MLVU information sheet

Please include this page in your report either at the start or at the end, before the appendix. Do not change the formatting.

Group number

4

Authors

name	student number
Aleksey Martynyuk	2817134
Tim Schutte	2676847
Laura Weerstra	2676731
Othman Alhorani	2602046

Software used

- Python libraries:
  - tensorflow keras for the LSTM Model
  - sklearn & pandas for preprocessing
  - sqlalchemy, psycopg2 (database connection)
  - shapely, geopandas for spatial processing (rbf)
  - matplotlib for visualization
- Google cloud platform:
  - Cloud SQL - Postgres DB v15
  - Cloud Functions
  - Vertex AI - cloud hosted notebook servers with dedicated GPUs for model training
  - App Engine (python application webserver)

Use of AI tools

- `Built-in Colab Enterprise (Cloud version)`: Jupyter Notebook in-cell tab autocomplete: completing small parts of statements. Can be convenient at times. Not used for logic or generating whole blocks of code.
- `ChatGPT`: For LaTeX formatting issues, matplotlib plots formatting.

Otherwise, we did not use AI tools for generating text or code.

Link to code `https://github.com/storks-amsterdam/mlvu-group4-project`

Group disagreements None

Use of generalized LSTM RNN in forecasting air pollution for geographically spread data collection points

ALEKSEY M. MARTYNYUK, Vrije Universiteit Amsterdam, Netherlands
TIM SCHUTTE, Vrije Universiteit Amsterdam, Netherlands
LAURA WEERSTRA, Vrije Universiteit Amsterdam, Netherlands
OTHMAN ALHORANI, Vrije Universiteit Amsterdam, Netherlands

This report evaluates the efficacy of recurrent neural networks (RNNs), specifically Long Short-Term Memory (LSTM) networks. The research focused on LSTMs due to their effectiveness in processing time-series data and their ability to remember information for long periods -which is crucial for air quality prediction tasks- at different spatial locations using a high-resolution dataset. To construct the dataset, a network of air quality sensors was deployed on a busy street in Amsterdam, collecting several weeks of data. Among the models compared, the Long Short-Term Memory (LSTM) network demonstrated superior performance due to its proficiency in handling time-series data and learning long-term dependencies. The best results were achieved with a two-minute data sampling rate employing MinMax scaling, exhibiting predictive accuracy superior to the baseline model across multiple metrics. However, when applied to discrete spatial locations, the model's predictive accuracy varied, indicating a potential spatial component to air quality that requires further investigation. The research underlines the importance of considering spatial relationships in environmental data and suggests directions for future enhancements including multi-step forecasting and integration of graph neural network techniques.

CCS Concepts: • **Computing methodologies** → **Neural networks**; • **Applied computing** → **Environmental sciences**.

Additional Key Words and Phrases: LSTM, RNN, IoT, Sensors, Air Pollution, Particulate Matter, Amsterdam, Time-Series, Machine Learning, University Project

1 INTRODUCTION

Climate change poses a significant threat, necessitating accurate data measurements to assess the efficacy of emission reduction efforts [5]. The integration of advanced machine learning tools in data analysis and forecasting offers innovative solutions for environmental monitoring, enhancing both efficiency and cost-effectiveness [11].

Storks.ai [7] is focused on a development of a nitrogen emission monitoring system for livestock farms, aiming to identify a machine learning methodology capable of training a generalized model for predicting pollution metrics at specific spatial locations. This research explores the model's performance in generalizing emissions data across varied spatial points to predict emissions at a distinct location. To achieve this, 10 air quality sensors were installed on a busy Amsterdam street, to gather raw data over several weeks to develop a predictive model. This study examines the comparative effectiveness of a model trained on a comprehensive dataset against models trained on geographically segmented subsets. The proximity of sensors, ranging from 20 to 150 meters apart, provides an opportunity to evaluate the spatial variability in pollutant concentrations and their dispersion, essential for model accuracy. An LSTM model was selected for air quality prediction, because of its ability to learn long-range dependencies, a challenge for traditional RNNs. LSTMs, with their unique gating mechanisms, effectively manage information flow, ensuring relevant data retention over prolonged periods [3]. This attribute

Authors' addresses: Aleksey M. Martynyuk, a.m.martynyuk@student.vu.nl, Vrije Universiteit Amsterdam, Netherlands; Tim Schutte, t3.schutte@student.vu.nl, Vrije Universiteit Amsterdam, Netherlands; Laura Weerstra, l.c.m.weerstra@student.vu.nl, Vrije Universiteit Amsterdam, Netherlands; Othman Alhorani, o.alhorani@student.vu.nl, Vrije Universiteit Amsterdam, Netherlands.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2024 Copyright held by the owner/author(s).

renders them highly suitable for sequential data tasks, like time-series prediction. The project's objective is to leverage historical data for precise future timestamp predictions. Although a deep neural network could theoretically learn from all previous timestamp features, this method overlooks the data's sequential nature. Hence, the adoption of the LSTM model, a specialized RNN, is justified for its proficiency in discerning patterns from historical data, aligning with the project's need for a time-series data-compatible architecture [3].

2 DATA COLLECTION

There was decided on a custom data collection over utilizing pre-existing datasets. This was influenced by two primary considerations:

Firstly, the research aligns with Storks' business development objectives, developing a proprietary nitrogen measurement system for farms, based on application of machine learning techniques. This involves gaining proficiency in constructing sensor systems, data collection, and understanding the capabilities of communication and hardware components.

Secondly, the research necessitates a dataset with high spatial and temporal resolution to adequately address specific research questions. Publicly available datasets do not offer needed granularity for environmental metrics, necessitating the generation of a custom dataset to meet these criteria.

2.1 Hardware configuration

To collect data, our own manufactured custom air quality sensors were used, with the Sensirion SEN55 photonic-based measuring device at its core. Complete device houses SEN55, an ESP32 developement board with a built-in WiFi modem, a power converter and a battery cell in a custom 3D-printed case. A picture of the device can be seen in the Appendix A. The sensor deployment spans a two-block radius along Overtoom street, positioned at various heights. Of these, two sensors are connected to the power grid, operating with a sampling frequency of 30 seconds. The remaining eight sensors, powered by batteries, record data at a two-minute interval. Map of the device locations is provided in Appendix B.

The two-minute sampling rate was selected to optimize power usage, as the sensing module and WiFi connection demand significant power. Sensors are programmed to enter a power-saving mode after every two-minute cycle. This scheduling allows for roughly one week of continuous data collection with a 37 Wh battery cell.

During each sampling cycle, sensors initiate measurement, undergo a 10-second calibration phase, and then proceed to gather data. The collected measurements are transmitted via WiFi to a Google Cloud API endpoint in JSON format. Subsequently, the data undergoes validation before being stored in a relational database (Postgres DB version 15).

2.2 Data format

Metrics collected by the sensor are the following:

- Particulate matter: with mass concentration range 0 - 1000 $\mu g/m^3$, and particle size range PM1.0, PM2.5, PM4 and PM10.
- Relative humidity: with range 0 - 90 %RH.
- Temperature: range -10 - 50 °C.
- Volatile Organic Compounds (VOC): with measurement range 0 - 1000 ppm (ethanol equivalent).
- NOx (nitrogen oxide and nitrogen dioxide) index.

Full datasheet available at: https://sensirion.com/media/documents/6791EFA0/62A1F68F/Sensirion_Datasheet_Environmental_Node_SEN5x.pdf

2.3 Limitations

Reading measurements from battery-powered sensors creates specific issues in the raw data collected. These are addressed in the Methodology (see 'Data preprocessing').

2.3.1 Loss of data. Sensors are placed in a busy street (Overtoom) in Amsterdam, where there are many private WiFi access points which creates interference. At times, sensors lose communication link with the WiFi access point and stop transmitting data for some periods. These are usually relatively short (under 10 minutes). This creates gaps in the data for individual sensor locations.

2.3.2 Variance between timestamp of the different sensors. The sensors are not synchronized with each other and send data at particular intervals, with different interval start and end points. Even though they all sleep for exactly two minutes (or 30 seconds for grid connected devices), the starting point is random. Therefore, it can be expected that there will be at least 10 data points for a given 2-minute interval, with the exact timestamp of each data point being different.

2.3.3 Reset of VOC and NOx Indices. Following a forced reboot, which may occur due to network disconnections, the indices for Volatile Organic Compounds (VOC) and Nitrogen Oxides (NOx) are reverted to predefined default values. Subsequently, the accumulation of new historical data commences. It's noteworthy that this reset process exclusively impacts VOC and NOx indices, leaving other measurements unaffected.

2.4 Raw data transformations

To address these limitations, several transformations have been applied:

- Synchronising the beginning of the time window between steps, by averaging data points within the time window [8].
- Removing error values for NOx and VOC metrics, 0 and -1 respectively.
- Imputing values for removed datapoints using moving average window of 10 minutes on each side.
- date time features have been extracted from the timestamp, including hour of day and day of the week among others.

Two datasets have been created, one with a step of 2 minutes, and another with the step of 5 minutes. The former follows the original data more closely and allows for a bigger training dataset. Latter has a more smoothed data since the values are averaged within the 5 minute window. Both datasets have been used in a subsequent model training for comparison. Example of the resulting data for one of the locations (37) can be seen in Figure 1 (A comparable graph for devices #31 and #34 can be found in the Appendix C). There are no 0 values for VOC, but there are still some gaps in data present, for example between March 25th and 27th. Also, notice, that data for device 37 starts from march 15th, while other locations have been gathering data since March 2nd.

Dataset structure consists of timestamp, device location id, mass concentration of PM1.0, PM2.5, PM4 and PM10 respectively, VOC index, NOx index, temperature, humidity, hour of day, and day of week. The summary of two-minute dataset is provided in Table 1. Five-minute dataset has the same structure, but a smaller size of 41128 entries.

3 METHODOLOGY

3.1 Model

In this project the goal of the target of the model is to forecast the different types of local air pollution in the future using the preceding air pollution data. To suit the sequential nature of this data an LSTM-layer is used as the top layer of the model. Recurrent Neural Networks (RNNs) are designed to process sequences by maintaining a hidden state that captures temporal information [1]. The fundamental architecture of an RNN, as depicted on

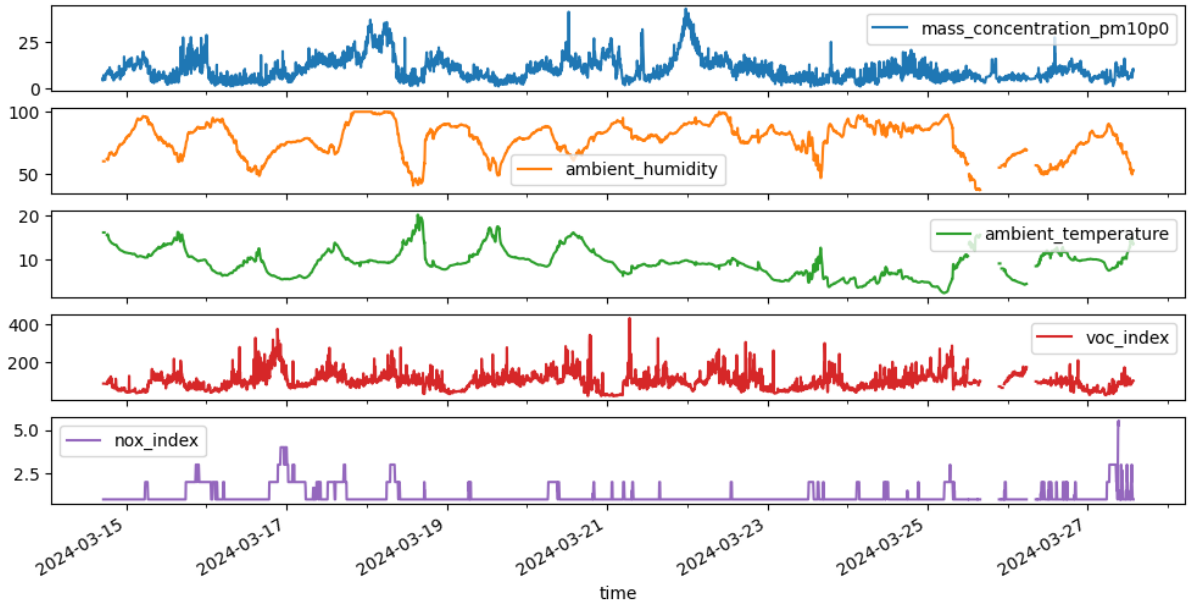

Fig. 1. Formatted sample for device #37 after applying uniform 2 minutes time window.

Column	Non-Null Count	Dtype
time	102760	datetime64[ns]
device_id	102760	int64
mass_concentration_pm1p0	102760	float64
mass_concentration_pm2p5	102760	float64
mass_concentration_pm4p0	102760	float64
mass_concentration_pm10p0	102760	float64
ambient_humidity	101627	float64
ambient_temperature	101627	float64
voc_index	101627	float64
nox_index	101627	float64
hour	102760	float64
minute_of_hour	102760	float64
day_of_week	102760	float64
day_name	102760	object

Table 1. Dataset with two-minute time step.
DatetimeIndex: 102760 entries, 2024-03-02 10:10:00 to 2024-03-27 14:34:00

the left of the image, includes a loop that enables the passage of this hidden state across time steps, effectively allowing the network to 'remember' previous inputs when making predictions. At each time step t, the network receives an input Xt and combines it with the previous hidden state ht-1, processed through a common activation

function such as the hyperbolic tangent (tanh), to produce the current hidden state ht. This state can then be used for predictions and is forwarded to the next time step, creating a temporal chain of information flow through the network [1].

Long Short-Term Memory (LSTM) networks, a specialized form of RNNs shown on the right side of the image, introduce a more intricate structure with gates that regulate information retention and elimination, making them adept at learning long-term dependencies. LSTMs contain three types of gates—forget, input, and output—that jointly manage the cell state and hidden state [6]. The forget gate decides which parts of the cell state may be discarded, allowing the network to 'forget' irrelevant information. The input gate controls the incorporation of new input information into the cell state, while the output gate determines the contribution of the current cell state to the hidden state. These gating mechanisms equip LSTMs with the capability to mitigate the vanishing gradient problem inherent to traditional RNNs [9], hence preserving the integrity of gradients over longer sequences and facilitating the learning of dependencies that span substantial temporal intervals. As a result, LSTMs have emerged as a powerful tool for time-series prediction, natural language processing, and other tasks requiring the modeling of complex temporal dynamics.

3.2 Feature design

3.2.1 Input sequence matrix. The input sequence matrix consists of a sequence of 1d vectors of length 39: 8 metrics of interest, and one-hot encoding of the day of the week (7) and hour of the day (24). From available time dimensions, time of day and day of week was selected as having most direct impact on weather and traffic patterns, which together cause most impact on air quality.

3.2.2 Target vector. The target is a 1d vector of length 8: the main metrics collected, without one-hot encoded time dimensions.

3.3 Data scaling

To ensure that all features contribute equally and do not dominate the optimization process, the features need to be scaled. Choice of the right scaler depends on the nature of data.

All metrics are strictly positive, but have different ranges. A 3-dimensional representation of PM10 (x-axis), NOx (as a color) and VOC (y-axis) is seen in Figure 2.

Unscaled values for mass concentration of PMs vary between 0 and 100. The threshold of 100 was imposed to cap outlier events which are brief and do not necessarily reflect trends, but still cause a change in overall concentration of PMs around the area. Excluding them would remove important events, but keeping original values have cause problems with scaling. Values for VOC index are well distributed between 0 and 500, and do not have outliers. NOx index values range between 0 and 10. Even though the sensing device's range is stated as between 0 and 500, in practice we have not seen values larger than 7, except for a couple of error terms. It is possible that calibration of the device sensitivity is needed to achieve higher measurements. Temperature (respectively humidity) values range between 0 and 30 (respectively 100). For these, no strange values or error measurements were present.

MinMax scalers are most common to use. However, because of the presence of outliers in PM data, it might cause issues. Robust Scaler deals with outliers, but it creates values outside of [0,1) range. Quantile Transformer deals both with outliers and low/high values, but it applies uniform probability density function to the data. Differences in applying these scalers are presented in Figure 3. More detailed charts are also included in Appendix D.

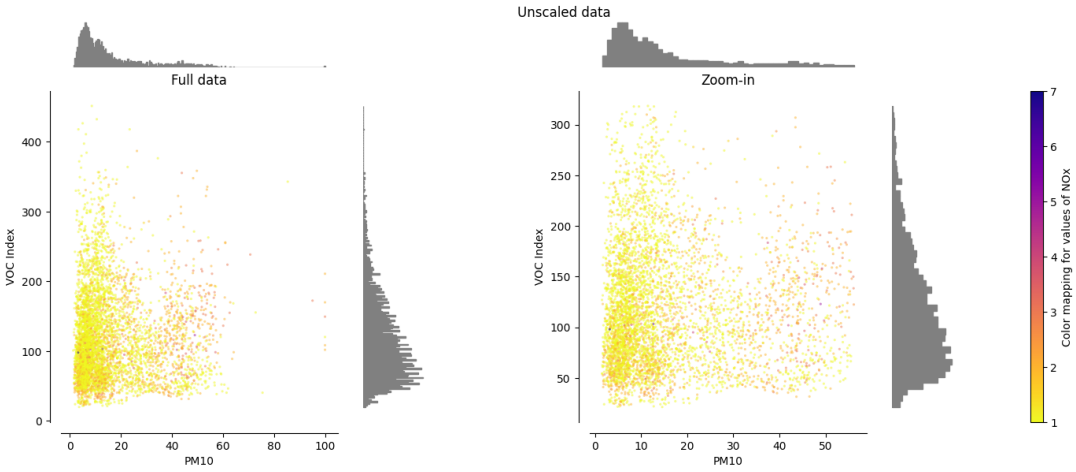

Fig. 2. Distribution of VOC index, NOx index and PM10 unscaled values.

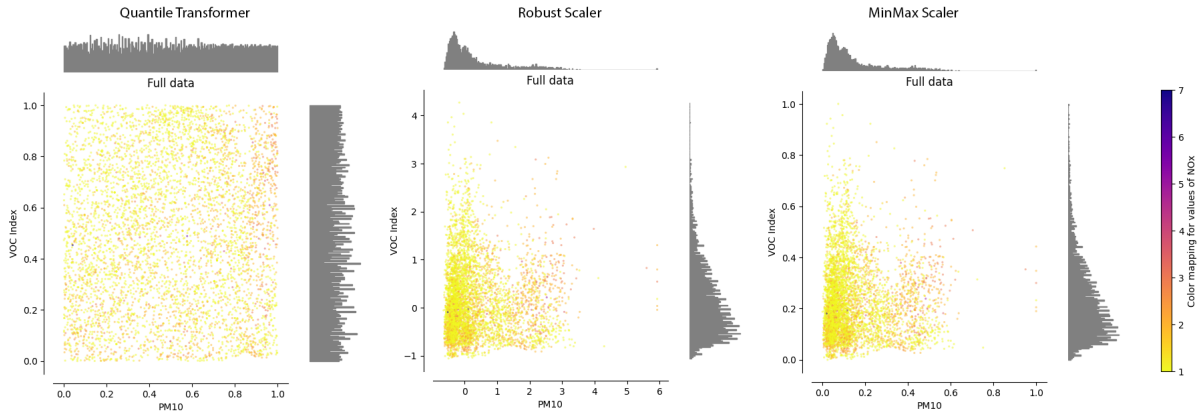

Fig. 3. Comparison of scalers based on VOC index and PM10 values.

3.4 Creating sequences

To create a proper 3-tensor of feature sequences as input for the LSTM layer the data first had to be organized accordingly. The dataset contains the features from different sensors in one table. The samples in each data sequence need to come from the same location for proper forecasting, and the sequences need to be of the same size to be used in one generalized model. Thus, the first step was splitting the dataset into dataframes for each sensor location.

As mentioned earlier, there are gaps in time-series due to data collection issues. When creating sequences, for each gap the datasets either had to be split into smaller datasets or the missing timestamps had to be imputed. We have decided to impute values for gaps less than 10 minutes, and then we tested different thresholds for dataset splitting, between 10 and 30 minutes. Ten-minute maximum gap had best validation results while training.

3.4.1 Length of input sequences. After testing various sequence lengths for the LSTM layer the resulting network ended up with a sequence length of 30. Values for this parameter from 20 up to 50 were tested. Above 30, however, the accuracy of the network did not seem to improve anymore. The training time did increase drastically, for this reason 30 was chosen as a window size for two-minute dataset, and 20 for five-minute dataset (in order for them to cover a similar time-span).

As a last step all the resulting dataframes were converted into a 3 tensor of time sequences and a 2-d array of the targets. Subsequently, the resulting 3 tensors of each dataset were merged back together into one large dataset.

3.5 Topography of the network

The architecture of neural networks often requires empirical experimentation, as no definitive guidelines exist for their configuration. In this study, a range of network topologies was evaluated, with variations in the number of hidden LSTM and Dense layers, as well as the number of nodes within each layer. The architecture ranged from one hidden LSTM layer and one hidden Dense layer to two of each. The nodes were arranged in a descending order, with the top LSTM layer's node count tested between 32 and 248. A final count of 124 nodes was chosen for the top LSTM layer, as higher counts did not result in lower validation loss and tended to exacerbate over-fitting, as indicated by a divergence between training loss and validation loss.

3.6 Hyperparameters

- Number of nodes and layers: The initial setup consisted of two layers (one LSTM and one Dense), expanding to four layers in subsequent trials. Larger node counts, particularly 248 LSTM nodes in the first layer, led to over-fitting, with the validation loss plateauing and exhibiting instability.
- Batch size: Options of 32, 64, and 128 were explored, with 128 emerging as optimal by mitigating over-fitting — a plausible outcome given the volatile nature of data across short time windows.
- Activation functions: In the LSTM layers, the default sigmoid and tanh activations suffice, obviating the need for additional activation functions due to their inherent non-linear expression capabilities.
- Learning rate: Adjustments were made between 0.001 and 0.0003, complemented by a Learning Rate on Plateau scheduler, reducing the learning rate upon stagnation (decay factor of 0.2, patience of 2 epochs, and a floor of 0.0001). This addressed the observed occasional increases in actual loss, suggesting the optimization process was leaping over potential minima.
- Early stopping: Implemented with a monitoring focus on validation loss and a patience of five epochs, early stopping aids in determining an appropriate number of epochs to prevent over-fitting.

These hyperparameters and model configurations were iteratively refined to balance the model's capacity to generalize against the risks of over-fitting, seeking to enhance predictive performance on unseen data. During training, 10% of the data was used for validation. With each training pass, when validation mean absolute error stayed consistently higher than training error, over-fitting was assumed. You can find an example of over-fitting in Appendix E.

4 RESULTS

4.1 Training, validation and test split

For all models, a training and test datasets were created, using 20%/80% split with the help of a built-in scikit-learn `train_and_test` method. Additionally, setting parameter `shuffle` to True reduced over-fitting in some cases.

During training, 10% of the data was used for validation. For each epoch trained, history of training and validation metrics is conveniently printed to the output and stored for later use. The results of training two-minute and five-minute datasets with different scaling is presented in Figures 4-11. Best performance was achieved

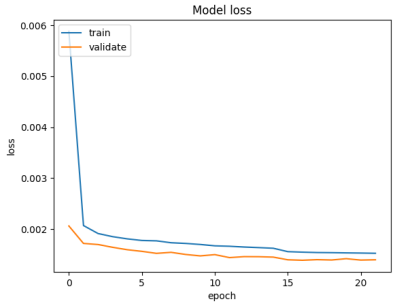

Fig. 4. Loss using two-minute dataset with MinMax scaling.

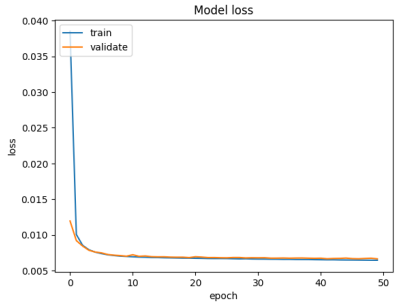

Fig. 5. Loss using two-minute dataset with Quantile scaling.

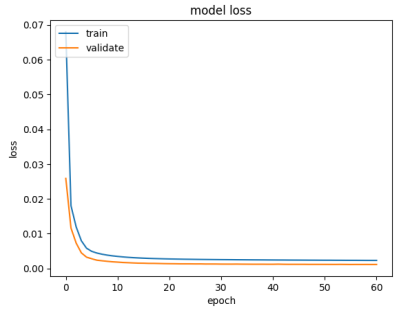

Fig. 6. Loss using five-minute dataset with MinMax scaling.

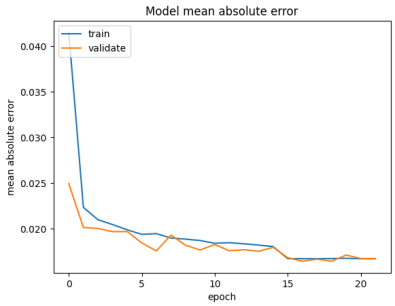

Fig. 7. Mean absolute error using two-minute dataset with MinMax scaling.

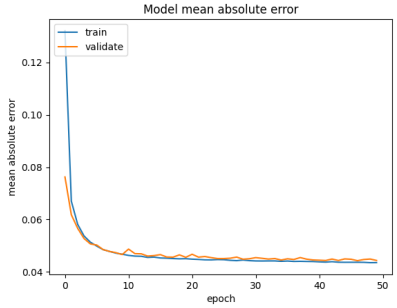

Fig. 8. Mean absolute error using two-minute dataset with Quantile scaling.

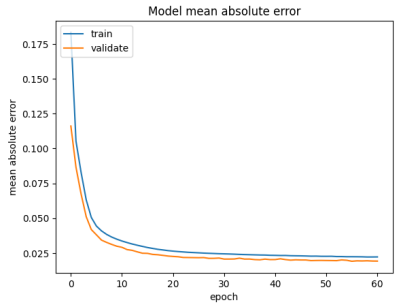

Fig. 9. Mean absolute error using five-minute dataset with MinMax scaling.

with two-minute dataset, using MinMax scaling: A validation error of less than 0.016. Five-minute dataset was only able to achieve a slightly higher error, at 0.019. A model with Quantile Transform applied was also kept, even though it achieved a much higher error, >0.04. The reason for keeping it, is that we think QT scaling made target values a bit larger than MinMax (since it takes care of the outliers), and therefore increased absolute error metrics. In that case, even with a higher error it could still have better prediction results.

4.2 Evaluating model on the test data. Prediction vs. baseline

After finalizing hyperparameteres and settling on several models, a test set was used to evaluate their performance. An output of the model was used to calculate a one step forward in the sequence, creating a prediction error and a baseline. As in similar approaches with LSTM, to measure baseline we used a previous value of the target metric as a base. Comparing prediction error with baseline error, the model using two-minute dataset, with MinMax scaling, showed best performance. For 5 metrics, PM1 PM2.5 PM4, PM10 and Humidity, prediction error was slightly less that the baseline. See result in the Table 2. Other 3 models did not have as good a prediction, with only marginal differences and in fewer metrics (for a full comparison see Appendix F).

Metrics	PM1	PM2.5	PM4	PM10	Humidity	Temperature	VOC Index	NOx Index
Prediction Error	1.862834	1.964249	2.094477	2.18898	17.003253	0.495768	0.148673	0.077545
Baseline Error	1.9762	2.22656	2.428068	2.537578	17.565135	0.372157	0.057658	0.046477

Table 2. Mean absolute error on testing dataset, predicted vs. baseline. Data: two-minute time-series, with MinMax scaling applied. Results are re-scaled.

4.3 Applying general model to unique locations

One of the main motivations for the study is to compare performance of the general model on specific location. After training model to a satisfactory level, and checking performance on a full test dataset, we split test dataset and evaluated model per device location. Results are shown in Table 3. Our assumption was, that if we trained a model using all points, it should create a generalized prediction for any location, with a similar accuracy. However, we found that it was not the case. As seen in the table, there is some difference in mean absolute errors between locations, with some outliers.

Device ID	31	32	33	34	35	36	37	38	39	40
Mean Absolute Error	0.0139	0.0155	0.0158	0.0189	0.0138	0.0166	0.0134	0.0187	0.0225	0.0167
Test Data Length	3483	2920	2716	2536	935	915	1759	1211	1828	1727

Table 3. Mean absolute error and sample size per device location. Model trained on two-minute dataset.

5 DISCUSSION

In assessing the spatial correlation of model performance, radial basis function (RBF) interpolation was employed to interpolate intermediary values for visualizing contour lines across a spatial street representation. This analysis leveraged the mean absolute error from various locations with a linear interpolation function. The resultant contour plots suggest a potential spatial relationship influencing model performance, as shown in Figures 10 and 11. However, these observations remain speculative and necessitate more in-depth analysis and potential model refinements for verification.

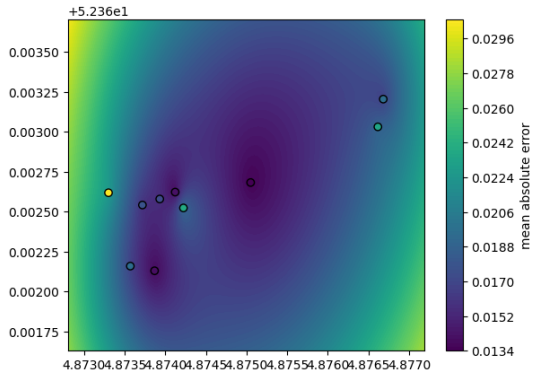

Fig. 10. RBF interpolation of mae for two-min dataset.

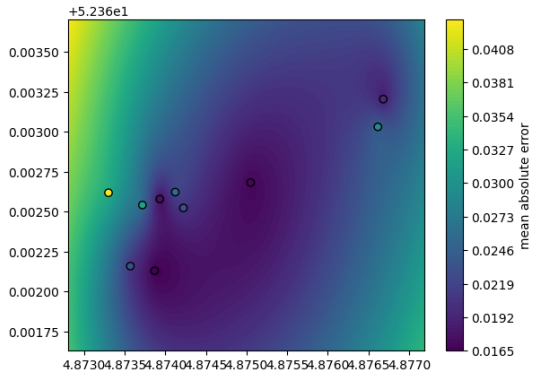

Fig. 11. RBF interpolation of mae for five-min dataset.

Several avenues could enhance this research further:

- Multi-step forecasting:
  The current model's single time-step prediction (t+1) could evolve into multi-step forecasting, predicting multiple future values simultaneously (t+1, t+2, t+3, etc.) [2]. This could be accomplished through either direct-based or iterate-based methods.[10]
- Graph-Enhanced LSTM:
  Incorporating graph neural networks (GNNs) with LSTM architecture could capture complex spatial-temporal interactions amongst data points[4], potentially enriching the model's spatial understanding and performance.
- Mobile sensor data:
  In this project the sensors' locations were fixed. Exploring data from mobile sensors could introduce dynamic spatial components to the model, diversifying the range of detectable environmental factors influencing air quality.
- Model input refinement:
  Presently, one-hot encoded time features are input directly into the LSTM layer. Given their linear nature, it may be more prudent to introduce these as metadata, utilizing multiple inputs within the keras.models.Model construct to inform the model more effectively.
- Feedback loops:
  As new data points are being logged by the sensors, integrating a feedback mechanism would allow the model to compare new incoming data with predicted values and adjust the model if necessary.

6 CONCLUSION

This study embarked on a quest to develop a generalizable machine learning model to predict specific air pollution metrics across varying spatial locations. Utilizing a dataset with high spatial and temporal resolution, generated from ten custom-built sensors deployed along a bustling Amsterdam thoroughfare, the research aimed to harness an LSTM model's strengths in time-series data prediction.

The overarching goal was to employ historical data to accurately forecast future air quality metrics. The training of various models culminated in the identification of an optimally performing model utilizing a two-minute dataset with MinMax scaling, surpassing baseline prediction errors for five critical metrics. However, when this generalized model was tested against individual spatial locations, the results unveiled a notable variance in mean absolute errors, contradicting the expectation of uniform prediction accuracy.

Future research endeavors will explore multi-step forecasting and the integration of graph neural networks with LSTM to enhance predictive capabilities and scrutinize the interconnections between distinct spatial data points. This iterative approach aims to refine the model's predictive precision and expand its analytical breadth, potentially reshaping environmental monitoring practices.

Keywords: LSTM, RNN, air pollution prediction, spatial analysis, time-series data, machine learning, sensor networks.

REFERENCES

[1] J. Brownlee. *An introduction to recurrent neural networks and the math that powers them.* 2024. URL: https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/.

[2] Jason Brownlee. *How to Get Started with Multi-Step Time Series Forecasting.* Accessed: 2024-03-30. n.d. URL: https://machinelearningmastery.com/multi-step-time-series-forecasting/.

[3] Colah. *Understanding LSTM Networks.* Aug. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[4] Zhilong Lu et al. "LSTM variants meet graph neural networks for road speed prediction". In: *Neurocomputing* 400 (2020), pp. 34–45. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.03.031. URL: https://www.sciencedirect.com/science/article/pii/S0925231220303775.

[5] World Health Organization. *Climate change and health.* 2022. URL: https://www.who.int/news-room/fact-sheets/detail/climate-change-and-health.

[6] N. Patel. *LSTMs explained: A complete, technically accurate conceptual guide with Keras.* Aug. 2019. URL: https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2.

[7] *Storks.ai.* https://storks.ai/. Accessed: 2024-04-01. 2024.

