Uncertainty in Deep Learning for Image Classification

Zurich University of Applied Sciences

School of Engineering

IDP Institute of Data Analysis and Process Design

Method Seminar HS2018

Elvis Murina

Institute of Data Analysis and Process Design

Zurich University of Applied Sciences

19.11.2018

Outline

- Definitions and History
  - AI, Machine Learning, Deep Learning
- Image Classification
  - Motivation, challenges
  - Traditional vs Deep Learning
  - Imagenet challenge
- Artificial neural networks
  - Fully connected vs Convolution neural networks
  - Stochastic gradient descent
  - Backpropagation and the chain rule
- Tricks to fight overfitting
  - Data Augmentation, Dropout
- Uncertainty
  - MC-Dropout and uncertainty in Deep Learning
- HCS Application
- Questions

What is Deep Learning?

Image source: https://knowyourmeme.com/photos/1244486-what-people-think-i-do-what-i-really-do

Definitions

Definitions

- AI
  - The effort to automate intellectual tasks normally performed by humans (François Chollet)
- Machine Learning
  - Field of study that gives computers the ability to learn without being explicitly programmed. (Arthur Samuel)
  - A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. (Tom M. Mitchell)
- Deep Learning
  - A subset of ML which uses deep artificial neural networks as models and does not require feature engineering.

Definitions

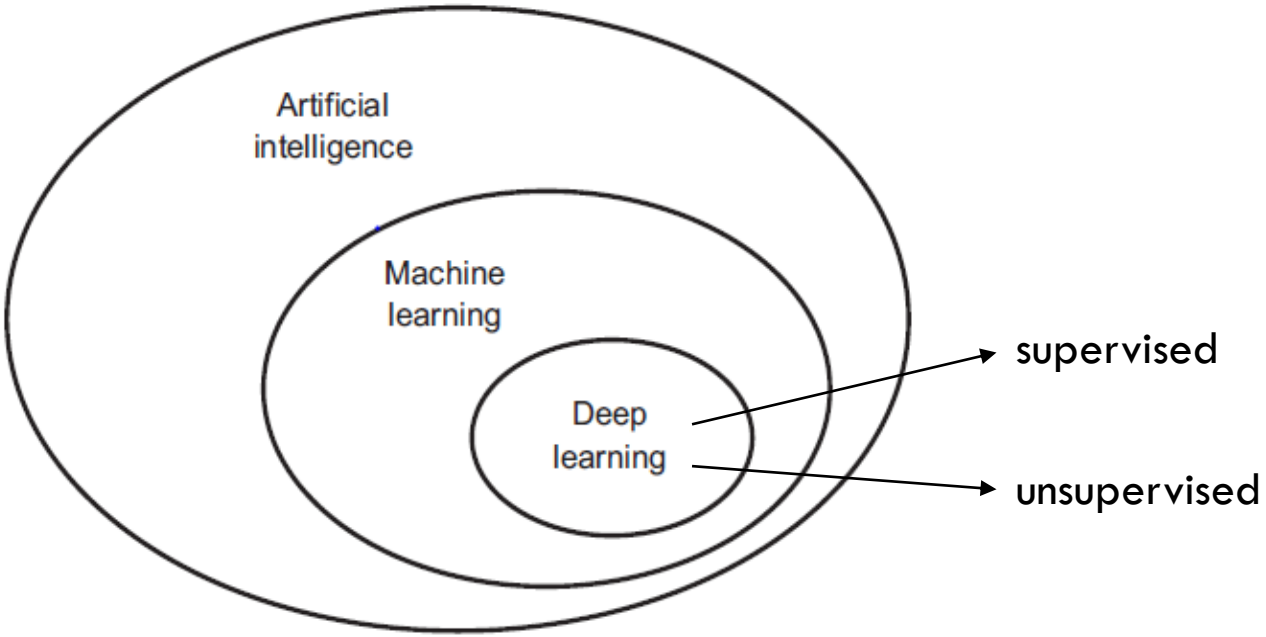


Image source: Deep Learning with Python

History of Neural Networks

History

NEURAL NETWORKS ARE AN OLD IDEA

One of the very first ideas in machine learning and artificial intelligence

- Date back to 1940s
- Many cycles of boom and bust
- Repeated promises of "true AI" that were unfulfilled followed by "AI winters"

"[The perceptron is] the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."

- Frank Rosenblatt, 1958

Image source: https://slides.com/beamandrew/deep-learning-101/#/12

History

- Linear
- No hidden layers
- Learnable weights
- Hype

- Simple non linear functions not solvable

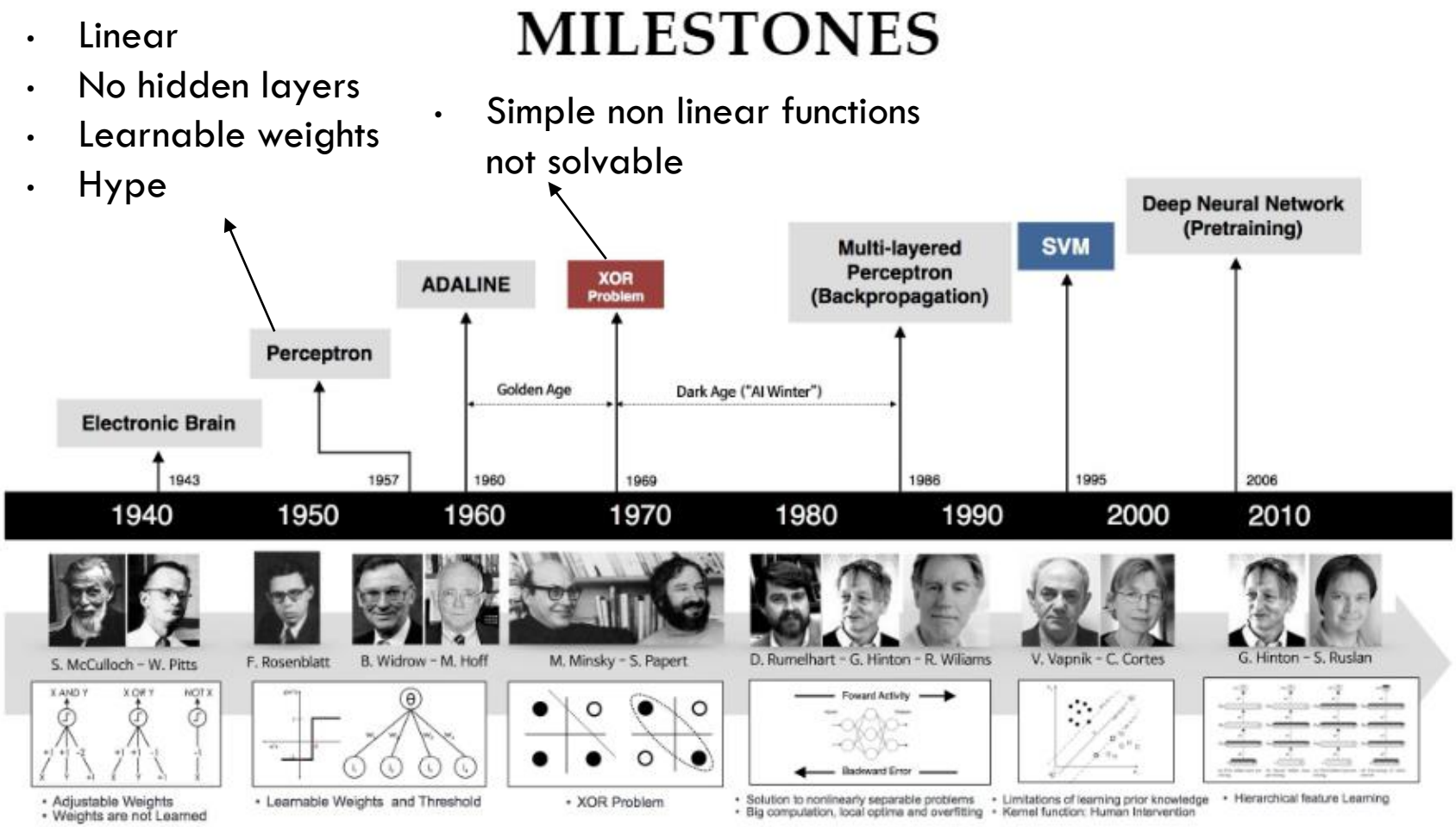


Image source: https://slides.com/beamandrew/deep-learning-101/#/12

Image Classification and other computer vison tasks

Image Classification

- A core task in computer vision

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

cat

Image source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture2.pdf (cs231n)

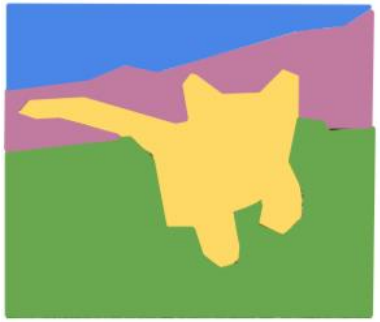
Image Classification

- Other task in computer vision

Image classification

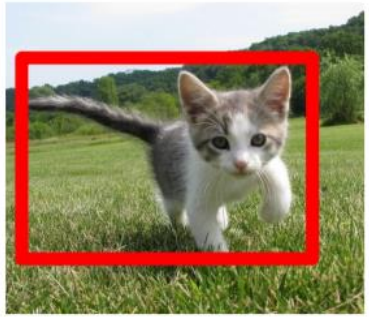
Cat

Semantic Segmentation



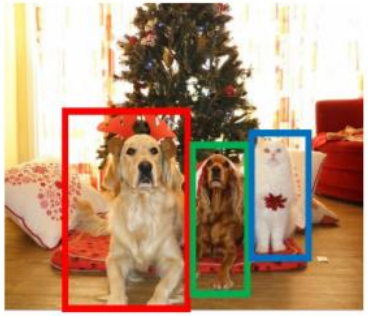
Grass, Cat, Tree, Sky

Object Detection



Cat

Object Detection



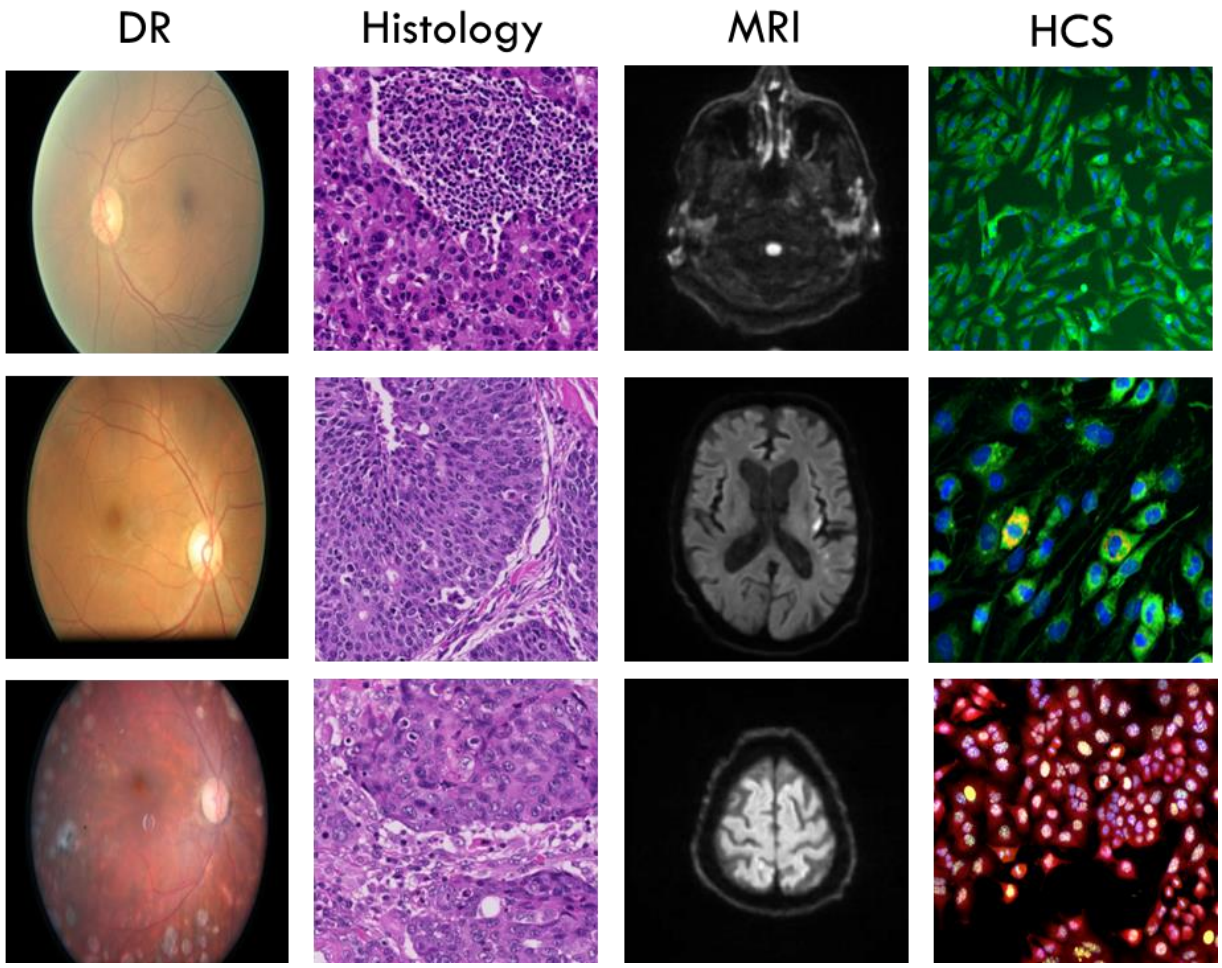
Dog, Dog, Cat

Image source: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture11.pdf (cs231n)

Motivation

Image source: http://cs231n.stanford.edu/slides/2016/winter1516_lecture1.pdf (cs231n)

Motivation



Challenges

- Sematic gap

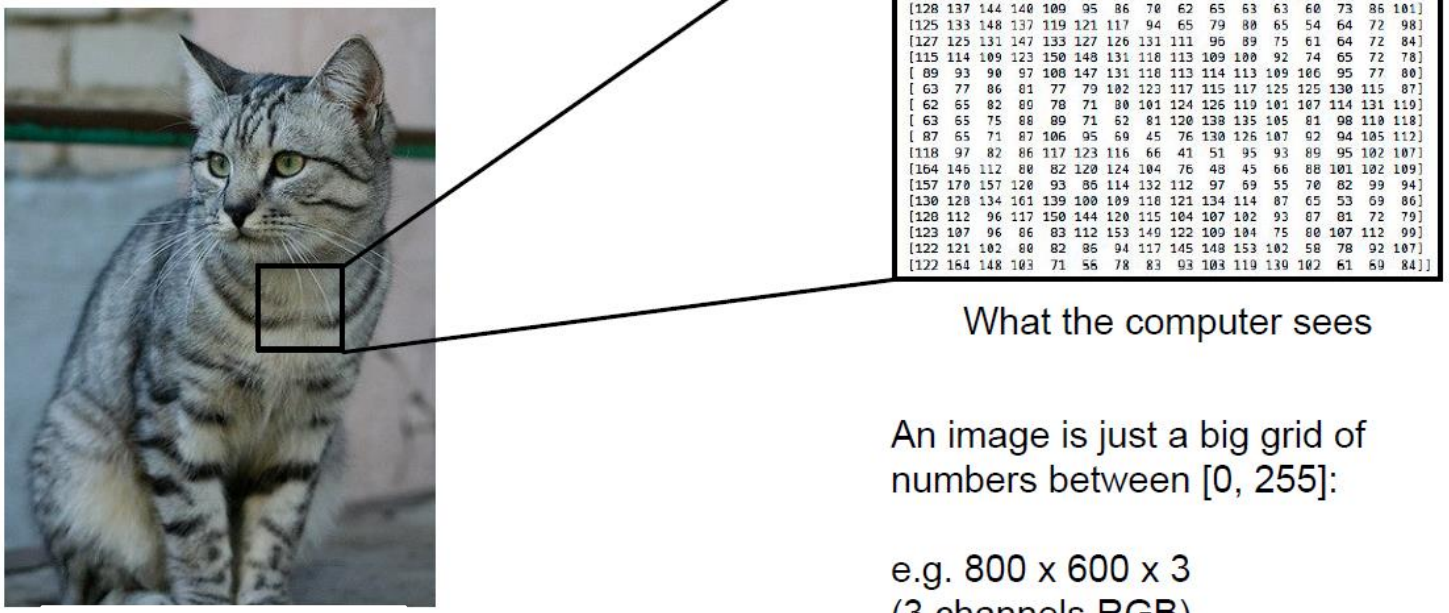


Image source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture2.pdf (cs231n)

Challenges

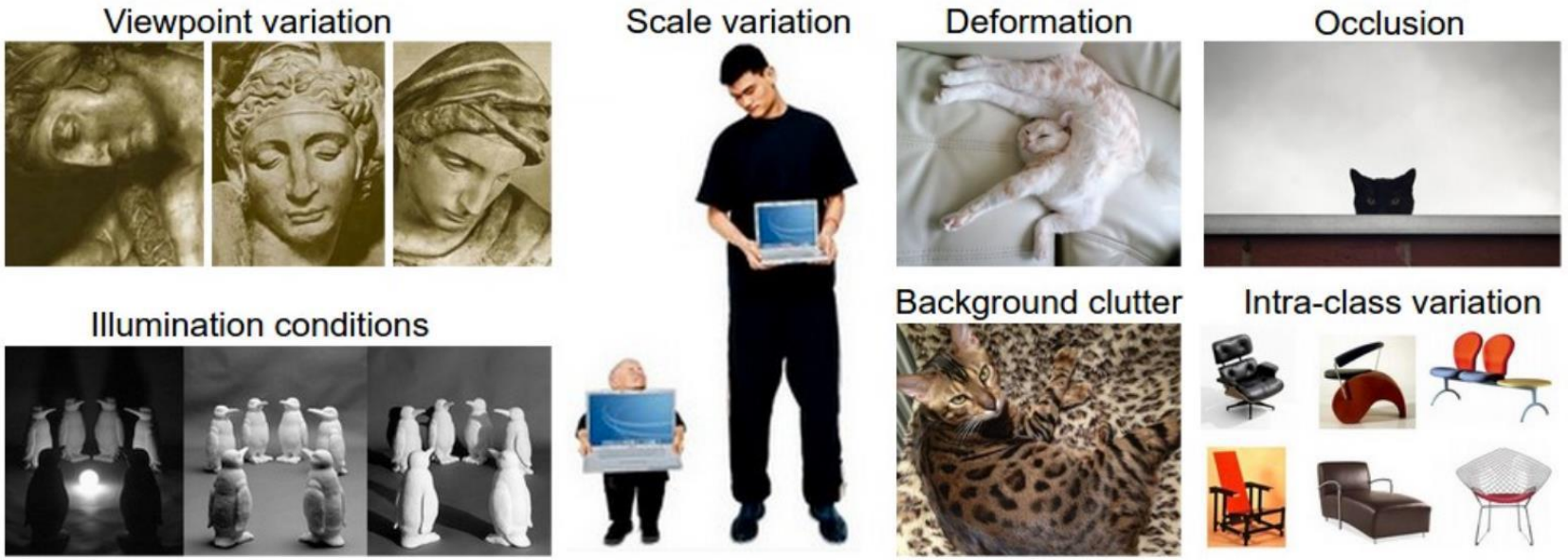


Image source: http://cs231n.github.io/classification/ (cs231n)

Old image classification paradigm


```
if  (eyes == 2) &
    (legs == 4) &
    (tail == 1 ) &
    …
then Print "Cat!"
```

Hard coded rules

Image source: https://www.zhaw.ch/storage/engineering/institute-zentren/iamp/sp_acss/Pelillo_Winterthur.pdf

