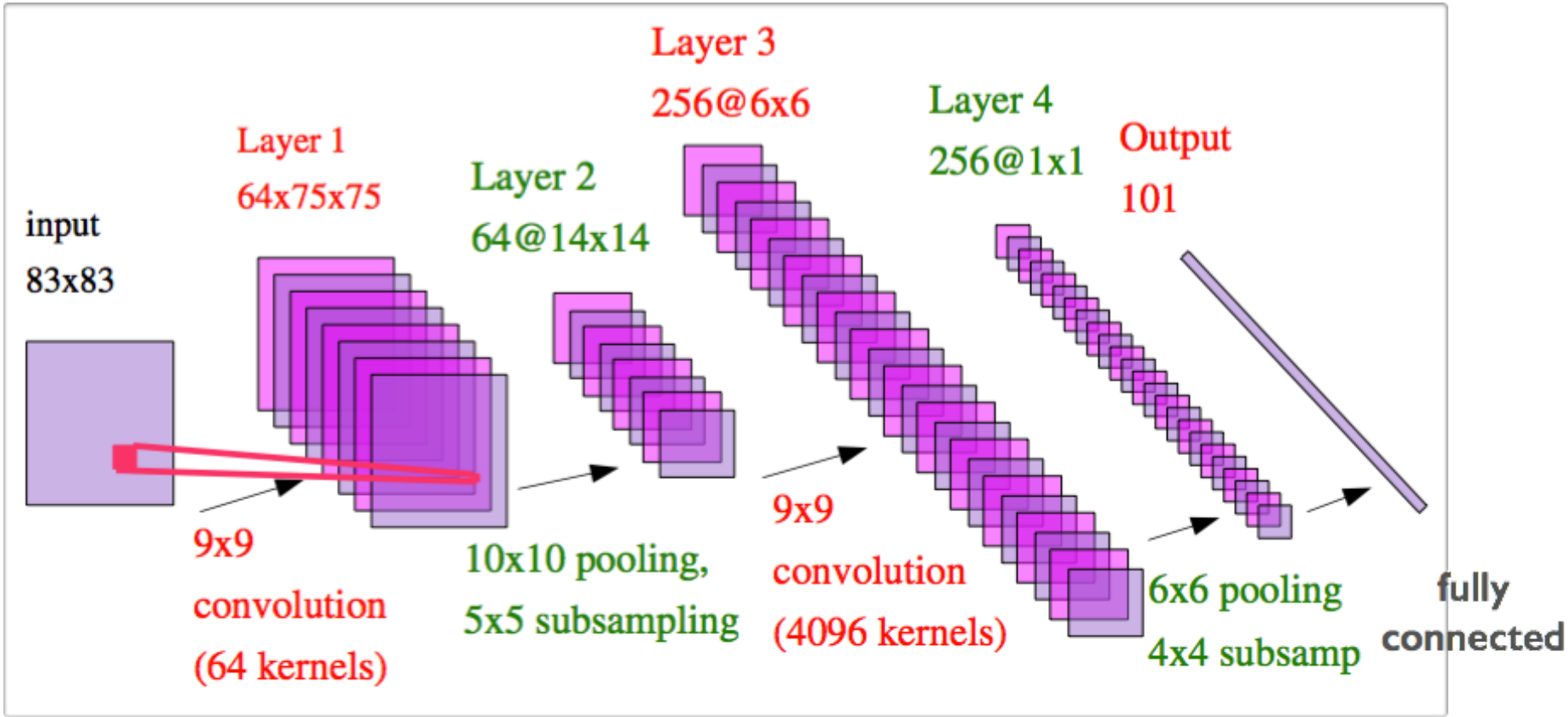
Example: Pooling



- By “pooling” (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

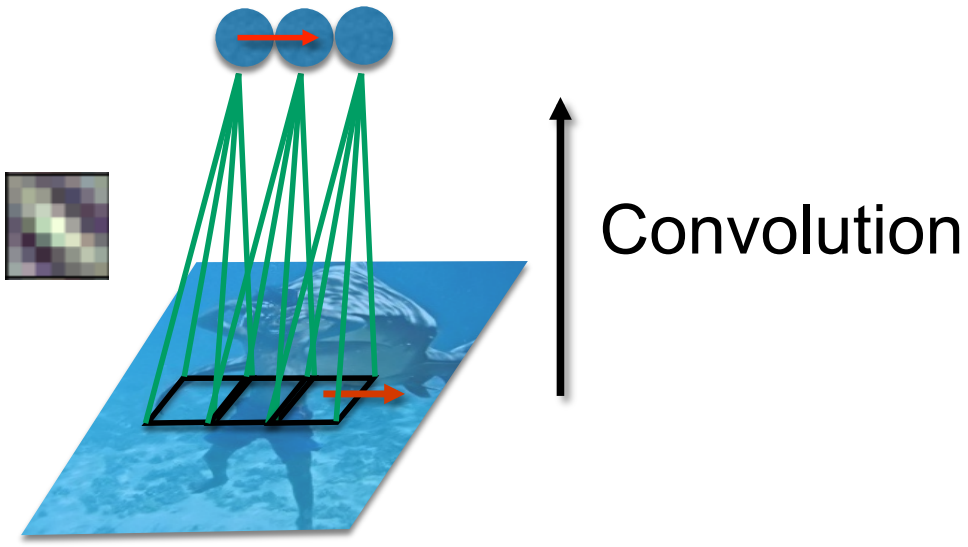
Convolutional Network

• Convolutional neural network alternates between the convolutional and pooling layers

