Abschlussbericht für das Abschlussprojekt Deep Learning

Tobias Giesler, Florian Graf, Tim Kleinoth

TH Köln

Cologne, Germany

tobias\_ralf.giesler@smail.th-koeln.de

florian.graf@smail.th-koeln.de

tim.kleinoth@smail.th-koeln.de

*Zusammenfassung*— Dieser Bericht beschreibt das Erstellen einer Deep Learning Architektur, welche die Handzeichen „Stein, Schere und Papier“ erkennen soll. Verwendet werden dabei die Datensätze von „Julien de la Bruère-Terreault“ [1], sowie ein eigens erstellter Datensatz.

Schlagwörter—Deep Learning, Bilderkennung, Künstliche Intelligenz

# Einleitung

Der Begriff „Künstliche Intelligenz“ (KI) beschreibt Computerprogramme, die versuchen eine menschliche Intelligenz nachzuahmen. Ziel dabei ist es ein Programm zu entwickeln, dass selbstständig Probleme löst, indem es auf unterschiedlichste äußere Einflüsse reagiert. Der Begriff Deep Learning beschreibt dabei eine gewisse Form einer künstlichen Intelligenz, welche aus mehreren sogenannten „Hidden Layers“ besteht. Diese Schichten beschreiben die Vernetzung mehrerer „Perceptrons“ also künstliche Neuronen (siehe Abbildung 1), die zum Anlernen eines Prozesses verwendet werden.

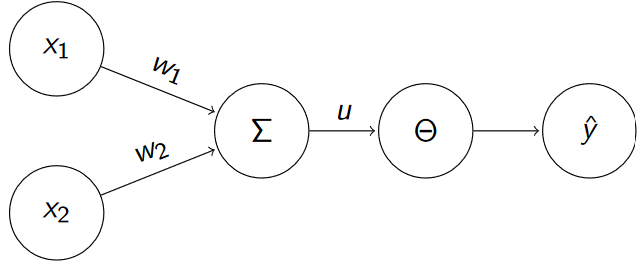


Abbildung 1: Darstellung eines Perceptrons [2]

Künstliche Intelligenzen finden schon heute in vielen Einsatzgebieten Verwendung. Zu diesen Einsatzgebieten zäht die Spracherkennung, Videoempfehlungen, Suchanfragenempfehlungen, oder (in der Industrie verwendet) „condition monitoring“, welches eine Überwachung von Sensorwerten durch eine künstliche Intelligenz beschreibt, um Ausfälle von Maschinen schon frühzeitig zu erkennen [3].

Ein weiteres großes Feld in dem künstliche Intelligenzen verwendet werden ist der Bereich der Bilderkennung. Bilderkennung beherbergt ein enormes Potential für Entwicklung. Bilderkennung kann und wird bereits in Bereichen wie Gesichtserkennung, autonomes Fahren, sowie die Überwachung von industriellen Anlagen verwendet.

In diesem Paper wird die Erstellung und Optimierung eines Deep Learning Algorithmus zum Erkennen von den Handzeichen „Stein, Schere und Papier“ beschrieben. Dabei wird auf unterschiedliche Möglichkeiten zur Verbesserung der sogenannten „Generalisierungsfähigkeit“ der KI eingegangen und deren Auswirkungen auf die Lernrate thematisiert.

# Aufbau und Optimierung des deep learning algorithmus

In diesem Kapitel wird der Grundaufbau des Algorithmus sowie Regularisierungsmethoden zur Verbesserung der „Generalisierungsfähigkeit“ thematisiert. Die Generalisierungsfähigkeit bezeichnet dabei die Fähigkeit eines neuronalen Netzes das durch den Trainingsdatensatz „Erlernte“ auf einen fremden Datensatz (zum Beispiel dem Validierungsdatensatz) anzuwenden. Eine gute Generalisierungsfähigkeit ist damit ein Indikator für ein gut trainiertes neuronales Netz.

Regularisierungsmethoden helfen dem neuronalen Netz eine höhere Generalisierungsfähigkeit zu erreichen. Regularisierungsmethoden zielen darauf ab „overfitting“, also die Überanpassung an ein neuronales Netz an einen Trainingsdatensatz zu vermeiden.

Des Weiteren wird auf den Trainingsdatensatz eingegangen.

Filer:

Input=300x200x3

Resize=224x224x3

Jeder Conv hat mehrere Filter und Outputs (eine Feature Map pro Filter). Jede Feature Map hat ein Neuron pro Pixel mit den selben weights, Bias usw..

