Deep learning

FFR135, Artificial Neural Networks

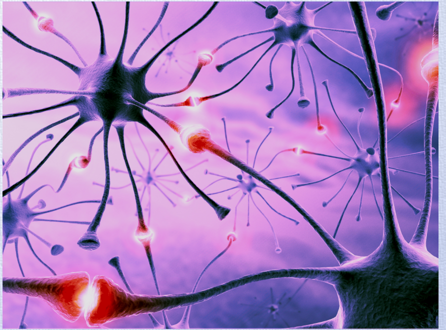
Olof Mogren

Chalmers University of Technology

October 2016

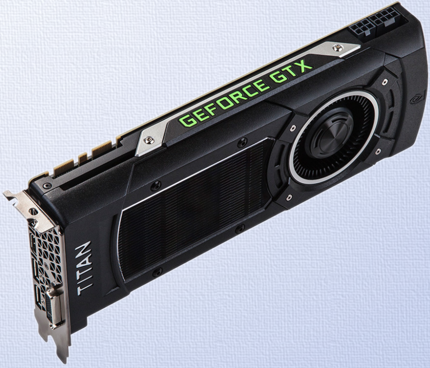
Deep learning

- Artificial neural networks
- Many layers of abstractions
- Outperforms traditional methods in:
  - Image classification
  - Natural language processing
    - Machine translation
    - Sentiment analysis
  - Speech recognition
  - Reinforcement learning



Semi-recent progress

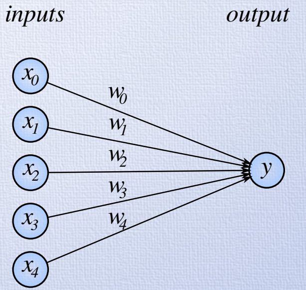
- 2006: Depth breakthrough: layerwise pretrained Restricted Boltzmann Machines
- GPUs
- Practical use
  Real applications from Google, Facebook, Tesla, Microsoft, Apple, and others!



A fast learning algorithm for deep belief nets; Hinton, Osindero, Tehi; Neural Computation; 2006

Perceptron

- 1943, McCulloch & Pitts (neuron model)
- 1958, Rosenblatt (perceptron)
- Linear (binary) classification of inputs
- Can not learn any non-linear function (e.g. XOR)



Modelling XOR

	1	1	0
x_0			
	0	0	1
		0	1
		x_1	

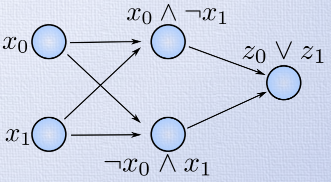
Modelling XOR

x_0 \ x_1	0	1
1	1	0
0	0	1

$x_0 \wedge \neg x_1$

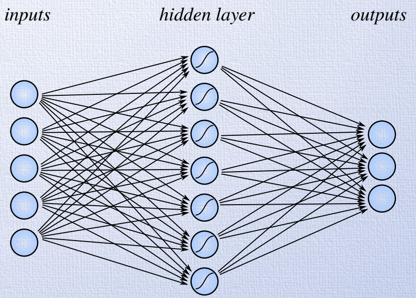
x_0 \ x_1	0	1
1	1	
0	0	1

$\neg x_0 \wedge x_1$



Multi-layer perceptron

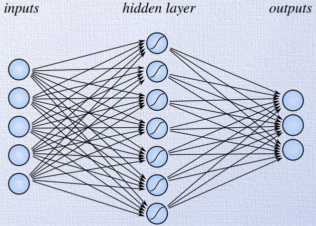
- Combining layers lets us represent non-linear functions
- Each layer:
  - Linear transformation: $\mathbf{a} = W\mathbf{x} + \mathbf{b}$
  - Non-linear (element-wise) activation: $\mathbf{h} = g(\mathbf{a})$



Modelling functions

- Universal function approximation
- Stacking layers: function composition
- Apply error/loss function to output
- Continuously differentiable; chain rule
- Propagating errors (backpropagation)
- (Mini-batch) Stochastic gradient descent (SGD)

details



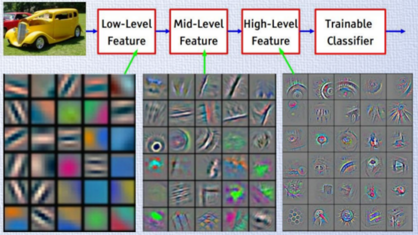
Motivation of depth

- More compact representation (exponentially)
- There are boolean functions that require
  - **Polynomial** number of units (**deep** architecture)
  - **Exponential** number of units (**shallow** architecture)
  - E.g., parity function (for *n* input bits):
    - efficiently represented with depth **O(log n)**
    - but **O(2^n)** gates if represented by a depth two circuit *(Yao, 1985)*

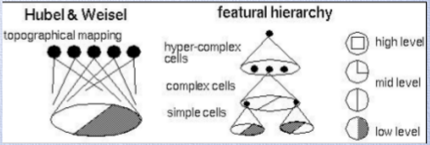
Exploring Strategies for Training Deep Neural Networks; Larochelle, Bengio, Louradour, Lamblin; JMLR 2009

Learning levels of representation

- Each layer:
  non-linear transformation of inputs:
  $\mathbf{h} = sigmoid(W\mathbf{x} + \mathbf{b})$
- Learning representations; abstractions
- No feature engineering!

