Jonas Glombitza
jonas.glombitza@fau.de

Astroparticle School 2022
Obertrubach, Germany

Convolutional Neural Networks

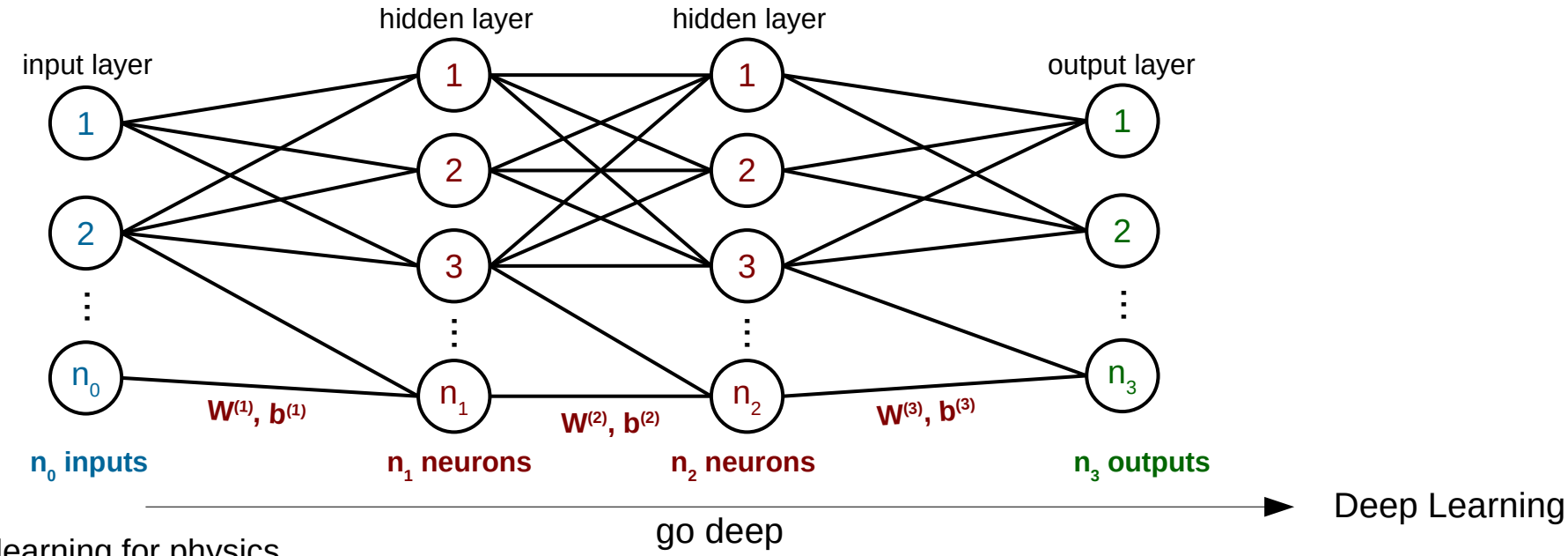
I. Processing image-like data
II. Incorporating symmetries into DNNs

https://github.com/DeepLearningForPhysicsResearchBook/deep-learning-physics

Neural Networks

Basic unit $\sigma(Wx + b)$ is called **node/neuron** (analogy to neuroscience)

- Strength of connections between neurons is specified by **weight matrix** W
- **Width:** number of neurons per layer
- **Depth:** number of layers holding weights (do not count input layer)

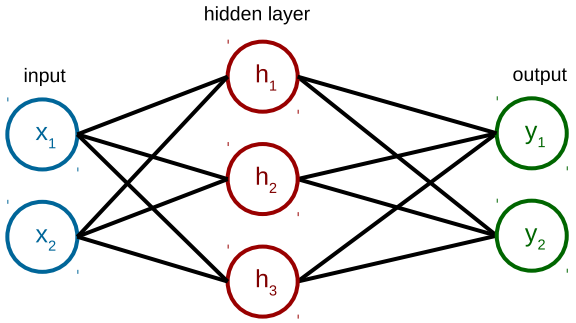


Recap: Neural Networks

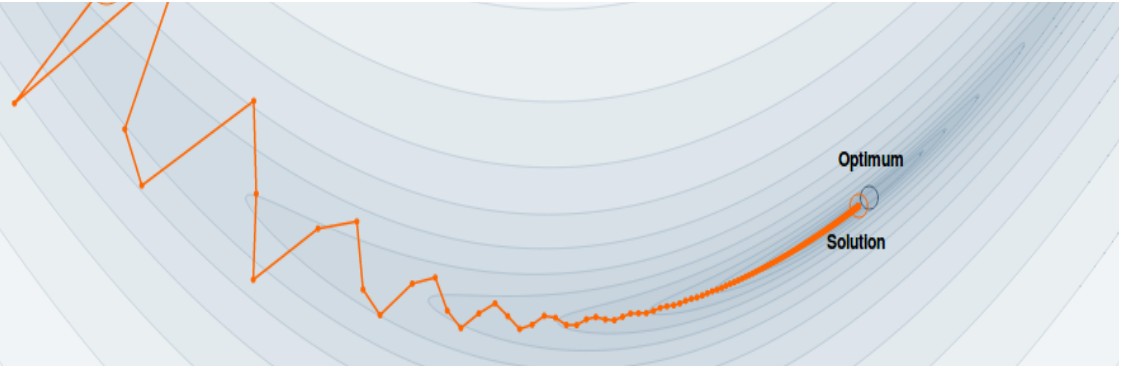
- Typical Machine Learning task
  - Labeled Data (x, y)
  - Model with adaptive weights $y_m(x, \theta)$
  - Objective $J(\theta)$
  - Optimization procedure

- Neural Network $y = \sigma(Wx + b)$
  - Matrix multiplication (adaptive superposition of features)
  - Add of bias
  - Nonlinear activation

- Deep Learning is form of representation learning
  - Stack multiple layers for increasing feature hierarchy



Recap: Optimization



Why Momentum Really Works, Distill

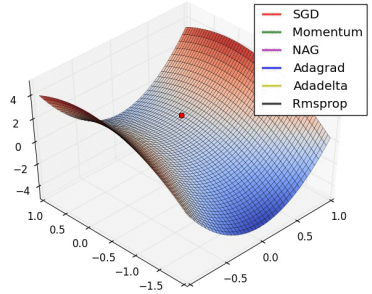
Mini-batch training: Use **stochastic gradient descent** algorithm (SGD)

Momentum: Use past gradients (velocity)

- Faster convergence by **damping oscillations** and increasing the step size for more informative gradients

Adaptive learning rate: Scaling using past gradients

- Use adaptive learning rate for each parameter

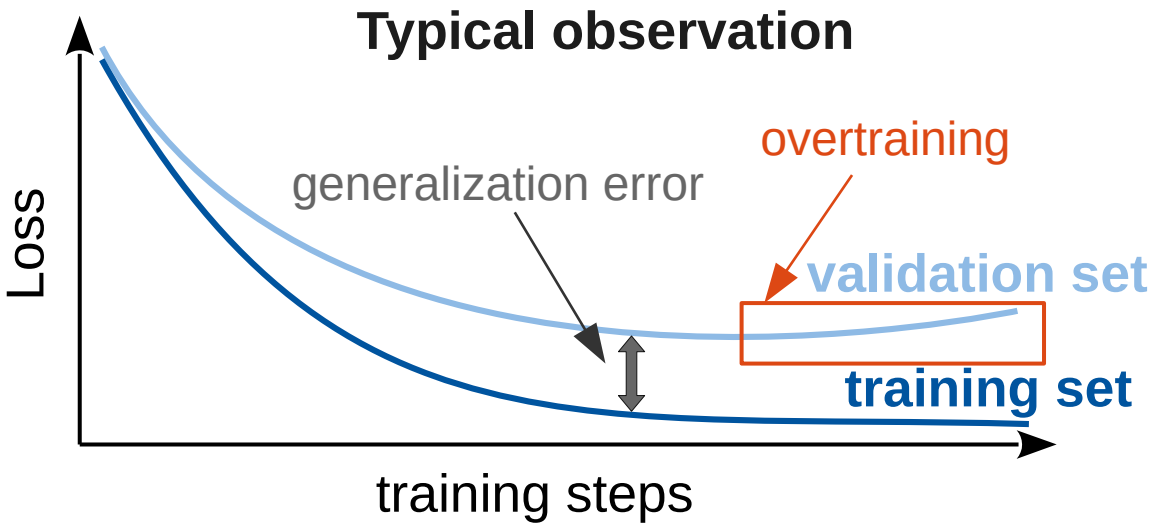


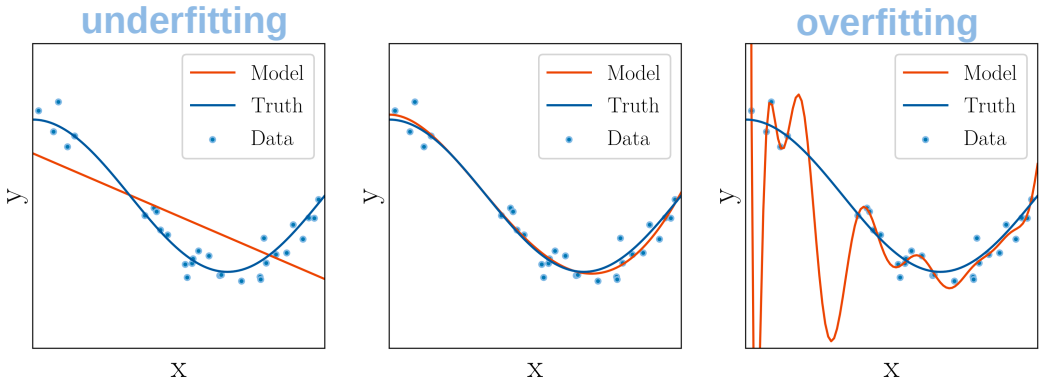
"Friends don't let friends use minibatches larger than 32" - *Yann LeCun*

Recap: Under- and Overtraining

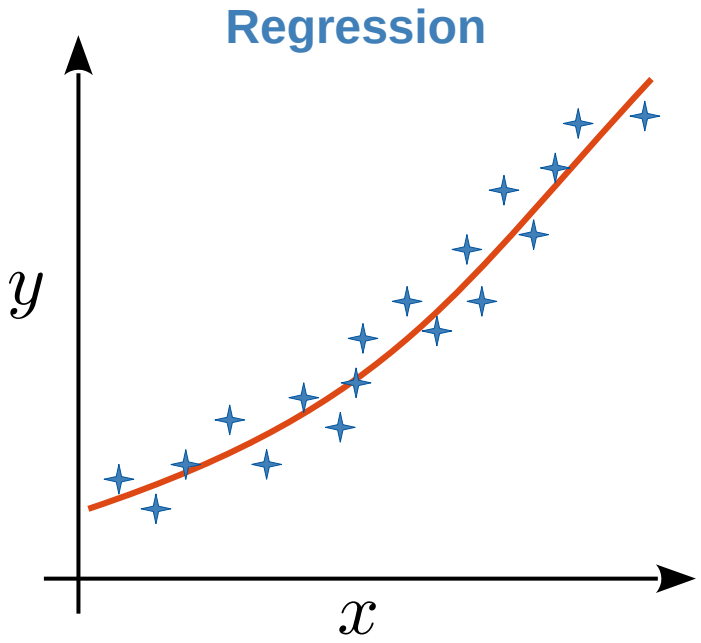
- Split data into 3 sets
  - training
  - test
  - validation
- What is a reasonably split?