Conv Layer 1, Filter 3x3:

* Nout=(Nin-F)/Stride+1
* (224-3)/1+1=222 => Feature Map

## Aufbau des neuronalen Netzes

Der Aufbau des neuronalen Netzes als ein „Fully Convolutional Network“ verwendet. (Aufbau der nicht optimierten KI beschreiben mit Begründung warum der Aufbau gewählt wurde).

Die Softmax-Funktion als Aktivierungsfunktion ist eine gängige Lösung, um eine Kombination mehrerer binärer Klassifikatoren zu verhindern. Die Funktion liefert ein Array von vier Werten mit der Gesamtsumme 1. Jedes Element des Arrays spiegelt die Wahrscheinlichkeit einer zu bestimmenden Kategorie wieder. Kapitel verfassen

## Ergebnisse der Lernrate einer nicht optimierten KI

Wie oben bereits erwähnt, soll Ziel dieses Projektes sein, eine Deep Learning Architektur zu entwerfen die selbstständig die Handzeichen „Stein, Schere, Papier“ erkennt. Dazu wurde eine Datensatz erstellt mit Bildern von den Handzeichen „Stein, Schere, Papier“ sowie „Rest“ welches ein den anderen drei Kategorien nicht zugehöriges Handzeichen darstellt. Zusätzlich wurde der bereits verwendete Datensatz von „Julien de la Bruère-Terreault“ verwendet. Ein Beispiel eines Fotos aus diesem Datensatz ist in Abbildung 2 dargestellt:



Abbildung 2: Beispielbild aus dem verwendeten Datensatz (Papier)

Der beschriebene Algorithmus wurde dann mithilfe des Trainingsdatensatzes trainiert, wobei die Trainingsergebnisse mithilfe eines Validierungsdatensatzes zusätzlich überprüft wurden. Anhand der erzielten Ergebnisse der entworfenen Deep Learning Architektur im Trainingsdatensatz sowie im Validierungsdatensatz kann man gut ein gewisses „Lernverhalten“ der entworfenen Architektur begutachten und beurteilen. Die Ergebnisse, die mit dem nicht optimierten Datensatz erzielt wurden, werden in Abbildung 3 und Abbildung 4 dargestellt:

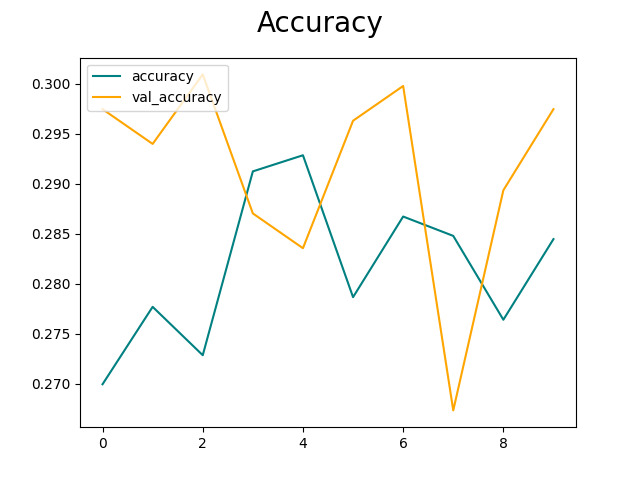


Abbildung 3: Testergebnis der Genauigkeit der nicht optimierten Deep Learning Architektur (Achsen noch beschriften! Genauere Legende, Deutscher Achsentitel)

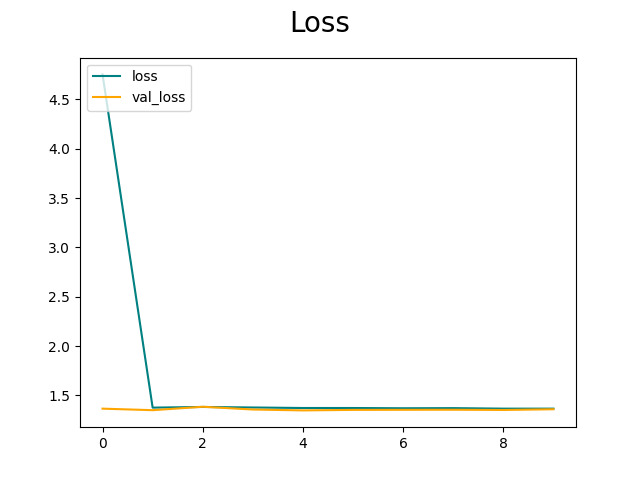


Abbildung 4: Testergebnisse der Loss Funktion der nicht optimierten Deep Learning Architektur (Besser andere Ergebnisse, da loss Funktion der Validation nicht funktioniert)

Interpretation der Ergebnisse hinzufügen. Warum ist das nicht so gut, was kann getan werden damit es verbessert wird.

