Deep Learning

Eigenschaften und Lernalgorithmen für rekurrente Netze

Alwine Schultze

Oktober 2023

Inhaltsverzeichnis

Abbildungsverzeichnis

Glossar

BiLSTM Bidirectional Long Short-Term Memory. 10, 11, 13

BPTT Backpropagation Through Time. 3, 4, 8

ESN Echo State Network. 11–13

FFN Feedforward Networks. 2, 3

GPT Generative Pre-trained Transformer. 13

GRU Gated Recurrent Unit. 10, 13

KI Künstliche Intelligenz. 1, 13

LLM Large Language Model. 13

LSTM Long Short-Term Memory. 1, 5–10, 12, 13, I

ML Machine Learning. 1

NLP Natural Language Processing. 8

NN Neuronale Netze. 1

RC Reservoir Computing. 11

RNN Recurrent Neural Network. 1, 3–7, 9, 11, 13, I, II

1 Einleitung

1.1 Problemstellung der Arbeit

In der Ära des maschinellen Lernens (ML) und der künstlichen Intelligenz (KI) haben neuronale Netze (NN) eine zentrale Bedeutung erlangt. Diese Modelle ermöglichen Maschinen, komplexe Muster und Beziehungen in Daten zu erkennen und Vorhersagen zu treffen, indem sie das menschliche Gehirn nachahmen (vgl. Sheikh, Prins und Schrijvers 2023, S. 31). Insbesondere rekurrente neuronale Netze (RNN) haben in Bereichen wie Spracherkennung, Maschinenübersetzung, Handschriftenerkennung und der Vorhersage von Aktienkursen oder dem Wetter revolutionäre Fortschritte erzielt (vgl. BigData-Insider 2019). Der Hauptvorteil von RNN gegenüber anderen Neuronalen Netzen ist ihre Möglichkeit sequentielle Inputdaten verarbeiten zu können, um zum Beispiel einen Aktienkursverlauf vorher zu sagen. Doch was genau macht diese Netze so mächtig und welche Herausforderungen bringen sie mit sich? Genau diese Fragen haben die hier vorliegende Arbeit motiviert.

1.2 Zielhierarchie

In dieser Arbeit liegt der Fokus darauf, das Konzept der rekurrenten neuronalen Netze eingehend zu untersuchen. Es wird nicht nur die allgemeine Struktur dieser Netze beleuchtet, sondern auch die besondere Bedeutung von LSTM-Netzen herausgestellt. Ein weiterer Schwerpunkt dieser Arbeit ist die Darstellung spezieller Varianten von RNN sowie die Betrachtung zukünftiger Perspektiven dieser Technologie.

1.3 Aufbau der Arbeit

Diese Arbeit befasst sich intensiv mit rekurrenten neuronalen Netzen (RNN) und ihren speziellen Varianten wie dem LSTM. Das Kapitel 2 *Rekurrente Neuronale Netze (RNN)* bietet eine Einführung in die Architektur von RNN sowie deren Trainingsmethodik und der Hauptproblematik beim Training. Im Anschluss widmet sich die Arbeit dem Kernthema *Long Short-Term Memory (LSTM)* im Kapitel 3, in dem sowohl der Aufbau als auch die Vorteile und Herausforderungen von LSTM behandelt werden. Kapitel 4 - *Entwicklungen und Varianten von RNNs* führt den Leser durch weitere Abwandlungen von rekkurenten neuronalen Netzen. Abschließend bietet die *Schlussfolgerung und Ausblick* aus Kapitel 5 eine Zusammenfassung der wichtigsten Erkenntnisse sowie einen Blick in zukünftige Entwicklungen und Potenziale im Bereich der RNN-Modelle.

2 Rekurrente Neuronale Netze (RNN)

2.1 Neuronale Netze

Nach Lipton, Berkowitz und Elkan (2015) sind neuronale Netze inspiriert von biologischen Modellen der Informationsverarbeitung und bestehen aus künstlichen Neuronen, die durch gerichtete Kanten verbunden sind. Diese Kanten repräsentieren in einem intuitiven Sinne die Synapsen eines biologischen neuralen Netzes (vgl. Lipton, Berkowitz und Elkan 2015, S. 6). Ein zentrales Element in diesem System ist die Aktivierungsfunktion, die jedem Neuron zugeordnet ist, oft eine Sigmoid- oder Tanh-Funktion (vlg. Schmidt 2019, S. 1). Diese Aktivierungsfunktion verarbeitet die gewichtete Summe der Eingangswerte, wobei jedes Gewicht einem spezifischen Übergang zwischen zwei Neuronen entspricht (vgl. Lipton, Berkowitz und Elkan 2015, S. 7).

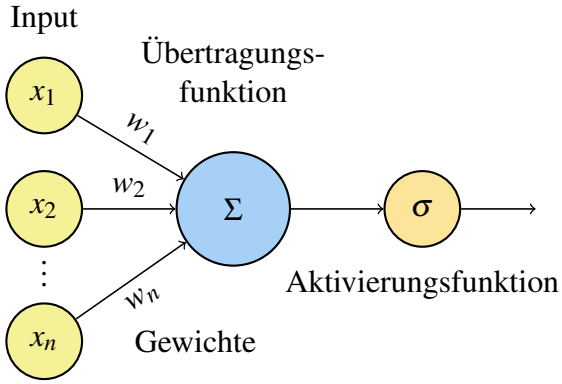

Abbildung 1: Perceptron (eigene Darstellung in Anlehnung an Lipton, Berkowitz und Elkan 2015)

Abbildung 1 visualisiert ein Perceptron, ein einfaches neuronales Netzmodell, das aus einem einzelnen künstlichen Neuron besteht. Es ist die grundlegendste Form eines neuronalen Netzes und dient oft als Einstiegspunkt für das Verständnis komplexerer Netzarchitekturen. Ein Perceptron erhält mehrere Eingangssignale $x_1, \ldots, x_n$, multipliziert jedes dieser Signale mit seinem zugehörigen Gewicht $w_1, \ldots, w_n$, und bildet daraus eine gewichtete Summe. Das resultierende Signal wird anschließend durch die Aktivierungsfunktion σ überführt, um das finale Ausgangssignal zu generieren.

Während ein einzelnes Neuron nur eine begrenzte Menge an Information verarbeiten und eine einfache Entscheidung treffen kann, ermöglicht die Verschaltung mehrerer Neuronen zu einem Netz das Lernen und Repräsentieren komplexer Muster. Die Art und Weise, wie Neuronen in einem Netz miteinander verbunden sind und wie Informationen durch das Netz fließen, bestimmt die spezifische Architektur des Netzes. Eine der grundlegendsten und am weitesten verbreiteten Architekturen in diesem Kontext sind die Feedforward-Netze (FFN). Informationen werden dabei in einer gerichteten Weise weitergegeben: von Eingabe zu Ausgabe ohne jegliche Rückkopplungen. FFNs neben genau einen Zustand auf und geben ein festes Ergebnis basierend auf diesem Zustand aus. Obwohl dies für viele Anwendungen geeignet ist, bei denen die Daten unabhängig und iden-

tisch verteilt vorliegen, eignen sie sich weniger für Szenarien, die zeitliche Abhängigkeiten zwischen Datenpunkten aufweisen. Demgegenüber stehen RNN, die sequentielle Daten verarbeiten und Informationen über mehrere Zeitschritte hinweg behalten können. Sie sind besonders darauf ausgerichtet, zeitliche Abhängigkeiten effektiv zu erkennen und zu berücksichtigen.

2.2 Rekurrente Neuronale Netze (RNN)

Nach Lipton, Berkowitz und Elkan 2015 erweitern rekurrente neuronale Netze (RNN) das Konzept von Feedforward-Netzen (FFN) durch die Einführung von Kanten, die benachbarte Zeitschritte überbrücken, wodurch dem Modell eine zeitliche Dimension hinzugefügt wird. Während in FFNs Zyklen zwischen den herkömmlichen Kanten nicht zulässig sind, erlauben RNN solche Zyklen. Dies schließt Selbstverbindungen eines Knotens über die Zeit hinweg ein (vgl. Lipton, Berkowitz und Elkan 2015, S. 10 ff.).

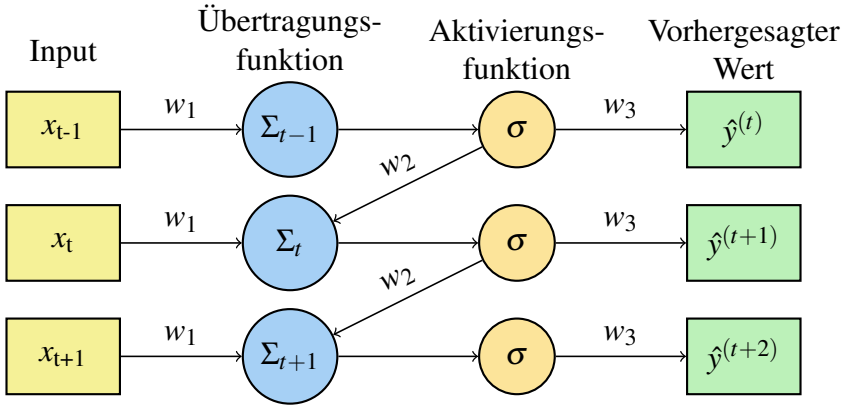

Abbildung 2: Eigene vereinfachte Darstellung eines „entfalteten“ RNN nach Ghojogh und Ghodsi 2023

Die zeitliche Dynamik eines einfachen rekurrenten Netzes lässt sich durch das *Entfalten* darstellen, siehe Abbildung 2. Anstelle einer zyklischen Betrachtung kann man sich das Netz als mehrschichtiges Modell vorstellen, wobei jede Schicht einem Zeitschritt x_t, entspricht. Dabei werden die stationären Gewichte w_1 bis w_3 über alle Zeitschritte konsistent gehalten (vgl. Lanquillon und Schacht 2023, S. 124). Dieses Modell ähnelt einem traditionellen Feedforward-Netz (FFN).

Der modifizierte Backpropagation-Algorithmus, bekannt als *Backpropagation through time* (BPTT), ermöglicht es, die optimalen Gewichtswerte während des Trainings zu bestimmen und so den Gesamtfehler des Netzes zu reduzieren (vgl. Ghojogh und Ghodsi 2023, S. 2). Der BPTT-Algorithmus stellt dabei die Differenz zwischen allen Ausgängen $\hat{y}^{(t)}$ und den Zielwerten $y^{(t)}$ über eine Verlustfunktion L dar und summiert jeden Verlustbegriff (vgl. Formel 1) der vorherigen Aktualisierungsschritte (vgl. Schmidt 2019, S. 2). Mit Hilfe der Kettenregel werden die Gradienten des Fehlers vom Ausgang des Netzes rückwärts berechnet, um die notwendigen Anpassungen der Gewichte zu ermitteln (vgl. Lipton, Berkowitz und Elkan 2015, S.11).

$$L = \sum_{t=1}^{\tau} L(\hat{y}^{(t)}, y^{(t)}) \tag{1}$$

Die Verwendung des BPTT-Algorithmus in RNN, kombiniert mit dem Teilen von Gewichten über viele Zeitschritte hinweg, kann jedoch eine Herausforderung darstellen. Zwei der signifikantesten Probleme, die dabei auftreten können sind das *Vanishing Gradient Problem* und das *Exploding Gradient Problem*, diese werden im Folgenden näher erläutert.

2.3 Vanishing/Exploding Gradient Problem

Das *Vanishing Gradient Problem* tritt häufig insbesondere in rekurrenten neuronalen Netzen (RNN) auf. Das Problem entsteht während des Trainingsprozesses, wenn der Gradient in den tieferen Schichten gegen Null konvergiert. Dies führt dazu, dass die Gewichte in diesen Schichten während des Gradientenabstiegsverfahrens kaum oder gar nicht aktualisiert werden, wodurch das Netz nicht effektiv trainiert werden kann (vgl. Lipton, Berkowitz und Elkan 2015, S. 13).

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2}$$

$$\sigma'(x) = \sigma(x)(1-\sigma(x)) \tag{3}$$

Abbildung 3 veranschaulicht die Sigmoid-Funktion und ihre Ableitung und unterstreicht die zentrale Bedeutung der Aktivierungsfunktion für dieses Problem. Es wird deutlich, dass die Ableitung der Sigmoid-Funktion, wie in den Formeln 2 und 3 dargestellt, Werte im Bereich von $[0, 0.25]$ annimmt. Als Folge davon führen Aktivierungsfunktionen in tieferen Netzschichten, die in gesättigten Zuständen (nahe 0) arbeiten, zu einem erheblich verringerten Gradienten.

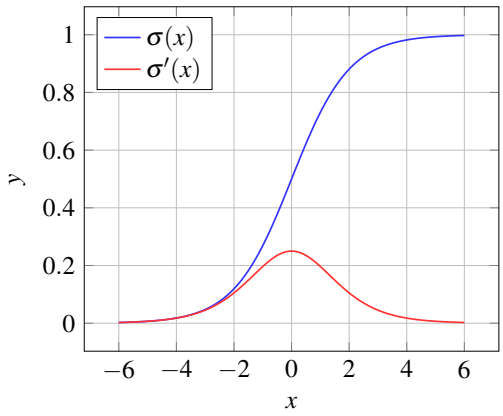

Abbildung 3: Sigmoid-Funktion und ihre Ableitung (Gradient)

Im Zuge der Backpropagation kann die Kombination dieses abgeschwächten Gradienten mit weiteren kleinen Gradienten zu einem drastischen Absinken des gesamten Gradienten führen. Demzu-

folge werden die Gewichtsaktualisierungen vernachlässigbar, was zur Manifestation des *Vanishing Gradient-Problems* führt (vgl. Lipton, Berkowitz und Elkan 2015, S. 13).

Es ist anzumerken, dass nicht ausschließlich die Sigmoid-Funktion diesem Problem unterliegt. Diverse andere Aktivierungsfunktionen können gleichermaßen beeinträchtigt werden, sofern ihre Gradienten (1. Ableitung) in bestimmten Bereichen gegen Null konvergieren.

Das *Exploding Gradient Problem* tritt dagegen eher auf, wenn die Gewichte des Netzes groß sind und es zu einer kumulativen Multiplikation großer Gradienten während der Backpropagation kommt. Das führt dazu, dass die Gewichtsaktualisierungen extrem groß werden und das Netz instabil wird. Es ist seltener, dass die Sigmoid-Aktivierungsfunktion direkt zu explodierenden Gradienten führt, da ihre Ableitungswerte begrenzt und klein sind. Jedoch können unsachgemäße Gewichtsinitialisierungen oder andere Faktoren zu diesem Problem führen, selbst wenn die Sigmoid-Funktion verwendet wird.

Die eben geschilderten Probleme beim Trainieren von RNN führten zur Einführung von Long Short Term Memory (LSTM) Netzen, die speziell entwickelt wurden, um das Problem des verschwindenden Gradienten zu behandeln. Im nächsten Abschnitt wird näher auf den Aufbau und die Funktionsweise von LSTM-Netzen eingegangen.

3 Long Short-Term Memory (LSTM)

3.1 Aufbau einer LSTM-Zelle

LSTM (Long Short-Term Memory) ist eine Art eines rekurrenten neuronalen Netzes, das entwickelt wurde, um dem Problem der verschwindenden Gradienten entgegenzuwirken (vgl. Lanquillon und Schacht 2023, S. 128). Das Hauptkonzept hinter LSTM besteht darin, zwei separate Pfade für langfristige und kurzfristige Erinnerungen zu verwenden, um Vorhersagen zu treffen. Im Gegensatz zu herkömmlichen RNN, die nur den versteckten Zustand verwenden, haben LSTM-Netze zwei Zustände: den *versteckten Zustand*, der Informationen über kurzfristige Abhängigkeiten, und den *Zellzustand*, der Informationen über langfristige Abhängigkeiten speichert (vgl. Zhang u. a. 2023). Obwohl LSTM-Einheiten oft ein Langzeitgedächtnis zugeschrieben wird, ist es strukturell treffender, dies als langes Kurzzeitgedächtnis zu bezeichnen, was den Namen „Long Short-Term Memory“ erklärt (vgl. Lanquillon und Schacht 2023, S. 129).

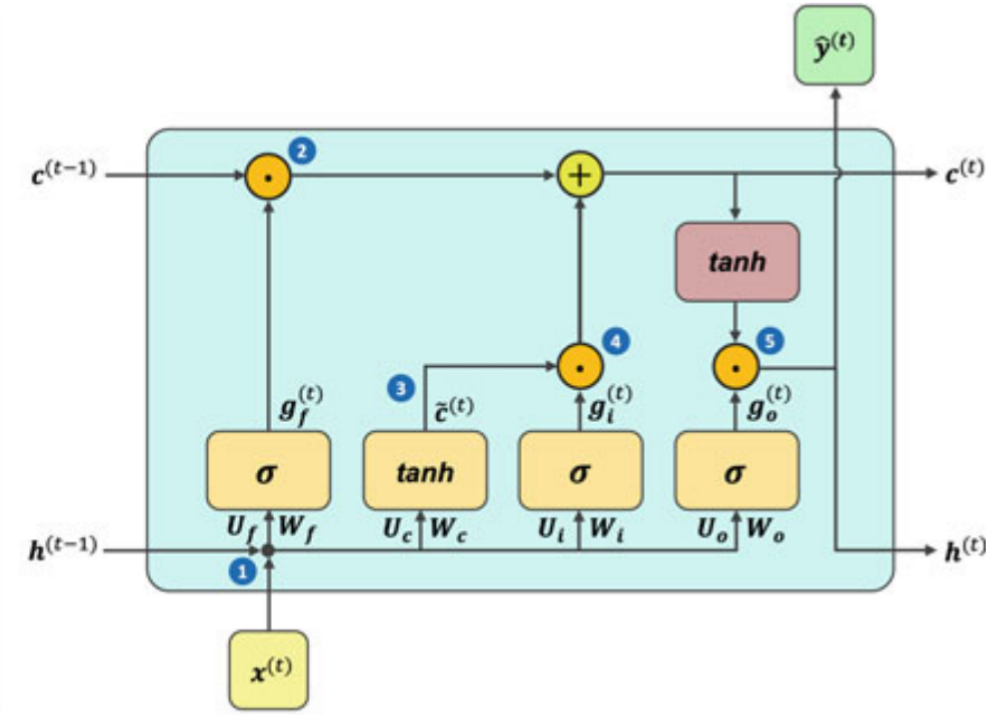

Abbildung 4: LSTM Zelle (Quelle: Lanquillon und Schacht 2023, S. 130)

LSTM-Einheiten verwenden spezielle Zellen und drei Gatter, um den Zellzustand $c^{(t)}$ zu steuern und zu modifizieren. In Abbildung 4 erkennt man, dass diese Gatter wie Ventile agieren, die Signale basierend auf ihrem Aktivierungswert durchlassen oder blockieren (vgl. Lanquillon und Schacht 2023, S. 129). In Lanquillon und Schacht (2023) wird die Struktur und Funktionsweise eines LSTM-Netzes detailliert und klar erläutert. Im Folgenden werden die zentralen Aspekte dieser Darstellung in komprimierter Form präsentiert.

(1) Konkatenation Nach Lanquillon und Schacht 2023 werden bei der Konkatenation die aktuelle Eingabe $x^{(t)}$ und der vorherige interne Zustand $h^{(t-1)}$ miteinander verbunden (vgl. Abbildung 4). Trotz dieser Verbindung werden zwei Gewichtsmatrizen W und U verwendet, um spezifische Aufgaben wie das *(2) Forget Gate*, *(4) Input Gate* und *(5) Output Gate* zu trainieren. Diese Gates regulieren, wie stark Signale durchgelassen werden. Damit ergeben sich im Vergleich zu einfachen RNN viermal so viele Gewichte, die während des Lernens angepasst werden müssen Lanquillon und Schacht 2023, S. 129 ff.).

(2) Forget Gate Das Forget Gate eines LSTM-Netzes verwendet Aktivierungswerte, bezeichnet als $g_f^{(t)}$, um zu bestimmen, welche Informationen des vorherigen Zellzustands $c^{(t-1)}$ bewahrt werden sollen (vgl. Lanquillon und Schacht 2023, S. 130). Dieses Tor ermöglicht es dem LSTM, durch Anpassen der Gewichtsmatrizen W_f und U_f selektiv bestimmte Informationen zu verwerfen, insbesondere basierend auf der aktuellen Eingabe und dem jüngsten internen Zustand. Dieser Mechanismus ist entscheidend, um das Netz effizient zu trainieren und sicherzustellen, dass es nur relevante Informationen über längere Sequenzen hinweg speichert.

(3) Zellzustand Der Zellzustandskandidat in einem LSTM-Netz wird aus der aktuellen Eingabe und dem letzten internen Zustand unter Verwendung der Gewichtsmatrizen W_c und U_c berechnet (vgl. Lanquillon und Schacht 2023, S. 130 ff.). Im Gegensatz zu anderen Aktivierungen verwendet der Zellzustandskandidat die *tanh*-Aktivierungsfunktion. Dieser Kandidat spielt eine zentrale Rolle bei der Aktualisierung des tatsächlichen Zellzustands $c^{(t)}$ im Netz.

(4) Input Gate Das Input Gate bestimmt, welche Teile des Zellzustandskandidaten dem durch das Forget Gate modifizierten Zellzustand hinzugefügt werden (vgl. Lanquillon und Schacht 2023, S. 131). Es entscheidet damit, welche neuen Informationen im aktuellen Zellzustand gespeichert werden sollen.

(5) Output Gate Das Output Gate wählt spezifische Inhalte des Zellzustands für den aktuellen internen Zustand und die Ausgabe aus. Diese Inhalte werden mittels der *tanh*-Aktivierungsfunktion in den Wertebereich [-1, 1] transformiert, um sicherzustellen, dass die Zellzustandswerte innerhalb dieses Bereichs liegen (vgl. Lanquillon und Schacht 2023, S. 131). Diese Transformation ist notwendig, da sich die Zellzustandswerte über die Zeit durch wiederholtes Hinzufügen von Zellzustandskandidaten ansammeln können. Es gibt keine zusätzliche Gewichtung des transformierten Zellzustands, weshalb die Transformation ähnlich wie bei einfachen RNN durchgeführt wird.

3.2 Vorteile und Anwendungen von LSTM

Der besondere Aufbau von LSTM-Netzen ermöglicht es, Informationen über längere Zeiträume hinweg effizient zu verarbeiten, was sie besonders geeignet für die Verarbeitung von sequenzielle Daten macht. Zu ihren Hauptvorteilen gehören die Fähigkeit, lange Abhängigkeiten in Daten zu erkennen, das Problem des verschwindenden Gradienten zu überwinden und ihre Flexibilität in verschiedenen Anwendungen.

LSTM wird häufig im Rahmen von NLP-Aufgaben eingesetzt, wie z.B. maschinelles Übersetzen, Textgenerierung und Sentiment-Analyse. Sie können die Struktur und Bedeutung von Sätzen erfassen, was sie zu einer wertvollen Technologie in diesem Bereich macht. Außerdem eignen sie sich durch ihren Aufbau auch für Zeitreihenprognosen, zum Beispiel in Bereichen wie der Finanz- und Wettervorhersage. Hier können LSTM-Netze Muster in Daten über längere Zeiträume erkennen und genaue Vorhersagen treffen. Ein dritter Anwendungsbereich solcher Netze ist die Musikkomposition, hier werden sie eingesetzt, um neue Melodien oder Harmonien zu generieren, die auf bestehenden musikalischen Daten basieren.

3.3 Trainingsverfahren und Herausforderungen bei LSTM-Netzen

Backpropagation through time (BPTT) stellt eine zentrale Methode für das Training von neuronalen Netzen dar. Die Schwierigkeiten beim Training des Netzes, die durch den sogenannten *vanishing* oder *exploding* Gradienten verursacht werden, wurden schon in den Abschnitten 2.2 und 2.3 besprochen. Diese spezifischen Schwierigkeiten waren maßgeblich für die Entwicklung des LSTM-Netzes. Dennoch existieren weitere Trainingsmethoden sowie zusätzliche Herausforderungen, die bei der Schulung von LSTM-Netzen berücksichtigt werden müssen. Die nachfolgenden Abschnitte bieten einen detaillierten Einblick in weitere Aspekte des Training und welchen Problemen damit entgegengewirkt werden kann.

Gradient Clipping Während des Trainingsprozesses von neuronalen Netzen, insbesondere bei tiefen Architekturen oder bei der Verarbeitung von langen Sequenzen, können die Gradientenwerte manchmal extrem groß werden. Dieses Phänomen, bekannt als das *exploding* Gradient-Problem, kann dazu führen, dass das Modell nicht konvergiert oder sogar numerische Instabilitäten während des Trainings auftreten (vgl. Kapitel 2.3). Um diesem Problem entgegen zu wirken, wird die Technik des Gradient Clipping angewendet (vgl. Lanquillon und Schacht 2023, S. 111). Dabei werden Gradientenwerte, die einen bestimmten Schwellenwert überschreiten, auf diesen Wert begrenzt. Dies stellt sicher, dass die Gewichtsaktualisierungen während des Trainingsprozesses kontrolliert und stabil bleiben, was zu einer effizienteren und zuverlässigeren Konvergenz des Modells beiträgt.

Regularisierungsmethoden für Overfitting LSTM-Netze können, ähnlich wie andere neuronale Netze, dazu tendieren, die Trainingsdaten übermäßig genau zu adaptieren, was zu *Overfitting* führen kann. Dieses Phänomen tritt auf, wenn das Netz die spezifischen Muster und das Rauschen in den Trainingsdaten zu genau erfasst, was zu einer schlechteren Performance bei neuen, unbekannten Daten führt (vgl. Lanquillon und Schacht 2023, S. 31). Um Overfitting zu bekämpfen, werden verschiedene Regularisierungstechniken eingesetzt.

Dropout ist eine solche Technik, die auch in LSTM-Netzen angewendet werden kann. Dabei werden während des Trainings zufällig ausgewählte Neuronen deaktiviert, sodass sie keinen Einfluss auf das Vorwärts- und Rückwärtspropagieren haben. Dies fördert eine gleichmäßigere Gewichtsverteilung im Netz.

L1- und L2-Regularisierung beeinflussen die Gewichtswerte in LSTM-Netzen. L1-Regularisierung, oft als Lasso bezeichnet, fördert schlanke Modelle, indem sie bestimmte Gewichtswerte zu Null treibt, wodurch weniger wichtige Merkmale entfernt werden. L2-Regularisierung, bekannt als Ridge, bestraft große Gewichtswerte, was die Modelle glatter macht. Während L1 bestimmte Merkmale eliminiert, sorgt L2 dafür, dass alle Merkmale mit eingeschränkter Intensität beitragen. Beide Techniken modulieren die Gewichtswerte, um Overfitting zu minimieren und die Modellstabilität zu erhöhen.

Während LSTM-Netze und die genannten Regularisierungstechniken wesentliche Fortschritte in der Verarbeitung sequenzieller Daten ermöglicht haben, steht die Forschung in diesem Bereich nicht still. Es wurden weitere Varianten von RNN entwickelt, die das Ziel verfolgen die Effizienz zu steigern oder bestimmte Herausforderungen zu adressieren. Einige von ihnen werden im nächsten Kapitel beschrieben.

4 Entwicklungen und Varianten von RNNs

4.1 Gated Recurrent Unit (GRU)

Die Gated Recurrent Unit (GRU) wurde eingeführt, um adaptiv Abhängigkeiten über verschiedene Zeitskalen hinweg zu berücksichtigen. Nach Chung u. a. (2014) besitzt die GRU, anders als die LSTM, keine getrennten Speicherzellen. Zudem gibt die GRU bei jeder Aktivierung ihren gesamten Zustand preis, wohingegen die LSTM dies gezielter macht. Ähnlich dem Forget Gate der LSTM-Einheit, jedoch mit einem einfacheren Mechanismus, erlaubt das *Reset-Gate* der GRU, den zuvor berechneten Zustand komplett zu vergessen (vgl. Chung u. a. 2014, S. 4).

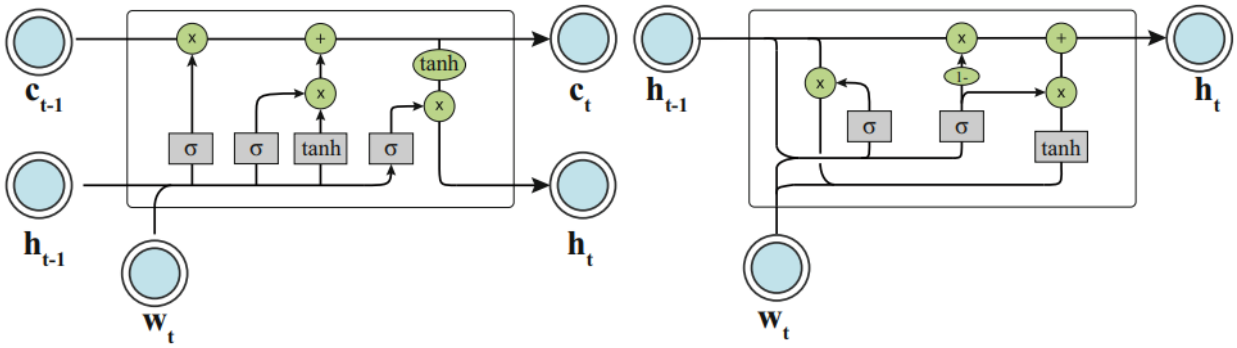

Abbildung 5: Aufbau einer LSTM und GRU im Vergleich (Quelle: Liu, Lin und Sun 2023, S. 89)

Die GRU verwendet im Vergleich zum LSTM weniger Parameter, was auch in Abbildung 5 klar ersichtlich ist. Dies führt zu einer höheren Effizienz und kann als vereinfachte Version des LSTM betrachtet werden. Dennoch ist es schwierig zu bestimmen, welche Einheit generell besser abschneidet. Vorläufige Experimente haben gezeigt, dass beide Einheiten in bestimmten Anwendungen, wie z.B. der maschinellen Übersetzung, vergleichbar abschneiden. Es bleibt jedoch unklar, ob dies auch für andere Aufgaben gilt (vgl. Chung u. a. 2014, S. 5).

4.2 Bidirectional Long Short-Term Memory (BiLSTM)

Die Bidirectional Long Short-Term Memory (BiLSTM) stellt eine Weiterentwicklung der klassischen LSTM-Architektur dar. Ihre Besonderheit liegt darin, dass sie in der Lage ist, sequenzielle Daten unter Berücksichtigung sowohl rückwärtiger als auch zukünftiger Informationen zu verarbeiten (vgl. Aggarwal 2023). Während eine reguläre LSTM Informationen lediglich in einer festen Richtung verarbeitet, besteht die BiLSTM aus zwei LSTM-Einheiten: Eine verarbeitet die Daten von Beginn bis Ende, während die andere dies in umgekehrter Reihenfolge durchführt. Der Aufbau, dargestellt in Abbildung 6, ermöglicht es, sowohl aus vorherigen als auch aus kommenden Datenpunkten innerhalb einer Sequenz zu lernen (vgl. Aggarwal 2023).

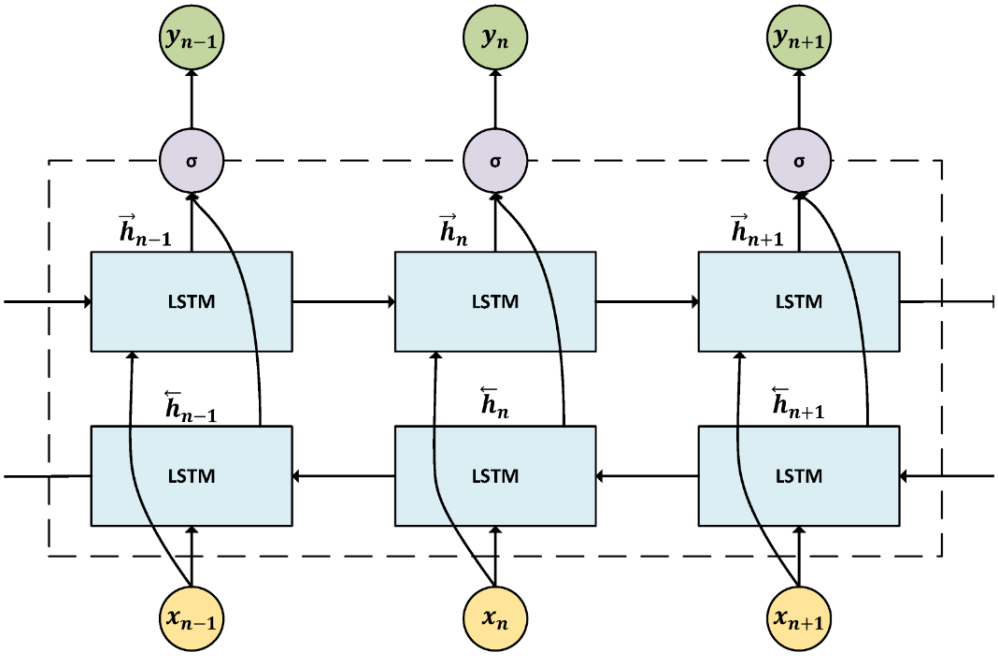

Abbildung 6: Aufbau eines bidirectionalen LSTM (BiLSTM) (Quelle: Magadán u. a. 2023, S. 2)

Dieser bidirektionale Ansatz zeigt seine Stärken insbesondere in Bereichen wie der maschinellen Übersetzung oder Textklassifikation, in denen Kontextinformationen aus beiden Richtungen von entscheidender Bedeutung sind. Obwohl die BiLSTM viele Vorteile mit sich bringt, ist ihre Leistungsfähigkeit stark von der jeweiligen Anwendung und den zugrundeliegenden Daten abhängig. Daher ist es essenziell, sowohl die BiLSTM als auch andere RNN-Modelle zu prüfen, um die optimalste Architektur für eine gegebene Problemstellung auszuwählen.

4.3 Echo State Networks (ESN)

Echo State Networks (ESN) gehören zur Familie des Reservoir Computing (RC). Sie bieten eine interessante Alternative zu den herkömmlichen Gradientenabstiegsmethoden beim Training von RNN (vgl. Sun u. a. 2020, S.1).

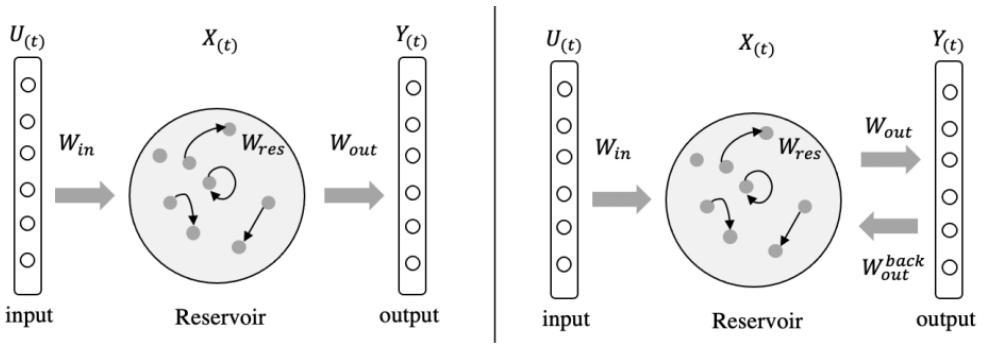

Abbildung 7: Einfaches Echo State Netz (Quelle: Sun u. a. 2020, S. 2)

In Abbildung 7 wird die Grundarchitektur eines ESN illustriert, bestehend aus Eingabe-, Reservoir-

und Ausgabekomponenten. Das Eingabesignal $U(t)$ wird über Gewichtungen W_{in} in das Reservoir überführt, welches dynamische Zustände $X(t)$ mittels interner Gewichtungen W_{res} generiert. Dieses Reservoir verarbeitet und speichert temporäre Informationen des Signals. Die Ausgabe $Y(t)$ wird durch die Gewichtungen W_{out} bestimmt. Der zweite Teil der Abbildung zeigt eine Erweiterung mit einem zusätzlichen Rückkopplungsmechanismus $W_{backout}$, der die Ausgabe rückwärts in das Reservoir leitet. Dies erweitert das Spektrum an Dynamiken, die das Netz erfassen kann. Ein Schlüsselmerkmal der ESNs, das nicht direkt in der Abbildung ersichtlich ist, ist die Echo-State-Eigenschaft, welche sicherstellt, dass der Einfluss des Anfangszustands nach einer gewissen Zeit verschwindet (vgl. Sun u. a. 2020, S.2).

ESNs haben in Kombination mit verschiedenen Neural Network Strukturen herausragende Leistungen erzielt und finden Anwendung in verschiedenen Bereichen, einschließlich Industrie, Medizin und Finanzwesen (vgl. Sun u. a. 2020, S.2). Stellt man LSTM und ESN gegenüber so zeichnen sich der LSTM-Ansatz insbesondere durch ihre Fähigkeit aus, langfristige Abhängigkeiten in Daten zu lernen, während ESNs im Reservoir Computing-Kontext eine schnelle und effiziente Trainingsmethode bieten. LSTM-Netze sind oft besser geeignet für komplexe sequenzielle Daten, während ESNs in Anwendungen mit schnellen Echtzeitvorhersagen oder begrenzten Trainingsdaten Vorteile bieten. Die Wahl zwischen LSTM und ESN hängt stark von der spezifischen Anwendung und den Datenanforderungen ab.

5 Schlussfolgerung und Ausblick

Die verschiedenen Entwicklungen und Varianten von RNN wie GRU, BiLSTM und ESN zeigen die kontinuierliche Anstrengung der Wissenschaftsgemeinschaft, die Leistung von sequenziellen Modellen zu optimieren. Während die GRU als vereinfachte und weniger parameterreiche Alternative zur LSTM entwickelt wurde, ermöglicht die BiLSTM das Lernen von Kontextinformationen aus beiden Richtungen einer Sequenz. Das ESN, welches im Kontext des Reservoir Computing steht, bietet einen effizienten Ansatz für das Training von RNN, insbesondere wenn Echtzeitvorhersagen oder begrenzte Daten im Vordergrund stehen.

Allerdings gibt es in der Welt der RNN keine Einheitslösung. Die Wahl des geeigneten Modells hängt stark von der jeweiligen Anwendung, den Daten und den spezifischen Anforderungen ab. Auch wenn Modelle wie GRU und BiLSTM in der Lage sind, lange sequenzielle Abhängigkeiten zu lernen, können sie in bestimmten Situationen, insbesondere bei extrem langen Sequenzen oder komplexen temporalen Strukturen, an ihre Grenzen stoßen. Außerdem gibt es abgesehen von den in der Arbeit genannten Abwandlungen von RNN auch zahlreiche weiter Modelle, die nicht aufgeführt wurden.

Ein Blick in die nahe Zukunft zeigt außerdem, dass die Weiterentwicklung von RNN nicht stillsteht. Die Transformer-Architektur, ursprünglich für maschinelle Übersetzungsaufgaben entwickelt, hat in den letzten Jahren aufgrund ihrer parallelen Verarbeitungsfähigkeiten und der Fähigkeit, sowohl kurz- als auch langfristige Abhängigkeiten in Daten zu erfassen, an Popularität gewonnen. Die gegenwärtigen Large Language Modelle (LLM) sind in der Regel vortrainierte Transformer-Modelle, wie zum Beispiel ChatGPT, Bart oder LLaMA.

Fortschritte in der Hardware-Technologie, wie spezialisierte KI-Chips oder Quantencomputer, könnten die Leistungsfähigkeit und Effizienz von RNN erheblich steigern. Dies könnte die Tür zu bisher unvorstellbaren Anwendungen öffnen, von extrem präzisen Wettervorhersagemodellen bis hin zu neuartigen, menschenähnlichen KI-Systemen.

Es gibt auch ein wachsendes Interesse an neuromorphen Ansätzen, die von der Arbeitsweise des menschlichen Gehirns inspiriert sind. Diese Ansätze könnten helfen, Energieeffizienz und Echtzeitverarbeitungsfähigkeiten zu verbessern und gleichzeitig die Fähigkeit von RNN zu erweitern, komplexe, dynamische Datenquellen zu verstehen.

Zusammengefasst stehen die RNN und ihre Derivate trotz der beeindruckenden Fortschritte der letzten Jahre erst am Anfang ihres Potenzials. Die Kombination aus neuen Architekturen, Hardware-Innovationen und biologischen Inspirationen lässt eine aufregende Zukunft für diese Technologie erwarten.

Literaturverzeichnis

Lanquillon, Carsten und Sigurd Schacht (2023). *Knowledge Science – Grundlagen*. Springer Fachmedien Wiesbaden. ISBN: 978-3-658-41689-8.

Liu, Zhiyuan, Yankai Lin und Maosong Sun (2023). *Representation Learning for Natural Language Processing*. 2. Aufl. Springer Singapore. ISBN: 978-981-99-1599-6.

Sheikh, Haroon, Corien Prins und Erik Schrijvers (2023). *Mission AI - The New System Technology*. Springer. ISBN: 978-3-031-21447-9.

Online-Quellen

Aggarwal, Raghav (2023). *Bi-LSTM*. Zugriff am 10. Oktober 2023. URL: https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0.

BigData-Insider, Redaktion (2019). *Was ist ein rekurrentes neuronales Netz (RNN)?* Zugriff am 28. August 2023. URL: https://www.bigdata-insider.de/was-ist-ein-rekurrentes-neuronales-netz-rnn-a-843274/.

Chung, Junyoung u. a. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. Zugriff am 07. Oktober 2023. arXiv: 1412.3555 [cs.NE].

Ghojogh, Benyamin und Ali Ghodsi (2023). *Recurrent Neural Networks and Long Short-Term Memory Networks: Tutorial and Survey*. Zugriff am 18. September 2023. arXiv: 2304.11461 [cs.LG].

Lipton, Zachary C., John Berkowitz und Charles Elkan (2015). *A Critical Review of Recurrent Neural Networks for Sequence Learning*. Zugriff am 18. September 2023. arXiv: 1506.00019 [cs.LG].

Magadán, Luis u. a. (2023). *A Robust Health Prognostics Technique for Failure Diagnosis and the Remaining Useful Lifetime Predictions of Bearings in Electric Motors*. Zugriff am 17. Oktober 2023. URL: https://www.mdpi.com/2076-3417/13/4/2220.

Schmidt, Robin M. (2019). *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*. Zugriff am 18. September 2023. arXiv: 1912.05911 [cs.LG].

Sun, Chenxi u. a. (2020). *A Review of Designs and Applications of Echo State Networks*. Zugriff am 17. Oktober 2023. arXiv: 2012.02974 [cs.LG].

Zhang, Aston u. a. (2023). *Long Short-Term Memory (LSTM)*. Zugriff am: 21. September 2023. URL: https://d2l.ai/chapter_recurrent-modern/lstm.html.