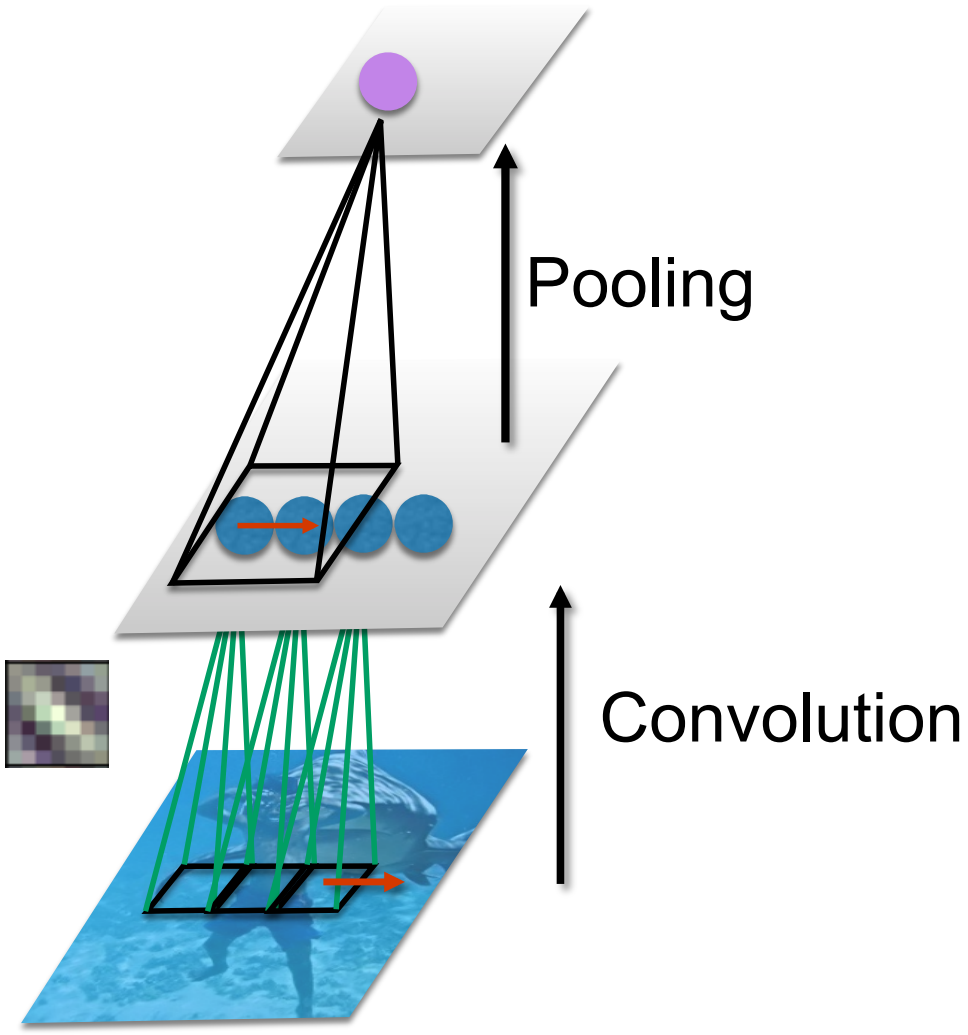


From Yann LeCun's slides

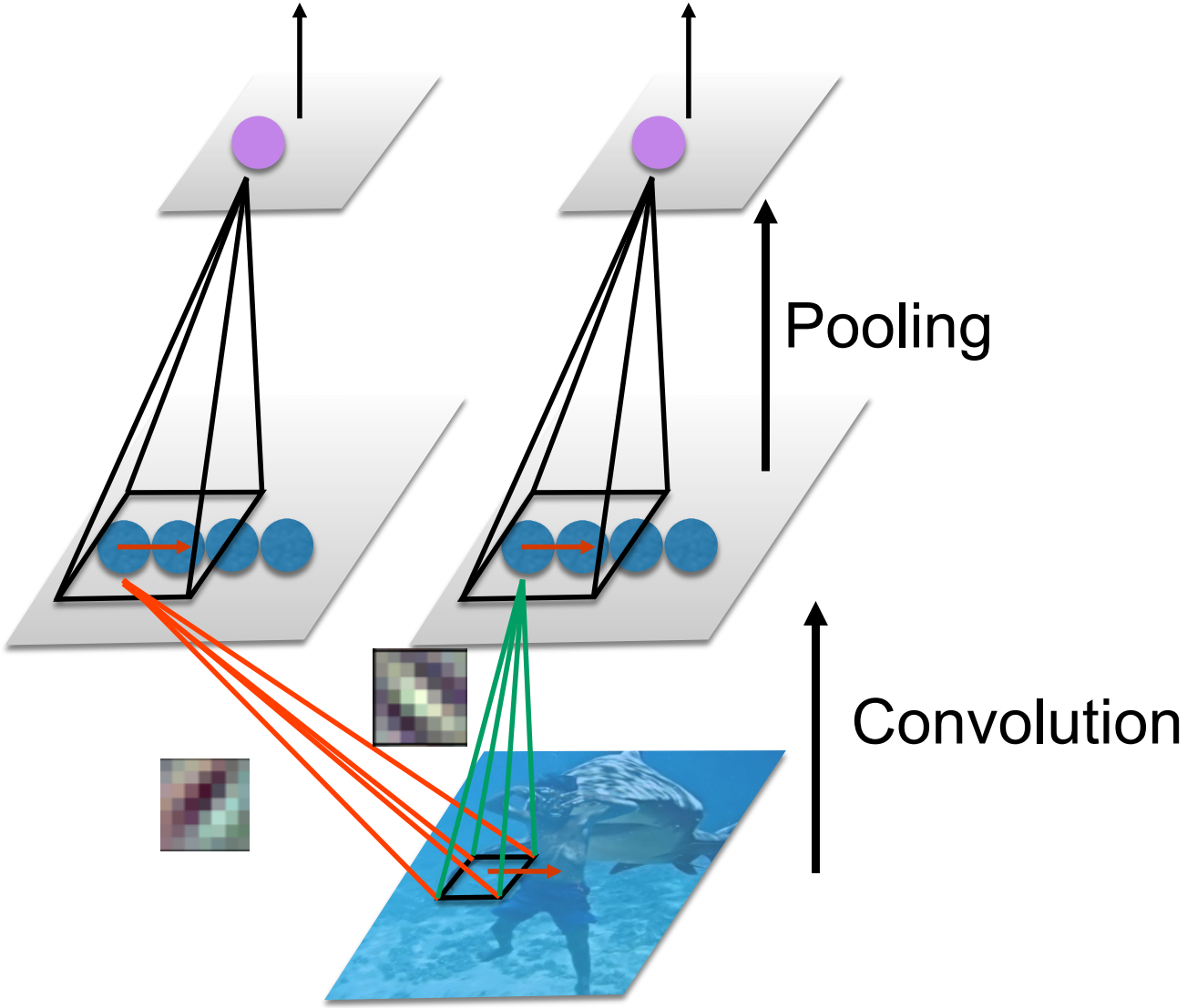
Deep Convolutional Nets



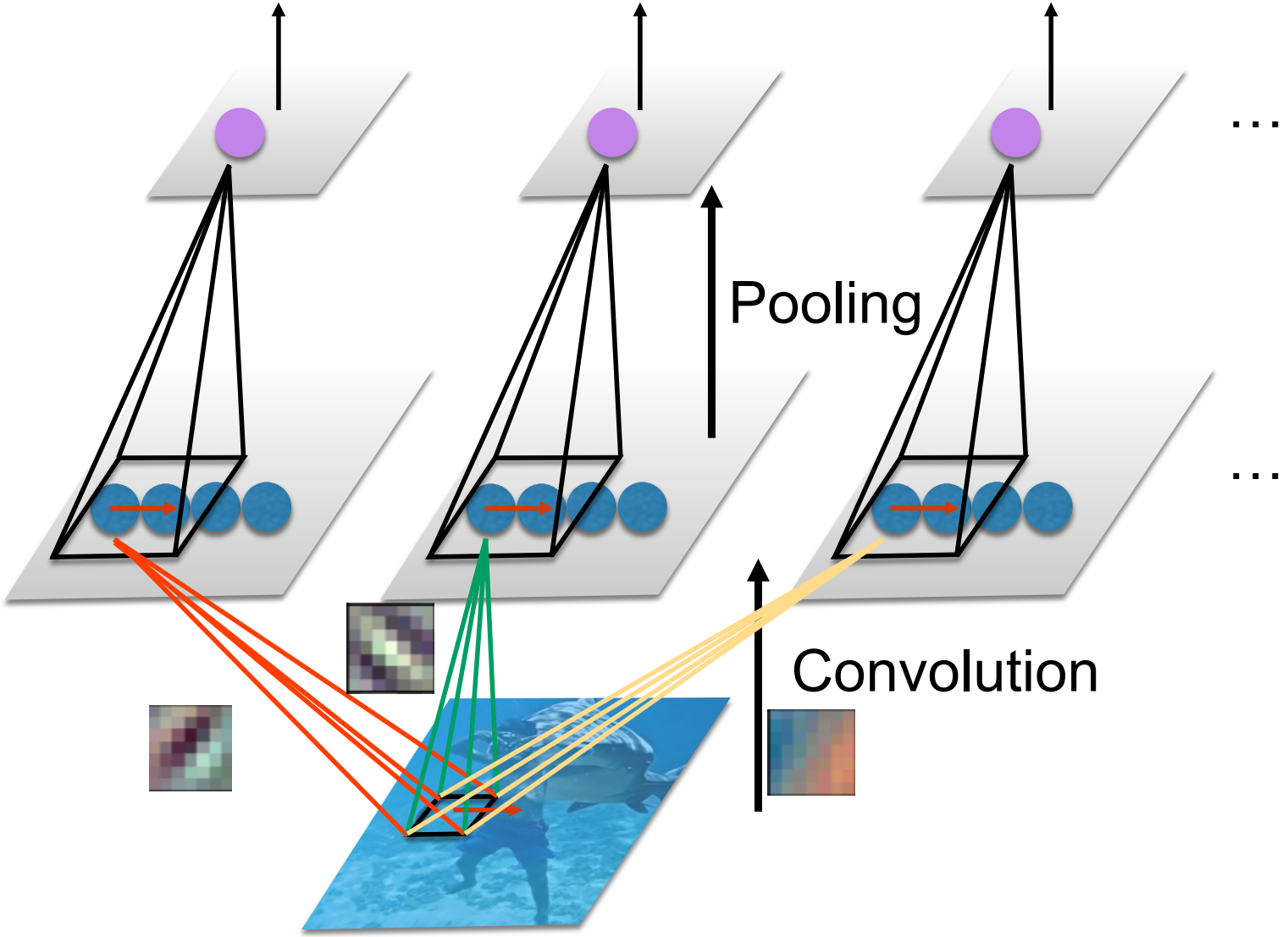
Deep Convolutional Nets



Deep Convolutional Nets



Deep Convolutional Nets



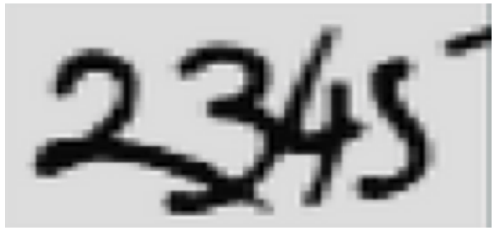
Convolutional Network

- For classification: Output layer is a regular, fully connected layer with softmax non-linearity
  - Output provides an estimate of the conditional probability of each class

- The network is trained by stochastic gradient descent
  - Backpropagation is used similarly as in a fully connected network
  - Pass gradients through element-wise activation function
