BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Cluj Napoca Temperature Prediction

– DS - Intelligent Modeling report –

Stancioiu Laurentiu
Data Science for Industry and Society
laurentiu.stancioiu@stud.ubbcluj.ro

2022-2023

Abstract

This paper aims to present a comprehensive comparison between the Transformer architecture, specifically an Autoformer[7], and Long Short-Term Memory (LSTM) [2]networks in the context of short-term temperature forecasting for the city of Cluj Napoca. We also want to see if transformers have the potential to beat the "*No free lunch theorem*"[5]. The results of the study show that while for long term temperature forcasting transformers clearly have an advantage over LSTM's, for short term predictions, LSTM's are better.

Figure 1: Graphical abstract of proposed method

Contents

List of Tables

List of Figures

Chapter 1

1. Introduction

1.1 What? Why? How?

Since the infamous paper *Attention Is All You Need*[4] from 2017 the Transformer architecture took the Natural Language Processing world by storm. In recent years we have seen advances in Image Processing, with the most popular being Stable Diffusion. In conclusion, important areas of Machine Learning are dominated by a neural network architecture that uses an attention mechanism.

With all this advances within just a few years, an area of Machine Learning from which most real world apps are made is still dominated by more classical models, tabular data. It is so untouchable that for classification purposes, various implementations of Gradient Boosting still remain state of the art.

In contrast, neural networks have found an area in tabular data that work better than classical algorithms: forcasting. The state of the art until 2021 were Long-Short Term Memories (LSTM's)[2] a type of Recurrent Neural Network (RNN)[3]. But since 2021 two new models were developed which used the attention mechanism such as Informer, Autoformer[7] or the most recent by the same group from Tingshua University called TimesNet[6]. From their articles we can clearly see that transformers are SOTA for long-term weather prediction.

So we wanted to know if Transformers are also the best model for short-term predicions as well. After all, usually most people want to see what to wear for the following day or even the following hours, and only on special events such as holidays or weekends would require accuracy for such prediction lenghts like the one presented in figure 1.1. We will see the advantages of LSTM's such as better training and prediction time, but also for the transformer architecture, such as the prediction of spikes of temperatures which LSTM's can't handle.

Table 1: Multivariate results with different prediction lengths $O \in \{96, 192, 336, 720\}$. We set the input length I as 36 for ILI and 96 for the others. A lower MSE or MAE indicates a better prediction.

Models		**Autoformer**		Informer[41]		LogTrans[20]		Reformer[17]		LSTNet[19]		LSTM[13]		TCN[3]	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT*	96	**0.255**	**0.339**	0.365	0.453	0.768	0.642	0.658	0.619	3.142	1.365	2.041	1.073	3.041	1.330
	192	**0.281**	**0.340**	0.533	0.563	0.989	0.757	1.078	0.827	3.154	1.369	2.249	1.112	3.072	1.339
	336	**0.339**	**0.372**	1.363	0.887	1.334	0.872	1.549	0.972	3.160	1.369	2.568	1.238	3.105	1.348
	720	**0.422**	**0.419**	3.379	1.388	3.048	1.328	2.631	1.242	3.171	1.368	2.720	1.287	3.135	1.354
Electricity	96	**0.201**	**0.317**	0.274	0.368	0.258	0.357	0.312	0.402	0.680	0.645	0.375	0.437	0.985	0.813
	192	**0.222**	**0.334**	0.296	0.386	0.266	0.368	0.348	0.433	0.725	0.676	0.442	0.473	0.996	0.821
	336	**0.231**	**0.338**	0.300	0.394	0.280	0.380	0.350	0.433	0.828	0.727	0.439	0.473	1.000	0.824
	720	**0.254**	**0.361**	0.373	0.439	0.283	0.376	0.340	0.420	0.957	0.811	0.980	0.814	1.438	0.784
Exchange	96	**0.197**	**0.323**	0.847	0.752	0.968	0.812	1.065	0.829	1.551	1.058	1.453	1.049	3.004	1.432
	192	**0.300**	**0.369**	1.204	0.895	1.040	0.851	1.188	0.906	1.477	1.028	1.846	1.179	3.048	1.444
	336	**0.509**	**0.524**	1.672	1.036	1.659	1.081	1.357	0.976	1.507	1.031	2.136	1.231	3.113	1.459
	720	**1.447**	**0.941**	2.478	1.310	1.941	1.127	1.510	1.016	2.285	1.243	2.984	1.427	3.150	1.458
Traffic	96	**0.613**	**0.388**	0.719	0.391	0.684	0.384	0.732	0.423	1.107	0.685	0.843	0.453	1.438	0.784
	192	**0.616**	**0.382**	0.696	0.379	0.685	0.390	0.733	0.420	1.157	0.706	0.847	0.453	1.463	0.794
	336	**0.622**	**0.337**	0.777	0.420	0.733	0.408	0.742	0.420	1.216	0.730	0.853	0.455	1.479	0.799
	720	**0.660**	**0.408**	0.864	0.472	0.717	0.396	0.755	0.423	1.481	0.805	1.500	0.805	1.499	0.804
Weather	96	**0.266**	**0.336**	0.300	0.384	0.458	0.490	0.689	0.596	0.594	0.587	0.369	0.406	0.615	0.589
	192	**0.307**	**0.367**	0.598	0.544	0.658	0.589	0.752	0.638	0.560	0.565	0.416	0.435	0.629	0.600
	336	**0.359**	**0.395**	0.578	0.523	0.797	0.652	0.639	0.596	0.597	0.587	0.455	0.454	0.639	0.608
	720	**0.419**	**0.428**	1.059	0.741	0.869	0.675	1.130	0.792	0.618	0.599	0.535	0.520	0.639	0.610
ILI	24	**3.483**	**1.287**	5.764	1.677	4.480	1.444	4.400	1.382	6.026	1.770	5.914	1.734	6.624	1.830
	36	**3.103**	**1.148**	4.755	1.467	4.799	1.467	4.783	1.448	5.340	1.668	6.631	1.845	6.858	1.879
	48	**2.669**	**1.085**	4.763	1.469	4.800	1.468	4.832	1.465	6.080	1.787	6.736	1.857	6.968	1.892
	60	**2.770**	**1.125**	5.264	1.564	5.278	1.560	4.882	1.483	5.548	1.720	6.870	1.879	7.127	1.918

* *ETT* means the ETTm2. See supplementary materials for the **full benchmark** of ETTh1, ETTh2, ETTm1.

Figure 1.1: Table with multivariate results with different prediction lenghts (96, 192, 336, 720 hours) [7]

1.2 Paper structure and original contribution(s)

The research presented in this paper advances the theory, design, and implementation of two particular models. We implement a LSTM model and a Autoformer model with the scope of comparing them on how well they predict the temperature of the city of Cluj Napoca. The principal bibliographical references will be"Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting"[7] from Wu *et. all* and *Long-Short Term Memory*[**?**] from Sepp Hochreiter and Jürgen Schmidhuber.

