Jugend forscht Wettbewerbsrunde 2018

Warum Shakespeare wirklich Shakespeare ist

Eine Analyse künstlicher neuronaler Netze

Tobias Scheithauer

Im vergangenen Jahr habe ich ein künstliches neuronales Netz so trainiert, dass es mit erstaunlich hoher Genauigkeit den Autor von Texten erkennen kann. Hierbei gibt es wie bei jeder Anwendung künstlicher neuronaler Netze allerdings ein Problem: Niemand kann genau sagen, wie ein solches System zu Entscheidungen kommt, weswegen auch die Zuverlässigkeit dieser Technologie fraglich ist. Deshalb habe ich mir in dieser Wettbewerbsrunde das Ziel gesetzt, den Prozess der Entscheidungsfindung von künstlichen neuronalen Netzen zu verstehen. Dazu habe ich einen Algorithmus entwickelt, dessen Ausgaben mit statistischen und graphischen Methoden ausgewertet werden können.

Inhaltsverzeichnis

1 Einleitung

1.1 Die Fragestellung

Künstliche neuronale Netze *engl. (ANNs)* sind wohl eine der wichtigsten Entwicklungen seit der Erfindung moderner Computer.

> ANNs sind die Kernkomponente des Deep Learning. Sie sind flexibel, mächtig und skalierbar, was sie ideal für große und hochgradig komplexe Machine-Learning-Aufgaben einsetzbar macht, wie beispielsweise die Klassifizierung von Milliarden Bildern (Google Images), Spracherkennung (Apples Siri), Videoempfehlungen für Hunderte Millionen Nutzer pro Tag (YouTube) oder den Weltmeister im Brettspiel Go durch Analyse von Milliarden Partien und anschließendes Spielen gegen sich selbst zu schlagen (AlphaGo von DeepMind).
> [Géron, 2017]

In der Software, die ich für die vergangene Wettbewerbsrunde programmierte, ordnet ein solches ANN mit erstaunlich hoher Genauigkeit Texte den richtigen Autoren zu, sodass sie auch im internationalen Vergleich mit den besten Softwarelösungen für dieses Problem mithalten kann [Scheithauer, 2017]. Dies liegt vor allem daran, dass das künstliche neuronale Netz nahezu alle Muster, die für die Zuordnung wichtig sind, selbst erlernt. Herkömmliche Systeme, die ohne ein ANN arbeiten, verwenden stattdessen vorgegebene Features, wie die Satzlänge, den Wortschatz und ähnliches. Hierbei ergibt sich allerdings ein wichtiger Unterschied zwischen den vorhandenen Softwarelösungen und meiner Software: Sobald ANNs verwendet werden, ist der Entscheidungsprozess weder nachvollziehbar noch bewertbar und man muss blind auf die Ausgaben des neuronalen Netzes vertrauen. Da sich die Zuverlässigkeit nicht bewerten lässt, ist dies insbesondere beim Einsatz von ANNs in sicherheitsrelevanten Umgebungen, wie beispielsweise bei autonomen Autos im Verkehr ein großes Problem.

Auch bei meiner Anwendung stellt sich daher die Frage, auf welche Muster das trainierte Netz achtet, und ob manche dieser mit den in herkömmlicher Software verwendeten Entscheidungskriterien übereinstimmen. Herkömmliche Softwarelösungen zur Autorenerkennung nutzen das Vokabular, bestimmte Worteigenschaften, die Syntax und Anomalien. Bei ANNs ist das anders, denn sie bestehen aus einer Vielzahl von Variablen, die auf eine zwar mathematisch aber nicht inhaltlich verständliche Weise angepasst werden, bis das System gut funktioniert. Ich habe mir vorgenommen folgende Fragen zu beantworten, um den Entscheidungsprozess von ANNs für Menschen verständlich zu machen:

1. Wie lässt sich der Erfolg des neuronalen Netzes in der Autorenerkennung erklären?
2. Welche Kriterien kommen im Entscheidungsprozess meines ANNs zum Einsatz und stimmen sie mit denen herkömmlicher Systeme überein?
3. Ergeben sich durch die Beantwortung der Fragen Möglichkeiten zur Bewertung des ANNs?

1.2 Der Status Quo im Verstehen von ANNs

Durch die zahlreichen Anwendungsmöglichkeiten von künstlichen neuronalen Netzen stellen immer mehr Entwickler, aber stets auch Anwender die Frage, warum solche Systeme funktionieren. Zur Beantwortung dieser Frage gibt es zwei Herangehensweisen: Zum einen wird der Trainingsvorgang von ANNs mathematisch erklärt, zum anderen wird versucht, den Entscheidungsprozess der ANNs in der jeweiligen speziellen Anwendung zu erklären.

Für die Erklärung des Trainingsvorgangs gibt es bereits viele Lösungen. Einige davon nutzen Animationen und sind daher auch für Nicht-Mathematiker leicht verständlich [Sanderson,

2017], andere hingegen sind sehr auf die mathematische Korrektheit fokussiert und daher für viele Anwender nicht gut zu verstehen. Da die Erklärung des Trainingsvorgangs keine Bewertung der Entscheidungen zulässt, werde ich mich in dieser Arbeit nicht damit befassen. Stattdessen werde ich mich der Erklärung des Entscheidungsprozesses widmen. Hierfür gibt es vor allem für ANNs, die eine Bildklassifizierung durchführen sollen, viele Veröffentlichungen, da sich Visualisierungen der Variablen dieser ANNs sehr leicht interpretieren lassen. [Liu et al., 2017], [Harley, 2015], [Zeiler and Fergus, 2014]. Solche Visualisierungen sind für textverarbeitende ANNs wie meines nicht verständlich, weswegen diese Technik für mich nur für ein besseres Verständnis der Ausgaben vorgelagerter Rechenprozesse zum Einsatz kommt.

2 Methoden

2.1 Über das zu untersuchende künstliche neuronale Netz

Das künstliche neuronale Netz, das ich in der vergangenen Wettbewerbsrunde trainiert habe, ist ein sogenanntes 'Convolutional Neural Net', kurz CNN, was wörtlich übersetzt 'faltendes neuronales Netz' bedeutet. Als Eingabe kommt eine zweidimensionale Matrix (Eingabematrix) zum Einsatz, in der die Zeilen für je ein Zeichen des Eingabealphabets stehen und die Spalten je einer Zeichenposition im Eingabetext entsprechen. Alle Elemente einer Spalte sind dabei Nullen, bis auf das Element, das dem Zeichen des Eingabealphabets an der jeweiligen Position im Eingabetext entspricht. Dieses Element ist eine Eins. Da die Eingabetexte eine Länge von 1000 Zeichen haben und das Eingabealphabet aus 70 Zeichen (Alphabet, Zahlen und Sonderzeichen) besteht, hat die Eingabematrix eine Größe von 1000x70.

Das CNN besteht aus mehreren Schichten von Neuronen und ist dem biologischen Gehirn

$$\begin{array}{c} \\ O \\ P \\ Q \\ R \\ S \\ T \\ U \\ V \\ W \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Abb. 1: Beispiel einer Eingabematrix für die Zeichenfolge 'WORT'.

nachempfunden. Namensgebend sind dabei die sogenannten *convolutional Layer*, die für die Durchführung einer diskreten Faltung sorgen. Hierbei ist die Ausgabematrix einer Schicht die Eingabematrix der darüberliegenden Schicht. Vereinfacht wird dabei ein Filter (genauer Filtermatrix) über die Eingabematrix bewegt. Dabei wird mit jedem Schritt das Frobenius-Skalarprodukt der übereinanderliegenden Matrizen berechnet und das Ergebnis in eine neue Matrix eingetragen. Ich verwende dabei ausschließlich Filtermatrizen derselben Höhe wie die Eingabematrizen, sodass für das Entstehen einer Matrix mehrere Filtermatrizen pro Convolutional Layer verwendet werden. Für jede Filtermatrix entsteht dabei eine Zeile der Ausgabematrix dieser Schicht.