New Machine learning paradigm

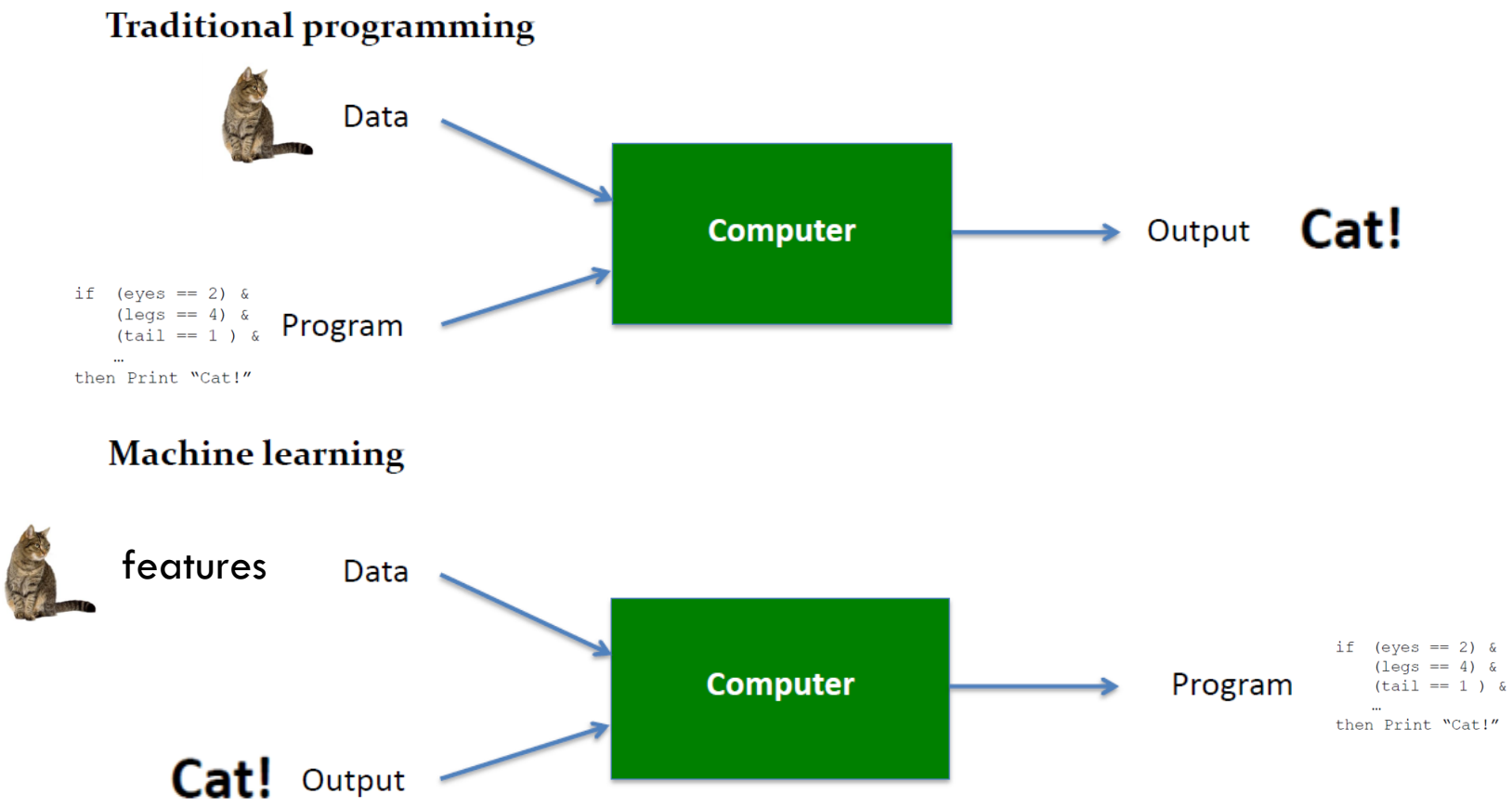


Image source: https://www.zhaw.ch/storage/engineering/institute-zentren/iamp/sp_acss/Pelillo_Winterthur.pdf

Deep Learning paradigm

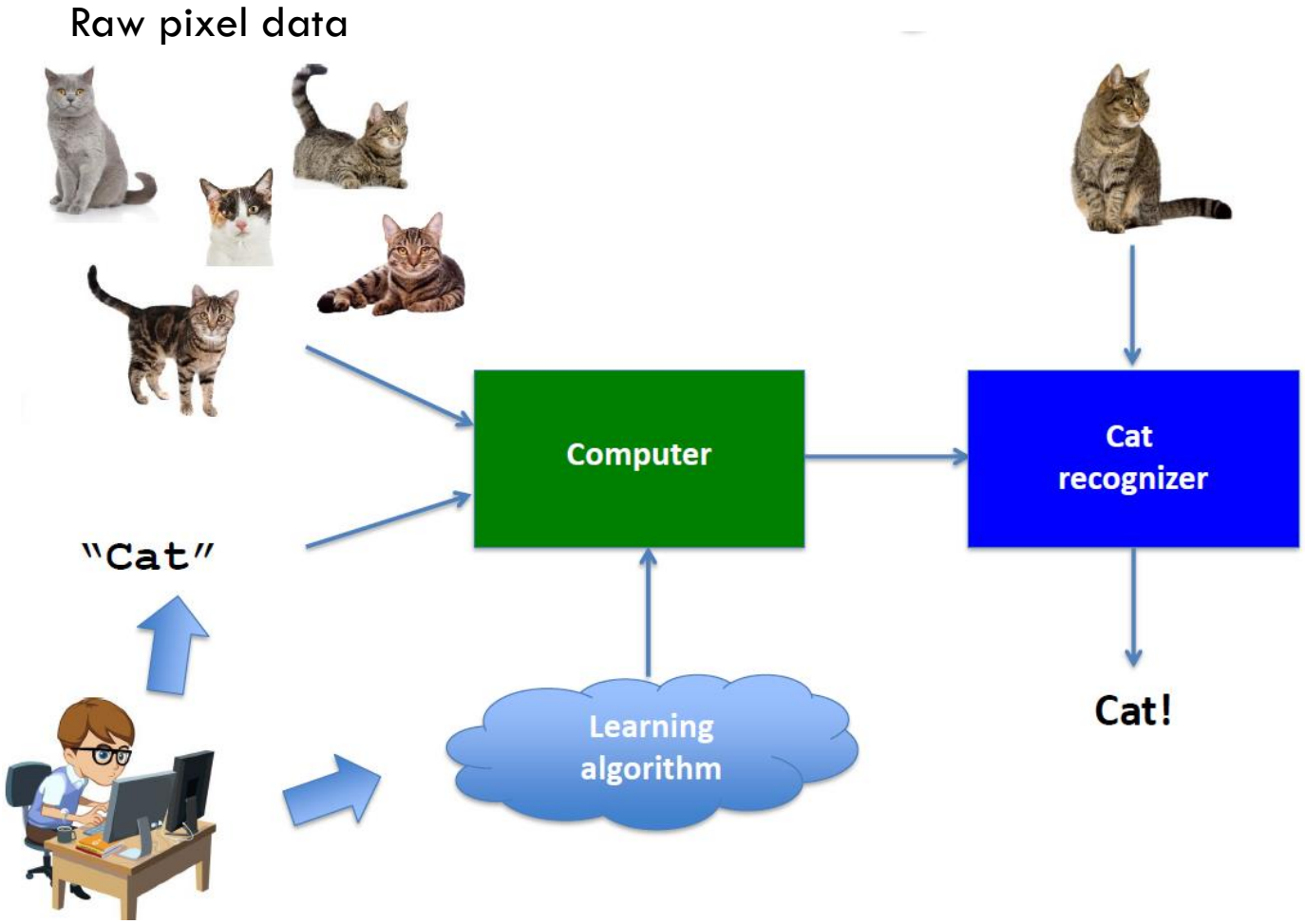
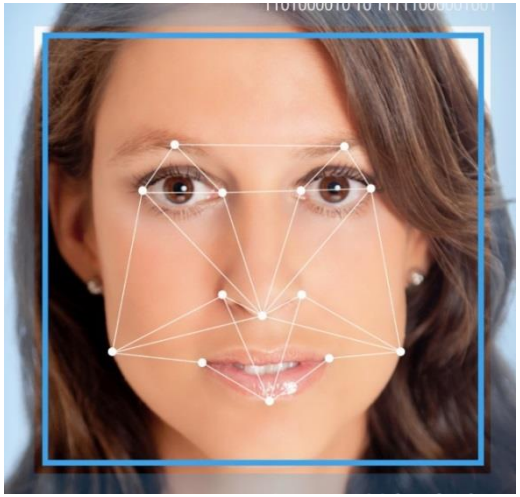


Image source: https://www.zhaw.ch/storage/engineering/institute-zentren/iamp/sp_acss/Pelillo_Winterthur.pdf

Traditional image analysis (ML) vs. Deep Learning



Traditional:
Extract handcrafted features and use these features to train/fit a model (SVM, RF) and use fitted model to perform classification/prediction.

Deep learning:
In deep neural networks start with raw data and learn during training/fitting to extract appropriate hierarchical features and to use them for classification/prediction.

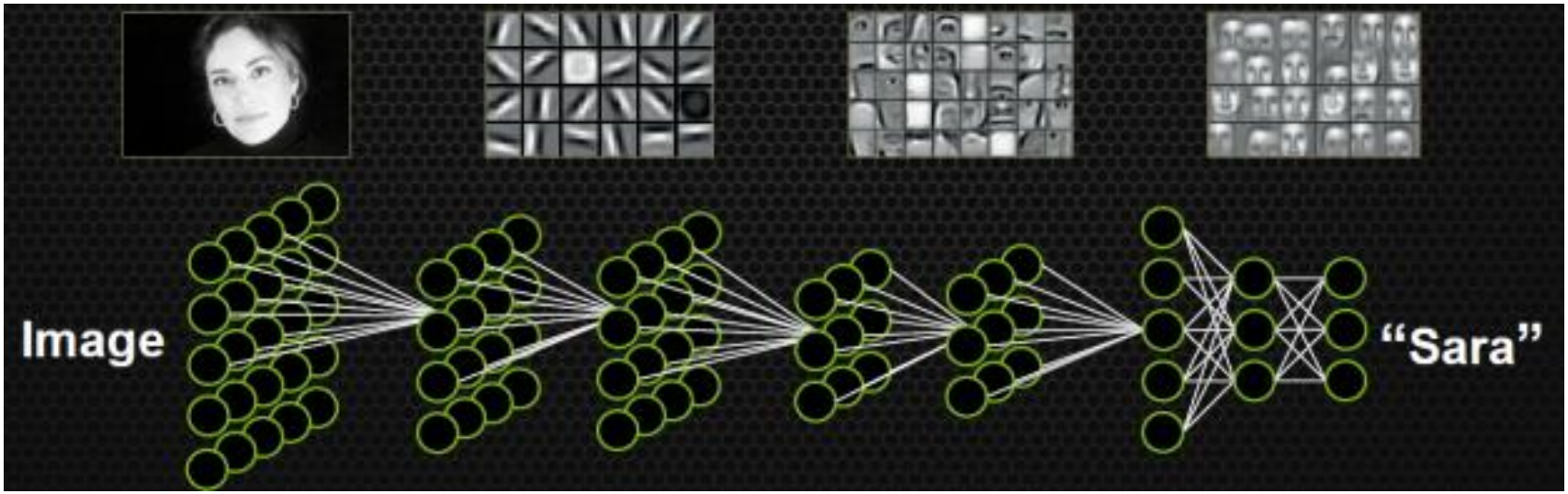
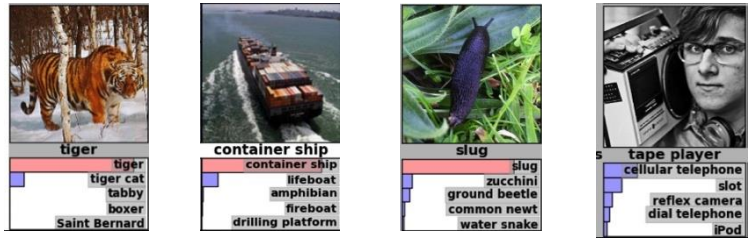


Image source: https://developer.nvidia.com/discover/artificial-neural-network

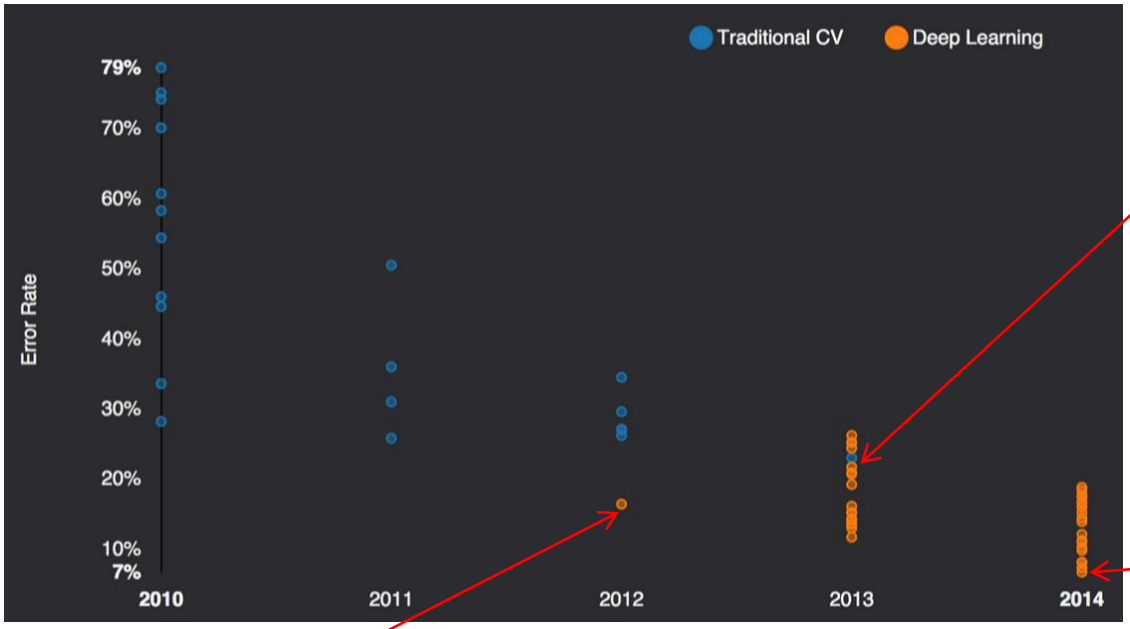
Imagenet
The Deep Learning revolution

ImageNet competition

1000 classes
1 Mio samples



Human: 5% misclassification



Only one non-CNN approach in 2013

GoogLeNet 6.7%

A. Krizhevsky
first CNN in 2012

2015

4.95% Microsoft (surpassing human performance 5%)
4.8% Google (further improved to 3.6%)
3.57% Microsoft (Resnet winner 2015)

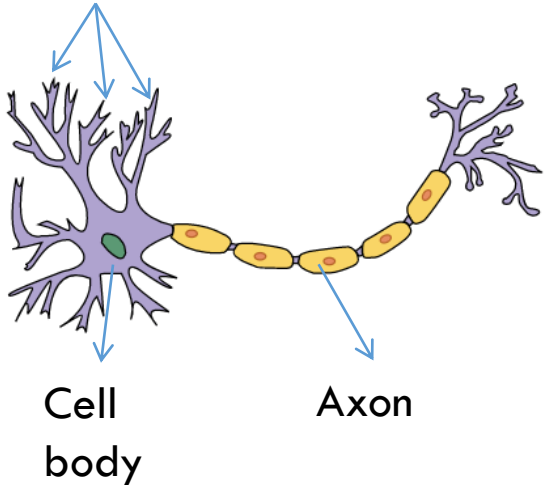
Image source: https://medium.com/global-silicon-valley/machine-learning-yesterday-today-tomorrow-3d3023c7b519

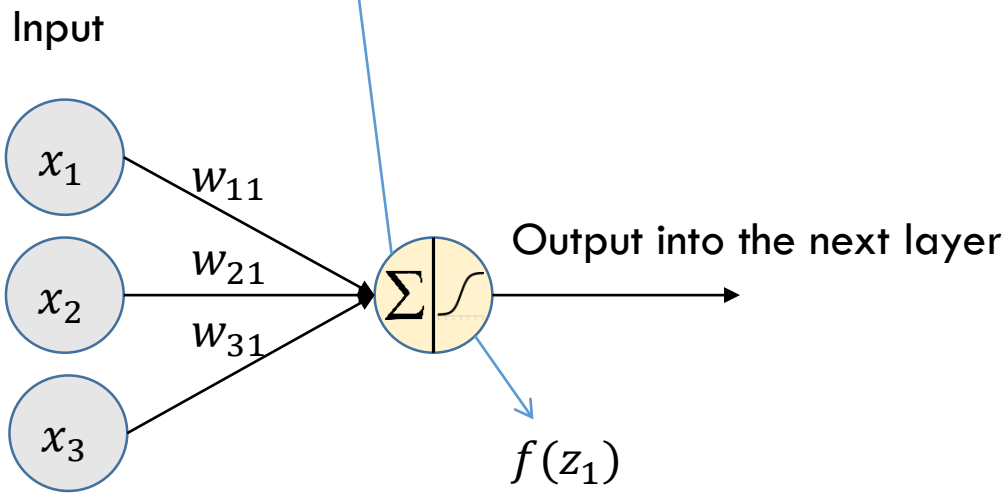
Artificial neural networks

Artificial neural networks

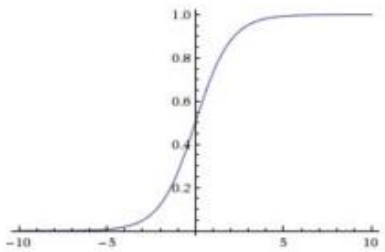
$$z_1 = \sum x_1 * w_{11} + x_2 * w_{21} + x_3 * w_{31} + b$$