The first chapter is a short introduction in the method used, then we will talk about the problem which we tried to overcome, some related work, and finally we will describe every step of our approach and come up with some conclusions.

Our main contribution would be the comparison of two different models on the weather data of Cluj Napoca, how well they behave, and if we can use a single model for all our predictions.

Chapter 2

Scientific Problem

2.1 Problem definition

Weather is getting harder and harder to predict. The same models that worked 20 years ago, do not work anymore because of more and more unforseen phenomena. As a famous meteorologist from Romania, Florin Busuioc would recall in an interview, when he first started they could predict weather with a small amount of error for one or two weeks. Nowadays even the daily forecast is getting harder and harder to predict using the same mathematical algorithms.

The question is: do we know of such an algorithm which learns from past data and can accurately predict never-before-seen data with a high degree of precision. Of course, we would point to neural networks, which can accurately predict the next word in NLP, or if a dog would like to go to the other side of the street when it comes to image/video labeling. Researchers and weather apps surely use neural networks in order to predict what would happen in 6 hours, tomorrow or in the next weeks. But is there an algorithm which can predict long-term forcasts but also short-term ones? This is the question which we try to answer in this report. Are we getting closer and closer of an algorithm which can be used for all areas of Machine Learning or like the *No free Lunch Theorem* states, that there's no point in searching for the "best" algorithm in a general sense, because what's best depends on the specifics of the problem you're dealing with. In our case the specifics of the problem would be the time of prediction

Chapter 3

State of the art/Related work

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN)[3] architecture that was developed to address the issue of long-term dependencies in sequence prediction problems.

Traditional RNNs have a significant limitation; they struggle to retain information from earlier time steps as the sequence gets longer, an issue known as the vanishing gradient problem. LSTMs overcome this problem with a sophisticated cell state architecture that allows them to store and retrieve information over long periods of time, making them highly effective for many sequence prediction problems.

Imagine you're reading a book, and you come across a paragraph that introduces a new character. As you continue reading, you encounter references to that character throughout the story, and you're able to remember who they are and the role they play, even if there are long gaps between their appearances. If we transpose this to weather data you could say that last week was an unusual week in terms of temperatures. They were higher by 2-3 degrees than last year for example. But if we put the sequence lenght until last year (which is not recomended) then the algorithm remembers that this could be an anomaly, and wouldn't let temperature skyrocket following last weeks trend.. The LSTM network comprises LSTM cells, each of which includes an input gate, a forget gate, and an output gate. These gates together control the flow of information into, out of, and within the cell.

- The input gate decides how much new information should be stored in the cell. It determines the relevance and importance of the current input based on the context.

- The forget gate determines how much information should be forgotten or discarded from the cell's memory. It helps the LSTM focus on the most relevant information and filter out what is not needed.

- The output gate controls how much information from the cell should be shared with the next layers in the neural network. It selects the most useful information to pass on as output.

- These gates use sigmoid functions, enabling them to output values between 0 and 1, representing the amount of information to retain or discard.

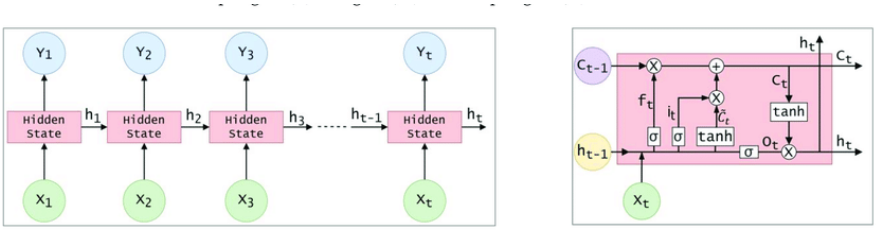

Figure 3.1: The architectures for time-series forecasting model: (a) Recurrent Neural Network (RNN); (b) Long-Short Term Memory (LSTM).[1]

Transformers are a type of model architecture used in the field of deep learning, particularly in natural language processing (NLP). But recently Transformers also do image generation (Dall-E), classification time-series forcasting or anomaly detection. They rely entirely on a mechanism called 'attention' to draw global dependencies between input and output. The key innovation of Transformers is the 'self-attention' mechanism, which allows them to weigh and prioritize different parts of the input when producing an output.

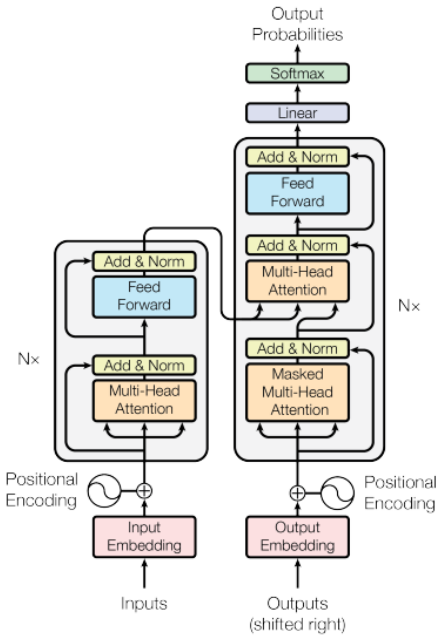

Figure 3.2: Attention model architecture[4]

The multi head attention layer helps the model focus to different aspects of the input looking for different patterns, information or context. Imagine a group of people that have different knowledge and they are trying to make a decision as a group. Not all people have the same information, but by working in a group, or in parallel in the case of the algorithm, their finding produce a more comprehensive and meaningful solution just because of those different findings. If we transpose it to our problem, then one of these attention heads may have an expertise in wind direction, while the other may be good at finding the right humidity and the other can describe how much rain will there be in the following hour. Individually these information can have a more or less correlation to predicting the temperature for the next day. But together, they have a higher chance of giving the right prediction.

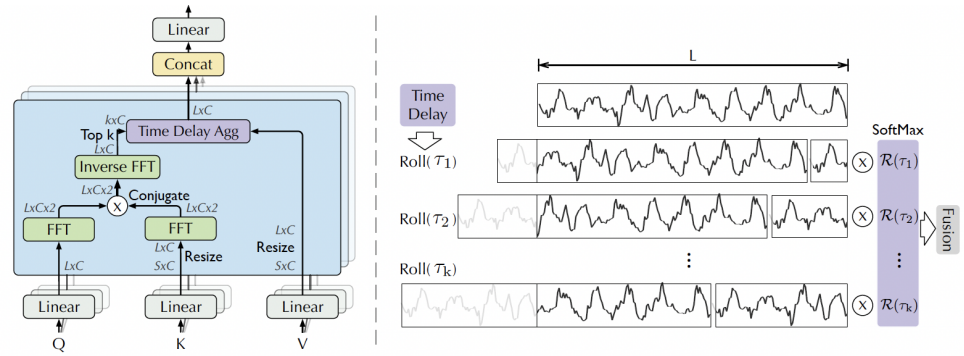

Figure 3.3: Auto-Correlation model architecture (left) and Time-Delay Aggregation (right)[7]

In the case of the Autoformer model architecture, the difference between the Auto-Correlation mechanism and self-attention is that the first discovers period-based dependencies by calculating the series autocorrelation and aggregates similar sub-series by time delay aggregation. The easiest way to think of this would be related to figure 4.4 d) where we see that we take a sub-series of the full series and try to see if there are any patterns which we could learn from the past sub-series. Self attention only takes into account scattered points in our time dataset. By seing the trends of the sub-series we make a calculation of the following sub-series which needs to be predicted. These calculations are made using Fast Fourier Transforms (FFT).