In meinem künstlichen neuronalen Netz kommen sechs faltende Schichten zum Einsatz. In den ersten beiden Schichten kommen Filter mit einer Breite von sieben Neuronen zum Einsatz. In den restlichen vier Schichten haben die Filter jeweils eine Breite von drei Neuronen. Häufig werden in CNNs zusätzlich sogenannte Pooling Layer eingesetzt, was bei dem von mir untersuchten ANN nicht der Fall ist. Dies liegt daran, dass ich nach einem Hinweis aus dem letzten Jahr diese Pooling Layer entfernt habe, ohne dass sich dabei die Genauigkeit verringert hat. Allerdings ist

$$conv(\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 1 & 2 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 2 \\ 1 & 2 \\ 2 & 1 \\ 1 & 1 \end{pmatrix}) = \begin{pmatrix} 3 & 4 & 4 & 2 & 2 & 3 & 2 \\ 4 & 3 & 2 & 4 & 3 & 2 & 4 \end{pmatrix}$$

Abb. 2: Beispiel einer Convolution.

dabei die Effizienz durch einen geringeren Rechenaufwand gestiegen. Aus der Ausgabematrix der sechsten faltenden Schicht wird mithilfe eines sogenannten *Fully-connected Layer* ein Ausgabevektor berechnet. Dieser wird mit der Softmaxfunktion so normiert, dass sich ausschließlich positive Werte ergeben, die sich zu Eins addieren. So ergeben sich die Wahrscheinlichkeiten zur Zugehörigkeit der eingegebenen Texte zu den einzelnen Autoren. Hierbei steht jede Dimension also für einen Autor.
Das von mir untersuchte ANN ist auf die Unterscheidung der fünf bekannten deutschen Autoren Goethe, Kafka, Kleist, Raabe und Schiller trainiert. Am Anfang des Trainings werden alle Filtermatrizen mit zufälligen Werten initialisiert. In kleinen Paketen werden dann immer mehrere für das Training ausgewählte Texte auf einmal in das neuronale Netz eingegeben. Während des Trainings wird für jedes eingegebene Paket ein Fehlerwert ermittelt, der die Abweichung der Ausgaben des ANNs von der gewünschten Ausgabe berechnet. Dieser wird während des Trainings durch eine Anpassung der Filtermatrizen mit jedem Schritt geringer. Sind alle Trainingstexte in das neuronale Netz eingegeben worden, beginnt der Evaluationsprozess. Dafür werden Texte, die nicht für das Training verwendet wurden, in das ANN eingegeben und der Fehlerwert berechnet, ohne eine Optimierung der Filtermatrizen durchzuführen. Dadurch wird gewährleistet, dass das neuronale Netz nicht mit diesen Daten trainiert wird, um so verlässliche Aussagen über den Trainingsfortschritt machen zu können. Nach Abschluss der Evaluation folgt eine weitere Trainingsrunde. Dies wird wiederholt bis der Trainingsvorgang abgebrochen wird.
Zum Teil nach: [Scheithauer, 2017]

2.2 Der NeuronRank-Algorithmus für künstliche neuronale Netze

Die Methoden der Analyse bildverarbeitender ANNs stützen sich vor allem auf die Visualisierung der Filtermatrizen. Diese Herangehensweise eignet sich für die Untersuchung meines textverarbeitenden ANNs, weil sich solche Visualisierungen für Textdaten nicht ohne weiteres deuten lassen. Daher habe ich für die Analyse der Eingaben, die für die Aktivierung einzelner Neuronen wichtig sind, selbst einen Algorithmus entwickelt.
Das ANN lässt sich mathematisch als gerichteter, kantengewichteter, azyklischer Graph modellieren. In einer solchen Darstellung bilden die Neuronen die Knoten und die Verbindungen der Neuronen die Kanten. Diese Modellierung ist prinzipiell vergleichbar mit der Modellierung des Internets, auf der der PageRank-Algorithmus aufbaut. Dieser dient der Suchmaschine Google als Grundlage zur Bewertung und Gewichtung von Internetseiten [Page et al., 1999] und wird zur Sortierung der Suchergebnisse verwendet. Dazu wird für jede Internetseite ein Wichtigkeitswert aus der Summe der Wichtigkeitswerte der auf eine Seite verlinkenden Internetseiten berechnet. Ich habe mich bei der Erstellung meines Algorithmus von dieser Funktionsweise inspirieren lassen. Mithilfe meines Algorithmus sollen für je ein zu untersuchendes Neuron die Wichtigkeitswerte der Neuronen untersucht werden, die zur Berechnung dessen Aktivierung eine Rolle spielen. Durch die Eigenschaft des ANN-Graphen, gerichtet zu sein, lassen sich die Wichtigkeitswerte Schicht für Schicht von dem ausgewählten Neuron bis zur Eingabeschicht des ANNs berechnen. Da der Graph des ANNs anders als der Graph für den PageRank-Algorithmus ein kantengewichteter Graph ist und in den Kantengewichten die Information des Einflusses zwischen den Neuronen steckt, ist es Aufgabe meines Algorithmus, die Wichtigkeitswerte der Neuronen mit den Kantengewichten zu verrechnen. Angelehnt an die zentrale Formel des PageRank-Algorithmus

ist die folgende Formel als Grundlage meines Algorithmus entstanden:

$$r_i = \sum_{j \in J} r_j * w_{i,j} \tag{1}$$

r_i: Rang des Neurons i
J: Menge aller Neuronen, zu denen eine Verbindung von i besteht
$w_{i,j}$: Gewicht der Verbindung von Neuron i zu Neuron j

Für die Berechnung des Wichtigkeitswertes r eines Neurons i werden alle Wichtigkeitswerte der Neuronen J der über i liegenden Neuronenschicht, die mit i verbunden sind, mit dem jeweiligen Kantengewicht der Verbindung der beiden Neuronen $w_{i,j}$ multipliziert und zusammenaddiert. Die so entstandene Summe ist der Wichtigkeitswert des Neurons r_i.
Als Eingabe benötigt mein Algorithmus eine Matrix, die die Maße der Neuronenschicht des zu untersuchenden Neurons hat. Mit dieser Matrix lässt sich durch Maskierung mit einer Eins und Nullen auswählen, welches Neuron untersucht werden soll. Die Ausgabe besteht aus einer Liste von Matrizen, die den Neuronenschichten unter dem betrachteten Neuron entsprechen. In diesen Matrizen repräsentiert jedes Element ein Neuron und gibt dessen Wichtigkeitswert für die Aktivierung des ausgewählten Neurons an. Diese Werte lassen sich wie folgt interpretieren: Das Vorzeichen des Wertes gibt an, ob ein positiver oder negativer Einfluss von einer positiven Aktivierung des jeweiligen Neurons auf die Aktivierung des zu untersuchenden Neurons ausgeht. Eine negative Wirkung ist vergleichbar mit einer Synapse in einem biologischen Gehirn, deren Aktivierung das nachgeschaltete Neuron hemmt. Der Betrag der Wichtigkeitswerte gibt die Stärke des Einflusses der Neuronen auf das ausgewählte Neuron an. Die Ausgaben des Algorithmus sind die Grundlage für statistische und graphische Methoden, die ich verwendet habe, um die Zusammenhänge im ANN zu verstehen.

