Maturitätsarbeit an der Kantonsschule Uster

Fachschaft Informatik

Vorhersage der Ozonkonzentration mithilfe von neuronalen Netzen

Diego Muzzarelli

08. Januar 2025

BetreuerIn: Julia Imhof
ExpertIn: Theresa Luternauer

Abstract

Ozon zählt zu den bedeutendsten bodennahen Luftschadstoffen, da hohe Konzentrationen ein ernstzunehmendes Gesundheitsrisiko darstellen. Die Vorhersage der Ozonkonzentration hat daher in den letzten Jahren stark an Bedeutung gewonnen. Traditionell basieren Prognosen auf Modellen, die chemische und physikalische Prozesse simulieren oder Trends in den Daten durch statistische Methoden identifizieren.

Seit etwa einem Jahrzehnt werden zunehmend auch Long Short-Term Memory (LSTM)-Modelle, eine Art neuronaler Netze, für die Vorhersage von Schadstoffkonzentrationen eingesetzt.
Diese Arbeit untersucht die Anwendbarkeit von LSTM-Modellen zur Vorhersage der Ozonkonzentration in der Stadt Zürich. Verschiedene LSTM-Modelle wurden entwickelt und trainiert, um die optimalen Konfigurationen der Modellarchitektur- und Trainingsparameter zu ermitteln. Die Evaluation erfolgte sowohl quantitativ anhand von Fehlermetriken wie MSE, MAE und dem Pearson-Koeffizienten als auch qualitativ durch die Analyse der Vorhersagekurven.

Die Ergebnisse zeigen, dass die Modelle zeitliche Trends der Ozonkonzentration erfolgreich erfassen. Dabei lieferten kleinere Modelle mit nur einem LSTM-Layer die präzisesten Vorhersagen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Mit dem Klimawandel und der Zunahme globaler Treibhausgasemissionen, welche seit dem 20. Jahrhundert rasant angestiegen sind (Ritchie et al., 2020), hat in den letzten 100 Jahren auch die Konzentration von Schadstoffen in der Luft, wie etwa Ozon (O_3) oder Feinstaub (PM_{10} und $PM_{2.5}$), zugenommen. Sie erreichte in den späten 1990er- und 2000er-Jahren Maximalwerte und ist seither wieder etwas zurückgegangen (Our World in Data, 2022).
Hohe Luftschadstoffkonzentrationen, welche vor allem in Grossstädten vorzufinden sind, können den Menschen, die dort leben, erhebliche Gesundheitsschäden zufügen. Dazu gehören Krankheiten wie Asthma, Herz-Kreislauf-Erkrankungen oder Lungenkrebs (Bundesamt für Umwelt, 2024; European Environment Agency, 2024). Zu den wichtigsten Schadstoffen zählen dabei Ozon (O_3), Schwefeldioxid (SO_2), Kohlenmonoxid (CO), Stickstoffdioxid (NO_2) sowie Feinstaubpartikel (PM_{10} und $PM_{2.5}$). Die Schweiz hat im Vergleich zum globalen Durchschnitt eine deutlich höhere Luftqualität (IQAir, 2024). Nichtsdestotrotz werden die von der WHO (World Health Organization) gesetzten Immissionsgrenzwerte für Schadstoffe (World Health Organization, 2021) teils überschritten (Bundesamt für Umwelt, 2024).
Aus diesen Gründen ist es wichtig, die Luftqualität bzw. die Konzentrationen von gesundheitsschädlichen Gasen und Feinstaub zu messen und vorhersagen zu können. Insbesondere hat die Fähigkeit, präzise Vorhersagen der Ozonkonzentration zu erstellen, in den letzten Jahren an Wichtigkeit gewonnen (Luo et al., 2024).

Das Ziel dieser Arbeit ist die Untersuchung der Anwendbarkeit von *Long Short-Term Memory* (LSTM)-Modellen zur Vorhersage der Ozonkonzentration (O_3) in der Stadt Zürich. Hierfür werden verschiedene Modellarchitekturen getestet, und die Qualität der Vorhersagen wird sowohl quantitativ als auch qualitativ evaluiert. Es soll somit herausgefunden werden, ob es den Modellen gelingt, die unterliegenden Trends der Konzentrationsentwicklung von Ozon zu erlernen.

LSTMs sind eine Art von neuronalen Netzen (Nigam, 2018), die für die zeitliche Regression, also die Vorhersage zeitlich veränderlicher Daten, verwendet werden.

Stand der Forschung

Schadstoffkonzentrationen werden gegenwärtig grösstenteils mithilfe von sogenannten *CTM* (engl. Chemical Transport Model)-Simulationen oder mit statistischen Methoden vorhergesagt.

CTM sind Programme, welche numerische Gleichungen lösen, die die Dispersions-, Umwandlungs-, Entstehungs- und Abbauprozesse von Schadstoffen in der Luft beschreiben (Liao Qi, X.L. Pan, Mingm Zhu, Zifa Wang, 2020). Mithilfe der momentanen Schadstoffkonzentrationen, meteorologischer Daten und dieser Gleichungen können die späteren Schadstoffkonzentrationen berechnet werden. Ein verbreitetes CTM ist das *Community Multiscale Air Quality (CMAQ) Model* (Appel et al., 2021).
Diese Modelle können sehr präzise Vorhersagen liefern, haben aber auch einige Nachteile: Ihre Vorhersagen hängen stark von der Genauigkeit der Eingabedaten ab. Ausserdem benötigen diese Simulationen eine hohe Rechenleistung und haben Mühe, aus grossen Datensets komplexe Beziehungen zwischen den verschiedenen Eingabedaten abzuleiten.

Statistische Methoden sind beispielsweise Modelle wie *Auto Regression Integrated Moving Average* (ARIMA) (Wikipedia, 2024b) oder *Multiple Linear Regression* (MLR) (Liao Qi, X.L. Pan, Mingm Zhu, Zifa Wang, 2020; Wikipedia, 2024g). Diese Methoden versuchen, Trends in den Daten zu identifizieren. Anders als CTMs, achten sie nicht auf chemische und physikalische Prozesse in der Luft. Wie CTMs haben auch sie Schwierigkeiten, nichtlineare und komplexe Beziehungen zwischen den Eingabedaten abzubilden.

Um die Limitierungen der erwähnten Methoden zu überwinden bzw. diese Methoden zu ergänzen, wurden in mehreren Studien LSTMs eingesetzt.
Diese zeichnen sich durch ihre Fähigkeit aus, zeitliche Muster in Daten effektiv zu erkennen und zu erlernen.
Bei einer Studie in Heilongjiang, China (Wu et al., 2024), lieferten die LSTM-Modelle die genauesten Vorhersagen für $PM_{2.5}$-Konzentrationen. Ähnliche Ergebnisse wurden in Madrid, Spanien, erzielt, wo LSTMs für die Vorhersage verschiedener Schadstoffkonzentrationen andere Regressionsmodelle wie die lineare Regression übertrafen (Navares & Aznarte, 2020).
LSTMs erwiesen sich aber auch in anderen Feldern, wie dem Aktienmarkt (Gourav Bathla, 2023), und dem Tourismusmanagement (Li & Cao, 2018), als zuverlässige und präzise Vorhersagemethoden.

2 Funktionsweise von LSTM

In diesem Kapitel wird erläutert, wie LSTMs funktionieren, und anhand ihrer inneren Struktur wird verdeutlicht, warum sie besonders gut für Daten mit zeitlichen Mustern geeignet sind.

LSTM-Modelle sind eine Art von neuronalen Netzen, die 1997 erstmals konzipiert wurden (Hochreiter & Schmidhuber, 1997). Neuronale Netze sind Algorithmen, die für maschinelles Lernen eingesetzt werden. Sie sind in ihrer Struktur an dem menschlichen Gehirn orientiert. Sie bestehen aus künstlichen Neuronen, die in Schichten (engl. Layers) angeordnet sind (Fumo, 2017; Nigam, 2018).
Ihr Aufbau wird nun illustriert:

1. Input Layer: Der Input Layer nimmt die Rohdaten in Form eines Vektors als Eingabe. Hierbei wird jedes Element im Vektor von jedem Neuron mit einem Gewicht w multipliziert und aufaddiert. Zu dieser Summe wird ein Wert, der *Bias* genannt wird, addiert. Die resultierende Summe wird in eine sogenannte *Aktivierungsfunktion* eingesetzt, die den Ausgabewert zum nächsten Layer weitergibt.

2. Hidden Layers: Diese sind Layers innerhalb des neuronalen Netzes, die die gleiche Operation wie der Input Layer durchführen mit den Werten, die vom vorigen Layer weitergereicht wurden.

3. Output Layer: Der Output Layer transformiert die Werte des vorangehenden Hidden Layers mit derselben Operation wie die anderen Layer und gibt einen Vektor als Output aus. Der Output Layer bestimmt somit die Outputdimension des neuronalen Netzes.

Zu den bekanntesten und meistgebrauchten Aktivierungsfunktionen gehören *Sigmoid*, *Tanh* und *ReLu* (Wikipedia, 2024a).
Die Layer eines solchen neuronalen Netzes werden auch *Dense-Layer* genannt. Ein solches neuronales Netz, das nur aus Dense-Layer besteht, wird auch *Feed Forward Neural Network* (FNN) genannt. Die Variablen oder Merkmale, die das Modell im Training verwendet, um Muster zu erkennen und Vorhersagen zu treffen, werden *Features* genannt.

Der Prozess des Weiterleitens von Information durch die Layers eines neuronalen Netzes nennt sich *Forward Propagation* (GeeksforGeeks, 2024b).
Forward Propagation von einem Layer $l-1$ zu einem Layer l kann als eine Matrix-Vektor-Operation beschrieben werden. W_l ist die Gewichtungsmatrix, die aus den Gewichtungen aller Neuronen in diesem Layer besteht. a_{l-1} und a_l sind die Outputvektoren der beiden Layer $l-1$ und l. b_l ist der Biasvektor und g ist die Aktivierungsfunktion:

$$a_l = g\left(W_l \cdot a_{l-1} + b_l\right)$$

Ein neuronales Netz wird mit Input-Output-Datenpaaren trainiert. Der Inputvektor wird hier X genannt und der erwünschte Output wird als Y bezeichnet. Der Outputvektor, der vom Modell produziert bzw. vorhergesagt wird, wird $\hat{Y}$ genannt.
Das neuronale Netz passt während des Trainings seine Gewichtungen und Biases so an, dass seine Vorhersagen $\hat{Y}$ möglichst den korrekten Werten Y entsprechen. Dabei wird der Fehler zwischen $\hat{Y}$ und Y mit einer Verlustfunktion $L(Y, \hat{Y})$ berechnet. Es wird dann die erste Ableitung der Verlustfunktion relativ zu jeder Gewichtung w und jedem Bias b berechnet und diese werden dann wie folgt verändert (LeCun et al., 2015; Palma, 2023):

$$w = w - \eta \cdot \frac{dL}{dw}$$

$$b = b - \eta \cdot \frac{dL}{db}$$

η ist die Lernrate und bestimmt, wie stark die Gewichtungen und Biases angepasst werden. Der Prozess des Anpassen der trainierbaren Parameter heisst *Backpropagation* (Wikipedia, 2024c).

Nun wird gezeigt, wie LSTM-Layers aufgebaut sind: LSTMs gehören zu den sogenannten *Recurrent Neural Networks (RNN)* (GeeksforGeeks, 2024a). Sie werden mit Daten trainiert, bei denen die Reihenfolge der Daten eine wichtige Rolle spielt. Anders als simple neuronale Netze haben LSTMs die Möglichkeit, Informationen in einem Langzeit- und in einem Kurzzeitgedächtnis in Form von Vektoren zu speichern. Das Langzeitgedächtnis wird *Cell State* C_t und das Kurzzeitgedächtnis wird *Hidden State* h_t genannt und ist jeweils der Output des Modells für den vorhergehenden Zeitschritt. Diese Gedächtnisse erlauben es den LSTMs, zeitliche Trends und Muster

zu erlernen, da sie sich merken können, wie sich die Daten verändern. Was in diesen Gedächtnissen gespeichert wird und wie der Datenfluss durch ein LSTM aussieht, wird durch vier Gates (engl. für Tore) reguliert (Hamad, 2023). Diese werden nun erläutert:

1. Forget Gate: Nimmt den Hidden State h_{t-1} des vorherigen Zeitschrittes und den Input X_t des aktuellen Zeitschrittes und produziert einen Wert zwischen 0 und 1. Dieser Wert entscheidet, wie viel vom Cell State C_{t-1} für den nächsten Zeitschritt beibehalten wird, indem C_{t-1} mit diesem Wert multipliziert wird.

2. Input Gate: Bestimmt, welche neuen Informationen im Cell State gespeichert werden. Es besteht aus zwei Komponenten:

   a) Tanh Layer: Kreiert einen Vektor mit Werten zwischen -1 und 1 mit derselben Dimension wie C_t

   b) Sigmoid Layer: Kreiert einen Skalierungsvektor mit Werten zwischen 0 und 1 mit derselben Dimension wie C_t

   Die beiden Vektoren, die diese Layer produzieren, werden elementweise multipliziert und werden dann zu C_{t-1} addiert. Dies ergibt dann C_t.

3. Output Gate: Produziert den Output des LSTM-Layers und den Hidden State. Hierbei durchlaufen der Input X_t und h_{t-1} ein Sigmoid Layer. C_t wird durch die Tanh Funktion geleitet. Anschliessend werden die beiden resultierenden Vektoren elementweise multipliziert und ergeben so den Output bzw. den nächsten Hidden State.

Abbildung 2.1 zeigt schematisch den Aufbau eines LSTMs. Die erwähnten Sigmoid und Tanh Layer sind Dense Layer mit den Aktivierungsfunktionen Sigmoid und Tanh. Als Input nehmen sie den Vektor, der aus der Verkettung von h_{t-1} und X_t resultiert. Sie haben ebenfalls einen Bias-Vektor.
Die Gleichungen unten zeigen die Operationen an den verschiedenen Gates (Hong et al., 2023):

$$
\begin{aligned}
f_t &= \sigma\left(W_f \cdot [h_{t-1}, X_t] + b_f\right) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, X_t] + b_i) \\
g_t &= \tanh\left(W_g \cdot [h_{t-1}, X_t] + b_g\right) \\
C_t &= f_t * C_{t-1} + i_t * g_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}, X_t] + b_o) \\
h_t &= o_t * \tanh(C_t)
\end{aligned}
$$

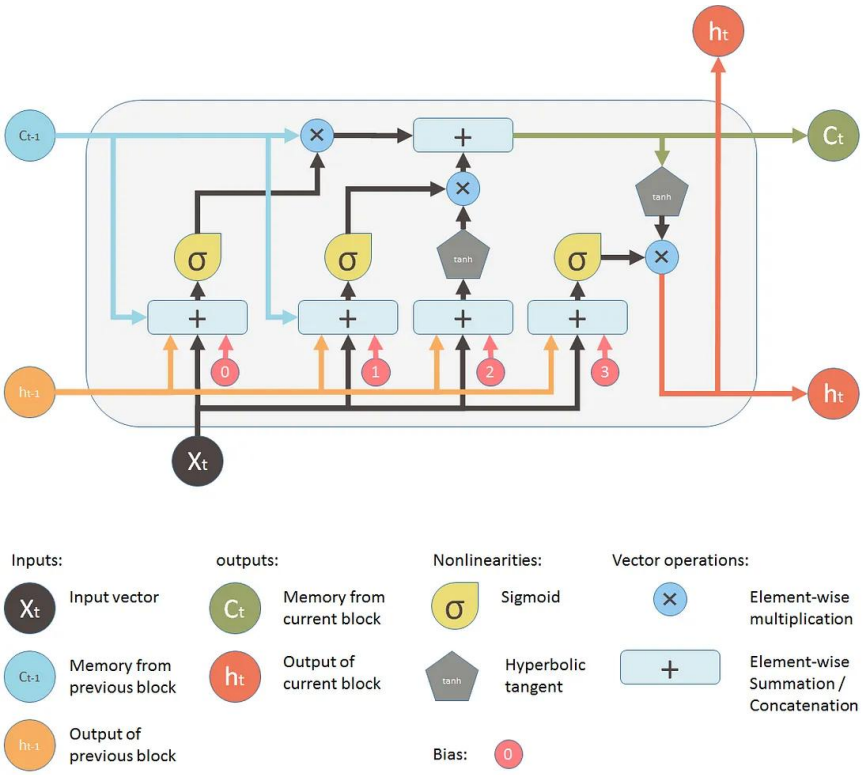


Abbildung 2.1: Diagramm mit dem Aufbau eines LSTM-Layers. Die Legende zeigt die Komponenten und Operationen innerhalb des LSTMs auf. (Quelle: Yan, 2016)

f_t ist der Vektor, den das Forget Gate produziert. i_t und g_t stehen für die Vektoren, die der Sigmoid Layer und der Tanh Layer des Input Gate produzieren. o_t steht für den Output des Sigmoid Layers im Output Gate.
$W_f, W_i, W_g, W_o, b_f, b_i, b_g$ und b_o stehen für die Gewichtungsmatrizen und Biases in den verschiedenen Layers. Das Sternsymbol ($*$) bezeichnet in diesem Zusammenhang die elementweise Multiplikation. Ähnlich wie bei FNNs passen LSTMs ihre trainierbaren Parameter durch Backpropagation an. Der Algorithmus ist in diesem Fall etwas komplexer und wird auch *Backpropagation through Time* (BPTT) (GeeksforGeeks, 2024a) genannt.