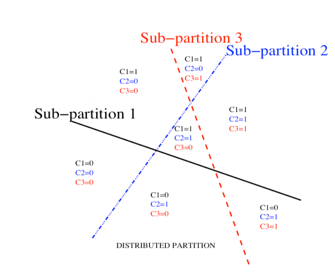


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]



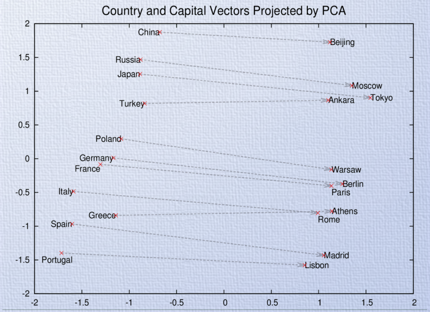
Distributed representations

- E.g.: big, yellow, Volkswagen
- Non-distributed representations:
  n binary parameters $\rightarrow$ n values
- E.g.: Clustering, n-grams, decision trees, etc.
- NNs learn distributed representations
- Distributed representations:
  n binary parameters $\rightarrow$ 2^n possible values



Example: Word embeddings

- Distributed representations for words
- word2vec, glove, etc.

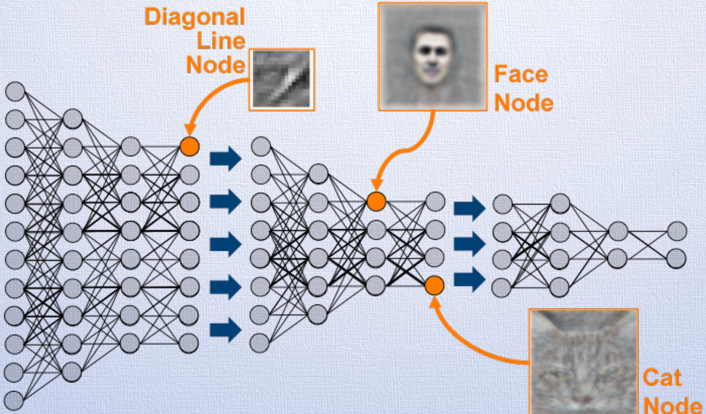


Deep learning in JavaScript

cs231n.stanford.edu

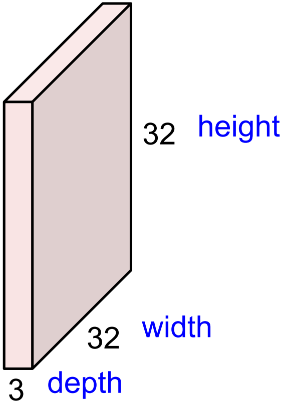
playground.tensorflow.org

Levels of abstractions



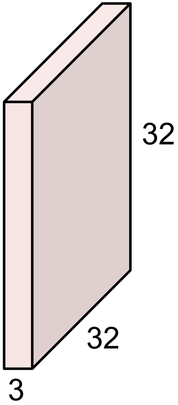
Convolution Layer

32x32x3 image

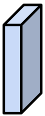


Convolution Layer

32x32x3 image



5x5x3 filter



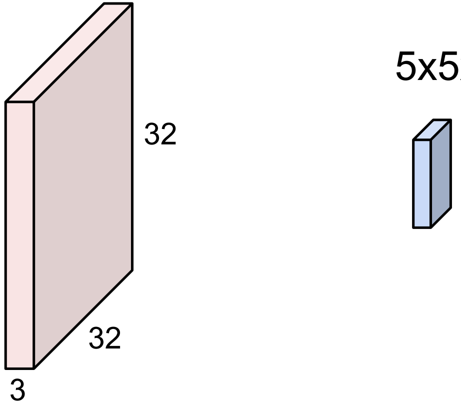
Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Convolution Layer

