Freie Universität Berlin

Bachelorarbeit am Institut für Informatik der Freien Universität Berlin

Evaluierung eines vortrainierten Deep Neural Network für Precipication Nowcasting auf deutschen Wetterdaten

Tarek Beutler
Matrikelnummer: 5071152
tareb98@zedat.fu-berlin.de

Betreuer/in: Dr. Annette Müller
Eingereicht bei: Prof. Dr. Daniel Göhring
Zweitgutachter/in: Prof. Dr. Nikki Vercauteren

Berlin, 23. Juni 2021

Zusammenfassung

Precipitation nowcasting beschreibt eine Niederschlagsvorhersage innerhalb eines kurzen Zeitrahmens. In dieser Arbeit wird die Leistung eines auf Daten aus Hong Kong trainierten neuronales Netzes für *precipitation nowcasting* auf deutschen Wetterdaten evaluiert. Dabei werden unterschiedliche Einflüsse auf die Leistung untersucht und gezeigt, dass die zeitliche Auflösung der Eingabedaten einen deutlichen Unterschied in der Vorhersagegenauigkeit macht. Anschließend werden Probleme und mögliche Verbesserungen diskutiert.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

23. Juni 2021

Tarek Beutler

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

Precipitation nowcasting beschreibt die detaillierte Vorhersage der Niederschlagsintensität in einem relativ kurzem zeitlichen Rahmen in einer lokalen Region, meistens zwischen 0 und 6 Stunden [1]. Dies macht es essentiell für viele reale Probleme, zum Beispiel Vorhersage der Wetterumstände für den Verkehr. Aktuelle Ansätze für *precipitation nowcasting* beschränken sich primär auf *Numerical Weather Prediction* (NWP) Modelle und solchen, die auf der Extrapolation von Radarechos basieren [2].
Dieses Problem kann auch als raum-zeitliches Vorhersage-Problem begriffen werden, also das Problem Vorhersagen zu treffen, welche auf Daten basieren, die sowohl räumliche als auch zeitliche Attribute besitzen [3]. Dies und die aktuellen Fortschritte im Bereich des *Deep Learning* haben dazu geführt, dass auch vermehrt maschinelles Lernen eingesetzt wird um *precipitation nowcasting* zu betreiben. So haben 2015 Shi et al. ein sogenanntes *Convolutional LSTM Network* (ConvLSTM) formuliert, welches spatiotemporale Korrelationen wesentlich besser erfasst und Niederschlag genauer vorhersagt, als Radarecho-Extrapolation Methoden auf dem aktuellen Stand der Technik [4]. 2017 erweiterten Shi et al. diesen Fortschritt nochmal mit einem verbesserten Modell, dem *Trajectory GRU* (TrajGRU) sowie verbesserten Fehlermethoden und einem standardisiertem Datensatz, um Modelle zu testen [5].

Beide Arbeiten basieren auf Wetterdaten aus und um Hong Kong. Das TrajGRU wurde dabei mit Eingabedaten von einer zeitlichen Auflösung von 6 Minuten trainiert. In dieser Arbeit soll die Wirkung der zeitlichen Auflösung der Eingabedaten auf die Leistung des vortrainierten TrajGRUs untersucht werden. Dabei werden mit Daten aus Deutschland und Hong Kong räumlich verschiedene Daten betrachtet. Anschließend sollen Methoden und Ansätze diskutiert werden, wie die Leistung des Modells auf fremden Daten verbessert werden kann.

1.2 Verwandte Arbeiten

Die in dieser Arbeit betrachteten Ansätze beschränken sich auf solche, welche rekurrente neurale Netze, also etwa *Long Short-Term Memory Networks* (LSTMs), als Basis benutzen.

Ein weiterer Ansatz ist *precipitation nowcasting* basierend auf U-Net [6], welches im Gegensatz zum ConvLSTM nur ein einfaches *Convolutional Neural Network* (CNN) ohne Gedächtnis ist. So benutzen Agrawal et al. in [7] U-Net um Vorhersagen auf Daten eines Doppler-Radar für drei verschiedene Kategorien von Niederschlag, eine Stunde in die Zukunft zu treffen. Ähnlich gehen Trebing et. al in [8] vor, welche U-Net als Basis nehmen und *Convolutional Block Attention Modules* [9] in das Modell einführen. Dies führt dazu, dass sich das Modell besser auf wichtige Eigenschaften, wie die räumliche Region der Daten, konzentrieren kann. Eines der neusten Modelle in diesem Bereich ist RainNet [10], welches lose auf U-Net und SegNet [11] basiert und ebenfalls auf deutschen Wetterdaten getestet wurde.

Im Gesamten kann gesagt werden, dass die Erforschung von auf *Deep Learning* basierten Methoden für *precipitation nowcasting* gerade erst begonnen hat und viele Architekturen noch ununtersucht sind.

2 Grundlagen

2.1 Convolutional Long Short-Term Memory Network

2.1.1 Convolutional Neural Networks

Ein CNN ist ein neuronales Netz, welches Faltungsoperatoren anstatt von normalen Matrixmultiplikationen in mindestens einer Ebene benutzt [12]. Eine Faltung ist eine Operation, welche zwei Funktionen f und g „faltet“ und so eine dritte Funktion $f * g$ produziert, wobei $*$ der Faltungsoperator ist [13]. Diese kann so ausgedrückt werden:

$$(f * g)(t) = \int f(x)g(t-x)dx$$

Im maschinellen Lernen bezeichnet $f(t)$ normalerweise die Eingabe und $g(t)$ den Kernel. Die diskrete Faltung wird wie folgt definiert:

$$(f * g)(t) = \sum_{x=-\infty}^{\infty} f(x)g(t-x)$$

CNNs werden oft für Daten benutzt, welche in einem Gitterformat vorliegen, wie zum Beispiel Bilder. Deswegen muss die Faltungsoperation auf eine zweidimensionale Eingabe und Kernel erweitert werden:

$$(f * g)(i,j) = \sum_{m}\sum_{n} f(m,n)g(i-m, j-n)$$

Ein CNN besteht aus einer oder mehreren Faltungsebenen. Eine Faltungsebene nimmt dabei die vorherige Eingabe und wendet darauf schrittweise die diskrete Faltung mit einem vorgegebenen Kernel an um die Ausgabe dann an die nächste Ebene weiterzugeben. Eine Ebene hat dabei dieselben Gewichte für jeden Ort der Faltung, was zu einer Ortsunabhängigkeit der Faltungsebenen führt.
Es ist üblich, dass jede Ausgabe der linearen Faltungsoperation in eine nicht-lineare Aktivierungsfunktion gegeben wird und anschließend anhand einer *pooling* Funktion zusammengeführt wird. Eine *pooling* Funktion ersetzt die Ausgabe des neuronalen Netz an einer bestimmten Stelle durch eine statistische Zusammenfassung der umliegenden Ausgabestellen [12]. *Max-pooling* zum Beispiel gibt die größte Stelle in einer quadratischen Umgebung zurück [14].

2.1.2 Long Short-Term Memory Network

Das LSTM ist ein *Recurrent Neural Network* (RNN) [16], welches seit 2016 immense Erfolge verzeichnet und den aktuellen Standard für sequentielle Probleme darstellt. Im Gegensatz zu den klassischeren *Feed-Forward Neural Networks* ermöglichen RNNs anhand ihrer inneren Zustände, die als eine Art Gedächtnis wirken, variable sequentielle Daten zu verarbeiten [17].

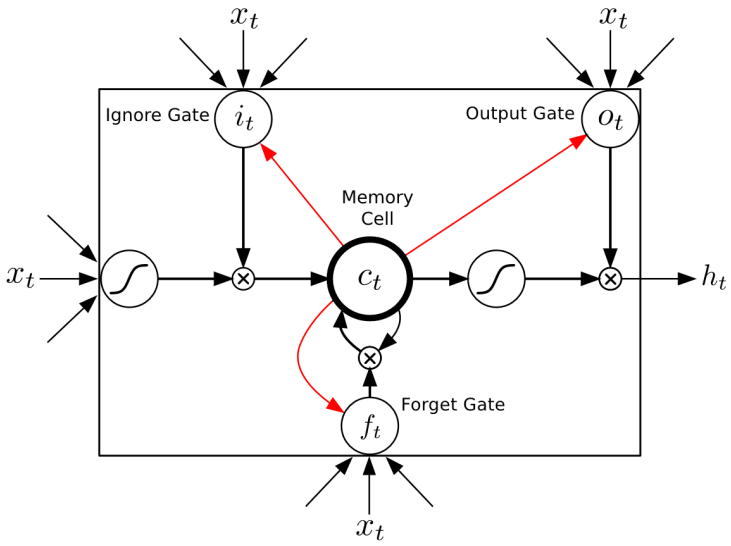


Abbildung 1: LSTM Architektur mit *peephole connections* (rot eingefärbt) [15]. Die kleinen Kreise mit einem $\times$ stellen das Hadamard-Produkt dar, die großen Kreise mit einer Kurve eine Aktivierungsfunktion.

Das LSTM ist eine spezifischere Variante eines RNNs. Es erweitert die klassische Struktur um mehrere Speicherzellen. Diese sind alleinstehende neuronale Netze, welche benutzt werden, um Informationen länger zu behalten (*memory cell*), diese anschließend besser zu selektieren (*selection gate*) und gegebenfalls auch wieder zu vergessen (*ignore gate*). Mathematisch wird dieses wie folgt definiert, wobei i das *ignore gate*, f das *forget gate*, c die *memory cell*, o das *output gate* und h der Ausgabezustand ist. Der Index t ist der aktuelle Zeitschritt, x die Eingabe, W die Gewichte und $\circ$ das Hadamard-Produkt [16]:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$h_t = o_t \circ tanh(c_t)$$

Das ConvLSTM baut auf einer Variante des LSTM auf, dem sogenannten *Peephole LSTM*. Dieses verbessert das herkömmliche LSTM und macht es sensitiver für kleine Unterschiede [18]. Dabei werden Verbindungen zwischen der *memory cell* und den restlichen Gates gezogen (sogennante *„peephole connections“*), was sich mathematisch wie folgt darstellen lässt [15]:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o)$$
$$h_t = o_t \circ tanh(c_t)$$

In Abbildung 1 ist die Architektur eines LSTM zu sehen, mit den hier hinzugefügten Verbindungen als Vergleich rot eingefärbt.

2.1.3 ConvLSTM

Das traditionelle LSTM ist zwar sehr gut für sequentielle Probleme, kodiert aber räumliche Informationen nicht besonders gut. In [4] wird deswegen das LSTM erweitert. Es werden alle Eingaben und Zellen um zwei Dimensionen erweitert, so dass die erste Dimension wie gehabt die zeitliche Abfolge darstellt und die beiden letzten Dimensionen die räumlichen Informationen. Anschließend wird die mathematische Definition des *Peephole LSTM* aus Sektion 2.1.2 erweitert, indem alle Matrixmultiplikationen mit Faltungsoperatoren ersetzt werden. Dies bildet das sogenannte ConvLSTM und ermöglicht, dass zukünftige Zustände einer Zelle anhand der Eingaben und vergangenen Zustände der lokalen Nachbarn bestimmt werden [4]:

$$i_t = \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + W_{ci} \circ c_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + W_{cf} \circ c_{t-1} + b_f)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ tanh(W_{xc} * x_t + W_{hc*}h_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + W_{co} \circ c_t + b_o)$$
$$h_t = o_t \circ tanh(c_t)$$

2.2 Trajectory Gated Recurrent Unit

2.2.1 Gated Recurrent Unit

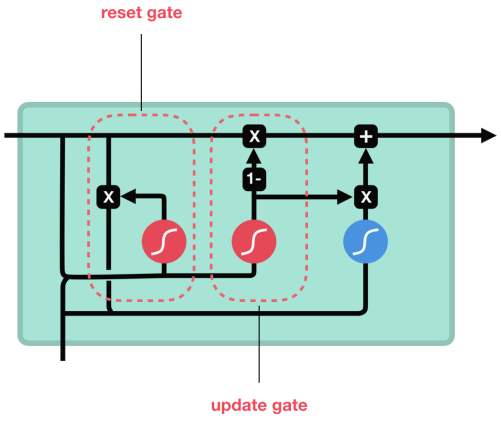


Abbildung 2: GRU Architektur [19]. Die kleinen Rechtecke mit einem $\times$ stellen das Hadamardprodukt dar, mit einem $+$ die punktweise Addition und mit einem $1-$ die Multiplikation mit -1. Die großen Kreise mit einer Kurve stellen eine Aktivierungsfunktion da.

Das TrajGRU basiert auf einer vereinfachten Variante des LSTMs, dem *Gated Recurrent Unit* (GRU). Dies ist ein RNN, welches dem LSTM sehr ähnlich ist, jedoch keine *selection cell* und somit eine Zelle weniger besitzt [20]. Stattdessen hat das GRU ein *reset gate*, was bestimmt wie die alten Informationen mit den neuen kombiniert werden, und ein *update gate*, welches bestimmt welche Informationen behalten werden sollen. Mathematisch wird dies ähnlich wie das LSTM definiert, wobei z_t das *update gate*, r_t das *reset gate*, H_t die gespeicherten und H'_t die neuen Informationen darstellen. f ist die gewählte Aktivierungsfunktion, $\circ$ das Hadamard-Produkt, x die Eingabe und t der aktuelle Zeitschritt.:

$$z_t = \sigma(W_{xz}x_t + W_{hz}H_{t-1})$$

$$r_t = \sigma(W_{xr}x_t + W_{hr}H_{t-1})$$

$$H'_t = f(W_{xh}x_t + R_t \circ (W_{hh}H_{t-1}))$$

$$H_t = (1 - z_t) \circ H'_t + z_t \circ H_{t-1}$$

Analog zum ConvLSTM wurde das ConvGRU eingeführt, in dessen Formeln wieder die Matrixmultiplikationen durch Faltungsoperationen ersetzt wurden [5]:

$$z_t = \sigma(W_{xz} * x_t + W_{hz} * H_{t-1})$$

$$r_t = \sigma(W_{xr} * x_t + W_{hr} * H_{t-1})$$

$$H'_t = f(W_{xh} * x_t + R_t \circ (W_{hh} * H_{t-1}))$$

$$H_t = (1 - z_t) \circ H'_t + z_t \circ H_{t-1}$$

Das GRU lässt sich durch weniger Parameter schneller trainieren als das LSTM und kann auf kleineren Datensätzen ähnliche oder bessere Ergebnisse als LSTMs erzielen [21]. Die Architektur ist in Abbildung 2 dargestellt.

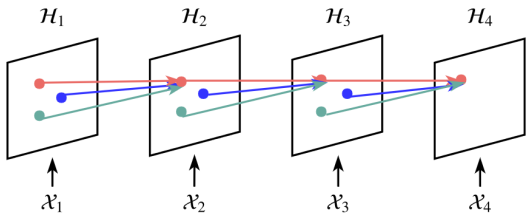
2.2.2 TrajGRU

Eins der größeren Probleme der ConvRNNs ist die Ortsunabhängigkeit der verwendeten CNNs. Seien $N^h_{i,j}$ die geordneten Umgebungssätze, wobei (i, j) der betrachtete Ort ist. Diese Sätze werden durch die Hyperparameter der Faltungen definiert, also die Kernelgröße, Dilation und Padding [22]. Da diese Hyperparameter statisch sind, ist der Umgebungssatz $N^h_{i,j}$ für alle (i, j) gleich. Allerdings haben viele Bewegungsmuster unterschiedliche Umgebungssätze für verschiedene betrachtete Orte, das heißt, diese sind ortsabhängig [5] und die ortsunabhängigen ConvRNNs haben Probleme, diese Muster korrekt zu erkennen.

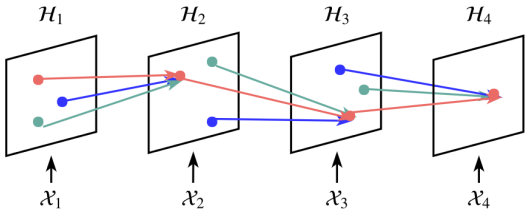
Shi et al. führen in [5] das TrajGRU ein, um dieses Problem zu beheben. Dieses neuronale Netz basiert auf dem ConvGRU und benutzt die aktuellen und vorherigen Informationen um einen lokalen Umgebungssatz für jeden Ort $\sum_{l=1}^{L}(p_{l,i,j}, q_{l,i,j})$ in jedem Zeitschritt zu generieren, wobei L die gewählte maximale Anzahl an lokalen Verbindungen ist. Es werden optische Flüsse[1] benutzt um die Indizes der Orte darzustellen:

$$U_t, V_t = \gamma(x_t, H_{t-1})$$

[1] Ein Vektorfeld, welches ein Bewegungsmuster darstellt [23].



(a) Statische Verbindungsstruktur eines ConvRNN.



(b) Dynamische Verbindungsstruktur des TrajGRU.

Abbildung 3: Vergleich der verschiedenen Verbinungsstrukturen [5].

$$z_t = \sigma(W_{xz} * x_t + \sum_{l=1}^{L} W_{hz}^l * \text{warp}(H_{t-1}, U_{tl}, V_{tl}))$$

$$r_t = \sigma(W_{xr} * x_t + \sum_{l=1}^{L} W_{hr}^l * \text{warp}(H_{t-1}, U_{tl}, V_{tl}))$$

$$H_t' = f(W_{xh} * x_t + R_t \circ (\sum_{l=1}^{L} W_{hh}^l * \text{warp}(H_{t-1}, U_{tl}, V_{tl})))$$

$$H_t = (1 - z_t) \circ H_t' + z_t \circ H_{t-1}$$

γ ist hier ein eigenes kleines neuronales Netz, welches die in U_t und V_t gespeicherten optischen Flüsse generiert. Die warp Funktion selektiert die jeweiligen Orte aus H_{t-1} welche von U_t und V_t angezeigt werden anhand eines *bilinear sampling kernel* [24]. Der Vorteil ist hier, dass die Verbindungsstruktur des Netzes durch das Subnetz γ gelernt werden kann [5]. Ein Vergleich der Verbindungsstrukturen eines herkömmlichen ConvRNNs und eines TrajGRUs ist in Abbildung 3 zu sehen.

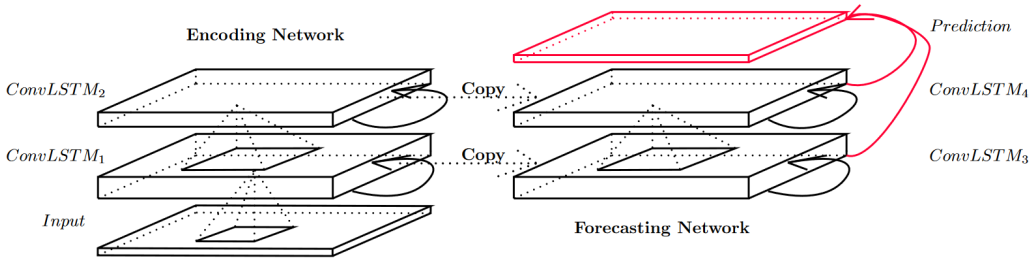


Abbildung 4: *Autoencoder* Struktur des ConvLSTM [4].

Ein komplettes TrajGRU Modell in der Praxis kann in seiner oberflächlichen Form als *Autoencoder* beschrieben werden, also ein neurales Netz, welches die eingegebene Sequenz zuerst zu einer fixen Größe enkodiert, um es anschließend wieder zu dekodieren und als Vorhersage auszugeben [25]. Dafür werden für Enkodierer und Dekodierer mehrere TrajGRU Ebenen „aufeinandergestapelt“. In Abbildung 4 ist die *Autoencoder* Struktur des ConvLSTM zu sehen, welche dem des TrajGRU stark ähnelt.

2.3 Transfer Learning

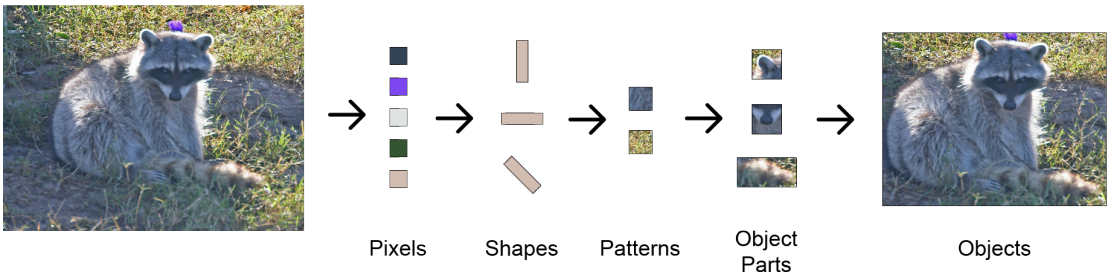


Abbildung 5: Feature-Learning Visualisierung anhand von ImageNet [26].

Viele *Deep Neural Networks* (DNN) funktionieren wie folgt: Die ersten Ebenen lernen generelle Eigenschaften der Daten und je tiefer die Ebenen gehen, desto spezieller werden die gelernten Eigenschaften in Bezug auf den Datensatz [27]. Zum Beispiel bei einem DNN, welches auf ImageNet [26] trainiert wird, lernen die ersten Ebenen soetwas wie Farbflächen und grobe Formen, während die tieferen Ebenen beispielsweise Kanten und kleine Farbunterschiede lernen. Dieser Prozess ist in Abbildung 5 bildlich dargestellt. Diese Eigenschaft ermöglicht uns, die Gewichte eines DNNs, welches auf einen bestimmten Datensatz trainiert wurde, auch für andere Datensätze zu benutzen, ohne das gesamte Netzwerk neu trainieren zu müssen. Dieser Prozess nennt sich *Transfer Learning*. Ein weiterer nützlicher Anwendungsfall ist, wenn der genutzte Datensatz sehr klein ist. Die generellen Eigenschaften können dann auf einem größeren, ähnlichen Datensatz gelernt werden und die spezielleren Eigenschaften dann auf dem kleinen Datensatz *finetuned* werden.

3 Hauptteil

3.1 Datengrundlage

3.1.1 HKO-7

Der HKO-7 Datensatz ist ein extra für *precipitation nowcasting* erstellter Benchmark-Datensatz. Er beinhaltet Radarecho Bilder eines Doppler-Radar, gesammelt von 2009 bis 2015 von einem 512km x 512km Gebiet über Hong Kong. Die Daten sind im 6-Minuten Takt aufgenommen und haben eine Auflösung von 480px x 480px. Zusätzlich liegen generierte Masken vor, um Störungen in den Daten auszugleichen. Für das Testen des TrajGRU Models wurden basierend auf den niederschlagreichsten Tagen,

812 Tage für das Training, 131 Tage für das Testen und 50 Tage für die Validierung ausgewählt.

3.1.2 KONRAD

Die *Konvektive Entwicklung in Radarprodukten* (KONRAD) ist ein „Verfahren zur automatischen Erkennung, Verfolgung und Vorhersage von Gewitterzellen auf der Basis von Wetterradardaten“ [28]. Es wird vom Deutschen Wetterdienst (DWD) für Unwetterwarnungen und den Katastrophenschutz genutzt.
KONRAD hat eine ungefähre Auflösung von 1 km x 1 km und wird in einem Intervall von 5 Minuten aus der Radarreflektivität der DWD Niederschlagsradare berechnet. Die Radarreflektivität berechnet sich aus der Rückstreuung von elektromagnetischen Wellen, welche von den Radargeräten emittiert wird [29]. Damit kommt es sowohl von der Methodologie als auch von den konkreten Eckdaten sehr nah an die originalen Daten aus dem HKO-7 Datenset heran. Jedoch hat es auch ähnliche Probleme. So kann es bei der Erfassung zu diversen Problemen kommen, wie etwa Störungen durch Flugzeuge und Vögel oder Dämpfungseffekte, welche Niederschlagsmengen verfälschen können [30]. Diese können zwar zum Teil ausgeglichen werden, trotzdem ist wie bei den Daten aus Hong Kong mit Störungen in den Daten zu rechnen.

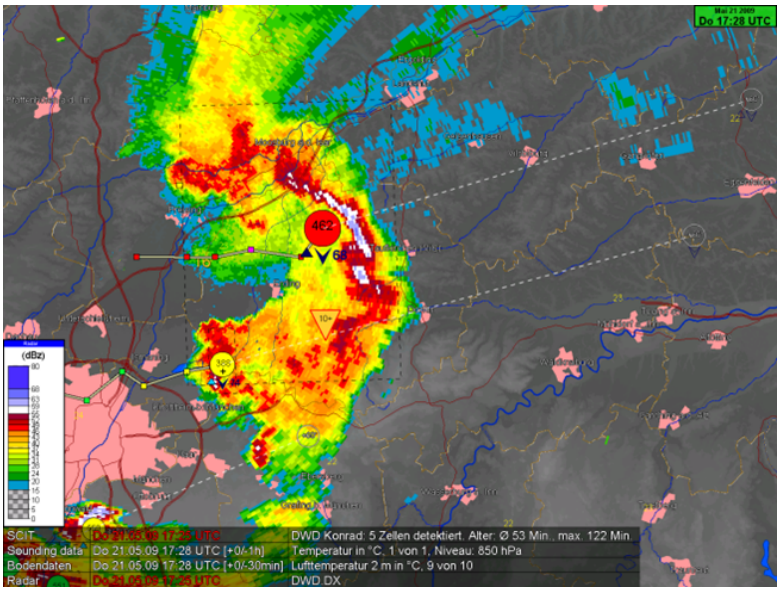


Abbildung 6: Radarreflektivität aus DWD-Niederschlagsscan und KONRAD-Zellerkennung, nordwestlich von München [28].

Das größte Problem liegt allerdings eher im Datenformat der KONRAD-Daten. Dieses setzt sich aus diversen Zellen zusammen, welche zusammenhängende Gruppen aus Pixeln sind, die eine bestimmte Radarreflektivität überschreiten. So ist es nicht möglich, die ursprünglichen Radardaten daraus zu konstruieren welche für das TrajGRU gebraucht werden, oder diese anderweitig sinnvoll in das Modell geben zu können.

3.1.3 COSMO-REA2

Das vom DWD geförderte Hans-Ertel-Zentrum für Wetterforschung bietet außerdem Regionale Reanalysen an, welche auf dem *Consortium for Small-scale Modeling* (COSMO) Modell basieren [31]. Die Reanalyse beschreibt ein Verfahren, in dem mehrere ältere Analysen von *numerical weather prediction* (NWP) Modellen erneut berechnet werden [32], welche dann mit der Datenassimiliation von historischen Messwerten angepasst werden. Dies hat den Vorteil, zurückliegende Erkenntnisse und modernere Methoden verwenden zu können.
Dafür werden mehrere Datensätze angeboten. COSMO-REA6, welches den gesamten europäischen Kontinent mit einer 6 km Auflösung abdeckt, COSMA-REA2, welches Deutschland und teile von Zentraleuropa mit einer Auflösung von 2 km abdeckt, sowie COSMO-ENS-REA12, welches auf COSMO-REA6 basiert, aber einen noch größeren Bereich mit einer 12 km Auflösung abdeckt [31]. Die Daten gehen von 2007 bis 2013 und sind stündlich akkumuliert.

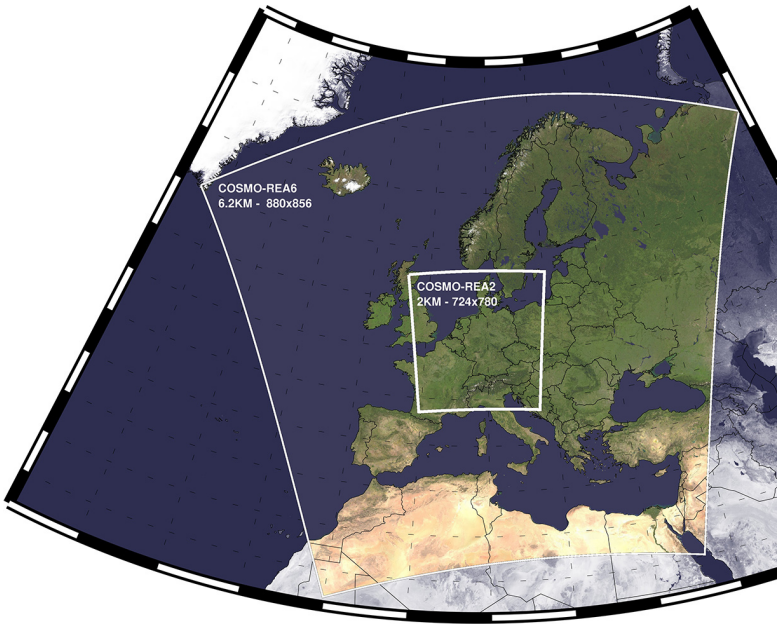


Abbildung 7: Übersicht der COSMO-REA Gebiete [31].

Mit diesen Datensätzen lässt es sich wesentlich einfacher arbeiten. Dadurch, dass bei Reanalysen sehr viele Observationen und vergangene Analysen benutzt werden, sind die Daten näher an einer „objektiven Wahrheit“ und beinhalten wesentlich weniger Störungen. Der gesamte Niederschlag wird in Millimeter pro Stunde gemessen und entspricht damit der Regenrate R [29]. Diese lässt sich gut weiterverwenden, um damit einen Datensatz für das TrajGRU zu erstellen. Der Datensatz wurde für diese Arbeit ausgewählt, da bis auf die zeitliche Auflösung die Eigenschaften relativ nahe an den HKO-7 Datensatz herankommne.

3.1.4 RADOLAN

Der DWD bietet außerdem Daten der *Radar-Online-Aneichung* (RADOLAN) an. Dabei handelt es sich um ein Reflektivitätskomposit von 17 Wetterradaren in Deutschland, welches in zeitlich sowie räumlich hochauflösende Daten liefert [33]. Die räumliche Auflösung beträgt dabei 1 km mit einer zeitlichen Auflösung von 5 Minuten. Dabei wurden vom Institut der Meteorologie an der Freien Universität Berlin die Daten der Regentage von 2017 bis 2020 gespeichert und für diese Arbeit zur Verfügung gestellt. Dieser Datensatz wurde benutzt um die Ergebnisse mit COSMO-REA2 vergleichen zu können.

3.1.5 Verarbeitung

Der COSMO-REA2 Datensatz liegt im *rotated spherical* Format vor [34], weswegen die Daten zuerst rotiert werden. Da den HKO-7 Daten so nah wie möglich gekommen werden soll, wird aus den rotierten COSMO-REA2 Daten sowie den RADOLAN Daten ein 480px x 480px großes Feld über Deutschland ausgeschnitten. Das heißt, dass bei einer Auflösung von 2 km und 1 km, die resultierenden Radardaten für COSMO-REA2 940 Kilometer und für RADOLAN 480 Kilometer umfassen. Die ro-

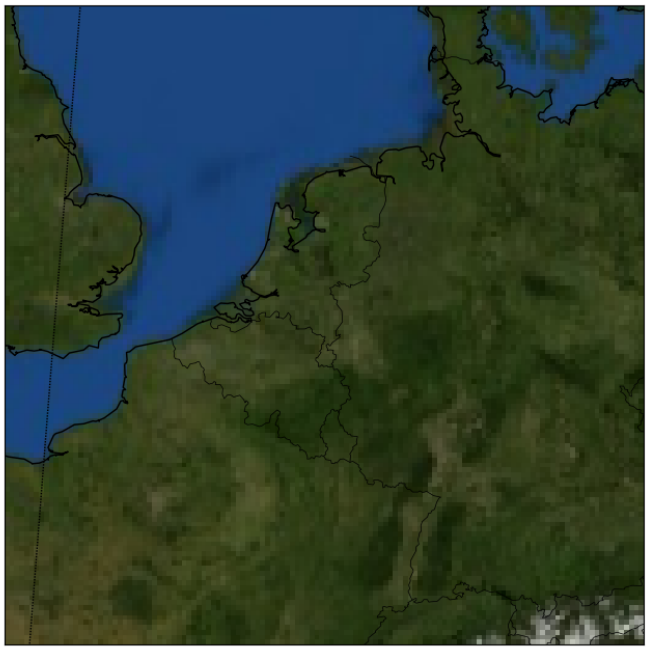


Abbildung 8: Ausschnitt der COSMO-REA2 Daten.

hen Daten müssen in Schwarzweiß-Bilder umgewandelt werden. Dies wird für die Eingabe der *Convolutional Layers* im TrajGRU benötigt, der Schwarzweiß-Kanal stellt dabei die Regenintensität da. Zum Umwandeln wird die Funktion $px(dBZ)$ aus [5] benutzt, um die logarithmische Radarreflektivität dBZ (*decibel relative to Z*, wobei Z der Reflektivitätsfaktor vom Radar zum Regentropfen ist [35]) linear in Pixelwerte zu transformieren:

$$px(dBZ) = \lfloor 255 \cdot \frac{dBZ + 10}{70} + 0.5 \rfloor$$

Der resultierende Wert wird anschließend noch auf den Intervall $[0, 255]$ begrenzt. Da die Daten aus COSMO-REA2 im mm/h Format vorliegen, müssen diese außerdem noch umgewandelt werden. Dafür kann die Marshall-Palmer Formel [36] benutzt werden:

$$R = (\frac{10^{(dBZ/10)}}{200})^{\frac{5}{8}}$$

Diese konvertiert die Radarreflektivität dBZ zur Regenrate R in mm/h, also das Format der COSMO-REA2 Daten. Nach dBZ umgestellt lautet diese:

$$dBZ = \frac{10 \cdot log(200 \cdot R^{\frac{8}{5}})}{log(10)}$$

Da die RADOLAN Daten bereits in dBZ vorliegen, können diese direkt mit der oben beschriebenen Funktion $px(dBZ)$ in Schwarzweiß-Bilder umgewandelt werden.

3.2 Auswertung

Um die verschiedenen Datensätze zu testen, wurde das in Sektion 2.2 beschriebene TrajGRU benutzt. Hierbei wurde eine vortrainierte Version aus [5] verwendet. Konkret handelt es sich dabei um ein TrajGRU mit 3 Ebenen in einer Autoencoder-Struktur, mit jeweils 13, 13 und 9 lokalen Verbindungen. Dieses wurde auf dem HKO-7 Datensatz trainiert. Dabei wurden 812 Tage für das Training, 50 Tage für die Validierung und 131 Tage für das Testen, aus den Tagen mit dem meisten Niederschlag selektiert. Das Modell wurde dabei mit dem Algorithmus Adam [37] optimisiert, mit einer Lernrate von 10^{-4} und einem Momentum von 0.5. Das Training nutzt eine Batchgröße von 4 und wurde basierend auf der Summe der Fehlerfunktionen *Balanced Mean Squared Error* (B-MSE) und *Balanced Mean Absolute Error* (B-MAE) früh gestoppt. Diese sind nach den Proportionen des Niederschlags gewichtete MSE und MAE Funktionen.

Dieses Modell wurde anschließend auf vier verschiedene Datensätze angewendet:

Datensatz	Zeitliche Auflösung (min)	Räumliche Auflösung (km)	Lage
RADOLAN	5	1x1	Deutschland
COSMO-REA2	60	2x2	Deutschland
HKO-7	6	1x1	Hong Kong
HKO-7 (1H Resampled)	60	1x1	Hong Kong

Tabelle 1: Übersicht über die Eigenschaften der Datensätze.

RADOLAN, COSMO-REA2, HKO-7 und eine Version von HKO-7, dessen zeitliche Auflösung auf eine Stunde resampled wurde. Eine Übersicht über die Eigenschaften dieser Datensätze ist in Tabelle 1 zu sehen.

4 Evaluierung

4.1 Methodik

Um die Ergebnisse des Modells auf den Datensätzen zu evaluieren, wurde sich an der Methodik in [5] orientiert. Das Modell erhält jeweils 5 Bilder als Eingabe und versucht die nächsten 20 Bilder vorherzusagen.
Um eine umfassendere Evaluierung der Vorhersagen zu liefern, werden zwei im *precipitation nowcasting* übliche Skill Scores benutzt, der *Critical Success Index* (CSI) und *Heidke Skill Score* (HSS) [38]. Diese stellen eine Messung dar, wie akkurat eine Vorhersage verglichen mit den tatsächlichen Daten ist. Genauer misst der CSI, welcher Anteil der Ereignisse richtig vorhergesagt wurde, während der HSS die Genauigkeit der getroffenen Vorhersage verglichen mit einer rein zufälligen anzeigt. Bei beiden Scores wäre ein Wert von 1 eine perfekte Vorhersage, während ein Wert von 0 keine Verbesserung bedeutet.
Diese beiden Scores werden jeweils für verschiedene Grenzwerte der Regenrate R berechnet (0.5, 2, 5, 10 und 30). Dafür werden die Pixelwerte der *ground truth* und Vorhersage anhand ihres Grenzwertes r binär nach 0 und 1 konvertiert. Danach werden die *True Positives (TP)* (Vorhersage = 1, *ground truth* = 1), *False Negatives (FN)* (Vorhersage = 0, *ground truth* = 1), *False Positives (FP)* (Vorhersage = 1, *ground truth* = 0) und *True Negatives (TN)* (Vorhersage = 0, *ground truth* = 0) berechnet. Um CSI und HSS zu berechnen werden folgende Formeln verwendet [5]:

$$\mathrm{CSI} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP}}$$

$$\mathrm{HSS} = \frac{\mathrm{TP} \cdot \mathrm{TN} - \mathrm{FN} \cdot \mathrm{FP}}{(\mathrm{TP} + \mathrm{FN})(\mathrm{FN} + \mathrm{TN}) + (\mathrm{TP} + \mathrm{FP})(\mathrm{FP} + \mathrm{TN})}$$

Um das Modell an sich zu evaluieren werden die gewöhnlichen Fehlermessungen *Mean Squared Error* (MSE) und *Mean Absolute Error* (MAE) verwendet. Da die Proportionen der Regenverteilung im Datensatz jedoch stark unterschiedlich sind, werden in [5] der *Balanced Mean Squared Error* (B-MSE) und *Balanced Mean Absolute Error* (B-MAE) eingeführt. Für diese wird jedem Pixel ein Gewicht $w(r)$ anhand des Regenbetrags r zugewiesen, und nach diesen der MSE und MAE gewichtet:

$$w(r) = \begin{cases} 1, & r < 2 \\ 2, & 2 \leq r < 5 \\ 5, & 5 \leq r < 10 \\ 10, & 10 \leq r < 30 \\ 30, & r \geq 30 \end{cases}$$

4.2 Ergebnisse

Die Ergebnisse des Modells auf den Testsätzen von HKO-7, COSMO-REA2 und RADOLAN kann man in Abbildung 9 sehen. Die Unterschiede zwischen den Datensätzen sind bei beiden Skill Scores etwa ähnlich. HKO-7 schneidet als der Datensatz, auf dem das Modell trainiert wurde, mit Abstand am besten ab. RADOLAN ist bis

Regenrate R (mm/h)	Anteil HKO-7 (%)	Anteil COSMO-REA2 (%)	Regenstufe
$0 \leq x < 0.5$	90.25	98.6	Kaum bemerkbar
$0.5 \leq x < 2$	4.38	1.4	Leicht
$2 \leq x < 5$	2.46	0.003	Leicht bis mittel
$5 \leq x < 10$	1.35	0	Mittel
$10 \leq x < 30$	1.14	0	Mittel bis stark
$30 \leq x5$	0.42	0	Unwetterwarnung

Tabelle 2: Vergleich zwischen Proportionen der Regenrate R in HKO-7 [5] and COSMO-REA2.

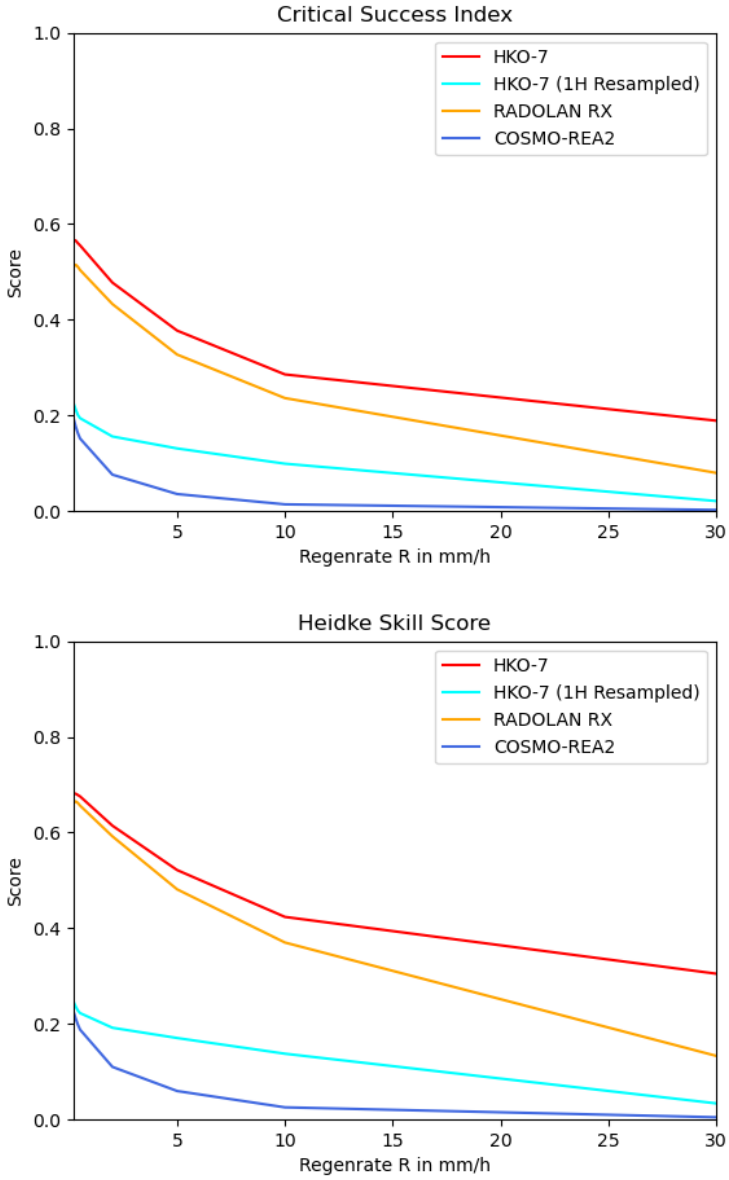


Abbildung 9: TrajGRU Benchmark Ergebnisse auf den verschiedenen Datensätzen.

auf sehr starke Regenereignisse nur minimal schlechter. COSMO-REA2 schneidet mit Abstand am schlechtesten ab und tendiert stark Richtung 0 für Ergebnisse $r \geq 2$.
Dies liegt an der unterschiedlichen Verteilung der Regenereignisse in den beiden Datensätzen, welche in Tabelle 2 zu sehen sind. Während in beiden Datensätzen starke Regenereignisse seltener auftreten als schwächere, gibt es im COSMO-REA2 Satz keine einzige Messung, welche durchschnittlich ≥ 5 mm/h Niederschlag hat und gerade mal zwei, welche einen Niederschlag ≥ 2 mm/h angeben.
Diese Divergenz zwischen den beiden Datensätzen kann man vermutlich mit der Intervallgröße der Messungen erklären. HKO-7 benutzt Daten eines Doppler-Radar, welche alle 6 Minuten aufgenommen werden [5], also 240 Bilder pro Tag. Die COSMO Reanalysen sind hingegen nur stündlich erhältlich. Da starke Regenereignisse normalerweise nur von kurzer Dauer sind, gehen diese vermutlich im Durchschnitt der stündlichen Reanalysen unter.

Die offene Frage ist warum zwei verschiedene Datensätze mit deutschen Daten so unterschiedlich abschneiden, denn auch für Regenereignisse mit weniger als 0.5 mm/h schneidet COSMO-REA2 deutlich schlechter ab als RADOLAN. Dieser hat einen 5-Minuten Intervall und weist ähnlich gute Ergebnisse auf wie der originale HKO-7 Datensatz. COSMO-REA2 hingegen hat einen Intervall von einer Stunde und einen sehr großen Unterschied was die Frequenz der Daten sowie die Länge der Eingaben und Vorhersagen angeht. Um zu testen, ob die zeitliche Auflösung die beobachteten Unterschiede erklären kann, wurde das Modell nochmal auf eine Version von HKO-7 angewendet, dessen Intervallgröße auf eine Stunde resampled wurde. Dafür wurden die Regenereignisse im Stundentakt akkumuliert. Die Ergebnisse auf diesem Datensatz sind ebenso in Abbildung 9 zu sehen und sind nur marginal besser als die von COSMO-REA2. Dies legt nahe, dass die Leistung des Modells stark von der Intervallgröße der Daten bzw. der zeitlichen Länge der versuchten Vorhersage abhängt.

4.3 Diskussion

4.3.1 Auswahl der deutschen Daten

Für die Auswahl des deutschen Datensatz wurden primär KONRAD und COSMO-REA2 aus Sektion 3.1 betrachtet. KONRAD war die erste Wahl aufgrund der hohen Auflösung und der Ähnlichkeit vom zeitlichen Aufnahmeintervall zum HKO-7 Datensatz. Jedoch ist KONRAD primär auf die Unwetterwarnung ausgerichtet, weswegen sich der Aufbau der Daten vorallem nach Zugrichtung und Intensität von Gewittern richtet, und nicht nach einer granularen Übersicht von Niederschlag, weswegen schlussendlich COSMO-REA2 für diese Arbeit ausgewählt wurde. RADOLAN wurde hinzugezogen, um die Ergebnisse auf COSMO-REA2 vergleichen zu können.

4.3.2 Anpassungen an den Datensatz

Wie in Sektion 4.2 bereits erwähnt, bestehen starke Differenzen zwischen den Datensätzen in den Proportionen der Regenrate. Als Lösung weisen Shi et al. in [5] jedem Pixel ein Gewicht $w(r)$ anhand der Regenrate r zu, um danach die MSE und MAE Scores danach zu gewichten (siehe Sektion 4.1).

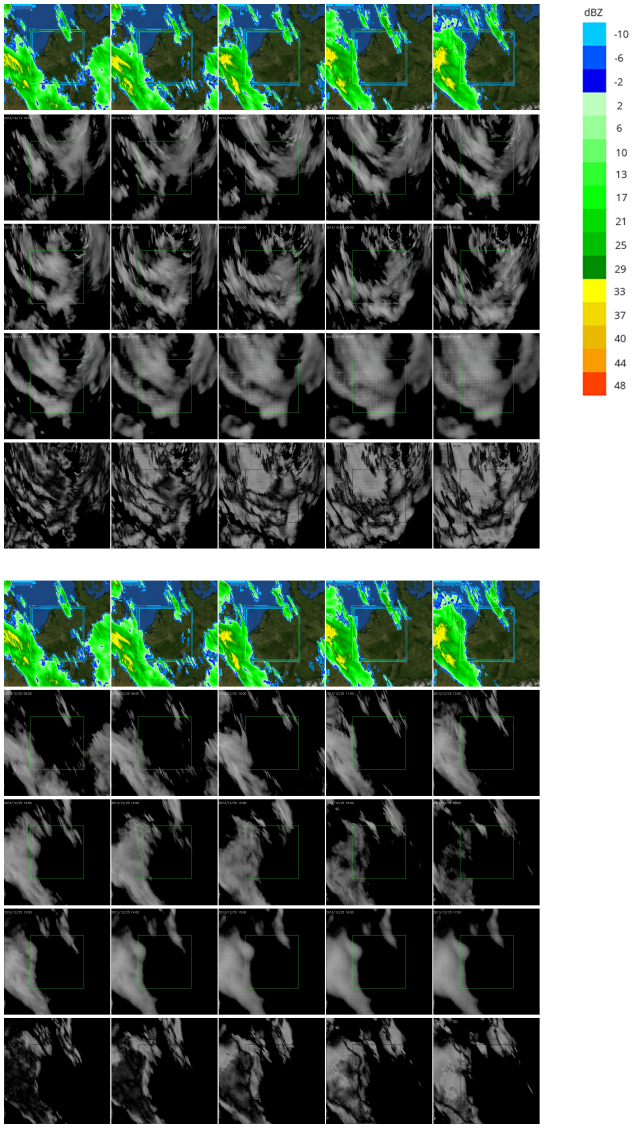


Abbildung 10: Zwei Samples des TrajGRU *precipitation nowcasting* auf COSMO-REA2. Die erste Reihe sind die Eingabedaten in Farbskala über der Landkarte. Die zweite Reihe ist die rohe Eingabe. Die dritte Reihe ist die *ground truth*. Die vierte Reihe ist die Vorhersage. Die fünfte Reihe ist die Differenz zwischen *ground truth* und Vorhersage.

Um die Leistung auf COSMO-REA2 zu verbessern, wurde eine neue Gewichtungsfunktion gefunden, welche den COSMO-REA2 Datensatz besser abbildet. In Abbildung 11 ist die Verteilung des Regenfalls in COSMO-REA2 etwas granularer dargestellt. Anhand dieser Kategorisierung wurde eine Gewichtungsfunktion $w'(r)$ aufgestellt:

$$w'(r) = \begin{cases} 1, & r < 0.1 \\ 2, & 0.1 \leq r < 0.2 \\ 5, & 0.2 \leq r < 0.3 \\ 10, & 0.3 \leq r < 1 \\ 30, & r \geq 1 \end{cases}$$

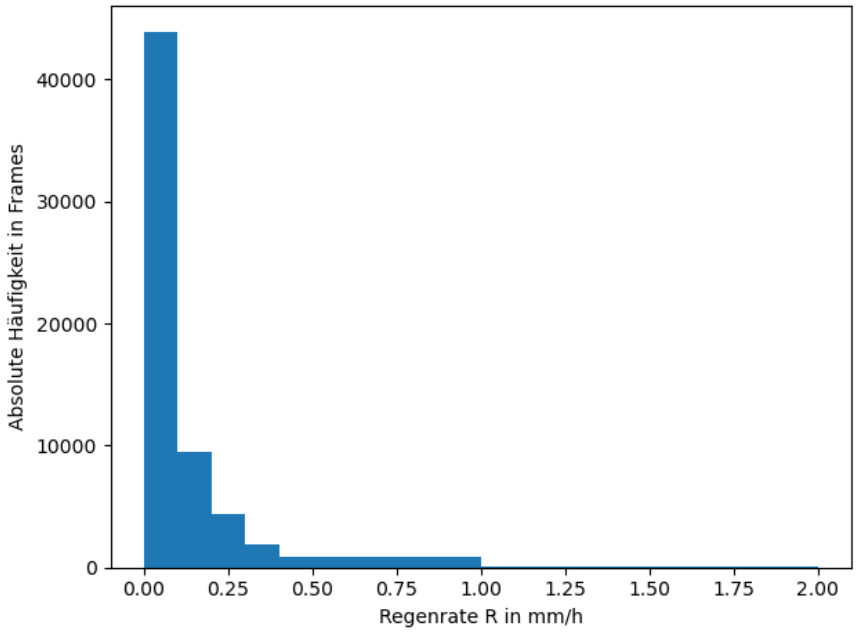


Abbildung 11: Verteilung des Regenfalls in COSMO-REA2.

Gewichtung	CSI					HSS				
	$r \geq 0.1$	$r \geq 0.2$	$r \geq 0.3$	$r \geq 0.4$	$r \geq 0.5$	$r \geq 0.1$	$r \geq 0.2$	$r \geq 0.3$	$r \geq 0.4$	$r \geq 0.5$
$w(r)$	0.216128	0.19306	0.17632	0.162802	0.151824	0.243693	0.225246	0.210645	0.198362	0.188091
$w'(r)$	0.216101	0.193032	0.176292	0.162774	0.151797	0.243638	0.225192	0.210593	0.198312	0.188047

Tabelle 3: TrajGRU Benchmark Ergebnisse auf COSMO-REA2 mit unterschiedlichen Gewichtungsfunktionen.

Dies macht allerdings kaum einen Unterschied in der Leistung des Modells, wie in Tabelle 3 zu sehen.

5 Fazit

5.1 Zusammenfassung

In dieser Arbeit wurde ein auf Daten aus Hong Kong vortrainiertes Modell für *precipitation nowcasting* auf deutschen Daten angewendet. Dabei wurde die Hypothese aufgestellt, dass die zeitliche Auflösung der Eingabedaten einen wesentlichen Unterschied in der Leistung des Modells macht.

Die in Sektion 4.2 diskutierten Ergebnisse legen nahe, dass die Veränderung der zeitlichen Auflösung der Daten die Leistung des vortrainierten TrajGRUs stark verändern. So sind die Vorhersagen des TrajGRU auf Daten mit einer zeitlichen Auflösung von einer Stunde deutlich schlechter, als auf Daten die näher an der ursprünglichen Trainingsauflösung von 6 Minuten sind.

Interessanterweise scheint die räumliche Lage der Daten keinen großen Einfluss zu haben. So sind Vorhersagen, welche auf deutschen Daten getroffen wurden, nur etwas schlechter, als Vorhersagen basierend auf den ursprünglichen Daten aus Hong Kong. Diese könnten mit verschiedenen Ansätzen wie *Transfer Learning* sogar noch verbessert werden.

5.2 Zukünftige Arbeit

5.2.1 Transfer Learning

Es würde sich anbieten, *Transfer Learning* auf dem vortrainierten TrajGRU Modell anzuwenden. Wie in Sektion 2.3 erklärt, eignet sich diese Methode, wenn anzunehmen ist, dass sich die generellen Features vom vortrainierten und neuen Datensatz nicht stark unterscheiden oder der neue Datensatz wesentlich kleiner ist. Beide Fälle treffen hier zu: COSMO-REA2 ist durch die stündlichen Intervalle in etwa 10 mal kleiner als HKO-7. Ersterer besteht aus nur 24 Bildern pro Tag, während letzterer aus 240 Bildern pro Tag besteht. RADOLAN ist ebenfalls deutlich kleiner als HKO-7, durch die selektive Auswahl der Regentage. Außerdem können wir annehmen, dass die generellen Features des Niederschlags sich stark ähneln, es jedoch spezifischere regionale Unterschiede gibt, welche das auf HKO-7 vortrainierte TrajGRU nicht gut vorhersagen kann. Deswegen ist anzunehmen, dass *Transfer Learning* die Leistung des Modells auf den deutschen Daten wesentlich erhöhen könnte.
Wie in Sektion 2.2 erläutert, besitzt das TrajGRU Modell die Struktur eines *Autoencoder*. Die Ebenen dessen funktionieren wie oben beschrieben: Die äußersten Ebenen des Kodierers lernen die generellen Features und die inneren die spezielleren. Symmetrisch dazu dekodieren die inneren Ebenen des Dekodierer die speziellen und die äußeren Ebenen die generellen Features. Ein Ansatz, um *Transfer Learning* auf das TrajGRU anzuwenden, könnte also sein, die Gewichte aller äußeren Ebenen des *Autoencoders* zu fixieren. Anschließend würden die innersten Ebenen sowohl des Kodierers als auch des Dekodierers zurückgesetzt und auf den deutschen Daten erneut trainiert werden.

5.2.2 Vergleich mit anderen Modellen

Interessant für das Subjekt dieser Arbeit wäre auch ein Vergleich mit anderen auf *Deep Learning* basierenden Modellen für *precipitation nowcasting* bezüglich Generalisierung auf verschiedenen Datensätzen. In Sektion 1.2 wurden mehrere CNN basierte Ansätze vorgestellt. Es könnten diese (besonders U-Net basierte) und RNN basierte (wie dem hier besprochenen TrajGRU) Modelle auf Datensätzen aus anderen Ländern angewendet und so verglichen werden, welche Architekturen am besten auf andere Daten generalisieren. Dafür würden sich die hier verwendeten Datensätze (HKO-7 und RADOLAN), aber auch andere wie zum Beispiel NEXRAD [39] anbieten, welches Reanalysen aus den Vereinigten Staaten anbietet.

Literatur

[1] World Meteorological Organization. *Nowcasting*. https://www.wmo.int/pages/prog/amp/pwsp/Nowcasting.htm. [Online, aufgerufen am 22. Dezember 2020].

[2] J. Sun, M. Xue, J. W. Wilson, I. Zawadzki, S. P. Ballard, J. Onvlee-Hooimeyer, P. Joe, D. M. Barker,P. W. Li, B. Golding, M. Xu, and J. Pinto. „Use of NWP for Nowcasting Convective Precipitation: Recent Progress and Challenges“. In: *Bulletin of the American Meteorological Society* 95 (2014).

[3] Charu C. Aggarwal. *Data Mining: The Textbook*. Springer, 2015.

[4] Xingjian Shi u. a. „Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting“. In: *Advances in Neural Information Processing Systems*. 2015. URL: https://proceedings.neurips.cc/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf.

[5] Xingjian Shi u. a. „Deep learning for precipitation nowcasting: a benchmark and a new model“. In: *Advances in Neural Information Processing Systems*. 2017.

[6] Olaf Ronneberger, Philipp Fischer und Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[7] Shreya Agrawal u. a. *Machine Learning for Precipitation Nowcasting from Radar Images*. 2019. arXiv: 1912.12132 [cs.CV].

[8] Kevin Trebing, Tomasz Stanczyk und Siamak Mehrkanoon. *SmaAt-UNet: Precipitation Nowcasting using a Small Attention-UNet Architecture*. 2021. arXiv: 2007.04417 [cs.LG].

[9] Sanghyun Woo u. a. *CBAM: Convolutional Block Attention Module*. 2018. arXiv: 1807.06521 [cs.CV].

[10] G. Ayzel, T. Scheffer und M. Heistermann. „RainNet v1.0: a convolutional neural network for radar-based precipitation nowcasting“. In: *Geoscientific Model Development* 13.6 (2020), S. 2631–2644. DOI: 10.5194/gmd-13-2631-2020. URL: https://gmd.copernicus.org/articles/13/2631/2020/.

[11] Vijay Badrinarayanan, Alex Kendall und Roberto Cipolla. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. 2016. arXiv: 1511.00561 [cs.CV].

[12] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[13] Eric W. Weisstein. *Convolution*. https://mathworld.wolfram.com/Convolution.html. [Online, aufgerufen am 24. Dezember 2020].

[14] Zhou und Chellappa. „Computation of optical flow using a neural network“. In: *IEEE 1988 International Conference on Neural Networks*. 1988, 71–78 vol.2. DOI: 10.1109/ICNN.1988.23914.

[15] Alex Graves. *Generating Sequences With Recurrent Neural Networks*. 2014. arXiv: 1308.0850 [cs.NE].

[16] Sepp Hochreiter und Jürgen Schmidhuber. „Long Short-term Memory“. In: *Neural computation* 9 (Dez. 1997), S. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[17] Samuel Dupond. „A thorough review on the current advance of neural network structures“. In: *Annual Reviews in Control* 14 (2019), S. 200–230.

[18] Nicol N. Schraudolph Felix A. Gers und Jürgen Schmidhuber. „Learning Precise Timing with LSTM Recurrent Networks“. In: *Journal of Machine Learning Research* 3 (2002), S. 115–143.

[19] Michael Phi. *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21. [Online, aufgerufen am 26. Mai 2021].

[20] Kyunghyun Cho u. a. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 [cs.CL].

[21] Junyoung Chung u. a. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. arXiv: 1412.3555 [cs.NE].

[22] Fisher Yu und Vladlen Koltun. *Multi-Scale Context Aggregation by Dilated Convolutions*. 2016. arXiv: 1511.07122 [cs.CV].

[23] James J. Gibson. *The Perception of the Visual World*. Houghton Mifflin, 1950.

[24] Eddy Ilg u. a. *FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks*. 2016. arXiv: 1612.01925 [cs.CV].

[25] Dor Bank, Noam Koenigstein und Raja Giryes. *Autoencoders*. 2020. arXiv: 2003.05991 [cs.LG].

[26] J. Deng u. a. „ImageNet: A Large-Scale Hierarchical Image Database“. In: *CVPR09*. 2009.

[27] Jason Yosinski u. a. *How transferable are features in deep neural networks?* 2014. arXiv: 1411.1792 [cs.LG].

[28] Deutscher Wetterdienst. *Konvektive Entwicklung (KONRAD)*. https://www.dwd.de/DE/forschung/wettervorhersage/met_fachverfahren/nowcasting/konrad_node.html. [Online, aufgerufen am 13. November 2020].

[29] Andreas Bott. *Synoptische Meteorologie: Methoden der Wetteranalyse und -prognose*. Springer Spektrum, 2016.

[30] Deutscher Wetterdienst. *Qualittätssicherung von Wetterradardaten*. https://www.dwd.de/DE/forschung/wettervorhersage/met_fachverfahren/wetterradarverfahren/qualitaetssicherung_wetterradardaten_node.html. [Online, aufgerufen am 13. November 2020].

[31] Hans-Ertel-Zentrum für Wetterforschung. *COSMO Regional Reanalysis - Overview*. https://reanalysis.meteo.uni-bonn.de/?Overview. [Online, aufgerufen am 20. November 2020].

[32] Deutscher Wetterdienst. *Regionale Reanalyse*. https://www.dwd.de/DE/klimaumwelt/klimaueberwachung/reanalyse/reanalyse_node.html. [Online, aufgerufen am 20. November 2020].

[33] Deutscher Wetterdienst. *RADOLAN (Radar-Online-Aneichung): Analysen der Niederschlagshöhen aus Radar- und stationsbasierten Messungen im Echtzeitbetrieb*. https://www.dwd.de/DE/leistungen/radolan/radolan.html. [Online, aufgerufen am 1. Mai 2021].

[34] Hans-Ertel-Zentrum für Wetterforschung. *COSMO-REA2 Specifications*. https://reanalysis.meteo.uni-bonn.de/?COSMO-REA2___Specifications. [Online, aufgerufen am 20. November 2020].

[35] National Weather Service. *Weather Glossary*. https://w1.weather.gov/glossary/index.php?letter=d. [Online, aufgerufen am 13. Dezember 2020].

[36] DesktopDoppler. *NWS NEXRAD*. https://web.archive.org/web/20160113151652/http://www.desktopdoppler.com/help/nws-nexrad.htm#rainfall%20rates. [Online, aufgerufen am 20. November 2020].

[37] Diederik P. Kingma und Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[38] Robin J Hogan, Christopher AT Ferro, Ian T Jolliffe, and David B Stephenso. „Equitability Revisited: Why the 'Equitable Threat Score' is not equitable". In: *Weather and Forecasting* 25 (2010).

[39] Dr. Jian Zhang und Dr. Jonathan Gourley. *Multi-Radar Multi-Sensor Precipitation Reanalysis*. 2018. DOI: 10.25638/EDC.PRECIP.0001.