Machine Learning und tiefe neuronale Netze mit TensorFlow

David Baumgartner

BACHELORARBEIT

Nr. XXXXXXXXXX-A

eingereicht am
Fachhochschul-Bachelorstudiengang

Software Engineering

in Hagenberg

im Januar 2017

Diese Arbeit entstand im Rahmen des Gegenstands

. . . . . .

im

Wintersemester 2016/17

Betreuer:

Stephan Dreiseitl, FH-Prof. PD DI Dr.

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 14. Januar 2017

David Baumgartner

Inhaltsverzeichnis

Vorwort

Kurzfassung

Abstract

Kapitel 1

Einleitung

Hintergrund - Motivation - Zielsetzung
Warum - wieso - wesshalb?

1.1 Hintergrund

1.2 Motivation

1.3 Zielsetzung

Kapitel 2

Begriffe im Maschinellen Lernen

Diese Erklärung der Begriffe und Elemente verfolgt zwei Ziele. Zum einen stellt dies Grundlage des gesamten Themas dar und soll für Interessierte die nicht so vertraut sind, eine Einführung in die Thematik bieten. Und zum anderen werden viele dieser Begriffe erläutert, welche noch häufig zum Einsatz kommen (u.A. Neuron, Aktivierungsfunktion, ...).

2.1 Data Science

Data Science wird generell als die Extraktion von Wissen aus Daten bezeichnet. Dabei werden die Fachbereiche Statistik und Mathematik, Informatik und Machine Learning sowie einige weitere, mit diesem Begriff zusammengefasst. Das Gebiet für sich wird auch als Berufstätigkeit bezeichnet, wobei meist spezialisierte Formen für die Berufsbezeichnung verwendet werden.

Damit Wissen aus Daten überhaupt extrahiert werden kann, muss ein ganzer Prozess durchlaufen werden. Dieser beginnt mit dem zusammentragen von Rohdaten aus der Realität, welche zu diesem Zeitpunkt noch keinen Zusammenhang offenbaren. Im zweiten Prozessschritt werden diese Daten meist umgebaut und neu sortiert, wobei dieser Schritt nicht immer erforderlich ist. Auf die zurecht gelegten Daten besteht nun die Möglichkeit Modelle, Algorithmen sowie weitere Extraktionen durchzuführen. Die erneut extrahierten Daten werden weiterer Folge als Ausgangsdaten verwendet. Auf diese Daten ausgeführte Modelle und Algorithmen liefern Ergebnisse die visuell dargestellt für eine größer Gruppe an Personen geeignet sind. Aus diesem gelernten Wissen besteht zusätzlich die Möglichkeit dieses zum Generieren von neuen Daten zu verwenden und neue Modelle zu entwickeln, die zum Beispiel Vorgänge in der Natur noch akkurater widerspiegeln.

2.2 Machine Intelligence

Machine Intelligence ist ein Begriff der in dieser Form noch nicht definiert worden ist. Einige namhafte Unternehmen wie Google Inc. und Microsoft Corporation bieten jeweils unterschiedliche Definitionen oder Beschreibungen. Die Definitionen dieser Firmen weicht nur unwesentlich von einander ab. Dieser wird als Überbegriff über das gesamte Gebiet mit Machine Learning, Künstlicher Intelligenz, Konversationsintelligenz und alle Themen die in näherer Beziehung dazu stehen verwendet.

2.3 Machine Learning

Machine Learning definiert eine große Anzahl an Theorien und Umsetzungen von nicht explizit programmierten Abläufen. Diese wurden aus Studien in den Bereichen der Mustererkennung und der rechnerischen Lerntheorie mit Künstlicher Intelligenz teilweise entwickelt. Dieses Gebiet umfasst im Jahr 2016 aber sehr viel mehr. So existieren zusätzliche Ansätze aus dem Bereich der Biologie wie zum Beispiel Neuronale Netzwerke, die dem Gehirn nachempfunden sind und genetische Algorithmen, die der Weiterentwicklung eines Lebewesens ähneln. Ein ganz anderer Zugang wurde in der Sowjetunion verfolgt, mit sogenannten 'Support Vektor Machines', bei welchen man einen rein mathematischen Ansatz anstrebt.