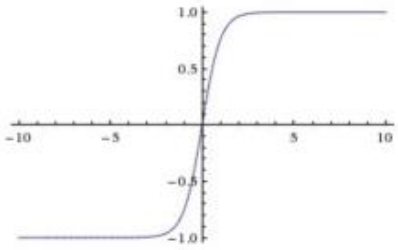
Weighted sum of x and W

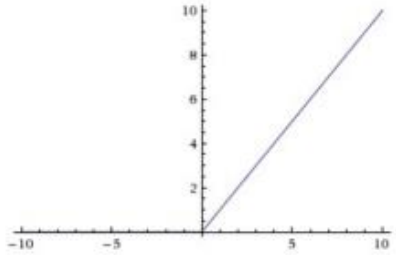




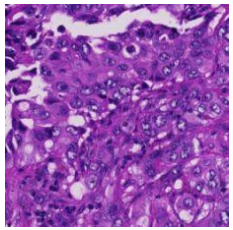
Non–linear activation function for z1



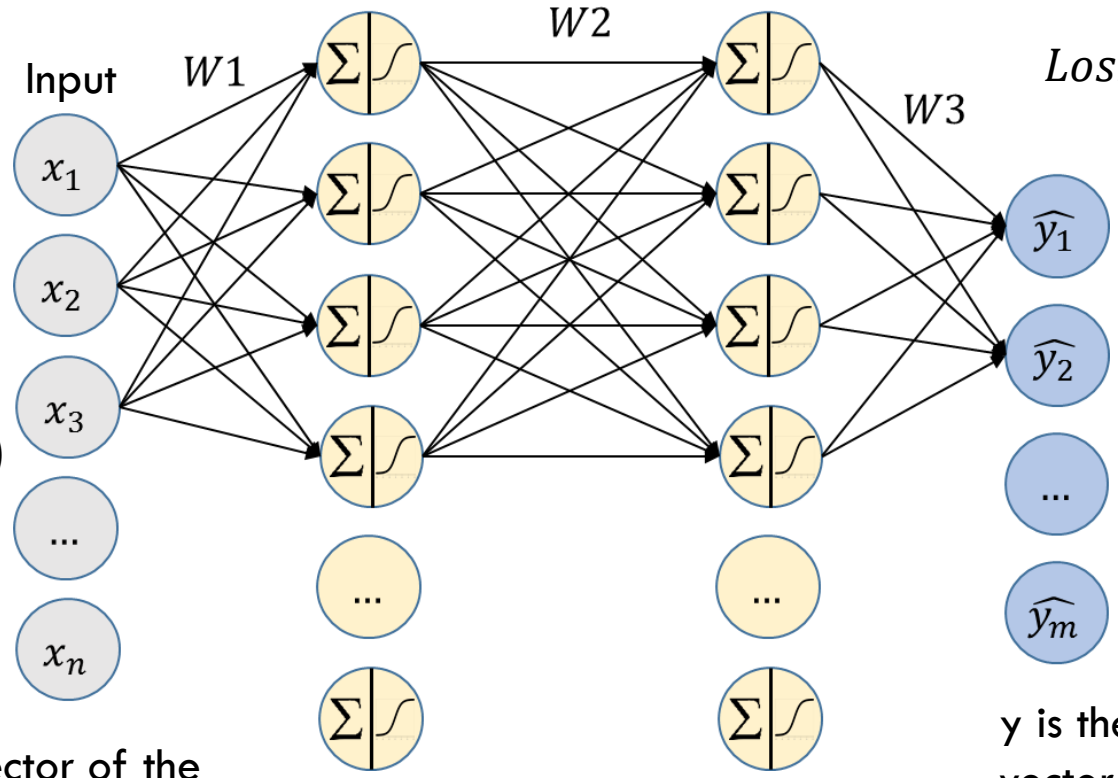




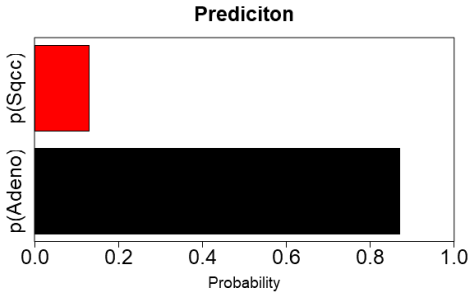
Artificial neural networks



(100x100x3)



$$Loss = -\frac{1}{N}\sum_{i=1}^{N} y_i \left(\log(\hat{y}_i)\right)$$



Example for two classes

x is the vector of the flattened input image

W are the weight matrices that are learned during the training phase of the network

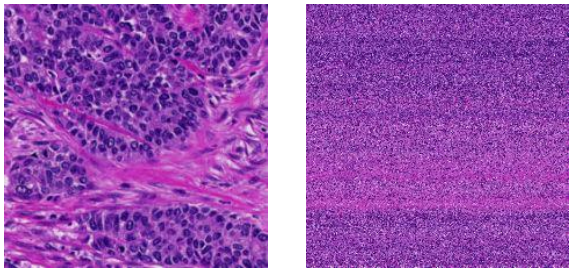
y is the probability vector for each class

$$Softmax = \frac{e^i}{\sum_{k=1}^{m} e^k}$$

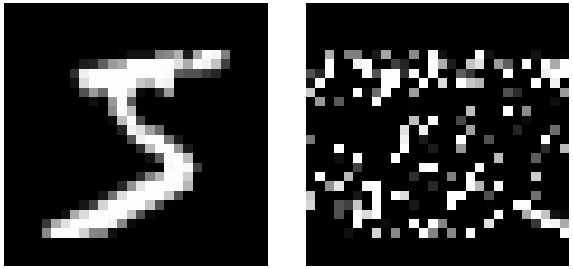
Artificial neural networks

- Two main disadvantages
  - Images are often big and therefore you need a lot of weights (also for the hidden layers)
  - The spatial information of the image gets lost, because you flatten the input.
  - Images left and right are the same for a fully connected network

Adeno vs Sqcc example

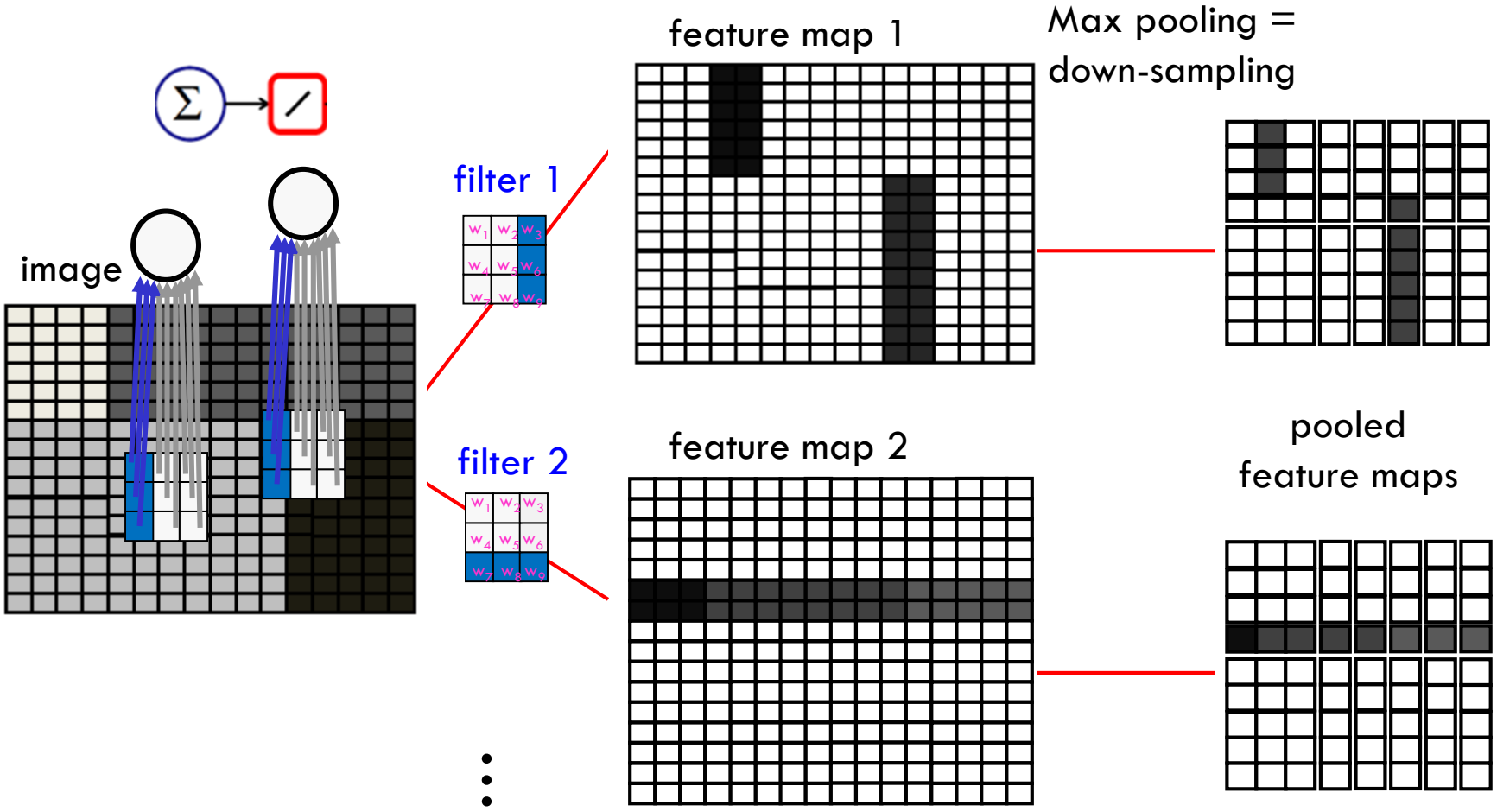


Mnist example



Convolutional neural networks

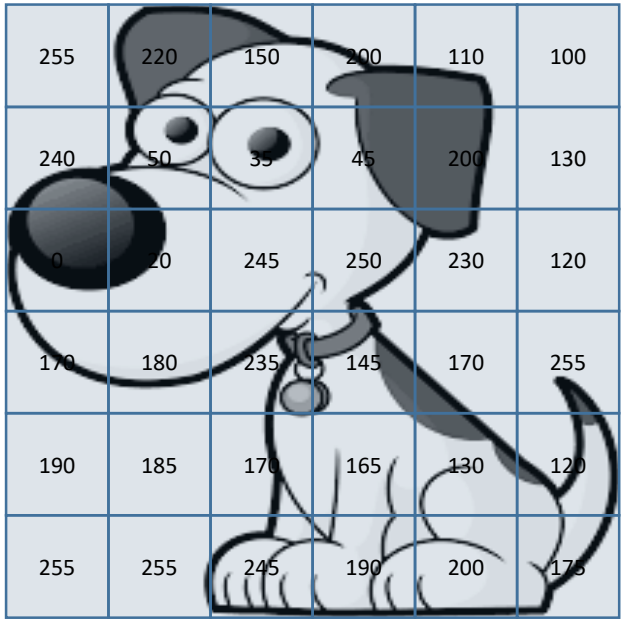
Convolutional neural networks



The weights of each filter are randomly initiated and then adapted during the training.

Convolutional neural networks

Input image 6x6x1



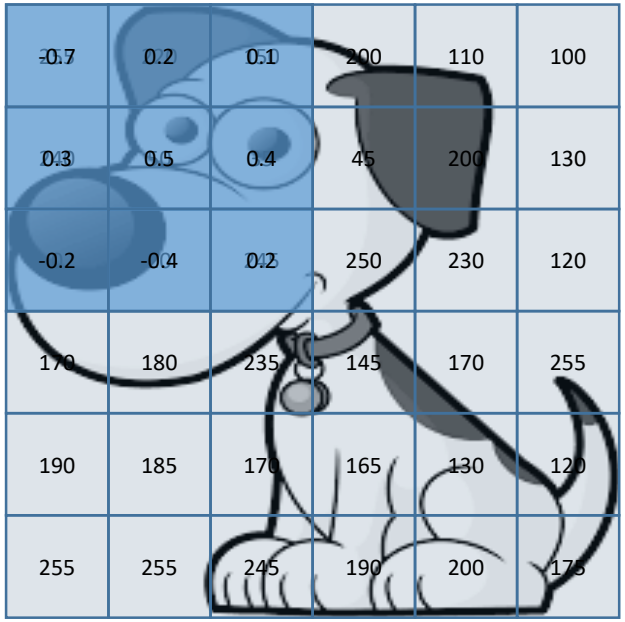
-0.7	0.2	0.1
0.3	0.5	0.4
-0.2	-0.4	0.2

3x3 filter

$$z = b + \sum_i x_i w_i$$

Convolutional neural networks

Input image 6x6x1



Feature map 4x4x1

32.5

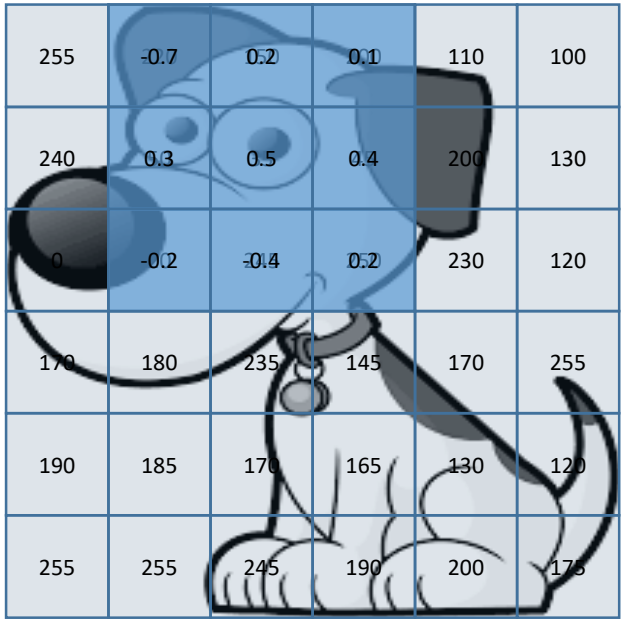
-0.7	0.2	0.1
0.3	0.5	0.4
-0.2	-0.4	0.2

3x3 filter

$$z = b + \sum_i x_i w_i$$

Convolutional neural networks

Input image 6x6x1



Feature map 4x4x1

32.5	-105.5

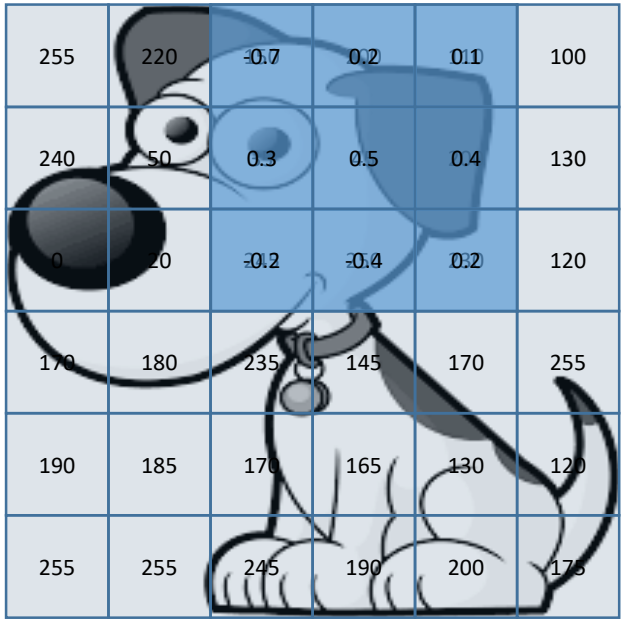
-0.7	0.2	0.1
0.3	0.5	0.4
-0.2	-0.4	0.2

3x3 filter

$$z = b + \sum_i x_i w_i$$

Convolutional neural networks

Input image 6x6x1



Feature map 4x4x1

32.5	-105.5	185.5

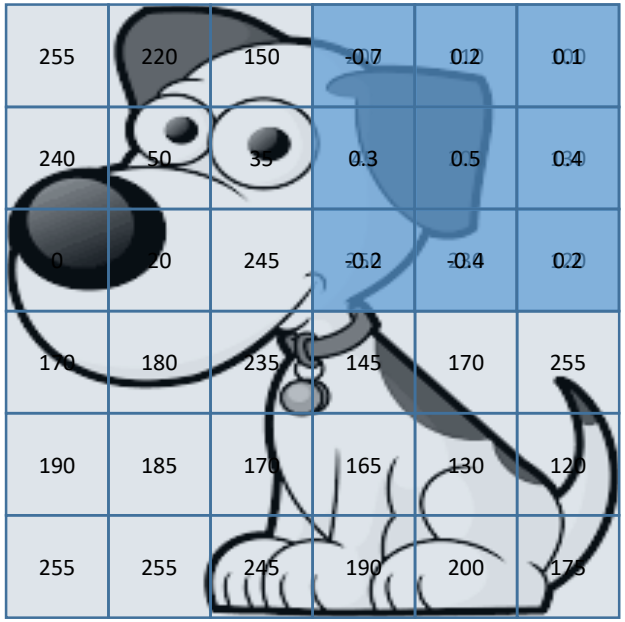
-0.7	0.2	0.1
0.3	0.5	0.4
-0.2	-0.4	0.2

3x3 filter

$$z = b + \sum_i x_i w_i$$

Convolutional neural networks

Input image 6x6x1



Feature map 4x4x1

32.5	-105.5	185.5	54

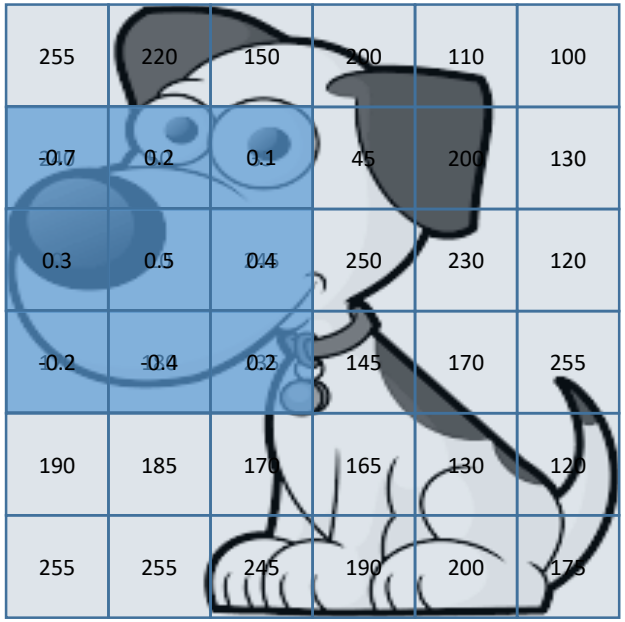
-0.7	0.2	0.1
0.3	0.5	0.4
-0.2	-0.4	0.2

3x3 filter

$$z = b + \sum_i x_i w_i$$

Convolutional neural networks

Input image 6x6x1



Feature map 4x4x1

32.5	-105.5	185.5	54
-105.5			

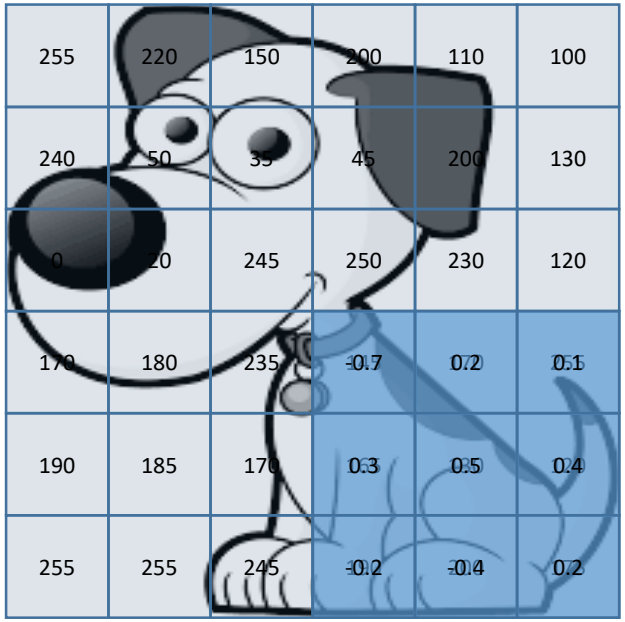
-0.7	0.2	0.1
0.3	0.5	0.4
-0.2	-0.4	0.2

3x3 filter

$$z = b + \sum_i x_i w_i$$

Convolutional neural networks

Input image 6x6x1



Feature map 4x4x1

32.5	-105.5	185.5	54
-105.5	104	217.5	31
-44	224	38.5	-18
-60.5	213.5	52.5	37.5

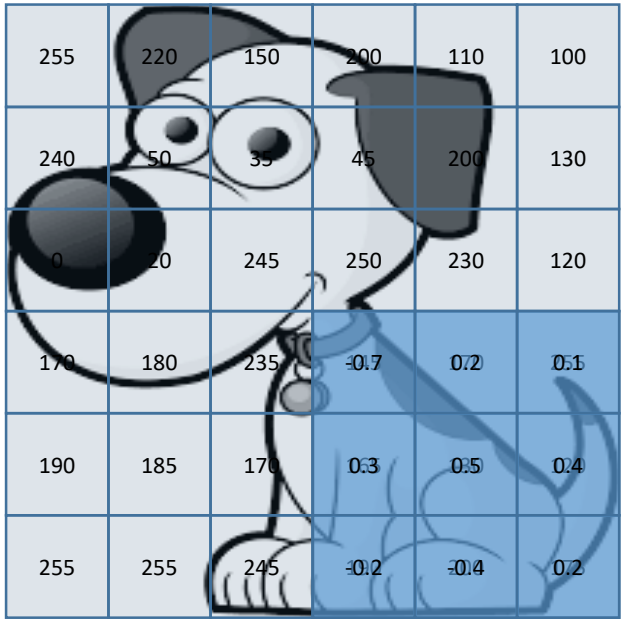
-0.7	0.2	0.1
0.3	0.5	0.4
-0.2	-0.4	0.2

3x3 filter

$$z = b + \sum_i x_i w_i$$

Convolutional neural networks

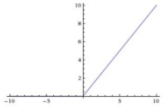
Input image 6x6x1



255	220	150	200	110	100
240	50	35	45	200	130
0	20	245	250	230	120
170	180	235	-0.7	0.2	0.1
190	185	170	0.3	0.5	0.4
255	255	245	-0.2	-0.4	0.2

Feature map 4x4x1

32.5	-105.5	185.5	54
-105.5	104	217.5	31
-44	224	38.5	-18
-60.5	213.5	52.5	37.5



Relu

32.5	0	185.5	54
0	104	217.5	31
0	224	38.5	0
0	213.5	52.5	37.5

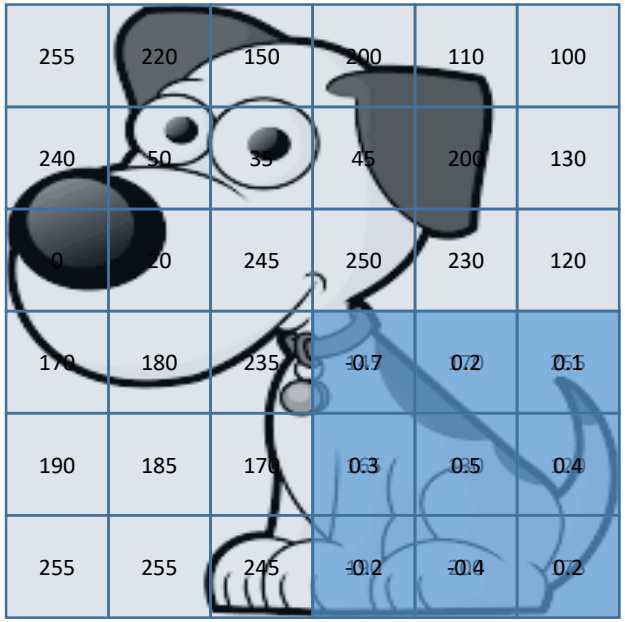
-0.7	0.2	0.1
0.3	0.5	0.4
-0.2	-0.4	0.2

3x3 filter

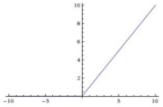
$$z = b + \sum_i x_i w_i$$

Convolutional neural networks

Input image 6x6x1



255	220	150	200	110	100
240	50	35	45	200	130
0	20	245	250	230	120
170	180	235	-0.7	0.2	0.1
190	185	170	0.3	0.5	0.4
255	255	245	-0.2	-0.4	0.2



Feature map 4x4x1

32.5	-105.5	185.5	54
-105.5	104	217.5	31
-44	224	38.5	-18
-60.5	213.5	52.5	37.5

Relu

32.5	0	185.5	54
0	104	217.5	31
0	224	38.5	0
0	213.5	52.5	37.5

Maxpool (2x2x1)

104

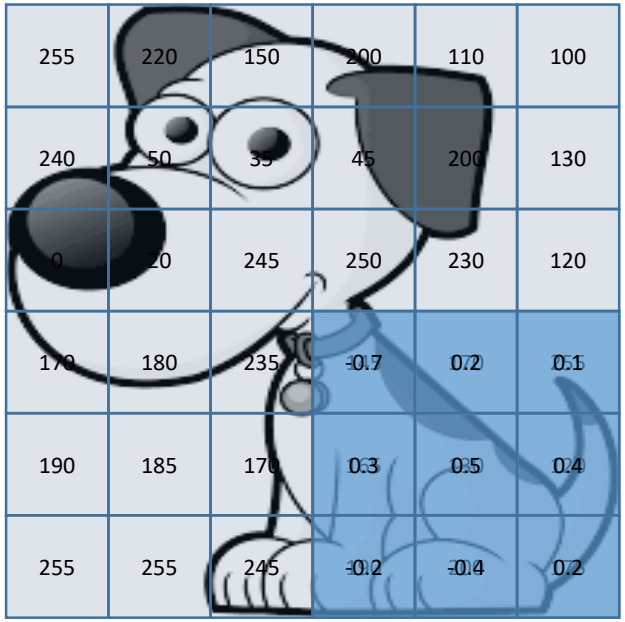
-0.7	0.2	0.1
0.3	0.5	0.4
-0.2	-0.4	0.2

3x3 filter

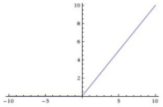
$$z = b + \sum_i x_i w_i$$

Convolutional neural networks

Input image 6x6x1



255	220	150	200	110	100
240	50	35	45	200	130
0	20	245	250	230	120
170	180	235	-0.7	0.2	0.1
190	185	170	0.3	0.5	0.4
255	255	245	-0.2	-0.4	0.2



Feature map 4x4x1

32.5	-105.5	185.5	54
-105.5	104	217.5	31
-44	224	38.5	-18
-60.5	213.5	52.5	37.5

Relu

32.5	0	185.5	54
0	104	217.5	31
0	224	38.5	0
0	213.5	52.5	37.5

Maxpool (2x2x1)

104	217.5

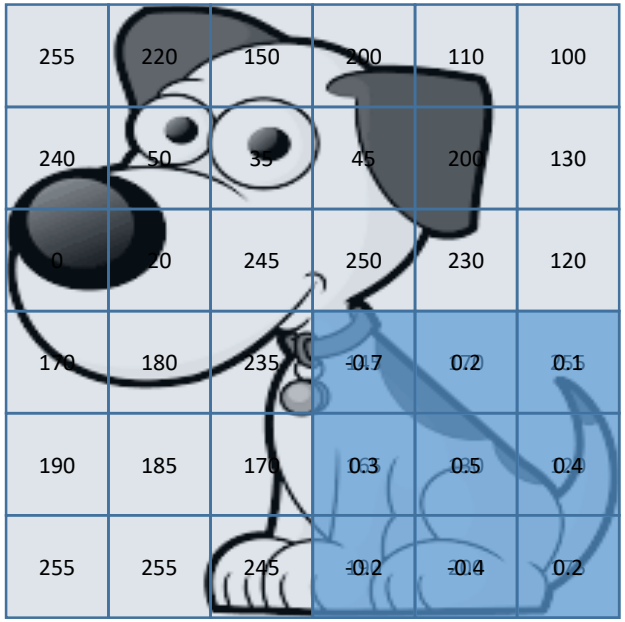
-0.7	0.2	0.1
0.3	0.5	0.4
-0.2	-0.4	0.2

3x3 filter

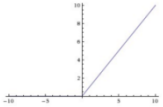
$$z = b + \sum_i x_i w_i$$

Convolutional neural networks

Input image 6x6x1



255	220	150	200	110	100
240	50	35	45	200	130
0	20	245	250	230	120
170	180	235	-0.7	0.2	0.1
190	185	170	0.3	0.5	0.4
255	255	245	-0.2	-0.4	0.2



Feature map 4x4x1

32.5	-105.5	185.5	54
-105.5	104	217.5	31
-44	224	38.5	-18
-60.5	213.5	52.5	37.5

Relu

32.5	0	185.5	54
0	104	217.5	31
0	224	38.5	0
0	213.5	52.5	37.5

Maxpool (2x2x1)

104	217.5
224	

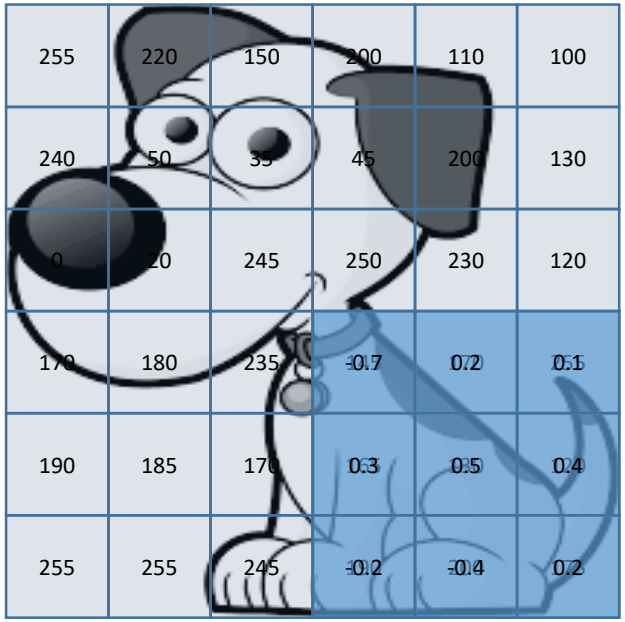
-0.7	0.2	0.1
0.3	0.5	0.4
-0.2	-0.4	0.2

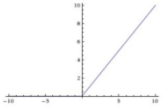
3x3 filter

$$z = b + \sum_i x_i w_i$$

Convolutional neural networks

Input image 6x6x1





Feature map 4x4x1

32.5	-105.5	185.5	54
-105.5	104	217.5	31
-44	224	38.5	-18
-60.5	213.5	52.5	37.5

Relu

32.5	0	185.5	54
0	104	217.5	31
0	224	38.5	0
0	213.5	52.5	37.5

Maxpool (2x2x1)

104	217.5
224	52.5

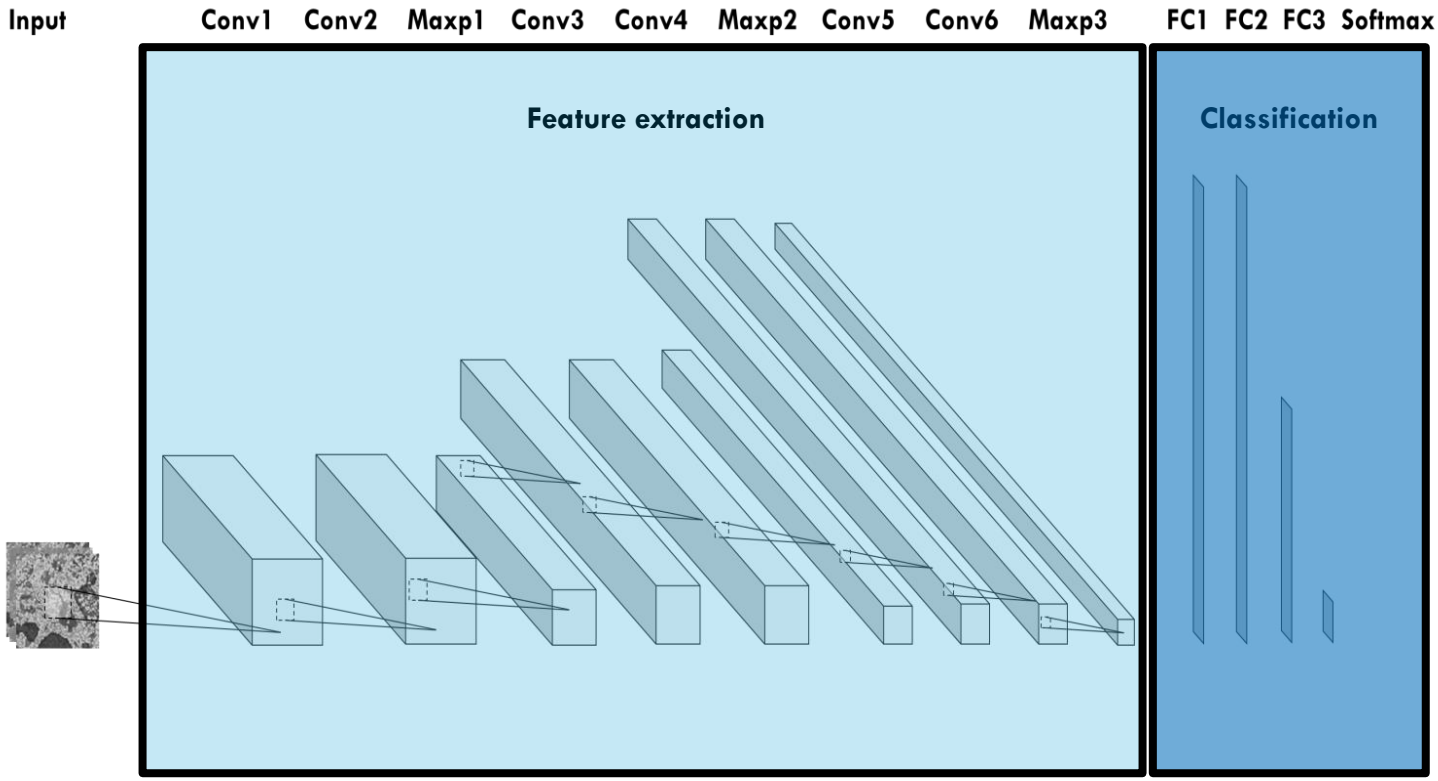
-0.7	0.2	0.1
0.3	0.5	0.4
-0.2	-0.4	0.2

3x3 filter

$$z = b + \sum_i x_i w_i$$

Convolutional neural networks

Typical architecture of a CNN

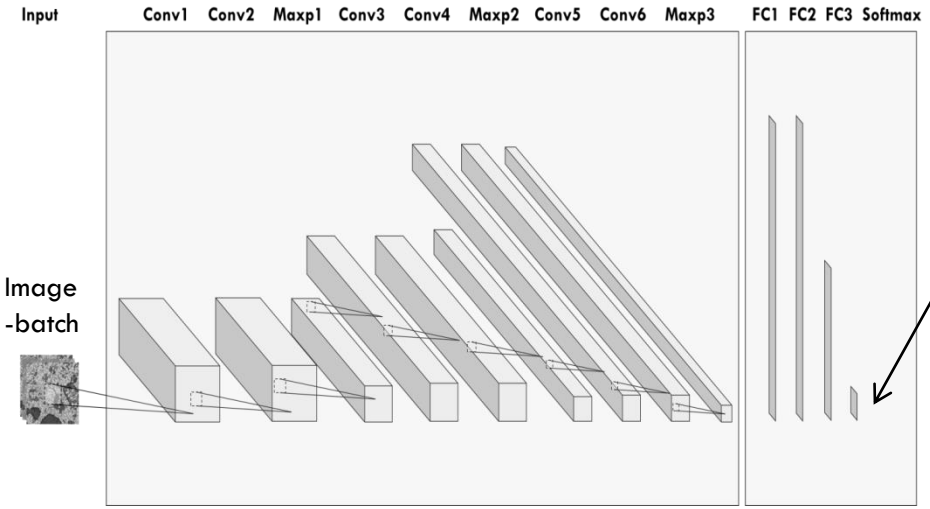


Spatial resolution is decreased with max-pooling while more abstract image features are detected in deeper layers.

Convolutional neural networks

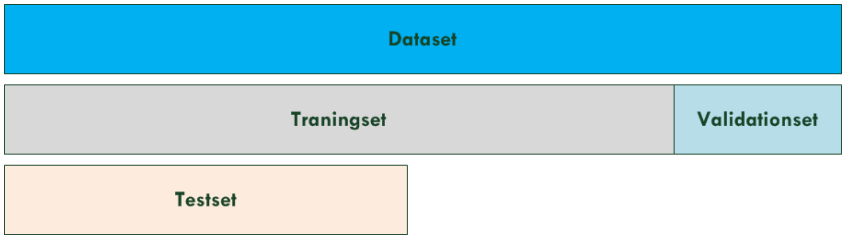
Forward pass

Calculate the output based on the current parameters



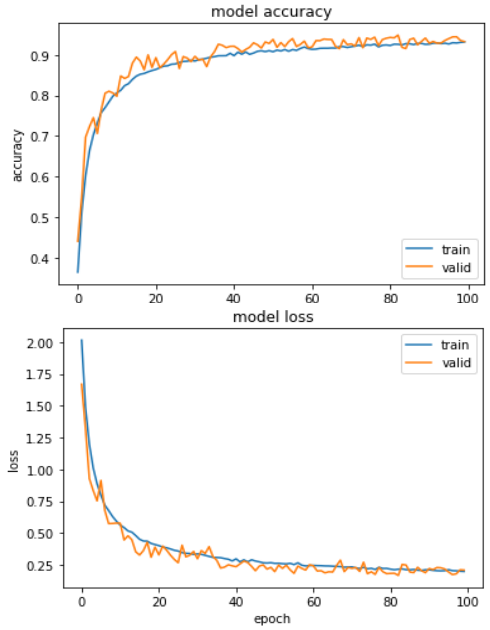
Backward pass

Update the parameters to minimize the loss



$$Loss = -\frac{1}{N}\sum_{i}^{N} y_i \left(\log(\hat{y}_i)\right)$$

Categorical crossentropy



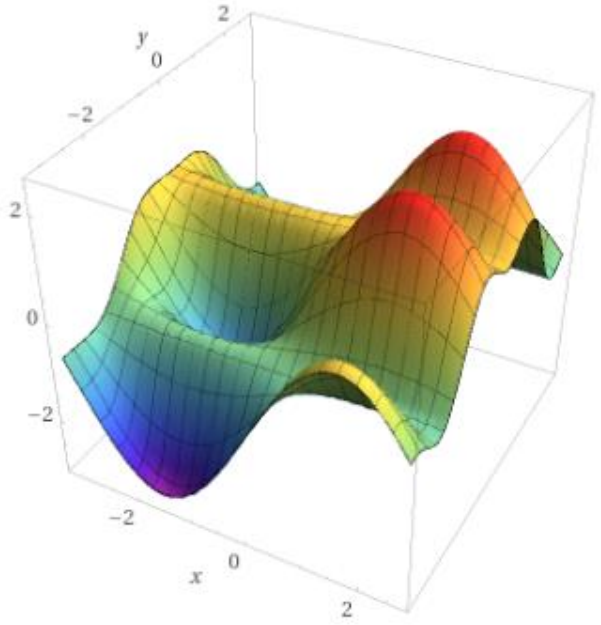
Weight update:

$$w_{i(t+1)} = w_{i(t)} - \eta * \frac{\partial L(w)}{\partial w_i}$$

Gradient descent, stochastic gradient descent online gradient descent and mini-batch gradient descent to minize the loss

Gradient descent in DL

Follow the slope!