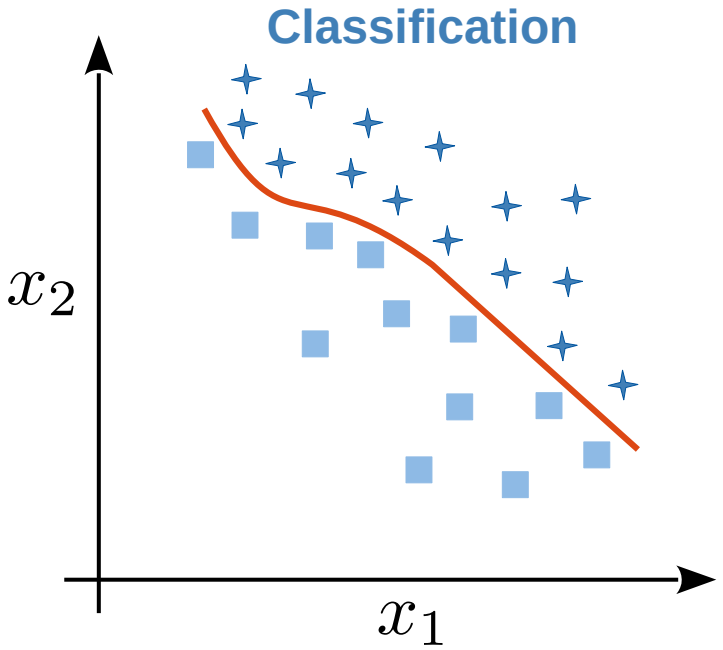
- During training monitor the loss separately for training and validation set
  - check for overtraining!
  - if loss stops decreasing
    - reduce learining rate
    - stop training





Machine Learning Tasks





- **Regression:** Predict continuous label y, e.g., particle energy
- **Classification:** Separate into different classes (cats, dogs, airplanes, …)

  Usually feature *different* <u>architectures</u> and <u>losses</u>

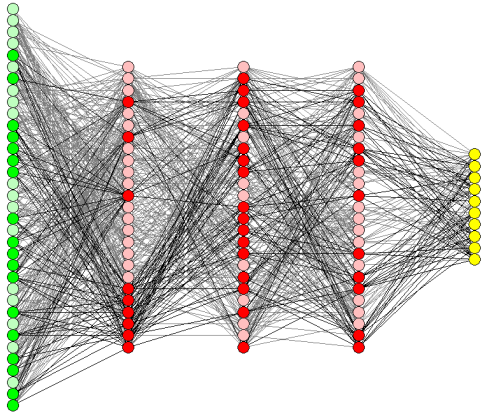
Training of a neural network

Regression

- activations are collected across the model
- try to get correct contribution of feature to final output value

→ yield correct amount at final node

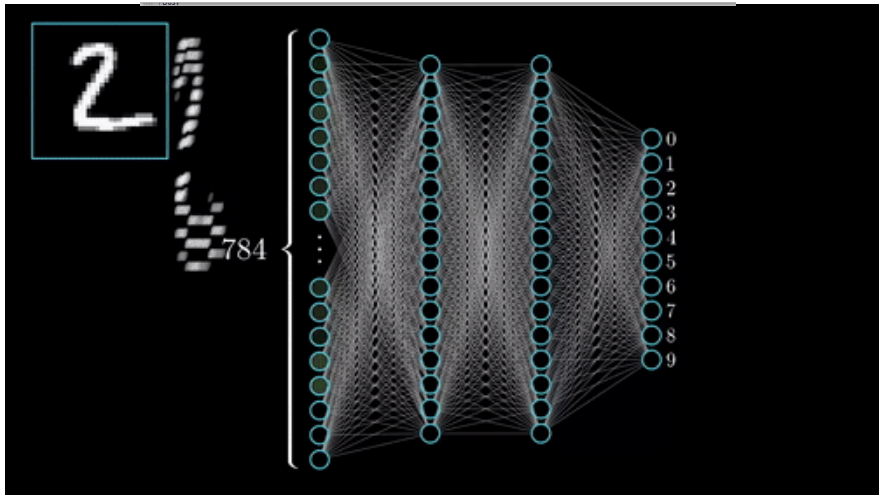
(no-rescaling – linear activation)



Classification

- activations of features are collected and pushed into the last nodes
- features describing class “2” are pushed into the class node

→ re-scaled into probabilities using softmax



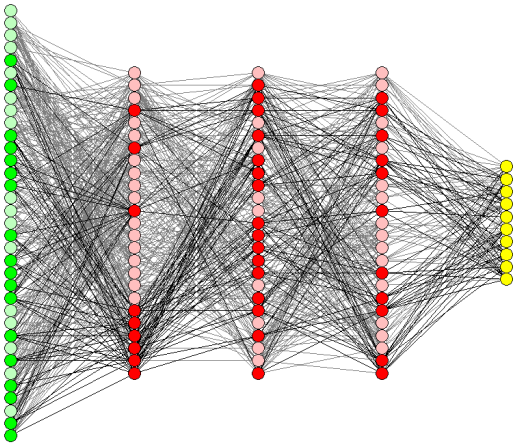
Training of a neural network

Regression

- activations are collected across the model
- try to get correct contribution of feature to final output value

→ yield correct amount at final node

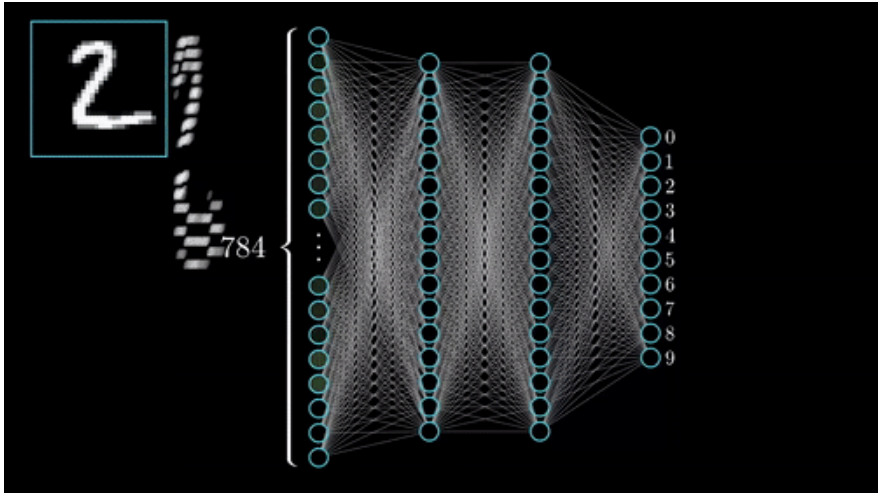
(no-rescaling – linear activation)



Classification

- activations of features are collected and pushed into the last nodes
- features describing class “2” are pushed into the class node

→ re-scaled into probabilities using softmax



Training of a neural network

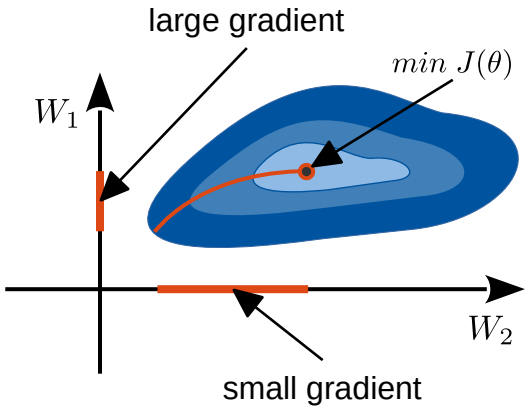
Regression and Classification

"Network learns underlying patterns / learns specific features"

underlying patterns: input data is correlated with training target (label)

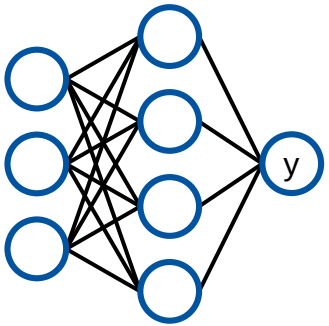
- when calculating loss → **forward pass**
  - networks is mapping (function) from input data to target (weights and biases)
- when estimating gradient → **backward pass**
  - if input to transformation is strongly correlated to target
  - ➔ will lead to a large gradient for adaptive parameter
  - ➔ will lead to a larger contribution in the next forward pass
  - ➔ larger contributions will affect the output most