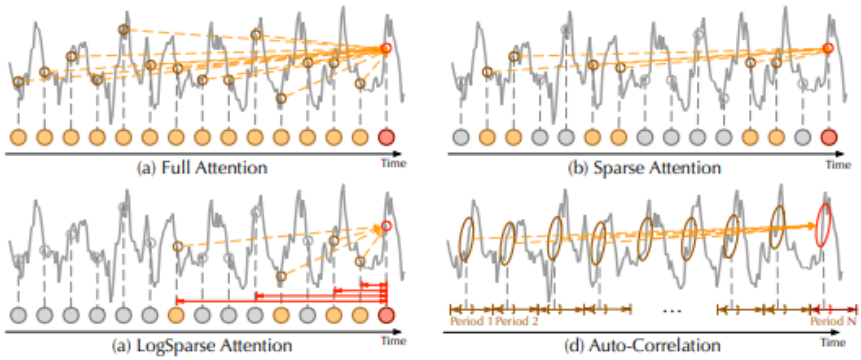

Figure 3.4: Auto-Correlation vs. self-attention family. Full Attention [41] (a) adapts the fully connection among all time points. Sparse Attention (b) selects points based on the proposed similarity metrics. LogSparse Attention (c) chooses points following the exponentially increasing intervals. Auto-Correlation (d) focuses on the connections of sub-series among underlying periods[7]

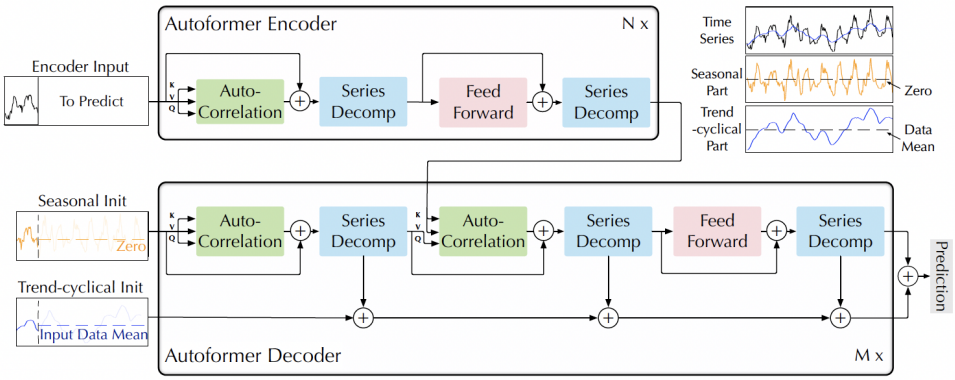

Figure 3.5: Autoformer architecture. The encoder eliminates the long-term trend-cyclical part by series decomposition blocks (blue blocks) and focuses on seasonal patterns modeling. The decoder accumulates the trend part extracted from hidden variables progressively. The past seasonal information from encoder is utilized by the encoder-decoder Auto-Correlation (center green block in decoder).[7]

Chapter 4

Investigated approach

4.1 Proposed approach - methodology

4.1.1 Exploratory Data analysis (EDA)

First, we need to see if the temperatures are between a certain threshold by month. As we can see in figure 4.1 the test data which was taken from April 2020 until May 2023 looks not far away from the training data. Also, as we could feel in recent years, the months of March and April have a tendency of being colder while December and January seem to be hotter on average.

	January	February	March	April	May	June	July	August	September	October	November	December
2008	-1.92	1.5	5.44	10.01	14.67	18.79	18.93	19.75	13.48	9.83	3.71	1.09
2009	-2.07	-0.82	3.56	12.51	15.39	18.13	20.46	20.12	16.62	9.5	5.92	-0.13
2010	-2.72	1.01	4.28	10.14	14.96	18.61	20.36	20.33	13.81	6.62	6.7	-2.09
2011	-3.31	-2.7	4.72	10.47	14.71	18.63	19.62	20.14	17.28	7.63	0.02	1.07
2012	-2.99	-7.15	4.07	11.21	15.42	19.89	23.2	21.42	17.59	10.31	4.58	-2.42
2013	-2.13	2.0	3.01	11.43	16.16	18.96	20.33	21.38	13.49	9.73	6.73	-2.06
2014	0.34	3.12	8.26	11.51	15.1	18.22	20.44	19.74	16.12	10.51	4.62	1.34
2015	-0.55	0.26	5.46	9.18	15.54	19.13	21.81	21.92	17.19	9.46	5.74	1.51
2016	-2.51	5.14	6.09	12.48	14.22	19.96	20.56	19.38	16.55	8.52	2.84	-2.77
2017	-6.52	1.73	8.32	9.46	15.56	20.24	20.8	22.19	15.55	9.84	4.96	1.49
2018	0.24	-0.17	3.32	14.94	18.16	19.35	20.28	21.98	16.3	12.03	5.37	-0.25
2019	-1.77	1.72	6.92	11.6	14.09	21.36	20.35	21.97	16.57	10.6	8.8	0.69
2020	-2.66	2.31	5.96	9.76	13.35	19.09	19.86	21.24	17.3	11.21	3.36	3.25
2021	-0.39	1.46	3.15	7.88	13.8	19.38	21.99	19.04	14.15	7.72	3.94	1.26
2022	-1.44	1.66	3.2	8.44	15.34	20.15	21.88	21.67	14.07	10.3	5.61	1.93
2023	3.08	0.53	5.64	8.05	12.75	-	-	-	-	-	-	-

Figure 4.1: Average of monthly temperatures from January 2008 up until May 2023

Next, we wanted to have a better understanding of the data we scraped from the API. For Data Collection we used https://openweathermap.org/history where we collected data from the first of January 2008 until the first of May 2023 at an hourly rate. That would make a total of 139394 entries. The API call resulted in 28 columns which were more or less relevant for our use case.