2 dimensional loss function. In DL we have millions dimensions!
We just know the current value (we are blind)

Image source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture3.pdf (cs231n)

Image source: https://www.matroid.com/blog/post/the-hard-thing-about-deep-learning

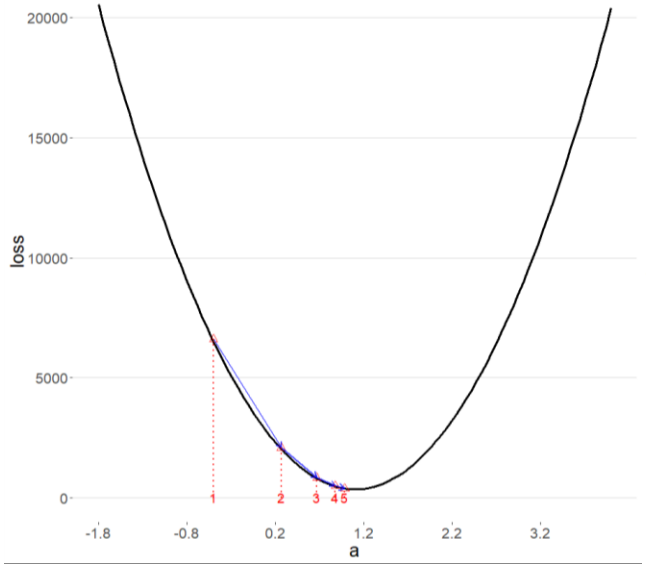
Stochastic gradient in DL

- Calculate the gradient (vector of partial derivatives) of the Loss w.r.t to all weights, update weights and repeat
- Take a small step (learning rate) in the negative direction of the gradient which is the steepest descent
  - GD: calculate all the gradients with **all** training data and do one update step
    - Often not possible because (memory) of the size of the training data and impractical.
  - SGD: calculate gradient (approximation) of **one random sample** of the training data and do one update step.
  - MBSGD: calculate gradients (approximation) of **mini-batch random sample** of the training data and do one update step.
  - OGD: same as SGD

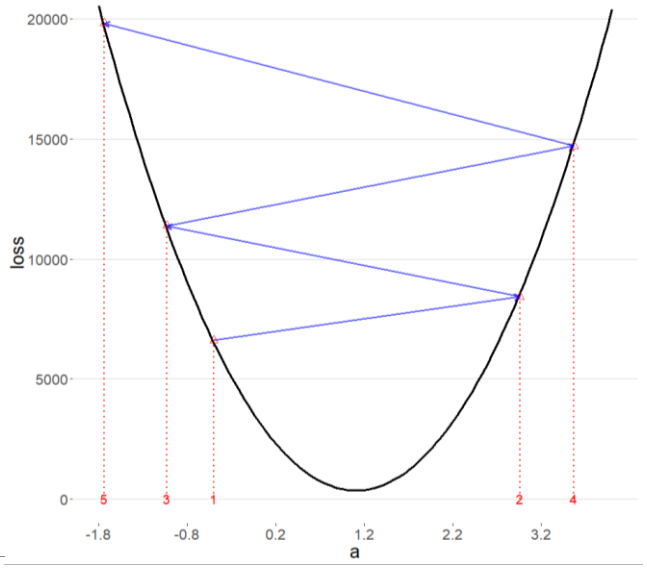
Stochastic gradient in DL

- Learning rate is the most important hyper parameter in gradient descent!
- Toy example simple linear regression (convex Loss function):

small learning rate



large learning rate



Keep clam and lower your learning rate!

Backpropagation and chain rule

Chain rule recap

- If we have two functions

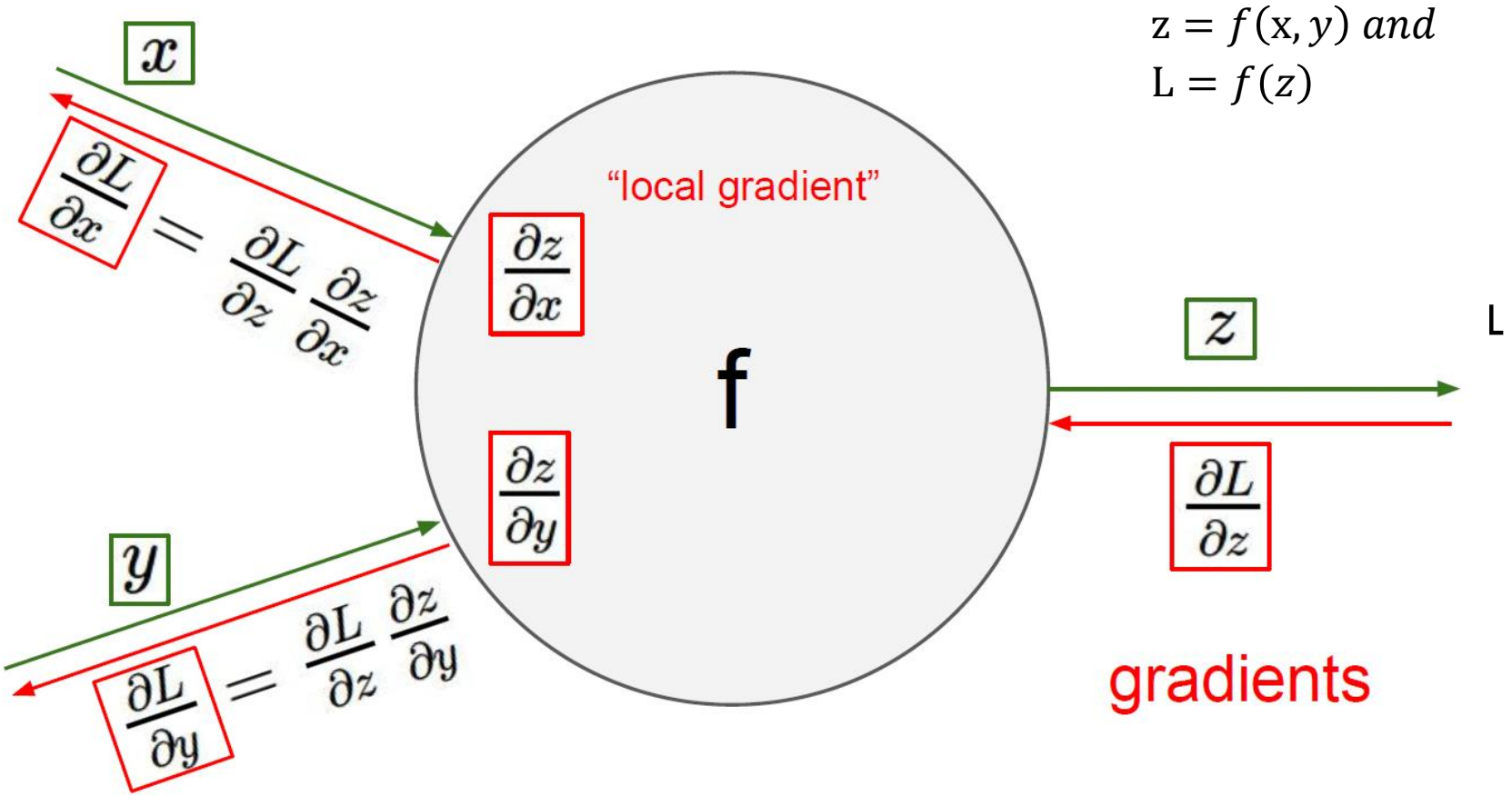
$$z = f(y)\ and$$
$$y = f(x)$$

then y and z are dependent variables.

- And by the chain rule:

$$\frac{\partial z}{\partial x} = \frac{\partial y}{\partial x} * \frac{\partial z}{\partial y}$$

Backpropagation and chain rule

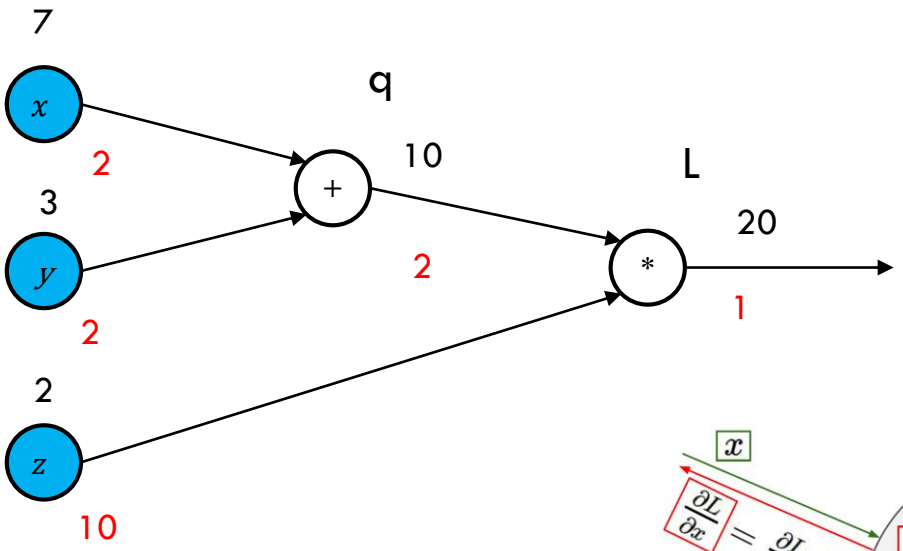


Simple example

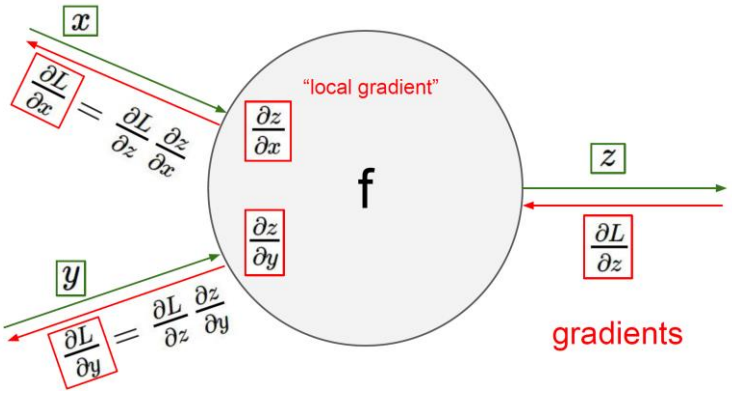
- **Multiplications do switch gradients (times the incoming gradient)**
- **Additions are passing the gradient trough**

$$\frac{\partial L}{\partial x} = ? ; \frac{\partial L}{\partial y} = ? ; \frac{\partial L}{\partial z} = ?$$

$$L = (x + y) * z$$



$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial q} * \frac{\partial q}{\partial x} = 1 * 2 = 2$$



$$f = a * b ; \frac{\partial f}{\partial a} = b ; \frac{\partial f}{\partial b} = a$$

$$f = a + b ; \frac{\partial f}{\partial a} = 1 ; \frac{\partial f}{\partial b} = 1$$

Forward pass

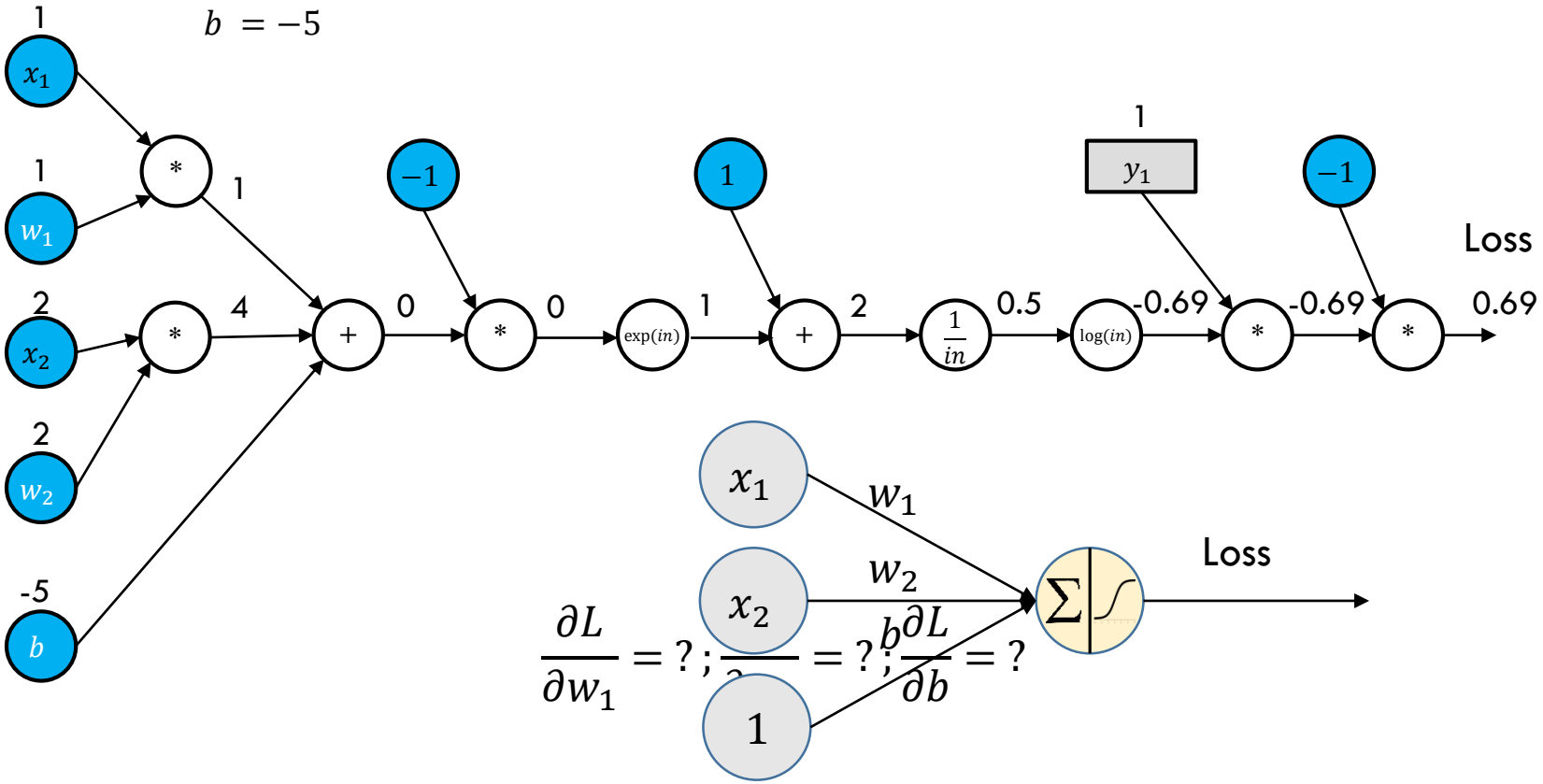
Training data:

$x_1 = 1$
$x_2 = 2$
$y_1 = 1$

Initial weights:

$w_1 = 1$
$w_2 = 2$
$b = -5$

$$p(y = 1|X) = \frac{1}{1 + e^{-(x_1 * w_1 + x_2 * w_2 + b)}}$$



Backward pass

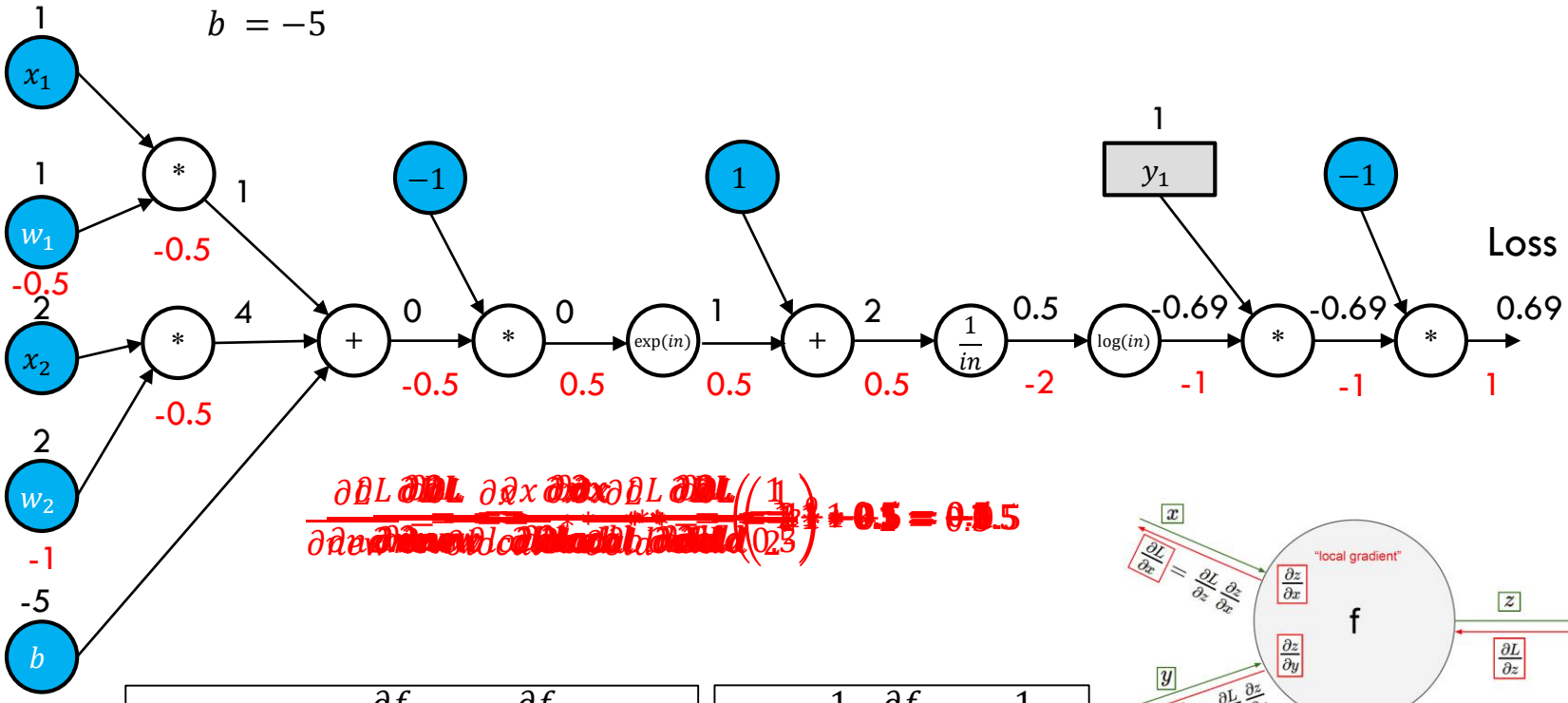
Training data:

$x_1 = 1$
$x_2 = 2$
$y_1 = 1$

Initial weights:

$w_1 = 1$
$w_2 = 2$
$b = -5$

$$p(y = 1|X) = \frac{1}{1 + e^{-(x_1 * w_1 + x_2 * w_2 + b)}}$$



$f = a * b\,;\ \frac{\partial f}{\partial a} = b\,;\ \frac{\partial f}{\partial b} = a$	$f = \frac{1}{a};\ \frac{\partial f}{\partial a} = -\frac{1}{a^2}$	
$f = a + b\,;\ \frac{\partial f}{\partial a} = 1\,;\ \frac{\partial f}{\partial b} = 1$	$f = e^a;\ \frac{\partial f}{\partial a} = e^a$	$f = \log(a)\,;\ \frac{\partial f}{\partial a} = \frac{1}{a}$