➔ network is improving and *learns* to solve the task

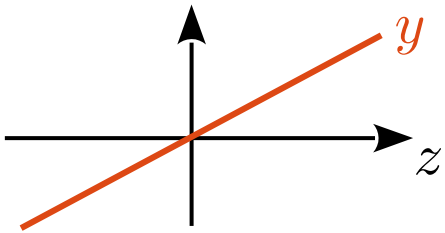


Classification vs. Regression

Regression



Linear

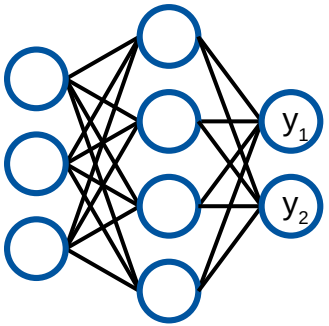


no activation function

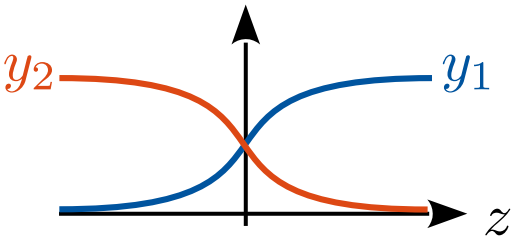
Minimize mean-squared-error

$$J(\theta) = \frac{1}{n}\sum_i [y_i - y_m(x_i)]^2$$

Classification



Softmax



$$y_j(z) = \frac{e^{z_j}}{\sum_i e^{z_i}}$$

Minimize cross entropy

$$J(\theta) = -\frac{1}{n}\sum_i y_i \log[y_m(x_i)]$$

Clarifying frequent misunderstandings

- **Use of activation functions -** layer without activation is usually meaningless
  - sigmoid **<u>only</u>** @ last layer in classification / regression @ last layer **<u>no</u>** activation
- **Universal approximation theorem is only a theoretic statement**
  - even such models exists → you have to find its design & **train** it → not easy!
- **Test and validation data are different**
  - validation: tune your DNN, e.g. train 10 DNNs & compare, monitor overtraining
  - test: check after you decide for one of the 10 models → ONCE!
- **Training networks is not random** → extract features out of patterns in data
  - retraining gives slightly different DNN → its feature sensitive to same patterns!
- **DNNs are not the holy grail** → simple fits can outperform DNNs
  - lots of data needed, challenge has to be complex and multi-dimensional

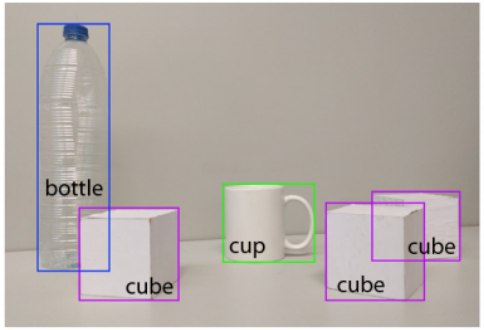
Natural Images

Automate task for humans, very challenging for machine learning models:

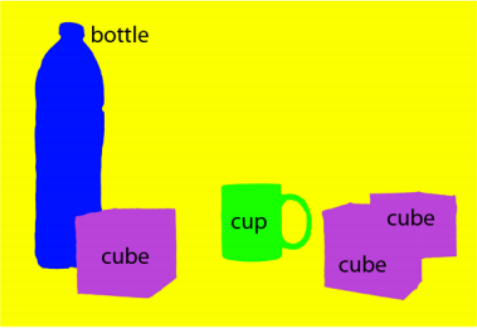
- High dimensional input (up to millions of pixels)
- Many possible classes depending on task
- Multiple variations
  - Viewing angle, light conditions, deformation, object variations, occlusions….

Computer Vision Tasks

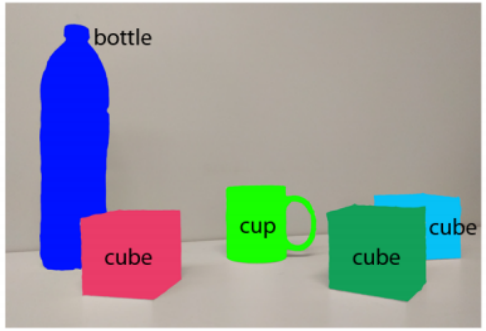
(a) Image classification



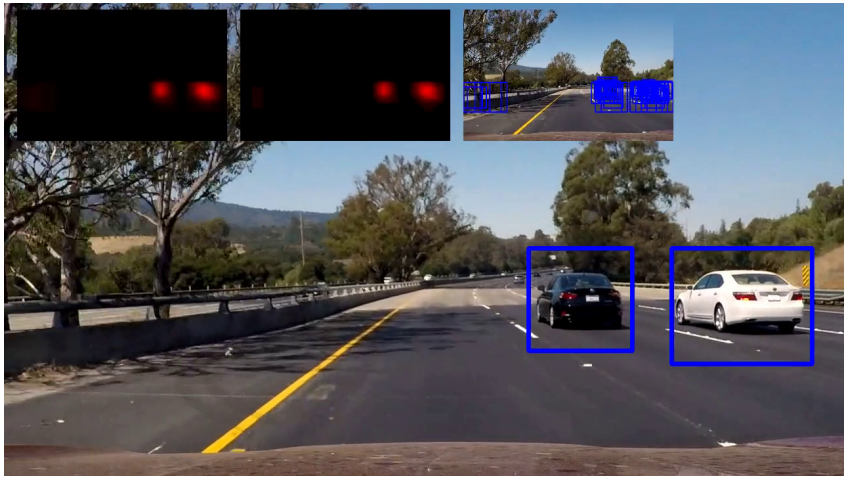
(b) Object localization



(c) Semantic segmentation

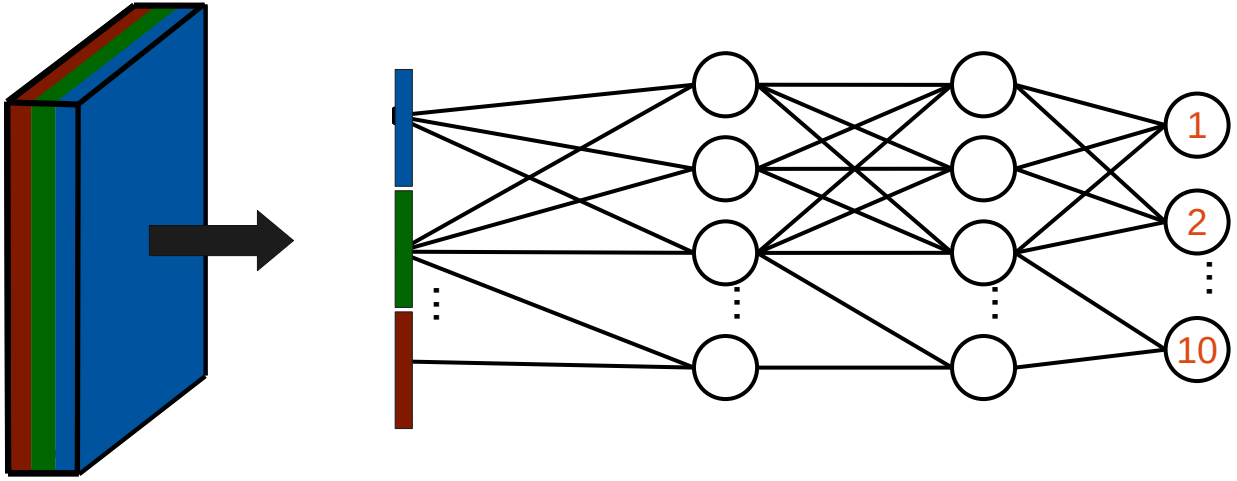


(d) Instance segmentation

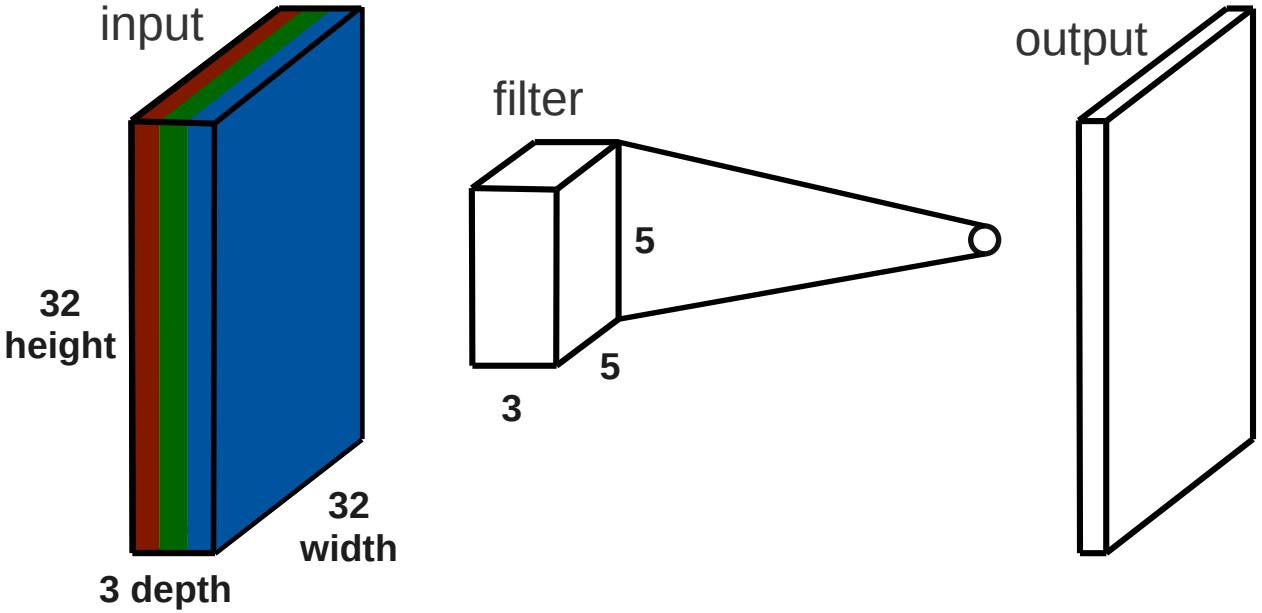


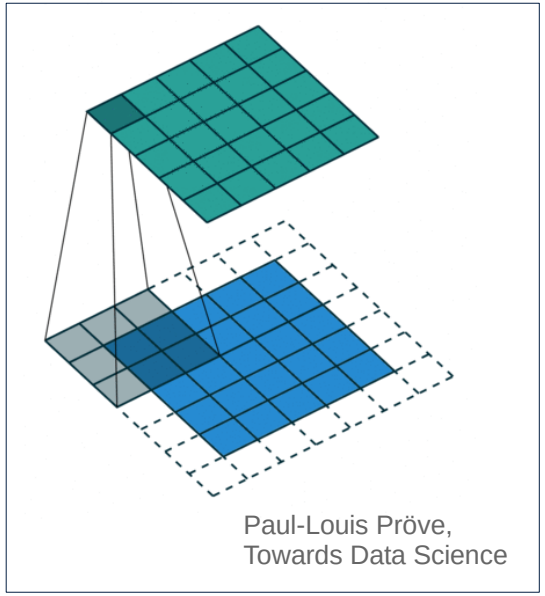
Fully Connected Network

- Input layer: Flatten image to 32 x 32 x 3 = 3072 vector
- Fully connected: every pixel connected with each other
- ✗ Huge number of adaptive parameters per layer
- ✗ No use of translational variance
- ✗ No prior on local correlations



