Deep Learning

Bilal Al Homsi

Einführung

- Deep Learning ist ein **überwachtes** Lernverfahren.
- Deep Learning ist in der Lage, Filter aus den Daten automatisch zu extrahieren und zu erlernen.
- Mit Deep Learning kann man folgendes erreichen:
  - **Klassifizierung**
  - **Lokalisierung**
  - **Objekterkennung**
  - **Objektsegmentierung**:
    - **Semantische** Segmentierung
    - **Instanz**-Segmentierung

Klassifizierung

- Ein künstliches neuronales Netz kann darauf trainiert werden, um neue Datenpunkte zu klassifizieren.
- Es gibt zwei Arten von Klassifizierung:
  - **Binäre** Klassifizierung
  - **Mehrklassen** Klassifizierung
- Der Trainingsprozess läuft wie folgt ab:
  - Es werden Trainingsdaten in das Modell eingespeist.
  - Das Modell versucht, aus den gelabelten Daten Features zu erlernen, um unbekannte Daten richtig zu klassifizieren.
  - Das Modell wird mit neuen unbekannten Daten evaluiert und getestet.

Lokalisierung

- Ein künstliches neuronales Netz kann darauf trainiert werden, um Objekte in einem Bild zu lokalisieren.
- Dabei werden die Koordinatenpunkte eines Begrenzungsrahmens (Bounding Box) ermittelt. Diese Box umrahmt das gesuchte Objekt.

Objekterkennung

- Ein künstliches neuronales Netz kann darauf trainiert werden, um Objekte in Bildern zu erkennen.
- Es werden dafür sowohl Klassifizierung als auch Lokalisierung der Objekte ermittelt.
- Es werden sowohl die Koordinatenpunkte als auch die Klasse des Objekts erkannt und ermittelt.

Objektsegmentierung

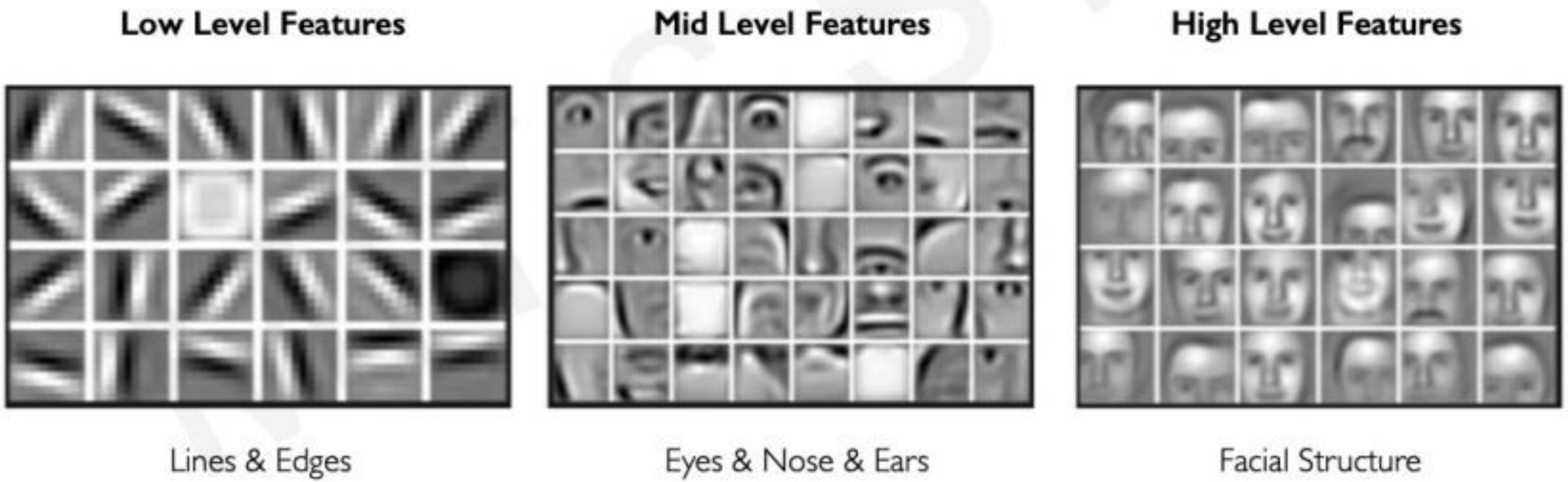
- Ein künstliches neuronales Netz kann darauf trainiert werden, um Objekte in Bildern zu segmentieren.
- Dabei wird zwischen **semantischer** und **Instanz**-Segmentierung unterschieden.
- Semantische Segmentierung bedeutet, dass alle Objekte einer Klasse als ein zusammenhängendes Segment interpretiert werden, währen es bei der Instanz Segmentierung darum geht, die einzelnen Objekte einer Klasse als Instanzen dieser Klasse behandelt und entsprechend separat segmentiert werden.

Low level Features, Mid level Features, High level Features

- **Low level Features:**
  - Pixelwerte
  - Farbinformationen
  - Einfache Kanten
- **Mid level Features:**
  - Texturmerkmale
  - Muster
  - Einfache Formen
- **High level Features:**
  - Komplexe Objekte
  - Zusammenhänge

- In der Bildverarbeitung sind **Features** *Merkmale*, die Objekte und deren Zusammenhänge in Bild beschreiben.
- Diese Features werden durch *Convolutional Layers* extrahiert und erlernt.

Beispiel



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

Deep Learning

- Um ein **künstliches neuronales Netz** zu trainieren, wird folgendes benötigt:
  - Ein qualitativer **Datensatz**
    - Es muss ein umfangreicher, vielfältiger Datensatz und mit ausreichender Varianz sein.
  - Eine geeignete **Netzarchitektur**
    - Es hängt von der Aufgabenstellung ab, aber je komplexer die Architektur ist, umso besser kann das trainieren.
  - Einen **leistungsstarken Rechner**:
    - Training auf der **CPU** langsam
    - Training auf der **GPU** schnell

Datensatz

- Um eine gute **Genauigkeit** mit den neuronalen Netzen zu erreichen, muss der Datensatz möglichst alle Fälle des gegebenen Problems abdecken.
  - **Umfangreich:**
    - Der Datensatz muss relativ groß sein, d.h. je mehr Daten enthalten sind, umso besser kann das Modell lernen.
  - **Vielfältig**
    - Der Datensatz muss möglichst alle Fälle über die Problemstellung abdecken.
  - **Mit ausreichender Varianz**
    - Varianz bedeutet, dass die gleichen Daten unterschiedliche Typen beinhalten.
  - **Ausgewogene Verteilung der Klassen im Datensatz:**
    - Die Klassen haben die gleiche Verteilung, d.h. jede Klasse hat die gleiche Anzahl an Datenpunkten.

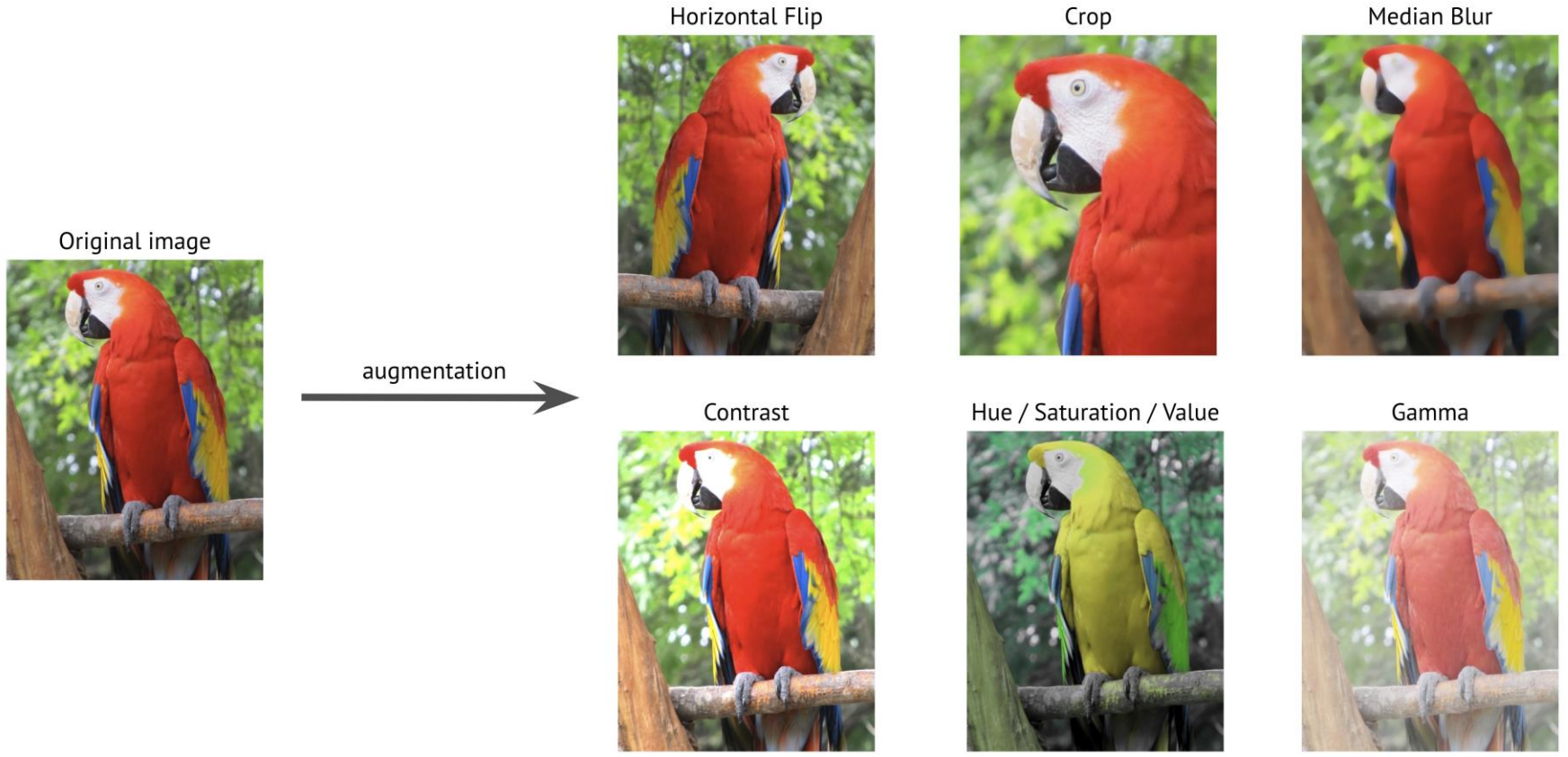
Datensatz

- Daten sind eine der wichtigsten Ressourcen und werden benötigt, um neuronale Netze für bestimmte Aufgaben zu trainieren.
- Es heißt, '*Wie deine Daten sind, ist so dein Modell*'
- Und bzgl. der Frage, wie kann an Daten herankommen kann?
  - Daten hängen stark von der Aufgabenstellung ab, sie werden z.B:
    - Selber gesammelt und vorverarbeitet.
    - Gekauft.
    - Von freien Internetquellen heruntergeladen.
    - Ect.

Augmentation

- Ist eine Technik, die in neuronalen Netzen verwendet wird, um künstliche Variationen von Trainingsdaten zu erzeugen.
- Beispiele sind also zufälliges:
  - Drehen
  - Zuschneiden
  - Spiegeln
  - Zoomen (rein und raus)
  - Ändern von Helligkeit und Kontrast
- Diese Technik dient dazu, dass der Datensatz mehr Bilder und Varianz erhalten kann.

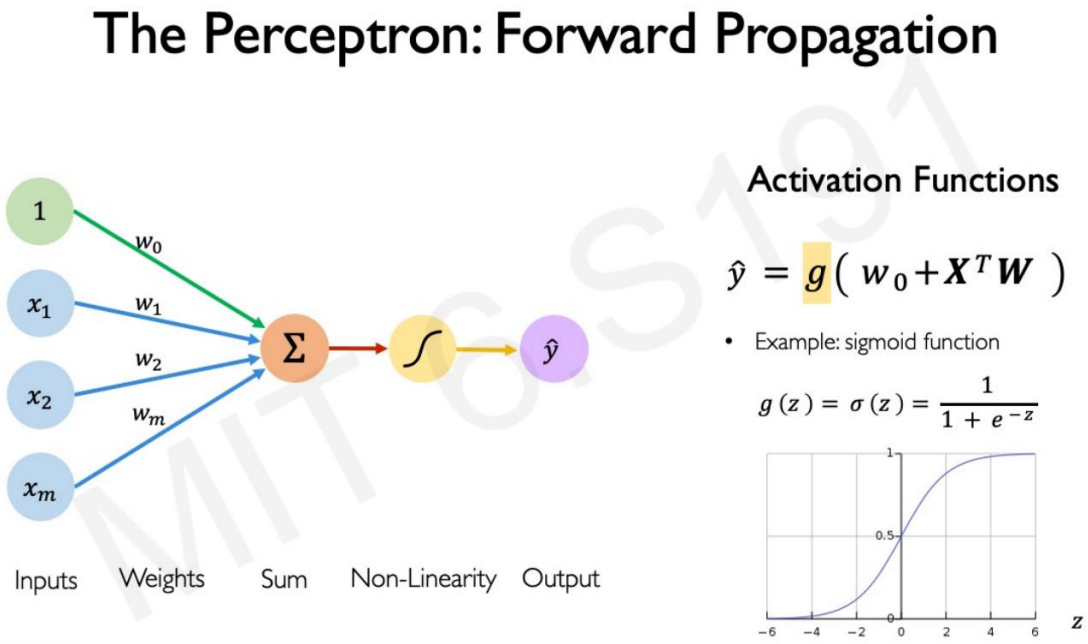
Beispiel für augmentierte Bilddaten



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

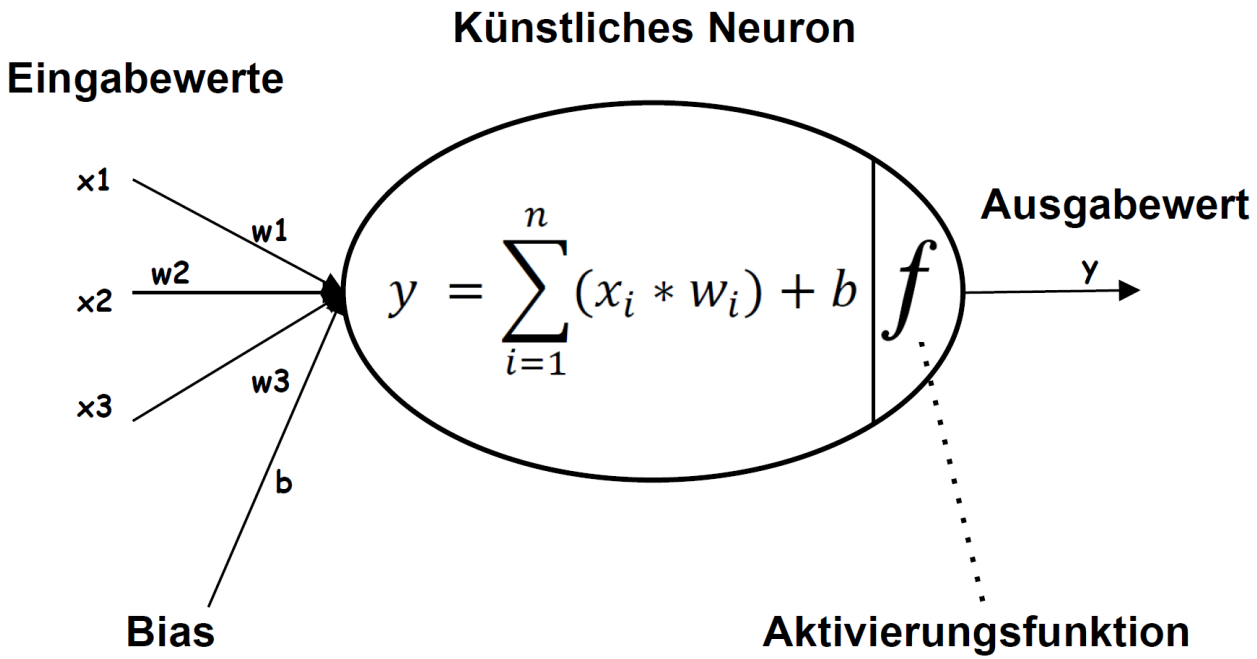
Das Perzeptron

- Die Eingabewerte werden mit den zufällig initialisieren Gewichten. Dann wird die Summe gebildet.
- $1*w_0 + x_1*w_1 + x_2*w_2 + x_3*w_3$
- Danach wird eine sogenannte Aktivierungsfunktion eingeschaltet. Diese ermöglicht es, manche Neuronen zu aktivieren oder nicht.



