

BACHELORTHESIS  
Matteo Stein-Cadenbach

# Merkmalsextraktion durch LSTM-Autoencoder am Beispiel von Aerosol-Rückstreuprofilen aus LIDAR-Ceilometern

---

FAKULTÄT TECHNIK UND INFORMATIK  
Department Informatik

Faculty of Computer Science and Engineering  
Department Computer Science

Matteo Stein-Cadenbach

Merkmalsextraktion durch LSTM-Autoencoder am  
Beispiel von Aerosol-Rückstreuprofilen aus  
LIDAR-Ceilometern

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung  
im Studiengang *Bachelor of Science Angewandte Informatik*  
am Department Informatik  
der Fakultät Technik und Informatik  
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Michael Neitzke  
Zweitgutachter: Prof. Dr. Ulrike Steffens

Eingereicht am: 17. Februar 2022

## **Matteo Stein-Cadenbach**

### **Thema der Arbeit**

Merkmalsextraktion durch LSTM-Autoencoder am Beispiel von Aerosol-Rückstreuprofilen aus LIDAR-Ceilometern

### **Stichworte**

Merkmalsextraktion, LSTM, Autoencoder, VAE, Deep-Clustering, Aerosol-Rückstreuprofile, Mustererkennung, Deep-Learning, Zufällige Zeitreihensegmentierung

### **Kurzzusammenfassung**

Durch Messungen vom deutschen Wetterdienst ist ein kontinuierlicher Datenstrom mit den Eigenschaften von Wetterphänomenen gegeben. Diese Phänomene sind von variabler Dauer und erscheinen zu unvorhersehbaren Zeitpunkten. Eine statische Fensterbreite zur Merkmalsextraktion durch Autoencoder erzeugt keine optimalen Segmente für die unüberwachte Detektion und Klassifikation der Phänomene. Die Klassifikation muss aufgrund fehlender Labels über ein Clustering erfolgen. Sie könnte durch eine unbekannt optimale Segmentierung des Datenstroms begünstigt werden.

Das Ziel der vorliegenden Arbeit ist die Untersuchung der Eignung von LSTM-Autoencodern für die Merkmalsextraktion unter einer Segmentierung mit variablen Fensterbreiten. Hierbei soll ein neuartiger Lösungsansatz als Platzhalter für die unbekannt optimale Segmentierung genutzt werden. Der Ansatz besteht aus der Abtastung des Datensatzes mit zufälligen, aber gleichverteilten Fensterbreiten.

Für eine überwachte Messung der Clusterqualität wurde ein gelabelter künstlicher Datensatz mit den für die Untersuchung notwendigen Eigenschaften der realen Daten hergestellt. Ein Prozess wurde entwickelt, um mithilfe der Labels eine ideale Partitionierung der Segmente zu bilden. Um zu überprüfen, dass die neuartige Segmentierung eine Annäherung an eine optimale Segmentierung darstellt, wurde sie einer statischen Segmentierung gegenübergestellt. Zuletzt wurden implementierte Varianten von LSTM-Autoencodern mit Clustering-Verfahren angewandt und miteinander verglichen.

Die Ergebnisse zeigen bessere Kennzahlen für die neuartige Segmentierung. Außerdem zeigen sie, dass bei Wahl unterschiedlicher maximaler Segmentbreiten für die Gleichverteilung eine stabile Clusterqualität gewährleistet ist, während dies bei Wahl unterschiedlicher statischer Segmentbreiten nicht der Fall ist. Im Vergleich der Architekturvarianten

---

ist festzustellen, dass ein VAE mit klassischen LSTM-Schichten zur Kodierung der Segmente am besten geeignet ist.

Weiterführende Forschung ist hinsichtlich alternativer Clustering-Methoden und einer gezielten Segmentierung vorzunehmen.

---

**Matteo Stein-Cadenbach**

**Title of Thesis**

Feature extraction using LSTM autoencoders on the example of aerosol backscatter profiles from LIDAR ceilometers

**Keywords**

Feature extraction, LSTM, Autoencoder, VAE, Deep Clustering, Aerosol backscatter profiles, Pattern recognition, Deep Learning, Random time-series segmentation

**Abstract**

A continuous data stream with properties of weather phenomena of variable duration and unpredictable appearance is given by measurements by the German Meteorological Service. The usage of a static window length for an Autoencoder-based feature extraction does not produce optimal segments in order to detect and classify the phenomena in an unsupervised manner. The classification has to be done through clustering due to missing labels. It could be improved by an unknown optimal segmentation of the data stream.

Goal of this thesis is to investigate the suitability of LSTM autoencoders for the feature extraction under usage of a segmentation with variable window lengths. Thereby a novel approach shall be used as placeholder for the unknown optimal segmentation. The data stream is segmented by using a sliding window with a random, but uniformly distributed window length.

In order to measure the clustering quality in a supervised way, a labeled synthetic dataset was created according to the mentioned real data properties. A process was defined to build an ideal partitioning of the segments for the calculation of rating indices. To verify that the novel segmentation method can serve as a valid placeholder for the optimal segmentation, it was compared to a segmentation with static window length. At last, implemented variants of LSTM autoencoder combined with clustering were applied and compared.

The results show better ratings for the new segmentation method. Furthermore, they show a stable clustering quality picking different maximal segment lengths for the uniform distribution. In contrast, the clustering quality differs highly when picking different static segment lengths. The comparison of the architecture variants shows that a VAE

---

with conventional LSTM layers is most suitable for encoding the segments. Further research could be done by using other clustering techniques and applying systematic segmentation methods.

# Inhaltsverzeichnis

Abbildungsverzeichnis	ix
Tabellenverzeichnis	xi
<b>1 Einleitung</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Zielsetzung . . . . .	2
1.3 Aufbau der Arbeit . . . . .	4
<b>2 Grundlagen</b>	<b>5</b>
2.1 Convolutional Layer . . . . .	5
2.2 LSTM . . . . .	7
2.3 ConvLSTM . . . . .	11
2.4 Autoencoder . . . . .	12
2.5 VAE . . . . .	13
2.6 Clustering . . . . .	17
2.7 Deep Clustering . . . . .	18
2.8 DEC . . . . .	19
<b>3 Analyse</b>	<b>23</b>
3.1 Künstliche Daten . . . . .	23
3.2 Reale Daten . . . . .	26
<b>4 Entwurf</b>	<b>31</b>
4.1 NaN-Werte . . . . .	31
4.2 Autoencoder . . . . .	32
4.2.1 Flaschenhals . . . . .	33
4.2.2 Architektur . . . . .	35
4.3 Clustering . . . . .	37
4.3.1 Deep Clustering . . . . .	37

4.3.2	Interne Kennzahlen . . . . .	37
4.3.3	Externe Kennzahlen . . . . .	39
<b>5</b>	<b>Implementierung</b>	<b>42</b>
5.1	Werkzeuge . . . . .	42
5.2	Datenformat . . . . .	43
5.3	Ordnerstruktur . . . . .	43
5.4	Konfiguration . . . . .	45
5.5	Datengenerierung . . . . .	46
5.6	Segmentierung . . . . .	48
5.7	Training . . . . .	49
<b>6</b>	<b>Ergebnisse</b>	<b>51</b>
6.1	Maximale Segmentbreite . . . . .	51
6.2	Vergleich variabler und fixer Breite . . . . .	52
6.3	Vergleich der Modelle . . . . .	54
6.4	Anwendung auf reale Daten . . . . .	55
6.5	Diskussion . . . . .	58
<b>7</b>	<b>Fazit</b>	<b>60</b>
7.1	Zusammenfassung . . . . .	60
7.2	Ausblick . . . . .	60
	<b>Literaturverzeichnis</b>	<b>62</b>
<b>A</b>	<b>Anhang</b>	<b>66</b>
A.1	Herleitungen . . . . .	66
	<b>Glossar</b>	<b>67</b>
	<b>Selbstständigkeitserklärung</b>	<b>74</b>

# Abbildungsverzeichnis

1.1	Problem der statischen schmalen Zeitscheibe . . . . .	2
1.2	Problem der statischen breiten Zeitscheibe . . . . .	3
2.1	Feature-Map-Erzeugung für eine 2-dimensionale Eingabe . . . . .	6
2.2	Äquivarianz bzgl. der Verschiebungstransformation . . . . .	6
2.3	Struktur einer LSTM-Zelle . . . . .	7
2.4	Verschwindender Gradient im RNN . . . . .	9
2.5	Verschwindender Gradient im LSTM . . . . .	10
2.6	Feature-Map gleichende Struktur im Cell-State vom ConvLSTM . . . . .	11
2.7	Struktur eines Autoencoders . . . . .	12
2.8	Struktur eines Variational Autoencoders . . . . .	13
2.9	Graphisches Modell ohne Variational-Inference . . . . .	14
2.10	Graphisches Modell mit Variational-Inference . . . . .	15
2.11	Erzwungene Ordnung im latenten Raum des VAEs . . . . .	17
2.12	Exemplarisches Netzwerk für DEC in Phase 2 . . . . .	20
2.13	Funktion $x^2$ im Intervall $[0, 1]$ . . . . .	21
3.1	Exemplarische Generierung künstlicher Daten . . . . .	25
3.2	Visuelle Maskierung negativer Werte . . . . .	28
3.3	Häufigkeitsverteilung vor und nach einer Log-Transformation . . . . .	30
4.1	1. Ansatz für den Flaschenhals des Autoencoders . . . . .	33
4.2	2. Ansatz für den Flaschenhals des Autoencoders . . . . .	33
4.3	3. Ansatz für den Flaschenhals des Autoencoders . . . . .	34
4.4	Entwurf LSTM-AE . . . . .	35
4.5	Entwurf LSTM-VAE . . . . .	36
5.1	Ordnerstruktur . . . . .	44
5.2	Ordnerstruktur eines Trainingprotokolls . . . . .	45

6.1	Durchschnittlicher Rekonstruktionsfehler nach 10 Epochen im Vergleich von maximalen Breitewerten . . . . .	52
6.2	Durchschnittlicher Rekonstruktionsfehler pro Epoche im Vergleich zwischen fixer Breite von 48 und variabler Breite mit 96 als Maximum . . . .	54
A.1	Herleitung $\mathbb{E}_a[f(a) + b] = \mathbb{E}_a[f(a)] + b$ . . . . .	66
A.2	Herleitung $\log \frac{a}{b} = -\log \frac{b}{a}$ . . . . .	66

# Tabellenverzeichnis

2.1	Deep-Clustering-Kategorien . . . . .	18
3.1	Repräsentanten interessanter Phänomene . . . . .	23
3.2	Künstliche RGB-Bilder als Repräsentanten interessanter Phänomene . . . . .	24
3.3	Provozierung der Zeitscheibenproblematik . . . . .	25
3.4	Fehlschlagende Rekonstruktion mit Min-Max-Skalierung . . . . .	26
3.5	Anteil negativer und invalider Werte im kalibrierten und bereinigten Datensatz $X_{Leipzig18}$ mit einer Gesamtgröße von 499 875 840 Werten . . . . .	27
3.6	Abbildung von logarithmischen Intervallen auf lineare Intervalle . . . . .	29
3.7	Rekonstruktionen von 3 repräsentativen Eingaben unter Anwendung verschiedener Datentransformationen nach jeweils 25 Epochen . . . . .	30
4.1	Vergleich der Ansätze zum Umgang mit NaN-Werten nach Trainings mit jeweils 25 Epochen . . . . .	32
4.2	Flags eines Segments mit Label [1, 0.33, 0.66] . . . . .	39
4.3	Klassen mit zugewiesenen Label-Zentroiden . . . . .	39
4.4	Beispiel einer Kontingenztabelle . . . . .	41
5.1	Datenformat des Datensatzes . . . . .	43
5.2	Konfigurationsparameter . . . . .	46
5.3	Attribute in <code>FigureProperties</code> . . . . .	47
6.1	Vergleich maximaler Segmentbreiten . . . . .	51
6.2	Vergleich fixer und variabler Segmentbreiten . . . . .	53
6.3	Kennzahlen unter Anwendung verschiedener Varianten mit einer maximalen Breite von 96 . . . . .	54
6.4	Interne Kennzahlen unter Anwendung verschiedener Varianten mit einer maximalen Breite von 96 für reale Daten . . . . .	55
6.5	Beschreibung der Cluster bei Anwendung eines LSTM-VAEs . . . . .	56

6.6	Beschreibung der Cluster bei Anwendung eines LSTM-AEs . . . . .	56
6.7	Beschreibung der Cluster bei Anwendung eines ConvLSTM-VAEs . . . . .	57
6.8	Beschreibung der Cluster bei Anwendung eines ConvLSTM-AEs . . . . .	57
6.9	Beschreibung der Cluster bei Anwendung eines LSTM-VAEs ohne IDEC .	58

# 1 Einleitung

## 1.1 Motivation

Der Deutsche Wetterdienst (DWD) führt mittels LiDAR-Ceilometern <sup>1</sup> vom Typ CHMK15K kontinuierlich Messungen der Atmosphärenbeschaffenheit durch. Bei einer Messung werden Laserstrahlen in die Atmosphäre gesendet und bei Reaktion mit den sich dort befindenden Partikeln und Molekülen teilweise reflektiert. Die Rückstreuung der Strahlen wird mit einem Empfangsteleskop erfasst und durch eine gegebene Signallaufzeit des Laserstrahls Höhenbereichen zugeordnet. (vgl. [OTT HydroMet Fellbach GmbH, 2021], Kap. 4) Aus den sequentiellen Messungen ergibt sich ein Datensatz mit einer zeitlichen und einer räumlichen Dimension. Er ermöglicht Rückschlüsse auf meteorologische Ereignisse wie z.B. Wolken, Nebel oder Niederschlag.

In [Gandraß, 2019] wurde ein prototypisches Verfahren entwickelt, um eine Detektion von Wetterphänomenen zu automatisieren. Das Verfahren besteht darin, den Datensatz mit einem Fenster sequentiell an der zeitlichen Achse abzutasten und je Abtastung mit einem CNN-Autoencoder einen schmalen Merkmalsvektor zu extrahieren. Der Merkmalsraum nimmt durch Nutzung von *Deep Clustering* eine clusterfreundliche Struktur an, während gleichzeitig die Qualität der Merkmalsrepräsentation erhalten bleibt. In der Arbeit konnten dadurch bereits erfolgreich einige für Wetterphänomene repräsentative Cluster entdeckt werden.

Die zeitliche Achse wurde mit einem statischen Fenster abgetastet. In Kapitel 3.4 aus [Gandraß, 2019] wurden verschiedene Fensterbreiten verglichen, wobei sich eine Breite von 32 Messungen als angemessen herausgestellt hat. Hierbei besteht die Einschränkung, raum-temporale Muster, die sich über einen längeren Zeitraum erstrecken, nicht oder nur teilweise erkennen zu können.

---

<sup>1</sup>*Light Detection and Ranging* (LiDAR) ist eine Technik zur Erfassung des räumlichen Umfelds mithilfe von Laserimpulsen.

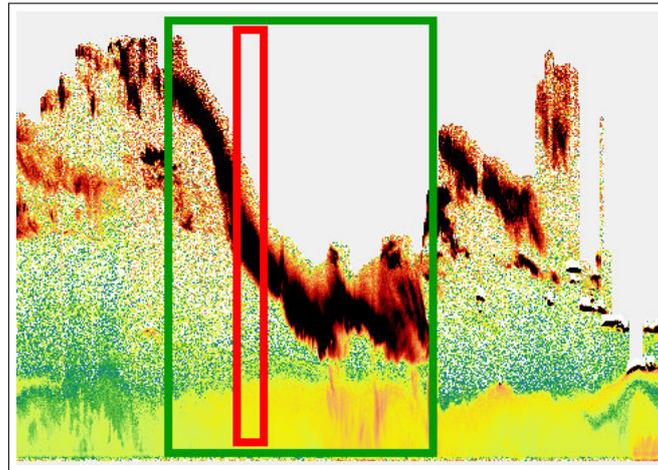


Abbildung 1.1: Problem der statischen schmalen Zeitscheibe

In Abbildung 1.1 wird das relevante Muster, für das idealerweise ein Cluster entdeckt wird, durch das grüne Rechteck markiert. Das rote Rechteck zeigt die statische schmale Zeitscheibe zu einem bestimmten Zeitpunkt. Hierbei wird der Nachteil einer kleinteiligen Segmentierung deutlich. Eine Gruppierung von Segmenten mit dem im grünen Rechteck enthaltenen komplexen Muster ist nicht möglich. Um dieses Problem zu behandeln, kann der Datensatz mit einer variablen Fensterbreite segmentiert werden. Eine variable Segmentierung kann erfolgen, indem gezielt Zeitpunkte ermittelt werden, zu denen ein Kriterium wie eine unerwartete Zustandsänderung erfüllt wird. Ein solcher Prozess wird *Change Point Detection* genannt.

### 1.2 Zielsetzung

Ziel dieser Arbeit ist die Untersuchung von Architekturen mit enkodierenden und dekodierenden LSTM-Faltungsnetzen unter der Verwendung von variablen Zeitscheiben. Ein weitergreifendes, aber abzugrenzendes Ziel wäre der Einsatz von Methoden der *Change Point Detection* zur zeitlichen Segmentierung des Datensatzes. Im Rahmen dieser Arbeit soll der Datensatz stattdessen zufallsbasiert segmentiert werden. Bei der Segmentierung soll sichergestellt werden, dass eine Gleichverteilung der möglichen Segmentbreiten in einem Bereich zwischen minimaler und maximaler Breite vorherrscht. Die zufällige Segmentierung soll sowohl das in Abbildung 1.1 dargestellte Problem der Abschneidung von zusammenhängenden Mustern als auch das in Abbildung 1.2 dargestellte Problem der Zusammenfassung separater Muster abschwächen. Die variable Segmentierung erhöht

die Wahrscheinlichkeit, dass optimale Segmente im aufbereiteten Datensatz auftauchen. Ein zu schmales Fenster schneidet mit höherer Wahrscheinlichkeit zusammenhängende Muster ab, sodass Ideal-Cluster nicht entdeckt werden könnten. Andererseits sorgt ein schmales Fenster für eine bessere Separation, da tendenziell nur ein Phänomen erfasst wird.

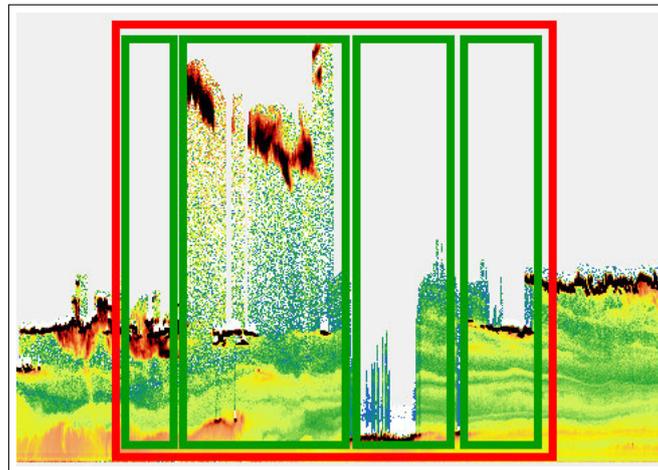


Abbildung 1.2: Problem der statischen breiten Zeitscheibe

Ein zu breites Fenster erfasst mit höherer Wahrscheinlichkeit gleichzeitig Muster unterschiedlicher Ideal-Cluster, sodass eine passende Separation dieser erschwert wird. Andererseits sorgt ein breites Fenster für die Erfassung zusammenhängender langer Merkmale. Die variable Segmentierung ist für den Vergleich der Architekturen geeignet, da sie für einen Trade-off zwischen diesen Abhängigkeiten sorgen kann. Der Vorteil besteht darin, dass mehr ideale Segmente als bei einer statischen Breite produziert werden. Je mehr ideale Segmente vorhanden sind, desto mehr ideale kleindimensionale Repräsentationen werden kodiert und desto mehr ideale Cluster-Repräsentanten existieren. Mit idealen Cluster-Repräsentanten gehen potenzielle Klassen repräsentierende Zentroiden einher. Hierbei ist anzumerken, dass die Phänomene und ihre (raum-)zeitlichen Muster eine variable Länge aufweisen können und daher die Argumentation, dass eine optimale statische Breite einen besseren Trade-off liefern könnte, abgeschwächt wird.

## 1.3 Aufbau der Arbeit

In Kapitel 2 werden die technischen Grundlagen der für die Architekturen und Auswertungen genutzten Methoden erläutert. Für den Vergleich der Clusterqualitäten unter Anwendung unterschiedlicher LSTM-Architekturen werden Labels benötigt. Daher wird in Kapitel 3.1 analysiert, wie ein künstlicher Datensatz nach dem Vorbild der realen Daten und eine Provozierung der Zeitscheibenproblematik kreiert werden soll. Um sicherzustellen, dass eine optimale Transformation der realen Daten vorgenommen wird, werden diese in Kapitel 3.2 analysiert. In Kapitel 4 werden die zu untersuchenden Architekturen entworfen. Hierbei werden ausführlich die Entwurfsentscheidungen dokumentiert. In Kapitel 5 werden Implementierungsdetails präsentiert. In Kapitel 6.1 wird auf Basis der künstlichen Daten ein angemessener Wertebereich für die maximale Segmentbreite hinsichtlich Trainingsdauer, Kompressionskapazität und Clusterqualität ermittelt. Um die in Kapitel 1.2 genannte Annahme, dass eine variable zufällige Breite statt einer fixen Breite von Vorteil sein kann, zu überprüfen, wird in Kapitel 6.2 ein Experiment zum Vergleich beider Varianten durchgeführt. In Kapitel 6.3 werden die Clusterergebnisse der Architekturen auf Basis der künstlichen Daten und unter Verwendung einer in Kapitel 6.1 ermittelten angemessenen maximalen Segmentbreite miteinander verglichen. In Kapitel 6.4 werden die Ergebnisse von Kapitel 6.3 durch Anwendung auf die DWD-Daten manuell validiert.

## 2 Grundlagen

### 2.1 Convolutional Layer

Ein Convolutional-Layer lernt die Gewichte von  $N$  Filtern. Ein Filter  $f$  hat die gleiche Anzahl an Dimensionen wie die der Eingabedaten, aber für gewöhnlich kleinere Dimensionsgrößen. Er wird über die Eingabedaten geschoben, sodass alle atomaren Datenpunkte mindestens einmal erfasst werden. Für jede Abtastung wird geprüft, ob das durch den Filter repräsentierte Feature erkannt wird, und ein Maß an Aktivierung berechnet. Die Aktivierung wird in der Praxis statt mittels der Faltungsoperation ( $\star$ ) mit der Kreuzkorrelation ( $\star$ ) folgendermaßen berechnet:

$$(K \star I)(i_1, \dots, i_n) = \sum_{b_1, \dots, b_m}^{c_1, \dots, c_m} (I[i_1, \dots, i_1 + c_1 - 1; \dots; i_n, \dots, i_n + c_m - 1] \circ K)_{b_1, \dots, b_m},$$

wobei die Argumente  $i_1, \dots, i_n$  die Position des Fensters in der Eingabe  $I$  angeben,  $\circ$  das Hadamard-Produkt berechnet und  $K$  den Filter  $f$  mit den Dimensionsgrößen  $c_1, \dots, c_m$  darstellt. Die Notation  $[\dots; \dots]$  zeigt die Bildung einer Submatrix: Pro Dimension (mit einem Semikolon getrennt) werden die zu behaltenden Zeilen/Spalten referenziert. Die Indizes einer Matrix starten bei 1. (vgl. [Ian Goodfellow, 2016], Kap. 9)

Die Abtastungen ergeben zusammen eine Feature-Map  $a_f$ . Die Schicht gibt nach Anwendung aller Filter  $A_F = \{a_{f_1}, \dots, a_{f_N}\}$  aus.

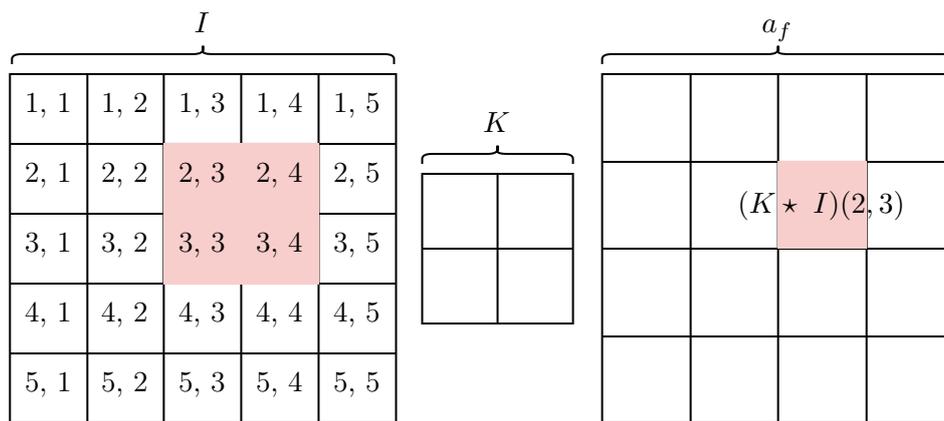


Abbildung 2.1: Feature-Map-Erzeugung für eine 2-dimensionale Eingabe

Das Konzept der Faltung weist verschiedene Eigenschaften auf, die für ein neuronales Netzwerk von Vorteil sein können. Zum einen ist die Filtermatrix normalerweise im Verhältnis zur Eingabe eher klein. Ein Neuron der Ausgabeschicht ist bei einem Kernel mit  $k$  Elementen nur mit  $k$  Neuronen verknüpft. Statt einer Laufzeit von  $\mathcal{O}(mn)$  für  $m$  Eingaben und  $n$  Ausgaben (in einer voll verknüpften Schicht) wird also lediglich eine Laufzeit von  $\mathcal{O}(kn)$  benötigt. Zum anderen werden die Gewichte des Kernels für mehrere Funktionen im Modell wiederverwendet. Dies reduziert die Anzahl an zu speichernden Parametern. Darüber hinaus ist die Faltungsoperation  $c$  äquivariant zu bestimmten Transformationen.

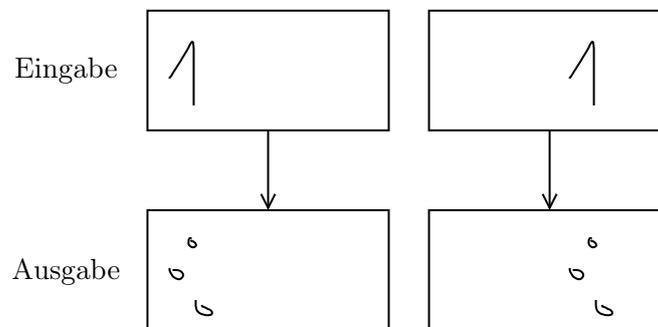


Abbildung 2.2: Äquivarianz bzgl. der Verschiebungstransformation

Sie ist dann äquivariant zu einer Transformation  $g$ , wenn  $c(g(x)) = g(c(x))$ . Dies ist zum Beispiel bei einer Verschiebung des Eingabebilds der Fall. Das kann von Vorteil sein, wenn Merkmale existieren, die an unterschiedlichen Stellen auftauchen. Die Information über ihre Position ist in der Ausgabe enthalten. (vgl. [Ian Goodfellow, 2016], Kap. 9)

## 2.2 LSTM

Die Architektur des LSTMs [Hochreiter und Schmidhuber, 1997] wurde entwickelt, um dem Problem des verschwindenden Gradienten von rekurrenten neuronalen Netzen (RNN) entgegenzuwirken. In einem herkömmlichen RNN können keine Langzeitabhängigkeiten gelernt werden, weil bei *Backpropagation Through Time* (BPTT) durch wiederholte Multiplikation von Werten unter 1 die Gradienten bei weit zurückliegenden Zeitpunkten verschwinden.

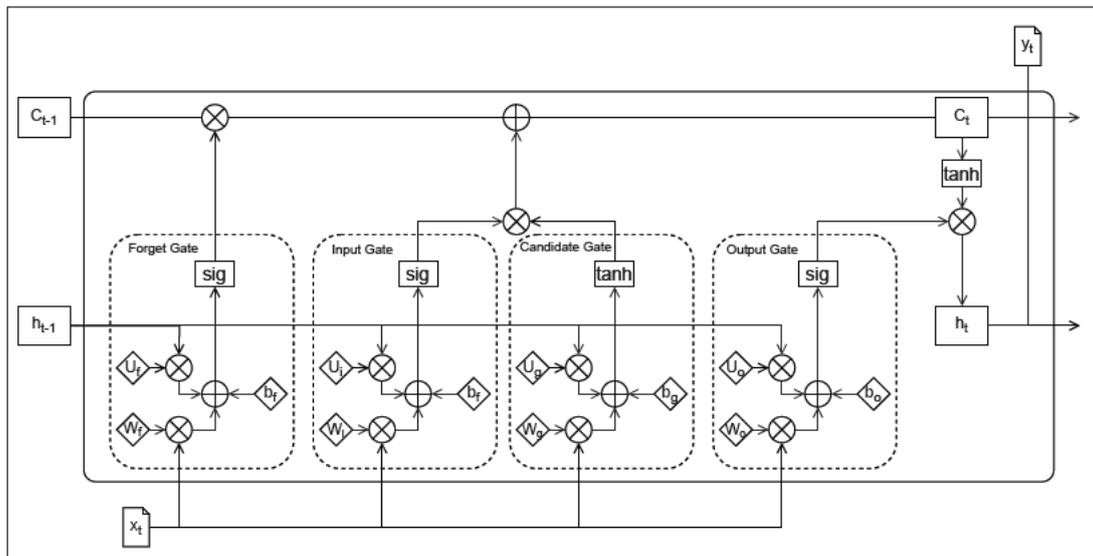


Abbildung 2.3: Struktur einer LSTM-Zelle

In Abbildung 2.3 wird eine typische Variante einer LSTM-Zelle zum Zeitpunkt  $t$  gezeigt. (vgl. [Ian Goodfellow, 2016], Kap. 10.10.1) Neben dem Hidden-State  $h$  und der Eingabe  $x$ , welche auch in einem RNN zu finden sind, besteht ein Cell-State  $c$ , der als ein Gedächtnis fungiert. Im Unterschied zum RNN wird der Hidden-State auf Basis des Cell-States berechnet und als Ausgabe  $y$  zum Zeitpunkt  $t$  ausgegeben ( $h_t = y_t$ ). Der Cell-State „verhält sich wie ein Akkumulator der Zustandsinformationen“ ([Shi u. a., 2015], Kap. 2.2). Der Zugriff auf seine Daten wird über Gates gesteuert.

A nice metaphor for these gating function is that of a differentiable if statement, where  $c$  is the condition and  $v$  is the value. If  $c$  is true ( $x \gg 0$ ),  $v$  will be passed on; if it is false ( $x \ll 0$ ), it will not. ([Bayer, 2015], Kap. 1.3.4)

Im Folgenden sind die in Abbildung 2.3 gezeigten Gates als Gleichungen formuliert. Die Sigmoid-Aktivierungsfunktionen in  $i_t$ ,  $o_t$  und  $f_t$  sorgen für eine Ausgabe im Intervall von 0 bis 1 und bilden kombiniert mit einer Multiplikation den genannten Schaltungsmechanismus.

$$\begin{aligned}d_t &= \tanh(W_d x_t + U_d h_{t-1} + b_d) \\i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f)\end{aligned}$$

Das Candidate-Gate  $d$  gibt basierend auf den neuen Informationen aus der Eingabe Änderungen aus, die dem Cell-State hinzugefügt werden können. Das Input-Gate  $i$  bestimmt, ob bzw. in welchem Maß diese potenziellen Änderungen dem Cell-State hinzugefügt werden. Das Output-Gate  $o$  bestimmt, welche Daten aus dem Cell-State in der Ausgabe des aktuellen Zeitpunktes bzw. im Hidden-State des nächsten Zeitpunktes enthalten sein sollen. Das Forget-Gate  $f$  gibt an, welche Informationen im Cell-State erhalten bleiben. In der ursprünglichen Version von [Hochreiter und Schmidhuber, 1997] bestand kein Forget-Gate. Das LSTM wurde in [Gers u. a., 1999] erweitert, weil ein kontinuierlicher Eingabestrom ein grenzenloses Wachstum der Werte im Cell-State verursachen kann. Dadurch kann der Cell-State seine Gedächtnisfunktion verlieren. Das Forget-Gate soll lernen, den Zustand im Cell-State zurückzusetzen, wenn eine gespeicherte Information nicht mehr benötigt wird. (vgl. [Gers u. a., 1999], Kap. 1) Cell-State und Hidden-State zum Zeitpunkt  $t$  werden gemäß nachfolgender Funktionen berechnet.

$$\begin{aligned}c_t &= f_t \circ c_{t-1} + i_t \circ d_t \\h_t &= o_t \circ \tanh(c_t)\end{aligned}$$

In [Gers und Schmidhuber, 2000] wurde ein Ansatz vorgestellt, mit welchem in der Mustererkennung des LSTMs der Zeitdauer zwischen Ereignissen eine höhere Berücksichtigung zukommen soll. Für das LSTM wurden Guckloch-Verbindungen (engl. *peephole connections*) eingeführt, wobei alle Gates zusätzlich eine eingehende Verbindung mit dem Cell-State als Quelle besitzen.

Im Folgenden wird durch eine Gegenüberstellung zum RNN gezeigt, wie das LSTM ein Verschwinden des Gradienten verhindert. Die Argumentation folgt der Beschreibung in Kapitel 1.3.3 und 1.3.4 von [Bayer, 2015]. Um dies aufzuzeigen, werden für die Kompo-

nennten des RNNs und LSTMs Skalare statt Matrizen verwendet.

Zunächst wird der Teil der herkömmlichen Gleichung für die Berechnung des Hidden-States in einem RNN entfernt, der für die Verarbeitung der Eingabe zuständig ist. So liegt folgende vereinfachte Formel vor:

$$h_t = \sigma(wh_{t-1})$$

Bei Anwendung von BPTT für den Fehlerrückfluss wird der Gradient durch Anwendung der Kettenregel von der Zelle mit Zeitpunkt T zur Zelle mit Zeitpunkt t zurückgegeben. Der kritische Abschnitt der Verkettung wird im Nachfolgenden beschrieben.

$$\nabla_t = \nabla_T \prod_{k=t+1}^T \frac{\partial h_k}{\partial h_{k-1}} \tag{2.1}$$

$$\begin{aligned} \frac{\partial h_t}{\partial h_{t-1}} &= \sigma'(wh_{t-1})'(wh_{t-1}) && \text{Kettenregel} \\ &= \sigma'(wh_{t-1})w && \text{(Ableitungsvariable ist 1)} \end{aligned}$$

$$\begin{aligned} \nabla_t &= \nabla_T \prod_{k=t+1}^T \sigma'(wh_{k-1})w \\ &= \nabla_T w^{T-(t+1)} \prod_{k=t+1}^T \sigma'(wh_{k-1}) && \text{Kommutativgesetz} \end{aligned} \tag{2.2}$$

Wenn  $w < 1$ , dann gilt: Je größer  $T - (t + 1)$ , desto eher wird sich 0 angenähert. Der Gradient „verschwindet“. Wenn  $w > 1$ , dann gilt: Je größer  $T - (t + 1)$ , desto eher wird sich  $\infty$  angenähert. Der Gradient „explodiert“.

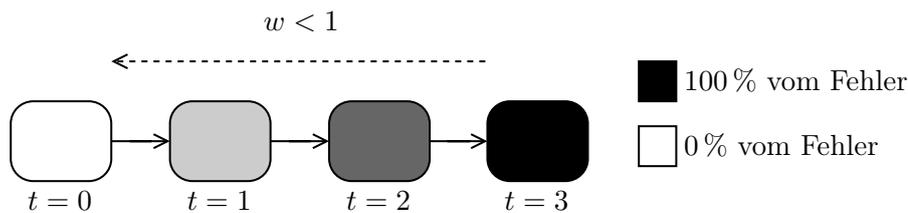


Abbildung 2.4: Verschwindender Gradient im RNN

Für die Betrachtung des entsprechenden kritischen Abschnitts bei einem LSTM wird eine ebenfalls vereinfachte Formel verwendet:

$$c_t = f_t c_{t-1}$$

Analog zur Berechnung des Gradienten zum Zeitpunkt  $t$  in einem RNN (siehe Gleichung 2.1) wird der gleiche Abschnitt des Fehlerrückflusses unter Anwendung der LSTM-Version von [Gers u. a., 1999] im Nachfolgenden beschrieben.

$$\begin{aligned} \nabla_t &= \nabla_T \prod_{k=t+1}^T \frac{\partial c_k}{\partial c_{k-1}} \\ \frac{\partial c_t}{\partial c_{t-1}} &= \frac{\partial f_t c_{t-1}}{\partial c_{t-1}} \\ &= c_{t-1} \frac{\partial f_t}{\partial c_{t-1}} + f_t \frac{\partial c_{t-1}}{\partial c_{t-1}} && \text{Multivariable Kettenregel} \\ &= c_{t-1} \frac{\partial f_t}{\partial c_{t-1}} + f_t && \text{Ableitungsvariable ist 1} \\ &= c_{t-1} 0 + f_t && \text{Fehler abschneiden (vgl. [Gers u. a., 1999], Kap. 3.2)} \\ &= f_t \\ \nabla_t &= \nabla_T \prod_{k=t+1}^T f_k \end{aligned}$$

Hierbei entfällt also der „intrinsic Faktor“ ([Bayer, 2015], Kap. 1.3.4)  $w^{T-(t+1)}$  aus Gleichung 2.2, der die Ableitung zu 0 oder  $\infty$  zieht. Im Falle eines RNNs ist für  $w \neq 1$  garantiert, dass der Gradient langfristig verschwinden oder explodieren wird. Im Falle eines LSTMs ist diese Garantie nicht gegeben. Darüber hinaus kann das Verschwinden des Gradienten durch das Forget-Gate gesteuert werden. Hierbei bietet sich an, dass die Gewichte des Forget-Gates zu Beginn des Trainings so initialisiert werden, dass ein konstanter Durchfluss des Gradienten möglich ist.

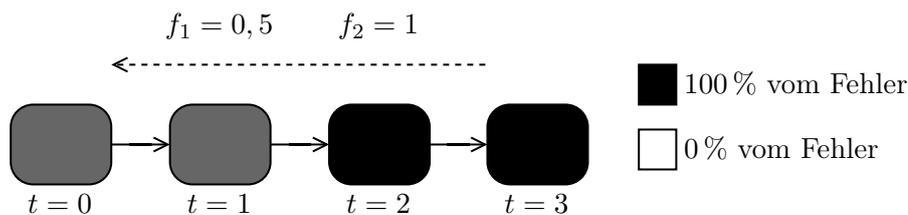


Abbildung 2.5: Verschwindender Gradient im LSTM

## 2.3 ConvLSTM

Das ConvLSTM [Shi u. a., 2015] wurde mit dem Ziel der Vorhersage von zukünftigen Regenintensitäten entwickelt. Dafür wurde ein Modell benötigt, das gut mit raumzeitlichen Datensequenzen umgehen kann. Das Modell hat eine LSTM-Version gemäß [Graves, 2014] als Grundlage.

$$\begin{aligned}
 i_t &= \sigma(W_i * x_t + U_i * h_{t-1} + V_i \circ c_{t-1} + b_i) \\
 o_t &= \sigma(W_o * x_t + U_o * h_{t-1} + V_o \circ c_t + b_o) \\
 f_t &= \sigma(W_f * x_t + U_f * h_{t-1} + V_f \circ c_{t-1} + b_f) \\
 d_t &= \tanh(W_d * x_t + U_d * h_{t-1} + b_d)
 \end{aligned} \tag{2.3}$$

Im Vergleich zur Version aus Kapitel 2.2 sind hier Gucklöcher inkludiert, wobei im Output-Gate eine Verbindung zum aktuellen Cell-State  $c_t$  statt zum alten Cell-State  $c_{t-1}$  besteht. Das ConvLSTM ersetzt alle Matrixmultiplikationen durch Faltungsoperationen (\*). Das Hadamard-Produkt wird durch  $\circ$  gekennzeichnet.

In [Shi u. a., 2015] wird als Beispiel für raumzeitliche Muster, die von dieser Architektur besser erkannt werden sollen, die Bewegung eines Objektes in Videosequenzen genannt.

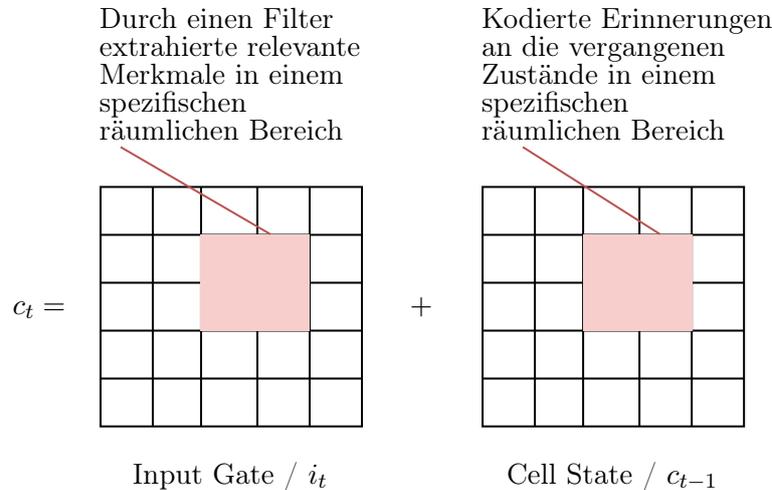


Abbildung 2.6: Feature-Map gleichende Struktur im Cell-State vom ConvLSTM

Die Formel von  $c_t$  kann simplifiziert werden, sodass sich  $c_t = c_{t-1} + i_t$  ergibt, wobei  $i_t = W_i x_t$ . In Abbildung 2.6 wird die Formel visualisiert, um erkennbar zu machen,

dass die Speicherung der Eingabeinformationen erzwungenermaßen positionsorientiert erfolgt. Erinnerungswürdige Merkmale (wie z.B. die diagonale Bewegung eines Balls), die an unterschiedlichen räumlichen Positionen auftauchen können, werden über Filter extrahiert. Hierbei ist auf die in Kapitel 2.1 genannten Vorteile hinzuweisen. Auf diese Weise erhält der Cell-State eine Struktur, die der einer Feature-Map gleicht. Der Formel 2.3 ist zu entnehmen, dass die Ausgabe ebenfalls eine solche Struktur aufweist.

### 2.4 Autoencoder

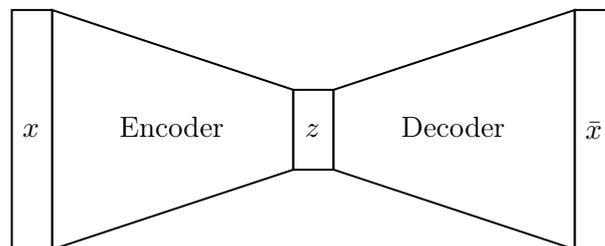


Abbildung 2.7: Struktur eines Autoencoders

Ein Autoencoder ist ein neuronales Netz, welches Eingangsdaten  $x$  mithilfe einer Funktion  $encode(x)$  in eine Repräsentation  $z_x$  überführt und mithilfe einer Funktion  $\bar{x} = decode(z_x)$  rekonstruiert. Die Funktionsparameter werden durch ein Training mit einer  $x$  mit  $\bar{x}$  vergleichenden Verlustfunktion gelernt. Üblicherweise werden Autoencoder zur Dimensionsreduktion und Merkmalsextraktion verwendet. Durch eine schmale Dimension der Schicht  $z$ , die Encoder und Decoder verbindet, kann das Netz dazu gezwungen werden, lediglich die für die Rekonstruktion relevantesten Aspekte der Eingabedaten zu erkennen und diese Informationen in  $z$  abzubilden. Ein Autoencoder mit solch einem Flaschenhals wird unternvollständig genannt.

## 2.5 VAE

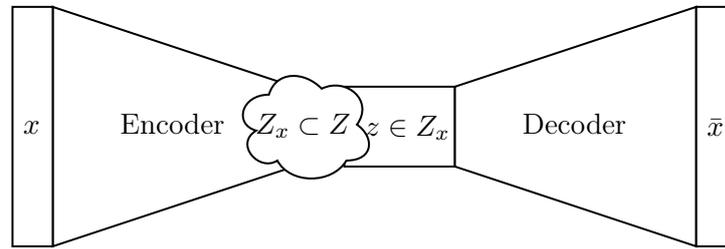


Abbildung 2.8: Struktur eines Variational Autoencoders

Ein Variational Autoencoder (VAE) [Kingma und Welling, 2014] ist ein generatives Modell. Eine konkrete Umsetzung dessen entspricht einem Autoencoder, der schmale Repräsentationen  $H_i$  lernt, deren Merkmalsausprägungen  $h_1..h_n$  die Parameter einer mehrdimensionalen Wahrscheinlichkeitsverteilung  $P_{H_i}$  darstellen. Der Decoder nimmt eine Stichprobe  $z$  von  $P_{H_i}$  entgegen, um die Eingangsdaten zu rekonstruieren. Abbildung 2.8 zeigt auf, dass der Encoder bei gegebener Eingabe  $x$  statt eines einzigen Werts für die latente Repräsentation  $z$  eine Menge von möglichen Werten vorhersagt ( $Z_x$ ). Die Menge wird durch  $P_{H_i}$  beschrieben.

Die Verlustfunktion ist bei Wahl der Normalverteilung für  $P$  gemäß [Kingma und Welling, 2014] gegeben durch:

$$L(x, \bar{x}, (\mu_1, \dots, \mu_k), (\sigma_1^2, \dots, \sigma_k^2)) \\ = L_{Rekonstruktion}(x, \bar{x}) + D_{KL}(\mathcal{N}((\mu_1, \dots, \mu_k), \text{diag}(\sigma_1^2, \dots, \sigma_k^2)) \parallel \mathcal{N}(0, I)), \quad (2.4)$$

wobei

- $x$ : die Eingabe ist.
- $\bar{x}$ : die Ausgabe ist.
- $(\mu_1, \dots, \mu_k)$ : Parameter  $(h_1, \dots, h_k)$  für den Erwartungswert sind.
- $(\sigma_1^2, \dots, \sigma_k^2)$ : Parameter  $(h_{k+1}, \dots, h_n)$  für die Varianz sind.
- $\mathcal{N}(0, I)$ : die Wahrscheinlichkeitsverteilung  $P(Z)$  ist.

Der linke Term in der Verlustfunktion berechnet den Rekonstruktionsfehlerwert und sorgt dafür, dass der Encoder repräsentative Merkmale extrahiert. Der rechte Term stellt eine Regularisierung dar, welche durch eine hohe KL-Divergenz bestraft, wenn sich  $P_{H_i}$  zu sehr von einer deklarierten „Code-Verteilung“ ([Ian Goodfellow, 2016], Kap. 20.10.3) entfernt. Eine Stichprobenentnahme von der Code-Verteilung soll eine mögliche Wertezuweisung der latenten Dimension  $z$  generieren, die von dem Decoder in eine den

Daten im Datensatz ähnelnde Ausgabe dekodiert werden kann. Als Code-Verteilung wird in [Kingma und Welling, 2014]  $\mathcal{N}(0, I)$  mit  $I$  als Identitätsmatrix verwendet. Die Verlustfunktion ergibt sich aus einer im Folgenden auf Basis von [Doersch, 2021] erläuterten Problemstellung, die aus Sicht der Bayesschen Statistik gestellt wird. Das Ziel ist es bei generativen Modellen wie VAE, die unbekannte Verteilung  $P_{data}$ , die die Daten  $X = x_1, \dots, x_N$  (Trainings- und Testdaten) generiert hat, mit einer Verteilung  $P_{model}$  zu modellieren, sodass Daten  $\bar{x} \sim P_{model}$  generiert werden können, die den Daten von  $X$  ähneln. (vgl. [Doersch, 2021], Kap. 1)

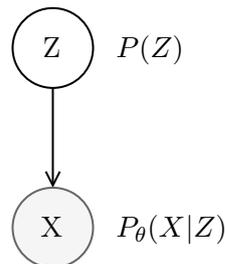


Abbildung 2.9: Graphisches Modell ohne Variational-Inference

Weil die Komplexität der Verteilung  $P(X)$  in der Praxis zu groß ist, als dass eine Annäherung möglich ist, werden latente Variablen  $Z$ , welche eine Abstraktion der beobachtbaren Variablen  $X$  darstellen, mit einer voraussichtlich weniger komplexen Verteilung  $P(Z)$  eingeführt. Dadurch liegt ein gerichtetes graphisches Modell mit einem Knoten der Zufallsvariablen  $Z$ , der auf den Knoten der Zufallsvariablen  $X$  zeigt, vor. Die Transformation von  $Z$  auf  $X$  erfolgt über eine Funktion  $P(X|Z)$  mit den Parametern  $\theta$ . Die Rand-Wahrscheinlichkeitsverteilung  $P(X)$  kann nun durch Anwendung der Produktregel der bedingten Wahrscheinlichkeiten auf die multivariate Wahrscheinlichkeitsverteilung  $P(X, Z)$  folgendermaßen formuliert werden:

$$P(X) = \int P(X|z; \theta)P(z) dz$$

Ziel ist die Maximierung der Log-Likelihood von  $P(X)$  über alle  $X$ . Wenn die summierte Log-Likelihood unter Stichprobenentnahme von  $P(z)$  maximal ist, dann liegt eine  $P_{data}$  optimal abbildende Verteilung  $P_{model}$  vor (vgl. [Ian Goodfellow, 2016], Kap. 5.5). Die Berechnung von optimalen Parametern  $\theta$  nimmt allerdings aufgrund der großen Anzahl an als Stichproben zu entnehmenden Werten für  $Z$  bzw. aufgrund des hochdimensionalen Integrals eine exponentielle Zeitkomplexität an (vgl. [Doersch, 2021], Kap. 2 und [Blei

u. a., 2017], Kap. 2.1). Diese Problematik kann dem Fluch der Dimensionalität zugeschrieben werden (siehe auch [Ian Goodfellow, 2016], Kap. 5.11.1). Daher wird der Raum von  $Z$  auf solche Werte eingeschränkt, die mit hoher Wahrscheinlichkeit  $X$  erzeugt haben. Dafür wird eine Funktion  $P(z|X)$  benötigt, die eine Verteilung über die Werte von  $Z$  liefert, welche mit hoher Wahrscheinlichkeit  $X$  produziert haben.

Diese Funktion kann mittels *Variational Inference* annähernd berechnet werden. Hierbei wird davon ausgegangen, dass eine günstige Deklaration der Verteilung  $P(Z)$  zur Modellierung ausreicht (siehe A-priori-Wahrscheinlichkeitsverteilung). Die wahre A-posteriori-Wahrscheinlichkeitsverteilung  $p(z|x)$  wird daraufhin mittels Optimierung eines Stellvertreters  $q(z|x)$  geschätzt, wobei  $q(z|x)$  und die A-priori-Wahrscheinlichkeitsverteilung  $p(z)$  einer definierten parametrisierten Wahrscheinlichkeitsdichtefunktion (z.B. Gauß-Funktion) entsprechen. (vgl. [Blei u. a., 2017], Kap. 2) Von dieser Funktion können dann Stichproben entnommen werden.

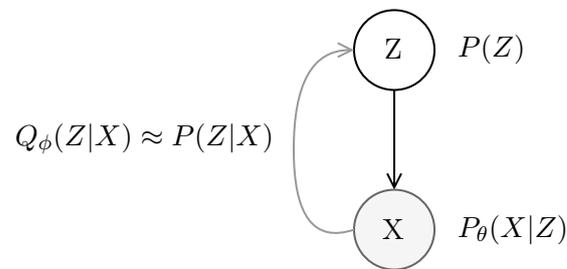


Abbildung 2.10: Graphisches Modell mit Variational-Inference

Das Ziel ist also nun, eine Verteilung  $Q(z|X)$  bzw.  $Q(z)$  zu finden. Dafür müssen die Parameter (sowohl  $\theta$  als auch  $\phi$ ) des Modells so trainiert werden, dass sich  $Q(z|x)$   $P(z|x)$  annähert:

$$\min D_{KL}(q(z|x) || p(z|x))$$

Durch Umformung ergibt sich für die KL-Divergenz:

$$\begin{aligned}
 & D_{KL}(q(z|x) \parallel p(z|x)) \\
 &= \mathbb{E}_{z \sim q(z|x)}[\log q(z|x) - \log p(z|x)] && \text{Definition } D_{KL}(P \parallel Q) \\
 &= \mathbb{E}_{z \sim q(z|x)}[\log q(z|x) - \log(p(z, x)/p(x))] && \text{Satz von Bayes} \\
 &= \mathbb{E}_{z \sim q(z|x)}[\log q(z|x) - (\log p(z, x) - \log p(x))] && \text{Log-Quotientregel} \\
 &= \mathbb{E}_{z \sim q(z|x)}[\log q(z|x) - \log p(z, x) + \log p(x)] && \text{Distributivgesetz} \\
 &= \mathbb{E}_{z \sim q(z|x)}[\log q(z|x) - \log p(z, x)] + \log p(x) && \text{A.1} \\
 &= \mathbb{E}_{z \sim q(z|x)}[\log q(z|x)/p(z, x)] + \log p(x) && \text{Log-Quotientregel} \\
 &= -\mathbb{E}_{z \sim q(z|x)}[\log p(z, x)/q(z|x)] + \log p(x) && \text{A.2}
 \end{aligned}$$

Der Erwartungswert ( $\mathbb{E}[X]$ ) stellt den sogenannten ELBO (*Evidence Lower Bound*) Term dar. Im Folgenden wurde nach  $\log p(x)$  aufgelöst.

$$\begin{aligned}
 \iff \log p(x) &= \mathbb{E}_{z \sim q(z|x)}[\log p(z, x)/q(z|x)] + D_{KL}(q(z|x) \parallel p(z|x)) \\
 &\geq \mathbb{E}_{z \sim q(z|x)}[\log p(z, x)/q(z|x)] && D_{KL} \geq 0
 \end{aligned}$$

Hierbei gilt, dass  $\log p(x)$  eine Konstante ist, die sich während der Optimierung des Stellvertreters  $q(z|x)$  niemals ändert. Die gezeigte Formel impliziert also, dass, wenn der ELBO maximiert wird, die KL-Divergenz minimiert wird. Somit kann die Maximierung vom ELBO als neues Ziel erhalten. Dies ist von Vorteil, da für die Berechnung des ELBO Terms sämtliche notwendige Komponenten zur Verfügung stehen.

*ELBO*

$$\begin{aligned}
 &= \mathbb{E}_{z \sim q(z|x)}[\log p(z, x)/q(z|x)] \\
 &= \mathbb{E}_{z \sim q(z|x)}[\log p(x|z)p(z)/q(z|x)] \\
 &= \mathbb{E}_{z \sim q(z|x)}[\log p(x|z) + \log p(z)/q(z|x)] && \text{Log-Produktregel} \\
 &= \mathbb{E}_{z \sim q(z|x)}[\log p(x|z)] + \mathbb{E}_{z \sim q(z|x)}[\log p(z)/q(z|x)] && \text{Definition } \mathbb{E}[X] \\
 &= \mathbb{E}_{z \sim q(z|x)}[\log p(x|z)] - \mathbb{E}_{z \sim q(z|x)}[\log q(z|x)/p(z)] && \text{A.2} \\
 &= \mathbb{E}_{z \sim q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x) \parallel p(z)) && \text{Definition } D_{KL}(P \parallel Q)
 \end{aligned}$$

Um den Wert zu maximieren, sollte also  $\log p(x|z)$  möglichst groß und die KL-Divergenz zwischen  $q(z|x)$  und  $p(z)$  möglichst klein sein. Daher ist das Ziel der Maximierung der Log-Likelihood und der Minimierung der KL-Divergenz zwischen der A-posteriori-

Wahrscheinlichkeitsverteilung  $q(z|x)$  und der A-priori-Wahrscheinlichkeitsverteilung  $p(z)$  gegeben. Dieses Ziel kann als Verlustfunktion (siehe Gleichung 2.4) verwendet werden, um optimale Parameter  $\theta$  und  $\phi$  zu lernen.

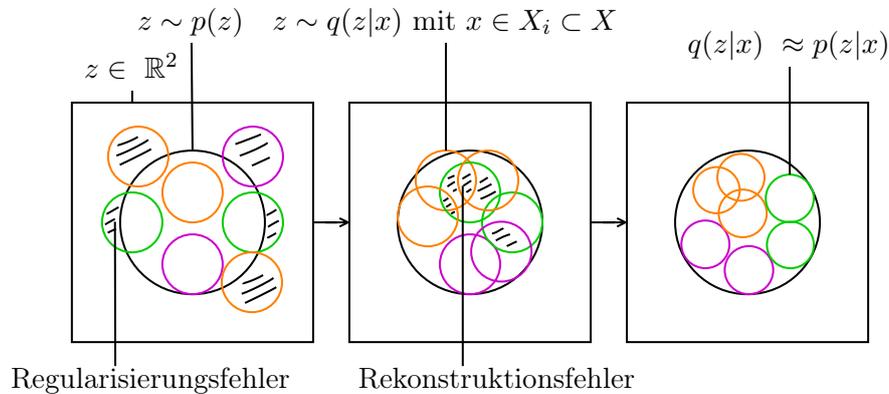


Abbildung 2.11: Erzwungene Ordnung im latenten Raum des VAEs

In Abbildung 2.11 wurden die Daten  $X$  zur Visualisierung in 3 Mengen aufgeteilt ( $X_i$ ), wobei die Elemente derselben Menge ähnliche und die Elemente unterschiedlicher Mengen unähnliche Merkmale aufweisen. Die Abbildung zeigt auf, wie während des Trainings vom VAE durch die Verlustfunktion eine Ordnung im latenten Raum geschaffen wird.

## 2.6 Clustering

Clustering ist eine Methode der unüberwachten Gruppierung von Objekten, bei der versucht wird, Gruppen zu finden, die jeweils möglichst Objekte mit ähnlichen Eigenschaften beinhalten und im Vergleich ihrer Objekte mit Objekten anderer Gruppen möglichst unterschiedliche Eigenschaften aufweisen. (vgl. [Guojun Gan, 2007], Kap. 1.1)

Clustering Probleme können in Hard-Clustering und Soft-Clustering unterteilt werden. Beim Hard-Clustering wird ein Datenpunkt lediglich einem Cluster zugeordnet. Beim Soft-Clustering wird dem Datenpunkt für jedes Cluster eine Zugehörigkeitswahrscheinlichkeit berechnet, sodass der Datenpunkt mehreren Clustern zugewiesen werden kann. (vgl. [Guojun Gan, 2007], Kap. 1.2.4) Die Datenpunkte können nach einem Soft-Clustering jeweils einem einzigen Cluster „hart“ zugeordnet werden, indem das Cluster mit der höchsten Zugehörigkeitswahrscheinlichkeit ausgewählt wird (vgl. [Guojun Gan, 2007], Kap. 8).

Um die Qualität des Clusterings auszuwerten, existieren drei allgemeine Kategorien: Externe Kriterien, interne Kriterien und relative Kriterien. Bei externen Kriterien wird eine intuitive Partition des Datensatzes gebildet und mit der tatsächlichen Partition verglichen, die durch das Clustering entsteht. (vgl. [Guojun Gan, 2007], Kap. 17.1.2) Bei internen Kriterien wird das Clustering Ergebnis nur auf Basis der Eigenschaften der Partition und der Eingangsdaten bewertet. Beliebte Maße sind Kompaktheit innerhalb der Cluster und Separation der Cluster untereinander (vgl. [Hassani und Seidl, 2017], Kap. 2.1). Diese validieren das primäre Ziel des Clusterings, ähnliche Datenpunkte zu gleichen Gruppen und unähnliche zu unterschiedlichen zuzuweisen. Bei relativen Kriterien werden die Clusterstrukturen von mehrfacher Ausführung eines Clustering Algorithmus mit unterschiedlichen Parametern anhand einer zuvor festgelegten Validitätskennzahlberechnung verglichen. (vgl. [Guojun Gan, 2007], Kap. 17.1.4) Auf diese Weise können möglichst optimale Parameterwerte ermittelt werden.

## 2.7 Deep Clustering

Clustering-Verfahren zeigen eher schwache Leistungen auf hochdimensionalen Datenpunkten (vgl. [Min u. a., 2018], Kap. 1), weswegen ein Verfahren zur Dimensionsreduktion benötigt wird. Autoencoder eignen sich insbesondere dazu, aus hochdimensionalen Daten für die Rekonstruktion wichtige Merkmale zu extrahieren und diese in eine komprimierte Repräsentation zu überführen. Methoden unter dem Schlagwort Deep-Clustering (DC) wurden entwickelt, um gleichzeitig eine Cluster-freundliche Repräsentation zu gewährleisten. In Tabelle 4 von [Min u. a., 2018] werden die für Autoencoder anwendbaren Methoden folgendermaßen kategorisiert:

Kategorie	Beschreibung
CDNN-bezogen	Optimieren des Netzwerks durch einen Clusteringfehlerwert (ohne Netzwerkfehlerwert)
AE-bezogen	Gleichzeitiges Optimieren eines Autoencoders (durch einen Rekonstruktionsfehlerwert) und von Clustering-Parametern (durch einen Clusteringfehlerwert)
VAE-bezogen	Inkludieren eines gaußschen Mischmodells (GMM) als Wahrscheinlichkeitsverteilung für den latenten Raum eines VAEs

Tabelle 2.1: Deep-Clustering-Kategorien

Zur Klasse der CDNN-bezogenen <sup>1</sup> Deep-Clustering-Methoden zählt u.a. *Deep Embedded Clustering* (DEC) [Xie u. a., 2016]. Im DEC-Verfahren wird ein Autoencoder vortrainiert. Danach wird der Decoder verworfen und durch eine Clusteringschicht ersetzt. In einem nachfolgenden Training wird eine initiale Clusterzuordnung verfeinert bzw. optimiert, wobei ein Clusteringfehlerwert von der Ausgabe der Clusteringschicht bis zur Eingabe zurückpropagiert wird. (vgl. [Xie u. a., 2016]) In Kapitel 2.8 wird das Verfahren im Detail beschrieben.

Zur Klasse der AE-bezogenen Deep-Clustering-Methoden zählt u.a. *Improved Deep Embedded Clustering* (IDEC) [Guo u. a., 2017]. Das IDEC-Verfahren erweitert DEC insofern, dass der Decoder nach dem Vortraining nicht verworfen wird. Stattdessen besteht eine Clusteringschicht parallel zum Decoder. Sowohl ein Rekonstruktionsfehler als auch ein Clusteringfehlerwert werden also gleichzeitig bis zur ersten Schicht zurückpropagiert. Dies hat das Ziel, lokale Strukturen - also z.B. die Korrelation zweier Features aufeinanderfolgender Schichten - möglichst zu erhalten, wenn diese eine besondere Relevanz zur Rekonstruktionsfähigkeit beitragen. In DEC dagegen besteht das Risiko, dass durch eine Anpassung des Netzwerks durch den Clusteringfehlerwert Merkmale bzw. Informationen verloren gehen, die für ein semantisch sinnvolles Clustering entscheidend sind. (vgl. [Guo u. a., 2017])

Zur Klasse der VAE-bezogenen Deep-Clustering-Methoden zählt u.a. *Variational Deep Embedding* (VaDE) [Jiang u. a., 2017]. In diesem Verfahren wird die multivariate gaußsche Verteilung eines VAEs, welche die latente Repräsentation generiert, durch ein GMM ersetzt. Die Initialisierung der Parameter des GMMs erfolgt nach einem Vortraining durch einen Autoencoder ohne Varianz auf Basis der durch den Autoencoder kodierten Repräsentationen. (vgl. [Jiang u. a., 2017])

## 2.8 DEC

Das DEC [Xie u. a., 2016] Verfahren besteht aus zwei Phasen. In der ersten Phase wird ein Autoencoder trainiert. Danach wird der Decoder verworfen und durch eine Clusteringschicht ersetzt. Für jedes Cluster werden z.B. durch Anwendung von K-Means initiale Zentroiden gebildet. In der zweiten Phase wird abwechselnd eine Ziel-Verteilung berechnet und die KL-Divergenz zu ihr per Fehlerrückfluss minimiert.

---

<sup>1</sup>CDNN ist eine Abkürzung für *Clustering Deep Neural Network*

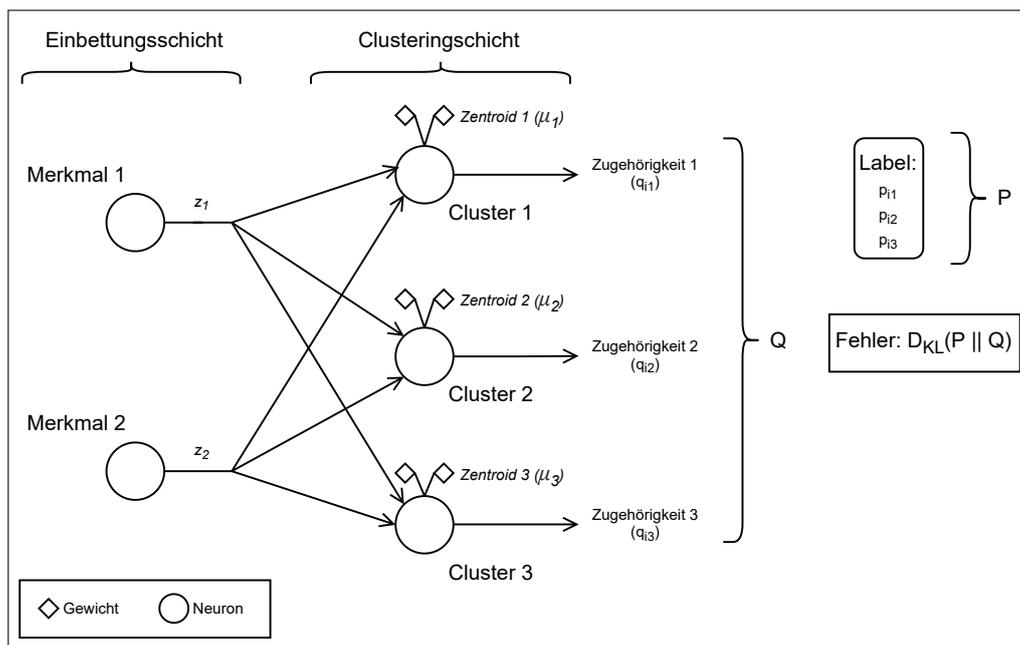


Abbildung 2.12: Exemplarisches Netzwerk für DEC in Phase 2

In Abbildung 2.12 wird der Flaschenhals des Autoencoders mit der Einbettungsschicht beschrieben, welche die extrahierten Merkmale der Eingabe  $x_i$  ausgibt. Die Clusteringsschicht umfasst Cluster repräsentierende Neuronen mit jeweils einem Zentroiden. Ein Neuron in der Clusteringsschicht ist verknüpft mit einem durch den Gradientenabstieg anpassbaren Zentroiden. Es empfängt die gesamte Ausgabe der Einbettungsschicht und gibt die Wahrscheinlichkeit der Zugehörigkeit zu einem Cluster aus. Die Ausgabe der Clusteringsschicht entspricht also einer Wahrscheinlichkeitsverteilung.

Die Zugehörigkeitswahrscheinlichkeit wird nach einem in Kapitel 3.3 von [van der Maaten und Hinton, 2008] beschriebenen Verfahren berechnet:

$$q_{ij} = \frac{(1 + \|\mathbf{z}_i - \boldsymbol{\mu}_j\|^2)^{-1}}{\sum_{j'} ((1 + \|\mathbf{z}_i - \boldsymbol{\mu}_{j'}\|^2)^{-1})}$$

wobei  $i$  eine Eingabe aus  $X$  und  $j$  ein Cluster aus  $C$  indiziert. Die Ähnlichkeit zwischen zwei Datenpunkten wird definiert als Wahrscheinlichkeit, dass der eine Datenpunkt  $z_i$  (*Embedded Point*) den anderen Datenpunkt  $\mu_j$  (Cluster-Zentroid) als seinen räumlich nächsten Nachbarn wählen würde (vgl. [van der Maaten und Hinton, 2008], Kap. 2). Die räumliche Nähe zwischen zwei Datenpunkten wird mit der euklidischen Distanz berechnet. Der berechnete Wert wird in die Wahrscheinlichkeitsdichtefunktion der studentschen

t-Verteilung mit einem Freiheitsgrad  $\alpha = 1$  gegeben, sodass die Nähe in eine Zugehörigkeitswahrscheinlichkeit transformiert wird. Das Resultat wird im Anschluss ins Verhältnis gesetzt zu der Grundmenge, die berechnet wird, indem die Wahrscheinlichkeiten der Nähe zu den jeweiligen Cluster-Zentroiden summiert werden. Dies hat zur Folge, dass  $Q$  eine normalisierte Wahrscheinlichkeitsfunktion für eine diskrete Zufallsvariable  $C$  mit dem Wert  $j \in \{1, \dots, N\}$  darstellt (vgl. [Ian Goodfellow, 2016], Kap. 3.3.1).

Als Verlustfunktion wird die KL-Divergenz zwischen  $Q$  und  $P$  genutzt.  $P$  ist hierbei die zuvor berechnete Zielverteilung. Ein Label für die Eingabe  $x_i$  repräsentiert  $P(C|x_i)$ .  $P(c_j|x_i)$  wird gemäß folgender Formel ermittelt und als Ziel-Zugehörigkeitswahrscheinlichkeit eines Labels der jeweiligen Eingabe aktualisiert:

$$p_{ij} = \frac{\frac{q_{ij}^2}{\sum_{i'} q_{i'j}}}{\sum_{j'} \left( \frac{q_{ij'}^2}{\sum_{i'} q_{i'j'}} \right)}$$

Zur Berechnung einer Ziel-Zugehörigkeitswahrscheinlichkeit wird im oberen Term die Zugehörigkeitswahrscheinlichkeit quadriert und ins Verhältnis gesetzt zur summierten Menge an Zugehörigkeitswahrscheinlichkeiten zum betrachteten Cluster. Die summierte Menge wird im Folgenden als Clusterdichte beschrieben. Das Quadrieren der aktuellen Zugehörigkeit hat zur Folge, dass Werte von ca. 30% bis 60% am stärksten abgeschwächt werden (siehe Abbildung 2.13).

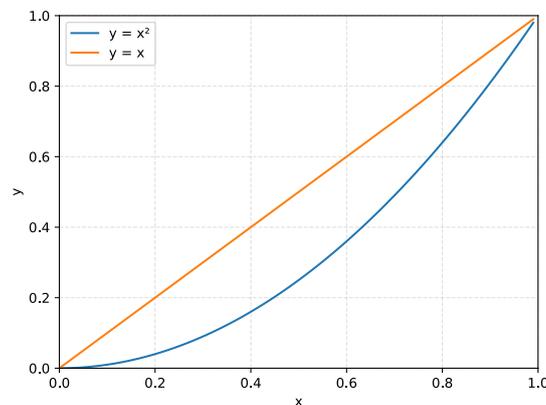


Abbildung 2.13: Funktion  $x^2$  im Intervall  $[0, 1]$

Hierdurch wird also das Ziel einer erhöhten Clusterreinheit signalisiert. Die Auswirkungen des Teilens durch die Clusterdichte kann durch folgende Logik beschrieben werden:

- Hohe Zugehörigkeit  $\wedge$  niedrige Clusterdichte  $\implies$  Zugehörigkeit stärken
- Niedrige Zugehörigkeit  $\wedge$  hohe Clusterdichte  $\implies$  Zugehörigkeit schwächen

Das Ergebnis wird wie zuvor bei der Berechnung der Zugehörigkeitswahrscheinlichkeit normalisiert.

## 3 Analyse

In diesem Kapitel werden eine Anforderungsanalyse für die Generierung künstlicher Daten und eine Analyse der DWD-Daten vorgenommen.

### 3.1 Künstliche Daten

Weil keine Labels zu den Wetterdienstdaten vorliegen und Labels für eine automatisierte Clusterbewertung wie z.B. der Zuordnung von Klassen zu Clustern zwingend notwendig sind, werden künstliche Daten erzeugt.

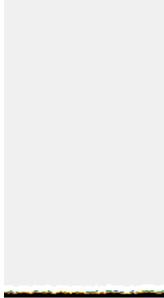
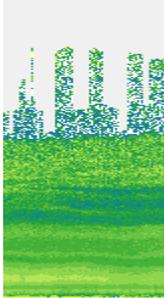
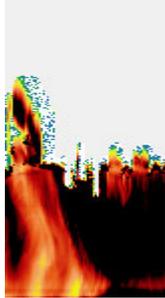
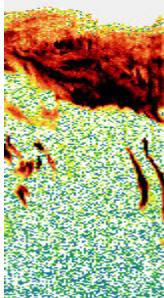
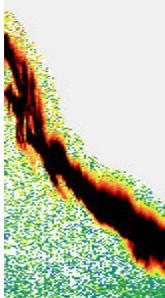
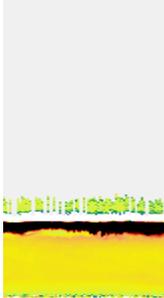
Hintergrund	Nebel	Grenzschicht	Fertransport
			
Niederschlag	Cirruswolke	Schauerwolke	Wasserwolke
			

Tabelle 3.1: Repräsentanten interessanter Phänomene

Zuerst werden Zeitabschnitte aus den realen Daten ermittelt, die relevante Phänomene repräsentieren. Unter Betrachtung der Visualisierung des zweidimensionalen Datensatzes haben die Phänomene differenzierbare Eigenschaften. Eine saubere Atmosphäre („Hintergrund“) zeichnet sich durch das Fehlen von anderen Phänomenen aus. Eine Cirruswolke befindet sich eher in oberen Höhenbereichen. Eine Schauerwolke ist durch eine V-förmige Struktur erkennbar. Sowohl Nebel als auch eine Wasserwolke zeichnen sich durch einen schmalen und horizontal linienförmigen Verlauf aus. Nebel unterscheidet sich primär von der Wasserwolke durch die Positionierung im Höhenbereich. Niederschlag ist durch einen vertikalen Schleier erkennbar. Aerosol der planetaren Grenzschicht wird durch gelbe Flächen indiziert, während Ferntransport-Aerosol hellgrün bis dunkelgrüne Flächen aufweist. Die enthaltenen Muster treten wiederkehrend auf, sodass das Finden eines jeweiligen Clusters möglich ist.

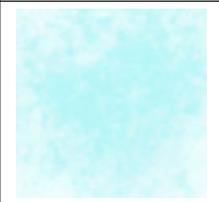
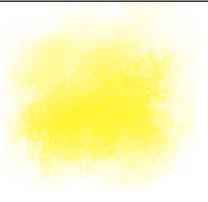
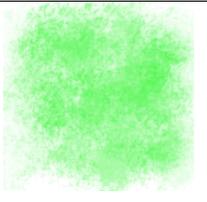
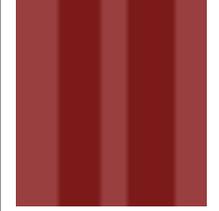
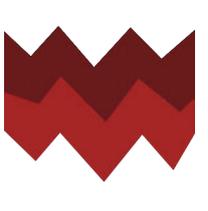
Hintergrund	Nebel	Grenzschicht	Ferntransport
			
Niederschlag	Cirruswolke	Schauerwolke	Wasserwolke
			

Tabelle 3.2: Künstliche RGB-Bilder als Repräsentanten interessanter Phänomene

Die ermittelten Repräsentanten der Wetterphänomene sollen durch simple RGB-Bilder repräsentiert werden. Jedes Bild hat eindeutige Merkmale. Dafür nimmt jedes zu differenzierende Bild tendenziell einen eigenen Farbraum ein und enthält eigene Formen. Mit den in Tabelle 3.2 dargestellten Bildern wird die in Kapitel 1 beschriebene Zeitscheibenproblematik nicht explizit mittels temporaler Muster provoziert. Eine kleinteilige Segmentierung eines einzigen Bildes weist hierbei Segmente mit starken Ähnlichkeiten auf.

a) Cirruswolke	b) Schauerwolke
	

Tabelle 3.3: Provozierung der Zeitscheibenproblematik

Um (raum-)temporale Abhängigkeiten zu inkludieren, kann wie in Tabelle 3.3, a) ein Farbverlauf hinzugefügt werden. Eine kleinteilige Segmentierung würde den Farbverlauf abschneiden. Die Reihenfolge bzw. Abhängigkeit würde vom neuronalen Netz daher nicht erfasst werden können. Die Segmente können fälschlicherweise unterschiedlichen Clustern zugeordnet werden. Um die Differenzierung zwischen zwei ähnlichen Mustern zu erschweren, kann wie in Tabelle 3.3, b) zwischen der Schauerwolke und der Wasserwolke eine räumliche Verzerrung über einen größeren Zeitabschnitt hinweg hinzugefügt werden. Hierbei würde eine kleinteilige Segmentierung von Wasserwolke und Schauerwolke ähnliche Segmente aufweisen. Die Segmente können daher fälschlicherweise gleichen Clustern zugeordnet werden.

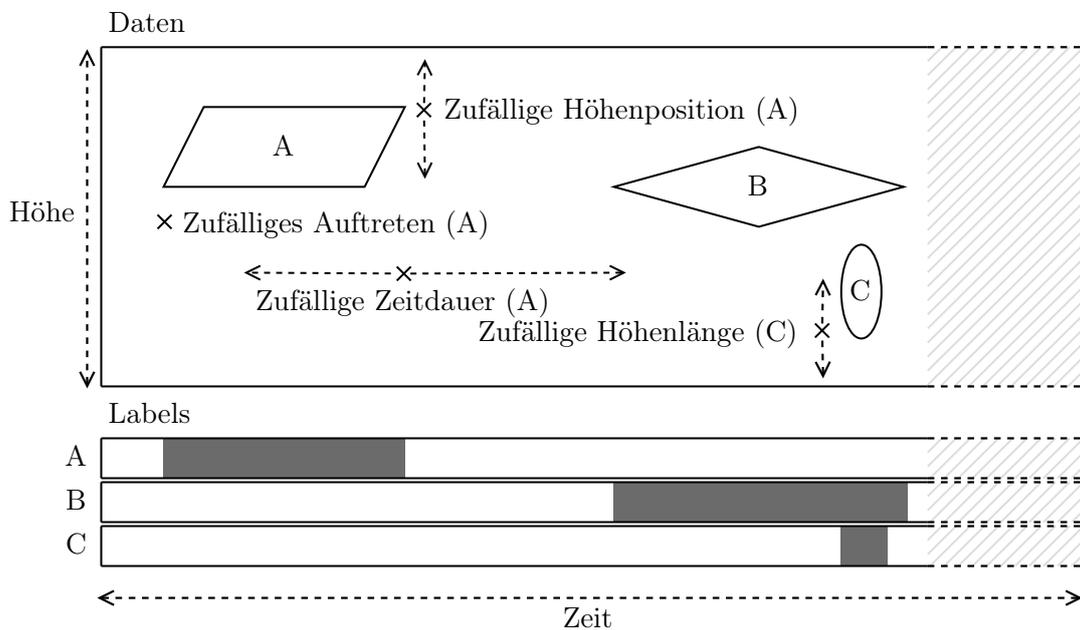


Abbildung 3.1: Exemplarische Generierung künstlicher Daten

Jedes dem jeweiligen Bild in Tabelle 3.2 entsprechende Phänomen (in Abbildung 3.1 durch A, B und C repräsentiert) hat spezifische Eigenschaften wie einen bestimmten Höhenbereich, eine Höhenlänge, eine typische Häufigkeit und Dauer seines Auftretens. Weil Phänomene im realen Datensatz von sehr variabler Dauer sein können, sollten hierfür breite Reichweiten für die Zufallswertgenerierung gewählt werden. Für ein Bild sollen Datenaugmentationsfunktionen mit zufallsbasierten Parametern angegeben werden können. Darüber hinaus soll ein Phänomen an ein anderes gekoppelt werden (z.B. Wasserwolke und Niederschlag) oder umgekehrt das andere ausschließen können (z.B. Wasserwolke und Schauerwolke). Für jeden Zeitpunkt wird mit einem Wahrheitswert hinterlegt, ob das Phänomen vorhanden ist oder nicht (in Abbildung 3.1 im Bereich „Labels“ durch jeweils graue oder weiße Farbe dargestellt).

### 3.2 Reale Daten

Es liegen Daten eines Ceilometers in Leipzig aus dem Jahr 2018 vor, welche im Folgenden mit  $X_{Leipzig18}$  bezeichnet werden. Sowohl ein gegebener CNN-Autoencoder als auch ein LSTM-Autoencoder tendieren dazu, für unterschiedliche Eingaben die gleiche Ausgabe und Repräsentation auszugeben, wenn sie mit den auf Basis von Min-Max-skalierten  $X_{Leipzig18}$ -Daten trainiert werden. Bei einer Standardisierung („z-Score“), welche dafür sorgt, dass die Daten einen Mittelwert von 0 und eine Standardabweichung von 1 haben, besteht das Problem zwar nicht, aber die Rekonstruktion weist starke Mängel auf.

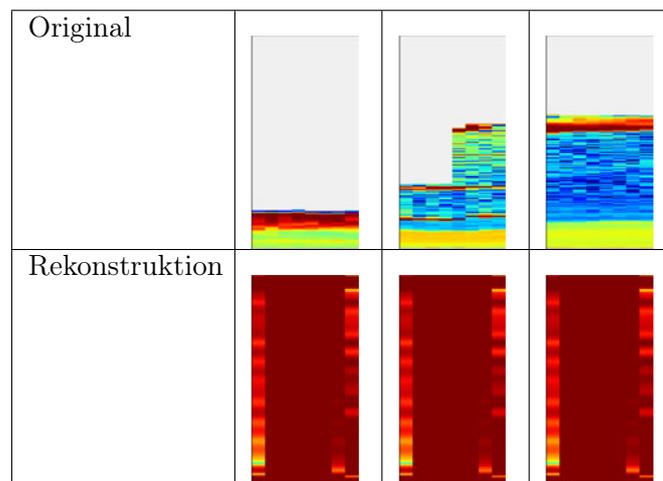


Tabelle 3.4: Fehlschlagende Rekonstruktion mit Min-Max-Skalierung

Der Fehlschlag, die Eingabe zu rekonstruieren, kann seine Ursache darin haben, dass die Skalierung Ausreißern eine zu hohe Bedeutung zukommen lässt. Eine andere Erklärung kann sein, dass die in  $X_{Leipzig18}$  ermittelten Min- und Max-Werte unangemessen sind, weil mit  $X_{Leipzig18}$  nur ein verhältnismäßig geringer Teil der durch den deutschen Wetterdienst erhobenen Gesamtdaten vorliegt.

Um sicherzustellen, dass eine optimale Datentransformation genutzt wird, wird der kalibrierte und bereinigte Datensatz untersucht.

Negative Werte		Unbekannte Werte	
Absolute Anzahl	Prozentualer Anteil	Absolute Anzahl	Prozentualer Anteil
21 139 129	4,22 %	358 338 937	71,68 %

Tabelle 3.5: Anteil negativer und invalider Werte im kalibrierten und bereinigten Datensatz  $X_{Leipzig18}$  mit einer Gesamtgröße von 499 875 840 Werten

Mithilfe der Werte aus Tabelle 3.5 kann für die negativen Zahlenwerte ein Anteil von 14,93 % an den validen Werten berechnet werden. Um diese Eigenschaft zu untersuchen, werden diejenigen Tagesabschnitte selektiert, die solche Werte beinhalten. Dabei stellt sich heraus, dass jeder Tagesabschnitt negative Werte aufweist.

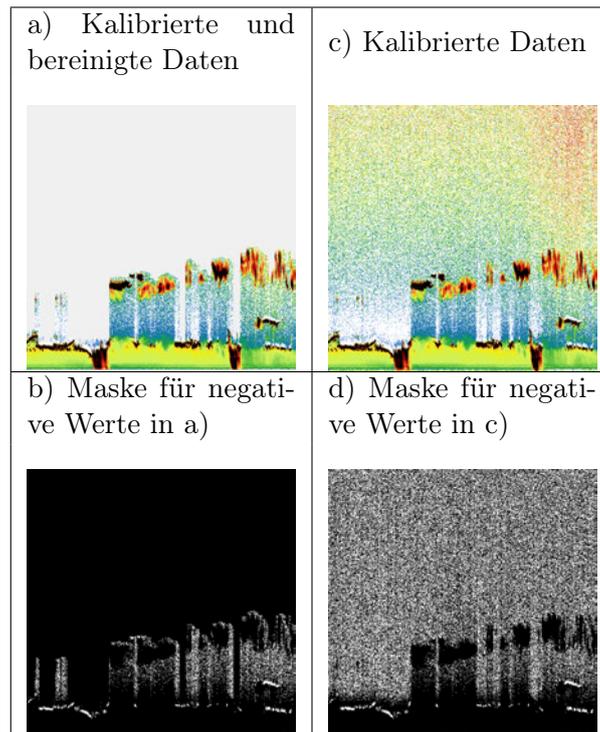


Abbildung 3.2: Visuelle Maskierung negativer Werte

In Abbildung 3.2, b) ist eine Maske dargestellt, die repräsentativ die tendenziellen Vorkommnisse der negativen Werte (in weiß) aufzeigt. In Abbildung 3.2, c) und d) wurde die Maske nicht bereinigten Daten gegenübergestellt. Da die negativen Werte oberhalb der Detektionsgrenze als Bestandteil von niedrigem SNR <sup>1</sup> zu verstehen sind, ergibt sich die Fragestellung, ob die negativen Werte unterhalb der Detektionsgrenze auch als nicht interpretierbar zu werten sind. Dieser Zusammenhang wurde vom DWD teilweise bestätigt. Negative Werte können zwar meistens als nicht interpretierbar gewertet werden. Wasserwolken weisen allerdings als typisches Merkmal oberhalb ihrer Ausprägung negative Werte auf.

Die invaliden Werte machen einen großen Anteil an den Gesamtdaten aus. Sie entstehen primär durch die Bereinigung mittels der maximalen Detektionshöhe. In Abbildung 3.2, a) weist die graue Farbe auf invalide Werte hin.

Im Folgenden wurden Ansätze zur Transformation der Daten mit dem Ziel, dem Problem der fehlschlagenden Rekonstruktionen entgegenzuwirken, abgearbeitet.

<sup>1</sup> *Signal-to-noise ratio* (SNR) ist ein Maß für die Qualität eines Signals.

Einer Erklärung <sup>2</sup> zur Interpretation der Rückstreusignale vom Ceilometer ist zu entnehmen, dass für die optische Mustererkennung eine logarithmische Farbskala verwendet werden kann. Die Distanzen zwischen den Farbübergängen sind disproportional. Während der rote Bereich <sup>3</sup> mit einer Intervallbreite von ca.  $249 \cdot 10^{-8}$  eher auf eine wässrige Eigenschaft der Atmosphärenbeschaffenheit hinweist, deutet ein gelber Bereich <sup>4</sup> mit einer Intervallbreite von  $34 \cdot 10^{-8}$  eher auf Eigenschaften eines Grenzschicht-Aerosols hin. Die folgende Transformation bedient sich dieser Interpretation. Hierbei wird eine Abbildung der 10 Intervalle der Farbskala auf eine lineare Skala von 0 bis 1 gemäß folgender Tabelle angewandt:

Vorher	Nachher	(1.8e-07, 3.5e-07)	f(5)
(Min, 1e-08)	f(0)	(3.5e-07, 6.9e-07)	f(6)
(1e-08, 2e-08)	f(1)	(6.9e-07, 1.34e-06)	f(7)
(2e-08, 4e-08)	f(2)	(1.34e-06, 2.63e-06)	f(8)
(4e-08, 9e-08)	f(3)	(2.63e-06, 5.12e-06)	f(9)
(9e-08, 1.8e-07)	f(4)	(5.12e-06, Max)	f(10)

$N = 11$

$f(t)$ : return  $(\frac{1}{N}t, \frac{1}{N}(t+1))$

Tabelle 3.6: Abbildung von logarithmischen Intervallen auf lineare Intervalle

Ein Wert, der in einem der Intervalle aus der ersten Zeile der Tabelle 3.6 liegt, wird auf das entsprechende Intervall in der zweiten Zeile skaliert.

Alternativ kann eine Log-Transformation genutzt werden. Dabei werden große Distanzen zwischen den vorkommenden Werten verkleinert und kleine Distanzen vergrößert. Hierfür ist erforderlich, dass die Werte größer als 0 sind. Dafür kann das Intervall des Datenbereichs um die Distanz zwischen dem niedrigsten negativen Wert und dem niedrigsten Wert  $> 0$  in den positiven Zahlenbereich geschoben werden. Eine andere Möglichkeit ist, die negativen Werte auf den niedrigsten Wert  $> 0$  zu setzen. In einer weiteren Variante werden die negativen Werte als nicht interpretierbar gewertet und somit den invaliden Werten zugeordnet. Alle genannten Varianten wurden angewandt. Im Folgenden wird die letztere aufgrund ähnlicher Resultate stellvertretend präsentiert.

<sup>2</sup>[https://www.dwd.de/DWD/forschung/projekte/ceilomap/files/Saharan\\_dust\\_example\\_en.pdf](https://www.dwd.de/DWD/forschung/projekte/ceilomap/files/Saharan_dust_example_en.pdf)

<sup>3</sup>ca. von  $263 \cdot 10^{-8}$  bis  $512 \cdot 10^{-8}$

<sup>4</sup>ca. von  $35 \cdot 10^{-8}$  bis  $69 \cdot 10^{-8}$

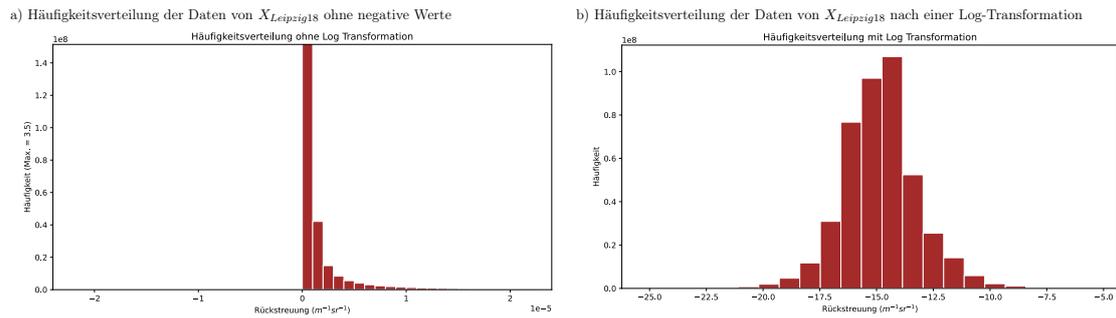


Abbildung 3.3: Häufigkeitsverteilung vor und nach einer Log-Transformation

Die Schiefe gibt in der Statistik die Asymmetrie einer Wahrscheinlichkeitsverteilung an. Mit *Fisher-Pearson coefficient of skewness* gemäß der Definition in Kapitel 1.3.5.11 von [Heckert u. a., 2013] wurde eine Schiefe von 38 für die Daten ohne negative Werte berechnet. Die Log Transformation kann genutzt werden, um die positive Schiefe zu reduzieren. Nach der Transformation ergab sich eine Schiefe von -1.

Der dritte Ansatz besteht aus der Verwendung deklarerter Min- und Max-Werte. Im Anhang A.3.1 von [Gandraß, 2019] wurden globale Minimum- und Maximum-Werte entdeckt, die testweise angewandt wurden.

Im Folgenden ist ein Vergleich der Rekonstruktionen unter Anwendung der Map- und der Log-Transformation sowie der angepassten Min-Max-Skalierung dargestellt.

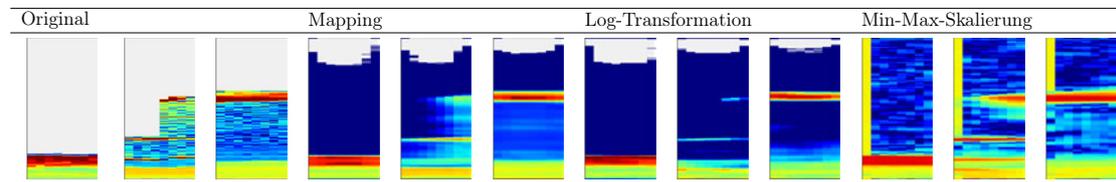


Tabelle 3.7: Rekonstruktionen von 3 repräsentativen Eingaben unter Anwendung verschiedener Datentransformationen nach jeweils 25 Epochen

Der Tabelle 3.7 ist zu entnehmen, dass die Min-Max-Skalierung den gegebenen anderen Transformationen überlegen ist. Einzig unter ihrer Anwendung ist der Autoencoder nach 25 Epochen in der Lage, die Wolkenschichten des zweiten Eingabebeispiels in angemessener Qualität zu rekonstruieren.

## 4 Entwurf

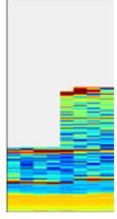
In diesem Kapitel werden sämtliche Entwurfsentscheidungen dokumentiert.

### 4.1 NaN-Werte

Im Folgenden wird sich mit der Frage auseinandergesetzt, wie mit den unbekanntem Werten umzugehen ist. Das Fehlen von Werten kann nützliche Informationen beinhalten.

There is information in missingness. ([Osborne, 2013], Kap. 6)

Im bisherigen Ansatz wird NaN durch 0 ersetzt. Die Visualisierung von Tabelle 3.7 legt nahe, dass das Netz nicht ausreichend zwischen bekannten und unbekanntem Datenpunkten unterscheidet, sodass die NaN-Werte nicht über einen definierten Schwellwert zurückgeholt werden können. Eine gute oder schlechte Differenzierung zwischen bekannten und unbekanntem Werten kann auf die Rekonstruktionsfähigkeit, die Interpretation der Eingabe einer Schicht und die latente Repräsentation Auswirkungen haben. Um die Differenzierung zu unterstützen, kann daher ein konstanter Abstand im Ziel-Wertebereich zwischen unbekanntem und bekannten Werten eingebaut werden. Alternativ kann der Eingabe ein weiterer Channel hinzugefügt werden, welcher als Maske, die die unbekanntem Werte markiert, fungiert. Die Maske kann der Interpretation unbekannter Werte dienlich sein und somit zur Präzision der Rekonstruktion im Training beihelfen. Unbekannte Werte werden mit 0 markiert, bekannte Werte mit 1.

Original	Standard	Konstanter Abstand	Maske	Konstanter Abstand und Maske	
					
MSE	0,01327	0,02600	0,00400	0,00402	

Konstanter Abstand bezeichnet den Bereich von 0 bis 0,1.

Für die Berechnung vom MSE wurden unbekannte Werte und Maske nicht betrachtet.

Tabelle 4.1: Vergleich der Ansätze zum Umgang mit NaN-Werten nach Trainings mit jeweils 25 Epochen

Im Rahmen der Erstellung von Tabelle 4.1 wurde die direkte Ausgabe des neuronalen Netzes über eine *Heatmap* visualisiert, die den Bereich von 0 bis 0,1 in den Fokus setzt. Hierbei wird deutlich, dass eine Maske deutlich zur Differenzierung zwischen bekannten und unbekanntem Werten beitragen kann. Die These zur Rekonstruktionspräzision wird nicht nur durch die präzise Rückholung der NaN-Werte gestützt: In der Tabelle ist der mittlere quadratische Fehler für lediglich Werte, die im Original bekannt sind, angegeben.

Alternativ kann außerdem die Nutzung eines RNNs, welches den Raum bis zur Wolkendecke abdeckt, zur Merkmalsextraktion der räumlichen Dimension genutzt werden. Dabei würde den unbekanntem Daten eher keine Informationsdichte zugesprochen werden. Weil in Kapitel 4.2.2 entschieden wurde, die räumliche Dimension mit einem CNN abzudecken, entfällt diese Variante.

## 4.2 Autoencoder

In diesem Kapitel werden Flaschenhals- und Architekturvarianten des LSTM-Autoencoders entworfen.

### 4.2.1 Flaschenhals

Um eine beliebig lange Sequenz auf eine komprimierte Größe zu reduzieren, wurden Entwurfsansätze für den Flaschenhals des Autoencoders zusammengetragen. Nachfolgend bezeichnet  $x$  die Eingabe,  $y$  die Ausgabe und  $h$  den Zustand der LSTM-Zelle. Die Variable  $t$  repräsentiert die zeitliche Dimension. Mit  $z$  wird die latente Repräsentation des Autoencoders bezeichnet. Hierbei ist im Vorfeld anzumerken, dass der Einfachheit halber die Notation eines RNNs für die nachfolgenden Abbildungen verwendet wird. In einem LSTM wird der Hidden-State als Ausgabe herausgegeben (siehe Abbildung 2.3).

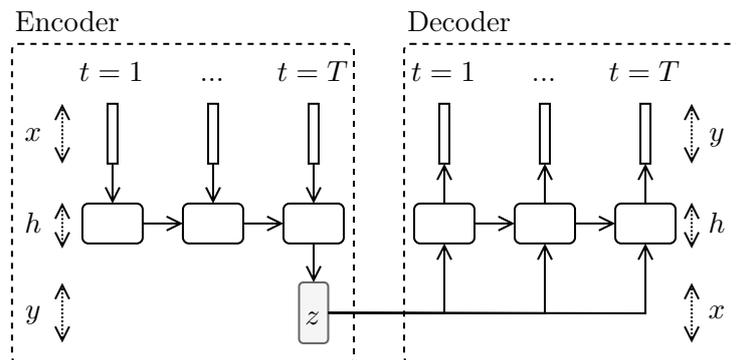


Abbildung 4.1: 1. Ansatz für den Flaschenhals des Autoencoders

Im ersten Ansatz ist die letzte Ausgabe des kodierenden LSTMs die latente Repräsentation. Diese wird unverändert repetitiv von  $t=1$  bis  $t=T$  als Eingabe in das dekodierende LSTM gegeben.

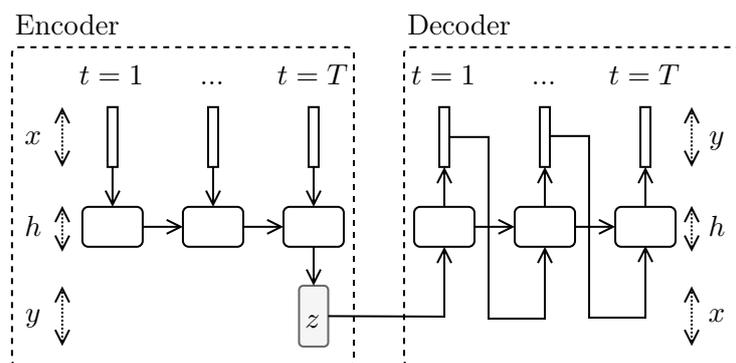


Abbildung 4.2: 2. Ansatz für den Flaschenhals des Autoencoders

Im zweiten Ansatz ist die latente Repräsentation ebenfalls die letzte Ausgabe des kodierenden LSTMs. Diese wird im dekodierenden LSTM als Eingabe für  $t=1$  verwendet. In

den darauffolgenden Schritten  $t > 1$  wird die Ausgabe von  $t-1$  als Eingabe hereingegeben. Alternativ kann die Extraktion der notwendigen Informationen für die Rekonstruktion einzelner Zeitpunkte auch in umgekehrter Reihenfolge von  $t=T$  bis  $t=1$  erfolgen.

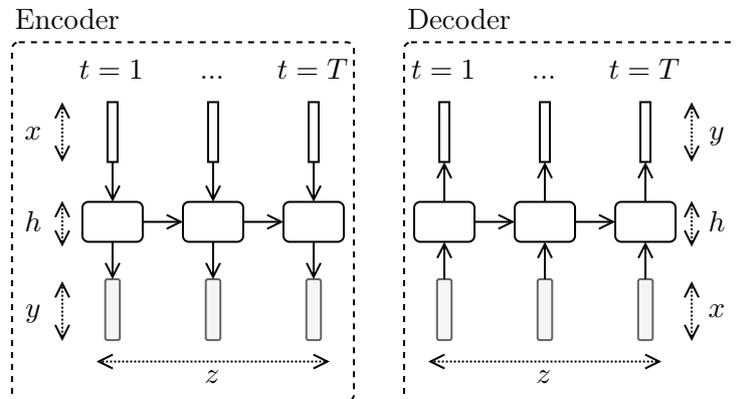


Abbildung 4.3: 3. Ansatz für den Flaschenhals des Autoencoders

Im dritten Ansatz wird die Eingabesequenz nicht in ihrer Länge komprimiert. Das kodierende LSTM gibt zu jedem Zeitpunkt  $t$  eine Ausgabe aus. Die Ausgabesequenz ist die latente Repräsentation, welche als Eingabesequenz in das dekodierende LSTM gegeben wird.

Im Folgenden wird diskutiert, welcher Ansatz zu bevorzugen ist. Weil eine Segmentierung mit zufälligen Segmentbreiten vorgenommen wird, erfordert die dritte Variante beim Clustering ein auf Sequenzen spezialisiertes Ähnlichkeitsmaßverfahren wie *Dynamic Time Warping* (DTW), das mit unterschiedlich langen Sequenzen umgehen kann. DTW ist ein langsames Verfahren („it is slow“ ([Guojun Gan, 2007], Kap. 6.6.3)). Alternativ liegen auch schnellere Varianten wie z.B. FastDTW [Salvador und Chan, 2007] vor. Der Ansatz ist allerdings aus verschiedenen Gründen nicht praktikabel: Zum einen muss die Größe der räumlichen Eingabedimension durch die vorherigen Schichten des Encoders auf ein abzuwägendes Minimum verkleinert werden, um einer hohen Zeitkomplexität im multidimensionalen Fall von DTW (siehe auch [Shokoohi-Yekta u. a., 2017]) entgegenzuwirken. Bei einer zu kleinen gewählten Größe besteht allerdings das Risiko des Informationsverlusts. Zum anderen wird die Auswahl des Deep-Clustering-Verfahrens in Kapitel 4.3.1 stark eingeschränkt und für die Verwendung eines VAEs in Kapitel 4.2.2 ist zusätzlicher Aufwand vonnöten. Daher wird vorgezogen, die latente Repräsentation auf eine fixe Größe zu setzen. Somit wird erster oder zweiter Ansatz präferiert.

Im Decoder des zweiten Ansatzes ist die Eingabe zum Zeitpunkt  $t$  bedingt durch die Ein- und Ausgabe zum Zeitpunkt  $t-1$ . Eine Abrufung einer relevanten Information zum

Zeitpunkt  $t$  kann also von einer vorherigen Abrufung von Informationen zum Zeitpunkt  $t - 1$  abhängen. Für eine Differenzierung von Objekten in einem anschließenden Clustering könnte dieses Verhalten von Nachteil sein. Eine latente Repräsentation, die ohne eine hierarchische Verarbeitung ihrer selbst alle Merkmale abrufbar beinhaltet, ist zu bevorzugen. Daher wird der erste Ansatz gewählt.

Als Alternative zu den genannten Ansätzen kann außerdem der letzte Hidden-State des kodierenden LSTMs als latente Repräsentation verwendet werden (vierter Ansatz). Dagegen spricht allerdings die gleiche Begründung wie für das Verwerfen des zweiten Ansatzes. Ein Vergleich von Ansatz 1, 2 und 4 unter Anwendung der in Kapitel 4.3.1 gewählten Deep-Clustering-Methode und/oder VAE könnte die Begründungen zu den Entwurfsentscheidungen stärken oder schwächen. Dies wird im Rahmen dieser Arbeit nicht abgedeckt.

### 4.2.2 Architektur

Folgende Architektur dient als Schablone für den Autoencoder:

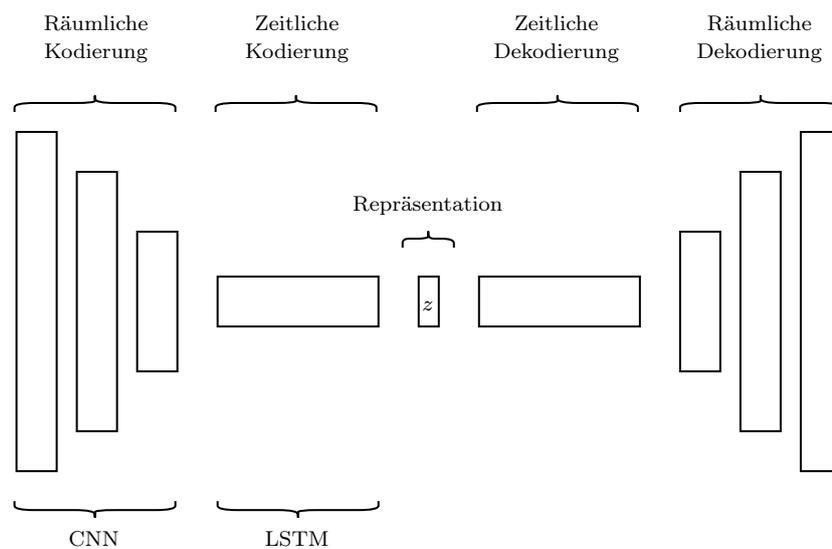


Abbildung 4.4: Entwurf LSTM-AE

Hierbei werden pro Zeitpunkt Merkmale aus der räumlichen Dimension mittels eines CNNs extrahiert. Die extrahierten Merkmale werden in ein die zeitlichen Muster kodierendes LSTM, das beliebig viele Eingaben akzeptiert, gegeben. Die letzte Ausgabe des

LSTMs stellt die latente Repräsentation der Eingabe dar und wird gemäß Abbildung 4.1 wiederkehrend als Eingabe in ein nachfolgendes dekodierendes LSTM gegeben. Ein CNN empfängt pro Zeitpunkt die Ausgabe vom LSTM und rekonstruiert den entsprechenden Teil der ursprünglichen Eingabe.

Klassische Autoencoder stellen ohne Weiteres nicht sicher, dass eine kleine Änderung in der latenten Repräsentation nur eine kleine Änderung in der Rekonstruktion zur Folge hat. Dies kann ein Clustering auf Basis der Repräsentationen erschweren. Die Qualität eines Clustering-Verfahrens mit einer Distanzmetrik wie bei K-Means ist tendenziell besser, wenn nahe Punkte (z.B. P1: (1, 5, 7, 3) und P2: (1, 5, 7, 2)) eine ähnliche Semantik aufweisen. Der Autoencoder stellt allerdings lediglich sicher, dass sich die Repräsentation eignet, damit der Decoder das Eingabebild rekonstruieren kann. Um sicherzustellen, dass eine kleine Änderung in der Repräsentation nur eine kleine Änderung in der Rekonstruktion zur Folge hat, eignen sich Variational-Autoencoder. Daher wurde für den nächsten Entwurf der Autoencoder durch einen Variational-Autoencoder ersetzt, sodass sich folgende Abbildung ergibt:

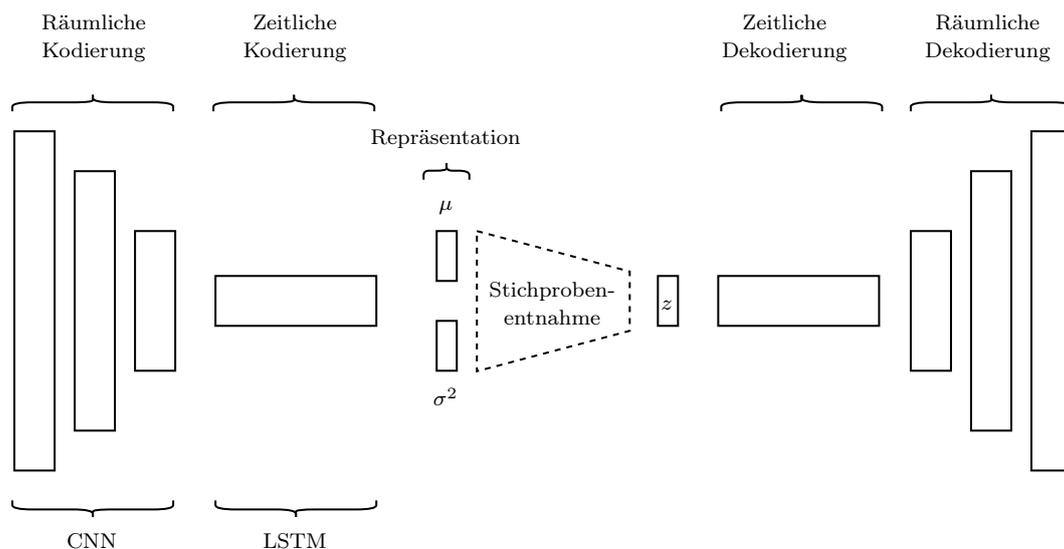


Abbildung 4.5: Entwurf LSTM-VAE

Hierbei werden zwei parallele Schichten, deren Ausgaben jeweils den Erwartungswert und die Varianz der latenten Variablen repräsentieren, und eine darauffolgende Schicht zur Stichprobentnahme im Flaschenhals vorgelagert. Die Wahrscheinlichkeitsverteilung  $P(z)$  wird also mit einer multivariaten Normalverteilung modelliert. Die Stichprobentnahme erfolgt mithilfe des *Reparameterization Tricks*, der besagt  $z = \mu + \sigma\epsilon$ , wobei

$\epsilon \sim \mathcal{N}(0, 1)$  (vgl. [Doersch, 2021], Kap. 2.2).

Für beide gezeigten Varianten sollen zusätzlich solche mit einem LSTM und solche mit einem ConvLSTM implementiert werden. In einem ConvLSTM ist die Ausgabe eine Sammlung von Feature-Maps, wobei eine Feature-Map ein Merkmal und der Inhalt der Feature-Map die Information über dessen räumliche Positionierung repräsentiert. Eine solche Struktur kann für ein Clustering von Vorteil sein. Für eine Darstellung dieser Struktur wird auf Abbildung 2.6 hingewiesen.

### 4.3 Clustering

Das Clustering erfolgt durch Wahl eines Deep-Clustering-Verfahrens und gruppiert die Eingabesequenzen auf Basis der latenten Repräsentation. Im Folgenden wird auf die Wahl des Verfahrens und der Kennzahlen für die Clusteringauswertung eingegangen.

#### 4.3.1 Deep Clustering

Um ein der Vorlage (siehe Abbildung 4.4) entsprechendes Clustering anzufügen, kommt am ehesten ein Deep-Clustering-Verfahren der Kategorie „AE-bezogenes Deep-Clustering“ in Frage. Repräsentative Methoden sind u.a. IDEC oder DCN. In den Messungen aus Tabelle 3 von [Yang u. a., 2017] ist kein nennenswerter Vorteil in der Verwendung von DCN gegenüber DEC zu erkennen, während in den Messungen aus Tabelle 2 von [Guo u. a., 2017] für die Verwendung von IDEC bessere Kennzahlen auftauchen. Daher wird IDEC als Verfahren gewählt. Weitere in [Min u. a., 2018] genannte Methoden könnten angewandt und verglichen werden. Dies ist aber nicht Bestandteil dieser Arbeit.

#### 4.3.2 Interne Kennzahlen

Weil mit IDEC ein Fuzzy-Clustering vorliegt, kann jeder Datenpunkt zu jedem Cluster mit unterschiedlichen Zugehörigkeitswahrscheinlichkeiten gehören. Ein *Fuzzy Validity Index* wird mit dem Ziel, ein Clustering-Modell zu finden, bei welchem die meisten Datenpunkte im Datensatz eine hohe Zugehörigkeitswahrscheinlichkeit zu einem Cluster erlangen, genutzt. Es bestehen dazu verschiedene Kennzahlenberechnungen. (vgl. [Guojun Gan, 2007], 17.4)

Für die Evaluierung wird entschieden, den *Partition Coefficient Index* ( $PK$ ) auszugeben.

$$PK = \frac{1}{n} \sum_{j=1}^c \sum_{i=1}^n p_{ji}^2,$$

wobei  $j$  das Cluster und  $i$  die Eingabe indiziert. Die Variable  $p_{ji}$  bezeichnet die entsprechende Zugehörigkeitswahrscheinlichkeit. Der schlechteste Wert für  $PK$  ist  $\frac{1}{c}$  und der beste Wert 1. (vgl. [Guojun Gan, 2007], 17.4.1) Die Berechnung basiert auf dem Ziel, dass die Zugehörigkeitswahrscheinlichkeit zu einem Cluster entweder möglichst nahe 0% oder möglichst nahe 100% sein sollte. Zur Veranschaulichung wird auf Abbildung 2.13 hingewiesen.

Um eine weitere Kennzahl präsentieren zu können, welche die Repräsentation der Exemplare zur Bewertung von Kohäsion und Separation einbezieht, wird der *Fukuyama-Sugeno Index* ( $FS$ ) gemäß Definition in Kapitel 17.4.3 von [Guojun Gan, 2007] genutzt.

$$\begin{aligned} FS_K &= \sum_{i=1}^n \sum_{j=1}^c p_{ij}^2 \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \\ FS_S &= \sum_{i=1}^n \sum_{j=1}^c p_{ij}^2 \|\boldsymbol{\mu}_j - \bar{\boldsymbol{\mu}}\|^2 \\ FS &= \frac{1}{n} (FS_K - FS_S), \end{aligned}$$

wobei  $x$  eine Repräsentation,  $\mu$  einen Cluster-Zentroiden und  $\bar{\mu}$  den Mittelwert der Cluster-Zentroiden darstellen. Der erste Term  $FS_K$  berechnet Kohäsion, indem die Distanz zwischen Datenpunkt und Cluster-Zentroiden gewichtet mit der jeweiligen Zugehörigkeitswahrscheinlichkeit berechnet wird. Ein niedriger Wert weist also auf hohe Kohäsion hin. Der zweite Term  $FS_S$  berechnet Separation, indem die Distanz zwischen Cluster-Zentroiden und Mittelwert der Cluster-Zentroiden gewichtet mit der jeweiligen Zugehörigkeitswahrscheinlichkeit berechnet wird. Ein hoher Wert weist daher auf hohe Separation hin. Die originale Definition wurde durch eine Gewichtung  $\frac{1}{n}$  ergänzt, um Ergebnisse bei unterschiedlicher Anzahl von Exemplaren vergleichen zu können.

### 4.3.3 Externe Kennzahlen

Die nachfolgend beschriebenen externen Kriterien können lediglich bei gelabelten Daten Anwendung finden, also primär bei dem künstlichen Datensatz. Bevor die Wahl der externen Kriterien erfolgt, sollte eine „ideale“ Partition gebildet werden.

Ein Segment hat ein Label, das sich gemäß Kapitel 3.1 folgendermaßen zusammensetzt: Angenommen, 3 Klassen liegen vor. Zu jedem Zeitpunkt wird angegeben, ob die jeweilige Klasse vorhanden ist (1) oder nicht (0). Für jede Klasse werden jeweils die Vorkommnisse im gesamten Segment addiert und in Verhältnis zu der Gesamtlänge des Segments gesetzt.

Zeitpunkt	00:00	00:01	00:02
Klasse 1	1	1	1
Klasse 2	1	0	0
Klasse 3	0	1	1

Tabelle 4.2: Flags eines Segments mit Label [1, 0.33, 0.66]

Im Idealfall werden Klassen repräsentierende Cluster gefunden. Da durch Kapitel 3.1 8 bekannte Klassen gegeben sind, wird eine Clusteranzahl von 8 gewählt.

Die Korrelation zwischen Cluster und Klasse wird mithilfe einer Korrelationsmatrix visualisiert. Dabei werden für jedes Cluster die Segmente mit der höchsten Zuweisungswahrscheinlichkeit gesammelt. Daraufhin wird für jedes Cluster die durchschnittliche Konzentration an Auftreten der jeweiligen Klassen berechnet. Auf diese Weise soll bereits ohne ideale Vergleichspartition ein Zusammenhang zwischen Cluster und Klasse indiziert werden.

Um die Segmente den Klassen zuzuordnen und auf diese Weise eine ideale Vergleichspartition zu erzeugen, könnten Klassen repräsentierende Segmente im Voraus für den künstlichen Datensatz generiert werden. Für eine zukünftige Anwendung auf einen teilweise gelabelten DWD-Datensatz wird stattdessen jeder Klasse ein „Label-Zentroid“ zugewiesen.

Klasse	Label-Zentroid
Hintergrundrauschen	1, 0, 0
Nebel	1, 1, 0
Biomasseaerosol	1, 0, 1

Tabelle 4.3: Klassen mit zugewiesenen Label-Zentroiden

Segmente, die als Repräsentanten einer Klasse gelten, können daraufhin auf Basis der Nähe ihres Labels zum Label-Zentroiden einer Klasse selektiert werden. Die Nähe wird über den mittleren quadratischen Fehler berechnet. Durch eine manuelle Auswertung hat sich gezeigt, dass mit einer Fehlertoleranz von 0,1 eine ausreichend angemessene Zuordnung produziert wird.

Weil nun eine ideale Partition vorliegt, können populäre externe Kennzahlen berechnet werden wie z.B. *Purity*. Weil ein positiv zu bewertender Wert für *Purity* leicht erzeugt werden kann, wenn die Anzahl der Cluster hoch ist (vgl. [Christopher D. Manning und Schütze, 2008], Kap. 16.3), wird *Normalized mutual info score* (NMI) verwendet. Darüber hinaus wird diese Kennzahl standardmäßig zur Messung in wissenschaftlichen Veröffentlichungen zu Deep-Clustering verwendet (vgl. [Min u. a., 2018], Kap. 2).

Für diese Kennzahl wird die diskrete Clusterzugehörigkeit mit einer Zufallsvariable modelliert. Die Zufallsvariablen  $U$  und  $V$  mit  $u, v \in \{1, \dots, k\}$  bei  $k$  Clustern geben die Clusterzugehörigkeit in der tatsächlichen und der idealen Partitionierung  $C^U$  und  $C^V$  an. Eine Partitionierung weist jedes Exemplar  $x$  aus dem Datensatz  $X$  mit  $|X|=n$  einem Cluster  $C_i$  hinzu. Eine Partition ist dann gegeben als  $C = \{C_1, \dots, C_k\}$ .

$$P(U = u) = \frac{|C_u^U|}{n}$$

$$P(V = v) = \frac{|C_v^V|}{n}$$

$$P(U = u, V = v) = \frac{|C_u^U \cap C_v^V|}{n}$$

$$MI(U, V) = D_{KL}(P(U, V) || P(U)P(V)) \quad (4.1)$$

$$NMI_{sqr}(U, V) = \frac{MI(U, V)}{\sqrt{H(U)H(V)}} \quad (4.2)$$

In Gleichung 4.1 wird gemäß Definition von  $D_{KL}(P || Q)$  der Informationsverlust gemessen, der einhergeht, wenn statt der multivariaten Wahrscheinlichkeitsverteilung  $P(U, V)$  die Verteilung  $P(U)P(V)$ , welche die totale statistische Unabhängigkeit von  $U$  und  $V$  repräsentiert, genutzt wird. Ein hoher Wert für die KL-Divergenz bedeutet, dass eine hohe Abhängigkeit zwischen  $U$  und  $V$  besteht. Wenn  $V = v$ , dann lässt auf  $U = u$  schließen. In Gleichung 4.2 wird der Wert auf das Intervall 0 bis 1 normalisiert. Dies ist wie dargestellt möglich, da  $MI(U, V) \leq \min(H(U), H(V)) \leq \sqrt{H(U)H(V)}$ , wobei  $H$  die Shannon-Entropie darstellt. (vgl. [Christopher D. Manning und Schütze, 2008], Kap. 16.3 und [Vinh u. a., 2010], Kap. 3)

Zusätzlich wird der *Adjusted mutual info score* (AMI), der eine alternative Form der

Normalisierung vornimmt, ausgegeben.

Um eine detailliertere manuelle Auswertung machen zu können, wird ein Mapping von Objektmengen durchgeführt. Dabei wird die Überschneidung einer beliebigen idealen Partition mit der tatsächlichen Partition durch eine Kontingenztabelle der Größe  $N\_CLUSTER \times M\_IDEAL\_CLUSTER$  offengelegt. Für jede Kombination an Cluster und idealem Cluster wird die Überschneidungsmenge  $I$  von den dem Cluster und dem idealen Cluster zugewiesenen Segmentmengen ermittelt und  $|I|$  als Wert in die entsprechende Zelle der Tabelle eingetragen.

Label-Zentroid	1	0	0	0	
	0	1	0	1	Summe
	0	0	1	1	
Cluster 0	43	2	0	0	45
Cluster 1	1	0	19	8	28
Cluster 2	3	31	5	6	45
Summe	47	33	24	14	118

$N\_CLUSTER = 3$

$M\_IDEAL\_CLUSTER = 4$

Tabelle 4.4: Beispiel einer Kontingenztabelle

## 5 Implementierung

Für die Implementierung lag bereits der vorhandene Prototyp von [Gandraß, 2019] vor. Insbesondere die Datenvorverarbeitung (Kalibrierung und Bereinigung) konnte hierbei wiederverwendet werden.

### 5.1 Werkzeuge

Als Programmiersprache wurde Python 3.9 verwendet. Python ist eine höhere Sprache, die mit einem Interpreter zu Maschinensprache übersetzt wird. Sie ist in der standardisierten Sprache ANSI C geschrieben und daher plattformunabhängig. Sie ist dynamisch typisiert und verwendet eine *Garbage Collection*. Darüber hinaus ist die Anwendung von sowohl objektorientierten, prozeduralen als auch funktionalen Programmierparadigmen möglich. (vgl. [Lutz, 2013], 1)

Als Framework für neuronale Netze wurden Tensorflow 2.5.0 und Keras 2.4.3 verwendet. Tensorflow ist eine primär von Google entwickelte Plattform, welche dazu dient, mathematische Ausdrücke über numerische Tensoren effizient zu verarbeiten. Sie kann sich GPUs und TPUs für eine möglichst parallelisierte Verarbeitung zu Nutze machen. Darüber hinaus bietet sie eine automatische Berechnung des Gradienten für einen differenzierbaren Term. (vgl. [Chollet, 2021], Kap. 3.1) Ein Term kann mittels der Konfiguration „Graph Execution“ in Tensorflow über einen Berechnungsgraphen abgebildet werden. Statt wie in der standardmäßig aktivierten „Eager Execution“ Operationen sequentiell abzuarbeiten, werden hierbei die Operationen durch gezielte Aufteilung des Graphen im Rahmen eines Optimierungsprozesses parallelisiert. Tensorflow liefert zusätzlich das Werkzeug Tensorboard, welches unter anderem die Visualisierung von Metriken, Berechnungsgraphen und Embeddings ermöglicht.

Keras bietet eine abstrakte API für Anwendungen von maschinellem Lernen. Die Bibliothek wird als Frontend bezeichnet, weil sie ein kompatibles Backend wie Tensorflow benötigt. Sie vereinfacht die Entwicklung von Lösungen, indem typische Anwendungsfälle

zusammengefasst werden. (vgl. [Chollet, 2021], Kap. 3.2)

Als Hardware wurde ein Computer mit einem Intel i7-6700, 32 GB RAM und einer NVIDIA GeForce GTX 1050 Ti verwendet.

## 5.2 Datenformat

In der Tabelle 5.1 wird das für das entwickelte Programm benötigte Datenformat beschrieben. Dabei wird ein Attribut einer obligatorischen Wertezuweisung zugeordnet, wobei ein Wert einem Tensor entspricht.

Attribut	Inhalt	Dimensionen
<code>data</code>	Hauptdaten	Zeitpunkt, Höhe, Channel
<code>image_data</code>	Zu Bilddaten transformierte Hauptdaten	Breite, Höhe, RGBA
<code>labels</code>	Pro Zeitpunkt ein NaN	Zeitpunkt, NaN
	Pro Zeitpunkt ein Array mit einem Wahrheitswert pro möglicher Klassenzugehörigkeit	Zeitpunkt, Klasse, Wahrheitswert
<code>times</code>	Pro Zeitpunkt ein Zeitstempel als Text	Zeitstempel

Tabelle 5.1: Datenformat des Datensatzes

Die Tensoren werden durch Numpy-Arrays implementiert. Die Attribute werden gemeinsam als .npz-Datei abgespeichert. Das .npz-Dateiformat dient der Archivierung mehrerer Variablen. Das entwickelte Programm erwartet mindestens valide Daten in `data` und `times` für eine korrekte Ausführung.

## 5.3 Ordnerstruktur

Für die Durchführung eines Trainings wird ein Wurzelverzeichnis mit der in Abbildung 5.1 gezeigten Struktur benötigt.

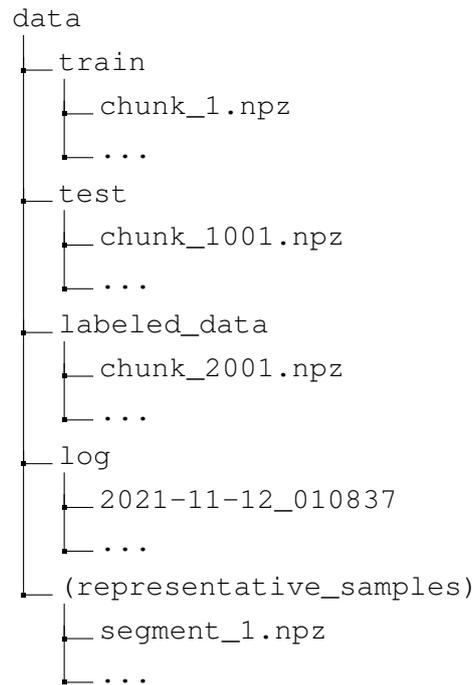


Abbildung 5.1: Ordnerstruktur

In den Ordnern `train` und `test` werden die Trainings- und Testdaten erwartet. Im Ordner `labeled_data` können Daten mit Labels für die Evaluation des Clustering abgelegt werden. Während der Durchführung eines Trainings wird im Ordner `log` ein Unterverzeichnis mit der in Abbildung 5.2 gezeigten Struktur angelegt. Optional kann im Wurzelverzeichnis ein Ordner `representative_samples` angelegt werden. In diesem werden Segmente erwartet, für welche Clusterzugehörigkeitswahrscheinlichkeiten ermittelt werden sollen.

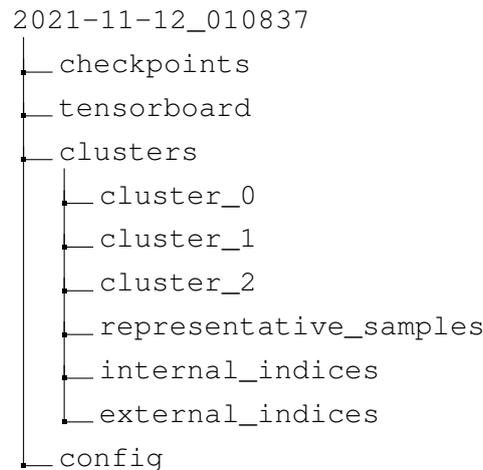


Abbildung 5.2: Ordnerstruktur eines Trainingprotokolls

Im Ordner `checkpoints` werden die durch den Callback `keras.callbacks.ModelCheckpoint` pro Epochendurchlauf erzeugten `.h5`-Dateien gespeichert, welche die Parameter des bis zum Callback-Zeitpunkt trainierten Modells beinhalten. Im Ordner `tensorboard` werden die für Tensorboard benötigten Dateien durch den Callback `keras.callbacks.TensorBoard` gespeichert. Im Ordner `clusters` werden Cluster zuzuordnende Unterverzeichnisse angelegt, in welchen Segmente mit hoher Zuweisungswahrscheinlichkeit zu finden sind. Außerdem befinden sich hier Abbildungen und Daten der Clustering-Auswertung. Darüber hinaus ist im Wurzelverzeichnis die für das Training genutzte Konfiguration gespeichert.

### 5.4 Konfiguration

In der Datei `config.py` befinden sich die Konfigurationsparameter für das Training. Die Tabelle 5.2 beinhaltet Beschreibungen der wichtigsten Parameter.

Parameter	Beschreibung
BATCH_SIZE	Anzahl der Segmente gleicher Größe in einem Batch
MIN_TIMESTEPS	Minimal erlaubte Segmentbreite
MAX_TIMESTEPS	Maximal erlaubte Segmentbreite
HEIGHT	Länge des Vektors pro Zeitpunkt
N_CHANNELS	Anzahl der Channels des Segments
N_EPOCHS	Anzahl der Epochen
N_RESEGMENTATIONS	Anzahl der Durchführungen einer Segmentierung mit anschließender Akkumulation der Segmente aller Durchführungen
N_CLUSTERS	Anzahl der zu ermittelnden Cluster
N_LABEL_FLAGS	Anzahl der Klassen bzw. idealen Cluster
DO_USE_CONV_LSTM	True: Nutzung von ConvLSTMs False: Nutzung von LSTMs
DO_USE_VAE_LAYER	True: Nutzung eines Variational Autoencoders False: Nutzung eines Autoencoders

Tabelle 5.2: Konfigurationsparameter

In der Konfigurationsdatei werden zudem das Wurzelverzeichnis von Abbildung 5.1, die Label-Zentroiden aus Kapitel 4.3.3 und weitere Parameter für die Auswertung gemäß Kapitel 4.3 festgelegt.

## 5.5 Datengenerierung

Mit der ausführbaren Datei `SyntheticSequenceGenerator.py` können künstliche Daten generiert werden. Höhe, Gesamtlänge und zuzuschneidende Blocklänge des zweidimensionalen Datensatzes können vorab festgelegt werden. Der Objekttyp `Figure-Properties` wird genutzt, mit welchem die spezifischen Eigenschaften eines Musters definiert werden können.

Attribut	Beschreibung
Muster	Rastergrafik mit RGBA-Farben
Länge	Potenzielle Anzahl der Spalten der Rastergrafik
Höhe	Potenzielle Anzahl der Zeilen der Rastergrafik
Höhenposition	Potenzielle Position in der Zeilen-Dimension der Datensatz-Rastergrafik
Augmentation	Datenaugmentationsfunktionen aus der Python-Bibliothek <code>imgaug</code>
BlockierendeMuster	<code>FigureProperties</code> für Muster, die nicht gleichzeitig zu diesem Muster erscheinen dürfen
AbhängigeMuster	<code>FigureProperties</code> für Muster, deren Auftreten vom Auftreten dieses Musters abhängen

Tabelle 5.3: Attribute in `FigureProperties`

Eine Hierarchie der Muster kann festgelegt werden. Ein Muster auf einer höherliegenden Ebene darf das unterliegende Muster überlappen oder verdecken. Ausnahmen können über das Attribut `BlockierendeMuster` aus Tabelle 5.3 festgelegt werden.

Initial liegt ein leeres und transparentes Bild vor, welches den gesamten Datensatz repräsentieren soll. Das Bild soll mit wiederkehrenden Mustern variabler Länge und zufälligem Auftreten gefüllt werden.

Entsprechend der festgelegten Hierarchie der Muster, wobei das hierarchisch niedrigste an erster Stelle steht, wird im Rahmen der Methode `GeneriereMehrereMuster` der in Algorithmus 1 abgebildete Prozess für die jeweiligen Muster durchgeführt.

**Algorithmus 1** *GeneriereEinMuster*

---

```
1: Eingabe: Datensatz D, FigureProperties F
2: Zeitpunkt T = 0
3: while T < D.Länge do
4:   if Zufälliger Wahrheitswert then
5:     Höhe H = Zufallswert(F.Höhe)
6:     Länge L = Zufallswert(F.Länge)
7:     Höhenposition P = Zufallswert(F.Höhenposition)
8:     if not D.Beinhaltet(F.BlockierendeMuster, T, L) then
9:       Muster M = Initialisiere(F.Muster, H, L)
10:      F.Augmentation(M)
11:      GeneriereMehrereMuster(M, F.AbhängigeMuster)
12:      D.Einfügen(M, T, P)
13:      T += L
14:    end if
15:  end if
16:  T += Zufallswert
17: end while
```

---

Zu zufälligen Zeitpunkten wird ein Auftritt des gerade betrachteten Phänomens getriggert. Daraufhin werden zufallsbasiert Eigenschaften wie Länge und Höhenpositionen des Phänomens generiert. Nach Erzeugung des Bildes wird es um abhängige Phänomene ergänzt. Falls keine blockierenden Muster an der planmäßig einzufügenden Stelle im Datensatz enthalten sind, wird das vollständig neue kreierte Muster in das Gesamtbild eingefügt.

## 5.6 Segmentierung

Ein Batch in einem LSTM benötigt gleich große Segmente. Gesucht wird also eine Menge  $S$  an gleichmäßig verteilten Zahlen zwischen `MIN_TIMESTEPS` und `MAX_TIMESTEPS`, sodass  $f(S) = \sum_s sb$  gleich der Gesamtlänge des Datensatzes ist, wobei  $b$  die Anzahl der Exemplare in einem Batch darstellt.

---

**Algorithmus 2** Segmentierung

---

- 1: **Eingabe:** Länge des Datensatzes  $L$ , Größe eines Batches  $B$ , Minimale Segmentlänge  $MIN$ , Maximale Segmentlänge  $MAX$
  - 2: **Ausgabe:** Segmentlängensammlung  $S$
  - 3: **while**  $\sum_s^n sB < L$  **do**
  - 4:     Ziehe Stichprobe aus einer gleichmäßigen Verteilung über  $[MIN, MAX]$
  - 5:     Füge Stichprobe  $S$  hinzu
  - 6: **end while**
  - 7: Entferne letzten Eintrag aus  $S$
- 

## 5.7 Training

Mit der ausführbaren Datei `Trainer.py` kann ein Training mit den in `config.py` definierten Einstellungen durchgeführt werden. Hierbei stehen vier verschiedene Modelle zur Auswahl: Das LSTM-AE, das ConvLSTM-AE, das LSTM-VAE und das ConvLSTM-VAE. Für alle Varianten wurde für die latente Repräsentation eine Dimensionsgröße von 32 gewählt. Die Datei enthält Methoden für die Experimentdurchführungen in Kapitel 6.

Die Datei `Evaluator.py` enthält Methoden zur Auswertung der Qualität der trainierten Modelle. Im Anschluss an das Vortraining des Autoencoders werden exemplarische Rekonstruktionen visualisiert, sodass die grundlegende Rekonstruktionsfähigkeit validiert werden kann. Außerdem werden die kleindimensionalen Repräsentationen („Embeddings“) von exemplarischen Eingaben mitsamt entsprechender originaler Bilddarstellung in Tensorboard kompatiblen Dateien gespeichert. Diese können im sogenannten „Embedding Projector“ von Tensorboard für eine t-SNE Analyse verwendet werden. Eine t-SNE Analyse ist zwar interessant, da die Berechnung der Zugehörigkeitswahrscheinlichkeit in DEC von t-SNE inspiriert wurde. Trotzdem ist sie lediglich für eine grobe Einschätzung der initialen Clusterqualität zu verwenden.

Darüber hinaus ist die Python Datei für die Auswertung des Clusterings gemäß der Entwurfsentscheidungen (siehe Kapitel 4.3.2 und 4.3.3) zuständig. Für die Ausgabe der dort beschriebenen Korrelationsmatrix werden die Bibliotheken `matplotlib` und `seaborn` verwendet. Für die Ausgabe der dort beschriebenen Kontingenztabelle werden die Bibliotheken `texttable` und `latexable` verwendet. Für die Berechnung der externen Kennzahlen konnte auf die Bibliothek `sklearn` zurückgegriffen werden.

Die Verwendung der in Kapitel 5.1 genannten GPU ist nur eingeschränkt möglich. Ein aufeinanderfolgendes Training kann zu Speicherproblemen führen, weil von Tensorflow keine Möglichkeit angeboten wird, den verwendeten Speicher wieder freizugeben. Dies

kann gelöst werden, indem für jedes Training ein eigener Kindprozess gestartet und beendet wird. Unter Verwendung einer maximalen Segmentbreite  $\geq 128$  bricht das Programm in der zweiten Phase des IDEC-Verfahrens aufgrund mangelnden GPU-Speichers ab. Daher muss die Verwendung der GPU für diese Anwendungsfälle deaktiviert werden.

# 6 Ergebnisse

## 6.1 Maximale Segmentbreite

Um den Bereich an möglichen maximalen Segmentbreiten einzuschränken bzw. eine angemessene maximale Segmentbreite bei gegebenen künstlichen Daten und fester Größe der latenten Repräsentation zu schätzen, werden mithilfe der Modell-Version LSTM-VAE und IDEC Trainings mit unterschiedlicher maximaler Segmentbreite durchgeführt.

Um unterschiedliche maximale Segmentbreiten miteinander vergleichen zu können, muss sichergestellt werden, dass annähernd eine Gleichverteilung der möglichen Breiten erzwungen wird. Dies wird durch die in Algorithmus 2 beschriebene Segmentierung gewährleistet.

Max. Breite	32	64	96	128	160	192	224
Epochen	10	10	10	10	10	10	10
PK	0,841	0,834	0,741	0,769	0,822	0,776	0,722
FS	38	29	28	28	39	28	28
NMI	0,758	0,713	0,727	0,525	0,453	0,387	0,381
AMI	0,757	0,712	0,725	0,521	0,445	0,377	0,369
Minuten	187	118	95	220	241	255	229

Tabelle 6.1: Vergleich maximaler Segmentbreiten

Als Auswahlkriterium für die besten maximalen Segmentbreiten wird zwischen kurzer Trainingsdauer, großer Breite und hoher externer Clustering-Kennzahlen abgewogen. Hierbei ergibt sich, dass die Breiten von 32 bis 96 angemessene Kandidaten darstellen. Die Breite 32 liefert den besten PK Wert und annähernd bestwertige externe Kennzahlen. Die Breite 64 liefert die besten externen Kennzahlen. Die Breite 96 weist zwar keine Bestwerte auf, aber bietet eine große Breite, während sich gleichzeitig Trainingsdauer und Kennzahlen in einem angemessenen Rahmen befinden.

Die Trainingsdauer wird durch die Anzahl an Batches beeinflusst. Durch den Segmentierungsprozess ergeben sich für kleinere maximale Segmentbreiten mehr Batches. Die Abweichungen wurden für maximale Segmentbreiten größer als 128 mithilfe des Konfigurationsparameters `N_RESEGMENTATIONS` abgeschwächt.

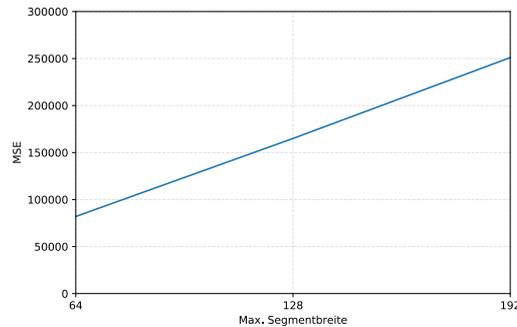


Abbildung 6.1: Durchschnittlicher Rekonstruktionsfehler nach 10 Epochen im Vergleich von maximalen Breitewerten

Abbildung 6.1 ist zu entnehmen, dass bei konstanter Erhöhung der maximalen Segmentbreite von 64 bis 192 eine konstante Erhöhung des Rekonstruktionsfehler nach gleicher Anzahl durchgeführter Epochen zu verzeichnen ist. Die Größe des Flaschenhalses im Autoencoder ist unter Verwendung der im Experiment genutzten Parameter also kein limitierender Faktor für eine verlustlose Komprimierung. Die Wahl einer gleichen Epochenzahl und die gegebenen Abweichungen in der Batchanzahl werden daher als angemessen bewertet.

## 6.2 Vergleich variabler und fixer Breite

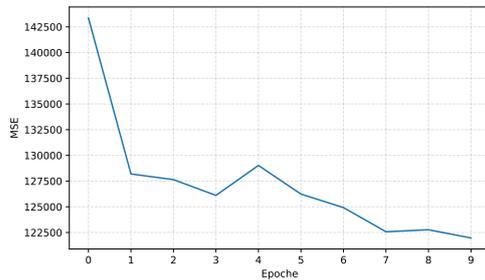
Um sicherzustellen, dass die Verwendung einer variablen Segmentbreite der Verwendung einer fixen Breite bei einem Datensatz mit zufällig auftretenden und zufällig lang andauernden zu differenzierenden Mustern zu bevorzugen ist, wird die Clusterqualität von beiden Szenarien verglichen. Hierfür werden die in Kapitel 6.1 als angemessen bewerteten maximalen Breitenwerte den jeweiligen Mittelwerten der möglichen Breiten als fixe Fensterbreite gegenübergestellt. Für die Untersuchung wird außerdem ein ungünstiges Szenario kreiert: Die Zeitscheibenproblematik wird nicht explizit durch signifikante temporale Muster wie in Tabelle 3.3 dargestellt provoziert.

	Fix	Variabel
Fixe Breite / Maximale Breite	8	16
Epochenanzahl	10	10
PK	0,808	0,857
FS	26	45
NMI	0,707	<b>0,738</b>
AMI	0,707	<b>0,737</b>
Fixe Breite / Maximale Breite	16	32
Epochenanzahl	10	10
PK	0,793	0,841
FS	26	38
NMI	0,721	<b>0,758</b>
AMI	0,720	<b>0,757</b>
Fixe Breite / Maximale Breite	32	64
Epochenanzahl	10	10
PK	0,710	0,834
FS	22	29
NMI	0,612	<b>0,713</b>
AMI	0,610	<b>0,712</b>
Fixe Breite / Maximale Breite	48	96
Epochenanzahl	10	10
PK	0,809	0,741
FS	27	28
NMI	0,454	<b>0,727</b>
AMI	0,450	<b>0,725</b>

Tabelle 6.2: Vergleich fixer und variabler Segmentbreiten

Trotz der repetitiven Eigenschaften der einzelnen Phänomene bzw. dem Fehlen von längeren temporalen Mustern in Tabelle 3.2 erzielt die Nutzung einer variablen Breite für alle Vergleiche die besseren externen Kennzahlen. Hierbei ist außerdem zu beobachten, dass die Clusterqualität bei erhöhter fixer Breite ab 16 stetig abnimmt, während die Clusterqualität bei beliebiger Wahl einer der gegebenen maximalen Breiten keine starken positiven oder negativen Steigungen aufweist.

a) Fixe Breite von 48



b) Maximale Breite von 96

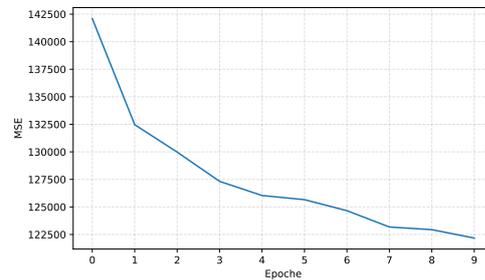


Abbildung 6.2: Durchschnittlicher Rekonstruktionsfehler pro Epoche im Vergleich zwischen fixer Breite von 48 und variabler Breite mit 96 als Maximum

Der Abbildung 6.2 ist zu entnehmen, dass die Größe des Flaschenhalses im Autoencoder kein limitierender Faktor für eine verlustlose Komprimierung ist und keinen Einfluss auf das Experiment hat. Die Wahl der gleichen Epochenzahl wird daher als angemessen bewertet.

### 6.3 Vergleich der Modelle

Um die performanteste Architekturvariante zu bestimmen, wurden entsprechend Trainings unter Verwendung des künstlichen Datensatzes und der in Kapitel 6.1 ermittelten größten angemessenen maximalen Segmentbreite durchgeführt.

ConvLSTM	Nein	Nein	Ja	Ja	Nein
VAE	Ja	Nein	Ja	Nein	Ja
IDEC	Ja	Ja	Ja	Ja	Nein
Epochen	10	10	10	10	10
PK	<b>0,741</b>	0,572	0,737	0,596	(0,416)
FS	28	<b>12</b>	65	27	(10)
NMI	0,727	0,708	0,712	0,669	<b>0,761</b>
AMI	0,725	0,706	0,710	0,667	<b>0,760</b>

Durch IDEC dupliziert sich die Anzahl an Epochen.

Tabelle 6.3: Kennzahlen unter Anwendung verschiedener Varianten mit einer maximalen Breite von 96

Die in Tabelle 6.3 gezeigten internen Kennzahlen PK und FS sind *Fuzzy-Validity-Indices*, welche die Qualität dahingehend prüfen, dass möglichst alle Elemente einem Cluster zu

0 % oder 100 % zugehörig sein sollen (siehe Kapitel 4.3.2). Dieses Ziel wird ohne Anwendung von IDEC nicht explizit verfolgt. Daher werden die internen Kennzahlen für die Variante ohne IDEC nicht zur Bewertung genutzt. Tabelle 6.3 ist zu entnehmen, dass unter Verwendung einer maximalen Breite von 96 die Variante mit VAE und herkömmlichen LSTM ohne Verwendung von IDEC beste externe Kennzahlen aufweist. Daher wird das LSTM-VAE ohne IDEC als die beste Variante für einen Datensatz mit den in Kapitel 3.1 beschriebenen Eigenschaften bewertet.

## 6.4 Anwendung auf reale Daten

Um sicherzustellen, dass die künstlichen Daten einen angemessenen Ersatz für die realen Daten darstellen, werden die Ergebnisse von Kapitel 6.3 durch Anwendung der realen Daten validiert. Das Training erfolgt mit einer maximalen Segmentbreite von 96.

ConvLSTM	Nein	Nein	Ja	Ja	Nein
VAE	Ja	Nein	Ja	Nein	Ja
IDEC	Ja	Ja	Ja	Ja	Nein
Epochen	10	10	10	10	10
PK	0,741	0,584	<b>0,825</b>	0,460	(0,216)
FS	24	<b>12</b>	55	42	(2)

Durch IDEC dupliziert sich die Anzahl an Epochen.

Tabelle 6.4: Interne Kennzahlen unter Anwendung verschiedener Varianten mit einer maximalen Breite von 96 für reale Daten

Weil keine Labels zur Verfügung stehen, wird die Clusteringqualität durch einen manuellen Abgleich der Segmente, die einem Cluster mit hoher Wahrscheinlichkeit zugeordnet werden, mit den optischen Erscheinungsbildern der Phänomene repräsentierenden Zeitabschnitte aus Tabelle 3.1 bewertet. Für jede Architekturvariante wird für jedes Cluster versucht, ein wiederkehrendes Muster zu erkennen. In den nachfolgenden Tabellen werden die Cluster beschrieben. Die Beschreibungen enthalten teilweise Höhenangaben, wobei ein hoher Höhenbereich eine Höhe von 10 bis 15 km beschreibt, ein mittlerer 5 bis 10 km und ein niedriger 0 bis 5 km. Mit Ampelfarben wird bewertet, ob ein Cluster ein gemeinsames Muster in den zugeordneten Segmenten aufweist.

- Ein gemeinsames Muster ist unter den zugeordneten Elementen erkennbar.
- Mehrere Muster sind unter den zugeordneten Elementen erkennbar.
- Kein wiederkehrendes Muster ist unter den zugeordneten Elementen erkennbar.

Cluster	Beschreibung	Reinheit
0	Eine dicke Wolkenschicht in niedriger bis mittlerer Höhe; Zusätzlich manchmal Wasserwolken in niedriger Höhe	
1	Wasserwolken in Bodennähe	
2	Grenzschicht-Aerosol; Manchmal mit einer über der Grenzschicht liegenden Wasserwolke	
3	Cirruwsolken in mittlerer Höhe	
4	Gleichzeitig Grenzschicht- und Ferntransport-Aerosole	
5	Cirruwsolken in mittlerer Höhe	
6	Schmale Cirruswolke in hoher Höhe	
7	Niederschlag	

Tabelle 6.5: Beschreibung der Cluster bei Anwendung eines LSTM-VAEs

Cluster	Beschreibung	Reinheit
0	Ausgeprägte Aerosol-Schichten mit Cirruwsolken in mittlerer bis hoher Höhe	
1	Niederschlag	
2	Dicke Wolkenschichten mit starker Rückstreuung in niedriger Höhe	
3	Grenzschicht-Aerosol; Manchmal Wasserwolken inklusive	
4	Unterschiedliche Wolken in niedriger bis mittlerer Höhe	
5	Nebel	
6	Gleichzeitig Grenzschicht- und Ferntransport-Aerosole	
7	Grenzschicht-Aerosol mit schmalen über Grenzschicht liegendem Ferntransport-Aerosol; Manchmal Wasserwolken inklusive	

Tabelle 6.6: Beschreibung der Cluster bei Anwendung eines LSTM-AEs

Cluster	Beschreibung	Reinheit
0	(Leeres Cluster)	
1	(Segmente mit niedriger Detektionsgrenze)	
2	(Starke Verschiebung der Detektionsgrenze in einem Segment)	
3	Gleichzeitig Grenzschicht- und Ferntransport-Aerosole; Zusätzlich manchmal verschiedene Wolkentypen	
4	Grenzschicht-Aerosol und Cirruswolken in mittlerer Höhe	
5	Ausgeprägte Ferntransport-Aerosol-Schicht mit dicken Wolkenschichten in mittlerer Höhe	
6	Ferntransport-Aerosol mit Cirruswolken in hoher Höhe	
7	Grenzschicht-Aerosol mit starkem Anteil von Ferntransport-Aerosolen; Zusätzlich manchmal Wasserwolken	

Tabelle 6.7: Beschreibung der Cluster bei Anwendung eines ConvLSTM-VAEs

Cluster	Beschreibung	Reinheit
0	(Niedrige Detektionsgrenze); Häufig Wasserwolken in niedriger Höhe	
1	(Schmale Segmente); Ausgeprägte Ferntransport-Aerosol-Schicht mit Wolken in niedriger bis mittlerer Höhe	
2	Cirruswolken in hoher Höhe	
3	(Detektionsgrenze in niedriger bis mittlerer Höhe)	
4	(Sehr schmale Segmente)	
5	Ferntransport-Aerosol mit Cirruswolken in mittlerer Höhe	
6	Grenzschicht-Aerosol; Zusätzlich manchmal Wasserwolken oder Nebel	
7	Grenzschicht-Aerosol; Zusätzlich manchmal Wasserwolken	

Tabelle 6.8: Beschreibung der Cluster bei Anwendung eines ConvLSTM-AEs

Cluster	Beschreibung	Reinheit
0	Cirruswolken mit Fallstreifen in niedriger bis mittlerer Höhe	
1	Wasserwolken	
2	Gleichzeitig Grenzschicht- und Ferntransport-Aerosole	
3	Dicke Schichten an Ferntransport-Aerosolen mit schmalen Wolken-Streifen in mittlerer Höhe; Zusätzlich manchmal Wasserwolken in niedriger Höhe	
4	Ferntransport-Aerosol mit Wasserwolken im Höhenbereich vom planetaren Grenzbereich	
5	Ferntransport-Aerosol mit Cirruwsolken in mittlerer Höhe	
6	Grenzschicht-Aerosol mit Cirruwsolken in hoher Höhe	
7	Niederschlag	

Tabelle 6.9: Beschreibung der Cluster bei Anwendung eines LSTM-VAEs ohne IDEC

Während in Kapitel 6.3 der ConvLSTM-VAE externe Kennzahlen vom gleichen Niveau wie der LSTM-AE aufweist, sind unter Anwendung der realen Daten deutliche Unterschiede zu verzeichnen. Der ConvLSTM-VAE schneidet deutlich schlechter ab. Ein LSTM-VAE mit und ohne IDEC werden am besten bewertet. Unter strenger Betrachtung der in den Tabellen 6.5 und 6.9 vorgenommenen Bewertungen weist das LSTM-VAE ohne IDEC die besten Clusteringergebnisse auf.

## 6.5 Diskussion

Im Folgenden wird versucht, die Ergebnisse aus Kapitel 6.3 und 6.4 zu erklären. Erwartet wurde, dass die Varianten mit einem VAE im Vergleich zu den Autoencodern bessere Clusteringergebnisse liefern. Hierbei ist auf Abbildung 2.11 hinzuweisen. Ein intrinsisches Clustering ist bereits während der ersten Phase von IDEC gegeben. Eine Generalisierung wird durch die trichterförmige Autoencoder-Architektur erzwungen. Über den Regularisierungsfehler wird eine Kraft hinzugeführt, die der Kohäsion dienlich ist, während durch den Rekonstruktionsfehler eine Kraft der Separation besteht. In der Begründung für den Entwurf einer Architektur mit VAE in Kapitel 4.2.2 wird argumentiert, dass bei der Eingabe eines Exemplars aus dem Datensatz für die latente Repräsentation statt einer

Punktschätzung ein potenzieller Merkmalsraum vorhergesagt wird, von welchem eine Rekonstruktion der Eingabe ähnelt. Zu Beginn der Einführung der Clusteringschicht von IDEC ist also bereits initial eine Clustering-freundliche Struktur der latenten Repräsentation dadurch gewährleistet, dass nahe Punkte ähnliche Eigenschaften aufweisen müssen. Die Erwartung, dass ein VAE signifikant bessere Clusteringergebnisse liefert, wurde unter Anwendung der künstlichen Daten zu einem gewissen Maß erfüllt, da eine deutliche Differenz zwischen den Kennzahlen zu verzeichnen ist. Unter Anwendung der realen Daten ist ein Vorteil des VAEs kaum noch erkennbar. Hierbei ergibt sich die Fragestellung, ob das Potenzial vom VAE aufgrund der Kombination mit IDEC nicht ausgeschöpft werden kann. Durch IDEC wird ein zusätzliches Ziel in der Verlustfunktion definiert, sodass ein ungünstiges Spannungsfeld zwischen den Zielen entstehen könnte. In Folge dieser Fragestellung wurden zusätzliche Messungen für die Verwendung eines VAEs ohne IDEC durchgeführt. Hierbei zeigt sich, dass die mehrfachen Ziele tatsächlich ein Problem darstellen können, da die berechneten externen Kennzahlen für die Anwendung auf künstliche Daten eine signifikant bessere Bewertung darstellen.

Dass die Verwendung eines ConvLSTMs eher ungewünschte Ergebnisse erzielt, kann der Frage geschuldet sein, ob überhaupt geeignete raum-temporale Merkmale bzw. Filter existieren. Das ConvLSTM erwartet in seinem originalen Verwendungszweck eine zweidimensionale Eingabe (z.B. ein Videoframe) pro Zeitpunkt, während die DWD-Daten lediglich eine eindimensionale Eingabe pro Zeitpunkt liefern. Ein weiterer Einflussfaktor kann die Konfiguration der Schichten wie z.B. die Wahl der Filtergröße und Filteranzahl sein. Bei Betrachtung der Tabellen 6.7 und 6.8 wird deutlich, dass in manchen Clustern Segmente Merkmale aufweisen, die im Zusammenhang mit der Segmentlänge oder Detektionsgrenze stehen. Dass der Segmentlänge eine höhere Bedeutung zukommt, kann den in Kapitel 2.2 und 2.3 genannten Guckloch-Verbindungen geschuldet sein. Die höhere Bedeutung der Detektionsgrenze kann ihre Ursache in der Existenz eines raum-temporalen Musters bezüglich einer Verschiebung dieser Grenze haben. Die Information über die Detektionsgrenze wird insbesondere durch die Datenaufbereitung eingefügt, bei welcher Bereiche oberhalb der Grenze bereinigt werden. Hierbei ist anzumerken, dass während der Trainings in Kapitel 6 keine Maskierung gemäß Kapitel 4.1 genutzt wurde. Eine solche Maskierung würde das genannte Verhalten voraussichtlich zusätzlich verstärken.

# 7 Fazit

## 7.1 Zusammenfassung

In dieser Arbeit wurde gezeigt, dass eine zufallsbasierte variable Segmentierung statt einer statischen Segmentierung eine Mustererkennung mittels Clustering in kontinuierlichen Sequenzen mit Mustern variabler Dauer und zufälligem Auftritt begünstigen kann. Eine interessante Beobachtung ist hierbei, dass die Wahl unterschiedlicher statischer Breiten zu großen Unterschieden in der Clusteringqualität führt, während bei Wahl unterschiedlicher maximaler Breiten für eine variable Segmentierung tendenziell eine Clusteringqualität auf vergleichbarem Niveau zu verzeichnen ist. Die Erkenntnisse indizieren eine Bestätigung der These, dass eine variable zufällige Segmentierung für einen Trade-off zwischen der Erfassung zusammenhängender Muster und der Separation unterschiedlicher Muster sorgt. Darüber hinaus stellt die variable Segmentierung einen sinnvollen Ersatz für eine gezielte Segmentierung dar, sodass eine einer gezielten Segmentierung bezogene Bewertung der in dieser Arbeit entworfenen Varianten von LSTM-Autoencodern vorgenommen werden kann.

Sowohl unter Anwendung der künstlichen als auch der realen Daten hat sich gezeigt, dass unter den Architekturvarianten ein VAE mit klassischen LSTMs zur Clustering-orientierten Kodierung variabel langer Segmente am besten geeignet ist. Darüber hinaus weist eine Kombination von VAE mit IDEEC keine besseren Clusteringergebnisse als ein VAE mit anschließendem K-Means auf.

## 7.2 Ausblick

In den Experimentdurchführungen dieser Arbeit wurde eine starke Komprimierung der räumlichen Dimension und eine schwache Komprimierung der zeitlichen Dimension genutzt. Domänen-Experten verwenden für eine Analyse andere Skalierungen. Eine Komprimierung der räumlichen Dimension durch eine Skalierung kann teilweise durch ein

tiefere CNN ersetzt werden. Die Zeitdimension sollte stärker komprimiert werden, so dass breitere zusammengehörige Strukturen mit der genutzten maximalen Segmentbreite besser erfasst werden können.

Die Verwendung von IDEC hat sich gegenüber einem VAE mit anschließendem K-Means nicht als vorteilhaft herausgestellt. Eine Anwendung des Deep-Clustering-Verfahrens Va-DE ist aufgrund der in der Veröffentlichung [Jiang u. a., 2017] gezeigten Kennzahlen interessanter.

Darüber hinaus wurden neben dem gewählten Ansatz für den Flaschenhals weitere Varianten entworfen, welche untersucht werden könnten (siehe Kapitel 4.2.1).

Das Labeln einer Teilmenge der DWD-Daten würde die Anwendung halb-überwachter Methoden ermöglichen. So könnte beispielsweise den Ideal-Cluster repräsentierenden Segmenten eine höhere Gewichtung für eine Cluster-Zentroid-Ermittlung zugewiesen werden. Als auf die Arbeit aufbauende Forschungsfrage ist eine Untersuchung unter Anwendung von Methoden der *Change Point Detection* zu empfehlen. In diesem Forschungsbereich wurden bereits verschiedene Deep-Learning-basierte Modelle angewandt. Beispiele sind unter anderem [Lee u. a., 2018] und [Deldari u. a., 2021]. Hierbei werden schmale nebeneinanderliegende Fenster über den Datensatz geschoben und ein Verfahren angewandt, um eine relevante Diskrepanz zwischen den Repräsentationen der Fenster festzustellen. In letzterem Beispiel wird *Contrastive Learning* angewandt, wobei die Erkennung einer Diskrepanz zwischen zwei Repräsentationen unterstützt wird, indem Positiv- und Negativbeispiele erzeugt werden. Die mittels solcher Methoden entstehenden Segmente können in einer nachfolgenden Phase mithilfe der Implementierung dieser Arbeit Clustern zugeteilt werden.

# Literaturverzeichnis

- [Bayer 2015] BAYER, Justin: *Learning Sequence Representations*, TU München, Dissertation, 2015
- [Blei u. a. 2017] BLEI, David M. ; KUCUKELBIR, Alp ; MCAULIFFE, Jon D.: Variational Inference: A Review for Statisticians. In: *Journal of the American Statistical Association* 112 (2017), Apr, Nr. 518, S. 859–877
- [Burkov 2019] BURKOV, Andriy: *Machine Learning kompakt*. 1. mitp-Verlag, 2019. – ISBN 9783958459960
- [Chollet 2021] CHOLLET, Francois: *Deep Learning with Python, Second Edition*. Manning Publications, 2021. – ISBN 9781617296864
- [Christopher D. Manning und Schütze 2008] CHRISTOPHER D. MANNING, Prabhakar R. ; SCHÜTZE, Hinrich: *Introduction to Information Retrieval*. Cambridge University Press, 2008
- [Deldari u. a. 2021] DELDARI, Shohreh ; SMITH, Daniel V. ; XUE, Hao ; SALIM, Flora D.: *Time Series Change Point Detection with Self-Supervised Contrastive Predictive Coding*. S. 3124–3135. In: *Proceedings of the Web Conference 2021*. New York, NY, USA : Association for Computing Machinery, 2021
- [Doersch 2021] DOERSCH, Carl: *Tutorial on Variational Autoencoders*. 2021
- [Gandraß 2019] GANDRASS, Niels: *Merkmalsextraktion durch Methoden des unüberwachten maschinellen Lernens zur Klassifikation von Aerosol-Rückstreuprofilen aus LIDAR-Ceilometern*. 2019
- [Gers und Schmidhuber 2000] GERS, F.A. ; SCHMIDHUBER, J.: Recurrent nets that time and count. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium* Bd. 3, 2000, S. 189–194 vol.3

- [Gers u. a. 1999] GERS, F.A. ; SCHMIDHUBER, J. ; CUMMINS, F.: Learning to forget: continual prediction with LSTM. In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)* Bd. 2, 1999, S. 850–855 vol.2
- [Graves 2014] GRAVES, Alex: *Generating Sequences With Recurrent Neural Networks*. 2014
- [Guo u. a. 2017] GUO, Xifeng ; GAO, Long ; LIU, Xinwang ; YIN, Jianping: Improved Deep Embedded Clustering with Local Structure Preservation. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, 2017*, S. 1753–1759
- [Guojun Gan 2007] GUOJUN GAN, Jianhong W.: *Data Clustering: Theory, Algorithms, and Applications*. Society for Industrial and Applied Mathematics (SIAM) and American Statistical Association (ASA), 2007. – ISBN 978-0-898716-23-8
- [Hassani und Seidl 2017] HASSANI, Marwan ; SEIDL, Thomas: Using internal evaluation measures to validate the quality of diverse stream clustering algorithms. In: *Vietnam Journal of Computer Science* 4 (2017), Aug, Nr. 3, S. 171–183. – ISSN 2196-8896
- [Heckert u. a. 2013] HECKERT, N. ; FILLIBEN, James ; CROARKIN, C ; HEMBREE, B ; GUTHRIE, William ; TOBIAS, P ; PRINZ, J: *NIST/SEMATECH e-Handbook of Statistical Methods*. NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, 2013
- [Hochreiter und Schmidhuber 1997] HOCHREITER, Sepp ; SCHMIDHUBER, Jürgen: Long Short-Term Memory. In: *Neural Comput.* 9 (1997), nov, Nr. 8, S. 1735–1780. – ISSN 0899-7667
- [Ian Goodfellow 2016] IAN GOODFELLOW, Aaron C.: *Deep Learning*. MIT Press, 2016. – ISBN 9780262035613
- [Jiang u. a. 2017] JIANG, Zhuxi ; ZHENG, Yin ; TAN, Huachun ; TANG, Bangsheng ; ZHOU, Hanning: Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence, AAAI Press, 2017 (IJCAI'17)*, S. 1965–1972. – ISBN 9780999241103
- [Kingma und Welling 2014] KINGMA, Diederik P. ; WELLING, Max: Auto-Encoding Variational Bayes. In: *2nd International Conference on Learning Representations*,

- ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014
- [Lee u. a. 2018] LEE, Wei-Han ; ORTIZ, Jorge ; KO, Bongjun ; LEE, Ruby: *Time Series Segmentation through Automatic Feature Learning*. 2018
- [Lutz 2013] LUTZ, Mark: *Learning Python, 5th Edition*. O'Reilly Media, Inc., 2013
- [van der Maaten und Hinton 2008] MAATEN, Laurens van der ; HINTON, Geoffrey: Visualizing Data using t-SNE. In: *Journal of Machine Learning Research* 9 (2008), Nr. 86, S. 2579–2605
- [Min u. a. 2018] MIN, Erxue ; GUO, Xifeng ; LIU, Qiang ; ZHANG, Gen ; CUI, Jianjing ; LONG, Jun: A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture. In: *IEEE Access* 6 (2018), S. 39501–39514
- [Osborne 2013] OSBORNE, Jason W.: *Best Practices in Data Cleaning*. SAGE Publications, Inc., 2013
- [OTT HydroMet Fellbach GmbH 2021] OTT HydroMet Fellbach GmbH (Veranst.): *User Manual Luft CHM 15k Ceilometer*. R17/R18. 2021
- [Salvador und Chan 2007] SALVADOR, Stan ; CHAN, Philip: Toward Accurate Dynamic Time Warping in Linear Time and Space. In: *Intell. Data Anal.* 11 (2007), oct, Nr. 5, S. 561–580. – ISSN 1088-467X
- [Shi u. a. 2015] SHI, Xingjian ; CHEN, Zhourong ; WANG, Hao ; YEUNG, Dit-Yan ; WONG, Wai-kin ; WOO, Wang-chun: Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. Cambridge, MA, USA : MIT Press, 2015 (NIPS'15), S. 802–810
- [Shokoohi-Yekta u. a. 2017] SHOKOOHI-YEKTA, Mohammad ; HU, Bing ; JIN, Hongxia ; WANG, Jun ; KEOGH, Eamonn: Generalizing DTW to the multi-dimensional case requires an adaptive approach. In: *Data Mining and Knowledge Discovery* 31 (2017), Jan, Nr. 1, S. 1–31. – ISSN 1573-756X
- [Vinh u. a. 2010] VINH, Nguyen X. ; EPPS, Julien ; BAILEY, James: Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. In: *J. Mach. Learn. Res.* 11 (2010), dec, S. 2837–2854. – ISSN 1532-4435

- [Xie u. a. 2016] XIE, Junyuan ; GIRSHICK, Ross ; FARHADI, Ali: Unsupervised Deep Embedding for Clustering Analysis. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, JMLR.org, 2016 (ICML'16), S. 478–487
- [Yang u. a. 2017] YANG, Bo ; FU, Xiao ; SIDIROPOULOS, Nicholas D. ; HONG, Mingyi: Towards K-Means-Friendly Spaces: Simultaneous Deep Learning and Clustering. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, JMLR.org, 2017 (ICML'17), S. 3861–3870

# A Anhang

## A.1 Herleitungen

$$\begin{aligned}\mathbb{E}_a[f(a) + b] &= \sum_a (f(a) + b)p(a) && \text{Definition } \mathbb{E}[X] \\ &= \left(\sum_a f(a)p(a)\right) + \left(\sum_a bp(a)\right) && \text{Distributivgesetz} \\ &= \left(\sum_a f(a)p(a)\right) + b\left(\sum_a p(a)\right) && \text{Distributivgesetz} \\ &= \left(\sum_a f(a)p(a)\right) + b && \text{(Summe ergibt 1)} \\ &= \mathbb{E}_a[f(a)] + b && \text{Definition } \mathbb{E}[X]\end{aligned}$$

Abbildung A.1: Herleitung  $\mathbb{E}_a[f(a) + b] = \mathbb{E}_a[f(a)] + b$

$$\begin{aligned}\log \frac{a}{b} &= \log a - \log b && \text{Log-Quotientregel} \\ &= \log a + (-\log b) \\ &= (-\log b) + \log a \\ &= -(\log b - \log a) && \text{Distributivgesetz} \\ &= -\log \frac{b}{a} && \text{Log-Quotientregel}\end{aligned}$$

Abbildung A.2: Herleitung  $\log \frac{a}{b} = -\log \frac{b}{a}$

# Glossar

$D_{KL}(P \parallel Q)$

*Siehe* KL-Divergenz.

$\mathbb{E}[X]$

Der Erwartungswert der Zufallsvariablen  $X$  ist ein gewichteter Durchschnitt der möglichen Werte, die  $X$  annehmen kann. Jeder Wert ist gewichtet hinsichtlich seiner Auftrittswahrscheinlichkeit. So gilt für eine diskrete Zufallsvariable

$$\mathbb{E}_{x \sim P}[f(x)] = \sum_x P(x)f(x),$$

wobei  $P(x)$  die Massenfunktion bezeichnet, und für eine stetige Zufallsvariable

$$\mathbb{E}_{x \sim p}[f(x)] = \int p(x)f(x) dx,$$

wobei  $p(x)$  die Dichtefunktion bezeichnet. (vgl. [Ian Goodfellow, 2016], Kap. 3.8)

A-posteriori-  
Wahrscheinlichkeitsverteilung

Die Kenntnis über einen tatsächlichen Parameterwert wird nach Beobachtung der Daten mit der A-posteriori-Verteilung  $p(z|x)$  repräsentiert. Die Kenntnis sollte in der bayesschen Schätzung durch Beobachtung der Daten genauer werden bzw. eine niedrige Shannon-Entropie gewinnen. Die Berechnung der Verteilung erfolgt in der bayesschen Schätzung über Anwendung des Satz von Bayes. (vgl. [Ian Goodfellow, 2016], Kap. 5.6)

A-priori-Wahrscheinlichkeitsverteilung Die Kenntnis über einen tatsächlichen Parameterwert wird vor Beobachtung der Daten mithilfe einer Festlegung einer A-priori-Verteilung  $p(z)$  repräsentiert. Die Kenntnis sollte in der bayesschen Schätzung ungenau sein bzw. eine hohe Entropie aufweisen. (vgl. [Ian Goodfellow, 2016], Kap. 5.6)

Bayessche Statistik In der klassischen Statistik wird Wahrscheinlichkeit als Häufigkeit der Auftretens eines Ereignisses interpretiert. In der Bayesschen Statistik wird Wahrscheinlichkeit als der Grad der Gewissheit über das Auftreten eines Ereignisses basierend auf einem persönlichen Kenntnisstand interpretiert. Eine zuvor festgelegte Kenntnis über eine Variable wird hierbei mithilfe der Beobachtung von Daten kombiniert, um den Kenntnisstand über die Variable zu verfeinern. (vgl. [Ian Goodfellow, 2016], Kap. 5.6)

Bedingte Wahrscheinlichkeit „Die bedingte Wahrscheinlichkeit  $P(X = x|Y = y)$  gibt die Wahrscheinlichkeit dafür an, dass die Zufallsvariable  $X$  einen bestimmten Wert  $x$  annimmt, unter der Voraussetzung, dass eine andere Zufallsvariable  $Y$  einen bestimmten Wert  $y$  besitzt.“ ([Burkov, 2019], Kap. 2.4) Sie kann folgendermaßen berechnet werden:

$$P(y|x) = \frac{P(y, x)}{P(x)}.$$

**BPTT** *Backpropagation Through Time* beschreibt den Fehlerückfluss in einem zeitlich aufgefalteten RNN. Für die Berechnung des Gradienten wird ein Fehler über die rekursive Verbindung vom Zeitpunkt  $t$  zum Zeitpunkt  $k$ , wobei  $k < t$  mithilfe der Kettenregel zurückgegeben:

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}}.$$

**Channel** Im Kontext dieser Arbeit beschreibt ein Channel eine Eigenschaft eines atomaren Datenpunktes. In einem RGB-Bild stellt beispielsweise ein Pixel einen atomaren Datenpunkt mit drei Channels dar.

**CNN** „Convolutional neural network“ beschreibt ein neuronales Netzwerk, welches Faltungen statt Matrixmultiplikationen in mindestens einer seiner Schichten nutzt. (vgl. [Ian Goodfellow, 2016], Kap. 9)

**Datenaugmentation** Mit Datenaugmentation werden Techniken beschrieben, mit welcher Transformationen an Originaldaten vorgenommen werden können, um die Vielfalt des Datensatzes zu vergrößern.

**Euklidische Distanz** Die euklidische Distanz berechnet den Abstand zwischen zwei  $n$  langen Vektoren  $p$  und  $q$  und ist gegeben durch

$$\sqrt{\sum_{i=1}^n (q_i - p_i)^2}.$$

(vgl. [Guojun Gan, 2007], Kap. 6.2.1)

Faltung

Die diskrete Faltung (nicht zu verwechseln mit der „Auffaltung“ im RNN) ist gegeben durch:

$$(f * g)(n) = \sum_{k \in D} f(k)g(n - k),$$

wobei  $f$  einen Filter und  $D$  einen diskreten Werteraum bezeichnet. Im multidimensionalen Fall ergibt sich

$$(f * g)(n, m) = \sum_k \sum_l f(k, l)g(n - k, m - l).$$

(vgl. [Ian Goodfellow, 2016], Kap. 9.1)

Fehlerrückfluss

Der Fehlerrückfluss (engl. *Backpropagation*) beschreibt den folgenden Prozess: Für die Parameter eines neuronalen Netzes wird der jeweilige Einfluss auf das Ergebnis einer Fehlerfunktion berechnet (Gradient). Durch eine dem Einfluss und der als Hyperparameter festgelegten Lernrate entsprechende Anpassung der Parameter wird der sogenannte Gradientenabstieg vollzogen. Auf diese Weise kann langfristig die Fehlerrate minimiert werden.

Gerichtetes graphisches Modell

Um eine Multivariate Wahrscheinlichkeitsverteilung zu repräsentieren, kann sie gemäß der Produktregel der bedingten Wahrscheinlichkeiten in mehrere Faktoren aufgeteilt werden. Diese Faktorisierung kann über einen Graphen mit Knoten und Kanten abgebildet werden. Ein eine Zufallsvariable  $C$  repräsentierender Knoten mit einer eingehenden Kante von der Zufallsvariablen  $B$  stellt dabei den Faktor  $P(C|B)$  dar. Ein eine Zufallsvariable  $A$  repräsentierender Knoten stellt den Faktor  $P(A)$  dar. (vgl. [Ian Goodfellow, 2016], Kap. 3.14)

**Gradient** Mit der Ableitung von  $f(x)$  kann die (positive oder negative) Steigung im Punkt  $x$  berechnet werden. Im Falle einer Funktion mit mehreren Variablen beschreibt der Gradient einen Vektor mit den partiellen Ableitungen für die jeweiligen Variablen.

**KL-Divergenz** Die KL-Divergenz ist gegeben durch

$$D_{KL}(P \parallel Q) = \mathbb{E}_{x \sim P} \left[ \log \frac{P(x)}{Q(x)} \right]$$

Mit ihr kann der Informationsverlust, der durch die Verwendung von  $Q$  statt  $P$  entsteht, gemessen werden (siehe Shannon-Entropie). Hierbei wird für jedes Ereignis  $x$  die Differenz des Informationsgehalts gewichtet mit der erwarteten Wahrscheinlichkeit  $p(X = x)$  berechnet. Die KL-Divergenz hat die Eigenschaft  $\geq 0$  zu sein. Wenn sie 0 ergibt, entspricht  $P$   $Q$ . (vgl. [Ian Goodfellow, 2016], Kap. 3.13)

**Klasse** Klasse im Sinne einer festgelegten Klassifizierung

**MSE** Der mittlere quadratische Fehler kann als Verlustfunktion zum Vergleich von tatsächlicher und erwarteter Ausgabe des Modells verwendet werden:

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \bar{Y}_i)^2,$$

wobei  $n$  die Größe eines Batches ist. (vgl. [Ian Goodfellow, 2016], Kap. 5.1.4)

**Multivariate Wahrscheinlichkeitsverteilung** Eine multivariate Verteilung  $P(X = x, Y = y)$  benennt die Wahrscheinlichkeit, dass  $X = x$  und  $Y = y$  gleichzeitig zutreffen.“ (vgl. [Ian Goodfellow, 2016], Kap. 3.3.1)



Satz von Bayes

Der Satz von Bayes besagt

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

und ergibt sich durch die Definitionen Bedingte Wahrscheinlichkeit und Produktregel der bedingten Wahrscheinlichkeiten. (vgl. [Burkov, 2019], Kap. 2.4)

Shannon-Entropie

Die Shannon-Entropie  $H(x) = -\mathbb{E}_{x \sim P}[\log P(x)]$  gibt den Erwartungswert für den Informationsgehalt der Zufallsvariablen  $X$  an. (vgl. [Ian Goodfellow, 2016], Kap. 3.13)

Wahrscheinlichkeitsverteilung

Eine Wahrscheinlichkeitsverteilung  $P(X)$  ordnet jedem möglichen Wert einer Zufallsvariablen eine Wahrscheinlichkeit zu.

Zufallsvariable

Eine Zufallsvariable  $X$  ist eine Variable, deren Werte mit bestimmten Wahrscheinlichkeiten auftreten.  $P(X = x)$  gibt die Wahrscheinlichkeit für den Wert  $x$  an. Die Summe der Wahrscheinlichkeiten ergibt 1. Die Zuordnung von Werten zu Wahrscheinlichkeiten  $P(X)$  kann bei einer diskreten Zufallsvariable mit einer Wahrscheinlichkeitsmassenfunktion und bei einer stetigen Zufallsvariable mit einer Wahrscheinlichkeitsdichtefunktion abgebildet werden. (vgl. [Burkov, 2019], Kap. 3.2)

## Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

\_\_\_\_\_

Ort                      Datum                                            Unterschrift im Original