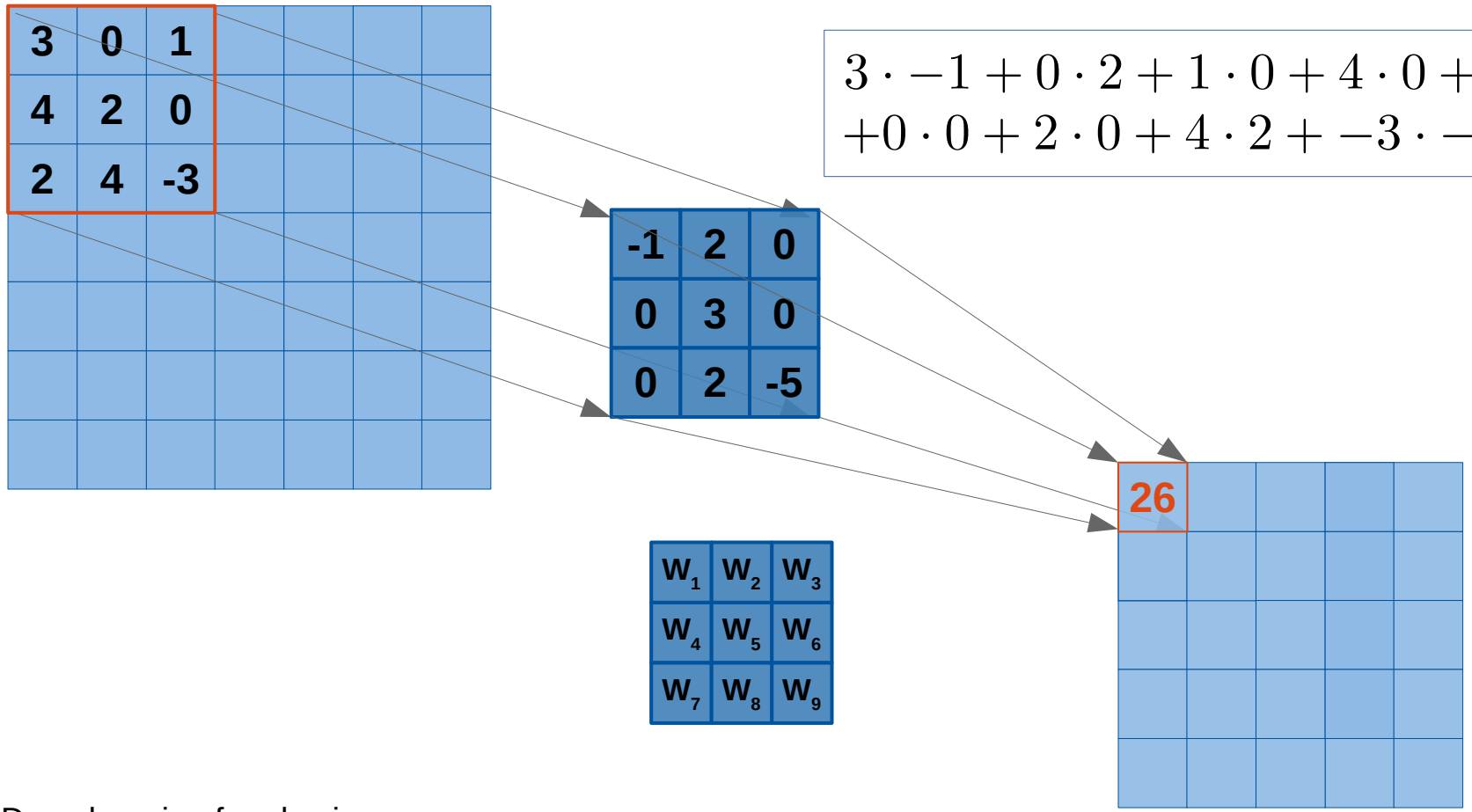
2D Convolutional Neural Networks





- Consider input volume (width x height x depth), e.g., 3 color channels
- Use convolutional filter with smaller width and height but same depth
- Slide filter over the entire volume and calculate linear transformation to get one output value for each position

Convolutional Operation



$$3 \cdot -1 + 0 \cdot 2 + 1 \cdot 0 + 4 \cdot 0 + 2 \cdot 3$$
$$+0 \cdot 0 + 2 \cdot 0 + 4 \cdot 2 + -3 \cdot -5 = 26$$

Convolutional filters

hand-designed filters

Edge

-1	-1	-1
-1	8	-1
-1	-1	-1

Diagonal edge

-1	-1	2
-1	2	-1
2	-1	-1

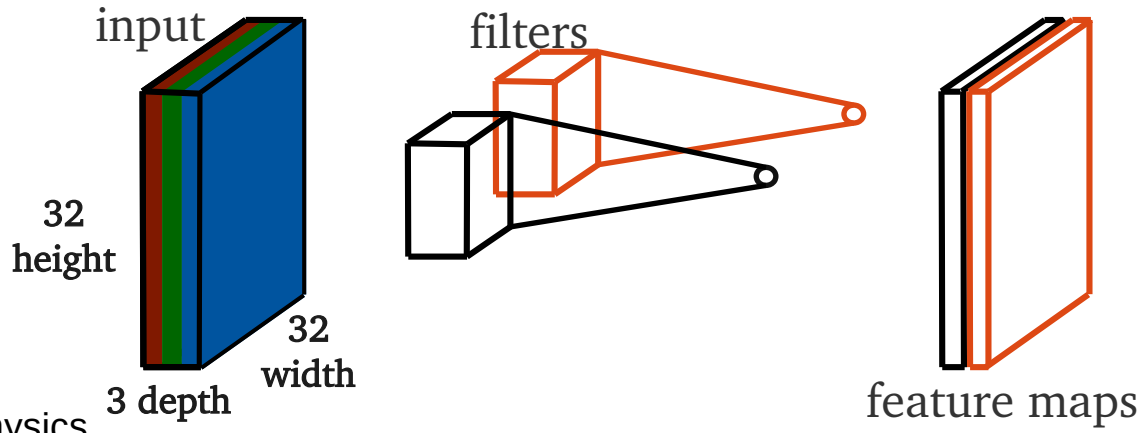
deep learning

Convolutional Networks

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

adaptive parameters

- scan input image for the presence of specific feature using **filters**
- use multiple filters and stack the results as **feature maps** (depth-wise stacking)

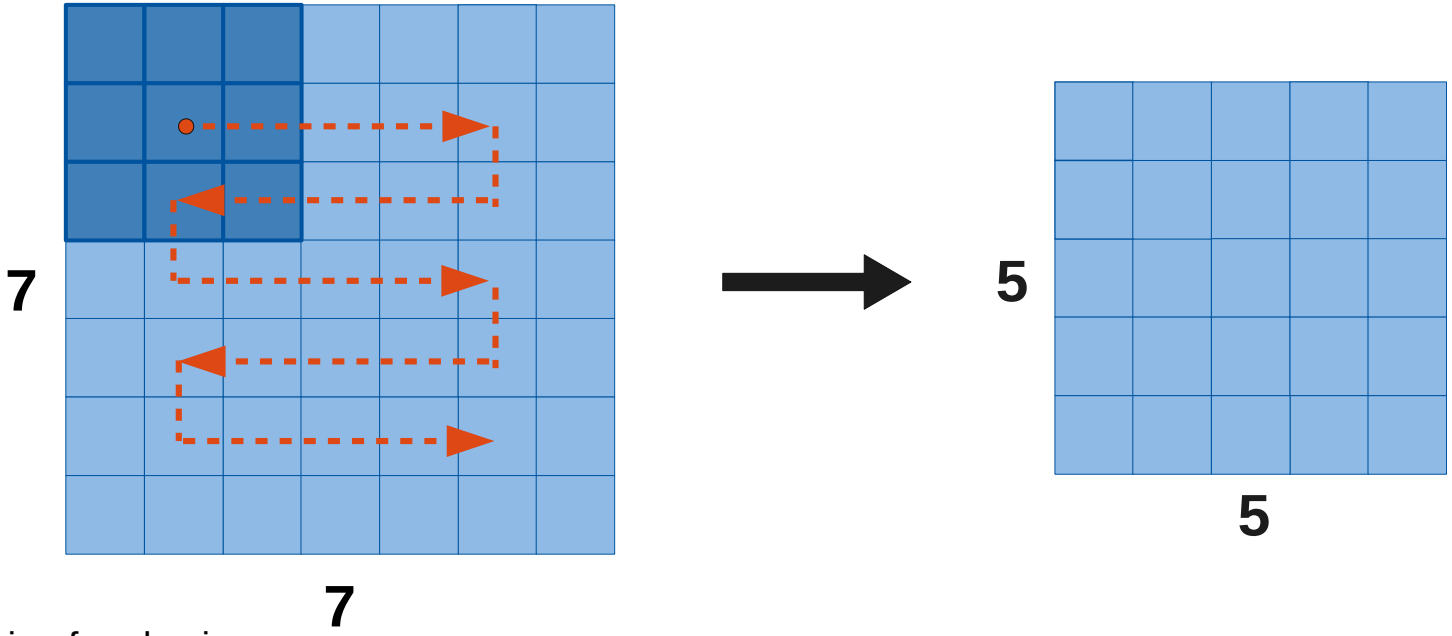


Spatial Output Size

Standard convolution reduces the output size due to extent of the filter

- Sets upper bound to the number of convolutional layers

- **Example:** Convolution with 3 x 3 filter

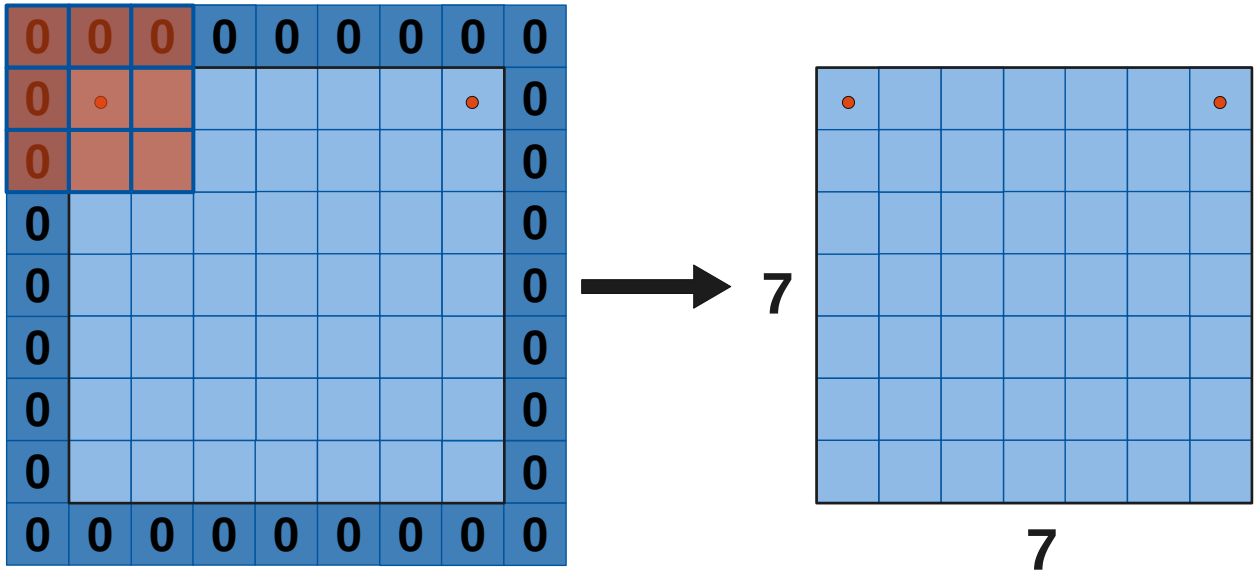


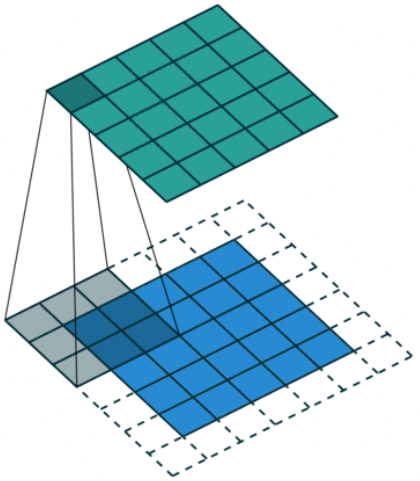
Padding

Add zeros around image borders to conserve the spatial extent of the input

- Prevents fast shrinking of the network input

- **Example:** Convolution with 3 x 3 filter and padding





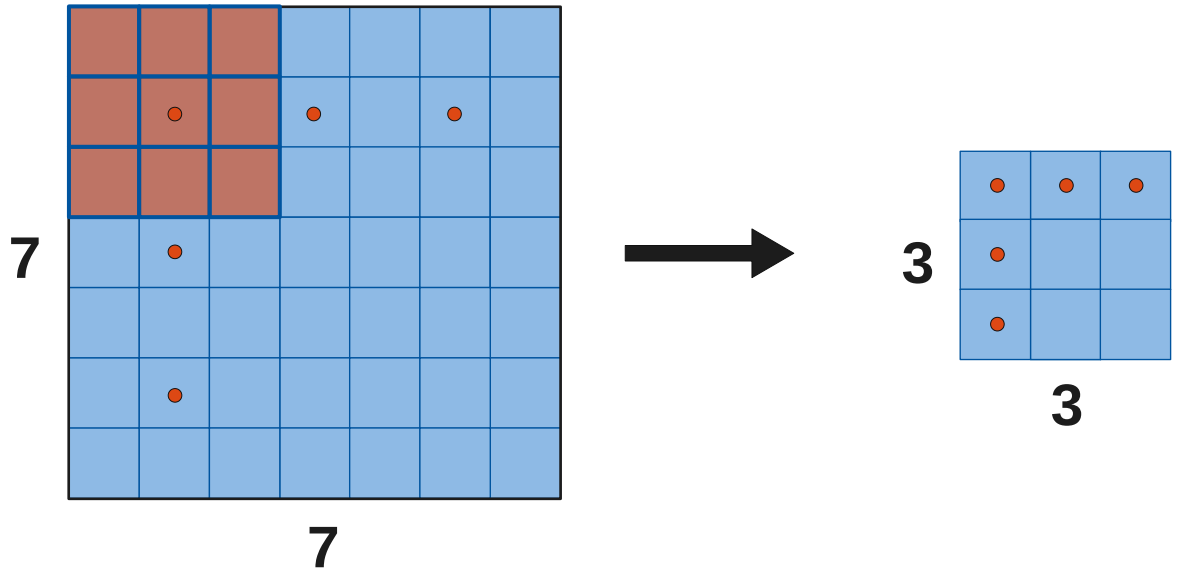
Paul-Louis Pröve,
Towards Data Science

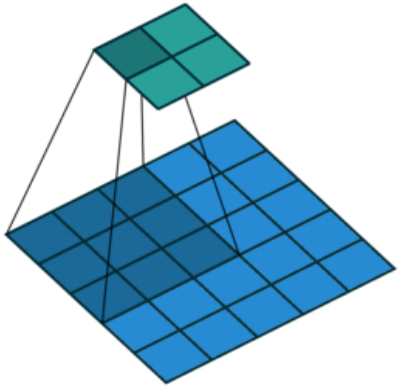
Striding

Using a larger stride when sliding over the input, reduces the output size

- Useful for switching to smaller image sizes / larger scales

- **Example:** Convolution with 3 x 3 filter and stride of 2





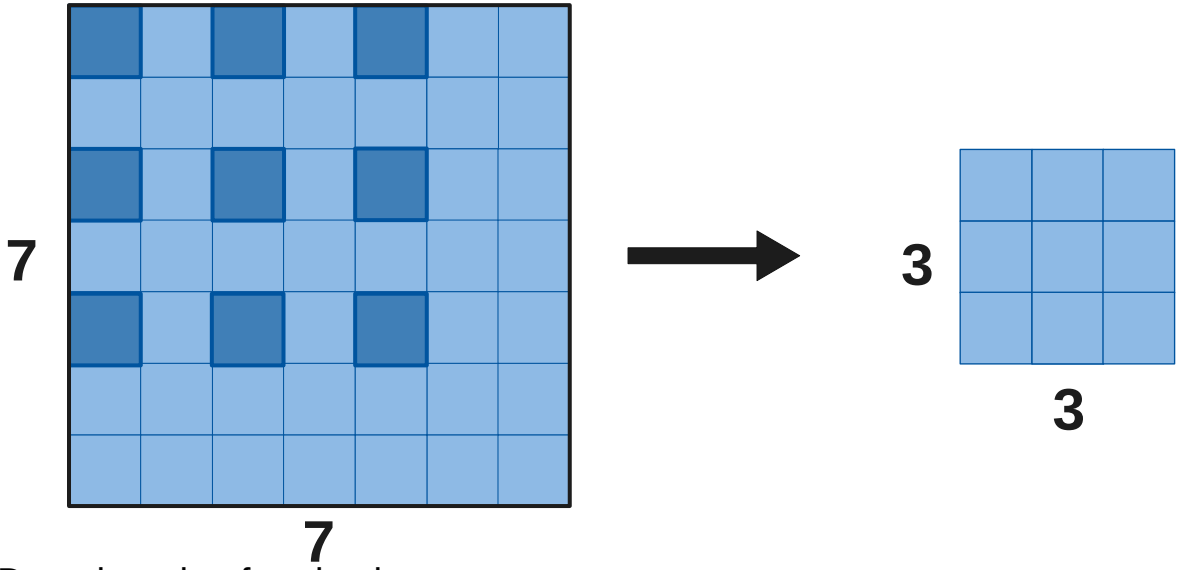
Paul-Louis Pröve,
Towards Data Science

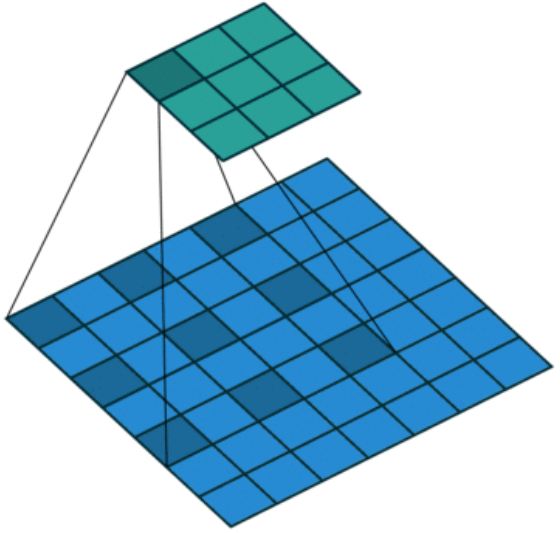
Dilating

Dilation leaves holes in where the filter is applied (also called **atrous convolution**)

- Useful for aggressively merging spatial information in large images
- Allows for a large field of view

- **Example:** Convolution with 3 x 3 filter and dilation 1





Paul-Louis Pröve,
Towards Data Science

Pooling

Sub-sample the input to reduce the output size

- Used to merge semantically similar features
- Make network invariant to small translations or perturbations

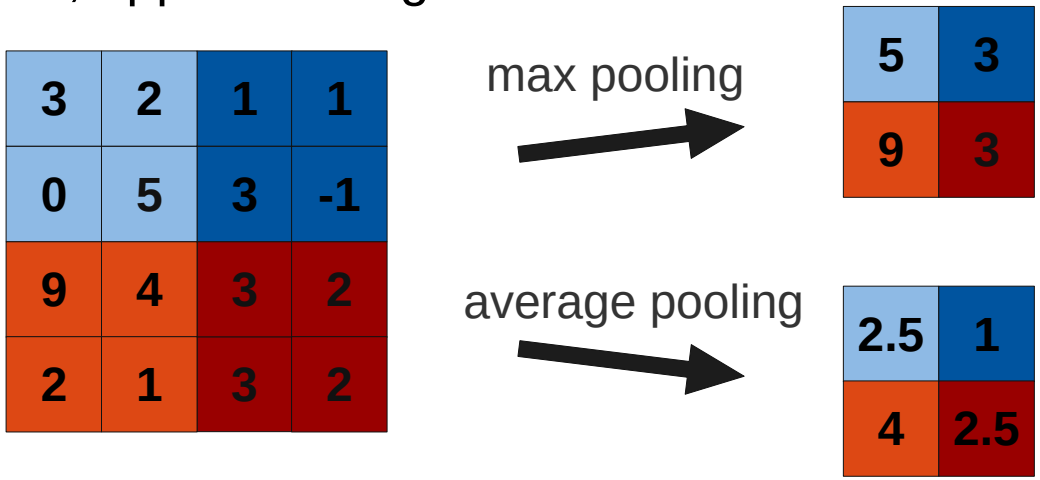
Average pooling: Take the mean of each patch → for some regressions preferable

Max pooling: Take the maximum of each patch

→ in practice often better performance, applies stronger constraint

- **Typical Pooling:**
  Pooling using 2 x 2 patches and a stride of 2

- **Overlapping Pooling:**
  3 x 3 patches with stride of 2

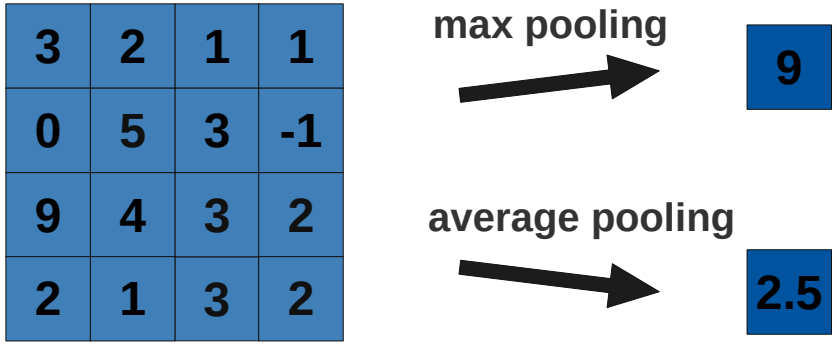


Global Pooling Operation

- Take maximum/average over complete image → usually second last layer
- Replace fully connected layers
  - Saves parameters in later layers of the models → prevent overfitting
- Can be seen as regularizer
  - Fully connected transformation matrix with diagonal shape
- Enforcing correspondences between feature maps and categories
- Allows object detection in the input space

"The pooling operation used in convolutional neural networks is a big mistake, and the fact that it works so well is a disaster"

- Geoffrey Hinton

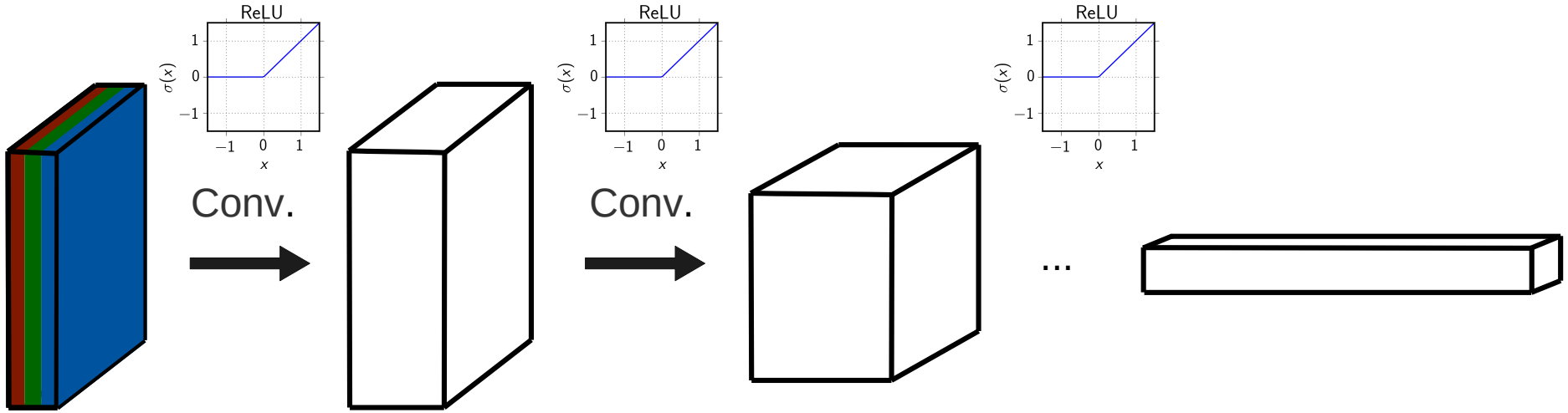


5 minutes break

Convolutional Pyramid

ConvNet architectures usually have a pyramidal shape. For deeper layers:

- Increasing of feature space
- Decreasing of spatial extent

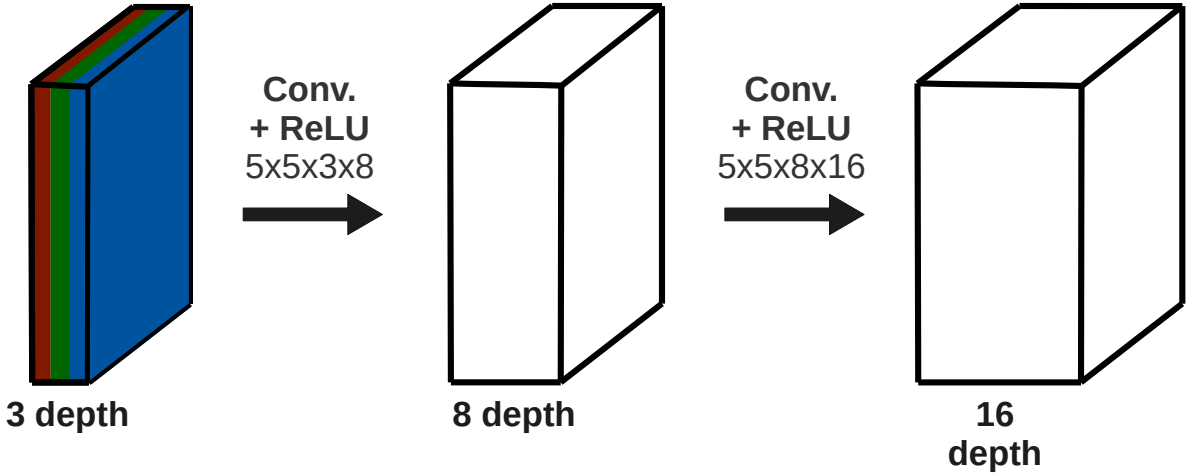


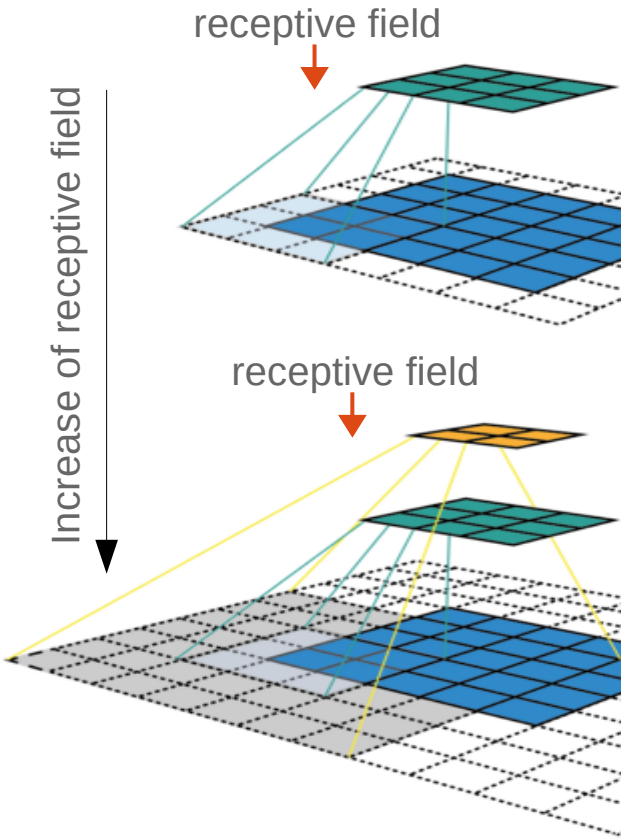
- Spatial information is converted to representational features with increasing hierarchy

2D Convolutional Operation

Stack multiple convolutional layers + activations

- Each convolution acts on feature map of previous layer
- Increasing feature hierarchy
- Increasing of receptive field



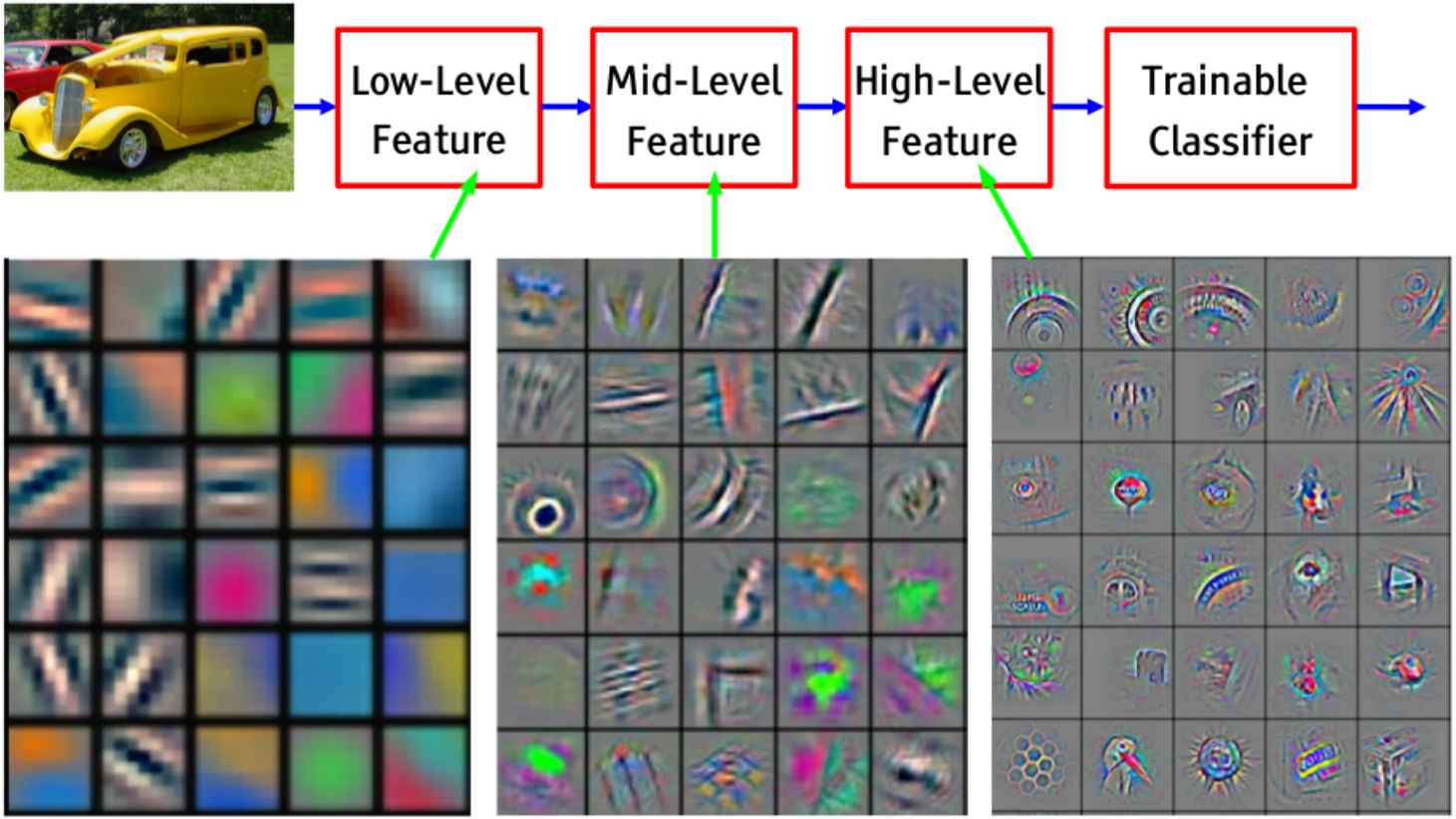


Visualization of CNNs – 5 mins

- Inspect the CNN → is the network translational invariant or equivariant
  - invariant: translation of the image yield exactly the same output?
  - or only equivariant? (DNN can reconstruct position of a pattern)

https://poloclub.github.io/cnn-explainer/

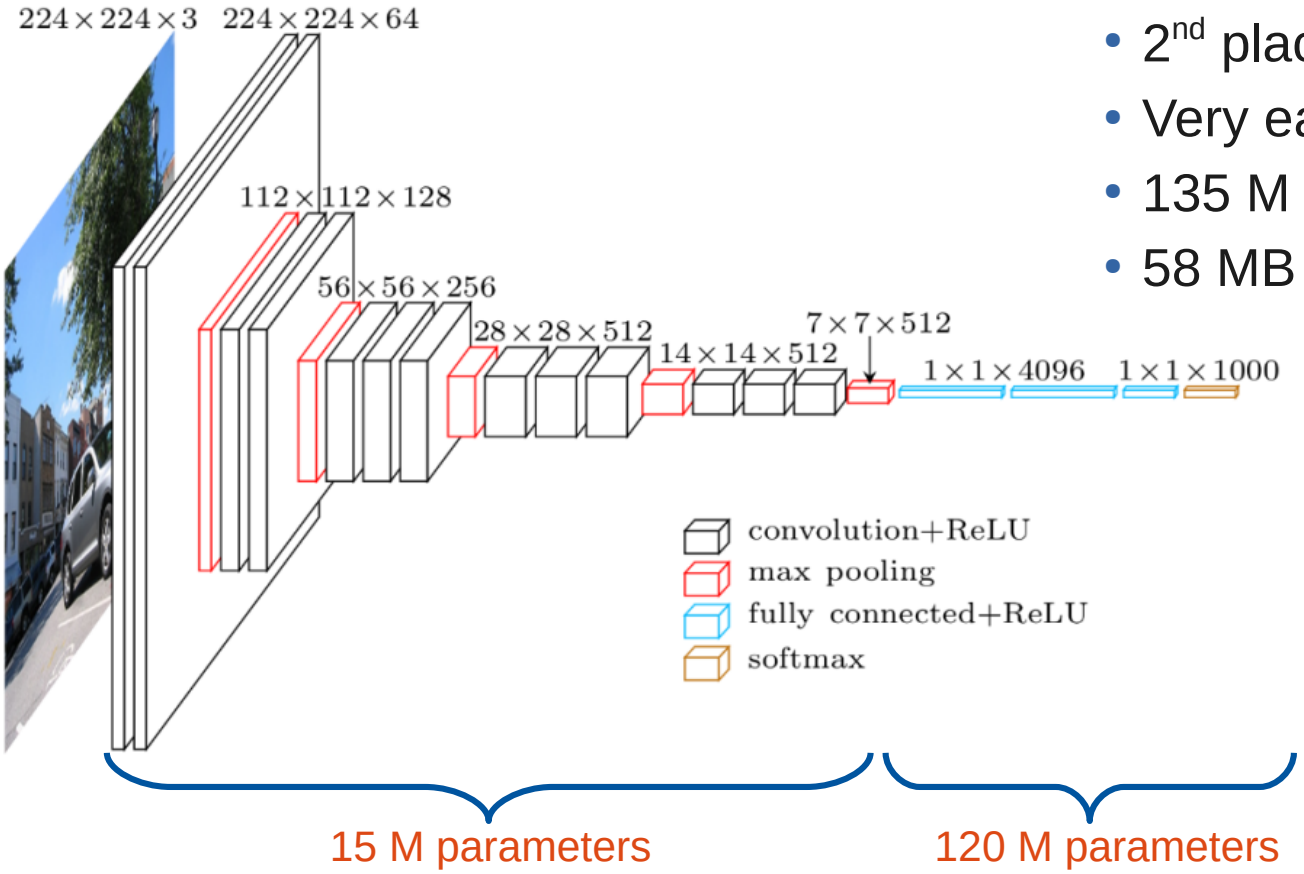
Feature Hierarchy



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

https://arxiv.org/abs/1311.2901

Example Architecture



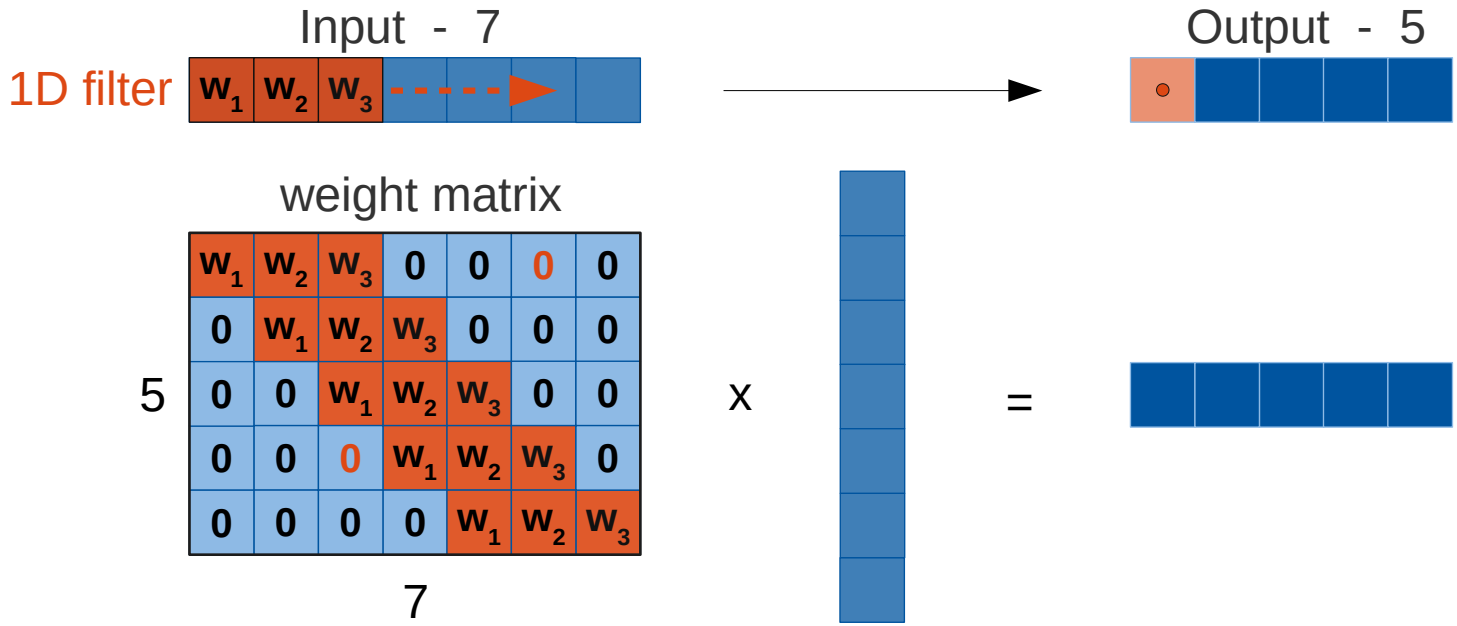
VGGNet 16

- 2nd place ILSVRC2014
- Very easy structure
- 135 M parameters → 530 MB
- 58 MB memory for activations

Simonyan, Zissermann
https://arxiv.org/abs/1409.1556

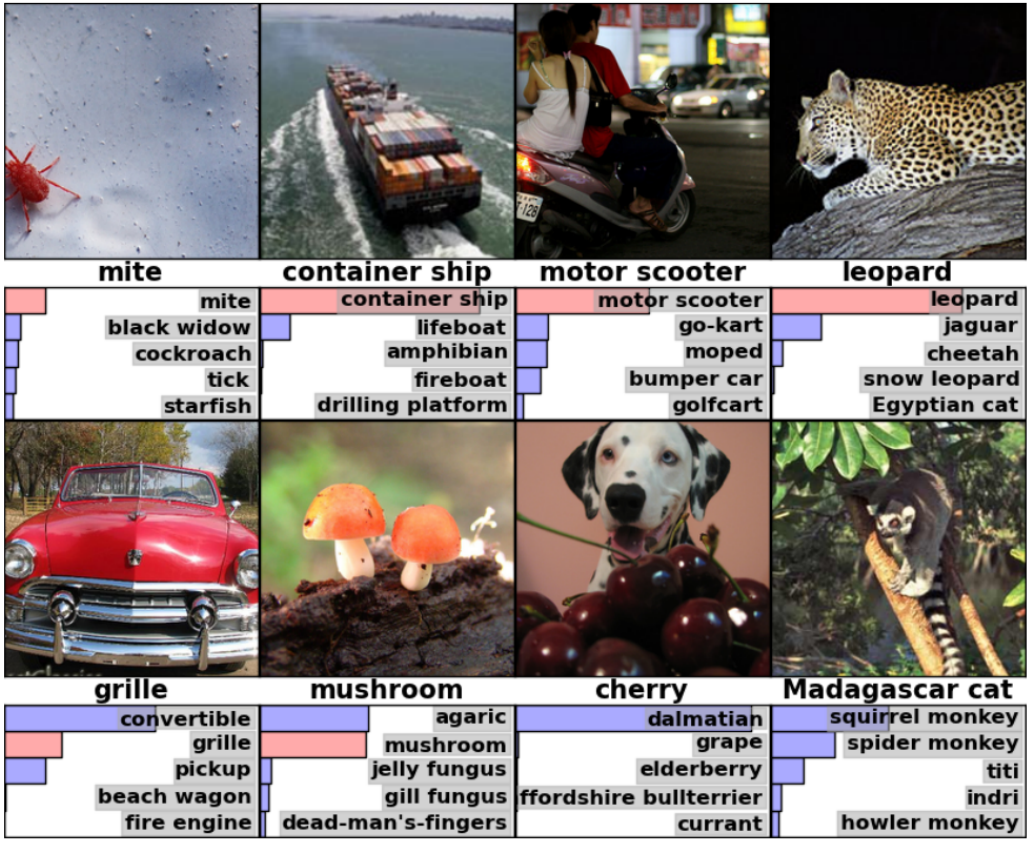
Convolutional Operation

- Fully connected layers are special case of convolutional layers

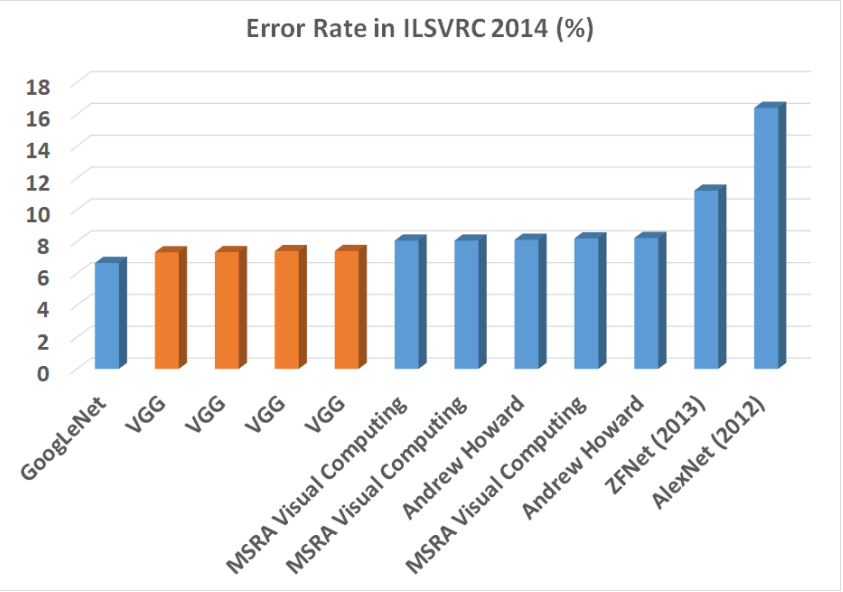


- Parameters greatly reduced due to **sparsity** and **weight sharing**
  - Strong prior on **local correlation** and **translational invariance**

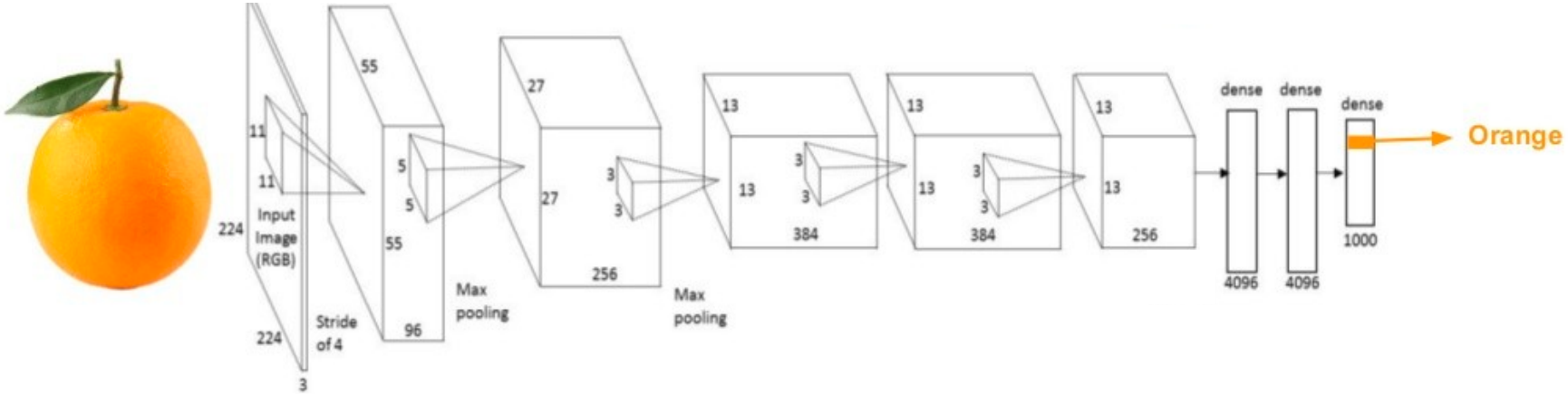
Results on ImageNet



AlexNet (2010) - http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

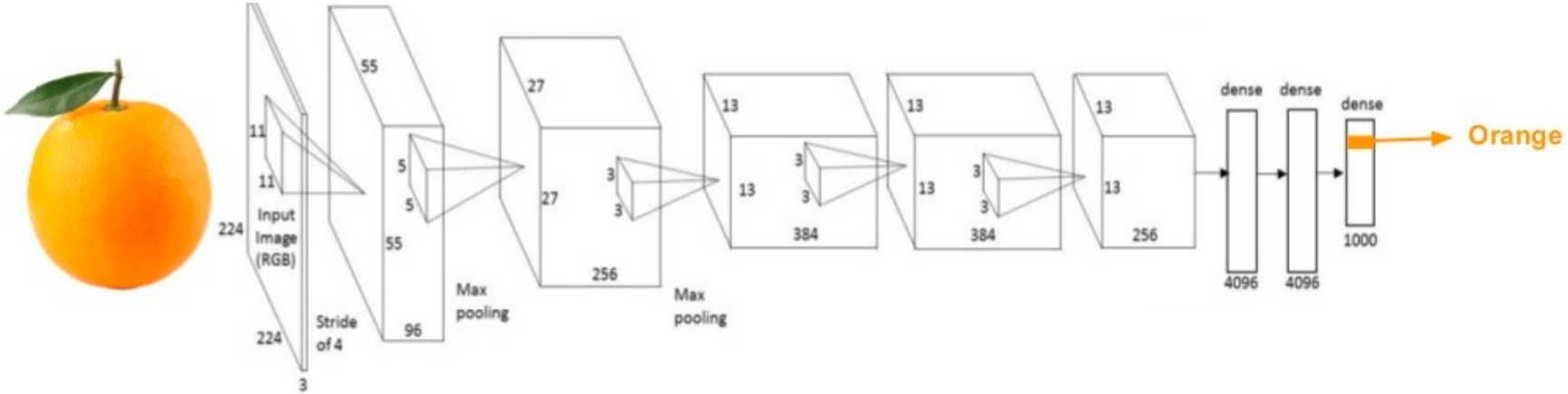


Dropout in CNNs



Where we have to apply dropout?

Dropout in CNNs



Where we have to apply dropout?

- Convolutional networks are less sensitive to over-fitting due to weight sharing
- Spatial Dropout: drop entire feature maps
- Use Dropout before MaxPooling layer
  - Make use of subdominant features

Clarifying frequent misunderstandings

- The **filters are no pre-defined** by the user → just width and depth and number
  - filters are adapted / learned by the CNN during training

- **Number of filters define number of new feature maps**
  - ten 3x3 filter applied to RGB image → 10 feature maps

- **Filter has the depth of the input image** (e.g. depth 3 for RGB images)
  - two 3x3 filter applied to RGB image → 2 feature maps, i.e. 2 channels
    → number of adaptive parameters = 3 x 3 x 3 * 2 + 2 = 56

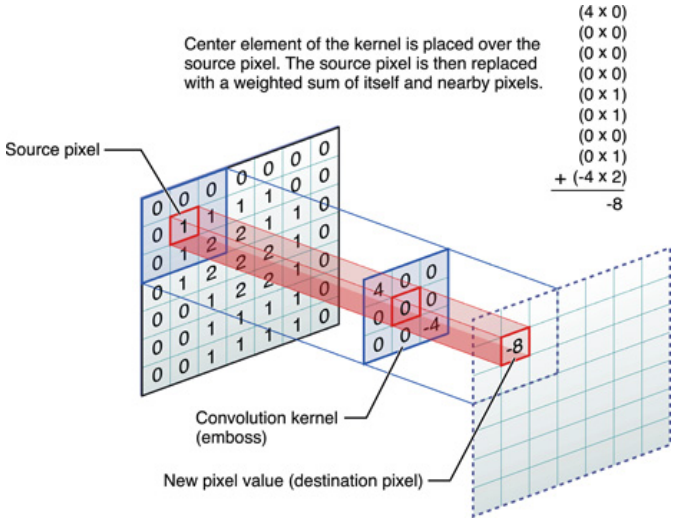
- **After each convolutional operation an activation is applied!** (usually)

- **CNN part is followed by a fully-connected part** (in most cases)
  → output is reshaped (flattened) to a vector → apply vanilla NN layer

Summary

- 2D Convolution acts on 3D input (width x height x depth)
- Slide small filter over input and make linear transformation (dot product + bias)
- Hyperparameter:
  - Size of filter, typically (1 x 1), (3 x 3), (5 x 5) or (7 x 7)
  - Number of filters (feature maps)
  - **Padding** (maintain spatial extent)
  - **Striding** or **pooling** (reduce spatial extent)

- Reduction of parameters using symmetry in data:
  - Prior on **local correlations** (use small filters)
  - **Translational invariance** (weight sharing)



References & Further Reading

- M. Erdmann, J. Glombitza, G. Kasieczka, U. Klemradt, Deep Learning for Physics Research, World Scientific, 2021, www.deeplearningphysics.org/
- I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, Chapter 7 / 8 / 9, MIT Press, 2016, www.deeplearningbook.org
- Xu et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, arXiv:1502.03044
- Y. LeCun, Y. Bengio, G. Hinton: Deep Learning, Nature 521, pages 436–444
- K. Simonyan, A. Zissermann: Very Deep Convolutional Networks for Large-Scale Image Recognition - ArXiv 1409.1556
- Toy Simulation: M. Erdmann, J. Glombitza, D. Walz, Astroparticle Physics 97, 46-53

Tryout Deep Learning Yourself!

Find many physics examples at:

http://www.deeplearningphysics.org/

For example:

- CNNs, RNNs, GCNs
- GANs and WGANs
- Anomaly detection, Denosing AEs
- Visualization & introspection and more