2.4 Neuronale Netzwerke

Neuronale Netzwerke sind im Jahr 2016 auch bekannt unter dem Begriff 'Deep Learning'.

Die Theorie und die ersten Grundlagen wurden im Jahre 1943 von Warrn McCulloch und Walter Pitts geschaffen, die ein Modell entwickelten, jedoch nicht die technischen Möglichkeiten hatten dieses umzusetzen. Dieses führte zur 'Threshold Logik'. Durch den 'Backpropagation'-Algorithmus im Jahre 1975 war es möglich, Netzwerke mit mehr als drei Ebenen zu trainieren.

Neuronale Netzwerke bestehen aus Neuronen, die miteinander verbunden sind und gemeinsam ein Netzwerk ergeben.Die Verbindungen sind nicht fest vorgegeben, sondern können auch zum Beispiel eine Schleife bilden.

2.5 Neuron

Abbildung (Aufbau eines Neurons) - TikZ Grafik

Ein Neuron wurde einer Nervenzelle in einem Gehirn nachempfunden mit den folgenden Bestandteilen:

Informationseingangsstrom ist der Dateneingang, wobei ein Neuron ein bis theoretisch beliebig viele solcher Eingänge haben könnte. Dies hängt von der jeweiligen Architektur des Netzwerks ab.

Informationsgewichtung bezeichnet die Gewichtung mit der der Eingangsstrom gewertet wird. So wird ein Informationseingangsstrom mehr oder weniger berücksichtigt. Diese Gewichtung wird durch den Backpropagation-Algorithmus angepasst und nachjustiert.

Kernfunktion bewirkt das Verarbeiten der gewichteten Informationseingänge. Im einfachsten Fall werden alle Werte aufsummiert. Es wäre aber möglich, jegliche Berechnung hier einfließen zu lassen, welche mehrere Werte verwendet und daraus einen neuen Wert berechnet.

Aktivierungsfunktion berechnet den Ausgang eines Neurons. Dabei wird eine weitere Funktion auf das im Kern berechnete Ergebnis ausgeführt und führt dazu, dass ein Ergebnis noch stärker ausgeprägt weitergegeben wird oder minimiert wird. Diese Aktivierungsfunktion ist meist die Sigmoid-Funktion oder eine lineare Funktion.

Die einfachste Repräsentation eines Neurons lässt sich mathematisch folgendermaßen darstellen. Im Kern wird eine Summenberechnung durchgeführt. Dabei werden die Eingangswerte und deren Gewichtung miteinander multipliziert, sowie diese Ergebnisse aufsummiert. Der griechische Buchstabe ϕ (phi) steht für die Aktivierungsfunktion des Neurons und stellt damit die Ausgabe des Neurons dar.

$$f(x, w) := \phi(\sum_{i} w_i * x_i) \tag{2.1}$$

Bias Neuron definiert einen Spezialfall eines Neurons, welches keine Dateneingänge somit auch keine Gewichtung hat und keine Berechnung im Kern durchführt. Dieses liefert nur einen konstanten Wert, wie zum Beispiel eine Eins. Durch die konstante Auslieferung wird auch die Aktivierungsfunktion überflüssig. Das Bias Neuron stellt somit einen stetigen Wert für das Netzwerk dar, beziehungsweise für die darauffolgende Ebene.

2.6 Ebenen/Layer

Ebenen sind Zusammenschlüsse von Neuronen, welche sich auf der selben Stufe befinden. Diese Neuronen sind aber nicht miteinander verbunden, son-

dern bekommen Daten aus der Ebene davor und geben diese an die darauffolgende Ebene weiter. Dieser Typ wird **Hiddenlayer** bezeichnet. Jedes Netzwerk benötigt zusätzlich zwei weitere Ausprägungen an Ebenen. Diese sind:

Inputlayer stellen den Übergang zwischen der Welt außerhalb des Neuronalen Netzwerks und dem Netzwerk dar. Diese Ebene nimmt die Daten ohne Gewichtung auf und gibt sie an die darauffolgende Ebene weiter.

