Bachelorarbeit

im Studiengang Computerlinguistik

an der Ludwig-Maximilians-Universität München

Fakultät für Sprach- und Literaturwissenschaften

Department 2

Hyperparameteranalyse linguistisch-informierter Konvolutionsnetze für Sentiment Analyse

vorgelegt von
Michael Hochleitner

Betreuer:	Prof. Dr. Hinrich Schütze
Aufgabensteller:	M.Sc. Sebastian Ebert
Bearbeitungszeitraum:	21. September - 30. November 2015

Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 30. November 2015

.....
Michael Hochleitner

Zusammenfassung

In der Bachelorarbeit wurden Experimente mit einem linguistisch-informierten Konvolutionsnetz für Sentiment Analyse durchgeführt. Es wurden Kombinationen der Hyperparameter Lernrate, Filteranzahl, k-Max-Pooling, L2-Regularisierung, Dropout und Anzahl verdeckter Neuronen getestet.

Die getesteten Werte sind in Tabelle 1 abgebildet. Die Konfiguration mit dem höchsten F1-Maß auf dem SemEval-Testset von 2013 setzt sich aus folgenden Werten zusammen: Lernrate 0,01, Filteranzahl 300, Max-Pooling 5, Regularisierungsparameter 0,00001, Dropout-Rate 0,5 und verdeckte Neuronen 100. Mit dieser Konfiguration wurde ein Makro-F1-Maß von 71% erreicht. Die Konfiguration mit dem niedrigsten F1-Maß auf den SemEval-Testdaten besteht aus: Lernrate 0,001, Filteranzahl 100, Max-Pooling 1, Regularisierungsparamter 0,0005, Dropout 0,8 und verdeckte Neuronen 0. Damit wurde ein Makro-F1-Maß von 51% erreicht. Der Unterschied der F1-Maße dieser beiden Kombinationen zeigt, dass durch eine gute Einstellung der Hyperparameter eine Verbesserung der Klassifikation erreicht werden kann.

Bei der Untersuchung der Ergebnisse von Modellen mit verschiedenen Hyperparametern hat sich gezeigt, dass die Lernrate den größten Einfluss auf das F1-Maß hat. Eine Erhöhung der Filteranzahl im Bereich von 100 bis 300 hat auch Verbesserungen gebracht. Beim k-Max-Pooling schneidet der Wert 5 besser ab als der Wert 1. Für das Regularisierungsparameter bei der L2-Regularisierung empfiehlt sich bei einer hohen Lernrate ein größerer Wert und für eine kleine Lernrate ein kleinerer Wert. Der Einsatz von Dropout zeigte Verbesserungen. Von den Werten 0,5 und 0,8 war 0,5 der bessere. Verdeckte Neuronen sollten eingesetzt werden. 100 verdeckte Neuronen brachten bessere Ergebnisse als 500.

Parameter	**Wert 1**	**Wert 2**	**Wert 3**	**Wert 4**
Lernrate	0,1	0,01	0,001	
Filteranzahl	100	200	300	400
Filterbreite	2			
Max-Pooling	1	5		
Regularisierungsparameter	0,00001	0,00005	0,0001	0,0005
Dropout	0	0,5	0,8	
verdeckte Neuronen	0	100	500	

Tabelle 1: getestete Parameter und Werte

Inhaltsverzeichnis

1 Einleitung

Die Bachelorarbeit befasst sich mit der Anwendung von künstlichen neuronalen Netzen (KNNs) in der Sentiment Analyse. Mit KNNs wurden State-of-the-Art Ergebnisse in Sentiment Analyse erreicht, z.B von Severyn und Moschitti (2015).

Ein Problem bei KNNs ist die große Anzahl von Hyperparametern, die eingestellt werden müssen. Wegen der Komplexität von KNNs und der vielen Parameter, die sich gegenseitig beeinflussen ist es schwierig theoretische Begründungen für die Einstellung eines Hyperparameters auf einen bestimmten Wert zu finden. Deshalb muss die Hyperparametereinstellung empirisch untersucht werden. Dafür werden die Werte einiger Hyperparameter manuell eingestellt und die Ergebnisse ausgewertet. Die Motivation ist das Erzielen besserer Ergebnisse durch eine Verbesserung der Hyperparametereinstellung. Das soll erreicht werden, indem aus den Ergebnissen der Experimente Anhaltspunkte für die Hyperparametereinstellung formuliert werden.

Die Arbeit beginnt mit einer Einführung in das Thema Sentiment Analyse in Kapitel 2 *Grundlagen*. Dabei wird geklärt, was man mit Sentiment Analyse erreichen will und welche Schwierigkeiten dabei auftreten. In Kapitel 2 befindet sich außerdem eine Einführung in neuronale Netze. Dort werden die Bestandteile von neuronalen Netzen, das allgemeine Vorgehen beim Training im Maschinellen Lernen und das Gradientenverfahren kurz geschildert. Außerdem werden Konvolution und Hyperparameter erklärt. Kapitel 3 behandelt den Einsatz von Neuronalen Netzen im Natural Language Processing. Dabei werden Wortrepräsentationen, die Umsetzung von Konvolution für Sätze, Max-Pooling, Softmax sowie Überanpassung, Regularisierung und Dropout besprochen. In Kapitel 4 *Experimente* befindet sich der Hauptteil der Arbeit: die Parameterstudie. Zunächst wird dort das für die Experimente verwendete System, die Hyperparameter und deren Werte beschrieben. Dann folgen Grafiken für alle getesteten Parameter in Form von Boxplots und Heatmaps für Parameterkombinationen. Am Ende des Kapitels steht eine Zusammenfassung der Ergebnisse. In Kapitel 5 *Verwandte Arbeiten* werden andere Arbeiten zur Satzklassifikation und Parameterstudien mit neuronalen Netzen beschrieben und mit den Ergebnissen der vorliegenden Arbeit verglichen. In Kapitel 6 *Zusammenfassung* werden die Ergebnisse zusammengefasst und ein Ausblick für weitere Untersuchungen gegeben.

2 Grundlagen

2.1 Sentiment Analyse

Folgender Überblick über das Thema Sentiment Analyse (SA) basiert auf „Opinion mining and sentiment analysis" von Pang und Lee (2008).

2.1.1 Motivation

Eine naheliegende Vorgehensweise, um gute Entscheidungen zu treffen, ist andere um Rat zu fragen. Dabei stellt sich die Frage, wie viele Meinungen eingeholt werden können oder sollten, um eine gute Entscheidung zu treffen. Vor Aufkommen des Internets beschränkte sich die Auswahl an Meinungen bei einer Privatperson auf die Familie, den Freundes- und Bekanntenkreis und Meinungen aus dem Radio, dem Fernsehen oder der Zeitung. Das Internet ermöglicht den Zugang zu einer Anzahl von Meinungen, die diese Auswahl bei weitem übersteigt. Im Internet werden Meinungen von Laien und Experten zu einer Vielzahl von Themen geäußert. Themen, die laut Pang und Lee von Interesse für Internetnutzer sind, sind vor allem Bewertungen von Produkten und Informationen über Wahlkampagnen. Meinungen zu diesen Themen werden von sehr vielen Leuten nachgeschlagen und geäußert.

Die Meinungen der Internetbenutzer über Produkte können einen großen Einfluss auf die Beliebtheit der Produkte haben. Einen Überblick über diese Meinungen zu haben, hilft einer Firma einzuschätzen, wie beliebt ihre Produkte sind. Aufgrund dieser Information kann entschieden werden, ob ein Produkt verändert, fortgeführt, oder nicht weiter produziert werden soll. Dieses Szenario ist ein Beispiel für eine Entscheidungsfindung, die durch die Meinung anderer beeinflusst werden kann. Die große Masse an Meinungen im Internet führt dazu, dass frühere Methoden der Medienbeobachtung, die einen Überblick über Bewertungen bilden, nicht mehr praktikabel sind. Deswegen sollen Systeme programmiert werden, die automatisch Meinungen extrahieren.

2.1.2 Teilaufgaben und Terminologie

Um automatisch Meinungen zu einem Thema zu extrahieren, müssen mehrere Teilaufgaben gelöst werden. Zuerst muss eine Auswahl von Texten erfolgen, die für das gewählte Thema relevant sind. Bei Internetseiten wie Amazon, auf denen Meinungen zu einem Thema (hier ein Produkt) gebündelt auftreten, ist dieser Schritt einfacher als bei Blogs, in denen oft Meinungen zu verschiedenen Themen geäußert werden. Wenn ein relevanter Text gefunden wurde, muss die dort ausgedrückte Meinung (das Sentiment) klassifiziert werden. Manche Internetseiten erleichtern die Meinungsextraktion dadurch, dass Bewertungen zusätzliche eine Bewertungsskala enthalten. Bei Bewertungen, die nur aus Text bestehen, ist die Zuweisung eines Sentiments schwieriger.

Wenn das Sentiment mehrerer Texte festgestellt wurde, soll die Summe der Meinungen sinnvoll dargestellt werden. Diese Darstellung kann ein Durchschnitt der Bewertungsskalen sein, wenn solche vorhanden sind, oder es können einige Meinungen ausgewählt und hervorgehoben werden. Weitere Darstellungsmöglichkeiten bestehen darin Übereinstimmungen und Unstimmigkeiten der verschiedenen Meinungen darzustellen, Meinungen Gruppen zuzuordnen, oder die Meinungen nach der Expertise der Verfasser zu gewichten.

Es gibt einige Fachbegriffe, die den Bereich, der hier dargestellt werden soll, beschreiben oder ihn eingrenzen: Subjectivity Analysis, Opinion Mining, Sentiment Analysis. In der Subjectivity Analysis (deutsch: Subjektivitätsanalyse) werden Texte, die Meinungen ausdrücken von objektiven Texten unterschieden. Im Opinion Mining werden zu einem Suchobjekt (z.B. einem Produkt) eine Liste seiner Eigenschaften und Meinungen über die Eigenschaften zusammengestellt. Der Begriff Sentiment Analysis (deutsch: Sentiment Analyse) kommt aus dem Bereich Natural Language Processing (deutsch: maschinelle Verarbeitung natürlicher Sprache oder Computerlinguistik). Unter diesen Begriff fallen Systeme, die erkennen, über was eine Meinung geäußert wird, von wem die Meinung geäußert wird oder ob die Stimmung als positiv, negativ oder neutral beschrieben werden kann. Die Fachbegriffe für das zuletzt genannte sind *sentiment polarity classification* und *polarity classification*. Im Folgenden wird dafür der Begriff Polaritätsklassifikation verwendet.

Das Thema dieser Arbeit ist Polaritätsklassifikation mit Twitter-Texten. Diese werden in drei Klassen eingeteilt: positiv, negativ und neutral.

2.1.3 Schwierigkeiten

Bevor die Sentiment Analyse angewandt wird, sollte festgestellt werden, ob die im Text ausgedrückte Meinung dem richtigen Autor zugeordnet wird. Diese Schwierigkeit tritt z.B. bei Zitaten auf.

Ein Ansatz zur Polaritätsklassifikation ist die Bestimmung der Polarität anhand von Stichwörten, die im Text vorkommen. Pang und Lee (2005) haben Sentiment Analyse mit einer Liste von Stichwörtern, die statistisch aus einem Korpus gewonnen wurden, durchgeführt. So haben sie fast 70% Genauigkeit erreicht. Dabei wurden auch Wörter als Indikator für eine Polarität entdeckt, die wenig intuitiv erscheinen, wie engl. *still* (deutsch: immer noch). Sie stellten fest, dass sich Stichwörter, die statistisch aus Trainingsdaten gewonnen werden, besser für Sentiment Analyse eignen als von Menschen ausgewählte Stichwörter. Es liegt also nahe, statistische Methoden aus dem Bereich des Maschinellen Lernens anzuwenden. In oben genannter Studie wurde mit Maschinellem Lernen eine Genauigkeit von über 80% erreicht. Trotzdem stellt sich die Frage, warum trotz der subjektiv großen Unterschiede der beiden Klassen *positiv* und *negativ* (zwei Pole eines Spektrums) die Klassifikation so schwer ist. Bei 80% Genauigkeit wird immerhin jeder fünfte Text falsch klassifiziert. Das liegt daran, dass Emotionen sehr subtil ausgedrückt werden können und die Form, in der sie geäußert werden, domain- und kontextabhängig ist. Pang und Lee nennen als Beispiel für Kontextabhängigkeit die Äußerung „go read the book". Im Kontext von Bücherbewertungen ist diese Äußerung ein starker Indikator für eine positive Bewertung. Im Kontext von Filmbewertungen ist das Gegenteil der Fall. Eine weitere Form der Kontextabhängigkeit ist die Reihenfolge, in der Hinweise auf Polaritäten geäußert werden. Sie kann trotz hoher Frequenz positiver Hinweise zu einer negativen Äußerung führen, wie folgendes Beispiel darstellt:

> This film should be *brilliant*. It sounds like a *great* plot, the actors are *first grade*, and the supporting cast is *good* as well, and Stallone is attempting to deliver a *good* performance. However, it can't hold up.

Pang und Lee (2008) Die hervorgehobenen Wörter weisen alle auf eine positive Rezension hin.

Diese Annahme wird durch den letzten Satz ins Gegenteil umgekehrt. Das ist für den menschlichen Leser einfach zu erkennen. Für ein statistisches System, das die Polarität nur von der Anzahl der Wörter, die auf eine positive bzw. negative Polarität hinweisen, abhängig macht, ist es nicht möglich.

2.2 Künstliche Neuronale Netze

Künstliche Neuronale Netze (KNNs) sind mathematische Modelle, die für Aufgaben wie Bilderkennung, Erkennung von Handschrift oder Sentiment Analyse eingesetzt werden. Nielsen (2015) Severyn und Moschitti (2015) Folgende Beschreibung von neuronalen Netzen basiert auf „Neural Networks and Deep Learning" von Nielsen (2015).

2.2.1 Beispiel für ein Neuron: einfaches Perzeptron

Der kleinste Baustein eines künstlichen neuronalen Netzes ist ein künstliches Neuron. Im Folgenden nur noch Neuron genannt. Eine einfache Ausführung eines Neurons ist das einfache Perzeptron. Das einfache Perzeptron bekommt mehrere Eingaben und gibt eine Ausgabe. Die Eingaben und die Ausgabe haben die Werte 0 oder 1. Wenn man ein einfaches Perzeptron für eine Entscheidungsfindung benutzt, steht die Eingabe 1 für eine Bedingung die zutrifft und 0 für eine Bedingung die nicht zutrifft. Abhängig davon wird eine Ausgabe gegeben, die die Antwort auf eine Entscheidungsfrage ist, die mit 0 für nein und 1 für ja beantwortet wird. Abbildung 1 zeigt ein Perzeptron mit drei Eingaben x_1, x_2, x_3.

