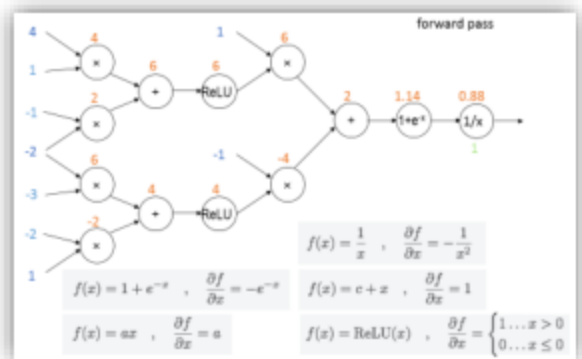
EcoVision

ETH zürich

Deep learning for geospatial data analysis

Jan D. Wegner and Nico Lang

(with slides borrowed from Konrad Schindler and Andrej Karpathy)



jan.wegner@geod.baug.ethz.ch

This talk…

1. **Deep learning theory**

…some dry theory

….ooohh noo

2. **Applications**

But later:…cool application

jan.wegner@geod.baug.ethz.ch

1. Deep Learning

- ✓ Why is this interesting?

  supports decision-making based on (a lot of) data and scales

- ✓ Why does this finally work? (it has been around since the 1980's…)

  massive amounts of training data available via the web
  significant hard-ware improvements, especially GPUs and RAM
  minor algorithmic improvements

- ✓ How can it help me?

  if you cannot model all causalities for solving a task but have massive amounts of data, machine learning does the job for you

What's new about it?

- Traditional machine learning:

Most of the "heavy lifting" in here. Final performance only as good as the feature set.

- Deep learning:

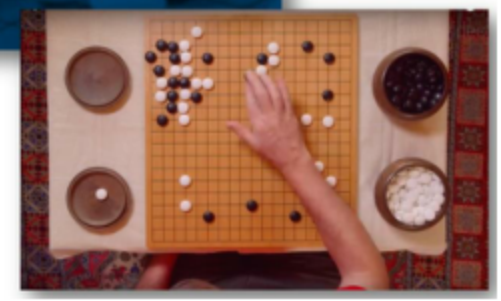
Features and model learned together, mutually reinforcing

slide credit: Michele Catasta

Deep learning can do fun stuff…

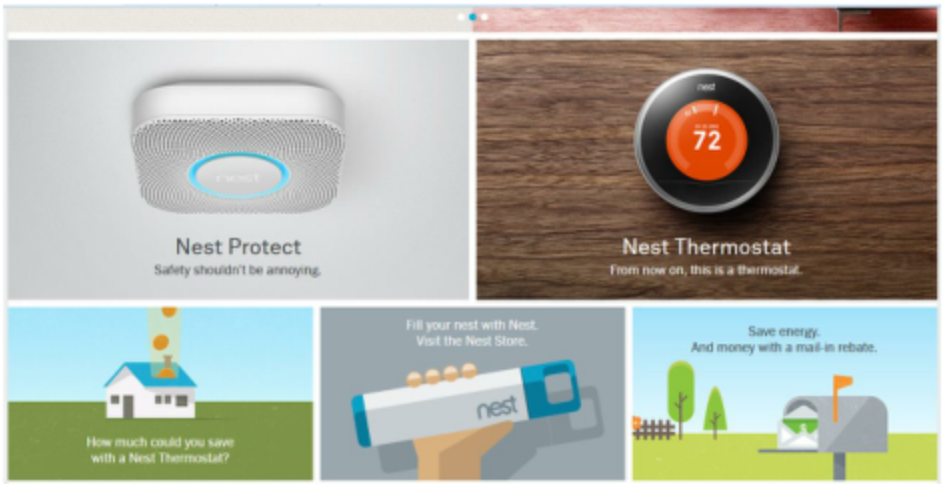
ETH Zurich Hönggerberg campus in the style of Van Gogh painted by a deep network (Google Deep Dream)

Machine beats human at game of Go – a milestone of artificial intelligence!



jan.wegner@geod.baug.ethz.ch

...but a lot of useful things, too!



Cont'd.

recommender systems

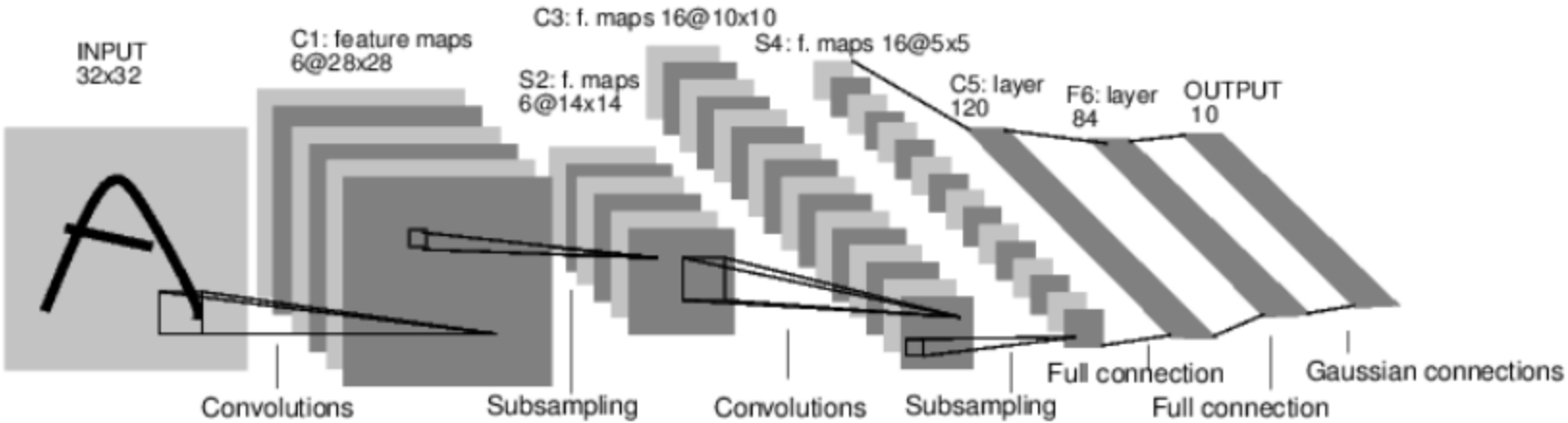
sentiment analysis

mapping

text mining

CNN components: overview

LeNet-5, LeCun 1980



There are many deep learning variants…

…we will focus on ***Convolutional Neural Networks (CNN)*** in the theoretical part

…but some basic neural networks in the practical part

1.1 CNN components

In the following: example case images, but DL/CNN can be applied to any other data like 3D point clouds, too !!!

transform original image layer by layer from original pixel values to final class scores

spatial extent of classified unit (image, pixel, bounding box) depends on task and needs specific network architecture.

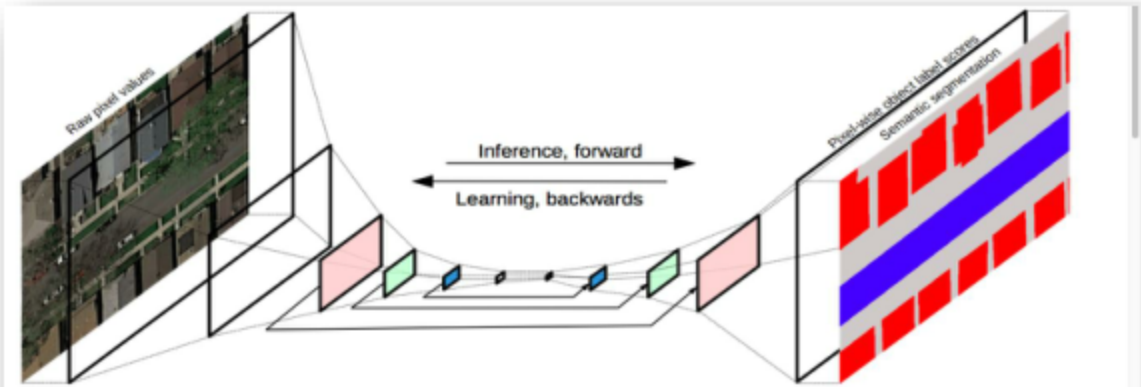
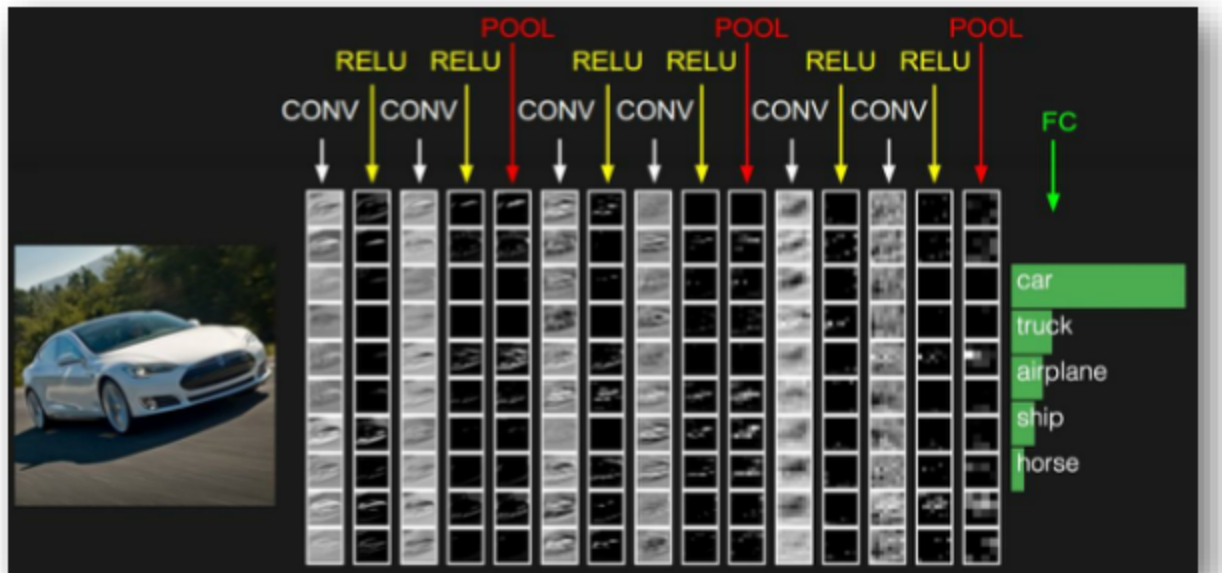


Figure 3.6: Conceptional illustration of the data flow through a fully convolutional network (FCN), which is used for the semantic segmentation of aerial images. Three "skips" are highlighted by pale red, pale green and pale blue, respectively. Figure is based on Figure 1 in Long et al. [20].

For images: Each Layer accepts an input 3D volume and transforms it to an output 3D volume *through a differentiable function.*

Components I

- INPUT: image (e.g., width 32, height 32, three color channels R,G,B)

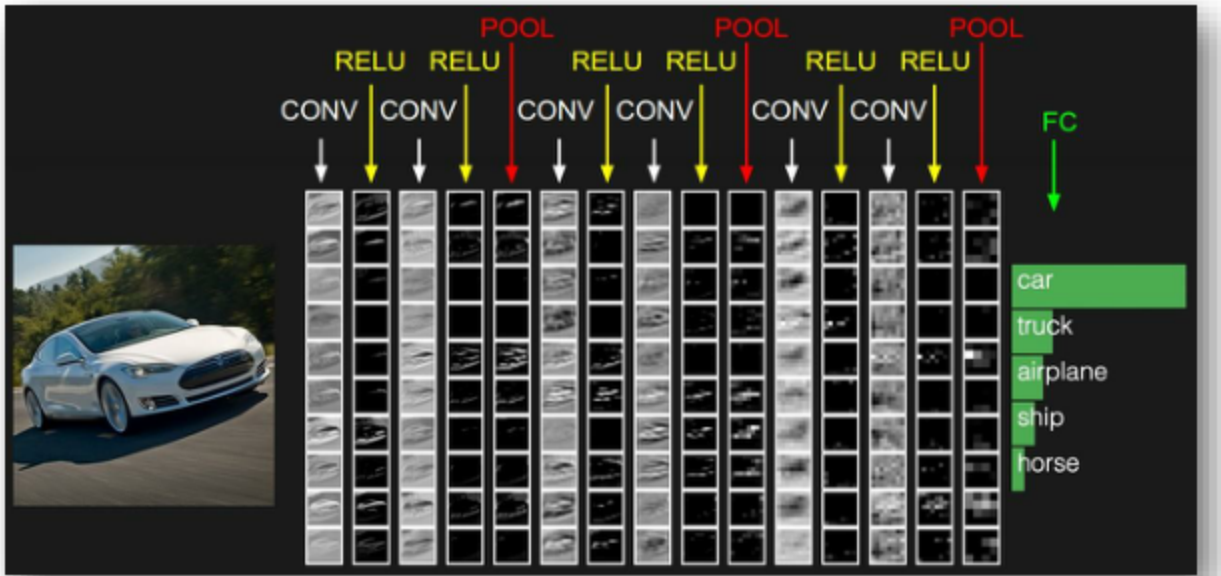


- CONV (convolutional layer): computes dot product between weights (filter kernel) and region in the input volume (may result in volumes like [32x32x12])

- RELU (rectified linear unit): layer will apply an elementwise activation function, such as max(0,x) ➔ thresholding at zero and leaving all other values unchanged (size of volume remains unchanged [32x32x12]). ➔ *you will play with activation functions during the practical part*

Components II

- POOL (pooling layer): downsampling operation along spatial dimensions (width, height), resulting in volume such as [16x16x12].



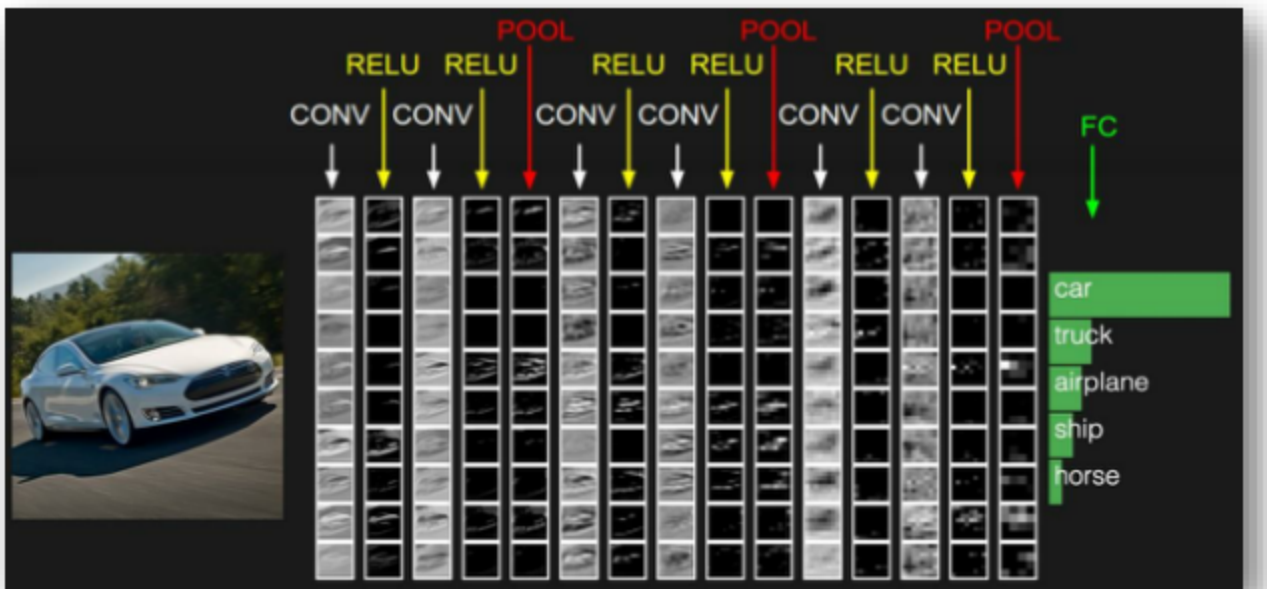
- FC (fully-connected layer): computes class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score (here: for 10 classes). Each element („neuron“) in this layer is connected to all the numbers in the previous volume

- + dropout & loss layer (and additional layers for specific CNN variants)

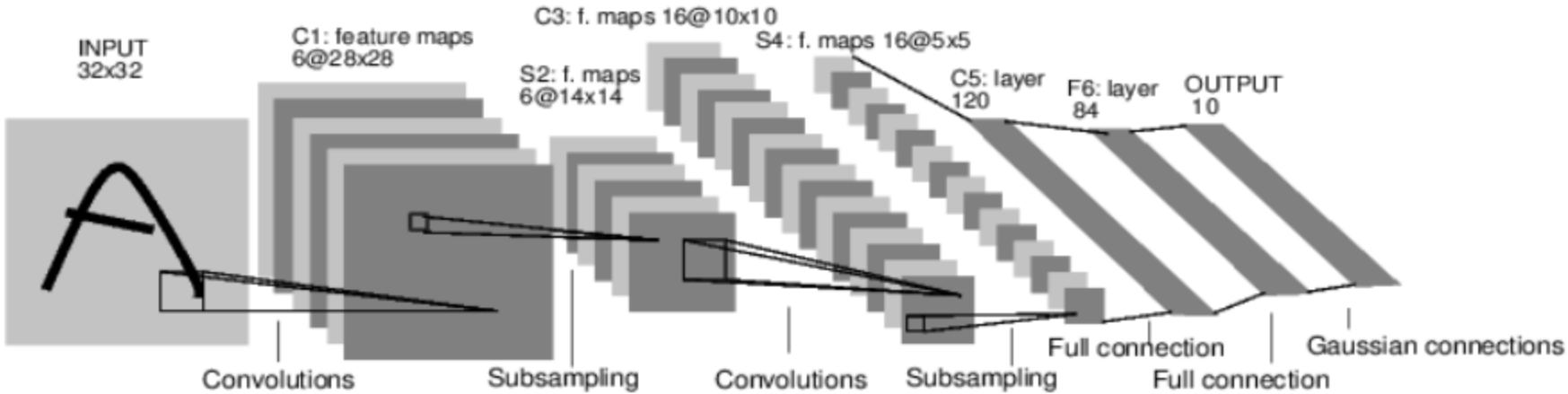
Components: good to know...

- some layers contain *parameters* that *are learned during training* time and others do not
- *CONV/FC layers perform transformations* that are a function of not only the activations in the input volume, but also of the parameters (weights and biases of the neurons)

  learn parameters in CONV/FC trained with (stochastic) gradient descent
- RELU/POOL layers will implement *a fixed function*.