2.3 Zu den Matrizen aus Neuronenschichten

Sowohl die Ausgabe des NeuronRank-Algorithmus, als auch die Ausgabe der Aktivierungen der Neuronen habe ich als Matrix modelliert. In diesen Matrizen ist jedem Neuron einer Schicht ein Element zugeordnet. Dabei befinden sich - sofern nicht anders notiert - alle Neuronen, die sich eine Filtermatrix teilen, in einer Zeile. Da ich in jeder faltenden Schicht genau 256 verschiedene Filtermatrizen benutze, gibt es in allen Matrizen der faltenden Schichten 256 Zeilen.

2.4 Methoden zur visuellen Analyse des künstlichen neuronalen Netzes

Ein wichtiger Bestandteil meiner Analysen des ANNs ist die Veranschaulichung von Daten. Dabei spielen sogenannte Heatmaps eine große Rolle, die ich zur Visualisierung von Matrizen verwende. Diese werden dann als ein Bild dargestellt, das genau die Maße der Matrix in Pixeln hat. Die Pixel werden anschließend abhängig von den Werten der Matrix eingefärbt. Um die Heatmap übersichtlicher zu machen, verwende ich sogenannte geclusterte Heatmaps, die häufig in der Bioinformatik zum Einsatz kommen. Hier werden die Spalten und Zeilen der Matrix umsortiert, sodass Cluster in den Daten erkennbar werden.

2.5 Verwendete 'Werkzeuge'

Im letzten Jahr hatte ich mich aus Effizienzgründen dazu entschieden, für die Konstruktion des ANNs das Framework TensorFlow [Abadi et al., 2015] zu verwenden. Da ich jetzt das mit TensorFlow trainierte ANN untersuche, nutze ich auch weiterhin dieses Framework mit der Programmiersprache Python. Des weiteren nutze ich die Softwarebibliotheken Matplotlib [Hunter, 2007] für einfache Visualisierungen, NumPy [van der Walt et al., 2011] für die Verarbeitung

multidimensionaler Daten, Pandas [McKinney, 2010] für die Verarbeitung tabellarischer Daten, Plotly [Plotly Technologies Inc., 2015] für aufwändige Visualisierungen und Seaborn [Waskom et al., 2017] für die Erstellung geclusterter Heatmaps. Diese Softwarebibliotheken sind für Datenanalysen mit Python weit verbreitet, da sie sehr auf Effizienz optimiert sind. Programmiert habe ich in der Jupyter Notebook Umgebung, mit der Daten interaktiv analysiert werden können. Während ich kleinere Berechnungen auf meinem eigenen Laptop durchführen konnte, habe ich für größere Berechnungen, wie beispielsweise das Generieren und anschließende Verarbeiten der Aktivierungen der Neuronen für eine sehr große Menge an Eingabedaten, mehr Rechenkraft benötigt. Diese habe ich auf dem Bioinformatik-Rechencluster der Medizinischen Hochschule Hannover bekommen.

3 Die Analyse des künstlichen neuronalen Netzes

Die Methoden, die ich zur Analyse des neuronalen Netzes verwende, lassen sich in drei Schritte aufteilen:

1. Die Untersuchung der Variablen in den Filtermatrizen
2. Die Untersuchung der Aktivierungen durch Eingabedaten
3. Die Untersuchung des Einflusses der Eingabedaten auf die Aktivierungen

3.1 Analyse des neuronalen Netzes ohne Betrachtung der Aktivierungen

Der erste Schritt in der Analyse der trainierten Variablen des neuronalen Netzes ist eine Visualisierung der Filtermatrizen in Form von Heatmaps. Dies ist nur für die erste Neuronenschicht sinnvoll, da bereits in den Aktivierungen der ersten faltenden Schicht jegliche interpretierbare Information über die Eingabeschicht verloren geht. Die Abbildung 3 zeigt eine solche Heatmap einer Filtermatrix der ersten Neuronenschicht. Da die Filter der ersten Neuronenschicht sieben Zeichen abdecken, sind diese Matrizen auch sieben Felder hoch. In diesen Filtermatrizen ist die Information enthalten, wie sich Kombinationen aus sieben aufeinanderfolgenden Zeichen in der Eingabe auf die Aktivierungen der ersten Neuronenschicht auswirken. Es lässt sich an den Unterschieden in der Farbintensität in dieser Heatmap jedoch nur erkennen, dass die Buchstaben des Alphabets eine deutlich größere Rolle spielen als die Sonderzeichen. In höheren Schichten werden die Zeichenkombinationen, die in der ersten Schicht detektiert werden, dann zu längeren Zeichenfolgen und anderen komplexeren Mustern kombiniert. Analog dazu haben Neuronen der unteren Schichten bilderverarbeitender ANNs häufig nur die Aufgabe sehr einfache Strukturen, wie waagerechte Linien zu erkennen. In den höheren Neuronenschichten werden diese einfachen Muster dann zu komplexen Strukturen, wie geometrischen Formen vereint [Zeiler and Fergus, 2014].

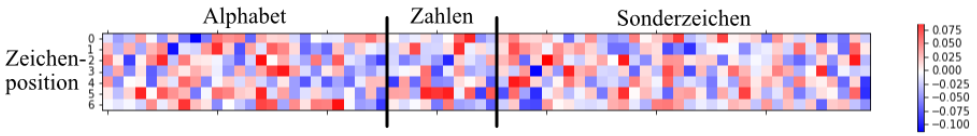

Abb. 3: Heatmap einer Filtermatrix aus der ersten Neuronenschicht. Jedes Pixel repräsentiert die Gewichtung eines Wertes der Eingabematrix für die Neuronen, die diese Filtermatrix nutzen. Da die Sonderzeichen in den Prosatexten, mit denen das ANN trainiert wurde, eine untergeordnete Rolle spielen, werden die dazugehörenden Bereiche im Training weniger stark angepasst. Dies ist hier an der geringeren Farbintensität sichtbar.

Um die Variablen und den Informationsfluss besser zu verstehen, bietet es sich an, das neuronale Netz als dreidimensionalen gerichteten Graphen darzustellen. Da das von mir verwendete

ANN aber fast 1,6 Millionen Neuronen und 330 Milliarden Verbindungen umfasst, ist es durch den hohen Rechenaufwand nahezu unmöglich darstellbar und auch für Menschen zu komplex, um es komplett zu begreifen. Deswegen habe ich mich dazu entschieden, nur jeweils die drei am stärksten gewichteten Neuronen darzustellen, die zu einem bestimmten Neuron der letzten Schicht positiv beitragen. Auch wenn diese Darstellung (Abbildung 4) schon sehr vereinfacht ist und nicht mehr alle wichtigen Informationen wie negative Verbindungen enthält, ist sie immer noch zu komplex. Allerdings lässt sich schon erahnen, wie kompliziert die Vernetzung ist, und dass es nahezu unmöglich ist, alle Verbindungen im ANN bis ins letzte Detail aufzuschlüsseln.

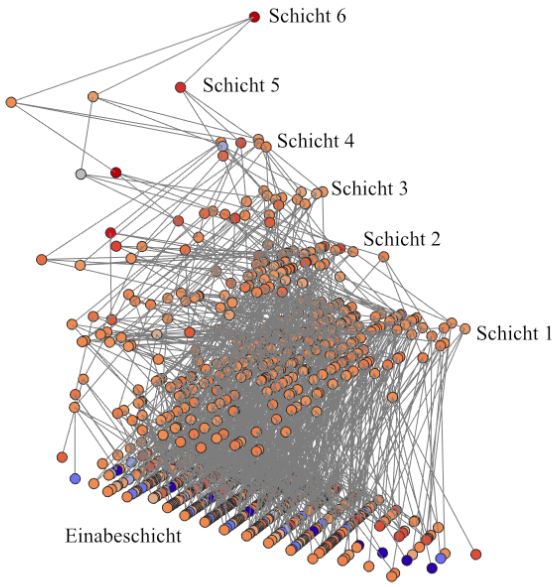

Abb. 4: Visualisierung eines Ausschnittes des ANNs als dreidimensionaler Graph. Zu sehen sind die Eingabeschicht (unten), sowie sechs Neuronenschichten. Dafür habe ich ein Neuron der sechsten Neuronenschicht ausgewählt und für jede Neuronenschicht darunter für jedes Neuron die jeweils drei am stärksten gewichteten Neuronen dargestellt. Die Farben wurden dabei abhängig von der Gewichtung der Neuronen ausgewählt. Erstellt mit Plotly.

