base.camp Talks
30.04.2020

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Eugen Ruppert
ruppert at informatik.uni-hamburg.de

BASE.CAMP TALKS
MACHINE LEARNING II – DEEP LEARNING

Zoom

Zoom
Nettiquette

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Turn on your video
- If available, use a headset
- Please mute your microphone when not in the conversation
  mute on/off: Alt + a – Push to Talk: Space
- We want to record the session and make it available for people who could not join today
  We take privacy seriously. Thus, we will blur/black out your faces, so don't worry about asking questions and giving feedback!

Introduction

base.camp Talks
Idea

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- lecture series on current and general topics for Computer Science
- practically oriented
- easy to follow
- interactive examples, not just theoretical knowledge

`https://www.inf.uni-hamburg.de/inst/basecamp/events/basecamp-talks.html`

base.camp Talks
Machine Learning Talks

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

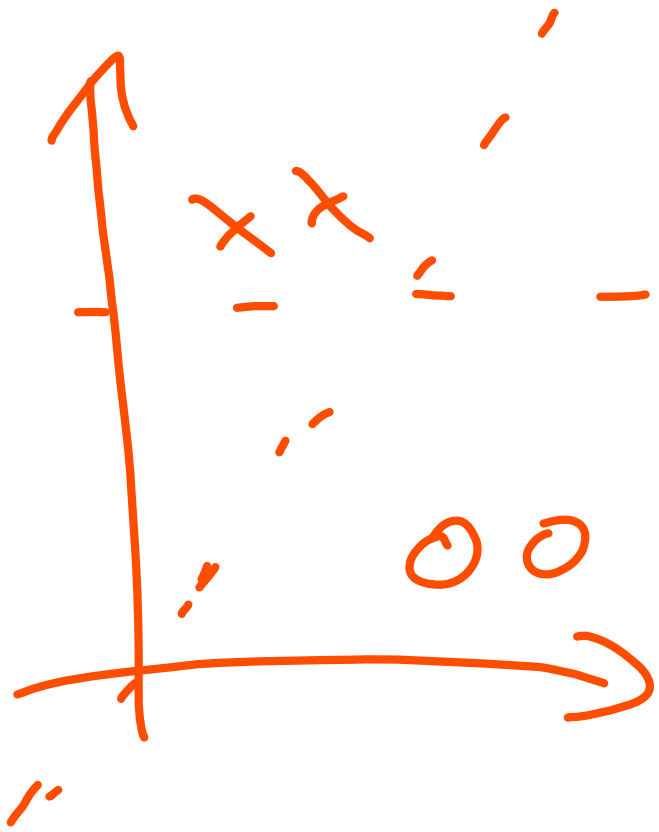
- practical introduction to ML
- understand the key terms for ML
- understand how to perform a ML project
- refresh your Machine Learning knowledge
- learn about Deep Learning
- code examples
- **not** a full lecture about ML and learning theory, partial derivatives, etc.
  There is a lecture ‘Machine Learning’ by Dr. Victor Uc Cetina

Perceptron

SGD Algorithm for Perceptron

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

for $t = 1, \ldots, T$ **do**
  for $i = 1, \ldots, n$ **do**
    if $y_i \langle x_i, w^{(t)} \rangle \leq 0$ **then**
      update $w^{(t+1)} = w^{(t)} + \eta^{(t)} y_i x_i$
    end if
  end for
end for



Perceptron example

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Consider the following data points, characterized by two features F and G; the variable Y represents the label:

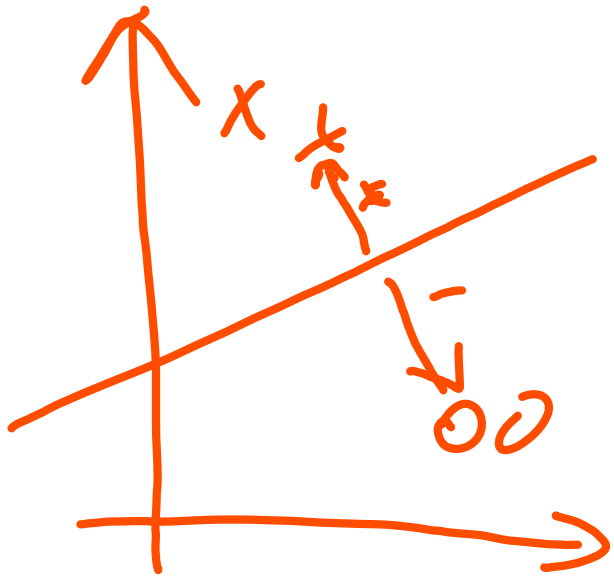
	F	G	Y
$\mathbf{x}_1$	-1	2	$-$
$\mathbf{x}_2$	3	0	$-$
$\mathbf{x}_3$	0	4	$+$
$\mathbf{x}_4$	1	5	$+$
$\mathbf{x}_5$	2	2	$+$

Perceptron example

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

In the first step we fold the distance to the hyperplane ρ into x_i.
This adds a new component -1 to every data point:

	X	Y
$\mathbf{x}_1$	$(-1, 2, -1)$	$-$
$\mathbf{x}_2$	$(3, 0, -1)$	$-$
$\mathbf{x}_3$	$(0, 4, -1)$	$+$
$\mathbf{x}_4$	$(1, 5, -1)$	$+$
$\mathbf{x}_5$	$(2, 2, -1)$	$+$



Perceptron example

Then we fold the label into the instance: $x_i \cdot y_i$

	X
$\mathbf{x}_1$	$(1, -2, 1)$
$\mathbf{x}_2$	$(-3, 0, 1)$
$\mathbf{x}_3$	$(0, 4, -1)$
$\mathbf{x}_4$	$(1, 5, -1)$
$\mathbf{x}_5$	$(2, 2, -1)$

Perceptron example

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Now we can apply the SGD algorithm to compute a weight vector w which classifies all examples x_i correctly. The learning rate η is set to 1, so we get the following update instruction for a misclassified x_i:

$$w^{(t+1)} = w^{(t)} + y_i x_i \tag{1}$$

Perceptron example

$$\sum_{i=1}^{n} w_i \cdot x_i = 1 \cdot -3 + -2 \cdot 0 + 1 \cdot 1 = -2$$

w	x_i	$x_i \cdot w$
$(0, 0, 0)$	$(1, -2, 1)$	0
$(1, -2, 1)$	$(-3, 0, 1)$	-2
$(-2, -2, 2)$	$(0, 4, -1)$	-10
$(-2, 2, 1)$	$(1, 5, -1)$	$+7$
$(-2, 2, 1)$	$(2, 2, -1)$	-1
$(0, 4, 0)$	$(1, -2, 1)$	-8
$(1, 2, 1)$	$(-3, 0, 1)$	-2
$(-2, 2, 2)$	$(0, 4, -1)$	$+6$
$(-2, 2, 2)$	$(1, 5, -1)$	$+6$
$(-2, 2, 2)$	$(2, 2, -1)$	-2
$(0, 4, 1)$	$(1, -2, 1)$	-7
$(1, 2, 2)$	$(-3, 0, 1)$	-1
$(-2, 2, 3)$	$(0, 4, -1)$	$+5$

w	x_i	$x_i \cdot w$
$(-2, 2, 3)$	$(1, 5, -1)$	$+5$
$(-2, 2, 3)$	$(2, 2, -1)$	3
$(0, 4, 2)$	$(1, -2, 1)$	-6
$(1, 2, 3)$	$(-3, 0, 1)$	0
$(-2, 2, 4)$	$(0, 4, -1)$	$+4$
$(-2, 2, 4)$	$(1, 5, -1)$	$+4$
$(-2, 2, 4)$	$(2, 2, -1)$	-4
$(0, 4, 3)$	$(1, -2, 1)$	-5
$(1, 2, 4)$	$(-3, 0, 1)$	$+1$
$(1, 2, 4)$	$(0, 4, -1)$	$+4$
$(1, 2, 4)$	$(1, 5, -1)$	$+7$
$(1, 2, 4)$	$(2, 2, -1)$	$+2$
$(1, 2, 4)$	$(1, -2, 1)$	$+1$

Perceptron example

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

At this point, all five examples are correctly classified, and we are done. In terms of the original data points (attributes *F* and *G*), the prediction is:

$$f(x) = x \cdot (1, 2) - 4 \tag{2}$$

We arrive at (2) if we compute the dot product of *x* and *w* and extract the constant -4, which is the same for all possible data points:

$$\begin{aligned} f(x) &= \langle x, w \rangle \\ &= \langle (F, G, -1), (1, 2, 4) \rangle \\ &= \langle (F, G), (1, 2) \rangle - 4 \end{aligned} \tag{3}$$

Perceptron example

We can confirm that the prediction (2) correctly classifies all the original examples:

$\boldsymbol{x}$	$f(\boldsymbol{x})$	Y
$(-1, 2)$	-1	$-$
$(3, 0)$	-1	$-$
$(0, 4)$	$+4$	$+$
$(1, 5)$	$+7$	$+$
$(2, 2)$	$+2$	$+$

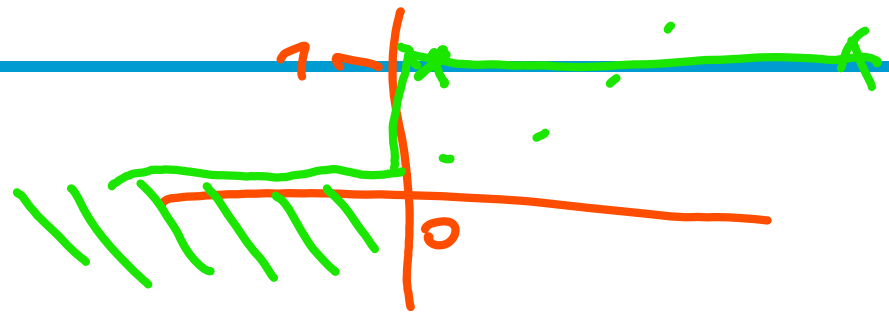
Machine Learning

Supervised ML Setup

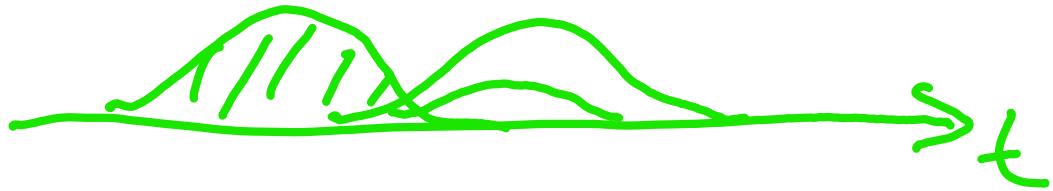
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- problem identification
- data collection and annotation
- selection of ML algorithm
- training and evaluation

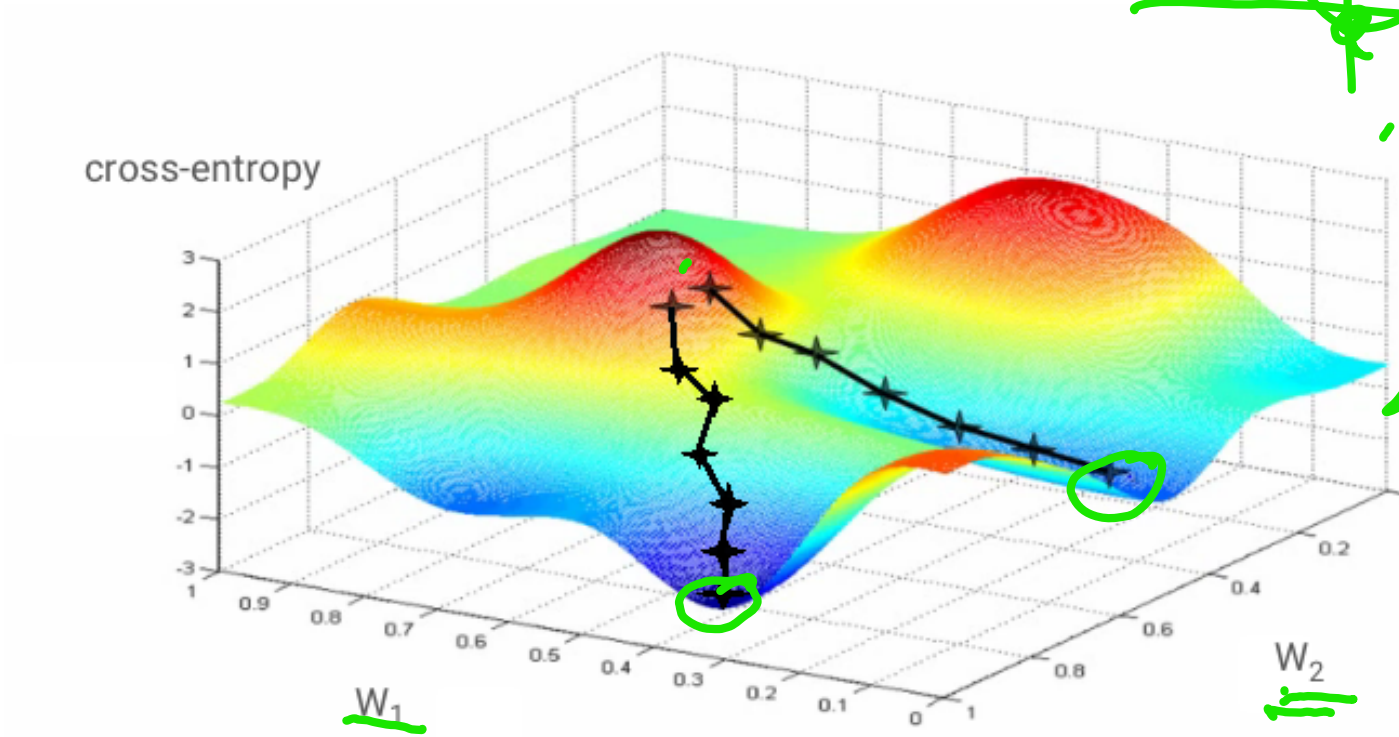
Training
Loss or Cost Function



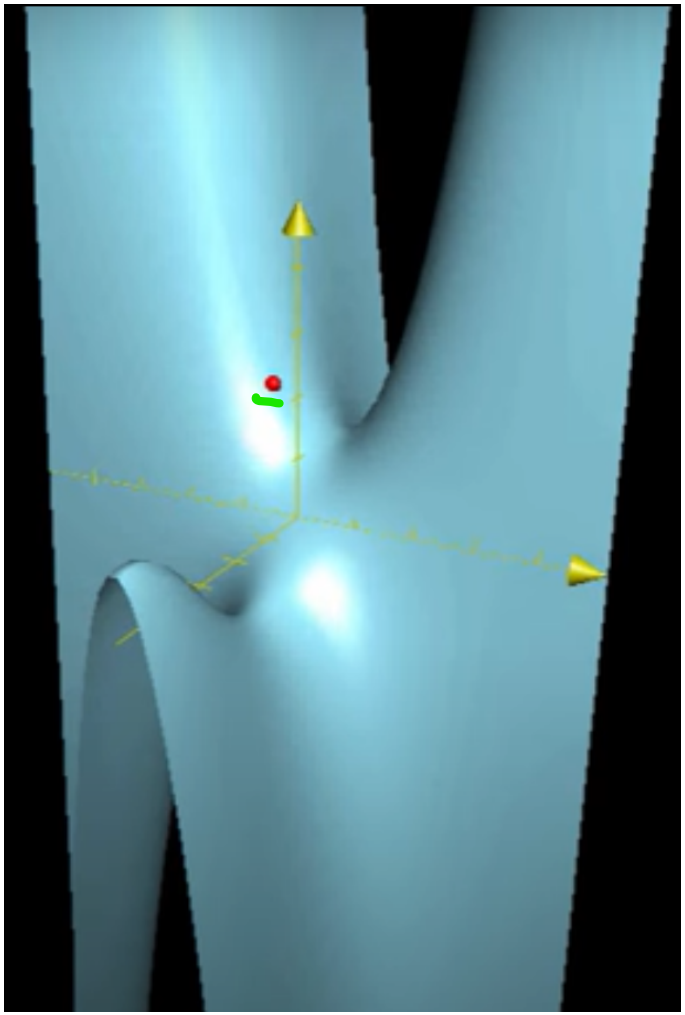
- errors during training are accumulated into a Loss
- ideal Loss functon scales with the magnitude of error
- training algorithm tries to minimize the Loss
- assumption: as all data is drawn independently, this will reduce errors on test data
- gradient of the error can be computed, so that a Gradient Descent algorithm can improve the model



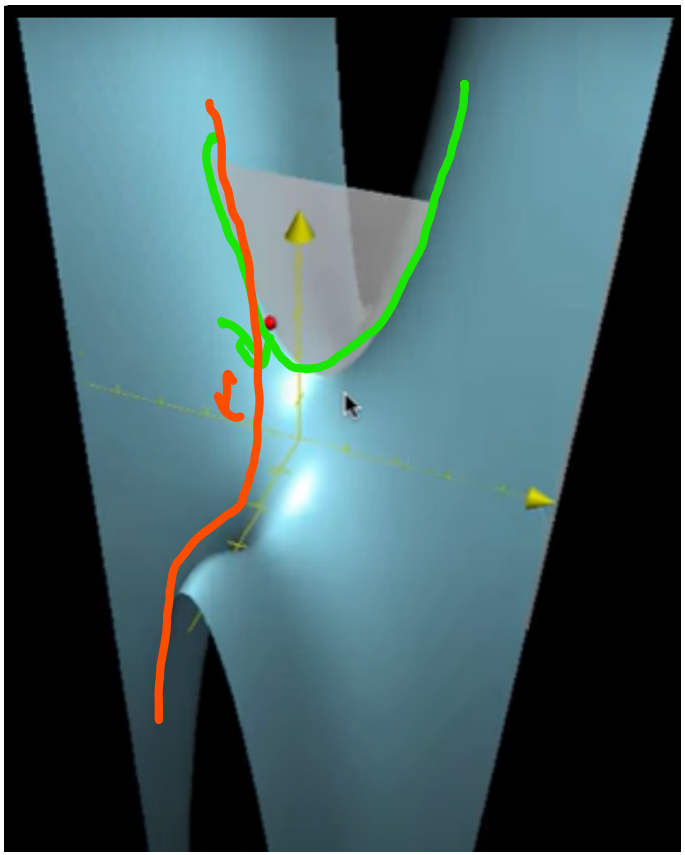
Stochastic Gradient Descent



Stochastic Gradient Descent
Partial Derivatives

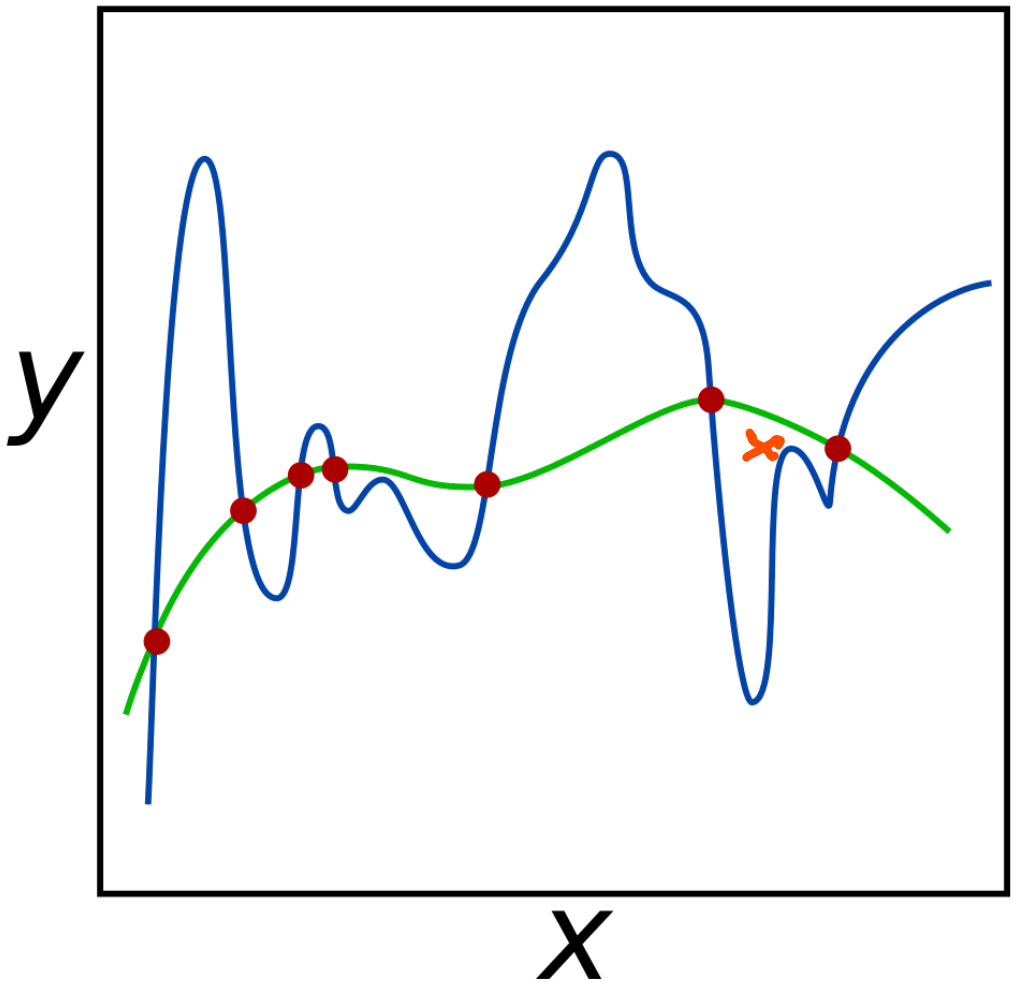


Stochastic Gradient Descent
Partial Derivatives

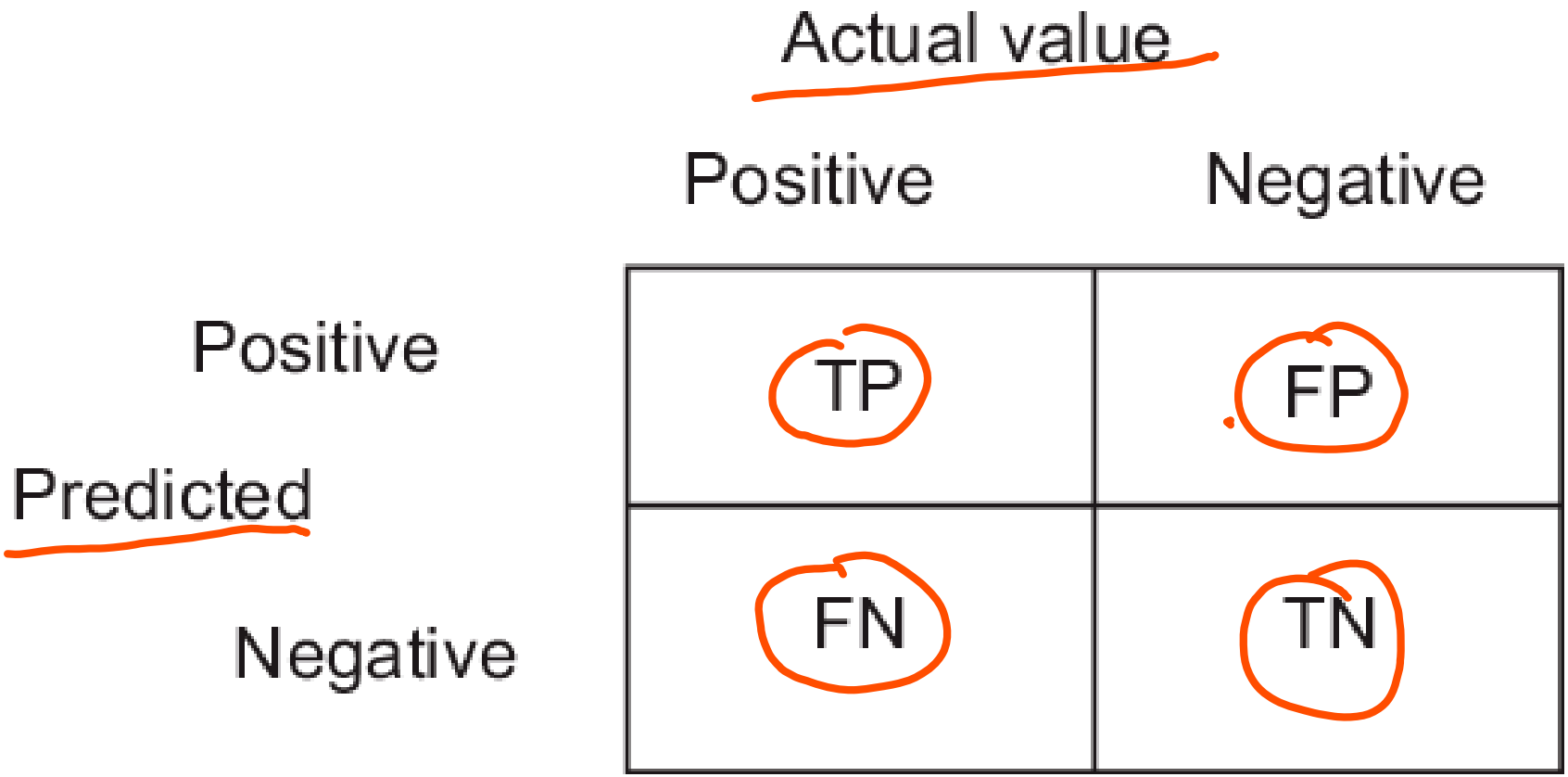


https://towardsdatascience.com/
a-quick-introduction-to-derivatives-for-machine-learning-people

Overfitting and Regularization



Evaluation


Actual value
Positive
Negative
Positive
Predicted
Negative
TP
FP
FN
TN

Evaluation

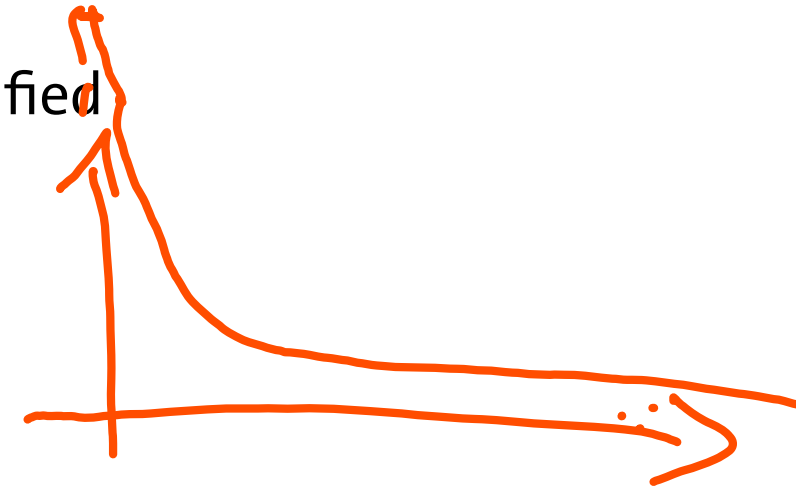
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Accuracy = TP / TP + FP + TN + FN
- Precision = TP / TP + FP
- Recall = TP / TP + FN
- F-score = 2 * Pr * Rec / Prec + Rec

Baselines

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- complex ML models need to be justified
- baselines can provide justification
- simple baselines
  - random
  - most frequent class
- strong baselines
  - perceptron
  - related work
  - simple neural networks



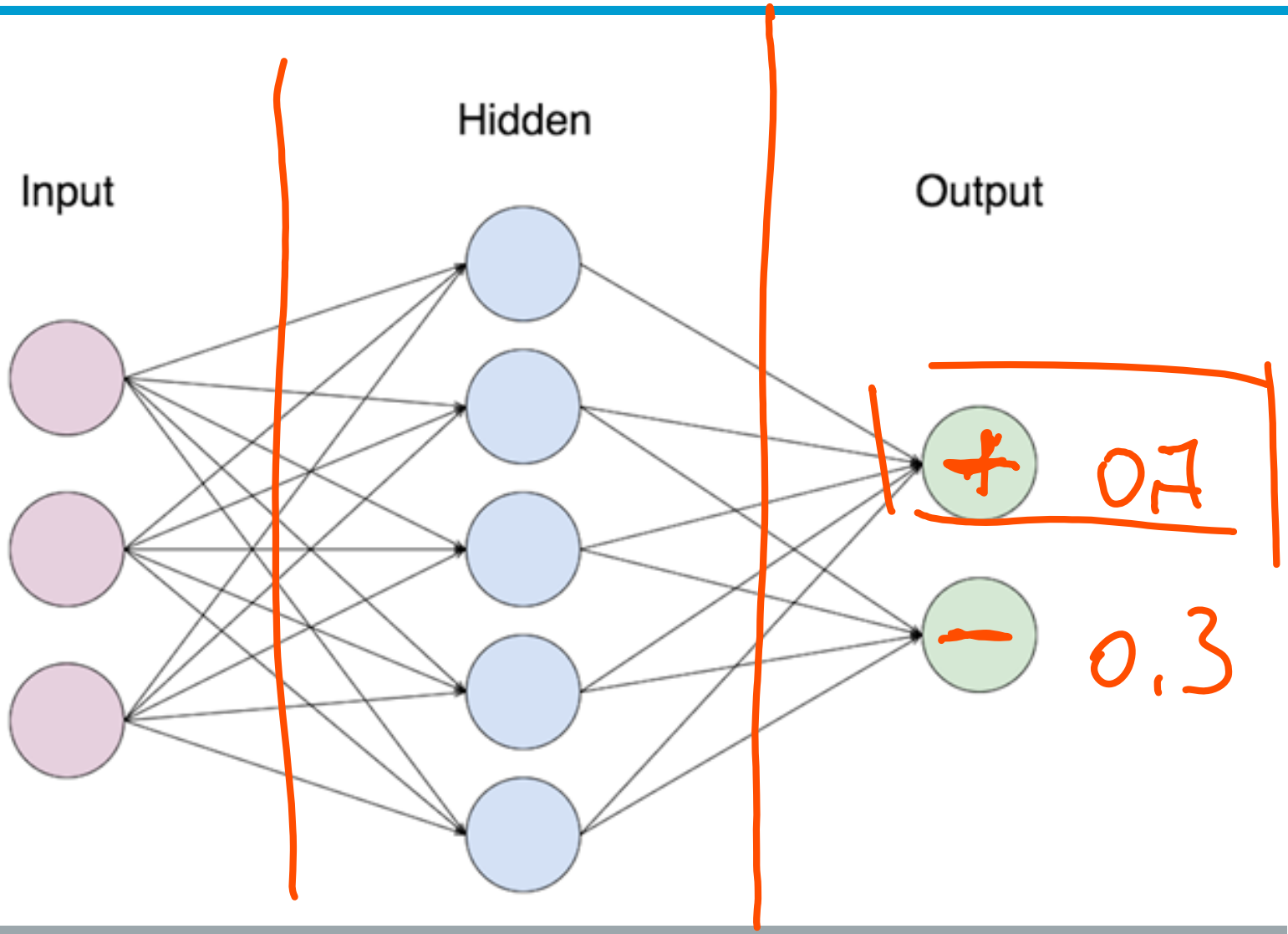
Deep Learning

Deep Learning

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

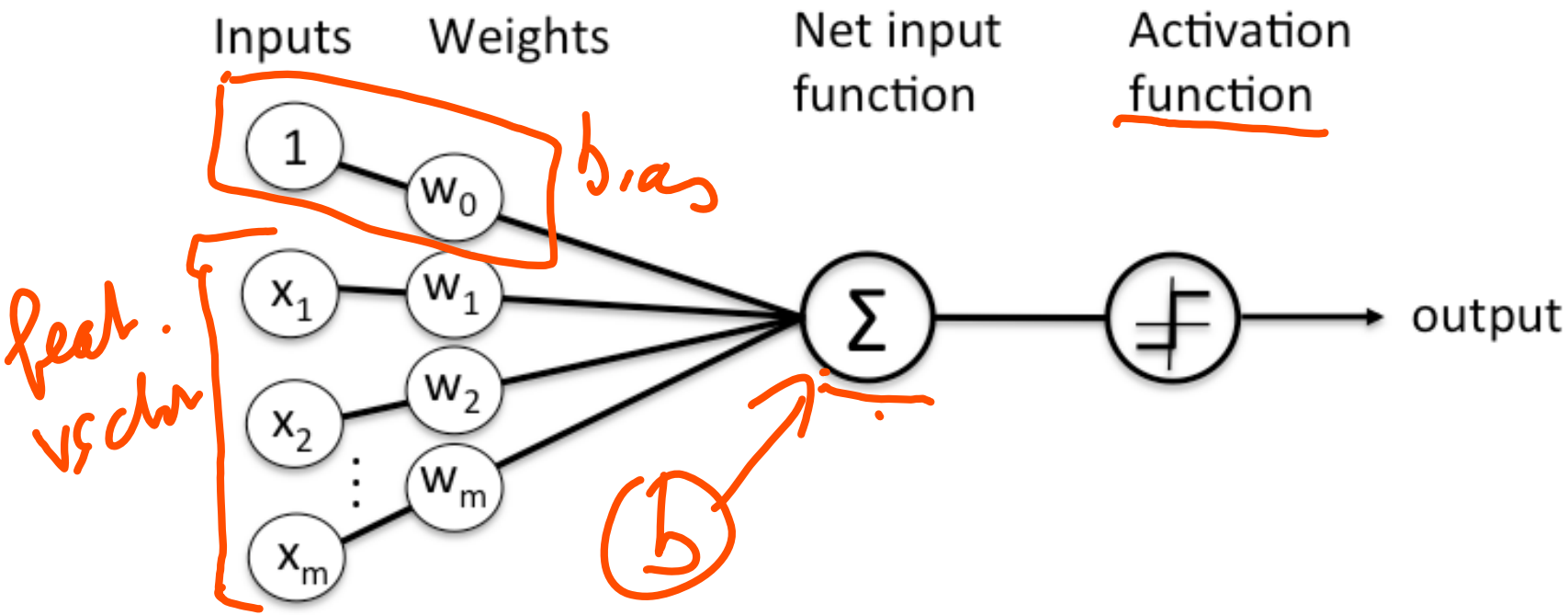
- ML with Deep Neural Networks
- inspired by neurons in the brain
- ‘automatic’ feature extraction
- several layers of processing, e.g. image recognition: lines – shapes – eyes – face – person
- large parameter space – benefits greatly from GPUs

Neural Network

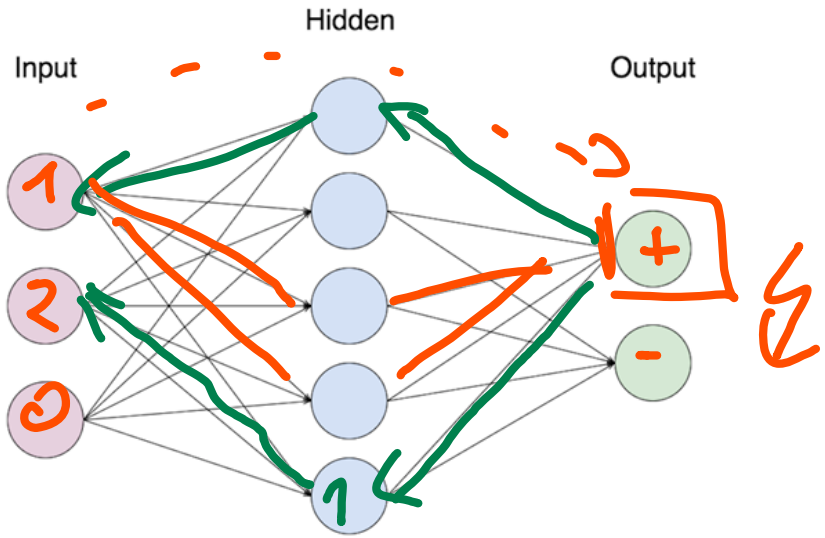


Neuron / Perceptron

- weighted sum of inputs
- bias
- activation function

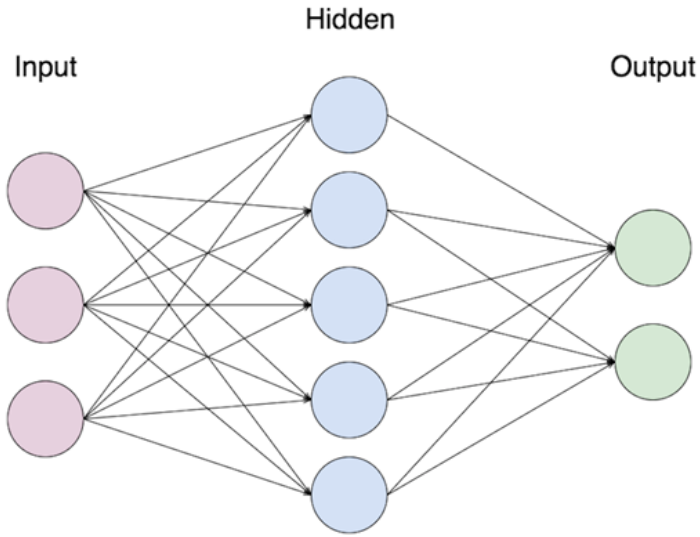


Forward and Back Propagation



- input is fed through the network (forward propagation)
- error is computed on the output layer
- error is back-propagated through all layers
- gradient of the error is identified, weights are updated

Batches and Epochs



- we compile loss on batches of input elements
  e.g. 1 batch = 100 input examples
- more robustness, stronger direction of the gradientIterations

- 1 epoch = all input examples have been processed
- with complex neural nets, many epochs are required

MNIST – Deep Learning “Hello World”

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MNIST

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Yann Lecun (Google Labs) 1998
- 60,000 training digits
- 10,000 evaluation digits
- dimension: 28x28 pixels (784 overall)
- `http://yann.lecun.com/exdb/mnist/`

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. ”Gradient-based learning applied to document recognition.” Proceedings of the IEEE, 86(11):2278-2324, November 1998.