Components: hyperparameters

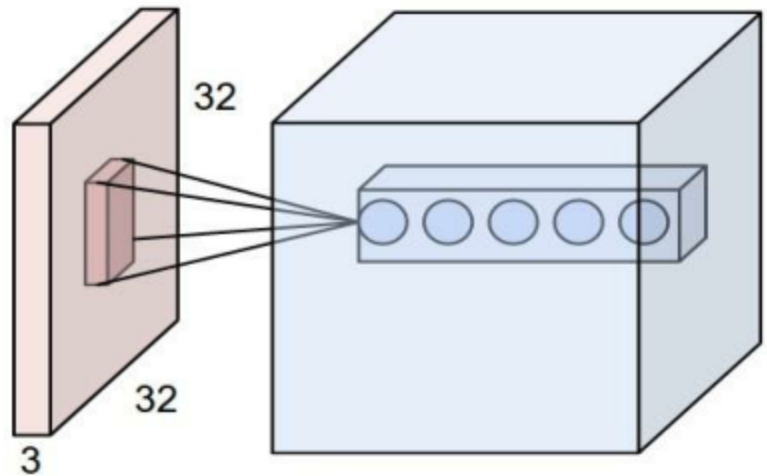


- **Receptive field**: local dimension of neuron (width and height of filter)
  ➔ note: always entire depth of input volume per filter/neuron
- **Depth**: number of neurons in CONV layer that connect to same region of input volume
- **Stride**: if stride is 1 we will allocate a new depth column of neurons to spatial positions only 1 spatial unit apart ➔ heavily overlapping receptive fields between columns
- **Zero-padding**: pad the input with zeros spatially on the border of the input volume

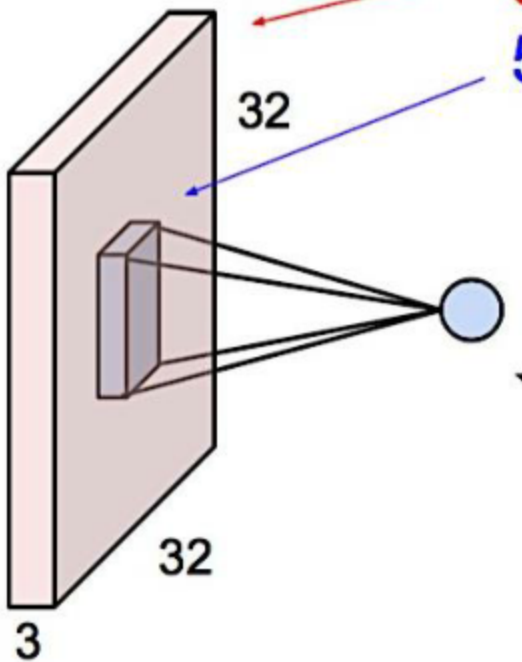
Components: convolutional layer

- CONV layer parameters consist of a set of ***learnable filters***
- ***Convolve*** (i.e., dot product) each filter across width and height of input volume during forward pass
- network will learn filters that activate (i.e., neurons fire) when they see some specific type of feature at some spatial position in the input.
- ***Stacking activation maps*** for all filters along depth dimension forms full output volume.

5 *neurons* in first CONV layer connected to local input region (e.g., 3x3) but full depth (all 3 color channels)



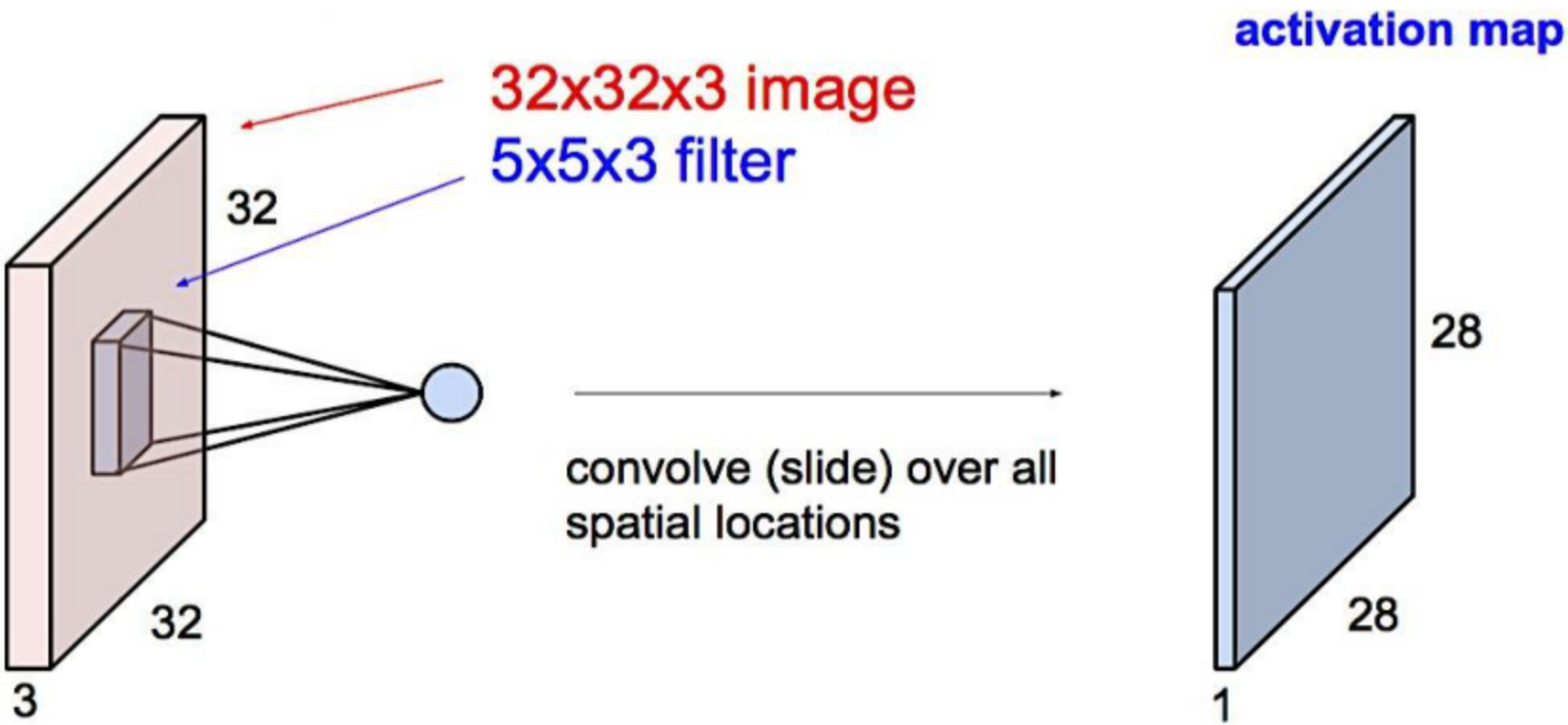
Components: CONV example



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)

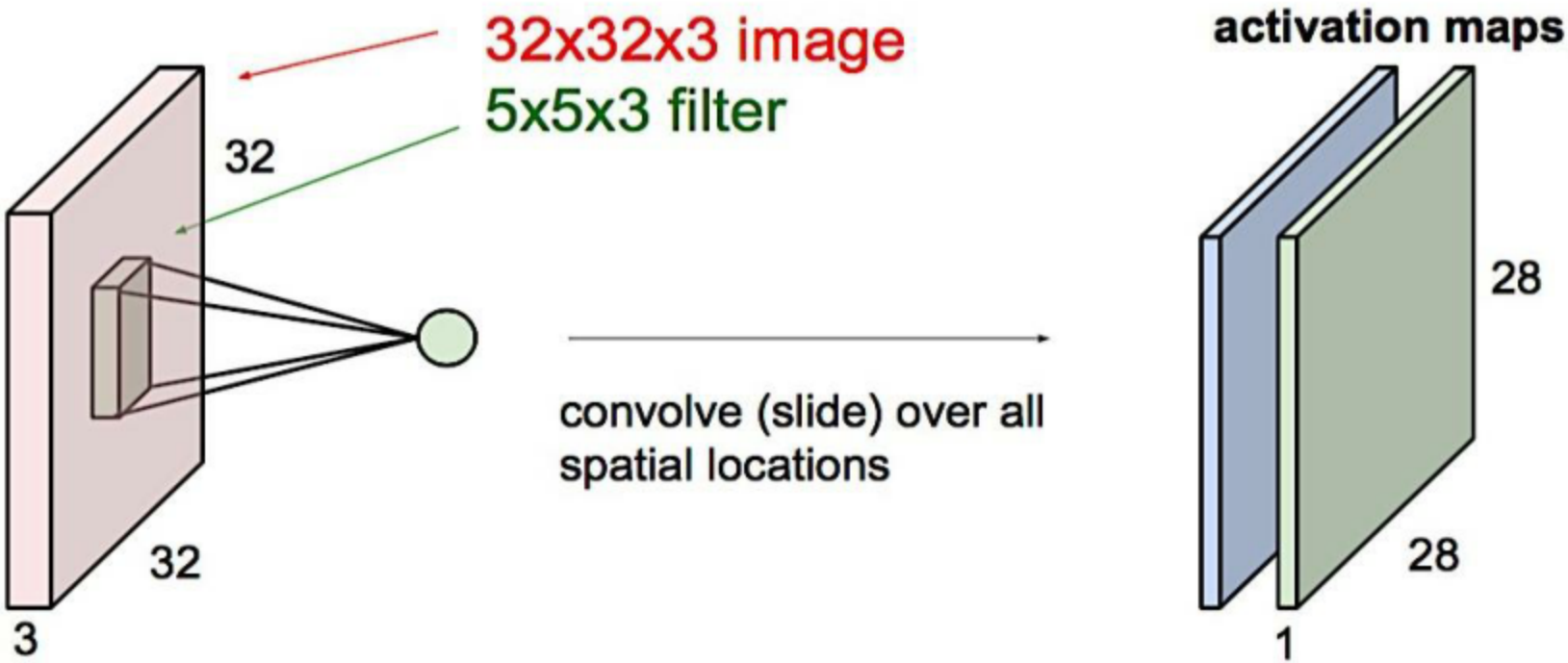
$$w^T x + b$$

Components: CONV example



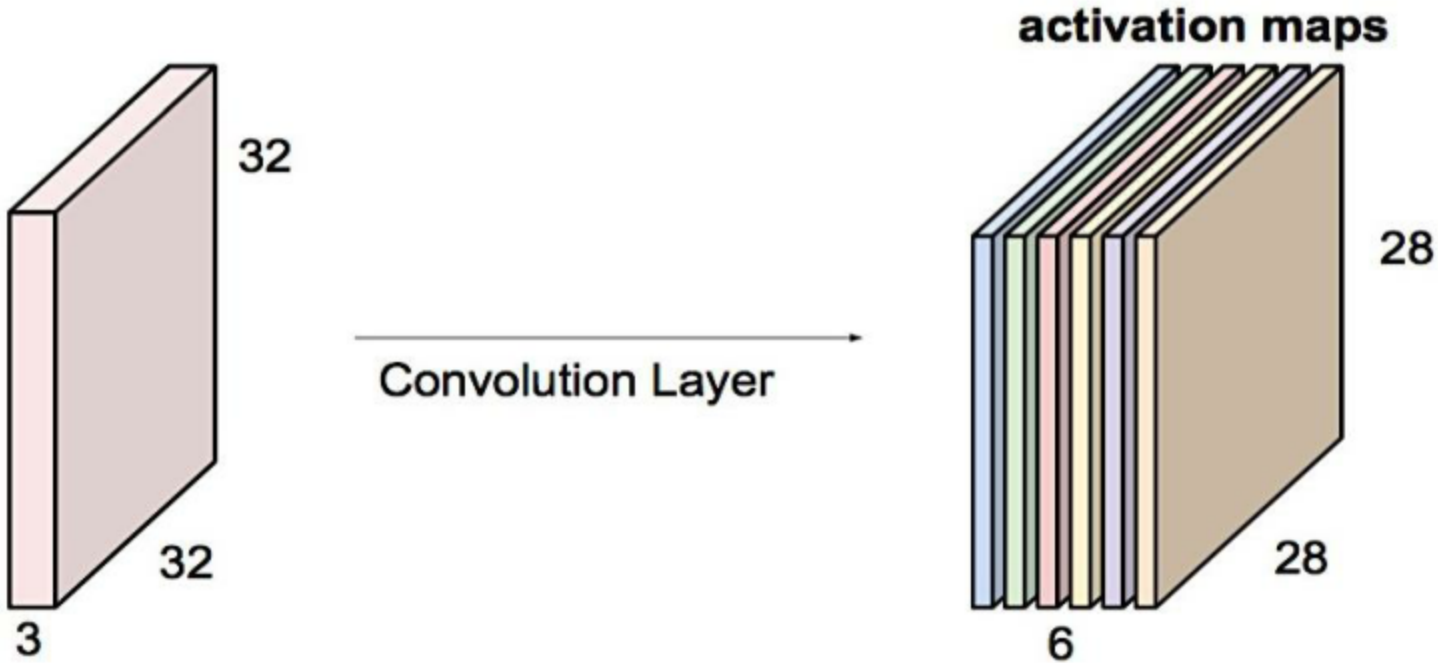
Components: CONV example

consider a second, green filter



Components: CONV example

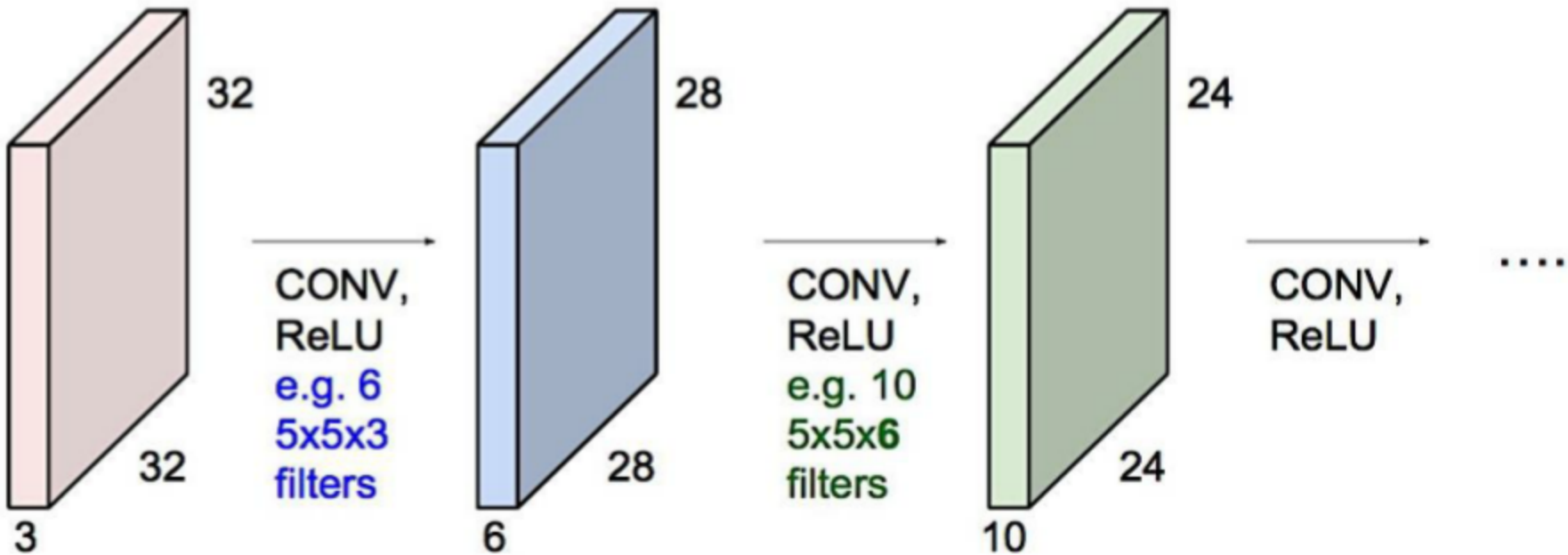
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



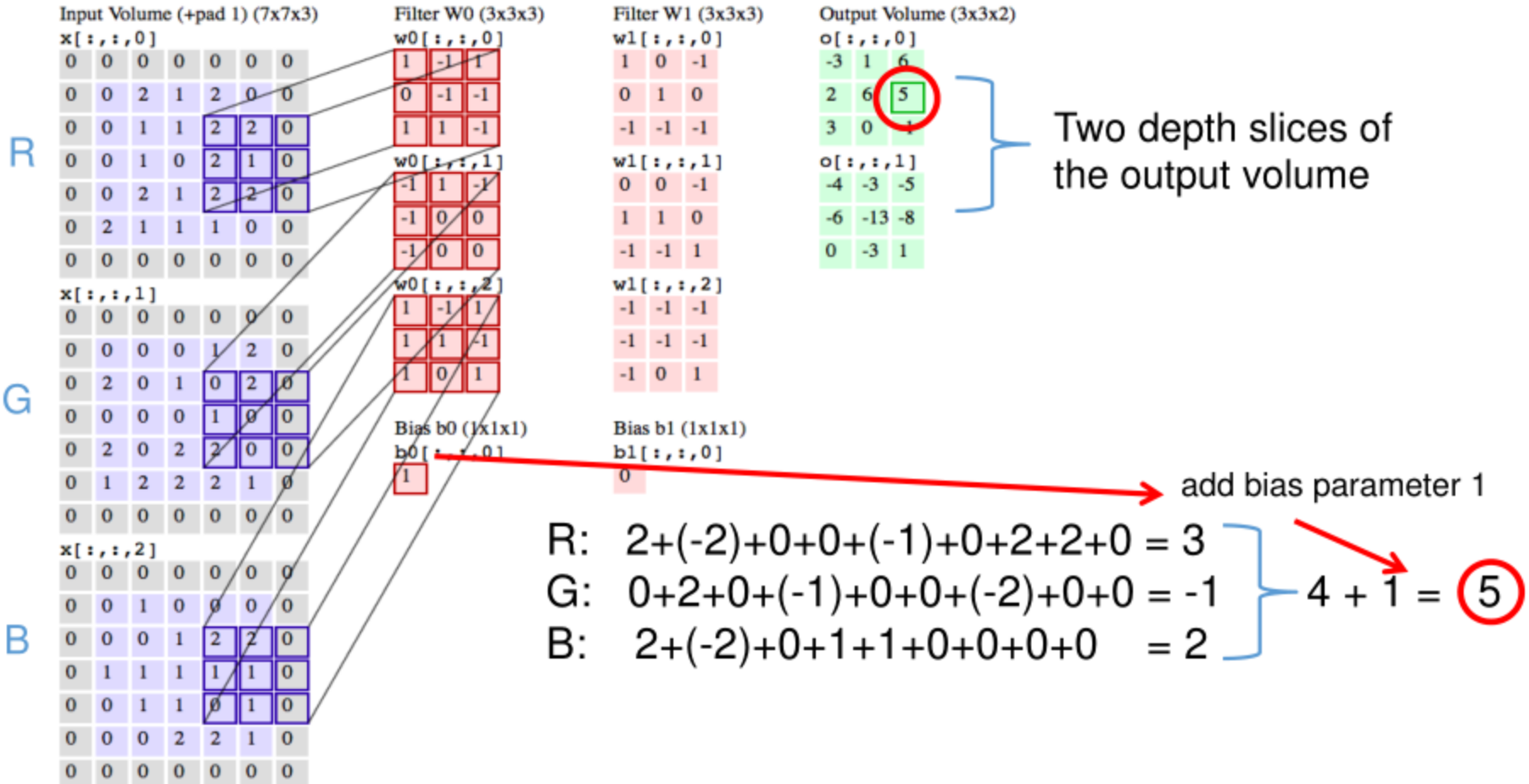
We stack these up to get a "new image" of size 28x28x6!