Aus der dreidimensionalen Darstellung des ANNs geht hervor, dass die Komplexität der Visualisierungen verringert werden muss, um die Vernetzung und den Entscheidungsprozess veranschaulichen zu können. Daher habe ich den NeuronRank-Algorithmus als Werkzeug zur Datenreduktion entwickelt.

Zunächst habe ich für jeden Autor die NeuronRanks aller Schichten des ANNs berechnet. In Abbildung 5 sind die NeuronRank-Matrizen der obersten faltenden Schicht und der Eingabeschicht des ANNs für jeweils jeden Autor als Heatmap zu sehen. Diese geben schon die ersten Hinweise auf die Anpassungen des neuronalen Netzes während des Trainings. In den berechneten Matrizen für die Eingabe sind senkrechte Linien zu erkennen, je ein Buchstabe scheint also über die gesamte Matrix unabhängig von der Position nahezu den gleichen NeuronRank zu haben. Das liegt daran, dass durch die Zusammenfassende letzte Neuronenschicht, also das Fully-connected-Layer, jegliche Information über die Position beim Zurückrechnen verloren geht. Die so entstandenen senkrechten Linien lassen nur vage Aussagen über die Häufigkeiten einzelner Buchstaben zu und geben keinen Einblick in Muster, die das neuronale Netz zur Entscheidung verwendet. Erkennbar ist außerdem, dass die Heatmaps der obersten Neuronenschicht eine hohe Farbintensität aufweisen. Dies bedeutet, dass die einzelnen Neuronen für den jeweiligen Au-

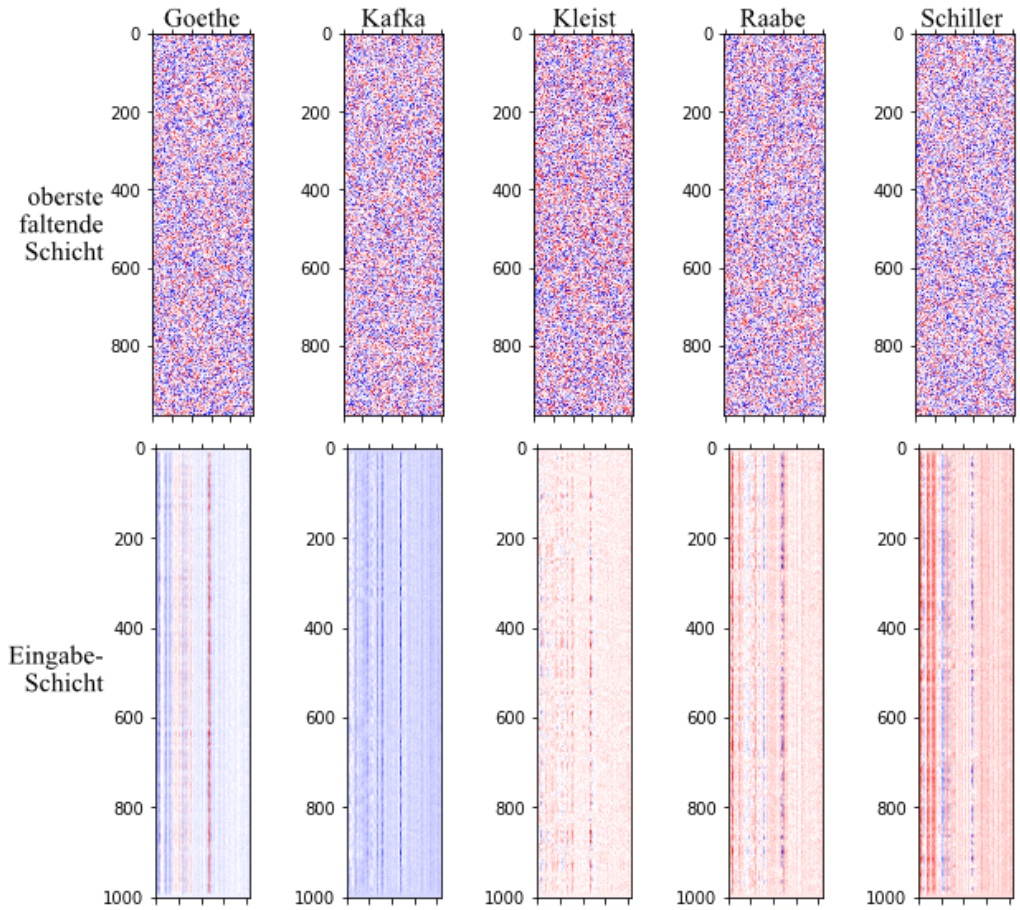

Abb. 5: Die NeuronRanks des letzten convolutional Layers, sowie der Eingabematrix für jeden Autor als Heatmap. Die Werte der Heatmap repräsentieren das Gewicht der zugehörigen Neuronen für den Wert des zum Autor gehörenden Elements im Ausgabevektor des neuronalen Netzes.

tor sehr diskriminierend wirken, obwohl das Gesamtbild sehr chaotisch wirkt. Die Unterschiede zwischen den NeuronRank-Matrizen der obersten Neuronenschicht sind im weiteren Verlauf zu untersuchen.

Die Summen der NeuronRanks derjenigen Neuronen, die denselben Filter besitzen, in Abbildung 5 also in einer Zeile stehen, werden in Abbildung 6 als geclusterte Heatmap dargestellt. Jede Spalte steht hier für einen Filter dieser Schicht, jede Zeile für einen Autor. Vergleicht man die Werte in einer Spalte, sieht man die Unterschiede der Gewichtung eines Filters für die verschiedenen Autoren. Es ist auffällig, dass es Filter in der letzen faltenden Schicht gibt, die für jeweils einen Autor sehr stark positiv (rot), für alle anderen Autoren jedoch eher neutral (weiß) oder sogar negativ (blau) gewichtet sind. Die meisten Filter lassen sich jedoch nicht auf diese Weise eindeutig einem Autor zuordnen. Zudem sind viele der diskriminierenden Filter in ihren Mustern nahezu identisch, sodass zu vermuten ist, dass hier ein hohes Potenzial der Reduktion der Komplexität des ANNs steckt.

3.2 Analyse des neuronalen Netzes mit Betrachtung der Aktivierungen

Zur Vervollständigung der Analyse des ANNs war es jetzt notwendig die Aktivierungen der Neuronen zu betrachten.