  - We also need to pass gradients through the convolution operation and the pooling operation

ConvNets: Examples

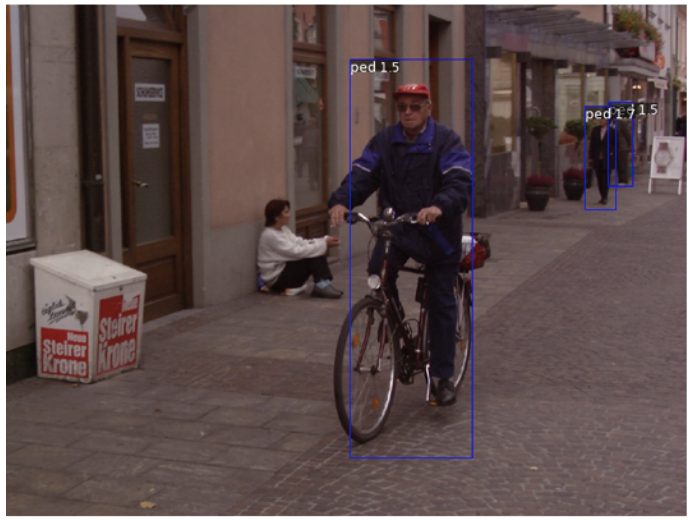
- Optical Character Recognition, House Number and Traffic Sign classification

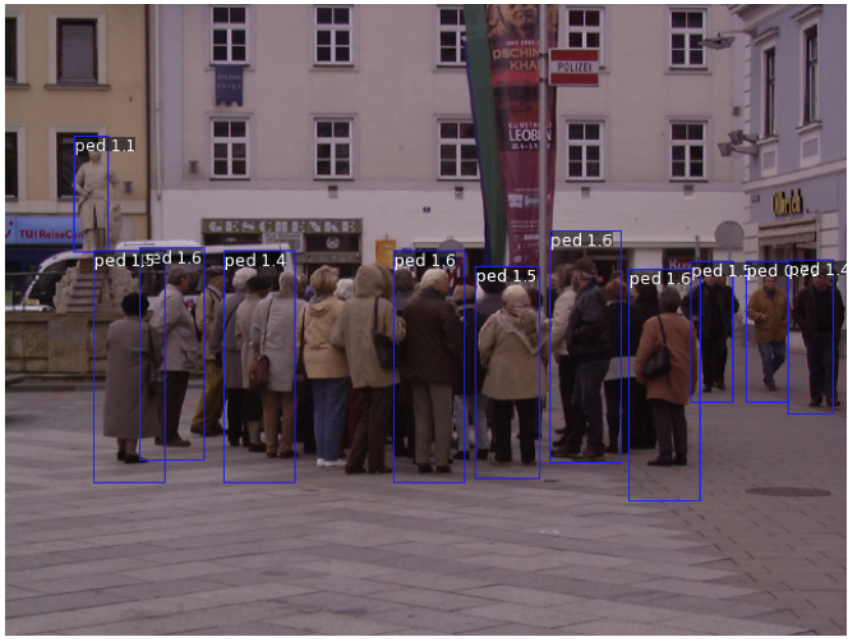


Ciresan et al. “MCDNN for image classification” CVPR 2012
Wan et al. “Regularization of neural networks using dropconnect” ICML 2013
Goodfellow et al. “Multi-digit nuber recognition from StreetView...” ICLR 2014
Jaderberg et al. “Synthetic data and ANN for natural scene text recognition” arXiv 2014

ConvNets: Examples

- Pedestrian detection

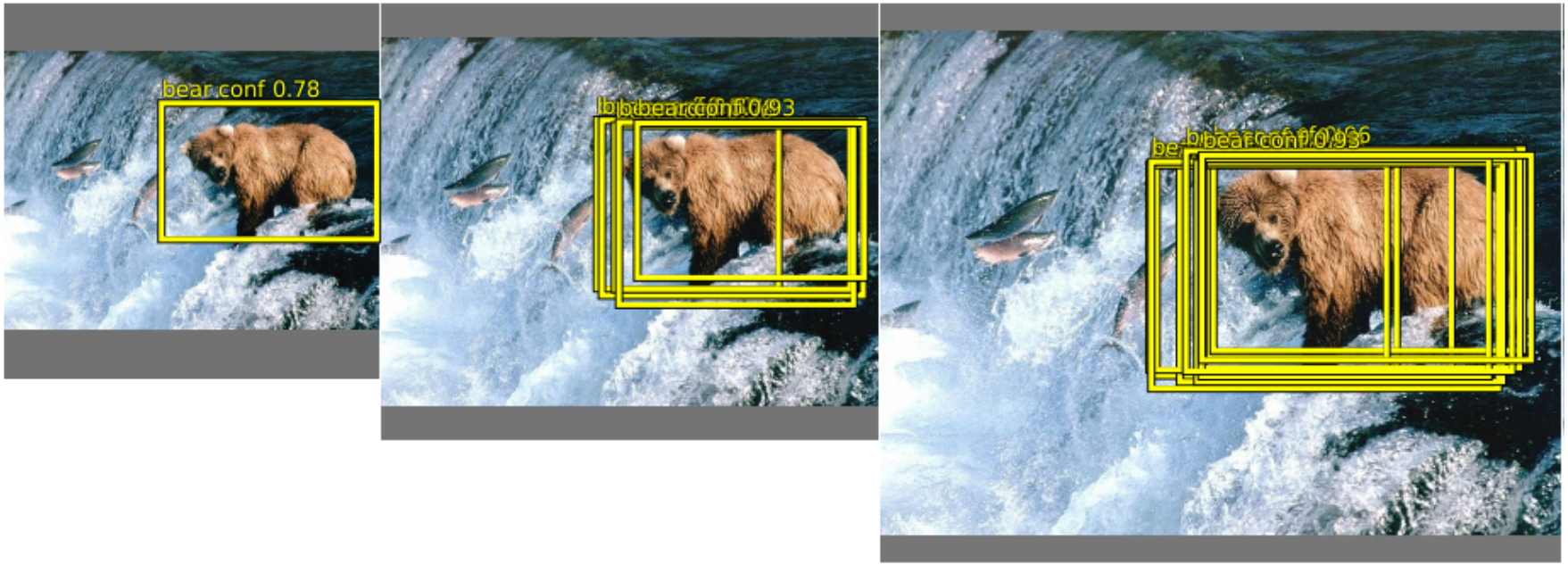




(Sermanet et al., Pedestrian detection with unsupervised multi-stage, CVPR 2013)