Components: CONV example

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



Components: CONV example



Components: parameter sharing

***Example*: Krizhevsky et al. 2012** (winner of ImageNet challenge 2012)

- Accepts images of size (227x227x3)
- First CONV layer neurons of receptive field size F = 11, stride S = 4, no zero padding P = 0
- (227 – 11)/4 + 1 = 55) and depth K = 96 ➔ CONV layer output volume has size (55x55x96)
  ➔ 290‘400 neurons

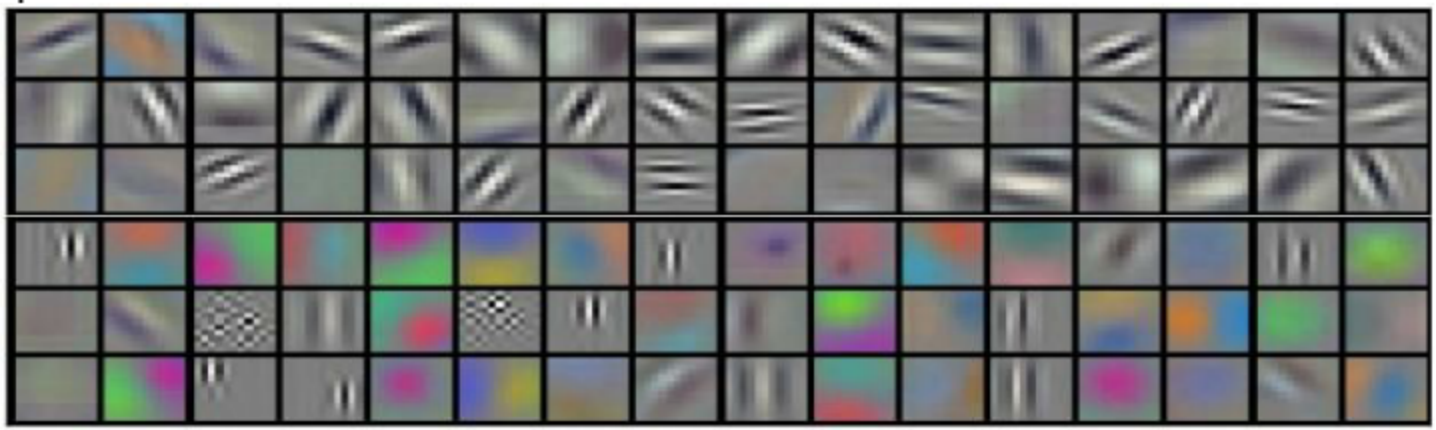
- Each neuron connected to local window of size (11x11x3)
  ➔ if all weights would be computed separately this means 11x11x3 = 363 weights (and 1 bias parameter) ➔ 290‘400 x 364 = 105‘705‘600 parameters!

- **solution:** share weights (and bias) across all neurons of a depth slice x,y
- ➔ only 96 unique sets of weights (one per depth slice) for total of 96x11x11x3 = 34‘848 unique weights (+96 biases)

Components: parameter sharing

If all neurons in a single depth slice use same weight vector, then the forward pass of a CONV layer can in each depth slice be computed as convolution of the neuron's weights with the input volume

Krizhevsky et al. 2012



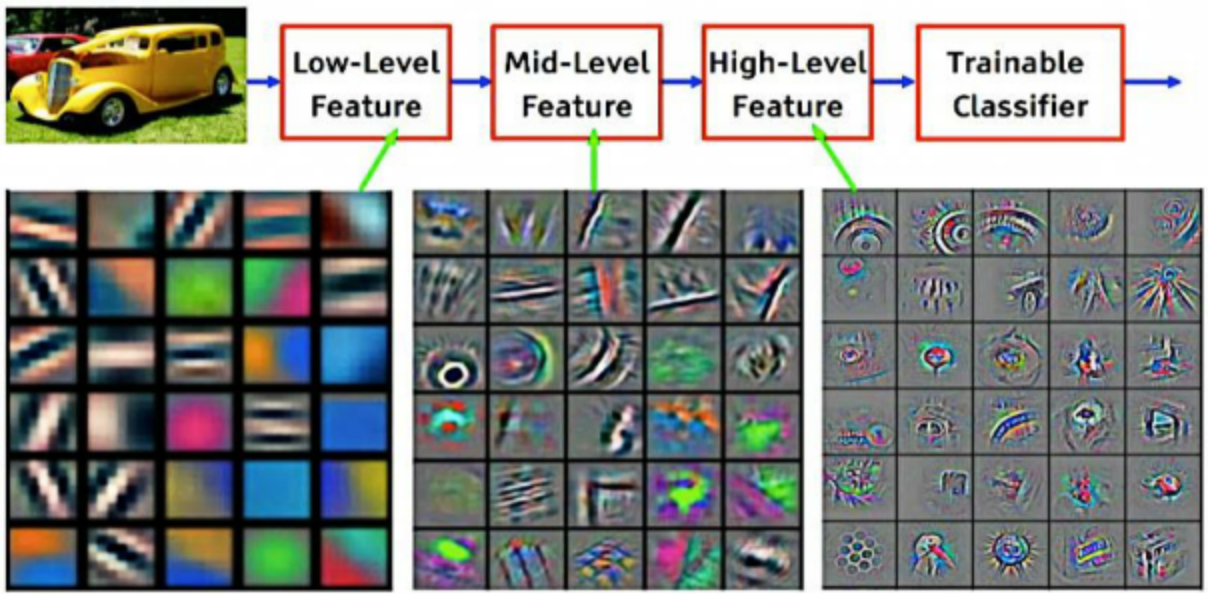
*96 **filters** of size (11x11x3), and each one is shared by the 55x55 neurons in one depth slice*

➔ Result of convolution is a so-called ***activation map*** (e.g., of size 55x55 per depth slice)

➔ Set of activation maps for each different filter are ***stacked along depth dimension to produce output volume*** (e.g., 55x55x96)

Components: CONV example

[From recent Yann LeCun slides]



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

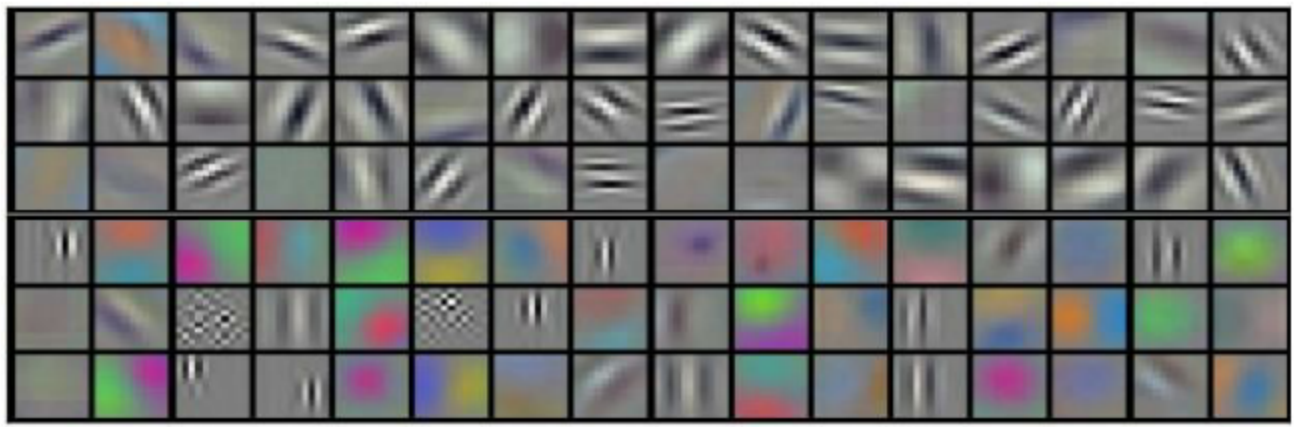
Low-level filters (features): learn primitve patterns, usually blobs and gradients

High-level filters (features): capture object parts (e.g., wheels, faces) and entire objects

Components: learned filters!

All filters *that convolve the image* ***are learned*** *discriminatively (via stochastic gradient descent and backpropagation) such that they* ***adapt to*** *the* ***specific data and specific task****!*

Krizhevsky et al. 2012



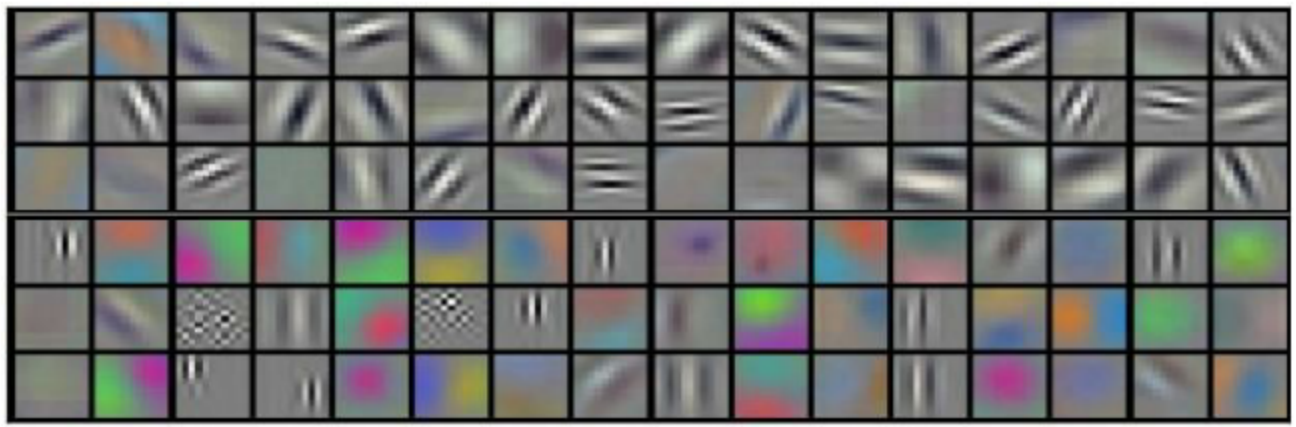
Backpropagation:

- For learning filters (given class labels) we use backpropagation.
- A backward pass for a convolution operation (for both the data and the weights) is also a convolution but with spatially-flipped filters.

Components: learned filters!

All filters *that convolve the image* ***are learned*** *discriminatively (via stochastic gradient descent and backpropagation) such that they* ***adapt to*** *the* ***specific data and specific task****!*

Krizhevsky et al. 2012



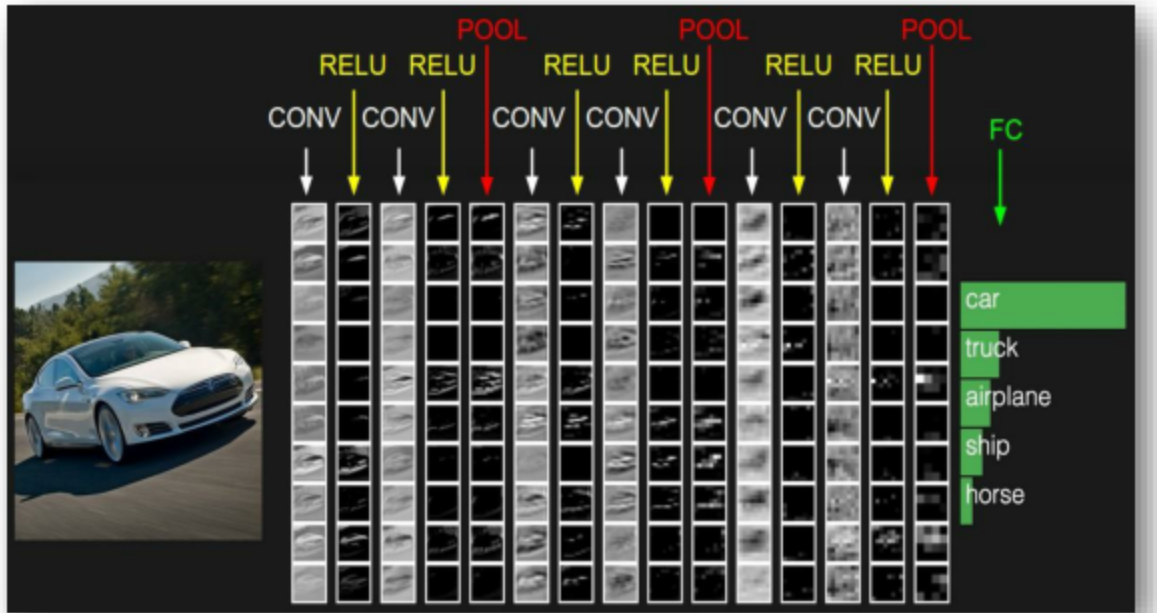
This is the main power of deep learning*!*

Components: pooling layer

Remember:
RELU layer applies max(0,x) to suppress negative values

Pooling:

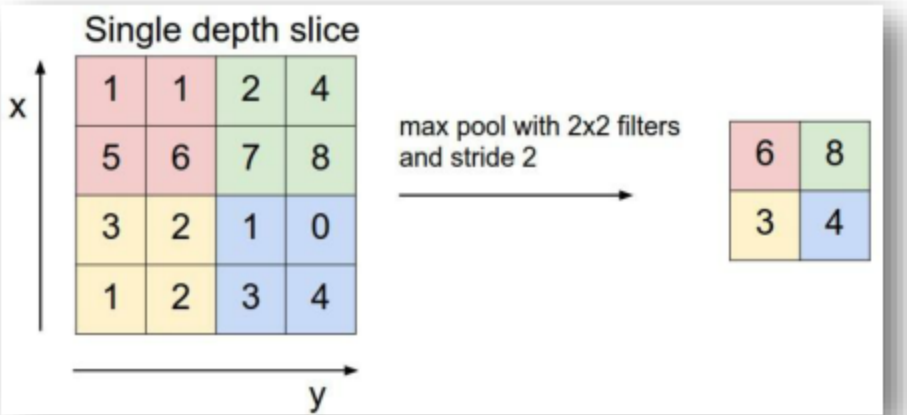
- progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network
  ➔ control (avoid...) overfitting



- MAX operation usually with filters of size 2x2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations.

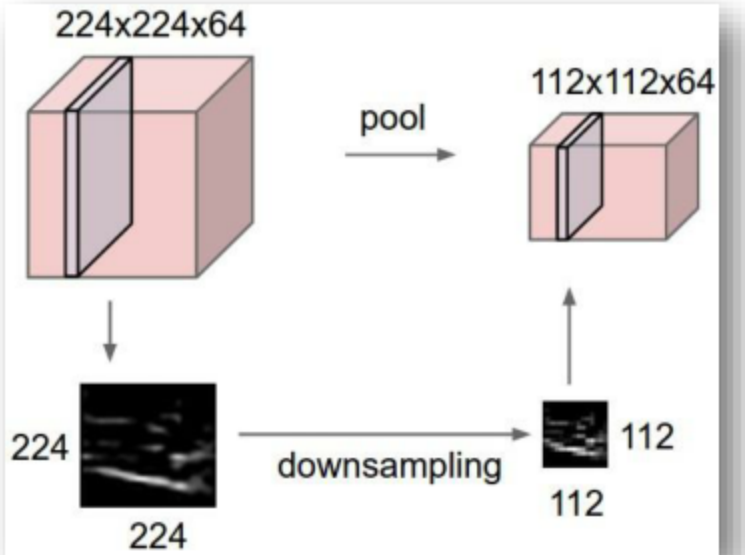
Components: pooling layer

Downsampling of input layer (activation map) with MAX pooling and stride 2



operates over each activation map independently

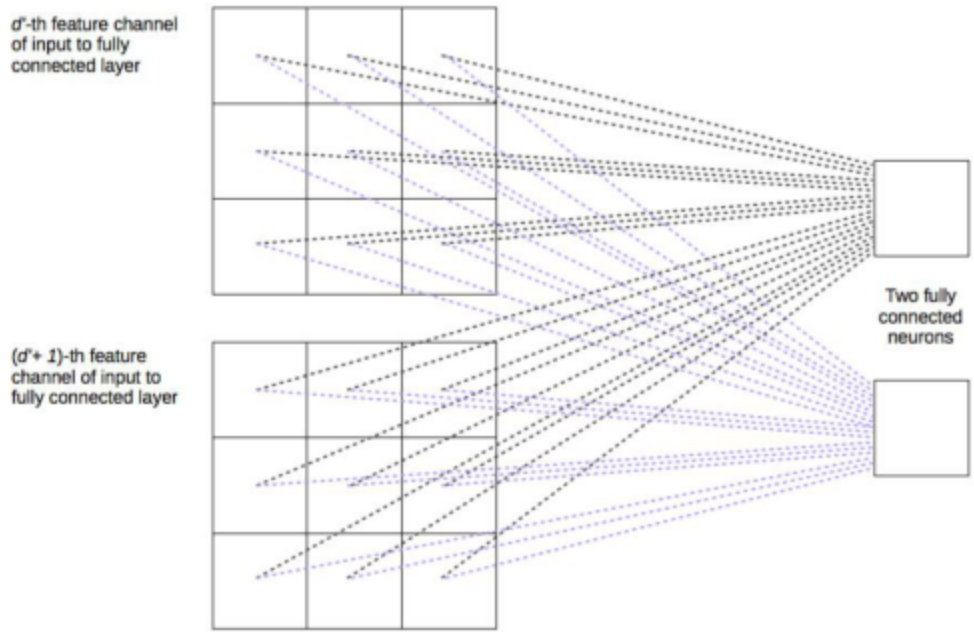
Shrinks width and height of activation maps but leaves depth unchanged



But: current trend in literature is towards discarding the pooling layer in modern CNNs ➔ only CONV layers for simplicity ➔ select larger CONV stride once in a while to reduce overfitting

Components: fully-connected (FC) layer

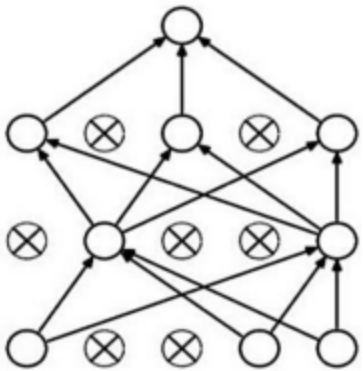
- usually the top layer(s)
- have full connections to all activations in the previous layer
- responsible for ”high-level reasoning”
- only followed by dropout, softmax and/or loss layers



- Technically, fully connected layers are convolutional layers with filters that have the same size as the input layer ➔ Parameters are thus learned like in the case of convolutional layers.