Outputlayer befindet sich am Ende eines Netzwerkes. Dieser Layer hat die Aufgabe, die Daten nach außer weiter zu geben anstatt an das darauffolgende Netzwerk. Hierbei werden die Informationen meist nur mehr für die Ausgabe aufbereitet.

Abbildung (Aufbau eines Einfachen FeedForward NN) - TikZ Grafik

2.7 Informationen Merken und wieder Erkennung

Durch das Anpassen der Gewichtungen bei jedem Dateneingangsstrom mit Hilfe des Backpropagation-Algorithmus ist es möglich, Zustände zu speichern und diese auch zu merken. Dies führt dazu, sollte ein ähnlicher Dateneingang stattfinden, wo zuvor schon ein ähnlicher in einem Trainingszyklus vorhanden war, dies erkannt wird. Dieser kann möglicherweise zu derselben Kategorie gehören, wie der zuvor schon bekannte gemachte und gelernte Dateneingang.

2.8 Backpropagation

Bis zum Jahre 1986 gab es kein automatisierte Möglichkeit, die Gewichtungen in einem Netzwerk anzupassen. In diesem Jahre entwickelten Rumelhart, Hinton & Williams eine mögliche Lösung, welche sehr ähnlich zu anderen Ansätzen von früher war (Werbos, 1974; Parker, 1985; Cun, 1985). Die zentrale Idee in ihrer Lösung liegt darin, die Abweichung des produzierten Ergebnisses zum wirklichen erwarteten Ergebnis zu bestimmen. Aufgrund dieses Fehlers lassen sich im Anschluss die Gewichtungen im Netzwerk von Ende zum Anfang nachjustieren. Diese Technik ermöglichte damit auch die Möglichkeit tiefe Netzwerke zu konstruieren und auch zu Trainieren.

Lernrate skaliert die Steigung des Lernprozesses, mit der Auswirkung ob schneller oder langsamer Gelernt wird. Eine Lernrate unter null würde die Lerngeschwindigkeit stark verlangsamen. Ein Wert über eins würde eine

hohen Lerngeschwindigkeit zur Folge haben. Eine zu hohe Rate würde nicht zum Konvergieren führen, sondern zum springen.

Momentum stellt wie die Lernrate eine Skalierung des Lernprozesses dar. Dabei wird mit dem Faktor die früheren Gewichtsupdates berücksichtigt. Dies führt dazu, dass locale Tiefpunkte überwunden werden können und das System doch zum globalen Tiefpunkt konvergiert.

2.9 Allgemeine Probleme

Nach dem aktuellen Stand der Dinge können Neuronale Netzwerke nicht jede Frage dieser Welt beantworten. Das Entwickeln eines neuen Netzwerks ist ein sehr schwierige und eine lang andauernde Aufgabe. Dabei können Fehler aufdrehten, welche natürlicher Natur sein können aber auch durch den Entwickler verursacht sein können.

Diese Arbeit wird auf die bekanntesten Probleme eingehen und auch Lösungen oder mögliche Lösungsansätze beinhalten.

Overfitting bezeichnet ein Problem welches nicht nur Machine Learning betrifft, sondern auch Menschen und andere Lebewesen. Zum Beispiel ein Student lernt auf eine Prüfung und ist im Besitz einer Klausur aus einem Vorjahr. Nach öfterem Durchspielen der Fragen und sich selbst testen, befindet er sich in der Lage diese Klausur mit einer sehr hohen Wahrscheinlichkeit zu bestehen. Dabei hat sich die Klausur in seinem Gehirn eingeprägt, aber nicht das Stoffgebiet zu welchem er eine Klausur schreiben muss. Das Problem wird als Overfitting bezeichnet und beschreibt, dass etwas gemerkt wurde aber nicht gelernt worden ist und somit eine Abwandlung von Informationen nicht wiedererkannt werden.

Gegenmaßnahme -> dropout, Regulatoren 1/2, learnrate, momentum

Daten die zum Trainieren von Netzwerken verwendet werde stellen selbst Probleme dar. Eine zu geringe Menge an Daten stellt den Entwickler vor das Problem, dass er für diese Daten ein akkurates Netzwerk entwickeln kann. Dieses Netzwerk würde aber möglicherweise nicht die gewünschten Resultate liefern, da der zur Verfügung gestellten Daten nur einen kleinen Teil des gesamten Repräsentieren. Zusätzlich kann es sein, dass die zur Verfügung gestellten Daten selbst nicht vollständig sind und somit wieder nur ein Subset verwendet werden kann.

Datensätze können ähnlich sein zu anderen, sind aber trotzdem wieder eigenjuniek. Dies bedeutet, dass ein Netzwerk welches für eine Problemstel-

lung entwickelt und trainiert wurde, nicht eins zu eins übernommen werden kann. Für diese Daten muss wieder ein Netzwerk entwickelt werden.

2.10 Trainieren

Überwachtes Trainieren definiert, dass die Daten welche zur Verfügung stehen aus zwei Teilen bestehen. Erstens aus den Daten selbst aus welchen gelernt und verstanden werden soll. Zweitens aus den Ergebnissen zu welchem das Netzwerk kommen sollte. Der bekannte Wert wird meistens als Lable bezeichnet. In diese Situation liefert das Netzwerk ein Ergebnis welches mit dem erwarteten Wert verglichen werden kann. Dieser Unterschied wird zum feststellen des Fehlers verwendet, welche besagt wie inkorrekt das Ergebnis ist. Des weiteren wird diese Fehlerwert für die Backpropagation (2.8) benötigt.

Unüberwachtes Trainieren kommt dann zu tragen wenn nur Daten zum Trainieren zur Verfügung stehen. Der erwartete Ausgang ist unbekannt. Diese Strategie wird in Fällen von Clustering (2.11.1) verwendet. Dabei sollen nicht bekannte Gruppen von zusammengehörende Beispiele identifiziert werden. Self-Organizing Maps (2.12.2) entdecken zusammen gehörende Muster und geben dies in einer Grafik zum Beispiel weiter zur weiteren Interpretation.

2.11 Domänenklassen

Neuronale Netzwerk können sehr vielseitig eingesetzt werden. Grundsätzlich lässt sich jedes Problem, welches als Funktion repräsentiert werden kann, durch ein Neuronales Netzwerk approximieren.

In dieser Arbeit werde sieben Hauptdomänen erklärt und beschrieben. Im speziellen wird auf die Neuronalen Netzwerke eingegangen, welche öfter zum Einsatz kommen und eingesetzt werden.

2.11.1 Clustering

Das Clustering Problem bezeichnet das Einordnen von Daten in Klassen oder Gruppierungen. Diese Gruppierungen können von einem Netzwerk selbst definiert werden oder manuell festgelegt werden. Die Anzahl der Gruppen in welche die Daten eingeordnet werden sollen können festgelegt werden aber auch im Falle einer Self-Oranizing-Map auch selbst definieren werden.

2.11.2 Regression

Regression beschreibt den Fall in welchem Daten generiert werden und das Kontinuierlich. Ein Anwendungsfall ist das Finder einer Zugrundeliegenden Funktion wo nur Resultate dieser Funktion vorliegen. So mit werden aus Daten weitere Daten erzeugt. So gibt es Abläufe in der Natur welche Approximiert werden um sie für weitere Systeme möglicherweise zu verwenden. [**bishop2006pattern**]

2.11.3 Classification

Das Classification Problem ist in gewisser Weise ähnlich zu Regression Problemen. Der Unterschied liegt im Ergebnis welche produziert werden. Hier werden Daten dem Netzwerk übergeben, und dieses muss vorhersagen zu welcher Klasse sie gehören. Dies wird in einer überwachten Umgebung durchgeführt. Die Klassen für die Vorhersage sind vorab schon bekannt und können mit den Daten aus dem Outputlayer des Netzwerks verglichen werden und infolge Justierungen durchgeführt werden. [**AI3**]

Ergebnisse einer Classification sagt aus, zu wieviel Prozent etwas auf den gegebenen Input zutrifft. Das gesamt Ergebnis ergibt immer 100 Prozent. Das Ergebnis bei einer Regression wird dabei nicht in Prozent angegeben sondern stellt einen konkreten Wert dar.

2.11.4 Predict

Predict Problemstellungen kommen im Kontext von Business beziehungsweise von E-Business zur Anwendung vor. Hier muss anhand von meist zeitgesteuerten Ereignisse eine Vorhersage getroffen werden. Zum Beispiel an der Börse ändern sich täglich die Kurse relativ rasch, sodass es für Menschen praktisch nicht mehr möglich ist, diesen zu Folgen. Im Falle der Börse sind zeitlich Kurse aus der Vergangenheit verfügbar. Diese werden als Trainingsset für ein Netzwerk verwendet, um den nächsten Tag möglicherweise vorherzusagen. Es kann somit als eine Spezialisierung von Regression und Classification angesehen werden, da Daten generiert werde diese aber mit einer Wahrscheinlichkeit.

2.11.5 Robotics

Auch bekannt unter dem Namen Robot-Learning. Dabei lernen Roboter eigenständig neue Techniken oder passen sich automatisch ihrer Umgebung an. Dabei gibt es Kernprobleme wie, dass in Realtime etwas zum Beispiel Dreidimensionales in einer höhere Dimension berechnet werden muss. Aus diesen Daten muss zur selben Zeit gelernt werden aber auch Aktionen eingeleitet werden, wie das steuern von Motoren, um zum Beispiel nicht Umzufallen.

2.11.6 Computer Vision

Computer Vision zielt darauf ab, einem Computer das Sehen und Verstehen von Bildern zu ermöglichen. Diese Technik findet im Jahr 2016 schon häufig Einsatz. So werden automatisiert Bilder analysiert, beschrieben sowie auch in Gruppen ein geordnet nach diversen Kategorien wie zum Beispiel Gesichtsgefühlszustände. Das Dienste werden auch kommerziell eingesetzt und auch angeboten. In autonom gesteuerten Fahrzeugen findet diese Technologie auch bereits Verwendung um Objekte zu erkennen und zu verstehen. So muss zum Beispiel ein Verkehrszeichen von einem Passanten unterschieden werden können.

2.11.7 Optimierung

Optimization bezieht sich auf eines der Grundprobleme der Informationstechnologie. So werden immer bessere schnellere Algorithmen entwickelt, welcher konkrete Probleme noch effizienter lösen können. Durch das Thema BigData entstand das Problem, sodass selbst sehr effiziente Algorithmen einige Problemstellungen nicht mehr in konstanter oder adäquater Zeit lösen können. Das 'Salesman' Problem gehört zu diesen Problemen. Durch Optimization wird in konstanter Zeit eine Lösung ermittelt, welche nicht die beste Lösung repräsentiert. Diese Lösung liegt aber im Rahmen von einer bestimmten definierten Toleranz und kann als Lösung verwendet werden. [**AI3**]

2.12 Neuronale Netzwerktypen

In den letzten Jahren heraus haben sich diverse gut funktionierende Neuronale Netzwerktypen gebildet, beziehungsweise sind entwickelt und erforscht worden. Diese Netzwerktypen definieren Richtlinien oder Ansätze zu möglichen Netzwerken, welche aber nicht komplett übernommen werden müssen, sonder einen kreativen Spielraum ermöglichen.

2.12.1 FeedForward

FeedForward Netzwerke (FFN) waren bis for einigen Jahren noch der Stand der Forschung. Auf ihnen basieren einige andere Typen von Netzwerken, die bekanntesten werden in dieser Arbeit noch behandelt. Ein FFN basiert auf den Grundlagen eines Neurons (Neuron 2.5), sowie dem ausbauen dieses zu Ebenen mit mehreren Neuronen (Layer 2.6). So ein Netzwerk besitzt einen Inputlayer, einen Hiddenlayer sowie einen Outputlayer. Sobald das Netzwerk mehrere Hiddenlayer aufweist wird es als Deep FeedForward Netzwerk (2.12.5) bezeichnet. Das FFN weist dabei eine Charakteristik auf, indem

dass der Datenfluss eindeutig definiert ist. Der Datenfluss beginnt bei dem Inputlayer und endet bei dem Outputlayer ohne dass ein Datenrückfluss zum Beispiel von dem Hiddenlayer in den Hiddelayer vorhanden ist. Dies würde eine Rekursion oder einem Kurzzeitgedächtnis entsprechen. Die einzelnen Ebenen müssen dabei aber nicht voll verbunden sein, sondern die Vernetzung kann selbst bestimmt werden.

2.12.2 Self-Organizing Map

Self-Organizing Map (SOM) findet vor allem im Bereich der Classification (2.11.3) Verwendung und wurden von Kohonen (1988) erfunden. Es ist nicht erforderlich einer SOM die Information zu geben, in wie viele Gruppen oder Klassen die Daten unterteilt werden sollen. Dadurch gehören es zu den Systemen, welche unsupervised trainiert werden. Außerdem besitzen sie die Möglichkeit, sich auch nach der Trainingsphase auf sich ändernde Eingangsdaten anzupassen. Kohonen entwarf die SOM mit zwei Ebenen, ein Inputlayer und ein Outputlayer ohne Hiddenlayer. Der Inputlayer propagiert Muster an den Outputlayer, wo der Dateneingang gewichtet wird. In dem Outputlayer gewinnt das Neuron, welches den geringsten Abstand zu den Eingangsdaten hat. Dies geschieht durch das Berechnen der euklidischen Distanz. Diese Art von Netzwerk kommt ohne Bias Werte (2.5) aus und es kommen ausschließlich Lineare Aktivierungsfunktionen zur Verwendung.

2.12.3 Hopfield Neuronal Network

Ein Hopfield Neuronal Network (HNN) ist ein einfaches Netzwerk welches aus einem Layer besteht. In diesem Layer sind alle Neuronen mit jedem anderen Neuron verbunden. Dieses Muster wurde von Hopfield (1982) erfunden. Im Gegensatz zu anderen Netzwerken können Hopfield Netzwerke in einer Matrix abgebildet werden, in welcher die Gewichtung zu den einzelnen Neuronen abgebildet werden. Die Neuronen selbst nehmen dabei den Zustand 1 für Wahr und -1 für Falsch an. Das Problem bei diesem Type ist, dass jedes Neuron auf dem Status der anderen aufbaut. Dies stellt ein Problem für die Reihenfolge der Berechnung dar, was zu einem nicht stabilen Zustand führt. Durch das Hinzugeben einer Energiefunktion kann festgestellt werden, in welchem Zustand sich das Netzwerk befindet.

2.12.4 Boltzmann Machine

Im Jahre 1985 stellten Hinton & Sejnowski das erste mal eine Boltzmann Maschine vor. Es stellt ein zwei Ebenensystem dar, mit einem Inputlayer und einem Outputlayer, wo jeder Knoten mit jedem verbunden ist außer mit sich selbst. Des voll vernetzte System unterscheidet eine Boltzmann Maschine von einer eingeschränkten Boltzmann Maschine (RBM), welche eine Grundlage für tiefes Lernen und tiefe Neuronale Netzwerke darstellt. In

einer RBM sind alle sichtbaren Neuronen mit allen Neuronen in dem Outputlayer verbunden. Die Verbindungen zwischen den Neuronen in dem selben Layer entfallen ebenfalls. Der alte uneingeschränkte Type der Boltzmann Maschinen kann für Optimierungsprobleme sowie für Mustererkennungen gut eingesetzt werden.

2.12.5 Deep FeedForward

Deep FeedForward Netzwerke unterscheiden sich zu normalen FeedForward Netzwerken in dem, dass sie mehrere Hiddenlayers beinhalten anstatt nur einem.

