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. Eine Verknüpfung von mehreren Perceptrons wird dabei als ein neuronales Netz bezeichnet und die Methode des machinellen Lernens die ein solches neuronales Netz mit zahlreichen Zwischenschichten („Hidden Layers“) verwendet, wird als Deep Learning bezeichnet.

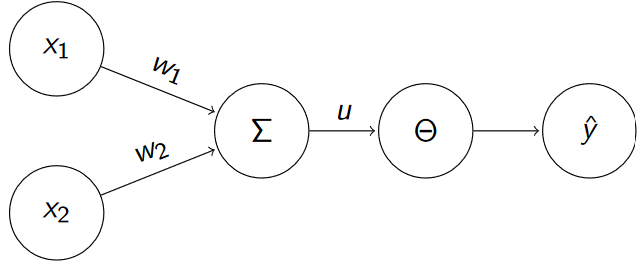


Abbildung 1: Darstellung eines Perceptrons [2]

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. In diesem Bereich wird Deep Learning als Teilgebiet einer Künstlichen Intelligenz verwendet. Bilderkennung beherbergt ein enormes technisches Potential für die Einbindung von neuronalen Netzen in Prozessen. Bilderkennung kann und wird bereits in Bereichen wie Gesichtserkennung, autonomes Fahren, sowie die Überwachung von industriellen Anlagen verwendet. Aber auch im medizinischen Bereich wie frühzeitige Krebserkennung kann die Bilderkennung durch ein neuronales Netz verwendet werden.

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

## Vorstellung des Datensatzes

Um das oben erwähnte Ziel eine Deep Learning Architektur zu entwerfen, die die Handzeichen „Schere, Stein und Papier“ erkennt, wurde ein Datensatz erstellt, der Bilder von den erwähnten Handzeichen enthält. Die Aufnahmen wurden immer so angefertigt das die Handinnenfläche von der Kamera weg zeigt. Zusätzlich wurden zufällige Bilder von Händen erstellt, die weder Stein Schere oder Papier darstellen sollen und mit dem Label „Rest“ gekennzeichnet wurden. Außerdem wurde ein bereits vorhandener Datensatz von „Julien de la Bruère-Terreault“ verwendet. Ein Beispiel eines Fotos aus diesem Datensatz ist in Abbildung 2 dargestellt:

Ein Bild, das grün, Spielhaus enthält.

Automatisch generierte Beschreibung

Abbildung 2: Beispielbild aus dem verwendeten Datensatz (Papier)

## 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

Der beschriebene Algorithmus wurde mithilfe des oben beschriebenen Datensatzes erwähnt. Dieser wurde unterteilt in einen Trainingsdatensatz, also ein Datensatz, mit dem das neuronale Netz trainiert wird und einem Validierungsdatensatz, der einen „fremden“ Datensatz darstellen soll, mit dem das trainierte neuronale Netz getestet wird. 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 dargestellt:

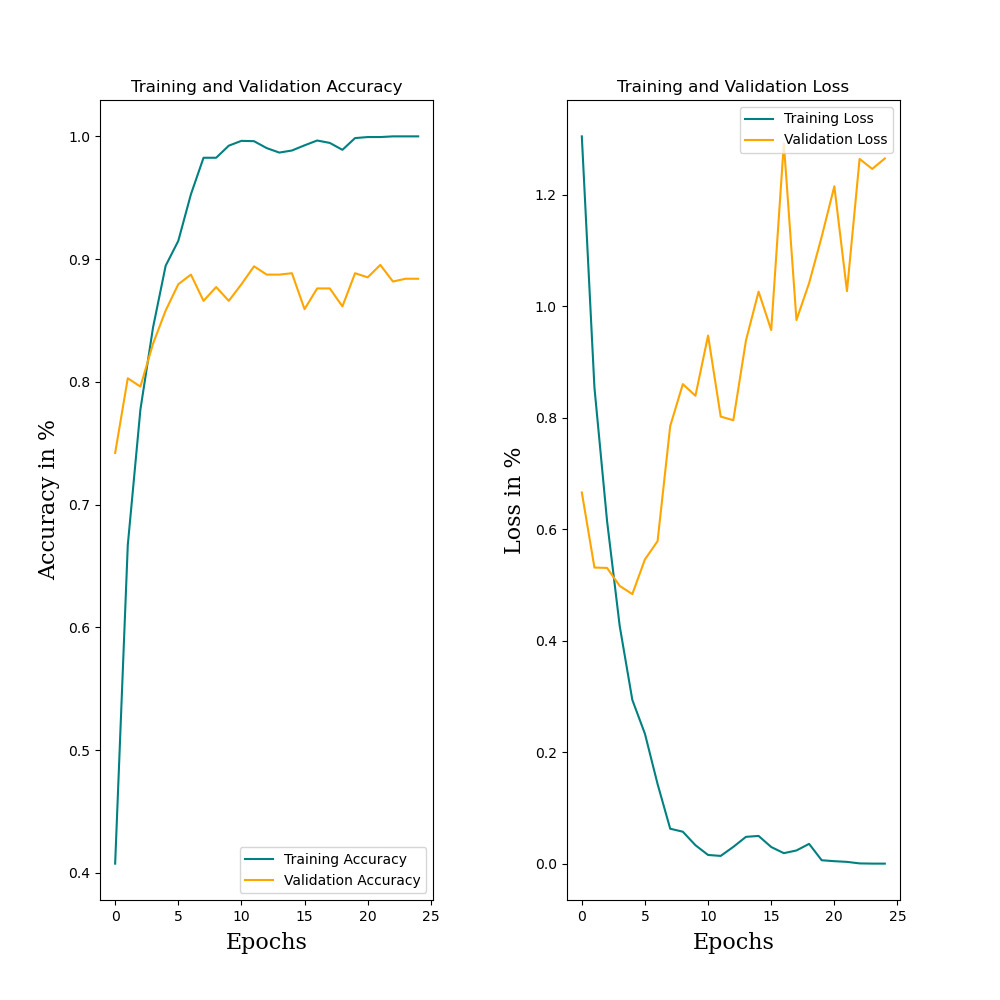


Abbildung 3: Ergebnisse des nicht optimierten neuronalen Netzes mit schlechter Generalisierungsfähigkeit

Betrachtet man die Ergebnisse so erkennt man hier ein Beispiel für das sogenannte „overfitting“. Overfitting bezeichnet die Überanpassung von einem neuronalen Netz an einen Trainingsdatensatz. In Abbildung 3 kann man erkennen das das neuronale Netz in der Genauigkeit der Bestimmung der Handzeichen im Trainingsdatensatz 100% erreicht, während im Validierungsdatensatz weniger Prozent erreicht werden (<90%). Betrachtet man die Loss-Funktion so erkennt man eine starke Abweichung der Werte des Trainingsdatensatzes zu den Werten des Validierungsdatensatzes. Während die Loss-Funktion im Trainingsdatensatz sich gegen 0 bewegt, so steigt die Fehlermenge des neuronalen Netzes im Validierungsdatensatz mit mehreren Epochen sogar an. Das neuronale Netz wird zu sehr an den Trainingsdatensatz angepasst und verliert somit an Generalisierungsfähigkeit. Da die Generalisierungsfähigkeit ein Maß für die Qualität eines neuronalen Netzes darstellt, ist dies ein unerwünschtes Phänomen. Um den Effekt des Overfittings zu verhindern, gibt es unterschiedliche Ansätze, die im Folgenden erläutert werden.

## Dataset Augmentation

„Dataset Augmentation“ beschreibt eine Regularisierungsmethode, die auf eine bessere Generalisierungsfähigkeit abzielt, indem sie mehr Daten aus einem Datensatz erzeugt. Mehr Daten im Datensatz 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 leicht 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.