ConvNets: Examples

- Object Detection

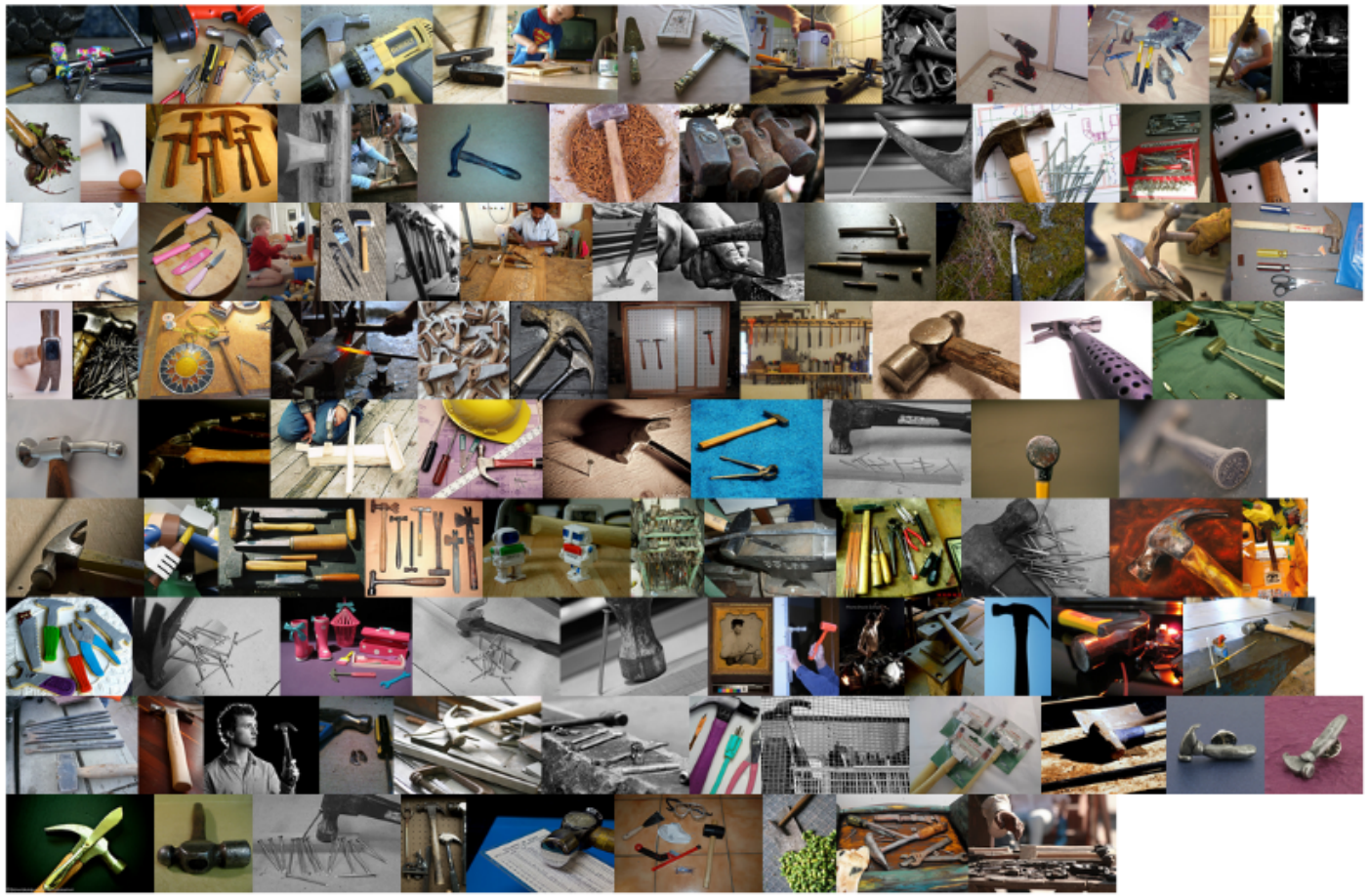


Sermanet et al., OverFeat: Integrated recognition, localization, 2013
Girshick et al., Rich feature hierarchies for accurate object detection, 2013
Szegedy et al., DNN for object detection, NIPS 2013

ImageNet Dataset

- 1.2 million images, 1000 classes

Examples of Hammer

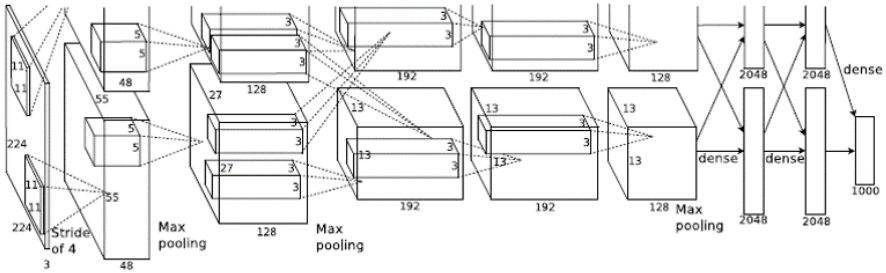


(Deng et al., Imagenet: a large scale hierarchical image database, CVPR 2009)

Important Breakthrough

- Deep Convolutional Nets for Vision (Supervised)

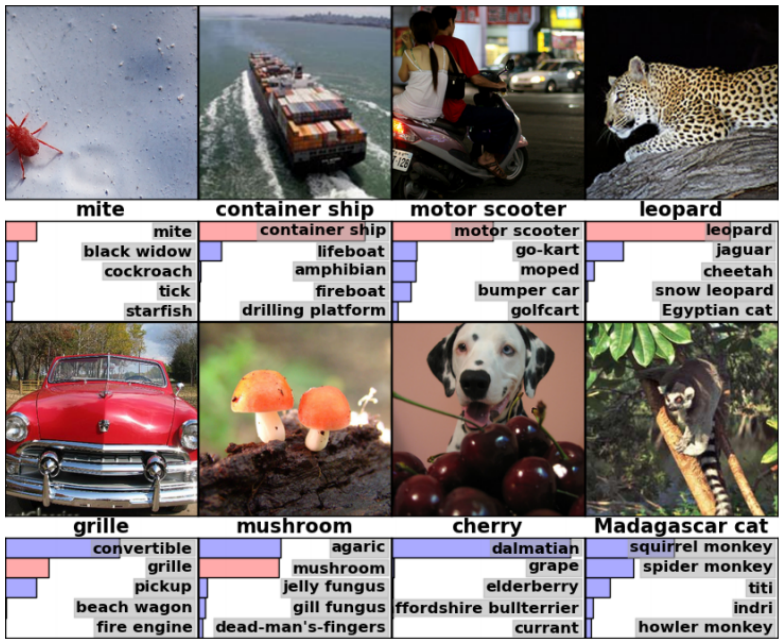
  Krizhevsky, A., Sutskever, I. and Hinton, G. E., ImageNet Classification with Deep Convolutional Neural Networks, NIPS, 2012.



