A2: Predictions of daily humidity in Delhi using multivariate climate data

Lasse Johansson, Joona Kareinen, Alex Karonen

December 20, 2024

1 Introduction

1.1 Project Overview

The objective of the project is to explore the forecasting power of neural network models trained to predict humidity in Delhi using multivariate climate data. The models that will be employed include Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Transformer Neural Networks (TNN). The models predictive performance will be compared against a chosen baseline model.

1.2 Code Availability

The source code developed for this project is available on Github[1].

1.3 Report Structure

The report is structured as follows: Section 2 introduces the dataset, covers exploratory analysis with time-series decomposition, and outlines the data preprocessing steps. Section 3 describes the architectures of the proposed models, the evaluation metrics, the optimization procedure, and the tools and resources used. Section 4 presents the experimental results, and provides a comparison of the models. Section 5 contains the ablation study for each model. Lastly, Section 6 discusses the results, summarizes the project, and proposes a potential future work.

2 Dataset and Preparation

2.1 Dataset Description

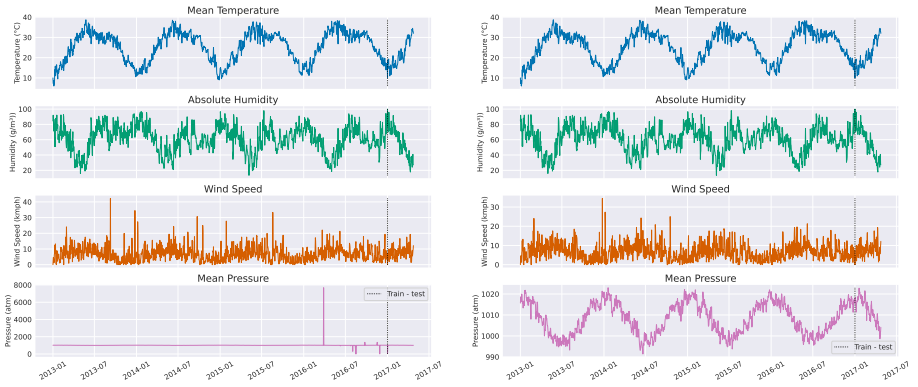
The dataset used in the project is originally from Weather Underground -service and it describes daily average meteorological conditions in Delhi, India, from

[1]Github link: `https://github.com/Jookare/ADAML-project2`

1st of January 2013 to 24th of April 2017. The dataset contains four parameters: mean temperature [°C], absolute humidity [g/m^3], wind speed [km/h], and mean pressure [atm]. The data is measured at a daily frequency, with the mean temperature obtained by averaging multiple 3 hour intervals, while other parameters are averaged across the entire day.

The dataset is synchronous across all variables and does not contain any missing values. However, the dataset contains few significant outliers in the pressure variable as the raw data ranged between -3 and 7700 (atm). These outliers were already removed for the exploratory analysis as otherwise the analysis does not bring any information about the mean pressure. The outlier removal process in explained in detail in Section 2.4. The raw and the cleaned time series are shown in Figure 1.

The dataset has 1575 days worth of data in total, with the data after 1st of January 2017 allocated for testing (total of 114 days). As visible in Figure 1, all variables exhibit clear seasonal trends.