[8] Author unknown. *Machine Learning Approaches for Time Series.* https://towardsdatascience.com/ml-approaches-for-time-series-4d44722e48fe. Accessed: 2024-03-30. Publication Year.

[9] Machine Learning for the Visual Understanding Group. *Lecture 07.* 2024. URL: https://mlvu.github.io/lecture07/.

[10] Liu Yunpeng et al. "Multi-step Ahead Time Series Forecasting for Different Data Patterns Based on LSTM Recurrent Neural Network". In: *2017 14th Web Information Systems and Applications Conference (WISA).* 2017, pp. 305–310. DOI: 10.1109/WISA.2017.25.

[11] S. Zhong et al. "Machine Learning: New Ideas and Tools in Environmental Science and Engineering". In: *Environmental Science & Technology* 55.11 (2021), pp. 7094–7105. DOI: 10.1021/acs.est.1c01339. URL: https://doi.org/10.1021/acs.est.1c01339.

A CUSTOM AIR QUALITY SENSOR DEVICE DEVELOPED BY STORKS ROBOTICS LAB

Fig. 12. Microcontroller (ESP32) with SEN55 without battery. Connected to power grid and sends data every 30 seconds.

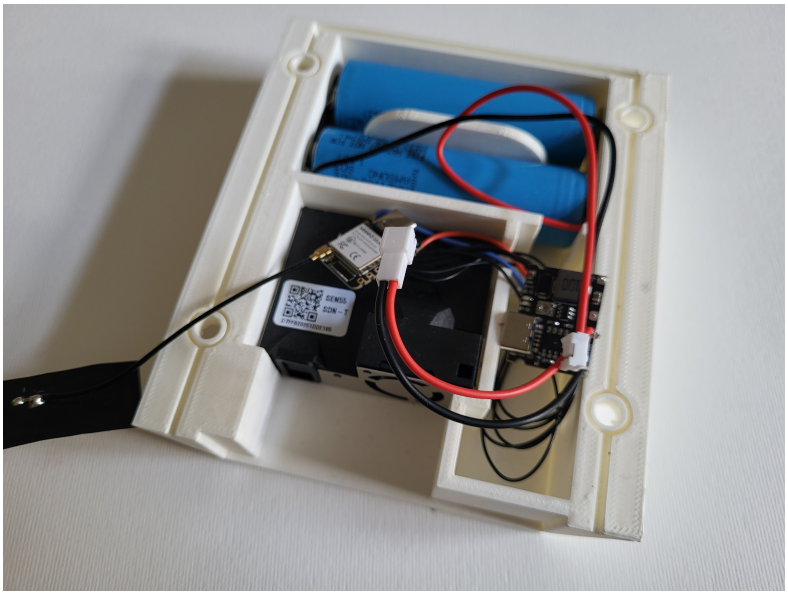

Fig. 13. Microcontroller (ESP32), SEN55, power converter and battery pack. Sends data every 2 minutes.

B MAP OF DEVICE LOCATIONS DISTRIBUTED OVER OVERTOOM STREET IN AMSTERDAM.

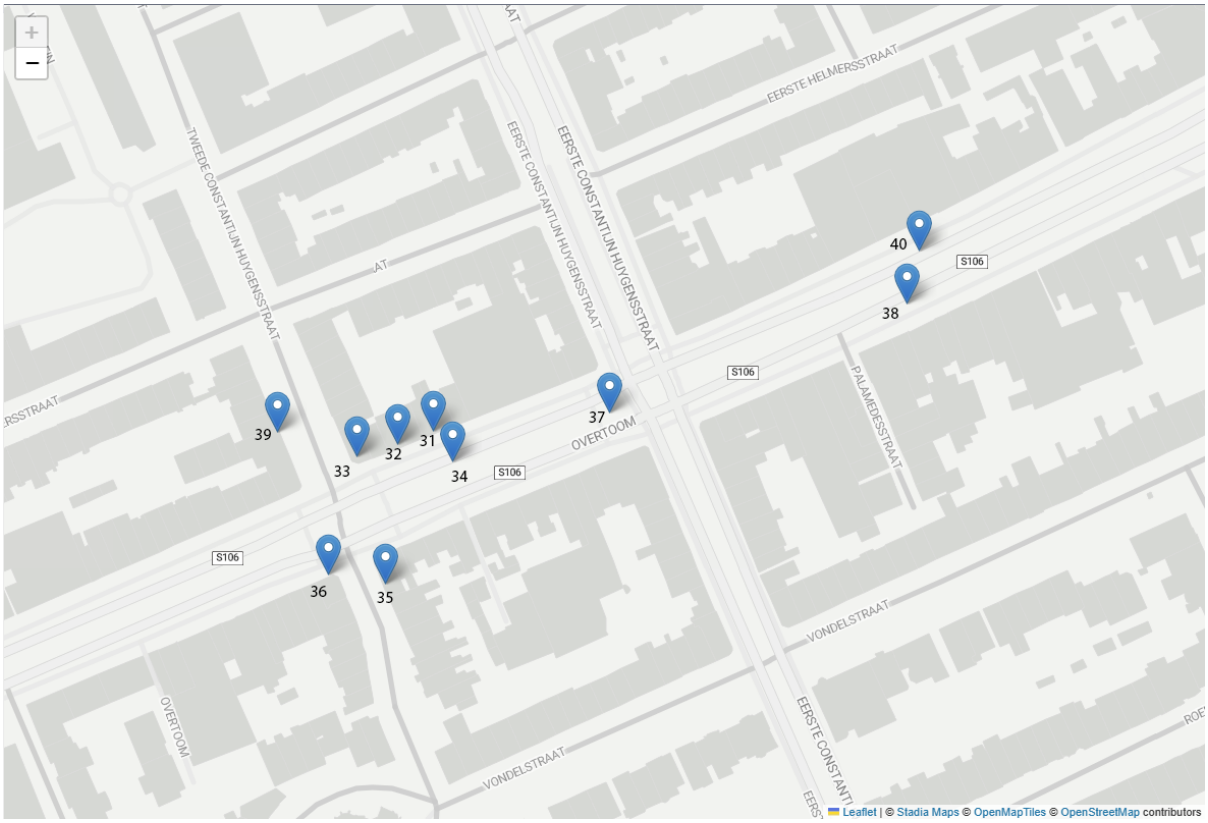

Fig. 14. Location of sensors placed around Overtoom street, Amsterdam.

C TIME-SERIES GRAPH OF PM10, NOX INDEX, VOC INDEX, TEMPERATURE AND HUMIDITY FOR DEVICES 31 AND 34.