Filters always extend the full depth of the input volume

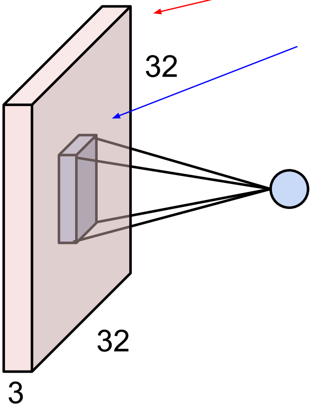
32x32x3 image

5x5x3 filter



Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

Convolution Layer



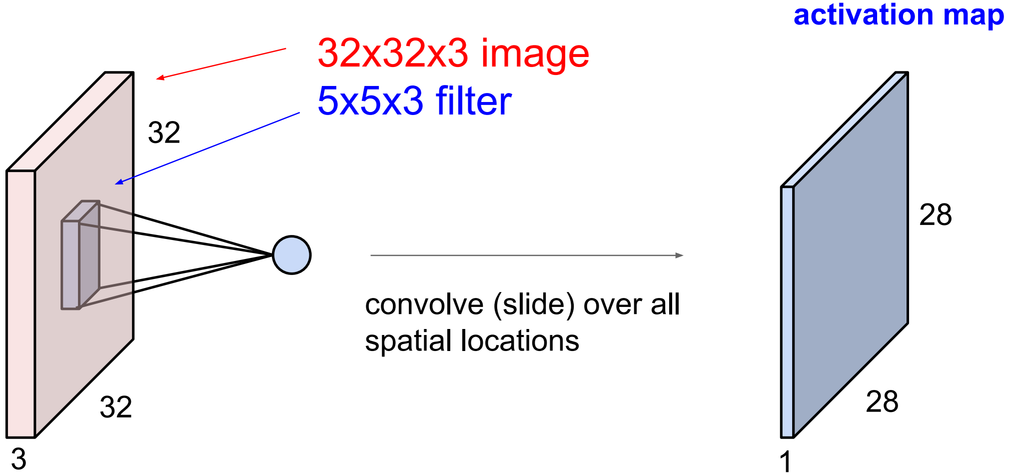
32x32x3 image

5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

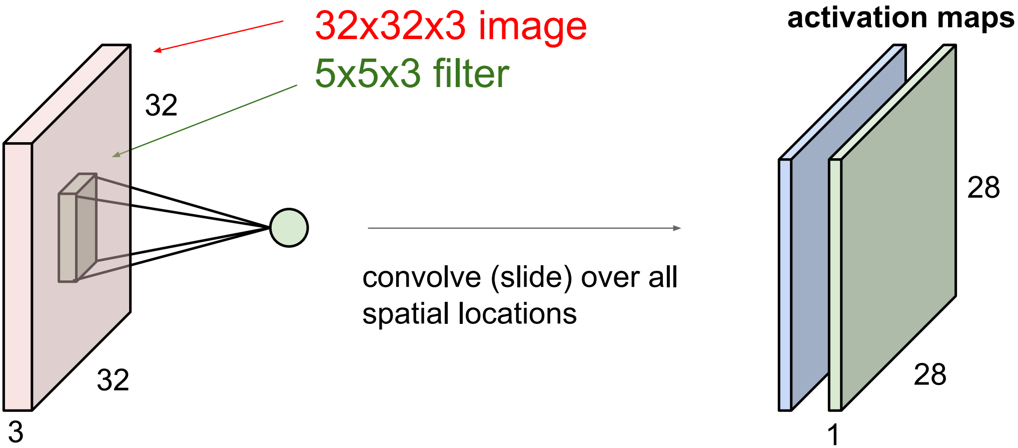
$$w^T x + b$$

Convolution Layer

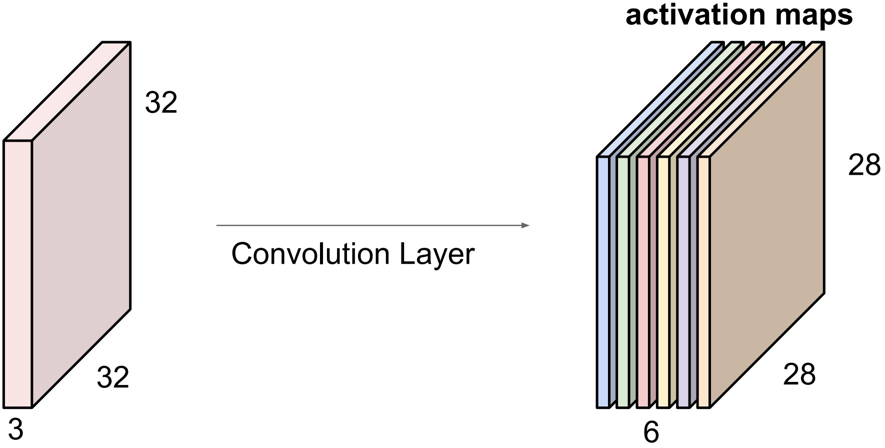


Convolution Layer

consider a second, green filter

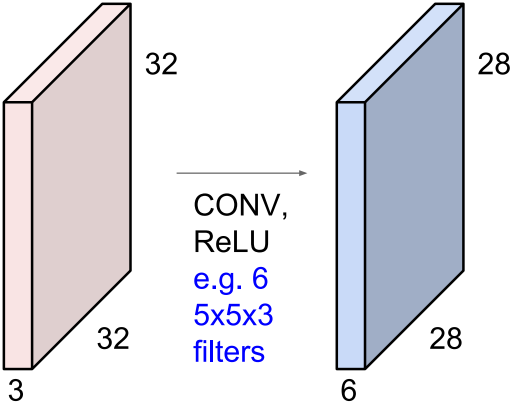


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

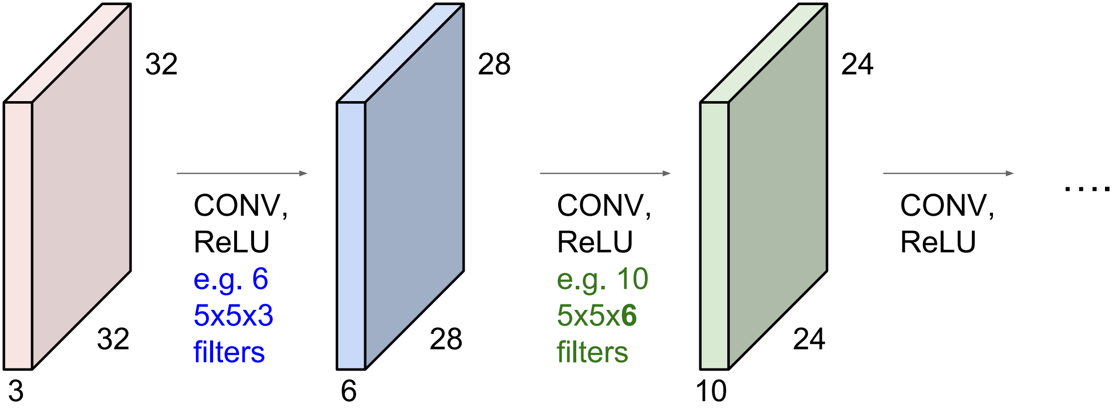


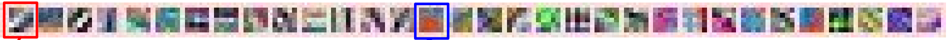
We stack these up to get a "new image" of size 28x28x6!

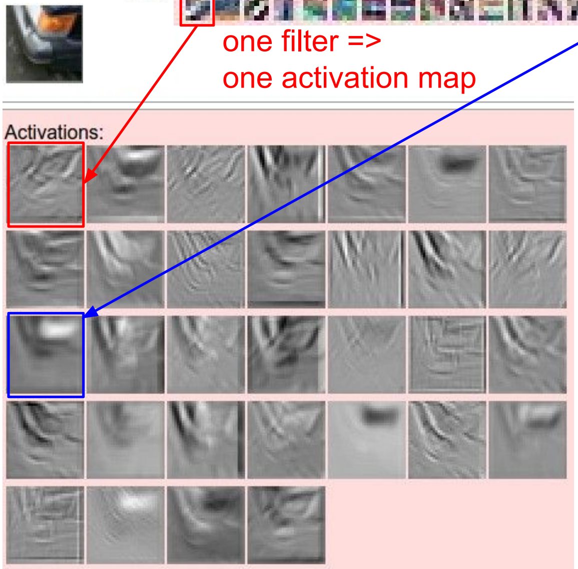
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions





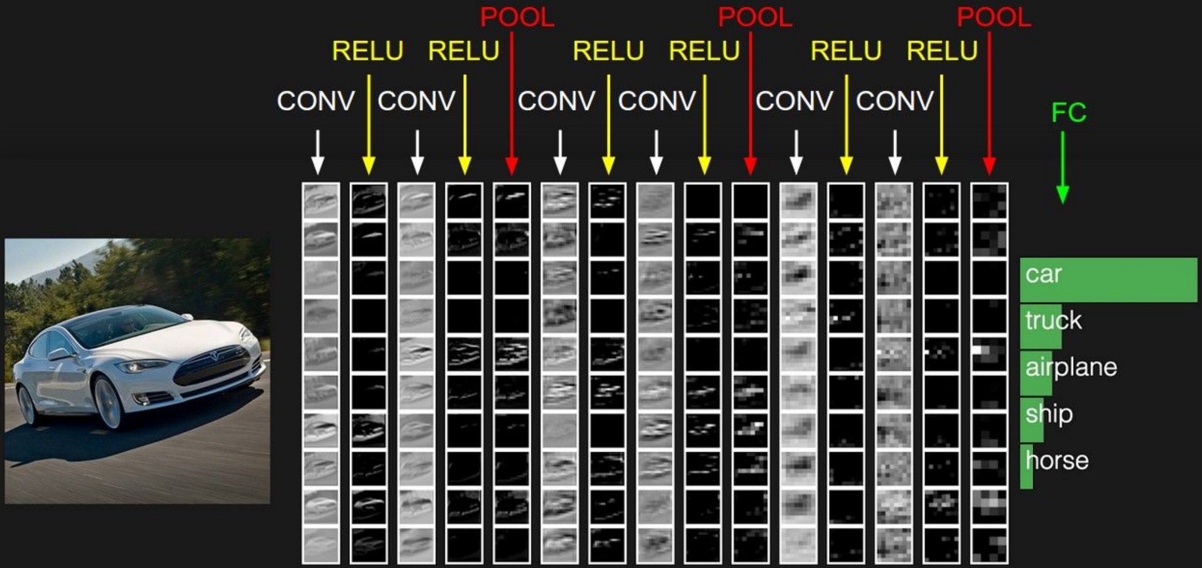


example 5x5 filters
(32 total)

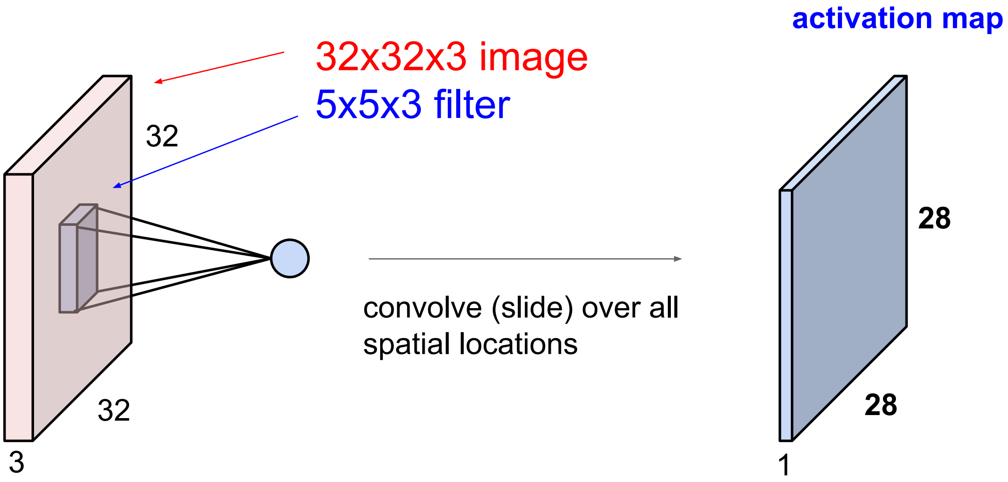
We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

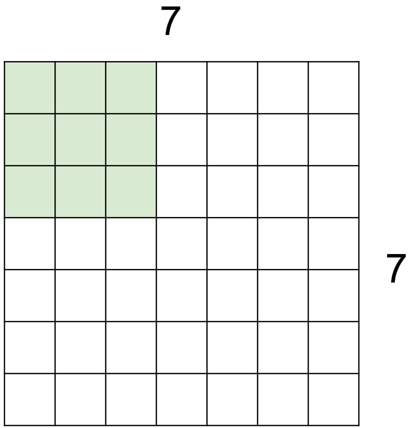
elementwise multiplication and sum of a filter and the signal (image) preview:


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

A closer look at spatial dimensions:

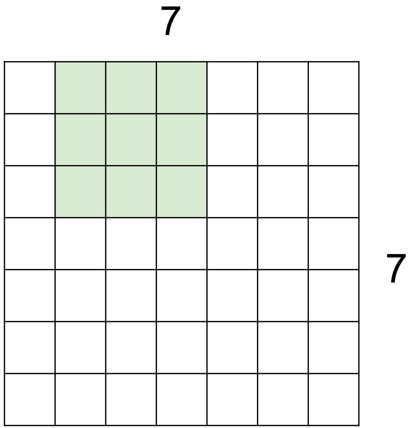


A closer look at spatial dimensions:



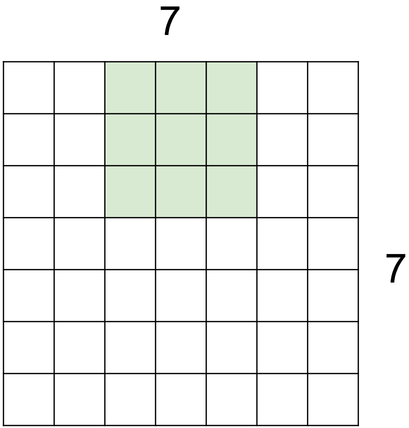
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