Components: dropout layer

FC layers occupy many parameters in a CNN ➔ slow to train and prone to overfitting
➔ use dropout layer (takes outputs of fully connected layers as inputs)



Forces the network to have a redundant representation.

At each training stage, individual input neurons of a dropout layer are dropped out of the net (or down-weighted), incoming and outgoing edges to dropped-out neurons are removed from the net. Before starting the next training stage, the removed neurons and their incoming and outgoing edges are reinserted into the network (and others are dropped).

Components: loss layer

loss functions used for any (non-CNN) classifer/regressor ➔ type of loss function depends on the kind of task that is learned

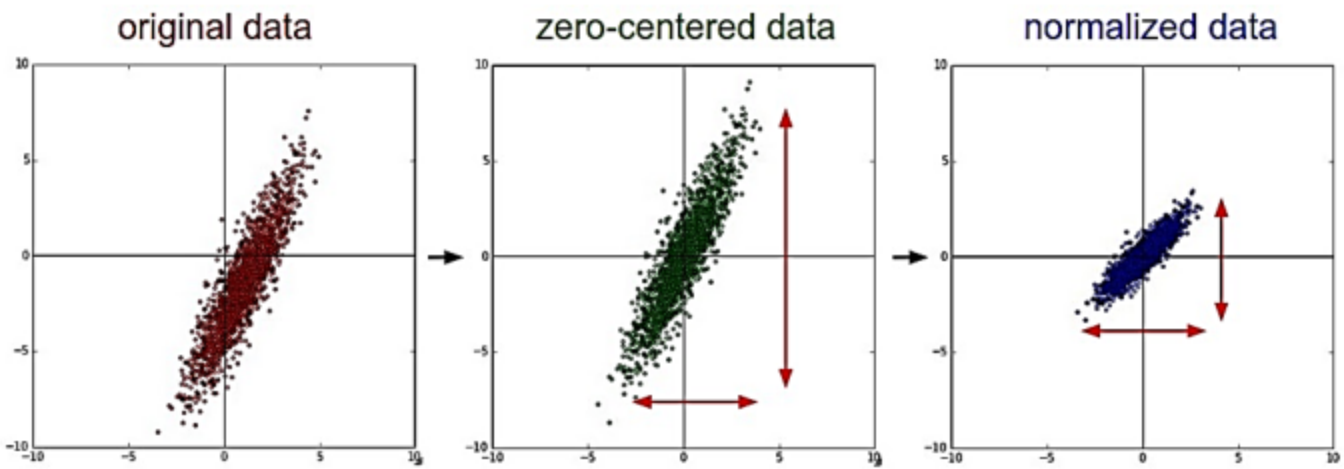
- **Softmax loss** or **multinomial logistic loss**: prediction of a class out of N mutually exclusive classes (e.g., multi-class classifiers)
- **Sigmoid cross-entropy loss**: prediction of N independent probability values in [0, 1]
- **Euclidean loss**: for regression, i.e. continuous, real-valued labels

Example: Semantic pixel-wise classification

- Compute (multinomial logistic) loss for each pixel of the input
- Sum all pixel-wise multinomial logistic losses to one image-wide multinomial logistic loss which evaluates the ”cost” associated with the entire image.

1.2 Data preprocessing

- mean subtraction: across every individual color channel (centers data cloud around origin)
- normalization: two common ways
  1. Divide each dimension by its standard deviation (after zero centering) or
  2. rescale such that Min and Max along each dimension is -1 and 1 respectively

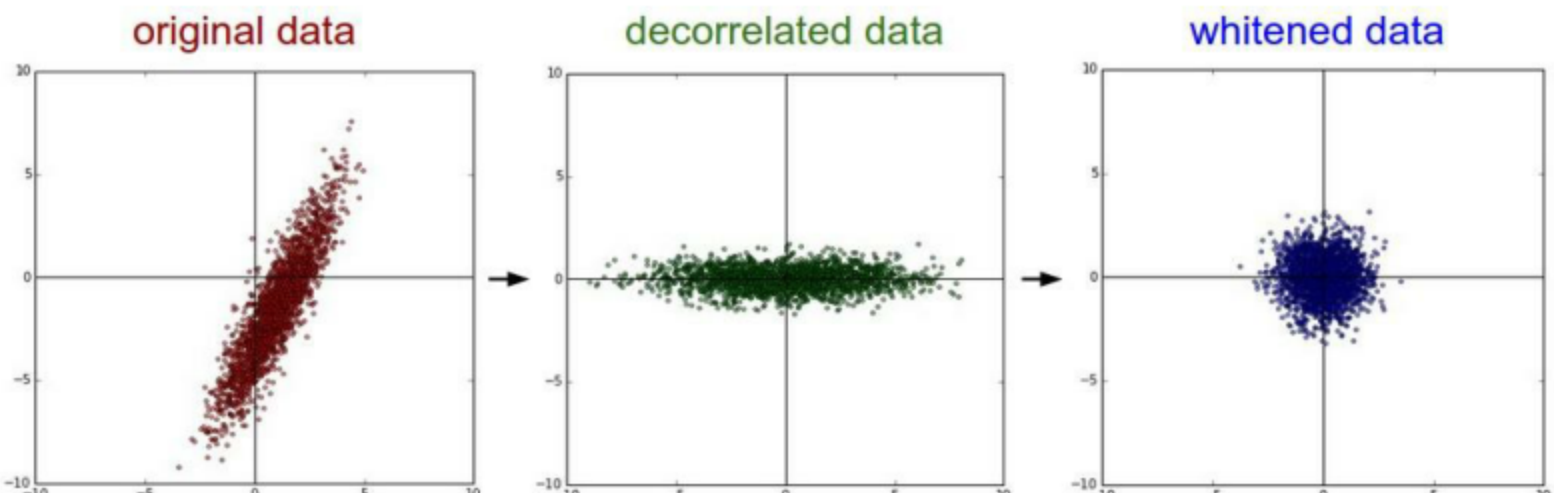


any preprocessing statistics (e.g., the data mean) must **only be computed on the training data**, and then **applied to the validation / test data**.

Data preprocessing

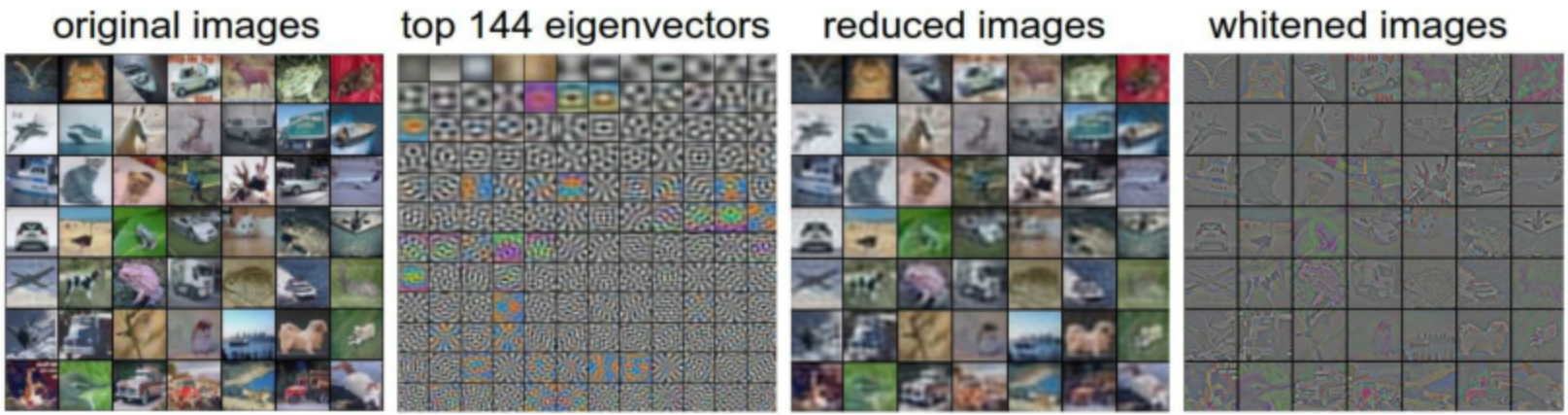
PCA and Whitening are another form of preprocessing (after data has been centered)

- **PCA**: compute covariance matrix ➔ SVD factorization to get eigenvectors ➔ decorrelate data by "rotation" of zero-centered data such that (orthonormal) eigenvectors are the new axis (and discard dimensions with very low variance)
- **Whitening**: take data in eigenbasis and divide every dimension by the eigenvalue to normalize scale (but: can exaggerate noise!)



Data preprocessing: example

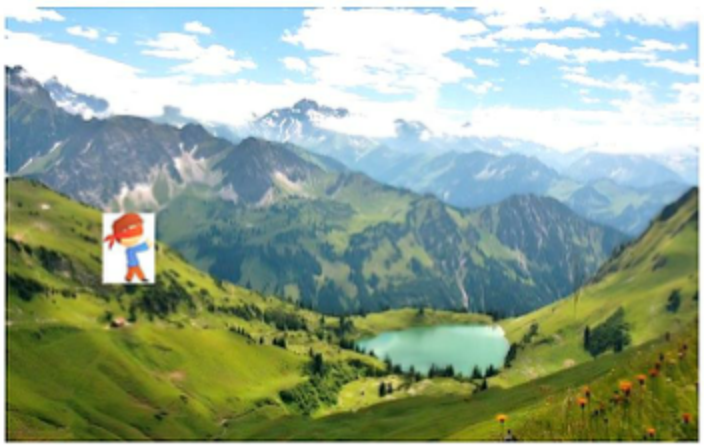
Some training images of CIFAR-10 benchmark:

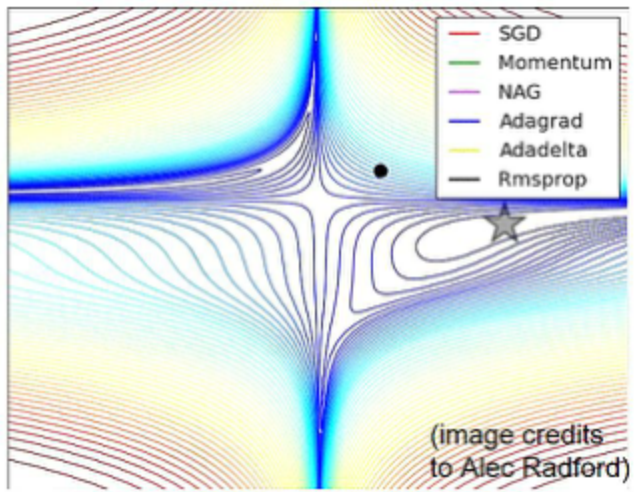


Note: *PCA and whitening are usually not used with Convolutional Networks…*
…but it is important to zero-center the data and normalization of every pixel is common, too.

Also: any preprocessing statistics (e.g. the data mean) must only be computed on the training data, and then applied to the validation / test data.

1.3 Training deep networks





Task during training: Minimize loss function ➔ find "globally lowest point"

Risk: deep learning non-convex ➔ get stuck in local minimum

…**but**: current understanding is that local minima in deep learning are very similar and usually close to the globally optimal value. *Not fully understood though!*

Training: overview

Today's quasi-gold standard is **Mini-batch (Stochastic) Gradient Descent**
➔ **parallelizable on GPU**

Iterate until convergence:

1. Randomly sample a batch (i.e., a (very small) subset) of data from the training data set
2. Forward propagation through all layers and compute loss
3. Backpropagation to calculate the gradients
4. Update the parameters using the gradient

Backpropagation

- compute the gradients step by step, from the loss layer towards the input, by applying the **chain rule** of differentiation

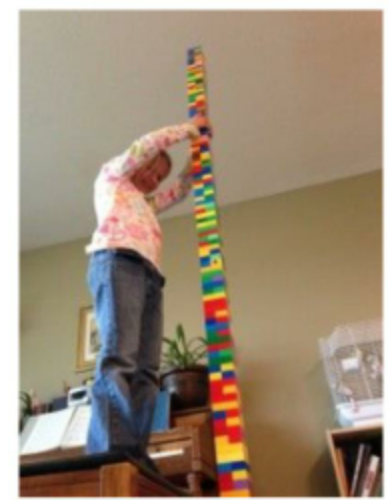
$$y = f(g(x)) \qquad \frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

- simple example

$$y = e^{ax} \qquad \frac{\partial y}{\partial x} = \frac{\partial y}{\partial q}\frac{\partial q}{\partial x} = ae^{ax}$$

$$y = e^q \qquad \frac{\partial y}{\partial q} = e^q$$

$$q = ax \qquad \frac{\partial q}{\partial x} = a$$

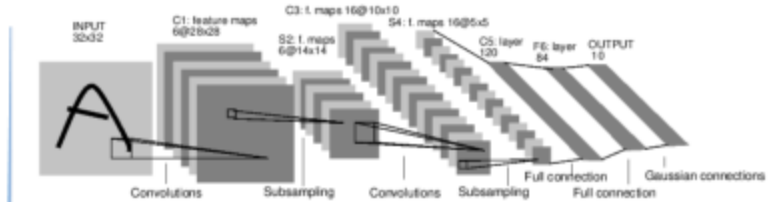


- deep network is just a more complicated function

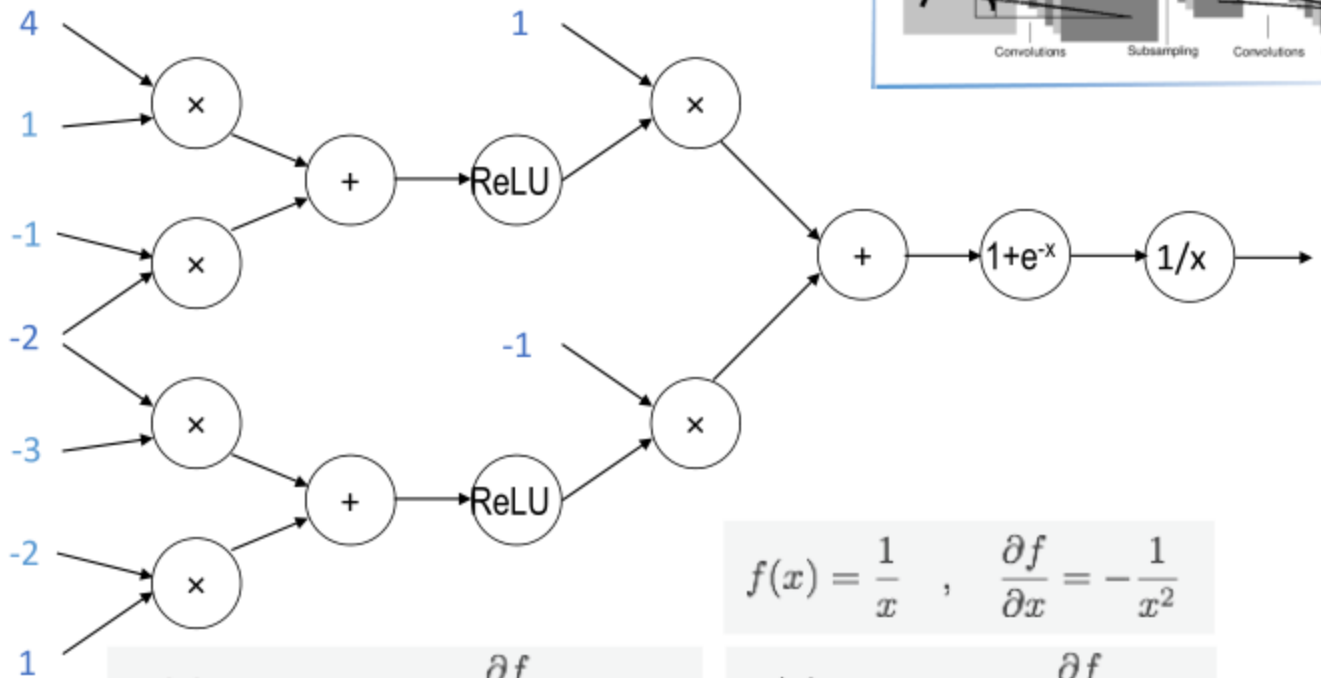
$$y = f(g(h(\dots(x))))$$

Backpropagation

Remember:



worked toy example



$$f(x) = \frac{1}{x} \quad , \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

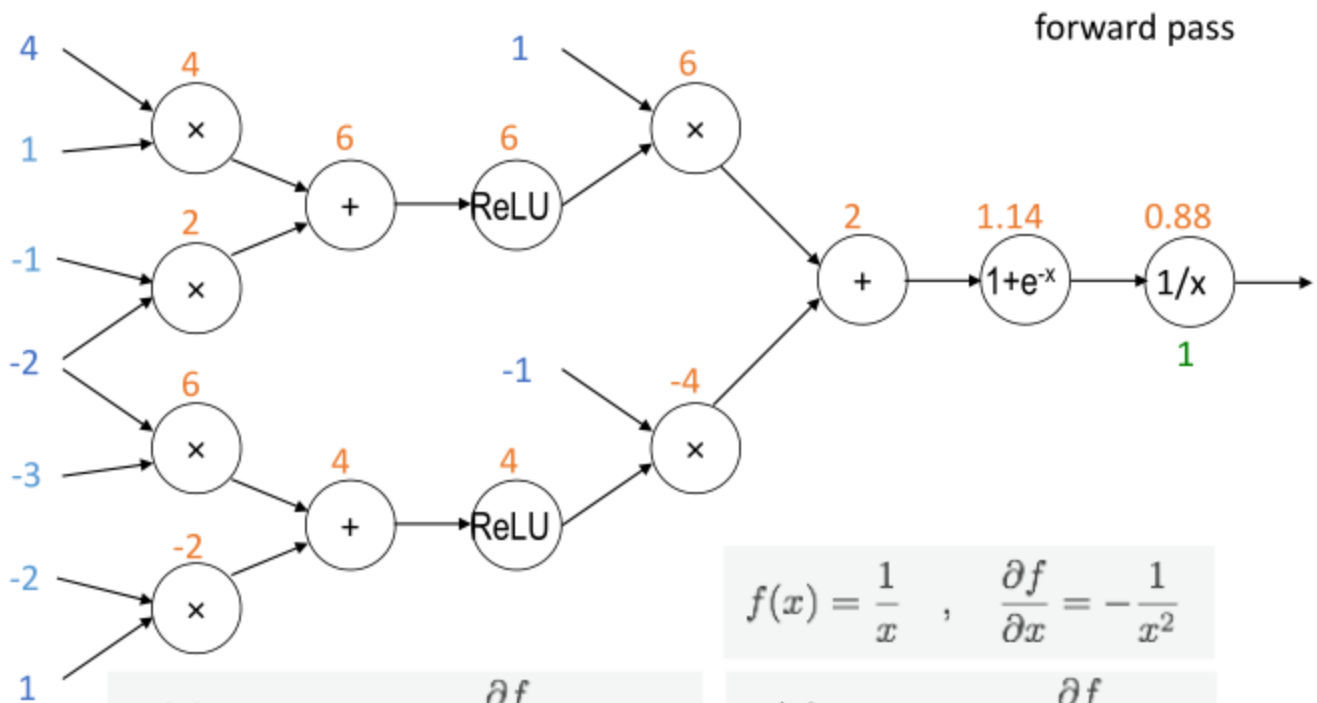
$$f(x) = 1 + e^{-x} \quad , \quad \frac{\partial f}{\partial x} = -e^{-x}$$

$$f(x) = c + x \quad , \quad \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \quad , \quad \frac{\partial f}{\partial x} = a$$

$$f(x) = \mathrm{ReLU}(x) \quad , \quad \frac{\partial f}{\partial x} = \begin{cases} 1 \ldots x > 0 \\ 0 \ldots x \leq 0 \end{cases}$$

