# ECSE-552 Assignment 4 Report

Rubert Martin Pardo, Theodore Janson, David “Graham” Smith

April 21, 2022

## 1 Introduction

In this homework assignment the goal was to create a time-series forecasting model. The time-series data under analysis was data related to weather measurements. Given a history of weather attributes from time $t-k$ to $t-1$, the values for four different features at time $t$ were predicted. The four different values to predict are:

- p(mbar) - Atmospheric pressure
- (degC) - Air temperature
- rh (%) - Relative humidity
- wv(m/s) - Wind velocity

The three different models investigated include:

- A fully connected dense network
- An LSTM Regressor
- A Transformer Encoder based model

## 2 Dataset

The dataset was provided as two *.csv* files: one containing the training dataset and the other containing the test data. The training data consists of 56072 samples while the test data consists of 14019 samples. This is roughly a ratio of $4:1$ in terms of training to test samples.

### 2.1 Features

Each training sample consists of fourteen features related to meterological measurements. These are:

1. Date Time - Date and time of the data record
2. p (mbar) - air pressure
3. T (degC) - air temperature
4. Tpot (K) - potential temperature
5. Tdew (degC) - dew point temperature
6. VPmax (mbar) - saturation water vapor pressure
7. VPact (mbar) - actual water vapr pressure
8. VPdef (mbar) - water vapor pressure deficit
9. sh (g/kg) - specfici humidity

10. H2OC (mmol/mol) - water vapor concentration
11. rho (g/m**3) - air density
12. wv (m/s) - wind velocity
13. max. wv (m/s) - maximum wind velocity
14. wd (deg) - wind direction

Wind velocity appears to be a misnomer and is more aptly called ”wind speed” as there is a wind direction measurement as well making the measurement not a velocity by definition [Kol08].

## 2.2 Pre-processing

The data and time values were converted from a string to an integer representing the hour the measurement was made. Min-Max scaling was applied to each feature as it showed an improvement in the networks' performance.

Basic statistical analysis revealed that there were some erroneous values stored within the wind velocity feature of the test data set. Several samples contained a value of $-9999$, which was quite clearly erroneous. These outliers were set to zero to minimize their impact. After removing the outliers, the basic statistics of the datasets are summarized in tables 1-4.

Table 1: p (mbar)

| | Train | Test | Diff |
|---|---|---|---|
| Mean: | 988.83 | 990.73 | -1.90 |
| Std: | 8.35 | 8.22 | 0.13 |
| Min: | 913.60 | 956.96 | -43.36 |
| Max: | 1013.91 | 1015.29 | -1.38 |

Table 2: T (degC)

| | Train | Test | Diff |
|---|---|---|---|
| Mean: | 8.99 | 11.30 | -2.31 |
| Std: | 8.44 | 8.09 | 0.35 |
| Min: | -22.76 | -13.67 | -9.09 |
| Max: | 35.65 | 37.28 | -1.63 |

Table 3: rh (%)

| | Train | Test | Diff |
|---|---|---|---|
| Mean: | 76.21 | 75.22 | 0.99 |
| Std: | 16.50 | 16.36 | 0.14 |
| Min: | 13.88 | 23.64 | -9.76 |
| Max: | 100.00 | 99.90 | 0.10 |

## 2.3 Feature Selection

It was hypothesized that several of the features would be periodic with respect to the duration of a day. Analysis via auto-correlation revealed several features were quasi-periodic with respect to 24 hours. However, it was hard to incorporate this information based on the limitations of the imposed upon the sequence length. The quasi-periodic nature of the air density is shown in Figure 1.

Table 4: wv (m/s)

| | Train | Test | Diff |
|---|---|---|---|
| Mean: | 2.15 | 2.05 | 0.10 |
| Std: | 1.54 | 1.54 | 0.00 |
| Min: | 0.00 | 0.00 | 0.00 |
| Max: | 14.01 | 12.05 | 1.96 |

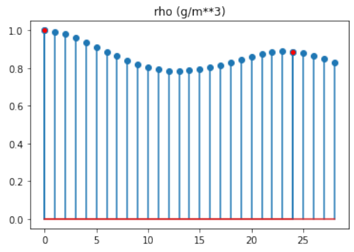

Figure 1: Air density auto-correlation

The next method of feature selection involved trying to quantify the relationships between the constituent variables. First the Pearson correlation coefficient was calculated, however this didn't yield good results and it was hypothesized there is likely non-linear relationships between the features. For this reason mutual information was used instead. A threshold of 1.25 was used to help narrow the selection.

The potential temperature was removed as it is merely the air temperature value converted to Kelvin. No new information is added as this is a linear conversion by just adding 273.

Taking these steps into account the end resulting features used are:

1. p (mbar)
2. T (degC)
3. rh (%)
4. wv (m/s) - wind velocity
5. VPmax (mbar)
6. rho (g/m**3) - air density
7. max. wv (m/s) - maximum wind velocity

## 3 Model Architectures

### 3.1 Dense Network

The dense network has the same number of layers as the other networks, which corresponds to the sequence length which they are trying to predict. The difficulty this this model however, is that it can't process a sequence naturally in the same way as the others. The sequence needs to be flattened into one vector first before being fed into the model. This means that the width of the input layer will be $k \times N$, where N Number of input features. Given that we have four output features, the final layer has 4. The intermediate layers are a fraction of the first layer based on the powers of 2. So the 2nd layer has half the number of neurons as the first layer and the third layer has 1/4 (and so on and so forth). ReLU was selected as the activation function in between layers. After the first layer there is a drop-out layer with a probability of 50%.

### 3.2 LSTM Regressor

The LSTM regressor model is rather simple. It consists of several LSTM modules stacked together. The number of layers corresponds to the length of the sequences we are feeding it during training. At the output of the last LSTM module is a linear layer which goes from the hidden size down to 4, corresponding to the number of features we wish to predict. Additionally, dropout is applied to the output of each LSTM layer.

### 3.3 Attention/Transformer Encoder

The third model implemented was a transformer encoder for time series data. As the full encoder-decoder transformer architecture has had much success in NLP, [eA17] looked to apply it to other sequential data and developed the first transformer based model for unsupervised representation of multivariate time series. The model specifically relies on the transformer encoder described in the seminal paper "Attention is All You Need" [eA17]. The model takes training samples $X \in \mathbb{R}^{w \times m}$, with $w$ being the sequence length, and $m$ being the number of variables (both supplemental features, and features to be learned). Each of these training examples is element multiplied by a corresponding mask matrix $M \in \{0,1\}^{w \times m}$, such that if one wants to predict feature 2 in the last day of a week-long sequence, then $X_{7,2} = 0$. This masked vector is passed through a linear layer, and projected into a d-dimensional space through multiplication by a learnable weight matrix $W \in \mathbb{R}^{d \times m}$. Positional encoding $W_{pos} \in \mathbb{R}^{w \times d}$ are then added to the linear layer's output. Learnable positional encoding are used, as [eA20] showed that they offer better performance than the sinusoidal, deterministic encodings proposed by [eA17]. Thus, the input to the transformer encoder is $U' = U + W_{pos}$. This is fed to the Query, Key and Value parameters of the self-attention layer in the first encoder layer which produces a representation of each feature that includes the encoded feature and its attention score. This is repeated over a determined number of attention layers with a determined number of self-attention heads per layer. Finally, the output of the transformer encoder is passed through another linear layer with ReLU and dropout on its output.

#### 3.3.1 Model Comparison

Naturally, the advantage of the LSTM network over the dense network is that the former learns the evolution of the sequence data through entire dataset, rather than being limited to the data contained in a single N-day sequence. To do this, the LSTM leverages the Markov property of the data such that the output of the previous day contains information related to the entirety of the 'previously seen' data, and is used as input on the current state. While the LSTM can retain previously seen information in hidden states, the encoding of the data from the previous time step considerably affects the representation of the data for next time step. After a few time steps, the effect of a given day on future hours may be greatly reduced. While this presents an issue in text data, as the effect of a word may transcend its local neighbourhood, it may not be an issue in time-series forecasting of weather, which is more Markovian in nature. There are a few theoretical advantages in using a transformer based model over an LSTM model. The transformer allows for learnable position encodings, in contrast to the LSTM, which has a fixed position. This allows for weather data from further in the past to potentially be weighed more heavily than the data the model has just seen. Additionally, the multiple attention heads in transformer models allows for the model to examine different representation subspaces, such as data separated by an hour, or by 5 hours, for example, and gives the model access to the entirety of the sequence's data without suffering information loss like the LSTM.

## 4 Method

### 4.1 Loss function

Each model is trained using the mean square error, or L2 Norm, loss function defined below:

$$MSE = \frac{1}{n}\sum(y - \hat{y})$$

## 4.2 Training Procedure

All training was done using Pytorch Lightning.

### 4.2.1 Dense and LSTM

Training for the Dense Network as well as the LSTM was nothing out of the ordinary. The foward pass on each model was calculated for each training batch. From there the the loss on the mode's output was calculated and the optimizer then updated the weights. For LSTM the Adam optimizer used and for the Dense Network Stochastic Gradient Descent was used.

### 4.2.2 Transformer Training

To train the transformer, the element inverse of the boolean mask used to set values to predict to 0 is applied to the output of the network and to the original sequence. This multiplies the values of interest by 1, and the rest of the features by zero, so that only the values of interest contribute to the loss function. The data split between training and validation was held constant at an 8:2 ratio. A minimum / maximum scaling transform from the Scikit-Learn library was applied to the entirety of the training set. It was saved so that the inverse transform could be applied onto the prediction during testing so that actual weather forecasting could be performed. Additionally, the dense network was trained using mini-batch stochastic gradient descent, while the LSTM and the transformer models were trained using the ADAM optimizers. The batch sizes and learning rates were selected for as hyperparameters. The loss function optimized was MSE loss, and the metrics used to validate performance include MSE and mean absolute percent error (MAPE).

## 4.3 Hyperparameter selection

Hyperparameter selection was done by first determining the optimal network architecture for the transformer model (number of transformer encoder layers, dimensionality of each layer, and the dimensionality of the linear on the initial input). This was done while holding the historical feature length, the batch size, learning rate, and the dropout probability constant. Rather than doing a complete grid search over the design parameters, various different potential architectures were tested, and the optimal one selected by lowest validation loss. Once the transformer model was established, the LSTM and dense network models would take on the same number of layers, but the hidden sizes by testing a few different configurations. Then the historical feature length was selected for each model by testing various values and optimizing on validation loss. Finally, the same was done for the batch, size, dropout and learning rate. The final transformer model determined uses an initial linear size of 32 neurons. Each encoder layer has four attention heads, one for each feature of to predict. There are two attention heads. This model used a sequence length of 12, and a dropout probability of 0.05. Training was done with a batch size of 64, and a learning rate $10^{-3}$. The dense network has 2 hidden layers, with the input being equal to the product of the number of input features and the historical sequence length. The number of neurons for first and second layers are each half the size of their input to reduce the size of the sequence representation. Hyperparameters included a historical sequence length of 4 hours, a dropout probability of 0.1, and a learning rate of $10^{-3}$. The model was trained with a batch size of 64 over 10 epochs. The LSTM network used a hidden size of 100 neurons, and a historical sequence length of 8, which is the number of LSTM layers. It used a dropout probability of 0.1, a learning rate of $10^{-3}$ and a batch size of 64.

# 5 Results

## 5.1 Model comparisons in terms of 4 attributes

The following three metrics will be utilized to compare the performance of each of the three proposed models:

- MAPE: The Mean Absolute Percentage Error is defined as

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

where $n$ is the number of measurements on each time series (single-valued) and $y_i$, $\hat{y}_i$ are the i-th values of the ground truth and predicted values of the time series. MAPE's advantages are it's scale-independency and easy interpretability. However, it generates infinite or undefined values for zero or close-to-zero actual values $\hat{y}_i$. Also, notice that using MAPE for features like Temperature makes no sense, since they have relatively arbitrary zero points.

- WMAPE: Weighted Mean Absolute Percentage Error is defined as:

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

  is a weighted version of the previous metric, on which the inverse of the true value $y_i$ is used as a weight when taking the average. In this way, the infinite error issue is overcome when very small values appear on the time series.

- MDRAE: Median Relative Absolute Error is defined as

$$Median_{i=1,n} \left( \frac{|y_i - \hat{y}_i|}{y_i - b_i} \right) \tag{3}$$

  . Here $b_i$ denotes a benchmark forecast result that correspond to the value of $y$ at an earlier time $b_i = y_{t-M}$, where $M$ is the seasonal period of the time series. If a model's forecast equals to the benchmark's forecast then the result is 1. If the benchmarks forecast are better than ours then the result will be above ¿ 1. If the model is better then it's below 1. MdRAE is more robust to outliers than MAPE, but also diverges when zero values appear on the time series.

Table 5.2 shows the performance of all the models with respect to each feature.

| | MAPE | | | | WMAPE | | | | MDRAE | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **p** | **T** | **rh** | **wv** | **p** | **T** | **rh** | **wv** | **p** | **T** | **rh** | **wv** |
| **Dense Network** | 0.27 | 1.0 | 1.0 | 1.0 | 0.23 | 1 | 1 | 1 | 1 | 6.3 | 5.1 | 1.7 |
| **LSTM** | 0.29 | 0.47 | 0.17 | 1.37 | 0.20 | 0.31 | 0.17 | 0.63 | 0.75 | 1.43 | 0.85 | 1.19 |
| **Transformer encoder** | **0.14** | **0.19** | **0.10** | **0.28** | **0.09** | **0.12** | **0.07** | **0.21** | **0.58** | **0.33** | **0.28** | **0.32** |

Table 5: Three metrics for comparing the proposed architectures when comparing each of the four physical features.

## 5.2 Overall performance

For each feature, the transformer model outperformed the LSTM and the dense model considerably, as seen in . As expected, the LSTM network performed better than the Dense network model. This is because the dense network cannot leverage any sort of recurrence, so all data in the sequence is treated the same with no attention paid to position. We did not expect to see such an increase in performance for the transformer over the LSTM. While transformers do perform considerably better than LSTM in text-like data, weather data is more Markovian in nature. This increase in performance is likely due to the fact that weather is not completely Markovian. Additionally, it is worth noting the dense network only managed to predict pressure to a reasonable level of error. It grossly underperformed when attempting to predict temperature, relative humidity and wind velocity, which are less prone to cycling through gradual risings and sudden drops. The LSTM network saw a large amount of error when attempting to predict wind velocity. This feature is likely more challenging to predict due to the fact that it is much less continuous in nature. Sensor readings may take place during gusts and lulls and create large discontinuities that would considerably effect regression tasks. In the case of the LSTM, if the last historical reading were to have taken place during a gust, it would bear little semblance to the data readings before it, breaking the Markovian property that the model relies on. The transformer would have performed better in this regard as it would have access to the previously seen data without the information loss suffered by the LSTM. Relative humidity and pressure, which are correlated phenomena, undergo the converse. They are more constant geo-temporaly, and can be more easily predicted, as evidenced by the models' performances with regard to this metric.

## 5.3 Error Propagation for Transformer/Encoder

To determine the effect of error propagation in the transformer model, three historical lengths were used: N = 4, 12, 32. The model attempts to predict N hours into the future. Essentially, the model is using predicted values on day $t$ to predict the values on day. $t + 1$ up to day $t + N$. Results in show that there is not a massive decrease in performance due to error propagation. Though performance, as shown in is still higher than with the LSTM model and the dense network models due to the considerable performance gap between the models.

| | p | T | rh | wv |
|---|---|---|---|---|
| 4 Hours | **0.17** | **0.17** | **0.11** | **0.27** |
| 8 Hours | 0.2 | 0.18 | **0.11** | 0.32 |
| 32 Hours | 0.24 | 0.23 | 0.15 | 0.35 |

Table 6: Mean Absolute Percentage Error for error propagation when predicting 4, 12, and 32 hours into the future using the transformer model.

## 5.4 Other Analysis

Additional analysis included removing the additional features used to supplement the model. These included saturation water vapor pressure, air density, and maximum recorded wind velocity. The length of the historical sequence was set to twelve hours to be consistent with the previous tests. The transformer model reported a mean absolute percentage score of 0.15, 0.24, 0.14, and 0.6 for pressure (p), temperature (T), relative humidity (rh) and wind velocity. The loss in performance was not substantial for pressure, but was greater for wind velocity and and relative humidity. This is likely because these features are correlated with other features we are seeking to predict. Notably, air density and maximum wind velocity are correlated with relative humidity and wind velocity respectively. It is worth noting that the model still performed better than the LSTM model, despite the loss of supplemental features. The final additional analysis was to observe the error propagation when the transformer model uses only a few hours worth of historical data to predict a long way into the future. For this analysis, a historical length of 3 hours was used to predict 15 hours into the future. The transformer model reported a mean absolute percent error score of 0.24, 0.26, 0.18, 0.42 for the four features described above. Still, the error remains reasonably low despite the constraints placed on the model. Again, this is likely due to the fact that many of the features have Markovian properties, so the error propagation remains relatively low. A much larger challenge lies in predicting a few days into the future, as weather forecasting stations attempt to do. This naturally comes with the constraint that weather data at a given station over this time frame is dependent on weather data of surrounding geography, which we do not have access to.

## References

[eA17] Ashish Vaswani et Al. Attention is all you need. 2017.

[eA20] George Zerveas et Al. A transformer-based framework for multi- variate time series representation learning. 2020.

[Kol08] Olaf Kolle. Documentation of the Weather Station on Top of the Roof of the Institute Building of the Max-Planck-Institute for Biogeochemistry. https://www.bgc-jena.mpg.de/wetter/Weatherstation.pdf, 2008. [Online; accessed 19-April-2022].