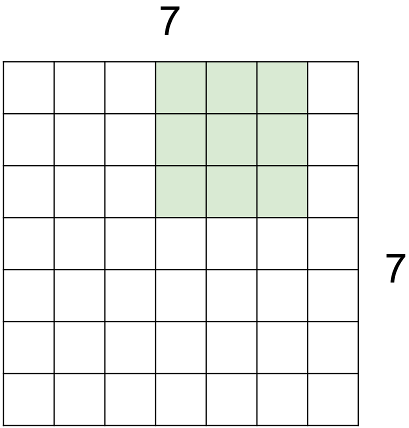
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



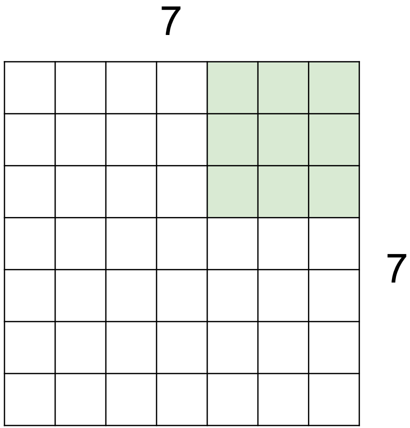
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

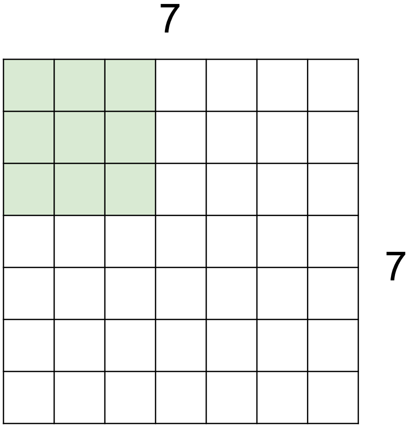
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

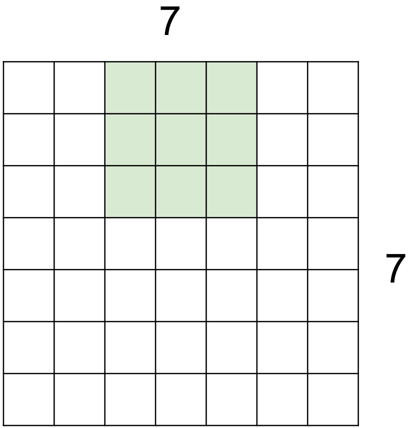
=> 5x5 output

A closer look at spatial dimensions:



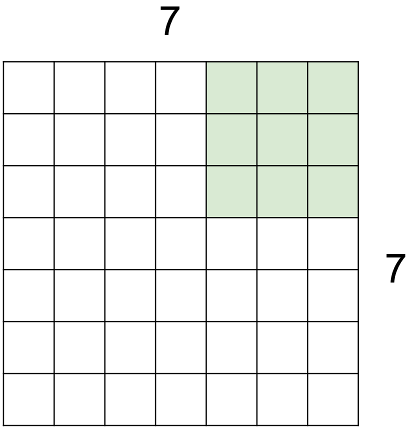
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



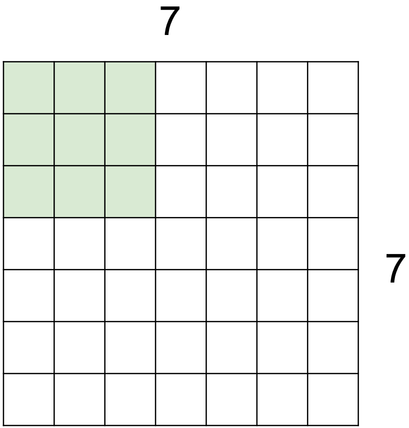
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



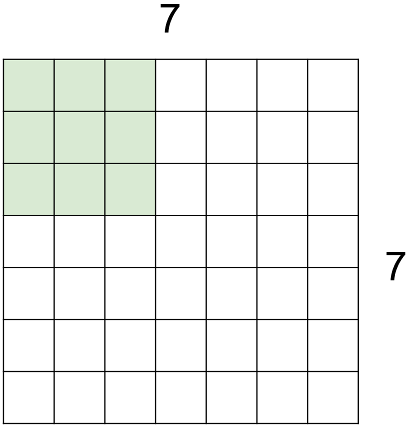
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

A closer look at spatial dimensions:



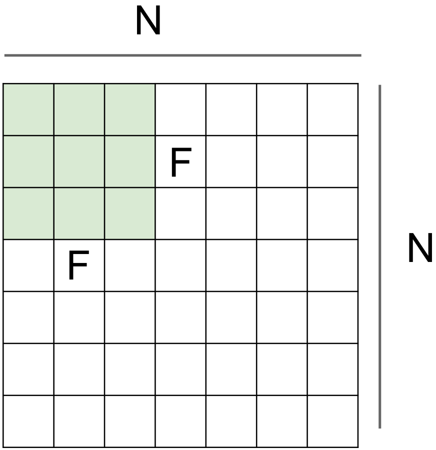
7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.