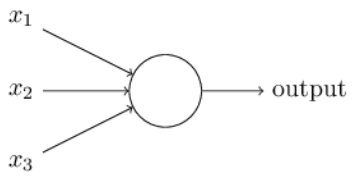

Abbildung 1: einfaches Perzeptron (Michael A. Nielsen, 2015)

Jede Eingabe wird mit einem Gewicht, das vorher festgelegt wird, multipliziert. Dann werden alle Gewichte zusammengezählt und mit einem Schwellwert verglichen. Die Ausgabe ist 1, wenn die Summe größer als der Schwellwert ist. Ist sie kleiner oder gleich dem Schwellwert, ist die Ausgabe dagegen 0. Für Gewichte w und Eingaben x berechnet sich die Ausgabe a bei Schwellwert s nach Formel 1 (nach Nielsen (2015)):

$$\mathrm{a} = \begin{cases} 0 & \text{wenn} \quad \sum_j w_j x_j \leq \mathrm{s} \\ 1 & \text{wenn} \quad \sum_j w_j x_j > \mathrm{s} \end{cases} \tag{1}$$

Der Einfluss einer Eingabe auf die Ausgabe wird durch ihr Gewicht bestimmt. Die Gewichte können angepasst werden, damit das Neuron bei einer bestimmten Eingabe eine bestimmte Ausgabe liefert. Formel 1 stellt dar, wie die Ausgabe eines Neurons in Abhängigkeit vom Schwellwert

berechnet wird. Öfter wird die Notation in Formel 2 (nach Nielsen (2015)) verwendet.

$$\mathrm{a} = \begin{cases} 0 & \text{wenn} \quad w \cdot x + b \leq 0 \\ 1 & \text{wenn} \quad w \cdot x + b > 0 \end{cases} \tag{2}$$

w und x sind Vektoren, die die Werte der Gewichte und Eingaben enthalten. Der Schwellwert wurde auf die andere Seite der Ungleichung gebracht und das Vorzeichen geändert.

Der Schwellwert wird Bias genannt. Der Bias ist ein Maß dafür wie einfach oder schwer es für ein Neuron ist 1 auszugeben. Wenn ein Neuron 1 als Ausgabe gibt, sagt man das Neuron wird aktiviert. Bei einem großen Bias ist es einfach ein Neuron zu aktivieren, da der neue Schwellwert 0 leicht überstiegen werden kann. Bei einem negativen Bias muss der Betrag der gewichteten Eingabe größer als der Bias sein, um das Neuron zu aktivieren.

Bis jetzt war die Bedingung für die Aktivierung eines Neurons das Überschreiten des Schwellwerts. Diese Bedingung kann durch andere Funktionen ersetzt werden. Diese Funktion heißt Aktivierungsfunktion. Um eine vorgegebenen Eingabe einer gewünschten Ausgabe anzupassen, möchte man, dass kleine Änderung an Gewicht und Bias nur kleine Änderungen in der Ausgabe ergeben. Das ist beim einfachen Perzeptron nicht der Fall. Hier kann eine kleine Änderung eines Gewichts oder Bias die Ausgabe von 0 nach 1 umkehren. Um mit diesem Problem umzugehen verwendet man Sigmoid-Neuronen. Diese und andere Neuronen werden im nächsten Kapitel behandelt.

2.2.2 Verschiedene Aktivierungsfunktionen für Neuronen

Neben dem einfachen Perzeptron gibt es noch andere Neuronen. Sie unterscheiden sich in der Aktivierungsfunktion. Beim einfachen Perzeptron ist das eine Stufenfunktion wie in Abbildung 2 dargestellt. In der Abbildung stehen auf der Achse, die mit Z markiert ist, die Werte, die die Summe der gewichteten Eingabe und des Bias annimmt: $Z = w \cdot x + b$. Das gilt auch für die folgenden Graphen anderer Aktivierungsfunktionen. In der abgebildeteten Grafik ist der Schwellwert 0.

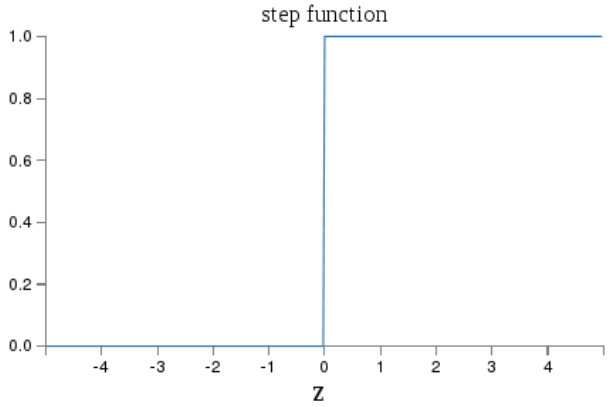

Abbildung 2: Stufenfunktion (Michael A. Nielsen, 2015)

Eine weitere Aktivierungsfunktion ist die Sigmoidfunktion. Um feinere Abstimmungen für die Ausgabe zu ermöglichen, sind beim Sigmoid-Neuron als Eingabe auch Zahlen zwischen 0 und 1 erlaubt. Die Aktivierungsfunktion σ ist in Formel 3 (übernommen von Nielsen (2015)) dargestellt.

$$\sigma(Z) = \frac{1}{1 + e^{-Z}} \tag{3}$$

In Formel 3 ist Z der Term $w \cdot x + b$: die Summe aus gewichteter Eingabe und Bias. Die Sigmoid-Funktion ist an jeder Stelle differenzierbar. Das ist wichtig für den Backpropagation-Algorithmus, einem Lernalgorithmus für KNNs. Abbildung 3 zeigt den Graphen der Sigmoidfunktion.

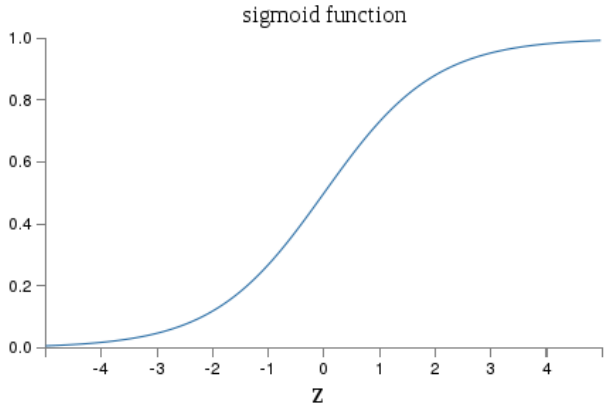

Abbildung 3: Sigmoidfunktion (Michael A. Nielsen, 2015)

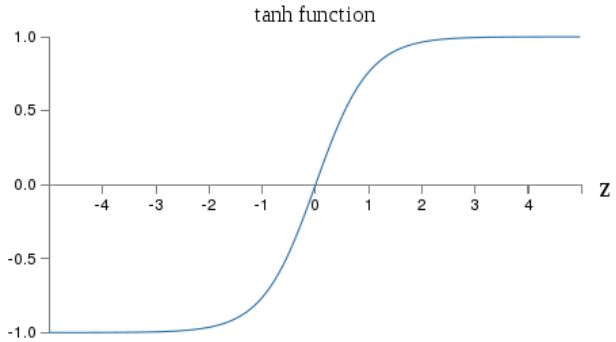

Abbildung 4: hyperbolische Tangensfunktion (Michael A. Nielsen, 2015)

Eine weitere Aktivierungsfunktion, die in Neuronen zum Einsatz kommt, ist die hyperbolische Tangensfunktion. Sie ist in Formel 4 (übernommen aus Nielsen (2015)) dargestellt. Ihr

Graph ist in Abbildung 4 dargestellt. Ein Neuron mit der hyperbolischen Tangensfunktion als Aktivierungsfunktion heißt tanh-Neuron.

$$\tanh(Z) \equiv \frac{e^{Z} - e^{-Z}}{e^{Z} + e^{-Z}} \tag{4}$$

Die Graphen der Sigmoid-Funktion und der hyperbolischen Tangensfunktion sind sehr ähnlich. Ein Unterschied ist, dass der Wertebereich bei der hyperbolischen Tangensfunktion im Bereich $[-1, 1]$ liegt, während er bei der Sigmoidfunktion in $[0, 1]$ liegt. Das muss bei der Ausgabe eines tanh-Neurons beachtet werden. Die hyperbolische Tangensfunktion ist an jeder Stelle differenzierbar.

Neben den genannten werden auch Neuronen mit einer stückweise linearen Funktion als Aktivierungsfunktion eingesetzt (engl. linear rectified unit oder rectifier). Abbildung 5 zeigt den Graphen einer stückweise linearen Funktion.
Die Gleichung für die stückweise lineare Funktion ist $f(Z) = \max(0, Z)$. Auch die stückweise lineare Funktion ist für das Gradientenverfahren und Backpropagation geeignet. Sie unterscheidet sich von der Sigmoidfunktion und der hyperbolischen Tangensfunktion dadurch, dass ihre Steigung für große positive Werte nicht 0 wird. Außerdem gibt sie für negative Werte immer 0 zurück. Die Steigung der Aktivierungsfunktion wird bei Backpropogation gebraucht. Wenn sie 0 wird, werden die Gewichte des betreffenden Neurons nicht mehr angepasst. Dieses Problem wird mit dem Einsatz der stückweise linearen Funktion umgangen. Andererseits hat sie den Nachteil, dass sie für negative Eingaben eine Steigung von 0 hat und in diesem Fall ihre Gewichte nicht angepasst werden.

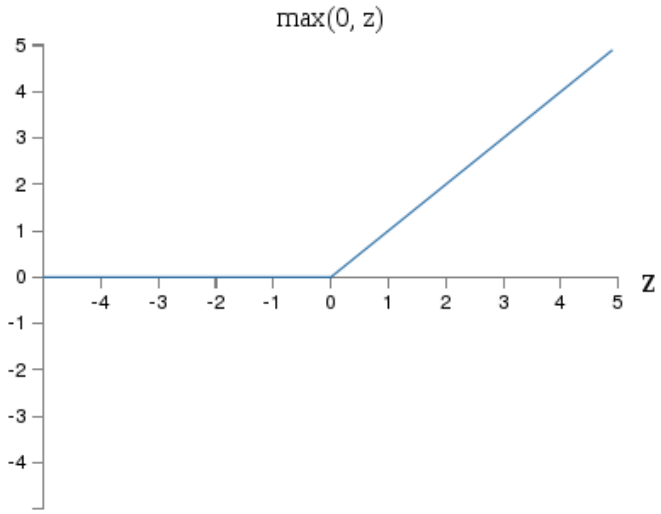

Abbildung 5: stückweise lineare Funktion (Michael A. Nielsen, 2015)

2.2.3 Schichten

Künstliche Neuronale Netze bestehen meist aus mehreren Neuronen, die in Schichten angeordnet werden. Die erste Schicht ist die Eingabeschicht, die letzte Schicht die Ausgabeschicht. Alle Schichten dazwischen sind verdeckte Schichten. Eine Möglichkeit die Schichten zu verbinden besteht darin, die Ausgaben jedes Neurons einer Schicht als Eingabe für alle Neuronen der folgenden Schicht zu verwenden. Abbildung 6 zeigt eine Skizze eines KNNs mit vier Schichten.

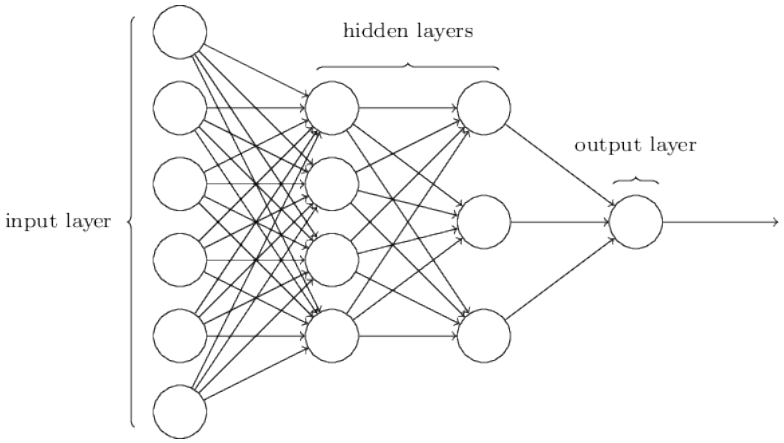

Abbildung 6: KNN mit 4 Schichten (Michael A. Nielsen, 2015)

Ein KNN, bei dem die Eingabe nur von links nach rechts weitergegeben wird, heißt feedforward-Netz. Es gibt auch Netze, in denen die Ausgabe eines Neurons wieder als Eingabe zu einem Neuron aus einem vorherigen Layer zurückgegeben wird. Solche Netze heißen rekurrente Netze.

2.2.4 Lernverfahren

Um ein KNN zu trainieren, werden Trainingsdaten, Developementdaten und Testdaten verwendet. Mit den Trainingsdaten werden Gewichte und Bias eingestellt, mit den Developementdaten werden geeignete Werte für Hyperparameter (siehe Kapitel 2.2.8) gefunden. Mit den Testdaten wird nach der Trainingsphase die Leistung des Systems gemessen. Alle drei Datensätze sind Sammlungen von Objekten wie Bildern oder Tweets, die klassifiziert werden sollen. Zu jedem Objekt ist das richtige Klassifikationsergebnis annotiert. In den Testdaten dürfen keine Beispiele aus den Trainingsdaten enthalten sein. Das Ziel ist es, nach dem Training Objekte, die nicht in den Trainingsdaten enthalten sind, korrekt zu klassifizieren. Deswegen wird die Leistung des Systems auf den Testdaten gemessen, die ungesehene Beispiele enthalten. Beim Training werden alle Trainingsbeispiele mehrere Male als Eingabe verwendet. Einen Trainingsdurchgang mit allen Trainingsbeispielen nennt man Epoche.

2.2.5 Gradientenverfahren

Um ein KNN zu trainieren, werden die Gewichte und die Bias angepasst. Bevor man Ausgaben hat, an denen man sich bei der Anpassung orientieren kann, werden sie zufällig initialisiert. Dann wird die Ausgabe mit der gewünschten Ausgabe verglichen und der Wert einer Kostenfunktion

berechnet, der ein Maß für den Abstand der Ausgabe zu der gewünschten Ausgabe darstellt. Diese Funktion wird minimiert, um das Netz zu verbessern. Dafür verwendet man das Gradientenverfahren. Dabei wird der Gradient ∇C der Kostenfunktion an der aktuellen Position v ermittelt und um einen Betrag $-\eta\nabla C$ gesenkt, wie in Formel 5 (nach Nielsen (2015)) dargestellt.

$$v' = v - \eta\nabla C \tag{5}$$

Dieser Schritt wird so oft durchgeführt bis man ein Minimum erreicht. Dabei ist η die Lernrate, die bestimmt wie groß die Schritte sind, die man nach unten geht. v' ist die neue Position, in der man sich nach Anwendung der Regel befindet. Der Gradient setzt sich aus $\partial C/\partial w$, der partiellen Ableitung der Gewichte, und $\partial C/\partial b$, der partiellen Ableitung der Bias, zusammen. In Formel 5 werden Gewichte und Bias angepasst, da sich die Position aus Werten für beide Variablen ergibt. In Formel 6 (nach Nielsen (2015)) sieht man, wie die Gewichte angepasst werden.

$$w'_k = w_k - \eta\frac{\partial C}{\partial w_k} \tag{6}$$

Ein Problem beim Gradientenverfahren ist, dass der Gradient der Kostenfunktion erst nach einer Epoche Training angepasst werden kann, wenn er als Mittel der Gradienten aller Trainingsbeispiele berechnet wird. Eine Epoche kann sehr lange dauern. Wenn dieses Problem auftritt, kann ein stochastisches Gradientenverfahren verwendet werden. Dabei wird der Gradient als Mittel der Gradienten einer zufälligen Untermenge der Trainingsdaten berechnet.