IMAGENET

1.2 million training images

1000 classes



Architecture

- How can we select the right architecture:
  - Manual tuning of features is now replaced with the manual tuning of architechtures

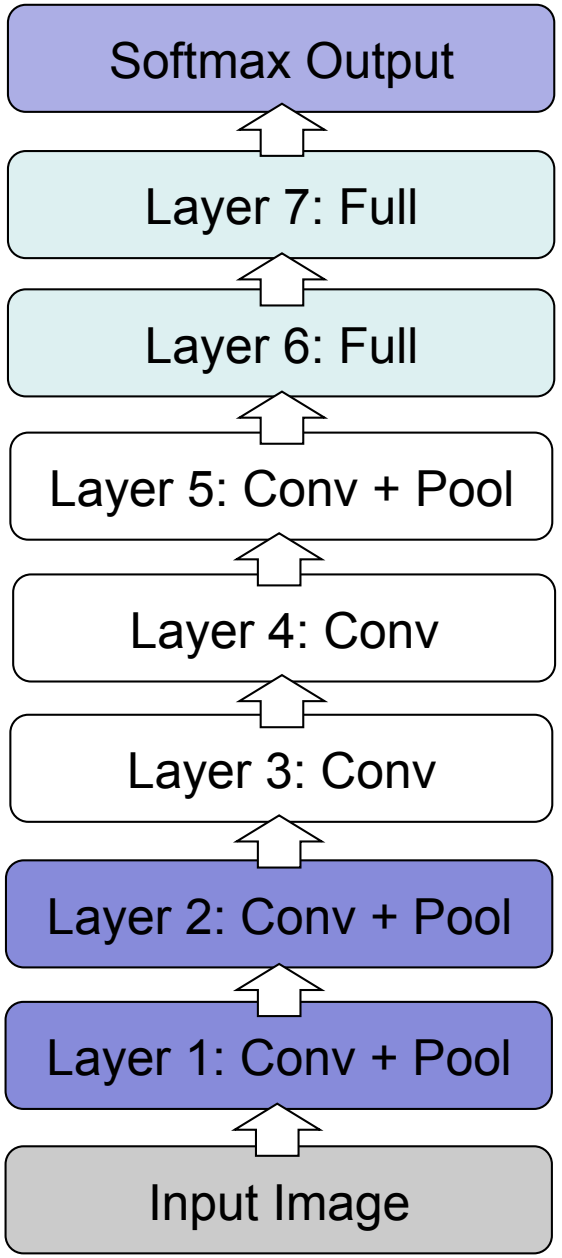
- Depth
- Width
- Parameter count

How to Choose Architecture

- Many hyper-parameters:
  - Number of layers, number of feature maps
- Cross Validation
- Grid Search (need lots of GPUs)
- Smarter Strategies
  - Random search
  - Bayesian Optimization

AlexNet

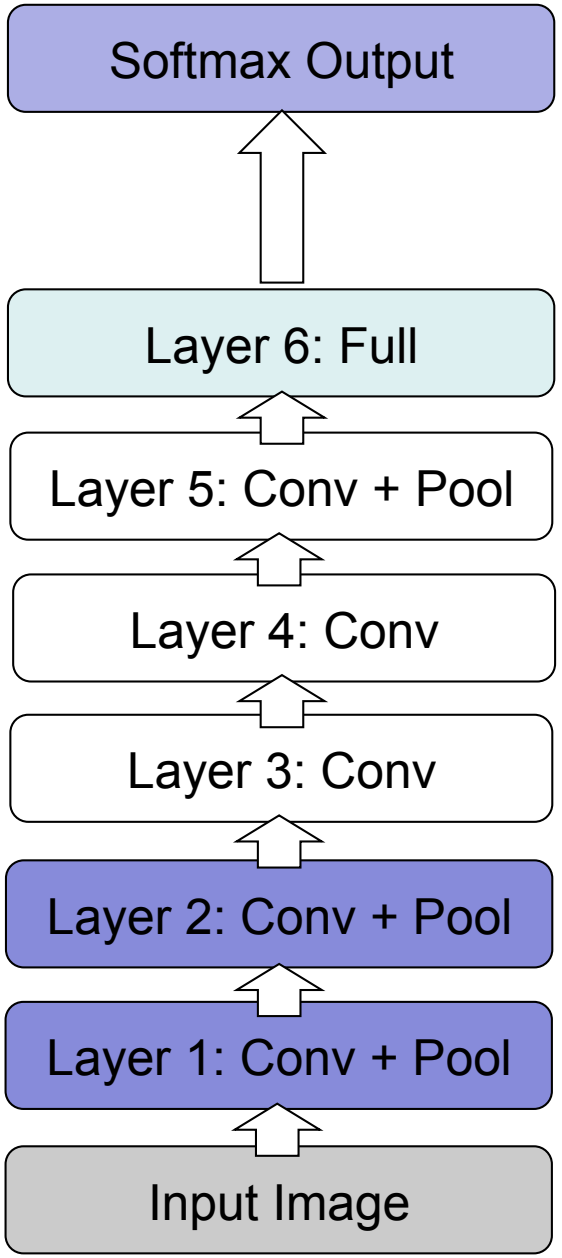
- 8 layers total
- Trained on Imagenet dataset [Deng et al. CVPR'09]
- 18.2% top-5 error



[From Rob Fergus' CIFAR 2016 tutorial]

AlexNet

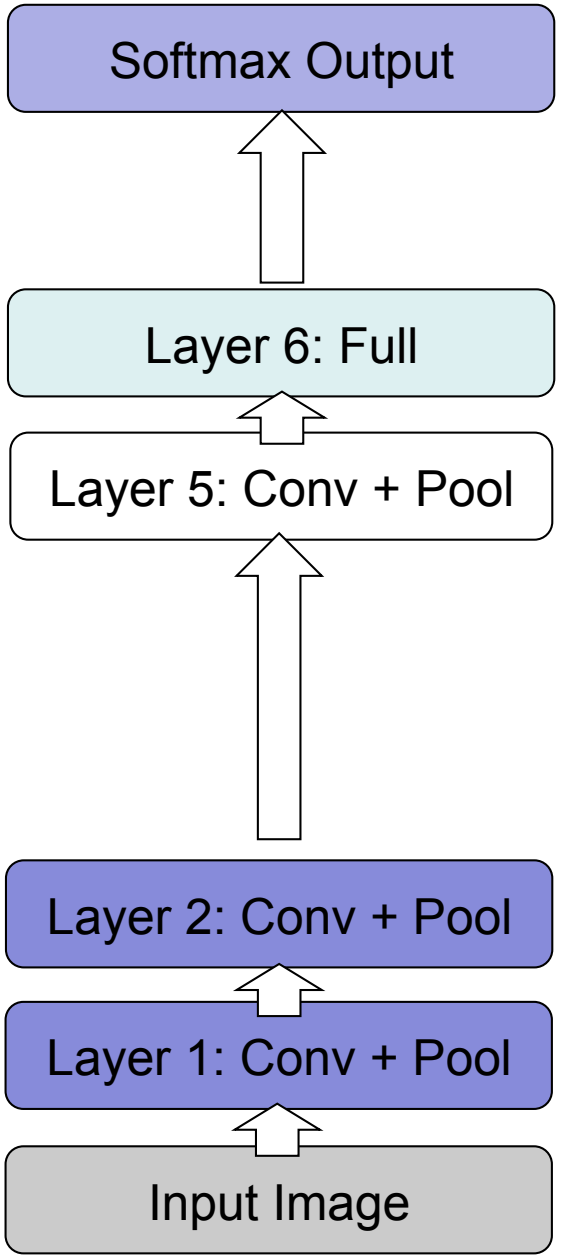
- Remove top fully connected layer 7
- Drop ~16 million parameters
- Only 1.1% drop in performance!



[From Rob Fergus' CIFAR 2016 tutorial]

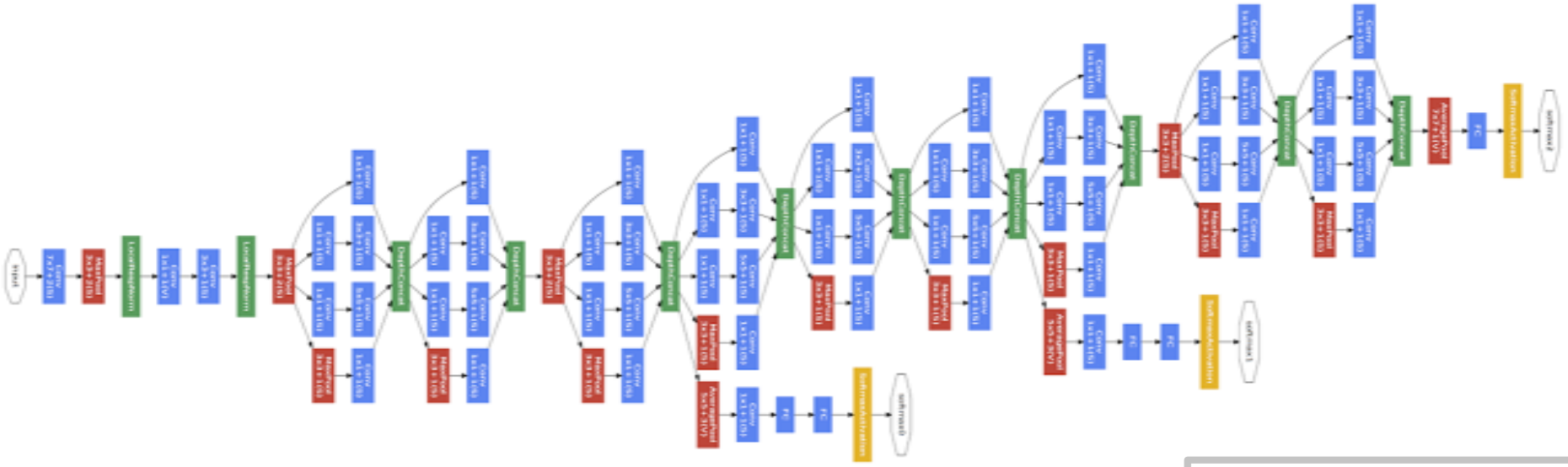
AlexNet

- Let us remove upper feature extractor layers and fully connected:
  - Layers 3,4, 6 and 7
- Drop ~50 million parameters
- **33.5 drop in performance!**
- Depth of the network is the key.



[From Rob Fergus' CIFAR 2016 tutorial]

GoogLeNet



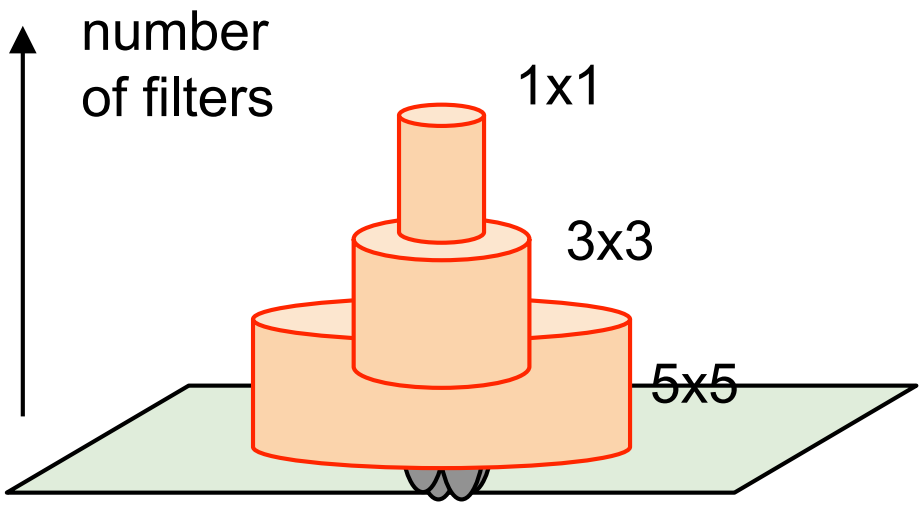
- 24 layer model that uses so-called inception module.

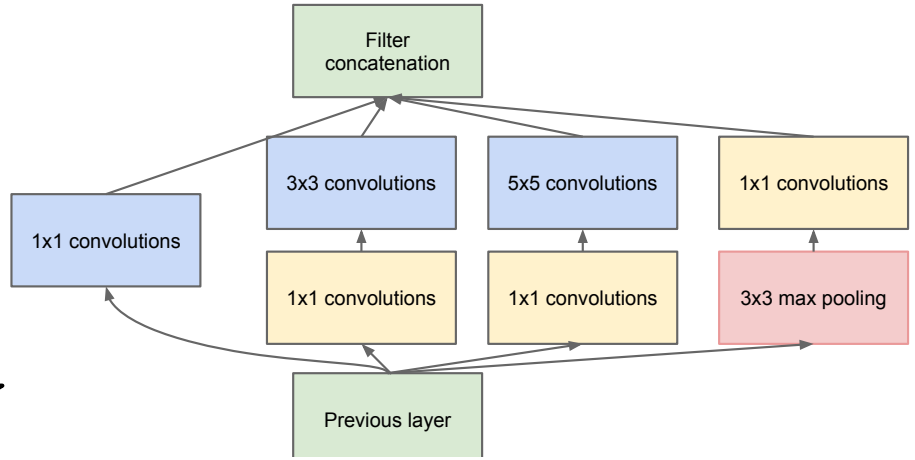
Convolution
Pooling
Softmax
Other

(Szegedy et al., Going Deep with Convolutions, 2014)

GoogLeNet

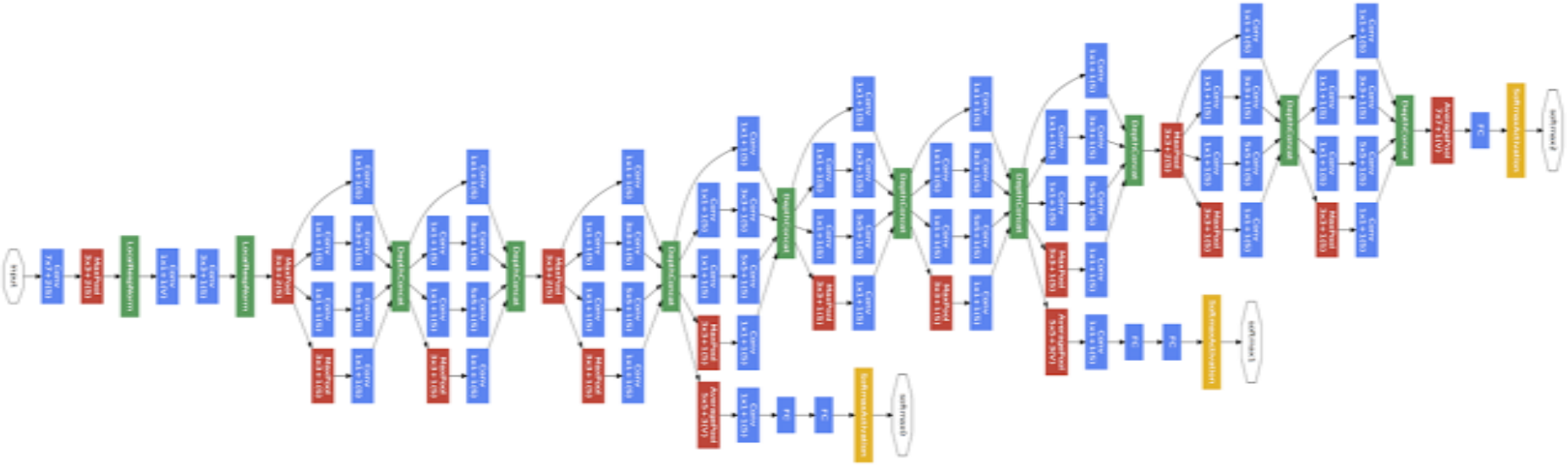
- GoogLeNet inception module:
  - Multiple filter scales at each layer
  - Dimensionality reduction to keep computational requirements down





(Szegedy et al., Going Deep with Convolutions, 2014)

GoogLeNet

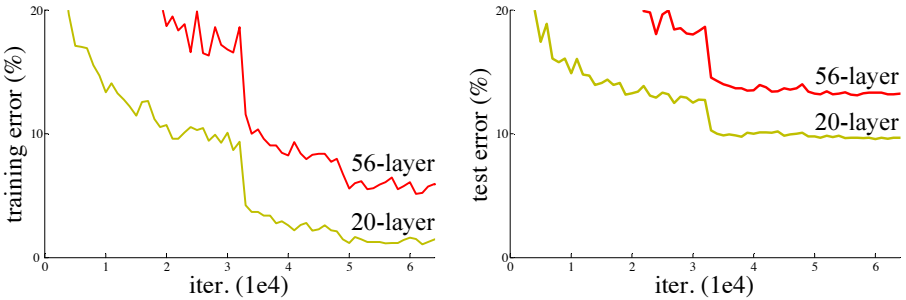


- Width of inception modules ranges from 256 filters (in early modules) to 1024 in top inception modules.
- Can remove fully connected layers on top completely
- Number of parameters is reduced to 5 million
- 6.7% top-5 validation error on Imagnet

(Szegedy et al., Going Deep with Convolutions, 2014)

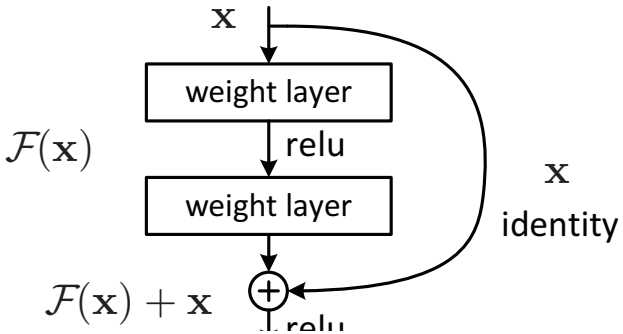
Residual Networks

Really, really deep convnets do not train well,
E.g. CIFAR10:



Key idea: introduce “pass through” into each layer

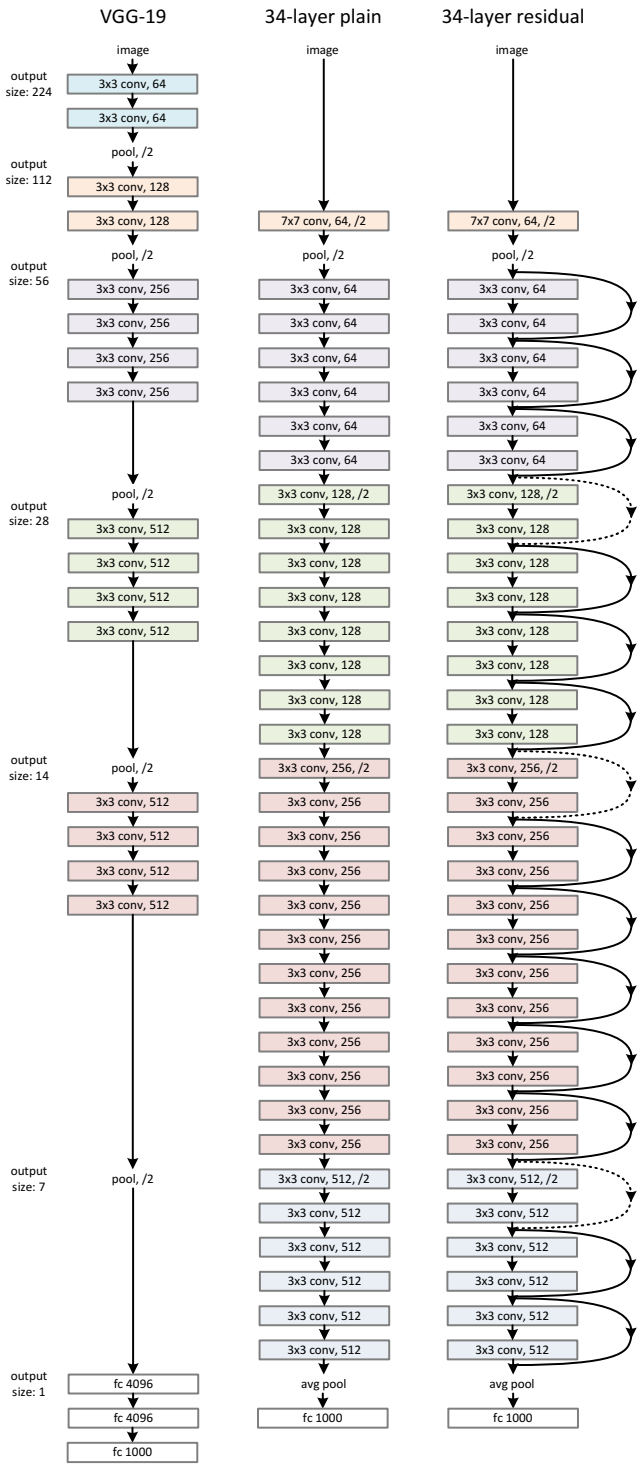
Thus only residual now needs to be learned



method	top-1 err.	top-5 err.
VGG [41] (ILSVRC’14)	-	8.43†
GoogLeNet [44] (ILSVRC’14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

With ensembling, 3.57% top-5 test error on ImageNet



(He, Zhang, Ren, Sun, CVPR 2016)

Choosing the Architecture

- Task dependent
- Cross-validation
- [Convolution → pooling]* + fully connected layer
- The more data: the more layers and the more kernels
  - Look at the number of parameters at each layer
  - Look at the number of flops at each layer
- Computational resources

[From Marc'Aurelio Ranzato, CVPR 2014 tutorial]

Optimization Tricks

• SGD with momentum, batch-normalization, and dropout usually works very well

• Pick learning rate by running on a subset of the data

- Start with large learning rate & divide by 2 until loss does not diverge
- Decay learning rate by a factor of ~100 or more by the end of training