Bei der Anwendung von Dataset Augmentation in diesem Projekt wurden die Bilder nach einem Zufallsprinzip gedreht und die Helligkeit wurde verändert. Beim Einfügen der Verschiebung der Bilder, in der Dataset Augmentation Funktion, ist in diesem Projekt ein Fehler aufgetreten, welcher starke Schwankungen in den Lernkurven verursacht hat. Bei Betrachtung der Inhalte der Datensätze ist dann aufgefallen das durch die Verschiebung einige Bilder keine, bzw eine kaum noch zu erkennende Hand enthielten. Dies wurde in der folgenden Abbildung festgehalten:

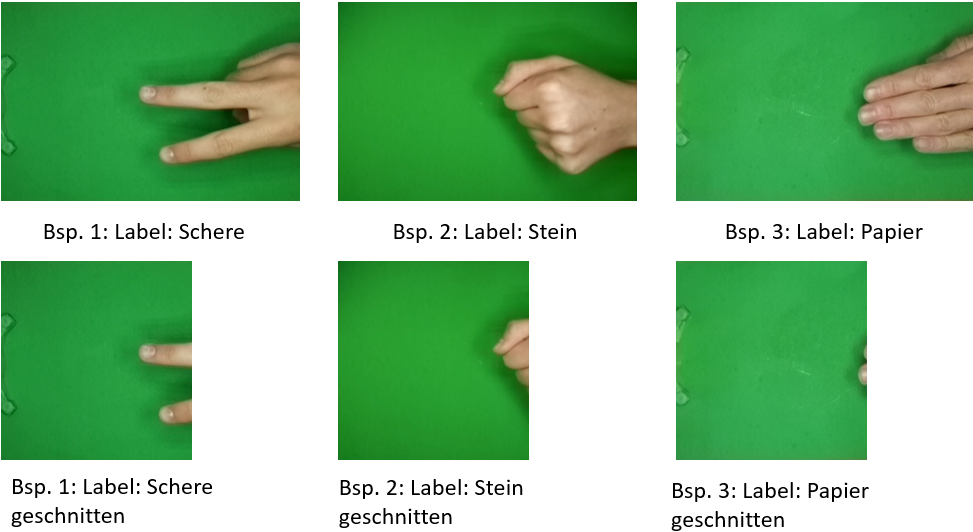


Abbildung 4: Abbildung bearbeitete Bilder. Durch die Verschiebung des Bildes ist eine Hand im Bild kaum noch enthalten

Eventuell noch Bild der Lernrate mit Verschiebung zeigen?

Nach Anwendung von Dataset Augmentation (ohne Verschiebung der Bilder) konnte ein Lernverhalten festgestellt werden welches in Abbildung 5 gezeigt wird. Vergleicht man nun Abbildung 4 mit Abbildung 5 so kann man erkennen, dass der Unterschied der Loss-Funktion vom Trainingsdatensatz zum Validierungsdatensatz geringer geworden ist. Auch die Genauigkeit in der Validierungsfunktion ist in ihrem Maximum ein wenig gewachsen. Darauf lässt sich zurückschließen dass die Generalisierungsfähigkeit des neuronalen Netzes ein wenig verbessert werden konnte.

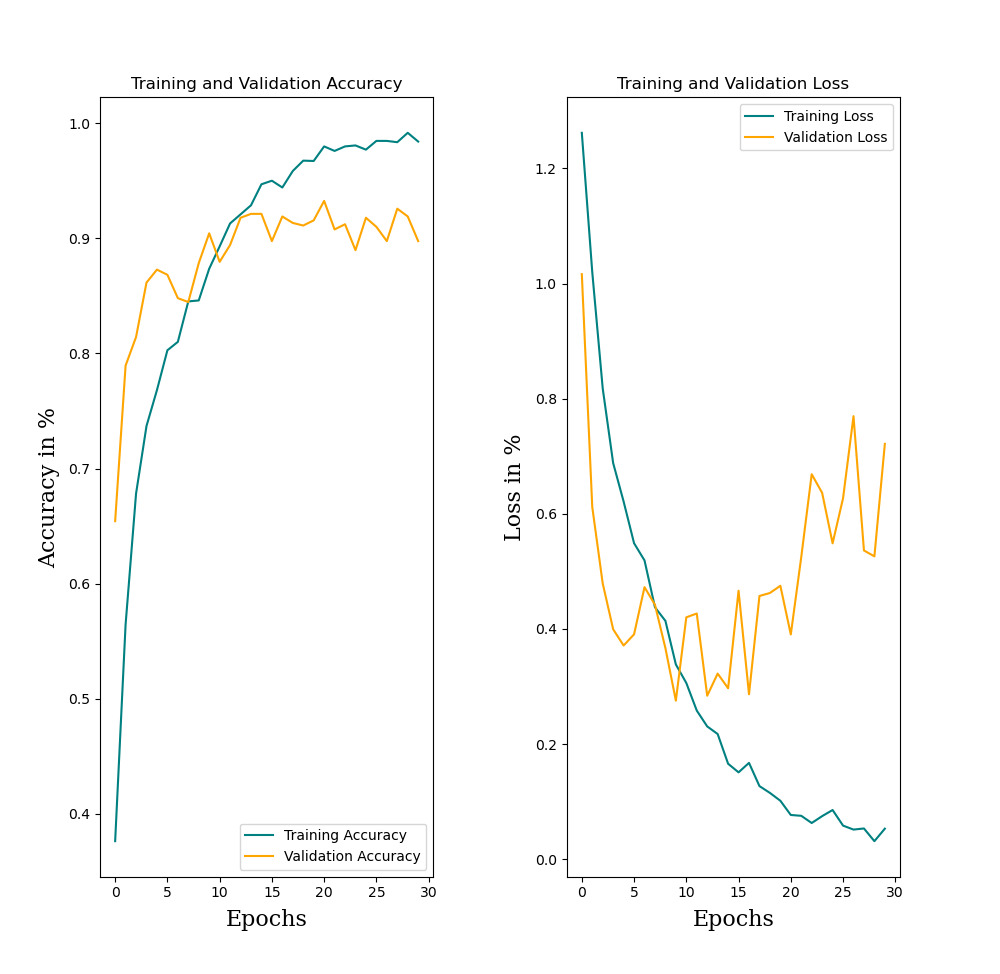


Abbildung 5: Lernverhalten des neuronalen Netzes mit Dataset Augmentation

## 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.

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.

Anwendung des Early Stoppings zeigen + Erklärung

## 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.

## 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.

## 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/>.

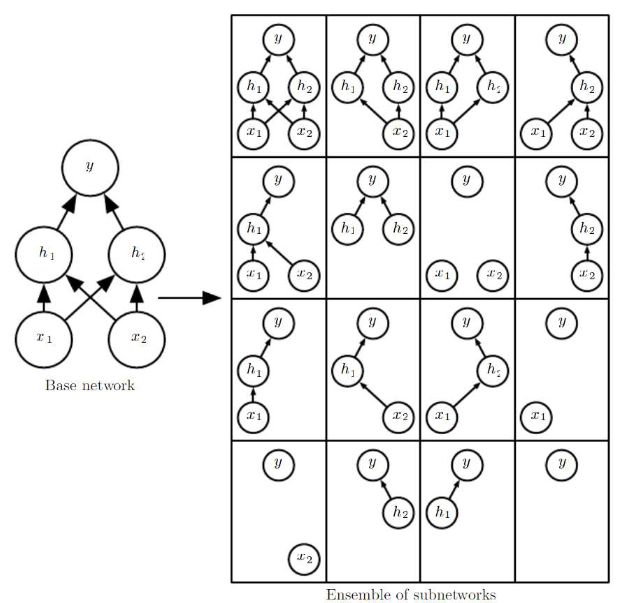


Abbildung 6: Aussetzen von Neuronen (Dropout)

##### 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