Getting Started with Tensorflow

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

```
pip3 install tensorflow
```

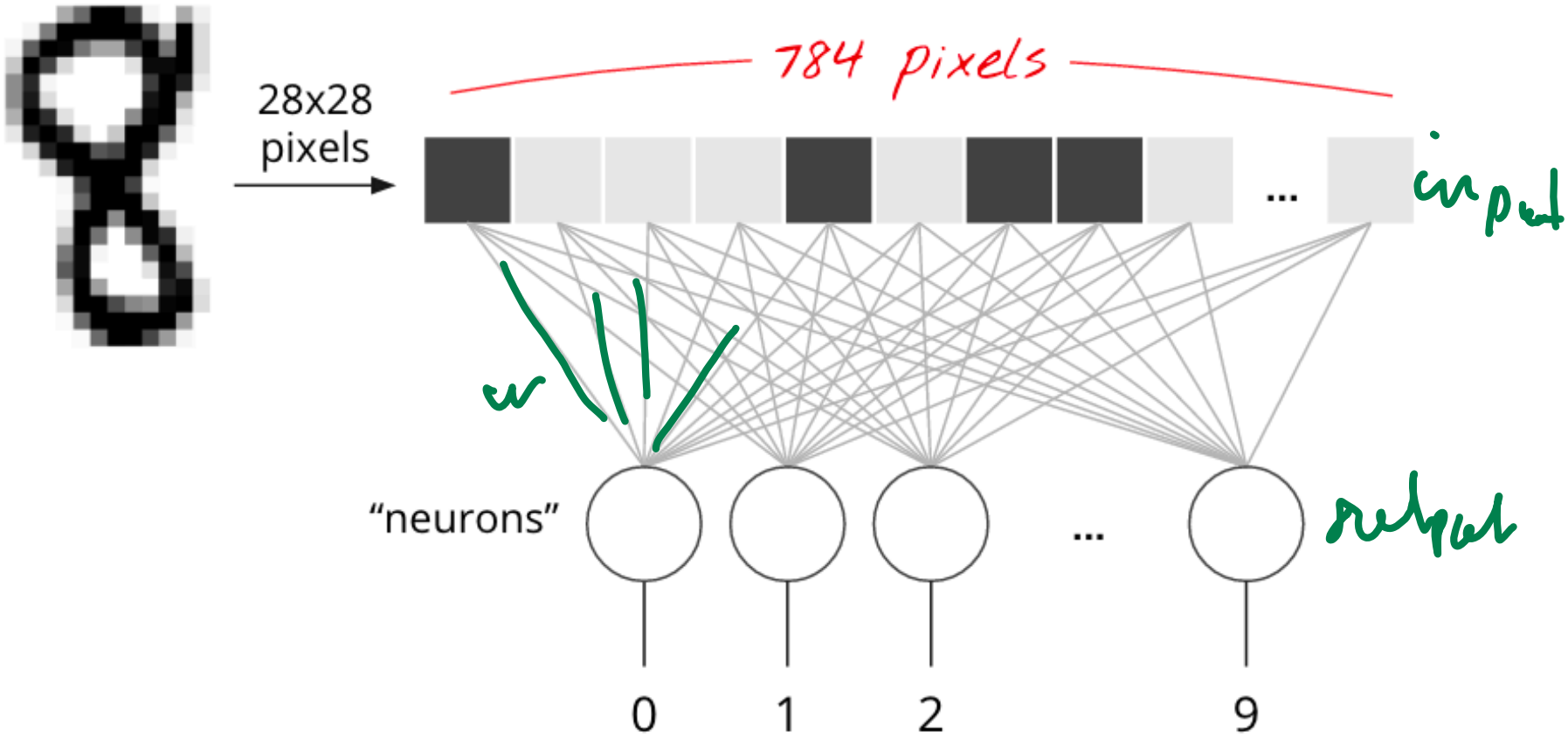
Tensor: multi-dimensional table:

grayscale image [28, 28]

color image [28, 28, 3]

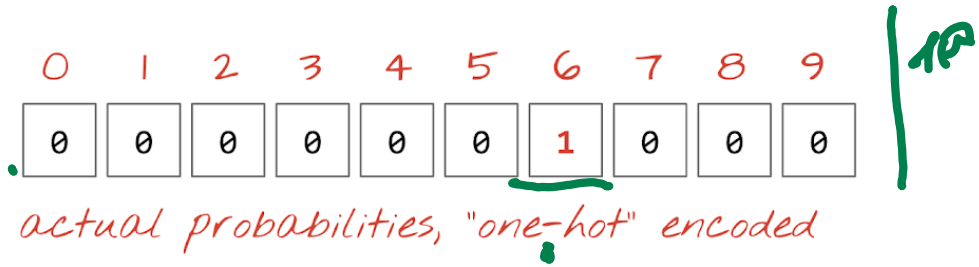
training batch [100, 28, 28, 3]

Dense Layer Classifier

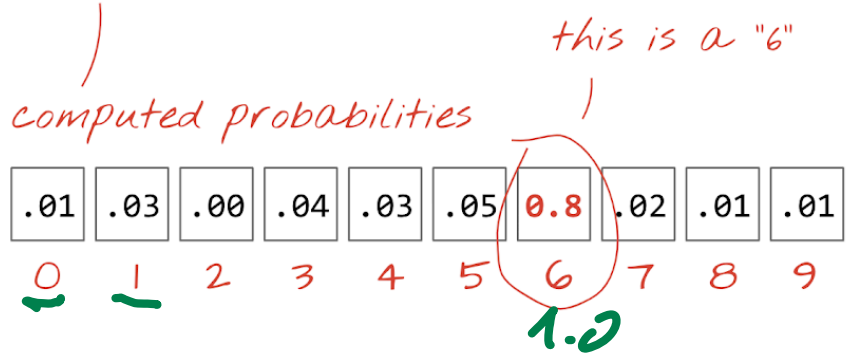


Cross Entropy Loss

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG



Cross entropy: $-\sum Y'_i . log(Y_i)$

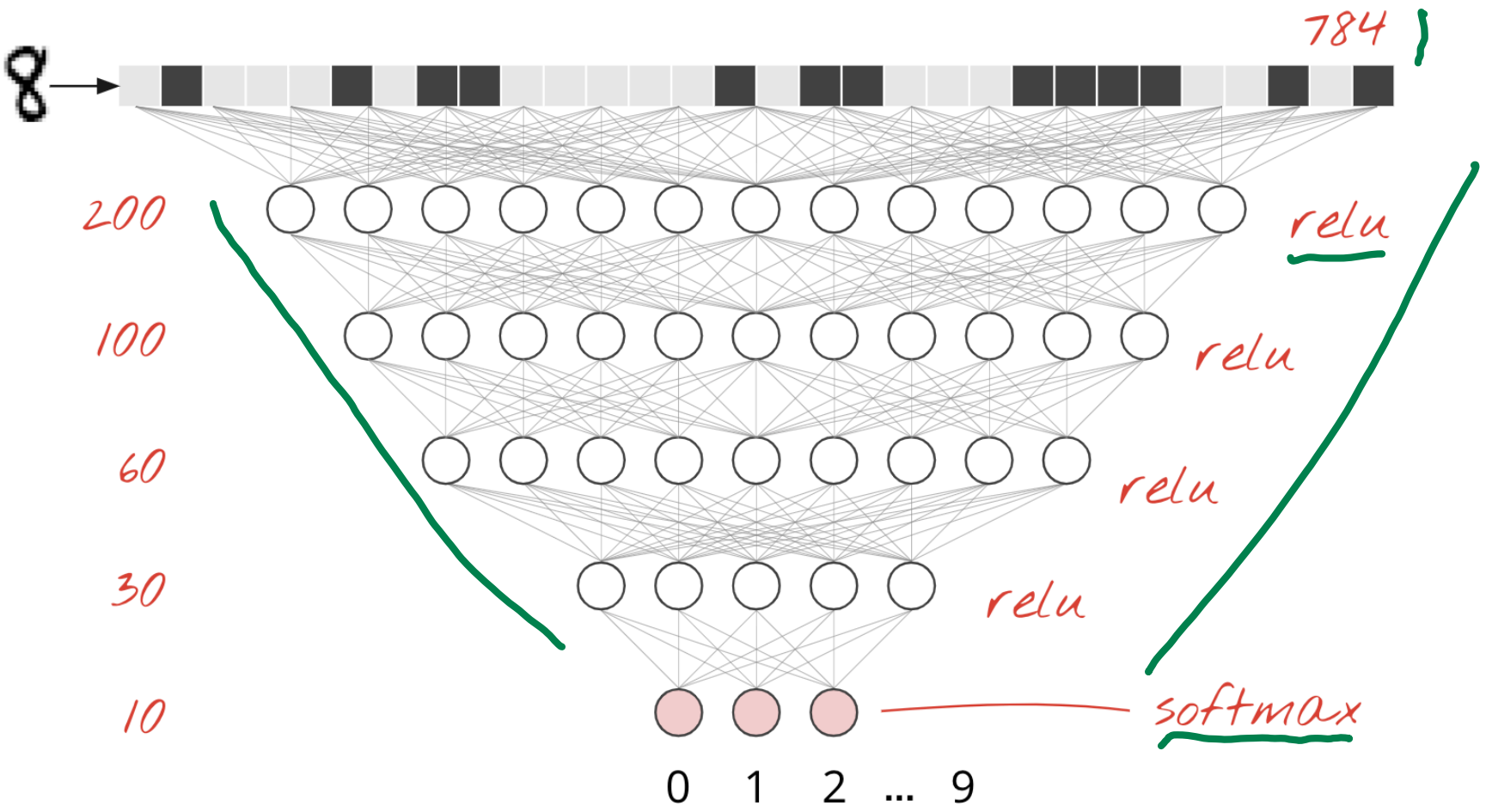


Off we Go!

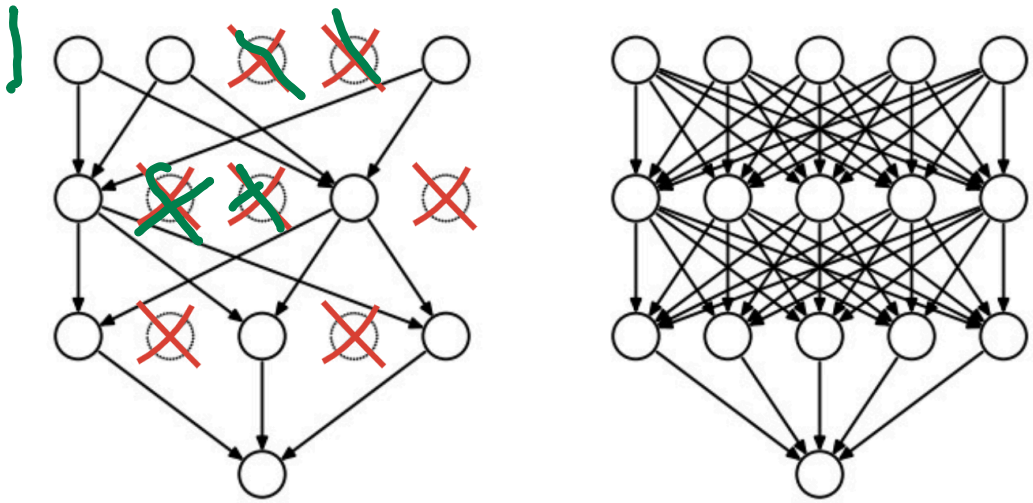
https://git.informatik.uni-hamburg.de/base.camp/ml-deep-learning/number-recognition-1.py

We need to go deeper!

Deep Neural Network



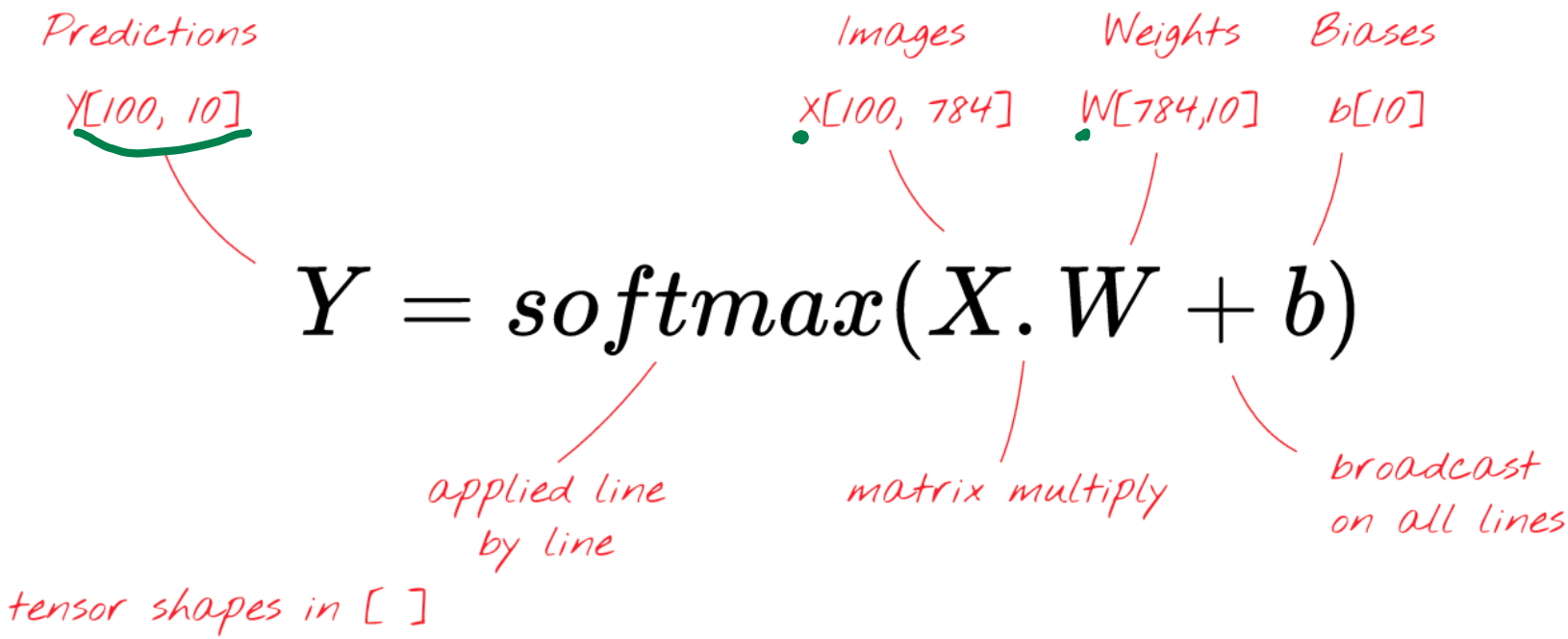
Dropout



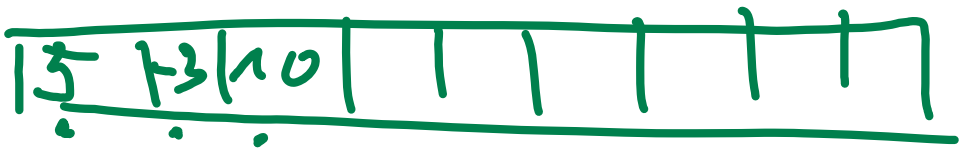
- in deep networks, neurons can provide ‘bad’ output
- following layers need to correct it
- dropout deactivates random nodes
- network becomes more robust

Activation Functions

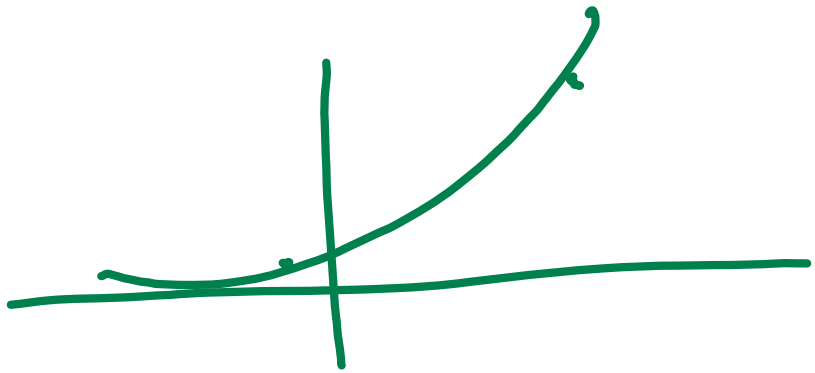
Softmax Activation



Softmax Activation

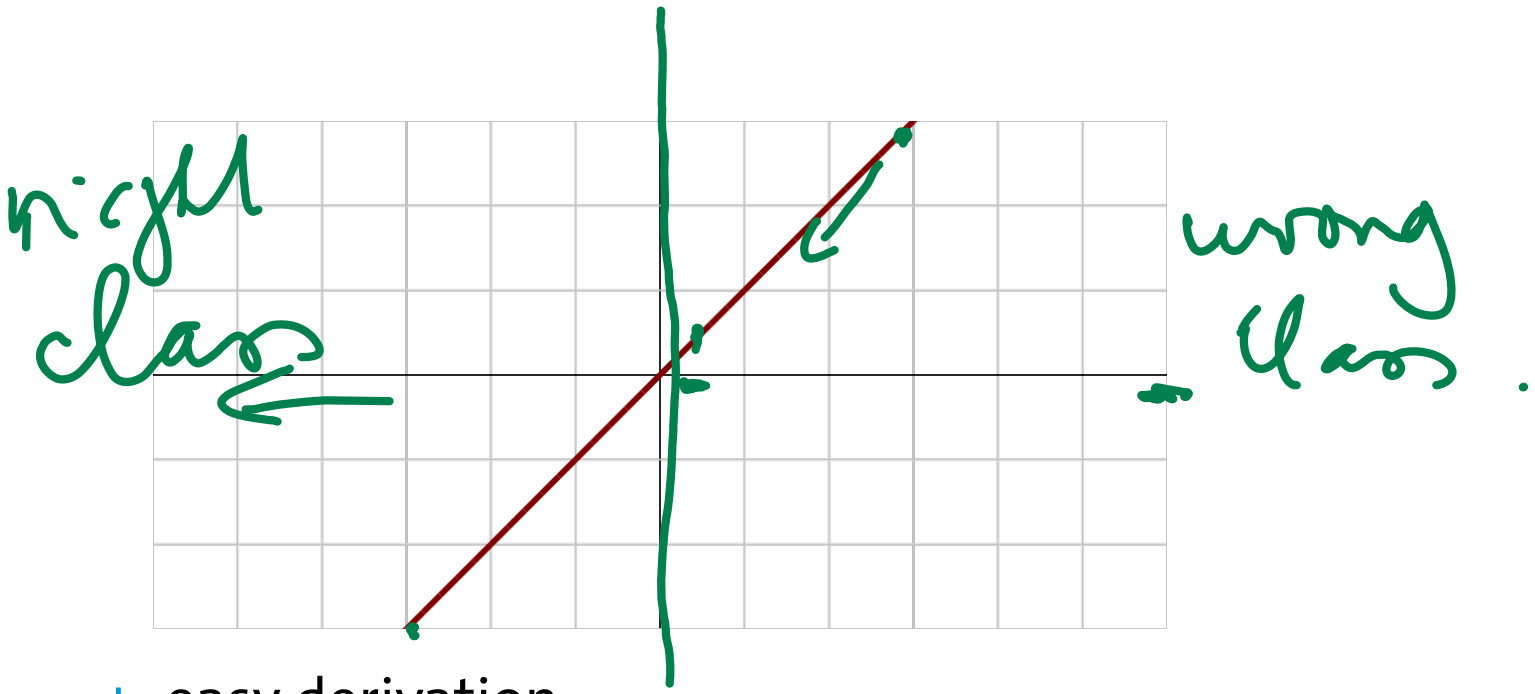


$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$



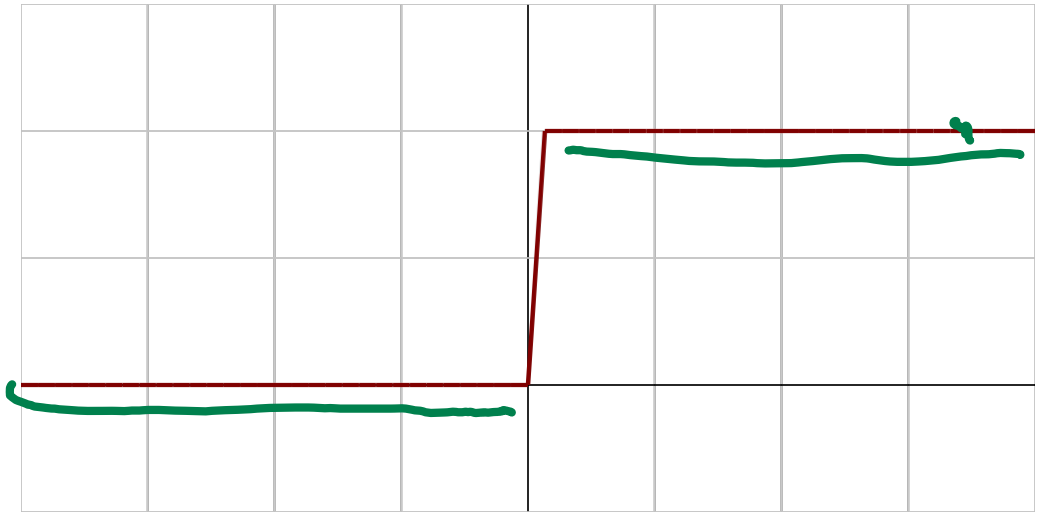
- on output layer
- transforms activations into probability distribution
- exponential funciton to increase contrast and give the highest activation a value close to 1.0

Linear/Identity Activation



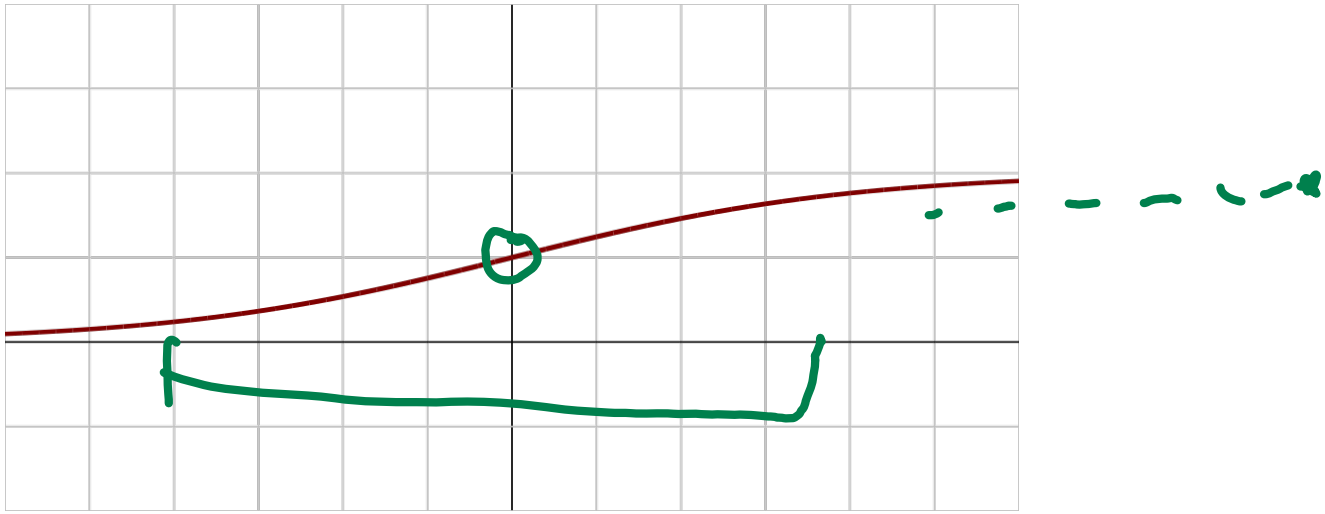
- \+ easy derivation
- – uniformity does not help differentiation
- – correct classifications contribute to learning
  → overfitting

Step Activation



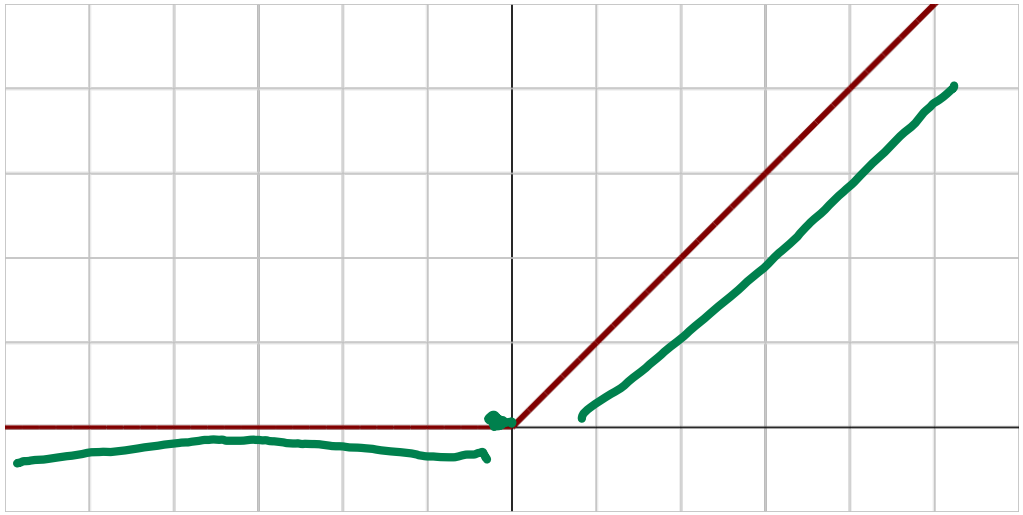
+ error class explicitly defined
+ only error class activates the neuron
– no learning is possible, gradient has no direction

Sigmoid Activation



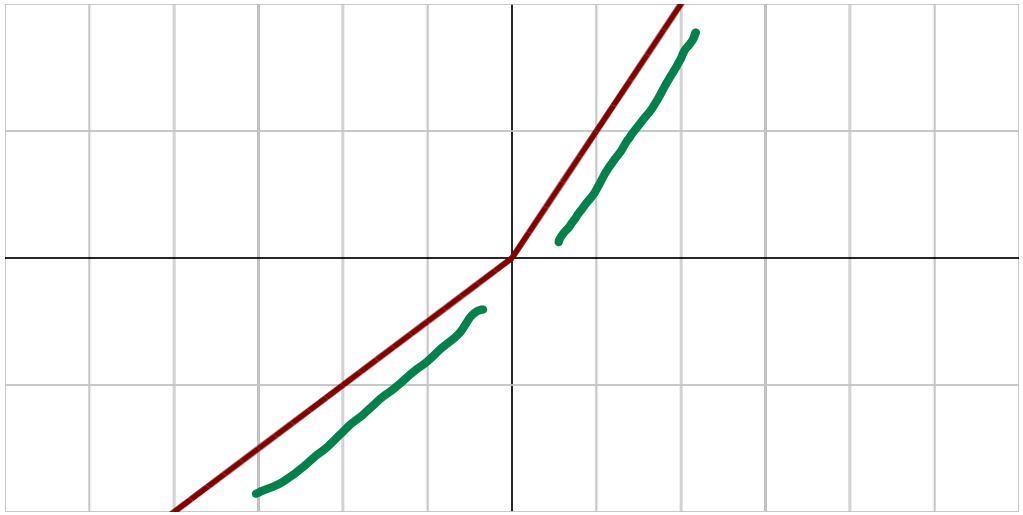
+ non-linear
+ strong differentiation at decision boundary
+ similar to brain neuron activation
– gradient becomes zero on large errors