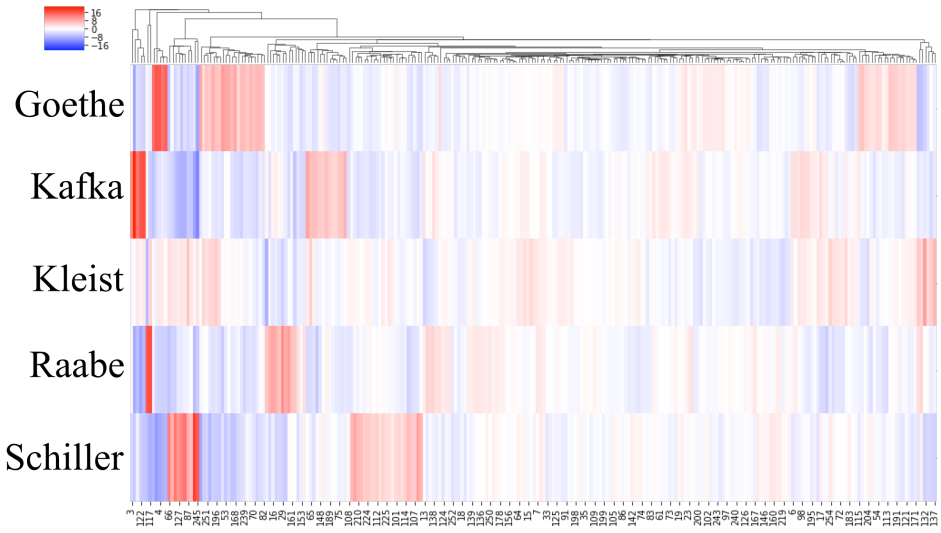

Abb. 6: Geclusterte Heatmap der summierten NeuronRanks derjenigen Neuronen der obersten Schicht, die sich einen Filter teilen. Für jeden Autor wurden separat NeuronRanks berechnet. Die Spalten stehen hier für die einzelnen Filter, die Zeilen für die Autoren.

3.2.1 Analyse der Aktivierungen ohne Betrachtung der Eingaben

Zur Analyse der Aktivierungen der Neuronen ohne Betrachtung der Eingaben habe ich alle Trainings- und Validierungstexte in das ANN eingegeben, die Aktivierungen gespeichert und für jeden Autor die aufsummierten Aktivierungen der einzelnen Neuronen untersucht. Hierbei fiel auf, dass durch die überlappenden Eingabetexte alle Neuronen einer Schicht in einer Zeile ungefähr gleich stark aktiviert wurden. Ich habe also durch diesen Ansatz jegliche Information über die Positionen der aktivierten Neuronen verloren. Dadurch war es möglich, ohne zusätzlichen Informationsverlust wie bei der Analyse der Filtermatrizen die Aktivierungsmatrizen der obersten faltenden Schicht zeilenweise aufzusummieren. Aus diesen Daten konnte ich eine geclusterte Heatmap (Abbildung 7) erstellen. Wie auch in der geclusterten Heatmap der Filtermatrizen steht in dieser Abbildung jede Zeile für einen Autor und jede Spalte für eine Gruppe von Neuronen, die sich einen Filter teilen.

Aus den Abbildungen 6 und 7 entsteht die Vermutung, dass einzelne Filter ohne Genauigkeitsverlust aus dem ANN entfernt werden können. Um dies genauer zu untersuchen, habe ich einen Teil der Aktivierungsdaten der obersten Neuronenschicht weiterführend analysiert. Hierzu wurden die Neuronen für jeden Autor entsprechend des NeuronRanks in vier Gruppen aufgeteilt. Für diese vier Gruppen habe ich die Aktivierungen der darin enthaltenen Neuronen für jeden eingegebenen Text gemittelt und eine Heatmap (Abbildung 8) erstellt. In diesen Heatmaps steht jede Spalte für einen Eingabetext, jede Zeile für einen Autor. Da die Eingabetexte nach Autor geordnet sind, lässt sich in den ersten beiden Heatmaps (extreme NeuronRanks) jeweils gut eine Diagonale von oben links nach unten rechts erkennen. Entsprechend der intuitiven Vermutung ist diese für die obersten 20% rot (hohe Aktivierung) und für die untersten 20% eher hell bzw. blau (geringe Aktivierung). Es lässt sich also gut erkennen, dass sich die Neuronen der äußeren NeuronRank-Werte gut voneinander unterscheiden lassen. Anders ist das bei der dritten (positive NeuronRanks) und vierten (negative NeuronRanks) Heatmap. Hier ist keine Diagonale zu erkennen. Entsprechend sind nicht nur die Neuronen mit dem höchsten oder niedrigsten NeuronRank für die Entscheidung des ANNs verantwortlich, sondern das Zusammenspiel aller Neuronen.

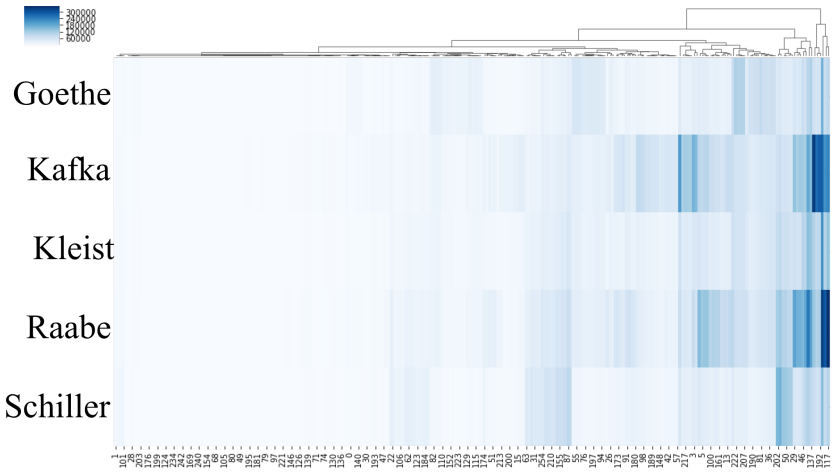

Abb. 7: Geclusterte Heatmap der Aktivierungen der Neuronen der obersten faltenden Schicht. Für alle Trainings- und Evaluierungstexte wurden die Aktivierungen gespeichert und zeilenweise aufsummiert. Die Spalten stehen für jeweils die Neuronen, die sich einen Filter teilen, die Zeilen für die Autoren.

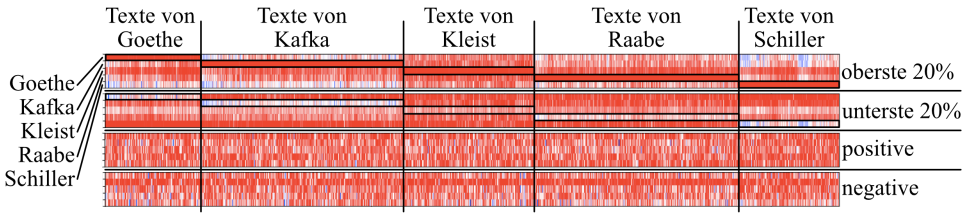

Abb. 8: Heatmaps von gemittelten Aktivierungen für vier verschiedene Gruppen von Neuronen. In jeder Matrix finden sich die Werte für jeweils eine Gruppe von Neuronen. Die Gruppen sind: alle Neuronen mit den höchsten (erste Heatmap) bzw. niedrigsten (zweite Heatmap) 20% der NeuronRanks und alle Neuronen mit positivem (dritte Heatmap) bzw. negativem (vierte Heatmap) NeuronRank für den jeweiligen Autor des Eingabetextes. In jeder Matrix steht jede Spalte für einen Eingabetext und jede Zeile für einen Autor, für den die NeuronRanks berechnet wurden.

3.2.2 Analyse des Einflusses der Eingabedaten

Um den Zusammenhang zwischen Filtern und Eingaben zu untersuchen, habe ich aus den bisherigen Analysen zwei Filter ausgesucht, deren zugehörige Neuronen ohne Berücksichtigung der Position im Text besonders für jeweils einen Autor aktiviert werden. Vergleicht man die NeuronRank-Matrizen der Eingabe für diese Filter, so fallen direkte Unterschiede auf (Abbildung 9)

Filter 42 (linke Heatmap) An den waagerechten Linien ist deutlich zu erkennen, dass vor allem einzelne Zeichen die Aktivierung der zugehörigen Neuronen stark beeinflussen. Die Zeile 37 steht für das Komma, welches durch die positiven NeuronRanks stark aktivierend auf die Neuronen des Filters wirkt. Wie zu erwarten kommt in den Textausschnitten, die die Neuronen dieses Filters besonders stark aktivieren, sehr häufig ein Komma vor.

Filter 152 (rechte Heatmap) Anders als beim Filter 42 reagieren die Neuronen dieses Filters nicht auf einzelne Zeichen, sondern auf komplexere Zeichenfolgen. Dies ist in der Heatmap an dem komplexeren Muster zu erkennen, welches jedoch nicht ohne weiteres auf das tatsächliche Eingabemuster schließen lässt, welches die Aktivierung der Neuronen beson-

ders steigen lässt. In den zu großer Aktivierung der Neuronen gehörenden Eingabetexten lassen sich hierzu allerdings Hinweise finden. Das Wort 'Prinz' aber auch die bestimmten Artikel der deutschen Sprache kommen in diesen Texten gehäuft vor. Was davon entscheidend ist, ist allerdings unklar.

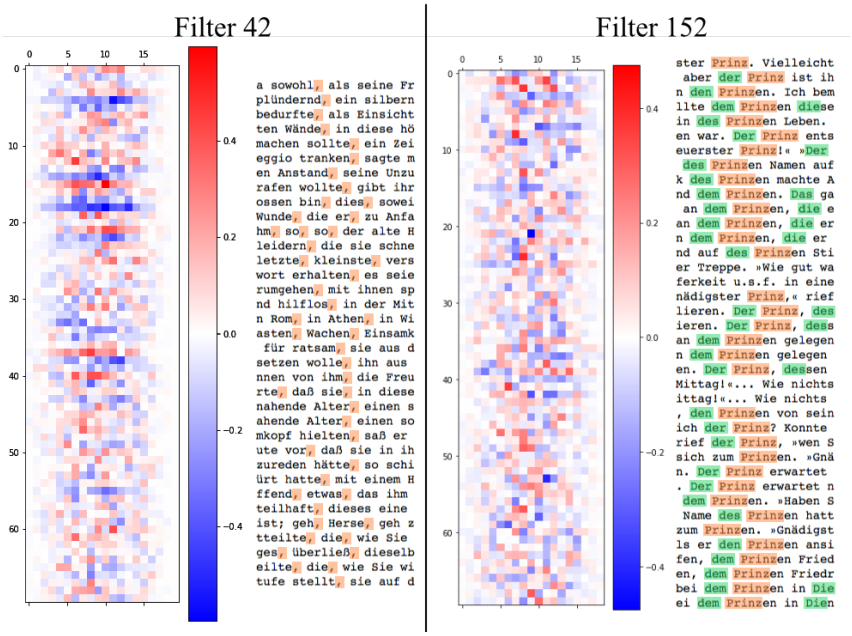

Abb. 9: Heatmaps der NeuronRank-Matrizen der Eingabe für zwei Filter der obersten faltenden Neuronenschicht. Jeweils rechts daneben befinden sich diejenigen Texte, die die Neuronen dieser Filter am stärksten aktivieren.

Da die NeuronRank-Matrizen nicht immer eine Vorhersage der Muster, die für den Entscheidungsprozess relevant sind, zulassen, ist die Untersuchung der Eingabetexte unerlässlich. Da aus der Darstellung der Eingabetexte in Abbildung 9 nicht eindeutig das Muster hervorgeht, ist es nötig die Filter noch weiter zu untersuchen und die Ergebnisse zu verifizieren. Dazu habe ich mithilfe von sogenannten Zeichen-n-Grammen gearbeitet. Diese kurzen Zeichenfolgen der Länge n (hier $n = 7$) habe ich aus den bereits untersuchten Texten gebildet und die Aktivierung der Neuronen der beiden Filter durch diese Zeichenfolgen untersucht. Durch die Länge sieben ist es möglich, das Vorkommen und die Umgebung der oben benannten Muster zu betrachten. Die Abbildung 10 zeigt die Aktivierungen der Neuronen der Filter durch die Zeichen-n-Gramme. Dazu habe ich die Sicherheitswahrscheinlichkeit berechnet, dass die Zeichenfolgen tatsächlich das untersuchte Muster enthalten, und die Punkte entsprechend der Sigma-Umgebung eingefärbt. Die umkreisten Punkte gehören zu den neben dem Plot aufgelisteten Zeichenfolgen, die jeweils mit einer Sicherheitswahrscheinlichkeit von $> 5\sigma \equiv 99,99966\%$ gut zur Unterscheidung der Filter geeignet sind. An ihnen lässt sich folgendes ablesen:

Filter 42 Wie bereits oben vermutet, ist das Komma das entscheidende Zeichen für diesen Filter. Insbesondere in Verbindung mit 'he' sorgt es für eine hohe Aktivierung der zu diesem Filter gehörenden Neuronen.

Filter 152 Durch die Darstellung wird deutlich, dass das Wort 'Prinz' für diesen Filter wichtig ist. Anders als oben finden sich hier keine Artikel mehr, sodass hiermit klar wird, welches das entscheidende Muster für die Aktivierung der zu diesem Filter gehörenden Neuronen ist.

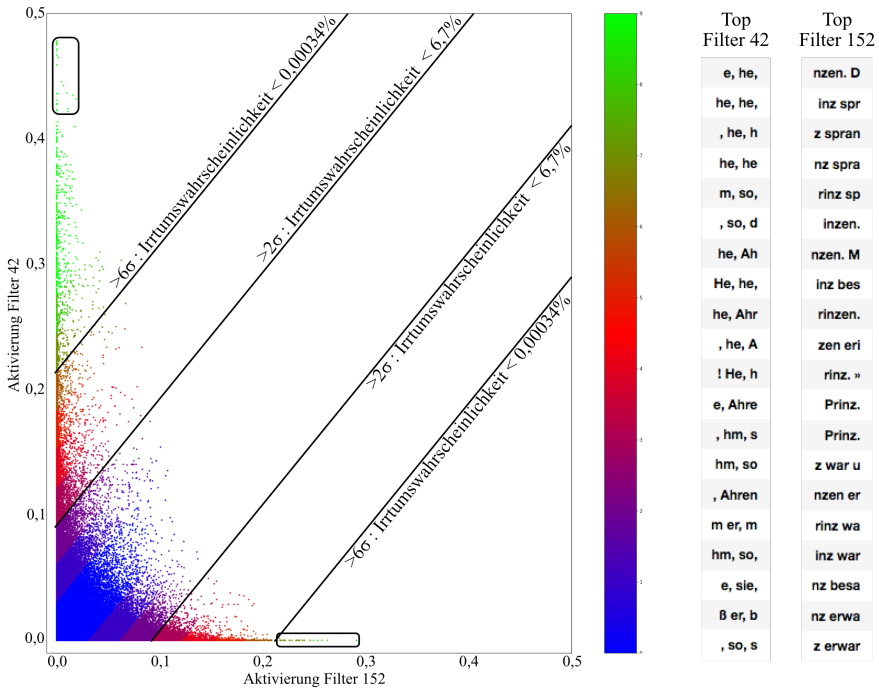

Abb. 10: Vergleich zweier Neuronengruppen, die sich einen Filter teilen, mithilfe von n-Grammen. Jeder Punkt repräsentiert ein n-Gramm der Länge sieben. Die Punkte sind eingefärbt nach Sigma-Umgebung, sodass sich direkt ablesen lässt, wie sicher es ist, dass einzelne Punkte zur Unterscheidung der Filter verwendet werden können. Neben dem Plot befinden sich die Listen derjenigen n-Gramme, die mit größter Wahrscheinlichkeit die sind, an denen sich die mit den Filtern zu erkennenden Muster ablesen lassen. Die dazugehörenden Punkte sind im Plot umkreist.

Eine gänzlich andere Methode an für die Entscheidungen des ANNs wichtige Neuronen zu gelangen, ist die Verwendung des NeuronRank-Algorithmus für das Täuschen des ANNs. Damit ist gemeint, dass eine von einem ANN korrekt klassifizierte Eingabe durch eine möglichst geringfügige Änderung nicht mehr richtig zugeordnet werden kann. Zu dieser Herangehensweise, die die potenzielle Unzuverlässigkeit von ANNs aufzeigt, gibt es in letzter Zeit immer mehr Veröffentlichungen [Su et al., 2017] [Jarmul, 2017]. Mithilfe des NeuronRank-Algorithmus habe ich bereits erste Versuche zum Täuschen des von mir untersuchten ANNs durchgeführt, da diese Methode für den Entscheidungsprozess wichtige Muster deutlich macht. Da eine ausführliche Erläuterung der Methode und der Ergebnisse allerdings den Umfang dieser Arbeit überschreiten würden, werde ich bei der Präsentation meiner Arbeit darauf gesondert eingehen.

4 Diskussion

Ich habe mir für diese Arbeit drei Fragen zu dem von mir entwickelten künstlichen neuronalen Netz gestellt, die ich mit meinen Analysen versucht habe zu beantworten:

1. Wie lässt sich der Erfolg des neuronalen Netzes in der Autorenerkennung erklären?
2. Welche Kriterien kommen im Entscheidungsprozess meines ANNs zum Einsatz und stimmen sie mit denen herkömmlicher Systeme überein?
3. Ergeben sich durch die Beantwortung der Fragen Möglichkeiten zur Bewertung des ANNs?

Um mich der ersten Frage zu nähern, habe ich zunächst versucht, eine Herangehensweise, die für die Analyse bildverarbeitender ANNs genutzt wird, anzuwenden. Das Ergebnis war wie intuitiv erwartbar nicht zufriedenstellend. Anstelle der ersten tatsächlich genutzten Muster konnte ich nur feststellen, dass die Buchstaben des Alphabets für die Entscheidung eine deutlich größere Rolle spielen, als Sonderzeichen. Mit einem Versuch das ANN dreidimensional darzustellen konnte ich zeigen, dass einfache Visualisierungen nicht für das Verstehen des Entscheidungsprozesses meines ANNs geeignet sind. Die Vernetzung der Neuronen ist zu komplex, weswegen ein Verständnis der Funktionsweise nicht ohne weiteres erlangt werden kann.

Ich habe den NeuronRank-Algorithmus von Grund auf selber entwickelt und implementiert, um die Informationen über die Struktur des ANNs zu vereinfachen. Bei der Entwicklung habe ich mich von dem PageRank-Algorithmus inspirieren lassen, der Internetseiten einen Relevanzwert auf Basis der Vernetzung zuordnet. Der NeuronRank-Algorithmus hat eine sehr ähnliche Funktionsweise, wie der PageRank-Algorithmus. Er ordnet jedem Neuron einen Wichtigkeitswert zu und sorgt dadurch dafür, dass eine Ordnung im ANN erkennbar wird. Hierzu müssen die Ausgaben des Algorithmus aber noch durch weitere Methoden der Statistik und Visualisierung weiterverarbeitet werden. Mithilfe des NeuronRank-Algorithmus lässt sich dann untersuchen, welche Eingabemuster die Neuronen besonders aktivieren.

Bei der ersten durch den NeuronRank-Algorithmus gestützten Analyse konnte ich die ersten Auswirkungen des Trainings des ANNs feststellen. Hierbei wurde deutlich, dass ich das Eingabealphabet stark verkleinern und so die Datenmenge und die Berechnungsgeschwindigkeit vergrößern kann. Bei einer Trainingsdauer von mehreren Tagen sind auch solche kleine Effizienzsteigerungen wichtig zu erkennen. Da sich diese Methode leicht auch auf andere neuronale Netze anwenden lässt, könnte sie dabei helfen, den durch die große Menge benötigter Rechenkraft hohen Stromverbrauch weltweit in vielen Anwendungen zu reduzieren.

In weiteren Untersuchungen bin ich darauf gestoßen, dass sich die Neuronen sehr einfach mithilfe ihrer Gewichtungen für die verschiedenen Autoren gruppieren lassen. So gibt es Gruppen von Neuronen, die besonders für einzelne Autoren wichtig sind. Hier zeigte sich das erste Mal, dass Ordnung in der großen unübersichtlichen Struktur des ANNs herrscht. Das Gruppieren der Neuronen scheint ein wichtiger Schritt zu einem großen Verständnis der Entscheidungsfindung zu sein. Bei einer Untersuchung der Aktivierungsdaten zeigte sich ein ähnliches Bild. Auch hier wurden Gruppen von Neuronen erkennbar, die direkt zur Unterscheidung zwischen den Autoren verwendet werden können. Dadurch entstand der Verdacht, dass einige Neuronen überflüssig sein könnten. Da die Filter auf unterschiedliche Muster gleichermaßen reagieren können, könnten eventuell einige Filter aufgrund ihrer Redundanz gestrichen werden. Dadurch würde sich wie auch bei der Verkleinerung des Eingabealphabets eine Effizienzsteigerung ergeben. Die Gruppen, die durch die Analyse der Aktivierungen erkennbar wurden, stimmen aber nicht gänzlich mit denen überein, die durch die Analyse der Filtermatrizen erkennbar wurden. Diese Nicht-Übereinstimmung lässt sich damit erklären, dass der NeuronRank-Algorithmus immer davon ausgeht, dass alle Verbindungen gebraucht werden, da alle Neuronen aktiviert sind. Dies ist jedoch aufgrund der Aktivierungsfunktion der Neuronen nicht der Fall: Sobald ein Neuron negativ aktiviert wird, wirkt es nicht hemmend auf die Aktivierung des nächsten Neurons, sondern wird gleich Null gesetzt. Entsprechend wird der Wert der Aktivierung für die Summe, die im nächsten Neuron gebildet wird, einfach nicht mehr verwendet - die Verbindung bzw. das Gewicht ist unwichtig.

Mithilfe einer neuen auf den NeuronRank-Daten basierenden Gruppierung der Neuronen fertigte ich Visualisierungen der Aktivierungen für diese Gruppen an. Hierbei wurde deutlich, dass keineswegs die Neuronen mit extremen NeuronRanks wichtig für die Entscheidungsfindung sind,

sondern vielmehr alle Neuronen und alle Filter zusammen die Entscheidung ausmachen. Erst wenn alle Neuronen am Prozess beteiligt sind, gelingt die Vorhersage. Sowohl die Möglichkeit, die Neuronen zu ordnen, als auch das Wissen, dass die Gewichtung der Neuronen allein keine Hilfe ist, zeigt, dass die Erforschung des 'Denkprozesses' des neuronalen Netzes nicht aussichtslos ist. Wichtig ist auch, dass sich hierdurch zeigt, dass die Variablen alleine nicht viel Information über das neuronale Netz bieten, obwohl sie genau das sind, was das ANN ausmacht.

Um genaue Aussagen über die für den Entscheidungsprozess wichtigen Muster treffen zu können, habe ich einzelne Filter genauer untersucht. Hierbei ist aufgefallen, dass sich mithilfe des NeuronRank-Algorithmus nur eingeschränkt Vorhersagen über die Eingaben, die ein Neuron aktivieren, treffen lassen. Auch lässt sich das Auftreten bestimmter Buchstaben oder Muster in den Eingabedaten mithilfe der NeuronRank-Matrizen nicht immer erklären. Dadurch wird meine Hypothese bestätigt, dass die Variablen für sich allein nicht ausreichend zur Analyse sind und die Aktivierungsdaten immer mit hinzugezogen werden sollten. Mithilfe einer Verifizierungstechnik, die auf Zeichen-n-Grammen basiert, war es mir möglich, zwei Muster als entscheidend ausmachen zu können. Die Kombination aus der Anwendung des NeuronRank-Algorithmus und dieser Verifizierungstechnik kann für alle Neuronen angewandt werden, um den Entscheidungsprozess des ANNs in seiner Gesamtheit zu verstehen. Dies ist allerdings mit einem enormen Zeitaufwand verbunden, weswegen ich noch nicht so weit fortschreiten konnte. Insbesondere die Analyse der von mir noch nicht untersuchten Neuronenschichten ist ein sehr interessantes Ziel, da dies für einen größeren Überblick über die Funktionsweise sorgen würde. Auch erscheint es sinnvoll nicht nur einzelne Filter, sondern ganze Filtergruppen vergleichend zu untersuchen. Diese Gruppen könnten beispielsweise die Neuronen sein, die sich in den bereits beschriebenen Untersuchungen eindeutig einem Autor zuordnen lassen.

Abschließend habe ich mit Versuchen zum Täuschen des neuronalen Netzes mithilfe des NeuronRank-Algorithmus erste Erfolge erzielt. Diese Erfolge, also das erfolgreiche Täuschen bei nur kleiner Änderung der Eingabedaten, zeigt, dass einige Muster, nach denen das ANN in der Eingabe sucht, essenziell für die Funktion des Netzes sind. Hier muss allerdings noch weiter erforscht werden, welche Muster das sind, was aufgrund der großen zu untersuchenden Datenmenge ein sehr rechenaufwändiger Prozess ist. Trotzdem sollte man diese Herangehensweise nicht aus den Augen verlieren, da insbesondere in der Forschung der Belastbarkeit von neuronalen Netzen die Suche nach Möglichkeiten des effizienten Täuschens immer stärker verfolgt wird. Entsprechend einfach wird es in Zukunft sein, mithilfe dieser täuschenden Eingaben und den daraus resultierenden Aktivierungen der Neuronen herauszufinden, welche Muster die kritischen für die Entscheidungsfindung sind. Die Herangehensweise des Täuschens bietet eine sehr gute Möglichkeit die Zuverlässigeit von ANNs zu bewerten.

Die Verbindung aus der Analyse der Täuschungsmöglichkeiten und meiner in dieser Arbeit beschriebenen Analyse von ANNs lassen sich verwenden, um weitere textverarbeitende ANNs untersuchen und bewerten zu können. Es gilt zu untersuchen, ob die präsentierten Methoden auch auf andere ANNs, wie beispielsweise bildverarbeitende, anwendbar sind. Ist dies der Fall, so lassen sich die Methoden nutzen, um für eine umfangreiche Verbesserung der Sicherheit zu sorgen, sobald ANNs verwendet werden. So könnten beispielsweise fehlerhafte Entscheidungen, wie sie jüngst bei der Steuerung autonomer Autos aufgetreten sind, bereits im Vorfeld erkannt und verhindert werden.

Ich plane in naher Zukunft meine Arbeit in einem wissenschaftlichen Journal zu veröffentlichen, um meine Methoden einem breiteren Publikum zur Verfügung zu stellen. Damit dies auch ohne Veröffentlichung geschehen kann, werde ich ein GitHub Repository mit den von mir erstellten Jupyter Notebooks veröffentlichen. Dadurch werden meine Untersuchungen leichter verständlich und können von allen Anwendern künstlicher neuronaler Netze leicht auf andere ANNs ange-

wandt werden. Dies ist auch im Sinne der offenen, für alle zugänglichen Wissenschaft und fördert das Verstehen von künstlichen neuronalen Netzen.

5 Zusammenfassung

Mein Ziel war es, den Erfolg des neuronalen Netzes in der Autorenerkennung zu erklären. Außerdem wollte ich herausfinden, ob sich das neuronale Netz mithilfe dieser Informationen bewerten lässt. Meine Analysen zeigten, dass das neuronale Netz unter anderem auf Wörter, aber auch auf die Verwendung von Kommata achtet. Diese Muster in den Eingabedaten werden auch von herkömmlichen Programmen für die Autorenerkennung verwendet. Für meine Analysen habe ich einen selbst entwickelten Algorithmus, der die Grundlage für die Verwendung bestehender Methoden darstellt, benutzt. Allerdings sind für ein vollständiges Verstehen der ANNs weiterführende Untersuchungen notwendig, die mit meiner Herangehensweise durchgeführt werden können. Noch nicht erklären konnte ich das Auftreten von Fehlern des neuronalen Netzes. Hier werde ich also in Zukunft auch noch weiter forschen. Neben Erkenntnissen über die Funktionsweise des ANNs konnte ich Möglichkeiten zur Verbesserung finden. Meine Analysen sind die Grundlagen für eine ausführliche Bewertung des ANNs. Da sie vermutlich auch auf andere künstliche neuronale Netze angewandt werden können, bieten sie eine Möglichkeit für eine größere Sicherheit bei der Verwendung von ANNs zu sorgen.

Literatur

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Géron, 2017] Géron, A. (2017). Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools, and Techniken für intelligente Systeme.

[Harley, 2015] Harley, A. W. (2015). An interactive node-link visualization of convolutional neural networks. In *International Symposium on Visual Computing*, pages 867–877. Springer.

[Hunter, 2007] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

[Jarmul, 2017] Jarmul, K. (2017). Deep learning blindspots. 34th Chaos Communication Congress.

[Liu et al., 2017] Liu, M., Shi, J., Li, Z., Li, C., Zhu, J., and Liu, S. (2017). Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):91–100.

[McKinney, 2010] McKinney, W. (2010). Data structures for statistical computing in python. In van der Walt, S. and Millman, J., editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56.

[Page et al., 1999] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab.

[Plotly Technologies Inc., 2015] Plotly Technologies Inc. (2015). Collaborative data science.

[Sanderson, 2017] Sanderson, G. (2017). Gradient descent, how neural networks learn — Chapter 2, deep learning. `https://www.youtube.com/watch?v=IHZwWFHWa-w`.

[Scheithauer, 2017] Scheithauer, T. (2017). Shakespeare oder nicht? Das ist hier die Frage! - Maschinelles Lernen zur Autorenattribution. Jugend forscht Projekt der Wettbewerbsrunde 2017.

[Su et al., 2017] Su, J., Vargas, D. V., and Kouichi, S. (2017). One pixel attack for fooling deep neural networks. *arXiv preprint arXiv:1710.08864*.

[van der Walt et al., 2011] van der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

[Waskom et al., 2017] Waskom, M., Botvinnik, O., O'Kane, D., Hobson, P., Lukauskas, S., Gemperline, D. C., Augspurger, T., Halchenko, Y., Cole, J. B., Warmenhoven, J., de Ruiter, J., Pye, C., Hoyer, S., Vanderplas, J., Villalba, S., Kunter, G., Quintero, E., Bachant, P., Martin, M., Meyer, K., Miles, A., Ram, Y., Yarkoni, T., Williams, M. L., Evans, C., Fitzgerald, C., Brian, Fonnesbeck, C., Lee, A., and Qalieh, A. (2017). mwaskom/seaborn: v0.8.1 (september 2017).

[Zeiler and Fergus, 2014] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.