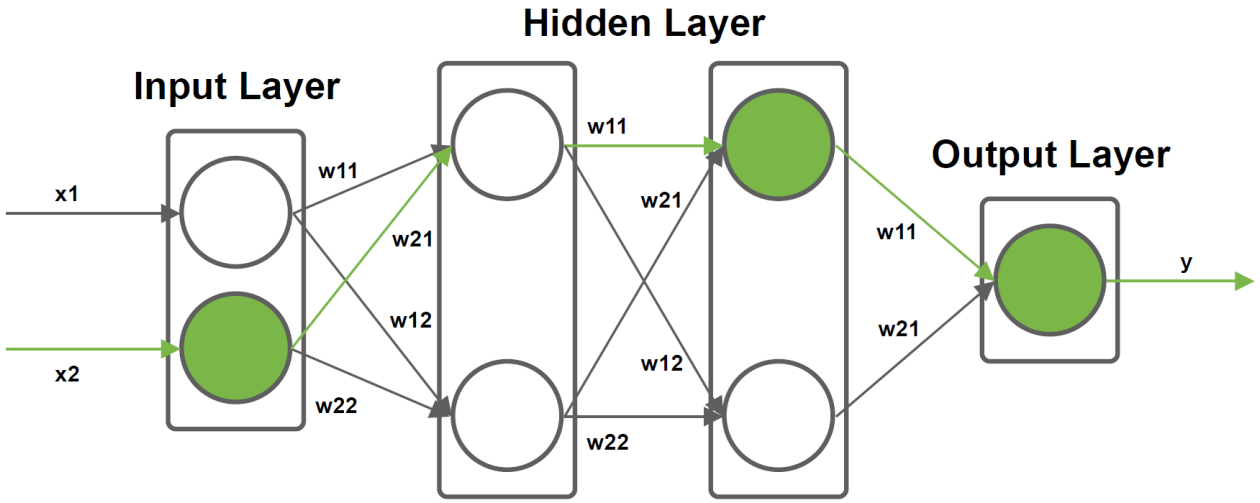
http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf

Das Perzeptron



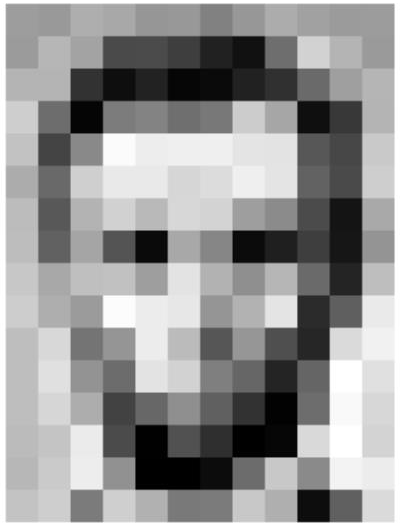
Einfaches künstliche neuronales Netz

- Folgendes neuronales Netz besteht aus einer *Eingabeschicht* **(Input Layer)**, einer *versteckten Schicht* **(Hidden Layer)** und einer *Ausgabeschicht* **(Output Layer)**.
- Die versteckten Schichten können beliebig viele sein. Dies definiert die **Tiefe** des neuronalen Netzes.
- Jede dieser Schichten kann beliebig viele Neuronen enthalten. Somit wird eine gewisse **Komplexität** des neuronalen Netzes erreicht.



Eingabeschicht

- Die Eingabeschicht enthält so viele Neuronen, wie die Eingabewerte. Das bedeutet, wenn z.B. die Bildgröße **28*28** Pixel ist, dann muss die Eingabeschicht **784 Neuronen** haben, wobei jedes Neuron einen Pixelwert annimmt.



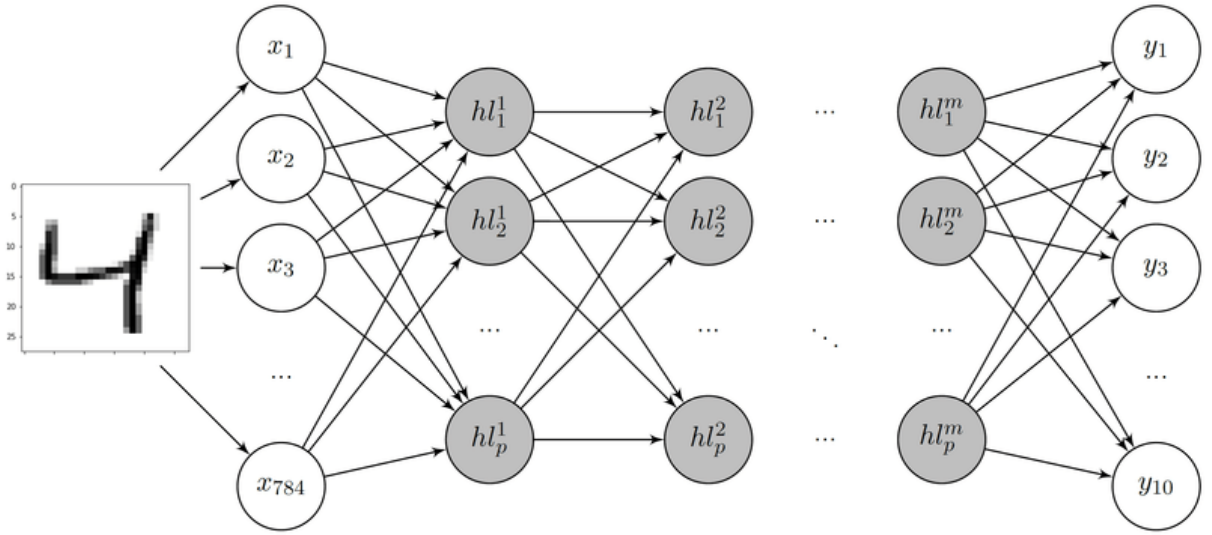
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	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
196	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
196	206	123	207	177	121	123	200	175	13	96	218

https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

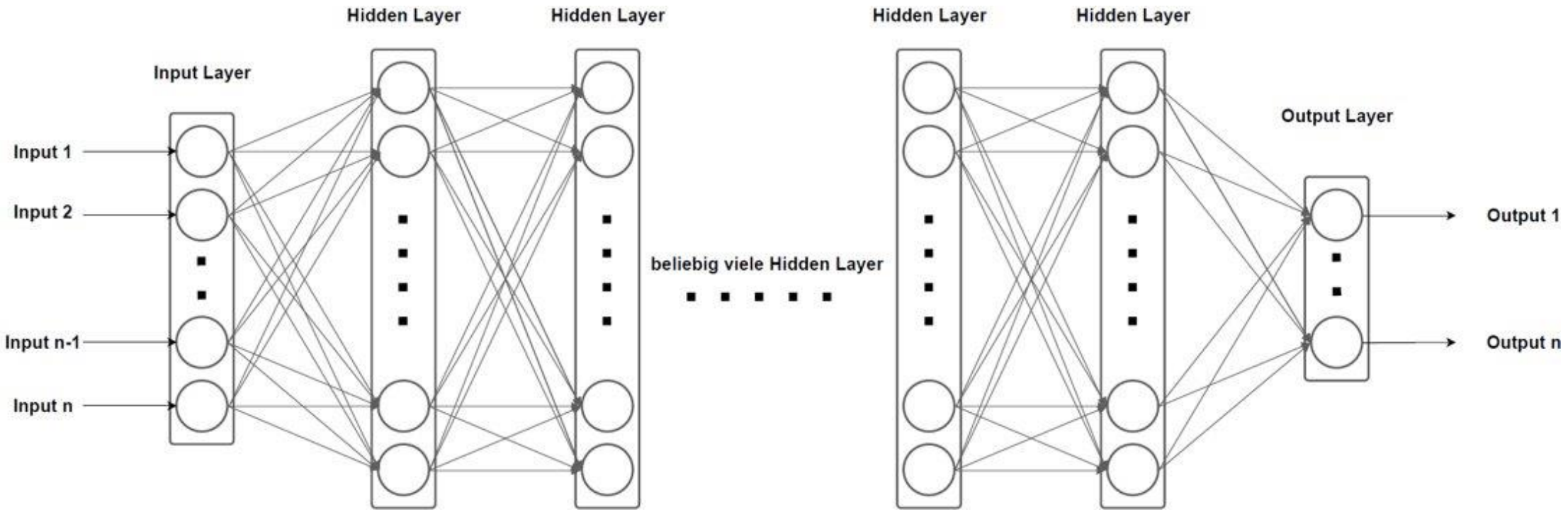
Ausgabeschicht

- Die Ausgabeschicht enthält so viele Neuronen, wie die Anzahl der Zielklassen ist. Das bedeutet, wenn es **10 Klassen** gibt, mit dem Ziel, das neue Bild in eine dieser Klassen zu klassifizieren, dann muss die Ausgabeschicht **10 Neuronen** haben, damit diese 10 Klassen repräsentiert werden können.

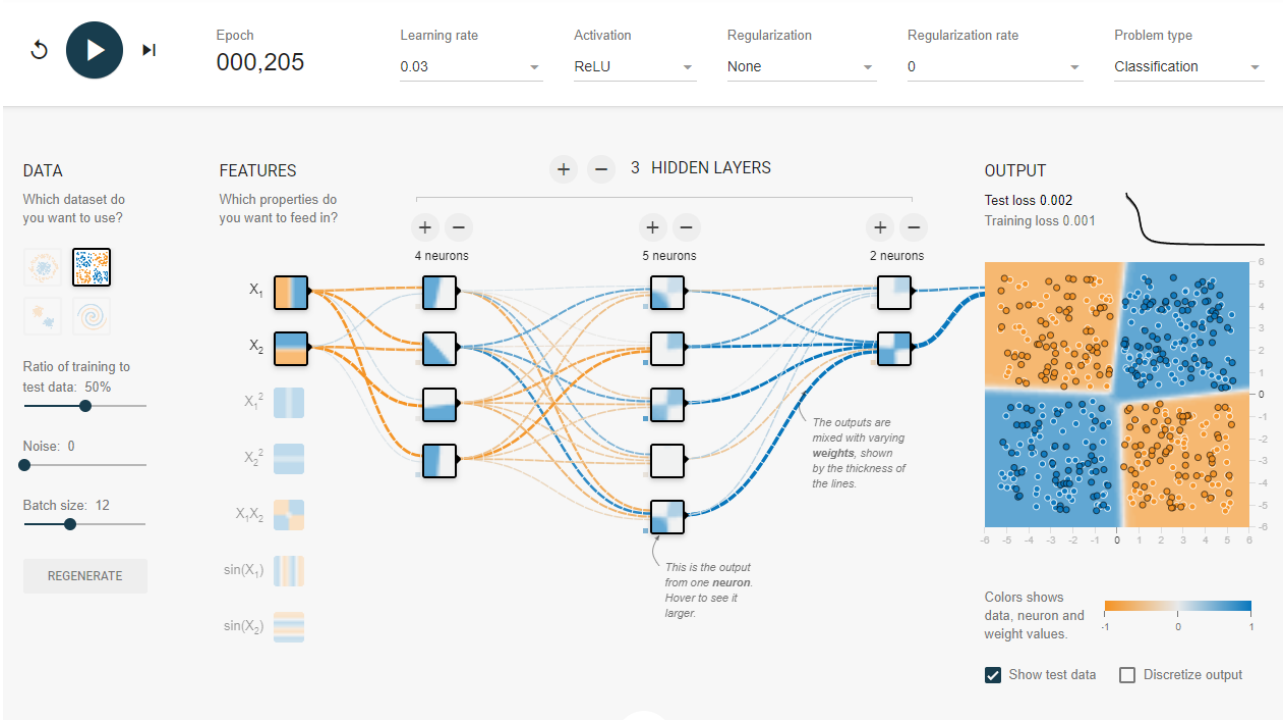


https://www.researchgate.net/figure/Artificial-neural-network-architecture-for-the-MNIST-dataset-classification-The-input_fig4_349991068

Ein tiefes neuronales Netz



Playground Tensorflow

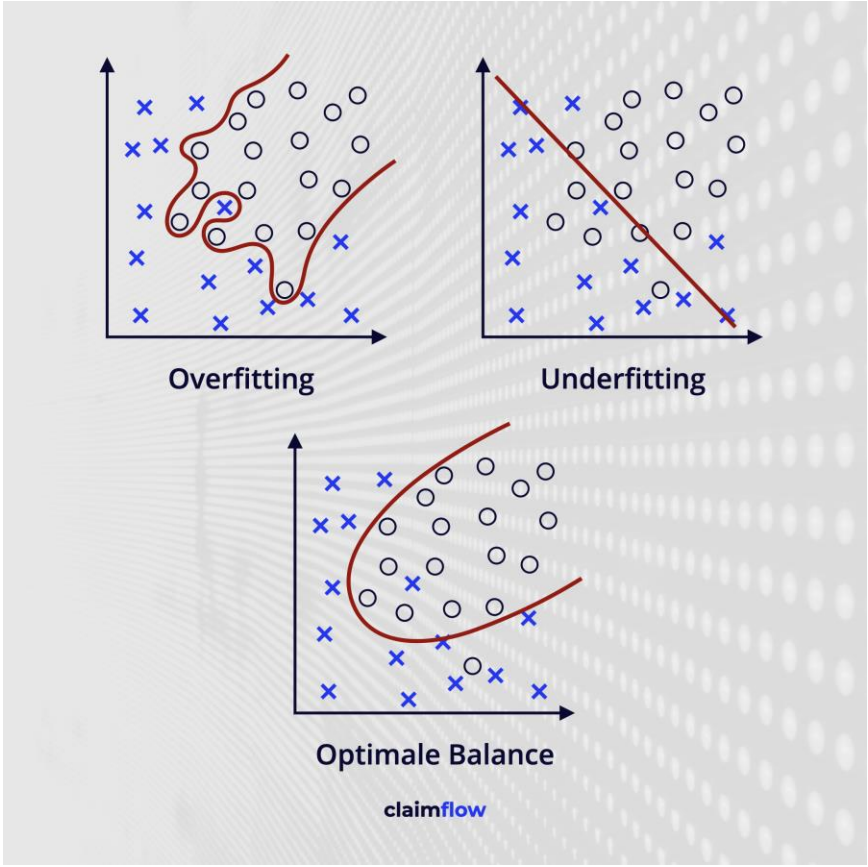


https://playground.tensorflow.org/

Overfitting, Underfitting und Optinale Balace

- **Overfitting:**

Es geht darum, dass sich das neuronale Netz zu stark an die Daten angepasst hat. Auf Trainingsdaten erzielt das Modell sehr gute Ergebnisse, aber auf die Testdaten nicht. Der Grund ist, dass das Modell die Features der Daten auswendig gelernt hat. Es kann also nicht generalisieren.

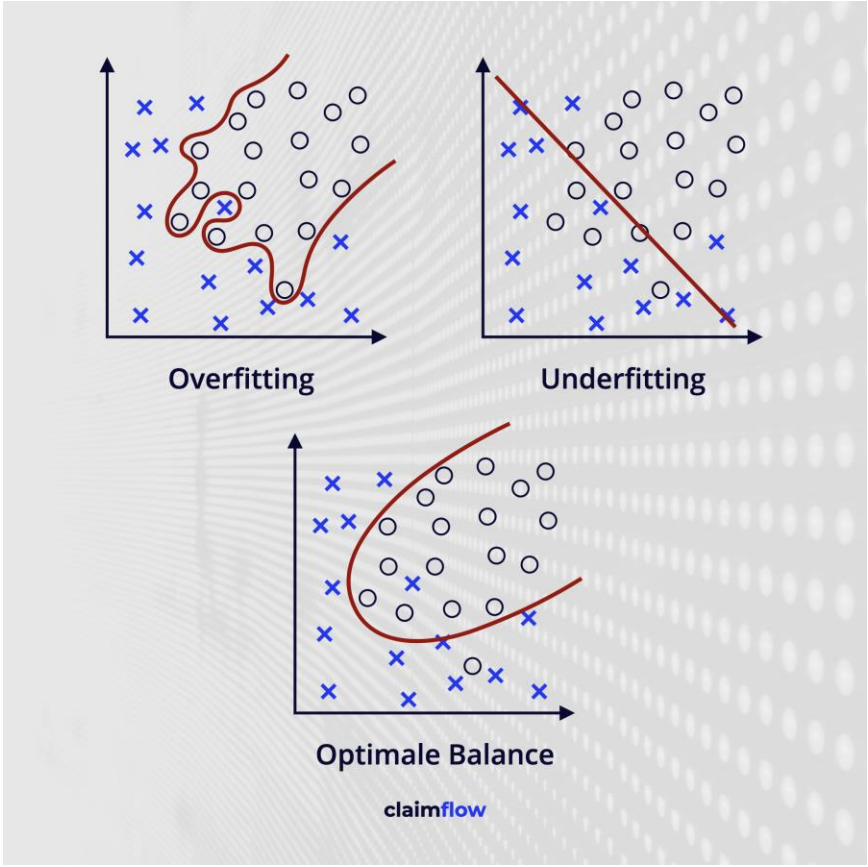


https://www.claimflow.de/post/overfitting-und-underfitting

Overfitting, Underfitting und Optinale Balace

- **Underfitting:**

Es geht darum, dass das neuronale Netz keine Features aus den Daten gelernt hat. Auf Trainingsdaten erzielt das Modell sehr schlechte Ergebnisse, aber auch auf die Testdaten. Der Grund ist, dass das Modell die Features der Daten nicht verstanden und gelernt hat. Es kann also nicht generalisieren.

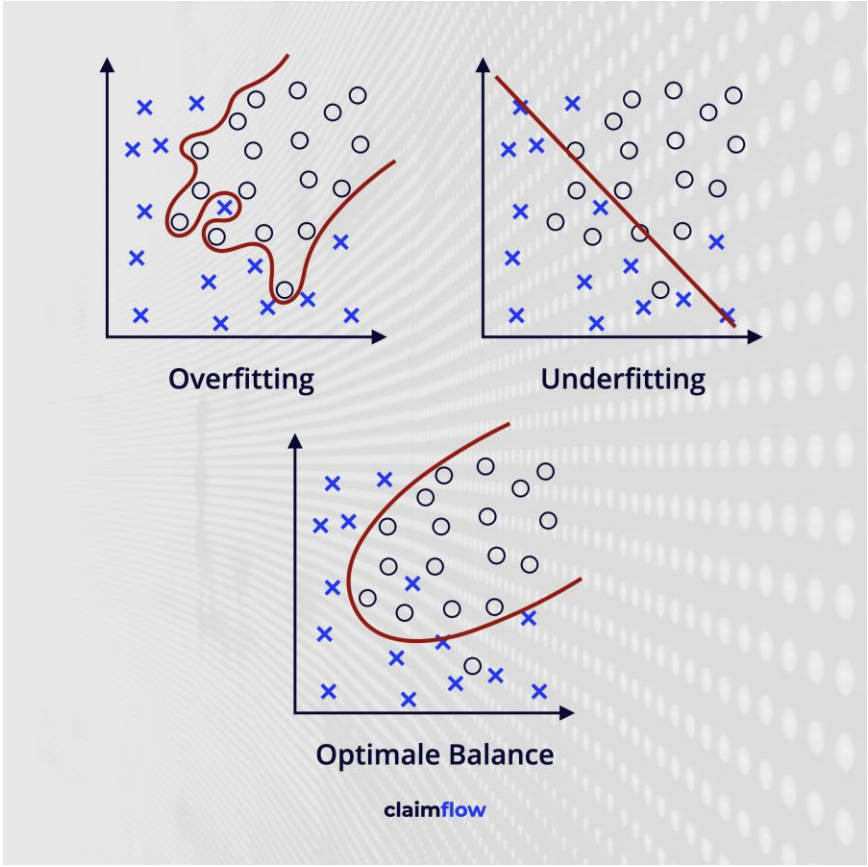


https://www.claimflow.de/post/overfitting-und-underfitting

Overfitting, Underfitting und Optinale Balace

- **Optimale Balance:**

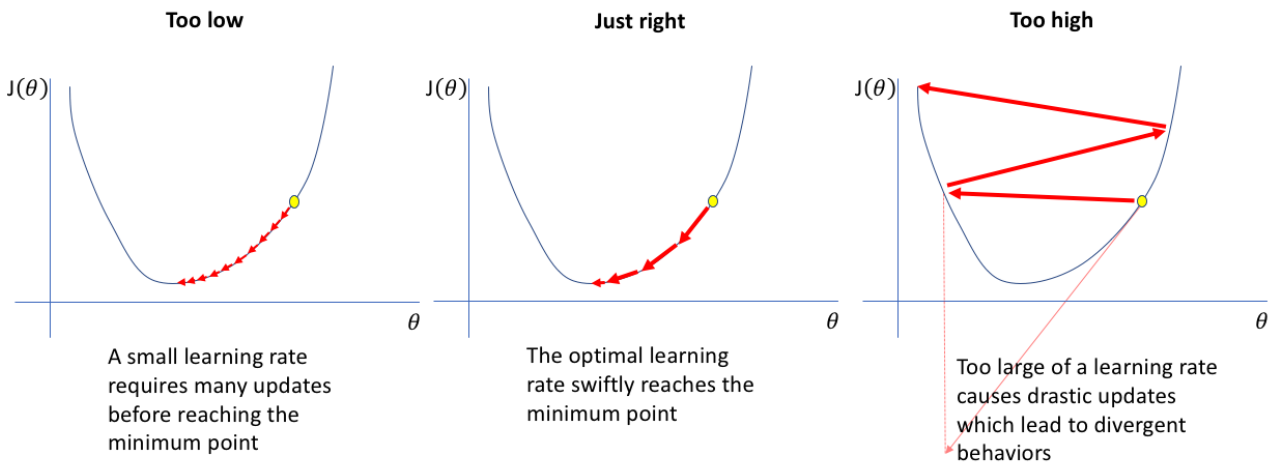
Es geht darum, dass das neuronale Netz ausreichend Features aus den Daten gelernt hat. Auf Trainingsdaten erzielt das Modell sehr gute Ergebnisse, aber auch auf die Testdaten. Der Grund ist, dass das Modell die Features der Daten verstanden und gelernt hat. Es kann also gut generalisieren.



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

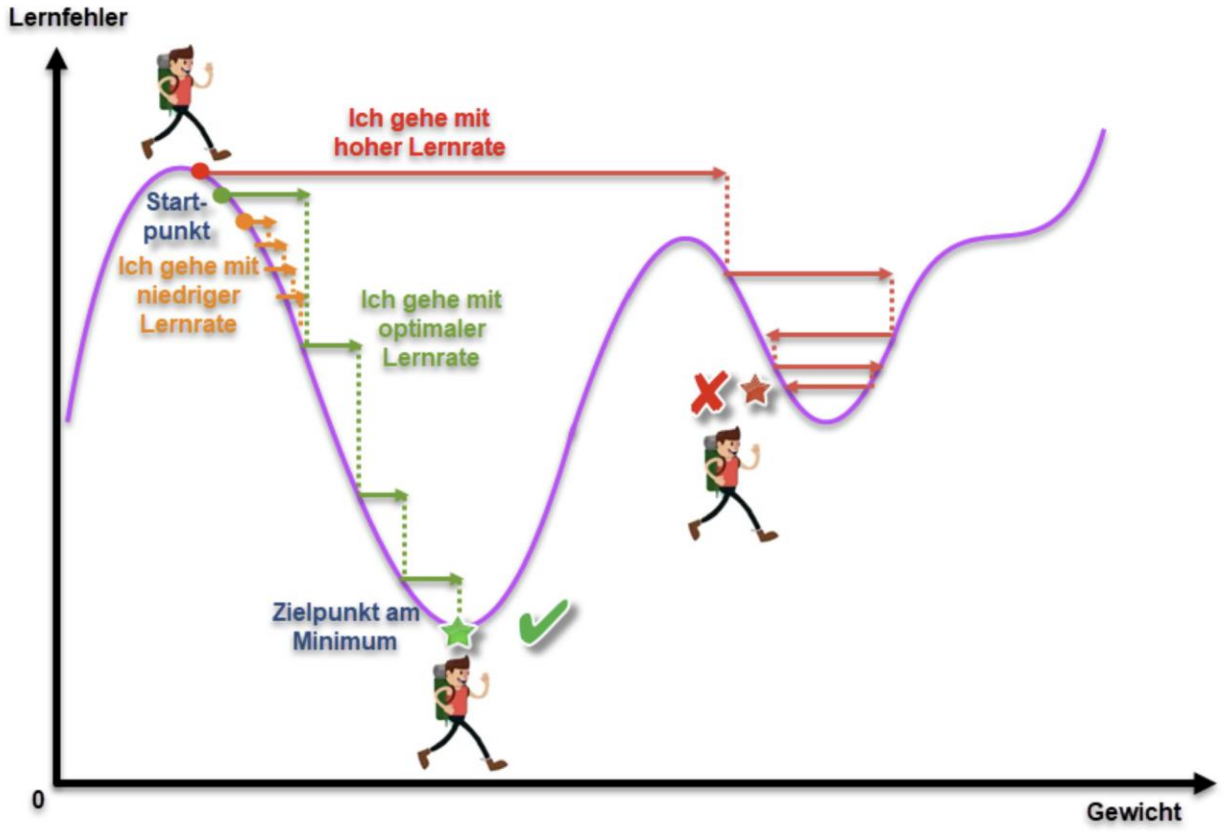
Lernrate

- Die Lernrate ist einer der wichtigsten Hyperparameter, die bei der Optimierung verwendet werden.
- Die Lernrate gibt an, wie groß die Schritte sind, während das Modell die Gewichtungen aktualisiert.

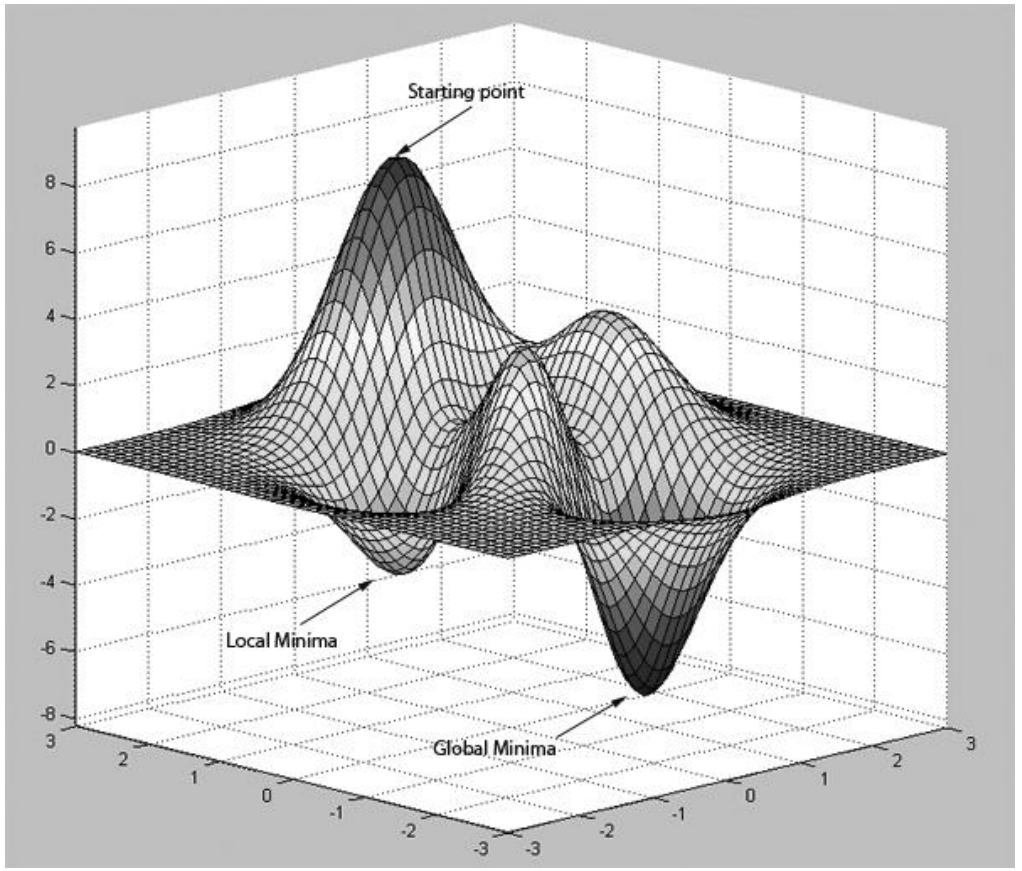


https://www.jeremyjordan.me/nn-learning-rate/

Beispiel für lokale und globale Maxima und Minima



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



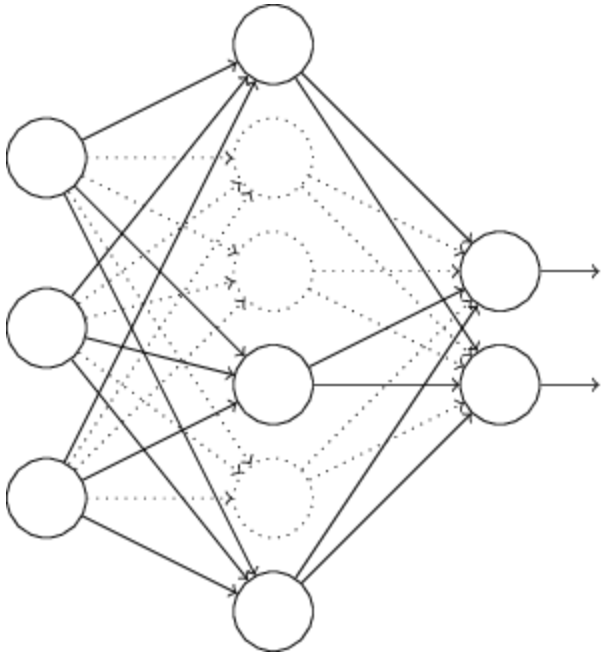
https://www.codeplanet.eu/tutorials/csharp/70-kuenstliche-neuronale-netze-in-csharp.html

Regularisierung

- Es geht darum, die Verallgemeinerung des Modells auf unsichtbare Daten zu verbessern.
- Es gibt u.a. zwei Optionen:
  - **Dropout:**
    - Es werden einige Verbindungen zu Neuronen absichtlich eliminiert.
  - **Early Stopping:**
    - Es wird an einer bestimmten Stelle ein Ende gesetzt, damit **Overfitting** und **Underfitting** vermieden werden können.

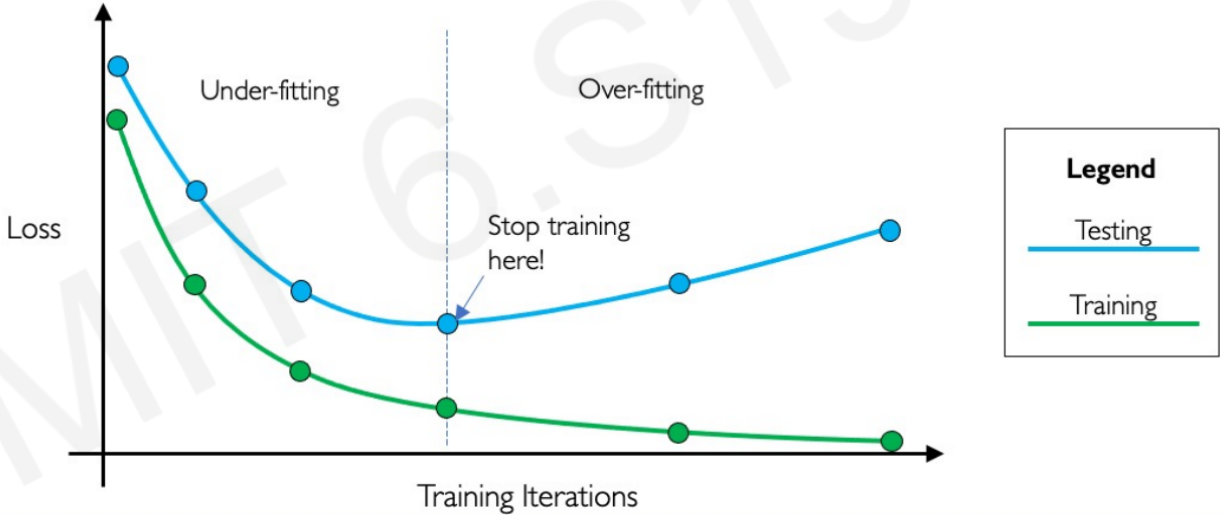
Regularisierung

Dropout



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

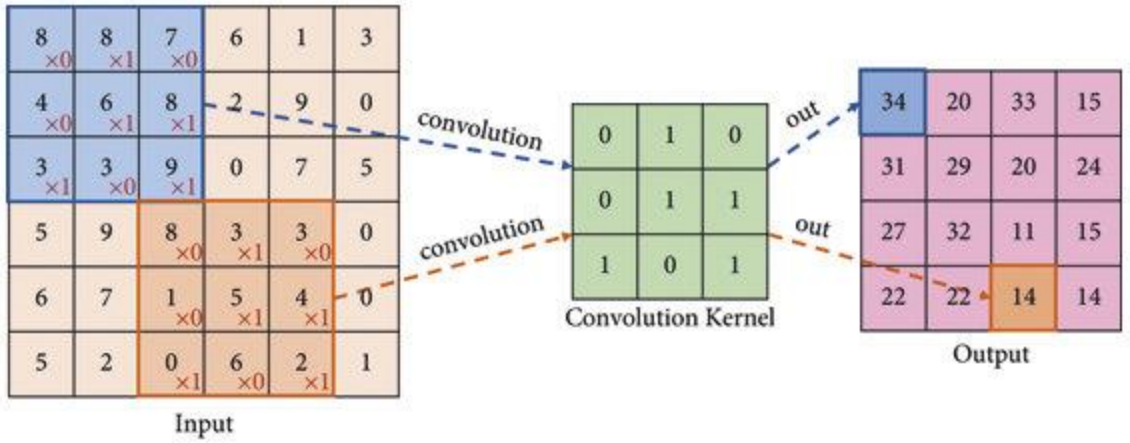
Early Stopping



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

Convolutional Operator

- Das ist eine spezielle Art von neuronalen Netzen, die für *Bilderdaten* eingesetzt werden. Es wird ein **Filter (Kernel)** auf das Bild verwendet.
- Dieser wird schrittweise von links nach rechts bewegt, dabei werden die Eingabewerte mit den Kernelwerten multipliziert und aufsummiert. Das Ergebnis dieser Operation wird schrittweise in eine **Feature Map** eingetragen.



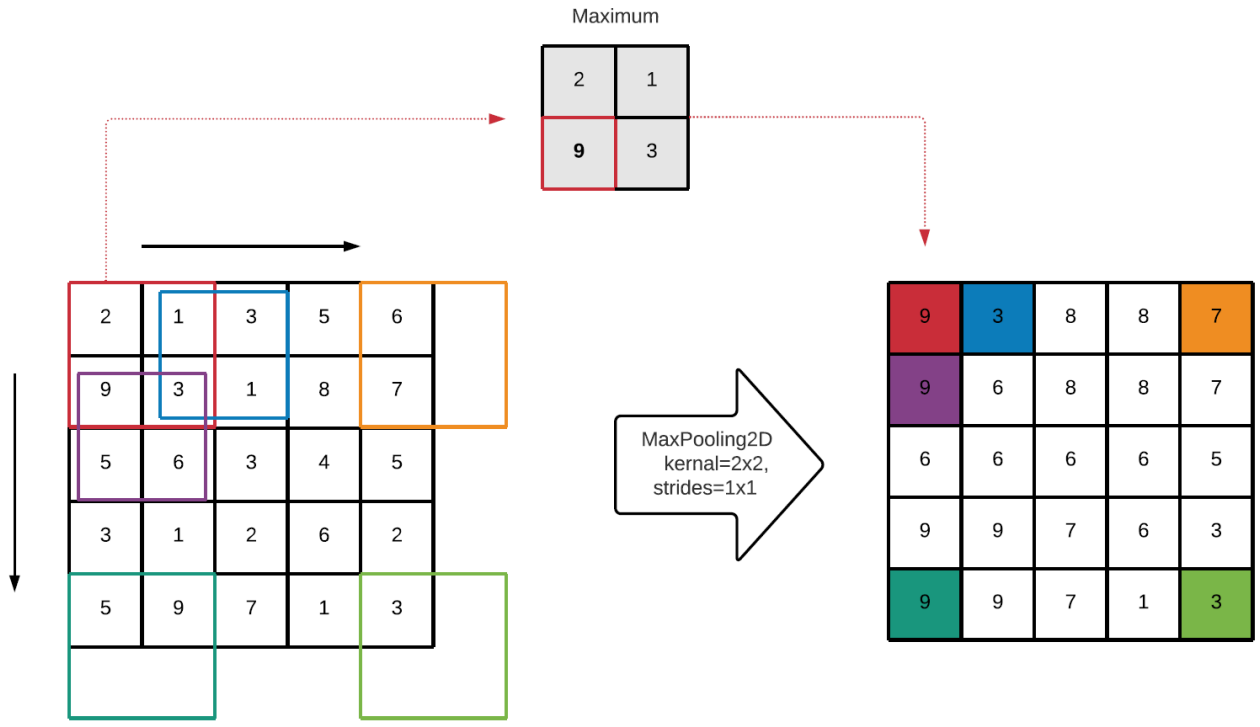
https://www.researchgate.net/figure/Convolution-with-convolution-kernel-of-size-33-stride-of-1-and-no-zero-padding_fig2_356590429

Convolutional Layer

- In der Convolutional Schicht kann folgendes bestimmt werden:
  - Die Anzahl der Filter
  - Die Größe des Kernels
  - Die Schrittgröße (*stride*)
- Mit der Convolutional Schicht kann folgendes erreicht werden:
  - Extraktion von Features und nützlichen Merkmalen
  - Reduzierung der Dimension der Bilddaten (Mehr Speicher und weniger Rechenleistung)

Pooling Operator

- Die Pooling Operation wird direkt nach der Convolutional Operation verwendet.
- Die Hauptidee ist, dass die Dimension der *Feature Map* zu reduzieren, wobei die relevanten Informationen beibehalten werden.

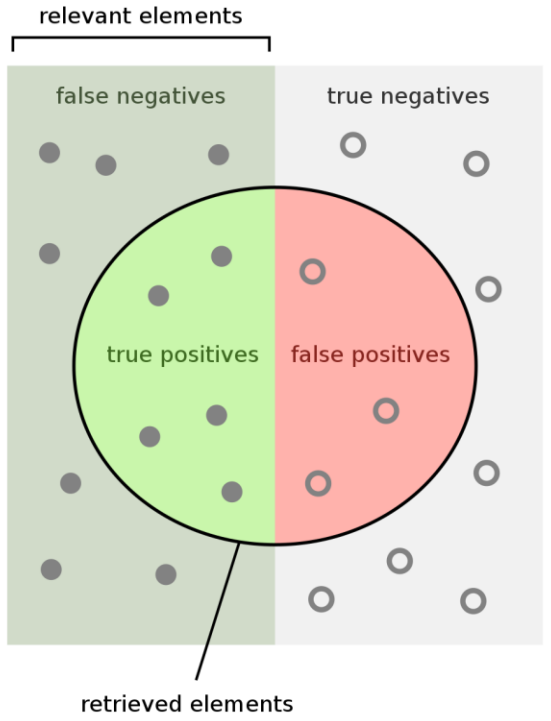


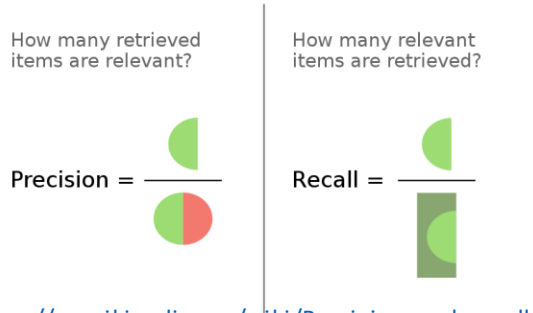
Pooling Layer

- In der Pooling Schicht kann folgendes bestimmt werden:
  - Die Größe des Kernels
  - Die Schrittgröße (*stride*)
  - Bounding Schicht (*padding*)
- Die Padding Schicht hilft dabei, dass die Dimension der Feature Map beibehalten wird. Das heißt, dass die Größe der Ein- und Ausgabe gleich ist.
- Mit der Pooling Schicht kann folgendes erreicht werden:
  - **MaxPooling2D**
    - Es wird die maximale Zahl im Pooling-Kernel gewählt.
  - **AveragePooling2D**
    - Es wird die durchschnittliche Zahl im Pooling-Kernel gewählt.

Recall und Precision

- Diese sind Metriken für die Evaluation der neuronalen Netze.
- **Recall**: Wie viele relevante Datenpunkte wurden gefunden.
- **Precision**: Wie viele gefundene Datenpunkte sind relevant bzw. richtig.

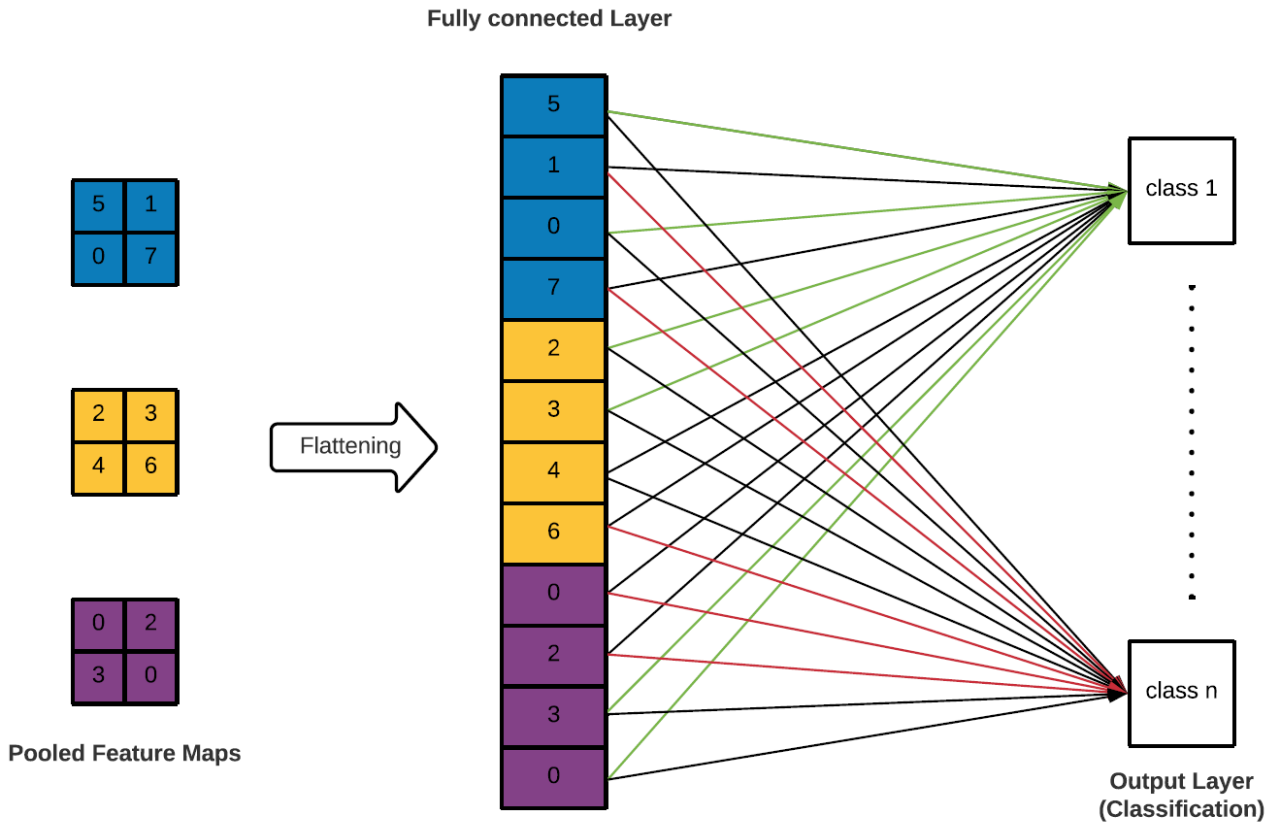




https://en.wikipedia.org/wiki/Precision_and_recall

Flattening

- *Feature Maps* aus der letzten *Convolutional Schicht* werden geglättet, sodass sie in das vollständig verbundene neuronale Netz eingefüttert werden.
- Jeder dieser Werte ist der Input für die Eingabeneuronen.



Projekt

- In diesem Modul ist ein Projekt vorgesehen. Dieses wird als Teil der Prüfungsleistung angerechnet.
- Das Projekt erfordert:
  - Einen Datensatz über mind. zwei Klassen.
    - Wenn nötig, sollte die Bilddaten vorverarbeitet werden.
  - Eine geeignete Archietektur für ein Convolutional neuronales Netz.
    - Es ist notwendig, verschiedene Konstellationen und Kombinationen der Hyperparameter auszuprobieren.
  - Jedes Ergebnis der Genauigkeit des Modells sollte mit der jeweiligen Kombination an Parametern gespeichert werden.
  - Es wird ein gutes Ergebnis für das gegebene Klassifikationsproblem erwartet.

Google Colab

- Das ist wie ein **Jupyter Notebook**, welches mit Cloud Ressoursscen ausgeführt werden kann.
- Der Code kann auf **CPU, GPU oder TPU** ausgeführt werden.
- Diese kann die Rechenleistung beschleunigen.
- Der Code kann mit **Google Drive** verbunden werden, um den *RAM* und die *Speicherkapazität* zu schonen. Die Daten können somit direkt über die Cloud geholt werden.
- Für mehr Informationen siehe: https://colab.google/