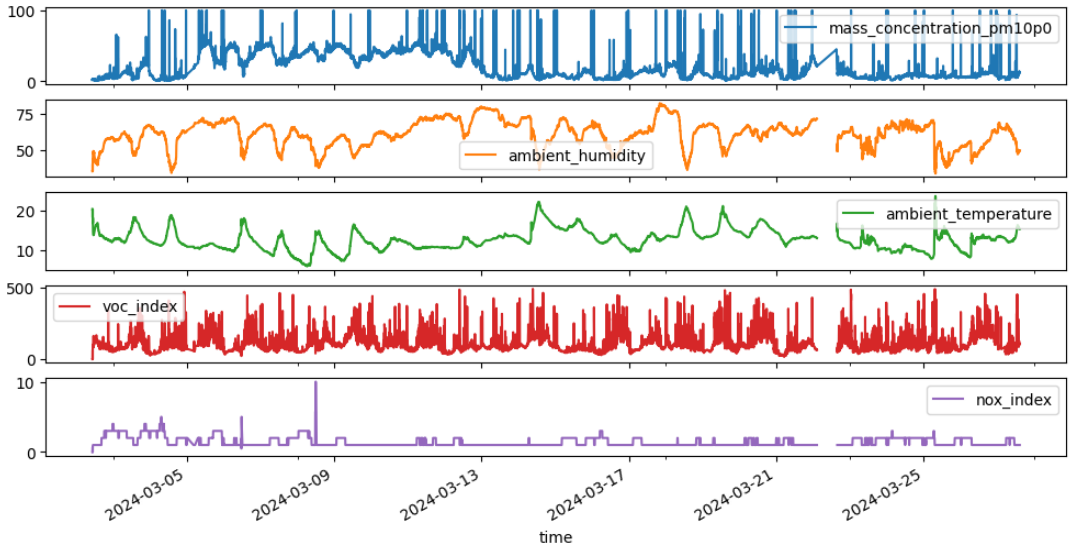

Fig. 15. Formatted sample for device #31 after applying uniform 2 minutes time window.

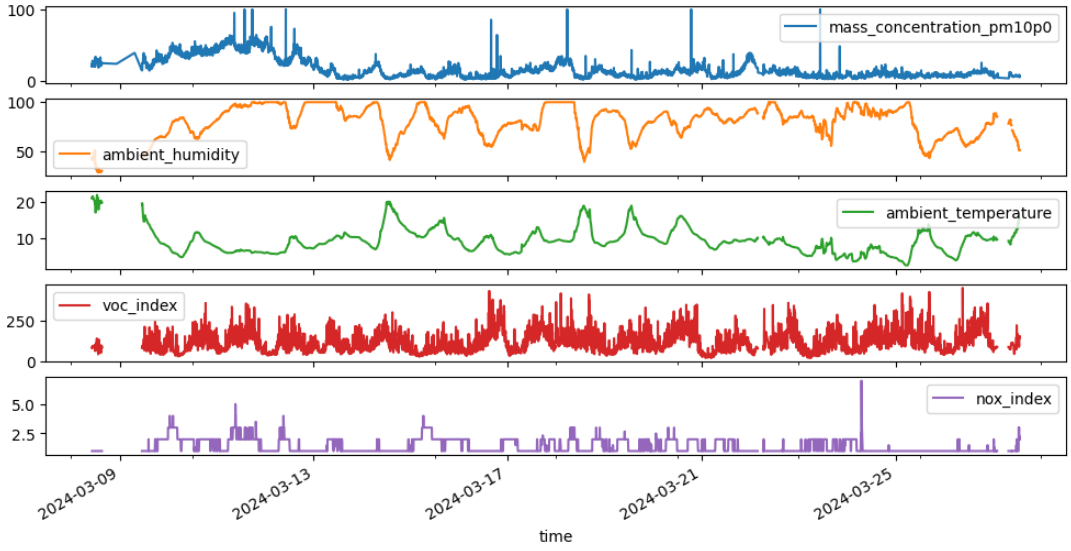

Fig. 16. Formatted sample for device #34 after applying uniform 2 minutes time window.

D COMPARISON OF TRANSFORMATIONS APPLIED TO VOC INDEX AND PM10 DATA.

The source code for visualizations can be found on the scikit-learn documentation website, at: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html

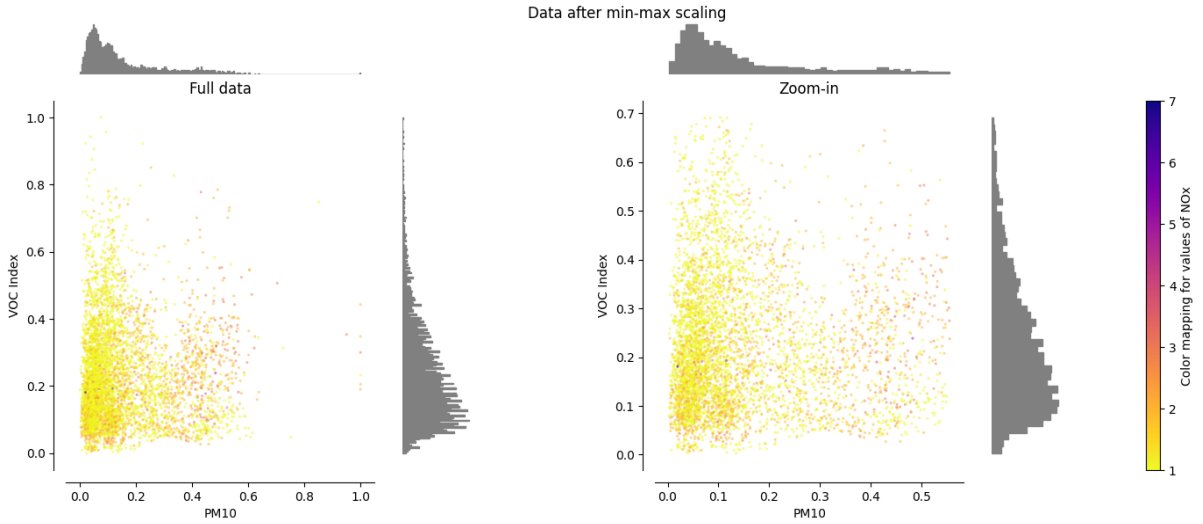

Fig. 17. MinMax Scaler. Full data and a 0 - 99 percentile range zoom-in.