Backpropagation



$$f(x) = \frac{1}{x} \quad , \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

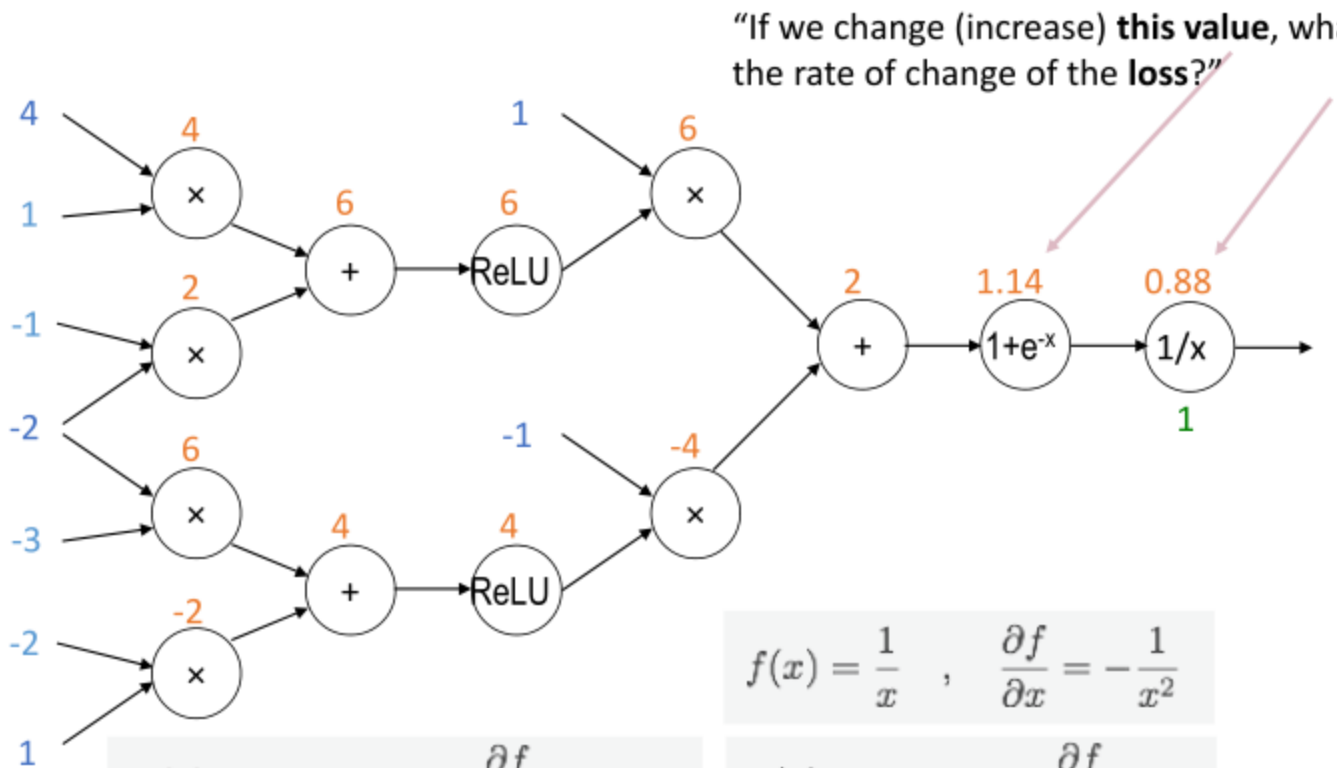
$$f(x) = 1 + e^{-x} \quad , \quad \frac{\partial f}{\partial x} = -e^{-x}$$

$$f(x) = c + x \quad , \quad \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \quad , \quad \frac{\partial f}{\partial x} = a$$

$$f(x) = \mathrm{ReLU}(x) \quad , \quad \frac{\partial f}{\partial x} = \begin{cases} 1 \ldots x > 0 \\ 0 \ldots x \leq 0 \end{cases}$$

Backpropagation



$$f(x) = \frac{1}{x} \quad , \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

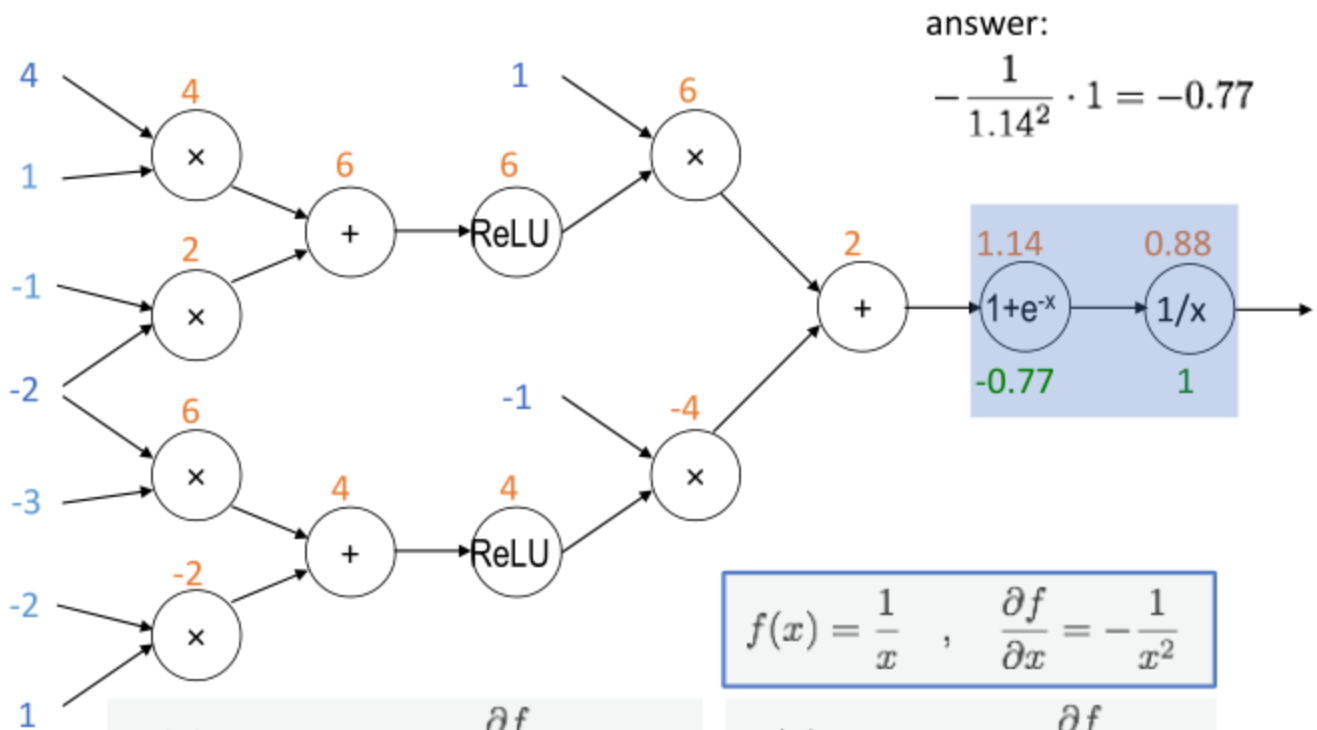
$$f(x) = 1 + e^{-x} \quad , \quad \frac{\partial f}{\partial x} = -e^{-x}$$

$$f(x) = c + x \quad , \quad \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \quad , \quad \frac{\partial f}{\partial x} = a$$

$$f(x) = \mathrm{ReLU}(x) \quad , \quad \frac{\partial f}{\partial x} = \begin{cases} 1 \ldots x > 0 \\ 0 \ldots x \leq 0 \end{cases}$$

Backpropagation



$$f(x) = \frac{1}{x} \quad , \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

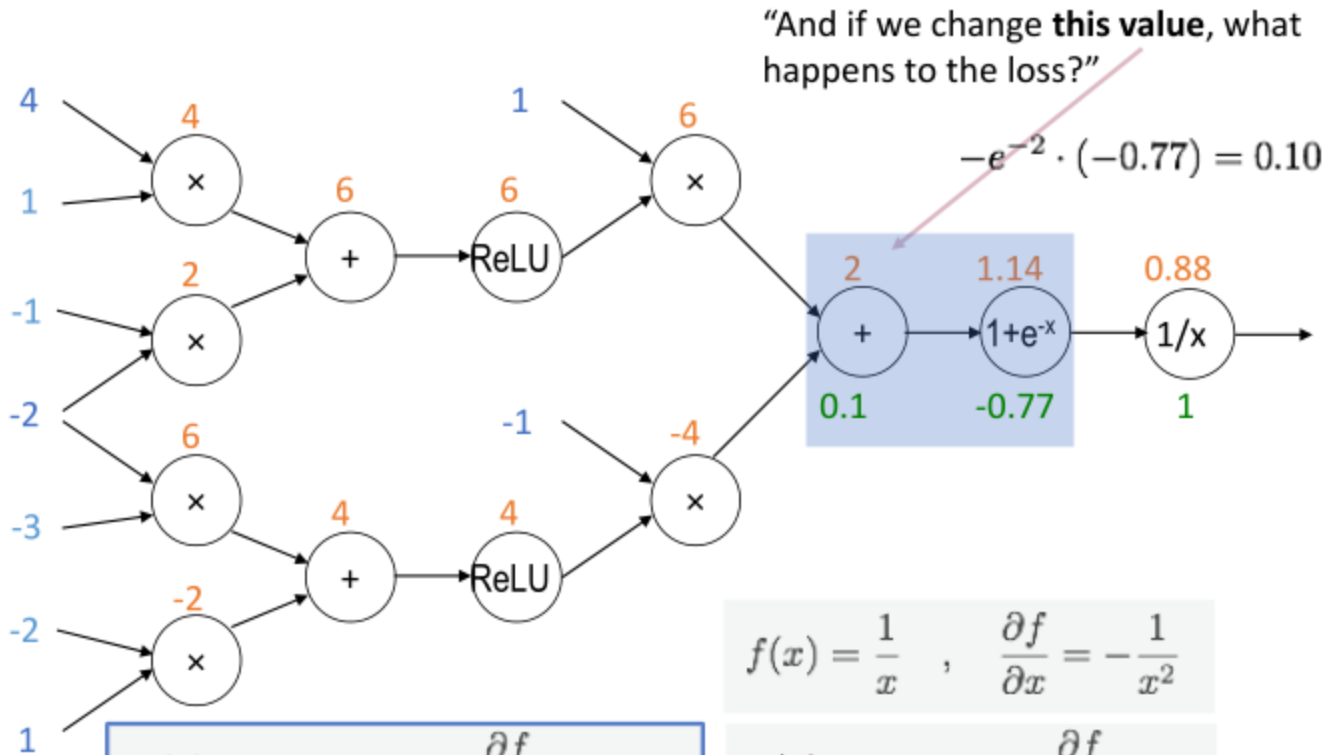
$$f(x) = 1 + e^{-x} \quad , \quad \frac{\partial f}{\partial x} = -e^{-x}$$

$$f(x) = c + x \quad , \quad \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \quad , \quad \frac{\partial f}{\partial x} = a$$

$$f(x) = \mathrm{ReLU}(x) \quad , \quad \frac{\partial f}{\partial x} = \begin{cases} 1 \ldots x > 0 \\ 0 \ldots x \leq 0 \end{cases}$$

Backpropagation



$$f(x) = 1 + e^{-x} \quad , \quad \frac{\partial f}{\partial x} = -e^{-x}$$

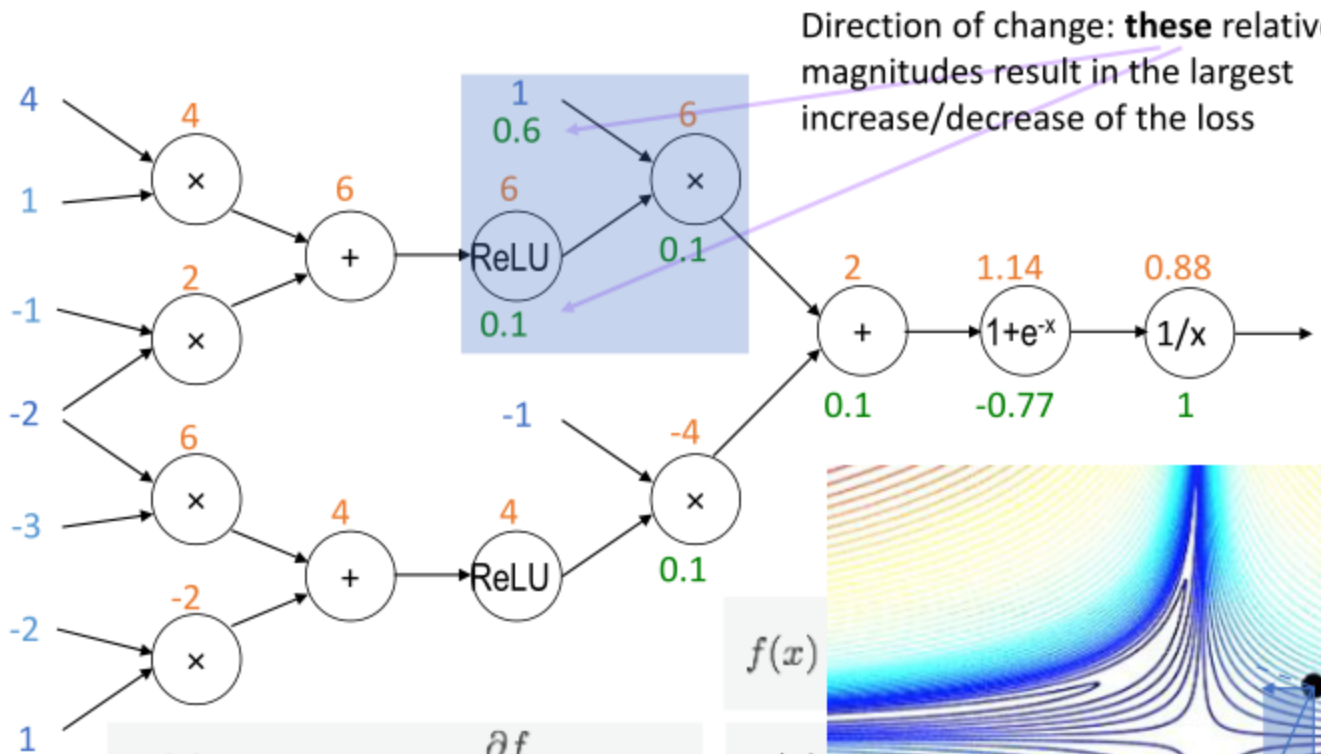
$$f(x) = \frac{1}{x} \quad , \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

$$f(x) = c + x \quad , \quad \frac{\partial f}{\partial x} = 1$$

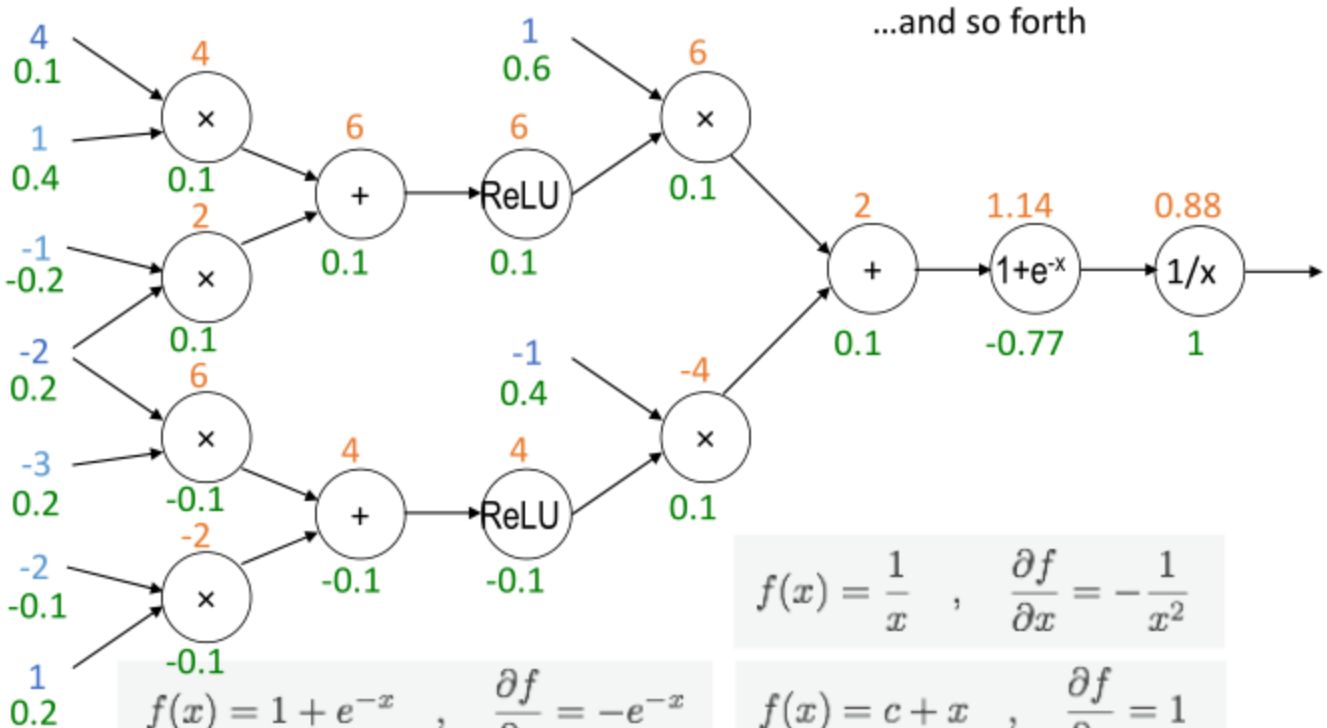
$$f(x) = ax \quad , \quad \frac{\partial f}{\partial x} = a$$

$$f(x) = \text{ReLU}(x) \quad , \quad \frac{\partial f}{\partial x} = \begin{cases} 1 \ldots x > 0 \\ 0 \ldots x \leq 0 \end{cases}$$