Forward pass

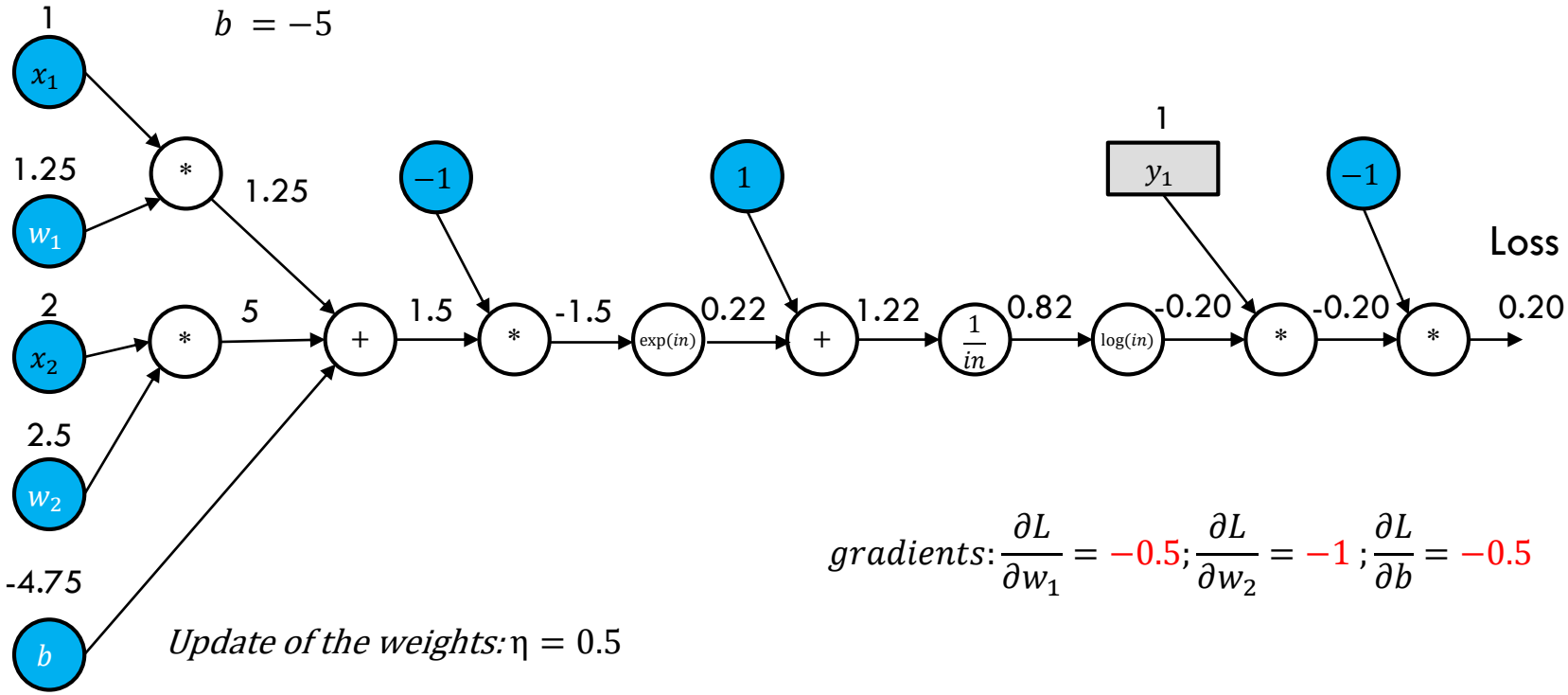
Training data:

$x_1 = 1$
$x_2 = 2$
$y_1 = 1$

Initial weights:

$w_1 = 1$
$w_2 = 2$
$b = -5$

$$p(y = 1|X) = \frac{1}{1 + e^{-(x_1 * w_1 + x_2 * w_2 + b)}}$$



$$gradients: \frac{\partial L}{\partial w_1} = -0.5; \frac{\partial L}{\partial w_2} = -1; \frac{\partial L}{\partial b} = -0.5$$

Update of the weights: $\eta = 0.5$

$$w_{1(t+1)} = w_{1(t)} - \eta * \frac{\partial L}{\partial w_1} = 1 - 0.5 * (-0.5) = 1.25$$

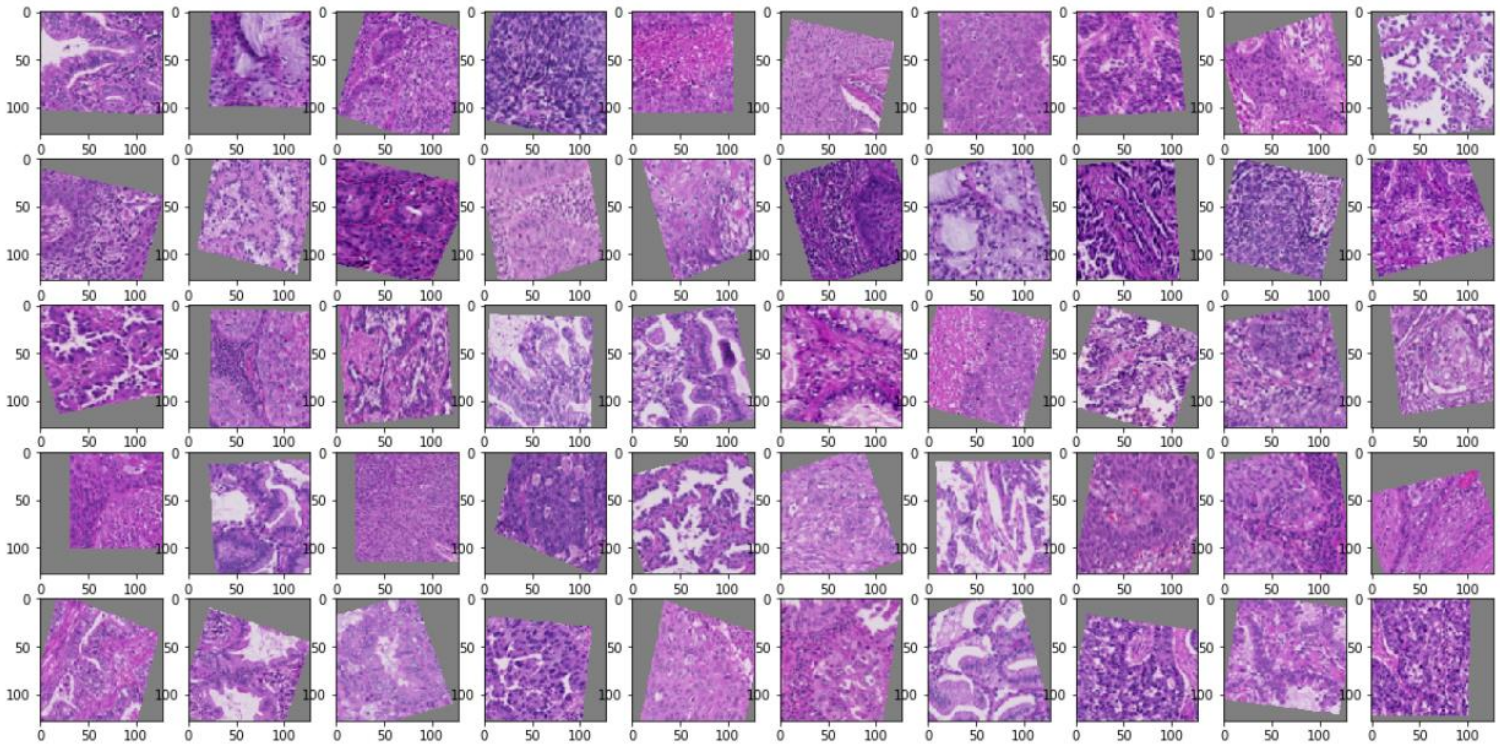
$$w_{2(t+1)} = w_{2(t)} - \eta * \frac{\partial L}{\partial w_2} = 2 - 0.5 * (-1) = 2.5$$

$$b_{(t+1)} = b_{(t)} - \eta * \frac{\partial L}{\partial b} = -5 - 0.5 * (-0.5) = -4.75$$

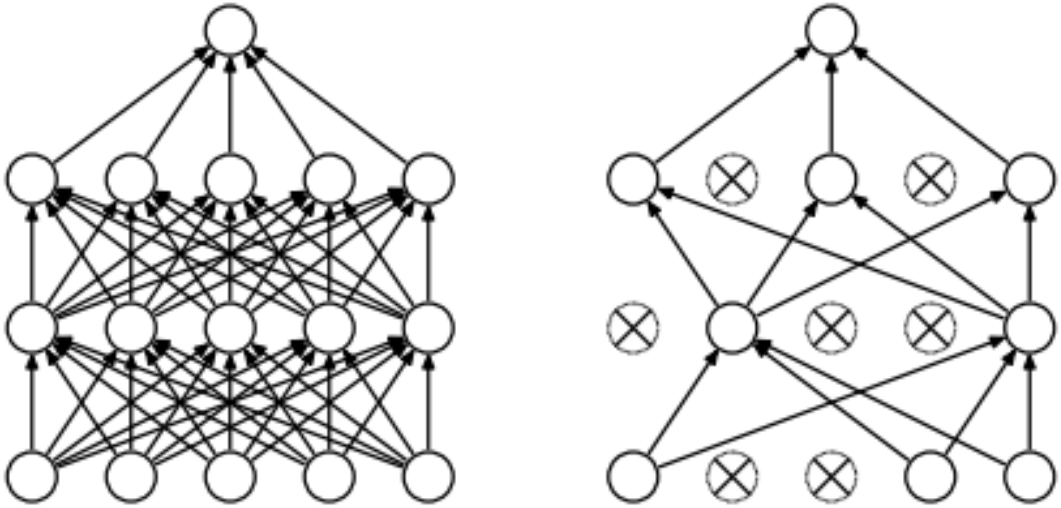
Overfitting

Fighting Overfitting

- Data Augmentation
  - Generate additional training data by rotating, zooming, shearing, flipping…



Fighting Overfitting



(a) Standard Neural Net

(b) After applying dropout.

- In each training step we train another sparse NN
- Dropout prevents co-adaptation and overfitting

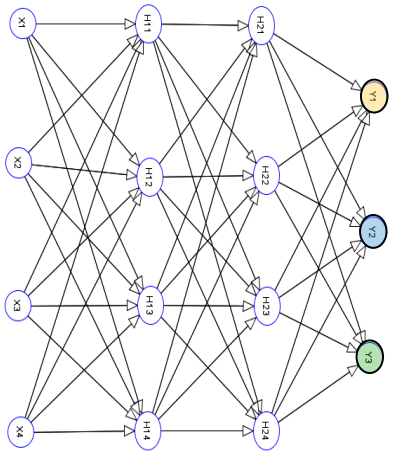
Image source: http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf

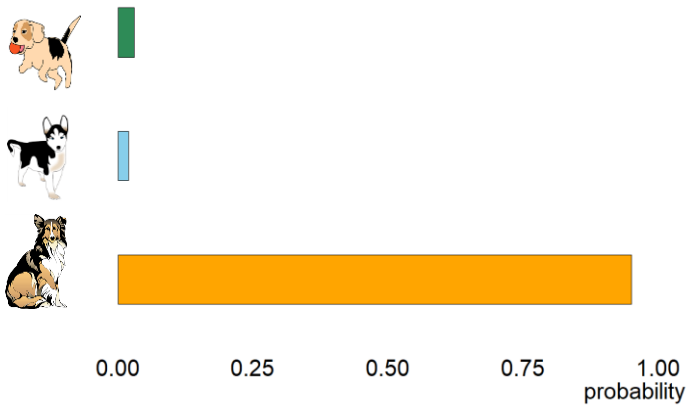
Image source: https://docs.google.com/presentation/d/1TVixw6ItiZ8igjp6U17tcgoFrLSaHWQmMOwjlgQY9co/pub?slide=id.g110257a6da_0_8135

Uncertainty

Uncertainty: A first thought experiment

Suppose you train a classifier on dogs and show it a new dog that was in the training set.

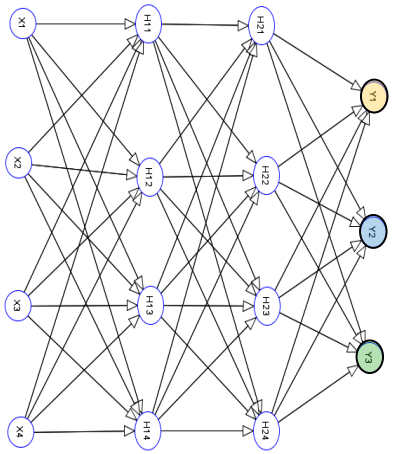


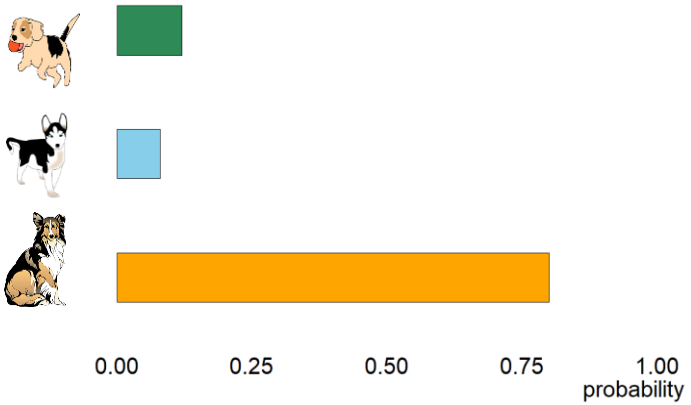


Uncertainty: A first thought experiment

Suppose you show the same classifier a cat

What will be the result?





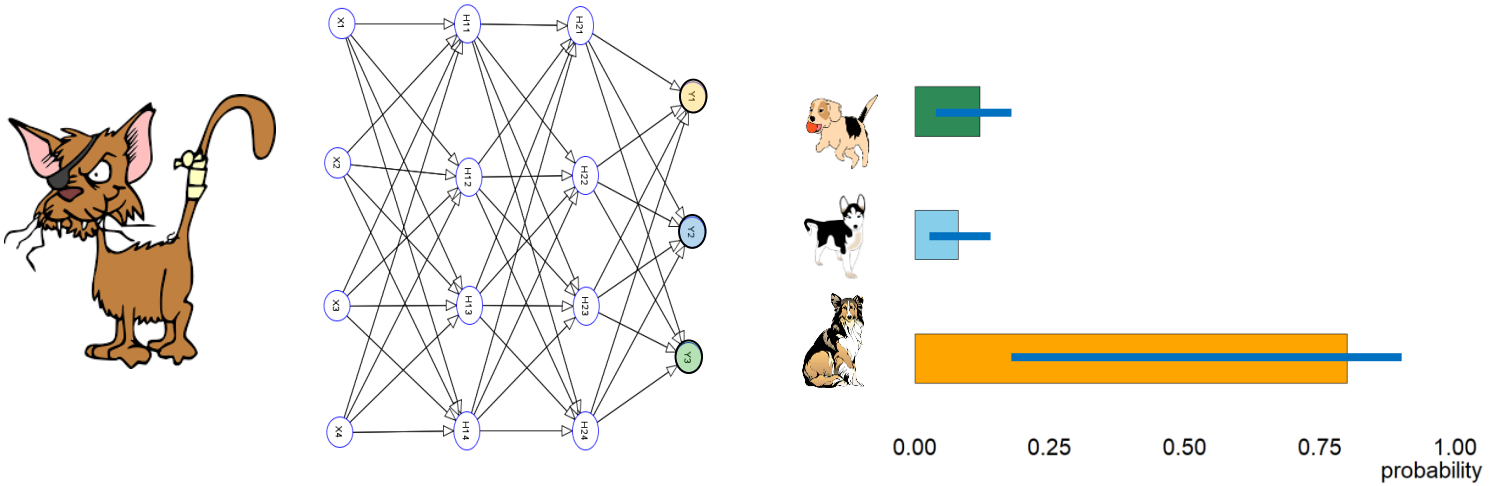
How can that be?

- Forced to classify it into one of the dogs.
- If it's a dog, than most probably a collie
- No confidence of the prediction given

Uncertainty: A first thought experiment

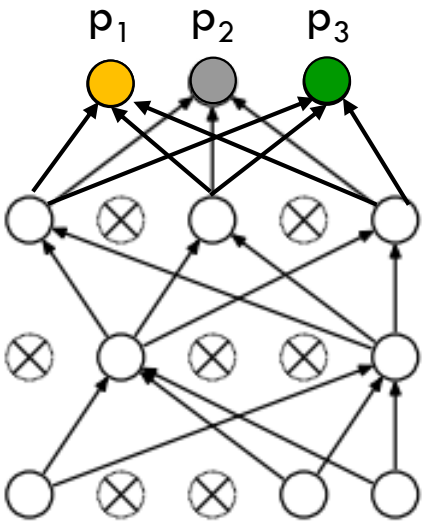
Suppose you show the same classifier a cat

What will be the result?



We need uncertainty measures to our predictions!

