

**Kompensationsarbeit im Modul Software Entwicklung**

Master:

“Data Science & Intelligent Analytics”

Betreuer :

Huber Stephan

Autor:

Jochen Hollich | 1810837475

Datum

10.12.2019

Inhalt

[Präambel 3](#_Toc27637381)

[Einleitung 4](#_Toc27637382)

[Abgrenzung der Begrifflichkeiten 4](#_Toc27637383)

[Abgrenzung Machine-Learning, Artificial-Intelligence und Deep-Learning 5](#_Toc27637384)

[Theorie Neuronale Netze 6](#_Toc27637385)

[Vorbild Biologie 6](#_Toc27637386)

[Überleitung von biologischen zu technischen Neuronalen Netzen 7](#_Toc27637387)

[Konzeptionelle Funktionsweisen Neuronaler Netze im Kontext Data-Science 12](#_Toc27637388)

[Convolutional Neuronal Network 12](#_Toc27637389)

[CNN Hauptkomponenten 12](#_Toc27637390)

[In der weiteren praktischen Ausarbeitung dieser Arbeit werden wir mit Bildmaterial aus den bekannten Datensätze MNIST und CIFAR10 arbeiten, daher verwenden wir in der Beschreibung der Funktionsweis von CNNs auch ein Beispiel der Bildanalyse. 12](#_Toc27637391)

[Input 12](#_Toc27637392)

[Convolutional Layer 13](#_Toc27637393)

[Pooling Layer 14](#_Toc27637394)

[Fully Connected Layer + 14](#_Toc27637395)

[Softmax Layer 14](#_Toc27637396)

[Output 14](#_Toc27637397)

[CNN Ablauf 14](#_Toc27637398)

[Rückschluss auf die Umsetzung in Python mit Tensorflow 14](#_Toc27637399)

[CNN-Architekturen 15](#_Toc27637400)

[Praktische Implementierung in Python 15](#_Toc27637401)

[Praktische Implementierung 15](#_Toc27637402)

[MNIST 15](#_Toc27637403)

[CIFAR10 16](#_Toc27637404)

[Abbildungsverzeichnis 18](#_Toc27637405)

[Literaturverzeichnis 18](#_Toc27637406)

# Präambel

Aufgrund des begrenzten Seitenspektrums werden in dieser Arbeit keine Begrifflichkeiten definiert.

# Einleitung

Viele Technologien der Menschheit finden Ihren Urspring in der Beobachtung natürlicher Phänome. So inspirierten uns Vögel hinsichtlich der Luftfahrt, die Lotuspflanze war das Vorbild für Ingenieure bei der Entwicklung von Autolacken, oder .Die gegenwärtige Phase wird auch als das digitale Zeitalter bezeichnet, somit erscheint es nun logisch eine Maschine nach dem Vorbild des menschlichen Gehirns intelligent zu gestalten.

Die Begriffe Artificial Intelligence, Machine-Learning und Deep-Learning sind Technologien, welche zugleich auch intensiv in den Medien diskutiert werden. Im Grunde genommen sind diese Disziplinen die praktische Implementierung und die technische Optimierung von mathematischen Algorithmen in einer rechenstarken, IT-gestützten Umwelt. Folglich werden unabhängig von der Komplexität der logischen und physischen Implementierung die Ziele der Algorithmik mit der Regression, Klassifikation (Binär, Multiclass, Multilabel, Multioutput), Clusterings oder Anomalie-Detection verfolgt. Auch wenn dies zunächst für den Leser abstrakt wirken mag, können viele Aufgaben aus dem privaten wie beruflichen Umfeld u.a. mit dieser Technologie automatisiert und beschleunigt werden.   
Veranschaulicht an dem aktuell diskutierten Thema „Autonomes Fahren“ spielt die Technologie von Computer Vision(Teilgebiet des Deep-Learnings) eine zentrale Rolle. Die Computer Vision befasst sich damit, aus digitalen Bildern und Videos Information herauszuarbeiten und diese Informationen für weitere Ver- und Bearbeitungsschritte bereitzustellen. Analog zu dem Beispiel des Autonomen Fahrens, kann selbige Technologie in zahlreichen anderen Anwendungsgebieten / Domänen wie bspw. der Biomedizin, Überwachung, Umweltwissenschaften, Sozialwissenschaften verwendet werden.

Vorliegende Arbeit befasst sich mit der Verwendung von Neuronalen Netzen, konkret Convolutional Neuronalen Netzwerken (CNN) vorwiegend im Bereich der Computer Vision. CNNs können jedoch ebenfalls mit einer hohen Erfolgsquote in weiteren Klassifizierungsfragestellungen verwendet werden. Um die theoretischen, beschriebenen Aspekte zu untermauern, wurden praktische Implementierungen der Konzepte in Jupyter-Notebooks auf Basis der Programmiersprache Python erstellt und an den entsprechenden Stellen dieser Arbeit verwiesen. Das gesamte Repository ist öffentlich zugänglich und kann unter [diesem Link](https://github.com/JHC90/SoftwareDev-Huber) heruntergeladen werden.