Parameter	Description
city_name	City name
lat	Geographical coordinates of the location (latitude)
lon	Geographical coordinates of the location (longitude)
main.temp	Temperature
main.temp_min	Minimum temperature at the moment (optional)
main.temp_max	Maximum temperature at the moment (optional)
main.feels_like	This temperature parameter accounts for the human perception of weather
main.pressure	Atmospheric pressure (on the sea level), hPa
main.humidity	Humidity, %
main.dew_point	Atmospheric temperature (varying according to pressure and humidity) below which water droplets begin to condense and dew can form
wind.speed	Wind speed
wind.deg	Wind direction, degrees (meteorological)
wind.gust	Wind gust
clouds.all	Cloudiness, %
rain.1h	Rain volume for the last hour, mm
rain.3h	Rain volume for the last 3 hours, mm
snow.1h	Snow volume for the last hour, mm (in liquid state)
snow.3h	Snow volume for the last 3 hours, mm (in liquid state)
weather.id	Weather condition id
weather.main	Group of weather parameters (Rain, Snow, Extreme etc.)
weather.description	Weather condition within the group
weather.icon	Weather icon id
visibility	Average visibility, metres
dt	Time of data calculation, unix, UTC
dt_iso	Date and time in UTC format
timezone	Shift in seconds from UTC

Figure 4.2: Parameter-Description table of all data taken from the API

From these resulted key-value pairs we cut those that were not important for us, or had categorical values. After the first cut, we remained with the features from which we gathered statistics present in figure 4.3

Name	count	mean	std	min	25%	50%	75%	max
T(degC)	139391.0	9.834	9.46	-23.02	2.24	9.7	16.91	37.61
Vis(m)	139391.0	8687.608	2751.263	49.0	9999.0	10000.0	10000.0	10000.0
Tdew(degC)	139391.0	5.085	7.569	-24.39	-0.45	5.15	11.35	25.19
Tfeel(degC)	139391.0	8.745	10.259	-29.27	0.69	8.8	16.7	37.74
Tmin(degC)	139391.0	8.543	9.542	-26.18	0.84	8.45	15.75	37.0
Tmax(degC)	139391.0	12.123	9.636	-21.25	4.7	12.0	19.4	42.7
p(mbar)	139391.0	1016.13	7.613	978.0	1012.0	1016.0	1021.0	1044.0
h(%)	139391.0	75.659	19.814	7.0	62.0	81.0	93.0	100.0
ws(m/s)	139391.0	2.319	1.694	0.0	1.03	2.0	3.1	18.0
wd(deg)	139391.0	154.977	117.502	0.0	60.0	130.0	270.0	360.0
rain_1h(mm)	139391.0	0.083	0.385	0.0	0.0	0.0	0.0	18.0
rain_3h(mm)	139391.0	0.025	0.409	0.0	0.0	0.0	0.0	27.0
snow_1h(mm)	139391.0	0.012	0.084	0.0	0.0	0.0	0.0	2.28
snow_3h(mm)	139391.0	0.005	0.11	0.0	0.0	0.0	0.0	9.0
clouds(%)	139391.0	38.215	38.417	0.0	0.0	20.0	75.0	100.0

Figure 4.3: Statistics of each feature in the dataset

As we can see after some feature selection we have 15 more columns left from which the temperature is our target. But, as it can be seen in fig 4.2., we also have 4 more temperatures. The next step was to wonder if these temperature have a strong correlation with one another. So we made a correlation

matrix seen in figure4.4.

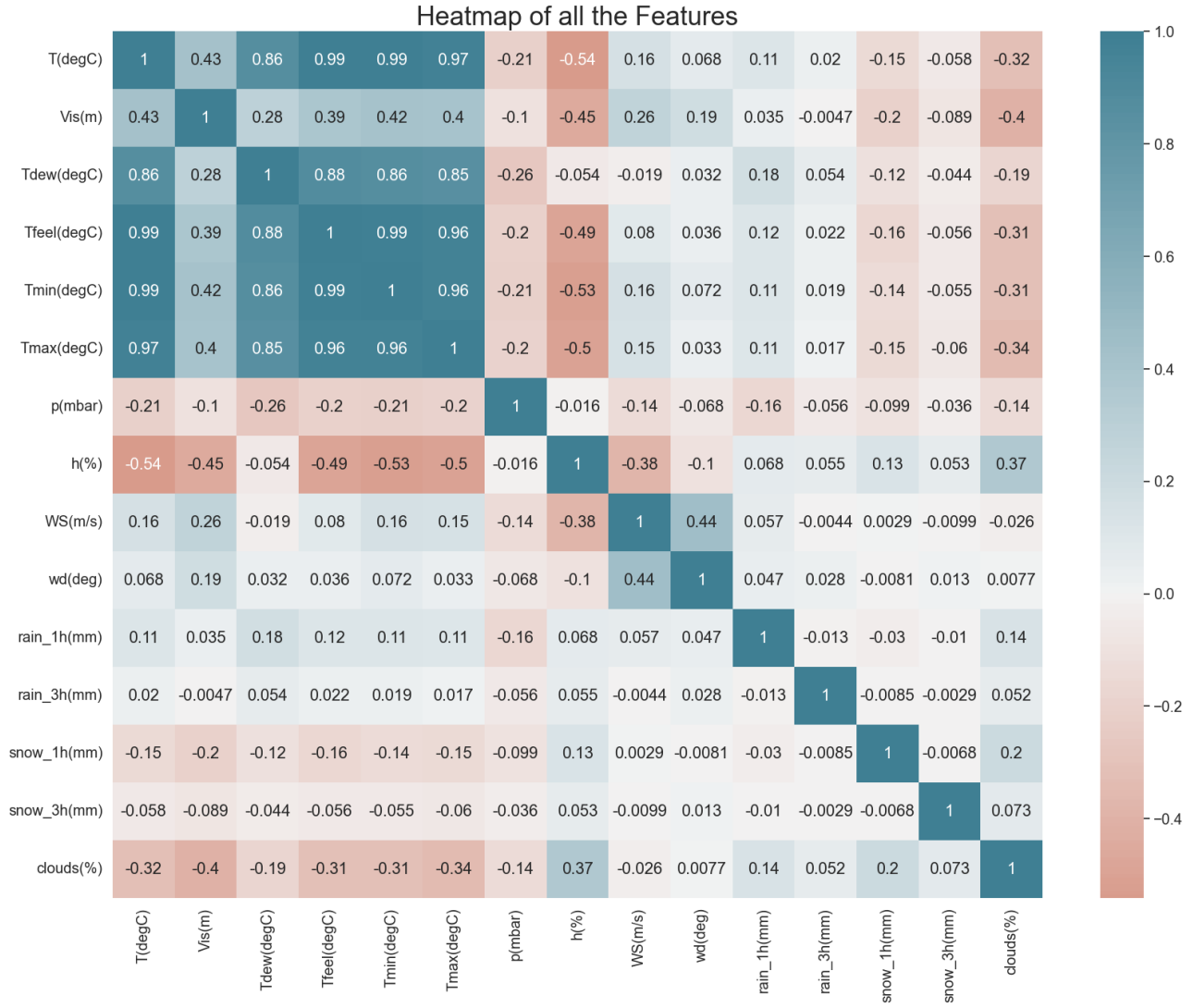

Figure 4.4: Correlation matrix of the features selected in the first step

This matrix was a graphical representation of what we suspected: that temperatures are proportionally correlated with one-another, so there is no need for us to have all of them in the dataset. So we selected the dew point to be our only temperature characteristic in the training dataset. As target we chose the temperature as this is a temperature forcasting. Furthermore, we can see that we have an inverse correlation between temperatures and humidity but no correlation between humidity and the dew point temperature. Also, there is a slight inverse correlation between the clouds and the visibility which would be expected. The final selected features were: Dew Point temperature, visibility, pressure, humidity, wind speed and wind degree, rain volumes and snow volumes.