Output size:

(N - F) / stride + 1

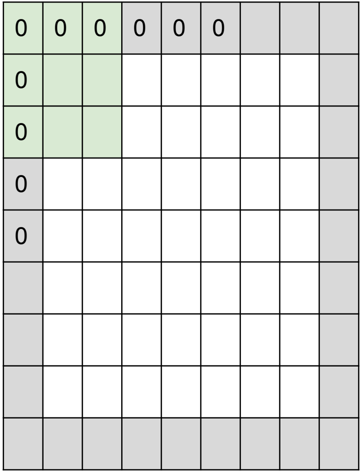
e.g. N = 7, F = 3:

stride 1 => (7 - 3)/1 + 1 = 5

stride 2 => (7 - 3)/2 + 1 = 3

stride 3 => (7 - 3)/3 + 1 = 2.33 :\

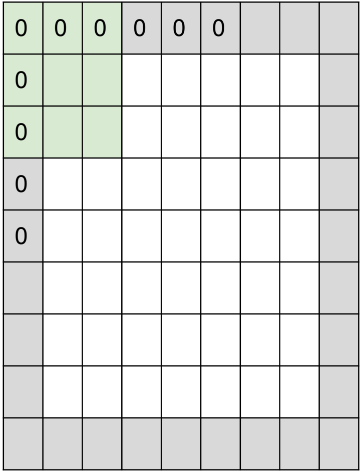
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

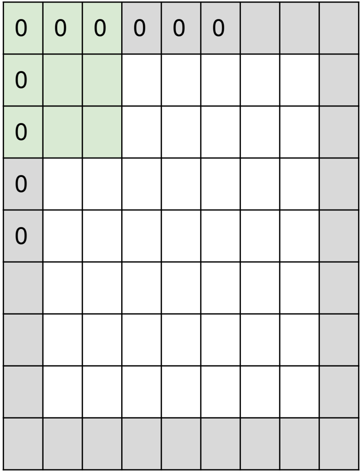
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

In practice: Common to zero pad the border

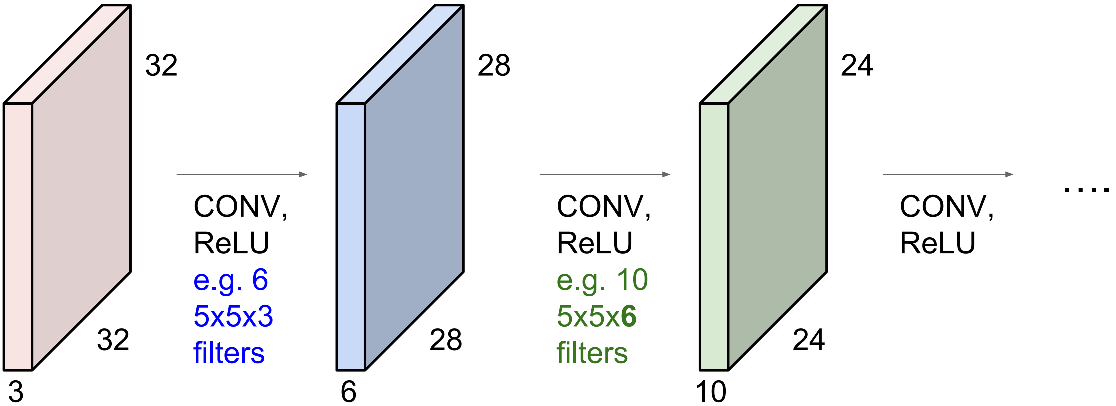


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Remember back to…

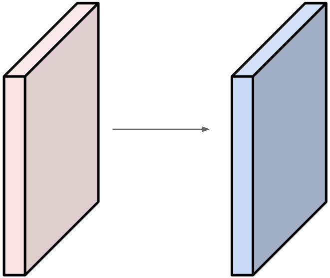
E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn’t work well.



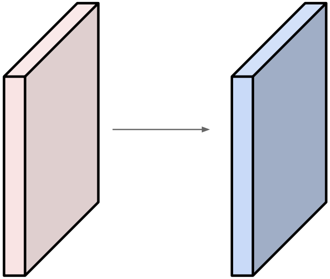
Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?



Examples time:



Input volume: **32x32x3**

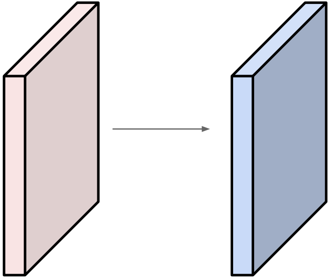
10 5x5 filters with stride 1, pad 2

Output volume size:

(32+2*2-5)/1+1 = 32 spatially, so

32x32x10

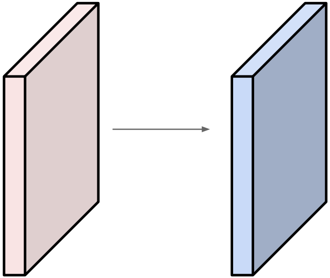
Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

Examples time:



Input volume: **32x32x3**

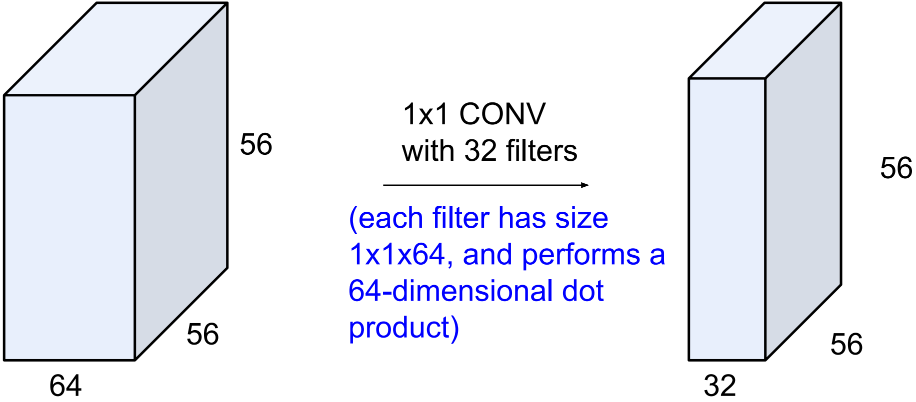
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

each filter has 5*5*3 + 1 = 76 params (+1 for bias)

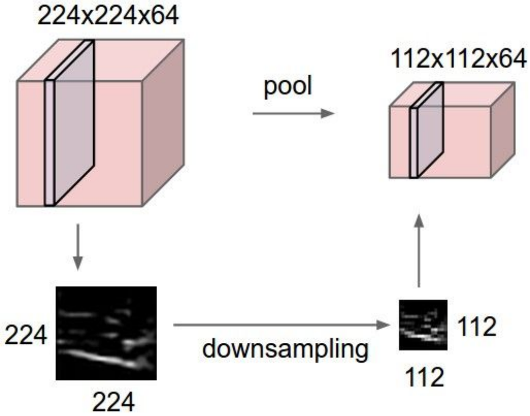
=> 76*10 = **760**

(btw, 1x1 convolution layers make perfect sense)

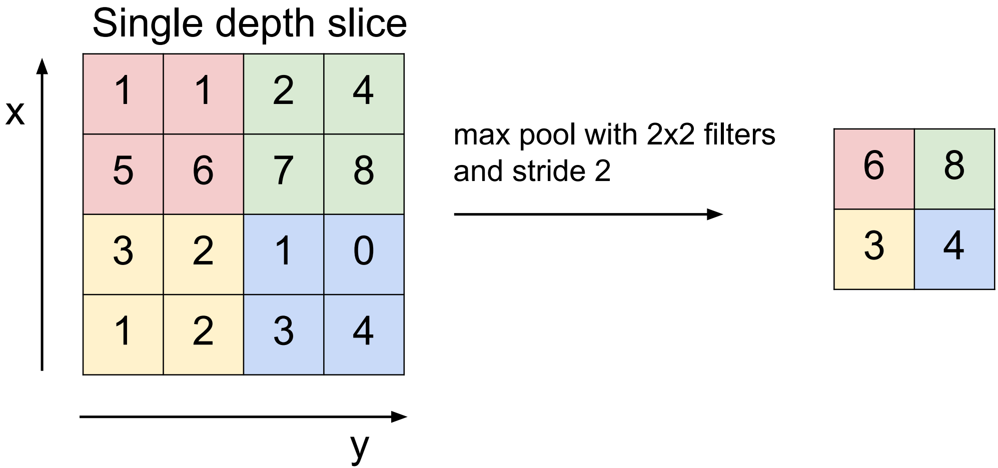


Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:

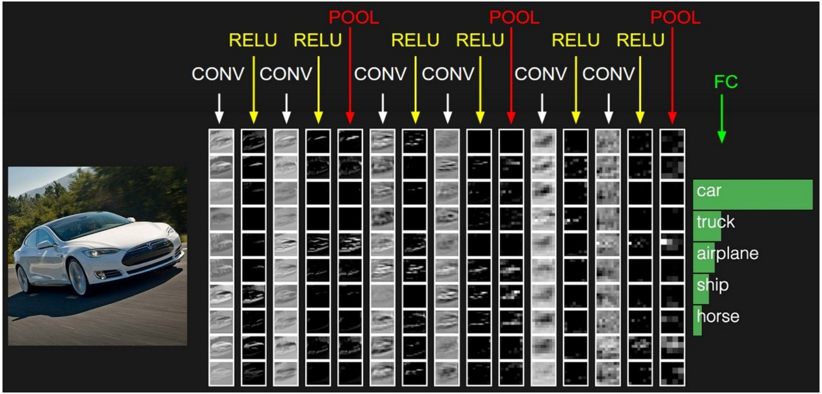


MAX POOLING


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Dropout

- During training:
- For each postactivation h_i, with probability p let $h_i = 0$
- Redundancy
- Equivalent to learning an ensamble of networks

Improving neural networks by preventing co-adaptation of feature detectors; Hinton, Srivastava, Krizhevsky, Sutskever, Salakhutdinov; (2012); arXiv:1207.0580

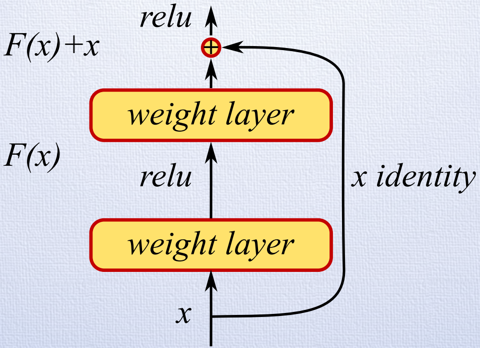
more on regularization

Batch normalization

- For each batch
- Normalize inputs to every layer to zero mean, unit variance.
- Helps with covariance shift

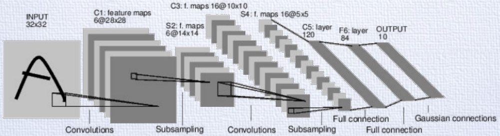
Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift; Ioffe, Szegedy; arXiv:1502.03167

Residual connections



Deep Residual Learning for Image Recognition; He, Zhang, Ren, Sun; arXiv:1512.03385

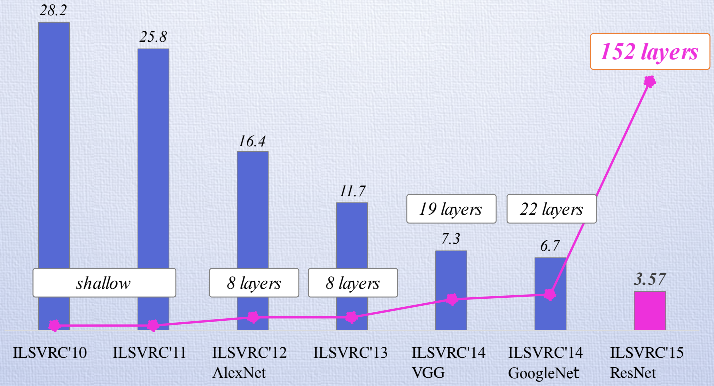
DEEPER AND DEEPER



[LeNet-5, LeCun 1980]

- 1998: LeNet-5; 3 layers
- 2012: AlexNet; 8 layers
- 2014: GoogLeNet; 22 layers (illustration)
- 2015: Residual Nets; 152 layers
- "Surpassed" human performance in 2015

Depth development

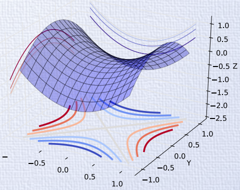


ImageNet Classification top-5 error (%)

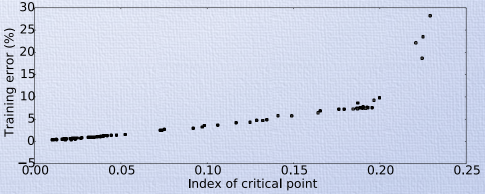
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Non-convex optimization

- Loss function non-convex
- Low-D: **local minima** dominate
- High-D: **saddle points** dominate
- Local minima are close to global minimum
- Convexity not needed



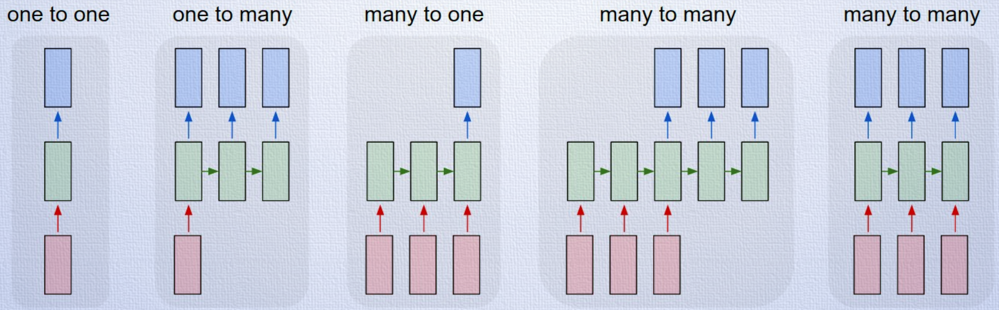
Yoshua Bengio



The loss surfaces of multilayer networks; Choromanska, et.al.; AISTATS 2015

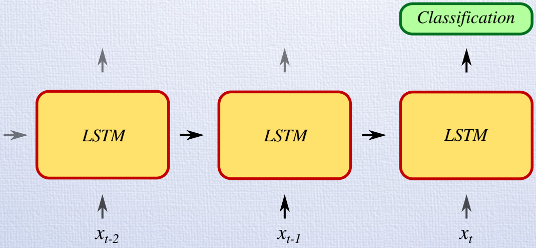
Identifying and attacking the saddle point problem in high-dimensional non-convex optimization; Dauphin, et.al.; NIPS 2014

SEQUENCE MODELLING



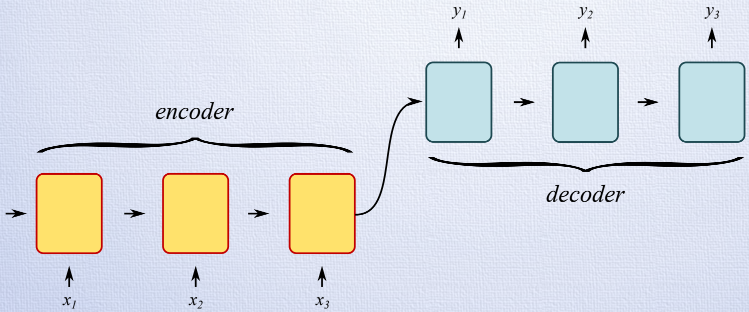
Andrej Karpathy
details

Sentiment analysis



- Binary sequence classification

Neural machine translation, NMT



Sequence to sequence learning with neural networks; Sutskever, Vinyals, Le; NIPS 2014

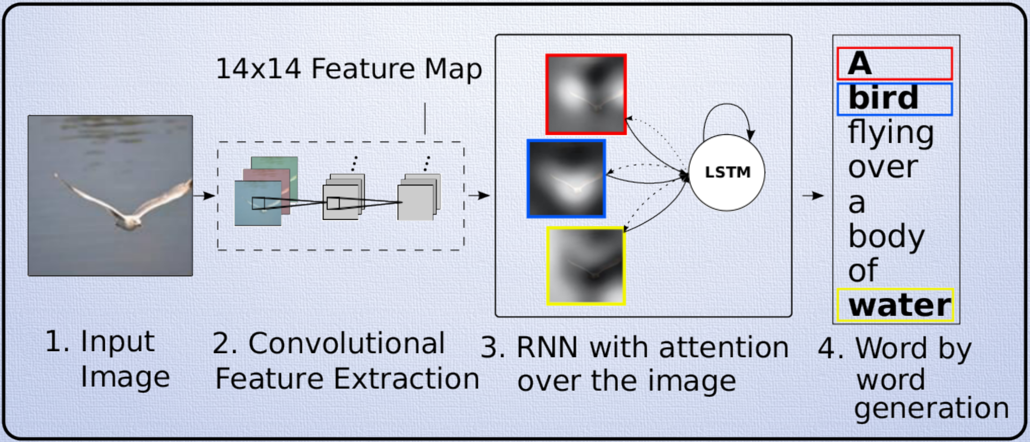
Neural machine translation by jointly learning to align and translate; Bahdanau, Cho, Bengio; ICLR 2015

Recent advances in NMT

- Subwords (BPE) (Sennrich et.al., ACL 2016)
- 8 layers deep LSTM model.
- Quantized weights $\in \{-1, 0, +1\}$
- Downpour SGD: parallell training
- 8GPUs, one host.
- Human evaluation:
  results comparable to human translators!

Google's neural machine translation system: Bridging the gap between human and machine translation; Yonghui Wu, et.al.; arXiv 1609.08144

Caption generation



more

http://**mogren.one**/