Rectified Linear Unit (ReLU) Activation



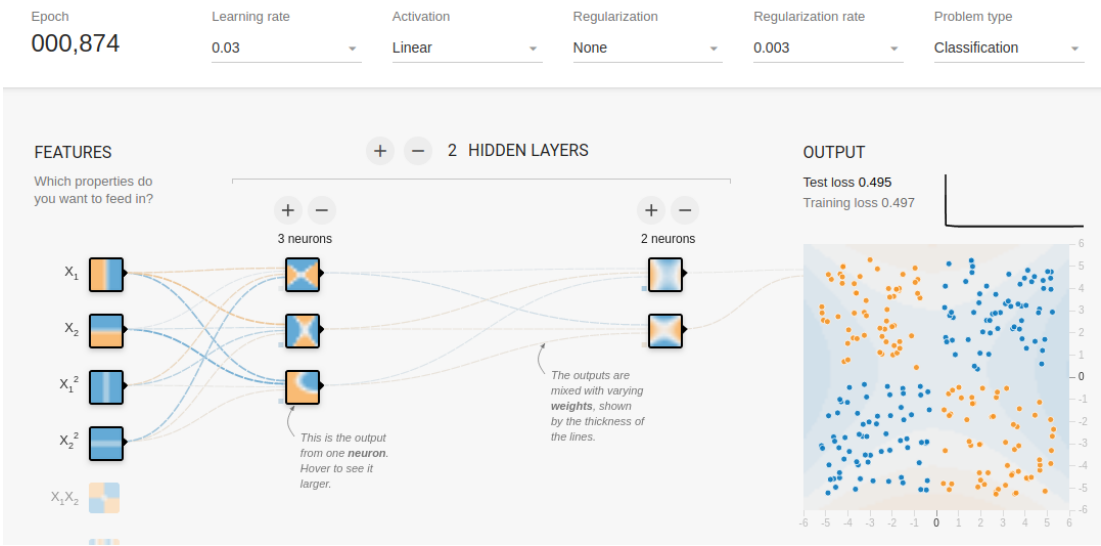
+ non-linear
+ can handle all errors margins
+ usually faster conversion than sigmoid
– cannot learn from positive examples

Leaky ReLU Activation



+ non-linear
+ can handle all errors margins
+ similar to brain neuron activation
+ usually faster conversion than sigmoid
+ learning from positive examples at a lower rate

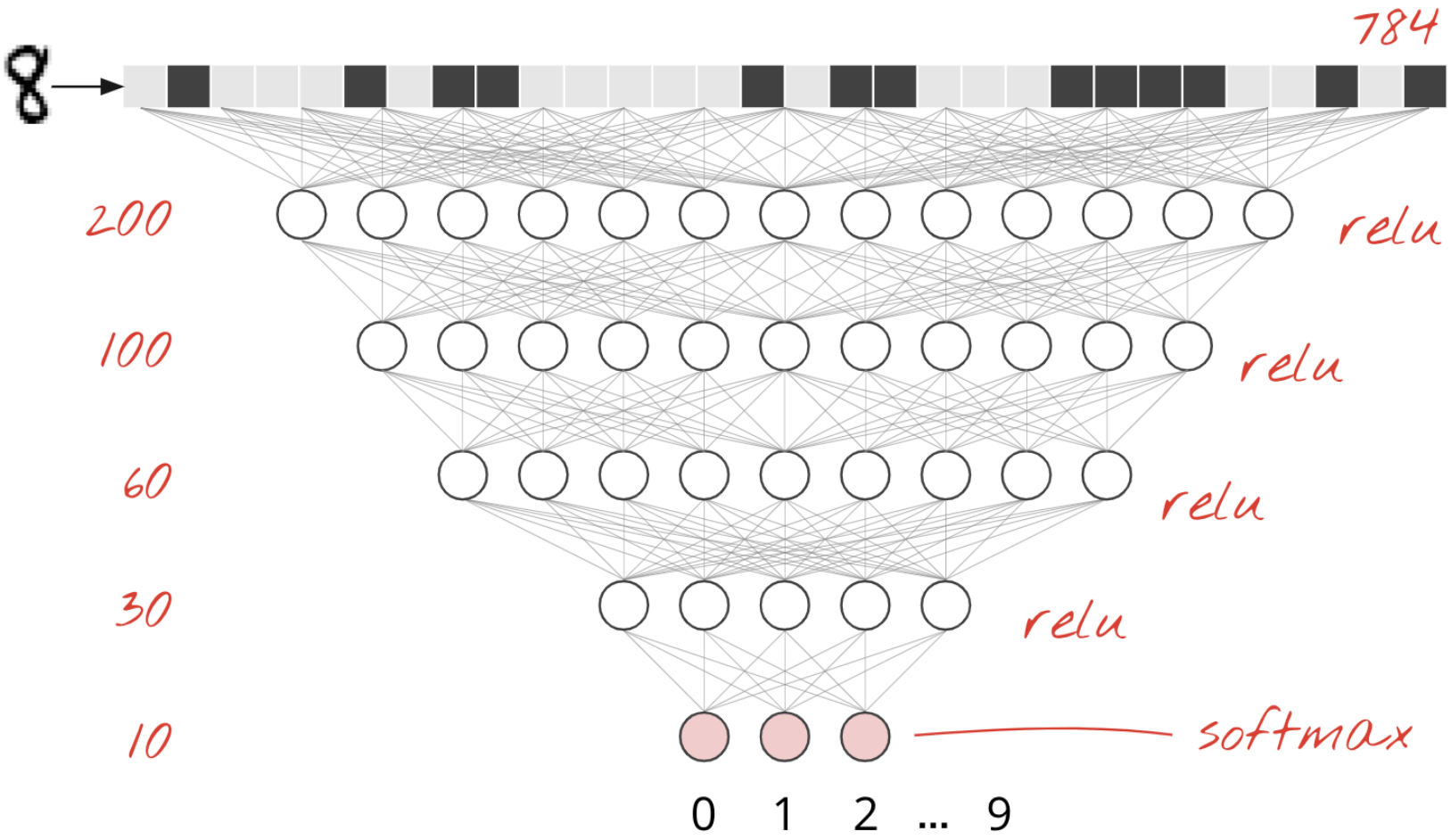
Universality



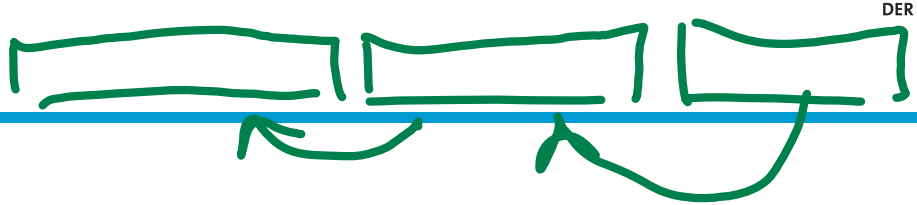
- non-linear data requires non-linear activation functions
- non-linear data requires non-linear feature representation
- you can **approximate any non-linear function** with non-linear activation functions

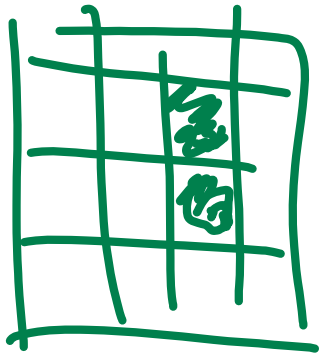
`http://playground.tensorflow.org/`

Deep Neural Network



Let's do it!

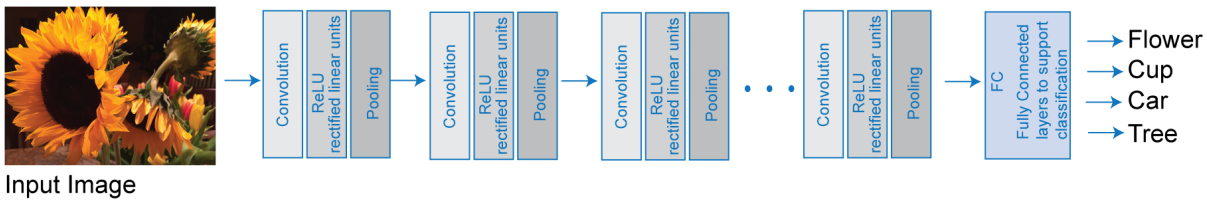


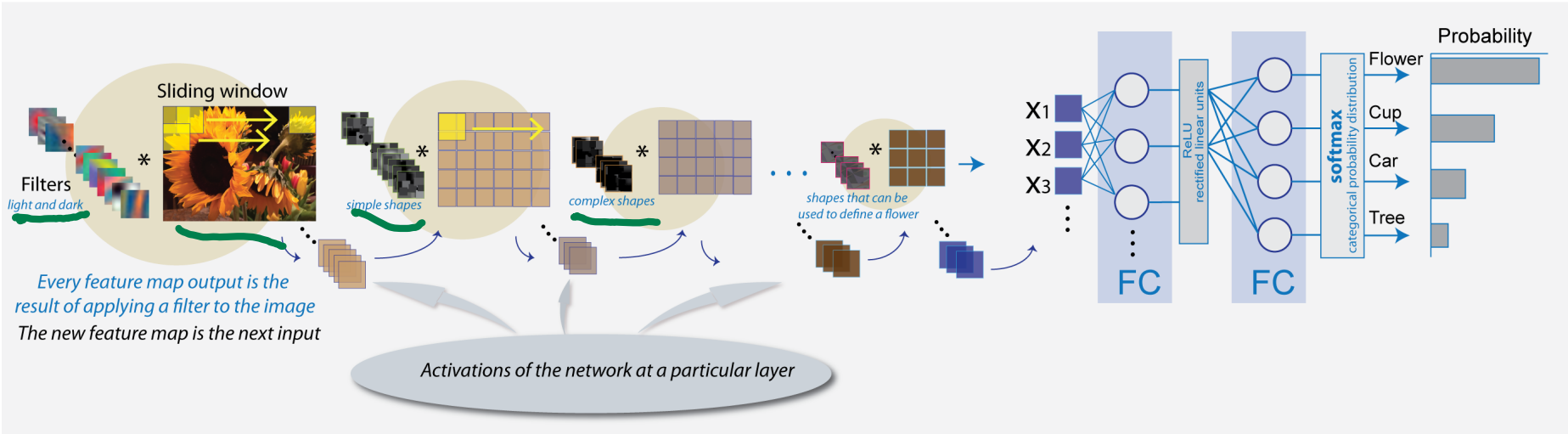


`https://git.informatik.uni-hamburg.de/base.camp/ml-deep-learning/number-recognition-2.py number-recognition-3.py`

I see 1-dimensional people?

Convolutional Neural Network





Layers

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

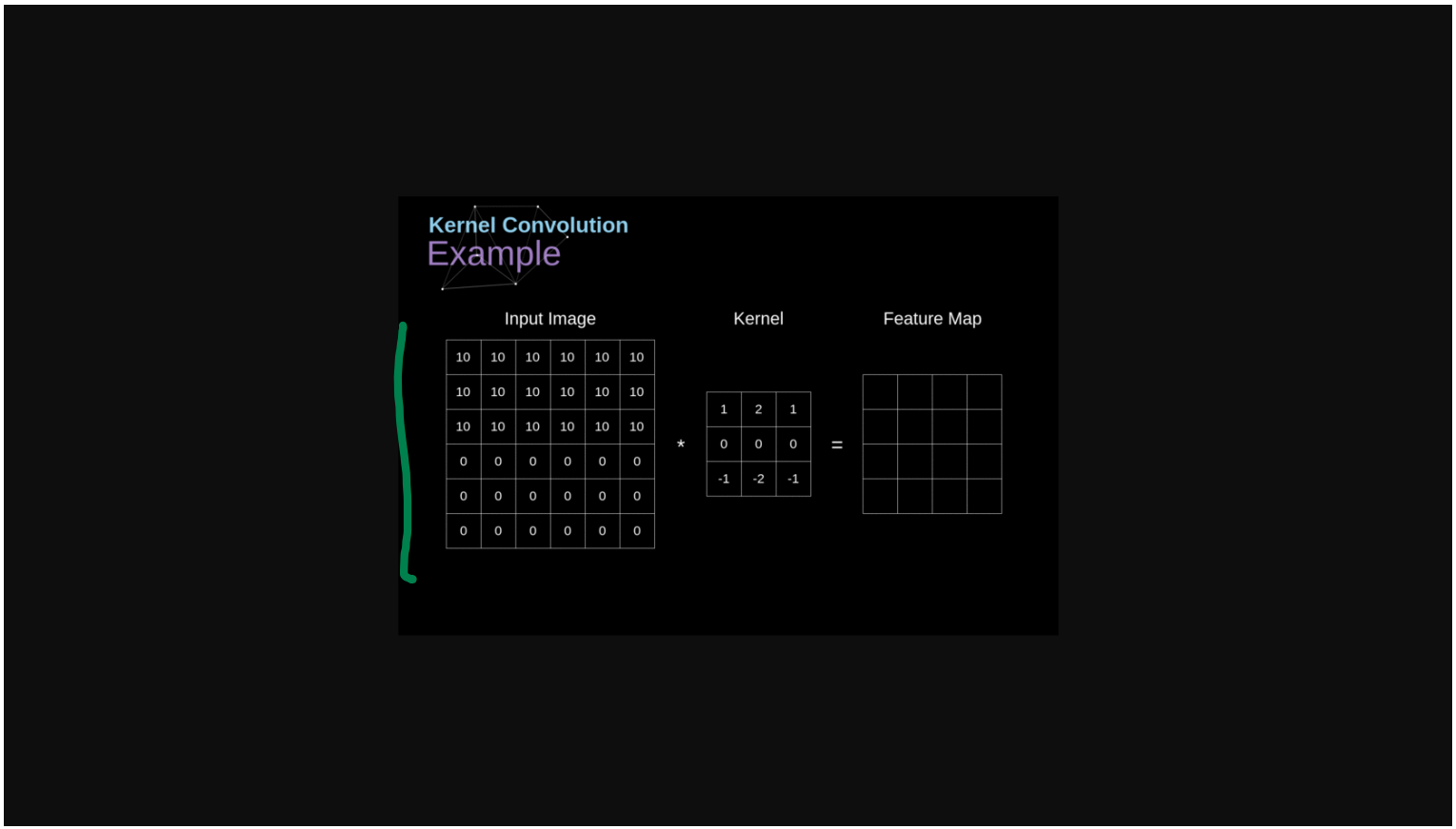
- Dense Layer (fully connected)
- Output Layer (w. number of classes)
- Convolutional Layer
- Pooling Layer

Convolutional Layer

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- automatically learned filters (also: kernels)
- identifying features in tensors
- important for image recognition and signal processing

Convolutional Layer

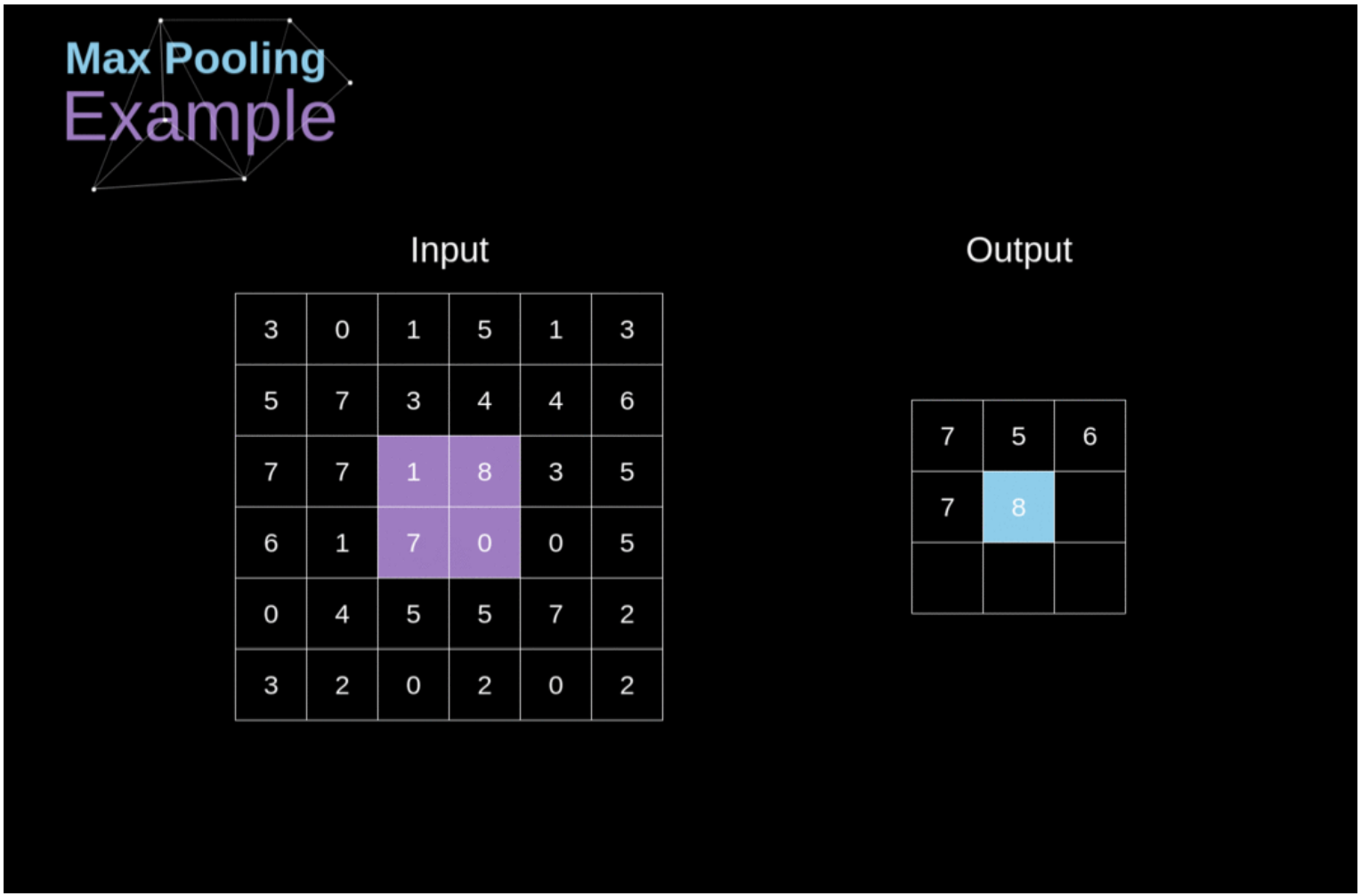


Pooling Layer

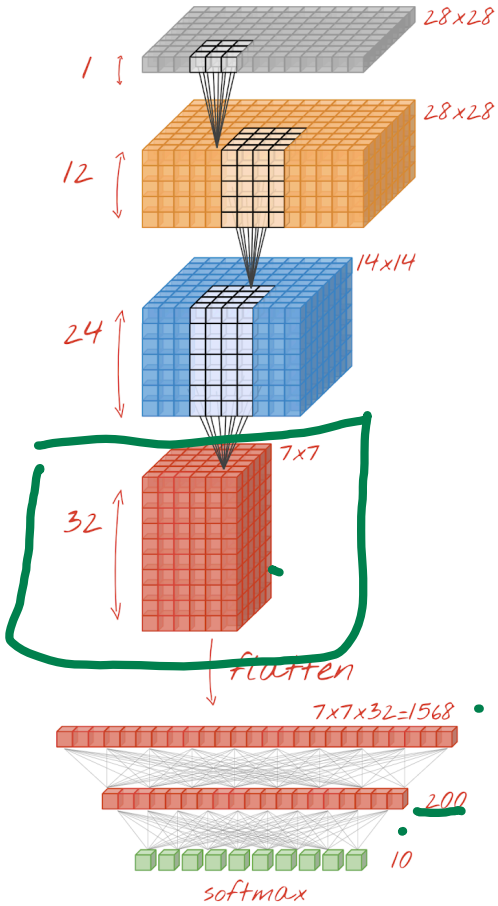
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- max pooling or average pooling
- identifying important points in tensor
- or: blurring the image for further processing (e.g. with convolutions)

Pooling Layer



CNN



Convolutional 3x3 filters=12
$W_1[3, 3, 1, 12]$

Convolutional 6x6 filters=24
$W_2[6, 6, 12, 24]$ stride 2

Convolutional 6x6 filters=32
$W_2[6, 6, 24, 32]$ stride 2

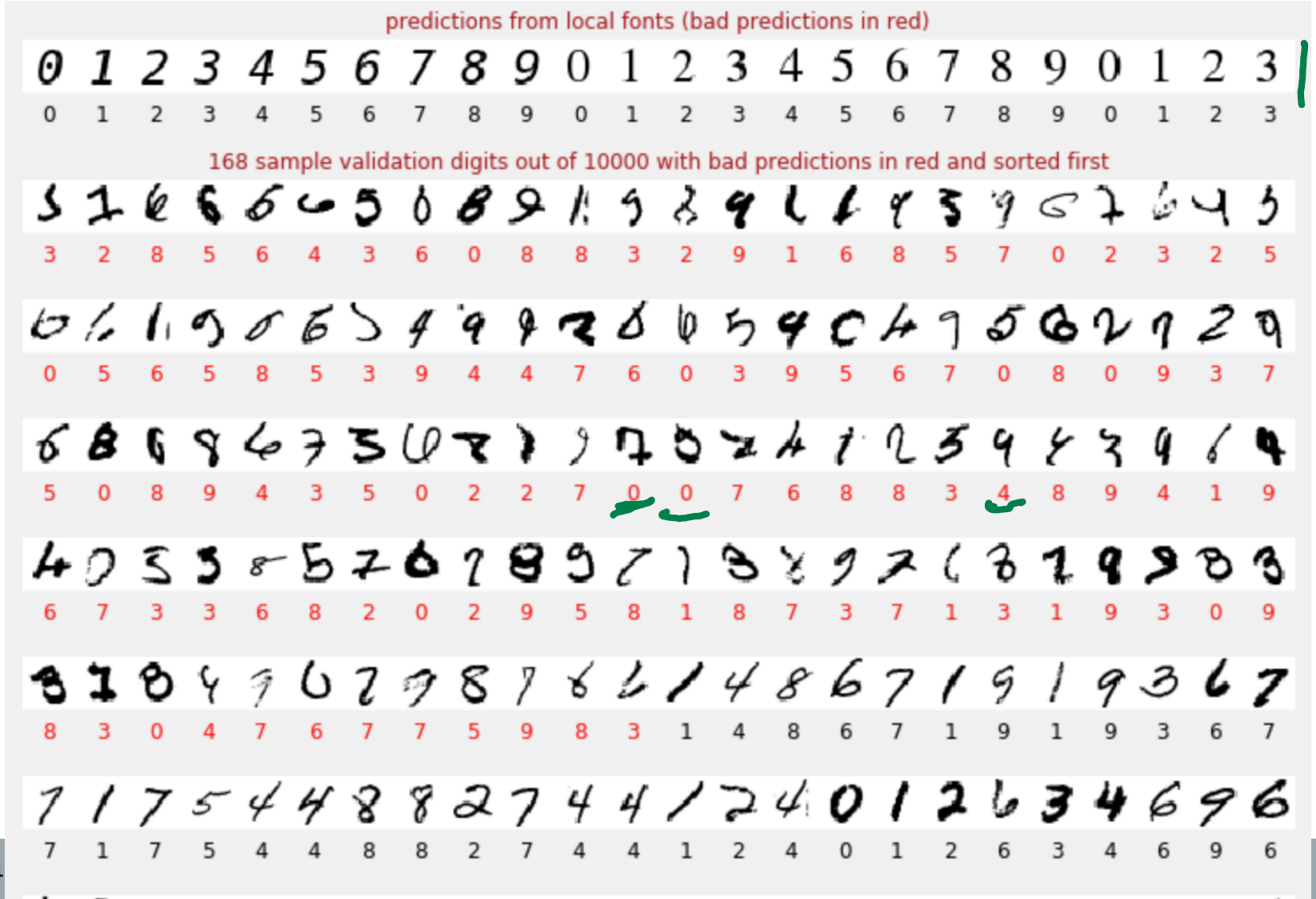
Dense layer
$W_4[1568, 200]$

Softmax dense layer
$W_5[200, 10]$

Let’s do it!

https://git.informatik.uni-hamburg.de/base.camp/ml-deep-learning/number-recognition-4.py

MNIST Results



MNIST Results

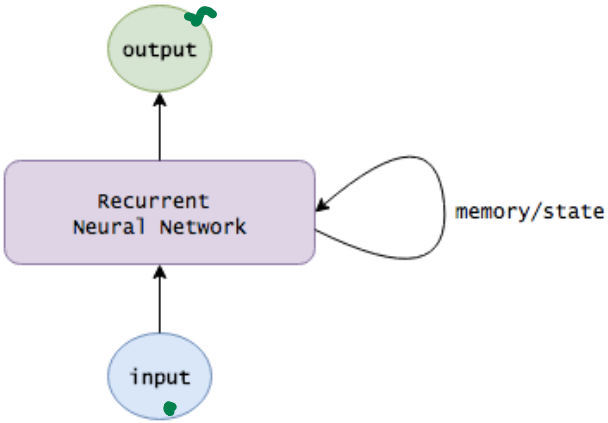
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