• Use ReLU nonlinearity

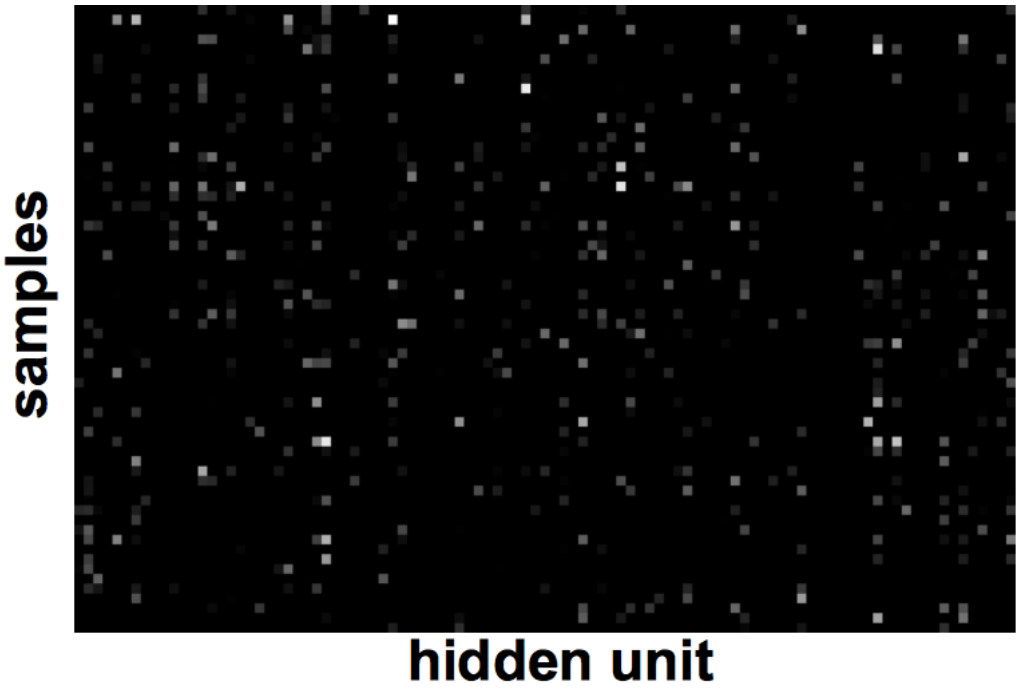
• Initialize parameters so that each feature across layers has similar variance. Avoid units in saturation.

[From Marc'Aurelio Ranzato, CVPR 2014 tutorial]

Visualization

- Check gradients numerically by finite differences

- Visualize features (features need to be uncorrelated) and have high variance



- Good training: hidden units are sparse across samples

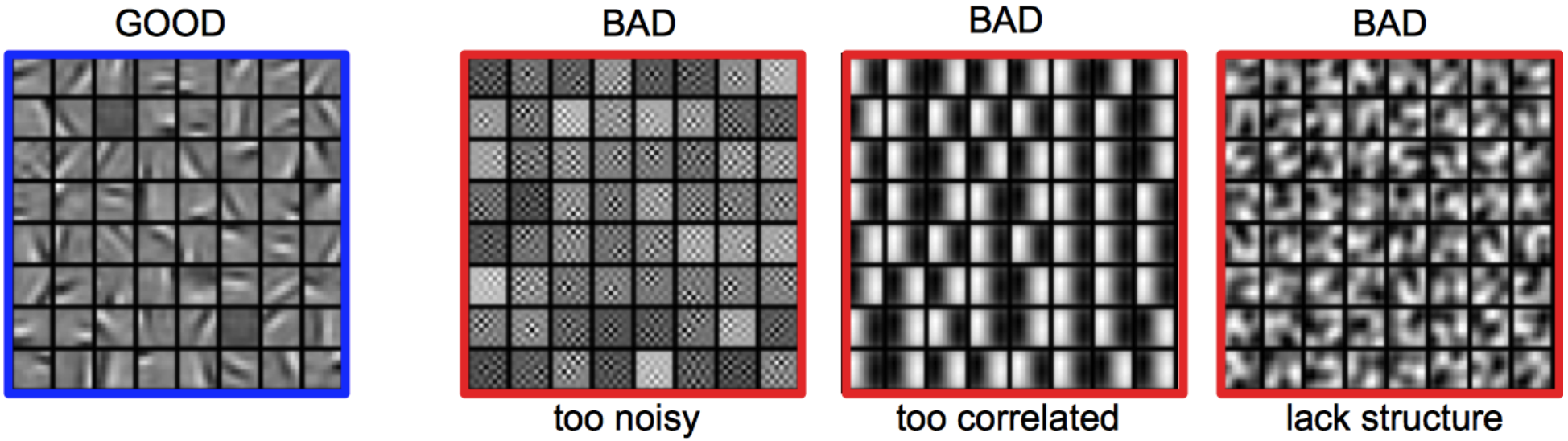
[From Marc'Aurelio Ranzato, CVPR 2014 tutorial]

Visualization

- Check gradients numerically by finite differences

- Visualize features (features need to be uncorrelated) and have high variance

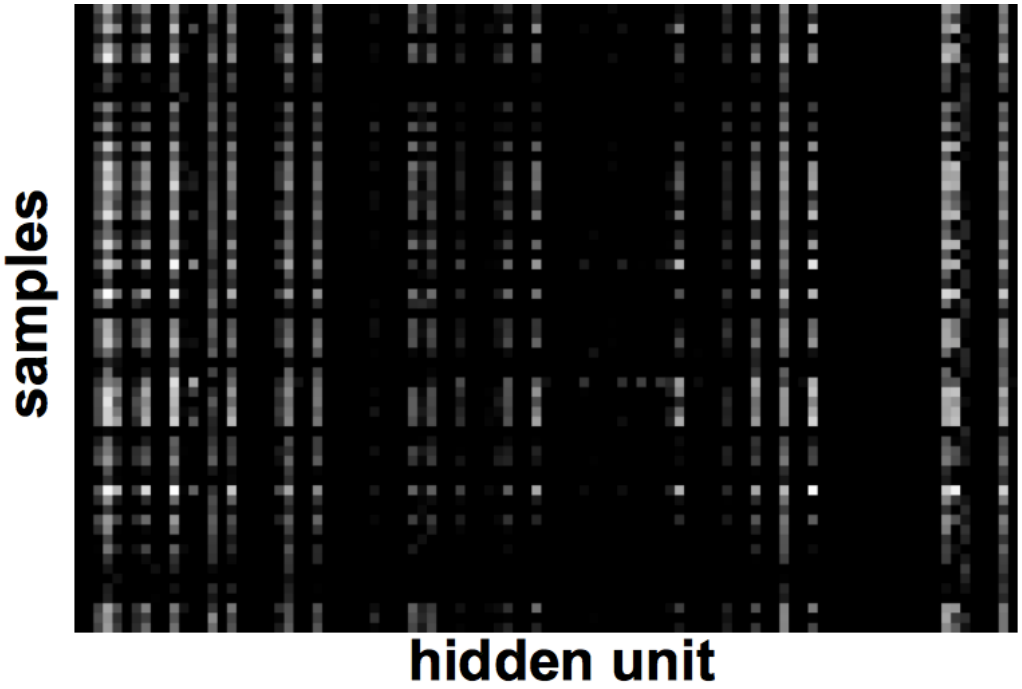
- Visualize parameters: learned features should exhibit structure and should be uncorrelated and are uncorrelated



Visualization

- Check gradients numerically by finite differences

- Visualize features (features need to be uncorrelated) and have high variance



- Bad training: many hidden units ignore the input and/or exhibit strong correlations