(a) Raw data

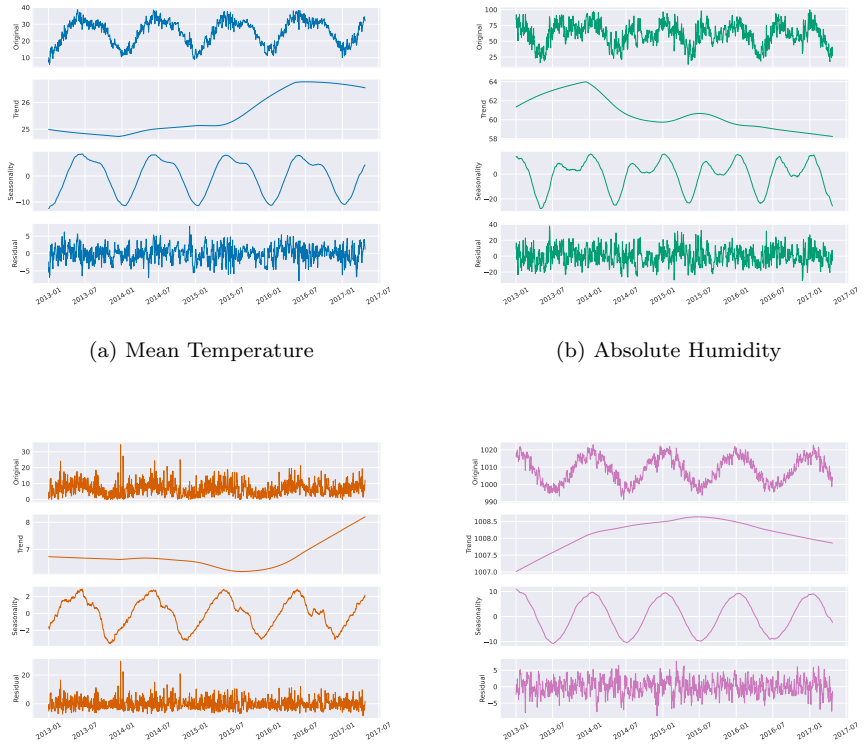
(b) Cleaned data

Figure 1: Time series plot for all variables in the dataset

2.2 Exploratory Analysis

To better understand the patterns in the data, the dataset was decomposed into three components: trend, seasonality, and residuals as shown in Figure 2. The decomposition was performed using 365-day seasonal cycle, with the seasonality smoothed using a 60-day moving average.

Multi-seasonal decomposition was also explored and tested, but we did not identify any additional meaningful seasonalities and we opted to use the single-seasonal decompose. In the figure the residuals capture the variance not included in the trend or in the smoothed seasonality. The 365 day seasonality captures the primary pattern in the data. The residuals can also capture the possible outliers such as sensor anomalies. However, in this case, the residuals appear to capture only normal fluctuations or noise inherent in the data.



(a) Mean Temperature

(b) Absolute Humidity

(c) Wind speed

(d) Mean pressure

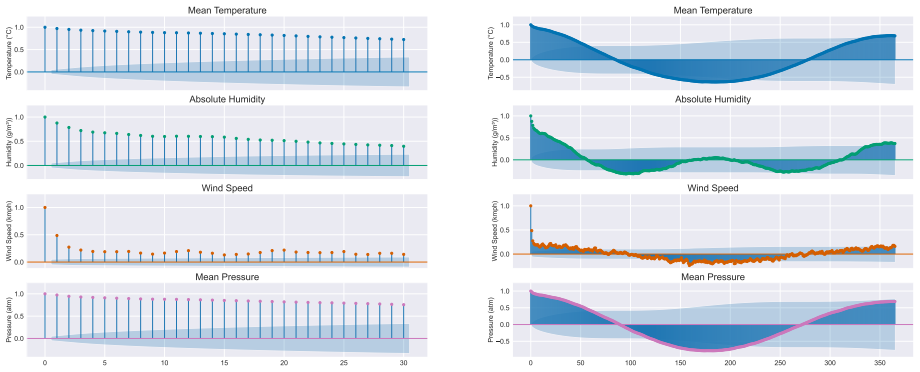
Figure 2: Time series decomposition with 365 day period

The autocorrelation plots for the dataset are shown in Figure 3. The short-term autocorrelation (30 days) shows how strongly the current values of the variables are related to the immediate past values. From the plot we can see that wind speed is the only variable where the correlation drops quickly. In the long term autocorrelation (365 days), the periodic nature of the variables is again clearly visible.

2.3 Data Partitioning

There exists a handful of ways to appropriately partition a time series data for model training purposes. Most commonly used partition techniques are rolling window, expanding window, blocking and leave-out-splitting methods.[2] Additionally, other non-traditional techniques include methods such as Markov cross-validation.[1] In this project, we aim to perform rolling cross-validation (CV) for model evaluation.

In rolling CV, a fixed length subset of data is used for training in each



(a) short-term (30 days) (b) long-term (365 days)

Figure 3: Autocorrelation plots for the training data

iteration, while a separated fixed length subset is used as a validation split. After each iteration, the training subset shifts forward to include the next portion of the data, and the validation subset moves forward as well. This means that the model is trained equally with each parts of the training data and validated with unseen data. Visualization of rolling CV is shown in Figure 4.

In defining the training and validation subset sizes the seasonality should be taken into an account in a way to ensure that the full trend is in each training subset. This can help the model better capture long-term trends and seasonal variations. Additional thing to note is that after rolling CV, the last validation period is also used to train the model. Following this training the model is tested with the testing dataset.

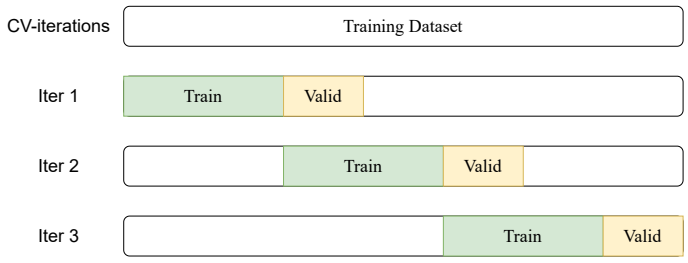


Figure 4: Rolling cross-validation

With models like Long Short-Term Memory (LSTM), a strong trend or seasonality can override the subtle signals of the data. Therefore, it can be beneficial to remove seasonality and trend from the data before feeding it to a model.[3] Common data scaling strategies for LSTMs include Min-Max scaling (scaling the input data between 0 and 1), or Z-score normalization (standardizing the data to have zero mean and unit variance).[4]

2.4 Data Preprocessing

Upon initial inspection of the raw data, few outliers were detected. Most notably the mean pressure contained some extreme values as the raw data ranged between -3 and 7700 (atm). These outliers were identified and removed using Z-scores, which measures how far each data point deviates from the mean in terms of standard deviations. The mean and standard deviation were computed for the first two years of the dataset, as this period showed no obvious outliers. A rolling window with size of 30 was used for the computation, and enough of these periods were concatenated to get values for the entire dataset (train and test). Z-scores were then computed for the whole period using the rolling mean and standard deviation values. If the absolute Z-score exceeded 4, the data point was considered an outlier and was replaced with the rolling mean value. This method of outlier replacement doesn't come without faults as it might detect some actual measurements such as high wind speeds caused by stormy weather as anomalous.

To help out models capture the seasonal nature of our data we plan to transform the day-of-year (d) into two supplementary features using sine and cosine transformations:

$$\sin_d = \sin\left(\frac{2\pi d}{365}\right), \qquad \cos_d = \cos\left(\frac{2\pi d}{365}\right). \tag{1}$$

This transformation projects the cyclical day-of-year onto a continuous, two dimensional circular space. These features capture the periodical nature of time without introducing artificial discontinuities (December 31st (365) to January 1st (1)).

To ensure that all data is evenly scaled, all numerical features are standardized using z-score normalization method:

$$z = \frac{x - \mu}{\sigma}, \tag{2}$$

where x is the data, μ is the mean of the feature across the training data, and σ is the standard deviation. This normalization scales the data to have zero mean and unit variance, reducing the risk of features with larger magnitudes (pressure) having too large influence to the models. Standardization is applied to the whole training set and the same parameters (μ and σ) are used to transform the testing set to ensure consistency.

3 Methods and Models

3.1 Baseline Method

A simple baseline model was created to predict the daily average humidity for the testing set. Given the strong seasonal behavior in meteorological data, common approached include auto-regressive model (AR) with defined context window or more advanced models like seasonal auto-regressive integrated moving average

(SARIMA) model. However, it was observed that the residuals seem to be normally distributed and exhibit the characteristics of white noise and thus the simplest (naive) baseline prediction may be the most effective - the last known value of humidity. The naive approach is especially effective for shorter forecasting period that do not exceed several days, as the seasonal fluctuations are relatively small.

3.2 Neural Network Models

The baseline architectures for the Recurrent Neural Network (RNN), the Long Short-Term Memory network (LSTM), and the Transformer Neural Network (TNN) can be seen in Figures 5, 6 and 7. The outline of the architectures will stay as shown, however the sizes of the layers i.e. how many units there are in each of the layers may vary. With transformer model it might be sufficient to use only the decoder for the predictions due to the limited size of the dataset. The final model compositions are going to be determined when we get to the model training and the ablation study will be created based on that. The initial results are obtained with models that have parameters shown in Tables 1 and 2.

Param	RNN	LSTM
Input size	(20, 6)	(20, 6)
Hidden size	128	128
Embedding size	128	128
Number of hidden layers	2	2
Dropout rate	20%	20%

Table 1: Initial parameters of Recurrent Neural Network models

Param	TNN	Decoder
Input size	(20, 6)	(20, 6)
Hidden size	128	128
Embedding size	128	128
Encoder depth	2	-
Encoder heads	4	-
Decoder depth	2	2
Decoder heads	4	4
Dropout rate	20%	20%

Table 2: Initial parameters of Transformer Neural Network models

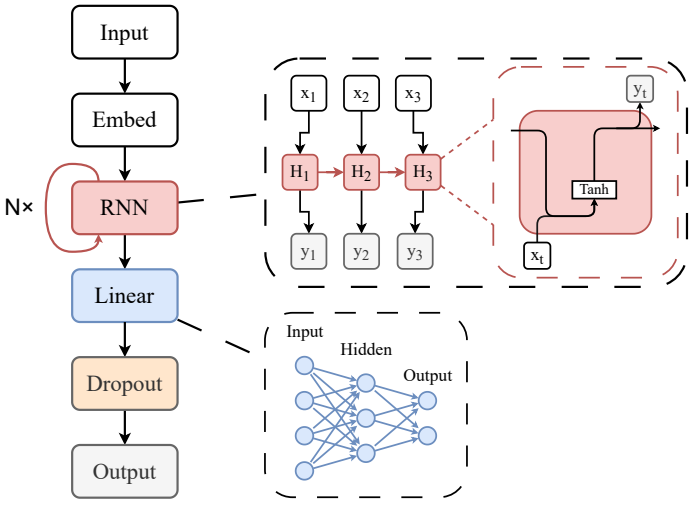


Figure 5: Baseline RNN model architecture

3.3 Model Evaluation and Optimization

The evaluation strategy for all models involves testing their prediction capabilities across multiple forecasting horizons: specifically, 1, 2, 5, and 10 days ahead. For each model, the forecasting performance is evaluated based on Root Mean Squared Error (RMSE) and Symmetric Mean Absolute Percentage Error (SMAPE). RMSE measures the magnitude of prediction errors and penalizes larger errors more heavily. It is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \tag{3}$$

where y_i is the a daily true humidity value and $\hat{y}_i$ is the model prediction. SMAPE provides a measure of prediction accuracy, and it is defined by:

$$\text{SMAPE} = \frac{2}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)}, \tag{4}$$

where y_i is the a daily true humidity value and $\hat{y}_i$ is the model prediction. The usage of metrics like RMSE and SMAPE allows the fair comparison between baseline model and more advanced architectures, such as RNN, LSTM, and Transformers, as long as the training, validation, and testing is kept same between the models.

For optimization, we employ the CV procedure to find optimal states for the model hyperparameters. Additionally, we will perform an ablation study to evaluate how each component of the model affects the performance. Furthermore, we will testing how the sizes of different layers affect the performance and

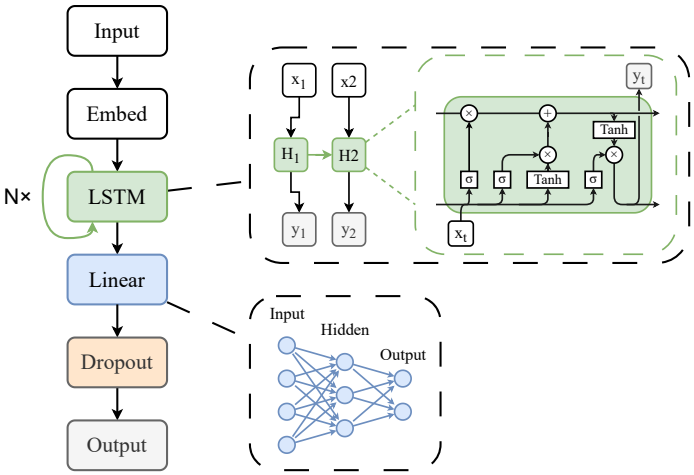


Figure 6: Baseline LSTM model architecture

we plan to study if using our proposed day-of-year preprocessing will improve the models' performance.

3.4 Tools and Resources

The models are created using Python and PyTorch, which is a machine learning library for Python. PyTorch was chosen as it is familiar to most of the team members and such using it makes the work a bit easier. In the model training we plan on using our own GPUs that we have on our desktop PCs. In case additional computational resources are needed, there are options like Google Colab and other computational servers the team members have access to.

We plan to divide the work as evenly as possible, taking into account the possible limitations in the available computational resources. In ideal case we divide the work so that each team member gets to implement at least one of the methods.

4 Experiments and Results

4.1 Baseline Model Results

The baseline model was evaluated on the testing set consisting of 114 daily values. The results of the naive approach (predicting the next day's humidity based on the last known value) are shown in Table 3. The horizon tells how many days ahead the forecasting was done.

As expected, the RMSE increases when the forecasting period gets longer, but it remains fairly small, suggesting that more sophisticated models may struggle to obtain significantly better performance. For reference, using the

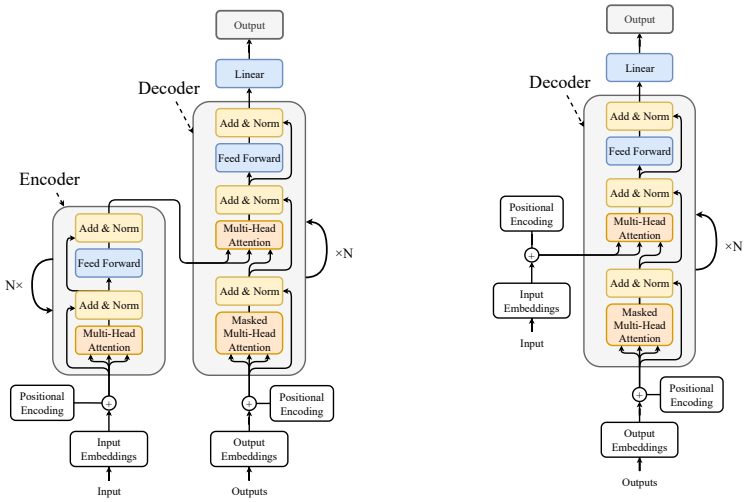


Figure 7: Proposed baseline transformer architectures: a) TNN, b) Decoder

Horizon	RMSE	SMAPE
1	7.42	0.114
2	9.13	0.145
5	10.42	0.161
10	11.26	0.189

Table 3: Results of the baseline model in terms of RMSE and SMAPE.

mean value (60.7) as a prediction for the whole testing set results in RMSE value of 19.5 for 1-day forecasting. Finally, one simple approach to make baseline predictions is to rely on the trends and seasonality of the training set. By making predictions this way for the testing partition one obtains an RMSE value of 13.7 for 1-day forecasting. In this assessment we use the last trend value of the training set and average seasonal value indexed according to the day-of-year.

4.2 Results for RNN and LSTM models

Initial training and results have been obtained for a RNN with an input window length of 20 days, linear embedding layer with 128 nodes, 2 RNN layers with 128 hidden recurrent nodes in each, and a dropout layer with a 20% dropout rate. The dropout is applied both between the RNN layers and before the final linear layer. An identical configuration was used for the training of the

LSTM model. For both models the training was done multiple times targeting a different forecasting window into the future each time (e.g., 1 day or 5 days ahead).

The training of the models was done over 100 epochs using Adaptive Moment Estimation (adam) optimizer with a learning rate of 10^{-4} and weight decay set to 0.03 to prevent overfitting. The loss function utilized in the training was the Mean Squared Error (MSE). The validation set contained the values for first 6 months of the next year. The training and validation loss metrics of the initial models are shown in Figure 8. From this we can see that the data of the second fold seems to be somewhat different from the other folds due to the model's tendency to overfit to the training data on that iteration. The results with trained RNN and LSTM models are shown in Table 4 together with the previously achieved baseline results for comparison. Figure 9 shows the 1 day and 10 day predictions with RNN and LSTM models.

Horizon	**Baseline**	**RNN**	**LSTM**
1	7.42	7.10	7.78
2	9.13	8.90	8.52
5	10.42	12.52	9.95
10	11.26	13.93	12.09

(a) RMSE

Horizon	**Baseline**	**RNN**	**LSTM**
1	0.114	0.110	0.124
2	0.145	0.149	0.136
5	0.161	0.200	0.162
10	0.189	0.214	0.194

(b) SMAPE

Table 4: RMSE and SMAPE values for different forecasting horizons (days) for the baseline, RNN and LSTM models (lower is better)

4.3 Results for Transformer and Decoder models

Using the same structure and evaluation basis that was used for RNN and LSTM, the forecasting accuracies of the transformer and decoder models were studied. The training was done using the exact same methods as was described in the previous section. Results in terms of RMSE and SMAPE can be seen in the Table 5. Training- and forecasting plots are also visualized and are visible in Figures 10 and 11.

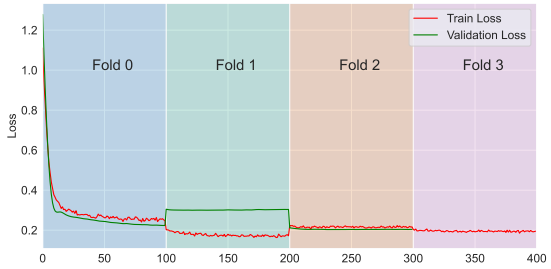
Horizon	**Baseline**	**TNN**	**Decoder**
1	7.42	6.90	7.04
2	9.13	7.49	7.28
5	10.42	9.96	9.39
10	11.26	10.64	10.01

(a) RMSE

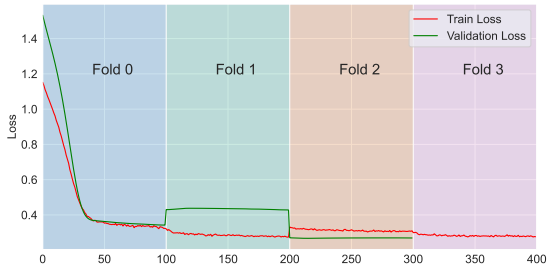
Horizon	**Baseline**	**TNN**	**Decoder**
1	0.114	0.110	0.114
2	0.145	0.120	0.114
5	0.161	0.183	0.150
10	0.189	0.209	0.168

(b) SMAPE

Table 5: RMSE and SMAPE values for different forecasting horizons (days) for the baseline, TNN, and Decoder models (lower is better)

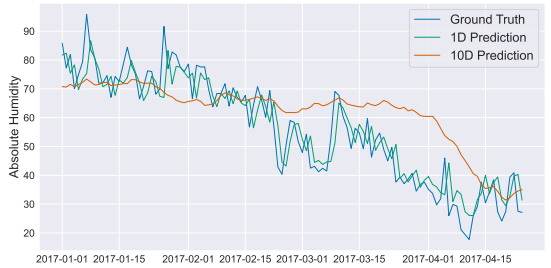


(a) RNN

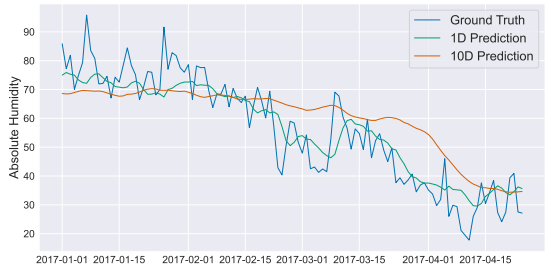


(b) LSTM

Figure 8: Training plots of RNN models



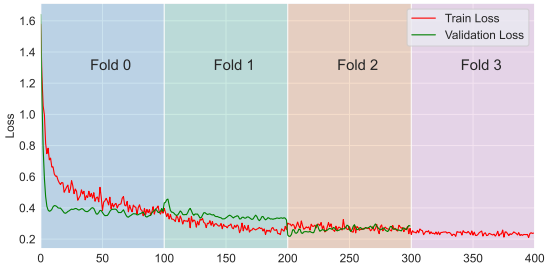
(a) RNN



(b) LSTM

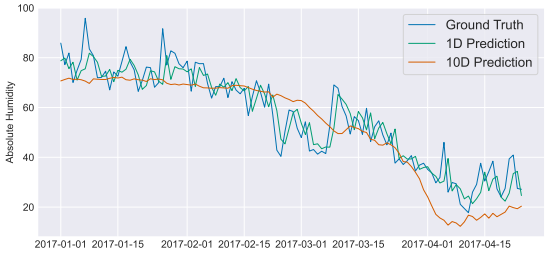
Figure 9: 1 day and 10 day predictions for the RNN models

(a) TNN

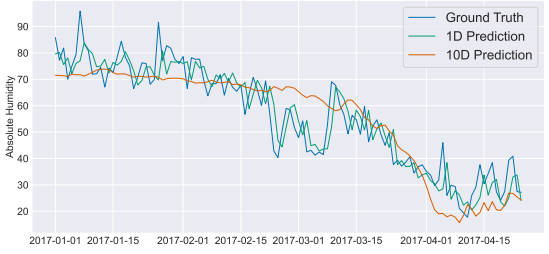


(b) Decoder

Figure 10: Training plots of initial models



(a) TNN



(b) Decoder

Figure 11: 1 day and 10 day predictions for the Transformer models

5 Ablation study

Ablation study was conducted for all the different model architectures by variating a) embedding dimension sizes, b) hidden dimension sizes, c) the number of layers and heads, d) window sizes, e) dropout rates, and g) the inclusion of day-of-year as an input variable. The results of the ablation study for different models have been shown in the Tables 6, 7, 8, and 9, for RNN, LSTM, TNN, and Decoder respectively. In the study the forecasting horizon was set to 1-day.

dim	RMSE	SMAPE
8	7.01	0.111
32	7.14	0.111
64	7.11	0.110
128	7.10	0.110
256	7.10	0.110

(a) Embedding dim.

dim	RMSE	SMAPE
8	8.09	0.135
32	7.04	0.112
64	7.07	0.111
128	7.10	0.110
256	7.23	0.112

(b) Hidden dim.

n	RMSE	SMAPE
1	7.06	0.109
2	7.10	0.110
4	7.07	0.110
8	7.09	0.112

(c) Layers

size	RMSE	SMAPE
5	7.13	0.111
10	7.08	0.110
20	7.10	0.110
30	7.07	0.109

(d) Window size

p	RMSE	SMAPE
0.2	7.10	0.110
0.4	7.14	0.111
0.6	7.09	0.110
0.8	6.97	0.109

(e) Dropout

day-of-year	RMSE	SMAPE
included	7.10	0.110
excluded	7.12	0.111

(f) Date as variable

Table 6: RNN ablation study

dim	RMSE	SMAPE
8	8.18	0.133
32	7.87	0.126
64	7.83	0.125
128	7.78	0.124
256	7.65	0.120

(a) Embedding dim.

dim	RMSE	SMAPE
8	10.66	0.174
32	8.31	0.131
64	8.01	0.127
128	7.78	0.124
256	7.66	0.121

(b) Hidden dim.

n	RMSE	SMAPE
1	7.29	0.117
2	7.78	0.124
4	9.21	0.150
8	19.14	0.312

(c) Layers

size	RMSE	SMAPE
5	7.70	0.123
10	7.62	0.121
20	7.78	0.124
30	7.88	0.125

(d) Window size

p	RMSE	SMAPE
0.2	7.78	0.124
0.4	7.79	0.124
0.6	7.85	0.125
0.8	8.05	0.130

(e) Dropout

day-of-year	RMSE	SMAPE
included	7.78	0.124
excluded	7.81	0.125

(f) Date as variable

Table 7: LSTM ablation study

dim	RMSE	SMAPE
8	9.67	0.152
32	7.90	0.116
64	6.96	0.107
128	6.90	0.110
256	7.08	0.111

(a) Embedding dim.

dim	RMSE	SMAPE
8	7.08	0.114
32	6.79	0.107
64	6.82	0.108
128	6.90	0.110
256	6.87	0.108

(b) Hidden dim.

n	RMSE	SMAPE
1	6.97	0.110
2	6.90	0.110
4	7.03	0.110
8	6.92	0.109

(c) Encoder depth

n	RMSE	SMAPE
1	7.00	0.113
2	7.01	0.114
4	6.90	0.110
8	6.94	0.110

(d) Encoder heads

n	RMSE	SMAPE
1	6.93	0.110
2	6.90	0.110
4	7.16	0.112
8	7.26	0.114

(e) Decoder depth

n	RMSE	SMAPE
1	6.95	0.112
2	6.90	0.110
4	6.90	0.110
8	6.92	0.110

(f) Decoder heads

size	RMSE	SMAPE
5	7.29	0.119
10	7.04	0.115
20	6.90	0.110
30	6.95	0.112

(g) Window size

p	RMSE	SMAPE
0.2	6.90	0.110
0.4	7.03	0.111
0.6	8.56	0.146
0.8	13.26	0.221

(h) Dropout

day-of-year	RMSE	SMAPE
included	6.90	0.110
excluded	6.90	0.110

(i) Date as variable

Table 8: TNN ablation study

dim	RMSE	SMAPE
8	11.20	0.184
32	7.71	0.119
64	6.79	0.108
128	7.04	0.114
256	6.96	0.112

(a) Embedding dim.

dim	RMSE	SMAPE
8	6.93	0.111
32	6.89	0.108
64	7.03	0.111
128	7.04	0.114
256	7.09	0.112

(b) Hidden dim.

n	RMSE	SMAPE
1	6.97	0.111
2	7.04	0.114
4	7.02	0.110
8	7.03	0.113

(c) Decoder depth

n	RMSE	SMAPE
1	6.97	0.112
2	6.89	0.110
4	7.04	0.114
8	6.98	0.113

(d) Decoder heads

size	RMSE	SMAPE
5	7.29	0.119
10	7.24	0.120
20	7.04	0.114
30	7.07	0.113

(e) Window size

p	RMSE	SMAPE
0.2	7.04	0.114
0.4	7.43	0.123
0.6	8.24	0.140
0.8	8.47	0.125

(f) Dropout

day-of-year	RMSE	SMAPE
included	7.04	0.114
excluded	7.01	0.111

(g) Date as variable

Table 9: Decoder ablation study

6 Discussion and conclusions

Humidity forecasts for different forecasting horizons were modelled using RNN, LSTM and Transformer model architectures. Depending on the model and choice of parameters the models provide either slightly better and sometimes slightly worse forecasting accuracy than the simple baseline model, depending on the forecasting horizon.

For the prediction of next day's humidity, Transformer and RNN models achieve the best accuracy in terms of RMSE and SMAPE indicators. For the prediction of 2-day or 5-day humidity, the LSTM model is decent, but transformer architecture still provides better prediction accuracy. With forecasting horizon of 10 days the best results are given by the decoder-only Transformer model. The accuracy is still only slightly better than the simple baseline model, showing that humidity (and weather more generally) cannot be predicted accurately several days ahead.

In Figure 9 it can be observed that the models trained for 10-day forecasts are smoother and more generalized whereas the 1-day forecast models - especially the RNN model - aim to capture the finer variability of humidity. From the shape of the RNN 1-day forecast, it can also be observed that it closely resembles the humidity value of 1 day before and thus is a fairly similar model than the baseline. The same behavior can be observed with the transformer models in Figure 11.

Ablation study was conducted for all different model architectures by varying embedding dimension, hidden dimension, the number of layers and heads, window sizes, and dropout rates. The RNN was shown to be insensitive to parameter value changes and the RMSE is contained between 6.97 and 8.09 during the ablation study. LSTM, which is consistently being outperformed by RNN, is slightly more sensitive to parameter variations, and there is one parameter configuration in which the model breaks in terms of RMSE (19.14 by setting layer count to 8). The inclusion of date (day of year) as a variable was also tested, and the inclusion provides slight (almost negligible) benefits. However, the date as a variable is expected to be useful only in the context of much larger forecasting windows. The transformer models ablation studies reveal similar findings that the models are not too sensitive to parameter changes. The only difference with transformer model is that they are a lot more sensitive to the embedding dimension compared to RNN models. Also the dropout affects the transformers a lot more.

Ultimately, the ablation study reveals that there are only minor improvements to be had by fine tuning the model parameters. By utilizing the results obtained in the ablation study and the principle of least complexity, the "optimized" models can be good in terms of performance while using less compute.

6.1 Future work

Based on the findings in this report, we conclude that with the given data, there is no significant difference between forecasting capabilities of the intro-

duced models. Thus one might want to consider the computational complexity compared against the forecasting power of the models which could prove useful depending on the application. The forecasting approach of the neural network models could be rethought such that the models would forecast a multivariate output i.e. give a prediction for each of the features simultaneously. This way the forecasting could be done in auto-regressive manner such that the model outputs would be the inputs for the next forecast. Such models could be beneficial in scenarios where acquiring new data is expensive.

References

[1] Jiang, Gaoxia, and Wang, Wenjian. "Markov cross-validation for time series model evaluations." Information Sciences, vol. 375, 2017. https://doi.org/10.1016/j.ins.2016.09.061.

[2] Bergmeir, Christoph, Hyndman, Rob J., and Koo, Bonsoo. "A note on the validity of cross-validation for evaluating autoregressive time series prediction." Computational Statistics & Data Analysis, vol. 120, 2018, pp. 70-83. https://doi.org/10.1016/j.csda.2017.11.003.

[3] Wu, Yuhan, Meng, Xiyu, Zhang, Junru, He, Yang, Romo, Joseph A., Dong, Yabo, and Lu, Dongming. "Effective LSTMs with seasonal-trend decomposition and adaptive learning and niching-based backtracking search algorithm for time series forecasting." Expert Systems with Applications, vol. 236, 2024. https://doi.org/10.1016/j.eswa.2023.121202.

[4] Karevan, Zahra and Suykens, Johan A. K. "Transductive LSTM for time-series prediction: An application to weather forecasting." Neural networks, vol. 125, 2020, pp. 1–9. https://doi.org/10.1016/j.neunet.2019.12.030.