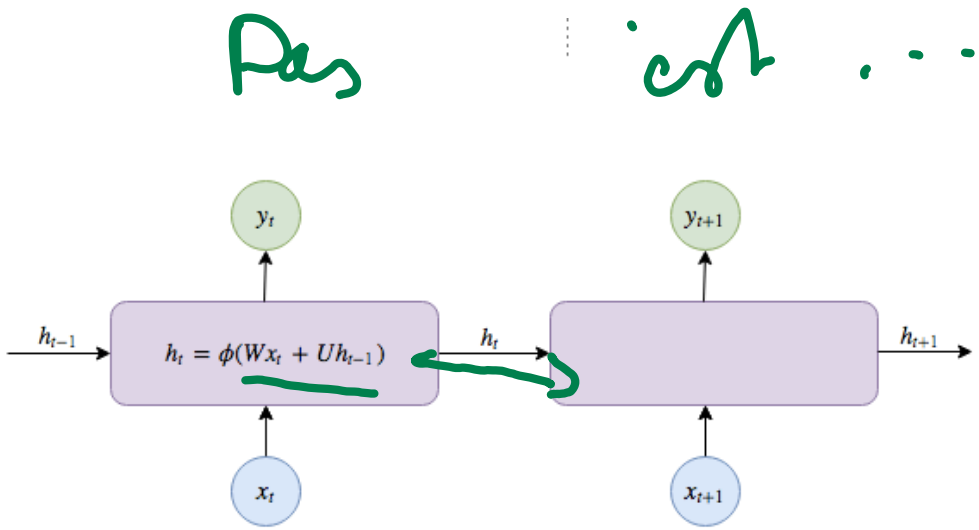
Approach	Error rate
Linear	7.6 – 12.0
KNN	0.5 – 5.0
SVM	0.6 – 1.4
Neural Nets	0.4 – 4.7
Convolutional Nets	0.2 – 1.7
Dense	7.3
Deep Network	1.7 – 2.1
Convolutional Network	1.1

our results

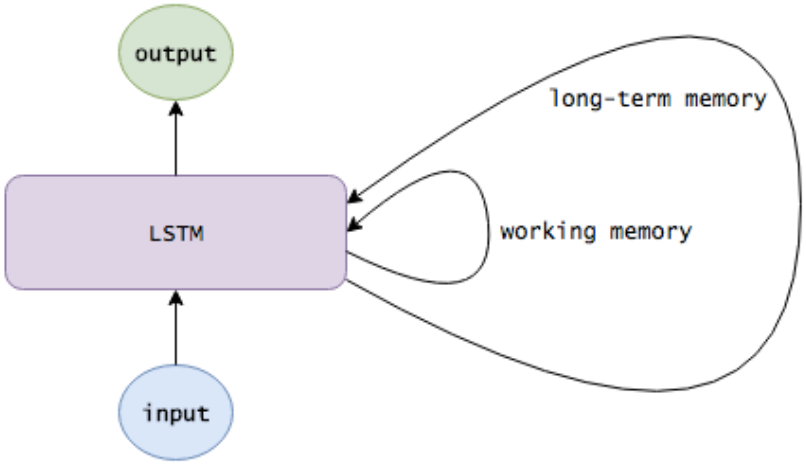
RNN, LSTM and others

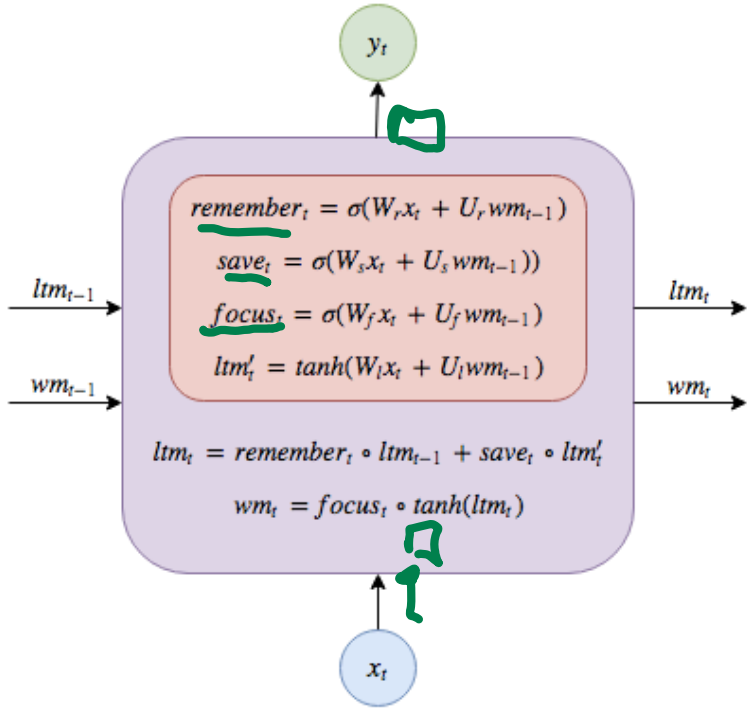
Recurrent Neural Networks





Long Short-Term Memory





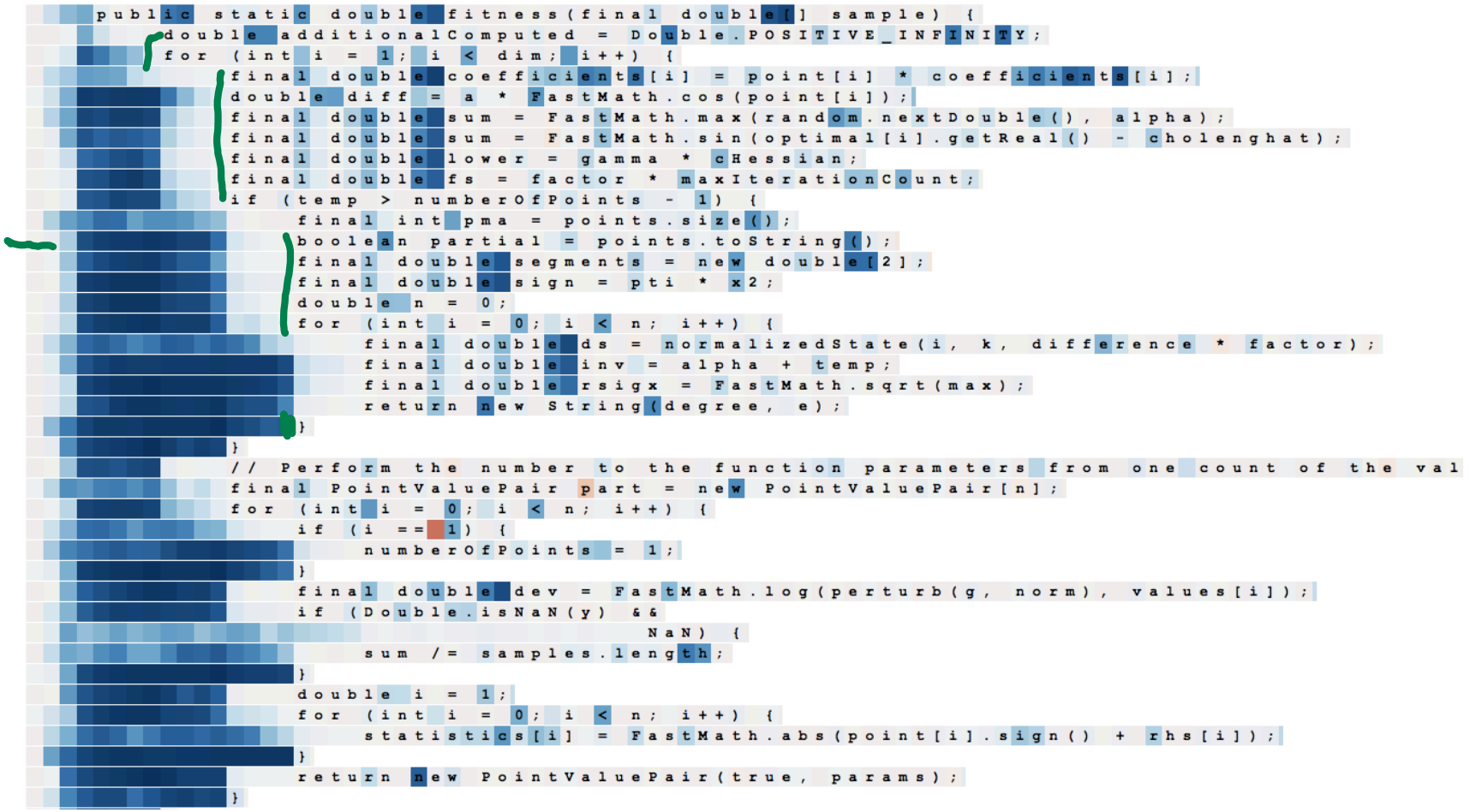
Long Short-Term Memory

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- LSTMs solve RNN problems like vanishing/exploding gradient
- input gate
- keep gate
- forget gate
- long-range dependencies are captured

LSTM to generate Code

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG



```
public static double fitness(final double[] sample) {
    double additionalComputed = Double.POSITIVE_INFINITY;
    for (int i = 1; i < dim; i++) {
        final double coefficients[i] = point[i] * coefficients[i];
        double diff = a * FastMath.cos(point[i]);
        final double sum = FastMath.max(random.nextDouble(), alpha);
        final double sum = FastMath.sin(optimal[i].getReal() - cholenghat);
        final double lower = gamma * cHessian;
        final double fs = factor * maxIterationCount;
        if (temp > numberOfPoints - 1) {
            final int pma = points.size();
            boolean partial = points.toString();
            final double segments = new double[2];
            final double sign = pti * x2;
            double n = 0;
            for (int i = 0; i < n; i++) {
                final double ds = normalizedState(i, k, difference * factor);
                final double inv = alpha + temp;
                final double rsigx = FastMath.sqrt(max);
                return new String(degree, e);
            }
        }
        // Perform the number to the function parameters from one count of the val
        final PointValuePair part = new PointValuePair[n];
        for (int i = 0; i < n; i++) {
            if (i == 1) {
                numberOfPoints = 1;
            }
            final double dev = FastMath.log(perturb(g, norm), values[i]);
            if (Double.isNaN(y) &&
                                    NaN) {
                sum /= samples.length;
            }
            double i = 1;
            for (int i = 0; i < n; i++) {
                statistics[i] = FastMath.abs(point[i].sign() + rhs[i]);
            }
            return new PointValuePair(true, params);
        }
```

Other Interesting Approaches

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Reinforcement Learning, automated learning in a situated environment
- Adversarial Learning, co-training 2 neural nets
- transfer learning
- embeddings (text, speech, graphs, images)

Reinforcement Learning

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- subfield of machine learning
- decision making and motor control
- agents reach goals in complex environments

`https://gym.openai.com/`

Frameworks, Tools and Resources

Tensorflow

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- by Google Brain team
- big ecosystem
- can model any kind of data flow
- mature

`https://www.tensorflow.org/`

Keras

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- by Google research
- modular
- extensible
- user-friendly (building blocks)
- rather an interface for Tensorflow

`https://keras.io/`

Keras


```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=[28, 28, 1]),
    tf.keras.layers.Dense(200, activation="relu"),
    tf.keras.layers.Dense(60, activation="relu"),
    tf.keras.layers.Dense(10, activation='softmax') # classifyi
])

# this configures the training of the model. Keras calls it "c
model.compile(
  optimizer='adam',
  loss= 'categorical_crossentropy',
  metrics=['accuracy']) # % of correct answers

# train the model
model.fit(dataset, ... )
```

PyTorch

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- by Facebook research
- automatic differentiation
- hot topic

`https://pytorch.org/`

PyTorch

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

```
input_size = 784
hidden_sizes = [128, 64]
output_size = 10

# Build a feed-forward network
model = nn.Sequential(nn.Linear(input_size, hidden_sizes[0]),
                      nn.ReLU(),
                      nn.Linear(hidden_sizes[0], hidden_sizes[
                      nn.ReLU(),
                      nn.Linear(hidden_sizes[1], output_size),
                      nn.LogSoftmax(dim=1))

# Train
logps = model(images)
loss = criterion(logps, labels)
loss.backward()
optimizer.step()
```

Ludwig

- by UBER AI LAbs
- automatic encoders for many data types
- no need for programming

`https://uber.github.io/ludwig/`

Ludwig

- MNIST example:
  `https://uber.github.io/ludwig/examples/#image-classification-mnist`

Resources

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Tensorflow Playground
  `http://playground.tensorflow.org/`
- Tensorflow Tutorial
  `https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/`
  Code: `https://github.com/GoogleCloudPlatform/tensorflow-without-a-phd/tree/master/tensorflow-mnist-tutorial`
- *Neural Networks and Deep Learning* book
  `http://neuralnetworksanddeeplearning.com/`
- Github Repository of examples in the slides:
  `https://github.com/basecamp-uhh/mnist-tutorial`

Conclusion

Conclusion

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- neural networks offer powerful ML techniques
- no feature engineering required
- but: network engineering and hyperparameter tuning
- not a means to solve everything but a nice tool in the toolbelt

Conclusion

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Take-away points:

- idea about neural networks
- knowledge to aid in understanding papers and lectures
- motivation to try out neural networks

Outlook

Outlook
Next events

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Feedback: `https://ep.mafiasi.de/p/basecamp-talks`
- Associated: 14.05.2020 - IBM Talk on Quantum Computing
- 28.05.2020: Otto Talk on Chat Bots

`https://www.inf.uni-hamburg.de/inst/basecamp/events/basecamp-talks.html`

See you soon in a base.camp event or your own project!

Fin