DeepL

Subscribe to DeepL Pro to translate larger documents.
Visit www.DeepL.com/pro for more information.

Deep learning

Bilal Al Homsi

Introduction

- Deep learning is a **supervised** learning process.
- Deep learning is able to automatically extract and learn filters from the data.
- The following can be achieved with deep learning:
    - **Classification**
    - **Localization**
    - **Object recognition**
    - **Object segmentation**:
        - **Semantic** segmentation
        - **Instance** segmentation

Classification

- An artificial neural network can be trained to classify new data points.
- There are two types of classification:
  - **Binary** classification
  - **Multi-class** classification
- The training process is as follows:
  - Training data is fed into the model.
  - The model attempts to learn features from the labeled data, to correctly classify unknown data.
  - The model is evaluated and tested with new unknown data.

Localization

- An artificial neural network can be trained to localize objects in an image.
- The coordinate points of a bounding box are determined. This box frames the object being searched for.

Object recognition

- An artificial neural network can be trained to recognize objects in images.
- Both the classification and localization of the objects are determined.
- Both the coordinate points and the class of the object are recognized and determined.

Object segmentation

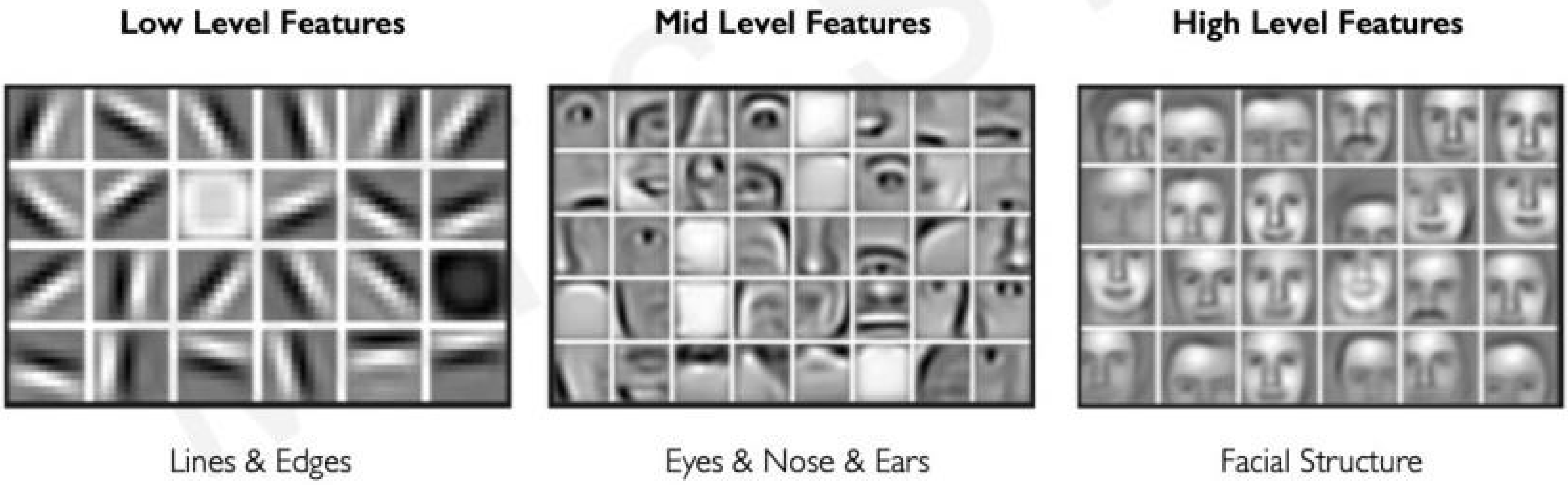
- An artificial neural network can be trained to segment objects in images.
- A distinction is made between **semantic** and instance **segmentation.**
- Semantic segmentation means that all objects of a class are interpreted as a coherent segment, whereas instance segmentation involves treating the individual objects of a class as instances of this class and segmenting them separately accordingly.

Low level features, Mid level features, High level features

- Low level features:
    - Pixel values
    - Color information
    - Simple edges
- Mid level features:
    - Texture features
    - Sample
    - Simple shapes
- High level features:
    - Complex objects
    - Contexts

- In image processing, **features** are *characteristics* that describe objects and their relationships in an image.
- These features are extracted and learned by *Convolutional Layers.*

Example



https://velog.io/@ktm1237/MIT-Introduction-to-Deep-Learning

Deep learning

- The following is required to train an **artificial neural network:**
  - A qualitative **data set**
    - It must be a comprehensive, diverse data set with sufficient variance.
  - A suitable **network architecture**
    - It depends on the task, but the more complex the architecture, the better the training.
  - A **powerful computer**:
    - Training on the **CPU** slow
    - Fast training on the **GPU**

Data set

- In order to achieve good **accuracy** with the neural networks, the data set must cover as many cases of the given problem as possible.
    - **Extensive:**
        - The data set must be relatively large, i.e. the more data it contains, the better the model can learn.
    - **Diverse**
        - The data set must cover as many cases as possible for the problem.
    - **With sufficient variance**
        - Variance means that the same data contains different types.
    - **Balanced distribution of classes in the data set:**
        - The classes have the same distribution, i.e. each class has the same number of data points.

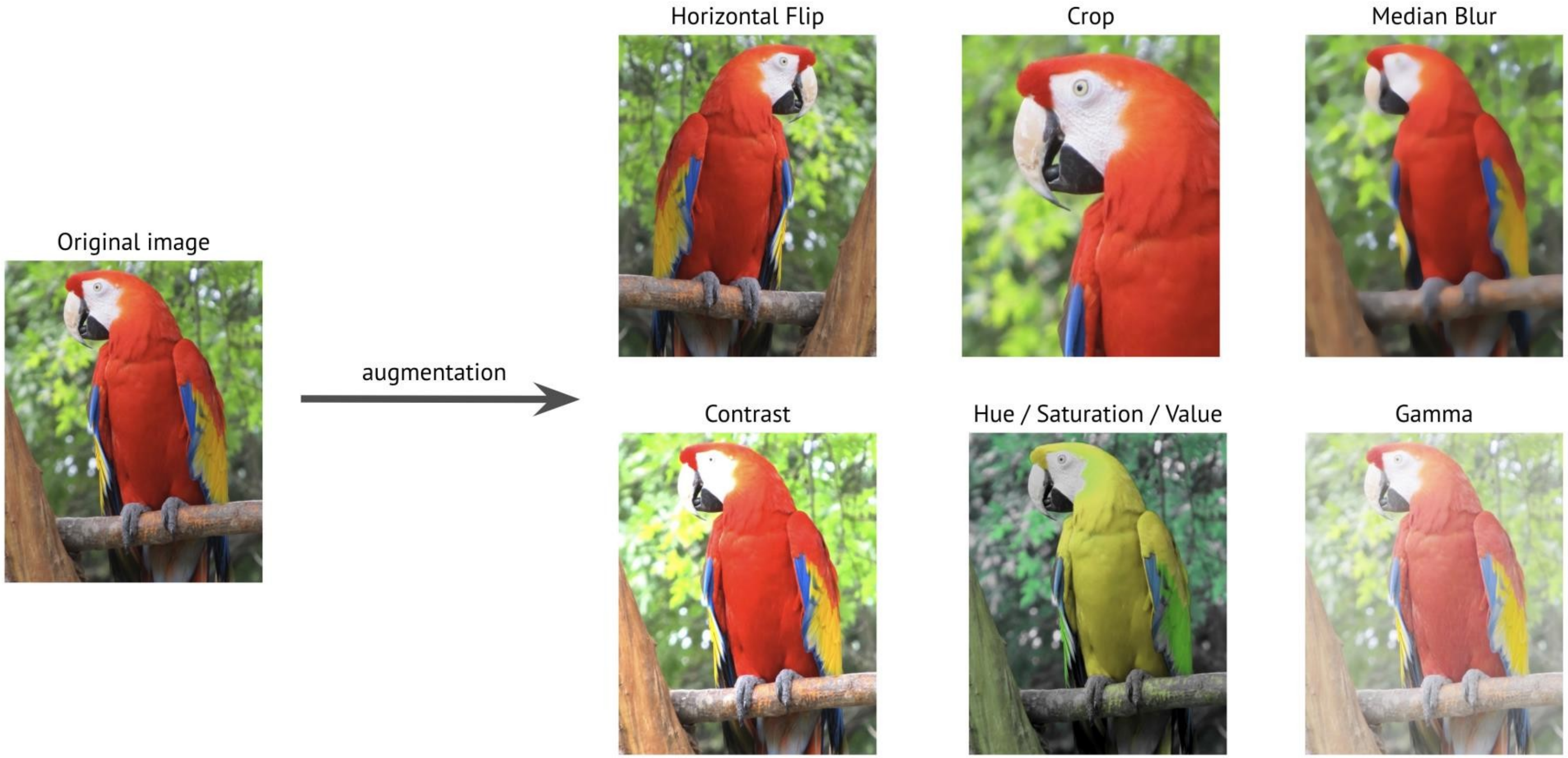
Data set

- Data is one of the most important resources and is needed to train neural networks for specific tasks.
- It is said, '*As your data is, so is your model*'
- And with regard to the question of how data can be obtained?
  - Data strongly depend on the task, they are e.g:
    - Self-collected and pre-processed.
    - Purchased.
    - Downloaded from free internet sources.
    - Ect.

Augmentation

- Is a technique used in neural networks to generate artificial variations of training data.
- Examples are therefore random:
  - Turning
  - Trimming
  - Mirror
  - Zooming (in and out)
  - Changing brightness and contrast
- This technique is used to give the data set more images and variance.

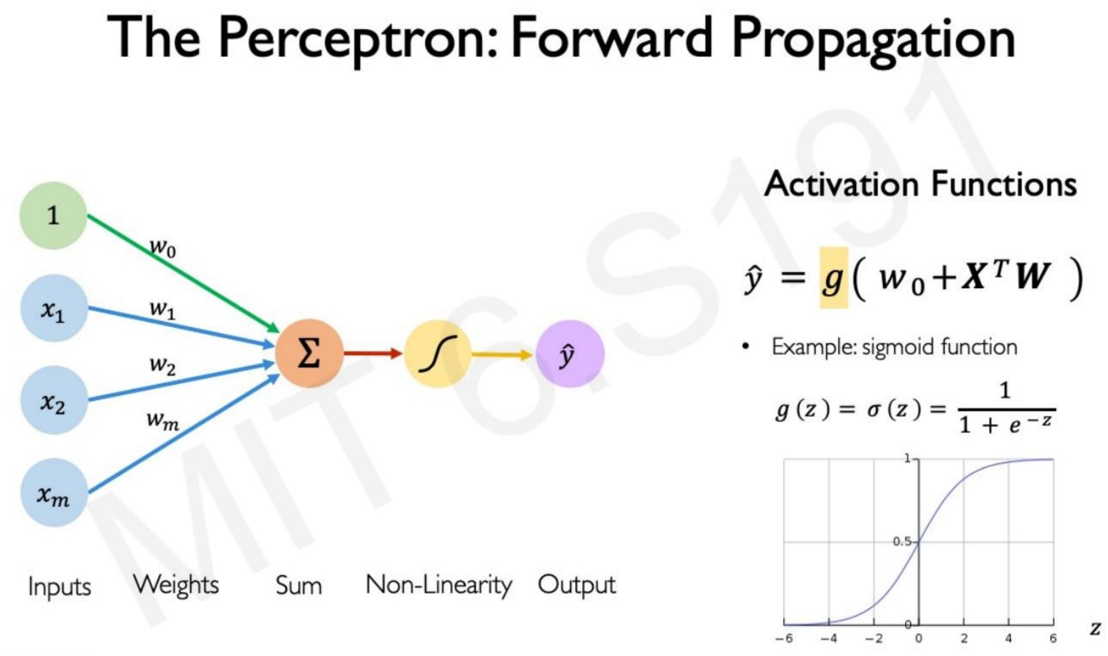
Example of augmented image data



https://albumentations.ai/docs/introduction/image_augmentation/

The perceptron

- The input values are initialized with the random weights. The sum is then calculated.
- 1*w0 +x1*w1 + x2*w2 + x3*w3
- A so-called activation function is then switched on. This allows some neurons to be activated or not.



The Perceptron: Forward Propagation

Activation Functions

$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W} \right)$$

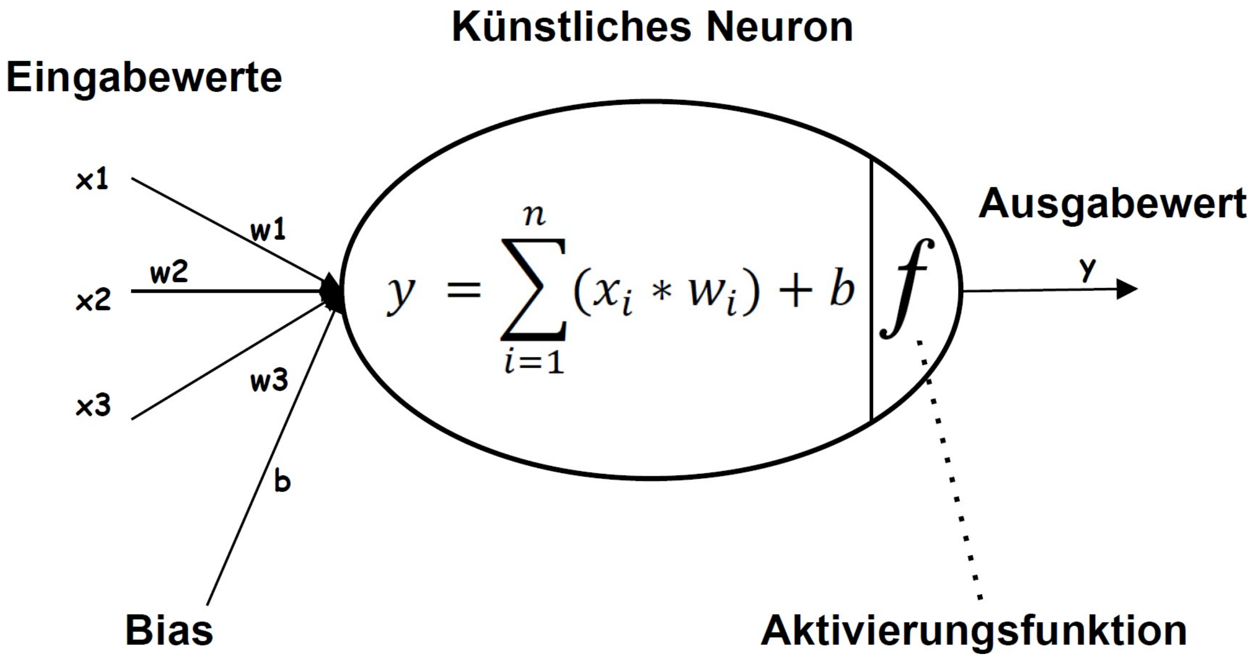
- Example: sigmoid function

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

MIT 6.S191

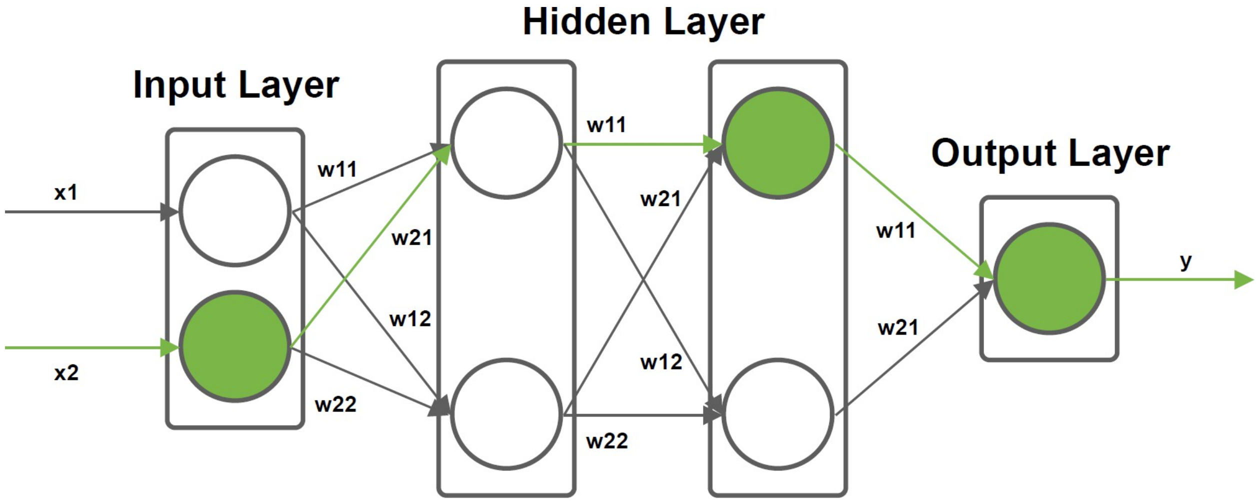
http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf

The perceptron



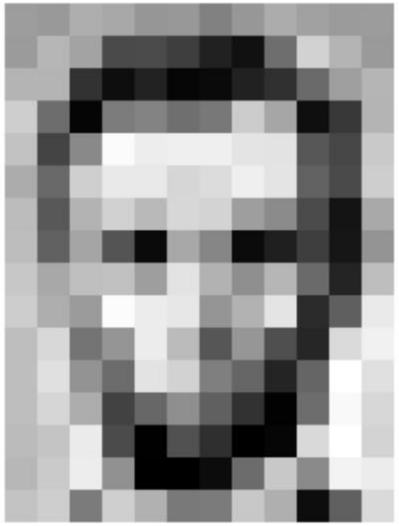
Simple artificial neural network

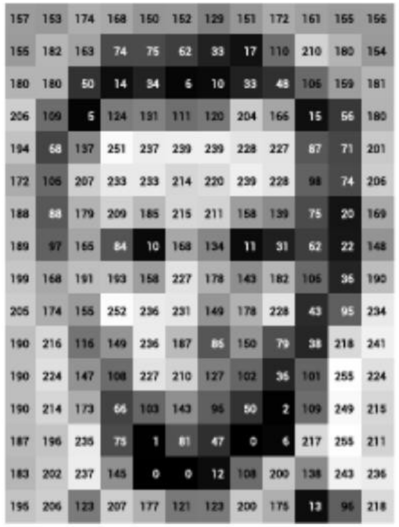
- The following neural network consists of an **input layer**, a **hidden layer** and an **output** *layer*.
- There can be any number of hidden layers. This defines the **depth** of the neural network.
- Each of these layers can contain any number of neurons. This gives the neural network a certain **complexity.**



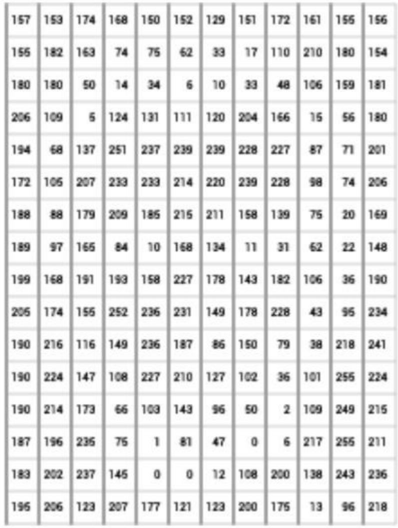
Input layer

- The input layer contains as many neurons as the input values. This means that if, for example, the image size is **28*28** pixels, then the input layer must have **784 neurons**, with each neuron assuming a pixel value.





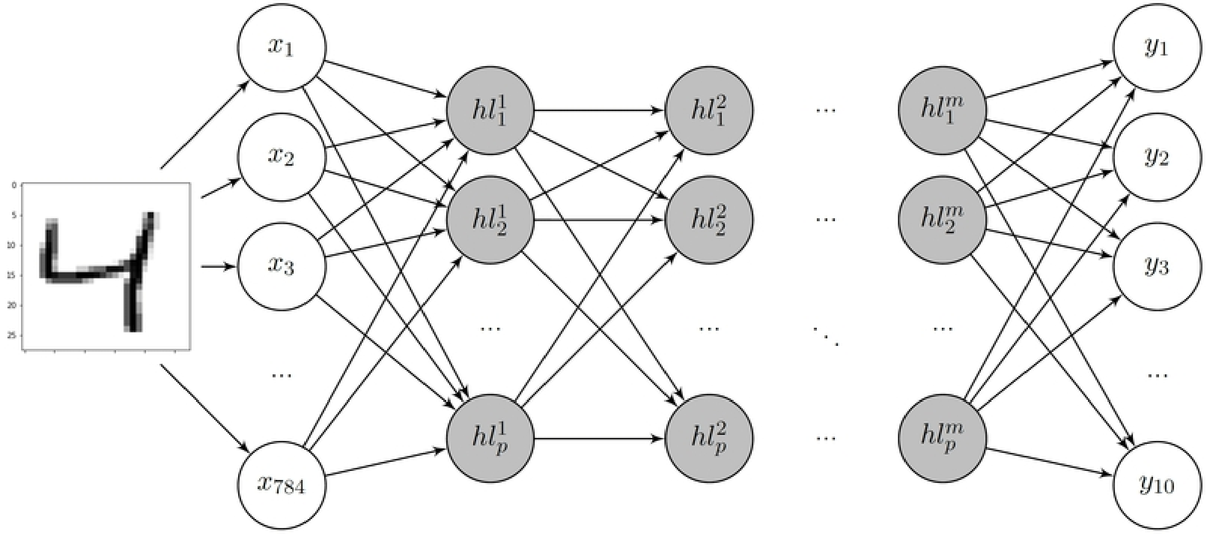
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Output layer

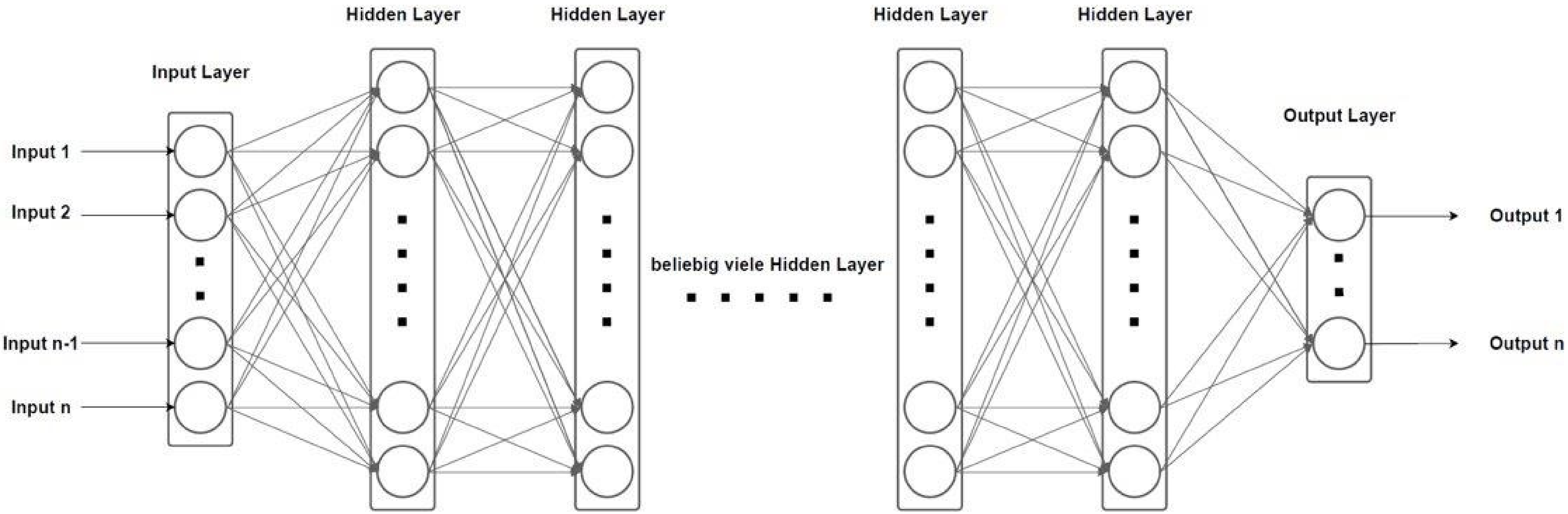
- The output layer contains as many neurons as the number of target classes. This means that if there are **10 classes** with the aim of classifying the new image into one of these classes, then the output layer must have **10 neurons** so that these 10 classes can be classified into one of these classes.
  can be represented.



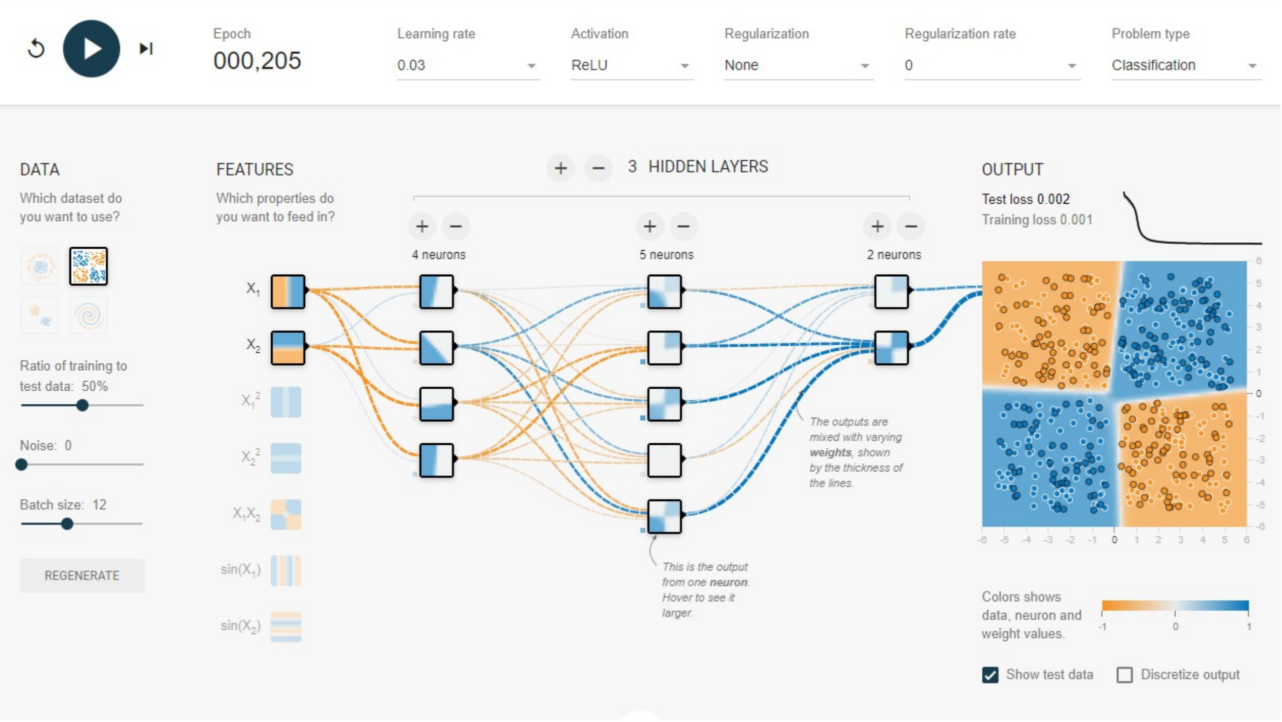
https://www.researchgate.net/figure/Artificial -neural-network-architecture-

for-the-MNIST-dataset-classification-The-input_fig4_349991068

A deep neural network



Playground Tensorflow



Epoch 000,205

Learning rate 0.03

Activation ReLU

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 12

FEATURES

Which properties do you want to feed in?

3 HIDDEN LAYERS

4 neurons

5 neurons

2 neurons

OUTPUT

Test loss 0.002

Training loss 0.001

The outputs are mixed with varying ***weights****, shown by the thickness of the lines.*

This is the output from one ***neuron****. Hover to see it larger.*

Colors shows data, neuron and weight values.

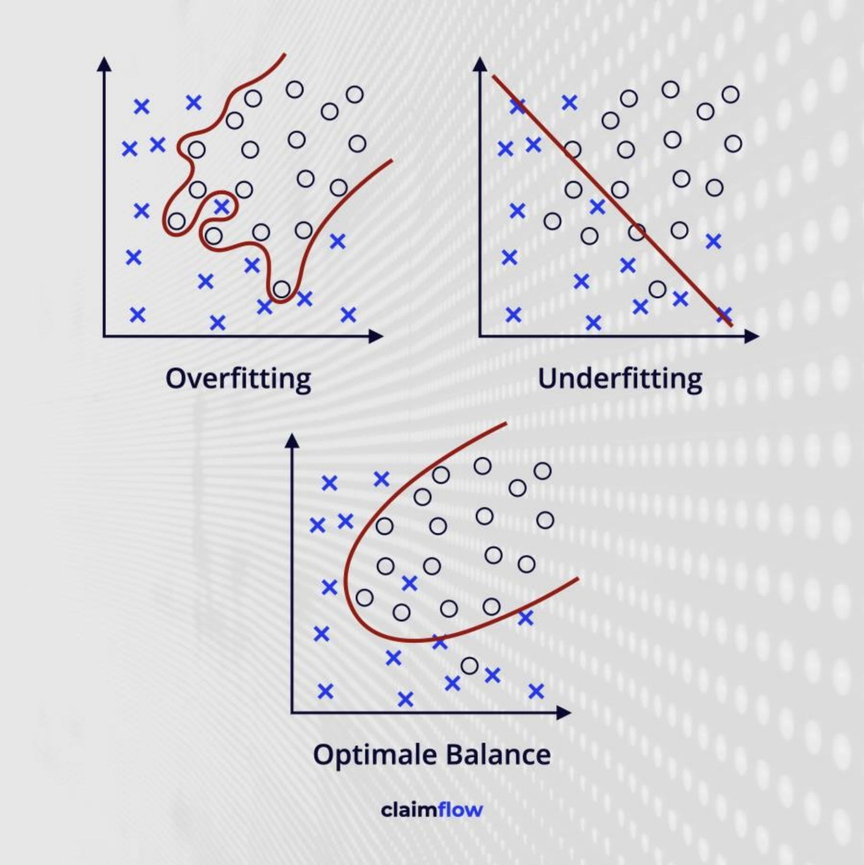
☑ Show test data ☐ Discretize output

https://playground.tensorflow.org/

Overfitting, underfitting and Optinal Balace

- **Overfitting:**

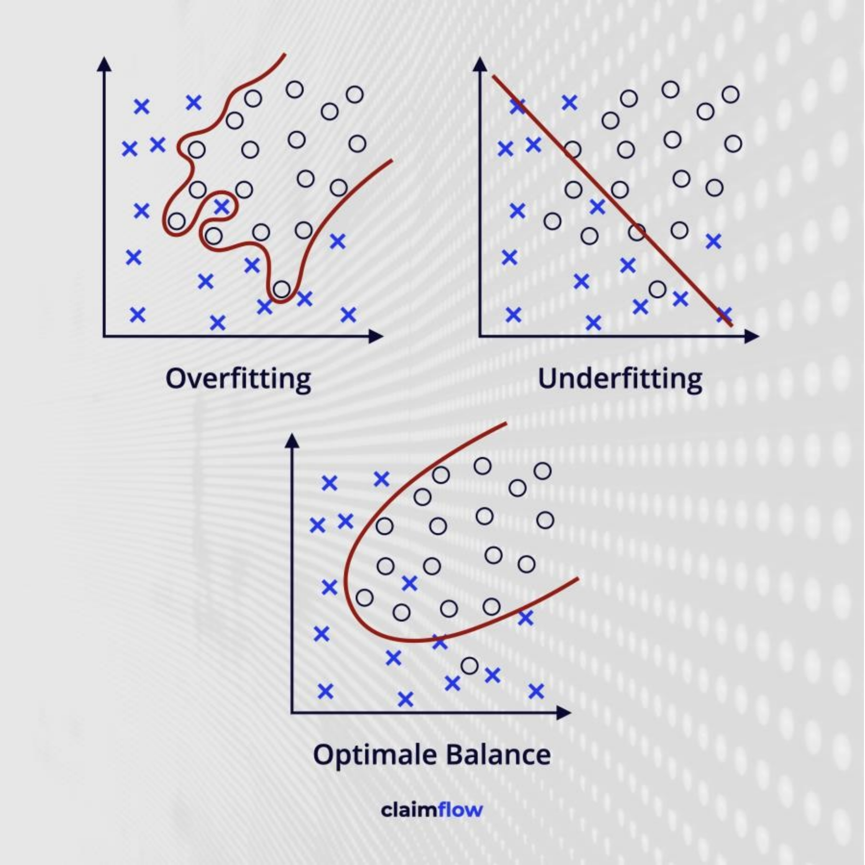
The issue is that the neural network has adapted too much to the data. The model achieves very good results on training data, but not on the test data. The reason is that the model has memorized the features of the data. It is therefore unable to generalize.



Overfitting, underfitting and Optinal Balace

- **Underfitting:**

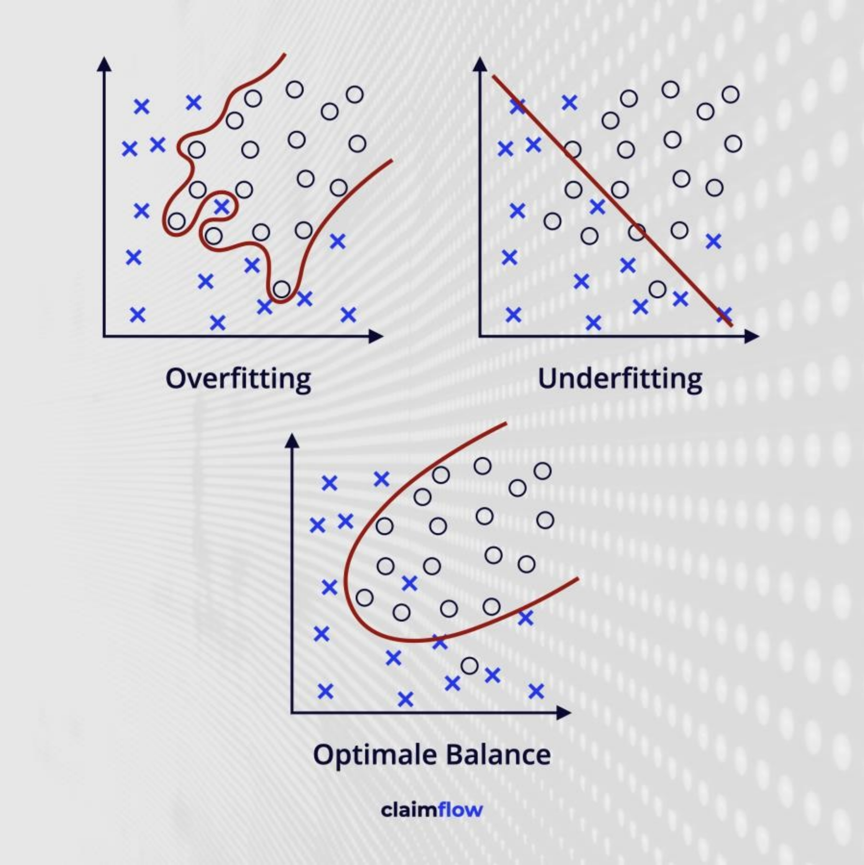
The issue is that the neural network has not learned any features from the data. The model achieves very poor results on training data, but also on the test data. The reason is that the model has not understood and learned the features of the data. It is therefore unable to generalize.



Overfitting, underfitting and Optinal Balace

- **Optimal balance:**

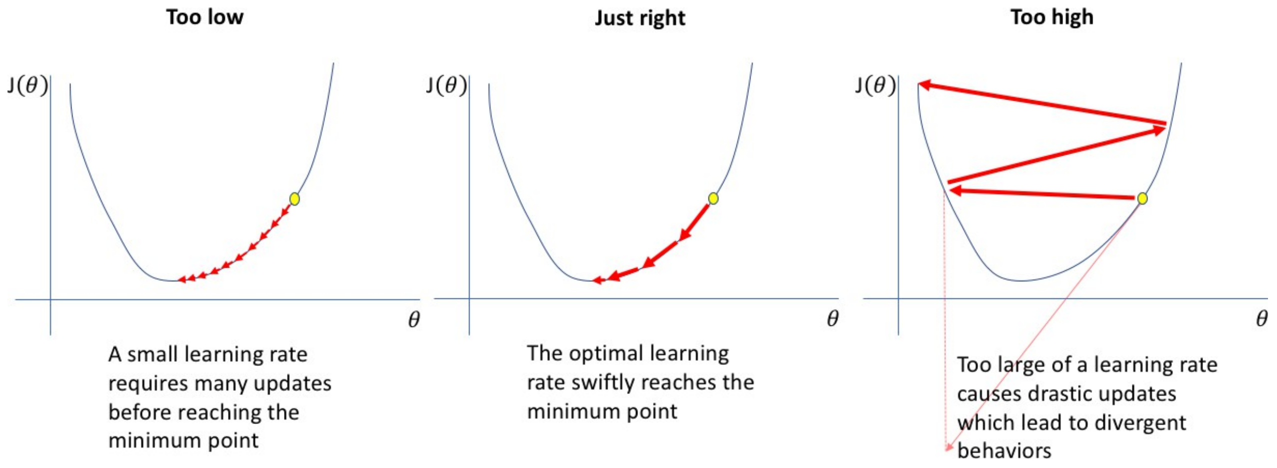
The point is that the neural network
has learned enough features from the data. The model achieves very good results on training data, but also on the test data. The reason for this is that the model has understood and learned the features of the data. It can therefore
generalize well.



https://www.claimflow.de/post/overfitting-und-underfitting

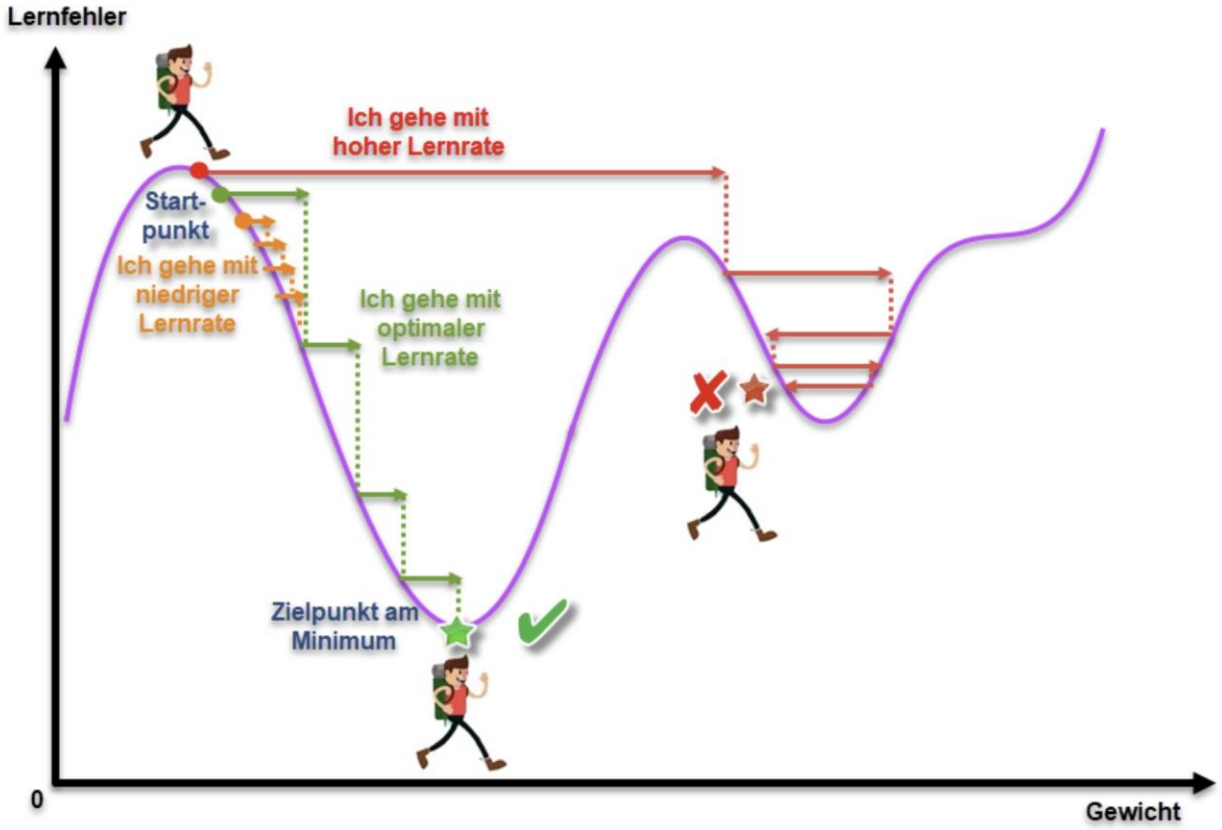
Learning rate

- The learning rate is one of the most important hyperparameters used in the Optimization used become.
- The learning rate indicates how large the steps are while the model is learning the Weightings updated.

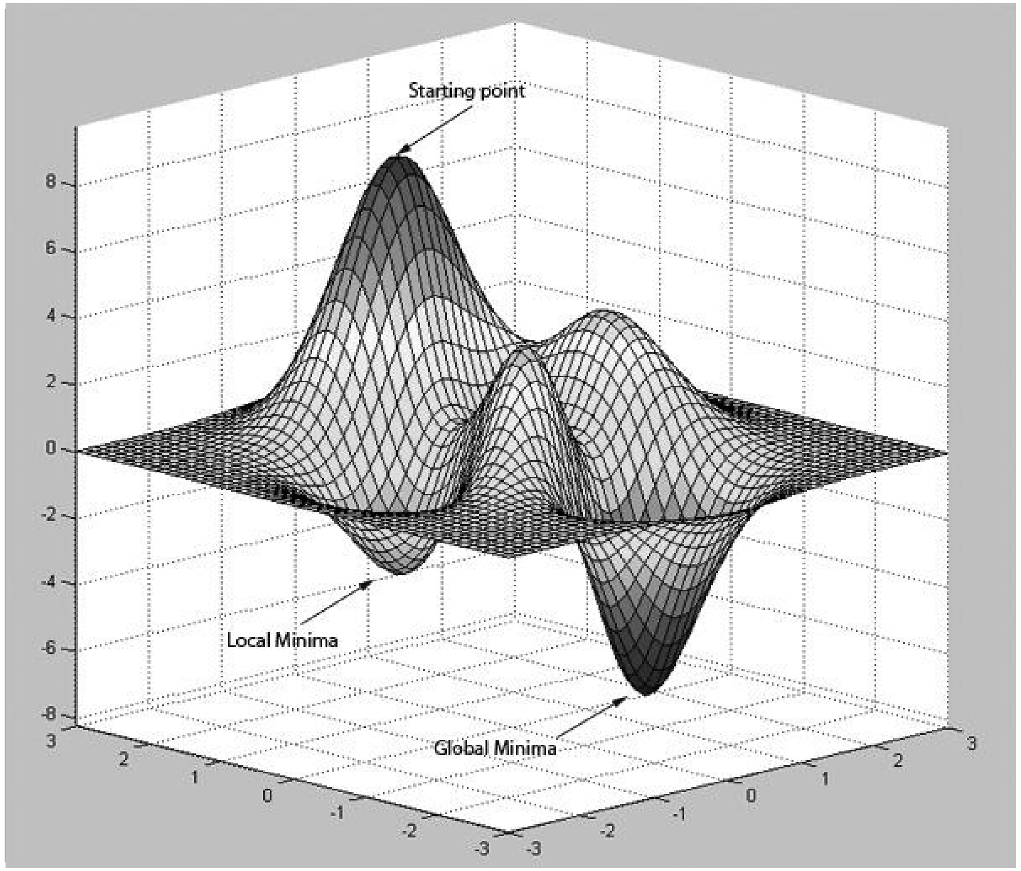

Too low
Just right
Too high
J(θ)
J(θ)
J(θ)
θ
θ
θ
A small learning rate requires many updates before reaching the minimum point
The optimal learning rate swiftly reaches the minimum point
Too large of a learning rate causes drastic updates which lead to divergent behaviors

https://www.jeremyjordan.me/nn-learning-rate/

Example of local and global maxima and minima


Lernfehler
Ich gehe mit hoher Lernrate
Start-punkt
Ich gehe mit niedriger Lernrate
Ich gehe mit optimaler Lernrate
Zielpunkt am Minimum
0
Gewicht

https://www.ngw.ch/wp-content/uploads/2019/01/Kinderuniversit%C3%A4t_Reichenbacher_09012019.pdf



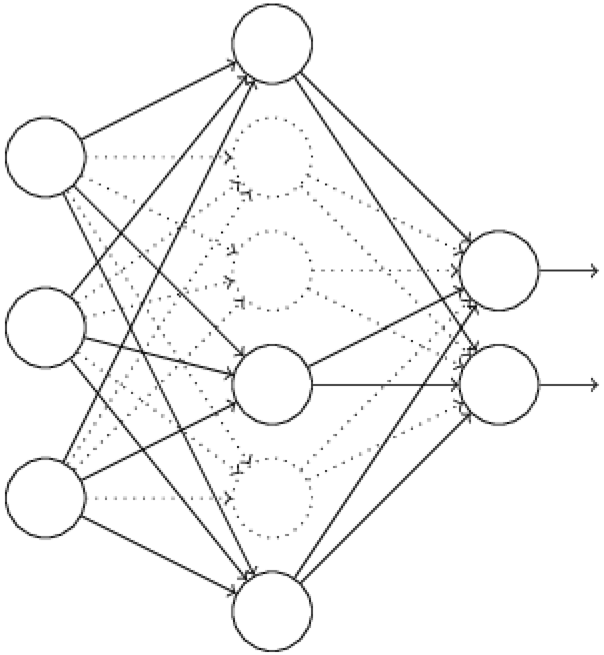
https://www.codeplanet.eu/tutorials/csharp/70-kuenstliche-neuronale-networks-in-csharp.html

Regularization

- The aim is to improve the generalization of the model to invisible data.
- There are two options:
  - **Dropout:**
    - Some connections to neurons are deliberately eliminated.
  - **Early Stopping:**
    - An end is set at a certain point so that **overfitting** and **underfitting can be avoided.** can be avoided.

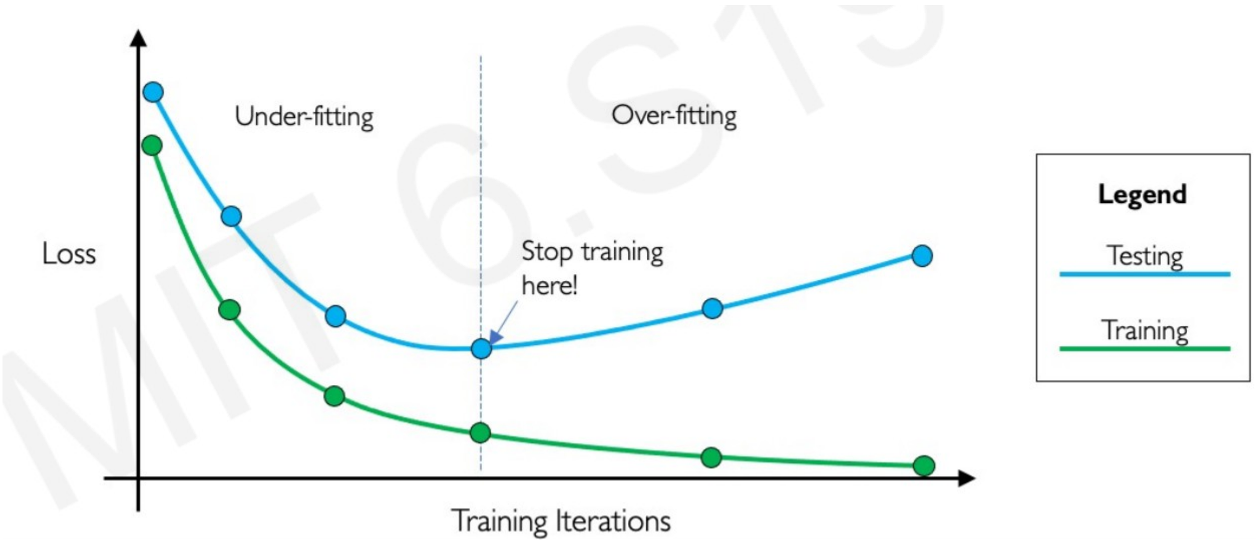
Regularization

Dropout



https://kharshit.github.io/blog/2018/05/04/dropout-prevent-overfitting

Early Stopping

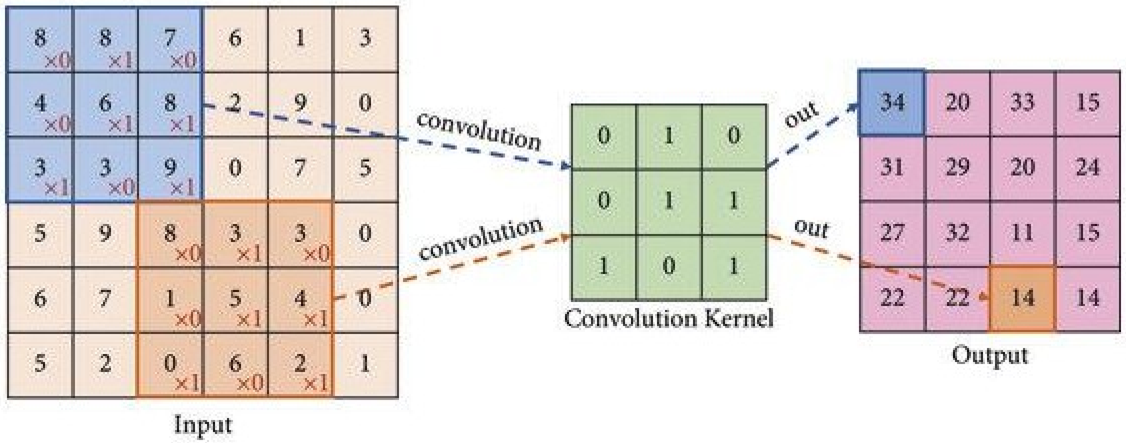


MIT 6.S19

http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf

Convolutional Operator

- This is a special type of neural network that can be used for *image data.* A **filter (kernel) is** used on the image.
- This is moved step by step from left to right, multiplying the input values by the kernel values and adding them up. The result of this operation is entered step by step in a **feature map.**



8 ×0	8 ×1	7 ×0	6	1	3
4 ×0	6 ×1	8 ×1	2	9	0
3 ×1	3 ×0	9 ×1	0	7	5
5	9	8 ×0	3 ×1	3 ×0	0
6	7	1 ×0	5 ×1	4 ×1	0
5	2	0 ×1	6 ×0	2 ×1	1

Input

0	1	0
0	1	1
1	0	1

Convolution Kernel

34	20	33	15
31	29	20	24
27	32	11	15
22	22	14	14

Output

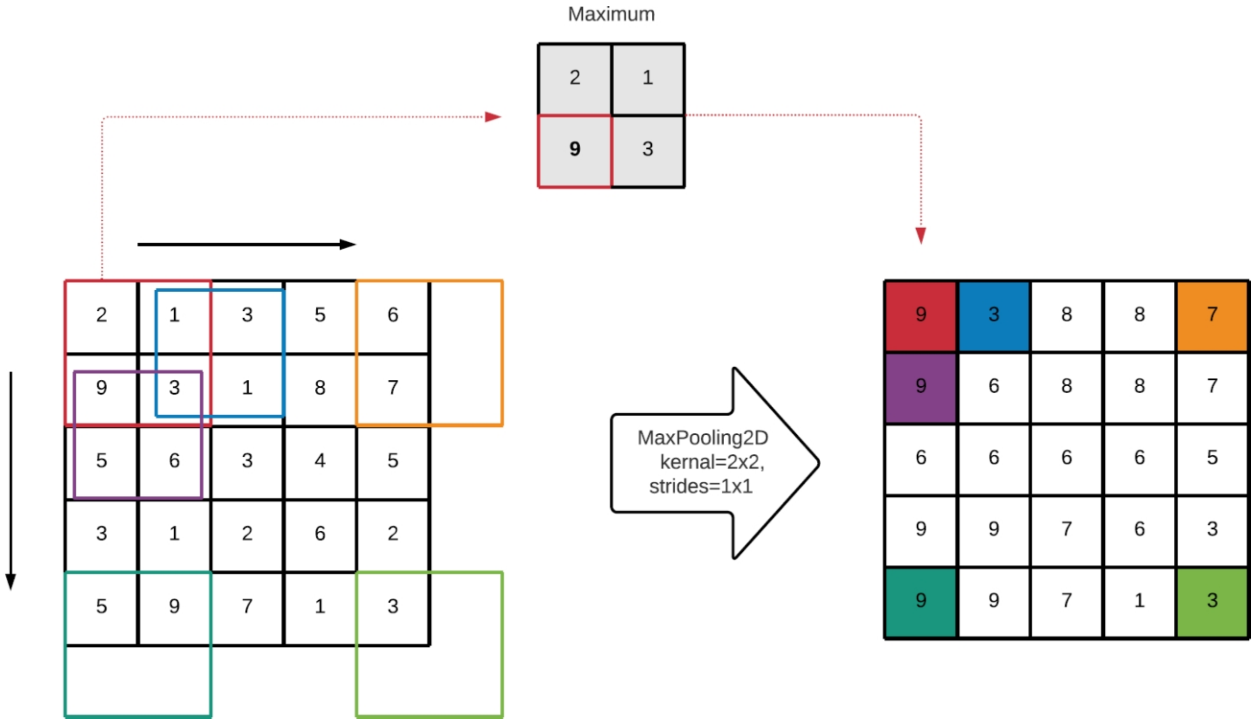
https://www.researchgate.net/figure/Convolution-with-convolution-kernel-of-size-33-stride-of-1-and-no-zero-padding_fig2_356590429

Convolutional layer

- The following can be determined in the Convolutional layer:
  - The number of filters
  - The size of the kernel
  - The stride size (*stride*)
- The following can be achieved with the Convolutional layer:
  - Extraction of features and useful characteristics
  - Reduction in the size of the image data (more memory and less computing power)

Pooling Operator

- The pooling operation is used directly after the convolutional operation.
- The main idea is that reduce the dimension of the *feature map* while retaining the relevant information.



Maximum

2	1
9	3

2	1	3	5	6
9	3	1	8	7
5	6	3	4	5
3	1	2	6	2
5	9	7	1	3

MaxPooling2D
kernal=2x2,
strides=1x1

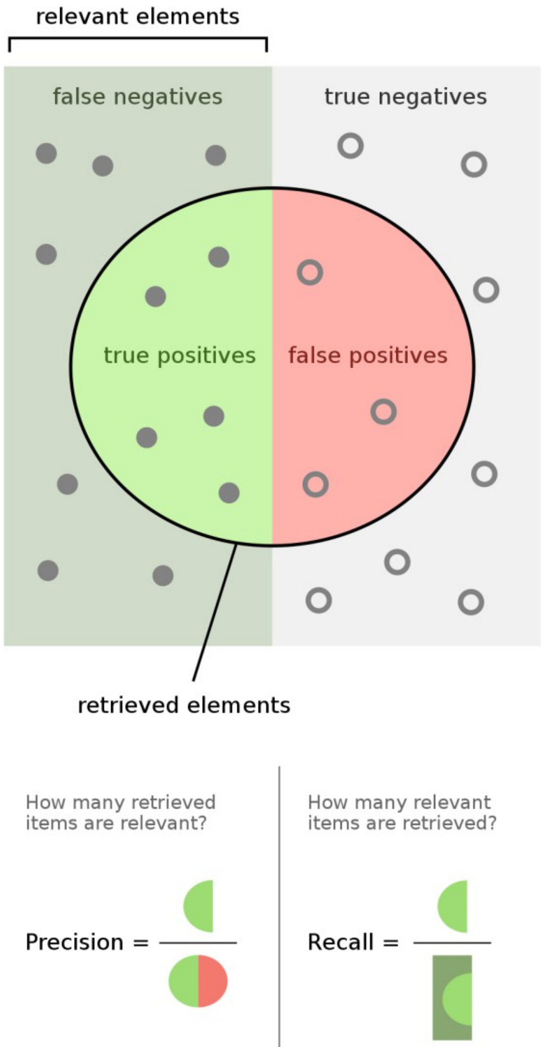
9	3	8	8	7
9	6	8	8	7
6	6	6	6	5
9	9	7	6	3
9	9	7	1	3

Pooling layer

- The following can be determined in the pooling layer:
  - The size of the kernel
  - The stride size (*stride*)
  - Bounding layer (*padding*)
- The padding layer helps to ensure that the dimension of the feature map is maintained. This means that the size of the input and output is the same.
- The following can be achieved with the pooling layer:
  - **MaxPooling2D**
    - The maximum number in the pooling kernel is selected.
  - **AveragePooling2D**
    - The average number in the pooling kernel is selected.

Recall and precision

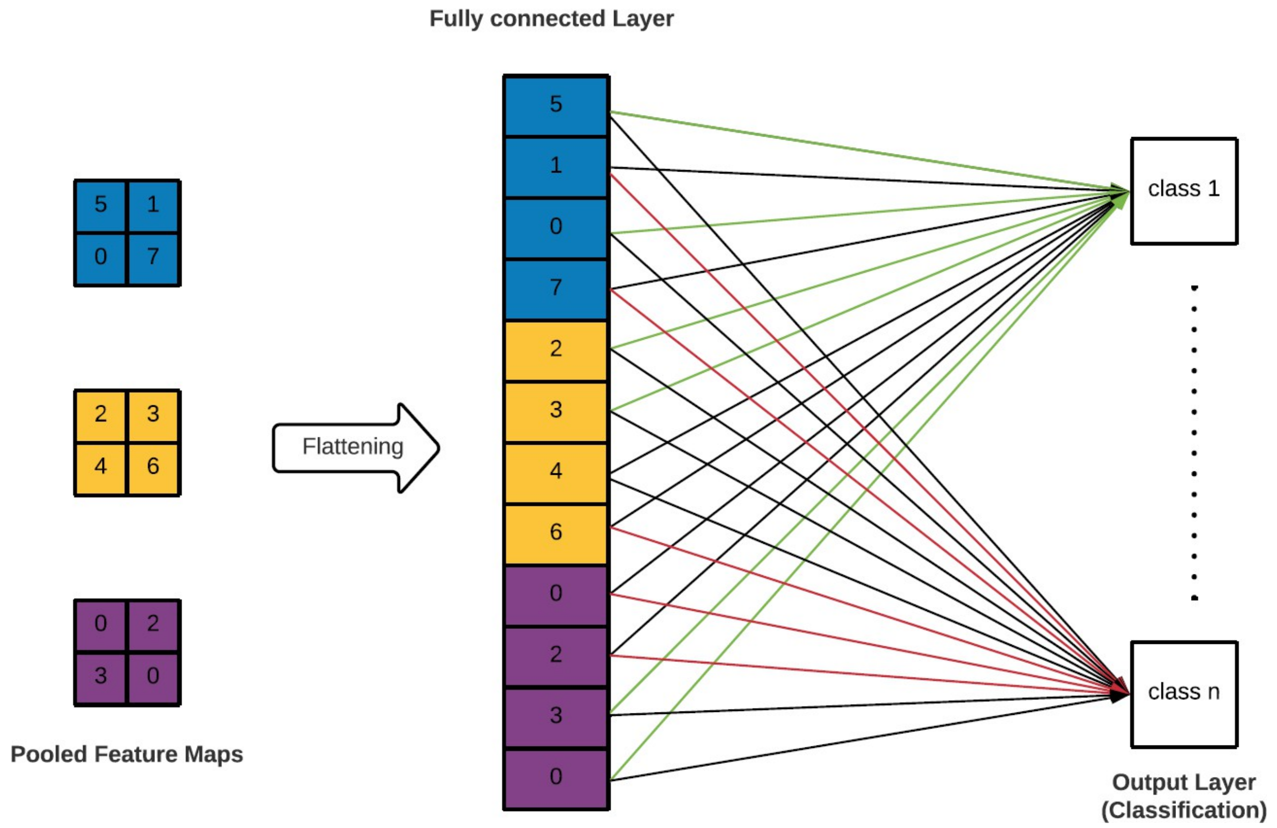
- These are metrics for evaluating the neural networks.
- **Recall**: How
  Many relevant data points were found.
- **Precision**: How
  Many data points found are relevant or correct.



https://en.wikipedia.org/wiki/Precision_and_recall

Flattening

- *Feature maps* from the last *convolutional layer* are smoothed so that they are fed into the fully connected neural network.
- Each of these values is the input for the input neurons.



5	1
0	7

2	3
4	6

0	2
3	0

Project

- A project is planned in this module. This is credited as part of the examination.
- The project requires:
  - A data record for at least two classes.
    - If necessary, the image data should be pre-processed.
  - A suitable architecture for a convolutional neural network.
    - It is necessary to try out different constellations and combinations of hyperparameters.
  - Each result of the accuracy of the model should be compared with of the respective combination of parameters.
  - A good result is expected for the given classification problem.

Google Colab

- It's like a **Jupyter notebook** that can be run with cloud resources.
- The code can be executed on **CPU, GPU or TPU.**
- This can accelerate the computing power.
- The code can be connected to **Google Drive to** save *RAM* and *storage capacity.* The data can therefore be retrieved directly via the cloud.
- For more information see: https://colab.google/