After this we had to see if we have any missing values in our features. There were missing values for the Dew Point temperatures and visibility, for which we chose to fill with the value above it, and

for rain and winter volumes, which we chose to fill with zeroes because we suspected from common sense that all values which were N/a are actually zeroes. The plots resulted from all data available 2008 - 2023 can be seen in figure 4.5

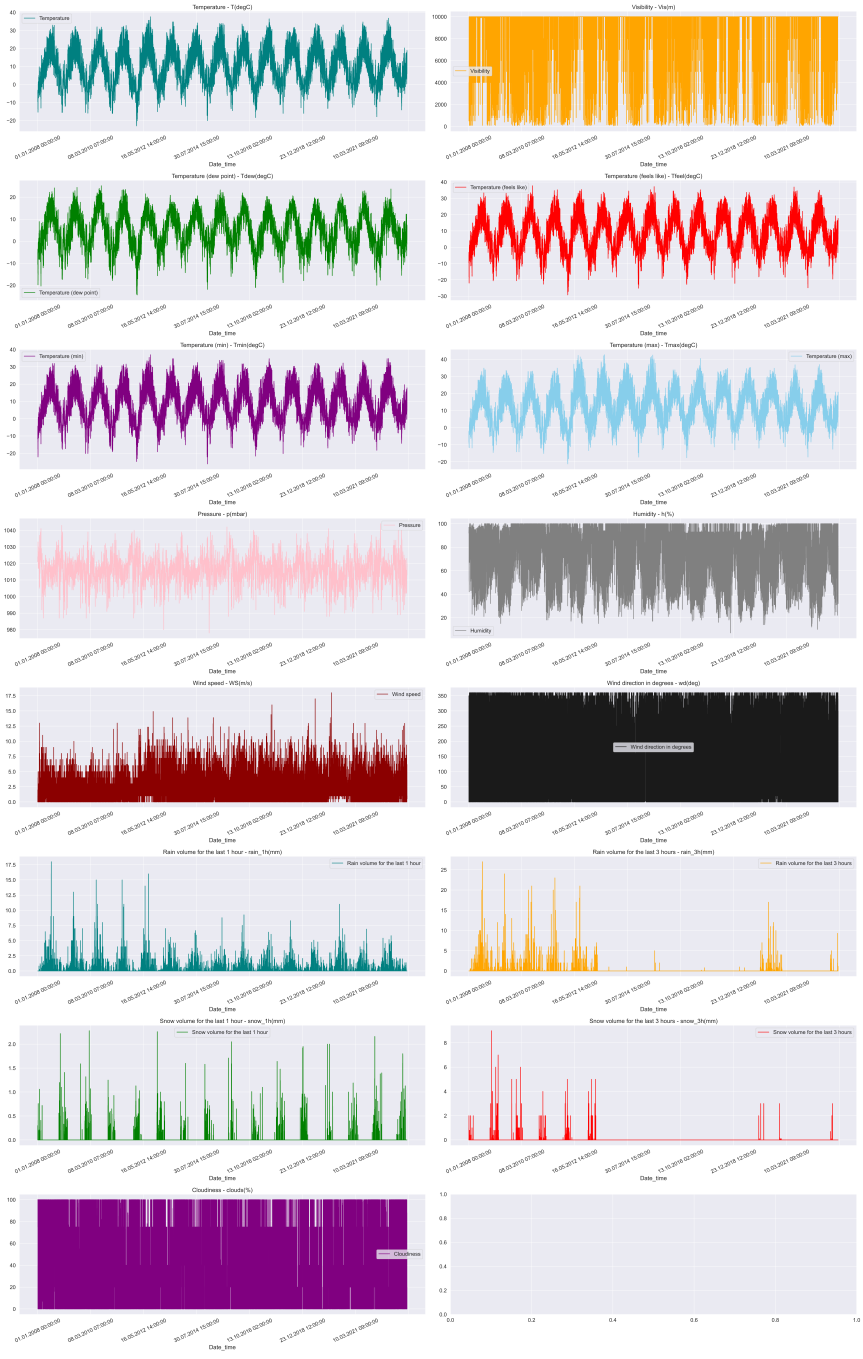

Figure 4.5: Features plot for data ranging from all the dataset

We can see that almost all features (except cloud percentage and wind degree) have seasonality. In the next step we wanted to see the distribution of our features and if they have any obvious anomalies. So we made the density plot at figure 4.6

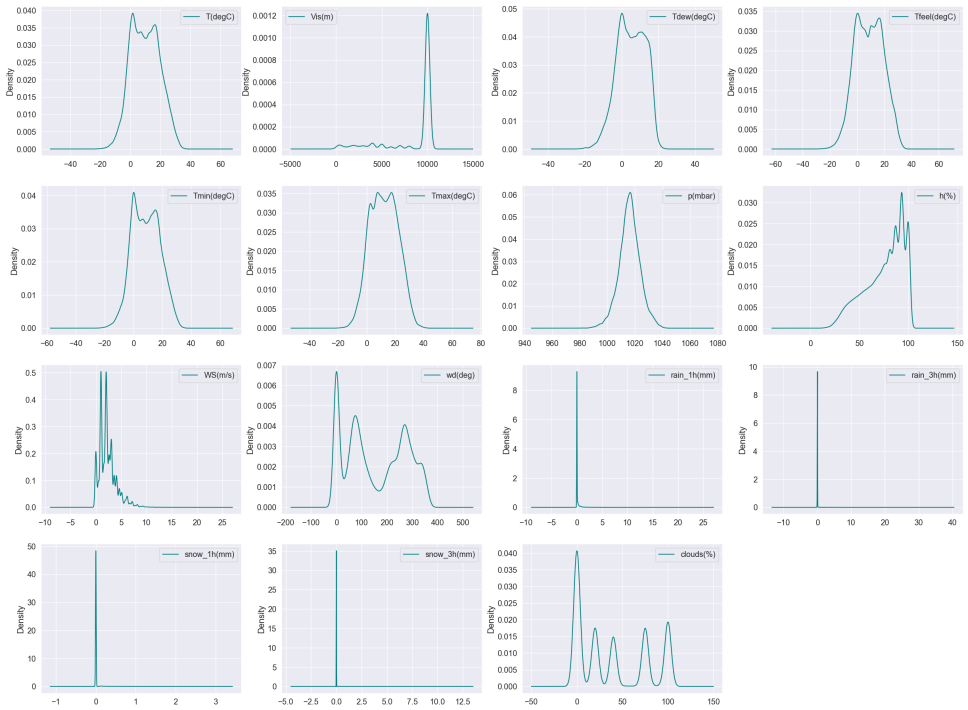

Figure 4.6: Density plot of all available features

4.2 Numerical validation

For LSTM we tried several architectures. We tried ConvLSTM's and also Stacked LSTM's but with no success. Stacked LSTM's were bringing good results, but surprisingly not better than a single LSTM. Also we tried different hyperparameters such as dropouts, batch size, learning rates , number of hidden units or l1 and l2 regularizers. The best hyperparameter combination was: learning rate of 0.001, 32 hidden units, a batch size of 256, a step of 1 (so no data was skipped) and 250 epochs. It is worth mentioning that we had checkpoints at each epoch, and we had an epoch delay of 10 before escaping to the best model checkpoint. This means that if the validation dataset hasn't improved after 10 epochs we would escape the training and save the weight of the best checkpoint. We had models for 6 , 8, 10, 12, 24 and 48 hours. All trainings aproximately 3-6 minutes on GPU and approximately 1 hour and 30 minute on CPU's. We used an Nvidia 1650 Ti 4GB GPU and an AMD Ryzen 7 4800H as CPU. We used tensorflow and keras as the frameworks for building the model. The furthest epoch we went to was 28, so the data didn't improve after 18 epochs. We tried with lower learning rates and/or higher dropouts, but we had overfitting, or the training model staying in a local minima. The optimizer used was Adam (Adaptive Moment Estimation).

The second algorithm we tried was the Autoformer. with this we modified the predict.ipynb file which was contained in this github repository. We also modified some .sh files or the dataloader.py

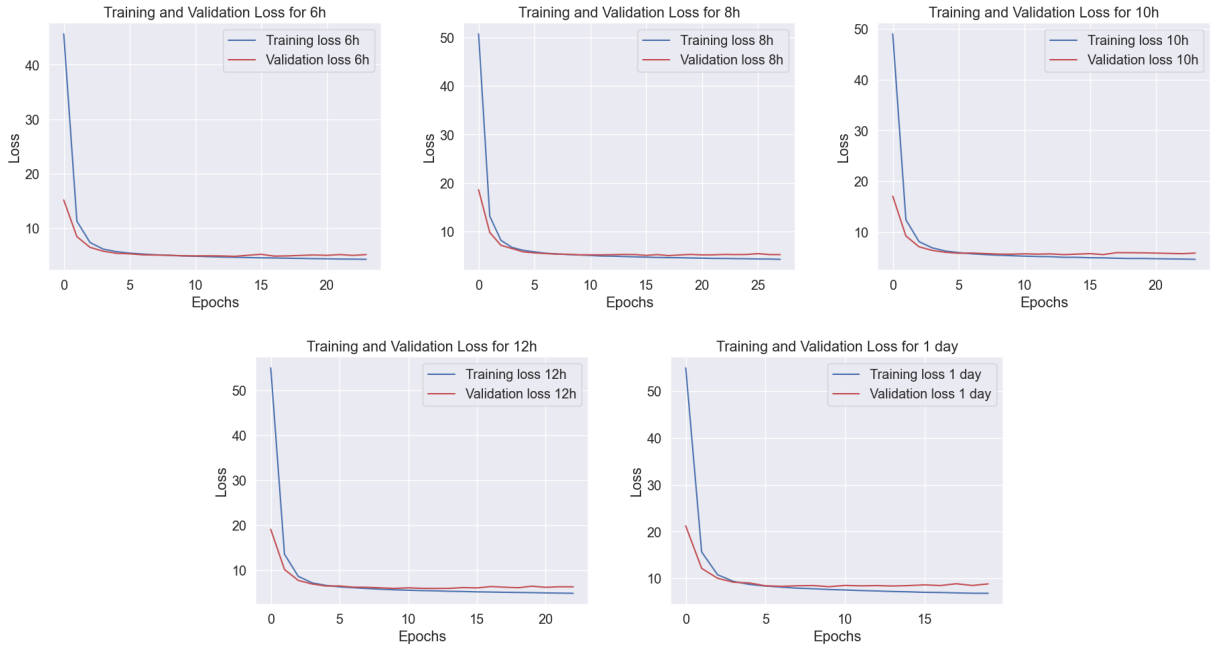

Figure 4.7: All training / validation loss plots for the LSTM architecture

file in order to work with our data. In this case we used the pytorch library for our training and had the following hyperparameters: dropout = 0.05, 2 encoding layers and one decoding layer, 3 AutoCorrelation heads. The activation function used was GELU (Gaussian Error Linear Unit), with 3 epochs and a batch size of 128. The learning rate was 0.0001. The training lasted for both models for about an hour on GPU's

4.2.1 Methodology

The metrics which we used for calculating the results of our validation data were Mean Squared Error (MSE) Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). MSE calcualtes the average of the squared differences between the predicted and the actual values and RMSE is the squared root of MSE. MAE calculates the average of absolute differences between the predicted and the actual values.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.1}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.2}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.3}$$

The difference between MAE and RMSE is the sensitivity to outliers. We use RMSE if the magnitude of the errors are important while MAE is used if we wanted to treat the errors equally. All these metrics are interpretable and they describe in our case how far away can the predicted temperature can be, on average from the actual temperature. We used 13 years of training data and 2 years of validation data. From the 29 initial features we ended up with 11 of them after the feature selection part of our comparison.

4.2.2 Data

We collected data from the OpenWeather API for a cost of 11 dollars, with the point of interest being Babes Bolyai University of Cluj Napoca situated on no. 1 Kogalniceanu street. After data cleaning we ended up with 139391 samples. Three of the samples were dropped because of extreme temperatures (2800 degrees) as we realized that there was an error when receiving the dataset.

4.2.3 Results

The results of the implementations are presented in table 4.1

Model	**RMSE**	**MAE**
LSTM 24 hours	3.01	2.39
Autoformer 24 hours	3.25	2.23
LSTM 12 hours	2.39	1.85
Autoformer 12 hours	2.75	2.08

Table 4.1: RMSE and MAE results of the implementations of LSTM and Autoformer

We can see from the results that the Autoformer is more affected by outliers than the LSTM as we have a smaller MAE but a bigger RMSE. Also, we can see that the shorter the time, the worse is the Autoformer at predicting the Temperature then the LSTM. In the following figures we can see the difference in predictions for the LSTM figures and for the Autoformer at figure . We can see that the LSTM is particularly bad in predicting extreme temperatures , while the transformer architecture mimics more or less some of the predictions but needs more iterations to get there for some of the predictions but for others it makes bad errors. This would be sign that the Autoformer was not trained long enough

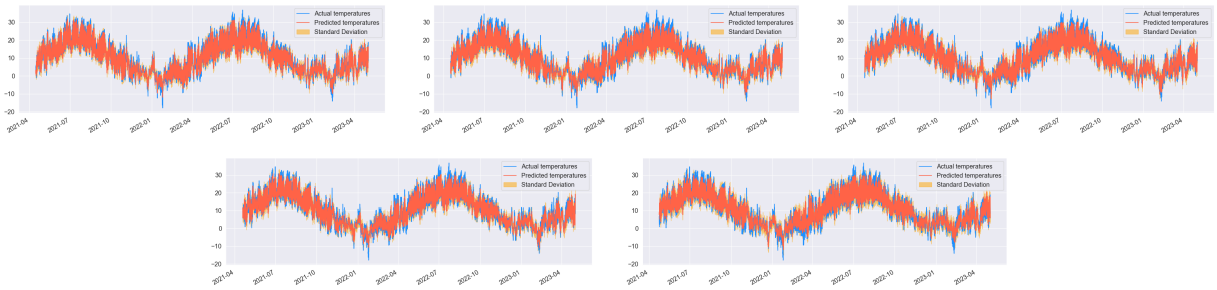

Figure 4.8: Actual vs. predicted values for LSTM's test data for 6 - 24 hours

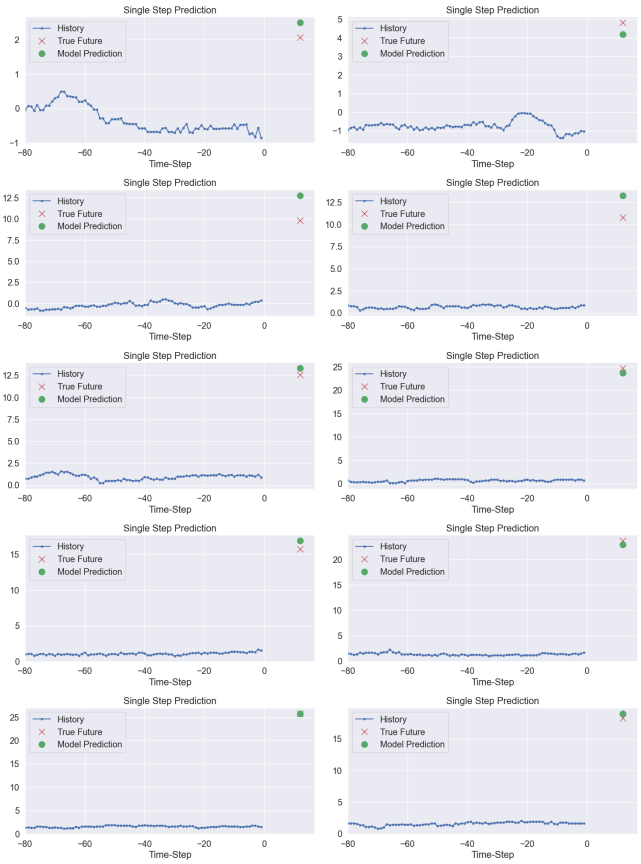

Figure 4.9: Single Step 12 hours prediction using the LSTM architecture

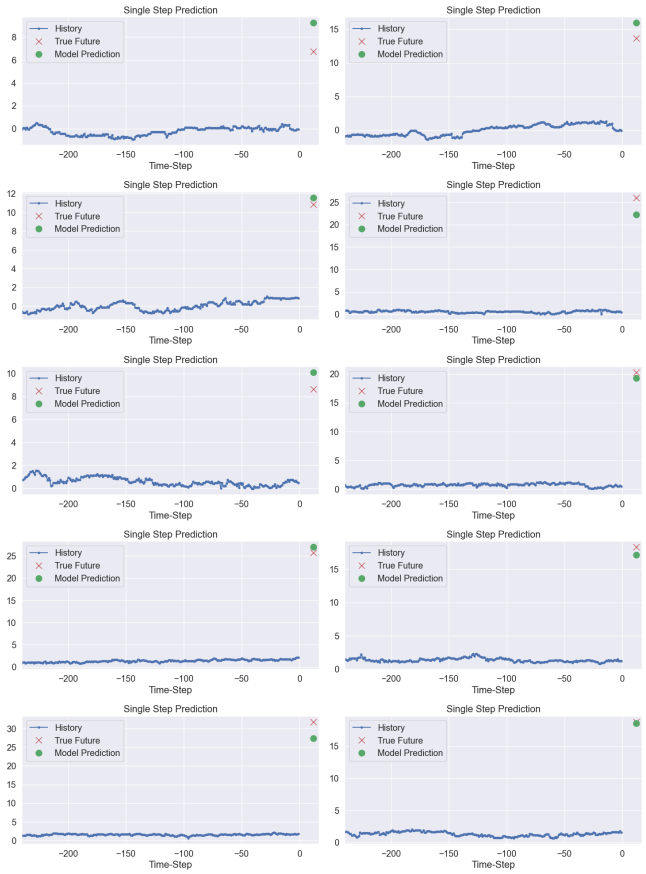

Figure 4.10: Single step 1 day prediction using the LSTM architecture

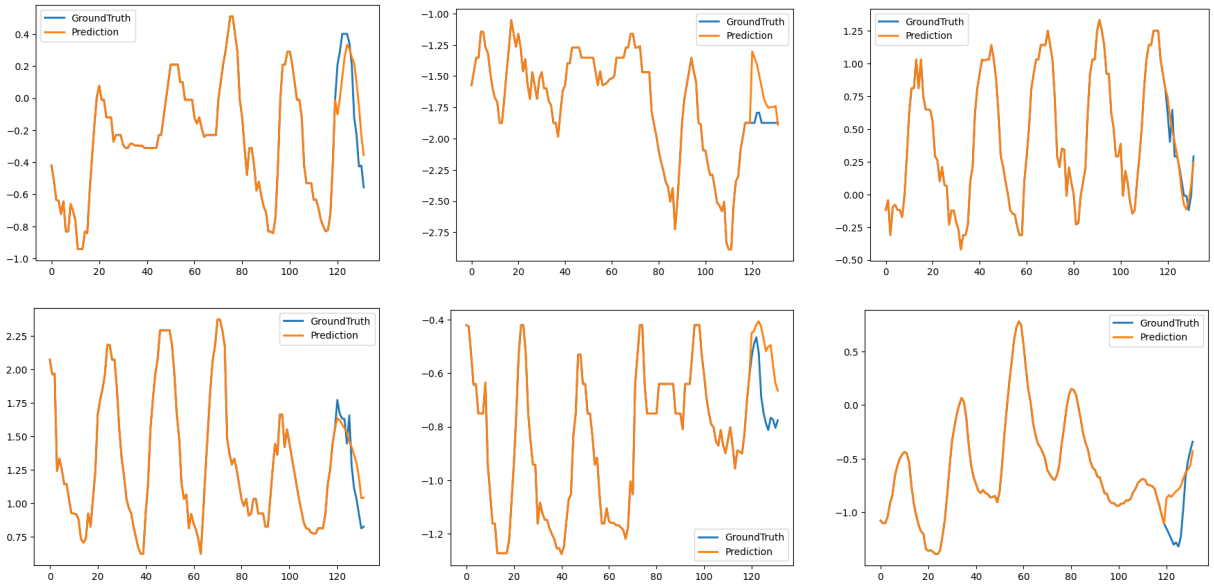

Figure 4.11: 12 hours prediction for the Autoformer architecture

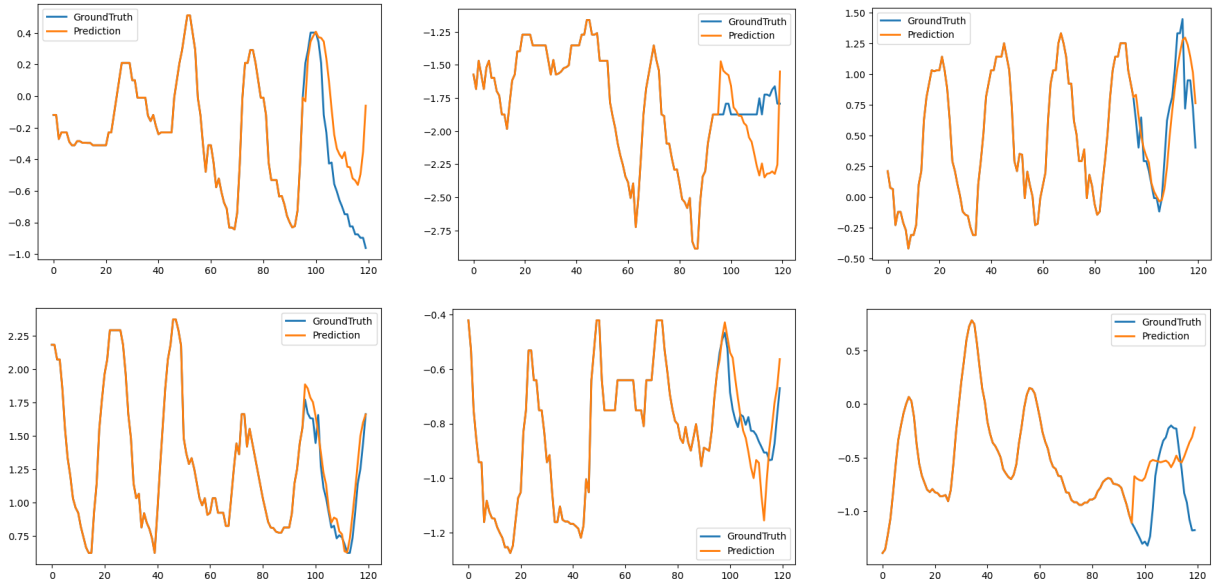

Figure 4.12: 24 hours prediction for the Autoformer architecture

4.3 Discussion

At the start we wanted to see if a transformer architecture can do better for short-term forcasting of temperatures than a LSTM. The results show that at small time frames between 6-12 hours the LSTM wins, and even for one day, while the Autoformer has a better MAE it is less stable then its competitor. Although because of the stability, we suspect that the Autoformer was not trained long enough. This is also because of the fact that transformers tend to need even more data then other neural network architectures.

At the beggining we stated that we wanted to see if we can have one model to rule them all. Well after some lenghty trainings of a lot of models we can say that, not for now, or at least not with this particular transformer architecture. Is there a better one. Well according to the same authors they actually built one called TimesNet[6] about a month ago, so we could try and see.

Chapter 5

Conclusion and future work

In conclusion our work presents a comparison between two different architectures based on neural networks, a LSTM and a transformer. We wanted to see if the transformer can have better predictions than the LSTM for short-term predicitons. Although our analysis is far from over, we provided a comprehensive and objective analysis between them, getting to the conclusion that no model can be the best solution for every dataset and that it depends on a multitude of factors, in our case the variable being the prediction lenght. We offer a SWOT analysis at figure 5.1 in which we ilustrate what we considered we did good, what we can do better or what are a couple of enhancements we could make to this report.

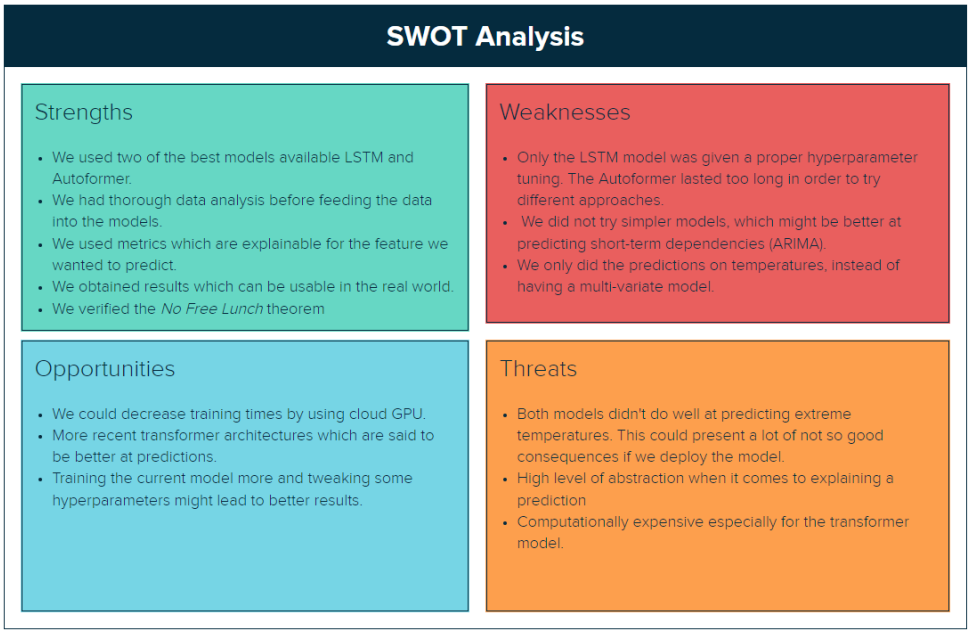

Figure 5.1: SWOT analysis of our report

Bibliography

[1] Hisham ElMoaqet, Mohammad Eid, Martin Glos, Mutaz Ryalat, and Thomas Penzel. Deep recurrent neural networks for automatic detection of sleep apnea from single channel respiration signals. *Sensors (Basel, Switzerland)*, 20, 09 2020.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[3] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. 1986.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[5] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[6] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis, 2023.

[7] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *CoRR*, abs/2106.13008, 2021.