Abschlussbericht Projekt „Big Data“
Universität Hamburg

Neuronale Netzwerke in der Klimamodellierung zur Prognose eines Zeitschritts

Vorgelegt von **Tobias Machnitzki**
tobias.machnitzki@studium.uni-hamburg.de
Finn Burgemeister
finn.burgemeister@studium.uni-hamburg.de

Eingereicht am 28. August 2018

Betreuer Dr. Julian Kunkel
Wissenschaftliches Rechnen, Universität Hamburg
Tobias Finn
Meteorologisches Institut, Universität Hamburg

https://github.com/fiburg/bdp_cnn

Kurzfassung

Neuronale Netzwerke werden in den Klimawissenschaften bisher nicht viel eingesetzt und wenn, dann oft nur, um bestimmte Prozesse isoliert zu betrachten. Grund für die bisherig geringe Nutzung von neuronalen Netzwerken ist, dass diese einen neuen Ansatz darstellen und nicht nur auf physikalischen Grundgleichungen basieren. Die vorliegende Arbeit beschäftigt sich daher damit ein neuronales Netzwerk einzusetzen, welches nicht nur zur Erkennung von Mustern dient, sondern versucht die 2 m-Temperatur-Prognose eines Klimamodells hinreichend genau zu approximieren. Das Ziel ist es zu zeigen, wozu selbst einfachste neuronale Netzwerke in der Lage sind, verglichen mit einem sehr komplexen gekoppelten Klimamodell.

Hierzu wird zunächst ein sehr einfaches Modell benutzt, zur Erläuterung der Unterschiede zwischen verschiedenen Typen künstlicher neuronaler Netzwerke - CNN und LSTM. Die Ergebnisse werden untersucht und verglichen. Da das LSTM für die Prognose von Zeitlich aufgelösten Daten deutlich besser ist, sodass es für das einfache Testmodell nahezu perfekte Prognosen wiedergibt (Korrelationskoeffizient r=1, RMSE=0.09 bei einem Wertebereich von ungefähr 40), wird dieses daraufhin auf das komplexe Klimamodell angewendet. Bei der Anwendung des neuronalen Netzwerks auf die Daten des Klimamodells werden zwar nur Spitzenwerte von r=0.99 und RMSE=2,45 K erreicht, in Anbetracht der Komplexität des zu approximierenden Modells sind diese Ergebnisse jedoch entsprechend gut.

Die Geschwindigkeit neuronaler Netzwerke wird des öfteren hervorgehoben, da sie verglichen mit der Laufzeit eines Klimamodells sehr viel geringer ist. Zwar kann das neuronale Netzwerk das Klimamodell nicht ersetzen, jedoch wird das Ziel einer hinreichend genauen Approximation erfüllt.

Inhaltsverzeichnis

1 Motivation

Allgemeine gekoppelte Zirkulationsmodelle simulieren die physikalischen Prozesse in der Atmosphäre, im Ozean und am Land und berücksichtigen den Zusammenhang und Austausch dieser Systeme. Die nicht-linearen Zusammenhänge zwischen den Modellkomponenten werden über Annahmen und Annäherungen vereinfacht, aber es verbleibt ein komplexes System ohne analytische Lösung. Ein tiefgehendes Verständnis der Physik, sowie große Rechnerkapazitäten werden demnach in der Klimamodellierung benötigt und daher sind vereinfachende Ansätze wünschenswert.

Künstliche neuronale Netzwerke werden bereits erfolgreich zur Mustererkennung in großen Datensätzen im Bereichen der Industrie, Wirtschaft und Wissenschaft angewendet, obgleich das Verständnis und die Optimierung dieser eine besondere Komplexität vorweisen. Das Ziel ist die Vorhersage einer Kategorie (Klassifikation) oder Variablen (Regression). In der Meteorologie gibt es bereits erfolgreiche Anwendungen neuronaler Netzwerke. Bezenac et al. (2017) verwenden *Deep-Learning*-Methoden zur Prognose der Wasseroberflächentemperatur aus Satellitendaten. Xingjian et al. (2015) stellen ein neuronales Netzwerk - Convolutional Long short-term memory - zur Niederschlags Kurzfristvorhersage vor. Im Bereich der Klimamodellierung versuchen Gentine et al. (2018) die Parametrisierungen von Konvektion mit einem neuronalen Netzwerk zu ersetzen. Moghim und Bras (2017) verwenden ein neuronales Netzwerk zur Verbesserung des BIAS von Klimavariablen, wie der Temperatur und des Niederschlags in einer bestimmten Region. Drastische Ansätze, wie die Verwendung von neuronalen Netzwerken zur zeitliche Prognosen von meteorologischen Variablen in der Klimamodellierung gibt es jedoch noch nicht. Dabei stellt sich die Frage, ob es möglich ist mit neuronalen Netzwerken Teile eines Klimamodells hinreichend genau zu approximieren. Mit diesem Ansatz wird in dieser Arbeit versucht die nicht-linearen Zusammenhänge des komplexen Klimasystems eines Modells mit einem geeigneten neuronalen Netzwerk zu verstehen, ohne die verwendeten physikalischen Gleichungen zu kennen.

Das Projekt „Neuronale Netzwerke in der Klimamodellierung zur Prognose eines Zeitschritts“ wurde im Rahmen der Lehrveranstaltung *Projekt Big Data* der Universität Hamburg durchgeführt. Zur Beantwortung der Zielfragestellung - *Ist es möglich Klimamodelle mit Hilfe neuronaler Netzwerke hinreichend genau zu approximieren?* - wird eine Variable eines Klimamodells vorhergesagt. Der vorliegende Abschlussbericht gibt einen Überblick über das Projekt. Die Projektplanung umfasst die Methodik, verwendeten Tools und Struktur. Ein erster Ansatz entsprechend der Methodik wird im Abschnitt des Lorenzmodells verfolgt, mit der Zusammenfassung der Theorie, Datengrundlage, des neurnalen Netzwerks und der Ergebnisse. Im Abschnitt des Klimamodells wird, analog zum Aufbau des Abschnitts des Lorenzmodells, die eigentliche Zielfragestellung diskutiert. In einem abschließenden Fazit wird kurz der Wert dieser Arbeit diskutiert.

2 Projektplanung

Die Methodik des Projekts umfasst abstrakt fünf Schritte. Zur Beantwortung des Projektsziels - der Prognose eines Zeitschritts einer Klimavariable mit neuronalen Netzwerken - wurde ein geeignetes neuronales Netzwerk ausgewählt (Abschnitt 3.3). Der zweite Schritt der Methodik ist das Verstehen des ausgewählten neuronalen Netzwerks anhand eines einfachen Modells (Abschnitt 3). Das geeignete neuronale Netzwerk wurde auf ein Ensemble einer Variable des Klimamodells trainiert (Abschnitt 4). Erweiternd wurde das neuronale Netzwerk auf die Variable aus mehreren Ensemble-Member angewendet. Die Optimierung eines neuronalen Netzwerk ist eine umfassende Aufgabe und wurde zum Ende des Projekts durchgeführt. Die Struktur dieses Abschlussberichts folgt der Methodik und chronologischen Durchführung.

Das Projekt ist in *Python* implementiert, da es hierfür bereits diverse Open Source Deep-Learning-Bibliotheken gibt. Die Maschine-Learning-Bibliothek *Keras* wurde neben *Tensorflow* und *Scikit-learn* primär in diesem Projekt verwendet. *Keras* basiert auf *Tensorflow*. Als Testsystem stand der Server *Breeze* des Max-Planck-Instituts für Meteorologie zur Verfügung, der 64 Intel Xeon Prozessoren mit jeweils 2,7 GHz Taktfrequenz und einen Arbeitsspeicher von 256 GB beinhaltet, somit konnten vergleichbare Ergebnisse der Laufzeiten erzielt werden.

Das Projekt wurde gemäß der objektorientierten Programmierung in vier Klassen unterteilt. Die Modellklasse beinhaltet das neuronale Netzwerk mit Methoden zur Initialisierung und Vorbereitung der Daten der Zeitserie. Der *DataHandler* ermöglicht die Aus- und Eingabe der Modellergebnisse im NetCDF-Format (Rew und Davis 1990). Die Daten des Klimamodells liegen ebenfalls als NetCDF-Datei vor. Die Klasse *Evaluater* beinhaltet Methoden zur Evaluation der Modellergebnisse mit statistischen Größen und Abbildungen. Die Klasse *Scaler* skaliert die Daten für das neuronale Netzwerk.

3 Lorenzmodell

Um zunächst neuronale Netzwerke kennenzulernen wurde mit einem sehr einfachen mathematischen Modell gearbeitet. Dieses Modell besteht nur aus einer zu lösenden Gleichung und es können so viele analytische Daten wie nötig zum Trainieren erstellt werden. Des weiteren ist die Ähnlichkeit des Modells zu einer meteorologischen Feldgröße, wie etwa der Bodentemperatur, gegeben, sodass gewonnene Erkenntnisse mit dem Lorenzmodell später direkt auf das Klimamodell angewendet werden können.

3.1 Theorie

Das Lorenzmodell ist ein mathematisches Modell welches von dem Mathematiker und Meteorologen Edward Lorenz im Jahr 1996 formuliert wurde (Lorenz 1996). Das Modell beschreibt ein dynamisches System, welches chaotisches Verhalten annehmen kann und wird verallgemeinert durch Gleichung 1 definiert.

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F \qquad i \in \mathbb{N} \tag{1}$$

Die Gleichung beschreibt die zeitliche Veränderung einer einzelnen fiktiven Variable x auf einem eindimensionalen Gitter mit i Gitterpunkten. F ist der Forcing-Term und definiert den von außen kommenden Antrieb des Systems. Die quadratischen Terme der Gleichung beschreiben eine Advektion und der Term $-x_i$ ist eine Dämpfung, damit eine einzelne Störung nicht unendlich anwachsen kann. Ein dynamisches System entsteht erst wenn $i \geq 3$.

3.2 Datengrundlage

Um ein Synoptisches System bestmöglich mit dem Lorenzmodell abzubilden wird ein Forcing von $F = 8$ verwendet, welches ein möglichst hohes Chaosverhalten des Systems erzwingt (Karimi und Paul 2010). Weiterhin werden 40 Gitterpunkte benutzt also i = {0, 1, 2, 3, ..., 39}. Dem Modell werden 1000 Tage Vorlauf zur Initialisierung gegeben, hiernach werden die Daten verwendet (Abbildung 1).

Die Länge der simulierten Zeitreihe, welche im Folgenden als Input-Datensatz dienen wird, beträgt 100 Jahre mit einer Ausgabefrequenz von 6 Stunden.

$$\frac{100 \cdot 365 \cdot 24\,h}{6\,h} = 146.000 \tag{2}$$

Der Input-Datensatz besteht aus 146.000 Zeitschritten und wird in Trainingsdaten, Validierungsdaten und Testdaten aufgeteilt, entsprechend der folgenden Verhältnisse:

- 2/3 Trainingsdaten,
- 1/6 Validierungsdaten,
- 1/6 Testdaten.

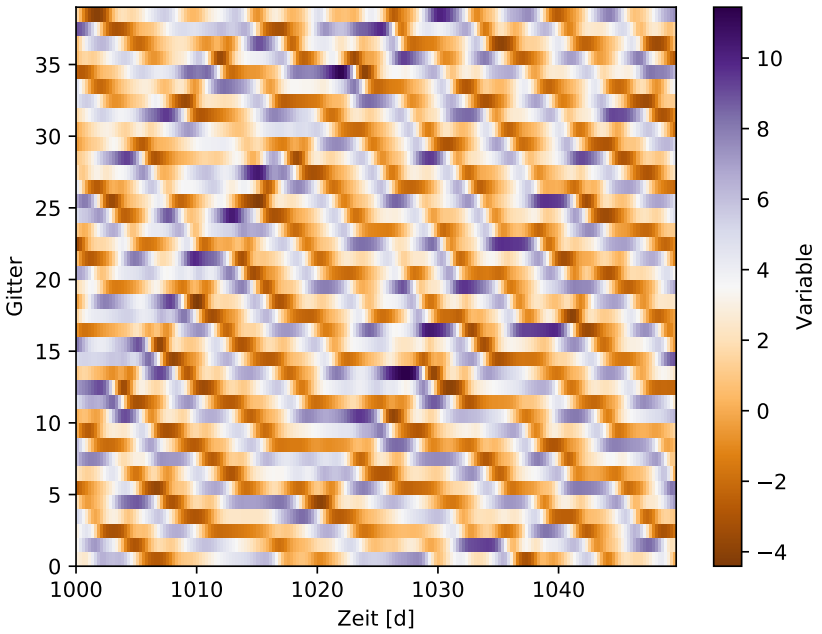

Abbildung 1: Die ersten 50 Tage (200 Zeitschritte) des Input-Datensatzes, erzeugt mit dem Lorenzmodell. Die x-Achse beginnt bei 1000 Tagen, da die ersten 1000 Tage als Initialisierungszeitraum für das Modell verwendet werden.

Für das Training des neuronalen Netzwerkes stehen somit noch 97.000 Zeitschritte zur Verfügung. Die Validierungsdaten werden verwendet, um während des Trainingvorgangs die Lernrate zu überwachen und ein Overfitting zu verhindern. Die Qualität des Modells wird zuletzt an den Testdaten ermittelt.

3.3 Neuronales Netzwerk

Generell gibt es zwei verschiedene Arten von künstlichen neuronalen Netzwerken. Zum einen gibt es Convolutional Neural Networks (CNN) - Faltungsnetzwerke -, zum Anderen gibt es Recurrent Neural Networks (RNN) - Rekurentes neuronales Netz. Eine Unterklasse der RNNs sind Long Short-Term Memory Networks (LSTM) - Langes Kurzzeitgedächtnis.

CNN

Heutzutage sind CNNs am weitesten verbreitet, diese eignen sich besonders um Strukturen in den Eingangsdaten zu ermitteln, weswegen sie häufig in der Bildverarbeitung eingesetzt werden. Das Erkennen der Strukturen funktioniert mittels der mathematischen Faltung, welche die ursprünglichen Eingangsdaten verkleinert. Soll das Modell später einen Datensatz zurückgeben, welcher die gleichen Dimensionen, wie der Input-Datensatz aufweist, muss der Input-Datensatz demnach erst künstlich vergrößert werden.

Der Gleichung 1 kann entnommen werden, dass es sich um ein Problem handelt, welches mit einer eindimensionalen Kugel verglichen werden kann. Aufgrund dieser Struktur werden die

Randbedingungen zyklisch gesetzt. Hierzu wird nach dem Gitterpunkt $i = 39$ der Gitterpunkt $i = 0$ angefügt und vor den Gitterpunkt $i = 0$ wird der Gitterpunkt $i = 39$ gesetzt.

Das CNN selbst besteht aus einem Input-Layer, zwei 2D-Convolutional-Layern und einem Output-Layer (Abbildung 2). Die Convolutional-Layer benutzen die „Relu“ und „Hard Sigmoid“ Aktivierungsfunktionen (Li und Karpathy 2017). Für das Lorenzmodell beträgt die Batch-Size 50 und die Neuronenanzahl 100. Die Input-Daten werden zufällig angeordnet und an das CNN weitergegeben. Dies führt dazu, dass ein Overfitting an die Trainingsdaten minimiert wird.

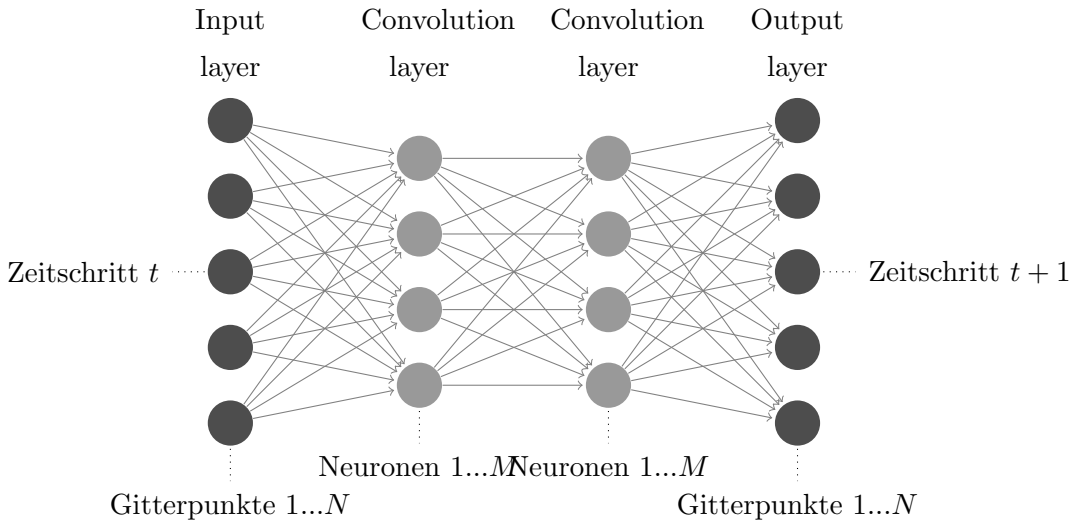

Abbildung 2: CNN mit zwei „Convolution Layer“. Für die Erhaltung der Gittergröße werden zyklische Randbedingungen angenommen (Padding, Reflection).

LSTM

Insbesondere beim prozessieren von sequentiellen Daten zeigen LSTMs ihre Stärke. Anders als CNNs versuchen LSTMs nicht aus einem einzelnen Input die signifikanten von den nicht relevanten Elementen zu trennen, sondern sind auf mehrere sequentielle Input-Daten ausgelegt. Aus den Veränderungen der Input-Daten erkennt das LSTM ein Muster. Da das LSTM eine gewisse Anzahl an sequentiellen Input-Daten benötigt, liefert dieses erst nach dieser Anzahl Vorhersagewerte, während das CNN gleich den nächsten Zeitschritt prognostiziert.

Über die besagten Input-Datenschritte baut sich ein Speicher (Internal State) auf, welchen das neuronale Netz benötigt um eine Vorhersage für einen weiteren Zeitschritt zu machen. Wird die Anzahl der Input-Datenschritte jedoch zu groß, so wächst der Internal State unbegrenzt an was letztendlich dazu führt, dass das LSTM abstürzt. Um dies zu verhindern, sollte entweder eine begrenzte Anzahl an Input-Datenschritten gewählt werden oder der Internal State muss an sinnvollen Punkten zurückgesetzt werden.

Für das Lorenzmodell wird ein Netzwerk mit drei Layern verwendet, einem Input-Layer, einem LSTM-Layer und einem Output-Layer. Das Input-Layer verwendet für das LSTM 10 Zeitschritte und das LSTM-Layer beinhaltet 100 Neuronen (Abbildung 4). Da das LSTM-Netzwerk keine Faltung verwendet, hat das ausgegebene Gitter die gleichen Dimensionen wie das Input-

Gitter. Der Traingsdatengenerator benutzt jeden Zeitschritt aus dem Trainingsdatenset (Kapitel 3.2) in allen möglichen Kombinationen, sodass möglichst viele Trainingsszenarien vorliegen. Für die erste vorhersage werden dafür die Zeitschritte $t = 0$ bis $t = 9$ verwendet um $t = 10$ vorherzusagen, für eine weitere Vorhersage werden $t = 1$ bis $t = 10$ verwendet um $t = 11$ vorherzusagen. Um ein Overfitting zu minimieren, werden ähnlich wie beim CNN die Inputdatensequenzen in zufälliger Reihenfolge dem neuronalen Netzwerk zugeführt, das heißt für das Training werden zunächst die Zeitschritte $t = 1$ bis $t = 10$ verwendet und zum Beispiel zufällig folgend $t = 11$ bis $t = 20$, statt chronologisch $t = 2$ bis $t = 11$. Die einzelnen Inputdatensequenzen sind jedoch bezogen auf die Zeitschritte konsistent.

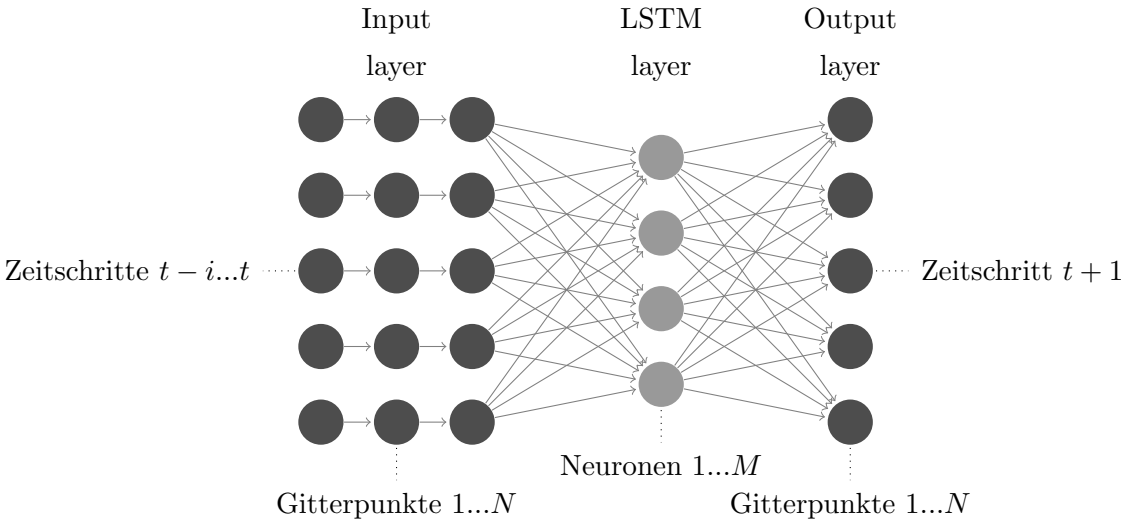

Abbildung 3: LSTM mit einem „Long Short-Term Memory layer“.

Abbildung 4: Neuronales Netzwerk mit einem LSTM Layer. Die Knoten des Inputlayers sind zur Darstellung der Inputzeitschritte mehrfach dargestellt.

3.4 Ergebnisse

Da die beiden getesteten neuronalen Netzwerktypen grundlegend sehr verschieden sind, ist eine Vergleichbarkeit zwischen beiden nur bedingt herzustellen. So wurden für das CNN und das LSTM jeweils 150 Neuronen und eine Batch-Size von 50 verwendet, da die Funktionsweise jedoch so verschieden ist, muss dies nicht unbedingt in einer guten Vergleichbarkeit resultieren. Es ist jedoch nicht das Ziel dieser Arbeit die beiden Typen bestmöglich zu vergleichen, sondern den für die Fragestellung besseren Netzwerktyp festzulegen.

Die Ergebnisse der beschriebenen Konfigurationen der Modelle sind der Abbildung 5 zu entnehmen. Bei nur einer Epoche sind beide Modellergebnisse sehr ähnlich mit RMSEs um die 0,35 und Korrelationen um 0,995. Schon nach 5 Epochen sticht das LSTM deutlich hervor, da der RMSE auf 0,128 sinkt, während das CNN einen RMSE von 0,277 behält. Ab 10 Epochen dominiert die Qualität der Ergebnisse des LSTMs deutlich. Der RMSE sinkt unter 0,1 und die Korrelation ist selbst nach Rundung auf die dritte Nachkommastelle 1,0. Das CNN wird mit mehr Epochen zwar auch genauer, es erreicht aber selbst nach 10 Epochen nur einen RMSE

von 0,261 bei einer Korrelation von 0,997. Für das CNN spricht allerdings die Laufzeit zum Trainieren des neuronalen Netzes. Die Laufzeit ist für das LSTM zwischen 5 und 10 mal länger als für das CNN, was aber auch auf die größere Anzahl an Input-Daten zurückzuführen ist. Wenn das Netzwerk erst einmal trainiert ist, macht sich bei der Prognose eines Zeitschrittes kaum ein Unterschied in der Laufzeit bemerkbar. Für dieses Projekt sind die Laufzeiten zum Trainieren der Netzwerke nicht primär von Interesse.

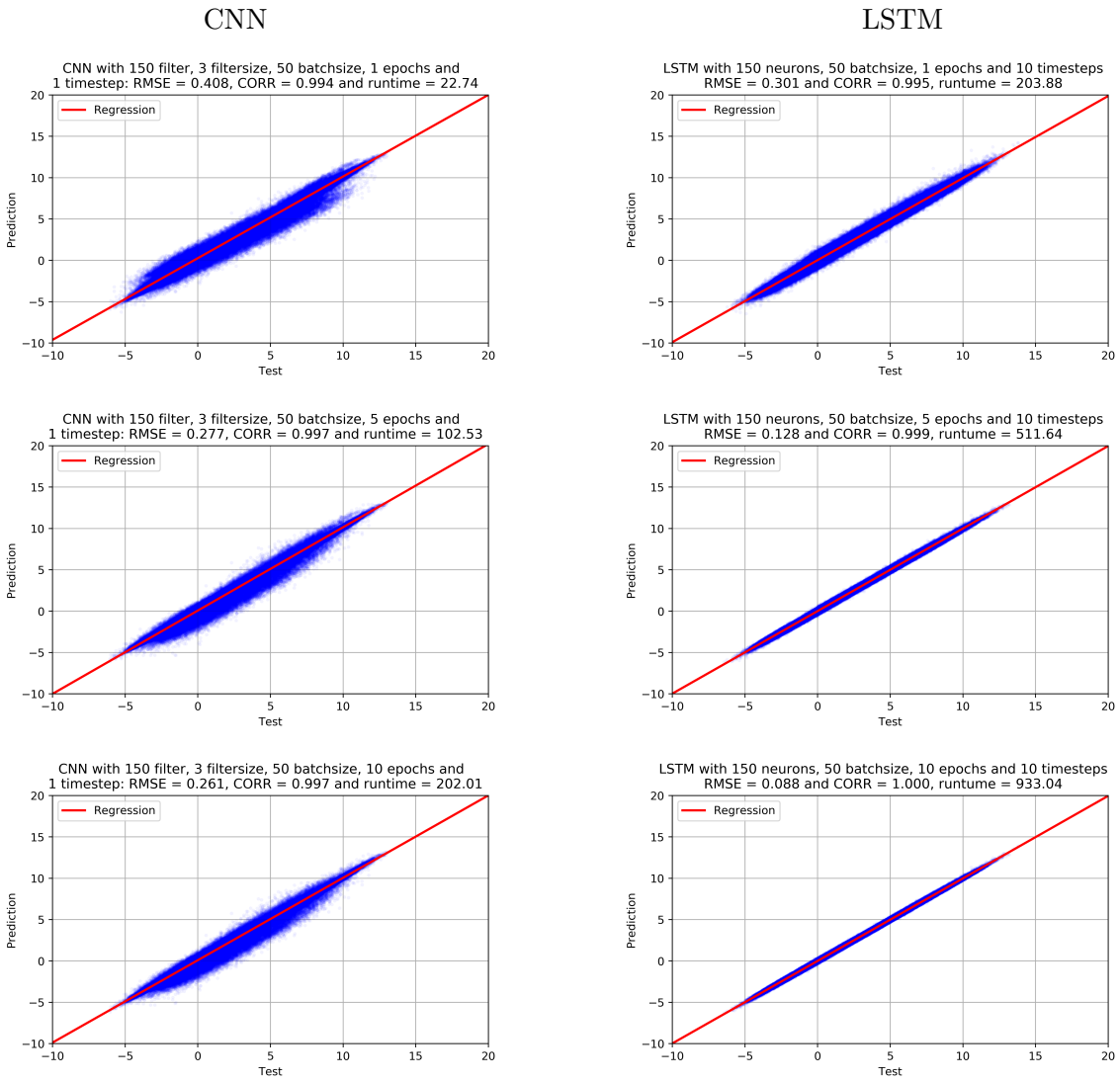

Abbildung 5: Ergebnisse des trainierten CNN (links) und des trainierten LSTM (rechts) für das Lorenzmodell, mit der jeweiligen Regressionsgeraden (rot), für die Epochen 1 (oben), 5 (mitte) und 10 (unten).

Beide Modelle liefern eine zufriedenstellende Genauigkeit. Für dieses Projekt wird das LSTM bevorzugt, da es im Vergleich zum CNN in dieser Konfiguration die besseren Ergebnisse liefert. Im Folgenden wird bei der Anwendungen auf Daten des Klimamodells demnach nur noch das LSTM verwendet.

4 Klimamodell

Die Erkenntnisse aus dem theoretischen Vorversuch konnten für die eigentliche Zielfragestellung verwendet werden. Das geeignete neuronale Netzwerk wurde im vorherigen Abschnitt festgelegt und auf die Variable eines Klimamodells angewendet. Im Folgenden wird das verwendete Klimamodell und das neuronale Netzwerk beschrieben. Zudem werden Optimierungsansätze des neuronalen Netzwerks erläutert und eine kurze Ergebnisübersicht dargestellt.

4.1 Datengrundlage

Als Grundlage der folgenden Untersuchung wird eine Variable eines globalen Klimamodells aus dem *Coupled Model Intercomparison Project phase 5* (CMIP5) des *1pctCO2* Experiments (Plesca et al. 2018; Taylor et al. 2012) verwendet. In diesem Experiment wird die CO_2-Konzentration jährlich um ein 1 % im Vergleich zum vorindustriellen Niveau, bis zur Vervierfachung, erhöht (Taylor et al. 2012). Das Szenario beginnt im Jahr 1850 und endet 2005. Weiterhin besteht das Experiment aus 64 Ensemble-Membern, wobei diese sich lediglich in leicht veränderten Anfangsbedingungen unterscheiden. Die horizontale Auflösung des Modells *Max-Planck-Institute Earth System Model* beträgt bei Betrieb mit geringer Auflösung 96 x 192 horizontale Gitterpunkte, bei einer vertikalen Auflösung von 26 Ebenen (MPI-ESM-LR) (Giorgetta et al. 2013). Die Ergebnisse werden als monatliche Mittelwerten ausgegeben. Als Variable wird die 2 m-Temperatur dieses Experiments verwendet. Diese Größe wurde aufgrund der erwarteten Komplexität, aber auch der Anschaulichkeit ausgewählt. Des weiteren wirkt sich der ansteigende CO_2-Gehalt in der Atmosphäre direkt auf die 2 m-Temperatur aus. Die Erwartung ist, dass bei einem erfolgreichen Training des LSTM an den ersten 97 Jahren (1850 bis 1947) der Temperaturanstieg auch in der Prognose der Testdaten zu erkennen ist, welche die letzten 41 Jahre umfassen (1964 bis 2005).

Das Training des neuronalen Netzwerkes erfolgt mit sechs der Ensemble-Member. Insgesamt besteht jedes Ensemble-Member aus 1872 2 m-Temperatur-Zeitschritten. Die ersten 1165 Zeitschritte jedes Ensemble-Members wurden für das Training des neuronalen Netzwerkes verwendet. Aus dem ersten Ensemble-Member wurden weiterhin 205 Zeitschritte zur Validierung nach jeder Epoche des Trainings benutzt, um das Overfitting an die Trainingsdaten zu minimieren. 502 Zeitschritte des ersten Ensemble-Members wurden zum Schluss als Testdatensatz ausgewählt, anhand welcher das trainierte neuronale Netzwerk unabhängig von den Trainings- und Validierungszeitschritten bewertet werden konnte. Die Aufteilung des Datensatzes ist angelehnt an die statistisch übliche Aufteilung bei der Verwendung von maschinellem Lernen (Abschnitt 3.2), von dieser wurde jedoch abgewichen aus empirischen Gründen bei der Arbeit mit Keras. Das Training des neuronalen Netzwerkes findet an den Trainingszeitschritten aller Ensemble-Member gleichzeitig statt.

4.2 Neuronales Netzwerk

Als neuronales Netzwerk wird gemäß des Abschnitts 3.3 ein LSTM verwendet. Das neuronale Netzwerk besteht analog zur Abbildung 4 aus einem Input-Layer, einem LSTM-Layer und

einem Output-Layer. Neuronale Netzwerke sind äußerst komplex und umfassen eine Vielzahl an Parametern. In diesem Projekt wurden ausgewählte Parameter variiert und empirisch festgelegt: Batch-Size, Neuronen, Input-Zeitschritte, Epochen, Optimizer, Learning-Rate, Aktivierungsfunktion und weitere. Die Batch-Size wurde zwischen 16 und 64 variiert. Die Anzahl der Neuronen wurde bei dem LSTM mit 50 festgelegt. Die Anzahl der Zeitschritte für das Input-Layer variieren zwischen 12 und 48 Zeitschritten. Es liegen Ergebnisse für eine Anzahl von Epochen zwischen 1 und 100 vor. Als Optimizer wurde der weitverbreitete Optimizer Adam verwendet, da andere Optimizer, wie Adadelta, hier zu keinen offensichtlichen Verbesserungen führten. Entsprechend dem Optimizer wurde eine Standard Learning-Rate von 0,001 verwendet. Erste Callbacks, wie ReduceLROnPlateau, der die Learning-Rate anpasst, wenn keine Verbesserung der Ergebnisse mit zunehmender Epochenzahl eintritt, wurden implementiert, aber nicht weiter untersucht. Die Vielzahl an Parametern des verwendeten neuronalen Netzwerks führt zu einigen möglichen Optimierungsansätzen.

4.3 Optimierungsansätze

Es wurden Optimierungen sowohl im Bereich der Trainingslaufzeit des Modells, als auch in Bezug auf die Genauigkeit angewendet. So wurde für die Performance empirisch eine optimale Batch-Size von 128 ermittelt (Tabelle 1), bei einer Anzahl von 50 Neuronen. Die Batch-Size ergibt sich aus der maximalen Größe, bei welcher die Genauigkeit des Trainingsergebnisses noch gewährleistet bleibt (Bengio 2012). Die Nutzung von mehr Neuronen ergab kaum einen Gewinn bezogen auf die Genauigkeit, jedoch stiegen die Laufzeiten. Des weiteren wurden die Daten mittels eines Datengenerators dem Netzwerk zugeführt. Der Generator lädt nur die aktuell benötigten Zeitschritte aus den Input-Daten, was eine hohe Speichereffizienz mit sich bringt.

Für die Genauigkeit wurde zudem getestet, inwiefern sich die Anzahl der Zeitschritte auf diese auswirkt. Die erste Idee war 12 Zeitschritte zu verwenden, aufgrund der monatlichen Schwankungen eines Jahres. Weitere Versuche wurden unter anderem mit 24 und 48 Zeitschritten durchgeführt, jedoch ohne merkliche Steigerung der Genauigkeit (Tabelle 1). Allerdings wurde die Laufzeit um ungefähr den gleichen Faktor vergrößert, wie die Änderung der Zeitschritte, sodass letztendlich 12 Zeitschritte verwendet wurden.

Nach Bengio (2012) wurde eine zufällige Reihenfolge für die Trainingsdaten des LSTM gewählt. Dies soll verhindern, dass eine zu starke Anpassung an die Trainingsdaten selbst erfolgt (Overfitting), sodass eine möglichst hohe Generalisierung des Modells erhalten bleibt. Empirisch konnten wir nach Tests mit und ohne zufälliger Reihenfolge jedoch keinen merkbaren Unterschied festestellen. Für die Experimente wurden dennoch immer Trainingsdaten in zufälliger Reihenfolge benutzt, da es keinen merkbaren Einfluss auf die Laufzeit gab.

4.4 Ergebnisse

Nach der Entwicklung und Optimierung des neuronalen Netzwerks wurde versucht repräsentative und vergleichbare Ergebnisse zu produzieren. Die jeweiligen Konfigurationen des LSTMs mit den Ergebnissen in Form des RMSEs, der Korrelation und der Laufzeit sind der Tabelle 1 zu

entnehmen. Die statistischen Größen, wie der RMSE und die Korrelation, wurden zwischen dem Klimamodell und dem neuronalen Netzwerk für jeden Zeitschritt gebildet und gemittelt. Die Ergebnisse der Konfiguration des LSTMs mit einer Batch-Size von 128, der Epochenanzahl von 20 und 12 Zeitschritten sind im Vergleich zu Ergebnissen anderer Konfigurationen optimal (Tabelle 1) und somit wird diese beispielhaft näher betrachtet.

Tabelle 1: Konfigurationen des LSTMs mit Ergebnissen. Die Neuronenanzahl beträgt 50.

Batch-Size	Epochen	Zeitschritte	RMSE	CORR	Laufzeit (s)
16	1	12	2.53258	0.99262	490
16	20	12	2.49101	0.99277	10010
16	20	12	2.51039	0.99278	9513
16	20	12	2.46181	0.99294	9246
16	25	48	2.60728	0.99217	29533
16	100	12	2.48967	0.99271	49574
64	20	12	2.45309	0.99293	2346
128	**20**	**12**	**2.44797**	**0.99302**	**1464**
128	20	12	2.48885	0.99292	1606

Die optimale Konfiguration des LSTMs hatte im Test eine Laufzeit von 24,4 Minuten, die primär durch das Training geprägt war. Werden die Differenzen zwischen den prognostizierten Temperaturen des LSTMs und der Temperatur des Klimamodells gebildet und über die Trainingsdaten gemittelt, ergibt sich ein BIAS von 0,22 K und eine Standardabweichung von 1,12 K (Abbildung 6). Der Mittelungszeitraum beträgt 40,75 Jahre. Es fällt auf, dass insbesondere im Bereich des mittleren Modellgebiets bezogen auf die Längengrade die Differenzen nahezu Null betragen. Die Differenzen am Karten- beziehungsweise Modellrandbereich bezüglich der Längengrade sind deutlich größer. Dieses Phänomen tritt bei allen Konfigurationen auf und lässt sich nicht eindeutig erklären. Eine Vermutung ist, dass zyklische Randbedingungen vernachlässigt werden. Die zweidimensionalen Temperaturfelder werden für das LSTM auf ein eindimensionales Feld reduziert, somit sind die Bereiche mit den großen Differenzen in Abbildung 6 Randbereiche. Grundsätzlich ist diese Vermutung aber nicht schlüssig, da die Gitterpunkte des Modells miteinander gekoppelt sein sollten. Nichtsdestotrotz sind die Ergebnisse als gut anzusehen, da lediglich aus den letzten 12 Monaten jeweils der Monat 13 prognostiziert wurde. Auf die Betrachtung einzelner Zeitschritte wird an dieser Stelle verzichtet, da die monatlichen Ergebnisse bei Klimabetrachtungen keine besondere Relevanz haben, sondern Mittelwerte über größere Zeiträume betrachtet werden. Werden einzelne Zeitschritte betrachtet, fallen deutliche Schwankungen in den Differenzen der Temperaturen von LSTM und CMIP5 Daten auf. Beispielsweise variiert der global gemittelte BIAS der Differenzen zwischen Zeitschritt 14 mit 0,75 K und Zeitschritt 15 mit -0,12 K.

Das globale Mittel der einzelnen Zeitschrittvorhersagen ist zusammen mit dem globalen Mittel des MPI-ESM-LR in Abbildung 7 aufgetragen. Eindeutig wird der ansteigende Temperaturtrend des Klimamodells auch vom LSTM wiedergegeben. Des weiteren ist zu erkennen, dass

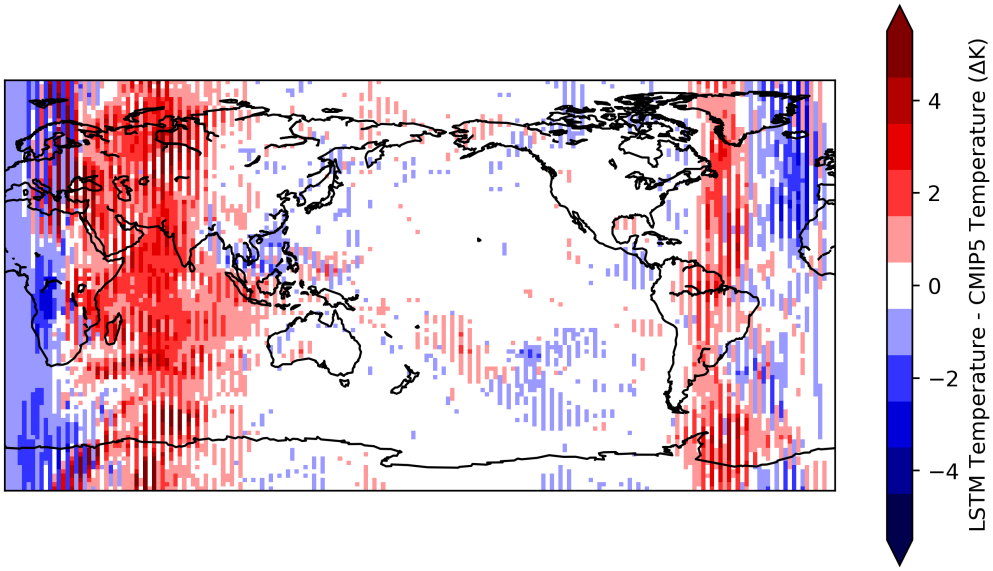

Abbildung 6: Differenzen der Temperaturen zwischen LSTM und CMIP5 Daten gemittelt über 40,75 Jahre.

das neuronale Netzwerk nicht nur eine Persistenzvorhersage macht, sondern Zusammenhänge interpretiert. Dies ist nicht trivial, da im Bereich der Klimaprognosen ein Persistenzmodell generell gut abschneidet. Das LSTM versucht den mittleren quadratischen Fehler beim Training möglichst zu minimieren und ein Persistenzmodell wäre demnach eine mögliche Lösung dieses Problems, jedoch für ein Prognosemodell nicht wünschenswert. Weiterhin lässt sich aus der Abbildung 7 ein positiver BIAS des LSTMs entnehmen. Eine mögliche Erklärung der Überschätzung der 2 m-Temperatur könnte der generellen globalen Erwärmung in den Trainingsdaten geschuldet sein, was auch die deutlich überschätzten Minima das MPI-ESM-LR begründen würde. Aufgrund der Komplexität eines neuronalen Netzwerkes ist dies nicht mit Sicherheit zu beweisen und bestenfalls eine gute Annahme.

Werden alle Prognosewerte des LSTM für jeden Gitterpunkt einzeln gegen die Modellwerte des MPI-ESM-LR aufgetragen, lässt sich der positive Offset nur noch schwer erkennen (Abbildung 8). Allerdings zeigt sich, dass für die Vorhersage der besonders hohen und niedrigen Temperaturen das LSTM gut abschneidet, sodass der Bereich zwischen 270 K und 300 K besonders gut vorhergesagt wird. Eine mögliche Erklärung für die gute Korrelation und den kleine BIAS in diesem Bereich ist, dass die monatlichen Mittelwerte dieser Temperaturen hauptsächlich aus den Tropischen Regionen kommen, wo die Variabilität der 2 m-Temperatur generell nicht sehr groß ist. Die niedrigen Temperaturen (< 240 K) werden jedoch etwas überschätzt, was an der Steigung und dem y-Achsenabschnitt der Regressionsgerade zu erkenne ist. Die logarithmische Farbskala ist zu beachten, weswegen die Auffächerung der Werte zwischen 240 K und 270 K mit einem viel geringeren Gewicht in die Ergebnisse eingeht, als der Bereich zwischen 270 K und 300 K.

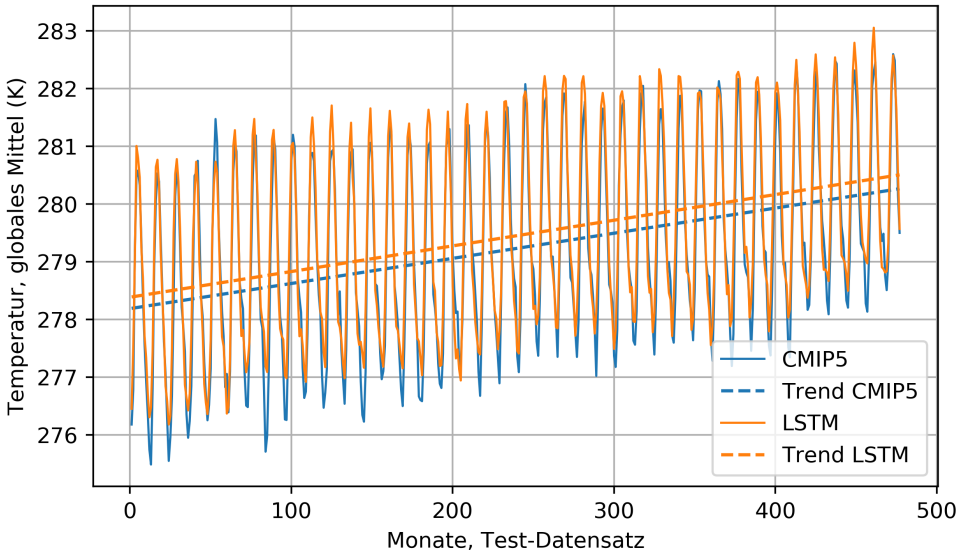

Abbildung 7: Globales Mittel der 2 m-Temperatur des CMIP5 und der Vorhersage des LSTM mit linearem Trend.

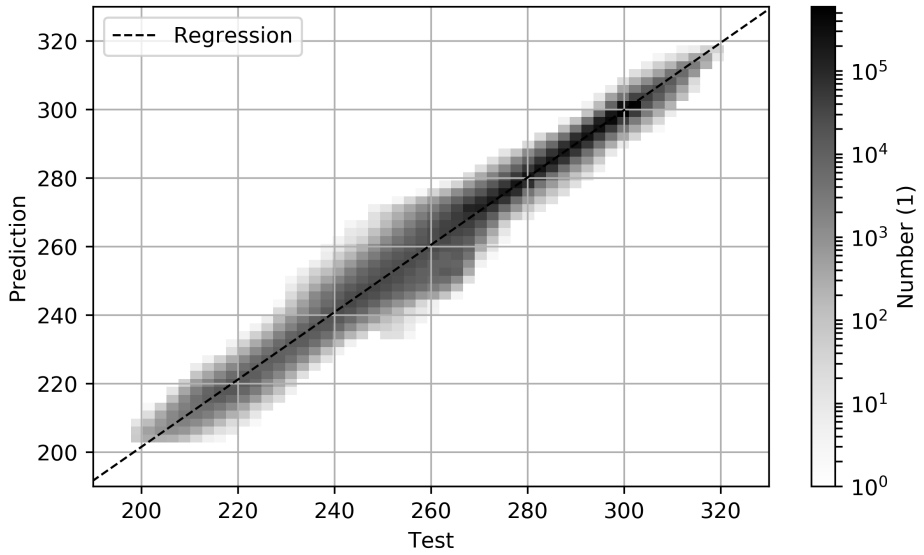

Abbildung 8: 2D-Histogramm der vorhergesagten (Prediction) und wahren (Test) 2 m-Temperatur.

5 Fazit

Nachdem das einfache neuronale Netzwerk aus Abbildung 4 ausgiebigen Tests in verschiedenen Konfigurationen unterzogen wurde, lässt sich sagen, dass die Erwartungen an neuronale Netzwerke zur Prognose von einzelnen Variablen in Klimamodellen weit übertroffen wurden. Sowohl für das analytisch lösbare Lorenzmodell, als auch für das komplexe MPI-ESM-LR überzeugt die Qualität der Ergebnisse, zeigt aber auf, dass weitere Untersuchungen notwendig sind.

Die Abweichungen im globalen Mittel über die 40,75 Testjahre (Tabelle 1) mit Spitzenwerten im RMSE von 2,45 K zeigen, dass das neuronale Netzwerk die Variabilität im Klimamodell gut wiedergibt. Natürlich kann das Klimamodell nicht ersetzt werden, da es sowohl zum Training des neuronalen Netzwerkes, als auch für das physikalische Verständnis benötigt wird. Vergleicht man jedoch die Laufzeit, welche das Klimamodell zur Prognose eines Zeitschrittes benötigt, mit der Laufzeit, die das trainierte LSTM für den gleichen Zeitschritt benötigt, so bietet sich eindeutig das LSTM an. Dies zeigt eine mögliche Anwendung des neuronalen Netzes auf. Zur Abschätzung eines Trends könnte das LSTM eine mögliche Alternative sein.

Auf der anderen Seite dürfen die Ergebnisse einzelner Zeitschritte nicht als wissenschaftliche Grundlage zur Untersuchung von Klimaphänomenen benutzt werden. Die Abweichung einzelner Gitterzellen ist dafür mit vereinzelt über 20 K zu hoch. Dies könnte bei einer Anwendung mit Wettermodellen, anstatt mit Klimamodellen anders sein, da der Zusammenhang zwischen chronologisch aufeinander folgenden Zeitschritten im Wettermodell viel eindeutiger ist, als im Klimamodell. Des weiteren ist die Auflösung des Gitters in Wettermodellen meist deutlich höher, weswegen auch die Zusammenhänge benachbarter Gitterzellen größer ist, was die Erkennung von Strukturen durch das neuronalen Netzwerk verbessern würde.

Es ist außerdem anzumerken, dass für das neuronale Netzwerk starkes Verbesserungspotential besteht. Das im Rahmen dieser Arbeit entwickelte Netzwerk ist darauf ausgelegt möglichst einfach und verständlich zu sein, daher würde ein komplexeres Netzwerk auch die komplexe Problemstellung weitaus genauer approximieren. Im Rahmen dieses Projekts konnte jedoch das Ziel einer hinreichend genauen Approximation erfüllt werden.

Literatur

Bengio, Yoshua (2012). „Practical recommendations for gradient-based training of deep architectures“. In: *CoRR* abs/1206.5533. arXiv: 1206.5533. URL: http://arxiv.org/abs/1206.5533.

Bezenac, Emmanuel de, Arthur Pajot und Patrick Gallinari (2017). „Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge“. In: *arXiv preprint arXiv:1711.07970*.

Gentine, P., M. Pritchard, S. Rasp, G. Reinaudi und G. Yacalis (2018). „Could Machine Learning Break the Convection Parameterization Deadlock?“ In: *Geophysical Research Letters* 45.11, S. 5742–5751. DOI: 10.1029/2018GL078202. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2018GL078202. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018GL078202.

Giorgetta, M. A., J. Jungclaus, C. H. Reick, S. Legutke, J. Bader, M. Böttinger, V. Brovkin, T. Crueger, M. Esch, K. Fieg et al. (2013). „Climate and carbon cycle changes from 1850 to 2100 in MPI-ESM simulations for the Coupled Model Intercomparison Project phase 5“. In: *Journal of Advances in Modeling Earth Systems* 5.3, S. 572–597.

Karimi, A. und M. R. Paul (2010). „Extensive chaos in the Lorenz-96 model“. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 20.4. DOI: 10.1063/1.3496397. eprint: https://doi.org/10.1063/1.3496397. URL: https://doi.org/10.1063/1.3496397.

Li, Fei-Fei und Andrej Karpathy (2017). *Convolutional Neural Networks for Visual Recognition*. Website. http://cs231n.stanford.edu/2017/; abgerufen am 6. März 2018.

Lorenz, Edward N (1996). „Predictability: A problem partly solved“. In: *Proc. Seminar on predictability*. Bd. 1. 1.

Moghim, Sanaz und Rafael L Bras (2017). „Bias correction of climate modeled temperature and precipitation using artificial neural networks“. In: *Journal of Hydrometeorology* 18.7, S. 1867–1884.

Plesca, E., V. Grützun und S. A. Buehler (2018). „How robust is the weakening of the Pacific Walker circulation in CMIP5 idealized transient climate simulations?“ In: *Journal of Climate* 31.1, S. 81–97.

Rew, R. und G. Davis (1990). „NetCDF: an interface for scientific data access“. In: *IEEE Computer Graphics and Applications* 10.4, S. 76–82. ISSN: 0272-1716. DOI: 10.1109/38.56302.

Taylor, Karl E, Ronald J Stouffer und Gerald A Meehl (2012). „An overview of CMIP5 and the experiment design“. In: *Bulletin of the American Meteorological Society* 93.4, S. 485–498.

Xingjian, SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong und Wang-chun Woo (2015). „Convolutional LSTM network: A machine learning approach for precipitation nowcasting“. In: *Advances in neural information processing systems*, S. 802–810.