# Abgrenzung der Begrifflichkeiten

Bevor im Folgenden auf die nähere Erläuterung von Neuronalen Netzwerken (NN) und anschließend auf die CNNs eingegangen wird, werden noch etwaige Begrifflichkeiten geklärt.

## Abgrenzung Machine-Learning, Artificial-Intelligence und Deep-Learning

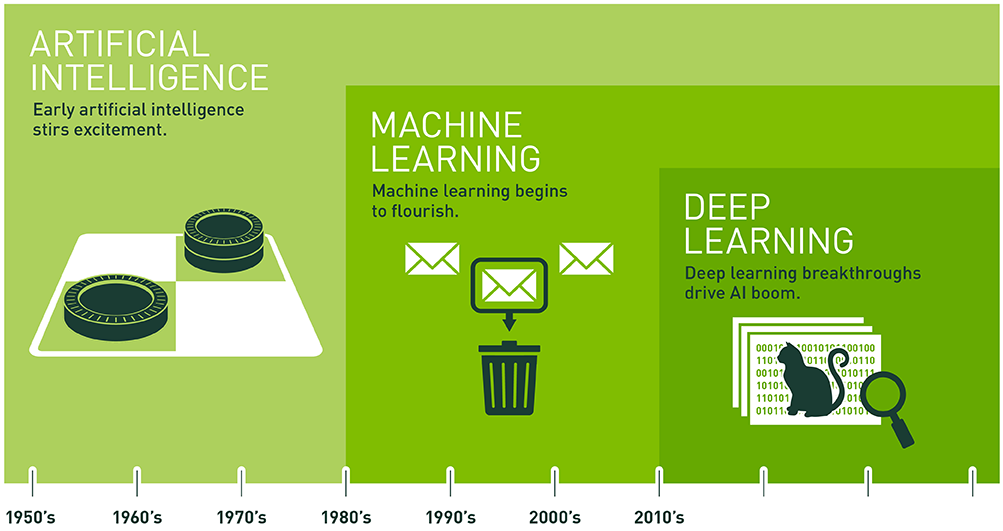


Abbildung : Entwicklung AI, ML, DL (https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/)

Die obenstehende Grafik veranschaulicht die Entwicklung der Begriffe anhand der zeitlichen Historie.

**Artificial Intelligence**

AI / KI zielt darauf ab menschliches Verhalten durch Verwendung der Gesetzte aus der Logiklehre, Entscheidungsbäumen und Konditionalverzweigungen imitieren. Berühmte Beispiele sind IBM’s Deep Blue (Schachautomatisierung| Schlug im Jahr 1996 den damaligen Weltemeister Garry Kasparov) oder Googles AlphaGo, welches ausschließlich für die Automatisierung des Spiels Go dient, und ebenfalls erflogreiche Go-Titel-Inhaber schlug. Der Unterschied der beiden Technologien lag darin, dass IBM’s Deep Blue jeden weiter möglichen Spielzug im Laufe eines Schachspiels„brute-frocte“. Dieser Ansatz war jedoch durch die höhere Komplexität des Spieles Go nicht übertragbar.

**Machine-Learning**

ML bietet für ein System die Möglichkeit aus bestehenden Daten zu lernen, ohne explizit auf den möglichen Output hin programmiert zu sein darauf hin.

Im Gegensatz zu dem Deep-Learning kann im Machine-Learning ein Mensch in die Datenanylse (noch) eingreift.

**Deep-Learning**

DL ist die ein Teilbereich des des ML. Hierbei werden Muster aus Datensätze mithilfe von Neuronalen Netzwerken extrahiert. Neuronale Netze wiederum basieren auf der Funktionsweise des biologischen Nervensystems. Somit kann der Prozess des Lernens automatisiert werden. Neuronale Netze adaptieren sich selbst während des Lernvorgangs, dem sogenannten „Trainings“. Diese entstehenden Modelle werden anschließend in der Verwendung Neuronaler Netze für die Prognose verwendet.

Der Kern der vorliegenden Arbeit befasst sich mit dem Thema Neuronale Netze und im Speziellen mit Convolutional Neuronal Networks. CNN ist ein Vertreter des Deep-Learning. Mit diesem Setup bietet sich grundsätzlich die Analyse von Bilddaten an.  
Im späteren praktischen Teil werden somit zu den Beispiel-Datensätzen MNIST und CIFAR10 entsprechend CNNs aufgebaut. Als Benchmark werden jedoch auch einfachere Klassifikationsmodelle aus dem Bereich des Machine-Learnings verwendet.

# Theorie Neuronale Netze

Neuronale Netze – sowohl bei biologischer als auch bei technischer Betrachtungsweise – verfolgen das Ziel mittels ähnlicher Konzepte zu lernen und zu prognostizieren. Dabei wird eine sich verändernde aber ähnliche Umwelt analysiert, um mit dem Erlernten Aussagen über ähnliche Bedingungen tätigen zu können. Die Komplexität der untenstehenden Analyse biologischer Nervensysteme / Neuronen übersteigt die hier ausgeführte Schilderung bei Weitem. Das folgende stark vereinfachte Beispiel wurde lediglich gewählt, um zu der Funktionsweise technischer Neuronaler Netze hinzuführen. Somit wurde nicht tiefer als nötig auf die Funktionsweise biologischer Neuronen und Nervensysteme eingegangen.

