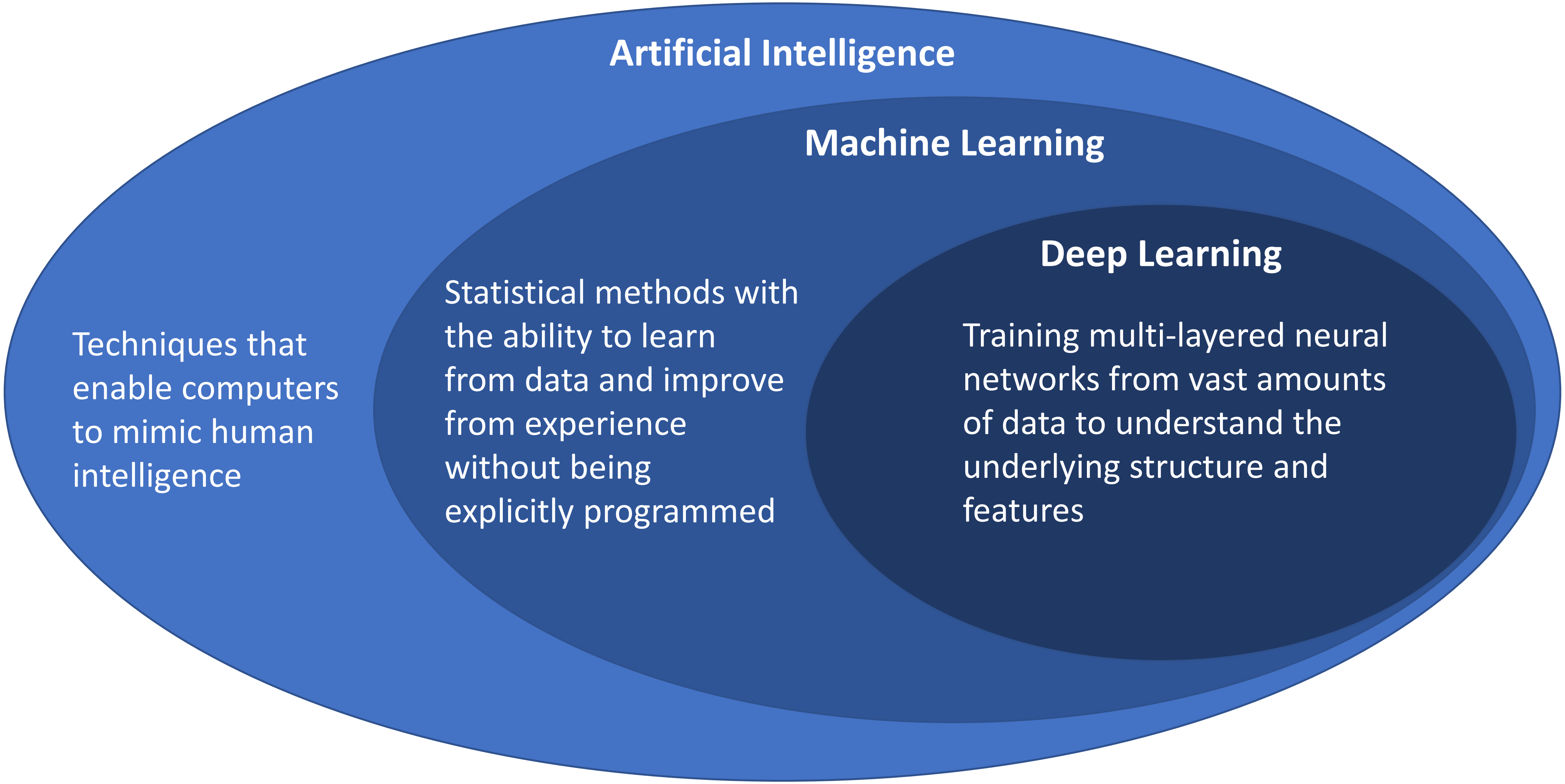
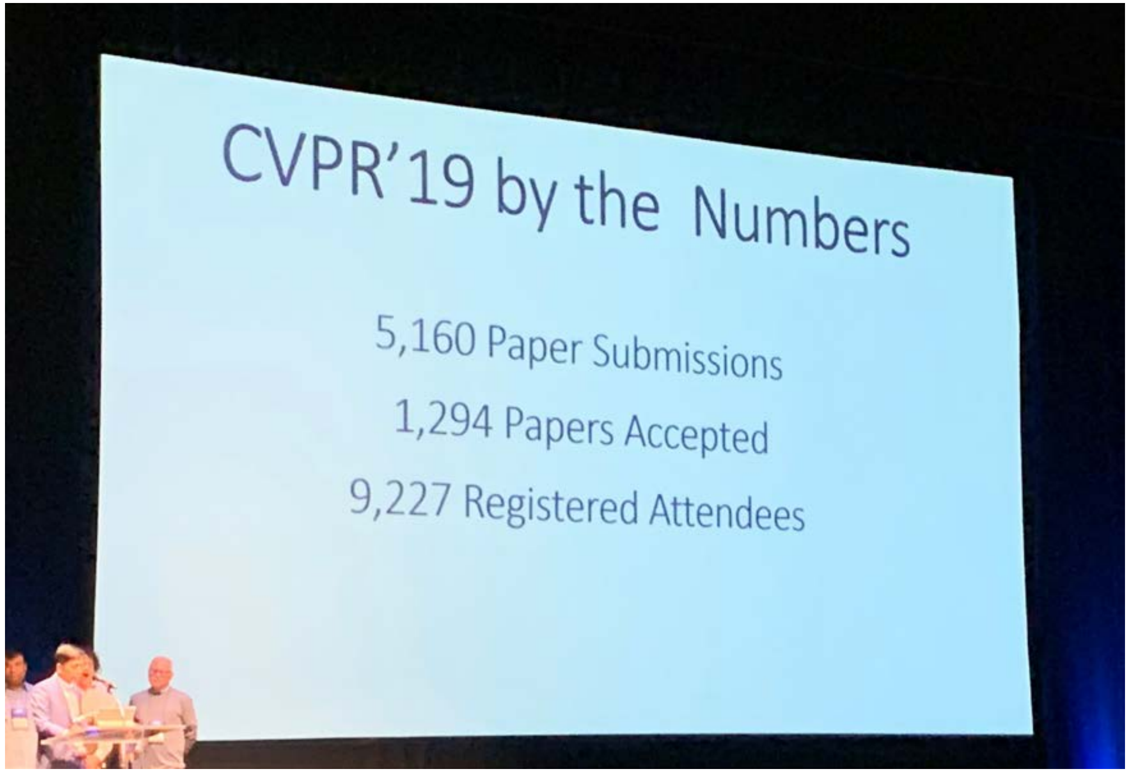
Introduction to Deep Learning

Prof. Dr. Martin Kada

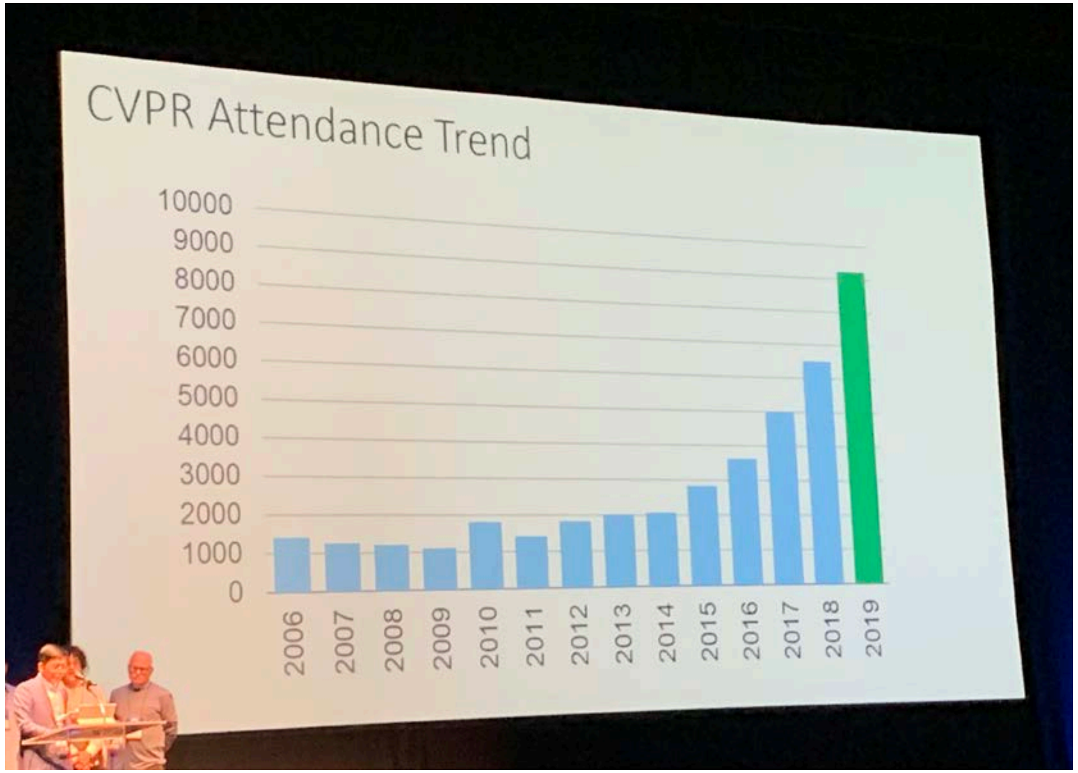
Artificial Intelligence (AI)



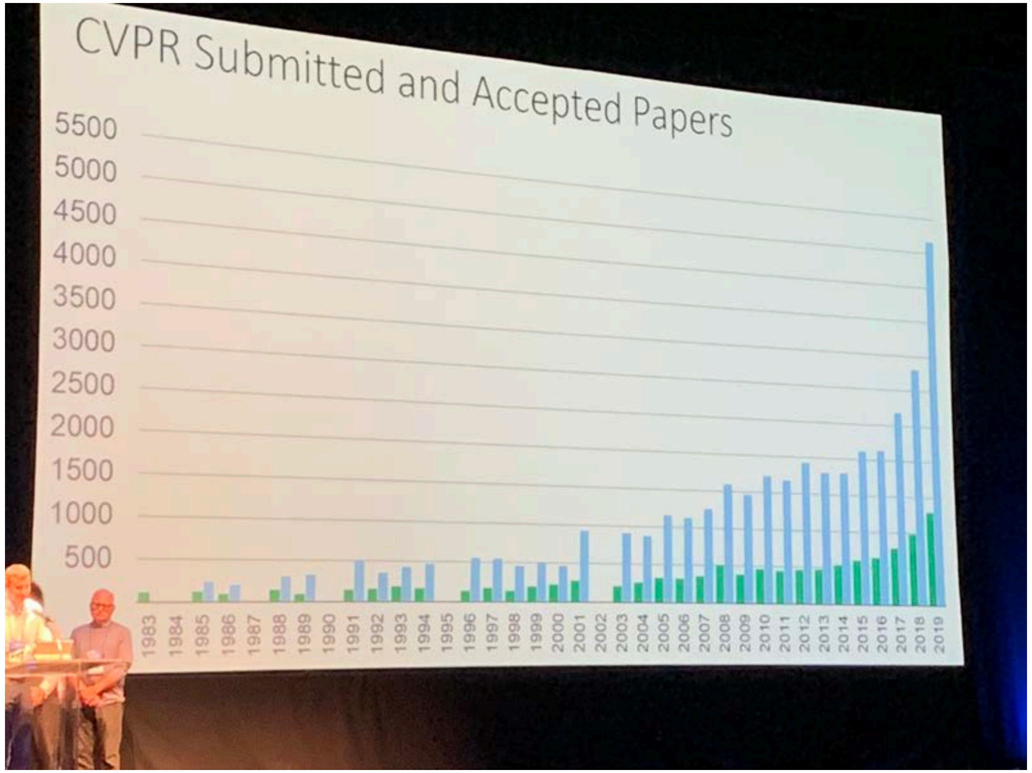
AI in Computer Vision



AI in Computer Vision



AI in Computer Vision



AI in Computer Vision

Autonomes Fahren: Softbank investiert zwei Milliarden Euro in GM-Tochter Cruise

US-Behörde genehmigt eine Zwei-Milliarden-Euro-Investition des japanischen Softbank-Konzerns in Cruise, der Roboterwagen-Tochter von General Motors.

Lesezeit: **1 Min.** In Pocket speichern 9

(Bild: General Motors)

07.07.2019 17:28 Uhr

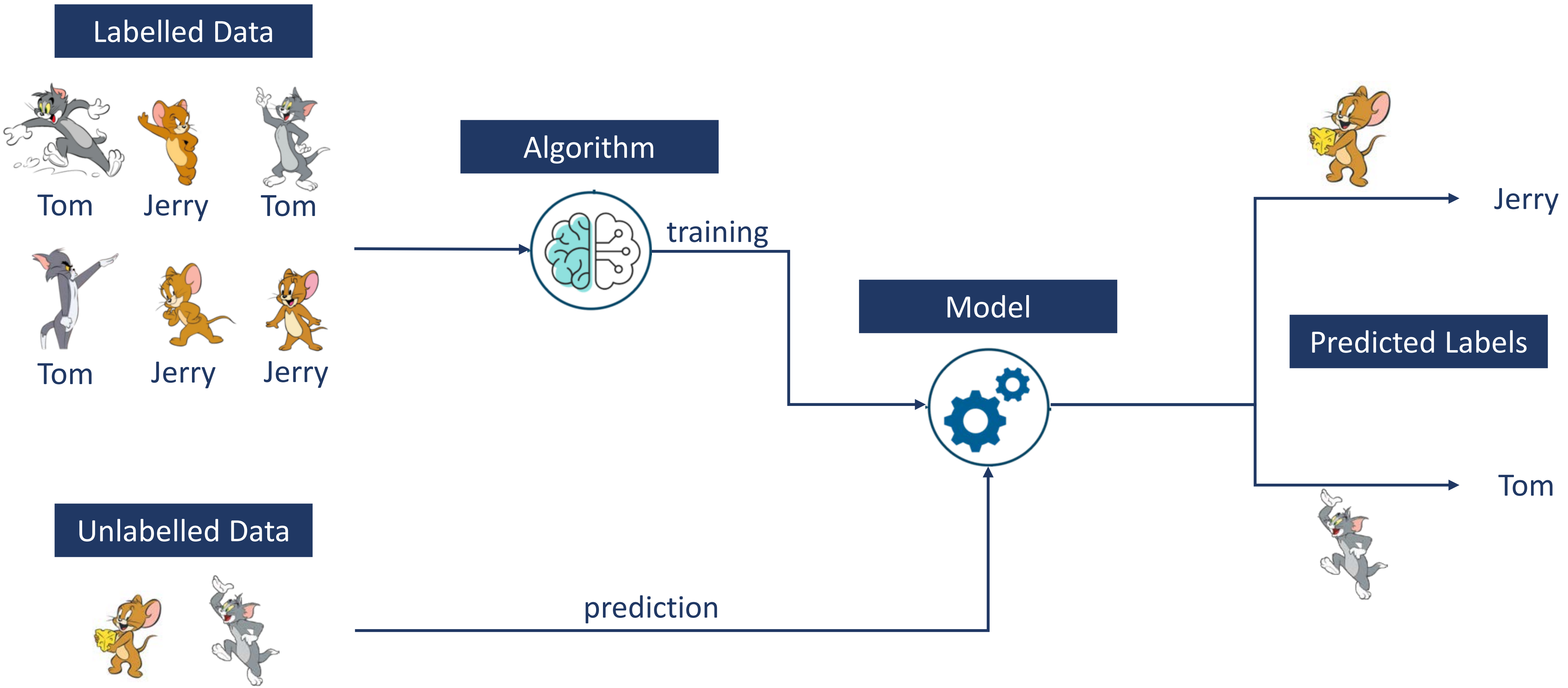
Von Bernd Mewes

AI Applications

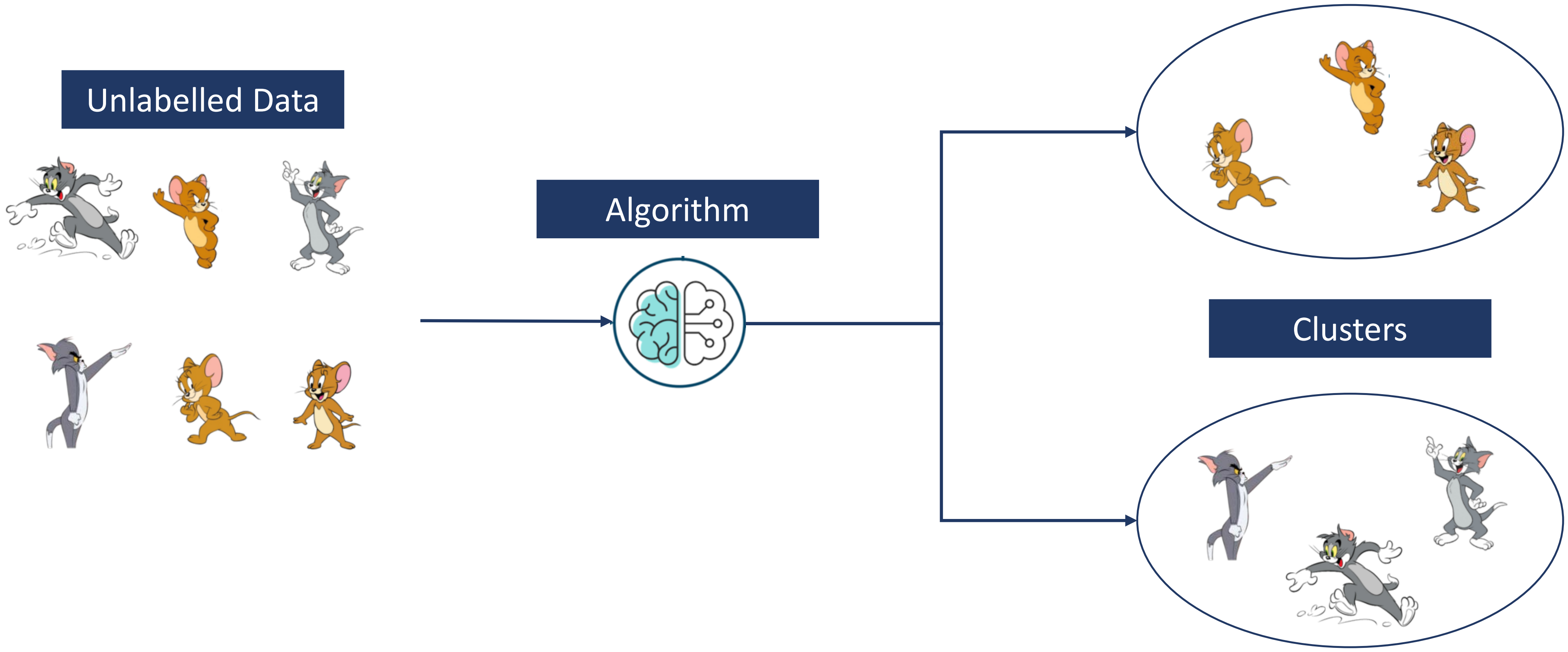
Goals for Today

- What we want to achieve today:
  - Theory of (Convolutional) Neural Networks
  - Arouse interest for further studies
- What we cannot achieve today:
  - Cover all details of Deep Learning
  - Go deep into Deep Learning for precision farming

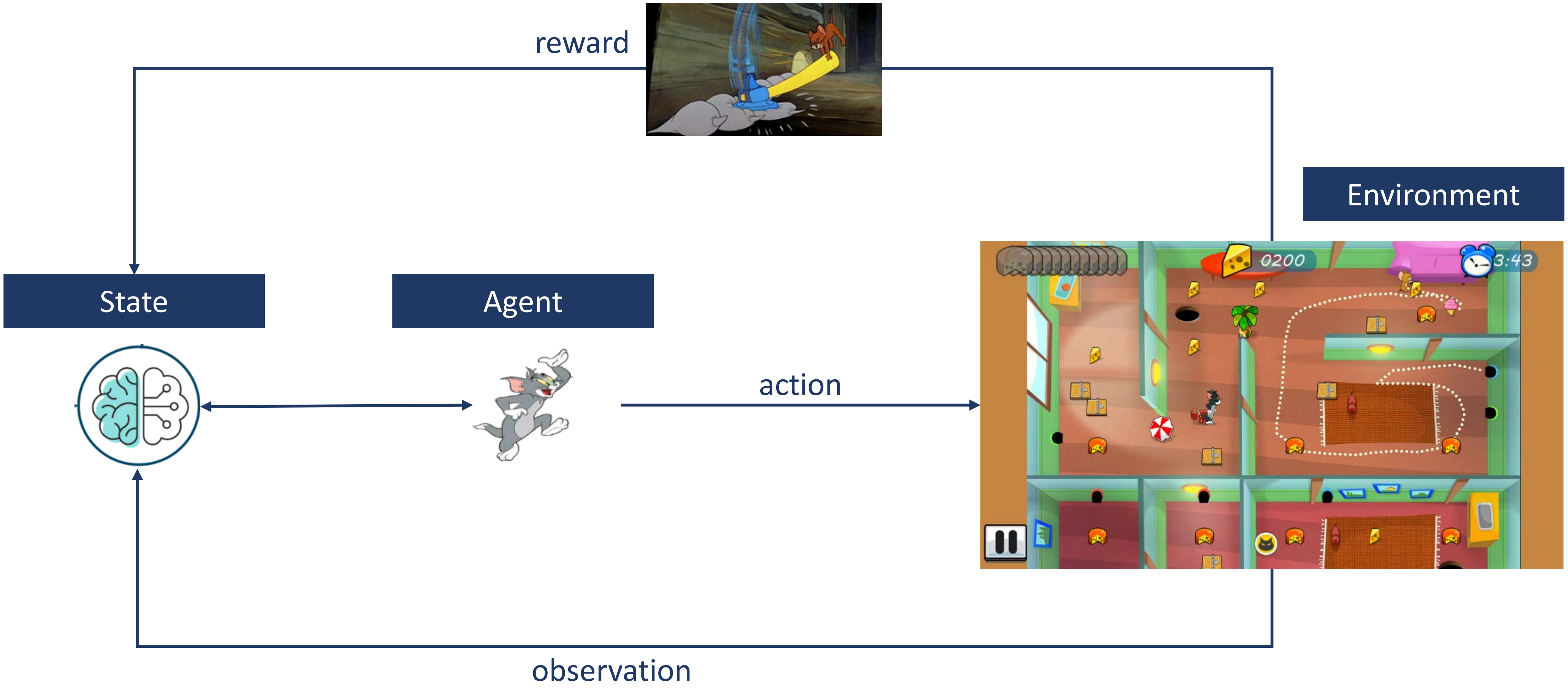
Supervised Learning



Unsupervised Learning



Reinforcement Learning



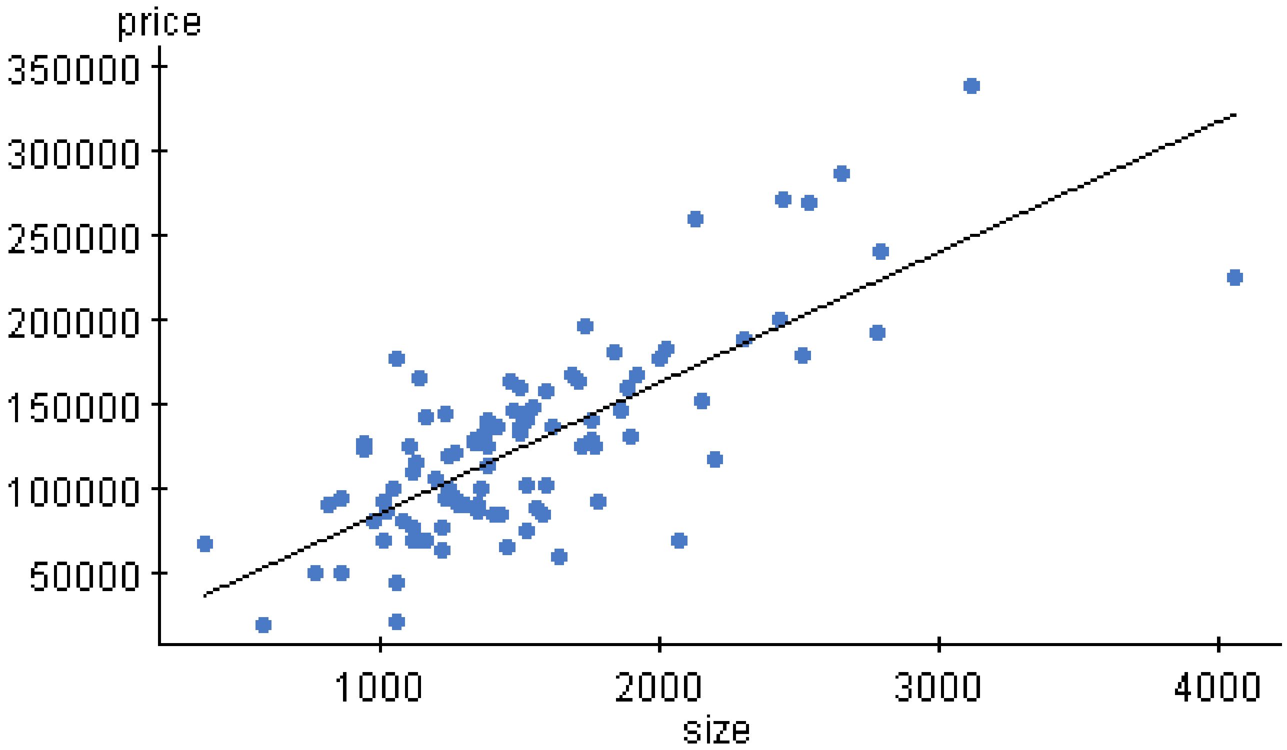
Categories of Machine Learning Algorithms

- Supervised learning
  - Given (training) data, which contains the correct answer for each dataset, the learning algorithm tries to find a hypothesis (model) that allows to predict the outcome for unseen datasets

- Unsupervised learning
  - The learning algorithm finds structure in the given data based on similarity and groups the data elements into clusters

- Reinforcement learning
  - The learning algorithm learns from rewards of previous decisions

Regression



- Learn a model by fitting a (straight) line through all (training) examples
- Predict the outcome for an unseen dataset by substituting the input values into the model

Linear Regression

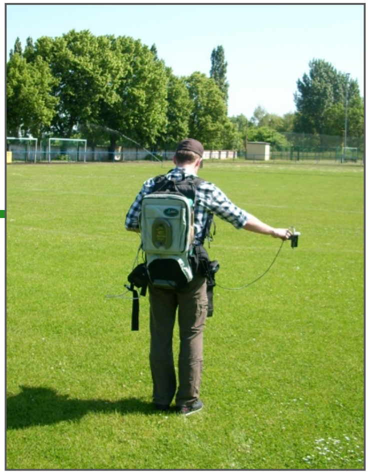
- Linear model that predicts a target value $\hat{y}$ by computing a weighted sum of the input features $(x_1, x_2, \ldots, x_n)$ plus a bias term b

$$\hat{y} = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

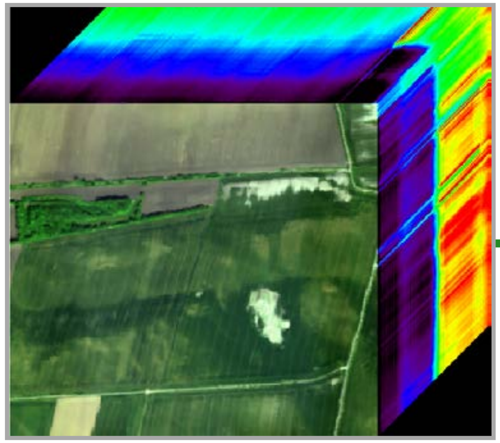
$$\boxed{\hat{y} = \sum_{i=1}^{n} w_i x_i + b}$$

Linear Regression for Crop Estimation

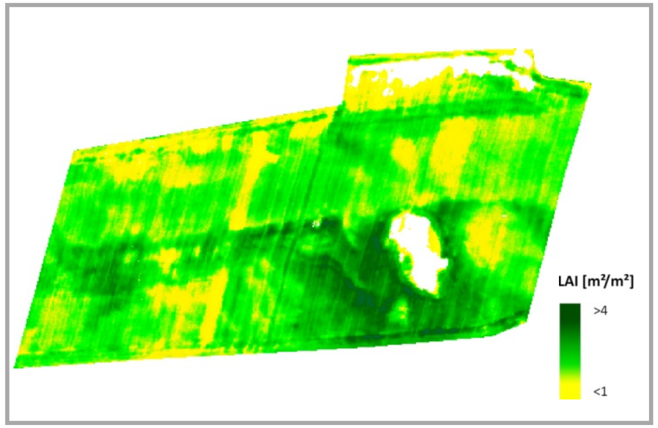
field measurements



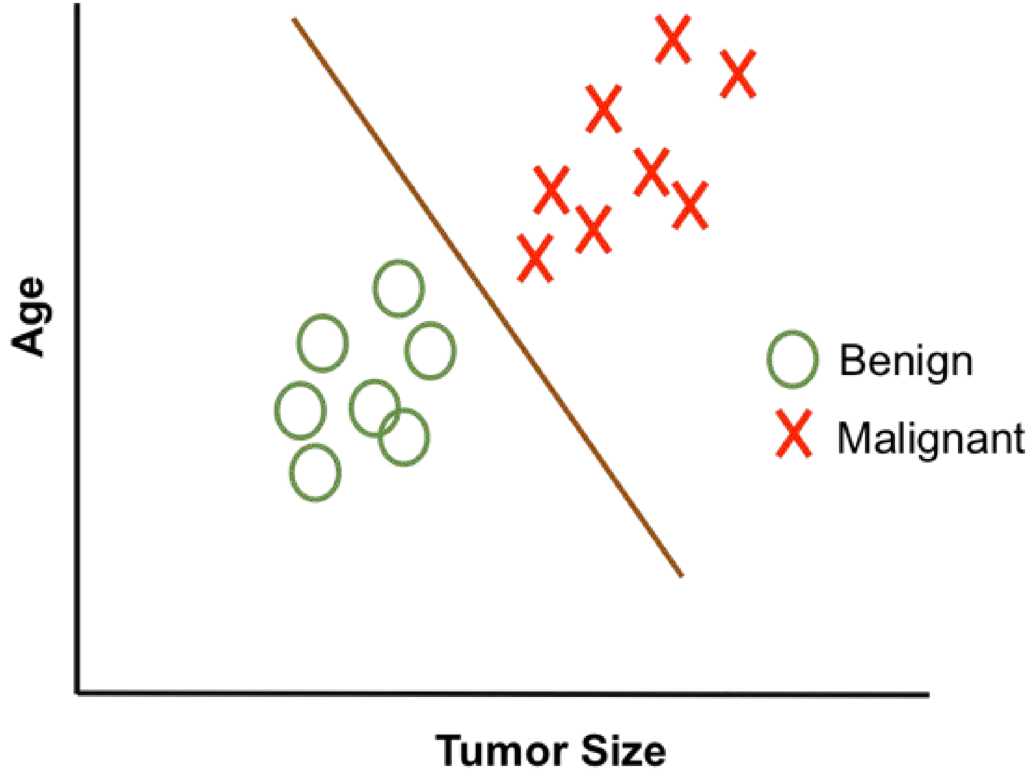
Linear regression model for vegetation parameters, e.g., for crop estimation



hyperspectral data



Classification



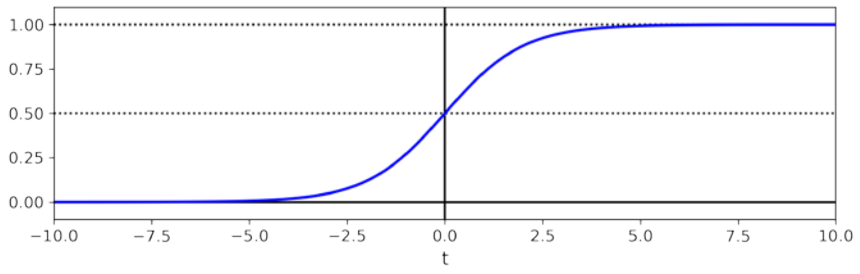
- Learn a model by finding a (straight) line that separates the (two) classes
- Predict the class by determining in which region your unseen input dataset lies

Logistic Regression

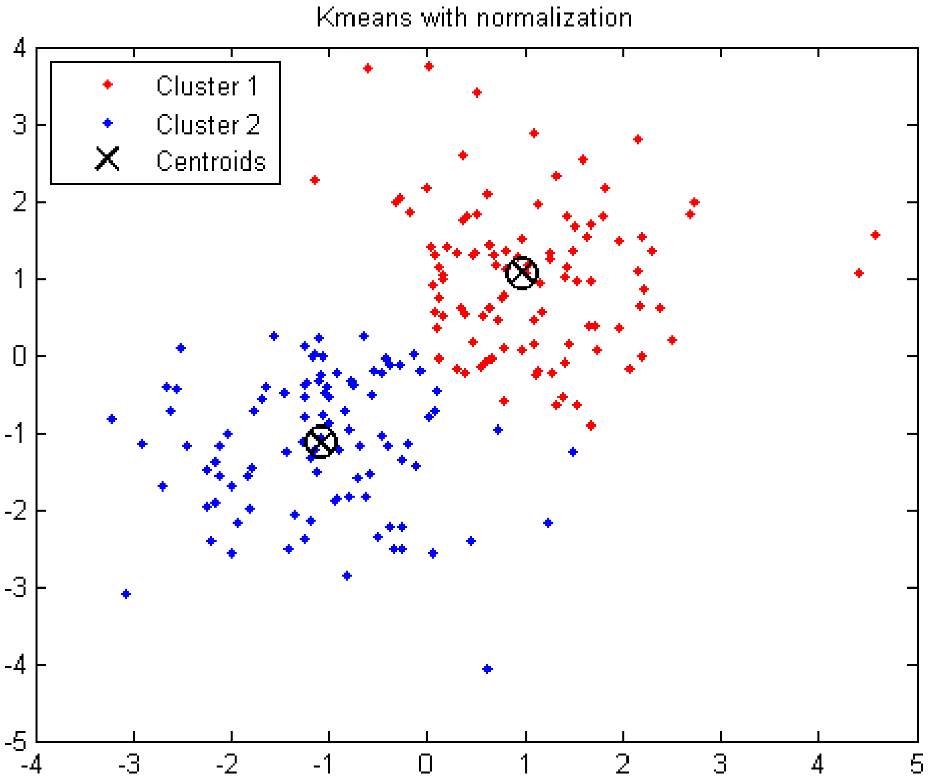
- Computes a probability that the object with the given input features belongs to the class (or does not belong to the class) by using the sigmoid logistic function

$$\boxed{\hat{p} = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right)}$$

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$



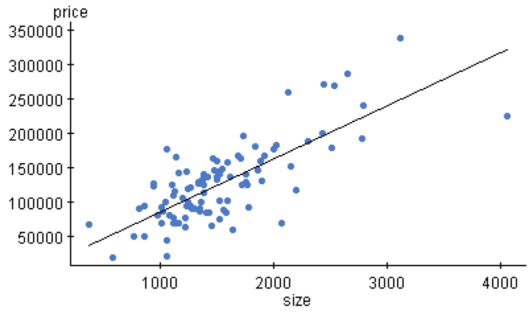
Segmentation



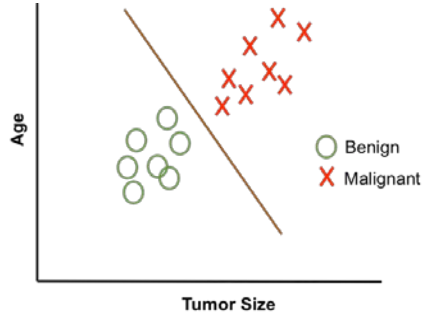
- Learn the structure of data by grouping similar examples into a set of clusters
- Predict the properties of an unseen dataset by its closeness to a cluster

Categories of Machine Learning Problems

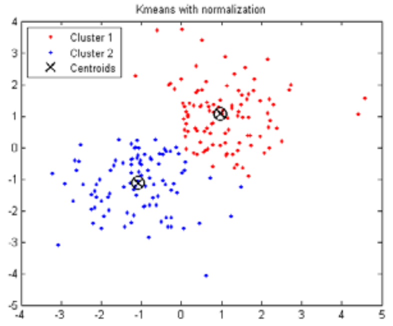
- Regression – supervised learning problem where the answer to be learned is a continuous value



- Classification – supervised learning problem where the answer is discreet (one of finitely many) values



- Segmentation – unsupervised learning problem where the structure to be learned is a set of clusters of similar examples



Machine Learning Process

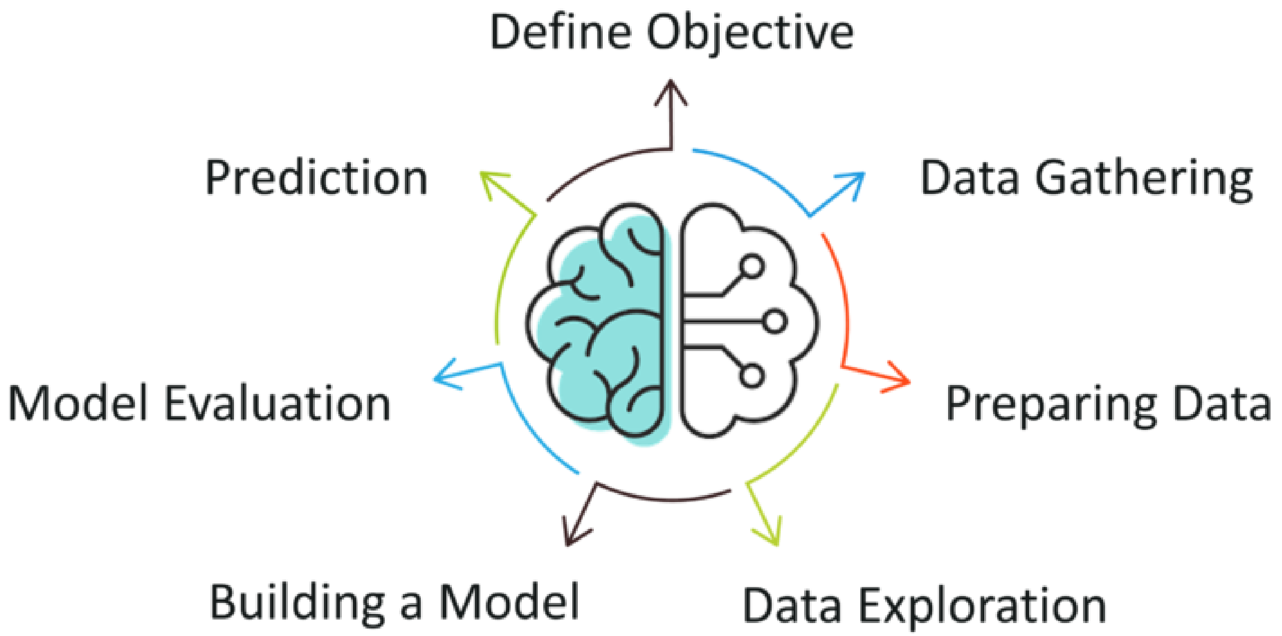
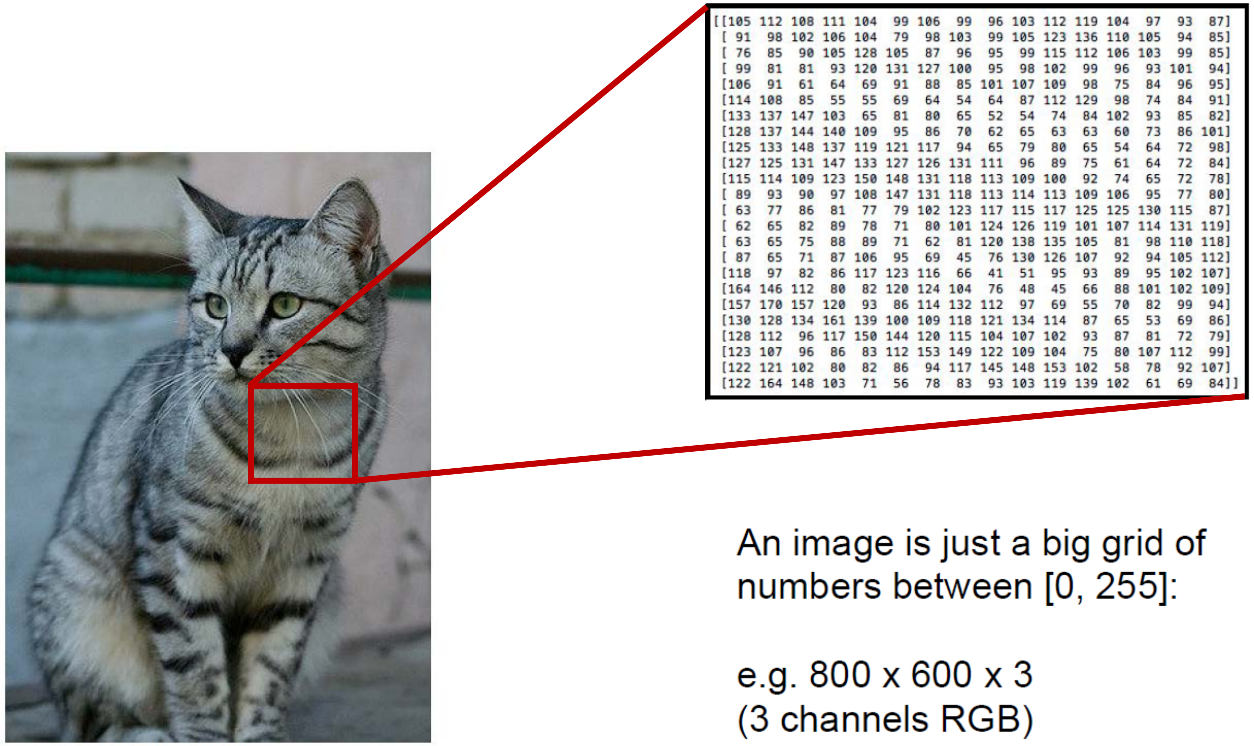


Image Classification



An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

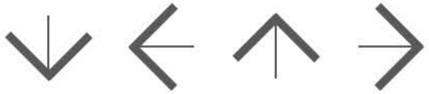
(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

Classical Approaches to Image Classification

Find edges

Find corners



?

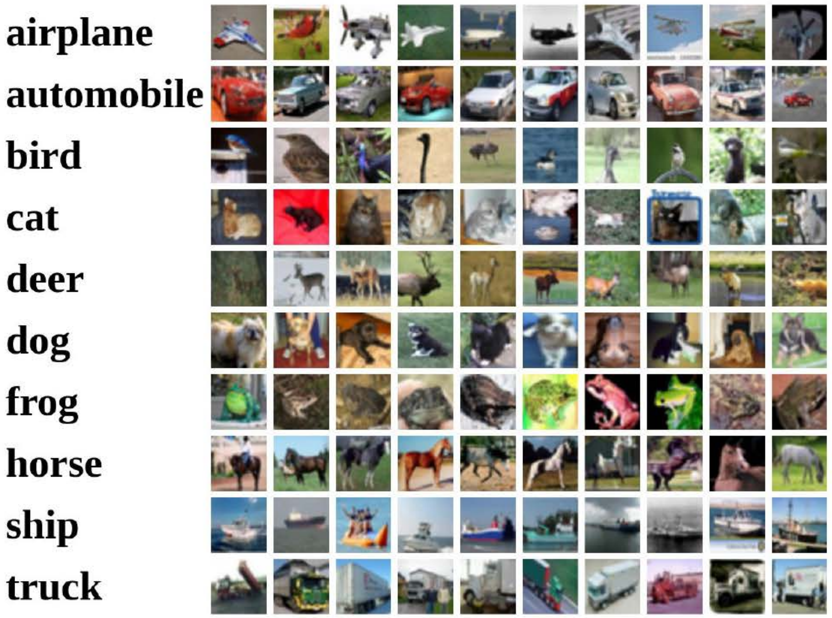
Define a set of rules that describes a cat from edge, corner and higher level features

John Canny, "A Computational Approach to Edge Detection", IEEE TPAMI 1986

Data-Driven Approach to Image Classification

1. Collect a (huge) dataset of labelled images
2. Train a classifier using machine learning techniques
3. Predict the class with the trained classifier

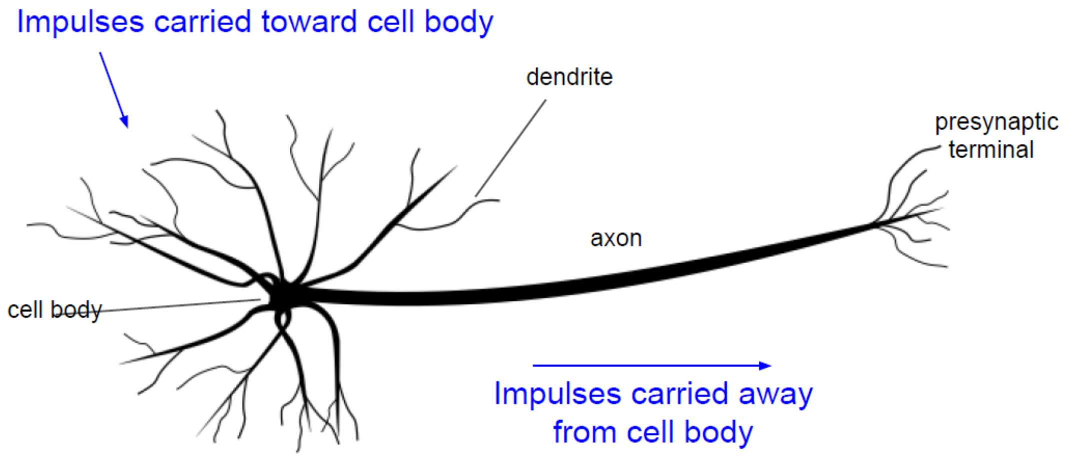
CIFAR10

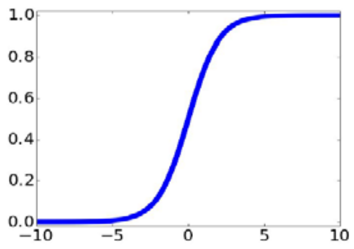


50,000 training images
each image is **32x32x3**

10,000 test images.

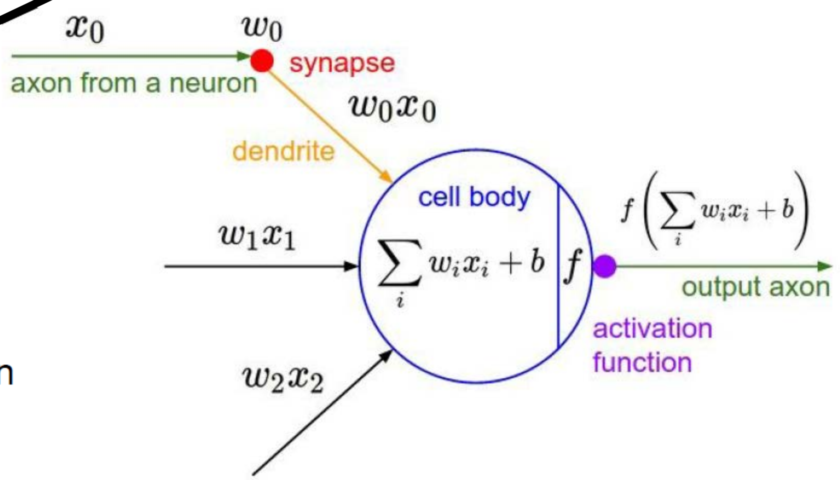
Neuron





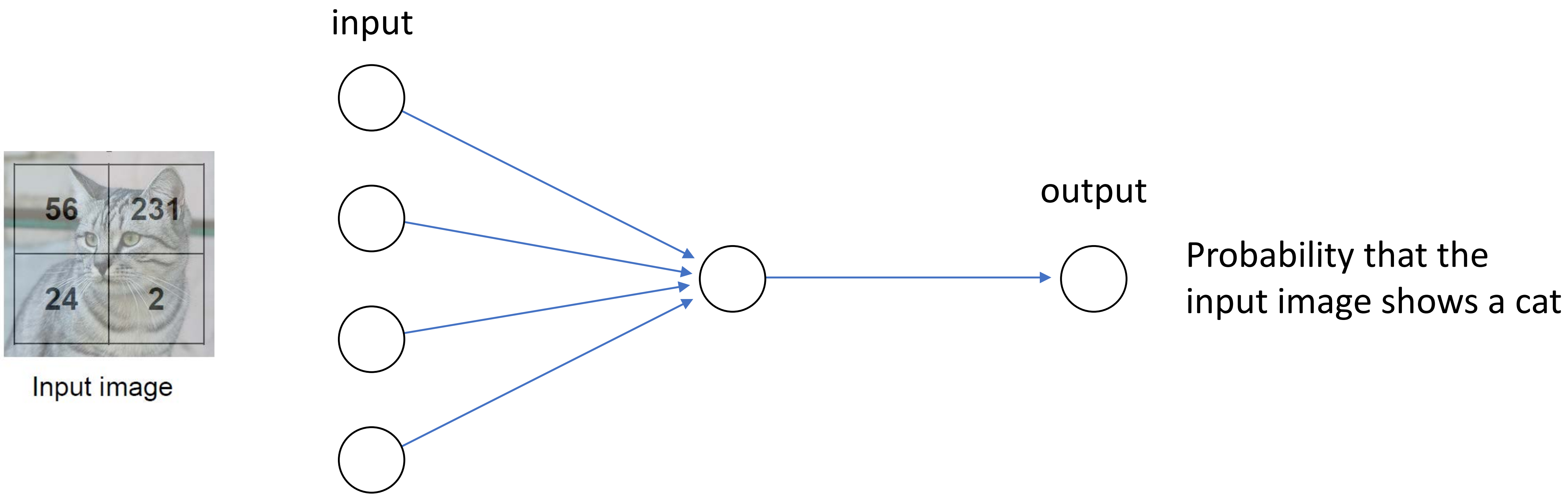
sigmoid activation function

$$\frac{1}{1+e^{-x}}$$



Perceptron

- Supervised learning algorithm for a binary classifier
- Takes vector data $\mathbf{x}$ as input and computes a single output value y



Mark I Perceptron

- First implementation of the perceptron algorithm by Frank Rosenblatt ~ 1957

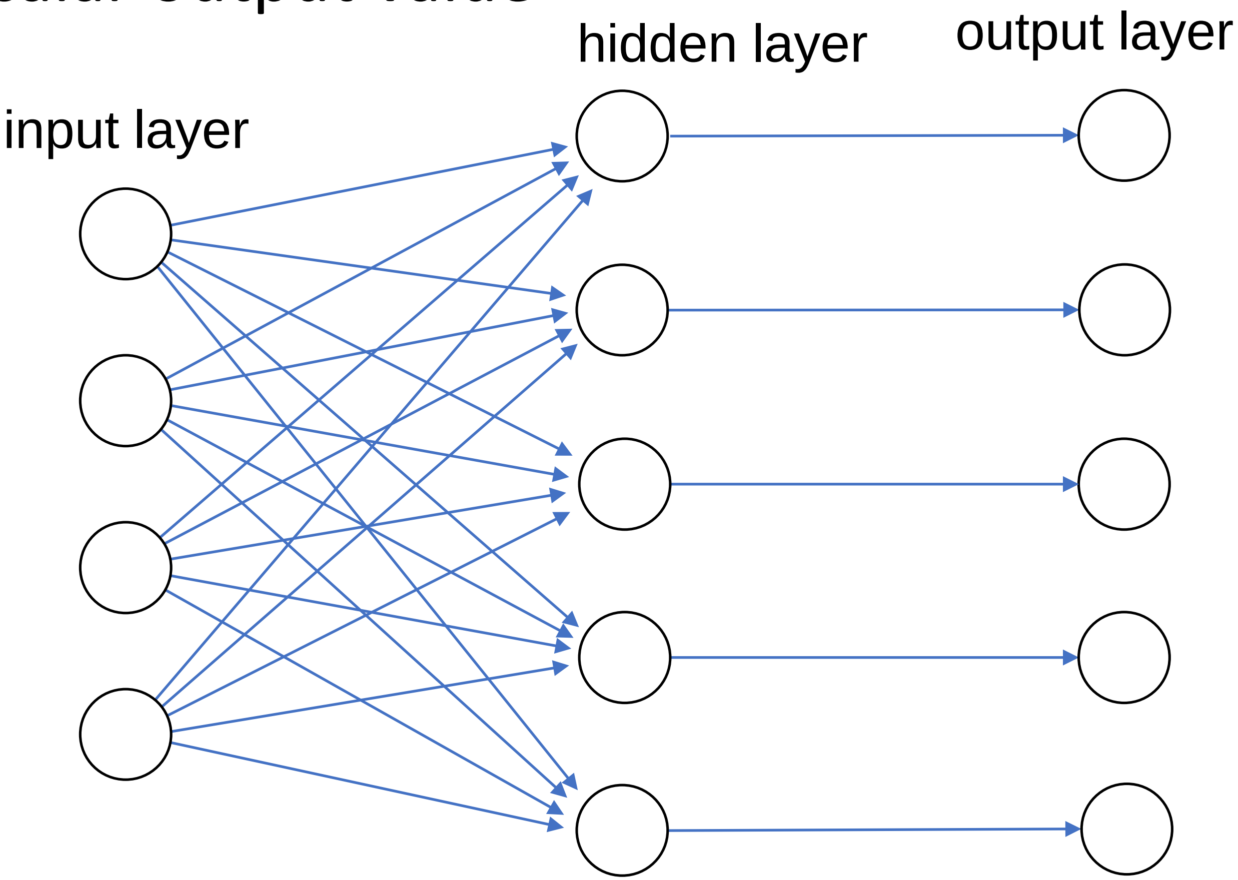
- 20 × 20 cadmium sulfide photocells (400 pixel image)

$$f(x) = \begin{cases} 1 & if \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0 & otherwise \end{cases}$$

- Recognizes letters of the alphabet

Multilayer Perceptron

- Many perceptron are grouped so that the output is a vector output instead of a scalar output value



each neuron of the output layer stands for a certain class

the output value of a neuron is the score for its class

the neuron with the highest score defines the predicted class

Parametric Approach to Linear Classifier

$$f(x,W) = Wx$$

Image

Array of **32x32x3** numbers
(3072 numbers total)

$f(\mathbf{x},\mathbf{W})$

10 numbers giving
class scores

W

parameters
or weights

Parametric Approach to Linear Classifier

$$\underbrace{\boxed{f(x,W)}}_{10\text{x}1} = \underbrace{\boxed{W}}_{10\text{x}3072}\overbrace{\boxed{x}}^{3072\text{x}1}$$

Image

Array of **32x32x3** numbers
(3072 numbers total)

$f(\mathbf{x},\mathbf{W})$

10 numbers giving
class scores

W
parameters
or weights

Parametric Approach to Linear Classifier

$$\underbrace{\boxed{f(x,W)}}_{10\text{x}1} = \underbrace{\boxed{W}}_{10\text{x}3072}\overbrace{\boxed{x}}^{3072\text{x}1} + \boxed{b}\ \ 10\text{x}1$$

Image

Array of **32x32x3** numbers
(3072 numbers total)

$\longrightarrow f(\mathbf{x},\mathbf{W}) \longrightarrow$ **10** numbers giving class scores

$\uparrow$

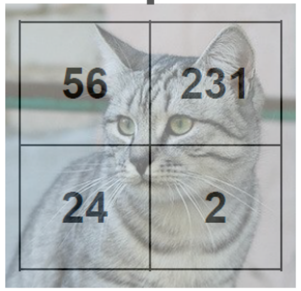
W

parameters
or weights

Parametric Approach to Linear Classifier

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

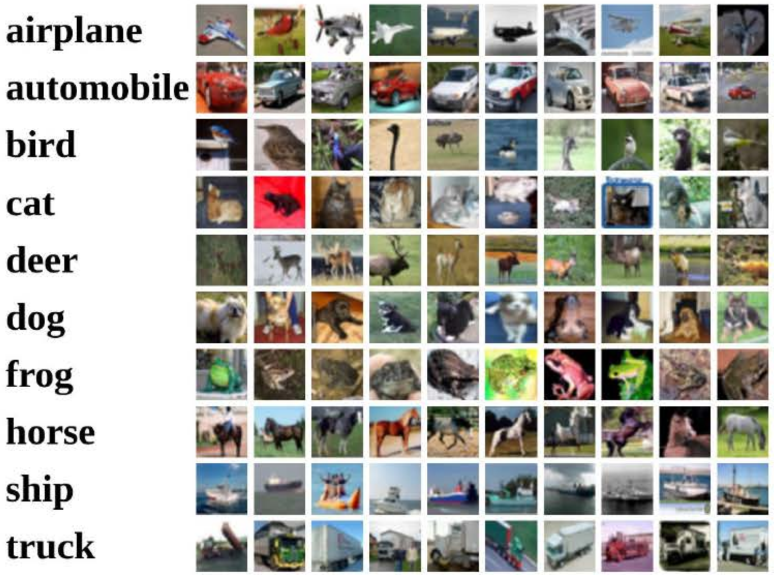
Stretch pixels into column



Input image

$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b} = \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix} \begin{array}{l} \text{Cat score} \\ \text{Dog score} \\ \text{Ship score} \end{array}$$

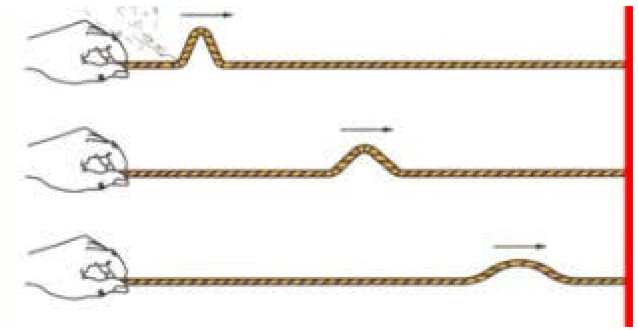
Interpreting a Linear Classifier

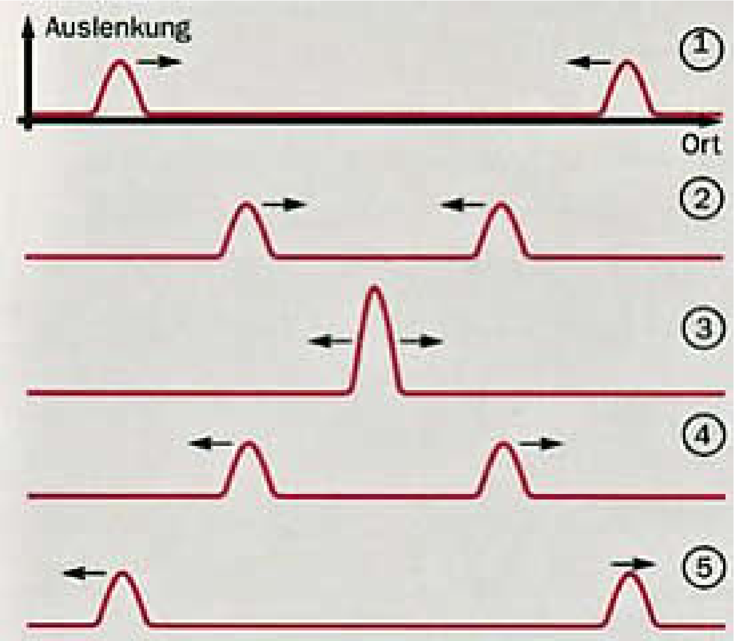


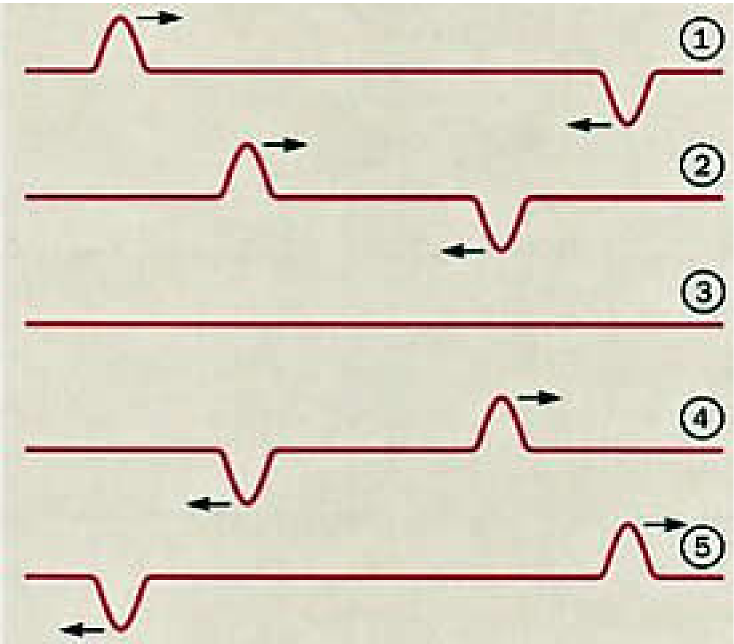
$$f(x,W) = Wx + b$$

Example trained weights of a linear classifier trained on CIFAR-10:

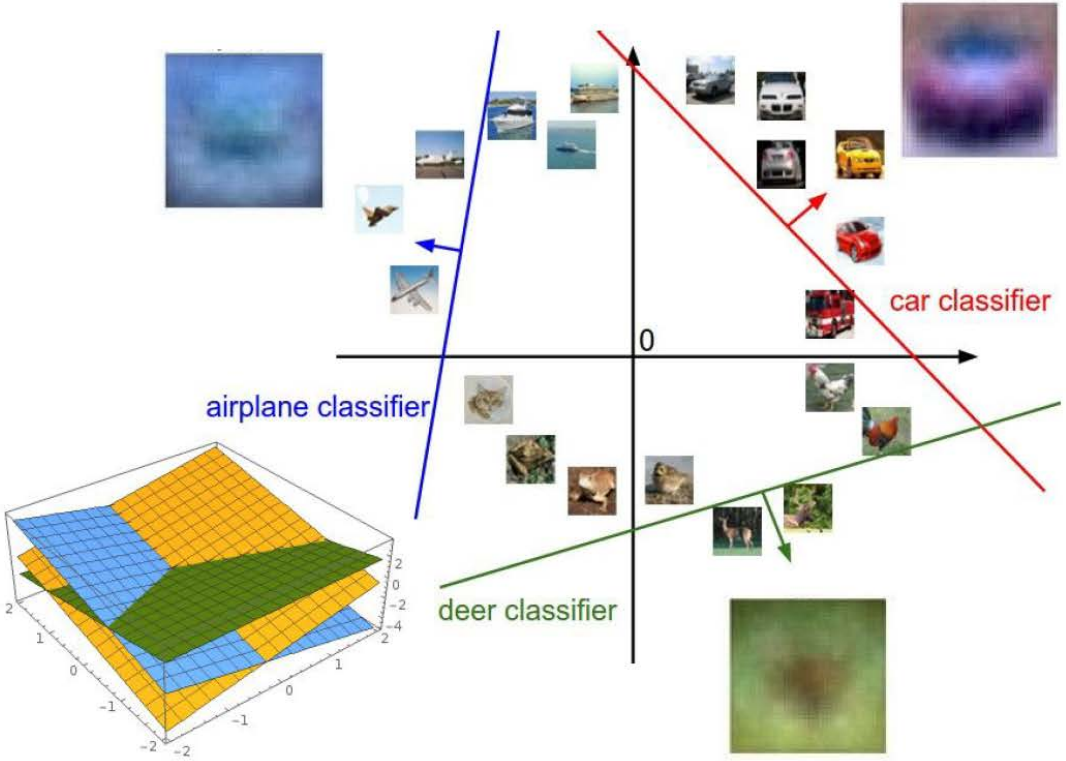
Interpreting a Linear Classifier







Interpreting a Linear Classifier



Plot created using Wolfram Cloud

$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers
(3072 numbers total)

German Traffic Sign Recognition Benchmark

- Single-image, multiple classes
- More than 40 classes
- More than 50,000 images
- Best recognition rates:
    1. 99.46% (committee of CNNs)
    2. 98.84% (human performance)
    3. 98.31% (multi-scale CNNs)

Training
Neural Networks

Prof. Dr. Martin Kada

Training Neural Networks

- Initialize the weights of the network
- Evaluate how good the network is
- (Stepwise) improve the network
  - Gradient descent
  - Backpropagation
  - Learning rate
- Activation functions

Weight Initialization

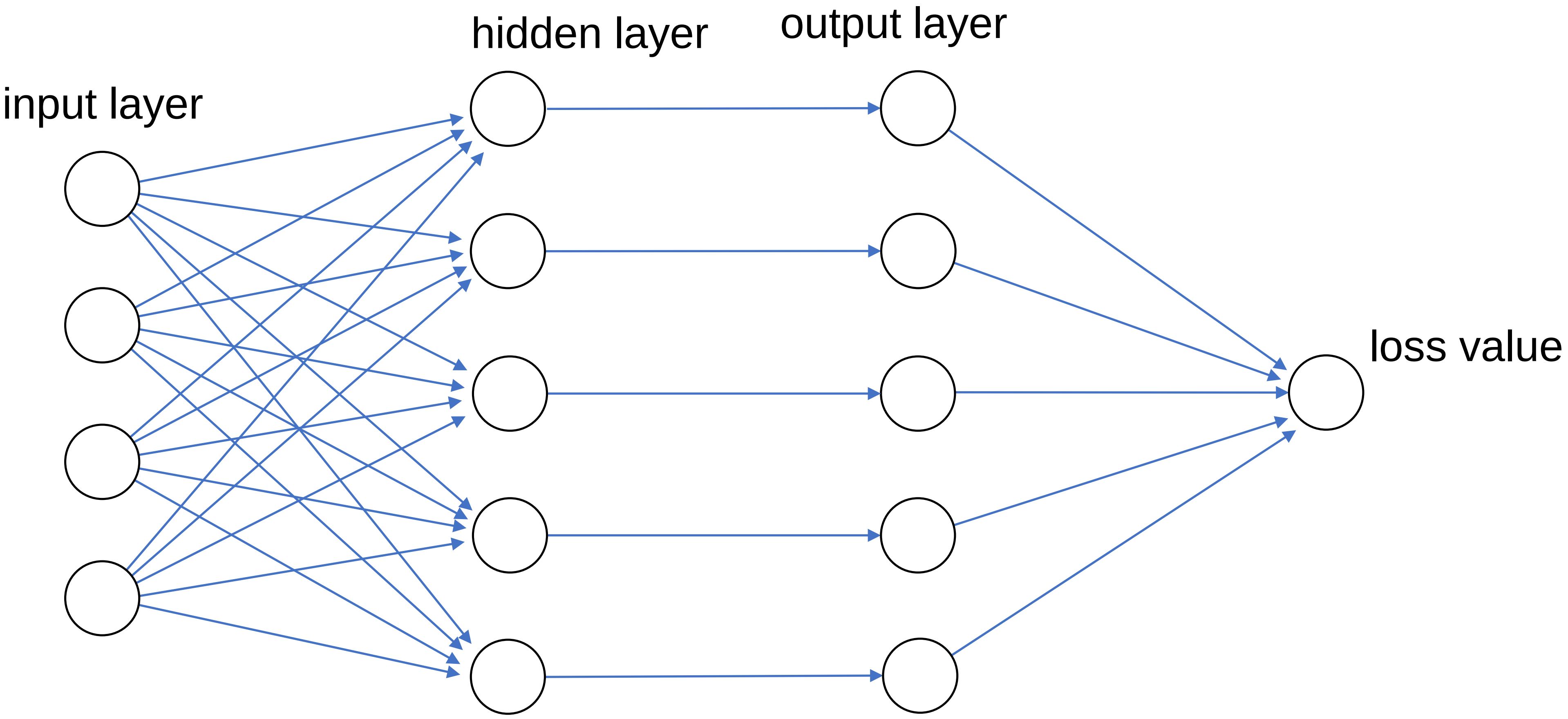
- Initialize all values of weight matrix W with random gaussian noise with zero mean and a user-defined (e.g. 0.01) variance
  - Works only good for shallow networks
  - Weights initialized too small,
    then the signal shrinks as it passes through each layer until it vanishes
  - Weights initialized too large,
    then the signal grows as is passes through each layer until it explodes

- Xavier initialization
  - Makes sure the weights are just right, keeping the signal in a reasonable range of values through many layers

$$Var(W) = \frac{2}{n_{in} + n_{out}}$$

Loss Function

- Quantifies how good the model is at the intended task



Softmax Loss Function

- Generalization of the logistic function to multiple classes

- Assumption: scores are unnormalized log probabilities of the classes

2. normalize to get probabilities

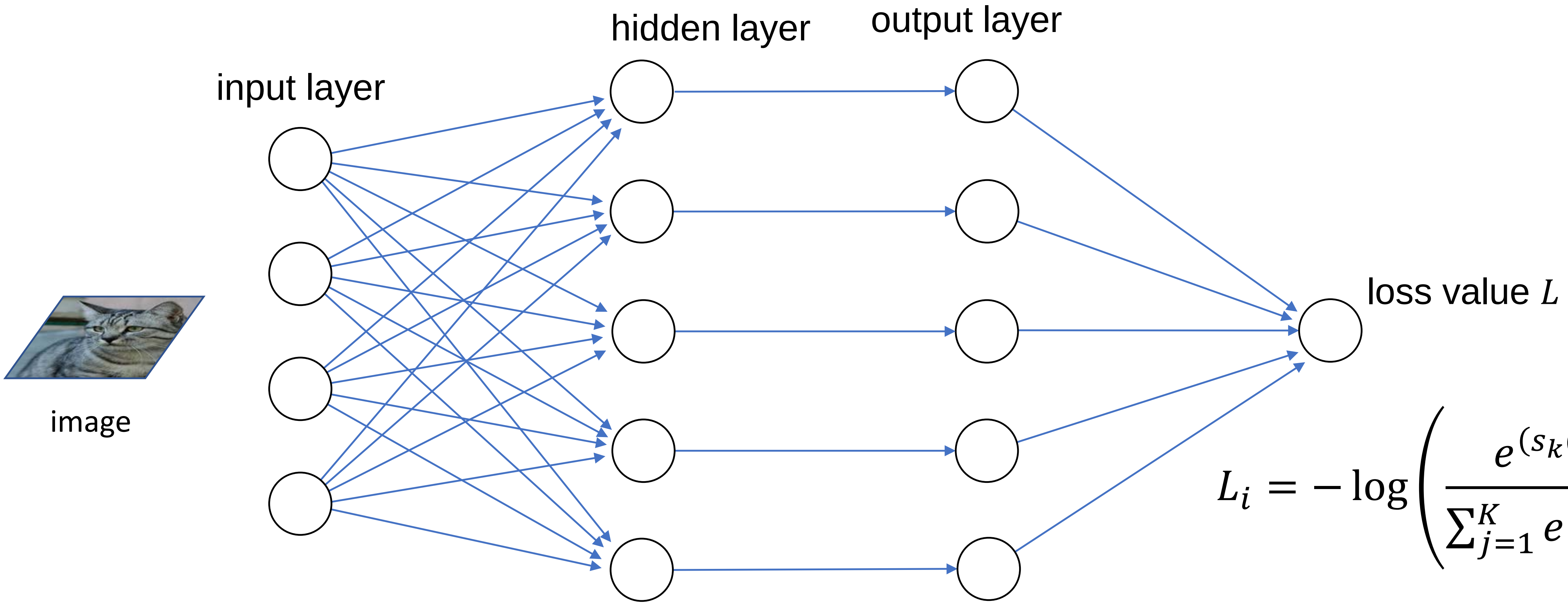
1. take the exponential of the scores

- Minimize the loss (for a given image i) w.r.t. the correct class k
  - Categorical Cross-Entropy loss (also called Softmax Loss) is a Softmax activation plus a Cross-Entropy loss

$$\boxed{L_i = -\log\left(\frac{e^{(s_k(\mathbf{x}))}}{\sum_{j=1}^{K} e^{(s_j(\mathbf{x}))}}\right)}$$

Softmax Loss Function

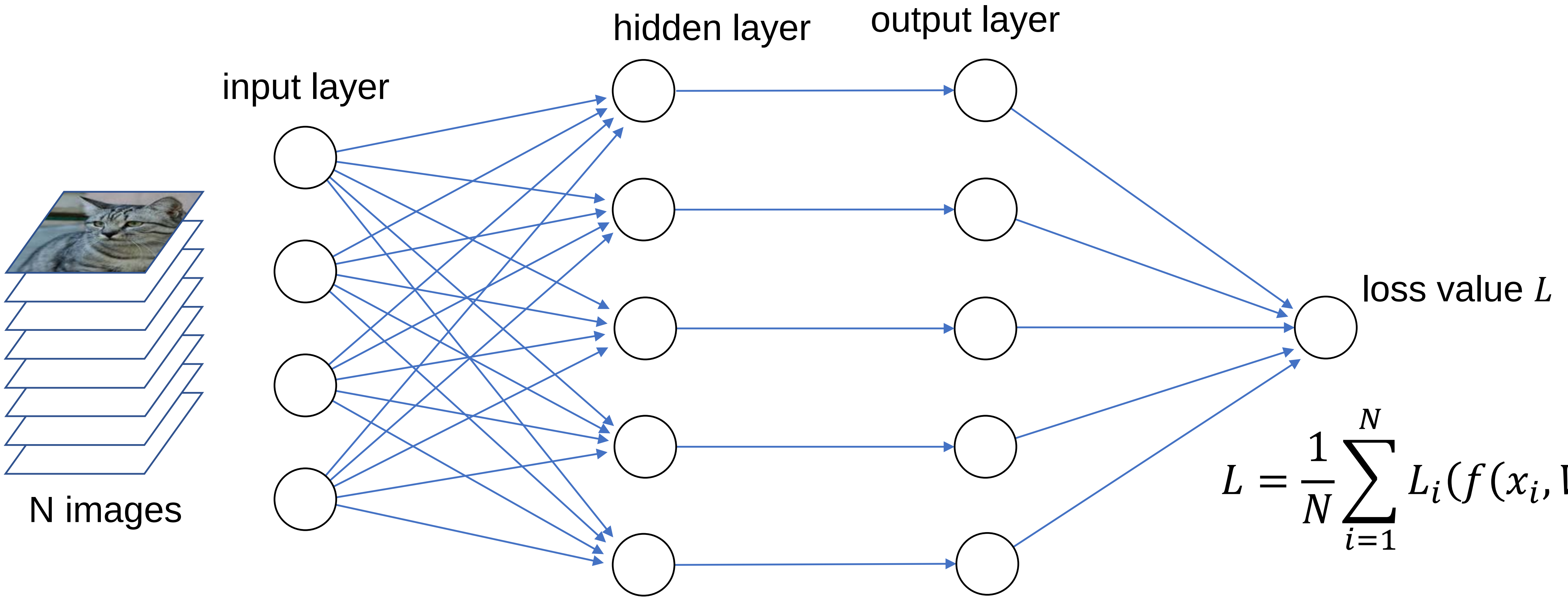
- Quantifies how good the model is at the intended task



$$L_i = -\log\left(\frac{e^{(s_k(\mathbf{x}))}}{\sum_{j=1}^{K} e^{(s_j(\mathbf{x}))}}\right)$$

Softmax Loss Function

- Quantifies how good the model is at the intended task



$$L = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)$$

Softmax Loss Function

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

unnormalized probabilities

cat	**3.2**		**24.5**		**0.13**	→ L_i = -log(0.13) = **0.89**
car	5.1	exp →	164.0	normalize →	0.87	
frog	-1.7		0.18		0.00	

unnormalized log probabilities

probabilities

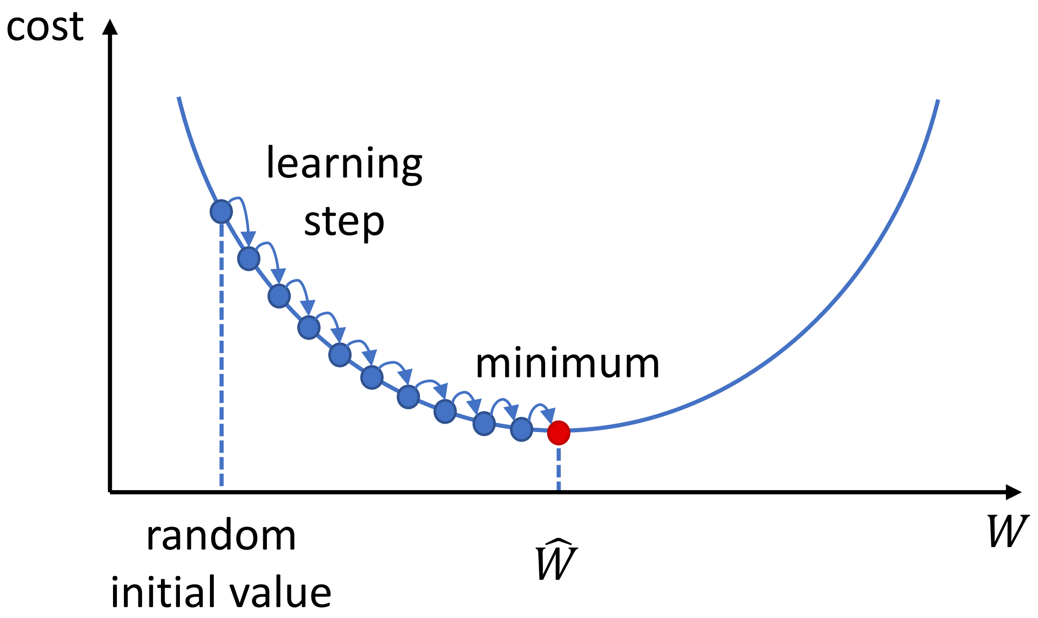
Gradient Descent

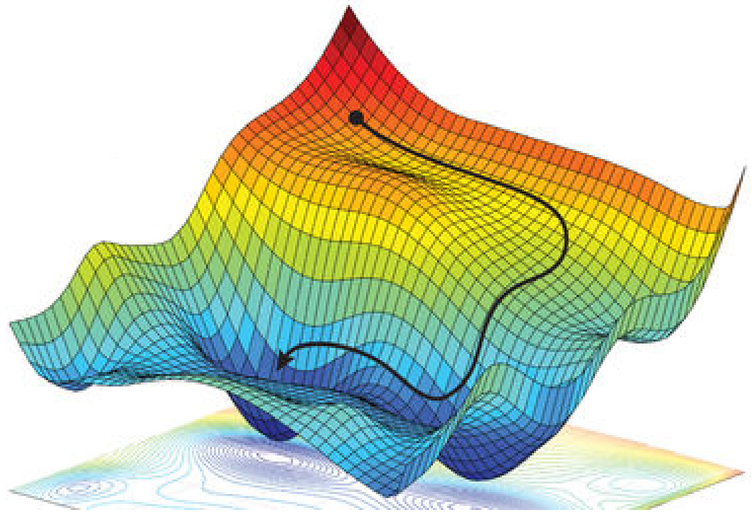
- Finds optimal weights by iteratively tweaking the model parameters W in order to minimize the cost function L

- Idea:
  - Go downhill in the direction of the steepest slope until you reach a valley
  - Measure the local gradient of the cost function with regard to the parameter vector W and go in the direction of descending gradient until a minimum is reached

Gradient Descent

- Algorithm:
  - Initialize W with random values (random initialization)
  - Gradually improve W by backpropagation to decrease the cost function
  - Stop when W converges to a minimum





Gradient Descent

- The loss L is a function of W

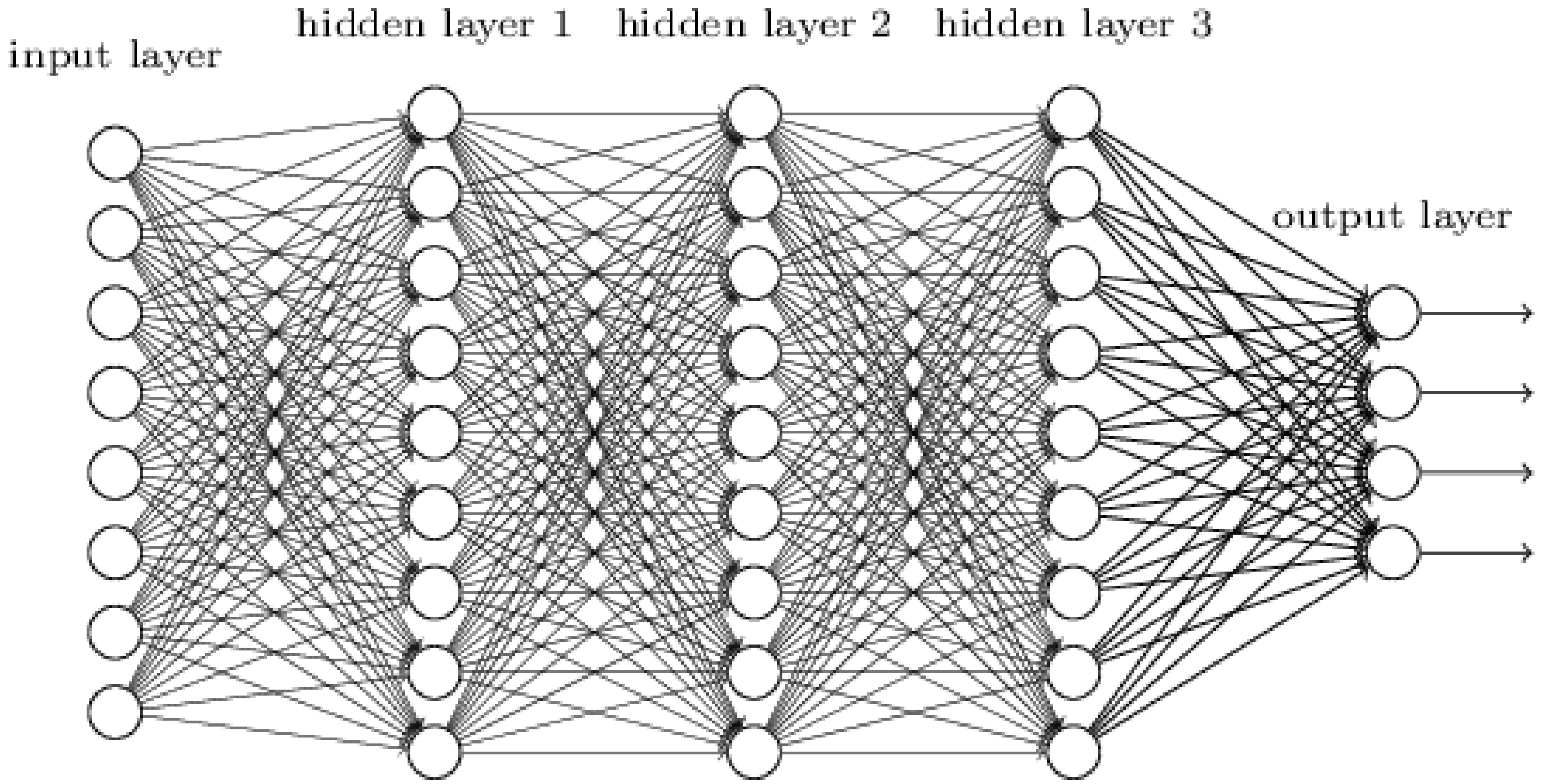
$$L = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)$$

- Calculate analytical $\boxed{\text{gradient vector } \nabla_w L}$

backpropagation

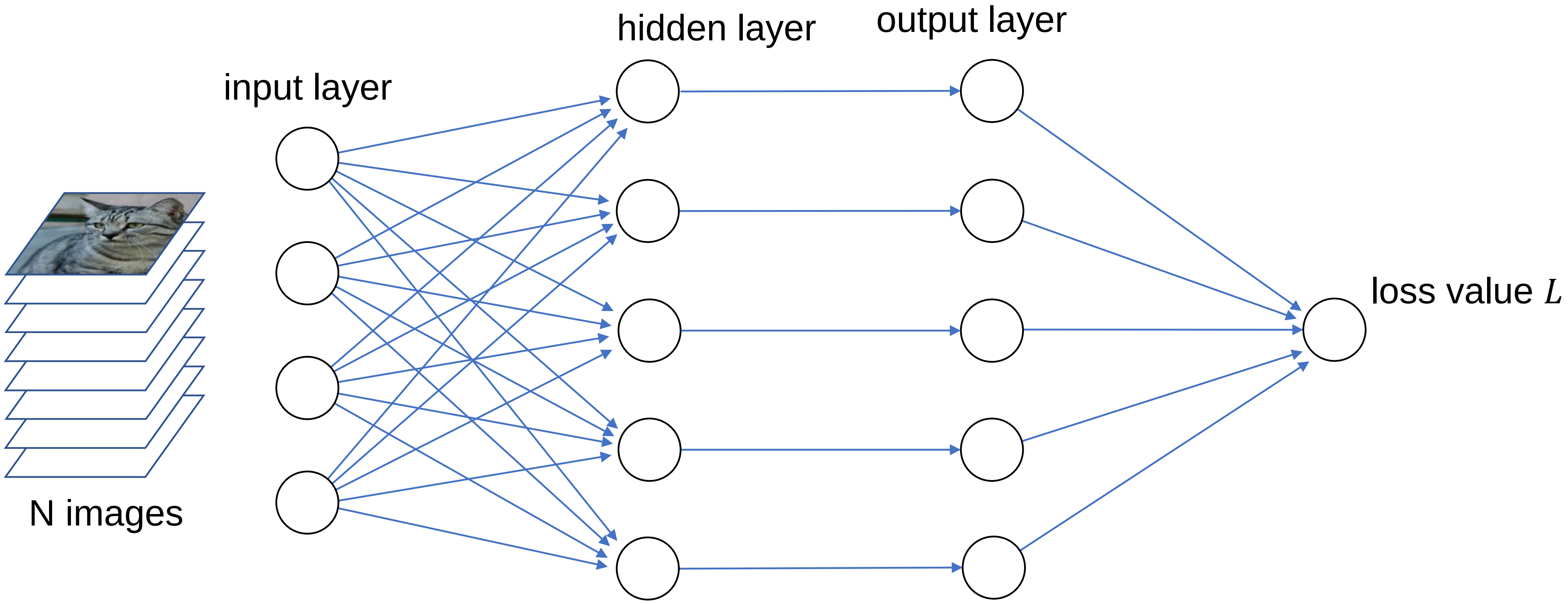
- Update W by computing $W' = W - \eta \cdot \nabla_w L$ where η is the learning rate

Gradient Descent



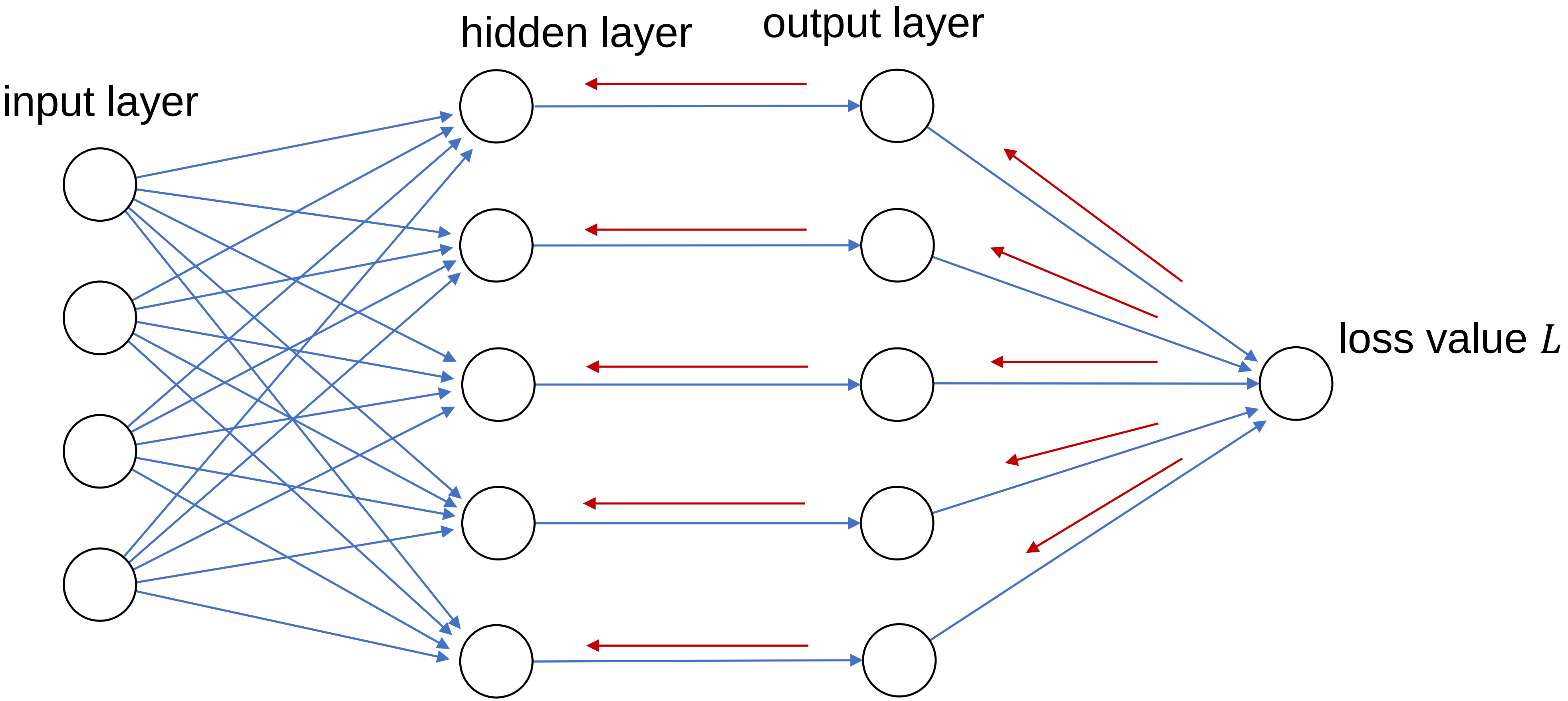
Backpropagation

- Forward pass – compute the aggregated output of all neurons and the loss



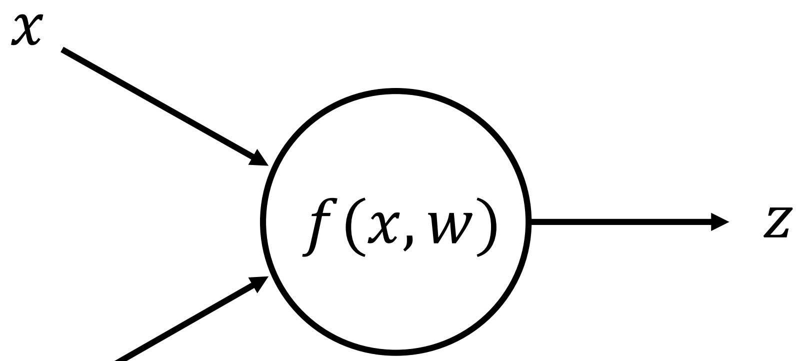
Backpropagation

- Backwards pass – update the weights W w.r.t. to the loss



Backpropagation

Forward pass

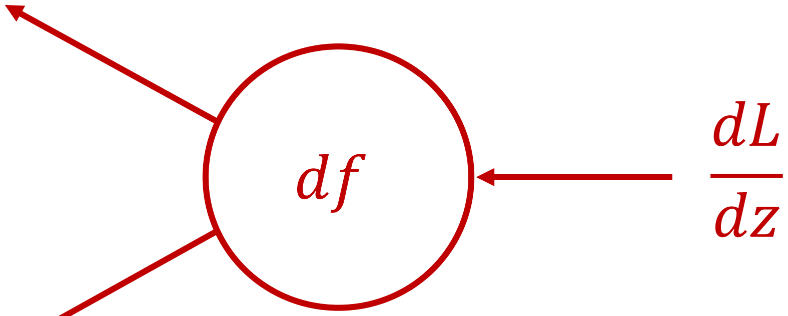


$$z = x \cdot w$$

Backward pass

$$\frac{dL}{dx} = \frac{dL}{dz}\frac{dz}{dx}$$

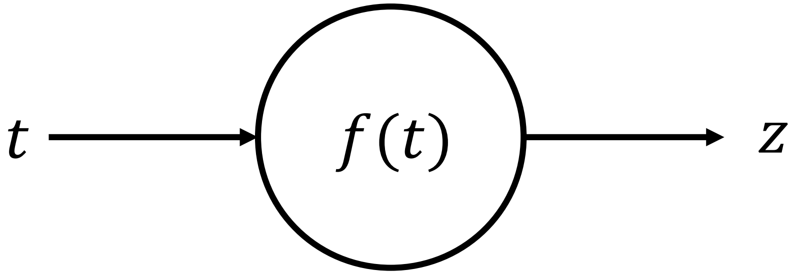
$$\frac{dL}{dw} = \frac{dL}{dz}\frac{dz}{dy}$$



$$\frac{dz}{dx} = w \qquad \frac{dz}{dw} = x$$

Backpropagation

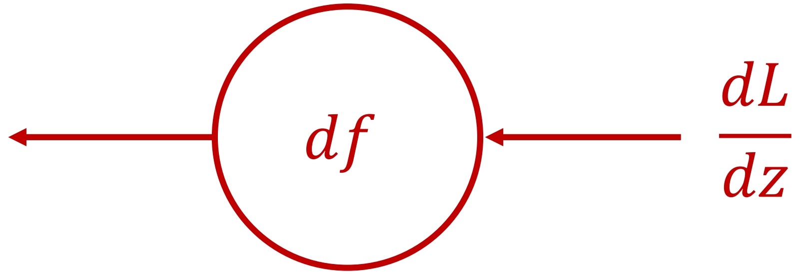
Forward pass



$$z = \sigma(t) = \frac{1}{1+e^{-t}}$$

Backward pass

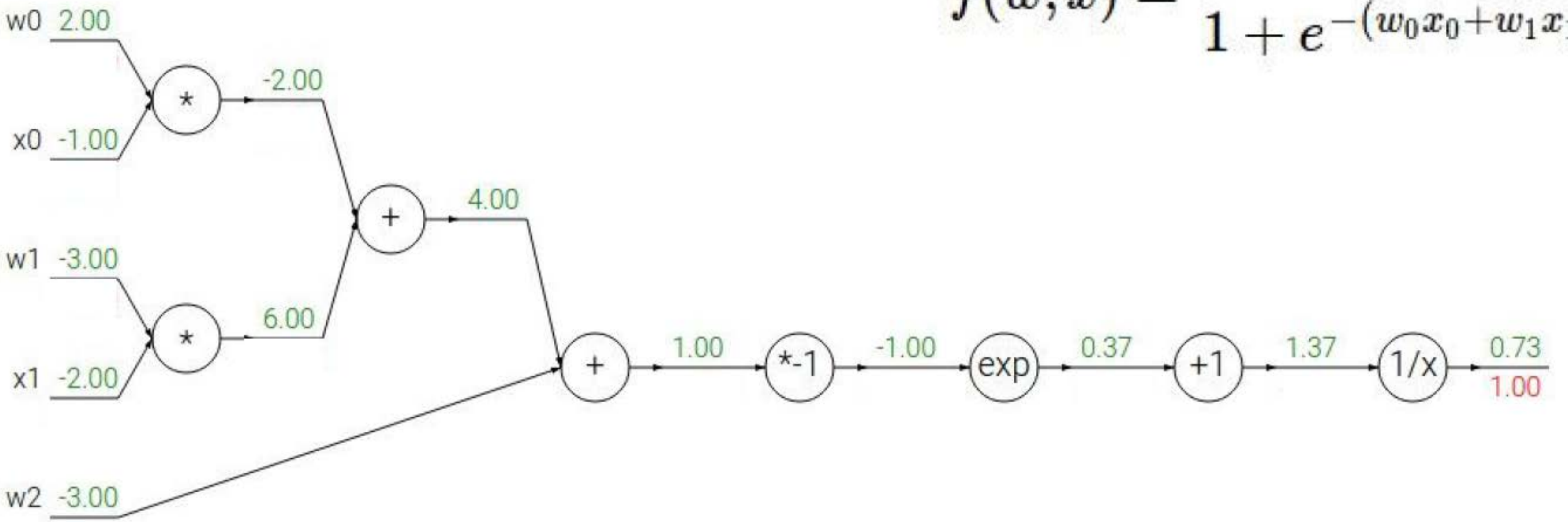
$$\frac{dL}{dt} = \frac{dL}{dz}\frac{dz}{dt}$$



$$\frac{dz}{dt} = (1-\sigma(t))\sigma(t)$$

Backpropagation

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

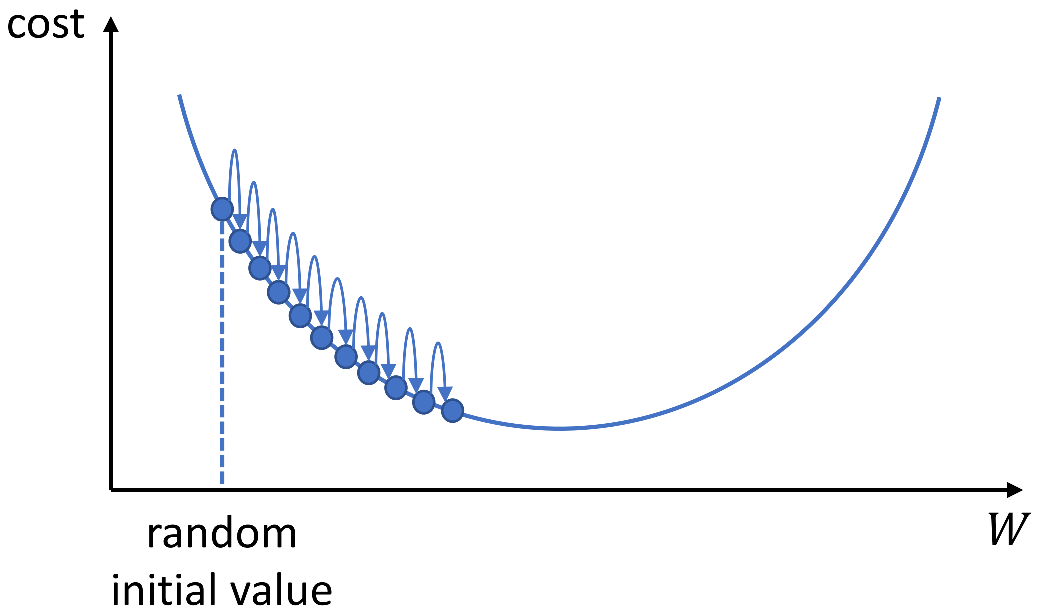


$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \qquad\qquad f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

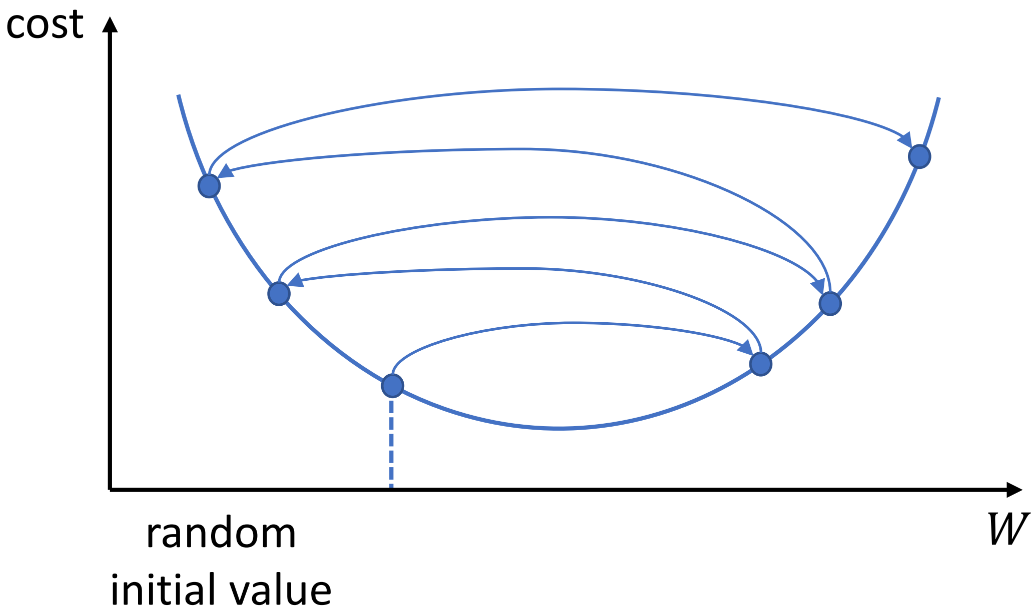
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \qquad\qquad f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Learning Rate

- The hyperparameter learning rate η determines how well the algorithm converges



If the learning rate is too small, then the algorithm will take many iterations to converge

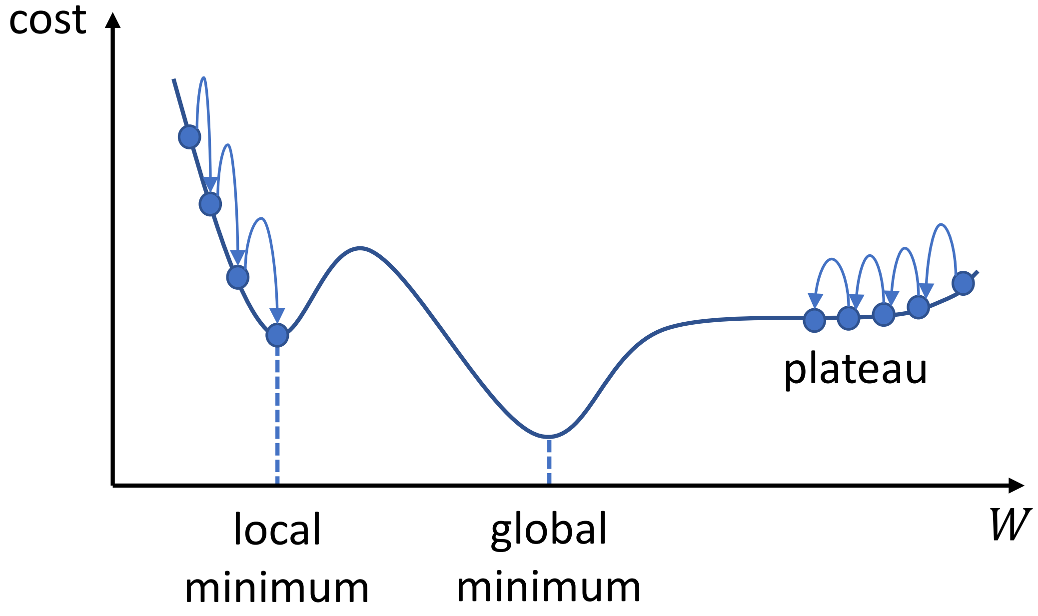


If the learning rate is too high, then the algorithm might overjump the minimum, possibly ending up even further away from the minimum than before $\rightarrow$ algorithm diverges and fails to find a good solution

Learning Rate

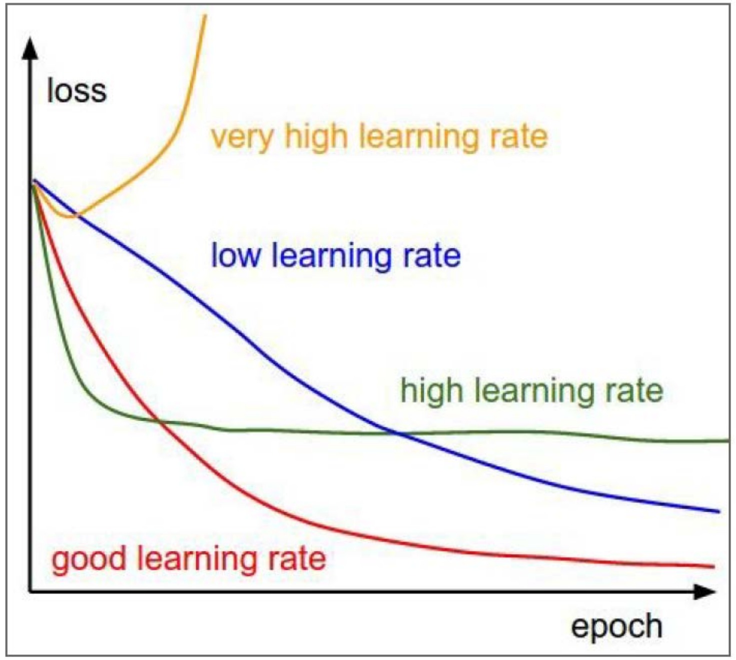
- Cost function might have valleys, ridges, plateaus and other irregular shapes → makes it difficult to converge to the minimum

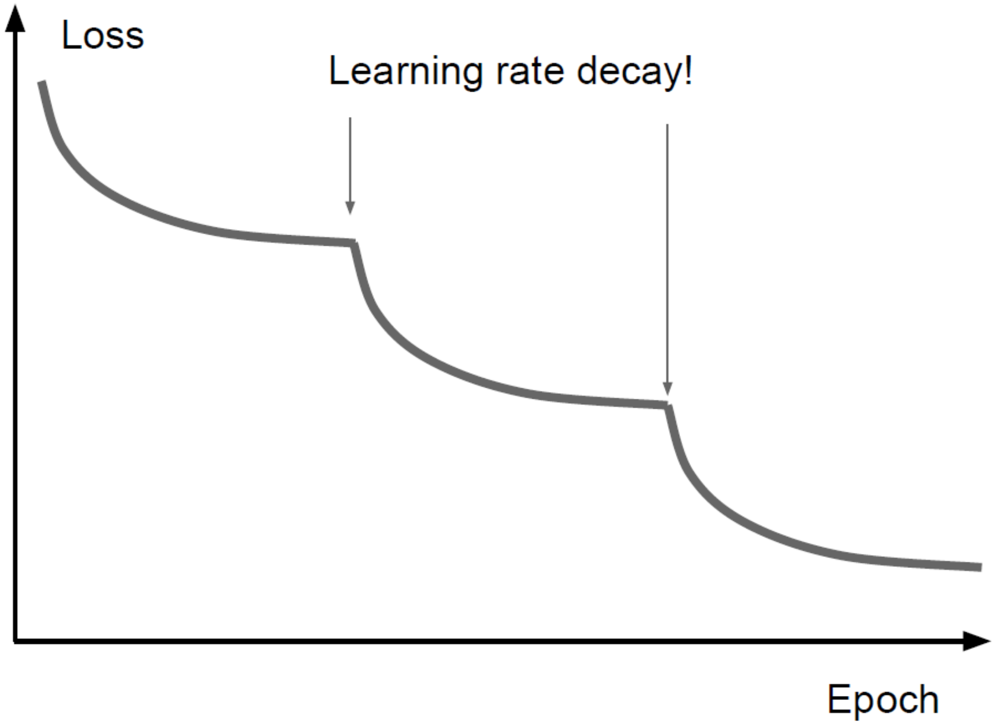
- Challenges:
  - Getting stuck in a local minimum, which is not as good as the global minimum
  - Taking very long to cross a plateau and unwillingly stopping too early before the global minimum is reached



Learning Rate

- How to get a good learning rate





Optimization Algorithms

- Adapt the learning rate to find the global minimum
- Build up "velocity" to overcome local minima and plateaus

(Stochastic) Gradient Descent

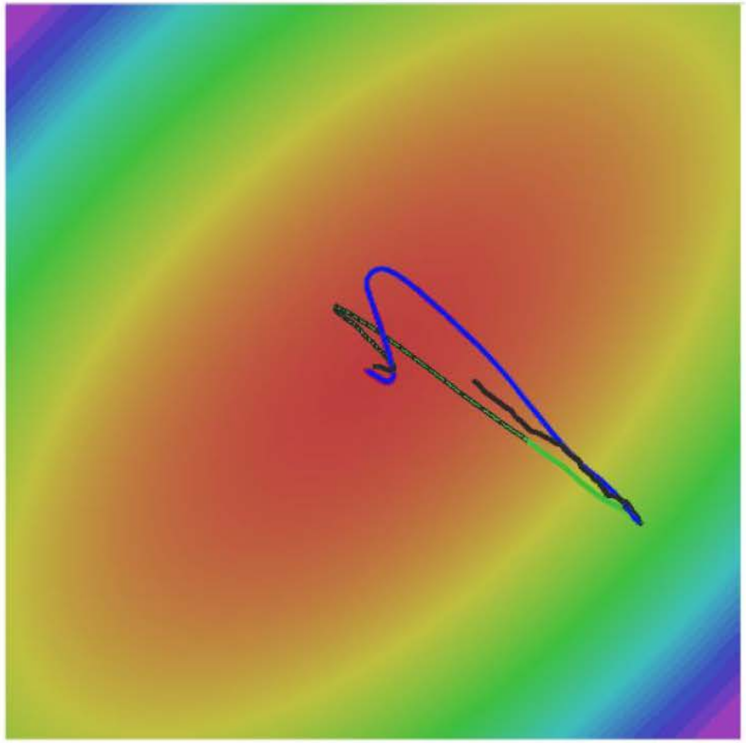
$$W_{t+1} = W_t - \eta \cdot \nabla_{w_t} L$$

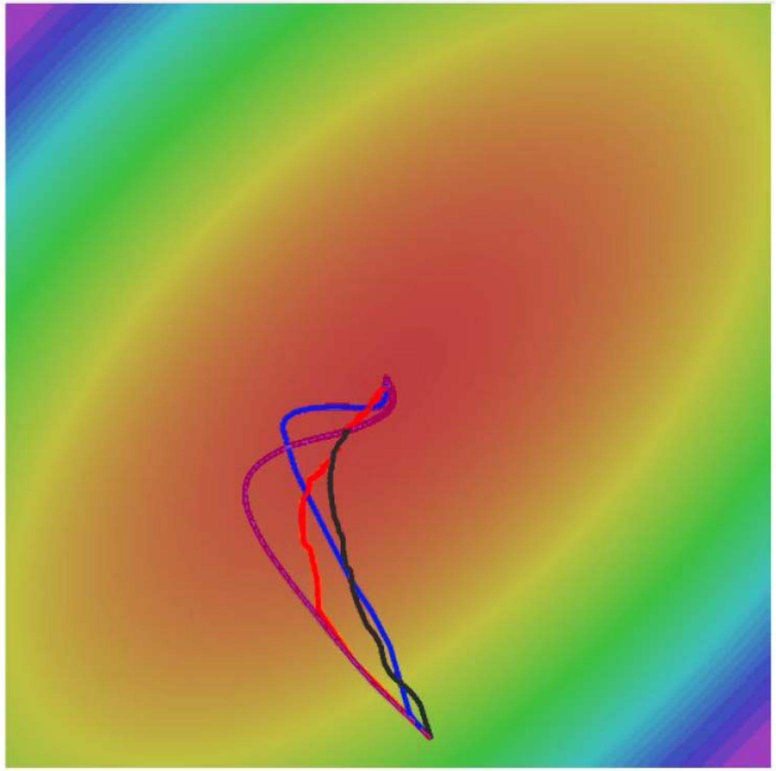
(Stochastic) Gradient Descent
+ Momentum

$$v_{t+1} = \rho v_t + \nabla_{w_t} L$$
$$W_{t+1} = W_t - \eta \cdot v_{t+1}$$

ρ gives "friction" (e.g. 0.9 or 0.99)

Optimization Algorithms

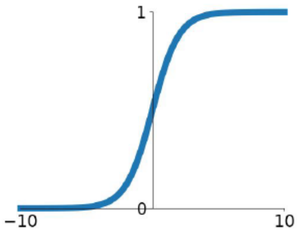




Activation Functions

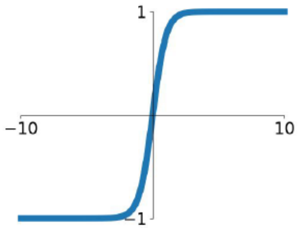
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



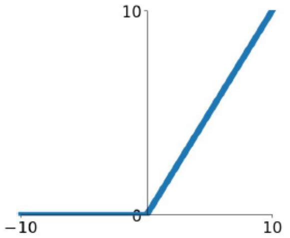
tanh

$\tanh(x)$



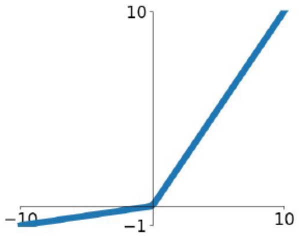
ReLU

$\max(0, x)$



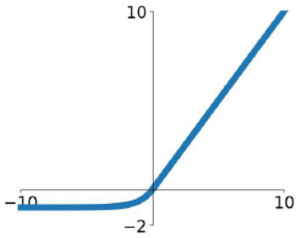
Leaky ReLU

$\max(0.1x, x)$



ELU

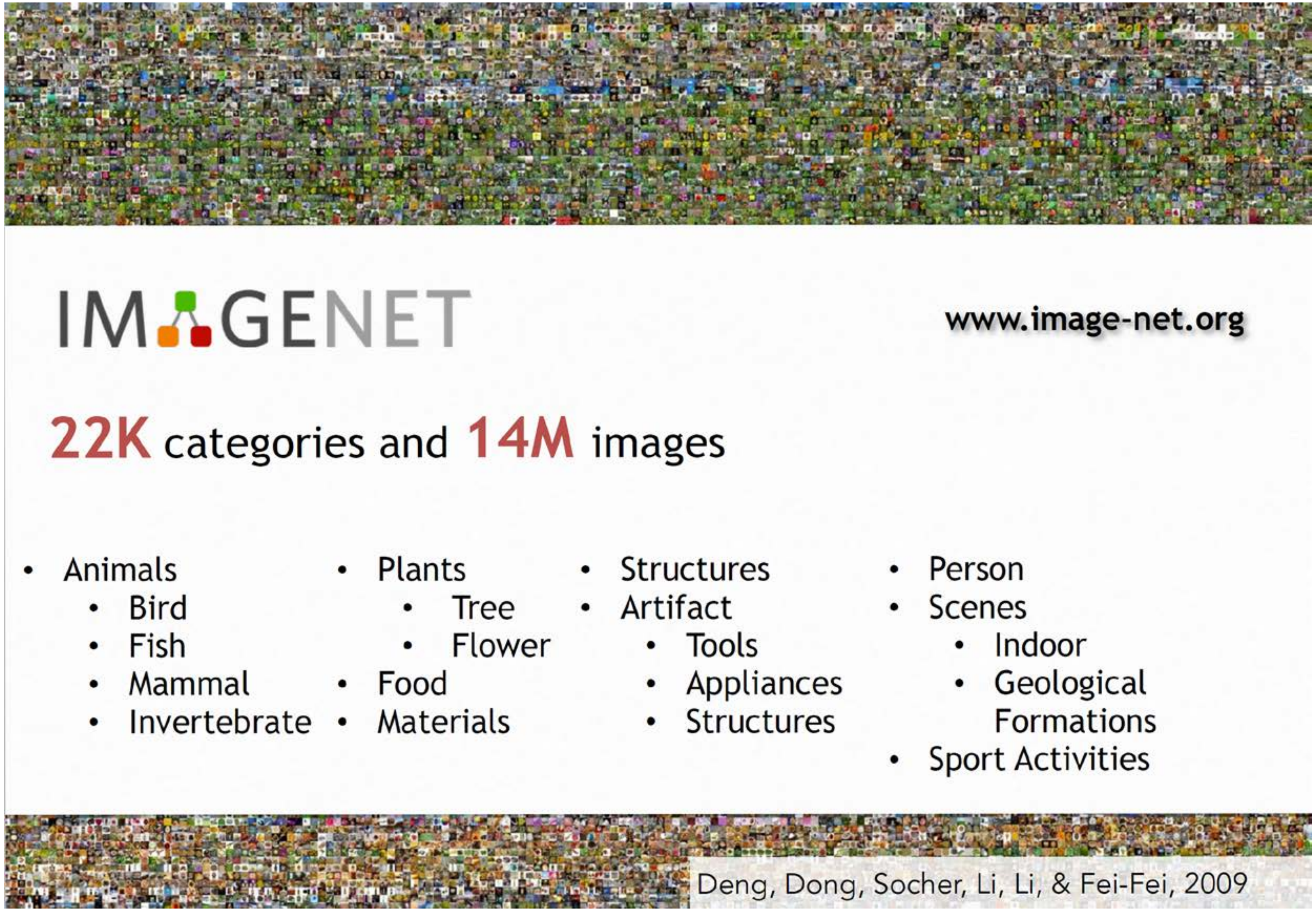
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



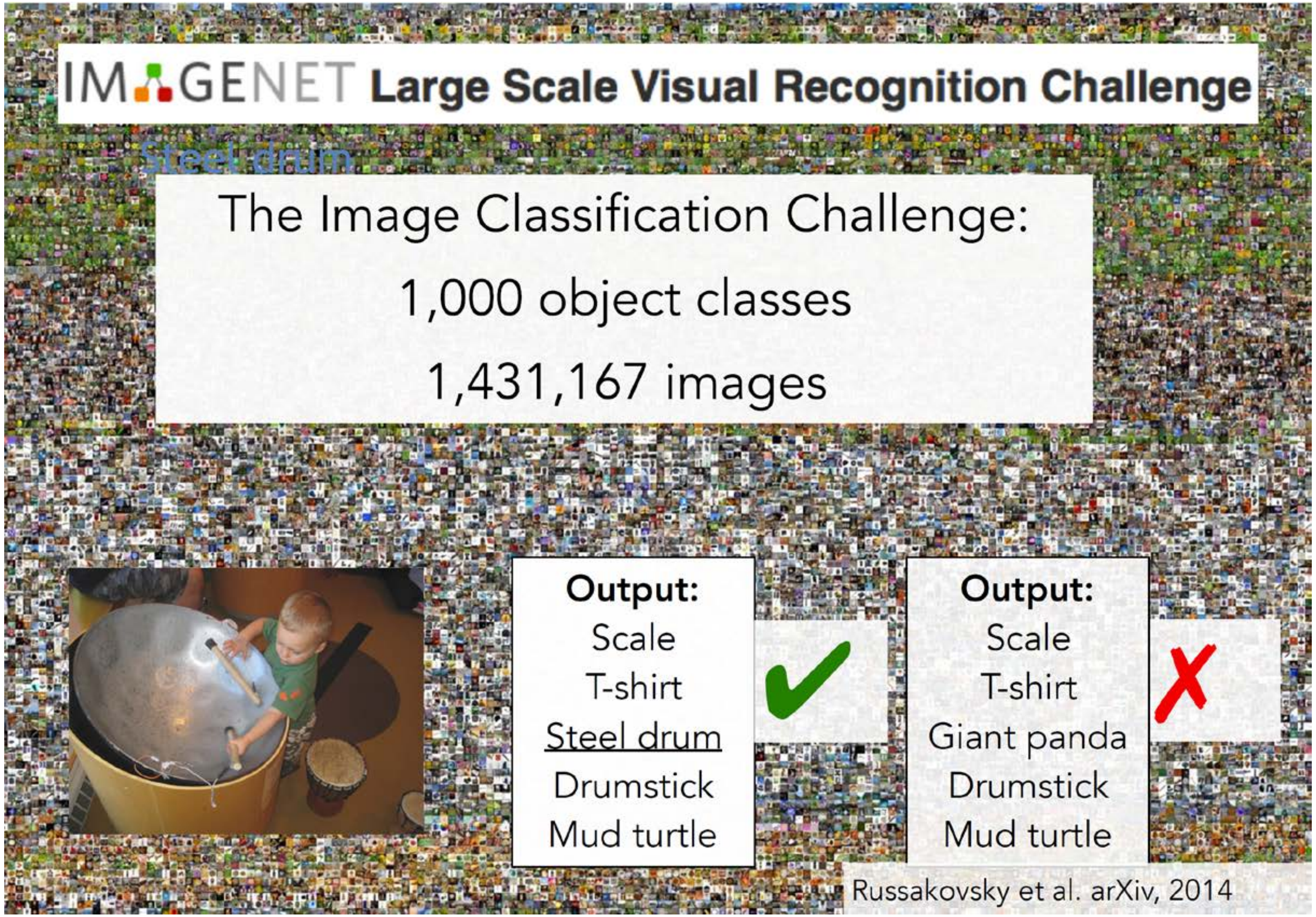
Convolutional Neural Networks

Prof. Dr. Martin Kada

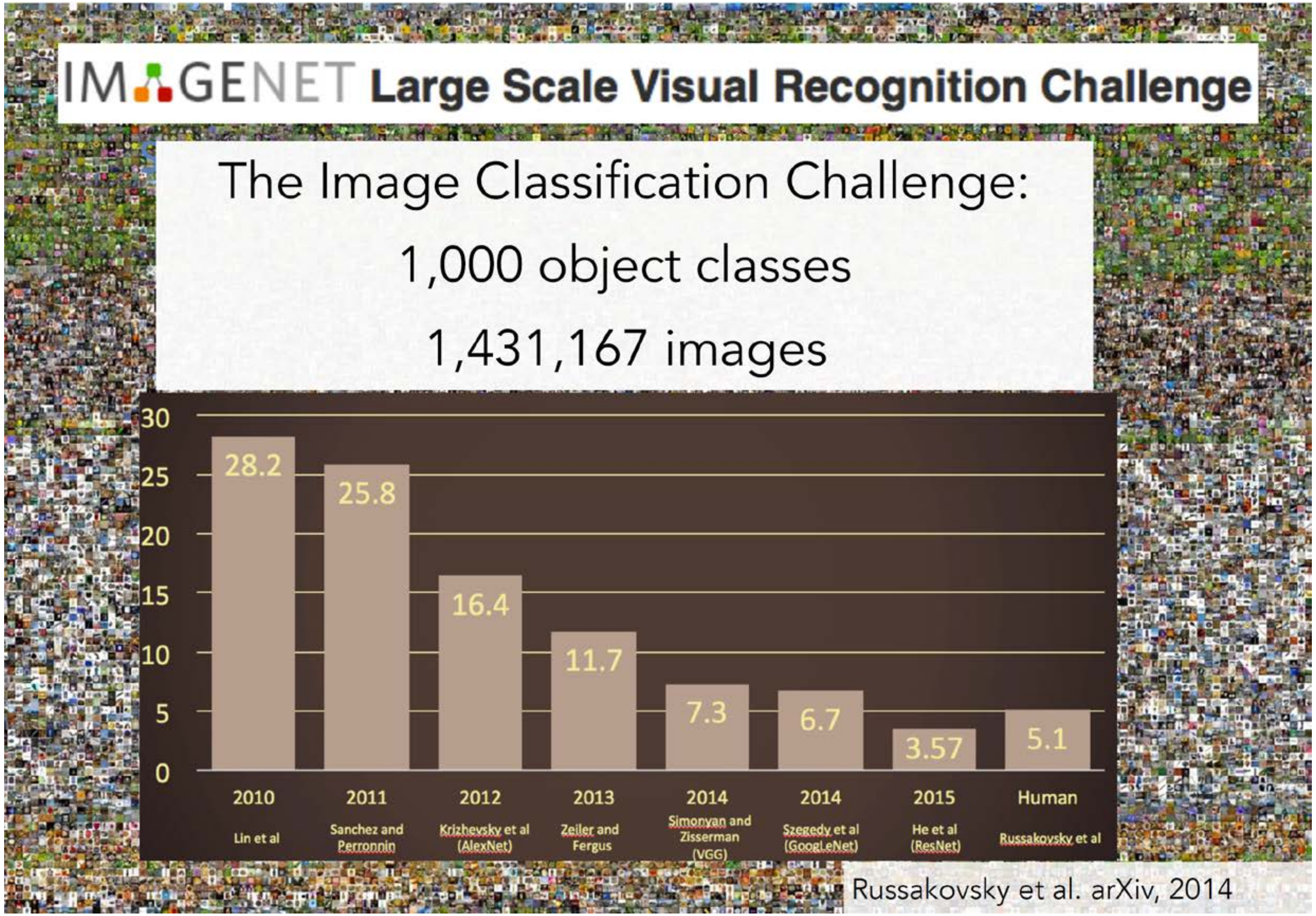
ImageNet


IMAGENET
www.image-net.org
22K categories and 14M images
Animals
Bird
Fish
Mammal
Invertebrate
Plants
Tree
Flower
Food
Materials
Structures
Artifact
Tools
Appliances
Structures
Person
Scenes
Indoor
Geological Formations
Sport Activities
Deng, Dong, Socher, Li, Li, & Fei-Fei, 2009

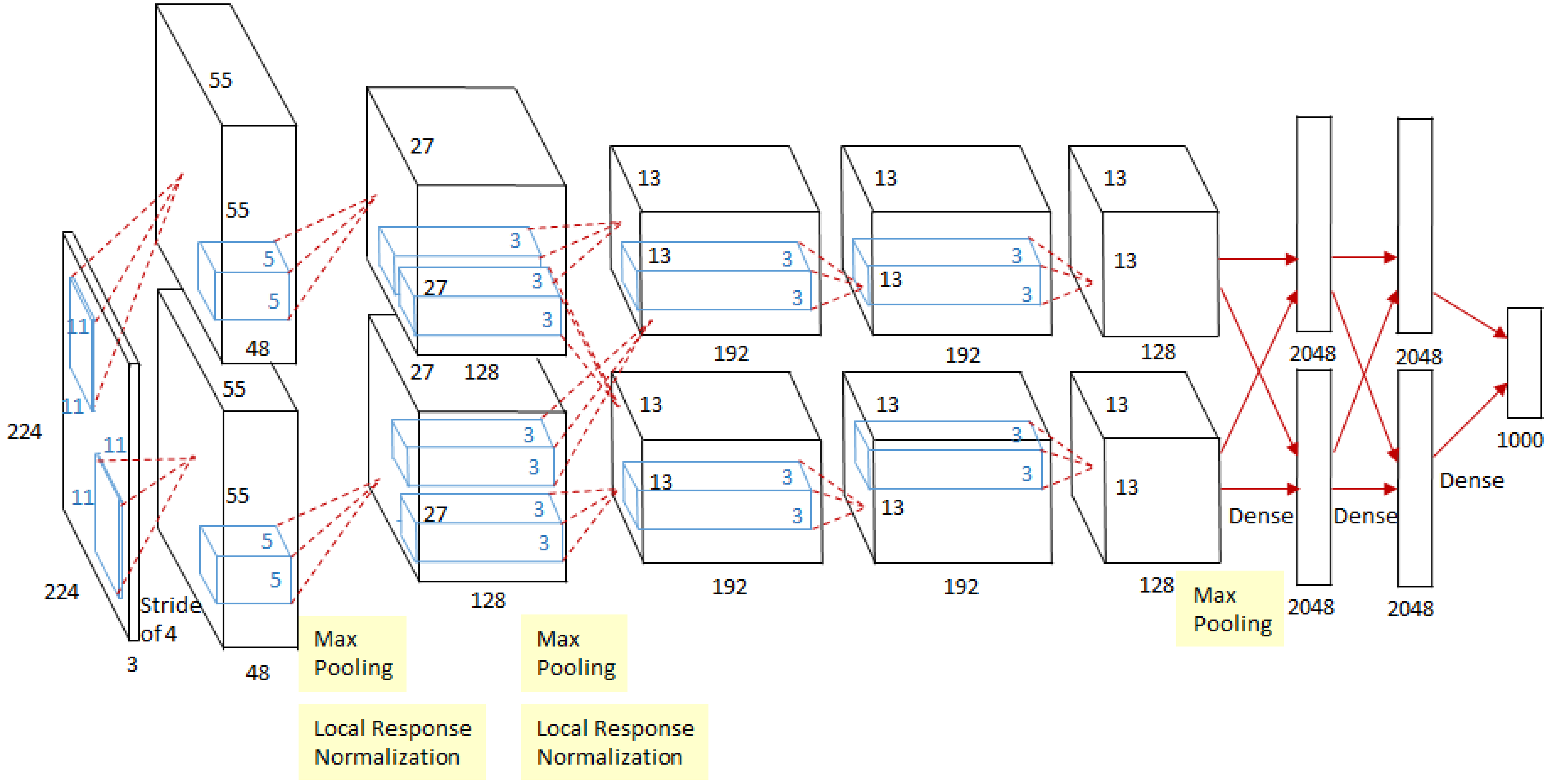
ImageNet


IMAGENET Large Scale Visual Recognition Challenge
Steel drum
The Image Classification Challenge:
1,000 object classes
1,431,167 images
Output:
Scale
T-shirt
Steel drum
Drumstick
Mud turtle
Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle
Russakovsky et al. arXiv, 2014

ImageNet


IMAGENET Large Scale Visual Recognition Challenge
The Image Classification Challenge:
1,000 object classes
1,431,167 images
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.57
5.1
2010
Lin et al
2011
Sanchez and Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler and Fergus
2014
Simonyan and Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
Human
Russakovsky et al
Russakovsky et al. arXiv, 2014

AlexNet


55
55
48
55
55
48
224
224
3
11
11
11
11
Stride
of 4
5
5
5
5
27
27
128
27
128
27
3
3
3
3
3
3
3
3
13
13
192
13
13
192
13
13
192
13
13
192
13
13
128
13
13
128
2048
2048
2048
2048
1000
Dense
Dense
Dense
Max
Pooling
Local Response
Normalization
Max
Pooling
Local Response
Normalization
Max
Pooling

VGG

[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

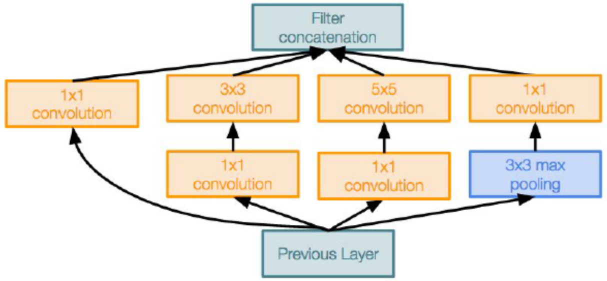
11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14

GoogLeNet

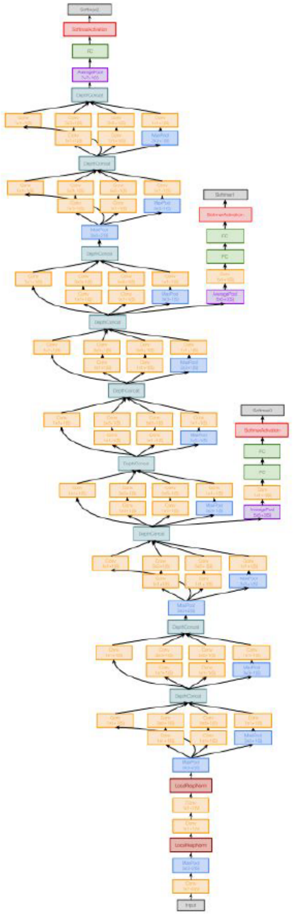
[Szegedy et al., 2014]

Deeper networks, with computational efficiency

- 22 layers
- Efficient “Inception” module
- No FC layers
- Only 5 million parameters!
  12x less than AlexNet
- ILSVRC’14 classification winner
  (6.7% top 5 error)



Inception module



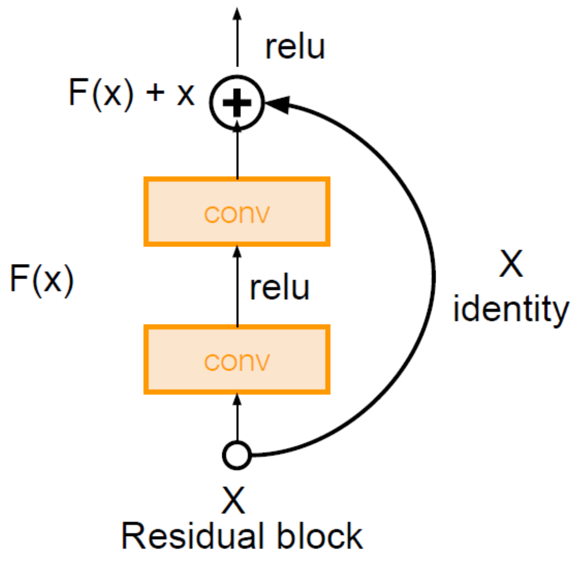
Hint: only the convolutional layers and the fully connected layer (at the head of the network) that predicts the class scores are counted into the 22 layers

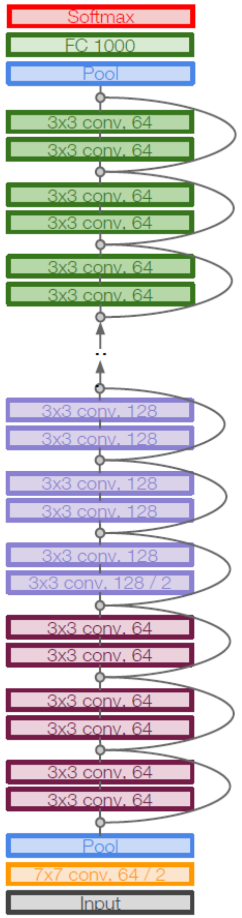
ResNet

[He et al., 2015]

Very deep networks using residual connections

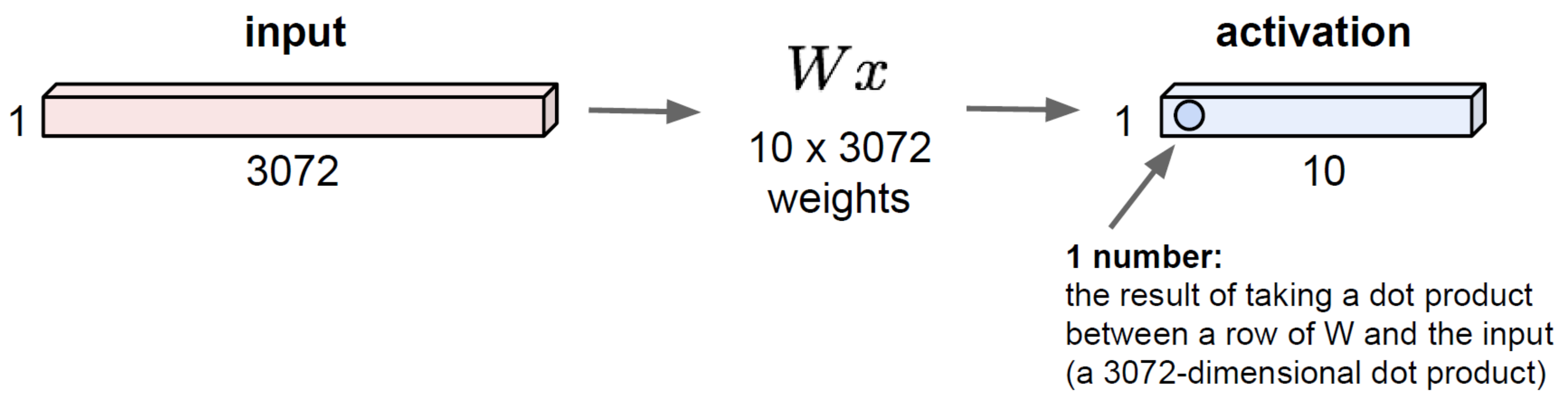
- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!





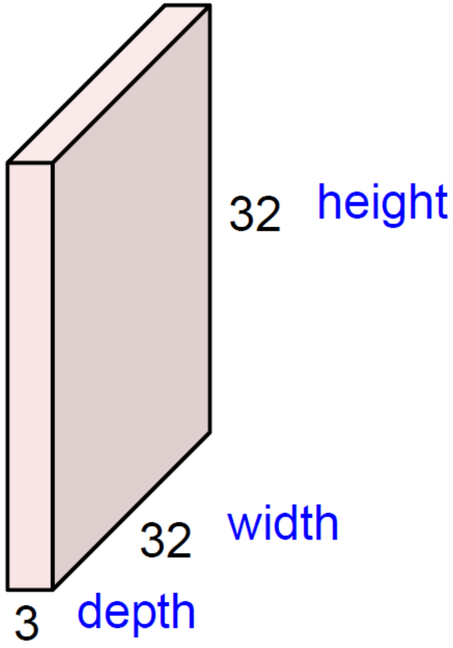
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

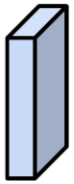


Convolution Layer

32x32x3 image -> preserve spatial structure



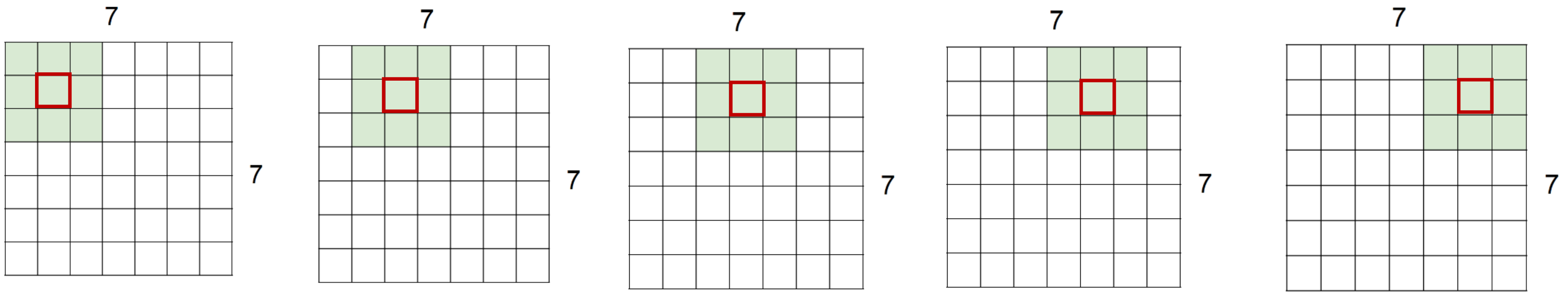
5x5x3 filter



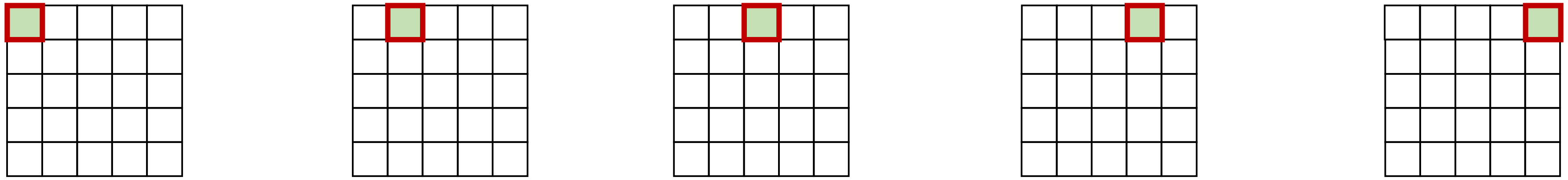
Convolve the filter with the image
i.e. "slide over the image spatially,
computing dot products"

Convolution Layer

7x7 input (spatially)
assume 3x3 filter



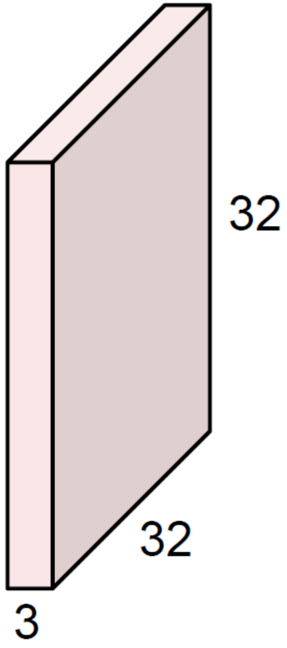
7×7 input with 3×3 filter $\Rightarrow$ 5×5 output



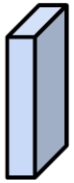
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

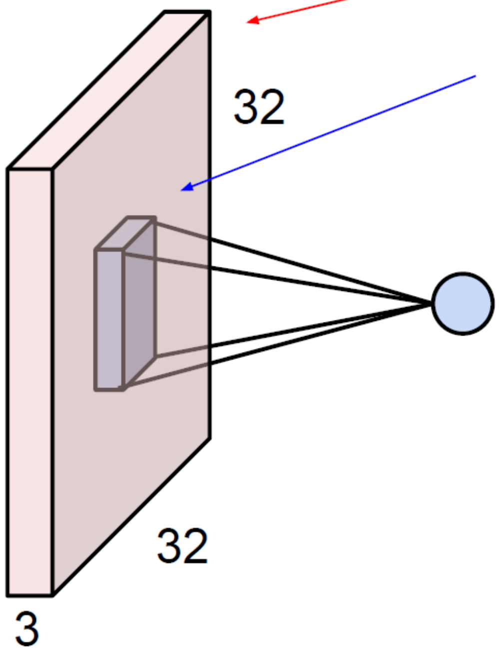


5x5x3 filter



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

Convolution Layer



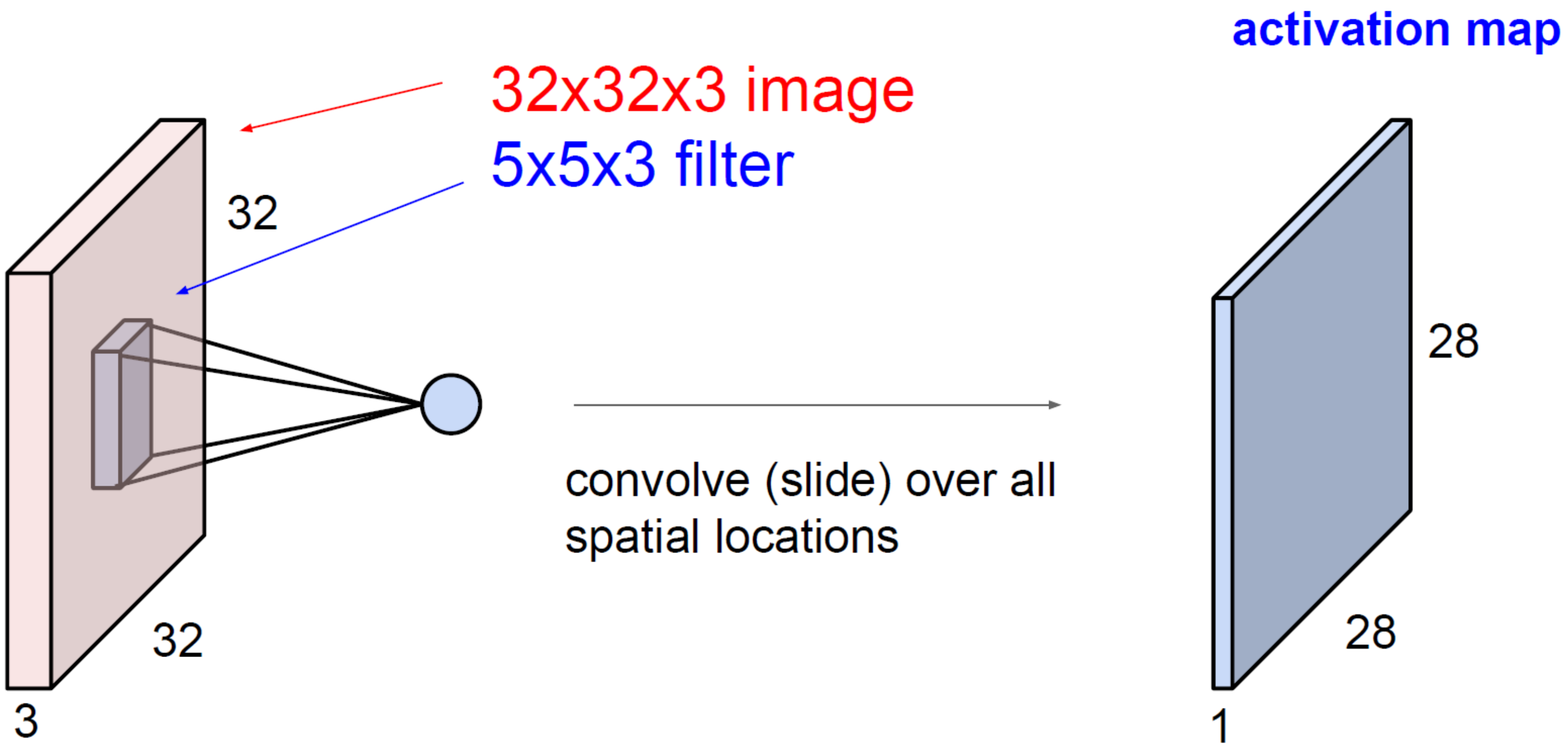
32x32x3 image

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)

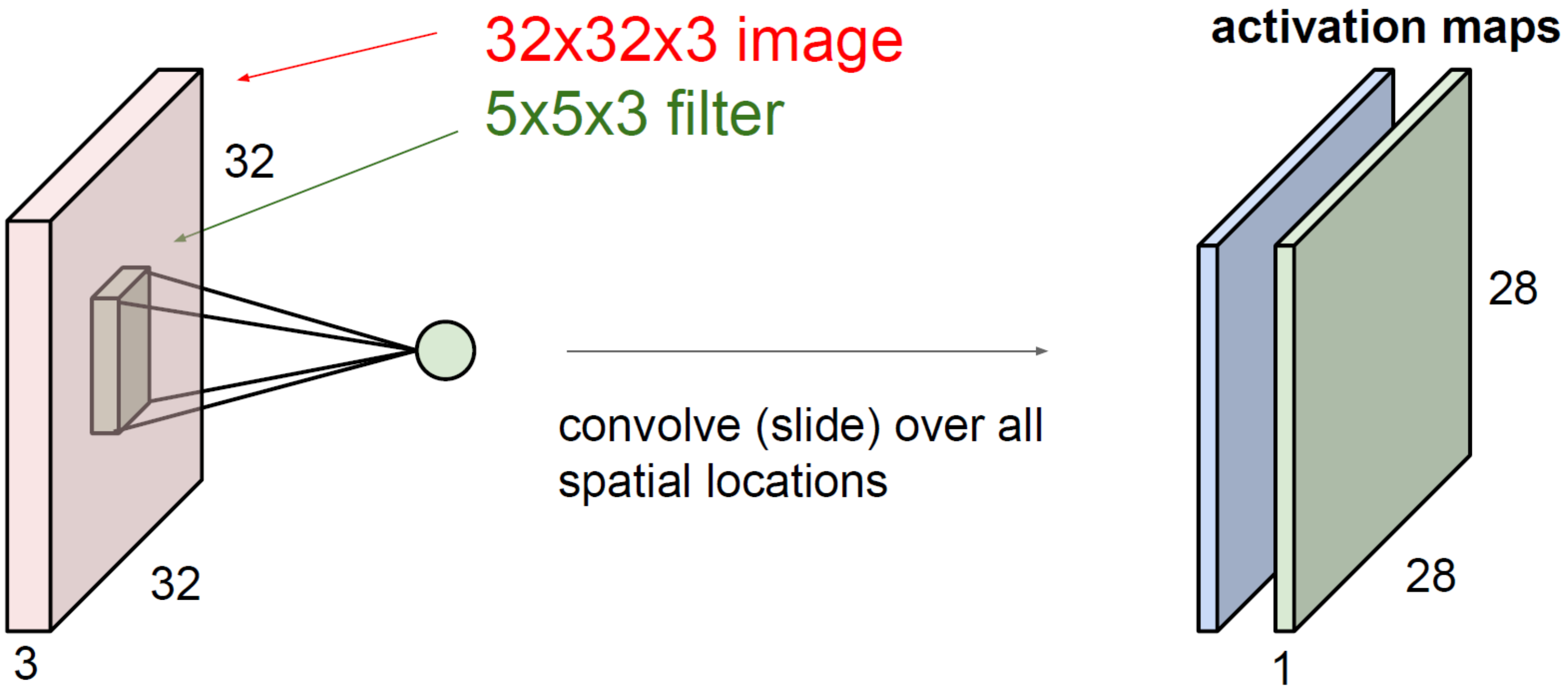
$$w^T x + b$$

Convolution Layer



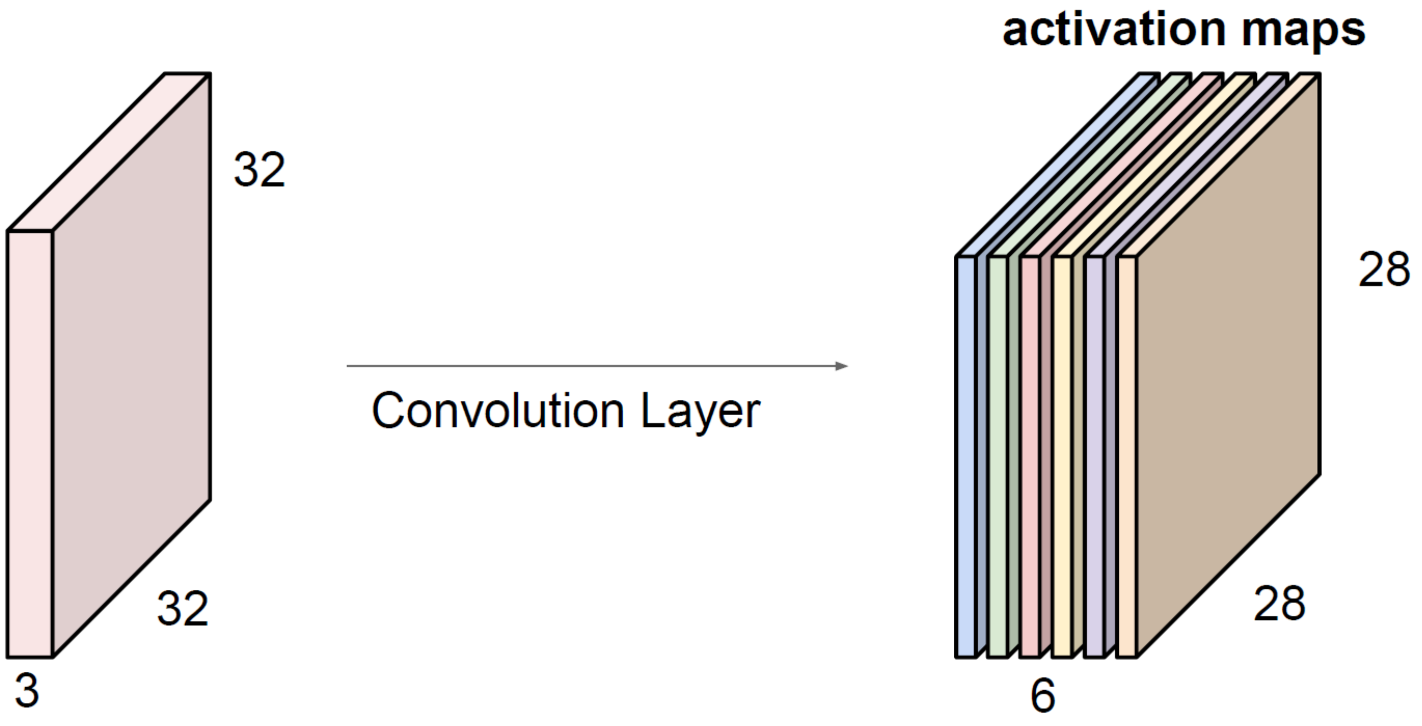
Convolution Layer

consider a second, green filter



Activation Maps

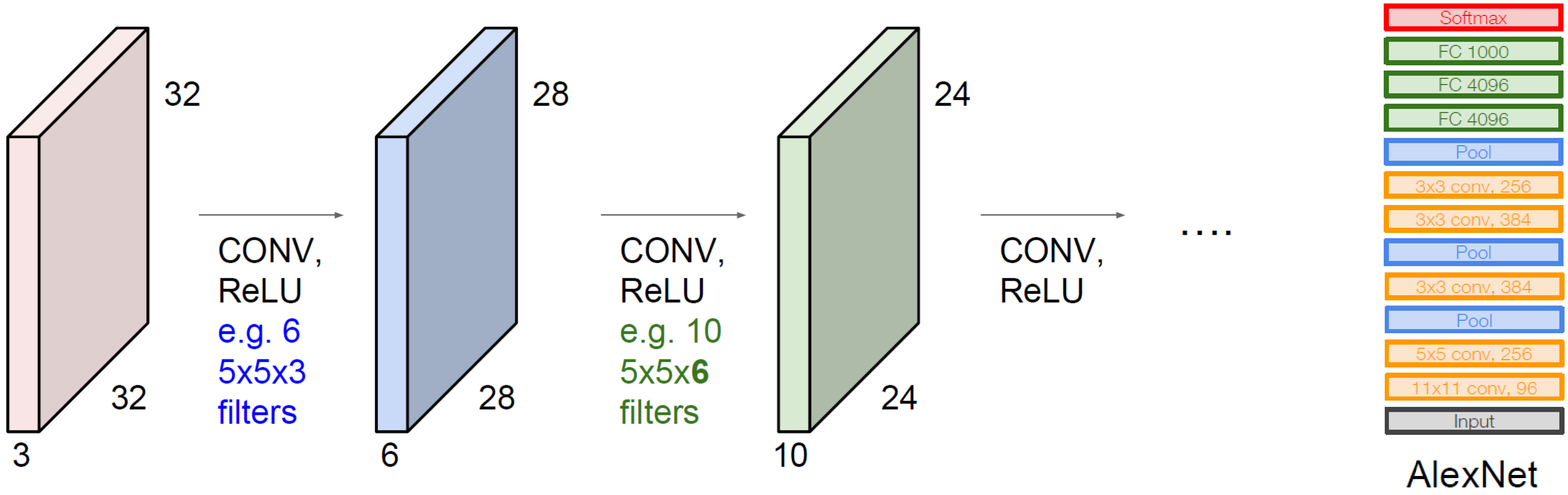
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



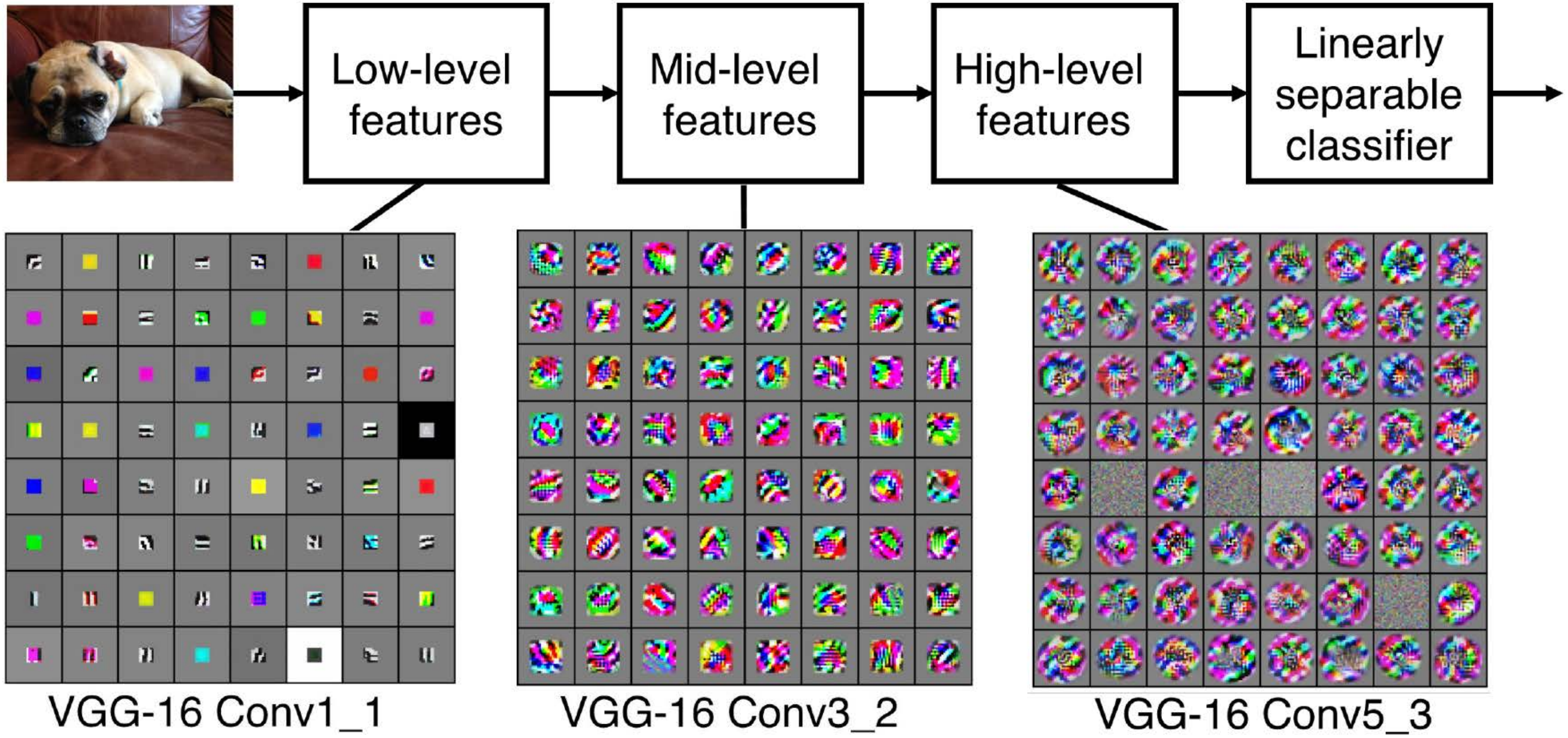
We stack these up to get a "new image" of size 28x28x6!

Convolutional Neural Network

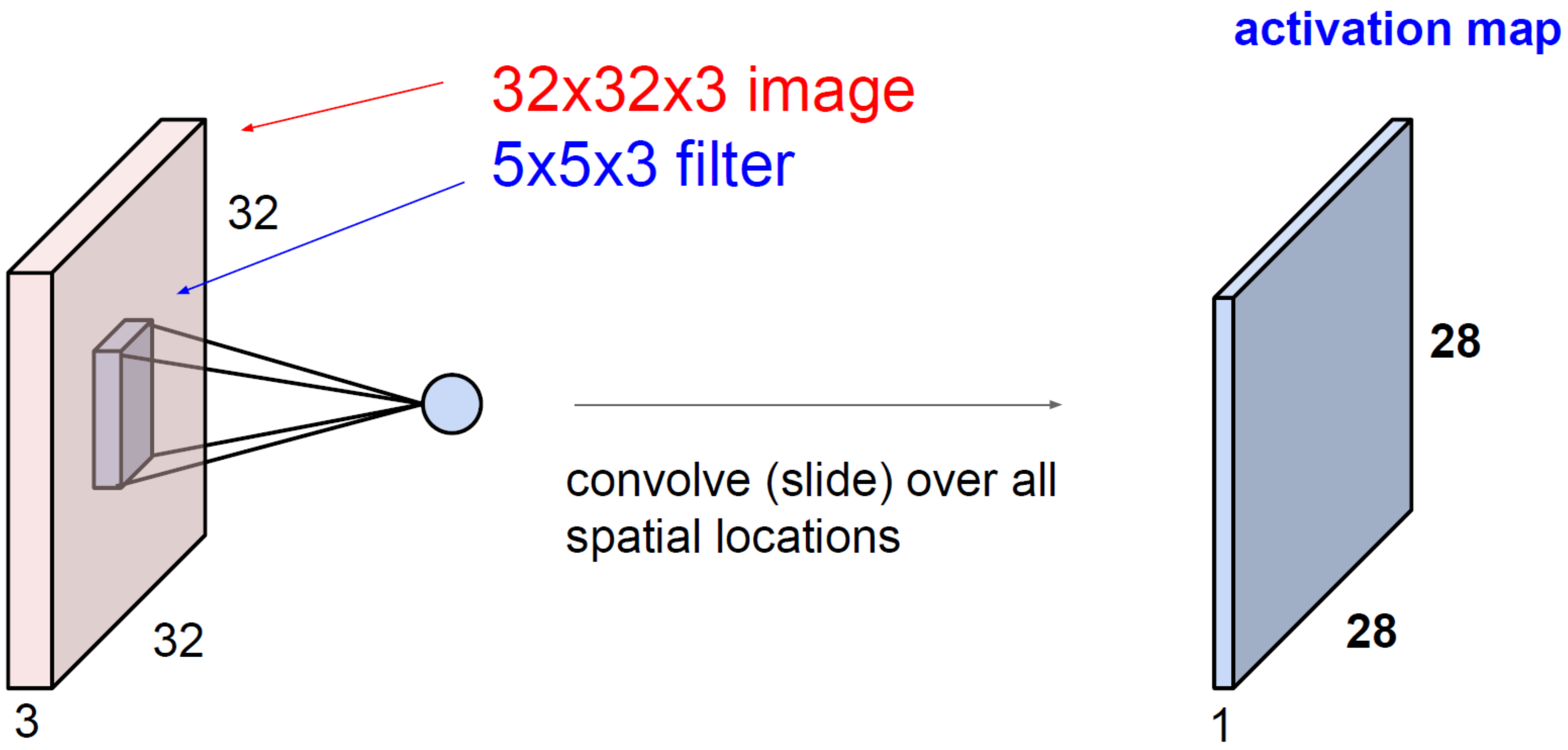
- A Convolutional Neural Network (CNN) is a sequence of Convolutional Layers, interspersed with activation functions



Convolutional Neural Network

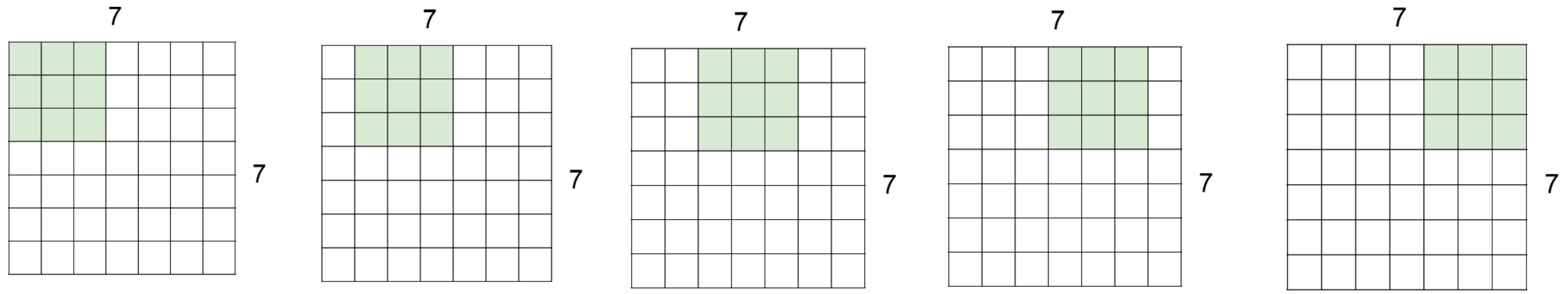


Stride



activation map
32x32x3 image
5x5x3 filter
32
32
3
convolve (slide) over all
spatial locations
28
28
1

Stride

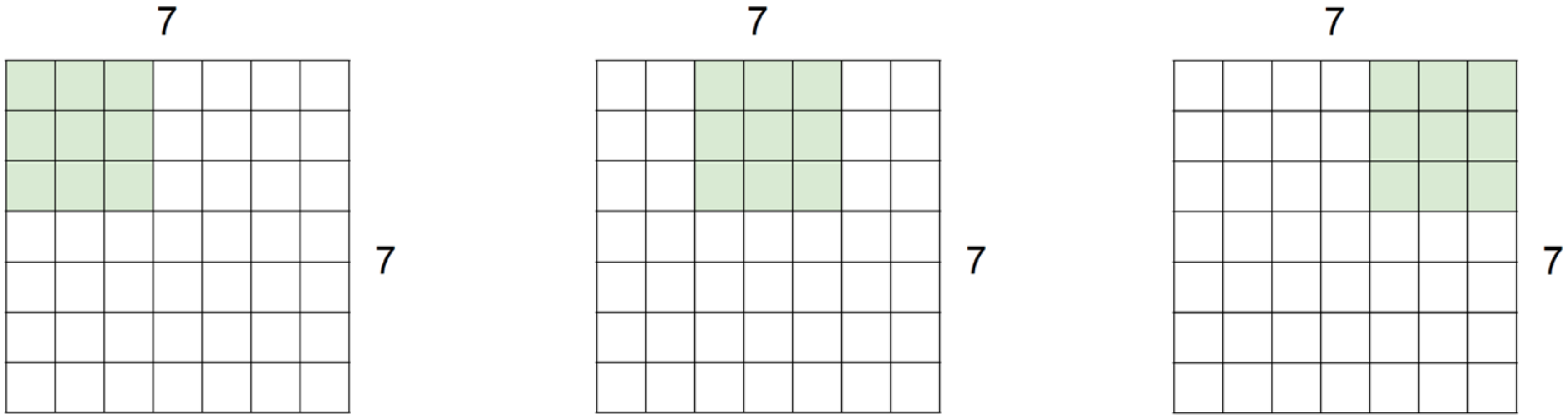


7x7 input (spatially)
assume 3x3 filter

applied **with stride 1**

=> 5x5 output

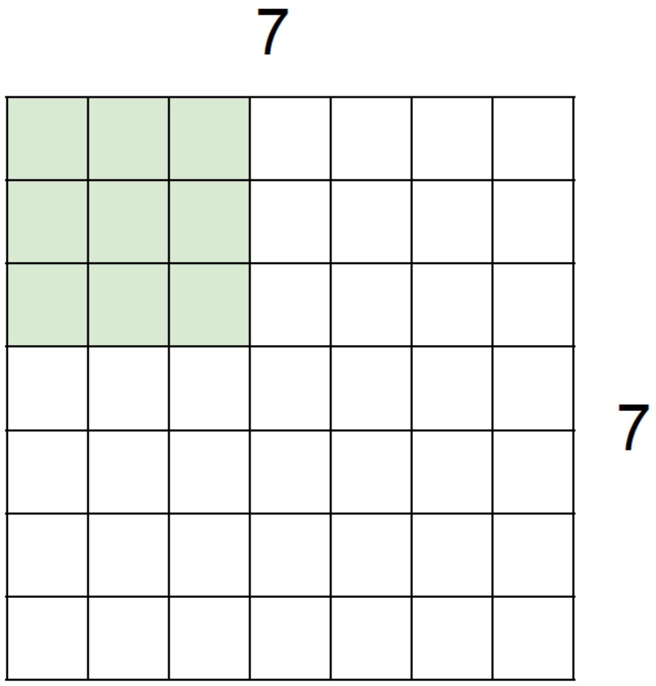
Stride



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

=> 3x3 output!

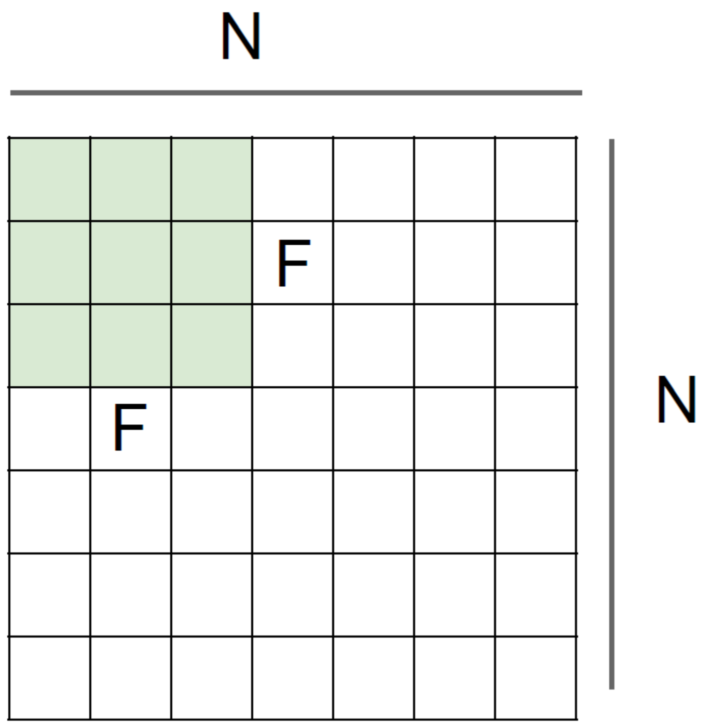
Stride



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

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

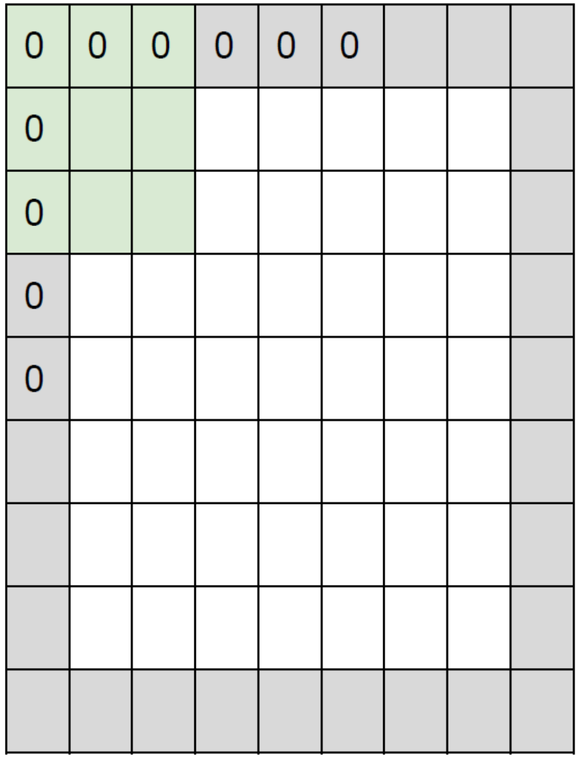
Stride



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 :\

Padding



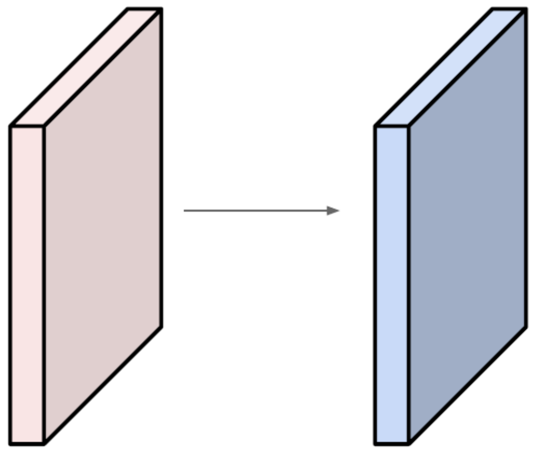
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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

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

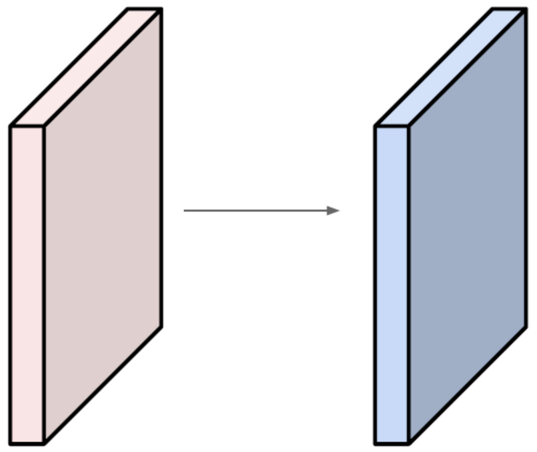
Examples



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



Input volume: **32x32x3**

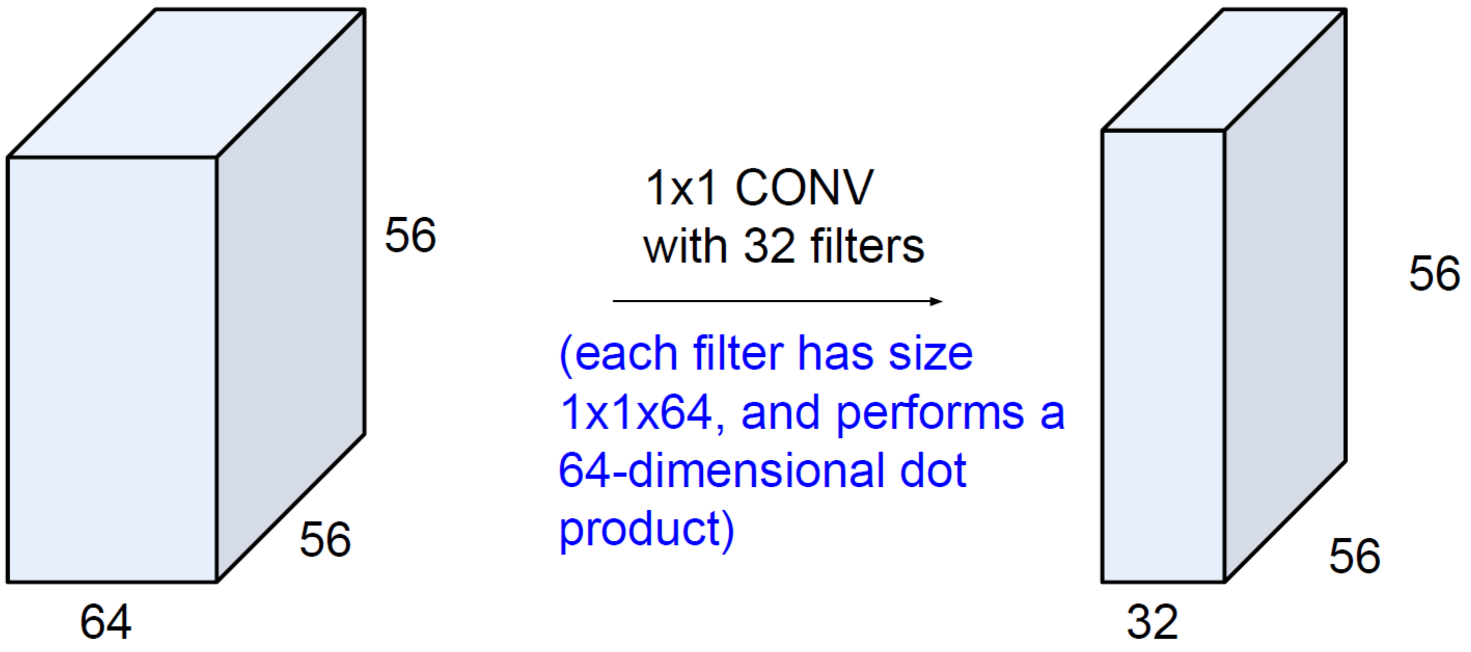
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

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

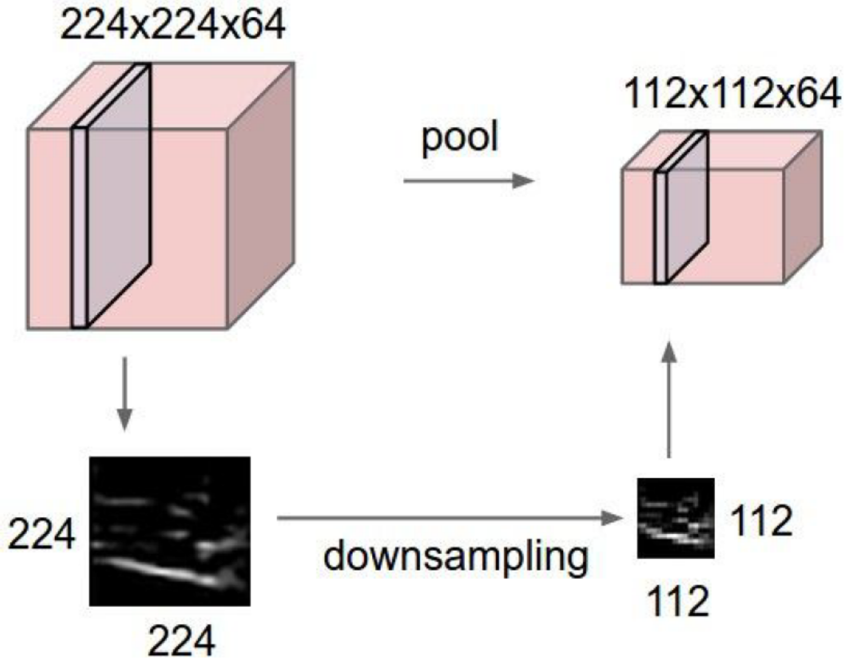
=> 76*10 = **760**

1×1 Convolution

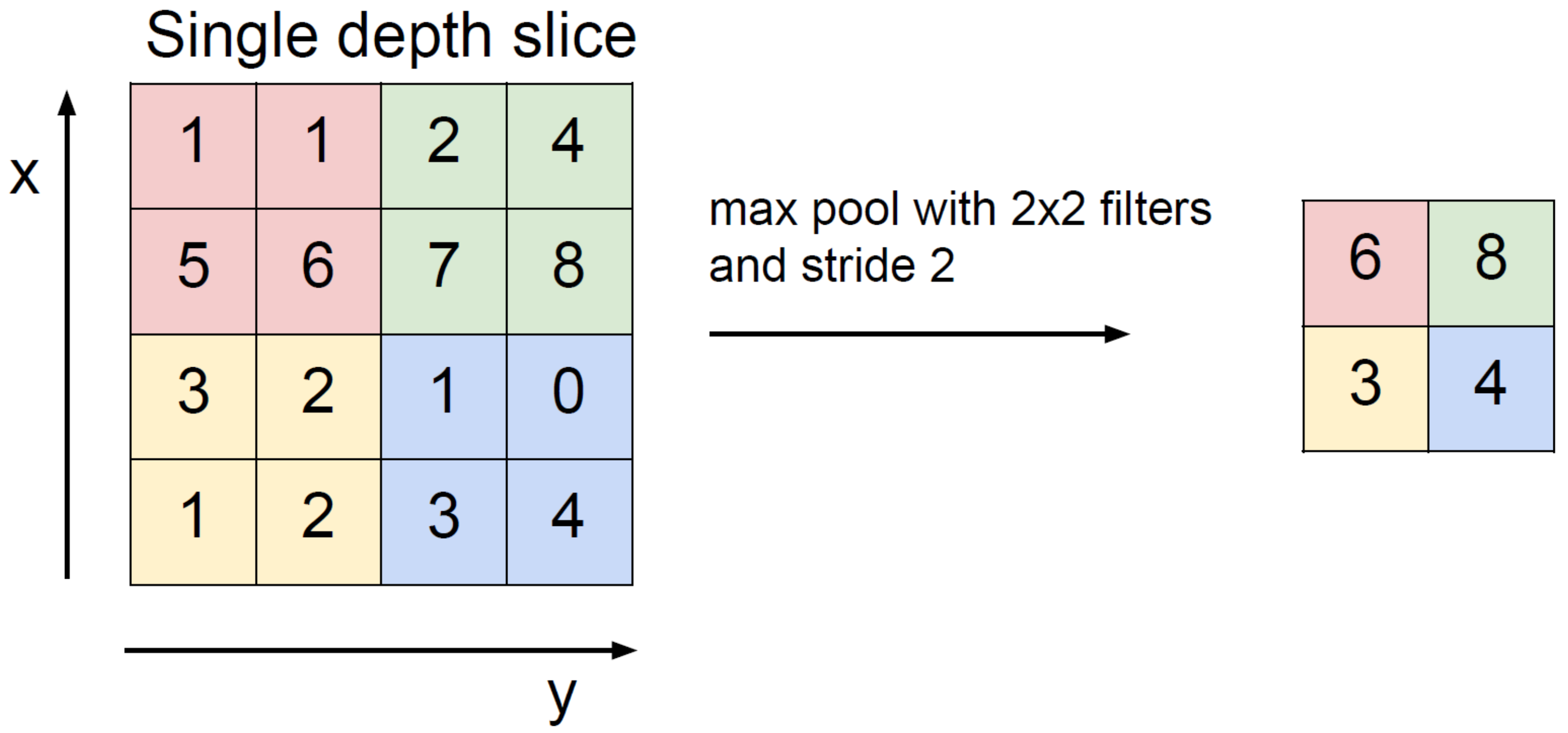


Pooling Layer

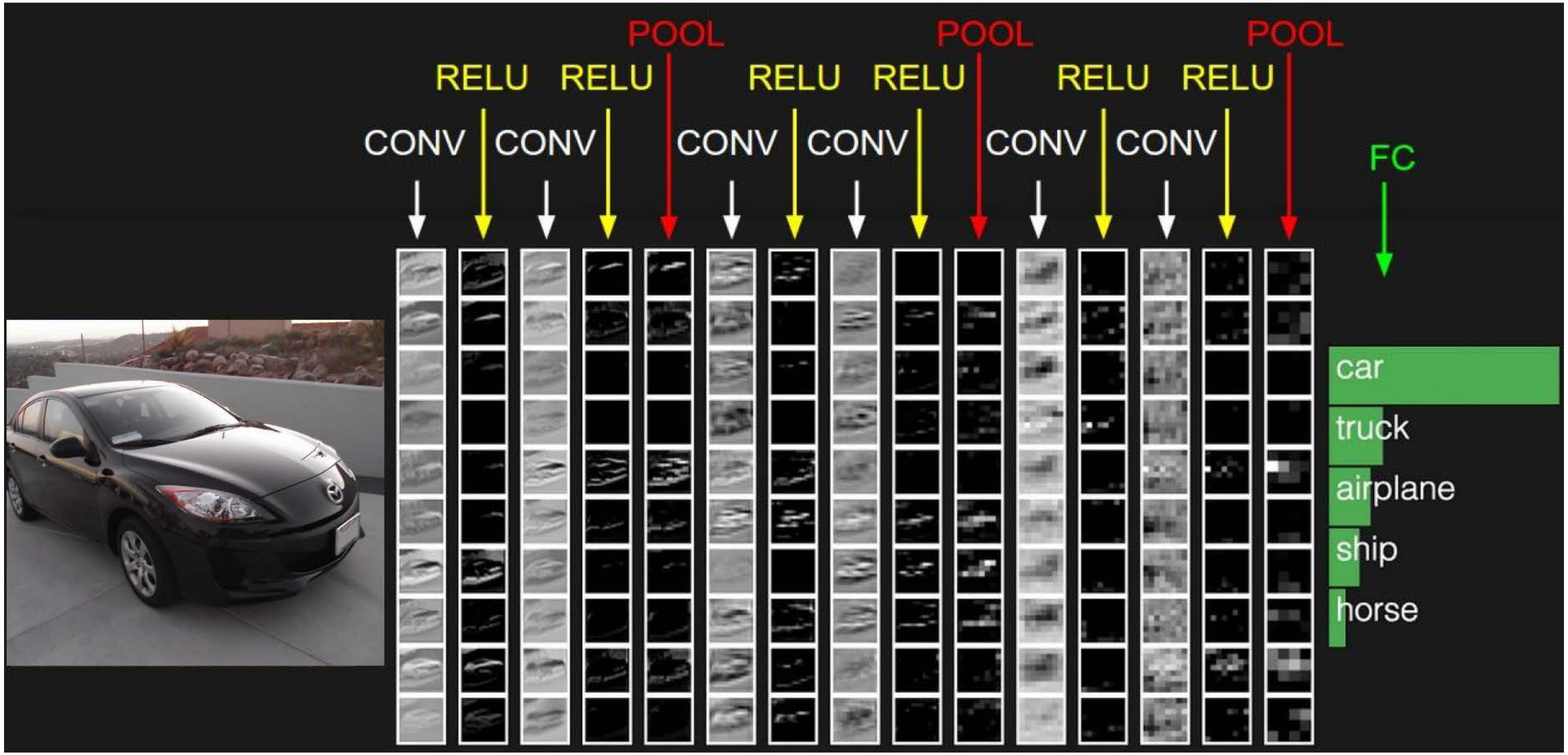
- makes the representations smaller and more manageable
- operates over each activation map independently:



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

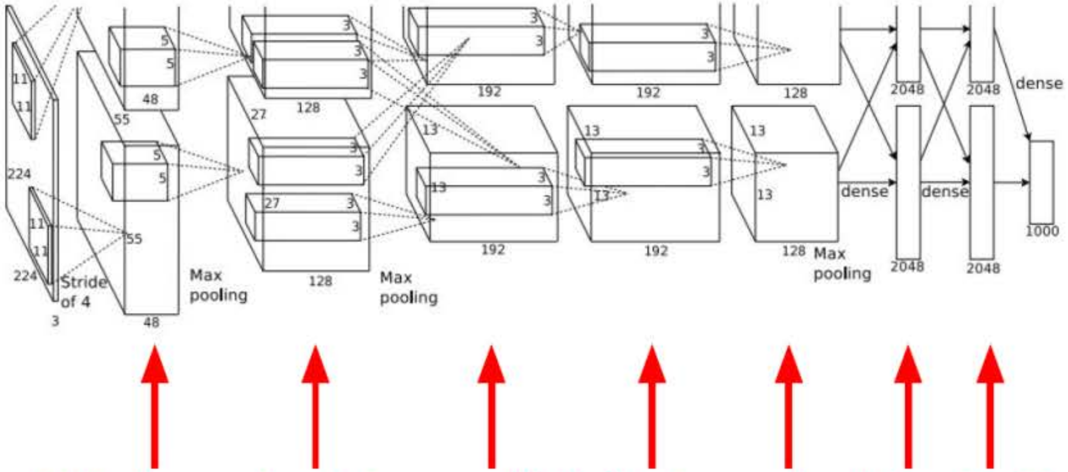
CNN for Image Classification



What’s going on inside ConvNets?

This image is CC0 public domain

Input Image:
3 x 224 x 224

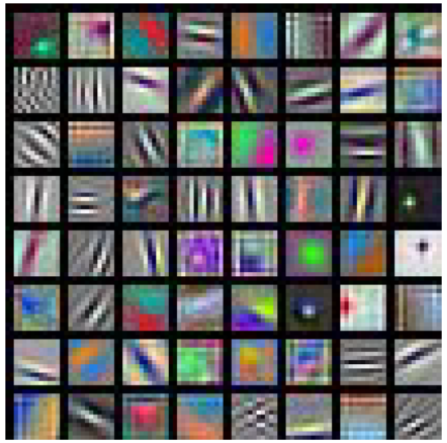


Class Scores:
1000 numbers

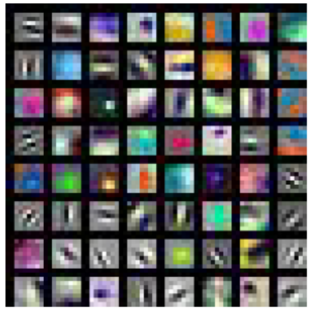
What are the intermediate features looking for?

Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figure reproduced with permission.

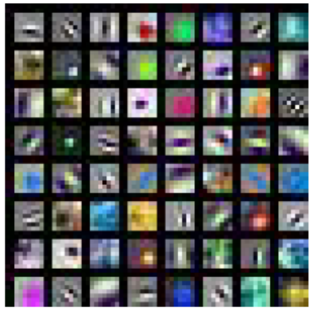
What’s going on inside ConvNets?



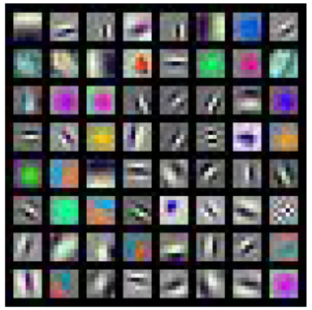
AlexNet:
64 x 3 x 11 x 11



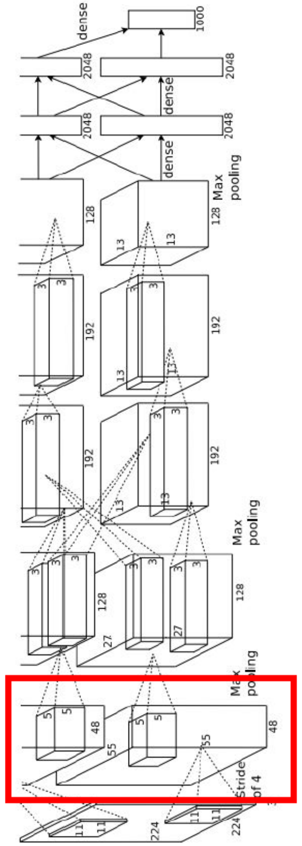
ResNet-18:
64 x 3 x 7 x 7



ResNet-101:
64 x 3 x 7 x 7

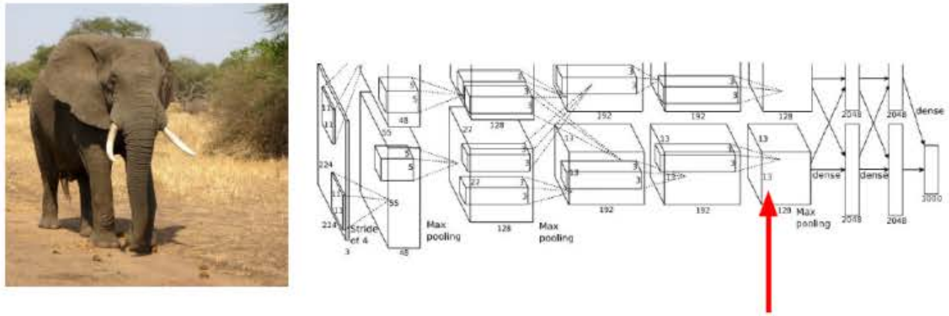


DenseNet-121:
64 x 3 x 7 x 7



Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv 2014
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016
Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017

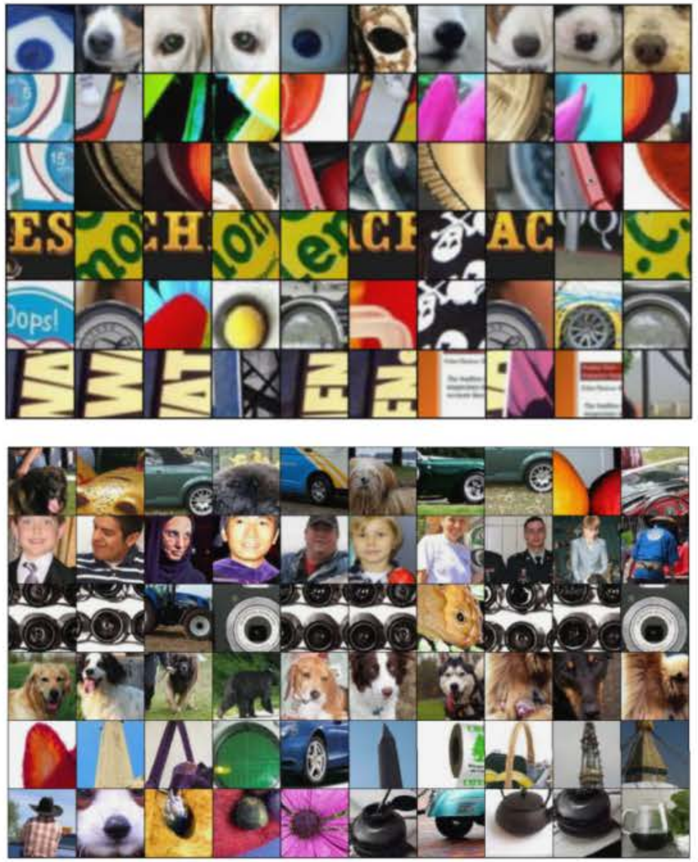
What’s going on inside ConvNets?



Pick a layer and a channel; e.g. conv5 is 128 x 13 x 13, pick channel 17/128

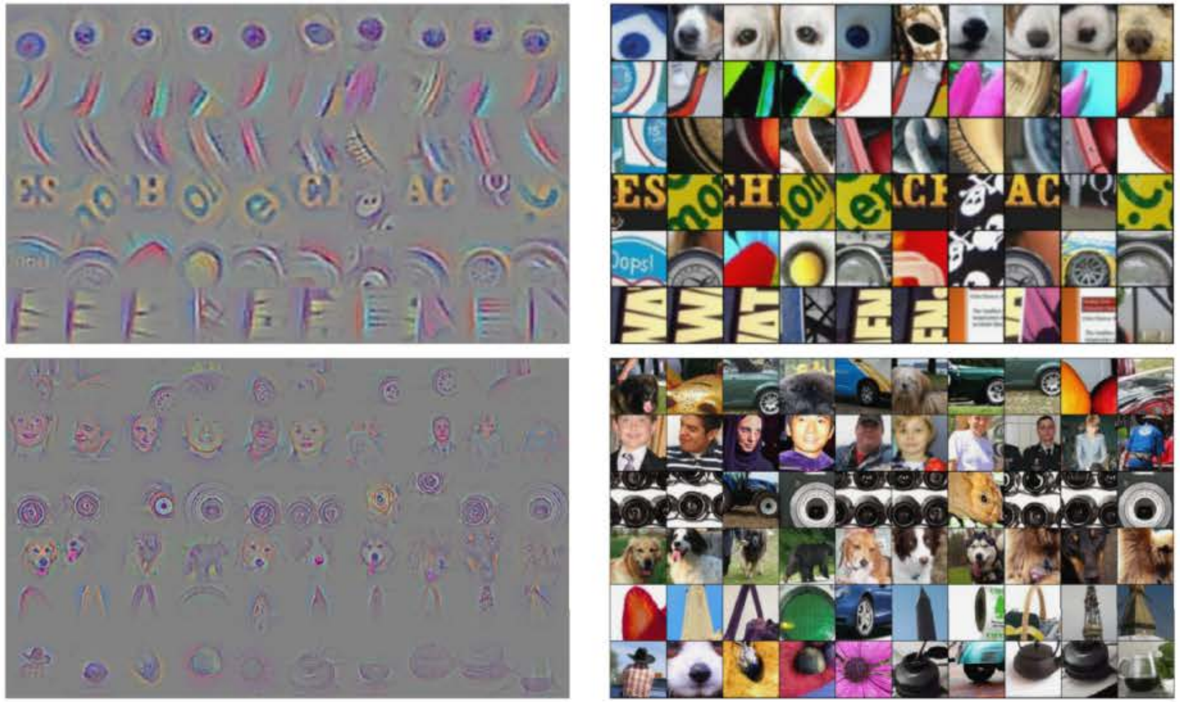
Run many images through the network, record values of chosen channel

Visualize image patches that correspond to maximal activations



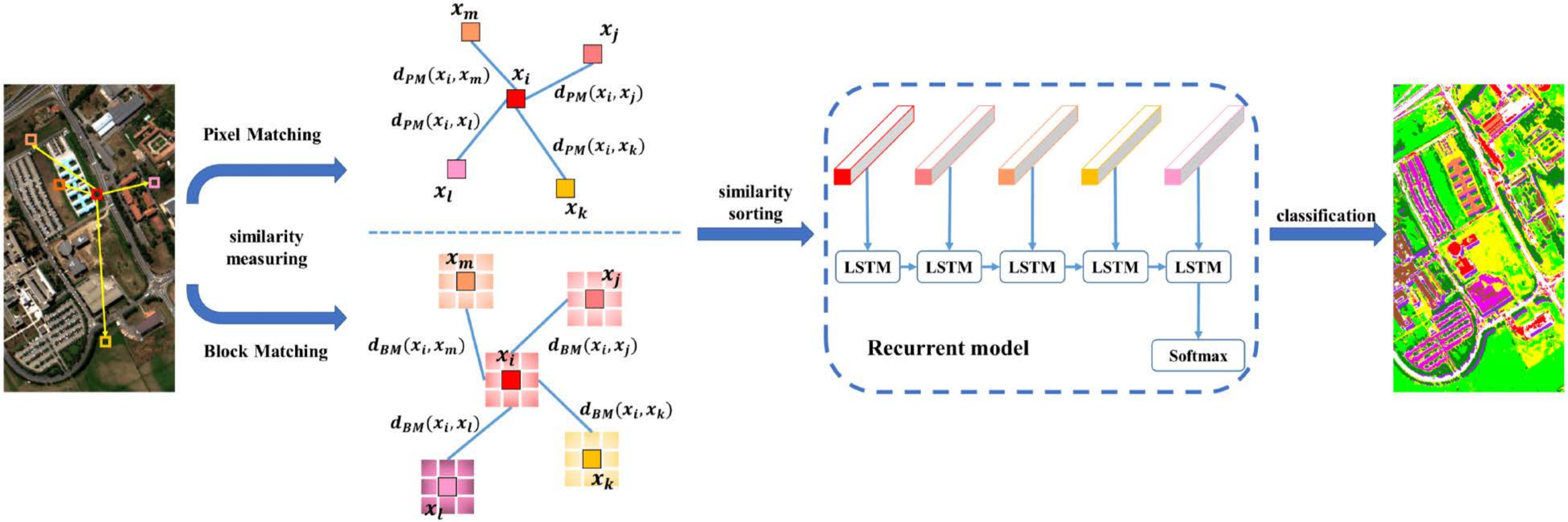
Springenberg et al, “Striving for Simplicity: The All Convolutional Net”, ICLR Workshop 2015

What's going on inside ConvNets?

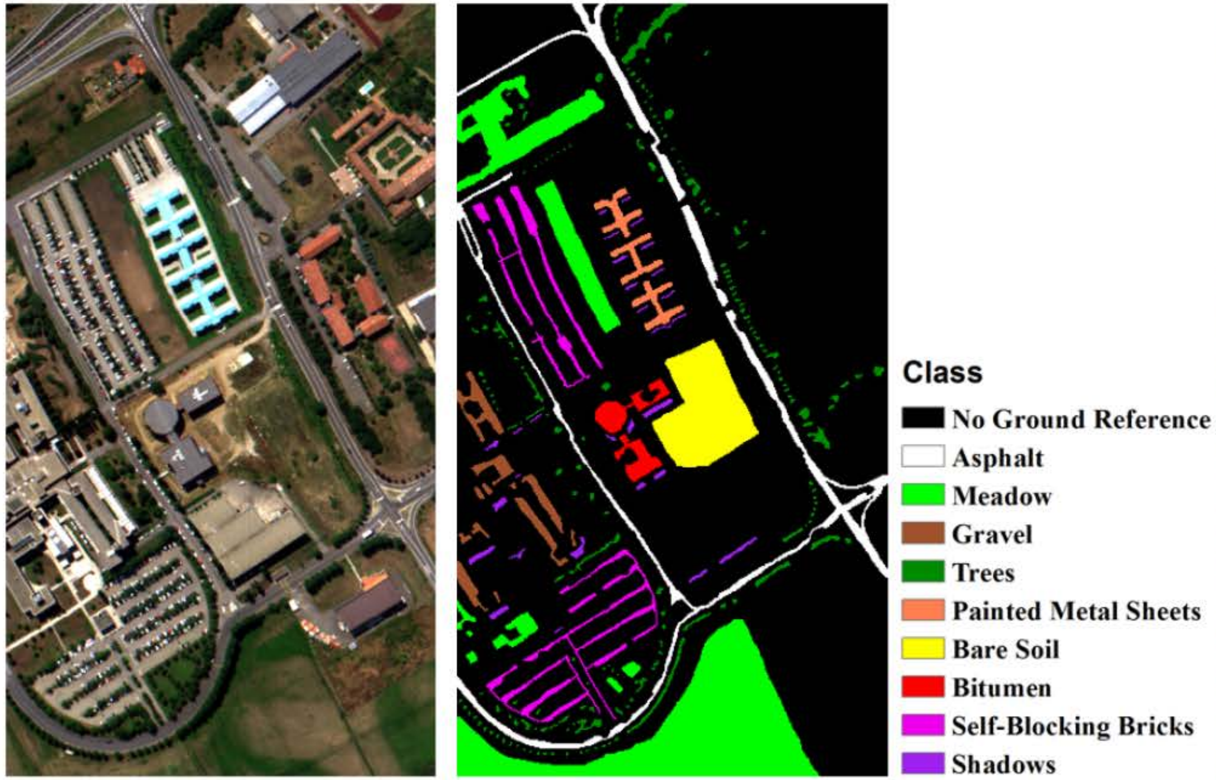


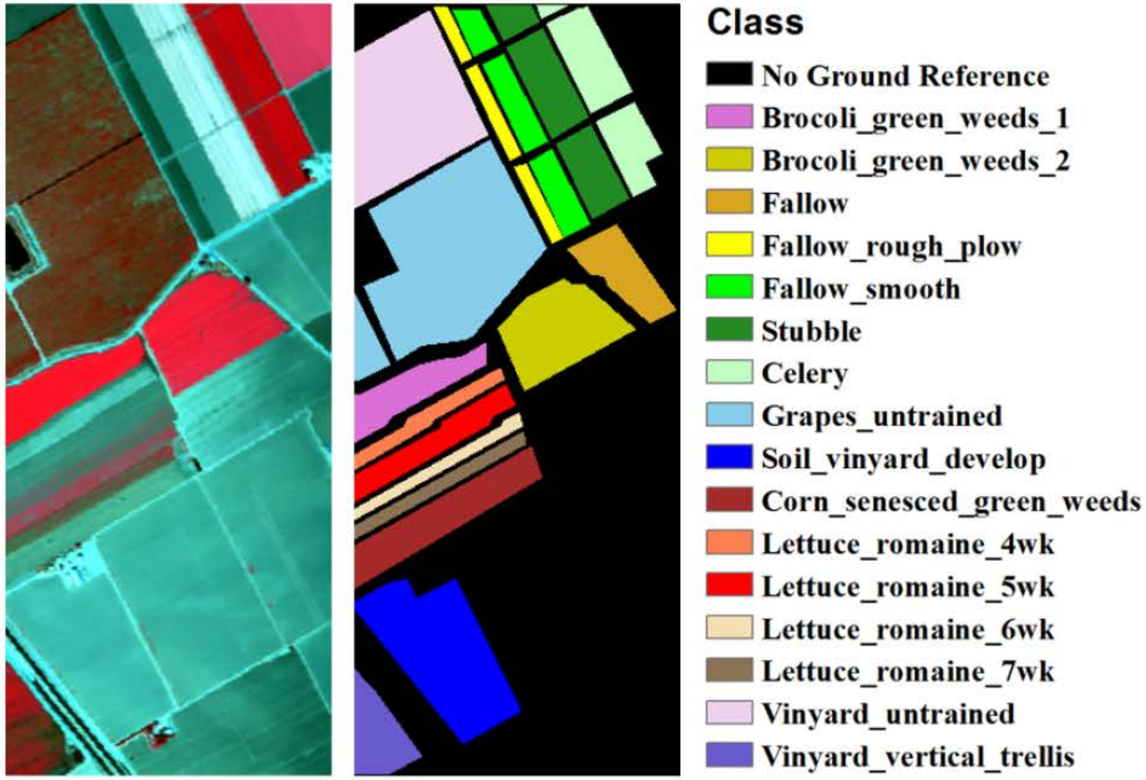
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

What to do now?



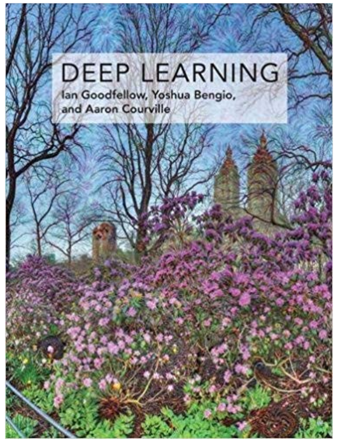
What to do now?





What to do now?

- Stanford course CS231n on
  "Convolutional Neural Networks for Visual Recognition"
  - PDF lecture presentation & YouTube lecture videos
    http://cs231n.stanford.edu/



- Deep Learning Book by Goodfellow, Bengio, Courville

- Machine Learning book by Géron