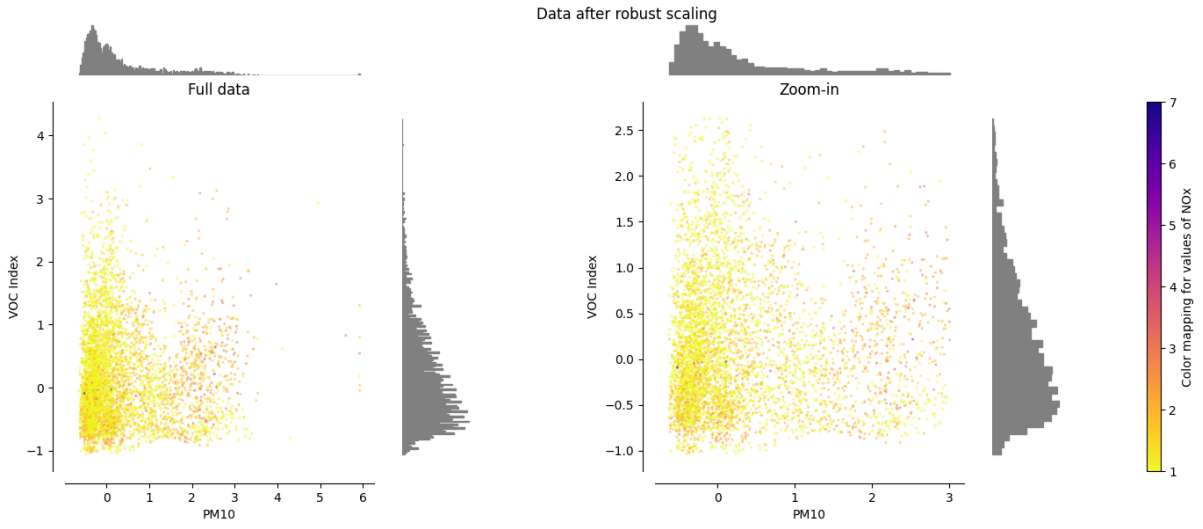

Fig. 18. Robust Scaler. Full data and a 0 - 99 percentile range zoom-in.

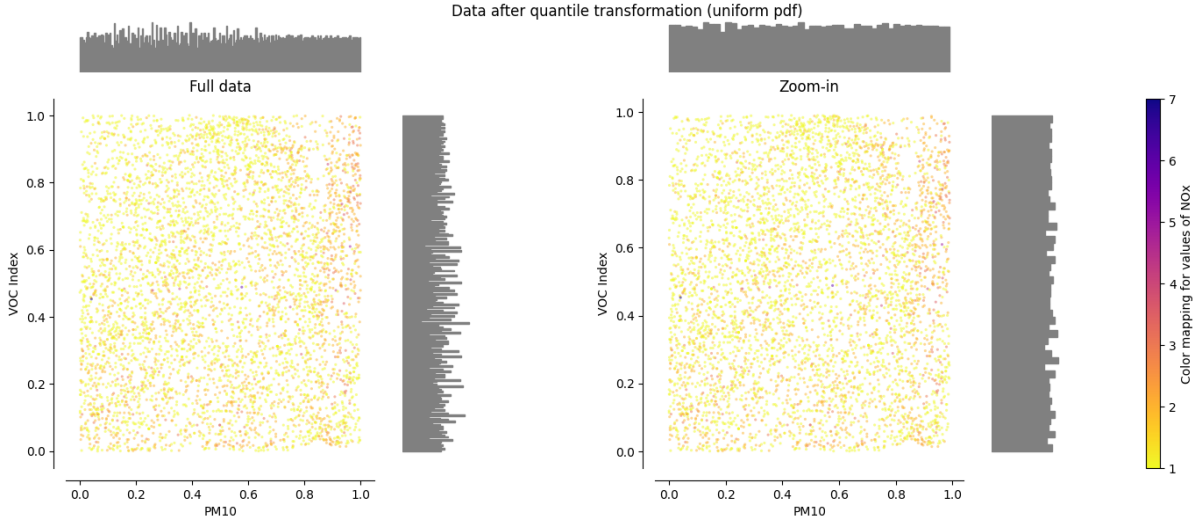

Fig. 19. Quantile Transformer. Full data and a 0 - 99 percentile range zoom-in.

E EXAMPLE OF OVER-FITTING FOR MODEL TRAINED ON FIVE-MINUTE DATASET WITH QUANTILE TRANSOFRM SCALING

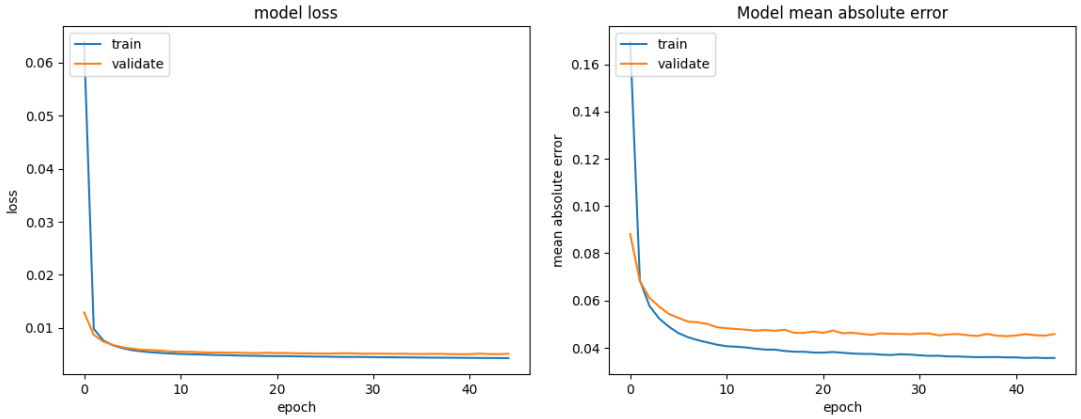

Fig. 20. Loss and mean absolute error (mae) for five-minute dataset, with Quantile Transform without shuffle.

F BASELINE VERSUS PREDICTION RESULTS FOR ADDITIONAL MODELS TRAINED

Metrics	PM1	PM2.5	PM4	PM10	Humidity	Temperature	VOC Index	NOx Index
Prediction Error	1.879993	2.091807	2.180341	2.385009	18.69542	0.67054	0.242827	0.104162
Baseline Error	2.059962	2.300725	2.513459	2.566677	18.017377	0.369182	0.058795	0.037368

Table 4. Mean absolute error on testing dataset, predicted vs. baseline. Data: two-minute time-series, with Quantile Transformation applied. Results are re-scaled.

Metrics	PM1	PM2.5	PM4	PM10	Humidity	Temperature	VOC Index	NOx Index
Prediction Error	1.761288	1.946745	1.945909	2.047001	20.049997	1.678788	0.438113	0.388244
Baseline Error	1.611014	1.879979	2.132054	2.270403	18.062048	0.724121	0.160803	0.162512

Table 5. Mean absolute error on testing dataset, predicted vs. baseline. Data: five-minute time-series, with MinMax scaling applied. Results are re-scaled.

Metrics	PM1	PM2.5	PM4	PM10	Humidity	Temperature	VOC Index	NOx Index
Prediction Error	1.953174	2.200589	2.359874	2.503794	19.003557	0.931512	0.349226	0.147992
Baseline Error	2.004278	2.212769	2.375451	2.464218	19.740372	0.552476	0.117314	0.064062

Table 6. Mean absolute error on testing dataset, predicted vs. baseline. Data: five-minute time-series, with Quantile Transformation applied. Results are re-scaled.

Received 29 March 2024; revised 1 April 2024