## Vorbild Biologie

Bevor nachfolgend technische Neuronale Netze analysiert werden, fokussiert sich diese Arbeit zunächst auf das biologische Vorbild dieser Technologie, bspw. dem menschlichen Nervensystem. Vereinfacht gesprochen werden Reize in Form von elektrischen Signalen aufgenommen, über das Nervensystem mittels elektrischer Impulse über unzählige, miteinander verflochtene Neuronen hinweg zu den entsprechenden Destinationen hin weitergeleitet und zuletzt an entsprechende Neurone verarbeitet ausgeben. Dabei bilden die Neuronen in unserer Betrachung die kleinsten individuellen Einzelsysteme für diese Weiterleitung.

In einem nächsten Schritt wird, vor der Betrachtung des „großen Ganzen“, der Funktionsweise des Nervensystems, auf die Funktionsweise eines einzelnen biologisch bipolaren Neurons eingegangen.

Ein einzelnes biologisches Neuron vereint die nachfolgenden Bestandteile mit den entsprechenden Funktionen in sich:

1. Nervenzelle | Neuron  
   Diese Bezeichnung beschreibt eine Zusammensetzung bestehend aus den folgenden Komponenten:
   1. Dendriten |Input  
      Dendriten sind die Teilbereiche bei welchen die elektrischen Signale (bspw. kommend von vorgelagerten Neuronen) eintreffen und zum Zellkern weitergeleitet werden. In unsere Betrachtungsweise hat ein Neuron mehr als ein eingehends Dendrit. Diese einkommenden Signale können reizend oder hemmend sein.
   2. Zellkörper / Schoma | Verarbeitung  
      Alle eintreffenden Signale von den Dendriten werden in der Zelle verarbeitet. Abhängig von dem Outcome dieser Verarbeitung wird der Status des Zellkörpers entweder in Form des Aktionspotentials aktiviert oder deaktiviert über das Axon ausgegeben. Die Höhe des Schwellwerts für die Zustandsänderung ist anhängig von dem jeweiligen Neuron.
   3. Axon | Output  
      Axonen sind Teilbereiche des Neurons welche das Signal des gegenwärtig fokussierten Neurons ausgeben.
2. Synapse   
   Als Synapsen bezeichnet man die Schnittstellen zwischen Axon und Dendrit

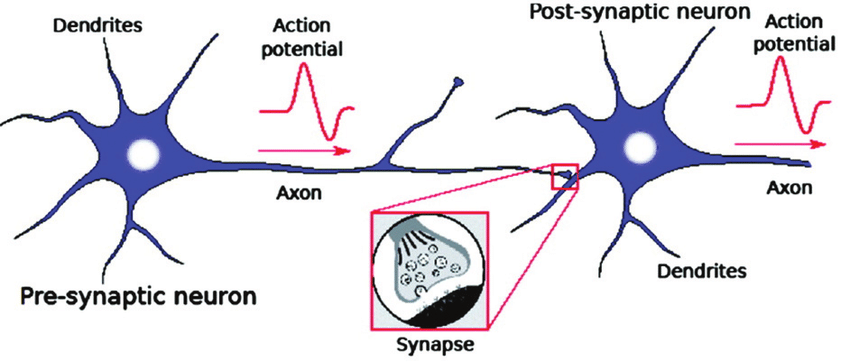


Abbildung 2, Neuron-Aufbau (https://medicalxpress.com/news/2018-07-neuron-axons-spindly-theyre-optimizing.html)

Zusammengefasst kommt über eine Synapse ein eingehendes elektrisches Signal am Dendriten unseres betrachteten Neurons an. Dieses Signal (und alle weiteren eintreffenden Signale kommend von weiteren dritten Neuronen) wird an den verarbeitenden Zellkern weitergeleitet. Abhängig von der Aktivierung des Zellkerns gibt dieser ein entsprechendes Aktionspotential an das Axom, welches wiederum über eine Synapse mit der nachgelagerten Nervenzelle verbunden ist.

**Biologische Neuronale Netze | Nervensysteme**

Wie oben geschildert können Neuronen über die Synapsen an dem Axon und Dendriten miteinander verbunden sein. Durch entsprechende parallele und sequentielle Anordnung von Neuronen entstehen somit komplexe Neuronen-Netzwerke, die sogenannten Nervensysteme. „Das menschliche Gehirn besteht aus etwa 1010 bis 1012 miteinander vernetzten Nervenzellen, den Neuronen“. ()

## Überleitung von biologischen zu technischen Neuronalen Netzen

Auf Basis des Grundverständnisses biologischer Netzwerke werden nachfolgend technische Neuronale Netzwerke betrachtet. Der Transfer des zuvor gewählten Beispiels der Beschreibung der Funktionsweise eines einzelnen biologischen Neurons in das technische Umfeld, lässt das „einfachste“ Perzeptron-Neuronale-Netzwerk entstehen.

Ein Neuronales Netz, im vorliegenden Fall das Perceptron besteht aus den nachfolgenden Bestandteilen mit den entsprechenden Funktionen:

1. Input-Layer

Dies ist der Startpunkt eines Neuronalen Netzes. In diesem Punkt werden die ggf. bereits vorverarbeiteten Daten in das NN „gespeist“.

1. Input eines Neurons | X  
   Der Input eines spezifischen Neurons ist die Summe aller einkommenden Signale multipliziert mit dem Gewicht, welches spezifisch für das einkommende Dendrit / Synapse gilt.
2. Gewicht | w

Technische Neuronale Netze sind durch die Gewichtung der gerichteten Dendriten geprägt. Diese Gewichte werden während des Trainings von Neuronalen Netzen adaptiert

1. Zellkörper | Actiavation-Function  
   Im Zellkörper fließen alle einkommenden Produkte (Input x Gewicht) aus den existenten Dendriten zusammen. Abhängig von der Höhe der Summe aller Produkte und der gewählten Aktivierungsfunktion (bspw.- „Sigmoid-Function“, Hyperbolic Tangent, „Rectified Linear Unit|ReLU“) wird ein Signal weitergeben oder nicht.
2. Output | y  
   Der Wert den ein technischen Neuron über das,
3. Hidden Layer

Zwischen Input und Output Layer befindet sich in einem Neuronalen Netzwerk mindestens ein Hidden-Layer. Je mehr Hidden Layer innerhalb eines Neuronalen Netzes bestehen, desto tiefer ist dieses Modell.

1. Der Output-Layer

Der Output-Layer „liegt hinter den Zwischenschichten und bildet die letzte Schicht in einem künstlichen neuronalen Netzwerk.“(<https://jaai.de/kuenstliche-neuronale-netze-aufbau-funktion-291/>)

1. .

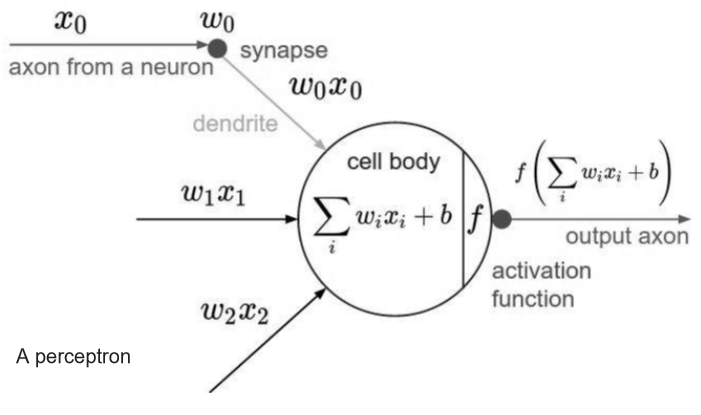


Abbildung , Perceptron Aufbau (Foliensatz Machine-Learning- Miroslav / Teil4 Deep-Learning)

**Technische Neuronale Netze | Architekturen**

Ähnlich den biologischen Neuronalen Netzen können technische Neuronale Netze ebenfalls parallel und sequentiell aufgespannt werden. Auch wenn sich hierdurch unterschiedliche Architekturen bilden können, gibt es folgende Gemeinsamkeiten: „Die Neuronen (auch Knotenpunkte) eines künstlichen neuronalen Netzes sind schichtweise in sogenannten Layern angeordnet und in der Regel in einer festen Hierarchie miteinander verbunden. Die Neuronen sind dabei zumeist zwischen zwei Layern verbunden (Inter-Neuronlayer-Connection), in selteneren Fällen aber auch innerhalb eines Layers (Intra-Neuronlayer-Connection).“()

|  |  |
| --- | --- |
| Beispiel: Single-Layer | Beispiel Multi-Layer |
|  |  |

Werden die beiden obenstehenden NNs verglichen, so werden folgende Gemeinsamkeiten ersichtlich:

1. Jedes Neuronale Netz hat ein Input-Layer. Das kann beispielsweise ein Vektor, ursprünglich kommend von einem digitalen Bild, sein.
2. Architektur
   1. Einerseits besteht die Architektur aus der Anzahl der Hidden-Layer (Hidden Layer = weder Input noch Output-Layer).
   2. Jeder Layer wiederum besteht aus einer Anzahl von Neuronen.
3. Jedes Neuronale Netz hat einen Output-Layer.
   1. Dies kann ein Single-Output (bspw. Implementierung einer binären Klassifikation) …
   2. …oder ein Multioutput (bspw. Implementierung einer Multi-Output-Klassifikation) sein.

Vice-Versa können folgende Unterschiede von Neuronalen Netzen detektiert werden:

1. Anzahl der Hidden Layer
2. Anzahl der einkommenden Dendriten eines individuellen Neurons
3. Anzahl der ausgehenden Axone eines individuellen Neurons

Des Weiteren können Neuronalen Netze auf unterschiedliche Arten, abhängig von dem Status Ihrer Verwendung, genutzt werden. Hierzu fokussiert man sich zunächst auf folgende Durchlaufmöglichkeiten:

1. **Forward-Propagation**Innerhalb der Forward-Propagation wird das Neuronale Netz von dem Input bis hin zu dem Output einmalig durchlaufen. Hierbei bleiben werden die Gewichte des Neuronalen Netzes **nicht** verändert.
2. **Backward-Propagation**Bei der Backward-Propagation wird das Neuronale Netz rückwärts ausgehend von dem Output hin zu dem Input durchlaufen. Hierbei werden die Gewichte des Neuronalen Netzes adaptiert.

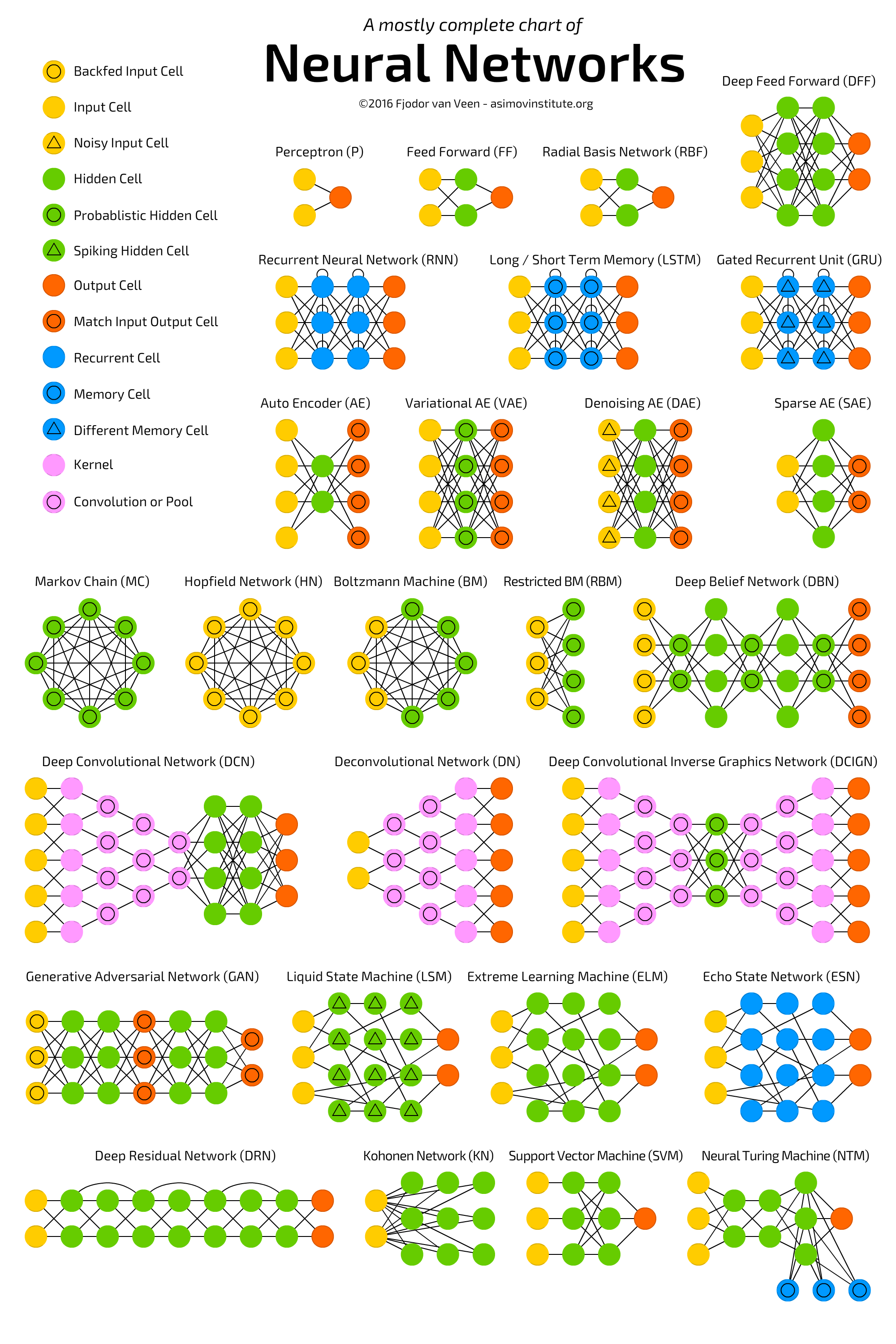
Diese Durchlaufmöglichkeiten kommen unterschiedlich zum Einsatz

1. **Verwendung von Forward- & Backward-Propagation im Training**Das Training eines Neuronalen Netzes ist der Prozess, in welchem die Gewichte des Neuronalen Netzes angepasst werden. Dies geschieht nach dem folgenden Konzept:
   * + 1. Die Gewichte eines neuronalen Netzes werden randomized gesetzt.
       2. Gelabelte Testdaten durchlaufen das Neuronale Netzwerk (Input, Gewicht, gewählte Aktivierungsfunktion).
       3. Der Output Y wird innerhalb der Cost-Function ausgewertet.
       4. Die Cost-Function wird innerhalb der Backward-Propagation (Anpassung der Gewichte der Neuronen) optimiert.

Diesen Prozess wird das „Lernen“ eines Neuronalen Netzes genannt.

1. **Verwendung der Forward-Propagation im trainierten Modell zur Prognose**Wurde ein Neuronales Netz trainiert, so wird dieses in der Regel für die Prognose verwendet. Prognostizieren bedeutet in diesem Fall, dass keine gelabelten Daten vorhanden sind. Ziel ist es mit den Input-Daten den Output vorherzusagen.  
   Technisch ausgedrückt bedeutet dies, dass das trainierte Neuronale Netzwerk einmalig mit der Forward-Propagation durchläuft und eine Prognose für den Output Y liefert.

Arten Neuronaler Netze

Somit kann u.a. auf Basis der unterschiedlichen Layeranzahlen, der unterschiedlichen Neuronenanzahl je Layer und der Forward- & Backward-Propagation unterschiedliche Arten von Neuronalen Netzen entstehen. Einen ersten Überblick liefert folgende Grafik:

## Konzeptionelle Funktionsweisen Neuronaler Netze im Kontext Data-Science

Wie bereits einleitend erwähnt, befassen im Allgemeinen die Ziele von Deep-Learning entweder die Regression, Klassifikation, Clusterings oder Anomalie-Detection. Im Vergleich zu den Machine-Learning Technologien werden im Deep-Learning nun leistungsstärkere Ansätze mittels Neuronaler Netze verwendet. Diese individuell auf den Anwendungsfall angepassten Netzwerke besitzen die Charakteristik, dass sie autark von Datensätzen lernen. Lernen heißt in diesem konkreten Beispiel, dass die Gewichte der einzelnen Neuronen innerhalb des Netzwerkes entsprechend angepasst werden.

Zusammenfassung Hyperparameter eines Neuronalen Netzwerkes

# Convolutional Neuronal Network

Ein CNN ist nun ein Neuronales Netz, welches wiederum nach dem Vorbild des Humanen visuellen Kortext entwickelt wurde. Diese Netzwerke werden vorrangig für die Bearbeitung von Video und Audiodateien verwendet.

Die beschriebenen CNNs welche im folgenden beschrieben werden, dienen der Image-Classification.

Um nun die Funktionsweise eines Convolutional Neuronal Networks zu verstehen reichern wir zunächst das Verständnis von Neuronalen Netze um die Eigenschaften der Convolutional Neuronal Networks an.

## CNN Hauptkomponenten

### In der weiteren praktischen Ausarbeitung dieser Arbeit werden wir mit Bildmaterial aus den bekannten Datensätze MNIST und CIFAR10 arbeiten, daher verwenden wir in der Beschreibung der Funktionsweis von CNNs auch ein Beispiel der Bildanalyse.

Ein CNN ist durch folgend veranschaulichte Komponten geprägt:



Abbildung ,Aufbau CNN, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Auf die entsprechenden Komponenten gehen wir nun im folgenden sequentiell ein.

### Input

Digitale Bilder sind zunächst geprägt durch die Pixelanzahl und das verwendet Farbschema. Durch eine Anordnung dieser Parameter entstehen Formen und Konturen, welche wir als Menschen als digitale Bilder konsumieren können. Technisch können wir uns die Darstellung als eine Matrix aufgespannt aus Pixelhöhe und Pixelbreite vorstellen, welche die jeweiligen Farbwerte beinhaltet.  
Eine Technische Herausforderung besteht nun darin, dass unterschiedlicher Bilder gleicher zugrundeliegender Motive für den Menschen klar unterscheidbar sind, für technische Systeme jedoch „auf den ersten“ Blick gänzlich diverse Objekte vorliegen.

Veranschaulicht an dem einfachen Beispiel der Schwarz-Weiß-Bilder „X“ und „O“ können folgende Änderungen aus technischer Perspektive festgehalten werden:

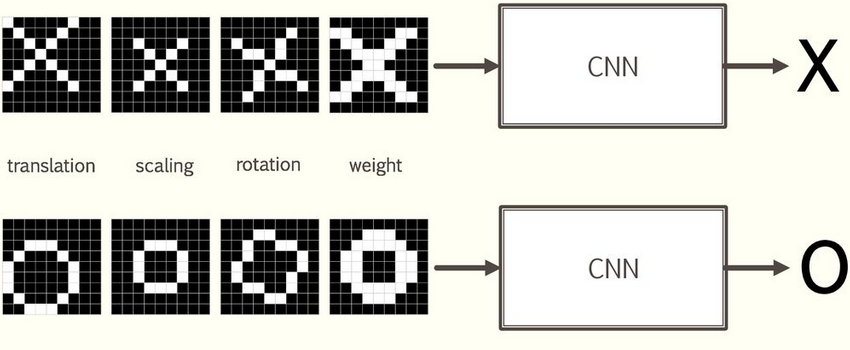


Abbildung ,Image-Unterschiede Schwarz/Weis einfacher Objeke , https://slideplayer.com/slide/14855329/

Um das Verständnis für die Bildbearbeitung und die oben beschrieben Bildunterschiede aus technischer Perspektive zu festigen wurden folgende Jupyter-Notebook ausarbeitungen angefertigt. Die Aufgabenstellungen stammen aus dem Buch „Programmieren-Trainieren“

1. Invert&Switch |Tranlation
2. Shaddow |Weight
3. Sizing | Scaling
4. Rotate | rotation

Ziel eines CNNs ist es nun Bilder gleicher Motive, jedoch auf unterschiedlicher Paramter dennoch mit einem Modell erfassen mit einer akzeptablen Accuracy zu klassifizieren.