3 Vorgehen und Methode

Die in diesem und den folgenden Kapiteln erwähnten Implementierungen wurden in der Programmiersprache Python geschrieben und befinden sich alle in den Ordnern `Data_Preparation` (bis Abschnitt 3.2.3) und `LSTM_Modell` (ab Abschnitt 3.3), auf welche im Github Repository: https://github.com/diegomuz/MA_NN_Modell zugegriffen werden kann.

3.1 Datenbeschaffung

Die in dieser Arbeit verwendeten Daten werden von drei Wetterstationen der Stadt Zürich bezogen. Es sind die Wetterstationen Stampfenbachstrasse, Rosengartenstrasse und Schimmelstrasse (siehe Abb. 3.1). Die Daten werden von der Stadt Zürich (Zürich, 2024) öffentlich zur Verfügung gestellt.

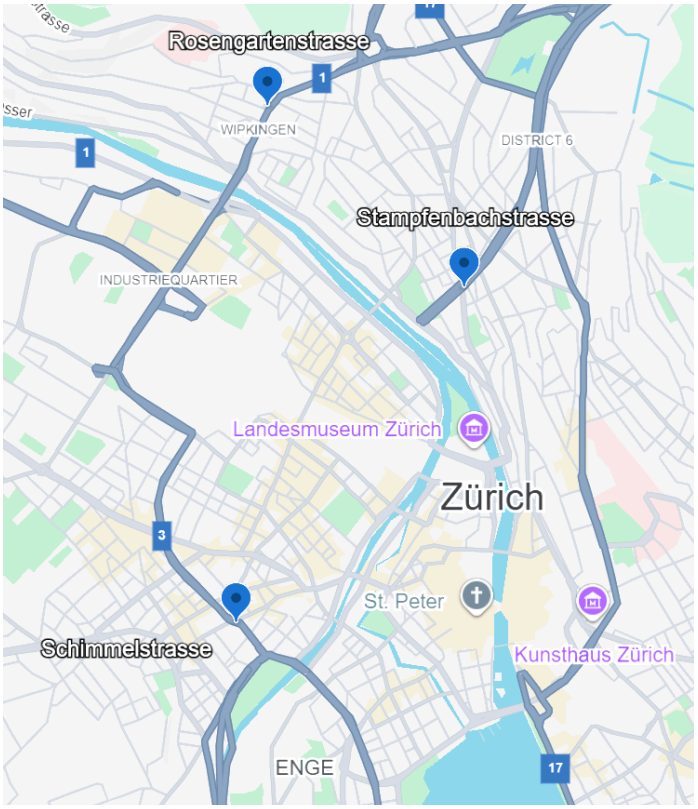


Abbildung 3.1: Positionen der drei Messstationen Stampfenbachstrasse, Rosengartenstrasse und Schimmelstrasse (Quelle: Google Earth)

Die Datensätze, welche zur Verfügung stehen, bestehen aus drei Datengruppen:

Erstens, die Luftqualitätsmessungen (Schadstoffmessungen) verschiedener Gase und Feinpartikel/Feinstäube, insbesondere:

- Ozon (O_3)
- Kohlenmonoxid (CO)
- Schwefeldioxid (SO_2)
- Stickstoffmonoxid (NO)
- Stickstoffdioxid (NO_2)
- Stickstoffoxide (NO_x)
- Feinstaub (PM_{10})
- Feinstaub ($PM_{2.5}$)

Die obigen Werte werden von allen Messstationen gemessen. Eine Ausnahme dazu sind CO und SO_2, welche nur an der Stampfenbachstrasse gemessen werden. Die Messungen erfolgen stündlich.

Zweitens, die Meteodaten, insbesondere:

- Temperatur (T)
- relative Luftfeuchtigkeit (Hr)
- Luftdruck (p)
- Regendauer (RainDur)
- Globalstrahlung (StrGlo)
- Windgeschwindigkeit (WVs) und Windrichtung (WD)

Die obigen Daten werden ebenfalls von allen Messstationen gemessen. Die Messungen werden stündlich erfasst.

Drittens, die Verkehrsdaten, insbesondere:

- Anzahl Zweiräder
- Anzahl Lastwagen

- Anzahl Personenwagen

Die Messungen der obigen Daten erfolgen stündlich an der Stampfenbachstrasse.

Die Daten für alle oben erwähnten Datengruppen stehen für die letzten vier Jahre in einer einheitlichen Struktur zur Verfügung.
Die Datensets wurden in dieser Arbeit in Form von CSV-Dateien mithilfe von `Download_Raw_Data.py` heruntergeladen.

Format der Rohdaten

In der Folge wird zuerst aufgezeigt, wie die Rohdaten formatiert sind. Anschliessend wird erklärt, wie sie zum Zwecke dieser Arbeit umstrukturiert werden.

Die Rohdaten sind wie folgt formatiert:
Jeder einzelne Datensatz umfasst die stündlichen Messungen aller Features (Luftqualität, Meteodaten und Verkehrsdaten). Die Messungen sind nach den Stationen, von denen sie aufgenommen wurden, gruppiert und nach Datum bzw. Uhrzeit geordnet. Abbildung 3.2 zeigt einen Ausschnitt der CSV-Datei für die Luftqualitätsmessungen von 2021, dessen Struktur als *Dataframe* mithilfe der Python Library *Pandas*[1] (Pandas, 2024) eingesehen werden kann.

	Datum	Standort	Parameter	Intervall	Einheit	Wert	Status
0	2021-01-01T00:00+0100	Zch_Stampfenbachstrasse	CO	h1	mg/m3	0.44	bereinigt
1	2021-01-01T00:00+0100	Zch_Stampfenbachstrasse	SO2	h1	µg/m3	4.88	bereinigt
2	2021-01-01T00:00+0100	Zch_Stampfenbachstrasse	NOx	h1	ppb	29.46	bereinigt
3	2021-01-01T00:00+0100	Zch_Stampfenbachstrasse	NO	h1	µg/m3	9.85	bereinigt
4	2021-01-01T00:00+0100	Zch_Stampfenbachstrasse	NO2	h1	µg/m3	41.24	bereinigt
...	...	...	...	...	...	...	...
210235	2021-12-31T23:00+0100	Zch_Rosengartenstrasse	NO	h1	µg/m3	53.78	bereinigt
210236	2021-12-31T23:00+0100	Zch_Rosengartenstrasse	NO2	h1	µg/m3	44.11	bereinigt
210237	2021-12-31T23:00+0100	Zch_Rosengartenstrasse	O3	h1	µg/m3	4.30	bereinigt
210238	2021-12-31T23:00+0100	Zch_Rosengartenstrasse	PM10	h1	µg/m3	61.90	bereinigt
210239	2021-12-31T23:00+0100	Zch_Rosengartenstrasse	PM2.5	h1	µg/m3	62.82	bereinigt

[210240 rows x 7 columns]

Abbildung 3.2: Aufbau des Luftqualitäts-Datensatzes in den CSV-Dateien: Der DataFrame umfasst 210'240 Reihen und 7 Spalten. (Quelle: Zürich, 2024)

In den Implementierungen `Heatmap.py` und `Training_Data_Plot.py` werden die Dataframes so umstrukturiert, dass es pro Spalte ein Feature hat. Dabei werden Messwerte für dasselbe Feature aus unterschiedlichen Stationen in verschiedenen Spalten gespeichert. In den Zeilen sind dann die stündlichen Messungen pro Feature aufgetragen (Abb. 3.3). So ist es leichter, die Daten featureweise zu analysieren.

[1] Die Pandas-library wird in allen nachfolgend erwähnten Programmen benutzt, da Pandas Dataframes ein praktisches Datenformat für den Umgang mit CSV-Dateien und für die Manipulation wie Visualisierung der darin enthaltenen Daten bieten.

	Zch_Rosengartenstrasse Hr	Zch_Rosengartenstrasse NO	Zch_Rosengartenstrasse NO2	...	Zch_Stampfenbachstrasse WVv	Zch_Stampfenbachstrasse Zweirad	Zch_Stampfenbachstrasse p
0	77.98	45.57	49.08	...	1.69	10.0	971.62
1	67.03	37.30	41.03	...	2.77	8.0	971.86
2	69.75	54.98	53.07	...	1.49	4.0	971.76
3	76.00	43.34	47.78	...	1.46	5.0	972.01
4	71.31	24.17	32.70	...	2.74	4.0	972.10
...	...	...	...	...	...	...	...
8755	89.17	11.13	37.89	...	1.34	13.0	960.54
8756	89.06	6.74	28.41	...	1.48	9.0	960.31
8757	85.77	4.65	16.27	...	1.89	13.0	960.82
8758	83.08	4.15	17.83	...	2.04	6.0	961.07
8759	82.33	6.08	19.25	...	2.53	21.0	961.40

[8760 rows x 45 columns]

Abbildung 3.3: Zusammengeführte und nach Features geordnete Luftqualitäts-, Meteo- und Verkehrsdaten (Quelle: eigene Darstellung)

3.2 Datenanalyse und Preprocessing

Im Abschnitt 3.2.1 werden die Rohdaten auf ihre Vollständigkeit und Plausibilität untersucht. Ebenfalls wird die Korrelation der Ozonkonzentration zu den anderen Features ermittelt. Schliesslich wird das Veränderungsverhalten der Ozonkonzentration über eine kürzere Zeitspanne analysiert. Im Abschnitt 3.2.2 wird die Struktur der Trainingsdaten eingeführt.
Schliesslich wird im Abschnitt 3.2.3 aufgezeigt, wie im Abschnitt 3.2.1 festgestellte Datenlücken und Anomalien für die Trainingsdaten bereinigt werden.

3.2.1 Analyse der Rohdaten

Die Vollständigkeitsanalyse der Daten zeigt (siehe Abb. 3.4), dass bei jeder Station Messungen fehlen. Die Datensätze bei der Stampfenbachstrasse sind sowohl für meteorologische Daten als auch für Luftqualitätsdaten vollständiger als bei den anderen Stationen.

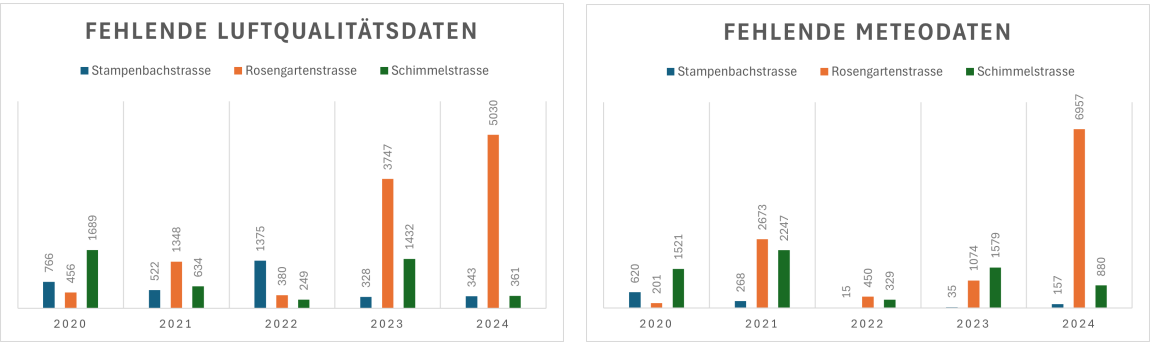


Abbildung 3.4: Anzahl fehlender Werte pro Station in den Jahresdatensätzen von 2020 bis 2024[2]. Die Luftqualitätsdatensätze umfassen 210'240 Messungen, die Meteodatensätze umfassen 192'720 Messungen. (Quelle: eigene Darstellung mit Excel)

Zudem sieht man im Diagramm weiter unten (Abb. 3.5), dass es neben fehlenden Daten (Lücken im Diagramm) auch Ausreisser hat.

[2]Der Datensatz für 2024 war zur Zeit der Erstellung dieser Grafik noch unvollständig und reichte bis September 2024.

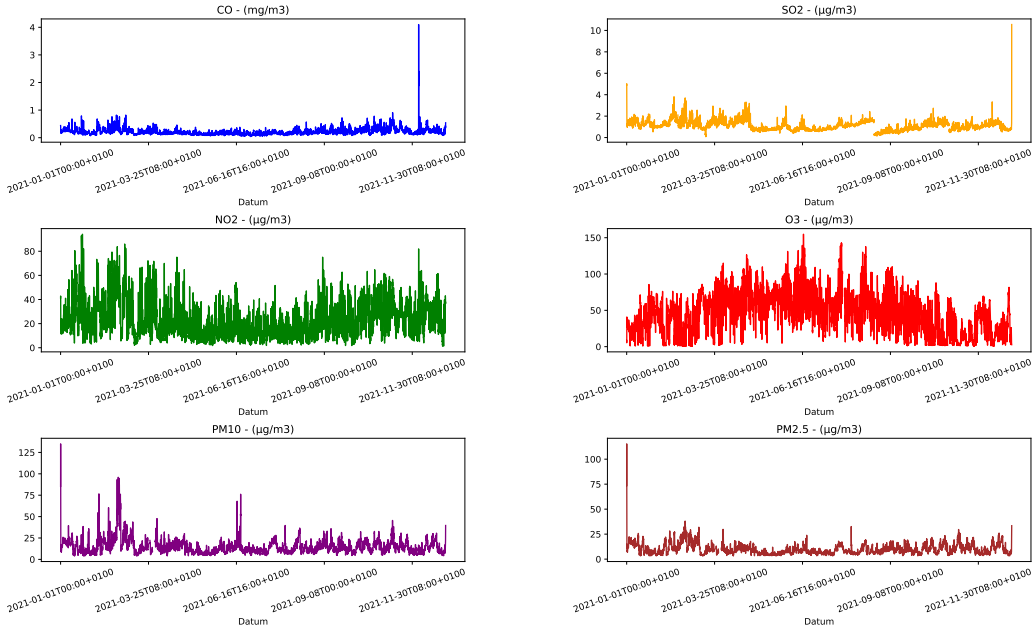


Abbildung 3.5: Diagramm mit den stündlichen Konzentrationsmessungen verschiedener Schadstoffe über das Jahr 2021. (Quelle: eigene Darstellung)

Nach weiterer Analyse sieht man, dass diese Ungenauigkeiten in den Daten bei allen Features anzutreffen sind.

Als Schlussfolgerung der Rohdatenanalyse liegt es nahe, die NaN-Werte (fehlende Messungen) durch näherungsweise korrekte numerische Werte zu ersetzen, da sie sonst beim Trainieren des Modells Fehler produzieren würden. Ebenfalls müssen Ausreisser entfernt bzw. durch richtige Werte ersetzt werden, da es sich bei diesen Werten höchstwahrscheinlich um fehlerhafte Messungen handelt, die das Training verzerren können. Das Modell könnte für solche Werte, die nicht mit dem erlernten Trend oder Muster übereinstimmen, übermässig hohe und fehlerhafte Verlustwerte berechnen, was zu einer falschen Anpassung seiner Gewichtungen führen würde. Ebenfalls wird die Stampfenbachstrasse für die Vorhersage der Ozonkonzentration gewählt, da sie sich aufgrund der höheren Datenvollständigkeit besser eignet, die Genauigkeit der Modellvorhersagen zu validieren. Zudem ist die Stampfenbachstrasse die einzige Station, an der die Konzentrationen von CO und SO_2 gemessen werden.

Als nächstes wird anhand einer Heatmap[3] mit dem *Pearson-Koeffizient* (Wikipedia, 2024l) die Korrelation zwischen der Ozonkonzentration und den anderen Luftqualitätsdaten, den Meteodaten sowie den Verkehrsdaten dargestellt.
Vereinfacht ausgedrückt misst der Pearson-Koeffizient, wie stark zwei Variablen linear zusammenhängen. Positive Werte deuten darauf hin, dass zwei Variablen a

[3]Für diese Visualisierung diente dieser Artikel als Orientierung (Brownlee, 2019)

und b einen positiven linearen Zusammenhang ($a = k \cdot b$, $k \geq 0$) haben. Negative Werte weisen auf einen negativen linearen Zusammenhang hin. Korrelationen von Betrag zwischen 0.5 und 1 werden als hohe Korrelation bezeichnet, wobei Korrelationen über einen Betrag von 0.7 sehr hoch sind. Werte, die betragsmässig zwischen 0.3 bis 0.5 liegen, werden als mittlere Korrelation bezeichnet und Werte unter 0.3 weisen auf eine sehr geringe bis fehlende Korrelation hin (DATAtab Team, 2024; Statistics Solutions, 2024).
Die Heatmap für die Datensätze des Jahres 2023 zeigt, dass die Ozonkonzentration an der Stampfenbachstrasse eine relativ hohe Korrelation mit CO, NO_x, NO_2 und NO hat (siehe Abb. 3.6). Mit den Feinpartikeln PM_{10} und $PM_{2.5}$ weist O_3 dagegen eine eher geringe Korrelation auf.

In der Heatmap werden auch die Korrelationen zu den Gasen und Feinstäuben an der Schimmel- und Rosengartenstrasse aufgezeigt. Obwohl sie etwas geringer ausfallen, bestätigen sie, mit Ausnahme des Falls von PM_{10} an der Rosengartenstrasse, die an der Stampfenbachstrasse festgestellten Zusammenhänge.

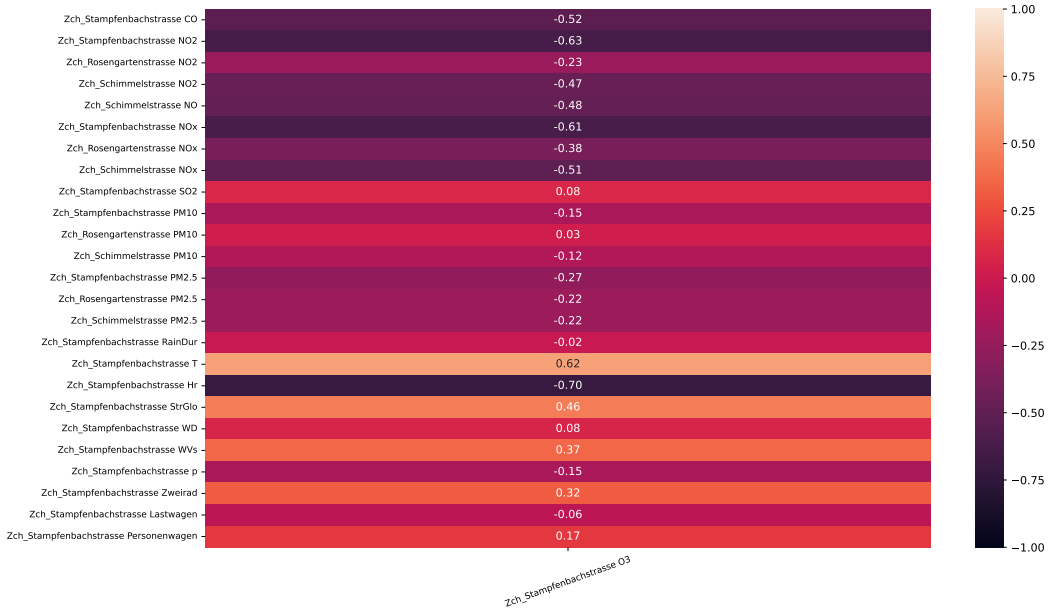


Abbildung 3.6: Heatmap der Pearson-Korrelation zwischen O_3 und den anderen Features des Datensatzes auf einer Skala von -1 bis 1. (Quelle: eigene Darstellung)

Die Ozonkonzentration hat ebenfalls eine hohe Korrelation mit der relativen Luftfeuchtigkeit (Hr), der Lufttemperatur (T) und eine mittlere Korrelation mit der Windgeschwindigkeit (WVs) und der Globalstrahlung (StrGlo).
Es wird ebenfalls beobachtet, dass gewisse Faktoren nur einen geringen Einfluss auf die O_3-Konzentration haben. Dazu gehören die Regendauer (RainDur), der Luftdruck (p), die Windrichtung (WD) die SO_2-Konzentration und die Verkehrsdaten an der

Stampfenbachstrasse (untere drei Zeilen).

Die Korrelationsanalyse führt zum Schluss, dass alle Verkehrsdaten aufgrund niedriger Korrelation und geringer Vollständigkeit nicht in das Modelltraining einbezogen werden. Die anderen Features, welche ebenfalls eine geringe Korrelation aufzeigen, jedoch höhere Datenvollständigkeit aufweisen, werden hingegen beibehalten.

Als nächstes wird der Konzentrationsverlauf von Ozon über eine Zeitspanne von 14 Tagen genauer betrachtet (siehe Abb. 3.7). Man stellt fest, dass die Fluktuationen eine Periodizität von ungefähr 24 Stunden haben. Die O_3-Konzentration ist am Tag hoch und sinkt dann in der Nacht.

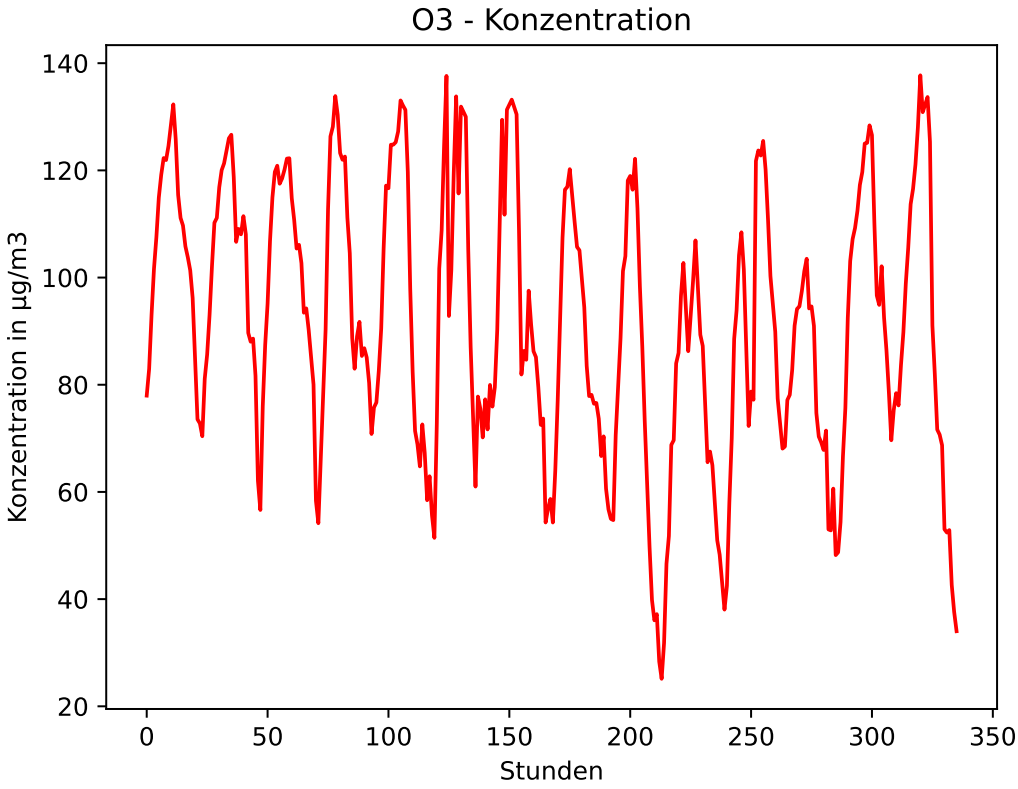


Abbildung 3.7: Verlauf der Ozonkonzentration über 14 Tage (336 Stunden). (Quelle: eigene Darstellung)

Um dem Modell ein Verständnis für die Tageszyklen zu vermitteln und ihm zeitliche Informationen bereitzustellen, werden die Zeitangaben mithilfe von Sinus- und Cosinusfunktionen in numerische Werte codiert und als Trainingsfeatures eingebunden:

Stunde im Tag:

$\sin_h = \sin(\frac{2\pi \cdot h}{24}), \quad \cos_h = \cos(\frac{2\pi \cdot h}{24})$

Tag im Jahr:
$\sin_{\mathrm{d}} = \sin(\frac{2\pi \cdot d}{365}), \quad \cos_{\mathrm{d}} = \cos(\frac{2\pi \cdot d}{365})$

Monat im Jahr:
$\sin_{\mathrm{m}} = \sin(\frac{2\pi \cdot m}{12}), \quad \cos_{\mathrm{m}} = \cos(\frac{2\pi \cdot m}{12})$

Diese Sinus- und Cosinusfunktionen haben einen Wertebereich zwischen -1 und 1 und eine Periode von 24 für die Stunden, von 365 für die Tage und von 12 für die Monate.

3.2.2 Struktur der Trainingsdaten

Im Folgenden wird beschrieben, wie die ursprünglichen Datensätze umstrukturiert werden und wie die Trainingsdatensätze gestaltet sind, die das Modell für den Lernprozess verwendet.
Ähnlich wie schon im Abschnitt 3.1 erklärt, werden die Dataframes, die man von den rohen CSV-Dateien erhält, so umgestaltet, dass die Messdaten nach den Features geordnet sind. Dieser Prozess wird für die Luftqualitätsdaten in `Air_Data_gather.py` und für die Meteodaten in `Meteo_Data_gather.py` durchgeführt. Anschliessend werden die Dataframes für die Meteo- und Luftqualitätsdaten in `Create_Training_Dataset.py` vereint.

Von jedem Luftqualitätswert und jedem meteorologischen Messwert werden zwei unterschiedliche Features kreiert. Zum ersten Feature gehören alle an der Stampfenbachstrasse erfassten Messungen für diesen Schadstoff oder Meteowert. Dieses Feature wird als *Hauptfeature* bezeichnet. Das zweite Feature, zu welchem im Trainingsdatensatz das Präfix *Cont-* für Kontext (engl. *Context*) hinzugefügt wird, ist der Durchschnitt der Messwerte von der Rosengartenstrasse und der Schimmelstrasse für diesen Schadstoff oder Meteowert. Falls ein Messwert bei einer der letzteren beiden Stationen fehlt, wird der Messwert der anderen Station statt dem Durchschnitt benutzt. So kann die Anzahl der fehlenden Werte für dieses Feature erheblich reduziert werden. Falls beide Werte fehlen, so muss dieser Datenpunkt ersetzt werden (siehe Abschnitt 3.2.3). Dieses Kontext-Feature wird in den Trainingsdatensatz (Abb. 3.8) integriert, mit dem Ziel, dem Modell ein besseres und erweitertes räumliches Verständnis der Schadstoffkonzentrationen und Wetterbedingungen zu vermitteln.

3.2.3 Behebung der Anomalien in den Trainigsdatensätzen

Nun wird präsentiert, wie die in Abschnitt 3.2.1 nachgewiesenen falschen bzw. fehlenden Datenpunkte identifiziert und ersetzt werden. Die Methode zum Entfernen von

	Datum	CO	SO2	NOx	NO	NO2	O3	PM10	PM2.5	Cont_NOx	...	Cont_RainDur	Cont_WD	Cont_WVv	Cont_WVs	sin_h	cos_h	sin_d	cos_d	sin_m	cos_m
0	2020-01-01T00:00+0100	0.42	2.09	33.91	20.41	33.55	1.75	35.2205	23.902	40.730	...	0.00	222.775	0.275	0.700	0.000000	1.000000	1.721336e-02	0.999852	5.000000e-01	0.866025
1	2020-01-01T01:00+0100	0.44	2.16	34.71	21.22	33.84	1.90	35.2205	23.902	50.275	...	0.00	57.780	0.365	0.625	0.258819	0.965926	1.721336e-02	0.999852	5.000000e-01	0.866025
2	2020-01-01T02:00+0100	0.50	1.51	33.46	20.43	32.67	1.86	35.2205	23.902	44.555	...	0.00	228.275	0.495	0.830	0.500000	0.866025	1.721336e-02	0.999852	5.000000e-01	0.866025
3	2020-01-01T03:00+0100	0.46	2.38	28.60	15.87	30.36	1.86	35.2205	23.902	40.015	...	0.00	220.715	0.750	1.035	0.707107	0.707107	1.721336e-02	0.999852	5.000000e-01	0.866025
4	2020-01-01T04:00+0100	0.42	1.66	28.19	13.54	33.14	1.81	29.4100	23.902	38.950	...	0.00	56.820	0.270	0.680	0.866025	0.500000	1.721336e-02	0.999852	5.000000e-01	0.866025
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
35059	2023-12-31T19:00+0100	0.20	0.78	10.20	2.13	16.23	60.75	4.4000	3.600	22.445	...	5.24	168.380	0.265	0.530	-0.965926	0.258819	6.432491e-16	1.000000	-2.449294e-16	1.000000
35060	2023-12-31T20:00+0100	0.21	1.15	8.02	1.40	13.18	62.74	9.0600	8.150	15.820	...	0.00	151.030	0.110	0.560	-0.866025	0.500000	6.432491e-16	1.000000	-2.449294e-16	1.000000
35061	2023-12-31T21:00+0100	0.19	1.27	5.32	0.89	8.80	65.21	11.8900	10.810	10.315	...	0.00	132.540	0.210	0.780	-0.707107	0.707107	6.432491e-16	1.000000	-2.449294e-16	1.000000
35062	2023-12-31T22:00+0100	0.17	1.13	3.83	0.88	5.96	71.38	5.6200	4.730	12.200	...	0.10	125.985	0.285	0.735	-0.500000	0.866025	6.432491e-16	1.000000	-2.449294e-16	1.000000
35063	2023-12-31T23:00+0100	0.19	1.57	8.50	2.01	13.18	61.50	12.3600	10.210	12.710	...	0.00	182.835	0.500	0.840	-0.258819	0.965926	6.432491e-16	1.000000	-2.449294e-16	1.000000

[35064 rows x 36 columns]

Abbildung 3.8: Dataframe mit der Struktur der Trainingsdaten. (Quelle: eigene Darstellung)

NaN-Werten hängt von der Korrektheit der vorhandenen Daten ab, weshalb zunächst Ausreisser behandelt und anschliessend die fehlenden Werte ersetzt werden.

Methode um Ausreisser zu entfernen

Bei der Darstellung der Daten in einem Histogramm (Wikipedia, 2024f; Yi, 2024) und einem Boxplot (Wikipedia, 2024d) bekommt man ein besseres Bild der Verteilung der Daten und kann die Lage von Ausreissern im Vergleich zu regulären Datenpunkten visualisieren. Das Histogramm zeigt auf, wie oft ein Messwert im Datensatz vorkommt. Der Boxplot zeigt das erste, das zweite und das dritte Quartil, wobei das zweite Quartil (Q_2) dem Median entspricht. Die beiden äusseren Geraden sind die untere und die obere "Antenne" engl. auch *Whisker* genannt. Nach der *Turkey's Fence* Definition von Ausreissern (Wikipedia, 2024d, 2024j) signalisieren die Bereiche ausserhalb der beiden Antennen potenzielle Ausreisserwerte. Die Antennen befinden sich hierbei 1.5-mal den Interquartilabstand (IQR) vom nächsten nebenstehenden ersten (Q_1) oder dritten Quartil (Q_3) entfernt. Die obere Antenne wäre also bei $Q_3 + 1.5 \cdot IQR$. Im folgenden Diagramm (Abb. 3.9) wird die Verteilung der Messwerte für das Gas CO während des Jahres 2021 aufgezeigt.

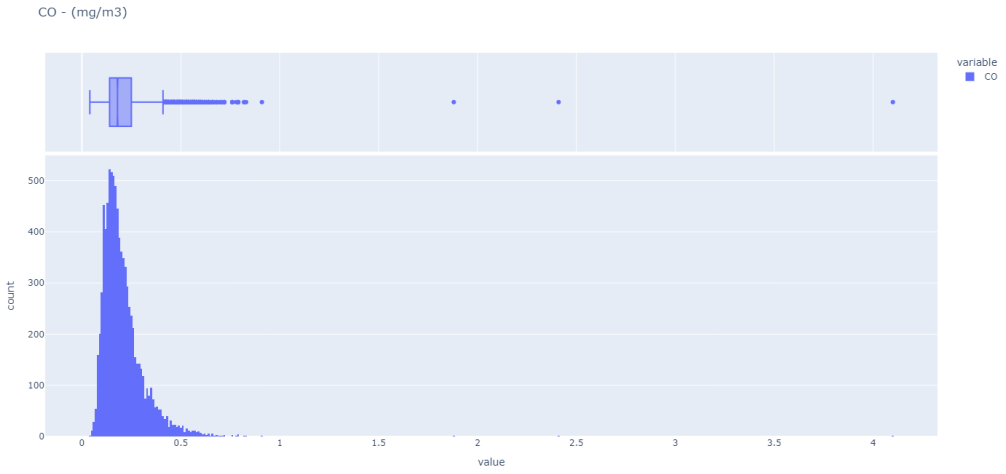


Abbildung 3.9: Histogramm (unten) und Boxplot (oben) der CO-Messwerte für die Stampfenbachstrasse im Jahr 2021. (Quelle: eigene Darstellung)

Wie man dem Bild entnimmt, liegen die meisten Datenpunkte relativ eng beisammen

in einem Bereich von 0,1 mg/m^3 bis 0,43 mg/m^3. Es gibt jedoch einige Ausreisser (gemäss der Turkey's Fence Definition), die sich nicht allzu weit von der oberen Grenze befinden (Konzentrationen bis 0,7 mg/m^3). Diese Werte sollten nicht als Ausreisser klassifiziert und ersetzt werden. Es gibt aber vereinzelt Messwerte, die deutlich höher sind als der Wert der oberen Antenne und sich weit entfernt von allen anderen Messwerten befinden. Bei diesen Werten ist davon auszugehen, dass es sich um Messfehler handelt. Zudem stellt man fest, dass die Häufigkeit solcher Ausreisser im Datensatz abnimmt, je weiter der Abstand zur oberen Antenne ist.

Nach genauerer Untersuchung solcher Diagramme für alle Features wurde beschlossen, Ausreisser wie folgt zu klassifizieren:

Ausreisser sind über 1.7-mal den Interquartilabstand vom nächsten Quartil entfernt und haben eine Auftrittshäufigkeit, die geringer als 5 Mal pro Jahr ist.

Die Wahl des Faktors 1.7 statt 1.5 für die Distanz zum nächsten Quartil wurde getroffen, weil bei Faktor 1.5 zu viele Werte als Ausreisser identifiziert wurden, obwohl sie in einer Histogramm Abbildung noch sehr nahe an den Nicht-Ausreisser-Werten lagen. Der Entscheid richtete sich am Ziel einer minimalen Datenausscheidung. Dabei soll wertvolle Information für das Modelltraining beibehalten werden und nur extreme Ausreisser ersetzt werden.

Aus diesem Grund wurde auch das zweite Kriterium einer Auftrittshäufigkeit unter 5 eingeführt. Beim Veranschaulichen der Daten fiel nämlich auf, dass bei einigen Features gewisse Datenpunkte zwar ausserhalb der festgelegten Quartilgrenzen liegen, aber dennoch eine relativ zu den Datenpunkten innerhalb der Grenzen vergleichbare Häufigkeit aufweisen.

Im Programm `Meteo_Data_gather.py` und `Air_Data_gather.py` werden Ausreisser nach den oben erklärten Kriterien featureweise identifiziert und dann ersetzt durch den Durchschnitt von den Messwerten der vorhergehenden und der nachfolgenden Stunde. Falls die Datenpunkte unmittelbar neben dem Ausreisser ebenfalls Ausreisser oder NaN-Werte sind, so wird das nächste, zwei Stunden vom detektierten Ausreisser entfernte Tupel untersucht. Dieses Vorgehen wird wiederholt, bis das Programm ein Messwertepaar findet, mit dessen Durchschnitt es den Ausreisser ersetzen kann. Dieser iterative Prozess hat ein Limit von maximal 20 Wiederholungen.
Für die Luftqualitätsdaten wurden in den Trainingsdatensätzen von 2020 bis 2023 im Durchschnitt 3.1% der Daten als Ausreisser identifiziert und ersetzt. Bei den Meteodaten lag dieser Wert im Durchschnitt bei 3,9% .

Methode um fehlende Werte zu ersetzen

Das Ersetzen von fehlenden Daten folgt nach der Behandlung der Ausreisser und wird im Programm `Create_Training_Dataset.py` ausgeführt. Dieser Prozess wird im Bereich des maschinellen Lernens auch *Imputation* genannt. Hier wurde als Imputationsmethode der *KNN-Imputer* von der Python Library *scikit-learn* (scikit-learn, 2024a) eingesetzt. Die Informationen aus diesen drei Artikeln zu KNN-Imputation (Browlee, 2020; Geeksforgeeks, 2024; Kyaw Saw Htoon, 2020) dienten zur Implementierung dieses Algorithmus. KNN steht für k-Nearest-Neighbors auf Englisch und ist ein Machine Learning Algorithmus, der für Regression verwendet wird. Seine Funktionsweise wird nun grob erklärt:
Der Algorithmus behandelt jeden Zeitschritt, also jede Stunde im Trainingsdatensatz als einen n-dimensionalen Ortsvektor, wobei n der Anzahl Features im Datensatz entspricht. Bei jedem dieser Vektoren, die mindestens eine Komponente haben, deren Wert NaN ist, wird der Euklidische Abstand (Wikipedia, 2024e) berechnet zu allen anderen Ortsvektoren im Datensatz, die keine fehlende Komponente aufweisen, wobei der NaN-Wert für die Berechnung der Euklidischen Abstands ausgelassen wird. Anschliessend wird die NaN-Komponente des Vektors ersetzt durch den Durchschnitt dieser Komponenten in den k nächsten (kleinster Euklidischer Abstand) Ortsvekoren, wobei k ein einstellbarer Parameter ist, der hier auf 5 gesetzt wurde.
Da die Features im Datensatz alle verschiedene Einheiten und Skalen haben, werden bei der Berechnung des Euklidischen Abstands Features mit grösseren numerischen Werten höher gewichtet als jene mit kleineren numerischen Werten. Um diese mögliche Verzerrung beim Imputationsalgorithmus zu vermeiden, werden die Daten zuerst mit dem *MinMaxScaler* von *scikit-learn* (scikit-learn, 2024b) auf einen Bereich zwischen 0 und 1 skaliert und nach der Imputation wieder zurückskaliert.

Bereinigte Datensätze

Nach dem Ersetzen von Ausreissern und NaN-Werten sehen die in Abbildung 3.5 dargestellten Messwerte fürs Jahr 2021 wie folgt aus (Abb. 3.10):

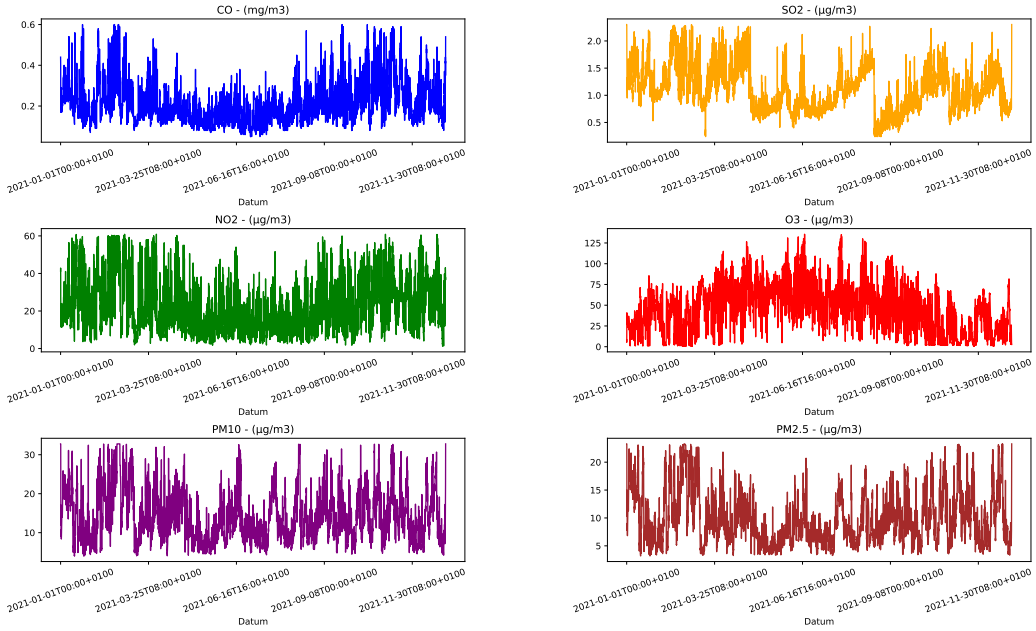


Abbildung 3.10: Stündliche Konzentrationswerte der Schadstoffe für das Jahr 2021 nach Ersetzen von NaN-Werten und Ausreissern im Datensatz. (Quelle: eigene Darstellung)

3.3 Modelltraining und Experimente

Das LSTM soll mit den bereinigten Trainingsdatensätzen von den Jahren 2020 bis 2023 trainiert werden. Dabei werden mehrere LSTM-Modelle trainiert, welche die erwartete künftige Ozonkonzentration vorhersagen. Es sollen mehrere Konfigurationen der veränderbaren Parameter in der Modellarchitektur und im Trainingsdurchlauf getestet und verglichen werden. Diese veränderbaren Parameter werden in der Folge noch genauer erklärt.

Die verschiedenen LSTM-Modelle werden mithilfe der Python Libraries *Keras* (Chollet, 2024c), *Tensorflow* (Brain, 2024), *NumPy* (Oliphant, 2024) und der vorhin bereits erwähnten *Pandas* im Programm `Model_Training.py` kreiert und trainiert. Folgende Artikel dienten als Orientierung und als Anleitung für den Umgang mit Keras: (Brownlee, 2022b) und (Brownlee, 2020) von der Platform *Machine Learning Mastery* sowie (Ryan T.J.J, 2020) und (Glancszpigel, 2020) von der Platform *Medium*. Nun wird erläutert, wie die LSTM-Modelle gebaut und trainiert werden:

Als Verlustfunktion werden die Modelle *MSE* (Wikipedia, 2024i), was für engl. *mean squared error* steht, verwenden. Diese ist eine der typischsten Verlustfunktionen für Regressionsprobleme. Die Abweichung zwischen der Modellvorhersagen $\hat{Y}$ und dem echten Wert Y, wobei $\hat{Y}$ und Y Vektoren mit n Komponenten sind, wird wie folgt

berechnet:

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

Um die Lernrate der Modelle während des Trainings dynamisch anzupassen, wird der *Optimizer Adam* (Chollet, 2024a) verwendet.
Der Gesamttrainingsdatensatz wird aufgeteilt in Trainingsdaten und Testdaten mit einem Verhältnis von 80:20. Somit reichen die Trainingsdaten vom 1. Januar 2020 bis zum 14. März 2023. Sie werden verwendet, um die Gewichtungen und Biases des Modells anzupassen bzw. für den Lernvorgang des Modells. Anhand der Testdaten wird die Genauigkeit der Modellvorhersagen für Daten, welche das Modell nicht aus dem Training kennt, evaluiert.

Für den Modellbau wird die Klasse *Sequential* (Chollet, 2024d) von *Keras* verwendet. Sie erlaubt es, Modelle durch Schichtung von Layers zu erstellen.
Die Modelle sollen aus zwei Layers bestehen. Das erste Layer soll ein LSTM-Layer sein, welcher die zeitlichen Trends der Konzentrationsentwicklung von Ozon erlernen soll. Das zweite Layer wird dann ein Dense-Layer sein, welches aus dem mehrdimensionalen Output des LSTMs einen eindimensionalen Output, nämlich die Konzentration von Ozon, extrahiert.
Der Modellaufbau wird auf Abbildung 3.11 schematisch dargestellt.

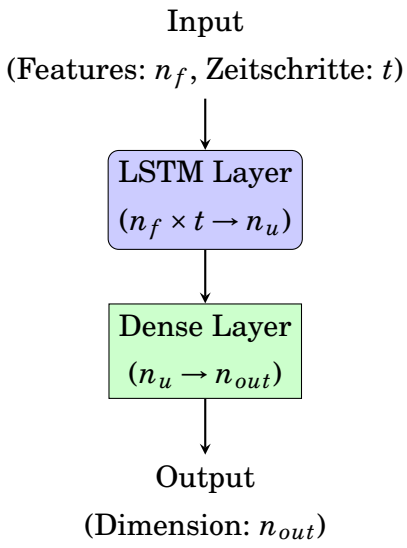


Abbildung 3.11: Diagramm der geschichteten Layer im LSTM-Modell mit den jeweiligen Input- und Outputdimensionen. Vertikale Pfeile zeigen die Verbindung zwischen den Layers, horizontale Pfeile die Konversion der Inputdimension in die Outputdimension. (Quelle: eigene Darstellung mit TikZ)

Es werden nun die veränderbaren Parameter der Modellarchitektur und des Modelltrainings vorgestellt, deren Einfluss auf die Vorhersagequalität später analysiert wird. Abbildung 3.12 gibt eine Übersicht über diese Parameter und ihre Funktion in der Modellarchitektur sowie im Training.

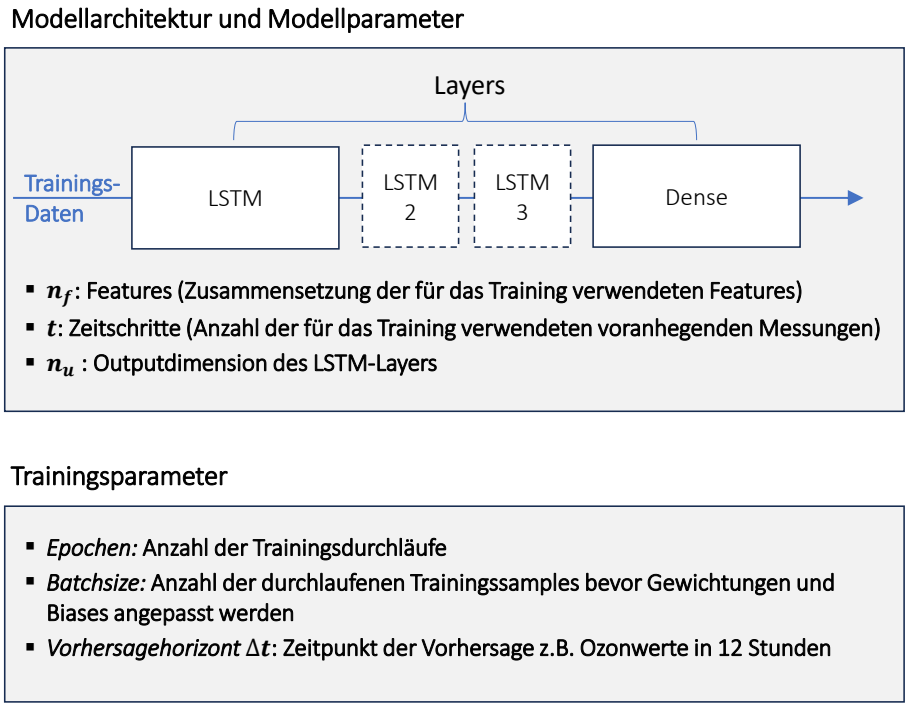


Abbildung 3.12: Zusammenfassung der Parameter in der Modellarchitektur und im Modelltraining (Quelle: eigene Darstellung mit Powerpoint)

Die veränderbaren Parameter der Modellarchitektur sind:

Zum einen die Anzahl der Inputfeatures n_f, d.h. welche der vorhandenen Luftqualitätsfeatures, Meteofeatures und zeitliche Features dem Modell zur Verfügung gestellt werden. n_f kann variiert werden, da wie in Abbildung 3.6 ersichtlich gewisse Meteo- oder Luftqualitätsfeatures redundant sein könnten.

Zum anderen sind es die Anzahl der vorangehenden Stundenmessungen t, anhand derer das Modell die Vorhersagen macht. $n_f \times t$ ist die Inputdimension des Modells.

Hinzu kommt die Outputdimension des LSTM-Layers n_u, die gleichzeitig die Inputdimension des nächsten Layers definiert. Darüber hinaus entspricht n_u der Dimension der Lang- und Kurzzeitgedächtnisse C_t und h_t des LSTM-Layers, wodurch sie die Merkfähigkeit und Komplexität des Modells bestimmt. Die Outputdimension des

Dense-Layers n_{out} wird auf 1 gesetzt.

Es wird ebenfalls experimentiert, welche Auswirkung das Hinzufügen eines oder zwei weiteren LSTM-Layer in das Modell, also die Erhöhung der Modellkomplexität, auf die Vorhersagegenauigkeit hat. Die weiteren LSTM-Layer würden direkt hinter das erste LSTM-Layer eingefügt werden. Ihre Outputdimension würde ebenfalls auf n_u gesetzt. In anderen Worten wären $n_u = n_{u2} = n_{u3}$.
Der Aufbau aus drei LSTM-Layer und einem Dense-Layer wies eine starke Leistung auf bei der Vorhersage von Schadstoffkonzentrationen in einer Studie, welche mit Messdaten aus Indien durchgeführt wurde (Ghufran Isam Drewil, 2022).

Weiter gibt es noch Parameter, welche nicht die Architektur des Modells verändern, sondern das Trainingsverhalten beeinflussen. Diese sind die *Batchsize*, die Anzahl *Epochen* (Brownlee, 2022a) und der Vorhersagehorizont Δt:

Die Batchsize definiert nach wie vielen Trainingssamples, also Input und Output Tupel, das Modell beim Durchgehen durch den Trainingsdatensatz die seine Gewichtungen und Biases anpasst. Ist die Batchsize beispielsweise 32, so macht das Modell Vorhersagen mit 32 Input-Output-Paaren und berechnet die Abweichung (MSE) seiner Vorhersagen für jedes dieser Paare. Dann passt er seine Gewichtungen an, wobei er hierfür die mittlere Abweichung für alle 32 Samples verwendet.
Für die Modelle, welche hier getestet werden, wird die Batchsize auf 32 gesetzt. Dieser Wert erwies sich als gut bei der vorhin erwähnten Studie (Ghufran Isam Drewil, 2022).

Die Epochen stehen für die Anzahl Iterationen des Lernprozesses. Dies heisst, wie oft das Modell mit gegebener Batchsize den gesamten Trainingsdatensatz durchläuft und seine Gewichtungen bzw. Biases anpasst. Mit zunehmender Anzahl an Epochen kann das Modell seine Gewichtungen theoretisch weiter optimieren. Ab einem bestimmten Punkt jedoch besteht die Gefahr, dass das Modell nicht mehr die zugrunde liegenden Trends, welche die zeitliche Entwicklung der Konzentrationen von Ozon beschreiben, erlernt, sondern stattdessen beginnt, die Trainingsdaten nahezu auswendig zu lernen. In solchen Fällen kann das Modell zwar sehr präzise Vorhersagen für die Trainingsdaten machen, hat jedoch Schwierigkeiten, wenn es auf neue, unbekannte Daten angewendet wird. Dieses Problem ist auch bekannt als *Overfitting* (Wikipedia, 2024k). Um Overfitting zu vermeiden, werden in dieser Arbeit sogenannte *Model Checkpoints* und *Early Stopping* Mechanismen von Keras (Chollet, 2024b) ins Training eingebaut, welche das Training automatisch stoppen, falls nach einer gewissen Anzahl Epochen, die hier auf vier gesetzt wurde, keine Minimierung der Verlustfunktion für die Testdaten zu beobachten ist. Ausserdem wird auch die letzte Version des Modells (bzw.

seiner Gewichtungen und Biases), bei welcher eine Minimierung der Verlustfunktion vorhanden war, abgespeichert.

Der Vorhersagehorizont Δt bestimmt, wie viele Stunden in der Zukunft die Stunde liegt, für die eine Vorhersage gemacht wird. Ist Δt beispielsweise 12, so wird die O_3-Konzentration in 12 Stunden vorhergesagt. Δt dient dazu, das Verhalten des Modells bei Vorhersagen für Zeitpunkte weiter in der Zukunft zu untersuchen.

Um dem Modell das Lernen zu erleichtern, werden alle Inputdaten mit dem bereits in Abschnitt 3.2.3 erwähnten MinMaxScaler auf einen Bereich zwischen 0 und 1 skaliert und dann bei der Vorhersage der Ozonkonzentration wieder zurück skaliert.

Auf den Tabellen 3.1 und 3.2 wird präsentiert, welche Konfigurationen der genannten veränderbaren Parameter für die Modellstruktur und für das Modelltraining getestet werden.

Tabelle 3.1: Tabelle mit den Werten der untersuchten Parameter in der Modellarchitektur

Anzahl LSTM Layer	n_u	t	Δt
1	32	6	1
2	64	12	12
3	128	24	24
		36	

Tabelle 3.2: Tabelle mit den untersuchten Inputfeature-Konfigurationen für das Modelltraining

n_f
nur O_3-Konzentration (1 Feature)
nur O_3-Konzentration + zeitliche Features (7 Features)
nur Hauptfeatures mit hoher Korrelation[4]+ zeitliche Features (16 Features)
Haupt- und Kontextfeatures mit hoher Korrelation + zeitliche Features (24 Features)
alle Features (35 Features)

[4]Zu den Features mit hoher Korrelation gehören: O_3, CO, NO_x, NO, NO_2, Temperatur, Luftfeuchtigkeit, Globalstrahlung und Windgeschwindigkeit.

4 Ergebnisse

Die genannten Konfigurationen der Modell- und Trainingsparameter (siehe Tab. 3.1 und Tab. 3.2) werden quantitativ, aber auch qualitativ analysiert.[1] Dazu werden Diagramme verwendet, welche die Modellvorhersagen für die Testdaten den tatsächlichen Werten gegenüberstellen, sowie Tabellen, in denen die Fehlermetriken der Modellvorhersagen aufgeführt sind.

Um die Genauigkeit der Vorhersagen zu bestimmen, werden die Fehlermetriken RMSE (Wikipedia, 2024m) (engl. Root Mean Square Error), MAE (Wikipedia, 2024h) (engl. Mean Absolute Error) und die Pearson-Korrelation berechnet.
Anbei sind die Formeln für die Berechnung der beiden Fehlermetriken RMSE und MAE:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Hierbei ist:

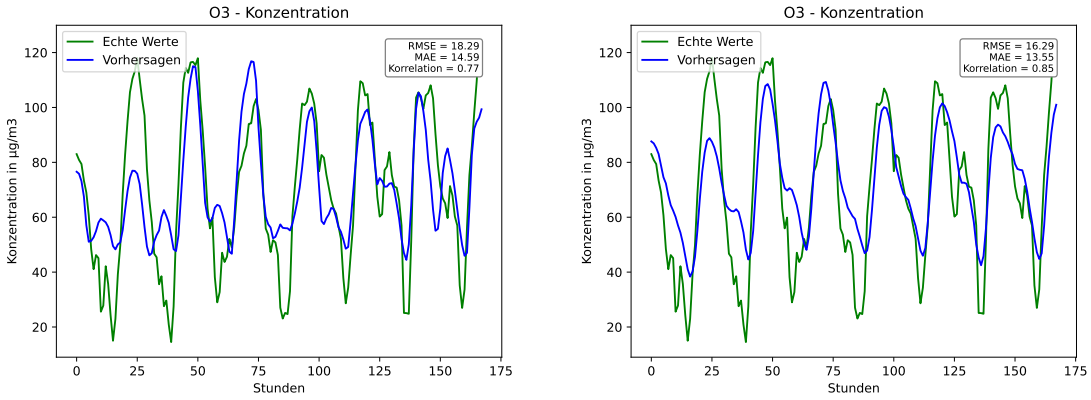
- n die Anzahl Datenpunkten
- y_i der echte Wert der i-ten Datenpunkts
- $\hat{y}_i$ der vorhergesagte Wert des i-ten Datenpunkts

Der erste Parameter, der untersucht wurde, ist die Anzahl Inputfeatures n_f.
Es werden fünf Modelle verglichen (siehe Tab. 3.2), wobei bei allen Modellen n_u auf 32 und t auf 12 gesetzt wurde. Die Resultate sind auf den Abbildungen 4.1, 4.2 und 4.3 ersichtlich.

[1] Die Modellevaluation erfolgt in den Programmen `Evaluate_Model.py`, `Compare_Models` und `Plot_Training_Progress.py`.

Die ersten beiden wurden nur mit der Ozonkonzentration trainiert, einmal mit den zeitlichen Features und einmal ohne. Die drei weiteren Modelle auf Abbildung 4.2 wurden mit unterschiedlichen Kombinationen der Haupt- und Kontextfeatures trainiert. Ebenfalls wurde der Effekt des Hinzufügen der Features mit geringer Korrelation[2] evaluiert.
Abbildung 4.3 zeigt erneut den Vergleich des MSE für Trainings- und Testdaten der fünf Modelle.
Zum einen sieht man, dass die Vorhersagen für die Modelle, welche mit den Zeitfeatures trainiert wurden, bessere Fehlermetriken erreichten. Zum anderen ist ersichtlich, dass die Modelle von den Kontextfeatures profitieren. Interessant ist auch, dass die Qualität der Modellvorhersagen durch das Einbinden der Features mit geringer Korrelation weiter zunimmt.

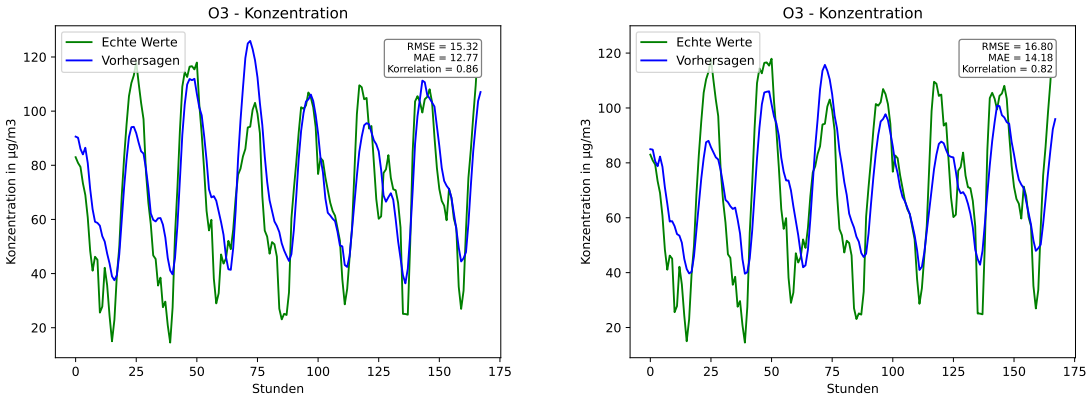
Auf Grundlage dieser Erkenntnisse wird beschlossen, die Modelle in den folgenden Experimenten mit allen Features im Datensatz zu trainieren.



(a) **(b)**

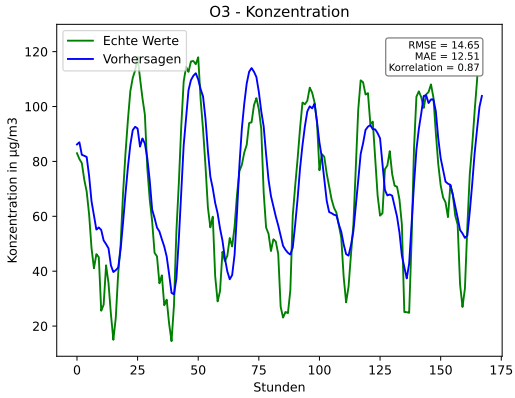
Abbildung 4.1: Graphen und Fehlermetriken der stündlichen Vorhersagen des Modells mit $t = 12$ und $n_u = 32$ bei $\Delta t = 12$: Training nur mit der Ozonkonzentration (a), zeitliche Features zusätzlich ins Training eingebunden (b). (Quelle: eigene Darstellung)

[2]Zu den Features mit geringer Korrelation gehören PM_{10}, $PM_{2.5}$, SO_2, der Luftdruck, die Windrichtung und die Regendauer.



(a)

(b)



(c)

Abbildung 4.2: Graphen und Fehlermetriken der stündlichen Vorhersagen des Modells mit $t = 12$ und $n_u = 32$ bei $\Delta t = 12$ für verschiedene Inputfeature-Konfigurationen: Nur Hauptfeatures mit hoher Korrelation + zeitliche Features (a), Haupt- und Kontextfeatures mit hoher Korrelation + zeitliche Features (b) und alle Features (c). (Quelle: eigene Darstellung)

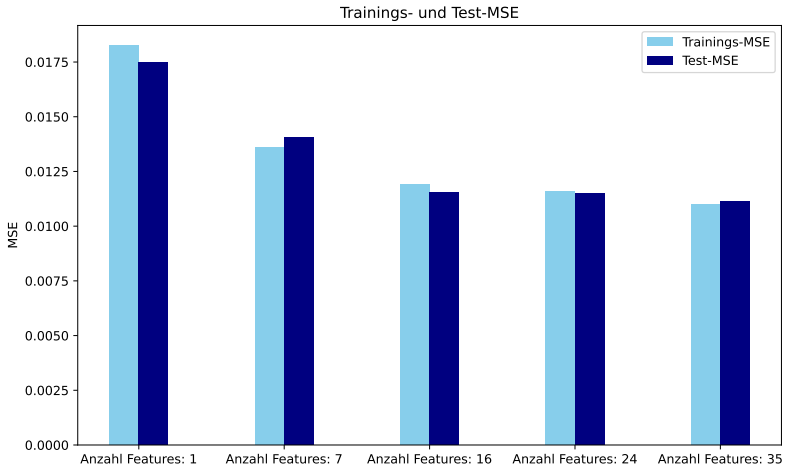
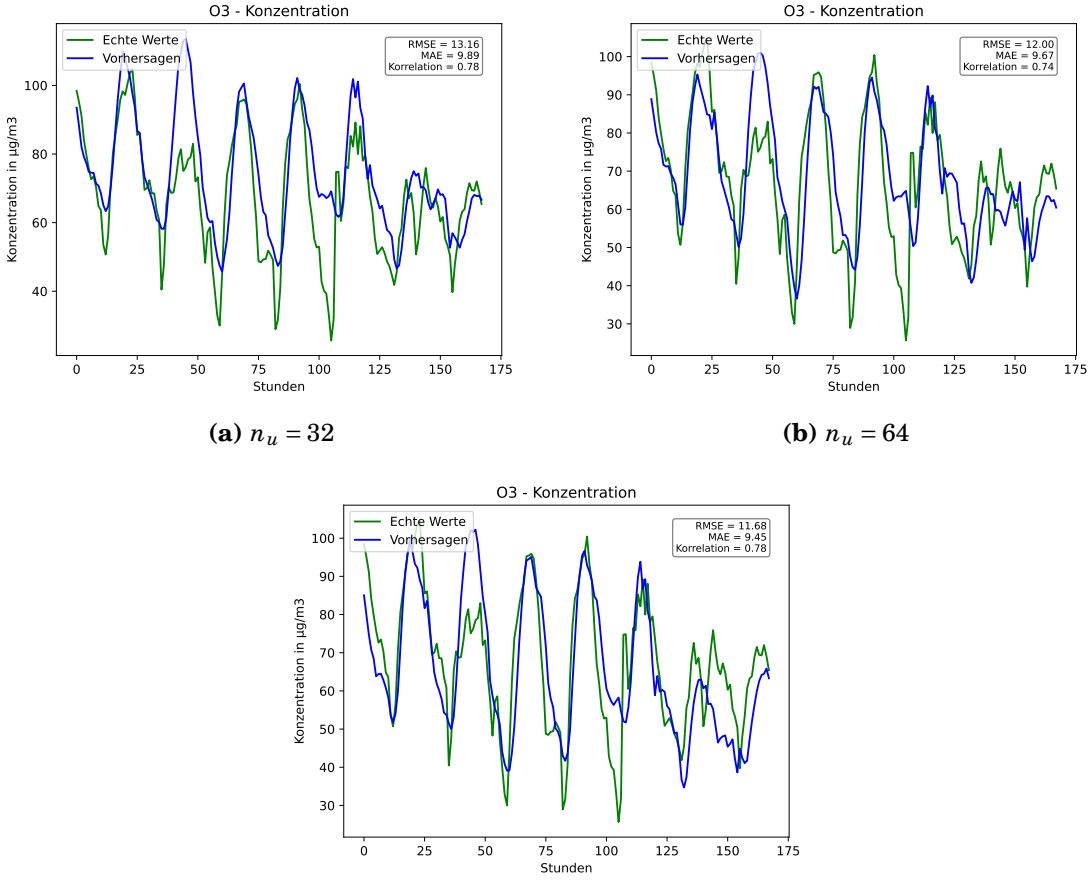


Abbildung 4.3: Balkendiagramm der Trainings- und Testfehler für verschiedene Inputfeature-Konfigurationen n_f. Der MSE bezieht sich auf die zwischen 0 und 1 skalierten Daten während des Trainings. (Quelle: eigene Darstellung)

Als Nächstes wurde untersucht, welchen Einfluss die Grösse der Outputdimensionen n_u und der Parameter t für verschiedene Δt haben.
In der Tabelle 4.1 sind die Fehlermetriken für die getesteten n_u und t über den gesamten Trainingsdatensatz zu sehen. Fett gedruckt sind jeweils die besten Fehlermetriken für ein bestimmtes Δt. Wie der Analyse zu entnehmen ist, lieferte das Modell mit der Outputdimension 128 des LSTM-Layers für $\Delta t = 1$ und $\Delta t = 12$ die präzisesten Vorhersagen. Für $\Delta t = 24$ waren die des Modells mit $n_u = 64$ leicht besser. Man sieht auch, dass die geringsten Fehlermetriken für die meisten getesteten Modellkonfigurationen bei $t = 12$ erzielt wurden. Ebenfalls wird deutlich, dass die Modelle mit höherer Komplexität, also die mit $n_u = 64$ und $n_u = 128$, im Schnitt bessere Vorhersagen lieferten, als das kleinere Modell mit $n_u = 32$.
Abbildung 4.4 zeigt einen Vergleich zwischen den LSTM-Outputdimensionen n_u, der die numerischen Ergebnisse veranschaulicht.



(a) $n_u = 32$

(b) $n_u = 64$

(c) $n_u = 128$

Abbildung 4.4: Graphen und Fehlermetriken der Vorhersagen des Modells mit $t = 12$ und $\Delta t = 12$ für verschiedene Outputdimensionen n_u des LSTM-Layers. (Quelle: eigene Darstellung)

In Abbildung 4.5 ist zu erkennen, dass die Modelle den Trend der Daten in Zeitspannen mit relativ periodischen Veränderungen der Ozonkonzentration gut erfassen können. Dabei fällt auch auf, dass die Vorhersagekurve des Modells deutlich geschmeidiger verläuft als die tatsächlichen Daten und wesentlich weniger Zacken aufweist.

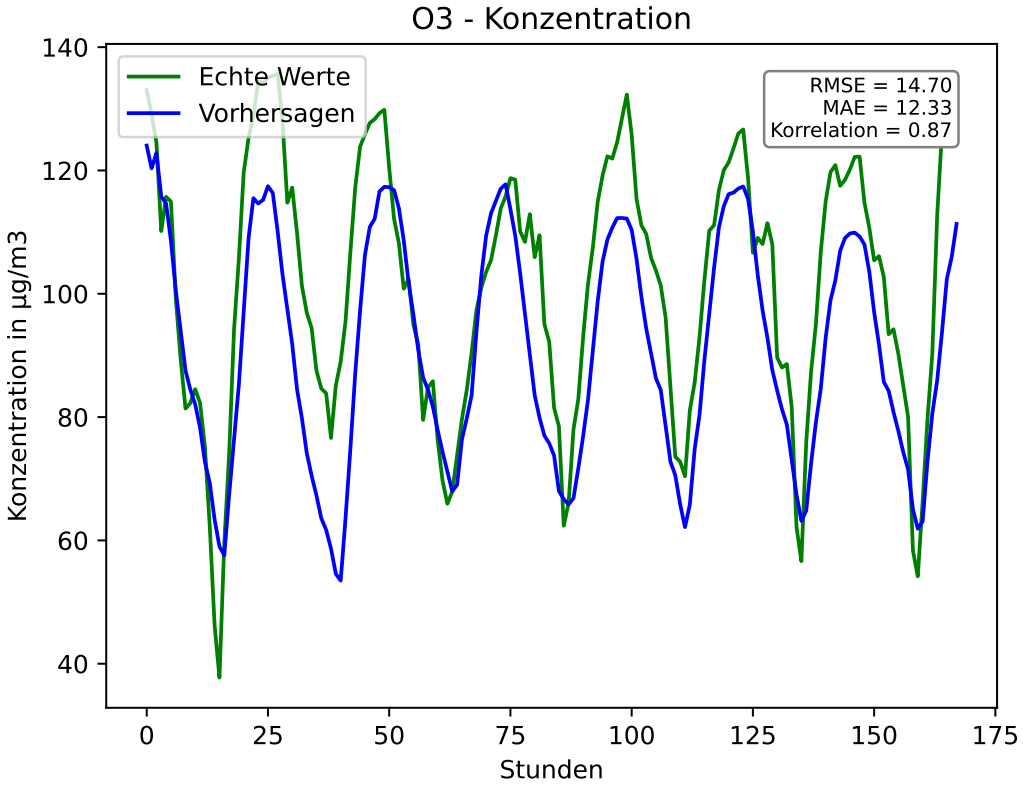


Abbildung 4.5: Graph und Fehlermetriken der Vorhersagen eines Modells mit $n_u = 64$ und $t = 12$ für $\Delta t = 24$. (Quelle: eigene Darstellung)

Für Zeitspannen, die jedoch keine so regelmässige Periodizität aufweisen, haben die Modelle etwas Schwierigkeiten, präzise Voraussagen zu machen. Wie man der Abbildung 4.6 entnimmt, kann das Modell diese plötzlichen, etwas unregelmässigen Fluktuationen der Ozonkonzentration, die im Bild als vertikal gerichtete Zacken sichtbar sind, nicht gut vorhersagen. Auch ersichtlich ist, wie das Modell von einer periodischen Veränderung der Ozonkonzentration ausgeht und deswegen zwischen der Stunde 100 und der Stunde 125 in der Abbildung eine Erhöhung der Ozonkonzentration vorhersagt. Die Ozonkonzentration bleibt in der Wirklichkeit jedoch tief. Ebenfalls ist zu erkennen, dass das Modell – mit Ausnahme des Bereichs zwischen Stunde 100 und Stunde 125 – die Ausschwankungen der Konzentration mit einer geringeren Amplitude vorhersagt als die tatsächlichen Werte.

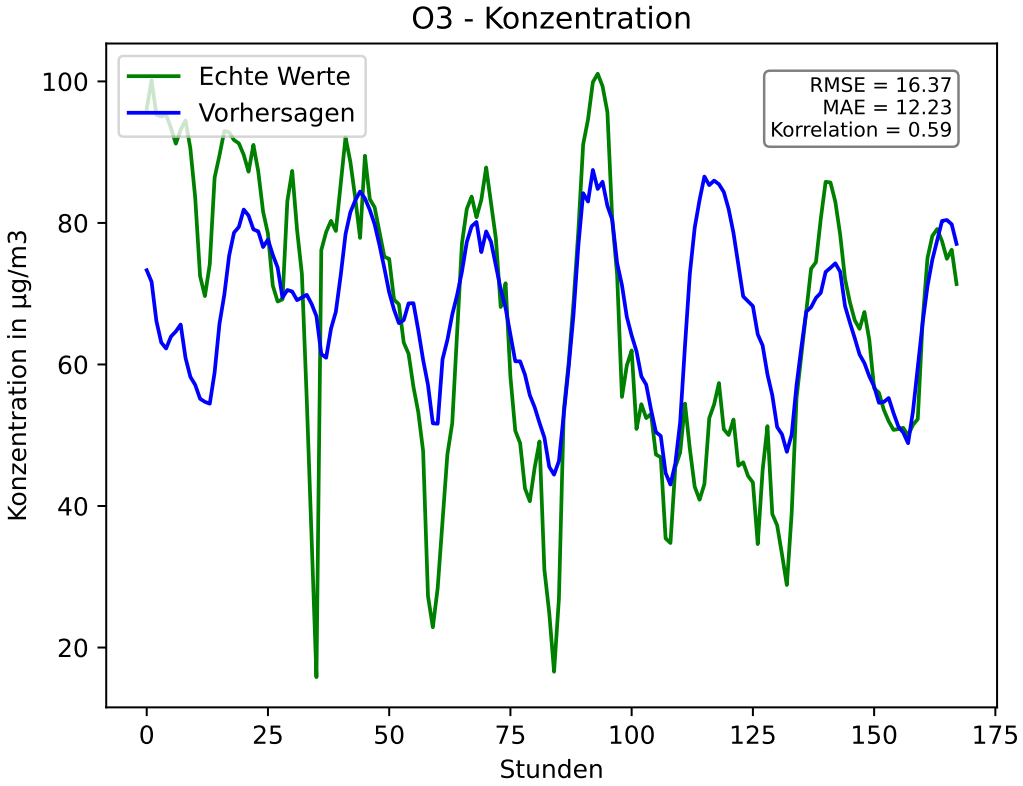
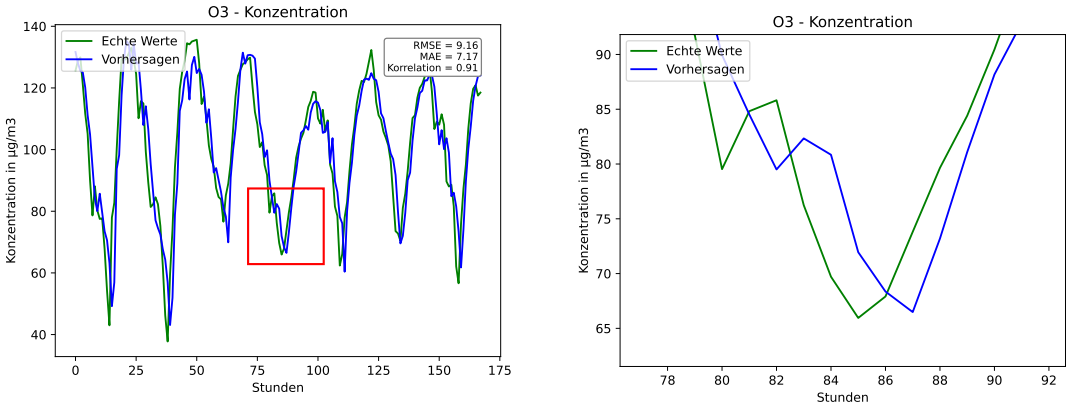


Abbildung 4.6: Graph und Fehlermetriken der Vorhersagen eines Modells mit $n_u = 64$ und $t = 12$ für $\Delta t = 24$. (Quelle: eigene Darstellung)

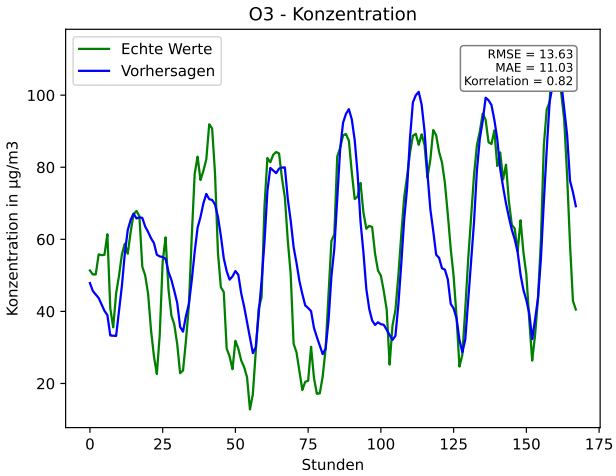
Eine weitere Beobachtung, die nicht direkt aus der Tabelle mit den Fehlermetriken hervorgeht, ist, dass die Modelle für $\Delta t = 1$ teilweise dazu neigen, die Muster in den Datenveränderungen nicht wirklich zu erlernen. Stattdessen geben sie häufig den Wert der vorangegangenen Stunde als Vorhersage aus. Dies wird in Abbildung 4.7 deutlich. Dort zeigt sich, dass die Vorhersagen nicht mehr geschmeidig, sondern relativ sprunghaft sind und der Verlauf der vorhergesagten Konzentrationen an einigen Stellen dem um eine Stunde verschobenen Verlauf der tatsächlichen Konzentrationen stark ähnelt.



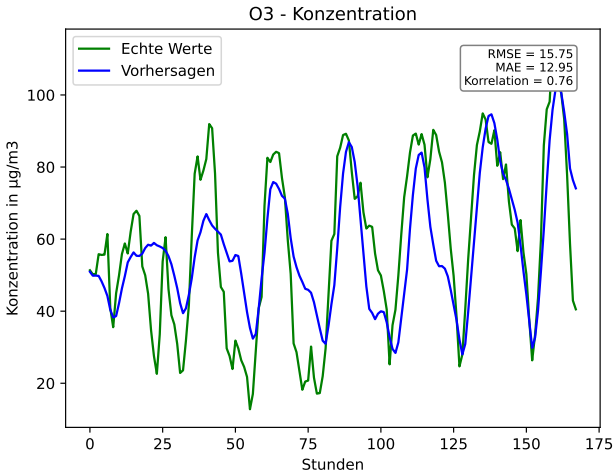
(a) **(b)**

Abbildung 4.7: Graphen der Vorhersagen des Modells mit $n_u = 32$ und $t = 12$ für $\Delta t = 1$: Vorhersagen über 168 Stunden (a) und Vergrösserung des rot umrahmten Bereichs (b). (Quelle: eigene Darstellung)

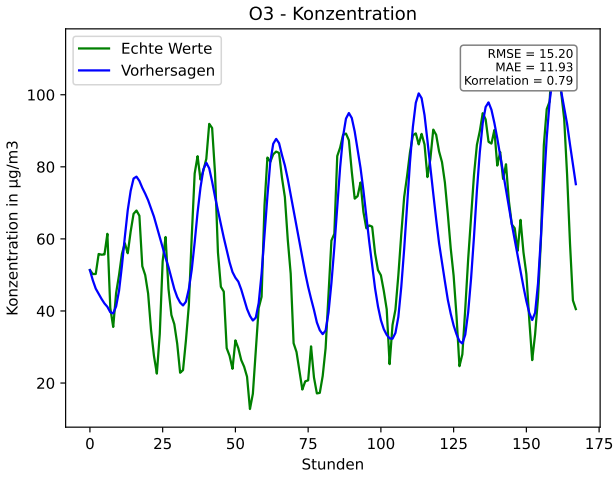
Schliesslich wird untersucht, ob das Hinzufügen eines oder zwei weiteren LSTM-Layer zu einer Verbesserung der Vorhersagen führen kann. Die Tabellen 4.2 und 4.3 zeigen die Fehlermetriken für die Modellvorhersagen über den Testdatensatz für das Modell mit zwei respektive drei LSTM-Layer. Wiederum wurde mit den in Tabelle 3.1 gezeigten Konfigurationen von n_u und t experimentiert. Der Vollständigkeit halber wurden die Fehlermetriken für $\Delta t = 1$ miteinbezogen, obwohl auch bei Modellen mit mehreren LSTM-Layern festgestellt wurde, dass die Vorhersage der Modelle lediglich den Ozonwert der vorherigen Stunde wiedergibt.
Aus den Tabellen lässt sich feststellen, dass das Modell mit nur einem LSTM-Layer die besten Fehlermetriken für alle Δt erzielte. Zudem zeigt sich eine Tendenz, dass die Vorhersagegenauigkeit im Durchschnitt abnimmt, je mehr LSTM-Layer die Modelle enthalten.
In Abbildung 4.8 ist erkennbar, dass die Vorhersagekurven der Modelle mit mehreren LSTM-Layern geschmeidiger und gleichmässiger verlaufen, während die des Modells mit nur einem LSTM-Layer etwas kantiger sind und dem Muster der echten Daten präziser folgen.
Wie in Abbildung 4.9 zu sehen ist, zeigen alle drei Arten von LSTM-Modellen eine deutliche Tendenz zum Overfitting. Dies wird daran erkennbar, dass die Verlustfunktion (MSE) für den Testdatensatz während des Trainings bereits nach wenigen Epochen aufhörte zu sinken und stattdessen wieder anstieg. Ausserdem fällt auf, dass Modelle mit einer höheren Anzahl an LSTM-Layern stärker zum Overfitting neigen. Dies erkennt man daran, dass das Training nach einer geringeren Anzhal an Epochen automatisch gestoppt wurde.



(a) 1 LSTM-Layer

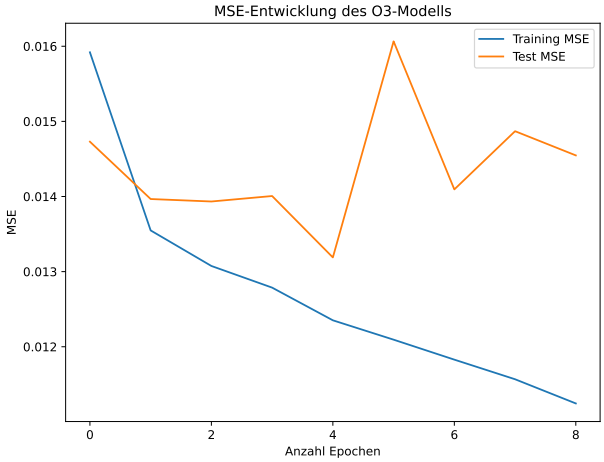


(b) 2 LSTM-Layer

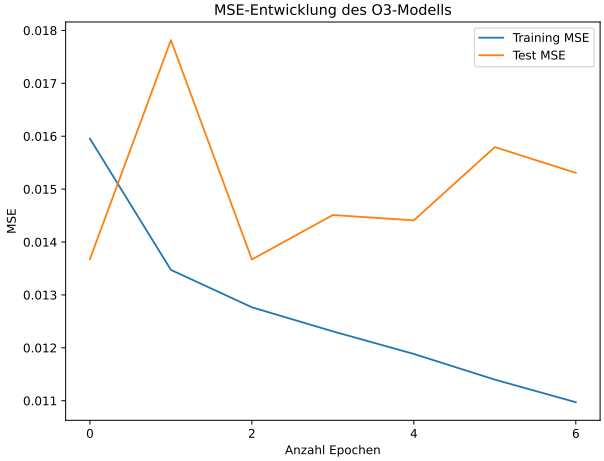


(c) 3 LSTM-Layer

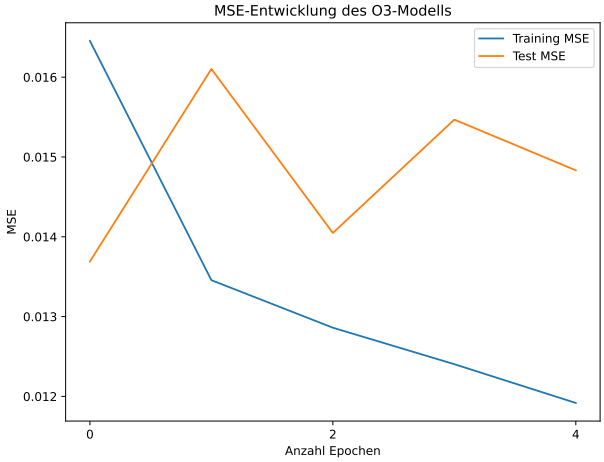
Abbildung 4.8: Graphen und Fehlermetriken der Vorhersagen des Modells mit $n_u = 32$ und $t = 12$ für $\Delta t = 24$ bei unterschiedlicher Anzahl an LSTM-Layern. (Quelle: eigene Darstellung)



(a) 1 LSTM-Layer



(b) 2 LSMT-Layer



(c) 3 LSTM-Layer

Abbildung 4.9: Graphen der Entwicklung des Trainings und Test-MSE des Modells mit $n_u = 32$ und $t = 12$ für $\Delta t = 24$ bei unterschiedlicher an Anzahl LSTM-Layern. (Quelle: eigene Darstellung)

Tabelle 4.1: Tabelle mit Fehlermetriken für verschiedene Konfigurationen der LSTM-Outputdimension n_u und des Parameters t für das Modell mit einem LSTM-Layer. Die besten Werte der Fehlermetriken für jedes Δt sind fett hervorgehoben.

$n_u = 32$		t			
		$t = 6$	$t = 12$	$t = 24$	$t = 36$
$\Delta t = 1$	RMSE:	10.87	10.88	10.85	10.64
	MAE:	7.96	7.87	7.92	7.73
	Korrelation:	0.93	0.93	0.93	0.93
$\Delta t = 12$	RMSE:	16.87	16.43	16.94	17.15
	MAE:	13.46	13.02	13.54	13.69
	Korrelation:	0.81	0.82	0.81	0.81
$\Delta t = 24$	RMSE:	18.08	17.85	18.82	18.07
	MAE:	14.80	14.36	15.30	14.70
	Korrelation:	0.78	0.78	0.76	0.78
$n_u = 64$		t			
		$t = 6$	$t = 12$	$t = 24$	$t = 36$
$\Delta t = 1$	RMSE:	10.88	10.89	11.08	10.84
	MAE:	7.91	7.93	8.05	7.85
	Korrelation:	0.93	0.93	0.92	0.93
$\Delta t = 12$	RMSE:	16.47	16.36	16.31	16.67
	MAE:	13.04	13.14	13.19	13.26
	Korrelation:	0.82	0.82	0.82	0.81
$\Delta t = 24$	RMSE:	17.92	**17.58**	18.41	18.16
	MAE:	14.58	**14.15**	14.87	14.72
	Korrelation:	0.78	**0.79**	0.77	0.77
$n_u = 128$		t			
		$t = 6$	$t = 12$	$t = 24$	$t = 36$
$\Delta t = 1$	RMSE:	10.95	10.61	**10.49**	10.74
	MAE:	7.96	7.68	**7.59**	7.81
	Korrelation:	0.92	0.93	**0.93**	0.93
$\Delta t = 12$	RMSE:	16.53	**16.18**	16.67	16.83
	MAE:	13.01	**12.93**	13.23	13.44
	Korrelation:	0.82	**0.83**	0.81	0.81
$\Delta t = 24$	RMSE:	17.97	18.20	17.72	17.97
	MAE:	14.44	14.68	14.33	14.60
	Korrelation:	0.78	0.78	0.79	0.78

Tabelle 4.2: Tabelle mit Fehlermetriken für verschiedene Konfigurationen der LSTM-Outputdimension n_u und des Parameters t für das Modell mit zwei LSTM-Layer. Die besten Werte der Fehlermetriken für jedes Δt sind fett hervorgehoben.

$n_u = 32$		t			
		$t = 6$	$t = 12$	$t = 24$	$t = 36$
$\Delta t = 1$	RMSE:	10.98	10.78	10.95	10.56
	MAE:	8.00	7.88	7.97	7.70
	Korrelation:	0.92	0.93	0.93	0.93
$\Delta t = 12$	RMSE:	16.93	16.70	16.49	16.59
	MAE:	13.51	13.43	13.23	13.23
	Korrelation:	0.81	0.82	0.82	0.82
$\Delta t = 24$	RMSE:	17.87	18.22	**17.68**	18.34
	MAE:	14.39	14.87	**14.21**	15.07
	Korrelation:	0.78	0.77	**0.78**	0.77
$n_u = 64$		t			
		$t = 6$	$t = 12$	$t = 24$	$t = 36$
$\Delta t = 1$	RMSE:	10.61	10.66	10.69	10.58
	MAE:	7.70	7.73	7.82	7.69
	Korrelation:	0.93	0.93	0.93	0.93
$\Delta t = 12$	RMSE:	17.09	16.74	16.97	16.91
	MAE:	13.73	13.29	13.62	13.42
	Korrelation:	0.80	0.81	0.81	0.81
$\Delta t = 24$	RMSE:	18.16	18.09	17.90	17.79
	MAE:	14.19	14.73	14.45	14.48
	Korrelation:	0.78	0.78	0.78	0.78
$n_u = 128$		t			
		$t = 6$	$t = 12$	$t = 24$	$t = 36$
$\Delta t = 1$	RMSE:	10.68	11.18	10.79	**10.35**
	MAE:	7.79	8.21	7.79	**7.52**
	Korrelation:	0.93	0.92	0.93	**0.93**
$\Delta t = 12$	RMSE:	17.43	16.65	16.54	**16.32**
	MAE:	13.90	13.26	13.21	**13.11**
	Korrelation:	0.80	0.81	0.82	**0.82**
$\Delta t = 24$	RMSE:	17.92	18.12	18.42	17.95
	MAE:	14.55	14.59	14.99	14.61
	Korrelation:	0.78	0.78	0.76	0.78

Tabelle 4.3: Tabelle mit Fehlermetriken für verschiedene Konfigurationen der LSTM-Outputdimension n_u und des Parameters t für das Modell mit drei LSTM-Layer. Die besten Werte der Fehlermetriken für jedes Δt sind fett hervorgehoben.

$n_u = 32$		t			
		$t = 6$	$t = 12$	$t = 24$	$t = 36$
$\Delta t = 1$	RMSE:	10.77	**10.51**	10.92	10.66
	MAE:	7.81	**7.63**	8.02	7.76
	Korrelation:	0.93	**0.93**	0.93	0.93
$\Delta t = 12$	RMSE:	16.82	16.54	16.44	17.27
	MAE:	13.50	12.89	13.13	13.95
	Korrelation:	0.81	0.82	0.82	0.82
$\Delta t = 24$	RMSE:	**17.74**	18.25	18.35	18.63
	MAE:	**14.55**	14.72	14.67	15.05
	Korrelation:	**0.78**	0.77	0.77	0.76
$n_u = 64$		t			
		$t = 6$	$t = 12$	$t = 24$	$t = 36$
$\Delta t = 1$	RMSE:	10.79	10.70	11.00	10.78
	MAE:	7.87	7.74	8.05	7.89
	Korrelation:	0.93	0.93	0.92	0.93
$\Delta t = 12$	RMSE:	16.64	**16.35**	17.53	17.21
	MAE:	13.20	**12.98**	14.06	13.64
	Korrelation:	0.81	**0.82**	0.79	0.80
$\Delta t = 24$	RMSE:	18.30	18.39	18.07	19.26
	MAE:	14.78	14.77	15.58	15.63
	Korrelation:	0.77	0.77	0.78	0.75
$n_u = 128$		t			
		$t = 6$	$t = 12$	$t = 24$	$t = 36$
$\Delta t = 1$	RMSE:	10.69	10.82	10.95	10.67
	MAE:	7.76	7.91	8.02	7.78
	Korrelation:	0.93	0.93	0.92	0.93
$\Delta t = 12$	RMSE:	17.44	16.44	16.95	17.22
	MAE:	14.08	13.25	13.62	14.01
	Korrelation:	0.80	0.82	0.81	0.81
$\Delta t = 24$	RMSE:	17.78	18.31	18.57	18.35
	MAE:	14.52	14.95	14.71	14.78
	Korrelation:	0.78	0.77	0.77	0.77

5 Interpretation und Diskussion

Die gewonnenen Erkenntnisse und Beobachtungen werden im Folgenden kommentiert und erläutert.
Das erste Experiment zeigte, dass die Modelle davon profitierten, zeitliche Features in den Trainingsdatensatz miteinzubeziehen. Dies bestätigt die Vermutung, dass die LSTM-Modelle dank dieser Informationen die unterliegenden periodischen Muster in der zeitlichen Veränderung der Ozonkonzentration besser erlernen konnten. Zudem zeigte sich, dass die Modelle präzisere Vorhersagen lieferten, wenn neben der Ozonkonzentration der vorherigen Stunden auch die Messwerte anderer Schadstoffkonzentrationen und meteorologischer Daten berücksichtigt wurden. Selbst Features mit einer vergleichsweise geringen Korrelation zur Ozonkonzentration trugen zur Verbesserung der Vorhersagegenauigkeit bei. Eine Studie (Cifuentes et al., 2021), die sich mit der Vorhersage der Ozonkonzentration in Manizales, Kolumbien, befasste, bestätigte ebenfalls, dass die LSTM-Modelle, die zusätzlich zur Ozonkonzentration mit einer Vielzahl weiterer Luftqualitäts- und Meteofeatures trainiert wurden, die präzisesten Vorhersagen erzielten.

Im nächsten Experiment wurde gefunden, dass für ein Modell mit einem LSTM-Layer und einem Dense-Layer die Modelle mit den grösseren LSTM-Outputdimensionen $n_u = 64$ und $n_u = 128$ die besseren Vorhersagen lieferten.
Die Modelle mit einer höheren LSTM-Outputdimension verfügen über mehr trainierbare Parameter (Gewichtungen und Biases), wodurch sie vermutlich komplexere Muster besser modellieren können. Sie verfügen ausserdem über ein grösseres Lang- und Kurzzeitgedächtnis, was vermutlich dazu führte, dass sich die Modelle zeitliche Entwicklungen besser merken konnten.
Es wurde ebenfalls festgestellt, dass die Modelle für ein t von 12 die besten Vorhersagen lieferten. Dies suggeriert, dass die Information über Messungen, die mehr als 12 Stunden in der Vergangenheit liegen, dem Modell kein wertvolles Wissen bringen und die Vorhersagequalität reduzieren.

Weiter wurde festgestellt, dass die Modelle für Zeitspannen, in denen sich die Ozonkonzentration regelmässig und periodisch veränderte, bessere Vorhersagen lieferten als für Zeiträume, in denen der Verlauf der Konzentration von diesem regelmässigen

Muster abwich. Dies lässt sich vermutlich damit erklären, dass LSTM-Modelle in der Lage sind, Trends in den Daten zu erlernen, wie zum Beispiel die Periodizität oder auch eine generelle langfristige Entwicklung des Datenverlaufs in eine bestimmte Richtung, wie etwa eine Erhöhung der Ozonkonzentration über mehrere Tage. Dies führt jedoch dazu, dass plötzliche Unregelmässigkeiten in den echten Daten, die von den erlernten Mustern und Regeln abweichen, vom Modell nicht erwartet werden und daher falsch vorhergesagt werden.

Damit wird eine Schwäche von LSTM-Modellen deutlich: Sie haben Schwierigkeiten mit Unregelmässigkeiten in den Daten. Solche Unregelmässigkeiten sind, wie vermutet, bei Messungen von Gaskonzentrationen zu erwarten, da äussere Faktoren wie beispielsweise einströmende Wärme- und Kaltfronten oder auch wechselnde Windlagen vom Modell nicht vorhergesehen werden.
Lösungen zu diesen Herausforderungen werden bereits im Bereich der Wettervorhersagen eingesetzt. Hier wird unterschieden zwischen lokalen und globalen Modellen (IBM, 2023). Globale Modelle werden trainiert, grobe Wettervorhersagen über den ganzen Globus zu produzieren. Lokale Modelle sind dagegen auf ein kleines Gebiet spezialisiert, in welchem sie präzisere Vorhersagen liefern sollten. Hierbei dienen die Vorhersagen des globalen Modells dem lokalen Modell als Kontext bzw. verhelfen ihm, Faktoren oder Änderungen von ausserhalb des Bereichs, welches das lokale Modell kennt, in seine Vorhersagen miteinzubeziehen.
Bei Luftqualitätsvorhersagen, wie der Vorhersage der Ozonkonzentration, wird bereits experimentiert, ähnliche Verfahren mit lokalen und globalen Modellen zu implementieren. Die EPA (United States Environmental Protection Agency) entwickelt bereits ein solches Modell für Luftqualitätsvorhersagen (EPA, 2024).

Als Nächstes zeigten die Untersuchungen, dass die Modelle für Vorhersagen der Ozonkonzentration der nächsten Stunde ($\Delta t = 1$) zu overfitten tendierten. Sie erlernten nicht die zugrundeliegenden Muster der Datenentwicklung, sondern lieferten als Vorhersage näherungsweise die Ozonkonzentration der vorangehenden Stunde.
Betrachtet man die Entwicklung der Verlustfunktion (MSE) während des Trainings (siehe Abb. 5.1), zeigt sich, dass sie sowohl für die Trainingsdaten als auch für die Testdaten bereits nach einer Epoche auf ein sehr niedriges Niveau sinkt und im Falle der Testdaten danach kaum noch Verbesserungen erzielt. Im Gegenteil, sie steigt teilweise sogar wieder an. Diese Beobachtung lässt vermuten, dass das Problem für das Modell zu wenig komplex ist. Das Modell scheint die Muster in der Entwicklung der Konzentration nicht wirklich zu erlernen, sondern approximiert lediglich den Wert der vorangehenden Stunde, multipliziert mit circa 1.
LSTM-Modelle sind zwar in der Lage, mit ihren trainierbaren Parametern komplexe

Probleme zu modellieren und möglicherweise einen Grossteil der Problemkomplexität zu erfassen. Sie können jedoch keine Muster oder Komplexitäten „erfinden“, bzw. würden in diesem Fall dann overfitten, und sind daher bei geringer Problemkomplexität in ihrem Lernpotenzial stark eingeschränkt.

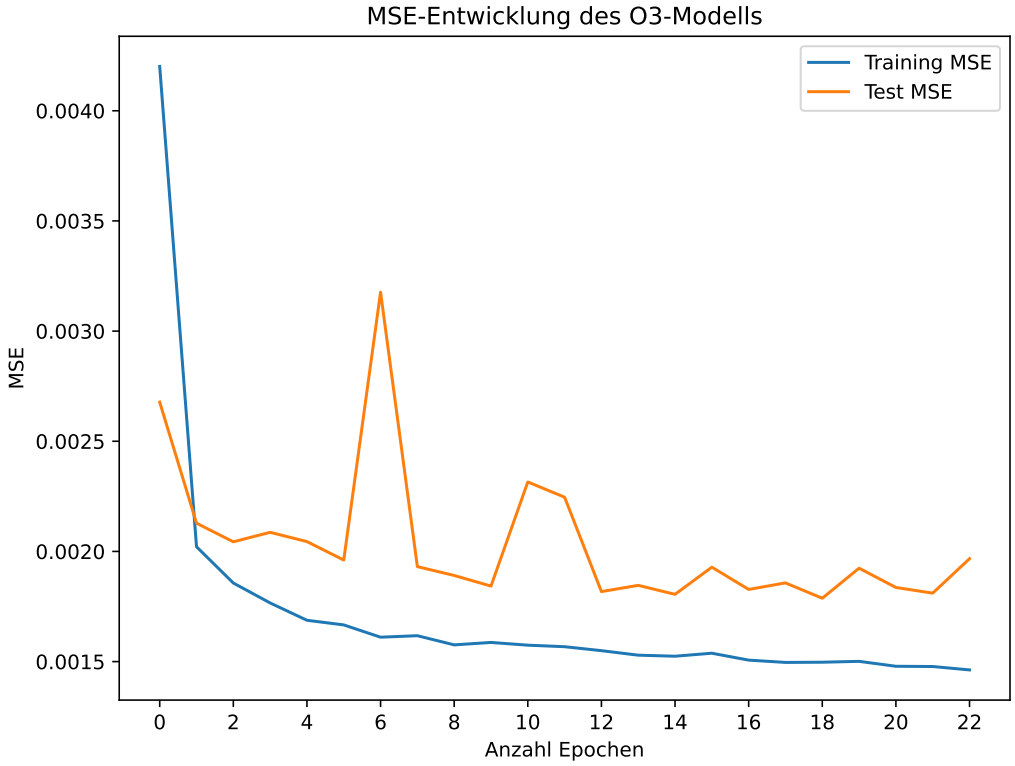


Abbildung 5.1: Im Graph ist die Entwicklung der MSE-Metrik während des Trainings eines Modells mit $n_u = 32$ und $t = 36$ für ein Δt von 1 ersichtlich. (Quelle: eigene Darstellung)

Die geringe Problemkomplexität bei Vorhersagen mit $\Delta t = 1$ wird auf die geringe stündliche Änderungsrate der Ozonkonzentration zurückgeführt, die für die Trainings- und Testdaten im Durchschnitt 5.1% beträgt. Daher liefert das Multiplizieren des vorherigen Wertes mit circa 1 bereits eine relativ präzise Vorhersage, was die Modelle entsprechen erlernten.

Schliesslich wurde untersucht, welchen Effekt eine weitere Erhöhung der Modellkomplexität durch das Hinzufügen eines oder zweier LSTM-Layer auf die Vorhersagequalität hat. Entgegen den Erwartungen wirkte sich das Hinzufügen zusätzlicher LSTM-Layer negativ auf die Vorhersagegenauigkeit aus. Den Modellen gelang es schlechter, feinere Muster in den Daten zu erkennen, und die Vorhersagekurven wurden deutlich gleichmässiger. Dies deutet darauf hin, dass diese Modelle lediglich in der Lage waren, grössere und auffälligere Trends, wie die 24-stündige Periodizität, zu erlernen.

Eine Studie aus Basel (Ryan Prater & Dornberger, 2024) untersuchte ebenfalls LSTM-Modelle für verschiedene Regressionsprobleme und stellte fest, dass eine Erhöhung der Modellkomplexität die Vorhersagequalität negativ beeinflussen kann. Zudem wurde eine bekannte Tatsache bestätigt: LSTM-Modelle benötigen mit zunehmender Anzahl von Layers und damit steigender Komplexität deutlich mehr Trainingsdaten, um von der hohen Anzahl an trainierbaren Parametern zu profitieren, ohne dabei zu overfitten (Srivatsavaya, 2023a). In dieser Arbeit könnte es also sein, dass der Trainingsdatensatz für die LSTM-Modelle mit mehr Layers nicht ausreichend gross war.
Es wurde auch festgestellt, dass alle LSTM-Modelle, unabhängig davon, ob sie aus einem oder mehreren LSTM-Layers bestanden, bereits nach wenigen Epochen zu overfitten begannen. Diese Beobachtung liefert eine wertvolle Erkenntnis über das Lernverhalten von LSTM-Modellen bei der Vorhersage der Ozonkonzentration: Sie erreichen das Minimum der Verlustfunktion für die Testdaten bereits nach wenigen Durchgängen durch den Trainingsdatensatz. Ohne den Einsatz von Methoden wie Early Stopping oder Model Checkpoints neigen sie jedoch dazu, weiter zu trainieren, was zu Overfitting führt und die Vorhersagegenauigkeit für neue, unbekannte Daten deutlich verringert. Der spezifische Grund für dieses Overfitting wurde in dieser Arbeit nicht weiter untersucht, ist jedoch eine bekannte Schwäche von LSTM-Modellen (Kinoyama et al., 2021; Srivatsavaya, 2023b).