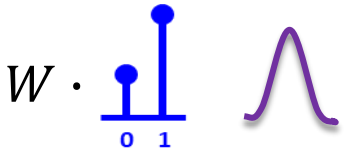
MC Dropout and Bayesian Neural Networks

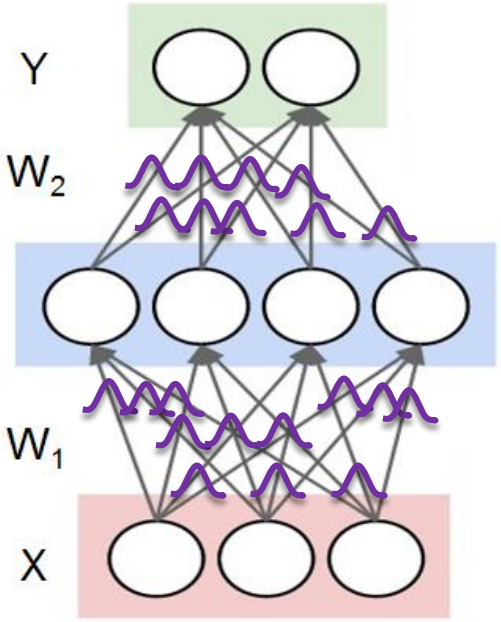


MC Dropout
Randomly drop nodes in each run
→ Ususally done during training

Dropout in test time

Yarin Gal* (2015): we learn a whole weight distribution



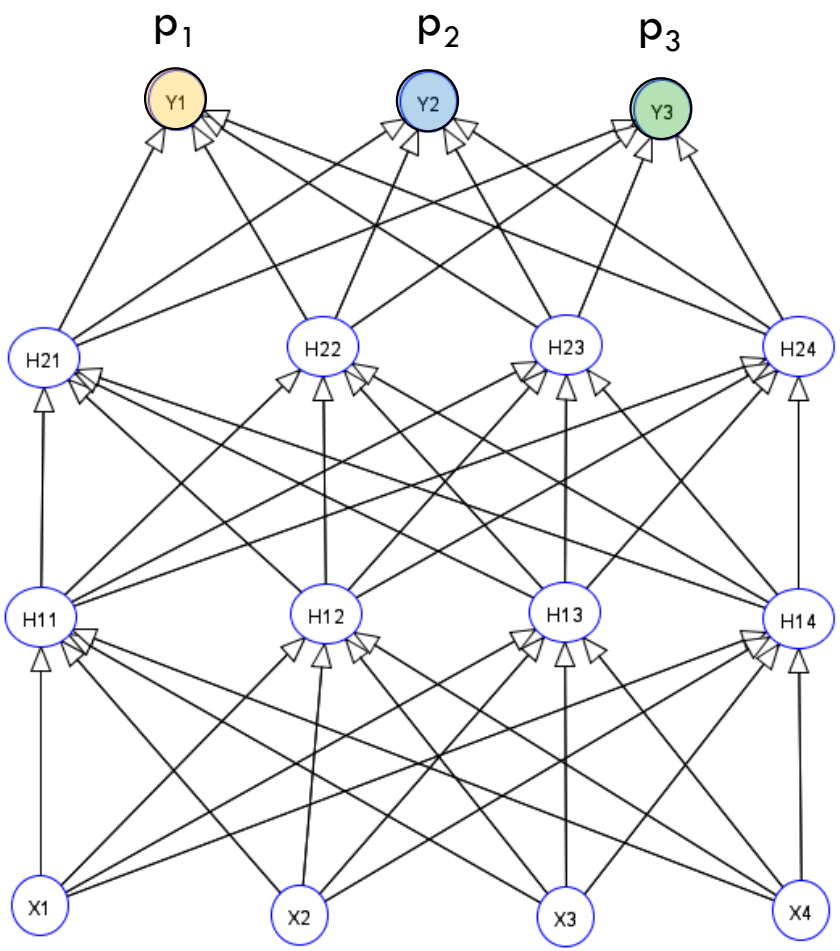


Bayesian NN

→ We should sample from weight distribution during test time

No MC Dropout

Output: probabilities for each class



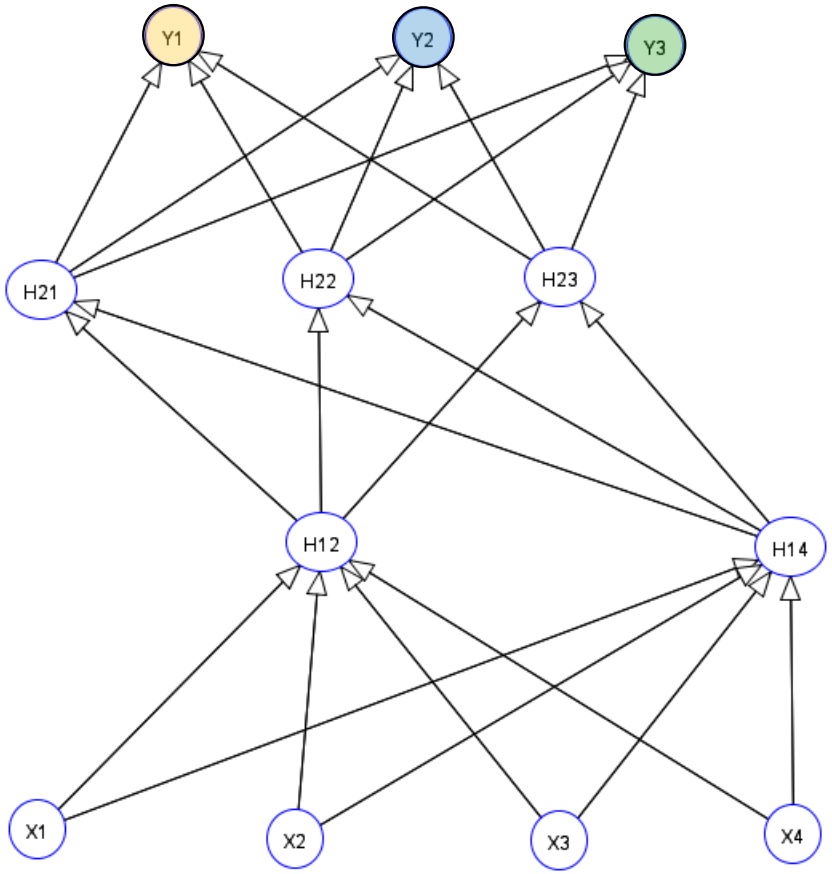
Done in training anyway

Why not use it also at test time

Input: image pixel values

Use dropout at test time: Run 1

p_1=0.08 **p_2=0.89** p_3=0.03



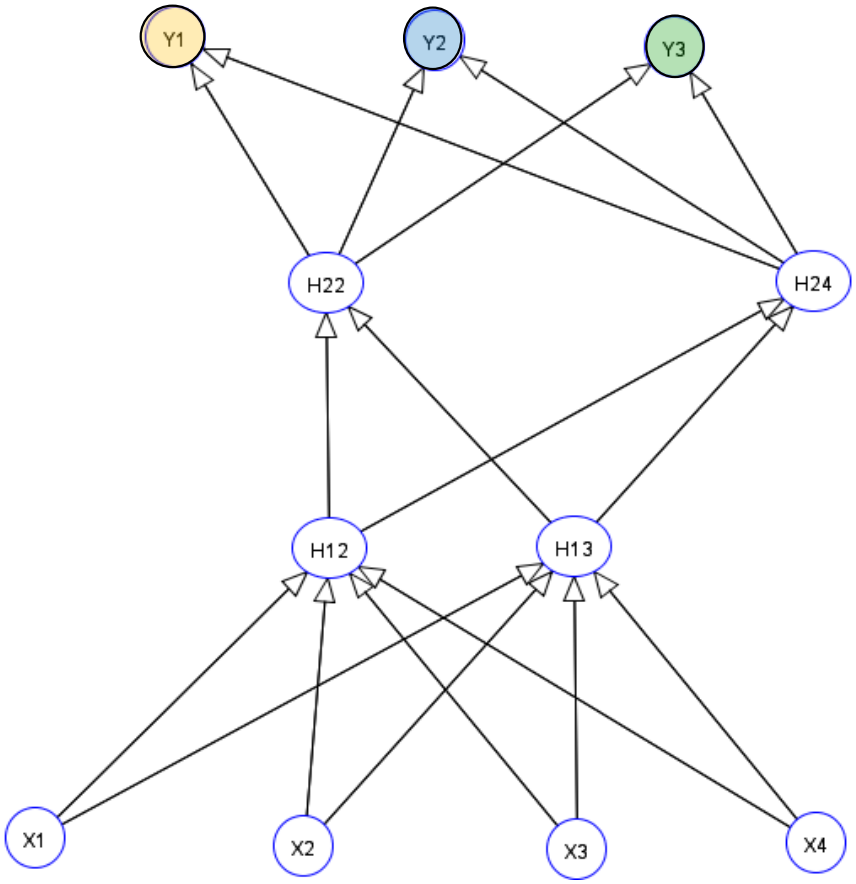
Output depends on dropout

Stochastic dropout of units

Same input image

Use dropout at test time: Run 2

p_1=0.11 **p_2=0.81** p_3=0.08



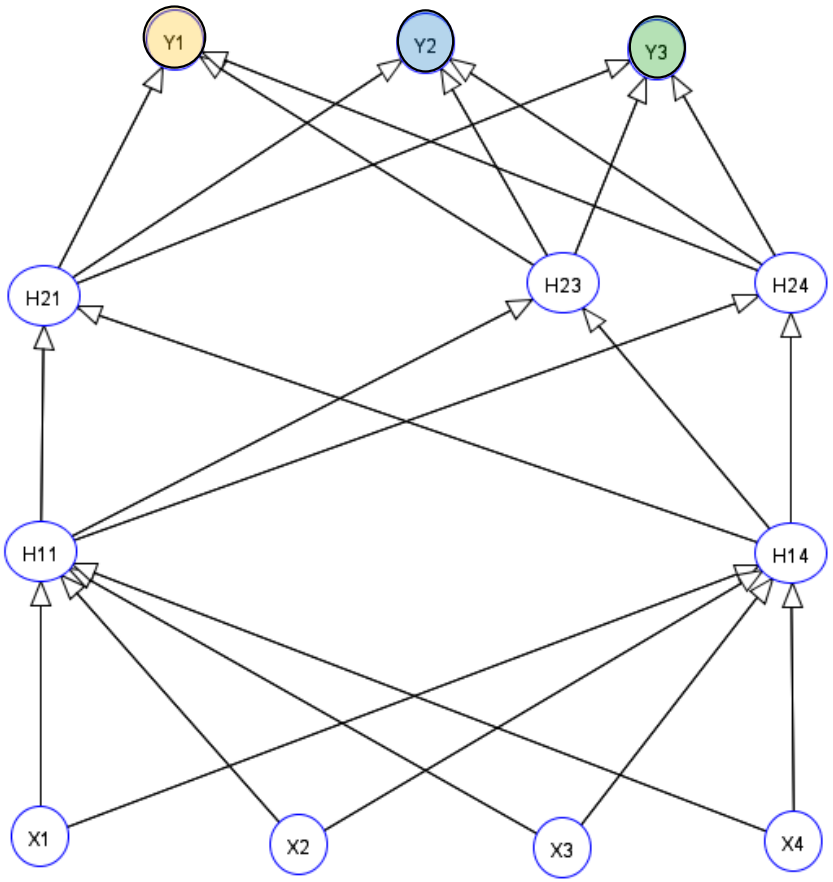
Output depends on dropout

Stochastic dropout of units

Same input image

Use dropout at test time: Run 3

p_1=0.03 **p_2=0.94** p_3=0.03



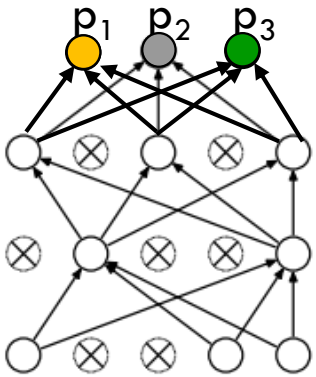
Output depends on dropout

Stochastic dropout of units

Same input image

…Repeat 500 times

Distributions of predicted probabilities



From the predicted distributions we can derive different measures

Probability estimates

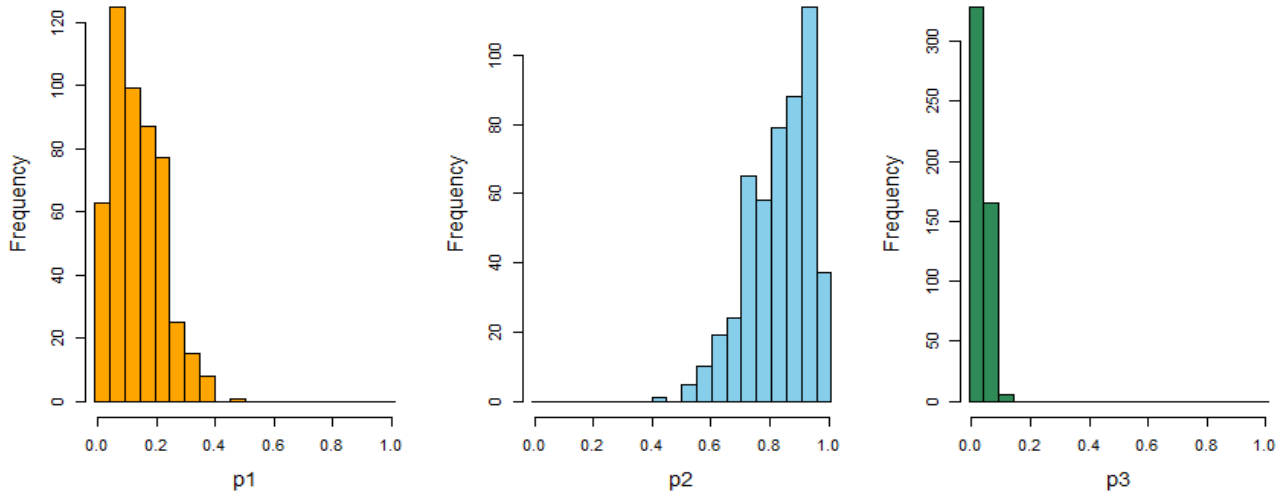
- p_{max} (when **not** using mc dropout)
- p^*_{max} (when using mc dropout)

Uncertainty estimates

- σ^* total standard deviation
- PE^* entropy

Predicted Entropy:

$$PE = -\sum_{l=1}^{L}\left(\bar{p}\_c\_l \cdot \log\left(\bar{p}\_c\_l\right)\right)$$



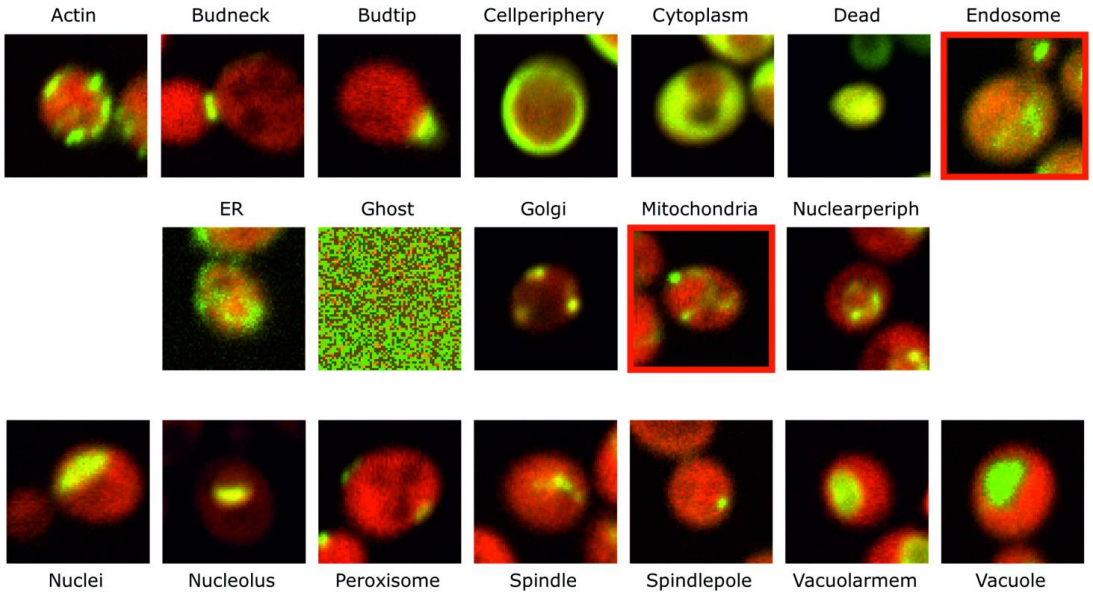
HCS application

HCS yeast protein localization data set

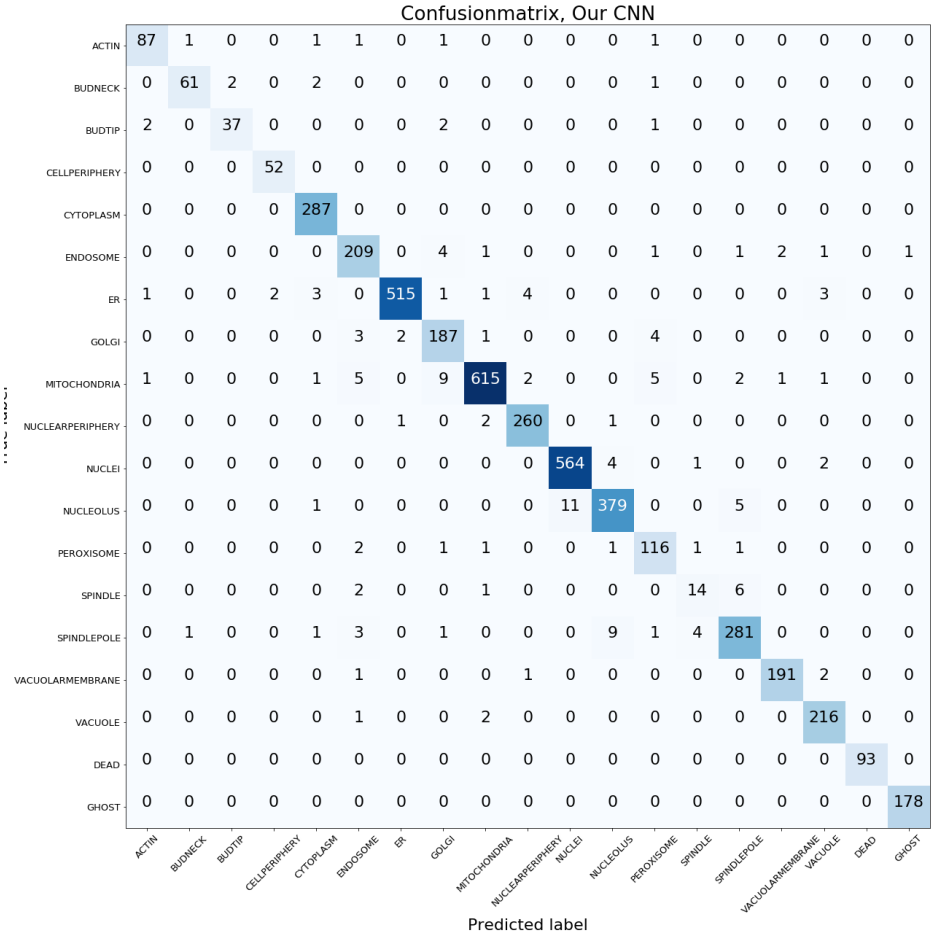
Data from budding yeast protein localizations

(all available from https://github.com/okraus/DeepLoc)

- 19 Classes
- 2 Channels
- 21882 64x64x2 segmented images in training set
- 4491 validation and 4516 for testing



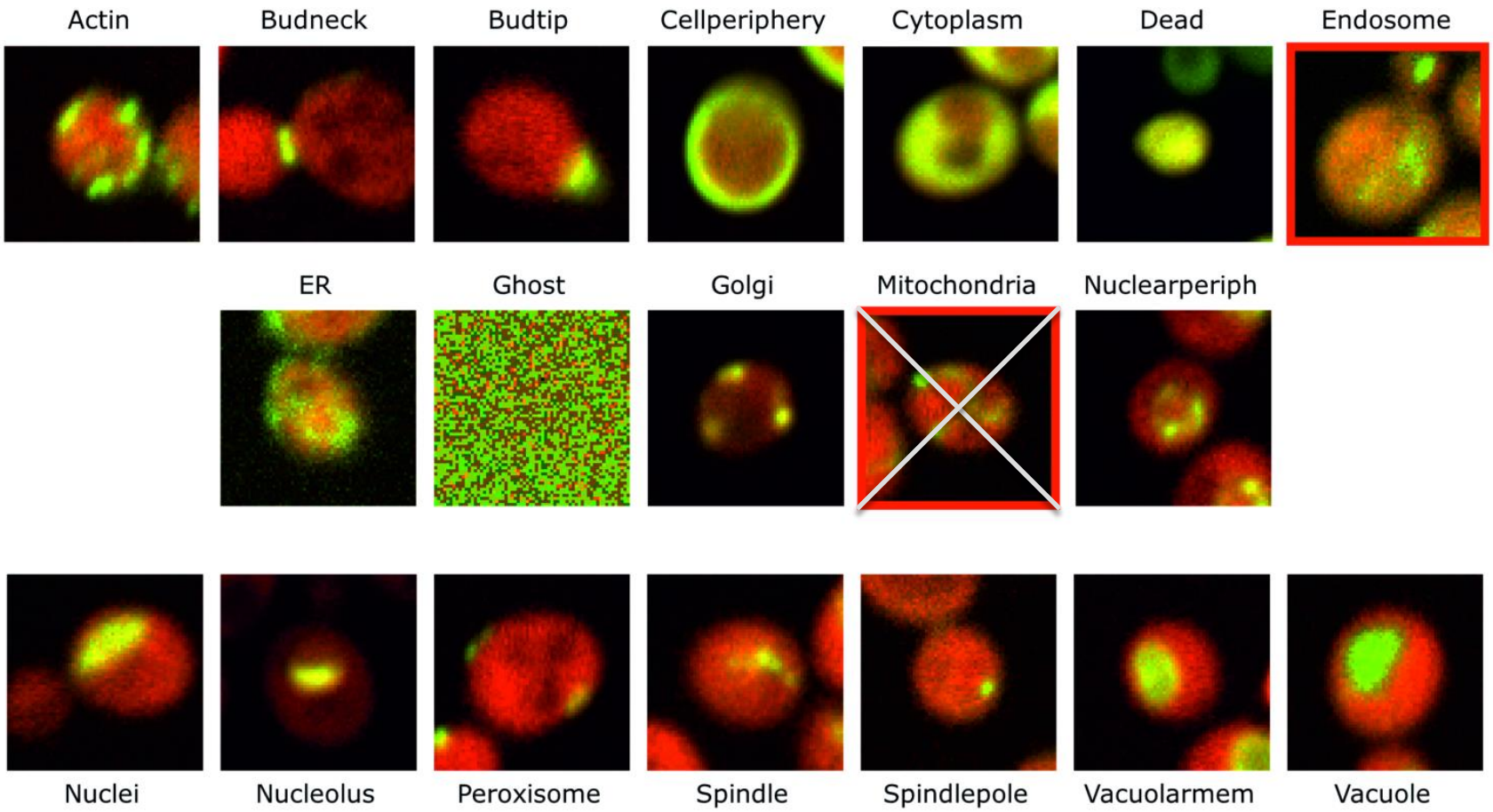
Classification performance



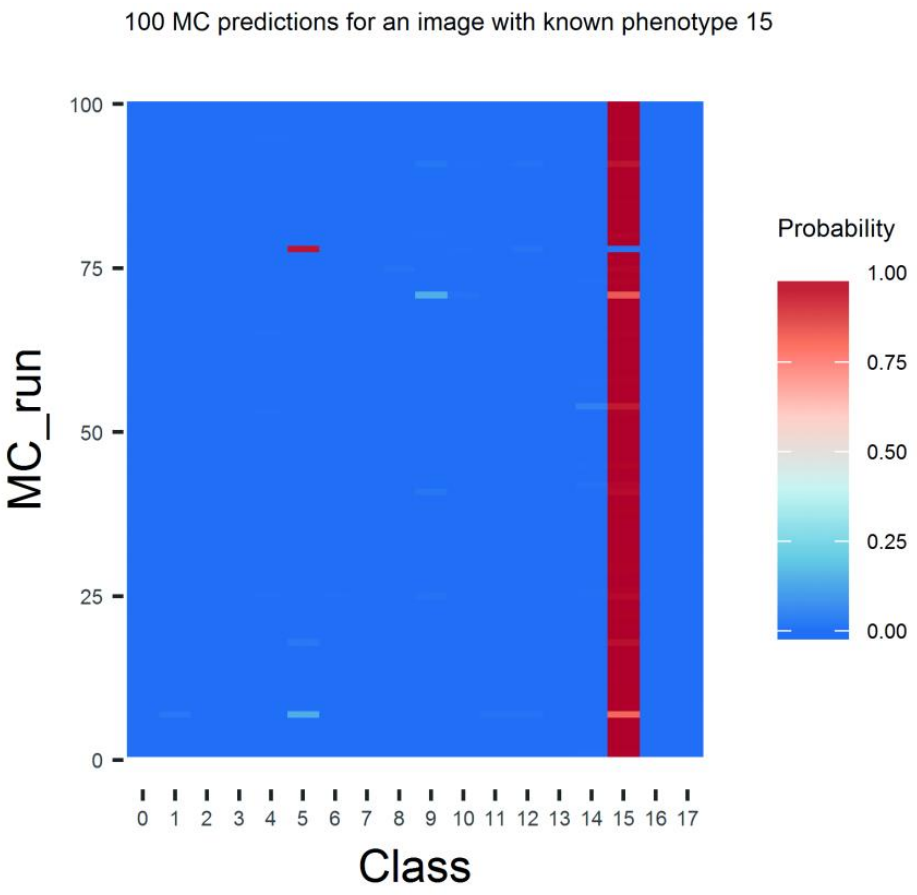
	Our_CNN	#Cells	Label
0	0.9457	92.0	ACTIN
1	0.9242	66.0	BUDNECK
2	0.881	42.0	BUDTIP
3	1.0	52.0	CELLPERIPHERY
4	1.0	287.0	CYTOPLASM
5	0.95	220.0	ENDOSOME
6	0.9717	530.0	ER
7	0.9492	197.0	GOLGI
8	0.9579	642.0	MITOCHONDRIA
9	0.9848	264.0	NUCLEARPERIPHERY
10	0.9877	571.0	NUCLEI
11	0.9571	396.0	NUCLEOLUS
12	0.9431	123.0	PEROXISOME
13	0.6087	23.0	SPINDLE
14	0.9336	301.0	SPINDLEPOLE
15	0.9795	195.0	VACUOLARMEMBRANE
16	0.9863	219.0	VACUOLE
17	1.0	93.0	DEAD
18	1.0	178.0	GHOST

Overall test acc: 96.3% [95.7%,96.8%]

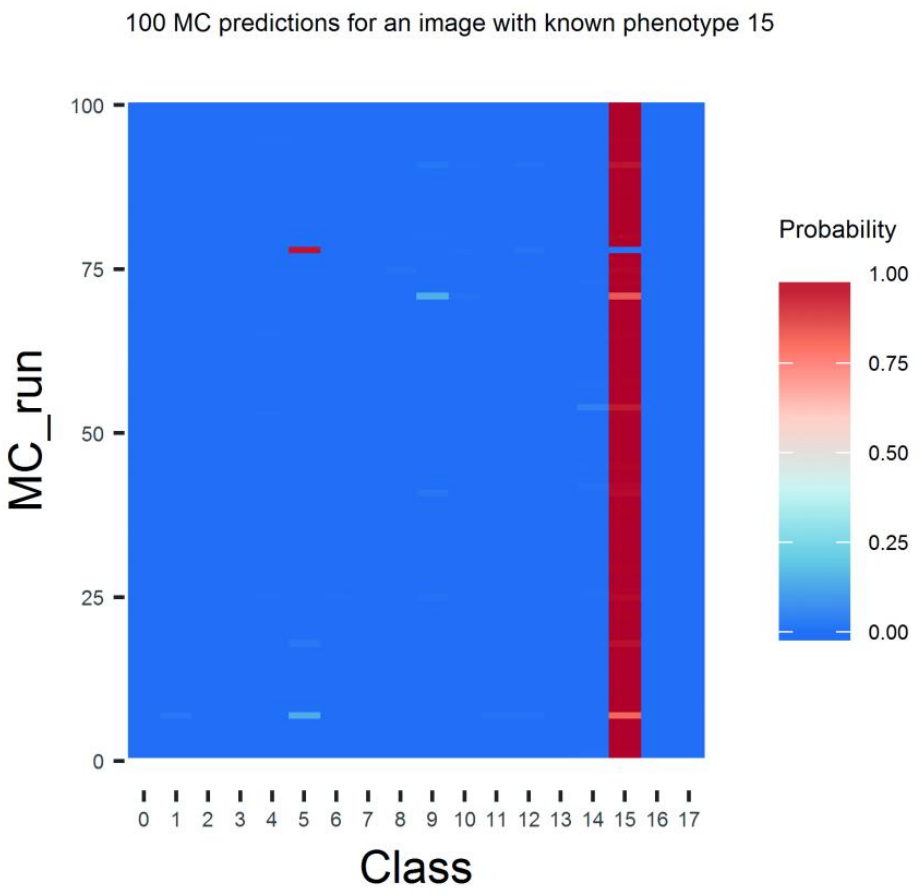
Experiment with unknown phenotype

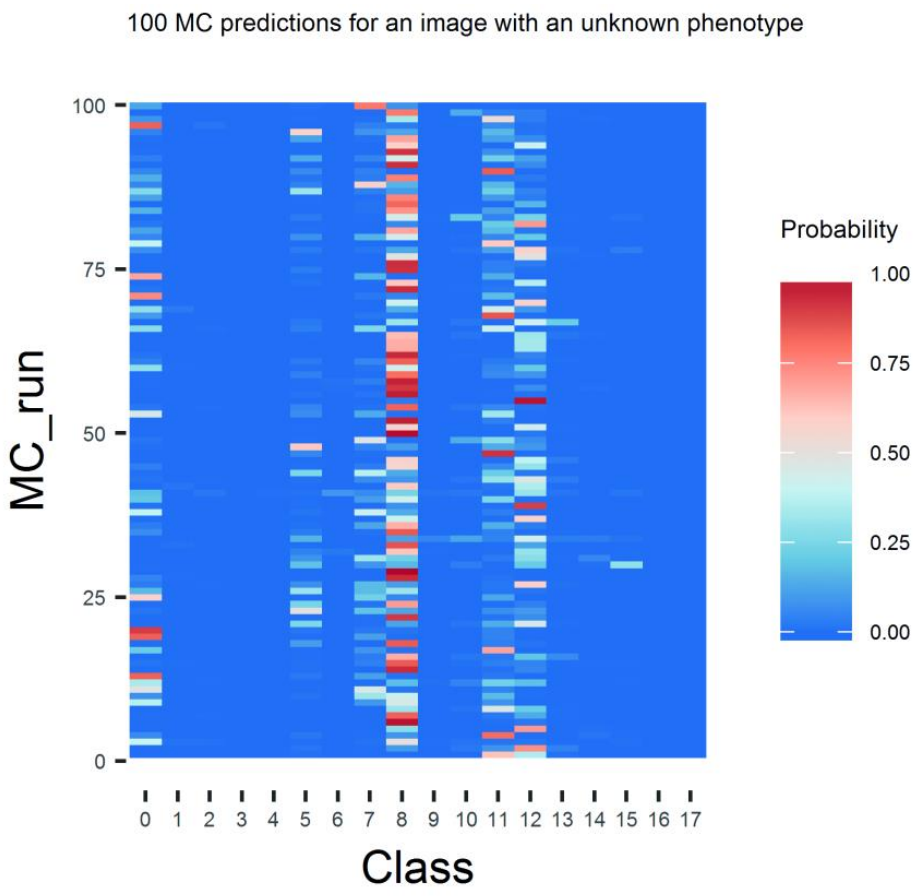


Experiment with unknown phenotype



Experiment with unknown phenotype

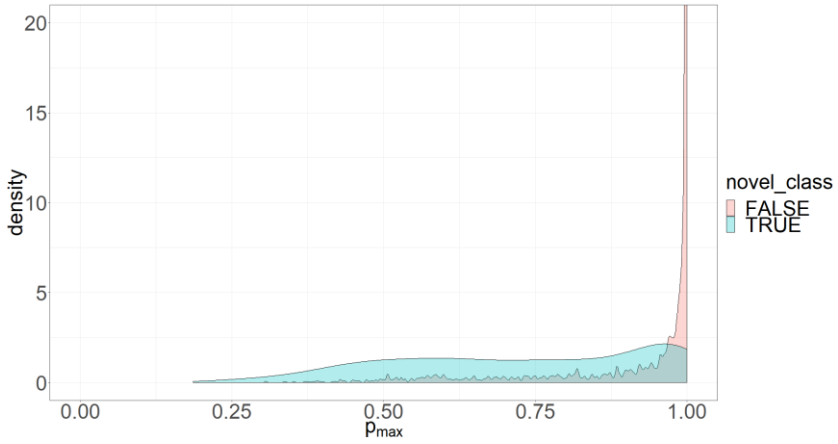




Experiment with unknown phenotype

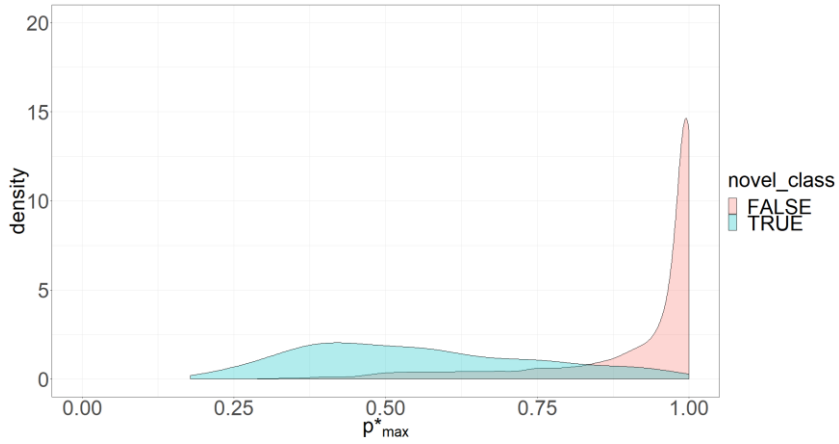
No MC Dropout (classical)
Probability estimate p_{max}

- accuracy (only not novel classes considered):
- **0.9367 [0.9286, 0.9442]**



MC Dropout
Probability estimate p^*_{max}

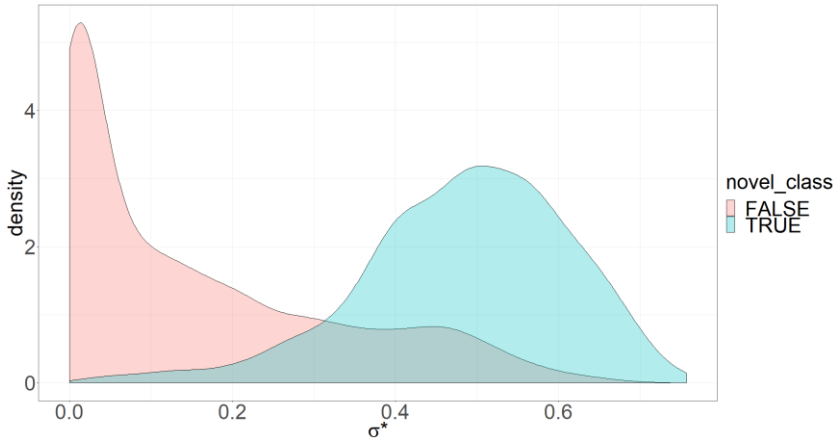
- accuracy (only not novel classes considered):
- **0.9543 [0.9472, 0.9607]**



Experiment with unknown phenotype

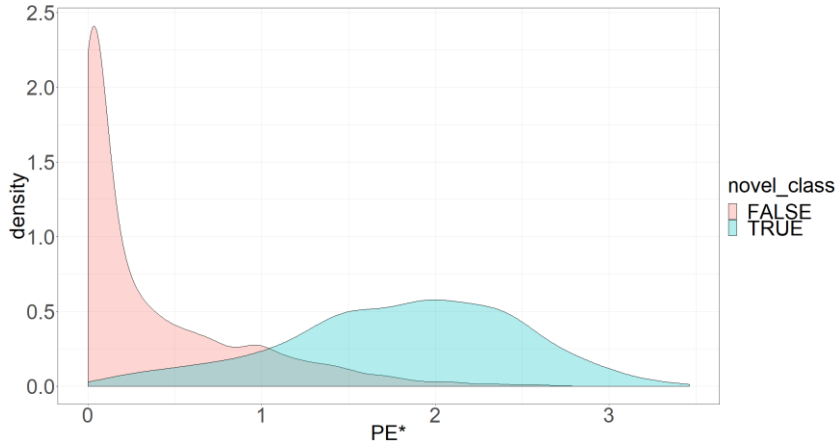
MC Dropout
Uncertainty estimate σ^*

- High σ^* indicates a class not seen in during traning
- Low σ^* indicates a confident prediction

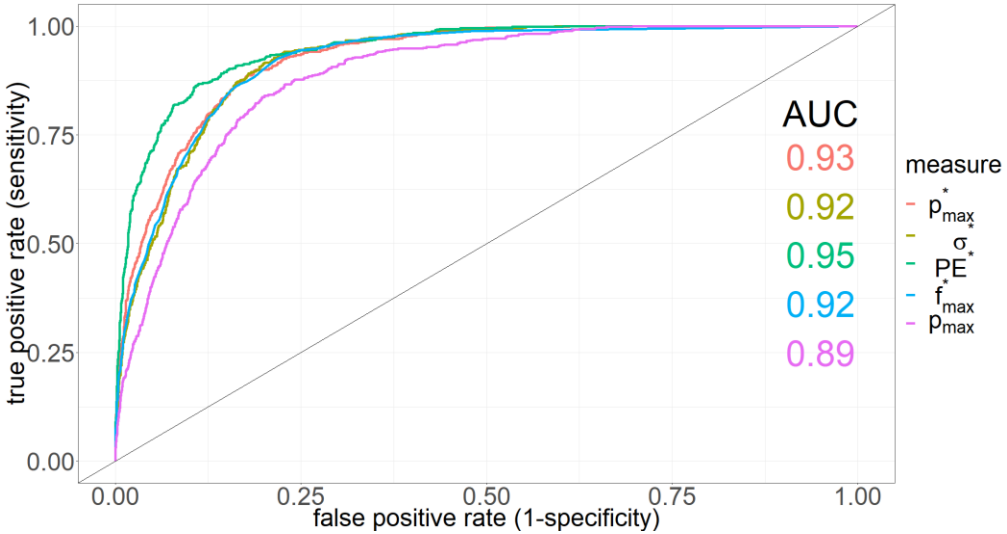


MC Dropout
Uncertainty estimate PE^*

- High PE^* indicates a class not seen in during traning
- Low PE^* indicates a confident prediction



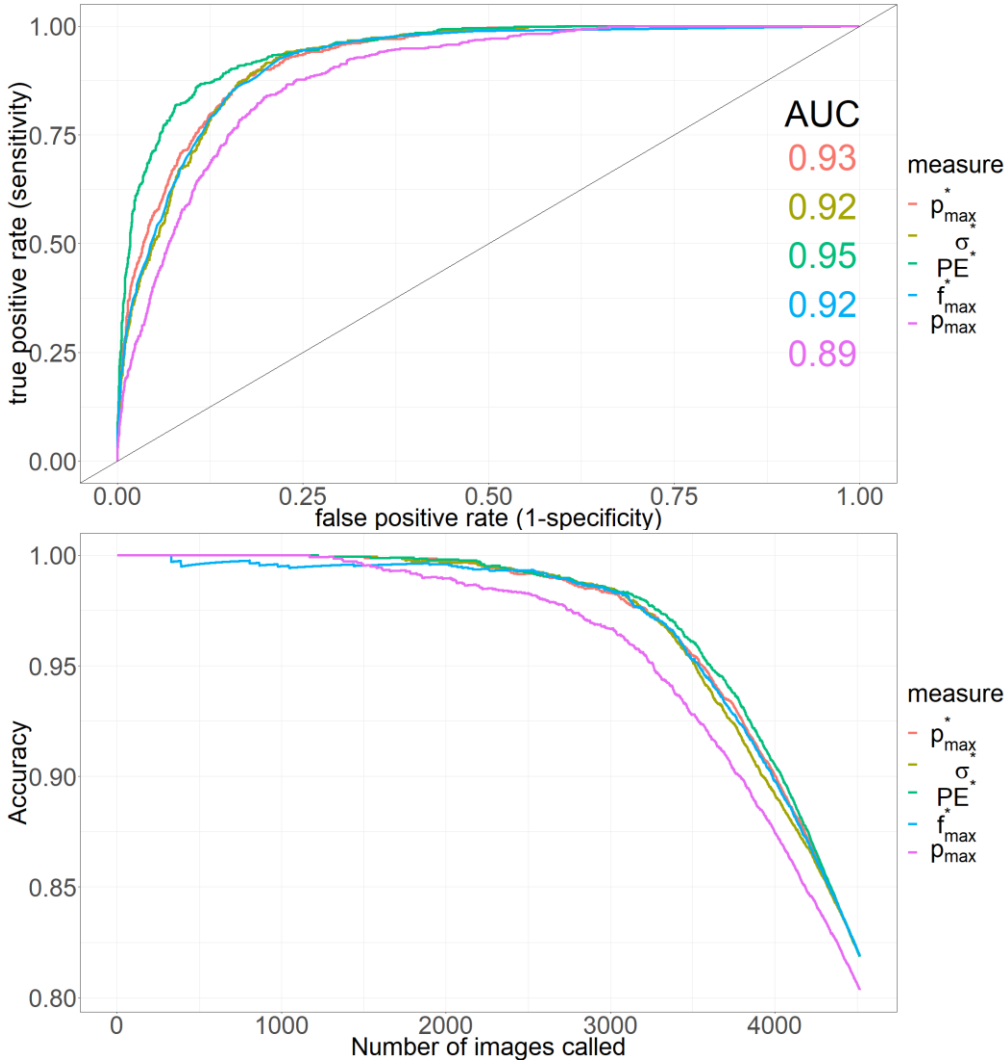
Experiment with unknown phenotypes



Discriminative power between known and unkown class

- All **MC Dropout based appoaches are superior** to the non-MC based approaches

Experiment with unknown phenotypes



Discriminative power between known and unkown class

- All **MC Dropout based appoaches are superior** to the non-MC based approaches

Filter uncertain predictions

- In the lift chart, we order the classified images according to the certainty of their call
- p_{max} which corresponds to the classical probability without MC dropout, is clearly worse compared with the MC dropout approaches

Main Rescources

- Stanford university, CS231: Convolutional Neural Networks for Visual Recognition
  - Li Fei Fei, Andrej Karpathy, Justin Johnson, Serena Yeung
- ZHAW, CAS Machine Intelligence, Modul B: Deep Learning
  - Beate Sick, Oliver Dürr
- MSE, Machine Learning FS2018
  - Oliver Dürr, Thilo Stadelmann
- Deep Learning with Python
  - François Chollet (Google)
- Uncertainty in Deep Learning
  - Yarin Gal (University of Cambridge)

Thank you, Questions?