KOMPENSATIONSARBEIT IM MODUL SOFTWARE ENTWICKLUNG

MASTER:

"DATA SCIENCE & INTELLIGENT ANALYTICS"

Betreuer :

Huber Stephan

Autor:

Jochen Hollich | 1810837475

Datum

10.12.2019

Inhalt

Präambel

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

Einleitung

Viele Technologien der Menschheit finden Ihren Ursprung in der Beobachtung natürlicher Phänome. So inspirierten uns Vögel hinsichtlich der Luftfahrt, die Lotuspflanze war das Vorbild für Ingenieure bei der Entwicklung von Autolacken und die Fähigkeit des Gekos kopfüber entlang zu spazieren stand Pate für ultrastarke Haftfolien. Die gegenwärtige Phase wird auch als das digitale Zeitalter bezeichnet. „*Das „Digitale Zeitalter" zeichnet sich durch mehrere zentrale Eigenschaften aus, die mit den Begriffen Digitalisierung, Vernetzung, Mobilität und Miniaturisierung erklärt werden können"(* https://www.teamnext.de/academy/artikel/das-digitale-zeitalter/*).* Mit den Konzepten der Bionik und den Treibern unserer Zeit erscheint es somit 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 heruntergeladen werden.

Abgrenzung der Begrifflichkeiten

Bevor in der folgenden Ausarbeitung auf die nähere Erläuterung von biologische Nervensysteme, technische Neuronalen Netzwerken (NN) und anschließend auf die CNNs eingegangen wird, werden noch etwaige Begrifflichkeiten geklärt.

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

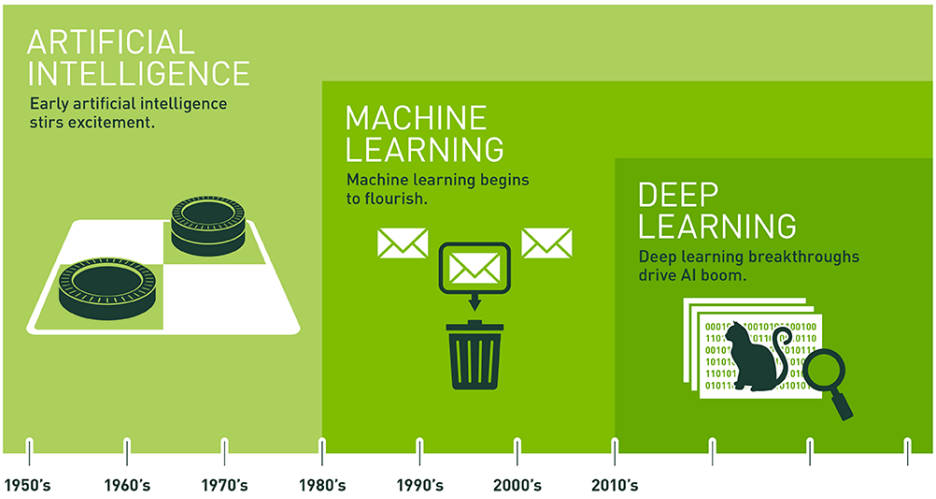


Abbildung 1: Entwicklung AI, ML, DL (https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/)

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 erfolgreiche 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-forcte". Dieser Ansatz war jedoch durch die höhere Komplexität des Spieles Go nicht übertragbar. Die „Intelligenz" von Alpha Go liegt in der Art wie das Spiel gelernt wurde. Hierbei wurde ein Reinformen-Algorithmus implementiert, in welchem Alpha-Go lernte Go effizient zu spielen.

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 Datenanalyse (noch) eingreifen.

Deep-Learning

DL ist die ein Teilbereich 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 die eigenen Parameter(Gewichte) 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 Neuronaler Netze

Neuronale Netze – sowohl bei biologischer als auch bei technischer Betrachtungsweise – verfolgen das Ziel zu lernen, um auf Basis des Erlernten Prognosen geben zu können. 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 biologische 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 Betrachtung in dieser Arbeit die kleinsten individuellen Einzelsysteme für diese Signal-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:
   a. 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 eingehendes Dendrit. Die eintreffenden Signale können reizend oder hemmend sein.
   b. 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. Die Ausgabe eines Neurons hängt somit einerseits von den eintreffenden Signalen und der Verarbeitung der jeweiligen Nervenzelle ab
   c. 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

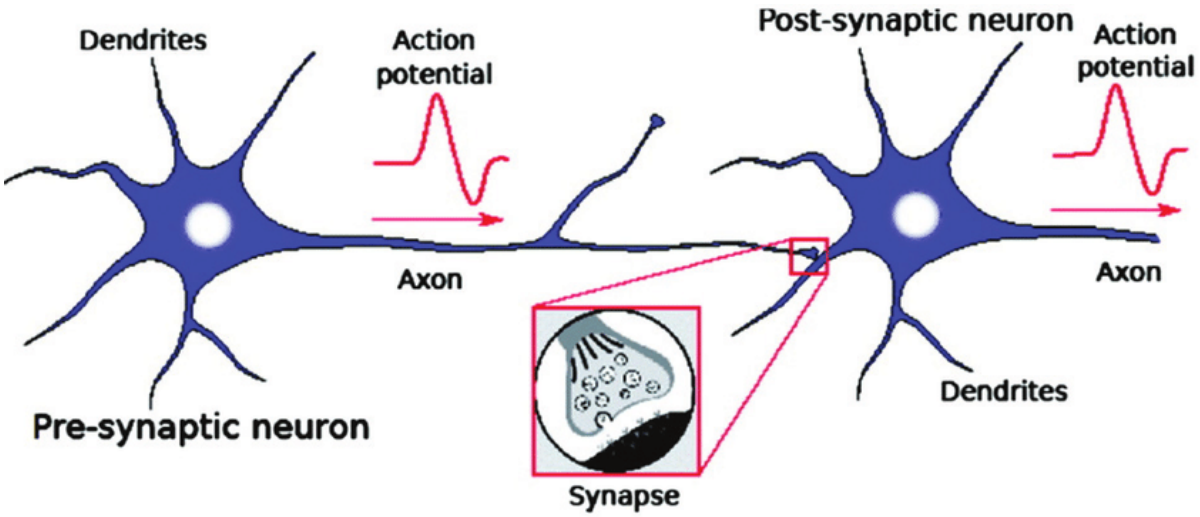


Abbildung 22, Neuron-Aufbau (https://medicalxpress.com/news/2018-07-neuron-axons-spindly-theyre-optimizing.html)