Fazit und Ausblick

In dieser Arbeit wurden bedeutende Erkenntnisse über den Einsatz von LSTM-Modellen für die Vorhersage der Ozonkonzentration gewonnen. Es wurde festgestellt, dass die Modelle in der Lage sind, die Muster, die die zeitliche Veränderung der Ozonkonzentration bestimmen, erfolgreich zu erlernen. Zudem zeigte sich, dass, entgegen der Erwartungen, kleinere Modelle mit weniger LSTM-Layern bessere Vorhersagen lieferten.
Für zukünftige Arbeiten wäre es interessant, LSTM-Modelle auch für die Vorhersage der Konzentration anderer Schadstoffe zu testen. Zudem könnte man untersuchen, wie sich die Modelle durch bessere Trainingsmethoden, alternative Architekturen und ergiebigere Datensätze weiter verbessern lassen, um die Vorhersagegenauigkeit zu erhöhen.

Quellenverzeichnis

Appel, K. W., Bash, J. O., Fahey, K. M., Foley, K. M., Gilliam, R. C., Hogrefe, C., Hutzell, W. T., Kang, D., Mathur, R., Murphy, B. N., Napelenok, S. L., Nolte, C. G., Pleim, J. E., Pouliot, G. A., Pye, H. O. T., Ran, L., Roselle, S. J., Sarwar, G., Schwede, D. B., … Wong, D. C. (2021). The Community Multiscale Air Quality (CMAQ) model versions 5.3 and 5.3.1: system updates and evaluation. *Geoscientific Model Development*, *14*(5), 2867–2897. https://doi.org/10.5194/gmd-14-2867-2021 (siehe S. 2).

Brain, G. (2024). *API Documentation*. Zugriff am 24. November 2024 unter https://www.tensorflow.org/api_docs (siehe S. 18).

Browlee, J. (2020). *kNN Imputation for Missing Values in Machine Learning*. Zugriff am 16. November 2024 unter https://machinelearningmastery.com/knn-imputation-for-missing-values-in-machine-learning/ (siehe S. 17).

Brownlee, J. (2019). *Time Series Data Visualization with Python*. Zugriff am 11. November 2024 unter https://machinelearningmastery.com/time-series-data-visualization-with-python/ (siehe S. 11).

Brownlee, J. (2020). *Dropout with LSTM Networks for Time Series Forecasting*. Zugriff am 24. November 2024 unter https://machinelearningmastery.com/use-dropout-lstm-networks-time-series-forecasting/ (siehe S. 18).

Brownlee, J. (2022a). *Difference Between a Batch and an Epoch in a Neural Network*. Zugriff am 2. Januar 2025 unter https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ (siehe S. 21).

Brownlee, J. (2022b). *Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras*. Zugriff am 24. November 2024 unter https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/ (siehe S. 18).

Bundesamt für Umwelt. (2024). *Luftqualität in der Schweiz*. Zugriff am 7. Dezember 2024 unter https://www.bafu.admin.ch/bafu/de/home/themen/luft/fachinformationen/luftqualitaet-in-der-schweiz.html (siehe S. 1).

Chollet, F. (2024a). *Adam*. Zugriff am 30. November 2024 unter https://keras.io/api/optimizers/adam/ (siehe S. 19).

Chollet, F. (2024b). *Callbacks API*. Zugriff am 2. Januar 2025 unter https://keras.io/api/callbacks/ (siehe S. 21).

Chollet, F. (2024c). *Keras 3 API documentation*. Zugriff am 24. November 2024 unter https://keras.io/api/ (siehe S. 18).

Chollet, F. (2024d). *The Sequential class*. Zugriff am 24. November 2024 unter https://keras.io/api/models/sequential/ (siehe S. 19).

Cifuentes, F., Gálvez, A., González, C. M., Orozco-Alzate, M., & Aristizábal, B. H. (2021). Hourly Ozone and PM2.5 Prediction Using Meteorological Data – Alternatives for Cities with Limited Pollutant Information. *Aerosol and Air Quality Research*, *21*(9), 200471. https://doi.org/10.4209/aaqr.200471 (siehe S. 35).

DATAtab Team. (2024). *Pearson Korrelation*. Zugriff am 10. November 2024 unter https://datatab.de/tutorial/pearson-korrelation (siehe S. 12).

EPA. (2024). *The Next-Generation Air Quality Model*. Zugriff am 28. Dezember 2024 unter https://www.epa.gov/cmaq/next-generation-air-quality-model (siehe S. 36).

European Environment Agency. (2024). *How air pollution affects our health*. Zugriff am 7. Dezember 2024 unter https://www.eea.europa.eu/en/topics/in-depth/air-pollution/eow-it-affects-our-health (siehe S. 1).

Fumo, J. (2017). *A Gentle Introduction To Neural Networks Series — Part 1*. Zugriff am 30. Dezember 2024 unter https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc (siehe S. 3).

GeeksforGeeks. (2024a). *Introduction to Recurrent Neural Networks*. Zugriff am 30. Dezember 2024 unter https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/ (siehe S. 4, 6).

GeeksforGeeks. (2024b). *What is Forward Propagation in Neural Networks?* Zugriff am 29. Dezember 2024 unter https://www.geeksforgeeks.org/what-is-forward-propagation-in-neural-networks/ (siehe S. 4).

Geeksforgeeks. (2024). *How KNN Imputer Works in Machine Learning*. Zugriff am 16. November 2024 unter https://www.geeksforgeeks.org/how-knn-imputer-works-in-machine-learning/ (siehe S. 17).

Ghufran Isam Drewil, R. J. A.-B. (2022). Air pollution prediction using LSTM deep learning and metaheuristics algorithms. *Measurement: Sensors*, *24*(100546). https://doi.org/10.1016/j.measen.2022.100546 (siehe S. 21).

Glancszpigel, F. M. (2020). *Basic LSTM model for predicting stock prices (Python)*. Zugriff am 24. November 2024 unter https://fglancszpigel.medium.com/basic-lstm-model-for-predicting-stock-prices-python-2bf6ba470e85 (siehe S. 18).

Gourav Bathla, H. A., Rinkle Rani. (2023). Stocks of year 2020: prediction of high variations in stock prices using LSTM. *Mulitmedia Tools and Applications*, *82*, 9727–9743. https://doi.org/https://doi.org/10.1007/s11042-022-12390-5 (siehe S. 2).

Hamad, R. (2023). *What is LSTM? Introduction to Long Short-Term Memory*. Zugriff am 28. Dezember 2024 unter https://medium.com/@rebeen.jaff/what-is-lstm-introduction-to-long-short-term-memory-66bd3855b9ce#:~:text=LSTM%20networks%20introduce%20memory%20cells,out%20of%20the%20memory%20cell. (siehe S. 5).

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Comput.*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 (siehe S. 3).

Hong, F., Ji, C., Rao, J., Chen, C., & Sun, W. (2023). Hourly ozone level prediction based on the characterization of its periodic behavior via deep learning. *Process Safety and Environmental Protection*, *174*, 28–38. https://doi.org/https://doi.org/10.1016/j.psep.2023.03.059 (siehe S. 5).

IBM. (2023). *What are weather models?* Zugriff am 28. Dezember 2024 unter https://www.ibm.com/think/topics/weather-models#:~:text=For%20a%20computer%20program%20to,in%20regular%20recurring%20time%20steps. (siehe S. 36).

IQAir. (2024). *World's most polluted countries and regions*. Zugriff am 7. Dezember 2024 unter https://www.iqair.com/world-most-polluted-countries (siehe S. 1).

Kinoyama, R., Perez, E. A. M., & Iba, H. (2021). Preventing Overfitting of LSTMs using Ant Colony Optimization. *2021 10th International Congress on Advanced Applied Informatics (IIAI-AAI)*, 343–350. https://doi.org/10.1109/IIAI-AAI53430.2021.00061 (siehe S. 38).

Kyaw Saw Htoon. (2020). *A Guide To KNN Imputation*. Zugriff am 16. November 2024 unter https://medium.com/@kyawsawhtoon/a-guide-to-knn-imputation-95e2dc496e (siehe S. 17).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, *521*, 436–44. https://doi.org/10.1038/nature14539 (siehe S. 4).

Li, Y., & Cao, H. (2018). Prediction for Tourism Flow based on LSTM Neural Network [2017 INTERNATIONAL CONFERENCE ON IDENTIFICATION,INFORMATION AND KNOWLEDGEIN THE INTERNET OF THINGS]. *Procedia Computer Science*, *129*, 277–283. https://doi.org/https://doi.org/10.1016/j.procs.2018.03.076 (siehe S. 2).

Liao Qi, X.L. Pan, Mingm Zhu, Zifa Wang. (2020). *Deep Learning for Air Quality Forecasts. a Review*. Zugriff am 7. Dezember 2024 unter https://www.researchgate.net/publication/344086668_Deep_Learning_for_Air_Quality_Forecasts_a_Review (siehe S. 2).

Luo, Z., Lu, P., Chen, Z., & Liu, R. (2024). Ozone Concentration Estimation and Meteorological Impact Quantification in the Beijing-Tianjin-Hebei Region Based on Machine Learning Models [e2023EA003346 2023EA003346]. *Earth and Space Science*, *11*(2), e2023EA003346. https://doi.org/https://doi.org/10.1029/2023EA003346 (siehe S. 1).

Navares, R., & Aznarte, J. L. (2020). Predicting air quality with deep learning LSTM: Towards comprehensive models. *Ecological Informatics*, *55*, 101019. https://doi.org/https://doi.org/10.1016/j.ecoinf.2019.101019 (siehe S. 2).

Nigam, V. (2018). *Understanding Neural Networks. From neuron to RNN, CNN, and Deep Learning*. Zugriff am 30. Dezember 2024 unter https://medium.com/analytics-vidhya/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90 (siehe S. 1, 3).

Oliphant, T. (2024). *NumPy reference*. Zugriff am 24. November 2024 unter https://numpy.org/doc/stable/reference/index.html#reference (siehe S. 18).

Our World in Data. (2022). *Emissions of air pollutants, World, 1895 to 2022*. Zugriff am 7. Dezember 2024 unter https://ourworldindata.org/grapher/long-run-air-pollution?time=1895..latest (siehe S. 1).

Palma, S. (2023). *Understanding Weight Update in Neural Networks (Part 2)*. Zugriff am 4. Januar 2025 unter https://medium.com/@simon.palma/understanding-weight-update-in-neural-networks-part-2-d445cce09d67 (siehe S. 4).

Pandas. (2024). *pandas documentation*. Zugriff am 14. November 2024 unter https://pandas.pydata.org/docs/index.html (siehe S. 9).

Ritchie, H., Rosado, P., & Roser, M. (2020). Greenhouse gas emissions. *Our World in Data*. Zugriff am 7. Dezember 2024 unter https://ourworldindata.org/greenhouse-gas-emissions (siehe S. 1).

Ryan Prater, T. H., & Dornberger, R. (2024). Generalized Performance of LSTM in Time-Series Forecasting. *Applied Artificial Intelligence*, *38*(1), 2377510. https://doi.org/10.1080/08839514.2024.2377510 (siehe S. 38).

Ryan T.J.J. (2020). *LSTMs Explained: A Complete, Technically Accurate, Conceptual Guide with Keras*. Zugriff am 24. November 2024 unter https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2#:~:text=By%20default%2C%20an%20LSTM%20cell,all%20hidden%20states%E2%80%9D%20Default%3A%20False (siehe S. 18).

scikit-learn. (2024a). *KNNImputer*. Zugriff am 16. November 2024 unter https://scikit-learn.org/1.5/modules/generated/sklearn.impute.KNNImputer.html (siehe S. 17).

scikit-learn. (2024b). *MinMaxScaler*. Zugriff am 16. November 2024 unter https://scikit-learn.org/1.5/modules/generated/sklearn.preprocessing.MinMaxScaler.html (siehe S. 17).

Srivatsavaya, P. (2023a). *Advantages and Disadvantages of Using Multiple LSTM Layers*. Zugriff am 2. Januar 2025 unter https://medium.com/@prudhviraju.srivatsavaya/advantages-and-disadvantages-of-using-multiple-lstm-layers-9c513cd0002c (siehe S. 38).

Srivatsavaya, P. (2023b). *LSTM — Implementation, Advantages and Diadvantages*. Zugriff am 2. Januar 2025 unter https://medium.com/@prudhviraju.srivatsavaya/lstm-implementation-advantages-and-diadvantages-914a96fa0acb (siehe S. 38).

Statistics Solutions. (2024). *Pearson's Correlation Coefficient: A Comprehensive Overview*. Zugriff am 9. November 2024 unter https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/pearsons-correlation-coefficient/#:~:text=Perfect%3A%20Values%20near%20%C2%B11,0.49%20indicate%20a%20moderate%20correlation (siehe S. 12).

Wikipedia. (2024a). *Activation function*. Zugriff am 29. Dezember 2024 unter https://en.wikipedia.org/wiki/Activation_function (siehe S. 3).

Wikipedia. (2024b). *Autoregressive integrated moving average*. Zugriff am 7. Dezember 2024 unter https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average (siehe S. 2).

Wikipedia. (2024c). *Backpropagation*. Zugriff am 29. Dezember 2024 unter https://en.wikipedia.org/wiki/Backpropagation (siehe S. 4).

Wikipedia. (2024d). *Box-Plot*. Zugriff am 11. November 2024 unter https://de.wikipedia.org/wiki/Box-Plot (siehe S. 15).

Wikipedia. (2024e). *Euklidischer Abstand*. Zugriff am 16. November 2024 unter https://de.wikipedia.org/wiki/Euklidischer_Abstand (siehe S. 17).

Wikipedia. (2024f). *Histogramm*. Zugriff am 11. November 2024 unter https://de.wikipedia.org/wiki/Histogramm (siehe S. 15).

Wikipedia. (2024g). *Linear regression*. Zugriff am 7. Dezember 2024 unter https://en.wikipedia.org/wiki/Linear_regression (siehe S. 2).

Wikipedia. (2024h). *Mean absolute error*. Zugriff am 27. Dezember 2024 unter https://en.wikipedia.org/wiki/Mean_absolute_error (siehe S. 23).

Wikipedia. (2024i). *Mean squared error*. Zugriff am 24. November 2024 unter https://en.wikipedia.org/wiki/Mean_squared_error#Definition_and_basic_properties (siehe S. 18).

Wikipedia. (2024j). *Outlier*. Zugriff am 11. November 2024 unter https://en.wikipedia.org/wiki/Outlier (siehe S. 15).

Wikipedia. (2024k). *Overfitting*. Zugriff am 6. Dezember 2024 unter https://en.wikipedia.org/wiki/Overfitting (siehe S. 21).

Wikipedia. (2024l). *Pearson correlation coefficient*. Zugriff am 9. November 2024 unter https://en.wikipedia.org/wiki/Pearson_correlation_coefficient (siehe S. 11).

Wikipedia. (2024m). *Root mean square deviation*. Zugriff am 27. Dezember 2024 unter https://en.wikipedia.org/wiki/Root_mean_square_deviation (siehe S. 23).

World Health Organization. (2021). *What are the WHO Air quality guidelines?* Zugriff am 7. Dezember 2024 unter https://www.who.int/news-room/feature-stories/detail/what-are-the-who-air-quality-guidelines (siehe S. 1).

Wu, Z., Tian, Y., Li, M., Wang, B., Quan, Y., & Liu, J. (2024). Prediction of air pollutant concentrations based on the long short-term memory neural network. *Journal of Hazardous Materials*, *465*, 133099. https://doi.org/https://doi.org/10.1016/j.jhazmat.2023.133099 (siehe S. 2).

Yan, S. (2016). *Understanding LSTM and its diagrams*. Zugriff am 29. Dezember 2024 unter https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714 (siehe S. 6).

Yi, M. (2024). *A complete guide to histograms*. Zugriff am 11. November 2024 unter https://www.atlassian.com/data/charts/histogram-complete-guide#:~:text=What%20is%20a%20histogram%3F,value%20within%20the%20corresponding%20bin. (siehe S. 15).

Zürich, S. (2024). *Stadt Zürich Open Data*. Zugriff am 16. Dezember 2024 unter https://data.stadt-zuerich.ch/dataset (siehe S. 7, 9).

Eigenständigkeitserklärung

Hiermit bestatige ich, dass ich meine Maturitatsarbeit selbststandig und nur unter Zuhilfenahme der in den Verzeichnissen oder in den Anmerkungen genannten Quellen und KI-/LLM-Tools angefertigt habe. Die Mitwirkung von anderen Personen hat sich auf Beratung und Korrekturlesen beschrankt. Alle verwendeten Unterlagen und Gewahrspersonen sind vollstandig aufgefuhrt.

Ort, Datum:

Unterschrift: