ECSE 552 Deep Learning
Assignment 4

Gauri Sharma
261026894

Group No
20

Contents

1 Introduction **2**

1.1 Dataset . . . 2

2 Method **2**

2.1 Features . . . 2

2.2 Preprocessing steps . . . 3

2.3 Feature Selection and EDA . . . 4

2.4 Specific model architecture . . . 5

2.4.1 Baseline model consisting of only fully-connected (nn.Linear) layers . . . 5

2.4.2 Baseline model that uses at least one layer of LSTM . . . 6

2.4.3 Proposed Model . . . 6

2.5 Loss function . . . 6

2.6 Our model (advantages, special properties, difference from the other two, etc., hyper-parameters and how you selected them . . . 6

2.7 Training procedure . . . 7

3 Results **7**

3.1 Metrics used for comparison . . . 7

3.1.1 Mean Square Error (MSE) . . . 7

3.1.2 Mean Absolute Percentage Error (MAPE) . . . 7

3.1.3 Weighted Mean Absolute Percentage Error (WMAPE) . . . 8

3.2 Performance Comparison . . . 8

3.2.1 Overall Performance Comparison . . . 8

3.2.2 Model comparisons in terms of the 4 specified attributes . . . 10

4 Additional Analysis for your model **10**

4.1 Error propagation . . . 10

4.2 Other Analysis . . . 11

1 Introduction

The goal of this assignment is to perform time-series forecasting using deep learning methods for weather related data. The values for four distinct variables at time *t* need to be predicted given a history of weather attributes from time *t - k* to time *t - 1*. Three different models are supposed to be investigated for the assigned task. The idea is to predict the current and future time steps on the basis of previous time steps for specific values. The values that are to be predicted for time *t* and the models that need to be employed can be found in table 1:

Variables	Methods
p(mbar) - Atmospheric Pressure T(degC) - Air Temperature rh (%) - Relative Humidity wv(m/s) - Wind Velocity	Only fully-connected (nn.Linear) layers At least one layer of LSTM Own Model

Table 1: List of values to be determined and methods to be investigated

1.1 Dataset

Source: Max Planck Institute for Biogeochemistry
Duration: 2009 to 2016
Curator: Francois Chollet
Description: The meteorological time-series represented by this dataset were captured at the Jena, Germany-based Max Planck Institute for Biogeochemistry's meteorological station. Date-time and 14 climatic measurements/features, such as atmospheric pressure, temperature, and humidity, that were originally recorded every ten minutes during a nine-year period are included in the dataset. The data was scaled down for this assignment to match the requirements since just hourly predictions were required. There are 56072 time series data points (1 January 2009 to 24 May 2015) for training and 14019 time series data points (24 May 2015 to 31 December 2016) for testing. Test-to-train ratio is roughly (1:4).

Figure 1: Dataset

```
train_df.head() # Display the first 5 rows of the training data
```

	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
0	01.01.2009 01:00:00	996.50	-8.05	265.38	-8.78	94.4	3.33	3.14	0.19	1.96	3.15	1307.86	0.21	0.63	192.7
1	01.01.2009 02:00:00	996.62	-8.88	264.54	-9.77	93.2	3.12	2.90	0.21	1.81	2.91	1312.25	0.25	0.63	190.3
2	01.01.2009 03:00:00	996.84	-8.81	264.59	-9.66	93.5	3.13	2.93	0.20	1.83	2.94	1312.18	0.18	0.63	167.2
3	01.01.2009 04:00:00	996.99	-9.05	264.34	-10.02	92.6	3.07	2.85	0.23	1.78	2.85	1313.61	0.10	0.38	240.0
4	01.01.2009 05:00:00	997.46	-9.63	263.72	-10.65	92.2	2.94	2.71	0.23	1.69	2.71	1317.19	0.40	0.88	157.0

```
test_df.head() # Display the first 5 rows of the test data
```

	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
0	24.05.2015 19:00:00	990.00	17.93	291.93	9.23	56.73	20.58	11.68	8.91	7.37	11.79	1179.50	1.58	2.16	34.37
1	24.05.2015 20:00:00	989.86	16.92	290.93	9.52	61.65	19.31	11.90	7.40	7.51	12.03	1183.33	0.23	0.60	20.89
2	24.05.2015 21:00:00	990.11	15.82	289.80	9.71	66.97	18.00	12.06	5.95	7.61	12.18	1188.07	0.26	0.52	252.00
3	24.05.2015 22:00:00	990.01	15.02	289.01	9.50	69.53	17.10	11.89	5.21	7.50	12.01	1191.32	1.22	1.88	218.50
4	24.05.2015 23:00:00	989.89	14.14	288.14	9.42	73.20	16.15	11.83	4.33	7.46	11.95	1194.85	1.43	1.96	218.00

2 Method

2.1 Features

The meteorological data set consists of a collection of training samples, each of which is comprised of fourteen features related to various meteorological measurements. These features are as follows:

1. **Date Time** represents the date and time at which the meteorological data was recorded.
2. **Air Pressure (p (mbar))** represents the atmospheric pressure measured in millibars (mbar).
3. **Air Temperature (T (degC)** represents the temperature in degrees Celsius (degC).

Figure 2: Dataset Description

```
# describe the data
train_df.describe()
```

Python

	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
56072.000000	56072.000000	56072.000000	56072.000000	56072.000000	56072.000000	56072.000000	56072.000000	56072.000000	56072.000000	56072.000000	56072.000000	56072.000000	56072.000000	56072.000000
988.832928	8.987746	283.059875	4.551409	76.208224	13.170791	9.297106	3.873606	5.874655	9.404474	1217.704076	2.149853	3.558251	174.029077
8.350194	8.440986	8.518036	6.830767	16.498261	7.479040	4.150333	4.673326	2.634644	4.201705	40.129584	1.542047	2.337514	87.062724
913.600000	-22.760000	250.850000	-24.800000	13.880000	0.970000	0.810000	0.000000	0.510000	0.810000	1059.450000	0.000000	0.000000	0.000000
983.810000	2.880000	277.000000	-0.120000	65.497500	7.520000	6.050000	0.820000	3.820000	6.120000	1189.090000	1.010000	1.800000	123.400000
989.240000	8.970000	283.100000	4.745000	79.500000	11.470000	8.575000	2.090000	5.410000	8.680000	1214.850000	1.790000	3.000000	197.400000
994.310000	15.070000	289.130000	9.732500	89.600000	17.160000	12.080000	5.100000	7.630000	12.210000	1244.640000	2.890000	4.760000	233.900000
1013.910000	35.650000	309.730000	23.060000	100.000000	58.340000	28.250000	41.780000	18.070000	28.740000	1393.540000	14.010000	23.500000	360.000000

```
test_df.describe()
```

Python

	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
unt	14019.000000	14019.000000	14019.000000	14019.000000	14019.000000	14019.000000	14019.000000	14019.000000	14019.000000	14019.000000	14019.000000	14019.000000	14019.000000	14019.000000
ean	990.732392	11.301290	285.225804	6.576604	75.216103	15.199599	10.481350	4.718212	6.614139	10.584220	1209.490327	-0.086451	0.582371	177.828951
std	8.220120	8.092519	8.224984	6.045940	16.357764	8.514445	4.182481	5.661257	2.657874	4.235651	38.656354	146.298911	168.955876	84.758167
min	956.960000	-13.670000	259.150000	-15.300000	23.640000	2.120000	1.850000	0.010000	1.150000	1.840000	1103.790000	-9999.000000	-9999.000000	0.110000
5%	985.910000	5.080000	279.100000	2.030000	64.250000	8.775000	7.080000	1.090000	4.450000	7.130000	1181.300000	0.930000	1.680000	131.550000
0%	990.960000	11.070000	284.950000	6.800000	78.300000	13.200000	9.890000	2.600000	6.230000	9.980000	1209.160000	1.640000	2.800000	201.000000
5%	995.960000	17.040000	291.030000	11.350000	88.800000	19.460000	13.450000	6.070000	8.490000	13.570000	1235.705000	2.730000	4.600000	234.400000
nax	1015.290000	37.280000	311.210000	19.780000	99.900000	63.770000	23.100000	46.010000	14.760000	23.510000	1348.150000	12.050000	18.670000	359.900000

4. **Potential Temperature (Tpot (k))** represents the potential temperature in Kelvin (K).
5. **Dew Point Temperature (Tdew (degC))** is the value for Dew point temperature.
6. **Saturation Water Vapor Pressure (VPmax (mbar))** represents the maximum pressure of water vapor that can exist in the air at the given temperature and atmospheric pressure.
7. **Actual Water Vapor Pressure (VPact (mbar))** is the actual water vapor pressure value.
8. **Water Vapor Pressure Deficit (VPdef (mbar))** is the water vapor pressure deficit value.
9. **Specific Humidity (sh (g/kg))** represents the mass of water vapor per unit mass of air in grams per kilogram (g/kg).
10. **Water Vapor Concentration (H2OC (mmol/mol))** represents the concentration of water vapor in the air in millimoles per mole (mmol/mol).
11. **Air Density (rho (g/m^3)** is the value for Air density.
12. **Wind Velocity (wv (m/s))** is the value for Wind velocity.
13. **Maximum Wind Velocity (max. wv (m/s))** is the Maximum Wind velocity.
14. **Wind direction (wd (deg))** is the value for Wind direction.

2.2 Preprocessing steps

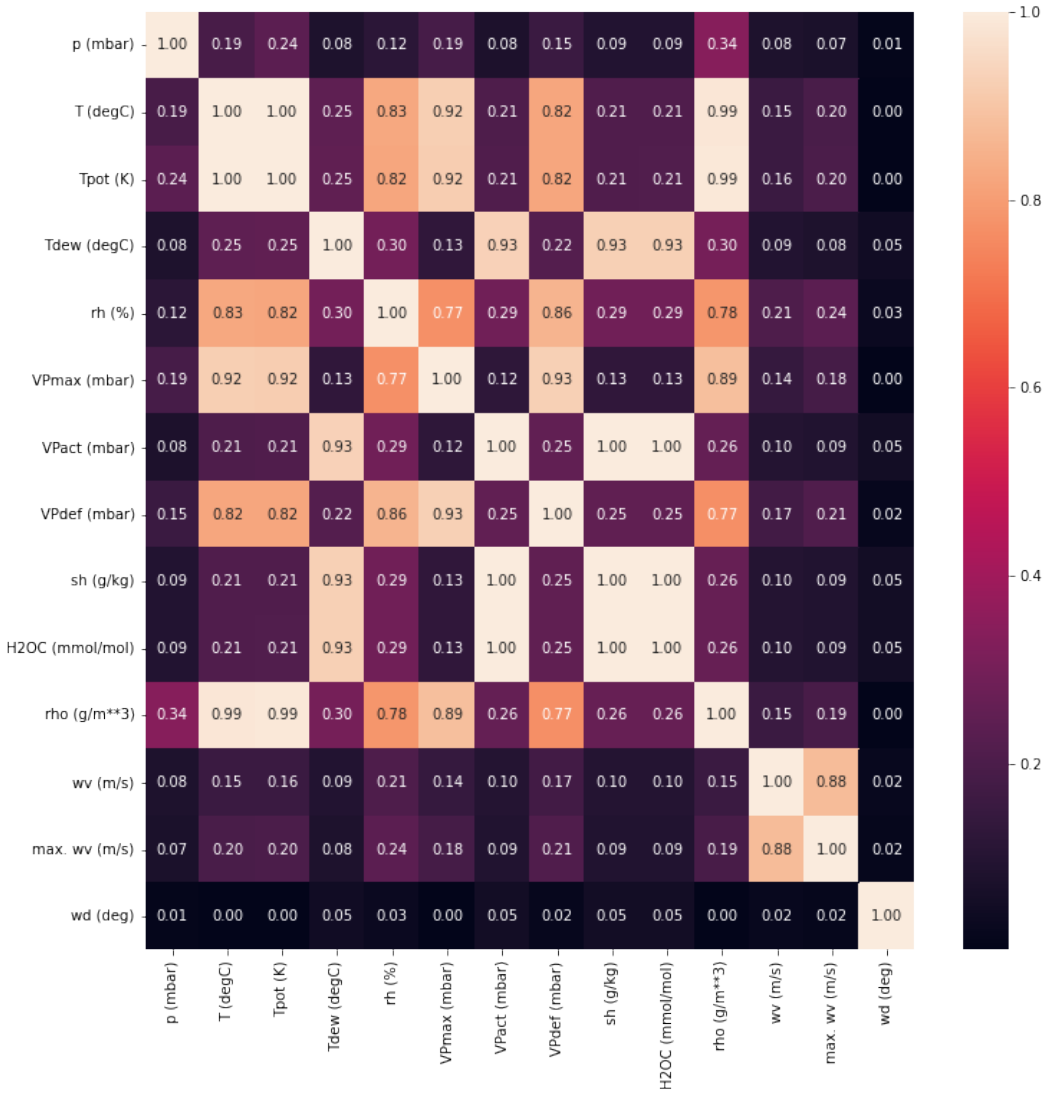
- The 'Date Time' column of the train and test dataframes is first converted into datetime format to work with the date and time values as independent columns, which is frequently helpful in time series analysis.
- "Date Time" column was deleted from train and test dataframes to isolate the date and time values from the other columns of the data so that we can model or analyse the data based on its other characteristics.
- The test data is cleaned to ensure that the data is suitable for analysis. Negative values, such as -9999, are replaced with more appropriate values. Specifically, negative wind velocity values in the 'wv (m/s)' column and negative max wind velocity values in the 'max. wv (m/s)' column are replaced with 0 in the test DataFrame. This step is important because negative values are not possible in real-life scenarios and could adversely affect the analysis results.
- The mean and standard deviation of the train dataframe are then determined to normalise the data using the mean and standard deviation. It makes it easier to compare to data from other attributes and scales.

- Using the estimated means and standard deviations, the code then normalises both the train and test dataframes. The standard deviation is divided by each value after the mean has been subtracted. This procedure aids in scaling the data and bringing it into a more uniform form.

2.3 Feature Selection and EDA

The pearson cross-correlation across the initial difference of all features is displayed in Figure 3. The Pearson correlation calculates the linear relationship between two continuous signals' covariance over time and displays it as a number between 0 and 1 (perfect correlation). To train the model, we selected a specific set of features from the 3, while discarding all other features that were not included in the set. The final set of features used for training includes air pressure p (mbar), air temperature T (degC), relative humidity rh (%), saturation water vapor pressure VPmax (mbar), water vapor pressure deficit VPdef (mbar), air density rho (g/m**3), wind velocity wv (m/s), and maximum wind velocity max. wv (m/s).

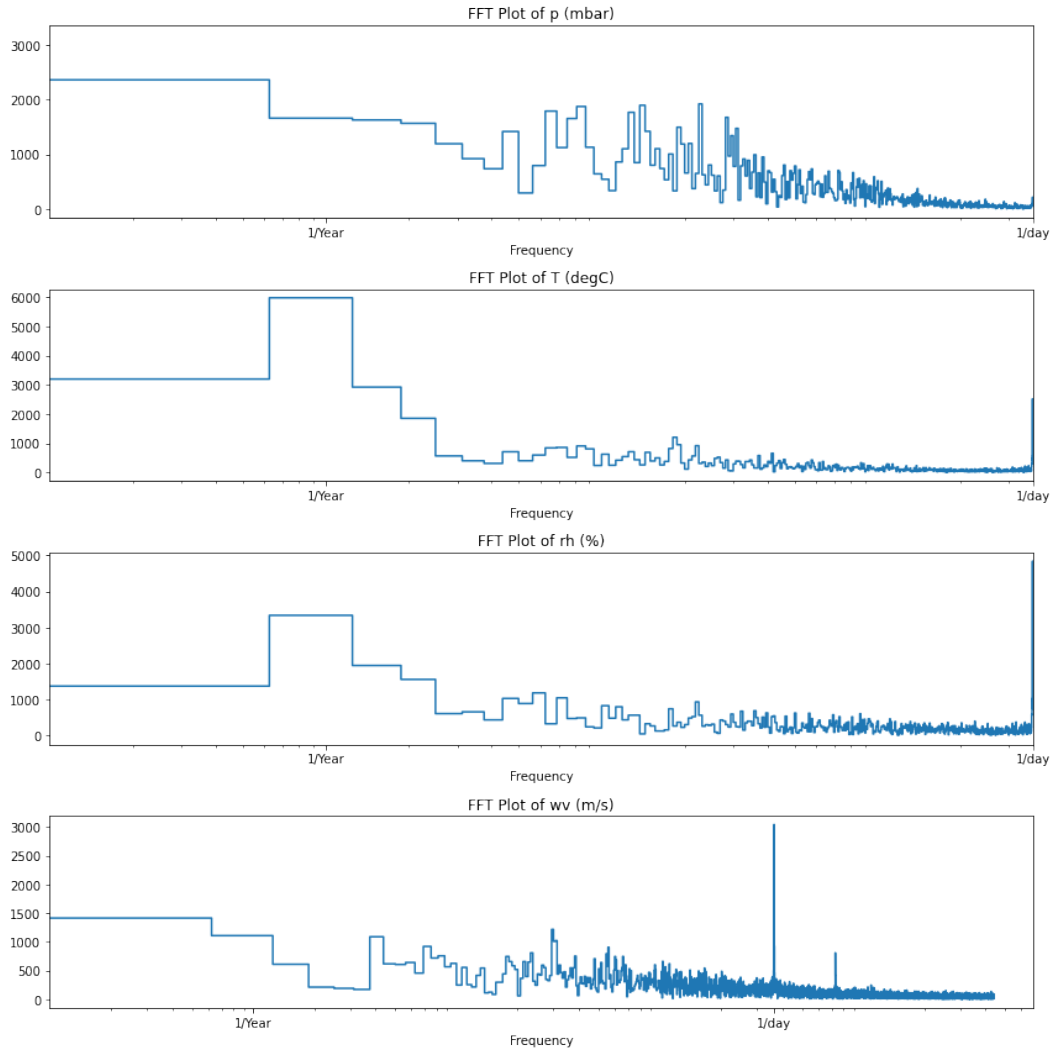
Figure 3: Heatmap



The FFT will allow us to see if the feature has any underlying periodic nature. The frequency of one year or one day corresponds to many characteristics. Features with a high one-day or twenty-four-hour

periodicity can be of interest to us. We display the fourier transform of the four forecasted weather qualities in Figure 4.

Figure 4: Discrete Fourier Transform of the Time Series Data



2.4 Specific model architecture

2.4.1 Baseline model consisting of only fully-connected (nn.Linear) layers

To define the model and its layers, the code makes use of PyTorch's nn.Module class. The nn.Module class is initially initialised as a superclass of the class pertaining to this model by the method super(Model1, self).init(). The completely connected layers are defined by the nn.Linear() method, and the first layer's output is activated by the ReLU using the F.relu() method. To convert the output tensor to a one-dimensional tensor, the mean of the output tensor along the second dimension is computed using the torch.mean() method. The model's forward pass, which applies the layers in order to the input tensor, is defined by the forward() method. The output tensor is finally reduced to a one-dimensional tensor by the x = torch.mean(x, dim=1) line, which computes the output tensor's mean along the second dimension. The code constructs a basic neural network model for converting input data into output data by utilising PyTorch's built-in functions and techniques.

2.4.2 Baseline model that uses at least one layer of LSTM

The provided code specifies the second model which forecasts the target variable from a set of input features using LSTM. An LSTM (Long Short-Term Memory) layer is the first layer of this model architecture, which is followed by a fully connected (linear) layer. SingleOutputLSTM, a unique class that extends PyTorch's nn.LSTM class, is used to define the LSTM layer. In order to return only the output from the LSTM layer and not the hidden state, the SingleOutputLSTM class overrides the forward method of the nn.LSTM class. By using the super().forward(x) method and returning the first output element, this is accomplished. The SingleOutputLSTM class is used by the respective model's class to initialise the LSTM layer, and nn.Linear is used to initialise the linear layer. The input tensor is first sent through the LSTM layer in the forward method of the MODEL class, and the final output of the LSTM layer is then applied with the linear layer. The output tensor is then delivered as an output.

2.4.3 Proposed Model

The proposed model accomplishes the task using a fully connected (linear) layer and a Gated Recurrent Unit (GRU) layer. The module's constructor defines the model's input, hidden, and output sizes. The fully connected layer is initialised with the hidden size as input and output, and the GRU layer is initialised with the input size and hidden size that are specified. The input tensor is passed through the GRU layer in the forward function to obtain the output tensor and the ultimate hidden state tensor. The output tensor is discarded, and only the final hidden state tensor is used in the procedure. The final output tensor is then obtained by passing the final hidden state tensor through the fully linked layer and returning it.

2.5 Loss function

Each model in the study is trained using the mean square error (MSE) or L2 Norm loss function. The MSE is defined as the average of the squared differences between the predicted values and the actual values in the dataset. The mathematical formula for MSE is as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{1}$$

where n is the number of samples in the dataset, y_i is the actual value of the i^{th} sample, and $\hat{y_i}$ is the predicted value of the i^{th} sample.

2.6 Our model (advantages, special properties, difference from the other two, etc., hyperparameters and how you selected them

1. Our proposed model has an advantage over the two baseline models, nn.Linear and LSTM model, as it uses a Gated Recurrent Unit (GRU) layer instead of a Long Short-Term Memory (LSTM) layer.
2. The GRU is a simpler version of LSTM and requires fewer parameters, which makes it computationally less expensive and faster to train. In comparison, the LSTM layer in second baseline model has more parameters and is more computationally expensive to train.
3. Additionally, the use of the GRU layer in our proposed model allows for the model to learn temporal dependencies in the input data, which is crucial for predicting weather patterns and climate changes. The GRU is also better suited for handling shorter sequences of data, which is appropriate for the meteorological dataset used in this study.
4. **Hyperparameters:** Using a range of learning rates, hidden unit counts, and batch sizes, we conducted a grid search to identify the model's hyperparameters. In order to reduce the mean square error (MSE) loss function, the hyperparameters' optimal values were picked based on the validation set's performance. A learning rate of 0.001, 64 hidden units, and a batch size of 64 were the final hyperparameters that were applied to our model.

Overall, our proposed model offers a simpler and more efficient alternative to the baseline models, while still achieving comparable or better prediction accuracy.

2.7 Training procedure

1. The training procedure for the three models involves preparing the data by pre-processing and selecting features, initializing the models with appropriate neural network architectures, selecting the mean square error (MSE) loss function and Adam optimizer, iterating over the training data in multiple epochs using small batches, evaluating the performance using the MSE metric, and selecting the best performing model based on the validation set performance.
2. Additionally, the training procedure also involves defining the training and testing functions for each model, which take in the pre-processed and selected features as input and compare them with the corresponding labels to calculate the loss. The loss is then backpropagated through the network to update the model parameters using the optimizer. The testing function evaluates the model's performance on a separate testing dataset and calculates the MSE loss between the predicted values and the true labels.

Model	Epochs	Batch Size	k	No. of Feature	No. of Label	Learning Rate
nn.Linear	20	32	4	8	4	1e-4
LSTM	20	32	4	8	4	1e-4
Our Model	20	32	4	8	4	1e-4

Table 2: Training setup for the three models

This table displays the training setup for the three models. Each row represents a model, and the columns represent the following training parameters:

- Epochs: The number of times the model will iterate through the entire training dataset.
- Batch Size: The number of samples to be processed in one iteration.
- k: The sequence length for the data.
- No. of Feature: The number of input features.
- No. of Label: The number of output labels.
- Learning Rate: The initial learning rate used for the optimizer.

The three models all have the same training setup, with 20 epochs, a batch size of 32, a sequence length of 4, 8 feature columns, 4 label columns, and an initial learning rate of 1e-4. The specific models being trained are nn.Linear, LSTM, and Our Model.

3 Results

3.1 Metrics used for comparison

3.1.1 Mean Square Error (MSE)

MSE is a common metric used to measure the average squared difference between the predicted and actual values in a regression problem. It is calculated by taking the average of the squared differences between the predicted and actual values for each data point. It is given as:

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

where n is the number of samples, y_i is the true value of the i^{th} sample, and $\hat{y}_i$ is the predicted value of the i^{th} sample.
The four weather attributes and the four overall attributes' mean squared error (MSE) is measured. We can observe that all three models' squared error graphs are comparable.

3.1.2 Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) is a commonly used metric to evaluate the accuracy of a model's predictions, particularly in the context of time series forecasting. It measures the average percentage difference between the predicted and actual values.

The formula for MAPE is:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{|y_i|} \tag{2}$$

where:

n is the total number of observations Y_i is the actual value of the ith observation $\hat{Y}_i$ is the predicted value of the ith observation The absolute difference between the actual and predicted values is divided by the actual value.

3.1.3 Weighted Mean Absolute Percentage Error (WMAPE)

Weighted Mean Absolute Percentage Error (WMAPE) is a modified version of the MAPE that takes into account the weight of each observation. It is commonly used in situations where some observations are more important than others. WMAPE is calculated as follows:

$$WMAPE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{\sum_{i=1}^{n}|y_i|} \tag{3}$$

where y_i represents the true value at time i, $\hat{y}_i$ represents the predicted value at time i, and n is the number of time steps in the forecast.

3.2 Performance Comparison

3.2.1 Overall Performance Comparison

1. **Train and Test Loss:** Mean squared error (MSE) is used for comparison. From the tables 3 and 4 below, we can conclude that LSTM baseline consistently achieves the lowest training and test losses across all epochs, indicating better performance compared to nn.Linear and our model. Additionally, baseline nn.Linear model seems to have a slightly higher loss compared to the other models, suggesting a potential underfitting issue.

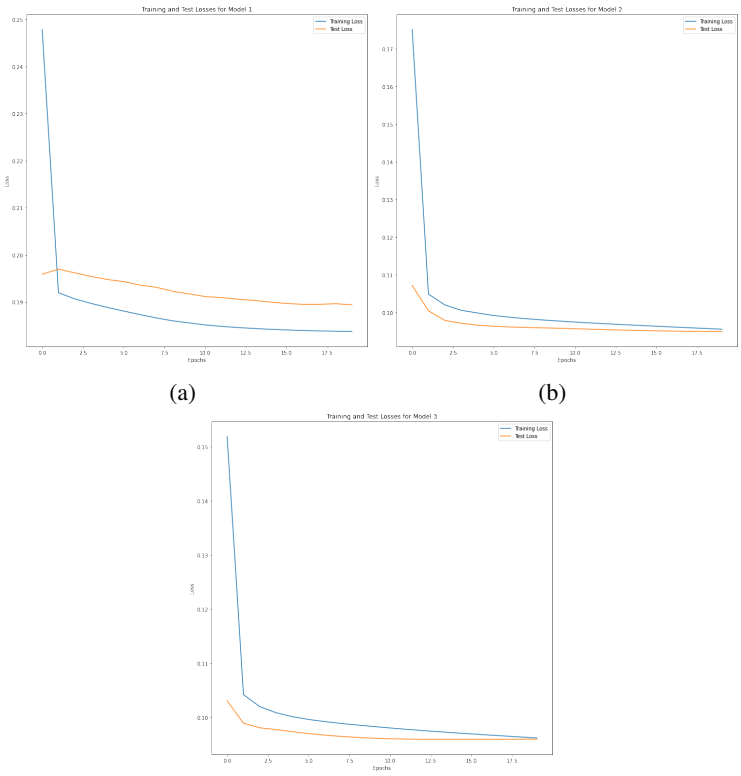
Epoch	nn.Linear	LSTM	Our Model
1	0.2479	0.1751	0.1519
2	0.1920	0.1049	0.1042
3	0.1907	0.1021	0.1020
4	0.1897	0.1006	0.1009
5	0.1889	0.0999	0.1002
6	0.1881	0.0992	0.0996
7	0.1873	0.0988	0.0992
8	0.1866	0.0984	0.0989
9	0.1860	0.0981	0.0986
10	0.1856	0.0978	0.0983
11	0.1852	0.0975	0.0981
12	0.1848	0.0973	0.0978
13	0.1846	0.0970	0.0976
14	0.1844	0.0968	0.0974
15	0.1842	0.0966	0.0972
16	0.1841	0.0964	0.0970
17	0.1840	0.0962	0.0968
18	0.1839	0.0960	0.0966
19	0.1838	0.0958	0.0964
20	0.1838	0.0956	0.0962

Table 3: Training Loss for all models

2. **MAPE** & **WMAPE** The table 5 reveals that LSTM outperforms nn.Linear and our model in terms of forecasting accuracy and performance, as it has the lowest MAPE and WMAPE values. nn.Linear exhibits the highest MAPE and WMAPE values, indicating less accurate

Epoch	nn.Linear	LSTM	Our Model
1	0.1959	0.1072	0.1031
2	0.1970	0.1005	0.0990
3	0.1962	0.0980	0.0981
4	0.1954	0.0972	0.0978
5	0.1948	0.0967	0.0974
6	0.1944	0.0964	0.0970
7	0.1936	0.0962	0.0967
8	0.1931	0.0961	0.0965
9	0.1923	0.0960	0.0963
10	0.1918	0.0959	0.0962
11	0.1912	0.0958	0.0961
12	0.1909	0.0956	0.0960
13	0.1906	0.0955	0.0960
14	0.1904	0.0954	0.0960
15	0.1900	0.0953	0.0960
16	0.1897	0.0952	0.0960
17	0.1895	0.0951	0.0960
18	0.1895	0.0951	0.0960
19	0.1897	0.0950	0.0960
20	0.1894	0.0950	0.0960

Table 4: Test Loss for all models



(a) (b)

(c)

Figure 5: Train vs Test Loss (a) nn.Linear (b) LSTM (c) Our Model

forecasts. Our Model demonstrates performance close to LSTM, although LSTM maintains a slight edge in terms of MAPE.

Table 5: MAPE and WMAPE Values

Model	MAPE	WMAPE
nn.Linear	3.8857	40.8730
LSTM	3.4571	36.4757
Our Model	3.4677	36.4757

3.2.2 Model comparisons in terms of the 4 specified attributes

MSE: By averaging the square error across all time points, the total mean square error is determined. The average MSE across the four meteorological attributes for each of the three models is the overall MSE metric. The table 6 shows the values for total mse for different models.

Table 6: MSE Values

Weather Attribute	nn.Linear	LSTM	Our Model
$\mathrm{p(mbar)}$	0.42	0.68	0.72
$\mathrm{T(degC)}$	1.28	0.86	1.01
rh (%)	21.09	15.31	16.45
$\mathrm{wv(m/s)}$	0.70	0.64	0.63
Overall MSE	5.89815	4.3725	4.7313

We can observe that the mean squared error (MSE) values for three different models, namely nn.Linear, LSTM, and Our Model, for four different weather attributes. The overall MSE values for each model are also shown. The table indicates that Our Model has the highest MSE for each weather attribute, but has a lower overall MSE compared to nn.Linear and LSTM. This suggests that Our Model may have a larger error for individual weather attributes, but performs better than the baseline model.

4 Additional Analysis for your model

4.1 Error propagation

The error propagation function calculates the loss at each iteration by comparing the model's predicted values to the actual values, and uses this information to adjust the model's parameters and improve its predictions. The output of this process includes the projected label information, time list, and loss list, which can be visualized using plots to gain insights into the model's performance over time. Here are the results in the form of tables (table 7) and plots (figure 6, 7, 8) for comparing the proposed architectures when comparing each of the four physical features.

Table 7: Results for Error Propagation

Model	Error Propagation
nn.Linear	0.04872
LSTM	0.03777
Our Model	0.04488

The above table shows the results of the error propagation process for three different models: nn.Linear, LSTM, and Our Model. The results indicate that the LSTM model had the lowest error propagation value of 0.03777, followed by Our Model at 0.04488, and nn.Linear with the highest value of 0.04872.

Figure 6: Error Propogation Curve for baseline model using nn.Linear

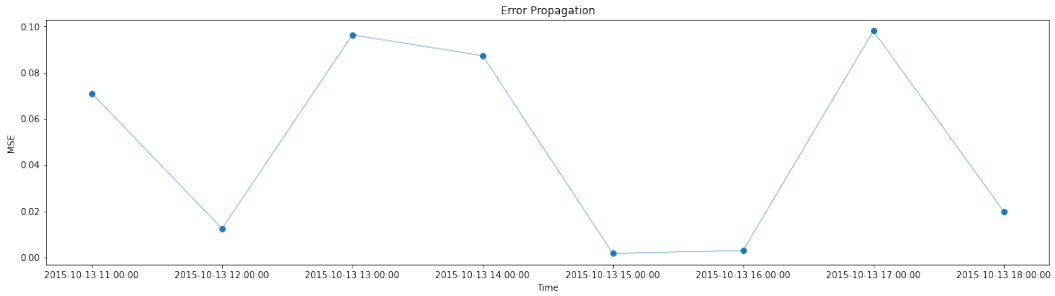
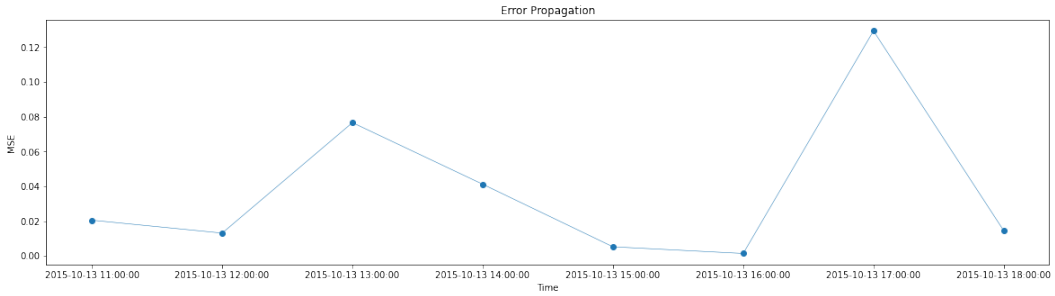


Figure 7: Error Propogation Curve for baseline model using LSTM



4.2 Other Analysis

1. A histogram is a graphical representation of the distribution of data. By creating a histogram of the actual and predicted values, you can visualize how well the predictions match the actual values. This can provide insights into the performance of the model, such as whether the predictions are biased towards certain values or whether there are any outliers in the predictions. The alpha parameter used in the code represents the opacity of the histogram bars.

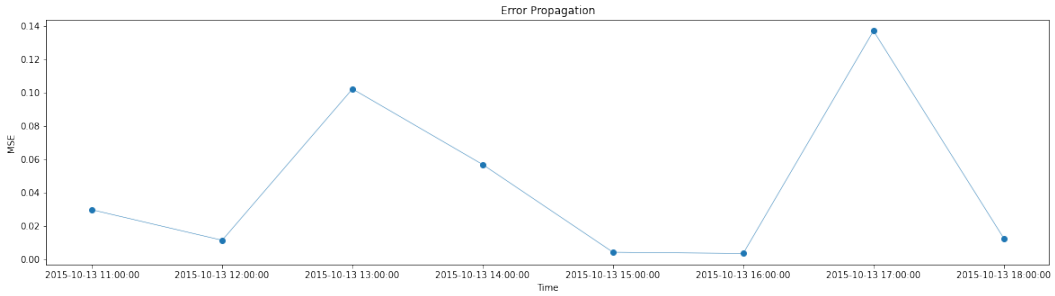
2. Similarly for line plot of the actual vs predicted values, it is another way to visualize the relationship between the actual and predicted values. By plotting the actual and predicted values on the same graph, we can see how well the predictions match the actual values over time or index. This can provide insights into the performance of the model over different time periods or for different subsets of the data.

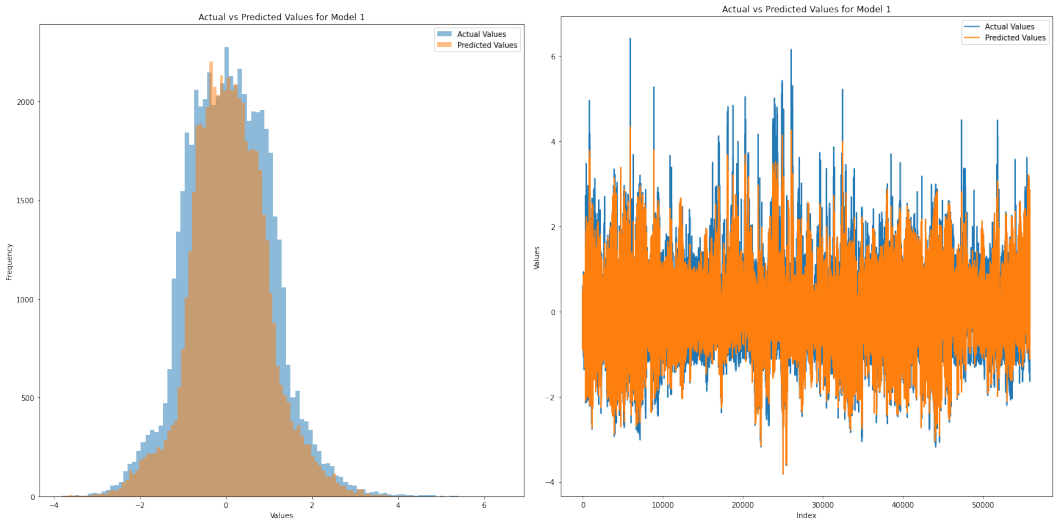
   The line plots and histogram plots can be found in figure 9, 10, 11.

3. Ensembling of 3 models: Ensemble learning is a technique where multiple models are combined to improve the overall prediction performance. We tried ensemble learning and it can be found in the in the optional section of the code file attached .

Overall, these analyses can provide valuable insights into the performance of the model and guide further optimization or improvement efforts.

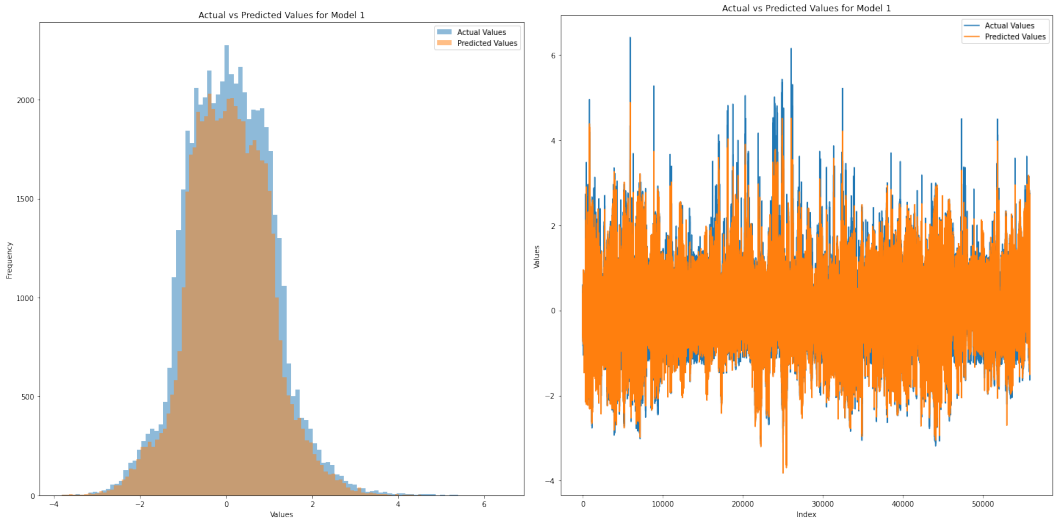
Figure 8: Error Propogation Curve for our model





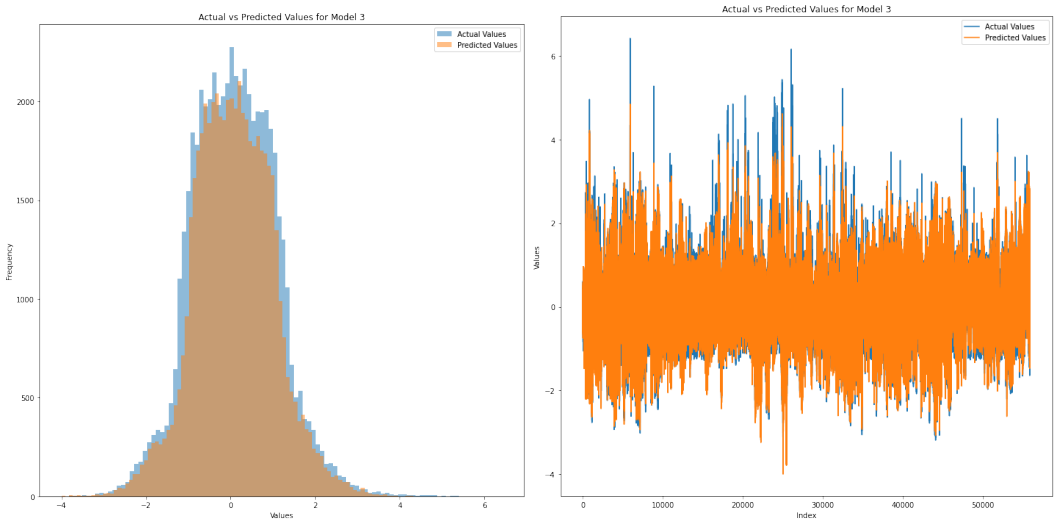
(a) (b)

Figure 9: Actual vs Predicted values for nn.Linear



(a) (b)

Figure 10: Actual vs Predicted values for LSTM



(a) (b)

Figure 11: Actual vs Predicted values for Our Model