Projekt: Edge AI DLBAIPEAI01_D

Fallstudie

Studiengang: Angewandte Künstliche Intelligenz

Sven Behrens

Matrikelnummer: 42303511

Prof. Dr. Bertram Taetz

12. Februar 2026

Inhaltsverzeichnis

Abbildungsverzeichnis **III**

Tabellenverzeichnis **IV**

Abkürzungsverzeichnis **V**

1 Einleitung **1**

2 Hauptteil **2**

2.1 Projektumgebung 2

2.2 Datengrundlage 2

2.3 Datenvorverarbeitung 3

2.4 Modelltraining 3

2.4.1 Train/Test-Split 3

2.4.2 PM2.5-Modelle 4

2.4.3 NO2-Modelle 4

2.5 Feature Importance und Feature-Reduktion 5

2.6 Edge-Optimierung 5

2.6.1 ONNX-Export 5

2.6.2 Benchmark 5

2.7 Alert-System 5

2.8 Edge-Inference-Pipeline 5

3 Fazit **5**

3.1 Zielerreichung und Projektergebnisse 5

3.2 Kritische Reflexion 5

3.3 Verbesserungspotenziale 5

3.4 Ausblick 5

Projektrepository **5**

Literaturverzeichnis **6**

Verzeichnis der Anhänge **7**

Anhang **7**

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

GRU Gated Recurrent Unit

MAE Mean Absolute Error

ONNX Open Neural Network Exchange

RMSE Root Mean Squared Error

WHO World Health Organization

1 Einleitung

Die Luftqualität in städtischen Gebieten stellt ein wachsendes Gesundheitsrisiko dar. Insbesondere Feinstaub (PM2.5) und Stickstoffdioxid (NO2) werden mit Atemwegserkrankungen, Herz-Kreislauf-Beschwerden und einer erhöhten Mortalität in Verbindung gebracht (Lelieveld et al., 2021; World Health Organization, 2016). Besonders gefährdet sind vulnerable Gruppen wie Kinder, ältere Menschen und Personen mit Vorerkrankungen (European Environment Agency, 2023). An stark befahrenen Straßen in der Nähe von Schulen und Wohngebieten können die Schadstoffkonzentrationen kurzfristig kritische Werte erreichen, ohne dass rechtzeitig Gegenmaßnahmen eingeleitet werden.

Klassische Umweltüberwachungssysteme erfassen zwar kontinuierlich Messdaten, leiten daraus jedoch keine unmittelbaren Handlungsempfehlungen ab. Die Daten werden zentral gesammelt und erst mit Verzögerung ausgewertet. Für zeitkritische Anwendungen wie die dynamische Verkehrssteuerung bei Schadstoffspitzen ist dieser Ansatz unzureichend. Edge AI bietet hier einen vielversprechenden Lösungsansatz: Durch die lokale Ausführung von Vorhersagemodellen direkt an der Messstation können Prognosen in Echtzeit erstellt und Maßnahmen ohne Umweg über zentrale Server ausgelöst werden.

Vor diesem Hintergrund wurde im Rahmen des Moduls „Projekt: Edge AI“ an der IU Internationalen Hochschule ein System entwickelt, das die Luftqualität an der Messstation Hamburg Habichtstraße vorhersagt. Die Station befindet sich an einer stark befahrenen Straße in unmittelbarer Nähe einer Schule und eignet sich daher besonders als Anwendungsfall. Das System prognostiziert sowohl die PM2.5- als auch die NO2-Konzentration eine Stunde im Voraus und leitet daraus ein dreistufiges Ampelsystem ab, das bei erhöhter Belastung automatisch Verkehrsmaßnahmen wie Geschwindigkeitsbegrenzungen oder Umleitungen auslösen kann.

Die methodische Vorgehensweise umfasst mehrere aufeinander aufbauende Schritte. Zunächst werden stündliche Messdaten der OpenAQ-Plattform vorverarbeitet und durch Feature Engineering angereichert. Anschließend werden vier Modelltypen, lineare Regression, Random Forest, Gradient Boosting und Gated Recurrent Unit (GRU), trainiert und verglichen. Durch eine systematische Feature-Importance-Analyse wird das Feature-Set von 28 auf 17 Merkmale reduziert. Die besten Modelle werden in das Open Neural Network Exchange (ONNX)-Format exportiert und hinsichtlich Inferenzzeit und Modellgröße für den Edge-Einsatz evaluiert. Abschließend wird eine Edge-Inference-Pipeline implementiert, die den gesamten Ablauf von der Sensorablesung bis zur Ampelentscheidung abbildet.

Das vorliegende Projekt gliedert sich wie folgt: Nach der Beschreibung der Projektumgebung und der Datengrundlage werden die Vorverarbeitung und das Modelltraining erläutert. Anschließend werden die Edge-Optimierung, das Alert-System und die Inference-Pipeline vorgestellt. Abschließend werden die Ergebnisse kritisch reflektiert und Verbesserungspotenziale aufgezeigt.

2 Hauptteil

2.1 Projektumgebung

Zu Beginn des Projekts wurde ein GitHub-Repository[1] angelegt, um eine nachvollziehbare Versionsverwaltung zu gewährleisten. Als Implementierungssprache wurde Python gewählt. Für die Reproduzierbarkeit wurde eine virtuelle Umgebung mit `venv` eingerichtet, in der alle Abhängigkeiten über eine `requirements.txt` installiert werden. Zu den zentralen Bibliotheken zählen `pandas` und `numpy` für die Datenverarbeitung, `scikit-learn` für die klassischen Machine-Learning-Modelle, `torch` (CPU-Version) sowie `onnxruntime` und `skl2onnx` für den ONNX-Export und die Edge-Inferenz.

Die gesamte Pipeline, von den Rohdaten bis zur Edge-Inference-Demo, lässt sich mit einem einzigen Befehl (`python run_all.py`) ausführen. Das Skript ruft alle Verarbeitungsschritte sequenziell auf und stellt sicher, dass jedes Teilskript im korrekten Arbeitsverzeichnis ausgeführt wird. Dadurch kann ein Tutor die Ergebnisse ohne manuelle Konfiguration reproduzieren.

2.2 Datengrundlage

Für das Training und die Evaluation der Modelle werden reale Luftqualitätsdaten benötigt. Im Vorfeld wurden drei Datenquellen evaluiert: das UCI Air Quality Dataset (De Vito, 2016) (globale Stadte), das Umweltbundesamt (Umweltbundesamt, 2025) (zuverlässig, aber manueller Download) und die OpenAQ-Plattform (OpenAQ, 2025) (globale API mit deutschen Stationen). Die Wahl fiel auf OpenAQ, da die Plattform einen automatisierten API-Zugang bietet und die Aufgabenstellung diese Quelle explizit als Beispiel nennt.

Als Messstation wurde zunächst Hamburg Max-Brauer-Allee II (Straßenstation) in Betracht gezogen. Eine Datenanalyse ergab jedoch, dass für PM2.5 lediglich 486 Datenpunkte aus dem Dezember 2025 vorlagen, während PM10, NO2 und CO über alle zwölf Monate hinweg knapp 8 000 Messpunkte aufwiesen. Da ein vollständiges Jahr für die Abbildung saisonaler Muster notwendig ist, wurde stattdessen die Station Hamburg Habichtstraße (OpenAQ Location ID 3010) gewählt. Diese Straßenstation befindet sich in unmittelbarer Nähe einer Schule und liefert für alle vier Parameter, PM2.5, PM10, NO2 und CO, jeweils rund 7 900 stündliche Messwerte über das gesamte Jahr 2025.

Der Datenabruf erfolgt über ein Python-Skript (`download_openaq.py`), das die OpenAQ v3-API monatsweise abfragt und die Ergebnisse als CSV im Long-Format speichert. Die monatsweise Paginierung ist notwendig, um die API-Limits von 1 000 Datensätzen pro Anfrage einzuhalten. Das resultierende Rohdatenfile enthält rund 31 500 Messungen mit den Spalten `parameter`, `value`, `datetime_utc` und `datetime_local`.

[1] https://github.com/svenb23/EdgeAI

2.3 Datenvorverarbeitung

Die Vorverarbeitung erfolgt in fünf aufeinander aufbauenden Schritten, die jeweils das Ergebnis des vorherigen Schritts einlesen und um neue Merkmale ergänzen.

Im ersten Schritt wird das Rohdatenfile vom Long-Format (eine Zeile pro Messung) in ein Wide-Format (eine Zeile pro Stunde, eine Spalte pro Schadstoff) überführt. Fehlende Werte werden linear interpoliert, wobei Lücken von maximal sechs Stunden geschlossen werden. Verbleibende Lücken werden entfernt.

Der zweite Schritt erzeugt zwölf zeitbasierte Merkmale. Neben den Rohwerten für Stunde, Wochentag und Monat werden binäre Indikatoren für Wochenende und Berufsverkehr (7–9 und 16–18 Uhr) erstellt. Zusätzlich werden Stunde, Monat und Wochentag zyklisch als Sinus- und Kosinuswerte kodiert, um die Periodizität für die Modelle abzubilden.

Im dritten Schritt werden sechs Lag-Features erzeugt, die die Schadstoffwerte der vorangegangenen Stunden als Prädiktoren bereitstellen. Für PM2.5 werden drei Lags (1h, 2h, 3h) erstellt, für NO2, CO und PM10 jeweils ein Lag von einer Stunde.

Der vierte Schritt berechnet gleitende Statistiken über verschiedene Zeitfenster. Für PM2.5 werden die rollierenden Mittelwerte über 3, 6 und 24 Stunden sowie die rollierende Standardabweichung über 3 Stunden berechnet. Diese vier Features erfassen lokale Trends und die kurzfristige Variabilität der Feinstaubkonzentration.

Im fünften Schritt werden drei schadstoffübergreifende Merkmale ergänzt: das Verhältnis PM2.5/PM10 als Indikator für den Anteil von Verbrennungspartikeln am Gesamtfeinstaub sowie die stündliche und dreistündliche Änderungsrate von PM2.5 (Δ1h, Δ3h).

Nach Abschluss aller fünf Schritte umfasst der Datensatz die vier Rohwerte sowie 25 engineerte Merkmale, insgesamt also 29 Spalten. Durch die Lag- und Rolling-Berechnungen gehen die ersten 23 Datenpunkte verloren, sodass der finale Datensatz rund 7 850 stündliche Beobachtungen enthält.

2.4 Modelltraining

2.4.1 Train/Test-Split

Für die Aufteilung des Datensatzes wird ein chronologischer 80/20-Split verwendet. Im Gegensatz zu einem zufälligen Split bewahrt diese Strategie die zeitliche Reihenfolge der Messdaten und verhindert Data Leakage, da das Modell ausschließlich auf vergangenen Daten trainiert und auf zukünftigen Daten evaluiert wird. Die ersten 80 % der Beobachtungen (ca. 6 280 Stunden) bilden den Trainingssatz, die letzten 20 % (ca. 1 570 Stunden) den Testsatz.

Als Zielvariablen werden die PM2.5- und die NO2-Konzentration jeweils eine Stunde in die Zukunft

verschoben. Der Wert von `target_pm25_1h` zum Zeitpunkt t entspricht somit dem tatsächlichen PM2.5-Wert zum Zeitpunkt $t + 1$. Analog wird `target_no2_1h` gebildet. Durch diese Verschiebung lernen die Modelle, aus den aktuellen Messwerten und den engineerten Merkmalen die Schadstoffbelastung der nächsten Stunde vorherzusagen. Die letzte Zeile des Datensatzes, für die kein Zielwert existiert, wird entfernt. Als Eingabemerkmale dienen alle 28 Features aus der Vorverarbeitung.

2.4.2 PM2.5-Modelle

Für die Vorhersage der PM2.5-Konzentration werden vier Modelltypen trainiert, die sich in Komplexität und Modellierungsansatz unterscheiden.

Die lineare Regression dient als Baseline-Modell. Sie wird ohne weitere Hyperparameter mit den Standardeinstellungen von scikit-learn trainiert und liefert einen ersten Referenzwert für die erreichbare Vorhersagegenauigkeit.

Das Random-Forest-Modell verwendet 100 Entscheidungsbäume mit einer maximalen Tiefe von 15. Durch die Begrenzung der Baumtiefe wird Overfitting reduziert, während die Ensemble-Methode dennoch nichtlineare Zusammenhänge erfassen kann. Die Berechnung erfolgt parallelisiert über alle verfügbaren CPU-Kerne.

Das Gradient-Boosting-Modell baut sequenziell 200 Bäume mit einer maximalen Tiefe von 5 und einer Lernrate von 0,1 auf. Die geringere Baumtiefe im Vergleich zum Random Forest kompensiert das Boosting-Verfahren durch die höhere Anzahl an Iterationen. Die Lernrate steuert den Beitrag jedes einzelnen Baumes zum Gesamtmodell.

Das GRU-Netzwerk ist das einzige sequenzielle Modell im Vergleich. Es verarbeitet Eingabesequenzen von sechs aufeinanderfolgenden Stunden, um die Konzentration der siebten Stunde vorherzusagen. Die Architektur besteht aus einer GRU-Schicht mit 32 Hidden Units, gefolgt von einer linearen Ausgabeschicht. Vor dem Training werden alle Features mit einem StandardScaler normalisiert. Das Modell wird über 30 Epochen mit dem Adam-Optimizer (Lernrate 0,001) und einer Batch-Größe von 64 trainiert. Als Verlustfunktion wird der Mean Squared Error verwendet.

Alle vier Modelle werden auf dem Testsatz anhand von Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) und R^2 evaluiert. Die klassischen Modelle werden als Pickle-Dateien gespeichert, das GRU-Modell als PyTorch-State-Dict zusammen mit dem zugehörigen Scaler.

2.4.3 NO2-Modelle

Für die Vorhersage der NO2-Konzentration werden dieselben vier Modellarchitekturen mit identischen Hyperparametern trainiert. Das Eingabe-Feature-Set bleibt unverändert bei 28 Merkmalen, lediglich die

Zielvariable wechselt von `target_pm25_1h` zu `target_no2_1h`. Diese einheitliche Konfiguration ermöglicht einen direkten Vergleich der Modellleistung zwischen beiden Schadstoffen und zeigt, wie gut die PM2.5-orientierten Lag- und Rolling-Features auch für die NO2-Vorhersage geeignet sind.

2.5 Feature Importance und Feature-Reduktion

2.6 Edge-Optimierung

2.6.1 ONNX-Export

2.6.2 Benchmark

2.7 Alert-System

2.8 Edge-Inference-Pipeline

3 Fazit

3.1 Zielerreichung und Projektergebnisse

3.2 Kritische Reflexion

3.3 Verbesserungspotenziale

3.4 Ausblick

Projektrepository

Der vollständige Quellcode ist im GitHub-Repository verfügbar: https://github.com/svenb23/EdgeAI

Literatur

De Vito, S. (2016). *Air Quality*. https://doi.org/10.24432/C59K5F

European Environment Agency. (2023, 24. April). *Air pollution and children's health*. https://www.eea.europa.eu/en/analysis/publications/air-pollution-and-childrens-health

Lelieveld, J., Hahad, O., Daiber, A., & Münzel, T. (2021). Luftverschmutzung und Herz-Kreislauf-Erkrankungen. *Aktuelle Kardiologie*, *10*(06), 510–515. https://doi.org/10.1055/a-1546-7355

OpenAQ. (2025). *OpenAQ – Open Air Quality Data*. https://openaq.org/

Umweltbundesamt. (2025). *Aktuelle Luftdaten*. https://www.umweltbundesamt.de/daten/luft/luftdaten

World Health Organization. (2016, 12. Mai). *Air pollution levels rising in many of the world's poorest cities*. https://www.who.int/news/item/12-05-2016-air-pollution-levels-rising-in-many-of-the-world-s-poorest-cities

Verzeichnis der Anhänge

Anhang