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 10^{10} bis 10^{12} miteinander vernetzten Nervenzellen, den Neuronen“. ()*

Lernen biologischer Nervensysteme

„Synapsen übertragen nicht nur elektrische Signale von einer Nervenzelle zur nächsten, sie können die Intensität des Signals auch verstärken oder abschwächen. Diese sogenannte synaptische Plastizität ist die Grundlage von Lernen und Gedächtnis“ (https://www.max-wissen.de/132855/Synapsen). Konkret bedeutet dies, wenn ein biologisches Individuum einen Vorgang (gleich ob kognitiver oder motorischer Art)übt, werden währenddessen die entsprechenden Synapsen und somit der nachgelagerte Zellkern häufiger getriggert. Durch diese frequente Reizung lassen diese Neuronen, durch das Üben und somit häufigere Nutzung der entsprechenden Synapse und Nervenzelle, ein Signal schneller durch das Neuron „passieren“. Diesen Phänomen nennt man Langzeitpotenzierung. Lernen findet somit in den Synapsen und dem Zellkern statt und bewirkt das Impulse effizienter von einem Neuron zum nächsten übertragen werden.

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 hier nachfolgend beschriebenen Fall das Perceptron besteht ausschließlich aus einem einzelnen Neuron. Die Bestandteile eines technischen Neurons und des Perceptron lauten wiefolgt:

a) Input-Layer
Dies ist der Startpunkt eines Neuronalen Netzes. In diesem Punkt werden die ggf. bereits vorverarbeiteten Daten in das NN „gespeist“. Neuronale Netze erwarten meist eine Anordnung der eingehenden Elemente ausgedrückt durch einen Mathematischen Vektor oder eine Matrize.

b) 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.

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

d) Zellkörper | Actiavation-Function f
In der Aktivierungsfunktion fließen alle einkommenden Produkte (Input*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“) wird ein Signal weitergeben oder nicht. Je nachdem wie stark ein Neuron aktiviert wurde gibt es ein entsprechend hohes oder niedriges Output-Signal weiter
Diese möglichen Funktionen

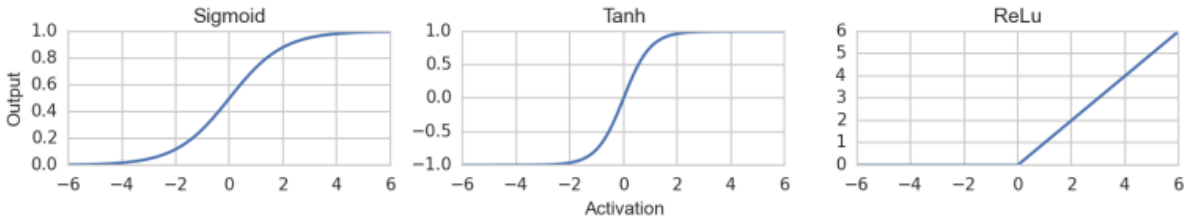


Abbildung 3,Darstellung gängiger Aktivierungsfunktionen,https://www.cbcity.de/tutorial-neuronale-netze-einfach-erklaert

e) Output Axon y
Ob und wie hoch ein Neuron ein Signal weitergibt(feuert) hängt von der Höhe des Inputs in Zusammenhang mit der Activation-Function ab. Die Weitergabe erfolgt dann auf dem Output-Axon.

f) Hidden Layer | Zwichenschicht
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. Dabei hat jedes Hidden-Layer während des Lifecycle des Modelles eine konstante Anzahl an Neuronen parallel geschaltet. Innerhalb eines Modelle mit mehrere Hidden Layer können die einzelnen Hidden-Layer eine unterschiedliche Anzahl an Neuronen besitzen. „Wenn jedes Neuron mit jedem Neuron des nächsten Layers verbunden ist, dann spricht man von ‘Fully Connected Layer’ (auch ‘Dense-Layer’). Es gibt dann so viele Gewichte w, wie es Verbindungen gibt.“(cbcity.de/tutorial-neuronale-netze-einfach-erklaert)

g) 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/). Ob lediglich eine einzelner Outputvalue oder mehrere mögliche Outputvalues ausgegeben werden, hängt von der Art der Implementierung bzw. des Zugrundeliegenden Use-Cases ab. „Wenn jedes Neuron mit jedem Neuron des nächsten Layers verbunden ist, dann spricht man von ‘Fully Connected Layer’ (auch ‘Dense-Layer’). Es gibt dann so viele

Gewichte w, wie es Verbindungen gibt.“(cbcity.de/tutorial-neuronale-netze-einfach-erklaert)

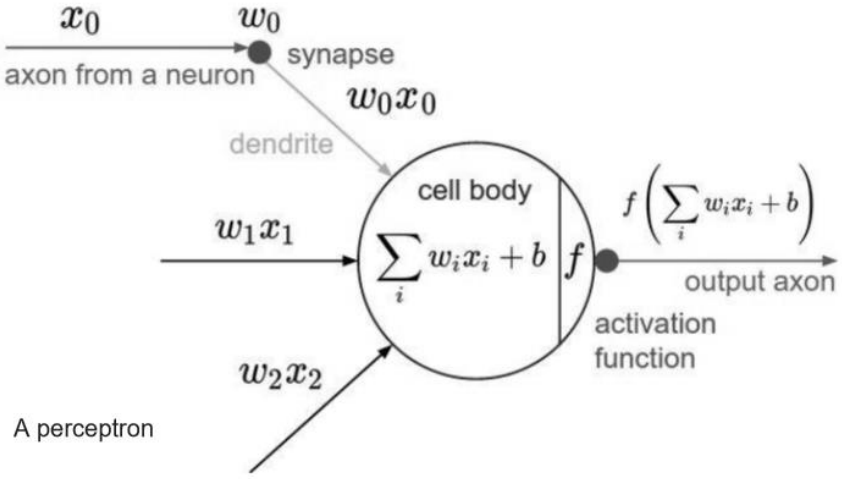
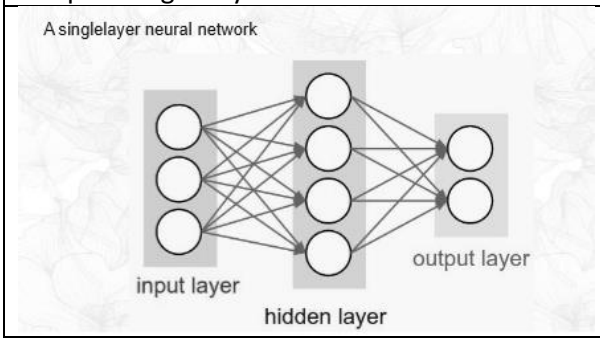
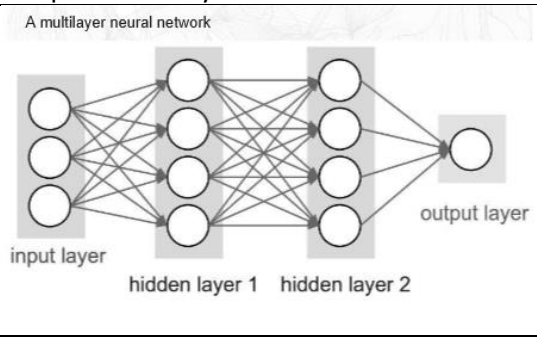


Abbildung 4, Perceptron Aufbau (Foliensatz Machine-Learning- Miroslav / Teil4 Deep-Learning)

Technische Neuronale Netze | Architekturen

Ähnlich den biologischen Neuronen können technische Neuronen ebenfalls parallel und sequentiell aufgespannt werden, dabei entstehen Neuronale Netzwerke. 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:

a) Jedes Neuronale Netz hat ein Input-Layer. Das kann beispielsweise ein Vektor, ursprünglich kommend von einem digitalen Bild, sein.(Beispile MLP | ! kein Beispiel CNN)
b) Architektur
    a. Einerseits besteht die Architektur aus der Anzahl der Hidden-Layer (Hidden Layer = weder Input noch Output-Layer).
    b. Jeder Layer wiederum besteht aus einer Anzahl von Neuronen.
c) Jedes Neuronale Netz hat einen Output-Layer.
    a. Dies kann ein Single-Output (bspw. Implementierung einer binären Klassifikation) …
    b. …oder ein Multioutput (bspw. Implementierung einer Multi-Output-Klassifikation/ MultiClass-Klassifikation) sein.

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

a) Anzahl der Hidden Layer
b) Anzahl der einkommenden Dendriten eines individuellen Neurons
c) 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:

a) **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.
b) **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. Dies ist ein zentrales Element des Lernens.

Diese Durchlaufmöglichkeiten kommen unterschiedlich zum Einsatz

a) **<u>Verwendung von Forward- & Backward-Propagation im Training</u>**
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. Diesen Prozess nennt man auch Initialisierung des Netzwerkes.
2) Gelabelte Trainingsdaten durchlaufen das Neuronale Netzwerk (Zentrale Parameter Input, Gewicht, gewählte Aktivierungsfunktion).
3) Der Output Y wird innerhalb der Cost/Loss-Function ausgewertet. Konkret heist das man vergleicht den prediktierten Wert mit den tatsächlich bekannten Wert der gelabelten Trainingsdaten Daten.
4) Die Cost-Function wird mittels der Backward-Propagation (Anpassung der Gewichte der Neuronen) optimiert. (z.B.: SGD, GD]). Konkret bedeutet dies, dass auf Basis der Ergebnisse der Costfunction durch das NN mittels der Backpropagation geprüft wird, welche Neuronen, für den Fehler verantwortlich waren. Diese Neuronen werden anschließend adaptiert. „Diese Suche nach der idealen Aktivierung bedeutet mathematisch das Finden von Minimalwerten im Fehlerraum“. Folgende Grafik veranschaulicht unterschiedliche Optimierungsstrategien grafisch:

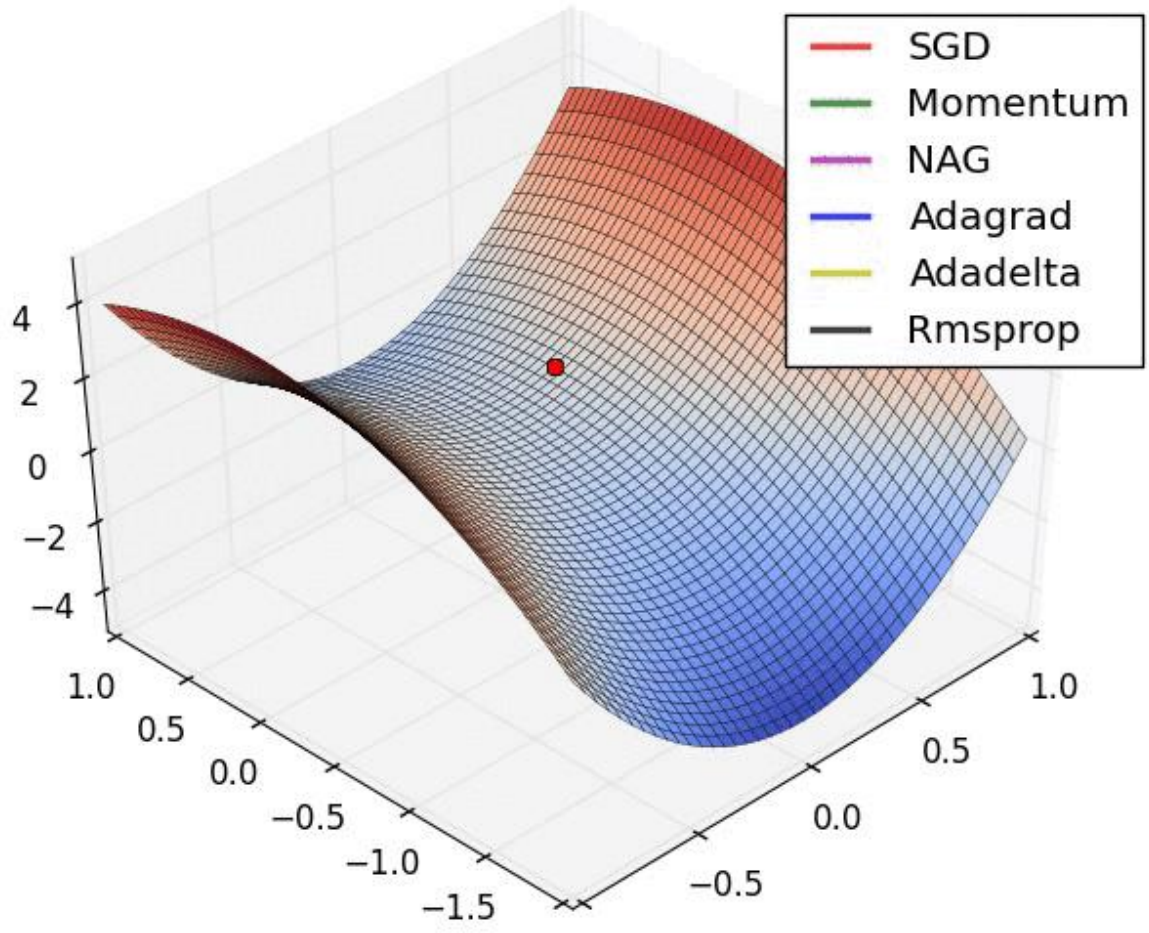


Abbildung 5,Mögliche Optimierungen der Fehlerfunktion,https://www.cbcity.de/tutorial-neuronale-netze-einfach-erklaert

5) Das angepasste Neuronale Netz wird erneut mit den angepassten Gewichten mit der nächsten Instanz der Trainingsdaten durchlaufen und erneut angepasst.

Dieser Prozess der Anpassung der Gewichte wird das „Lernen“ eines Neuronalen Netzes genannt.

b) **<u>Verwendung der Forward-Propagation im trainierten Modell zur Prognose</u>**

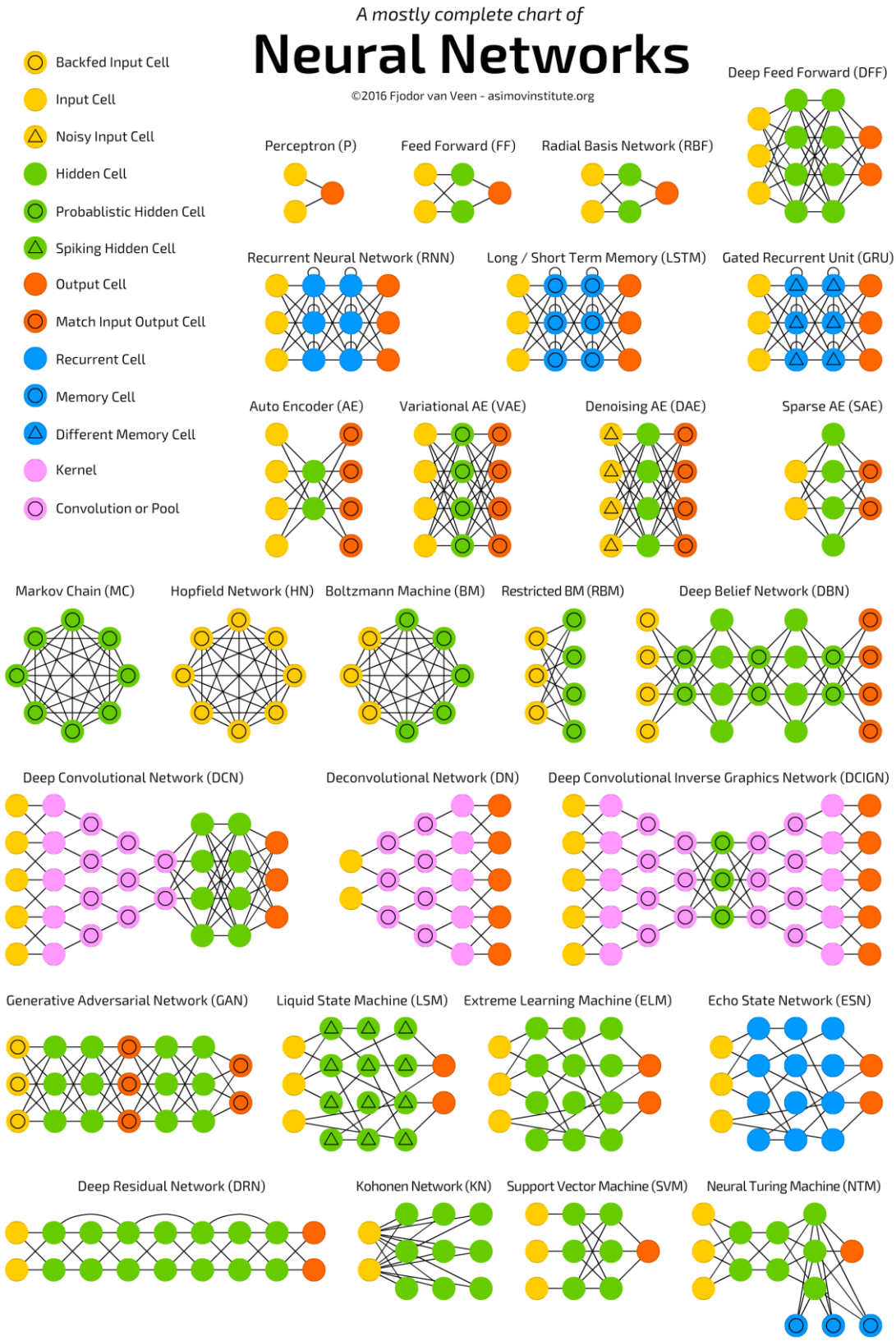
Wurde ein Neuronales Netz trainiert, so wird dieses in der Regel für die Prognose verwendet. Prognostizieren bedeutet in diesem Fall, dass keine gelabelten Daten vorhanden sind. Ziel ist es mit den Input-Daten den Output vorherzusagen.

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

Wie nun aufgezeigt wurde, unterscheiden sich Neuronale Netze in der Tiefe(Anzahl der Hidden Layer) in der Höhe (Anzahl der Neuronen in einem Layer).

Je nach Anordnung entstehen somit unterschiedliche Architekturen, welche wiederum je nach Anwendungszweck verwendet werden können. Ein Auflistung und Veranschaulichung geläufiger Architekturen findet sich in nachfolgender Grafik:



Konzeptionelle Funktionsweisen Neuronaler Netze im Kontext Data-Science

Wie bereits einleitend erwähnt, sind 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.

Beispiel Multi-Layer-Perceptron in Keras

Möchte man nun ein Neuronales Netz technisch operativ umsetzten so kann grundsätzlich nach „Schema-F“ vorgegangen werden. Für die Implementierung der Konezepte gibt es Zahlreiche Bibliotheken, welche die Ausarbeitung unterstützen.

Grundsätzlich gilt die Vorgehensweise

a) Daten vorverarbeiten, sodass diese ins NN übergeben werden können
b) Daten in Batches aufteilen
c) NN definieren
d) NN ausführen
e) Results bewerten
f)

Hyperparameter in NN

Paramter	**Beschreibung**	**BSP Paramter in Tensorflow**	**Mögliche Parameter**
Batch Size	In ML / DL Projekten haben wir häufig mit Datenmengen zu tun, welche so groß sind, dass diese nicht zeitgleich verarbeitet werden können. Aus diesem Grund teilt man die gesamten Daten in Batches der größe „batch Size“ auf. Jeder Batch wird letztlich sequentiell isoliert am NN trainiert.	Batch_size = 128	
Iteration	Teilt man die Anzahl der Traininsinstanzen erhält man die Batch Anzahl. Nachdem jeder Batch einmal das gesamte Netz durchlaufen muss bevor der gesamte Datensatz verarbeitet wurde, ist somit die Anzahl der Iterations gleich der Anzahl der Batches	Die Iteration wird indirekt über die Batch Size und die „Anzahl“ der Trainingsinstanzen bestimmt	
Epoche	Durchläuft ein Datensatz(oder einmal alle Batches) das Neuronale Netz, so ist dies eine Epoche. Währens des Trainings eines NN werden im Grunde genommen übe forward&Backpropagation die Gewichte des Netzes angepasst. Somit gibt es einen Mehrwert wenn der gesammte Datensatz „mehrfach“ (= mehrere Epochen) durch das Netz fließt und die Gewichte anpasst	epochs = 20	
Layer / Dense	Wie bereits erwähnt unterscheiden sich die Architektur Neuronaler Netze durch die unterschiedlichen Anzahl von	model.add(Dense(512, activation='relu', input_shape=(784,)))	Hier wird ein Layer mit 512 Neuronen zu dem Modell hinzugefügt, und anschließend die Aktivierungsfunktion aufgerufen.

	Layern, der Anzahl der Neuronen je Layer und die Aktivierungsfunktion der Neuronen		Relu(Alternativ übliche Aktivierungen = Hyperbolic tangent, sigmoid)
Dropout rate	Anzahl der Neuronen in jedem Layer welche während des Trainings(Konkret während eines Durchlaufs, welche neuronen konkret nicht berücksichtigt werden ist Zufall) nicht aktiv sind=> dient dem Entgegenwirken des Overfittings, nach Ablauf Training wird die Dropout-Rate deaktiviert	model.add(Dropout(0.2))	
compile	Mit diesem Befehl wird das bisher beschrieben NN initialisiert. Für die Initialisierung muss die loss function definiert werden, der Optimizer und die Evaluierungsmetric	model.compile(loss='categorical_crossentropy', optimizer=RMSprop(), metrics=['accuracy'])	**Loss Function** Die Lossfunction wird je nach Business-Case geändert. In Klassifikationsfragestellungen sind mögliche Funktionen die Categorial corssentropy oder die Log-Losst. Bei Regressionsfragestellung wird häufig die L1 oder L2 Norm verwendet. Mögliche Lossfunctions innerhalb von Keras befinden sich unter folgendem Link **Optimizer** Hier wird der Optimizer gewählt, nach welchem die Loss-Function optimiert werden soll **Metrics** Die Metric ist grundsätzlich ähnlich der Lossfuntion. Diese wird jedoch nicht während des Trainings sondern im späteren produktiven Einsatz angewendet(Nach der Verwendung der Trainingsdaten, werden die Testdaten für die Ermittlung der Accuracy verwendet)
Fit	Hier wird das Modell entsprechend ausgeführt	history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,	

		verbose=1, # show training progress during training Possible = 0,1,2
		validation_data=(x_test, y_test))	

Convolutional Neuronal Network

Ein CNN ist ein Neuronales Netz, welches nach dem Vorbild des Humanen visuellen Kortex entwickelt wurde. Das ursprüngliche Ziel von CNNs war die Entwicklung der Computer Vision. Diese Netzwerke werden vorrangig für die Bearbeitung von Video und Audiodateien verwendet.

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

Um die Funktionsweise eines Convolutional Neuronal Networks zu verstehen wird zunächst das Verständnis von Neuronalen Netzen, um die Eigenschaften, Layer und deren Funktionsweisen der CNNs angereichert.

CNN Hauptkomponenten

In der weiteren praktischen Ausarbeitung dieser Arbeit wird mit Bildmaterial aus den bekannten Datensätze MNIST und CIFAR10 gearbeitet. Daher wird in der Beschreibung der Funktionsweise von CNNs auch ein Beispiel der Bildanalyse verwendet.
Ein CNN ist durch folgende veranschaulichte Komponenten geprägt:

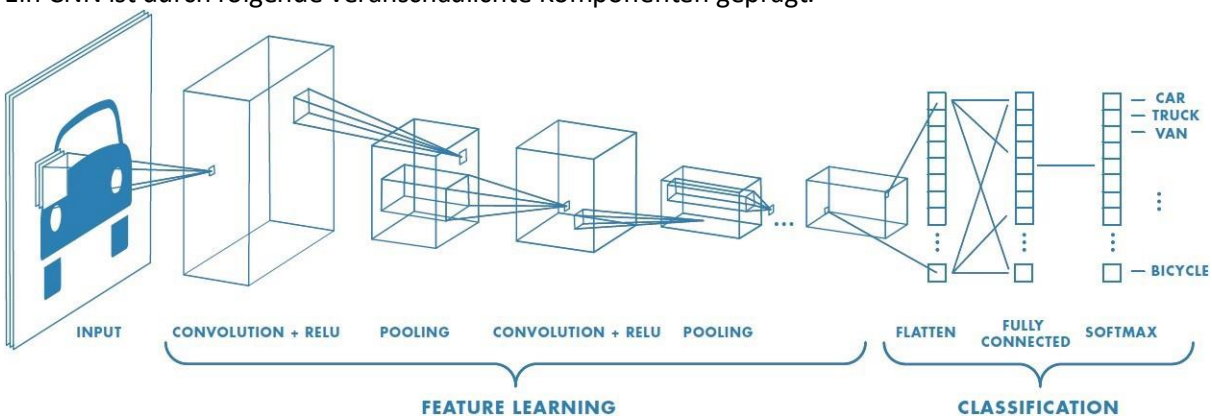


Abbildung 6,Aufbau CNN, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Auf die entsprechenden Komponenten wird nachfolgend sequentiell eingegangen.

Input

Ein markanter Unterschied zwischen „einfachen“ Neuronalen Netzen wie beispielsweise einem Multi-Layer-Perceptron und CNNs ist die Verarbeitung der Input-Daten. Ein Neuronales Netz erwartet einen Input-Vector, wohingehend ein CNN mit Matrizen als Input-Data beginnt.

Digitale Bilder sind zunächst durch die Pixelanzahl (Pixelhöhe x Pixelbreite) und das verwendete Farbschema (Schwarz / weiß | Binär | 2d oder Farbbilder | 3d) geprägt. Durch eine Anordnung dieser Parameter entstehen Formen und Konturen, welche von Menschen als digitale Bilder wahrgenommen werden und in Form von Foto- oder Video Material konsumiert werden können.
Technisch kann die Darstellung als eine Matrix, aufgespannt aus Pixelhöhe und Pixelbreite, vorgestellt werden, welche die jeweiligen Farbwerte beinhaltet.
Eine technische Herausforderung der Computer Vision besteht darin, dass unterschiedlicher Bilder gleicher zugrundeliegender Motive (Beispielsweise Bilderserie eines Autos aus unterschiedlichen Blickwinkeln) für den Menschen klar als dasselbe Objekt identifizierbar 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:

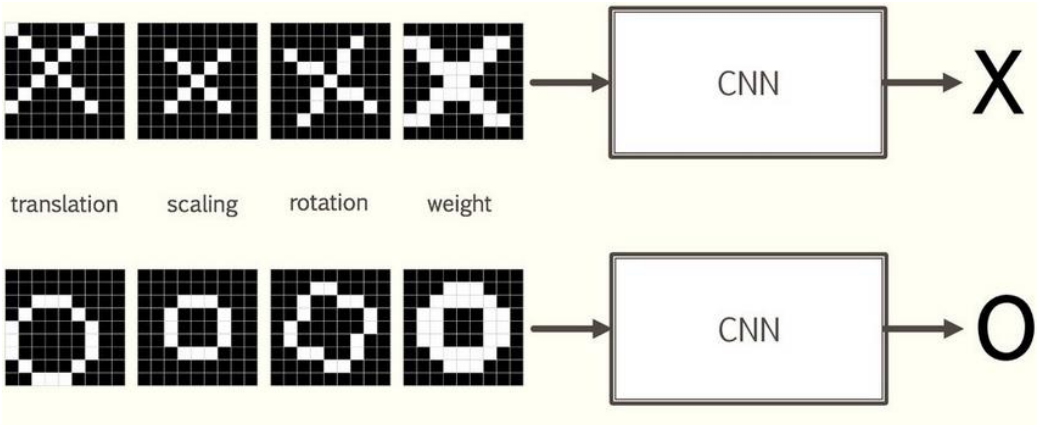


Abbildung 7,Image-Unterschiede Schwarz/Weis einfacher Objekte, https://slideplayer.com/slide/14855329/

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“ von Luigi Lo Iacono, Stephan Wiefling und Michael Schneider

1) Invert & Switch |Tranlation
2) Shaddow |Weight
3) Sizing | Scaling
4) Rotation | Rotation

Ziel eines CNNs ist es, Bilder gleicher Motive jedoch auf unterschiedlichen Parametern dennoch mit einem Modell zu erfassen und mit einer akzeptablen Accuracy zu klassifizieren.
In dem bisher beschriebenen Beispiel wurde lediglich auf Verformung“ von Schwarz-Weiß-Bildern von vergleichsweise einfachen Bilddateien konzentriert. Jedoch kann dieses Grundprinzip problemlos in komplexere Farbbilder übertragen werden.
Mögliche Unterschiede von Farbbildern

Convolutional Layer

Im ersten 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 / Faltungsmatritzen) durchsucht und die Ergebnisse in Form von Bildpatches in einem Stack abgelegt werden. Die Anzahl der Bildpatches ist somit gleich der Anzahl der gewählten Filter. Jedes entstehende Imagepatch innerhalb des Stapels spiegelt das Vorkommen und die Lokalisation des jeweiligen Filters in dem ursprünglichen Input-Bild.

Um diese Faltung zu vollziehen wird nach dem folgenden Schema vorgegangen:

1) Erstellen von Filtern aus vertikalen und horizontalen Strukturen. Diese Filter werden von dem CNN automatisch erstellt. Jeder einzelne Filter ist für sich gesehen statisch und ändert somit seine Werte nicht.
2) Die Filter werden sequentiell auf das ursprüngliche Input Bild angewendet.
    a. Jeder Filter gleitet über das Input-Image mit einer definierten Schrittweite (Stride) hinweg, diesen Prozess nennt man Padding.
    b. 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 Input-Bildes unter der „Blickwinkel“ des entsprechenden Filters. Dabei wird nicht mehr jeder

einzelne Pixel des Input Images, sondern je nach Filtergröße eine Gruppe von Pixeln innerhalb des Input-Bildes im Bildpatch repräsentiert.

4) Durch das Convolutional Layer entsteht aus dem Input-Bild ein Stapel von Bildern geprägt durch den gewählten Filter.

Ziel des Convolutional-Layers ist es die High-Level-Feature (ausgedrückt durch ortsunabhängige Ecken, Kanten, Linien und Farbtupfer) innerhalb des Input-Bildes in den entstehenden Bildpatches unabhängig der Lokalisation im Originalbild zu detektieren. In dem beschriebenen Beispiel findet zugleich eine Dimensionsreduktion zwischen dem Input-Image und dem Bildpatch im Convolutional-Stack statt. Diese Reduktion hängt von der Dimension und der Art der Implementierung des verwendeten Filters ab.

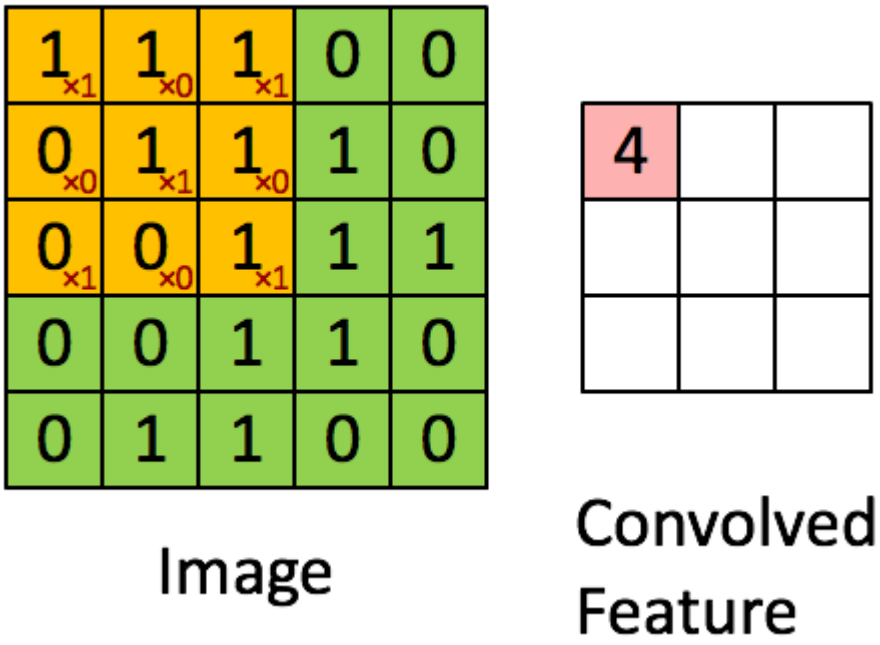


Abbildung 8,Veranschaulichung Convolutional Layer am Beispiel eines Filters, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Rectified -Linear Unit

Zwischen den einzelnen Layern 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 diese Nicht-Linearität jedoch zusätzlich verstärkt. Mit diesen Ausbau wiederum werden die zu extrahierenden Merkmale technisch deutlicherer.
Die ReLU-Funktion kann sowohl nach dem Convolutional als auch nach dem Pooling Layer stattfinden.

Pooling Layer

Je größer die Dimension eines Bildes (unabhängig ob Input oder Convoluted Image) desto intensiver ist die benötigte Rechenpower. Das Pooling Layer verfolgt das Ziel jedes Bild innerhalb des Convolutional-Image-Stack (Output aus dem Convolutional-Layer) in den Dimensionen zu reduzieren, die Ergebnisse somit zu aggregieren. Folglich ist der Output des Pooling Layers die

Dimensionsreduktion der Images im Convolutional-Stack, jedoch keine Änderung der Anzahl der Images in dem Stack.

Für diese Dimensionreduktion wird nach dem folgenden Schema vorgegangen:

1) Auswahl der Parameter (Fenstergröße, Schrittweite)
2) Für jedes Bild innerhalb des Convolutional Stacks
    a. Sliding
        i. Innerhalb jedes Zwischenstopps während des Slidings Auswahl des Maximal-Wertes
    b. Abspeichern der entstanden Dimensionsreduktion im Pooling-Stack

Zwischeneinschub

Die bisher beschriebenen Stationen aus Convolutional-Layer, ReLU und Pooling Layer innerhalb eines CNNs können beliebig oft wiederholt werden. Hierbei entsteht auch das Grundkonzept des Deep-Learning und die möglichen Architekturen

Typische Grenzen sind hierbei jedoch die physische Hardware, da durch jeden zusätzlichen Layer entsprechend mehr Leistung benötigt wird.

Fully Connected Layer

Der Fully Connected Layer greift die Bildpatches des zuletzt vorgelagerten Pooling-Stacks auf und flacht den Stack zu einem Vektor hin ab, diesen Prozessen nennt man auch „ausrollen“ oder „flatten“.Somit sind „*Die Output-Signale der Filter-Schichten [] unabhängig von der Position eines Objektes, daher sind zwar keine Positionsmerkmale mehr vorhanden, dafür aber ortsunabhängige Objektinformationen*“ (https://jaai.de/convolutional-neural-networks-cnn-aufbau-funktion-und-anwendungsgebiete-1691/). Mit diesen Eigenschaften sind die Probleme des MLP aufgehoben.

Ab dem Zeitpunkt des Vektor-Flattens werden die Technologien des bereits beschriebenen Neuronalen Netzes verwendet, um im vorliegenden Falle eine Klassifikation zu erstellen. Somit bildet der abgeflachte Vektor den Input eines Neuronalen Netzes, welches in unserem Falle zuletzt durch die Output-Neuronen klassifiziert. In praktischen Beispielen von MNIST und CIFAR10 sind die möglichen Output-Klassen bekannt und entsprechen in beiden Fällen jeweils einem Wert von 10. Somit wird die Anzahl der Output-Neuronen 10 sein.

Softmax Im Fully-Connected Layer

Die Softmax-Funktion ist Teil der Softmax-Regression (oder Multinomial Logistic Regression) die wiederum eine Verallgemeinerung der logistischen Regression ist. Sie wird häufig bei MultiClass-Klassifizierungsproblemen verwendet. Im Kontext der verwendeten Beschreibung bedeutet dies, dass der Output der letzten Neuronen kumuliert 1 beträgt.

Rückschluss auf die Umsetzung in Python mit Keras

Die Softmax-Funktion ist Teil der Softmax-Regression (oder Multinomial Logistic Regression) die wiederum eine Verallgemeinerung der logistischen Regression ist. Sie wird häufig bei MultiClass-Klassifizierungsproblemen verwendet. Im Kontext der verwendeten Beschreibung bedeutet dies, dass wenn die Anzahl(N) der möglichen Output-Klassen bekannt ist, der Outputvektor der Softmax-Funktion N-Dimensionen hat. Kummuliert man die WertendernOutput-Vektoren auf, so erhält man den Wert 1.

Python Implementierung Hyperparameter

Anzahl der Feature

Größe der Feature

Pooling

Fenstergröße

Stride

Fully-Connected

Anzahl der Neuronen

CNN-Architekturen

Weitere Anwendungsfelder von CNNs

In dieser Beschreibung wird überwiegend auf die Verwendung von CNNs im Kontext der Bildverarbeitung, der Computer Vision, eingegangen. Für diesen Anwendungsfall wurden CNNs auch initial hin entwickelt. Jedoch erweisen sich CNNs auch in anderen Bereichen wie beispielsweise dem Textmining, Zeitreihenanalysen usw. ebenfalls als

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

a) Die Darstellung im Online-Git-Repo gewährleistet wurde,
b) 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

1= Mnist-CNN

2 CIFAR10-CNN sind.

MNIST

Der Datensatz MNIST ist im Bereich der Bildklassifikation der Computer Vision mit dem „Hello-World“ Konzept der unterschiedlichen Programmierparadigmen zu vergleichen. 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	Dieses Notebook dient 1) Der Datenbeschaffung a) Download b) Entpacken 2) Der ersten Datensichtung a) Prüfung der Bildformate b) Erste Visuelle Veranschaulichung
1_Binary_Classifier.jpynb	1) Erstellen binärer klassifizier a) Modifizieren der Train-Data für BC b) Stochastic-Gradient-Descent c) Logistische Regression d) Random-Forrest-Classifier 2) Modelle werden im Directory „Model“ gespeichert
1b_Binary_Classifier_Evaluation.jpynb	1) Modelle werden aus Directory „Model“ geladen 2) Evaluierungen a) Accuracy b) CM i) Precision & Recall ii) F1-Score c) ROC-Kurve /AUC
2_Multiclass_Classfier	1) Erstellen Multi-Class Klassifizierer a) Stochastic Gradient Descent i) OVO ii) OVA b) Logistische Regression c) RF-Classifier
2b_Binary_Classifier_Evaluation.jpynb	2)
3_CNN	Aufbau und Vorverarbeitung eines CNNs
	3)

CIFAR10

Als weiterführende Übung wurde die Klassifikation des CIFAR10, Datensatzes verwendet. Als Leitlinie diente hierzu die initiale Aufgabenstellung.

Unterschiede zu MNIST:

1) Unterschiedliche Daten einlesbar

2) Komplexer in der Datenvorverarbeitung

Hierzu wurden folgende Jupyter-Notebook Aufbereitungen erstellt:

Titel	Beschreibung
0_Basics.ipynb	Dieses Notebook dient 1) Der Datenbeschaffung c) Download d) Entpacken 2) Der ersten Datensichtung c) Prüfung der Bildformate d) Erste Visuelle Veranschaulichung

Abbildungsverzeichnis

Literaturverzeichnis

Becker, R. (06. 02 2019). Convolutioal Neuronal Networks. Abgerufen am 19. 12 2019 von https://jaai.de/kuenstliche-neuronale-netze-aufbau-funktion-291/

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

MOESER, J. (kein Datum). KÜNSTLICHE NEURONALE NETZE – AUFBAU & FUNKTIONSWEISE. Von https://jaai.de/kuenstliche-neuronale-netze-aufbau-funktion-291/ abgerufen

Weiter noch einzuarbeitende Quellen

https://www.cbcity.de/tutorial-neuronale-netze-einfach-erklaert