Backpropagation



Backpropagation



$$f(x) = \frac{1}{x} \quad , \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

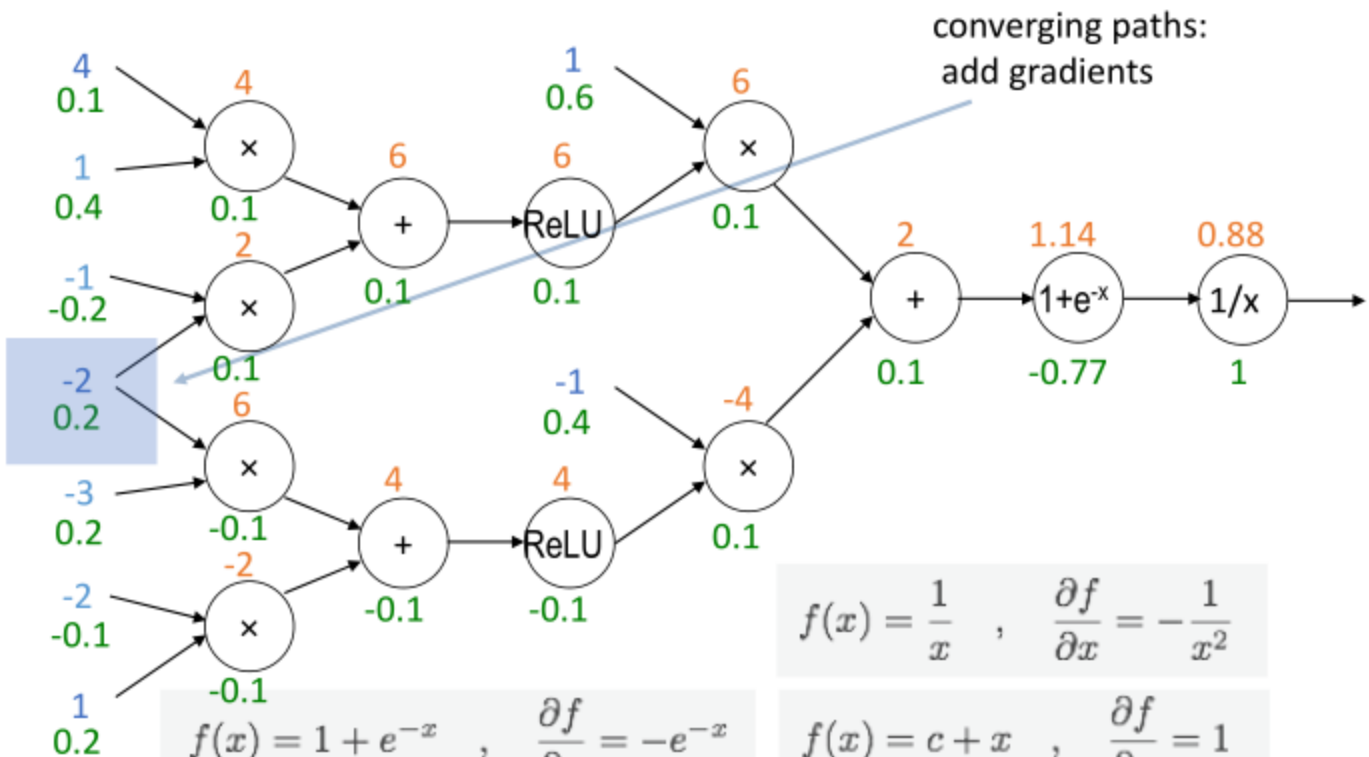
$$f(x) = 1 + e^{-x} \quad , \quad \frac{\partial f}{\partial x} = -e^{-x}$$

$$f(x) = c + x \quad , \quad \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \quad , \quad \frac{\partial f}{\partial x} = a$$

$$f(x) = \mathrm{ReLU}(x) \quad , \quad \frac{\partial f}{\partial x} = \begin{cases} 1 \ldots x > 0 \\ 0 \ldots x \leq 0 \end{cases}$$

Backpropagation



$$f(x) = \frac{1}{x} \quad , \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

$$f(x) = 1 + e^{-x} \quad , \quad \frac{\partial f}{\partial x} = -e^{-x}$$

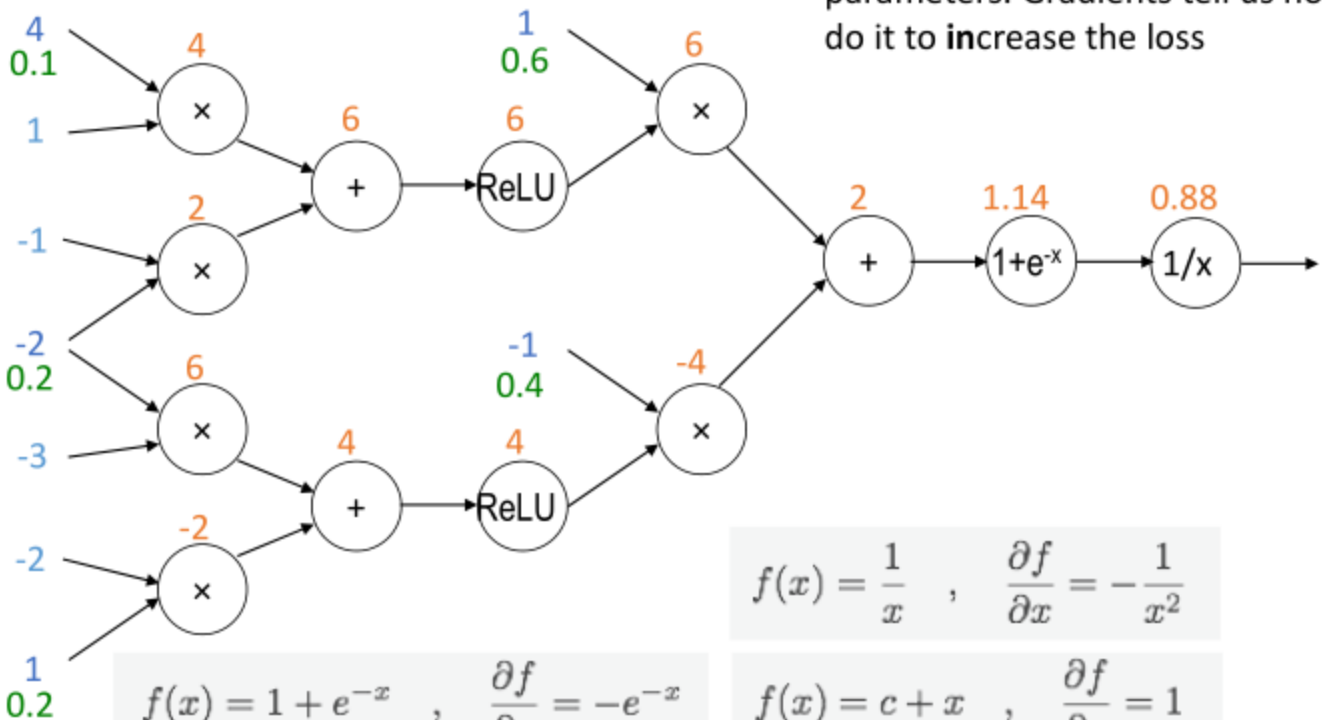
$$f(x) = c + x \quad , \quad \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \quad , \quad \frac{\partial f}{\partial x} = a$$

$$f(x) = \mathrm{ReLU}(x) \quad , \quad \frac{\partial f}{\partial x} = \begin{cases} 1 \ldots x > 0 \\ 0 \ldots x \leq 0 \end{cases}$$

Backpropagation

…we can only change the free parameters. Gradients tell us how to do it to **in**crease the loss



$$f(x) = \frac{1}{x} \quad , \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

$$f(x) = 1 + e^{-x} \quad , \quad \frac{\partial f}{\partial x} = -e^{-x}$$

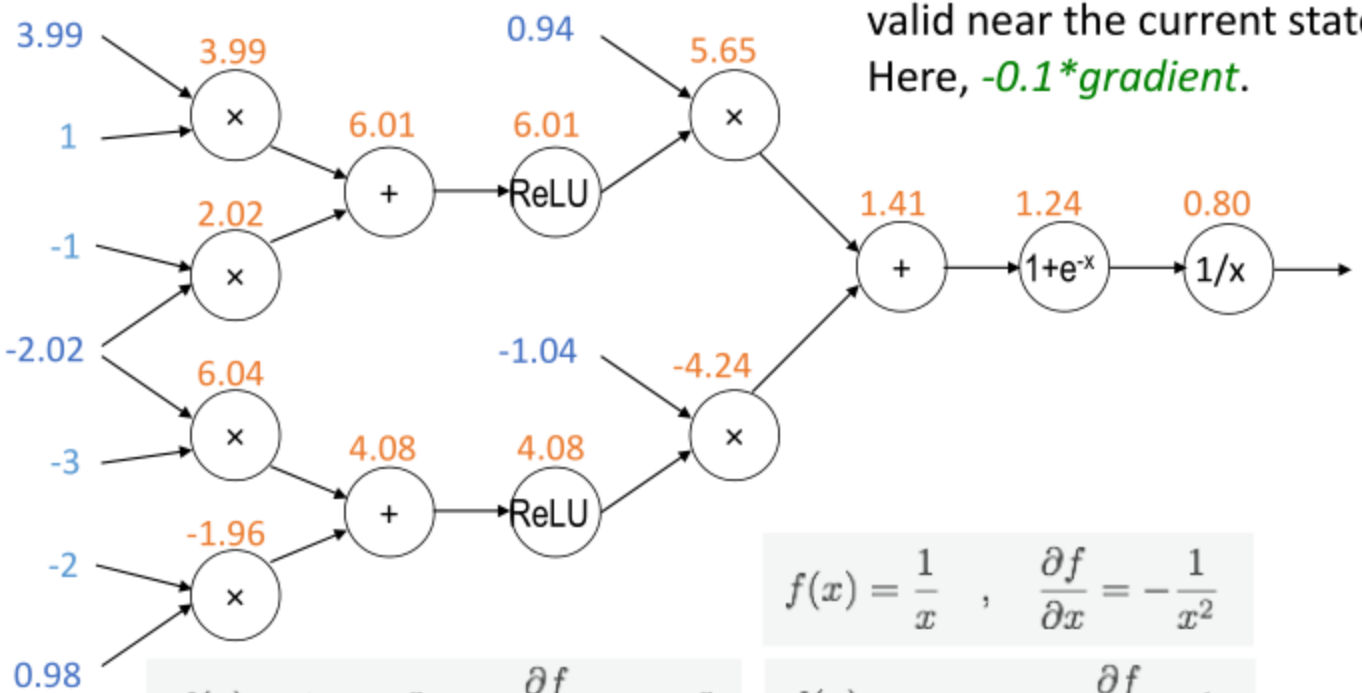
$$f(x) = c + x \quad , \quad \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \quad , \quad \frac{\partial f}{\partial x} = a$$

$$f(x) = \mathrm{ReLU}(x) \quad , \quad \frac{\partial f}{\partial x} = \begin{cases} 1 \ldots x > 0 \\ 0 \ldots x \leq 0 \end{cases}$$

Backpropagation

update parameters (forward pass) to **de**crease loss. Small steps – gradient only valid near the current state!
Here, *-0.1*gradient*.



$$f(x) = \frac{1}{x} \quad , \quad \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

$$f(x) = 1 + e^{-x} \quad , \quad \frac{\partial f}{\partial x} = -e^{-x}$$

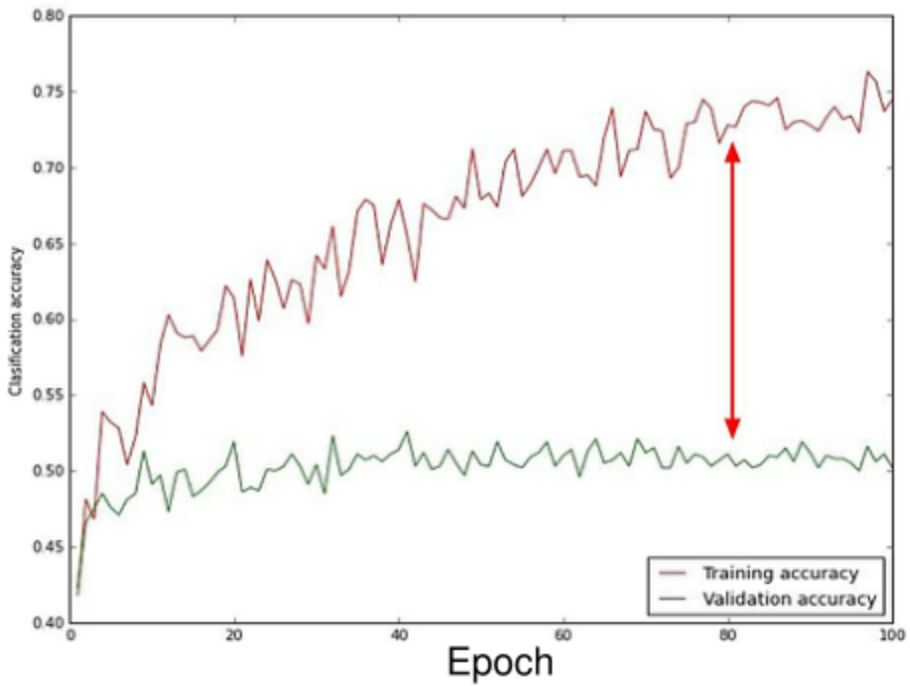
$$f(x) = c + x \quad , \quad \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \quad , \quad \frac{\partial f}{\partial x} = a$$

$$f(x) = \mathrm{ReLU}(x) \quad , \quad \frac{\partial f}{\partial x} = \begin{cases} 1 \ldots x > 0 \\ 0 \ldots x \leq 0 \end{cases}$$

CNN cook book recipes I

Monitor and visualize the loss curve (or classification accuracy):
always plot loss/accuracy curve on training data AND validation data!

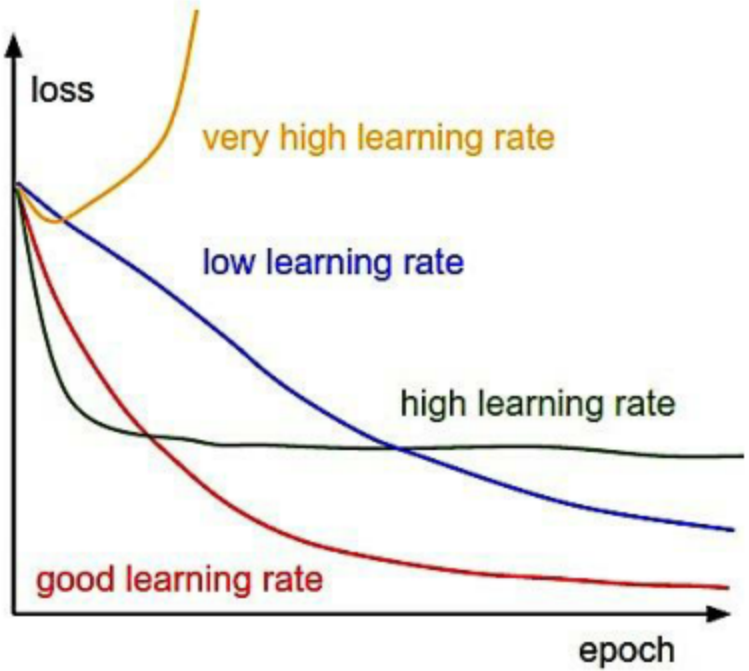


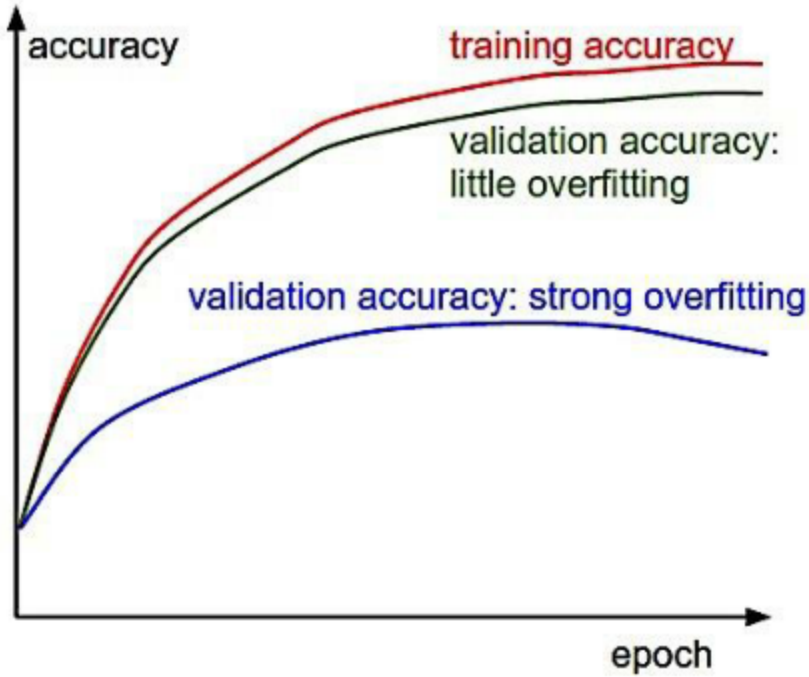
big gap = overfitting
=> increase regularization strength?

no gap
=> increase model capacity?

CNN cook book recipes I

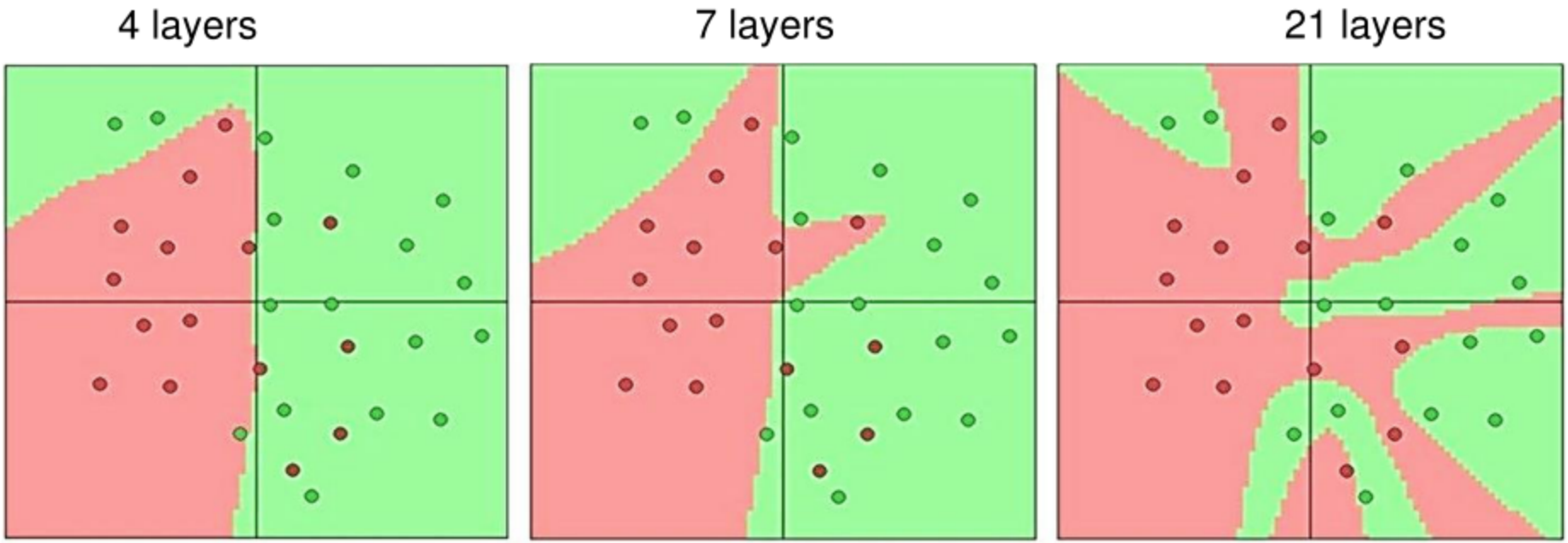
- learning is sensitive to hyper-parameter settings: initialisation, learning rate (decay)…
- monitor loss, and the prediction error for both the training data and a validation set





CNN cook book recipes II

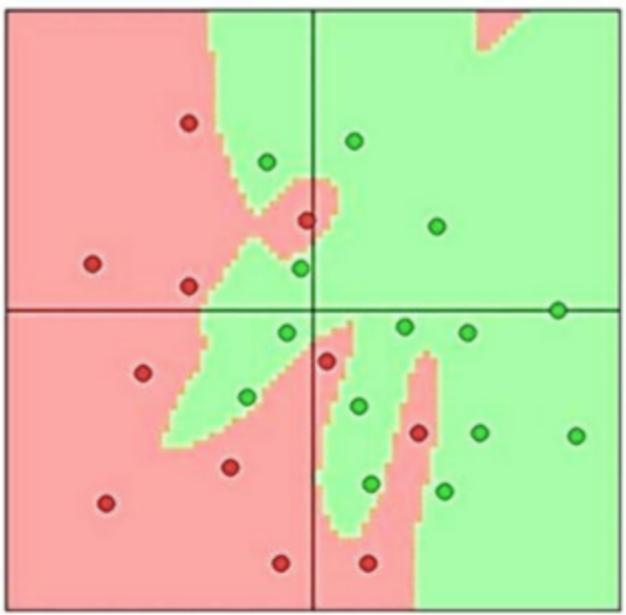
Setting the number of layers and their sizes



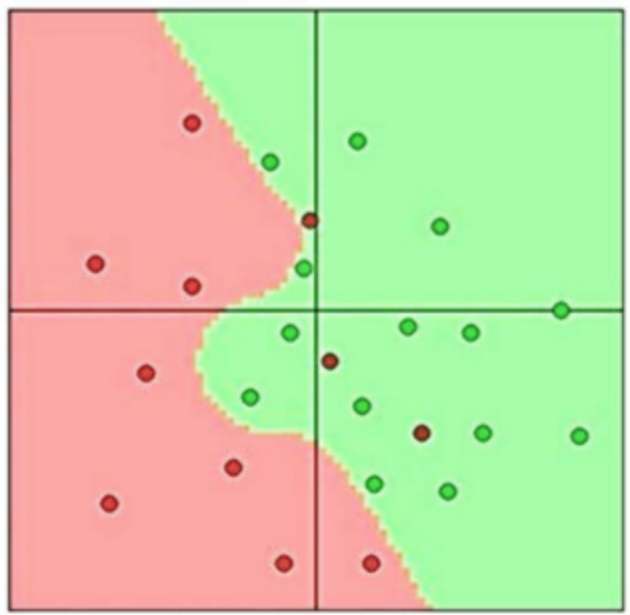
CNN cook book recipes II

Do not use size of network as a regularizer, but stronger regularization:

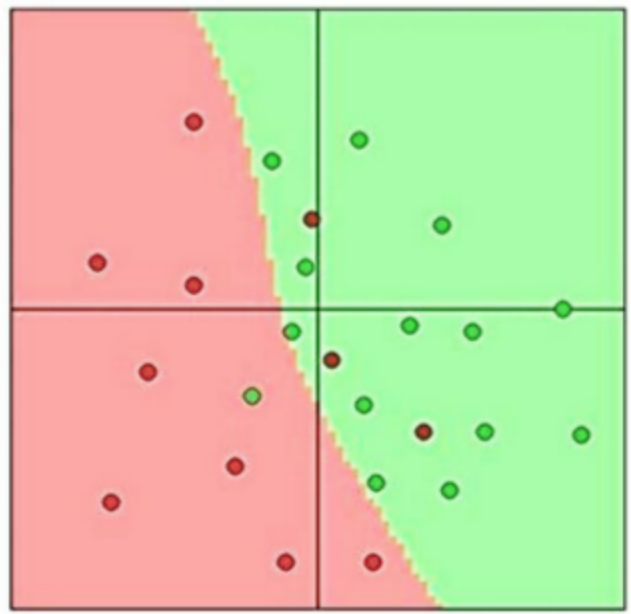
λ = 0.001



λ = 0.01



λ = 0.1



CNN cook book recipes III

Do not train DeepNets from scratch but make use of pre-trained models

Example: pre-training on cats and dogs…

…helps training tree detection…!

Why does this work…?

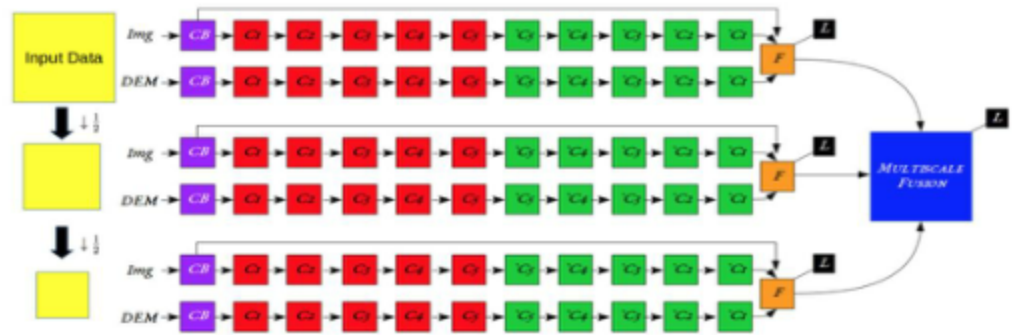
- Low layers capture basic image primitives that are universal for almost any kind of image object.
- Pre-training avoids having to re-train…training procedure can focus capacity on on assembling object parts.

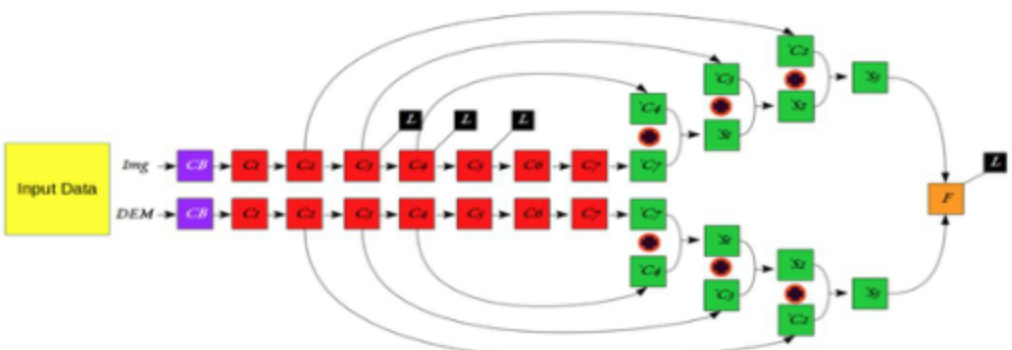
...so now cool stuff?

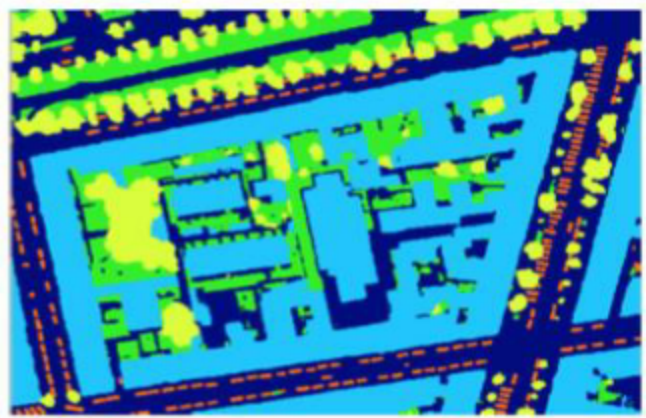
Wait, not yet...but getting better...

2. Applications

Semantic Segmentation







Marmanis, D., Schindler, K., Wegner, J.D., Galliani, S., Datcu, M., Stilla, U. : Classification with an edge: improving semantic image segmentation with boundary detection, ISPRS Journal of Photogrammetry and Remote Sensing, vol. 135, 2018, pp. 158 – 172.

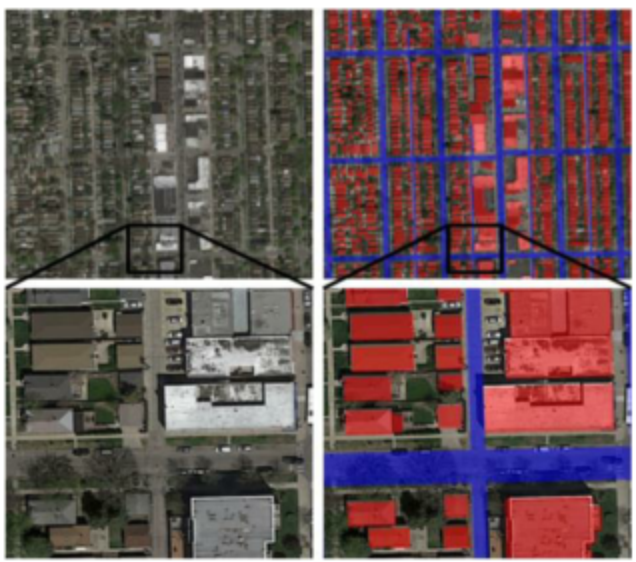
Learning from Massive Open Data

Perhaps the main limitation of deep learning: it is extremely data-hungry

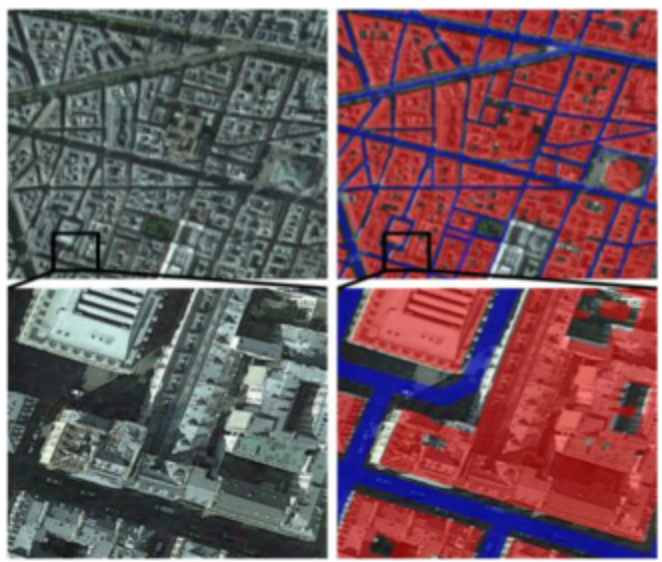
→ exploit huge amounts of free, but noisy online map data for training

OpenStreetMap

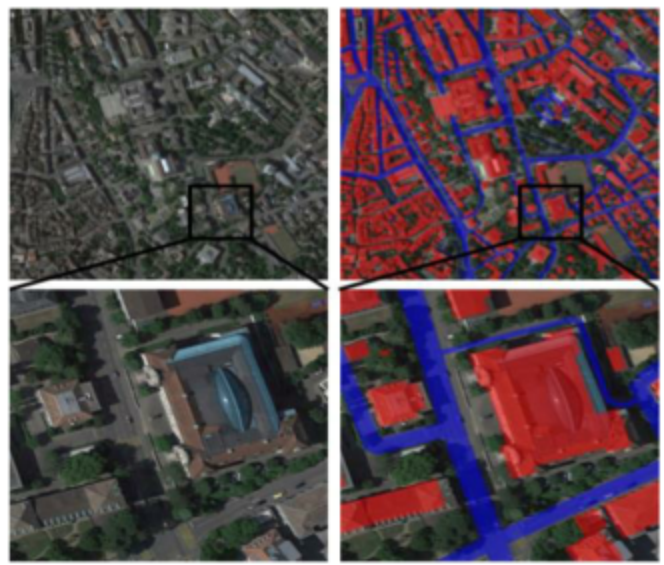
Kaiser, P., Wegner, J.D., Lucchi, A., Jaggi, M., Hofmann, T., Schindler, K.: Learning Aerial Image Segmentation from Online Maps, IEEE Transactions on Geoscience and Remote Sensing, vol 55(11), 2017, pp. 60 – 6068.



Chicago



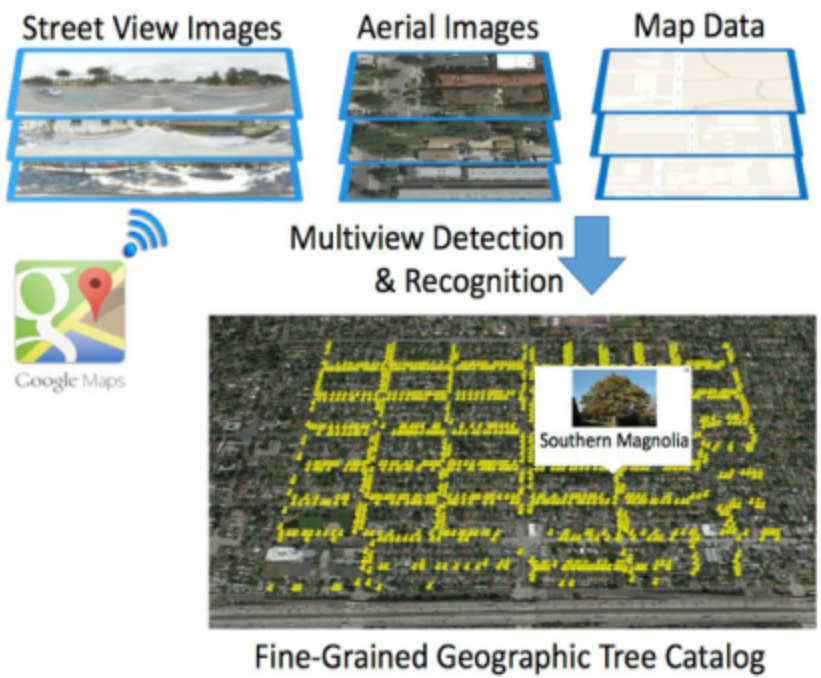
Paris



Zürich

EcoVision PRS Photogrammetry Remote Sensing ETH zürich

Mapping Trees with Open Data



Wegner, J.D., Branson*, S., Hall, D., Schindler, K., Perona, P.: Cataloging Public Objects Using Aerial and Street-level Images – Urban Trees, CVPR, 2016.*

Branson, S., Wegner*, J.D., Hall, D., Lang, N., Schindler, K., Perona, P.: From Google Maps to a Fine-Grained Catalog of Street trees, ISPRS Journal of Photogrammetry and Remote Sensing, vol. 135, 2018, pp. 13 – 30.*

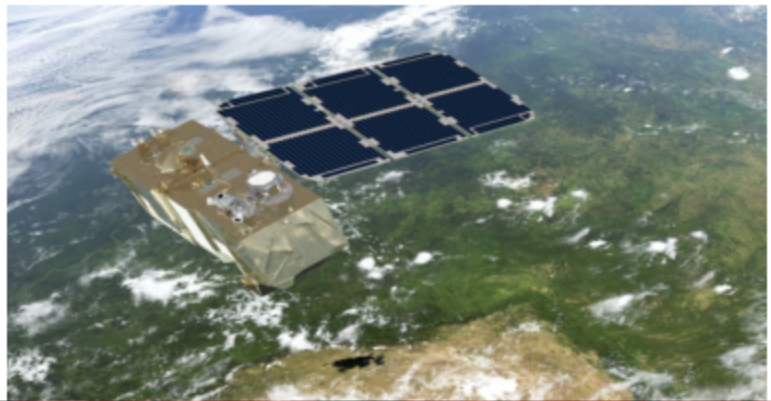
*(*authors contributed equally to this publication)*

EcoVision PRS Photogrammetry Remote Sensing ETH zürich

Agriculture and Environment

Jan Wegner
Nico Lang
Andres Rodriguez

Mapping biomass, tree crops, rainforest cut-down from Sentinel-2 (-1) data

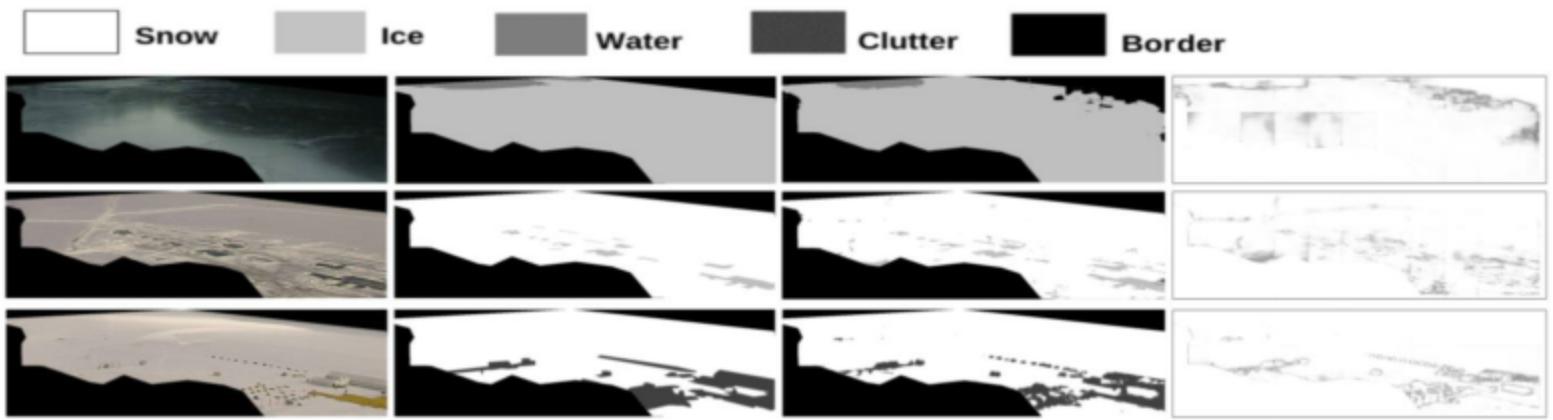


Lake Ice from Webcam Images and MODIS

Monitor ice / snow-on-ice / water for GCOS

Tom, M., Kälin, U., Sütterlin, M, Baltsavias, E., Schindler, K.: Lake Ice Detection in Low-Resolution Optical Satellite Images, ISPRS TC II Symposium (Riva del Garda, Italy, June 2018)

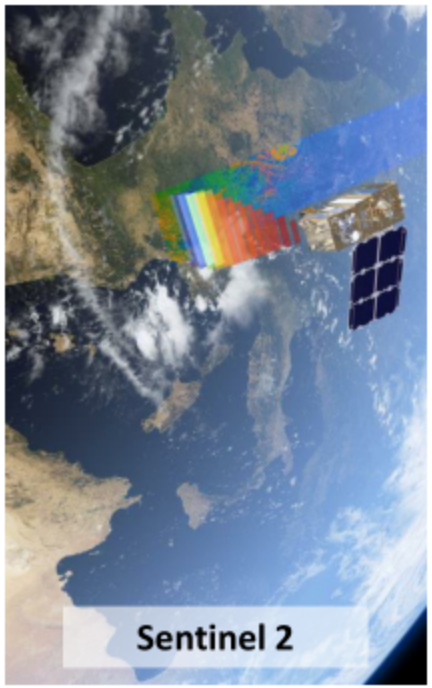
Xiao, M., Rothermel, M., Tom, M., Galliani, S., Baltsavias, E., Schindler, K.: Lake Ice Monitoring with Webcams , ISPRS TC II Symposium (Riva del Garda, Italy, June 2018)

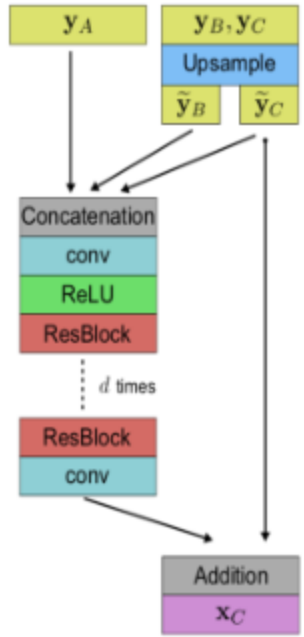


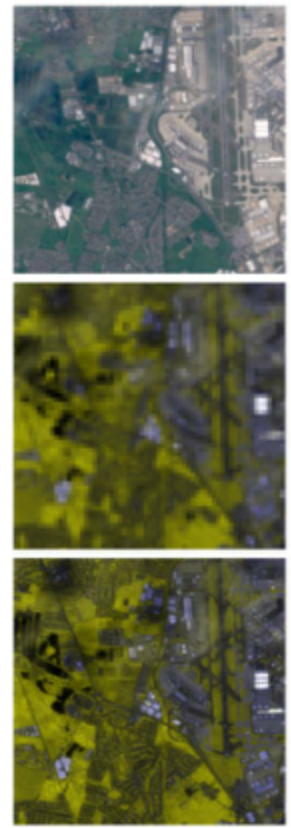
Multi-spectral Super-resolution

Upsampling of multi-resolution sensor bands

Lanaras C., Bioucas-Dias J., Baltsavias E., Schindler K.: Super-Resolution of Multispectral Multiresolution Images from a Single Sensor, CVPR workshop EarthVision, 2017.



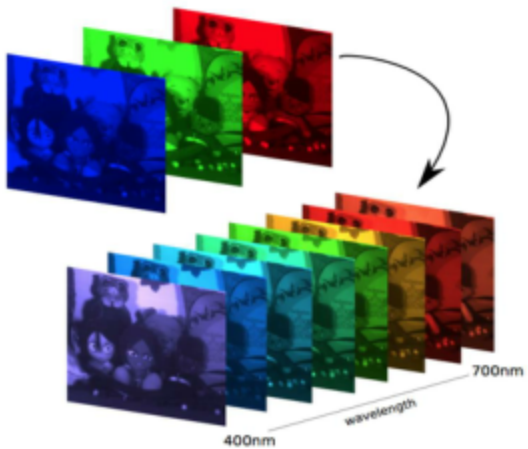


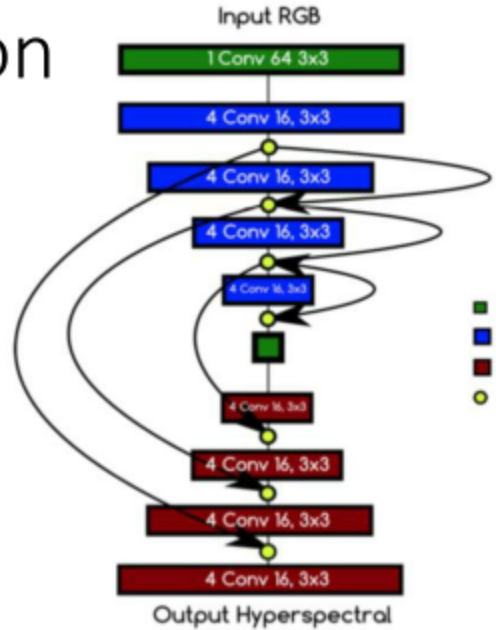


EcoVision PRS Photogrammetry Remote Sensing ETH zürich

Spectral Reconstruction

Blind spectral super-resolution

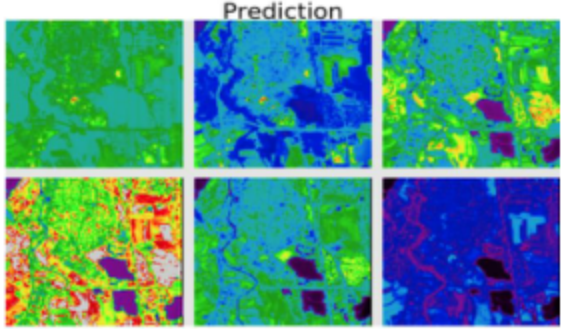




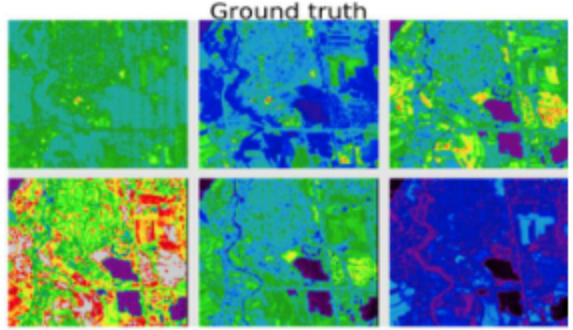
Convolution
Densenet + Max-Pooling
Densenet + sub-pixel Upsampling
Concatenate

S. Galliani, C. Lanaras, D. Marmanis, E. Baltsavias, K. Schindler: Learned Spectral Super-Resolution, Arxiv preprint 2017

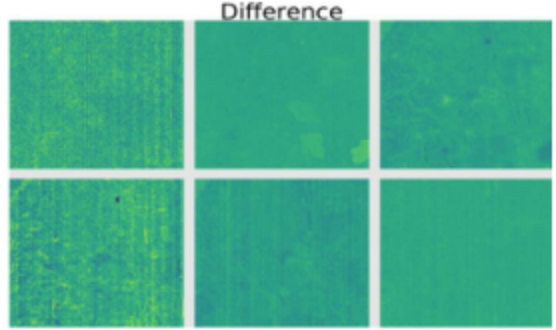
Prediction



Ground truth



Difference



Hyperion

EcoVision PRS Photogrammetry Remote Sensing ETH zürich

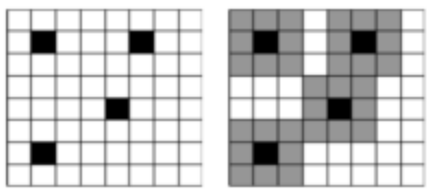
Point Cloud Processing

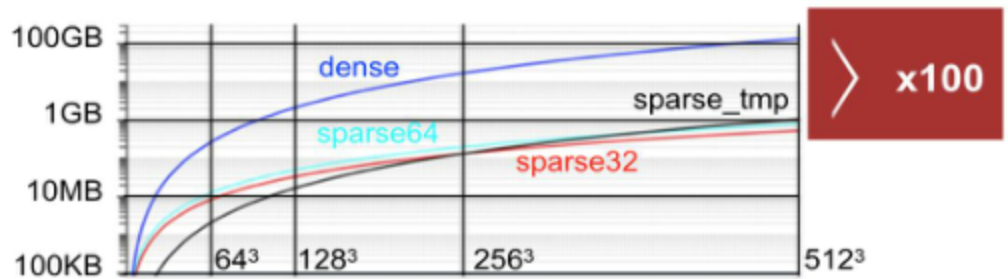
T. Hackel, M. Usvyatsov, S. Galliani, J.D. Wegner, K. Schindler: ILA-SCNN: Inference, Learning and Attention Mechanisms that Exploit and Preserve Sparsity in Convolutional Networks, Arxiv preprint, 2018.

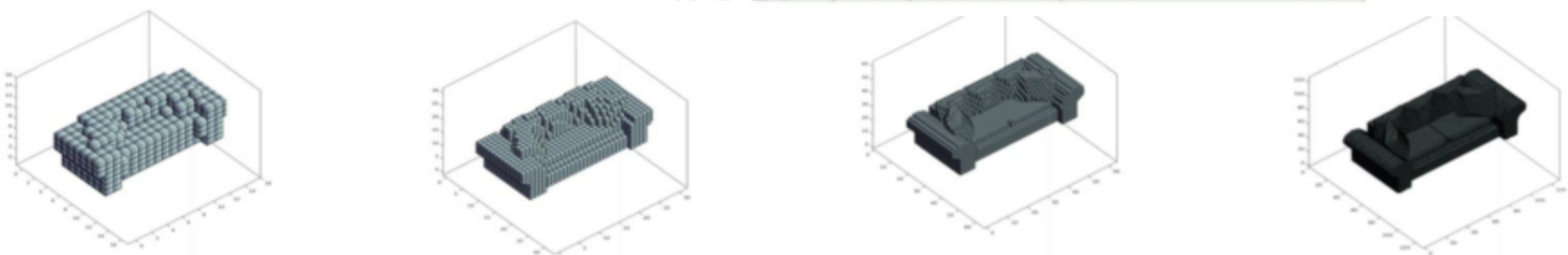
Bottleneck for deep learning is (GPU) memory
→ Exploit and preserve sparsity throughout the CNN

A gruelling programming exercise:
outsmarting your GPU (and the *TensorFlow* architects)

convolutions destroy sparsity





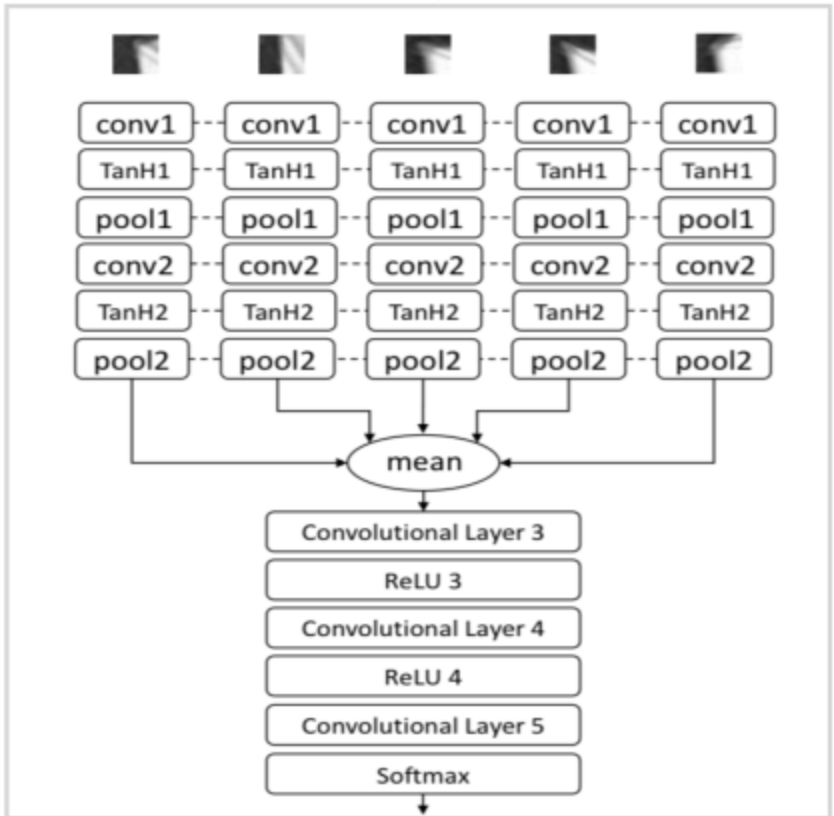


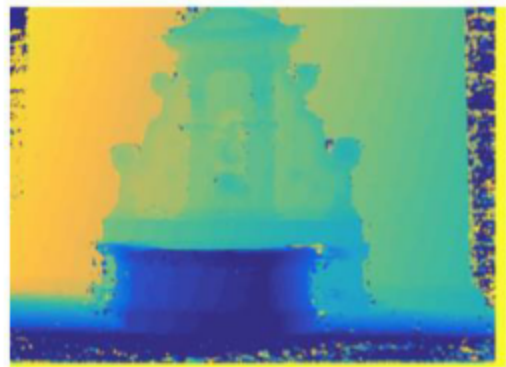
(a) $r_1 = 16^3, \rho_1 = 15.5\%$ (b) $r_2 = 32^3, \rho_2 = 9.1\%$ (c) $r_3 = 64^3, \rho_3 = 5.3\%$ (d) $r_4 = 128^3, \rho_4 = 2.7\%$

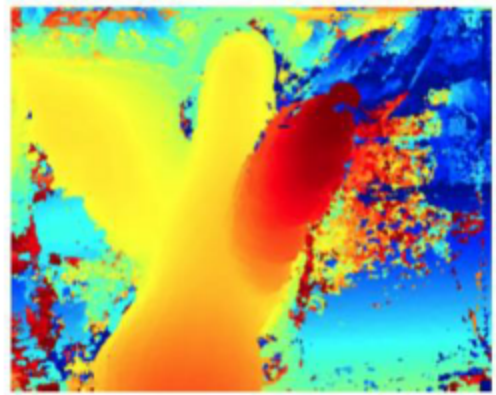
Surface Reconstruction

Hartmann W., Galliani, S., Havlena M., Van Gool, L., Schindler K.: Learned Multi-Patch Similarity, ICCV, 2017

Learned similarity measure across *multiple* images







Surface Reconstruction

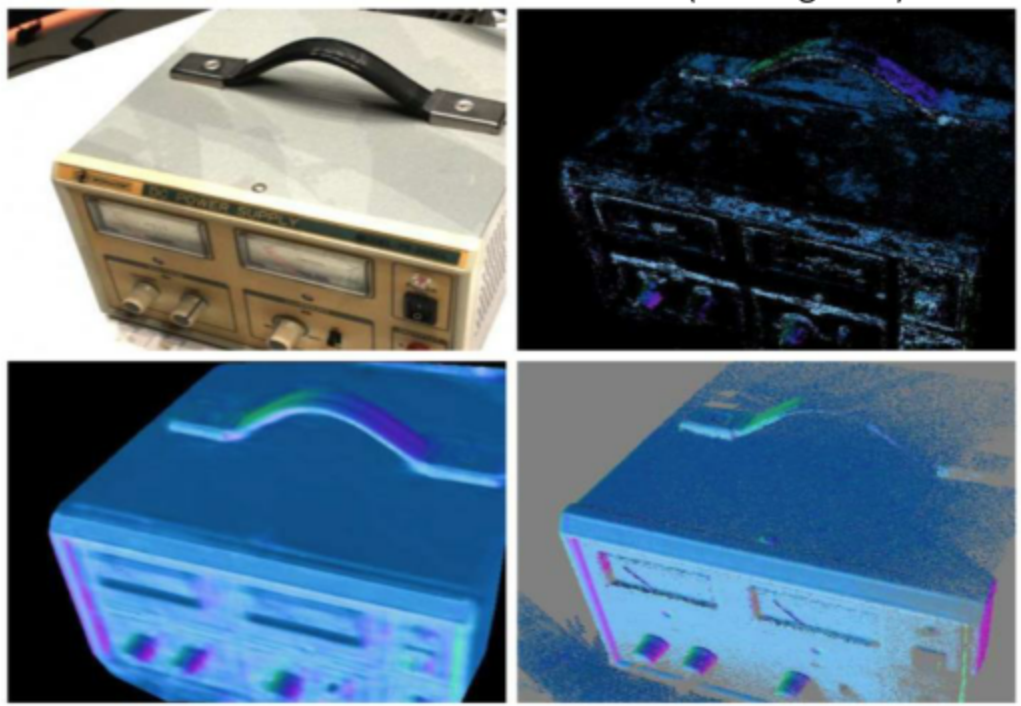
Learn regression from image appearance to normal vectors

Input image patch of size 16x16
Convolution with 16 kernels of dimension 5x5
Max pooling of size 2
Convolution with 50 kernels of dimension 5x5
Max pooling of size 2
Fully connected layer of size 512
Rectifier
Dropout with factor 0.5
Fully connected layer of size 2
Euclidean loss over angles

Silvano Galliani, Konrad Schindler: Just look at the image: viewpoint.specific surface normal prediction for improved multi-view reconstruction, CVPR, 2016.

image

points from MVS (training data)



estimated normals

ground truth

Ingredients for success

- ✓ Fully understand the problem you want to solve
- ✓ Collect as much data as you can get
- ✓ Get as much reference annotations as possible
- ✓ Never throw away your raw data!

…only using raw data unleashes the full potential of deep learning

- ✓ Rent cloud space or buy GPUs
- ✓ Always verify results using reference data!

Crowdsourcing platforms

CrowdFlower

clickworker.com

amazon mechanical turk™
Artificial Artificial Intelligence

Almost done…

…a slight warning before you get to euphoric:

- ✓ there is no magic, always carefully validate your results with manually collected reference data!
- ✓ even if results are excellent we may sometimes not understand causalities

The Hype Curve

jan.wegner@geod.baug.ethz.ch