Um den Gradienten ∇C zu berechnen wird der Backpropagation-Algorithmus verwendet.

2.2.6 Konvolutionsnetze

Als Konvolutionsnetz (engl. convolutional neural network) bezeichnet man ein KNN, in dem auf einer oder mehreren Schichten eine Konvolution durchgeführt wird. Lecun et al. (1998) zeigten, dass sich Konvolutionsnetze für die Erkennung von 2-dimensionalen Formen eignen. Nielsen (2015) behauptet, dass KNNs, in denen alle Neuronen zweier Schichten verbunden sind, die räumliche Struktur von Bildern nicht berücksichtigen. In Konvolutionsnetzen sind nicht alle Neuronen komplett mit den Neuronen der folgenden Schicht verbunden. Sie berücksichtigen räumliche Strukturen. Die Eingabe von Konvolutionsnetzen sind Matrizen. Bei der Konvolution werden mehrere Werte zu einem Wert zusammengefasst. In Abbildung 7 werden z.B. mehrere Spalten einer Matrix auf einen Wert abgebildet. Dabei wird ein Filter angewandt. In Abbildung 7 ist der Filter gelb markiert. Ein Filter ist eine Matrix, die Gewichte enthält. Von den Werten in der Eingabematrix und den Gewichten hängt das Ergebnis der Konvolution ab. Bei Severyn und Moschitti (2015) ist das Ergebnis der Konvolution die Summe des elementweisen Produkts der Werte aus der Eingabematrix und der Gewichtsmatrix. Eine Schicht, auf die Konvolution abgebildet wird, heißt Konvolutionsschicht. Die Anzahl der Werte, die zusammengefasst werden, hängen von der Größe des benutzten Filters ab. Die Größe des Filters kann in beiden Dimensionen variieren. Die Höhe des Filters macht den Unterschied der eindimensionalen Konvolution (Kalchbrenner et al. (2014)) zur zweidimensionalen Konvolution (Ebert et al. (2015), Kim (2014)) aus. Siehe Kapitel 3.2 und 3.5. Da ein Filter an verschiedenen Positionen der Matrix angesetzt wird, können laut Nielsen (2015) gleiche Merkmale an verschiedenen Positionen erkannt werden.

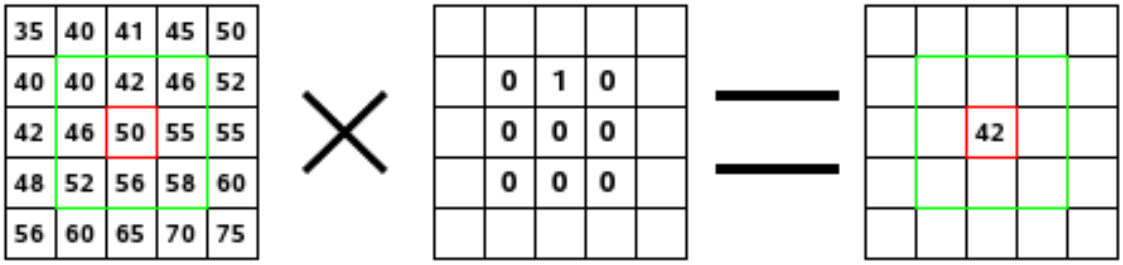

Abbildung 7: Konvolution

Auf der Konvolutionsschicht wird Max-Pooling eingesetzt. Dabei werden aus den Werten, die die Konvolution ergeben hat, der größte Wert ausgewählt. So werden nur besonders aussagekräftige Merkmale ausgewählt.

2.2.7 Softmax

Bei Klassifikationsaufgaben möchte man als Ausgabe eines neuronalen Netzes für jede Klasse eine Wahrscheinlichkeit haben. Dafür wird die Anzahl der Neuronen der Ausgangsschicht gleich der Anzahl der Klassen gewählt. Um die Ausgabe der vorherigen Schicht zu verarbeiten, nimmt man die Softmax-Funktion als Aktivierungsfunktion der Neuronen in der Ausgabeschicht. Sie ist in Formel 7 (nach Nielsen (2015)) dargestellt.

$$o_j^L = \frac{e^{Z_j^L}}{\sum_k e^{Z_k^L}} \tag{7}$$

Dabei ist o_j^L die Neuronenausgabe des Neurons j in der Ausgangsschicht L. Im Nenner wird über alle Neuronen k in L summiert. Z_j^L ist die gewichtete Eingabe des Neurons j in Schicht L. Z_k^L ist die gewichtete Eingabe des Neurons k in Schicht L. Die Neuronenausgaben aller Neuronen in der Ausgangsschicht ergeben in der Summe 1. So bekommt jede Klasse eine Wahrscheinlichkeit, die zwischen 0 und 1 liegt.

2.2.8 Hyperparameter

Hyperparameter sind Werte, die die Architektur und das Training eines KNNs beeinflussen. Durch eine gute Architektur und gute Hyperparameter für das Training kann die Leistung des KNNs verbessert werden. Hyperparameter können nicht aus den Trainingsdaten ermittelt werden. Zu den Hyperparametern gehören z.B. die Lernrate, die Anzahl der Neuronen in verdeckten Schichten und die Anzahl der Epochen. Um gute Werte für Hyperparameter zu finden, wird ein Netz auf den Trainingsdaten mit verschiedenen Hyperparameterkonfigurationen trainiert und auf den Developement-Daten getestet. Beim Vergleich der Ergebnisse kann man sehen welche Hyperparameter gut funktionieren.

3 Neuronale Netze im Natural Language Processing

Neuronale Netze werden im Natural Language Processing (NLP) z.B. für Klassifikationsaufgaben und zur Erstellung von Sprachmodellen verwendet. Bengio et al. (2003) Nach Kalchbrenner et al. (2014) eignen sich *word embeddings* (deutsch: Wortvektoren) gut als Wortrepräsentationen für den Einsatz in KNNs. Außerdem beschreiben Kalchbrenner et al. (2014) wie man mit Konvolution und Max-Pooling Merkmale ausfindig macht, die mehrere Wörter umfassen.

3.1 Wortrepräsentationen

Um natürliche Sprache maschinell zu verarbeiten muss eine geeignete Darstellung für Wörter gefunden werden. Diese kann als String oder in 1-aus-n-Code erfolgen. In beiden Fällen ist die Darstellung unabhängig von ihrem Kontext. Ähnlichkeiten können bei der Darstellung durch Strings durch die äußere Form oder weitere Verarbeitung wie POS-Tagging erkannt werden. Eine andere Form der Darstellung sind *word embeddings*, die im Folgenden Wortvektoren genannt werden. Bei der Darstellung eines Wortes durch einen Wortvektor wird der Kontext berücksichtigt. Außerdem kann man anhand der Vektoren Ähnlichkeiten zweier Wörter erkennen. Diese Ähnlichkeiten können syntaktisch oder semantisch sein. Vom Abstand der Vektoren kann auf ihre Ähnlichkeit geschlossen werden. Das zeigt auch folgendes Zitat von Mikolov et al. (2013b).

> For example, the male/female relationship is automatically learned, and with the induced vector representations, “King - Man + Woman” results in a vector very close to “Queen.”

Im Folgenden wird das Skip-Gram-Modell aus Mikolov et al. (2013a) beschrieben, das Wortvektoren aus sehr großen Datenmengen erstellt. Das Modell ist in Abbildung 8 dargestellt.

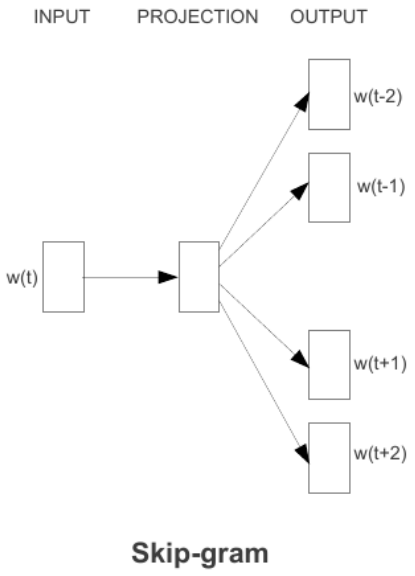

Abbildung 8: Skip-Gram-Modell (Mikolov et al. (2013a))

Beim Skip-Gram-Modell werden Wörter im Kontext eines gegebenen Wortes vorhergesagt. Die Vektoren der vorhergesagten Wörter werden mit Vektoren, die im Kontext des gegebenen Wortes vorkommen, verglichen und angepasst. Für Wörter mit ähnlichem Kontext ergeben sich ähnliche Wortvektoren.

Im Folgenden wird die Verwendung von Wortvektoren im linguistisch-informierten Konvolutionsnetz (lingCNN) von Ebert et al. (2015) beschrieben. Um für jedes Wort eines zu klassifizierenden Tweets einen Wortvektor zu finden, wurde eine Tabelle $LT \in \mathbb{R}^{|V|}$ erstellt (LT steht für Lookup-Table). V steht für das Vokabular. LT besteht aus zwei Matrizen P und Q: $LT = \begin{bmatrix} P \\ Q \end{bmatrix}$. $P \in \mathbb{R}^{d_p \times |V|}$ ist eine Matrix, in der sich Wortvektoren der Länge d_p befinden.

Diese wurden mit dem Skip-Gram-Modell aus Mikolov et al. (2013a) aus Twitter-Daten erstellt. Die Daten stammen aus dem Twitter Events Data Set (McMinn et al. (2013)) und den SemEval Trainingsdaten von 2013 (Nakov et al. (2013)). Das Vokabular besteht aus allen Wörtern aus den SemEval Trainingsdaten und den 50000 Wörtern mit der höchsten Frequenz aus dem Events Data Set. Die Wortvektoren haben 60 Dimensionen. Es wurde ein unbekanntes Wort *unknown* zum Vokabular hinzugefügt, um eine Repräsentation für ungesehene Wörter zu haben. Die Wortvektoren können während des Trainings verändert werden. So passen sie sich der vorliegende Aufgabe an. In diesem Fall ist das die Sentiment Analyse.

Der untere Teil von LT ist Q. In $Q \in \mathbb{R}^{d_Q \times |V|}$ befinden sich Vektoren, in denen pro Wort d_Q Merkmale gespeichert sind. Diese Merkmale werden durch Abgleich der Wörter mit Polaritätslexika und Nachschlagen von Werten für Wörter in Polaritätslexika gebildet. Ein Wortmerkmal ist ein binärer Sentiment-Indikator (engl. binary sentiment indicator). Dabei werden für jedes Wort zwei Binärstellen pro Lexikon gebildet, die darstellen, ob ein Wort im Lexikon positiv oder negativ annotiert ist. Für die binären Sentiment-Indikatoren wurden drei Polaritätslexika verwendet: MPQA (Wiebe und Cardie (2005)), opinion lexicon (Hu und Liu (2004)) und NRCC emotion lexicon (Mohammad und Turney (2013)). Eine weitere Art von Wortmerkmalen sind Sentiment-Werte (engl. Sentiment-Scores), die in einem Polaritätslexikon stehen. Zu jedem Wort ist dort ein Wert für jedes Sentiment angegeben. Die Punktzahlen wurden aus dem Sentiment140-Lexikon und dem Hashtag-Lexikon (Mohammad und Kiritchenko (2015)) entnommen. Ein weiteres Wortmerkmal ist binäre Negation (engl. binary Negation). Dabei wird jedes Wort zwischen einem Negationswort und der nächsten Punktuation als negiert markiert.

3.2 Konvolution

Um neuronale Netze für Aufgaben wie Polaritätsklassifikation zu verwenden ist es hilfreich, wenn man nicht nur Wortmerkmale, sondern auch Merkmale, die aus mehreren Wörtern bestehen, verwendet. Um Merkmale, die mehrere Wörter berücksichtigen, zu extrahieren, benutzt man Konvolution. Die Eindimensionale Konvolution nach Kalchbrenner et al. (2014) ist im Folgenden dargestellt. Die Konvolution ist eindimensional, weil immer nur eine Zeile eines Wortvektors für die Konvolution relevant ist. Zweidimensionale Kovolution ist in Kapitel 3.5 beschrieben. Dort wird auf alle Zeilen mehrerer Vektoren Konvolution angewandt. Bei Kalchbrenner et al. (2014) ist ein Satz als Sequenz $f \in \mathbb{R}^f$ dargestellt. $f_i \in \mathbb{R}^f$ ist ein Merkmal für das Wort an Stelle i in Satz f. $m \in \mathbb{R}^m$ ist ein Gewichtsvektor. Für die Konvolution wird das Skalarprodukt des Gewichtsvektors und jedes m-Grams des Satzes ausgerechnet. Das ist in Formel 8 (nach

Kalchbrenner et al. (2014)) dargestellt.

$$c_j = m \cdot f_{j-m+1:j} \tag{8}$$

$f_{j-m+1:j}$ steht für einen Abschnitt des Satzes f. Die Länge des Abschnitts ist abhängig von der Filterbreite und dem Parameter j. Wenn man $f \geq m$ wählt und j im Bereich $[m, j]$ liegt, nennt man das *narrow convolution* (deutsch: enge Konvolution). Bei enger Konvolution setzt man den Filter ganz links vom Satz an und schiebt ihn mit wachsendem j jeweils eine Position nach rechts. Bei *wide convolution* (deutsch: breite Konvolution) kann m beliebig gewählt werden und j liegt in $[1, f + m - 1]$. Die m-Gram-Größe kann so auch länger als der Satz gewählt werden und die Position, an der der Filter angesetzt wird, kann außerhalb des Satzes liegen. Enge und breite KOnvolution sind in Abbildung 9 dargestellt. Wenn Teile des Filters sich außerhalb der

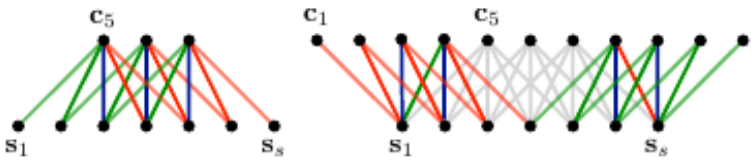

Abbildung 9: enge Konvolution und breite Konvolution (Kalchbrenner et al. (2014))

Satzlänge befinden, werden die fehlenden Werte mit 0 interpretiert. So werden mit verschiedenen Filterbreiten m Gewichte für N-Gramme der Länge m oder kleiner (wenn der Filter über die Satzgrenze hinausragt) gelernt. Bei breiter Konvolution wird jedes Gewicht im Filter für jedes Wort gelernt. Das ist bei enger Konvolution nicht der Fall.

3.3 k-Max-Pooling

k-Max-Pooling ist eine Modifikation von Max-Pooling. Anstatt nur den größten Wert aus einer Reihe von Werten auszuwählen, werden die k größten Werte ausgewählt. So bleiben Merkmale, die auch große Werte haben, erhalten. Es können dabei allerdings auch unwichtige Merkmale aufgenommen werden.

3.4 Architektur

Das lingCNN besteht aus vier Schichten. Der Eingangsschicht, einer Konvolutionsschicht, einer Pooling-Schicht und der Ausgangsschicht. Die Eingangsschicht besteht aus den konkatenierten Wortvektoren der oben genannten Tabelle LT, die jedem Token aus dem gegebenen Tweet einen Vektor zuteilt. Die Eingabe erfolgt in Form einer Matrix R. Siehe Formel 9 (übernommen von Ebert et al. (2015)).

$$R = \begin{bmatrix} | & | & | \\ LT_{.,t_1} & \cdots & LT_{.,t_n} \\ | & | & | \end{bmatrix} \tag{9}$$

Dabei steht t_1 für das erste Token im Tweet und t_n für das n-te Token. Auf der Eingangsschicht wird mit einer Anzahl von Filtern mit der Filtermatrix $M \in \mathbb{R}^{d \times m}$ eine zweidimensionale

Konvolution durchgeführt. d steht für die Länge der Vektoren und m für die Filterbreite. Die Konvolution erfolgt nach Formel 10 (übernommen von Ebert et al. (2015)).

$$u_h^{(1)} = \sum_{i=1}^{d} \sum_{j=1}^{m} M_{i,j} Z_{i,h+j} \tag{10}$$

Dabei wird die Filtermatrix M je nach Filterposition h mit einem anderen m-Gram aus der Tokensequenz multipliziert. (1) steht für die erste Schicht im KNN. Es wird breite Konvolution benutzt, was auch in Abbildung 10 zu sehen ist. Es werden verschiedene Filterbreiten und mehrere Filter pro Filterbreite eingesetzt. Aus der Konvolutionsschicht werden die größten Werte mit Max-Pooling in der zweiten Schicht gesammelt. Die Neuronen in der Pooling-Schicht sind ReLUs.

Zur Ausgabe der zweiten Schicht $u^{(2)}$ werden Satzmerkmale s konkateniert $[u^{(2)}s]$. Satzmerkmale sind Frequenzen für großgeschriebene Wörter, Wörter mit Buchstabenwiederholungen, Emoticons, Punktuationswiederholungen und negierte Wörter. Auf Satzebene werden aus den vorher erwähnten Sentiment-Werten vier Merkmale für Wörter errechnet: Anzahl der Wörter, die eine Sentiment-Wert haben, Summe der Sentiment-Werte der Einzelwörter, höchster Sentiment-Wert der Wörter und der Sentiment-Wert des letzten Wortes. Für die Berechnungen werden die fünf oben erwähnten Polaritätslexika eingesetzt. Die vier Merkmale werden außerdem für Part-of-Speech Tags, Hashtags und großgeschriebene Wörter berechnet.

In der Ausgangsschicht läuft die Ausgabe der Pooling-Schicht, die mit den Satzmerkmalen konkateniert wurde, durch die Softmax-Funktion und liefert eine Wahrscheinlichkeitsverteilung über die drei Polaritätsklassen.

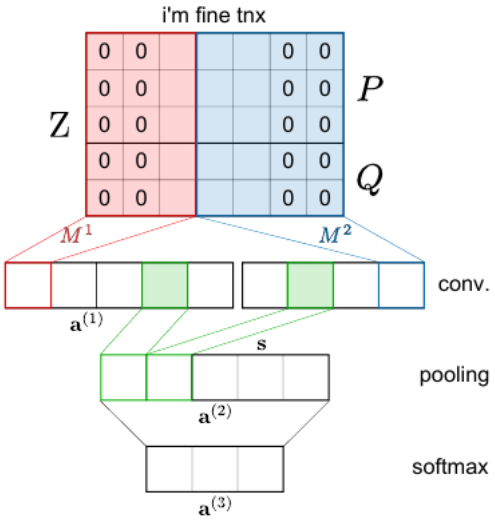

Abbildung 10: Linguistisch-informiertes Konvolutionsnetz Ebert et al. (2015)

3.5 Überanpassung, Regularisierung, Dropout

Ein neuronales Netz soll aufgrund von Merkmalen, die aus Trainingsdaten gelernt wurden, eine Entscheidung treffen. Dabei können Merkmale gelernt werden, die nur in den Trainingsdaten mit einer Klasse korrelieren. Diesen Vorgang nennt man Überanpassung. Um Überanpassung zu vermeiden, wird Regularisierung und Dropout verwendet. Eine Art der Regularisierung ist L2-Regularisierung, auch genannt *weight decay* (deutsch ungefähr: Gewichtsdämfpung). Bei L2-Regularisierung wird an die Kostenfunktion ein Term angehängt, der Einfluss darauf nimmt wie sich die Gewichte im Laufe des Trainings verändern. Eine L2-regularisierte Kostenfunktion ist in Formel 11 (übernommen und verändert aus Nielsen (2015)) zu sehen.

$$C = C_0 + \lambda \sum_w w^2 \tag{11}$$

Dabei ist C_0 eine Kostenfunktion, w steht für ein Gewicht und λ ist das Regularisierungs-Parameter. Durch den addierten Term wird die Kostenfunktion abhängig vom Wert der Gewichte. Das bedeutet, dass mit größeren Gewichten die Kosten größer werden. Ein KNN mit L2-Regularisierung wird also eher dazu neigen kleine Gewichte zu lernen, weil so die Kosten verringert werden. Die Stärke dieser Neigung kann mit dem Regularisierungs-Parameter eingestellt werden. Bei einem großen λ liefern die Gewichte einen größeren Beitrag zur Kostenfunktion und werden so beim minimieren der Kosten noch weiter verringert. Das Lernen von kleineren Gewichten führt dazu, dass das KNN besser generalisiert. Nielsen (2015)

Nielsen (2015) erwähnt Dropout als weiteren Ansatz, um Überanpassung zu vermeiden. Beim Dropout werden während des Trainings eine bestimmte Anzahl von Neuronen in einer verdeckten Schicht aus- und wieder eingeschaltet. Bevor Trainingsbeispiele eingegeben und die Gewichte nach z.B. einem Mini-Batch angepasst werden, werden einige Neuronen ausgeschaltet. Nachdem die Gewichte der übrigen Neuronen angepassten wurden, werden die vorher ausgeschalteten Neuronen wieder eingeschaltet und zufällig eine neue Menge von Neuronen gewählt, die für den nächsten Trainingsschritt ausgeschaltet werden. Dieser Vorgang geschieht nach jeder Aktualisierung der Gewichte und Bias. Wenn das Training beendet ist, werden alle Neuronen für die Klassifikation verwendet. Diese Vorgehensweise ähnelt dem Vorgang mehrere Netze zu trainieren und über die Ergebnisse zu mitteln. Nielsen (2015) Auch so kann Überanpassung vermieden werden.

4 Experimente

Es wurden mit dem System aus Ebert et al. (2015) Experimente durchgeführt, die zeigen sollen, welche Hyperparameter gute Ergebnisse liefern. Die verwendeten Wortvektoren und die Architektur wurde in 3.4 und 3.5 beschrieben. Es folgt eine Beschreibung der verwendeten Daten, der verwendeten Hyperparameter und die Auswertung der Ergebnisse.

4.1 A Linguistically Informed Convolutional Neural Network

Ebert et al. (2015) beschreiben ein Konvolutionsnetz, das Sentiment Analyse für Tweets durchführt. Ein Tweet ist eine Nachricht auf der Internetplattform Twitter. Sie enthält maximal 140 Zeichen. In Tweets werden gerne Hashtags und Emoticons benutzt. Im Folgenden werden das Preprocessing und die verwendeten Daten beschrieben.

4.1.1 Preprocessing

Ein Tweet nach dem Preprocessing sieht so aus:

> <user><user>#surface the forida mall , orlando , f1 is the closest $msft store to me . ;-) can't wait to be there tomorrow !

Im Preprocessing wurden die Wörter in den Tweets tokenisiert, alle Benutzernamen durch <user> ersetzt und alle URLs durch <web>. Außerdem werden alle Großbuchstaben in Kleinbuchstaben übersetzt und es werden POS-Tags annotiert. Folgen wiederholter Punktuation wie „???!!!!“ wurden durch eine Liste der vorkommenden Punktuationszeichen ersetzt.

4.1.2 Daten

Die Trainingsdaten stammen aus dem SemEval 2013 Task 2: *Sentiment Analysis in Twitter* Task B: Polaritätsklassifikation. Nakov et al. (2013) Über ein Jahr wurden Millionen Tweets gesammelt. Beliebte Themen wurde gefunden, indem Named-Entities mit hoher Frequenz gesucht wurden. Aus diesen Themen wurden Tweets, die Sentiment enthalten, ausgewählt. Dafür wurden die Wörter der Tweets mit einem Sentimentlexikon verglichen. Die Daten wurden über Mechanical Turk annotiert. Dabei wurde jedes Wort, das auf ein Sentiment hinweist, markiert und mit positiv/negativ/neutral annotiert. Außerdem wurde eine Polarität für den ganzen Tweet vergeben.

Für die Experimente wurden als Trainingsdaten die Trainingsdaten und die Developementdaten von SemEval 2013 benutzt. Als Developementdaten wurde die Testdaten von SemEval 2013 und die Testdaten des Sentiment140-Korpus benutzt. Statistiken über die Daten für Task B befinden sich in Tabelle 2.

	total	**pos**	**neg**	**neu**
training set	9845	3636	1535	4674
developement set	3813	1572	601	1640
SemEval test set	2390	1038	365	987
Sent140 test set	498	182	177	139

Tabelle 2: verwendete Daten (Ebert et al. (2015))

4.2 Parameterstudie

Parameter	**Wert 1**	**Wert 2**	**Wert 3**	**Wert 4**
Lernrate	0,1	0,01	0,001	
Filteranzahl	100	200	300	400
Filterbreite	2			
Max-Pooling	1	5		
Regularisierungsparameter	0.00001	0,00005	0,0001	0,0005
Dropout	0	0,5	0,8	
verdeckte Neuronen	0	100	500	

Tabelle 3: getestete Parameter und Werte

Es wurden Modelle mit verschiedenen Hyperparametern auf den Trainingsdaten trainiert und auf den Developementdaten evaluiert. In Tabelle 3 sind die getesteten Parameter aufgelistet. Wenn der Parameter *verdeckte Neuronen* einen Wert größer als 0 hat, gibt es zwischen Pooling-Schicht und und Ausgangsschicht eine weitere Schicht, in der alle Neuronen komplett mit der vorherigen und nachkommenden Schicht verbunden sind. Die Anzahl der Neuronen in der Schicht ist der Wert des Parameters *verdeckte Neuronen*. Die Dropout-Rate wird auf die verdeckten Neuronen angewandt. Wenn es keine verdeckten Neuronen gibt, wird die Dropout-Rate auf die Konvolutionsschicht angewandt. Aus den Werten für die sieben Parameter wurde jede Kombination getestet. Die besten und schlechtesten Kombinationen für das SemEval 2013 Testdaten und den Sentiment140 Korpus (Go et al. (2011)) befinden sich in den Tabellen 4 und 5. Auf dem SemeEval-Korpus ist der Abstand zwischen höchstem und niedrigstem F1-Maß 20%. Daran sieht man, dass es sich lohnt die Hyperparameter gut einzustellen. Auf dem Sentiment140-Korpus fällt der Unterschied mit 5% geringer aus.

Parameter	Höchtes F1-Maß	Niedrigstes F1-Maß
Lernrate	0,01	0,001
Filteranzahl	300	100
Filterbreite	2	2
Max-Pooling	5	1
Regularisierungsparameter	0,00001	0,0005
Dropout	0,5	0,8
verdeckte Neuronen	100	0
F1	**0,71**	**0,51**

Tabelle 4: höchste und niedrigste F1-Maße auf den SemEval 2013 Testdaten

Parameter	Höchtes F1-Maß	Niedrigstes F1-Maß
Lernrate	0,1	0,001
Filteranzahl	300	100
Filterbreite	2	2
Max-Pooling	5	1
Regularisierungsparameter	0,0001	0,0005
Dropout	0,5	0,8
verdeckte Neuronen	100	0
F1	**0,82**	**0,67**

Tabelle 5: höchste und niedrigste F1-Maße auf dem Sentiment140-Korpus

4.3 Boxplots

Um die Ergebnisse verschiedener Parameterwerte zu vergleichen, wurden Boxplots aus den Ergebnissen auf den SemEval 2013 Testdaten für alle getesteten Parameter erstellt. Ein Boxplot ist eine grafische Darstellung der Verteilung eines Wertes in Abhängigkeit von einem anderen Wert. Für die Parameterstudie wird das F1-Maß in Abhängigkeit von den verschiedenen Werten eines Parameters dargestellt. Ein Boxplot setzt sich aus charakteristischen Werten für die Verteilung zusammen: dem Median, dem oberen und unteren Quartil, zwei Antennen und den Ausreißern. Die Länge der Antennen ergibt sich aus dem Produkt eines Faktors und des Interquartilsabstands. Diese Länge wird vom oberen Quartil nach oben und vom unteren Quartil nach unten angetragen. Der größte Wert innerhalb dieser Länge wird als Antenne markiert. Alle Werte über der oberen und unter der unteren Antenne werden als Ausreißer bezeichnet.

Für die Erstellung der Grafiken wurde die Bibliotheken matplotlib[1], pandas[2] und seaborn[3] verwendet. Der Faktor, der die Länge der Antennen bestimmt, ist 1,5[4]. Die Spannweite ist der gesamte Bereich des Datensatzes. Zu dieser Größe zählen auch die Ausreißer. Einen Überblick über die Verteilungen des F1-Maßes bei allen Experimenten bieten die Abbildungen 11 und 12.

[1] `http://matplotlib.org/`
[2] `http://pandas.pydata.org/`
[3] `http://stanford.edu/~mwaskom/software/seaborn/`
[4] `http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.boxplot` besucht am 18.10.2015 14:44

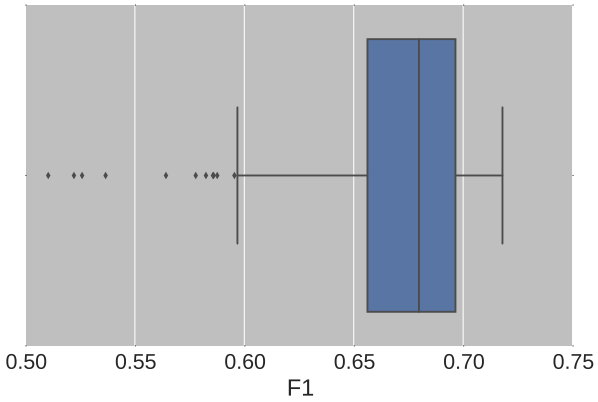

Abbildung 11: Verteilung der F1-Maße aller Experimente auf dem SemEval 2013 Test-Set

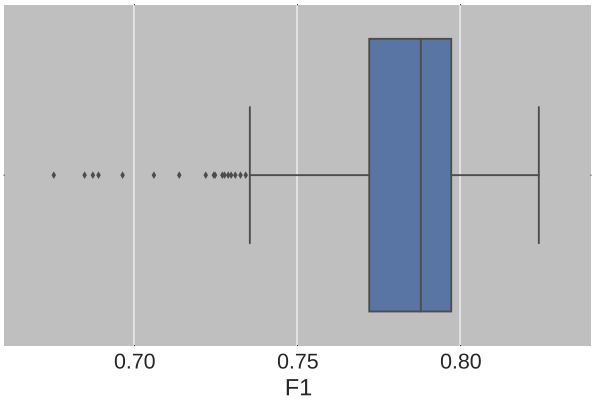

Abbildung 12: Verteilung der F1-Maße aller Experimente auf dem Sentiment140-Korpus

4.3.1 Lernrate

Die Lernrate wurde mit den Werten 0,1, 0,01 und 0,001 getestet. Der Boxplot (Abbildung 13) zeigt, dass alle charakteristischen Werte (Median, Maximum, Minimum, oberes und unteres Quartil) bei der Lernrate 0,01 die höchsten F1-Maße haben. Bei 0,1 sind diese Werte kleiner und bei 0,001 noch kleiner. Bei der Lernrate 0,01 haben die F1-Maße die kleinste Spannweite, bei 0,1 die zweitkleinste und bei 0,001 die größte. Eine kleine Spannweite wird als positives Merkmal interpretiert, weil die Streuung des F1-Maßes geringer ist. Auch der Interquartilsabstand ist bei 0,01 am kleinsten, bei 0,1 am zweitkleinsten und bei 0,001 am größten. Alle drei Lernraten haben Ausreißer nach unten. Bei 0,1 und 0,01 liegen die Ausreißer nah an der unteren Antenne. Bei 0,001 sind sie sehr weit von der Spitze unteren Antenne entfernt.

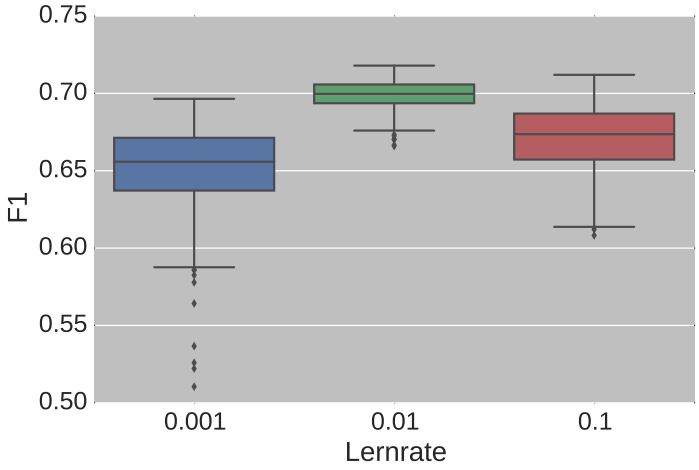

Abbildung 13: Boxplot des F1-Maßes für verschiedene Lernraten

Aufgrund der höheren Werte des Medians, der Extremwerte, des oberen und unteren Quartils, der kleineren Spannweite und des kleineren Interquartilsabstand schneidet bei dieser Experimentreihe der Wert 0,01 für die Lernrate am besten ab.

4.3.2 Filteranzahl

Für die Filteranzahl wurden die Werte 100, 200, 300 und 400 getestet. Im Boxplot in Abbildung 14 ist zu sehen, dass die oberen Antennen aller vier Werte sehr nah zusammen liegen. Der Wert der Spitze der oberen Antenne von 100 liegt im Vergleich mit den anderen niedriger. Bei den unteren Antennen gibt es größere Unterschiede. Die höchste untere Antenne hat die Filteranzahl 400. Dahinter liegen mit wenig Abstand zur nächsthöchsten unteren Antenne 300 und 200. Die untere Antenne von 100 liegt relativ weit zurück. Bei allen vier Verteilungen gibt es Ausreißer nach unten. Bei 300 und 400 liegen die Ausreißer näher an der unteren Antenne als bei 200. Bei 100 sind die Ausreißer sehr weit von der unteren Antenne entfernt.

Die oberen Quartile liegen sehr nah beieinander, wobei 400 das höchste obere Quartil hat und 100 das niedrigste. Bei den unteren Quartilen gibt es größere Unterschiede. Die unteren Quartile von 400 und 300 liegen sehr nah beieinander. Dahinter folgen mit größerem Abstand 200 und 100. Die kleinste Spannweite hat 400, dicht gefolgt von 300. 200 und 100 haben beide deutlich größere Spannweiten.

Beim Interquartilsabstand sind 400 und 300 sehr ähnlich und 200 und 100 folgen in größerem Abstand. Die Mediane von 300 und 400 liegen fast auf der selben Höhe, danach folgen 200 und 100. Die Werte 100 und 200 haben größere Spannweite, größeren Interquartilsabstand und

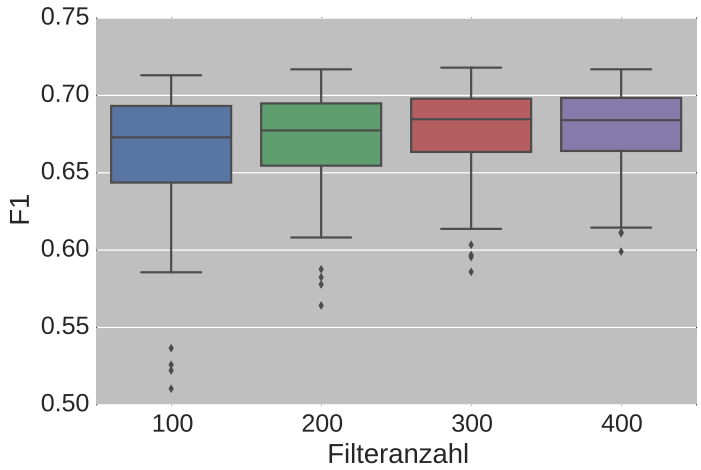

Abbildung 14: Boxplot des F1-Maßes für verschiedene Filteranzahlen

kleinere Mediane als 300 und 400. Die Maxima sind bei allen vier sehr nah zusammen, aber auch hier sind die von 300 und 400 höher als die der anderen zwei. Beim Vergleich zwischen 300 und 400 sind die Werte für Maximum, Interquartilsabstand und Median fast identisch. Der Abstand zwischen den Antennen und die Spannweite ist bei 400 geringer als bei 300. Es kann festgestellt werden, dass die Erhöhung der Filteranzhal von 100 auf 200 und von 200 auf 300 Verbesserungen bringen. Ob die Verteilung bei der Filteranzahl von 400 der Verteilung bei der Filteranzahl von 300 vorzuziehen ist, ist streitbar. Der Median liegt bei 300 höher, dafür gibt es bei 400 weniger Ausreißer.

4.3.3 Max-Pooling

Für Max-Pooling wurden die Werte 1 und 5 getestet. Der Boxplot (Abbildung 15) zeigt, dass die Spannweite bei 1 deutlich größer ist als bei 5. Das liegt vor allem an den Ausreißern nach unten, die bei 1 sehr weit von der unteren Antenne entfernt sind. Bei 5 gibt es auch Ausreißer nach unten. Hier sind sie aber viel näher an der unteren Antenne. Auch der Interquartilsabstand ist bei 5 kleiner als bei 1. Dieser Unterschied ergibt sich hauptsächlich durch das höhere untere Quartil von 1. Die oberen Quartile sind sehr nah zusammen, wobei das von 1 höher liegt. Das Maximum ist bei 5 minimal höher.

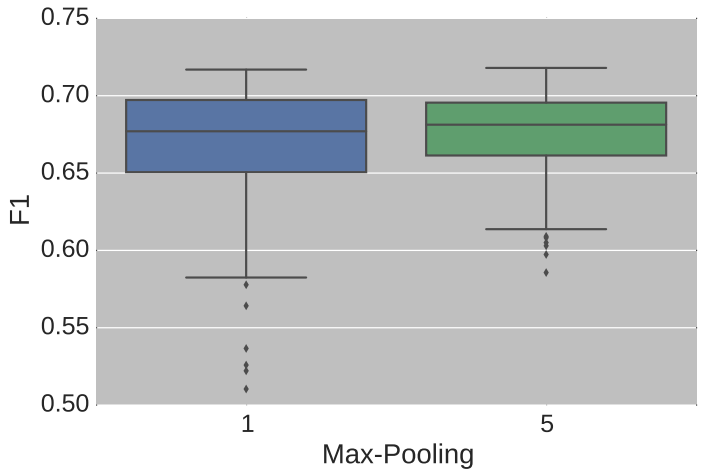

Abbildung 15: Boxplot des F1-Maßes bei verschiedenen Werten für Max-Pooling

Aufgrund der kleineren Spannweite, des kleineren Interquartilsabstands und des höheren Medians ist der Max-Pooling Wert von 5 dem von 1 vorzuziehen.

4.3.4 L2-Regularisierung

Als Regularisierungsparameter bei der L2-Regularisierung wurden die Werte 0,0005, 0,0001, 0,00005 und 0,00001 getestet. Beim Boxplot (Abbildung 16) fallen bei allen Verteilungen die Ausreißer auf. Jeweils ein Ausreißer ist sehr weit von der unteren Antenne entfernt. Die größte Spannweite hat 0,0005. Hier ist auch der Abstand des Minimums zur unteren Antenne am größten. Die Spannweiten der anderen drei Werte sind ähnlich groß. Je kleiner das Regularisierungsparameter, desto kleiner die Spannweite.

Die Abstände zwischen den Werten der Maxima sind klein. Dasselbe gilt für die oberen Quartile. Die unteren Antennen haben größere Abstände. Bei 0,0005 ist der Wert der unteren Antenne am größten. Die unteren Quartile wachsen mit der Größe des Regularisierungsparameters. Auch der Median wird mit größerem Regularisierungsparameter größer. Der Interquartilsabstand wird mit größerem Regularisierungsparameter kleiner.

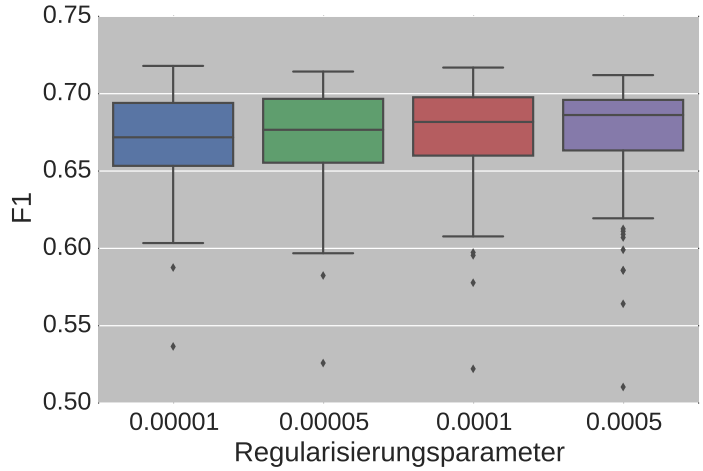

Abbildung 16: Boxplot des F1-Maßes bei verschiedenen Werten für das Regularisierungsparameter

In der Testreihe der Regularisierungsparameter hat sich gezeigt, dass mit größeren Regularisierungsparametern der Median und die Spannweite größer werden und der Interquartilsabstand kleiner wird. Für Maxima ist keine solche Tendenz erkennbar.

4.3.5 Dropout

Für Dropout wurden die Werte 0,0, 0,5 und 0,8 getestet. Der Boxplot (Abbildung 17) zeigt, dass eine Dropout-Rate von 0,5 im Vergleich zu 0,0 Verbesserungen beim Maximum, Median, oberer und unterer Antenne und beim oberen und unteren Quartil bringt. Der Interquartilsabstand und die Spannweite wurden größer. Bei der Änderung von 0,5 auf 0,8 nahmen Maximum, obere Antenne, untere Antenne, unteres Quartil und Median ab. Allein das obere Quartil wurde größer. Der Interquartilsabstand und die Spannweite wurde größer. Nur bei 0,8 gibt es Ausreißer nach unten.

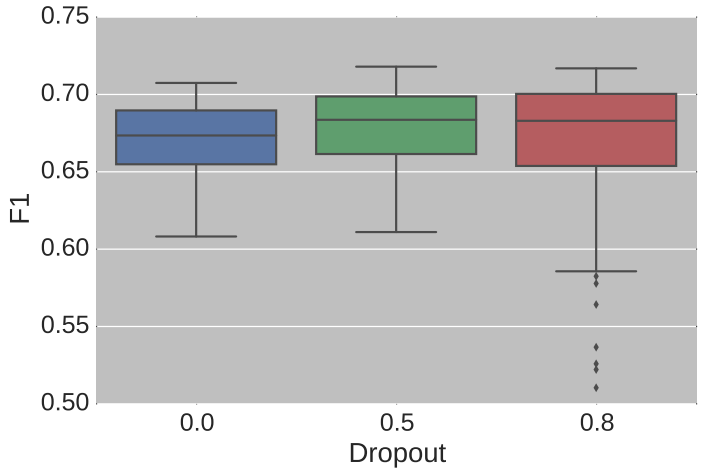

Abbildung 17: Boxplot des F1-Maßes bei verschiedenen Werten für Dropout

Die Dropout-Rate von 0,5 zeigt bessere Ergebnisse als gar kein Dropout. Bei 0,8 sind die Ergebnisse schlechter als bei 0,5. Möglicherweise gibt es zwischen 0,5 und 0,8 Dropout-Raten, die besser funktionieren als 0,5.

4.3.6 Verdeckte Neuronen

Für die Anzahl der verdeckten Neuronen wurden die Werte 0, 100 und 500 getestet. Der Einsatz von 100 verdeckten Neuronen zeigt im Vergleich mit 0 verdeckten Neuronen Verbesseurngen beim Median, dem unteren Quartil, der unteren Antenne und den Ausreißern. Der Median, das untere Quartil und die untere Antenne wurden größer. Die Anzahl der Ausreißer wurde geringer und die Spannweite kleine. Auch der Interquartilsabstand wurde kleiner. Bei einer weiteren Erhöhung der verdeckten Neuronen auf 500 verschlechterte sich der Median und das obere und untere Quartil. Der Interquartilsabstand wurde geringer und die untere Antenne liegt höher.

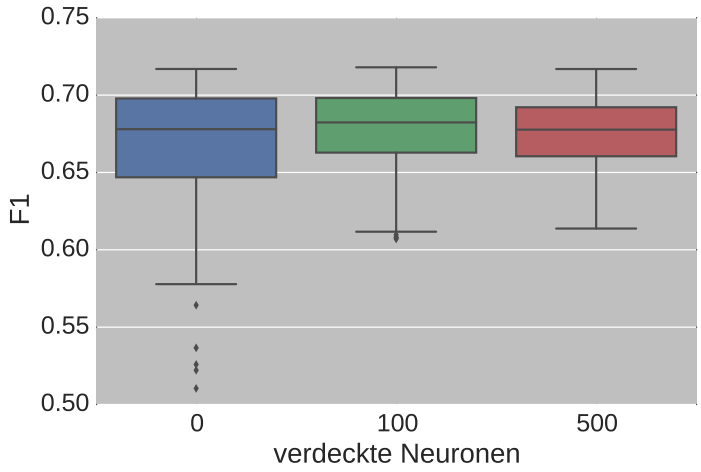

Abbildung 18: Boxplot des F1-Maßes bei verschiedener Anzahl verdeckter Neuronen

Aufgrund der besseren Werte der beiden Quartile und des Medians schneidet der Wert 100 für die Anzahl der verdeckten Neuronen in diesem Vergleich am besten ab.

4.4 Heatmaps

Um die gegenseitige Wirkung von Parameterpaaren zu untersuchen wurden Heatmaps für alle Parameterpaare erstellt. Dafür wurde der Durchschnitt der F1-Maße für alle Experimente berechnet, in denen die entsprechende Parameterkombination benutzt wurde.

4.4.1 Dropout und Regularisierungsparameter

Die Heatmap von Dropout und Regularisierungsparamter (Abbildung 19a) zeigt, dass bei den Dropout-Raten 0,0 und 0,5 jede Erhöhung des Regularisierungsparameters zu einem besseren F1-Durchschnitt führt. Bei der Dropout-Rate 0,8 wird der F1-Durchschnitt bei den ersten drei Werten für das Regularisierungsparameter noch besser. Danach wird er schlechter. Das beste durchschnittliche F1-Maß wird mit der Kombination der Dropout-Rate 0,5 und des Regularisierungsparameters 0,0005 erreicht. An dieser Grafik sieht man, dass es sich lohnt Dropout und L2-Regularisierung zu kombinieren, um Überanpassung zu vermeiden. Außerdem lässt sich erkennen, dass sich die beiden Methoden gegenseitig ergänzen.

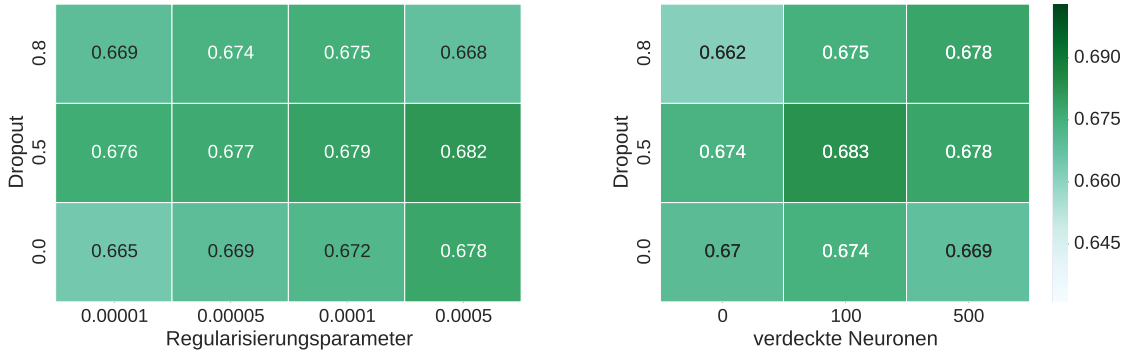

Abbildung 19: Heatmap von Dropout und Regularisierungsparameter und Dropout und verdeckten Neuronen

4.4.2 Dropout und verdeckte Neuronen

Die Heatmap von Dropout und verdeckten Neuronen (Abbildung 19b) zeigt, dass das höchste durchschnittliche F1-Maß mit der Kombination der Dropout-Rate 0,5 und der Anzahl der verdeckten Neuronen von 100 erreicht wurde. Bei 0 verdeckten Neuronen und der Dropout-Rate 0,8 wurde der schlechteste Durchschnitt gemessen. Bei gar keinem Dropout, also Dropout-Rate 0,0 wurde für alle Anzahlen von verdeckten Neuronen der schlechteste F1-Durchschnitt erreicht. Bei der hohen Anzahl von 500 verdeckten Neuronen ist das Ergebnis ohne Dropout schlechter als mit Dropout. Daran sieht man, dass ein KNN mit vielen Neuronen zur Überanpassung neigt. Dem wird mit Dropout entgegengewirkt. Bei 100 verdeckten Neuronen fällt der Wert ohne Dropout nicht so stark gegenüber denen mit Dropout ab. Hier ist die Überanpassung geringer. Trotzdem verbessert die Verwendung der Dropout-Rate von 0,5 den F1-Durchschnitt um 0,9%.

4.4.3 Filteranzahl und Dropout

Die Filteranzahl und die Dropout-Rate sind schwer in Verbindung zu bringen, weil sie unter Umständen auf verschiedenen Schichten des Neuronalen Netzes zum Tragen kommen. Die Dropout-Rate wird entweder auf die Konvolutionsschicht oder auf die verdeckten Neuronen angewandet, wenn diese vorhanden sind. Die Heatmap von Filteranzahl und Dropout (Abbildung 20a) zeigt, dass eine Erhöhung der Filteranzahl bei allen Dropout-Raten den Durchschnitt des F1-Maßes hebt, oder zumindest nicht verschlechtert. Bei der Dropout Rate 0,0 ist die Verbesserung des F1-Durchschnitts durch die Erhöhung der Filteranzahl am geringsten. Bei den höheren Dropout-Raten ist die Verbesserung des F1-Durchschnitts durch höhere Filteranzahlen größer. Das kann so interpretiert werden, dass bei mehr Filtern mehr Werte in der Konvolutionsschicht sind. In diesem Fall bringt Dropout mehr Verbesserungen.

Falls verdeckte Neuronen vorhanden sind, ist der Sachverhalt anders, weil Dropout auf die verdeckte Schicht angewandt wird. Um mehr Einsicht zu bekommen, sollte in zukünftigen Untersuchungen eine Darstellung der F1-Durchschnitte in Abhängigkeit von Filteranzahl und Dropout für Experimente mit verdeckten Neuronen und ohne verdeckten Neuronen erstellt werden. Die besten Werte für alle Filteranzahlen wurden mit der Dropout-Rate 0,5 erreicht.

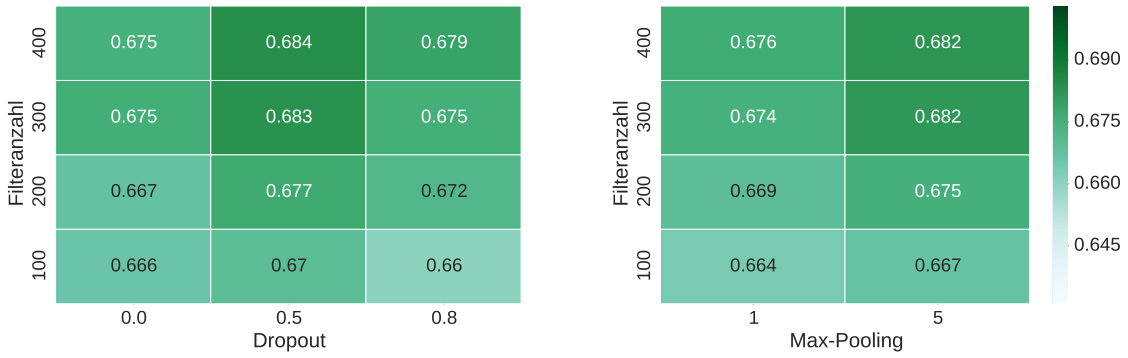

Abbildung 20: Heatmap von Filteranzahl und Dropout und Filteranzahl und Max-Pooling

4.4.4 Filteranzahl und Max-Pooling

Filteranzahl und Max-Pooling hängen in der Architektur des KNN sehr nah zusammen. Die Filteranzahl bestimmt die Anzahl der Vektoren, die von Max-Pooling zusammengefasst werden. In der Heatmap von Filteranzahl und Max-Pooling (Abbildung 20b) zeigt sich, dass das durchschnittliche F1-Maß mit wachsender Filteranzahl immer besser wird oder gleich bleibt. Das gilt für beide Max-Pooling Werte. Der Max-Pooling Wert 5 schneidet bei allen Filteranzahlen besser ab als 1. Es lohnt mehr als nur den höchsten Wert aus Konvolutionsschicht zu entnehmen. 5-Max-Pooling profitiert mehr von höheren Filteranzahlen als 1-Max-Pooling. Das liegt vermutlich daran, dass bei einer größeren Auswahl an Merkmalen durch 5-Max-Pooling weniger Merkmale verloren gehen, die nützlich für die Klassifikation sind.

4.4.5 Filteranzahl und Regularisierungsparameter

Die Interaktion von Filteranzahl und Regularisierungsparameter ist ähnlich zu der von Filteranzahl und Dropout. Eine höhere Filteranzahl begünstigt Überanpassung. Regularisierung wirkt dem entgegen. Die Heatmap von Filteranzahl und Regularisierungsparameter (Abbildung 21a) zeigt, dass größere Werte für beide Parameter das durchschnittliche F1-Maß erhöhen. Der Boxplot der Filteranzahl hat gezeigt, dass mehr Filter das F1-Maß verbessern. Der höchste F1-Durchschnitt in Abbildung 21 wird mit der höchsten Filteranzahl 400 und mit dem größten Regularisierungsparameter 0,0005 erreicht. Diese Beochachtung bestätigt die Werte aus den Boxplots für Filteranzahl und Regularisierungsparameter.

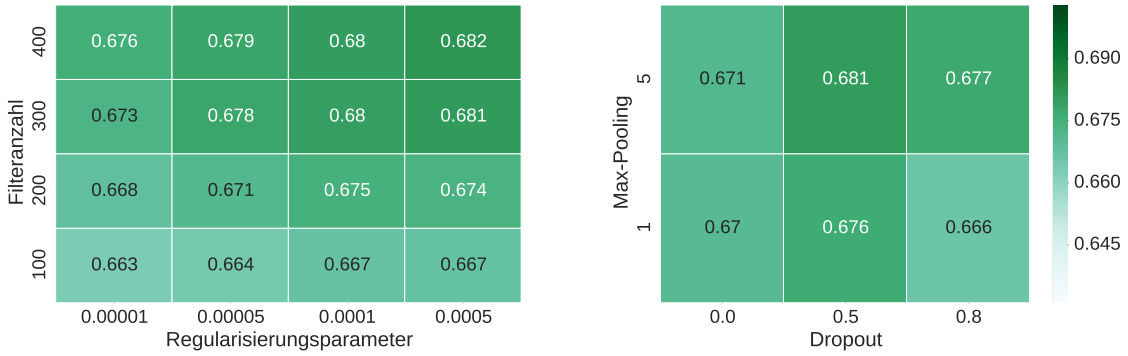

Abbildung 21: Heatmap von Filteranzahl und Regularisierungsparameter und Max-Pooling und Dropout

4.4.6 Max-Pooling und Dropout

Auch bei Max-Pooling und Dropout müsste für genauere Betrachtung eine Darstellung des F1-Durchschnitts in Abhängigkeit des Max-Poolings und der Dropout-Rate für Experimente mit und ohne verdeckte Neuronen erstellt werden. Die aktuelle Darstellung von Max-Pooling und Dropout (Abbildung 21b) zeigt, dass die Dropout-Rate von 0,5 für beide Max-Pooling Werte den besten F1-Durchschnitt erreicht. Verglichen mit 0,0 erhöht die Dropout-Rate 0,5 den F1-Durchschnitt bei beiden Max-Pooling Werten. Die weitere Erhöhung der Dropout-Rate auf 0,8 verschlechtert den F1-Durchschnitt bei beiden Max-Pooling Werten. Bei 1-Max-Pooling ist der F1-Durchschnitt mit Dropout-Rate 0,8 schlechter als ohne Dropout. Das liegt daran, dass beide Parameter zusammen zu weniger Werten auf der Pooling-Schicht führen. Beim niedrigsten Wert für Max-Pooling und dem höchsten Wert für Dropout ist die Anzahl an Werten auf der Pooling-Schicht am geringsten und zu klein. Bei allen Dropout-Raten wurde der F1-Durchschnitt durch Erhöhung des Max-Pooling Wertes größer.

4.4.7 Max-Pooling und verdeckte Neuronen

Die Max-Pooling-Schicht liegt im lingCNN vor der verdeckten Schicht. Bei größerem Max-Pooling bekommt die verdeckte Schicht mehr Eingaben, weil mehr Neuronen in der Pooling-Schicht sind. In der Heatmap (Abbildung 22a) sieht man, dass eine Erhöhung des Max-Pooling Wertes für jede Anzahl von verdeckten Neuronen eine Verbesserung des F1-Durchschnitts bringt. Bei beiden Max-Pooling Werten verbessert sich der F1-Durchschnitt mit der Erhöhung der Anzahl von verdeckten Neuronen von 0 auf 100. Daran sieht man, dass die Einführung von verdeckten Neuronen die Leistung des Systems positiv beeinflusst. Bei der weiteren Erhöhung von 100 auf 500 verdeckte Neuronen verschlechtert sich der F1-Durchschnitt bei beiden Max-Pooling Werten. Der beste Durchschnitt wurde mit der Kombination 100 verdeckte Neuronen und Max-Pooling von 5 erreicht.

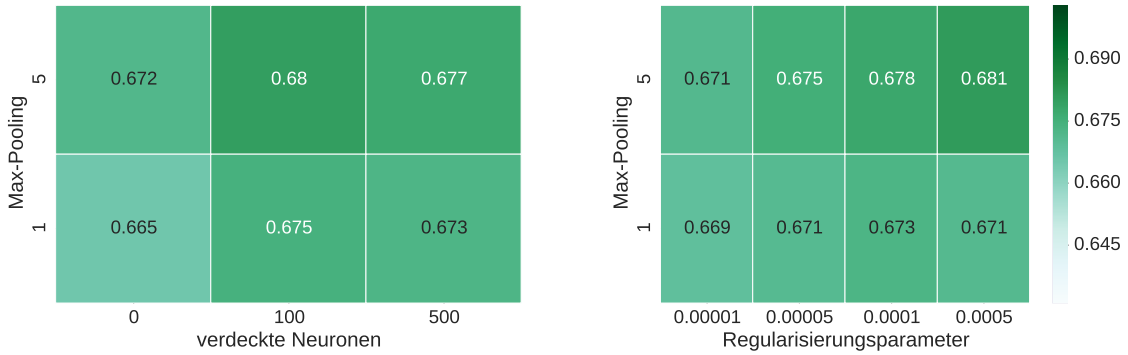

Abbildung 22: Heatmap von Max-Pooling und verdeckten Neuronen und Max-Pooling und Regularisierungsparameter

4.4.8 Max-Pooling und Regularisierungsparameter

Bei Max-Pooling und dem Regularisierungsparameter spielt Überanpassung eine Rolle. Durch mehr Neuronen in der Pooling-Schicht wird Überanpassung unterstützt. Bei größeren Werten für Max-Pooling empfiehlt sich als ein höheres Regularisierungsparameter. Das zeigt auch Abbildung 22b. Die Erhöhung des Regularisierungsparameters verbessert den F1-Durchschnitt bei 5-Max-Pooling. Beim 1-Max-Pooling bringen die Erhöhungen des Regularisierungsparameters nur kleine Verbesserungen im Vergleich zum 5-Max-Pooling.

4.4.9 Lernrate und Dropout

Die Heatmap von Lernrate und Dropout (Abbildung 23a) zeigt verschiedene Tendenzen. Bei der besten Lernrate 0,01 bringt die Einführung von Droput 0,5 eine Verbesserung des F1-Durchschnitts. Bei der Erhöhung der Dropout-Rate auf 0,8 verschlechtert sich der der F1-Durchschnitt minimal. Die F1-Durchschnitte sind höher als bei den anderen beiden Lernraten. Bei der Lernrate 0,1 wird der F1-Durchschnitt durch eine höhere Dropout-Rate besser. Bei der Lernrate 0,001 verschlechtert die Erhöhung der Dropout-Rate den F1-Durchschnitt. Diese Beobachtungen sind analog zur Heatmap von Lernrate und Regularisierungsparameter (Abbildung 23b). Bei größerer Lernrate werden die Gewichte schnell zu groß. Mehr Dropout wirkt dem entgegen, indem weniger Gewichte bei jeder Aktualisierung angepasst und potentiell erhöht werden können. Bei einer niedrigen Lernrate werden die Gewichte zu langsam groß genug und eine hohe Dropout-Rate verstärkt diesen Effekt weiter.

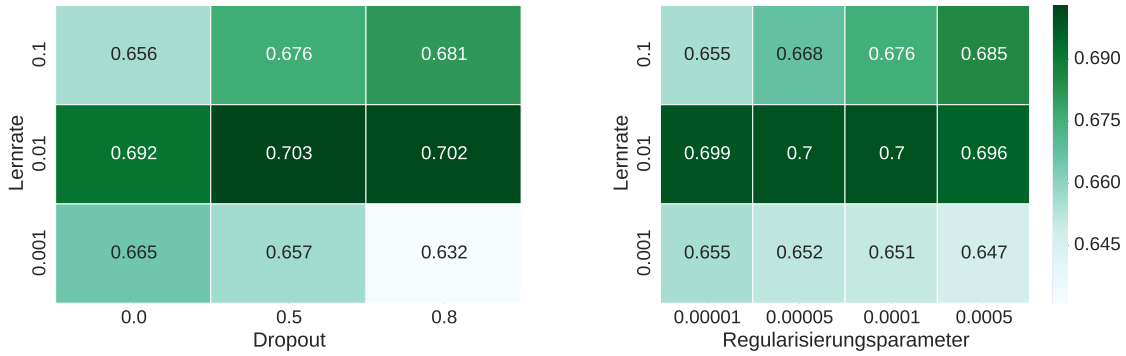

Abbildung 23: Heatmap von Lernrate und Dropout und Lernrate und Regularisierungsparameter

4.4.10 Lernrate und Regularisierungsparameter

Bei der Kombination Lernrate und Regularisierungsparameter (Abbildung 23b) sind bei der Lernrate 0,01 die besten Ergebnisse bei allen Werten für das Regularisierungsparameter zu sehen. Bei der Lernrate 0,1 wird der F1-Durchschnitt mit steigendem Regularisierungsparameter größer. Bei der Lernrate 0,001 hingegen wird der F1-Durchschnitt bei steigendem Regularisierungsparameter immer kleiner. Ein KNN mit hoher Lernrate neigt eher dazu die Gewichte schnell zu groß werden zu lassen. In diesem Fall kann mit starker Regularisierung durch ein hohes Regularisierungsparameter entgegen gewirkt werden. Bei einer niedrigen Lernrate ist das Gegenteil der Fall. Die Gewichte werden eher langsam angepasst. In diesem Fall ist weniger Regularisierung angebracht. Bei einer hohen Lernrate ist also ein hohes Regularisierungsparameter zu empfehlen, bei einer kleinen Lernrate ein kleines.

4.4.11 Lernrate und Max-Pooling

Zwischen Lernrate und Max-Pooling kann man sich im Bezug auf die Systemarchitektur schwer eine Wechselwirkung vorstellen. Durch höheres Max-Pooling gibt es mehr Gewichte, die angepasst werden müssen. In der Heatmap von Lernrate und Max-Pooling (Abbildung 24a) zeigen sich mit der Lernrate 0,01 bei beiden Max-Pooling Werten die besten Ergebnisse. Bei den beiden größeren Lernraten wird der F1-Durchschnitt mit größerem Max-Pooling schlechter. Bei der kleinsten Lernrate verbessert sich der F1-Durchschnitt durch höheres Max-Pooling.

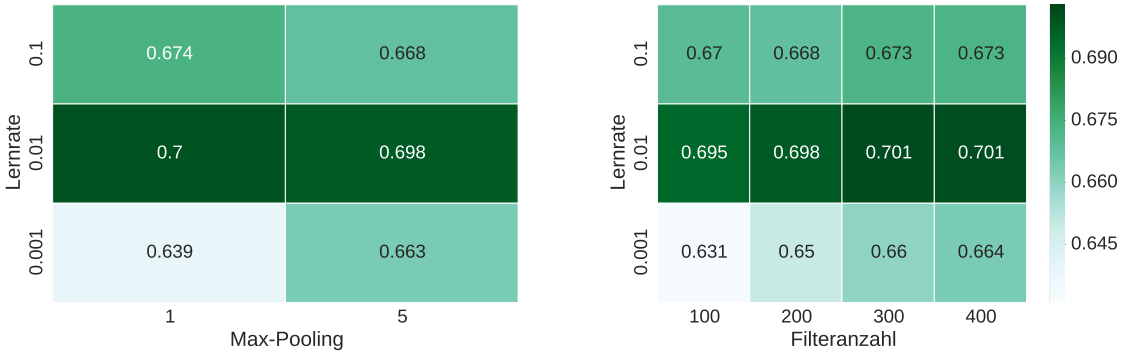

Abbildung 24: Heatmap von Lernrate und Max-Pooling und Lernrate und Filteranzahl

4.4.12 Lernrate und Filteranzahl

Das Verhalten des F1-Durchschnitts bei größerer Filteranzahl ist bei allen drei Lernraten ähnlich. In der Heatmap von Lernrate und Filteranzahl (Abbildung 24b) sieht man, dass die Erhöhung der Filteranzahl auf 200 und 300 den F1-Durchschnitt bei allen Lernraten verbessert. Bei der weiteren Erhöhung auf 400 bleibt der F1-Durchschnitt bei den größeren Lernraten gleich und verschlechtert sich bei der kleinsten Lernrate. Die Lernrate 0,01 zeigt mit bei allen Filteranzahlen die höchsten F1-Durchschnitte.

4.4.13 Lernrate und verdeckte Neuronen

An der Heatmap von Lernrate und verdeckten Neuronen (Abbildung 25a) sieht man, dass sich der F1-Durchschnitt bei den Lernraten 0,01 und 0,1 bei Erhöhung der Anzahl der verdeckten Neuronen von 100 auf 500 verschlechter. Bei der kleinsten Lernrate bringt die größere Anzahl von verdeckten Neuronen bessere Ergebnisse.

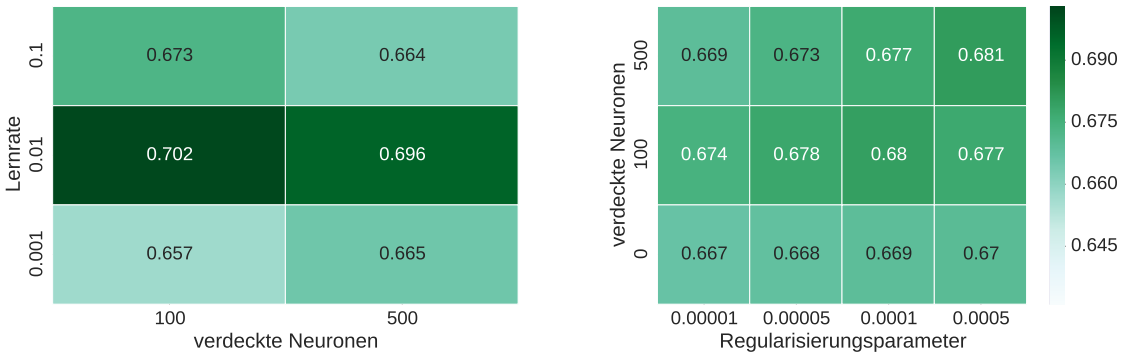

Abbildung 25: Heatmap von Lernrate und Dropout und verdeckten Neuronen und Regularisierungsparameter

4.4.14 Verdeckte Neuronen und Regularisierungsparameter

Durch mehr verdeckte Neuronen kann das KNN die Trainingsdaten präziser modellieren, wird aber auch anfällig für Überanpassung. Die Kombination von vielen verdeckten Neuronen mit dem größten Regularisierungsparameter bringt deswegen die besten Ergebnisse für diese Parameterkonfiguration. Die Heatmap von verdeckten Neuronen und Regularisierungsparamter (Abbildung 25b) zeigt, dass die Erhöhung des Regularisierungsparameters fast immer eine Verbesserung des F1-Durchschnitts bringt. Nur mit 100 verdeckten Neuronen verschlechtert sich bei der Erhöhung des Regularisierungsparameters von 0,0001 auf 0,0005 der F1-Durchschnitt. Bei den beiden größeren Regularisierungsparametern verbessert sich der F1-Durchschnitt mit jeder Erhöhung der Anzahl der verdeckten Neuronen. Bei den beiden kleineren Regularisierungsparametern gilt das nur für die Erhöhung von 0 auf 100 verdeckte Neuronen. Hier trifft wie bei der Kombination von Filteranzahl und Regularisierungsparameter die Beobachtung zu, dass bei mehr Werten auf einer Schicht eine größere Regularisierung besser funktioniert.

4.4.15 Filteranzahl und verdeckte Neuronen

Filteranzahl und verdeckte Neuronen hängen im KNN indirekt zusammen. Deswegen ist es schwierig eine intuitive Interpretation für ihr Zusammenwirken zu finden. Die Daten in der Heatmap (Abbildung 26) zeigen, dass bei 0 und 100 verdeckten Neuronen jede Vergrößerung der Filteranzahl den F1-Durchschnitt hebt. Bei 500 verdeckten Neuronen gibt es zwischen Filteranzahl 300 und 400 keine Verbesserung. Bei allen Filteranzahlen bringen 100 verdeckte Neuronen eine Verbesserung gegenüber 0 verdeckten Neuronen. Bei der weiteren Erhöhung auf 500 werden die F1-Werte schlechter.

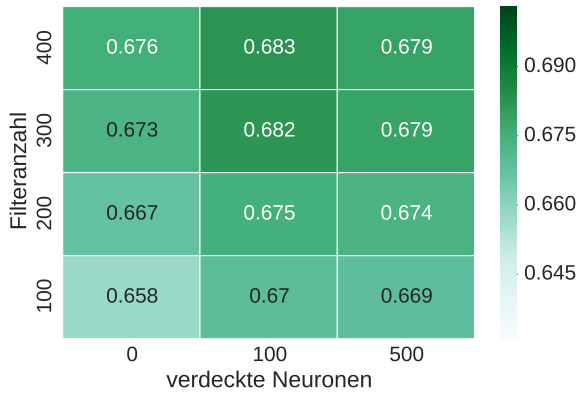

Abbildung 26: Heatmap von Filteranzahl und verdeckten Neuronen

4.5 Ergebnisse der Parameterstudie

Nicht aus allen Grafiken lassen sich allgemeine Regeln für die Hyperparametereinstellung von neuronalen Netzen ableiten. Einige Tendenzen sind jedoch klar erkennbar. Dazu gehört der große Einfluss der Lernrate auf die Leistung des Systems. Der Boxplot der Lernraten (Abbildung 13) zeigt die größten Unterschiede zwischen den getesteten Parametern. Es sollte bei der Einstellung der Hyperparameter also zuerst darauf geachtet werden, eine gute Lernrate zu finden. Bei den hier getesteten Werten 0,001, 0,01 und 0,1 war 0,01 der beste Wert. Aus den Heatmaps ergibt sich die Tendenz, dass die Erhöhung der Filteranzahl im Bereich von 100 bis 300 die Ergebnisse verbessert. Bei der Erhöhung von 300 auf 400 ergeben sich nur noch geringe Verbesserungen, oder die Leistung bleibt gleich. Das sind die zwei stärksten Beobachtungen aus der Versuchsreihe.

Dropout sollte eingesetzt werden. Dabei sollte die Dropout-Rate nicht zu groß gewählt werden. Unter den Werten 0,0, 0,5 und 0,8 war 0,5 der beste. Verdeckte Neuronen sollten benutzt werden. Dabei schneidet die Anzahl von 100 besser ab als 500. Beim k-Max-Pooling zeigen Heatmaps und Boxplots, dass der Wert 5 bessere Ergebnisse bringt als 1. Beim Regularisierungsparameter hat sich gezeigt, dass sein Einfluss stark von der Lernrate abhängt. Bei der kleinsten Lernrate ist das kleinste Regularisierungsparameter am besten, bei der größten Lernrate das größte. Bei der mittleren Lernrate hat es wenig Einfluss. Wenn eine gute Lernrate gefunden ist, sollten Tests ohne Regularisierung durchgeführt werden.

5 Verwandte Arbeiten

5.1 Satzklassifikation

Kim (2014) stellt ein Modell für Satzklassifikation vor, dass auf dem neuronalen Netz in Collobert und Weston (2008) basiert. Dabei kommen vortrainierten Wortvektoren aus dem CBOW-Modell von Mikolov et al. (2013a) zum Einsatz. Es wird untersucht, welche Auswirkungen die Anpassung der Wortvektoren beim Training des neuronalen Netzes hat. Dafür werden vier verschiedene Modellvarianten verwendet. Bei der ersten Variante werden die Wortvektoren zufällig initialisiert (CNN-rand). Bei der zweiten Variante werden die vortrainierten Vektoren von Mikolov et al. (2013a) verwendet und unbekannte Wörter zufällig initialisiert (CNN-static). Bei der dritten Variante werden die vortrainierten Wortvektoren verwendet und während des Trainings angepasst (CNN-non-static). Bei der vierten Variante sind die vortrainierten Wortvektoren in zwei *channels* vorhanden. Nur ein channel wird während des Trainings angepasst, der andere bleibt gleich. Es werden beide channels für die Konvolution eingesetzt (CNN-multichannel). So soll Überanpassung vermieden werden.