## Datenvorverarbeitung

Die Bilder des Datensatzes werden nicht direkt verwendet. Stattdessen werden dem Modell nur augmentierte Bilder zur Verfügung gestellt. Da die Augmentierungen nach dem Zufallsprinzip vorgenommen werden, können sowohl veränderte Bilder als auch detailgetreue Nachbildungen der Originalbilder (z. B. fast ohne Augmentierungen) erzeugt und beim Training verwendet werden.

Die Keras-Bibliothek für Deep Learning bietet die Klasse ImageDataGenerator an, welche zur Daten Augmentierung genutzt werden kann. Mithilfe des Datengenerators konnte ebenfalls der Validierungsdatensatz festgelegt werden <https://keras.io/api/preprocessing/image/>.

## Early Stopping

Eine der Verbesserungsmöglichkeiten der Generalisierungsfähigkeit ist das sogenannte „Early Stopping“. Early Stopping wird verwendet, um zu vermeiden, dass sich das entworfene neuronale Netz zu sehr an einen Trainingsdatensatz anpasst und damit übertrainiert wird (overfitting). Ein übertrainiertes Netz erzielt zwar im Trainingsdatensatz, durch die Überanpassung, bessere Ergebnisse, erzielt aber im bisher fremden Validierungsdatensatz schlechtere Ergebnisse. Ein Beispiel für overfitting ist in Abbildung 5 dargestellt:

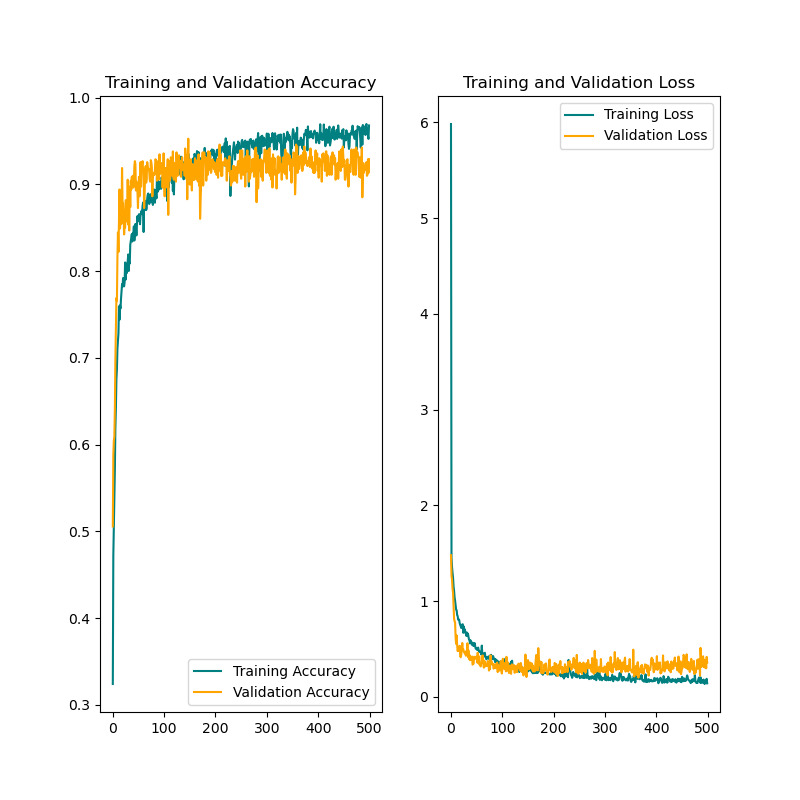


Abbildung 5: Beispiel für die Lernrate eines übertrainierten Netzes

Hier wurde das entworfene neuronale Netz über 500 Epochen trainiert. Man kann erkennen das ungefähr ab 120 Epochen die Genauigkeit beim Testen des Validierungsdatensatzes nicht mehr steigt. Auch die Loss Funktion des Validierungsdatensatzes bleibt ungefähr konstant, wobei man sogar in den späteren Epochen (im Mittel) einen leichten Anstieg erkennen kann, während die Genauigkeit des Trainingsdatensatzes steigt. Dies ist ein klassisches Beispiel für „overfitting“. Das trainierte Netz hat somit eine schlechtere Generalisierungsfähigkeit. Early Stopping kann hierbei gut ansetzen. Beim Early Stopping wird die Fehlerfunktion des Validierungsdatensatzes überwacht. TensorFlow erlaubt es, die Early Stopping Methode anzupassen, indem Parameter an den Algorithmus übergeben werden, z.B. der Validierungsverlust zur Beobachtung und die Geduld zur Bestimmung des Beobachtungsintervalls. Außerdem wird ein Checkpoint implementiert, um das Modell zu sichern, das die besten Ergebnisse erzielt hat. In dem Trainingsverlauf von Abbildung 5 könnte dies ungefähr bei 120 Epochen passieren.

Anwendung des Early Stoppings zeigen.

## Parameter Norm Penalties

Um die Generalisierungsfähigkeit der Deep Learning weiter zu verbessern kann man die Regularisierungsmethode „Parameter Norm Penalties“ anwenden. In dieser Methode wird ein Strafterm (Ω) verwendet, welcher, in Abhängigkeit von den Gewichten w (siehe Abbildung 1) eine gewisse Zielfunktion (J) verändert. Genauer wird dies in der folgenden Rechnung dargestellt.

+ α Ω(w) (1)

Das Ziel bei dieser Regularisierungsmethode ist es wieder overfitting zu vermeiden und somit eine bessere Generalisierungsfähigkeit zu erzielen. Thema nochmal ausführlicher beschreiben.

## Dataset Augmentation

„Dataset Augmentation“ beschreibt eine Regularisierungsmethode, die auf eine bessere Generalisierungsfähigkeit abzielt, indem sie mehr Daten aus einem Datensatz erzeugt. Mehr Daten sind grundsätzlich der erfolgversprechendste Weg, um eine bessere Generalisierungsfähigkeit zu erreichen. Im Falle der Bildbearbeitung werden hierbei mehr Daten erzeugt indem gewisse Filter verwendet werden. Die verwendeten Bilder aus dem Datensatz werden nach einem zufälligen Prinzip verschoben, gedreht, bzw. vergrößert oder ein Blur-Effekt wird eingesetzt, die dann veränderten Bilder werden erneut für das Trainieren des neuronalen Netzes verwendet. Somit erzeugt man sich aus dem vorhandenen Datensatz eine wesentlich größere Menge an Daten, wodurch wiederum eine bessere Generalisierungsfähigkeit erzielt werden kann.

Erzeugten Bilder anzeigen lassen und hier darstellen. Veränderung auf die Lernrate zeigen.

## Dropout

Dropout bezeichnet das zufällige Aussetzen von bestimmten Neuronen im Neuronalen Netz (siehe Abbildung 5). Durch das zufällige Aussetzen von Neuronen wird ein Effekt erzeugt, der umliegende Neuronen zwingt, sich stärker zu vernetzen, um die „Fehler“ von umliegenden Neuronen zu korrigieren. Das Model wird somit insgesamt robuster und erreicht wieder eine höhere Generalisierungsfähigkeit.

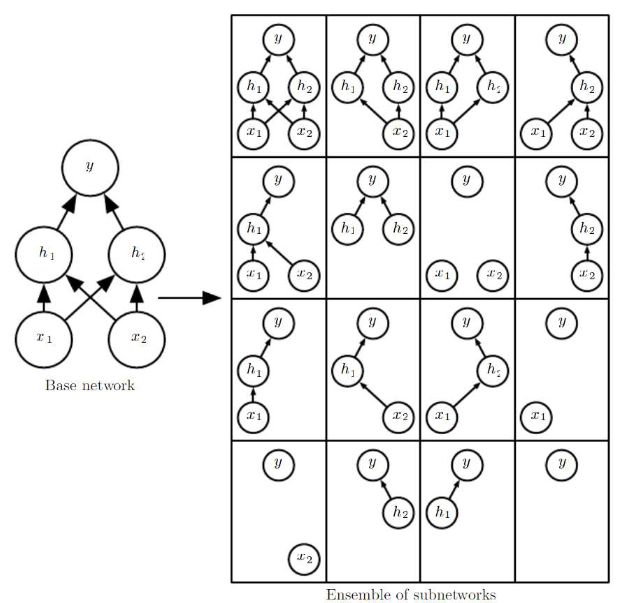


Abbildung 6: Aussetzen von Neuronen (Dropout)

##### Quellenangaben

1. Datensatz von Julien de la Bruère-Terreault: <https://www.kaggle.com/datasets/drgfreeman/rockpaperscissors>
2. Jan Salmen, Vorlesungsfolien „Deep Learning” S. 31
3. Geron Aurelien, “Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow” S. XV