In dem bisher Beschriebenen Beispiel haben wir uns lediglich auf „verformung“ von Schwart-Weis-Bilder von vergleichsweise einfachen Bilddateien konzentriert. Jedoch kann dieses Grundprinzip problemlos in komplexere Farbbilder übertragen werden. Mögliche unterschiede von Farbbildern

### Convolutional Layer

Der erste Schritt der Bearbeitung innerhalb eines CNN wird das Bild im Convolutional Layer gefaltet (engl to convolute = falten). Konkret bedeutet dies, dass das Input-Image zunächst nach bestimmten Kriterien (ausgedrückt durch Filter/Kernel/Faltungsmatitzen)durchsucht und die Ergebnisse abgelegt inform von Bildpatches in einem Stack abgelegt werden. Die Anzahl der Bildpatches ist somit gleich der Anzahl der gewählten Filter. Jedes entstehende Imagepatch innerhalb des Stapel spiegelt das Vorkommen und die Lokalisation des jeweiligen Filters in dem ursprünglichen Input-Bild.

Um diese Faltung zu vollziehen wird nach folgendem Schema vorgegangen:

1. Erstellen von Filtern aus Vertikalen und Horizontalen Strukturen. Diese Filter werden automatisch erstellt.
2. Die Filter werden sequentiell auf das ursprüngliche Input Bild angewendet
   1. Jeder Filter gleitet über das Input-Image mit einer definierten Schrittweite = Stride hinweg, diesen Prozess nennt man Padding
   2. Bei jedem „Zwischen-Stopp“ während des Paddings wird der Filter mit dem betrachteten Ausschnitt des Input-Bildes verglichen. Konkret werden die Werte des Filters mit den Werten des Bildausschnittes multipliziert und daraus ein Durchschnitt berechnet
3. Durch jeden Filter innerhalb des Padding entsteht somit eine komprimierte Darstellung des Bildes. Diese entstehenden Bilder sind außerdem durch die Charakteristik des Filters geprägt.
4. Durch das Convolutional Layer entsteht aus dem Input-Bild ein Stapel von Bilder geprägt durch den gewählten Filter

Ziel des Convolutional-Layers ist es die High-Level-Featrue (ausgedrückt durch Ecken&Kanten )innerhalb des Input-Bildes zu detektieren. In dem beschriebenen Beispiel findet hier Zugleich eine Dimensionsreduktion zwischen dem Input-Image und dem Bildpatch im Concolutional-Stack statt. Diese Reduktion hängt von der Dimension und der Art de Implementierung des verwendeten Filters ab.



Abbildung ,Veranschaulichung Convulutional Layer am Beispiel eines Filters, <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

### Rectified -Linear Unit

Zwischen den einzelnen Layer wird häufig eine erste Optimierung in Form von ReLUs verwendet. Ziel der Relu-Funktion ist es alle negativen Werte innerhalb der Bildpatches des vorgelagerten Stacks zu detektieren und diese dem Wert 0 zuzuweisen. Durch dieses Vorgehen wird die „nicht-linearität“ innerhalb der Bilder erhöht. Input Bilder sind grundsätzlich nicht Linear, durch diese Funktion wird dies Nichtlinearität verstärkt. Durch diesen Ausbau wiederum werden die zu extrahierenden Merkmale technisch deutlicherer.

### Pooling Layer

Je größer die Dimension eines Bildes (unabhängig ob Input oder Convoluted Image) desto intensiver ist die benötigte Rechenpower. Das Pooling Layer verfolgt somit nun das Ziel jedes Bild innerhalb des der Convolutional -Image-Stack(Output aus dem Convultional-Layer) zu in den Dimensionen zu reduzieren. Somit ist der Output des Pooling Layers die Dimensionsreduktion der Images im Convolutional Layer, jedoch keine Änderung der Anzahl der Images in dem Stack.

Für diese Dimensionreduktion wird nach folgendem Schema vorgegangen

1. Auswahl der Paramter (Fenstergröße, Schrittweite)
2. Für jedes Bild innerhalb des Convolutioal Stacks
   1. Sliding
      1. Innerhalb jedes Zwischenstops während des Slidings Auswahl des Maximal-Wertes
   2. Abspeichern der entstanden Dimensionsreduktion im Pooling-Stack

### Zwischeneinschub

Die bisher beschriebenen Stationen aus Convolutional-Layer, ReLU und Pooling Layer innerhalb eines CNNs können nun beliebig oft wiederholt werden. Hierbei entsteht nun auch das Grundkonzept des Deep-Leanring und die Möglichen [Architekturen](#_CNN-Architekturen)

Typische Grenzen sind hierbei jedoch die Physische Hardware, da durch jedes Zusätzliche Layer entsprechend mehr Leistung benötigt wird. Die

### Fully Connected Layer

Das Fully Connected Layer greift die Bildpatches des zuletzt vorgelagerten Poolings-Stacks auf und flacht den Stack zu einem Vektor ab. Dieser Vektor wird nun für die Klassifikation in unserer Aufgabenstellung verwendet

## CNN Ablauf

## Rückschluss auf die Umsetzung in Python mit Tensorflow

Hyper-Parameter

Convolutional

Anzahl der Feature

Größe der Feature

Pooling

Fenstergröße

Stride

Fully-Connected

Anzahl der Neuronen

# CNN-Architekturen

# Praktische Implementierung in Python

Um die hier behandelten theoretischen Konzepte zu Neuronalen Netzen und im speziellen zu CNNs zu untermauern wurden die diskutierten Technologien in folgenden Case-Studies verwendet. Diese Ausarbeitungen wurden in Jupyter Notebook ausgeführt, sodass

1. Die Darstellung im online-Git-Repo gewährleistet wurde
2. Die einzelnen Teilaspekte durch Kommentare versehen werden konnten.

## Praktische Implementierung

Im Folgenden finden sich die Querverweise zu den Jupyter-Notebooks, welche im Rahmen der Ausarbeitung dieser Kompensationsarbeit erstellt wurden. Dabei ist anzumerken, dass

Die thematisch passenden Notebooks im Kontext dieser schriftlichen Ausarbeitung sind

1= Mnist-CNN

2 CIFAR10-CNN

## MNIST

Der Datensatz [MNIST](http://yann.lecun.com/exdb/mnist/) ist im Bereich der Bildklassifikation der Computer Vision zu vergleichen mit den „Hello-World“ Konzept der unterschiedlichen Programmierparadigmen. Dieser Datensatz besteht aus 70000 handgeschriebenen Bildern der Ziffern 0-10.

Grundsätzlich behandelt diese Arbeit die Wirkungsweise von CNNs. Dennoch wurden Algorithmen außerhalb der NN-Technologien ebenfalls implementiert und ausgearbeitet. Dieses Vorgehen ermöglicht einerseits eine sinnvolle Hinführung zu dem Thema „CNN“ als auch ein Benchmark der Technologie.

Ziel der Aufgabenstellung ist es einen Classifier zu finden, welcher je nach verwendetem Algorithmus die besten Werte in der Accuracy, der Confusion Matrix und weiterführend der Precision&Reccall&F1-Score, ROC&AUC bzw. LOG-LOSS erbringt.

Hierzu wurden folgende Jupyter-Notebook Aufbereitungen erstellt:

|  |  |
| --- | --- |
| **Titel** | **Beschreibung** |
| [0\_Basics.ipynb](file:///C:\Users\silvi\AppData\Local\Temp\Projekte\MNIST\0_Basics.ipynb) | Dieses Notebook dient 1) Der Datenbeschaffung   1. Download 2. Entpacken   2) Der ersten Datensichtung   1. Prüfung der Bildformate 2. Erste Visuelle Veranschaulichung |
| 1\_Binary\_Classifier.jpynb | 1. Erstellen binärer klassifizier 2. Modifizieren der Train-Data für BC 3. Stochastic-Gradient-Descent 4. Logistische Regression 5. Random-Forrest-Classifier 6. Modelle werden im Directory „Model“ gespeichert |
| 1b\_Binary\_Classifier\_Evaluation.jpynb | 1. Modelle werden aus Directory „Model“ geladen 2. Evaluierungen 3. Accuracy 4. CM 5. Precision & Recall 6. F1-Score 7. ROC-Kurve /AUC |
| 2\_Multiclass\_Classfier | 1. Erstellen Multi-Class Klassifizierer 2. Stochastic Gradient Descent 3. OVO 4. OVA 5. Logistische Regression 6. RF-Classifier |
| 2b\_Binary\_Classifier\_Evaluation.jpynb |  |
| 3\_CNN | Aufbau und Vorverarbeitung eines CNNs |
|  |  |

## CIFAR10

Als weiterführende Übung wurde die Klassifikation des CIFAR10, Datensatzes verwendet. Als Leitlinie diente hierzu die initiale [Aufgabenstellung](file:///C:\Users\1810837475\Desktop\Operativ\CNN\Projekte\1_CIFAR10\WH1.pdf).

Unterschiede zu MNIST:

1. Unterschiedliche Daten einlesen
2. Komplexer in der Datenvorverarbeitung

Hierzu wurden folgende Jupyter-Notebook Aufbereitungen erstellt:

|  |  |
| --- | --- |
| **Titel** | **Beschreibung** |
| [0\_Basics.ipynb](file:///C:\Users\silvi\AppData\Local\Temp\Projekte\MNIST\0_Basics.ipynb) | Dieses Notebook dient 1) Der Datenbeschaffung   1. Download 2. Entpacken   2) Der ersten Datensichtung   1. Prüfung der Bildformate 2. Erste Visuelle Veranschaulichung |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Abbildungsverzeichnis

[Abbildung 1: Entwicklung AI, ML, DL (https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/) 4](#_Toc27570636)

[Abbildung 22, Neuron-Aufbau (https://medicalxpress.com/news/2018-07-neuron-axons-spindly-theyre-optimizing.html) 7](#_Toc27570637)

[Abbildung 3, Perceptron Aufbau (Foliensatz Machine-Learning- Miroslav / Teil4 Deep-Learning) 8](#_Toc27570638)

# Literaturverzeichnis

Learning, K. N. (2018). Künstliche Neuronale Netzwerke und Deep Learning. *Wirtschaftswissenschaften HTW SAAR*.