CNN-static zeigt Verbesserungen im Vergleich mit CNN-rand. Die CNN-multichannel Variante brachte nicht die gewünschten Verbesserungen und war nur auf manchen Datensätzen besser als die CNN-non-static Variante. Zur Untersuchung der Anpassung der Wortvektoren wurde einige Vektoren aus dem CNN-static mit dem CNN-non-static Modell verglichen. Dabei fiel auf, dass der Vektor für *good* in den vortrainierten Daten am nächsten am Vektor für *bad* ist. Nach der Anpassung durch das CNN-non-static Modell ist *good* näher an *nice*.

Severyn und Moschitti (2015) stellen ein System zur Sentiment Analyse von Tweets vor. Besonderer Wert wird dabei auf die Initialisierung der Werte der Wortvektoren gelegt. Es wird ein Konvolutionsnetz ähnlich zu dem in Kalchbrenner et al. (2014) und Kim (2014) verwendet. Die Eingabevektoren werden zuerst mit dem Skip-Gramm-Modell von Mikolov et al. (2013a) auf einem unüberwachtem Korpus von 50 Millionen Tweets gelernt. Dann werden die Wortvektoren mit distant supervision (entfernt überwachtes Lernen), beschrieben in Go et al. (2011), für Sentiment Analyse angepasst. Die so gewonnen und angepassten Vektoren werden an das Konvolutionsnetz übergeben und dort weiter angepasst. Das Konvolutionsnetz besteht aus der Eingangsmatrix der Wortvektoren, einer Konvolutionsschicht, einer Pooling-Schicht und einer Softmax-Schicht. Die weitere Voranpassung der Wortvektoren an die Aufgabe Sentiment Analyse bringt Verbesserung gegenüber den Daten, die nur mit dem Skip-Gramm-Modell trainiert wurden.

Kalchbrenner et al. (2014) stellen ein System zur semantischen Satzmodellierung vor. Anders als in den bisher vorgestellten Systemen besitzt das Netz zwei Konvolutionsschichten. Außerdem kommt Dynamic k-Max-Pooling zum Einsatz. Das ist eine Abwandlung vom bereits vorgestellten k-Max-Pooling. Dabei ist die Anzahl k der Merkmale, die aus der Konvolutionsschicht auf die Pooling-Schicht übertragen werden nicht statisch, sondern wird in Abhängigkeit von der Satzlänge und der Tiefe des Netzes gewählt.

5.2 Parameterstudien

Zhang und Wallace (2015) haben den Einfluss von Hyperparametern auf ein Konvolutionsnetz zur Klassifizierung von Sätzen untersucht. Dabei war die Fragestellung, bei welchen Hyperparametern es sich lohnt, Zeit in die Optimierung zu investieren und bei welchen es feststehende Werte gibt, die unabhängig von den verwendeten Daten gut funktionieren. Es wird eine Regularisierungstechnik verwendet, die bis jetzt nicht erwähnt wurde. Bei dieser Technik wird ein Schwellwert für die L2-Norm der Softmax-Schicht festgelegt und der Vektor angepasst, wenn dieser überschritten wird. Die Wortvektoren können während des Trainings angepasst werden (non-static) oder gleich bleiben (static). In der Arbeit wurden verschiedene Klassifikationsaufgaben mit sieben Datensätzen durchgeführt

Es wurden Wortvektoren von word2vec Mikolov et al. (2013a), GloVe Pennington et al. (2014), eine Kombination von beiden und Vektoren mit 1-aus-n-Code getestet. Die 1-aus-n-Code-Vektoren waren auf allen getesteten Datensätzen schlechter als word2vec und GloVe. Zwischen word2vec und GloVe konnte keine eindeutig bessere Wortrepräsentation bestimmt werden. Die Konkatenation von Vektoren beider Systeme brachte keine großen Verbesserungen. Die Tests für verschiedene Filterbreiten haben ergeben, dass gute Werte für die Filterbreite von der Satzlänge und damit vom Datensatz abhängt. Es wird empfohlen Filterbreiten von 2 bis 25 auszuprobieren.

Auch bei der Anzahl der Merkmalsbereiche für jede Filterbreite hängt der beste Wert vom Datensatz ab. Es wird empfohlen Werte zwischen 50 und 600 auszuprobieren. Bei 1000 und 2000 Merkmalsbereichen wird die Leistung schlechter. Bei der vorliegenden Arbeit wird der empfohlene Bereich auf die Werte 300 und 400 eingegerenzt. Dabei muss allerdings beachtet werden, dass andere Datensäzte als von Zhang und Wallace (2015) verwendet wurden.

Es wurden die Pooling-Strategien local max pooling, k-max pooling und *average pooling* (Pooling mit dem Durchschnitt, statt dem Maximum) auf den Merkmalsbereichen getestet. 1-max pooling, bei dem für jeden Merkmalsbereich nur der größte Wert ausgewählt wird, hat die besten Ergebnisse gezeigt. Das steht im Kontrast zur vorliegenden Arbeit, in der 5-Max-Pooling besser abgeschnitten hat.

Für Dropout werden Werte zwischen 0,1 und 0,5 empfohlen. In der vorliegenden Arbeit wurde hat sich der Wert von 0,5 besser als 0,0 und 0,8 erwiesen.

Zu den Orientierungswerten für die genannten Parametern stellen die Autoren fest, dass mehrere Testläufe mit gleicher Konfiguration unterschiedliche Ergebnisse liefern. Das liegt an den Zufallswerten beim stochastischen Gradientenverfahren, der zufälligen Initialisierung von Gewichten und der zufälligen Auswahl von Neuronen beim Dropout. Es wird empfohlen die Varianz der Ergebnisse bei mehreren Testläufen anzugeben, wenn die Leistung eines neuronalen Netzes wiedergegeben wird.

Peng et al. (2015) haben verschiedene Regularisierungstechniken für neuronalen Netze, bei denen Wortvektoren verwendet werden, getestet. Dabei wurde der Einfluss verschiedener Arten der Regularisierung und das Verhalten bei der Kombination verschiedener Regularisierungsstrategien untersucht. Die Anwendungen waren Relationsextraktion und Sentiment Analyse.

Es wurden vier Regularisierungstrategien getestet. Die erste ist L2-Regularisierung. Bei Sentiment Analyse brachte unter den Werten 10^{-5}, 10^{-4} und 10^{-3} der beste Wert 10^{-3} eine Verbesserung von 2,07%. Dieser Wert für das Regularisierungsparameter ist deutlich höher als die Werte, die in der vorliegenden Arbeit getestet wurden (maximal $5 \cdot 10^{-5}$). In zukünftigen Un-

tersuchungen könnten also auch höhere Werte zu guten Ergebnissen führen.

Die zweite Regularisierungsstrategie ist ein Schwellwert für die L2-Norm der Wortvektoren. Für Sentiment Analyse ergab diese Strategie keine signifikanten Verbesserungen.

Die dritte Strategie ist, die Anpassung der Wortvektoren nur bis zu einem bestimmten Abstand zur Ausgangsform zu erlauben. Diese Einschränkung der Anpassung der Wortvektoren brachte keine großen positiven Auswirkungen auf die Generalisierung

Die vierte Strategie ist Dropout. Es wurden die Werte 0,1, 0,2, 0,3, 0,4 und 0,7 getestet. Dabei brachte 0,1 mit 1,76% die größten Verbesserungen für Sentiment Analysis. Dieser Wert liegt zwischen den Werten 0,0 und 0,5, die in der vorliegenden Arbeit getestet werden und kann bei weiteren Untersuchungen berücksichtigt werden. Beim Vergleich von L2-Regularisierung und Dropout brachte L2-Regularisierung eine größere Verbesserung im F1-Maß. Die Kombination von Dropout und L2-Regularisierung verbesserte die Genauigkeit gegenüber dem Einsatz von nur L2-Regularisierung nicht. In der vorliegenden Arbeit ergänzten sich die beiden Regularisierungsstrategien für drei der vier getesteten Regularisierungsparameter.

6 Zusammenfassung

Diese Arbeit beschäftigt sich mit der Hyperparametereinstellung von Konvolutionsnetzen für Sentiment Analyse. Es wurde zuerst eine Einführung in das Thema Sentiment Analyse gegeben. Dabei wurde die Motivation, mehrere Meinungen für eine Entscheidungsfindung zu berücksichtigen, genannt. Es wurden die Teilaufgaben Textauswahl, Polaritätsklassifikation und Darstellung der Meinungen besprochen. In der Einführung in künstliche Neuronale Netze wurden die wichtigsten Neuronen einfaches Perzeptron, Sigmoidneuron, tanh-Neuron und stückweise-lineares-Neuron besprochen. Es wurden Schichten, das Lernverfahren mit Trainings-, Test- und Developementdaten und das Gradientenverfahren kurz erläutert. Außerdem wurden die Begriffe Konvolution, Softmax und Hyperparameter erklärt. Danach wurden neuronale Netze im Bezug auf Natural Language Processing betrachtet. Dabei wurden Wortvektoren, Konvolution über Sätze und k-Max-Pooling, die Systemarchitektur und Regularisierung und Dropout besprochen.

Im Hauptteil wurde das System, mit dem die Experimente durchgeführt wurden, beschrieben. Dabei wurde das Preprocessing, die Architektur und die verwendeten Daten dargestellt. Für den Vergleich verschiedener Werte eines Parameters wurden Boxplots erstellt. Um den gegenseitigen Einfluss von Parametern darzustellen wurden Heatmaps erstellt. Die Auswertung der Grafiken hat ergeben, dass die Wahl der Lernrate das System gegenüber den anderen getesteten Parametern am meisten verbessert. Sie ist also der Parameter, der als erstes optimiert werden sollte. Bei den getesteten Werten 0,001, 0,01 und 0,1 war 0,01 der beste Wert. Die Erhöhung der Filteranzahl im Bereich von 100 bis 300 hat die Ergebnisse verbessert. Unter den Dropout-Raten 0,0, 0,5 und 0,8 war 0,5 der beste. Auch der Einsatz von verdeckten Neuronen hat Verbesserungen gebracht. 100 verdeckte Neuronen brachten bessere F1-Maße als 500. K-Max-Pooling hat für den Wert 5 bessere Ergebnisse als für den Wert 1 gezeigt. Der Einfluss des Regularisierungsparameters war bei verschiedenen Lernraten unterschiedlich. Bei der kleinsten Lernrate 0,001 bringt das kleinste Regularisierungsparameter die größten Verbesserungen, bei der größten Lernrate 0,1 das größte. Bei der besten Lernrate 0,01 sollte getestet werden, ob L2-Regularisierung nötig ist. Außerdem wurde festgestellt, dass sich Dropout und L2-Regularisierung ergänzen.

Beim Vergleich der Ergebnisse mit Peng et al. (2015) hat sich gezeigt, dass für die L2-Regularisierung noch größere Werte ausprobiert werden könnten. Beim Dropout empfehlen Peng et al. (2015) 0,1. Dieser Wert wurde hier nicht getestet, sollte aber in weiteren Untersuchungen berücksichtigt werden. Für die Filteranzahl werden von Zhang und Wallace (2015) Werte von 50 bis 600 empfohlen. Dieser Wertebereich wird durch die in der vorliegenden Arbeit gefundenen guten Werte von 300 und 400 bestätigt bzw. eingeschränkt. In Zhang und Wallace (2015) wird 1-Max-Pooling empfohlen. In der vorliegenden Arbeit hat 5-Max-Pooling besser abgeschnitten. Außerdem widersprechen die Ergebnisse von Peng et al. (2015) den hier vorgestellten Ergebnissen, die besagen, dass sich L2-Regularisierung und Dropout ergänzen. Dabei muss allerdings beachtet werden, dass bei den beiden Untersuchungen verschiedene Systeme und Datensätze verwendet wurden.

Um weitere Erkenntisse über die Hyperparametereinstellung zu gewinnen, könnten Experimente ohne Regularisierung und Dropout durchgeführt werden und mit den vorliegenden Ergebnissen verglichen werden, um das Verhalten des Systems bei der besten gefundenen Lernrate zu untersuchen. Außerdem könnten weitere Experimente mit Werten zwischen den hier getesteten Parametern für verdeckte Neuronen, Dropout und Max-Pooling durchgeführt werden.

Literatur

Bengio, Y., Ducharme, R., Vincent, P., und Janvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research Volume 3.*

Collobert, R. und Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine Learning.*

Ebert, S., Vu, N. T., und Schütze, H. (2015). A linguistically informed convolutional network. In *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis.*

Go, A., Bhayani, R., und Huang, L. (2011). Twitter sentiment classification using distant supervision. In *Proceedings of the Workshop on Languages in Social Media.*

Hu, M. und Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Kalchbrenner, N., Grefenstette, E., und Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *In Proceedings of the Annual Meeting of the Association for Computational Linguistics.*

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

Lecun, Y., Bottou, L., Bengio, Y., und Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE.*

McMinn, A. J., Moshfeghi, Y., und Jose, J. M. (2013). Building a large-scale corpus for evaluating event detection on twitter. In *Proceedings of the ACM International Conference on Information & Knowledge Management.*

Mikolov, T., Chen, K., Corrado, G., und Dean, J. (2013a). Efficient estimation of word representations in vector space. *Computing Research Repository.*

Mikolov, T., tau Yih, W., und Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*

Mohammad, S. M. und Kiritchenko, S. (2015). Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence Volume 31.*

Mohammad, S. M. und Turney, P. D. (2013). Crowdsourcing a word-emotion association lexicon. *Computational Intelligence Volume 29.*

Nakov, P., Rosenthal, S., Kozareva, Z., Stoyanov, V., Ritter, A., und Wilson, T. (2013). Semeval-2013 task2: Sentiment analysis in twitter. In *Proceedings of the 7th International Workshop on Semantic Evaluation.*

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.

Pang, B. und Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Pang, B. und Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*.

Peng, H., Mou, L., Li, G., Chen, Y., Lu, Y., und Jin, Z. (2015). A comparative study on regularization strategies for embedding-based neural networks. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Pennington, J., Socher, R., und Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Severyn, A. und Moschitti, A. (2015). Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Wiebe, J. und Cardie, C. (2005). Annotating expressions of opinions and emotions in language. language resources and evaluation. In *Language Resources and Evaluation Volume 39*.

Zhang, Y. und Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *Computing Research Repository*.

Abbildungsverzeichnis

Tabellenverzeichnis