2.12.6 NEAT

NeuroEvolution of Augmenting Topologies (NEAT) Netzwerke sind relative jung, wobei NEAT für einen Algorithmus steht der Neuronale Netzwerke entwickelt. Sie wurden von Stanley und Miikkulainen (2002) entwickelt. Dieser Type verwendet genetische Algorithmen, um die Struktur und die Gewichtungen im Netzwerk zu optimieren. Die Input- und Outputlayer sind identisch zu einem FeedForward Netzwerke. Dafür fehlt diesem Type eine innere Struktur. Die Verbindungen sind lose und nicht klar definiert und können während dem Entwickeln entfernt werden aber auch wieder hinzugefügt werden.

Compositional pattern-producing network (CPPN) ist ein Netzwerk das andere Strukturen entwickelt und basiert dabei auf der Theorie von NEAT. Dies können Bilder aber auch andere Netzwerke sein, wobei meist Bilder generiert und weiter entwickelt werden. CPPN können im Gegensatz zu NEAT Netzwerken mit verschieden Aktivierungsfunktionen verwendet werden. Ein erzeugtes Netzwerke resultiert aber immer in einem regulären NEAT Netzwerk.

VL weglassen

HyperNEAT ist eines der bekanntesten CPPN Netzwerke welches keine Bilder produziert sondern neue Netzwerke erstellt. Mit der Fähigkeit andere Netzwerke zu kreieren, welche wiederum für ihre Aufgabe gute Ergebnisse liefern, ermöglicht es schneller Netzwerke zu kreieren und sich auf ändernde Probleme schneller anzupassen.

2.12.7 Convolutional neural network

Werden selbst nicht als komplettes eigenes Netzwerk verwendet, sondern in FeedForward Netzwerken verwendet. Im Speziellen wen es sich um Bilderkennung geht. Dabei werden entweder zwei Ebenen nicht voll vernetzt

sondern nur teilweise und somit Gewichtungen eingespart. Im zweiten Fall werden die Gewichtungen geteilt, sodass immer in die selbe Richtung verlaufende Verbindungen die selbe Gewichtung aufweisen. Dies ermöglicht es komplexe Strukturen zu speichern und trotzdem die Speicherauslastung niedrig zu halten und die Effektivität aufrecht zu halten.

2.12.8 Elman Network

2.12.9 Jordan Network

2.12.10 Recurrent Network

2.13 Domänen und Typen Matrix

2.14 Tensorflow Typen Unterstützung

Kapitel 3

TensorFlow Bibliothek

3.1 Python API

3.2 C++ API

3.3 Go API

3.4 TensorFlow Python

3.4.1 Graphs / Dataflowgraph

3.4.2 Operation

3.4.3 Tensor

3.4.4 Operationen

Konstanten, Zufallswerte

Variables

Transformationen

Mathematik

Flusskontrolle

Images / FFmpeg

Input und Readers

Neural Network

Running Graphs

Training

3.4.5 Probleme

NaN Problem

Kapitel 4

Facial Keypoints Detection

4.1 Ausgangssituation

4.2 Vorbereitung

4.2.1 Daten vorbereiten und normalisieren

4.2.2 Evaluation- und Errorfunktion

4.3 Neuronale Ebenen vorbereiten

4.4 Neuronale Ebenen verknüpfen

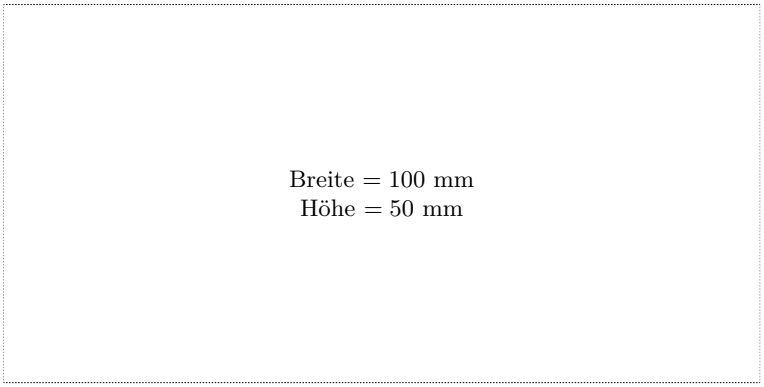
4.5 Trainieren

4.6 Validierungsresultate

Quellenverzeichnis

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —