Maximal Temperature forecasting under spatio-temporal interrelations using Machine Learning

Gonzalo Marco, María Eugenia Miranda, Pablo Rodriguez-Bocca, Gerardo Rubino

Maximal Temperature forecasting under spatio-temporal interrelations using Machine Learning *

Gonzalo Marco[1], María Eugenia Miranda[1], Pablo Rodríguez-Bocca[1], and Gerardo Rubino[2]

[1] Facultad de Ingeniería, Univ. de la República, Montevideo, Uruguay
{gonzalo.marco.mohotse, maria.eugenia.miranda, prbocca}@fing.edu.uy
[2] INRIA, Rennes, France gerardo.rubino@inria.fr

Abstract. In the forecasting of irregular time series using modern Machine Learning methods, some methodologies have become pretty popular, such as those implemented in the XGBoost tool. XGBoost has already shown to be efficient in many situations. However, in the case of a set of time series, for instance associated with a set of geographical points and exhibiting significant correlations, XGBoost needs an important amount of *features engineering* to be able to capture that supplementary and useful information. In these situations, the Graph Neural Networks (GNNs) family offers tools designed to take into account these spatial correlations automatically. In this paper, we consider the time series of the daily maximal temperatures collected at many meteorological stations in a wide area, and we try to forecast the maximal temperatures in the next few days at all those points. These are pretty irregular series, and we chose a particularly powerful algorithm belonging to the GNN class called Graph WaveNet for this task. The algorithm hasn't been applied before to this type of target. We explored its behavior with no need for any feature engineering, and we also compared it with that of XGBoost for which we should invest a significant effort to exploit the spatial aspects of data. We considered the forecasting up to 10 days, the meteorological range. Basically, Graph WaveNet behaves better for very short *prediction horizons* and becomes less accurate when the prediction horizon increases. XGBoost stays closer to the mean of the series when the horizon gets large (close to 10 days). The paper provides more details about the two tools' behavior.

Keywords: Irregular time series · Forecasting · XGBoost · Graph Neural Networks · Graph WaveNet.

* This work was supported by the "ClimateDL" project belonging to the *Climat AmSud* program, with program code 22-CLIMAT-02.

1 Introduction

Forecasting the future of a time series appears in uncountable situations and in all kinds of human activities [1]. In some of these cases, the forecasting task is hard, for instance when we know that the underlying phenomena is chaotic. In the specific area of weather forecast, we encounter these hard-to-forecast series, for instance, mean or maximal temperatures at specific places, at specific days or hours, at specific altitudes, etc. A particular case is the meteorological one, or short-range weather forecasting, where we want to forecast weather parameters (temperatures, precipitations, etc.) for the next days (meaning up to 10 days in the future). This is done by means of fine granularity physical models of the atmosphere and the oceans (huge systems of differential equations, also referred to as dynamical models), solved using extremely powerful supercomputers. This is called Numerical Weather Prediction (NWP); see [2] for a recent reference). Traditionally, time series forecasting is a statistical territory, where many techniques are available, and for long term forecasting they are the most used tools (see for instance [3]). Recently, Machine Learning has entered the field with an important number of methods, that have already proved to be efficient in several areas. Short-term weather prediction is however still a case pretty hard to attack, and dynamic models still dominate the operational forecasting services. ML techniques are starting to reach now the accuracy of NWP, at the cost of very large platforms with powerful hardware and the necessary huge amount of data. See [4] and [5] for two recent tools exhibiting very nice performances. In this paper we focus on the ML tools that can be used with standard hardware and software, needing fewer data than the models mentioned above, and we explore the capability of these approaches in approximating the target.

Let us first present more formally the problem addressed here, the forecast of the maximal temperature at a specific geographic point (a point in a region where there is a meteorological station taking measures), at a short horizon. More specifically, having chosen a unit of time (here, days), at some point in time t (seen as the *present*), we know the past of the series of maximal temperatures at the station plus possibly that of other supplementary variables, and we want to predict those maximal temperatures in the *close future*, say in the next 10 days. Now, instead of considering a single station, we want to make the prediction simultaneously for a set of N stations, with the same kind of data at each of them. An immediate observation is that the series at two different stations (for instance, geographically "close" to each other) will probably be strongly correlated and an efficient way of using this information should deal to more efficiency in the global forecast.

Formally, assume the stations ordered in some manner, that is indexed from 1 to N, and denote by $\mathbf{y}_t$ the vector of maximal temperatures (of all stations) at time t, $\mathbf{y}_t \in \mathbb{R}^N$. Let us denote by F the number of supplementary variables (precipitation, pressure, etc.), collectively called here the *features*, available at each of the N points, and by $\mathbf{X}_t$ the matrix $\left(\mathbf{X}_{s,f,t}\right)_{s,f}$ containing these values at time t, for all stations and features. That is, the scalar $\mathbf{X}_{s,f,t}$ is the value of feature f at station s at time t, and $\mathbf{X}_t \in \mathbb{R}^{N\times F}$. We then look for a function $f()$

(our *model*) able to predict the close future of the time series $\left(\mathbf{y}_t\right)$, using the T past values of the data (including the current one) and making the prediction for the next S values. We symbolically write

$$\mathbf{y}_{(t-T):t}, \mathbf{X}_{(t-T):t} \xrightarrow{f} \mathbf{y}_{(t+1):(t+S)},$$

where we have $\mathbf{y}_{(t-T):t} \in \mathbb{R}^{N \times T}$, $\mathbf{X}_{(t-T):t} \in \mathbb{R}^{N \times F \times T}$ and $\mathbf{y}_{(t+1):(t+S)} \in \mathbb{R}^{N \times S}$.

In the paper, we will use $T = 12$ (we will train the Machine Learning tools to make the predictions based on the last 12 days of historical data) and S will go from 1 to 10. We will use a different model for each value of S, that is, we will predict value $\mathbf{y}_{t+S}$ with a specific model $f_S()$ specifically trained for that forecast horizon.

The available data will be partitioned into the three classic sets according to the time index: $\mathcal{D}_{\text{train}}$ for training, $\mathcal{D}_{\text{val}}$ for validation and $\mathcal{D}_{\text{test}}$ for testing (see Section 5 for details).

To determine the quality of forecasting, we use the Root Mean Square Error (RMSE) formula over the testing dataset, given by Equation (1):

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{t \in \mathcal{D}_{\text{test}}} ||\mathbf{y}_t - \hat{\mathbf{y}}_t||^2}, \tag{1}$$

where $\mathbf{y}_t$ is the vector of ground truth (target) temperature values, and $\hat{\mathbf{y}}_t$ is the vector of forecasted temperature values with some model. RMSE is the standard method to measure the accuracy of temperature forecast models, and help us to directly compare our results with the previous ones. Moreover, it measures the error in the same units (here, in Celsius degrees) than the forecasting, which makes it easy to understand how the methods used behave.

The rest of the paper is structured as follows. In Section 2 we position this work in the existing research literature. Section 3 describes the two algorithms we decided to use for this exploration, in the Machine Learning area. One is Graph WaveNet [6], the other is XGBoost [7]. Section 4 presents the data we used in the paper. In Section 5 we describe some results illustrating the behavior of the two selected procedures. Section 6 contains the conclusions and some perspectives of this work.

2 Related work

There are many ways of attacking this type of problem, belonging to different families of techniques.

- The models used for short-range (for some authors, this is actually medium-range instead) weather forecasting (say, up to 10 days) are dynamical models, that is, differential equations solvers (also referred to as Numerical Weather Predictors (NWPs)). They are very performant until around 10 days in the

future [2]. The mathematical systems to solve are chaotic, meaning that progress (in number of days, for the prediction to be somehow useful) is extremely slow (chaotic behavior arrives exponentially fast). These systems are huge, and the solvers must run on very powerful hardware (in the range of the most powerful supercomputers available) to produce their output fast enough. In the considered area (meteorological predictions) these are the models used for predicting weather conditions in the next days.
- Other models are statistical (ARIMA, SARIMA, SARIMAX, etc. [1]). They exploit historical data to estimate how the future is correlated with the past, and for weather predictions they are in general used for long term predictions. They started to be developed early in last century, and have been successfully applied to uncountable problems in all possible areas.
- Others, more recent, are based in Machine Learning (ML) tools, such as the ones on which we will focus here. In many areas belonging to the general time series analysis, these techniques improve over the state of the art, using a variety of approaches. Weather prediction is a very particular field with specific difficulties, including the chaotic behavior of the corresponding dynamical systems, even if in some cases, using specific ML architectures such as Recurrent Networks or, in particular, Reservoir Computing [8], good results can be obtained for *simple* chaotic systems (Lorentz, etc.). This is the reason that makes interesting its exploration here.

In this paper, we focus then on standard ML tools, that is, on well-known techniques that have proven to be effective in many (other) problems, and we explore their capabilities in this type of prediction problems. In the general framework introduced in Section 1, we will only consider the case of $F = 1$, that is, a single feature, in our case the cumulated precipitations (see Section 4). But we will integrate the fact that we work on several spatial (geographic) points, where there are correlations (a priori, complex ones) between the time series elements associated with different stations (typically, the closer the stations, the higher the correlations).

The literature in the global weather or climate forecasting area using ML is huge, but specifically on meteorological predictions for the next days there are fewer publications. We mention here those that we found particularly interesting for our study, all very recent, all with pretty nice results, but none on the same problem we explored.

In [9] the author considers the prediction of the average temperature in a single geographical point. The prediction is done using as input the last 7 days, for the next day and globally for the next 10 days. There are 10 years of historical data and several other variables available (features). The paper compares 3 neural networks' architectures: a multi-level perceptron, the LSTM model and a combination between the latter and a CNN (Convolutional Neural Network). The best one was this last model, but the performances of the three tools were anyway close to each other.

In [10] the prediction concerns several meteorological variables in the very close future (up to 5 days). Many geographical points are considered (2048 sta-

tions). The data set is large (WeatherBench), with 40 years of registers. The input consists of the present observations, plus 6 and 12 hours before. The ML tool is a variant of a CNN called Resnet, using a network for each desired output, and also per unit of time in the future, or using the free-run idea where the prediction for the second day uses the precedent one, and so on. The paper also uses a physical or dynamical model for comparison purposes. The results are sometimes close to those of the latter.

In [11] there is another pretty recent work that, as the previous one, takes into account the spatial dependencies. The data consists in 5 years of registers for 30 points in the US and in Canada, for every hour and concerning 11 variables, plus a European data set with daily measures over 15 years at 19 stations. Both sets have data for many variables. The prediction scheme is similar to previous ones, and concerning the temperature, the data offers the average one. The selected ML architecture is a Transformer, but the author used other architecture with well-known (and good) performances, such as CNN, LSTM, etc.

In [12], the authors develop a new neural model to detect extreme temperatures by means of the prediction of the maximal one of the set of the next 7 days. The data consists only of previous maximal temperatures for a period of 58 years but concerning only three months (July to September), and a set of 651 stations. In the model, a decomposition procedure partitions the data taking into account the geographical side, then a specific Feed Forward neural network is used on each element (a Radial Basis Function Neural Network) and at the end, the outputs are combined to provide the global output of the model,

Before describing the algorithms used, let us mention two other papers that appeared at the end of last year, 2022, with a different goal than ours but related to it. Both concern large projects using powerful hardware and extremely rich data, with the goal of competing with NPW systems in short horizons (days). Both reach and even go beyond NPW, a clear breakthrough. In [4], the authors present the Pangu-Weather tool, strongly relying on deep architectures and showing better accuracy than the leaders of the NWP area. One month later [5] was uploaded, proposing Graphcast, that appears to be even more accurate than the former. Both offer several methodological improvements that we can't detail here. Graphcast belongs to the Graph Neural Networks (GNNs) family, as the Graph WaveNet technique we used in our paper. The GNN is combined with a particular multiscale internal mesh data representation. The tool is trained of a large European dataset with 4 measures per day in the period 1979..2018, with 227 variables at each of 1,038,240 geographical points.

3 Machine Learning tools

Machine Learning (ML) tools are starting to emerge in weather prediction, because some of their main properties (black box approaches, good scaling with data), in general in long-range predictions, and researchers start to look about their performances on medium to short-range ones.

Classical approaches in this area take care mainly of the temporal dependency of data. But it is clear that for this type of problems the spatial dependencies are important, and it is very hard to capture them in a white box manner. But ML is precisely performant in these contexts, that is, it is capable of capturing the effect of those correlations without trying to explain them.

In the sequel, we first focus on one well referenced tool called Graph WaveNet, where those spatial connections can take arbitrary forms (well represented by an arbitrary graph), that proved to be very efficient in many cases. We will compare it with XGBoost, a popular high performant technique with also a good record of good results in irregular time series forecasting problems, particularly sensitive to a good feature engineering pre-process. Let us introduce these two methods now.

3.1 Graph WaveNet

Graph WaveNet [6] is a deep architecture (many levels) commonly used in multivariate forecasting problems, which has the option to learn an adaptive dependency matrix through a node embedding technique, in order to capture the hidden spatial dependency in the data. It belongs to the Graph Neural Networks (GNNs) area.

The Graph WaveNet architecture consists of many stacked spatio-temporal layers connected to a single output layer. The input is transformed by a linear layer (a 2D convolution) and then passed to the stack. Each element of the stack has residual connections, and can also send data directly to the output, which simply uses $ReLu()$ neurons and linear transformations. Recall that the output is a scalar prediction per station, and per horizon. Figure 1 illustrates graphically the Graph WaveNet structure.

The first layer (called the "bottom" one) processes the short-term temporal data, and the last one (called the "top" layer) the long-term one. The spatial dependencies are processed inside each layer of the stack.

Inside each spatio-temporal layer we have a Gated Temporal Convolution module (G-TCN) and a Graph Convolution Network (GCN). We describe next both components.

Gated Temporal Convolution layer (G-TCN). A Temporal Convolution Network (TCN) is simply a dilated causal convolution followed by an activation function. Formally,

$$\mathbf{y} = \sigma(\mathbf{\Theta} \star \mathbf{X} + b), \tag{2}$$

where $\sigma()$ is an activation function, $\mathbf{X}$ is the set of input variables, $\star$ denotes the dilated causal convolution operation (we have $\mathbf{\Theta} \star \mathbf{X} = \sum_s \mathbf{X}(s)\mathbf{\Theta}(t - ds)$), with a hyper-parameter d, $\mathbf{\Theta} \in \mathbb{R}^K$ and b being the two parameters of the procedure. The G-TCN is made of two parallel TCN layers, TCN-a and TCN-b in Figure 1. The former uses a tanh() activation function, while the TCN-b uses a sigmoid one. Formally, the G-TCN implements the following equation

$$\mathbf{h} = \tanh(\mathbf{\Theta}_1 \star \mathbf{X} + b_1) \odot \sigma(\mathbf{\Theta}_2 \star \mathbf{X} + b_2). \tag{3}$$

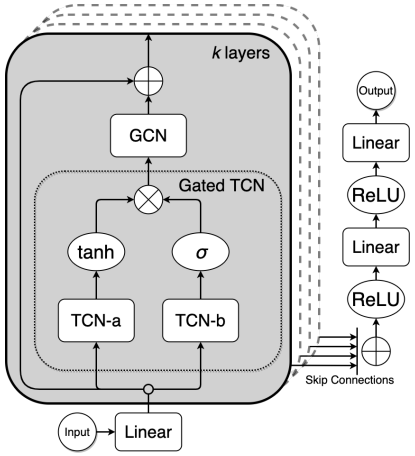


Fig. 1: The Graph WaveNet structure: the stack of spatial-temporal layers (we see the substructure of the first one, with a gray background) plus the output layer on the right side. After a linear transformation, the input attacks the spatio-temporal module of the first layer, who has its residual connection and also, its connection to the output layer. The source of the picture is [6].

Above, we have the input $\mathbf{X} \in \mathbb{R}^{N \times H \times T}$, the parameters $\mathbf{\Theta}_1$, $\mathbf{\Theta}_2$, b_1 and b_2, and $\odot$ represents the element-wise product. The parameter H belongs to the linear processing of the input.

Graph Convolution Network (GCN) The GCN layer models the spatial dependencies using an underlying directed graph G. This graph can be given externally, can be something received and modified during training by the architecture, or it can be totally learned. It has the task of improving each node's features, transforming those of its neighbors.

Denote by $\mathbf{A} \in \mathbb{R}^{N \times N}$ the adjacency matrix derived of G. Let $\mathbf{X} \in \mathbb{R}^{N \times H}$ be the input features, $\mathbf{Z} \in \mathbb{R}^{N \times H}$ the output signals, $\mathbf{W} \in \mathbb{R}^{F \times H}$ the layer's parameters, and $\mathbf{P}_f = \mathbf{A}/rowsum(\mathbf{A})$ and $\mathbf{P}_b = \mathbf{A}^\intercal/rowsum(\mathbf{A}^\intercal)$ the forward and the backward transition matrices. We can use these to fix the spatial relationships, and we will call this case *input adjacency*, where the output of the GCN layer is $\mathbf{Z} = \sum_{k=0}^{K} \left(\mathbf{P}_f^k \mathbf{X} \mathbf{W}_{k_1} + \mathbf{P}_b^k \mathbf{X} \mathbf{W}_{k_2} \right)$.

Also, we can prefer to completely learn the spatial relationship from scratch for our forecasting task; we will call this case *self-adaptive*; the output of the GCN layer is then $\mathbf{Z} = \sum_{k=0}^{K} \tilde{\mathbf{A}}_{adp}^k \mathbf{X} \mathbf{W}_{k_3}$, where $\tilde{\mathbf{A}}_{adp} = \text{SoftMax}\left(\text{ReLU}\left(\mathbf{E}_1 \mathbf{E}_2^\intercal\right)\right)$, and $\mathbf{E}_1, \mathbf{E}_2 \in \mathbb{R}^{N \times c}$ are learnable node embedding parameters.

Last, the general configuration called *input adjacency + adaptive* happens when we know a graph structure G that represents spatial dependencies, but we want to improve it and adapt it to the specific forecasting task. In this more

general case, the output of the GCN layer is computed as follows:

$$\mathbf{Z} = \sum_{k=0}^{K} \left(\mathbf{P}_f^k \mathbf{X}\mathbf{W}_{k_1} + \mathbf{P}_b^k \mathbf{X}\mathbf{W}_{k_2} + \tilde{\mathbf{A}}_{adp}^k \mathbf{X}\mathbf{W}_{k_3} \right). \tag{4}$$

The GCN output configuration of the stacked spatio-temporal layers (*input adjacency*, *self-adaptive*, or *input adjacency + adaptive*) is one of the most important hyper-parameters of the architecture, and we will discuss it for our problem in the results presented in Section 5.

3.2 XGBoost

The algorithm XGBoost (eXtreme Gradient Boosting [7]) implements several ideas, leading to an efficient procedure with a nice record of good performances. The *boosting* part is the usual method of combining several predictors to obtain a new one, in principle better than its components. XGBoost proceeds as follows: after building a first predictor, the global algorithm tries to predict its errors and to integrate this prediction into a new predictor, and then repeats the same procedure several times, improving the quality of the resulting model (here appears the *gradient* component of the name, in these sequence of iterations). This "gradient boosting" idea is used in XGBoost starting with decision trees as the initial components of the architecture, the ones that will be combined into global predictors during the training phase.

Face to our data having not only temporal dependencies but also spatial ones, XGBoost needs an important effort from the user in providing information about these dependencies, what is called *feature engineering*. This is necessary here in order to be able to compare this technique with the Graph Neural Network one. The data available includes many supplementary variables associated with each station, that is, with each geographical point. An important one is the volume of precipitations. For this purpose, we enriched the input to our model with features associated with each epoch, containing information about the precipitations, formally obtained by comparing the volume of the last m months with the historical data. Typically, m is 1, 3, 6, 9, 12, 24, and the obtained index is named SPI-m (SPI stands for Standardized Precipitation Index [13]). To these features, we add the daily volume of precipitations. Another example of supplementary feature is the season of the year, and also the year itself, etc. Concerning spatial data, we computed the geodesic distance between the stations, and we associated with each station the resulting 3 closest stations, assuming that they were the more influential ones. Finally, the features added to each station were the temperature and precipitations parameters corresponding to the oldest date used for training (that is, $t - T$).

The feature engineering process is often done incrementally, by adding them little by little and testing if they contribute to the performance. XGBoost also allows weighting the features according to their estimated (or known, etc.) relative impact on the global model, that can be useful to avoid putting very different features at the same level.

4 Available data

The data used in this paper come from the ClimateDL project, a joint cooperation between Argentina, Chile, Uruguay and France, belonging to the Climat AmSud program. Our database includes measures from the national meteorological services in Argentina[3], Chile[4], Uruguay[5], Paraguay[6] and Brazil[7]. As stated before, we only used the maximal temperature of each day, and the volume of precipitation measured in mm, also on a daily basis.

To have an idea of the global volume of the data, some figures: we had measures from $N = 137$ stations in different countries (Argentina, Uruguay, Brazil, Paraguay and Chile, see Fig. 2); they were collected in the period going from Jan 1 1977 to Dec 31 2018 (that makes 42 years or 15340 days); the volume of missing data was of about 4.8% of the total.

5 Results

The data set was decomposed into three sets $(60\%, 20\%, 20\%)$, as usual, with the following characteristics:

- $\mathcal{D}_{\text{train}}$: for training, all the records from Jan 1 1997 to Dec 10 2002;
- $\mathcal{D}_{\text{val}}$: for validation, the data from Jan 1 2003 to Dec 31 2010; and
- $\mathcal{D}_{\text{test}}$: for testing, from Jan 1 2011 to Dec 31 2018.

We used these datasets for optimizing the hyper-parameters first, for both models (XGBoost and Graph WaveNet) and then, for training and evaluating our final models. All these tasks were executed on the ClusterUY[8] platform.

5.1 Graph WaveNet hyper-parameter definition

Graph WaveNet has an important property: it is able to manage different configurations of the GCN layers (see Section 3.1). These configurations specify how the architecture learns the spatial interrelations between the geographical points. As that, they are one of the major hyper-parameters of the tool. We mentioned 3 configurations in Section 3.1): *input adjacency*, *self-adaptive*, and *input adjacency + adaptive*. The first and third options require an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. As stated in the original paper [6], we built the weighted adjacency matrix by computing the geodesic distances between the

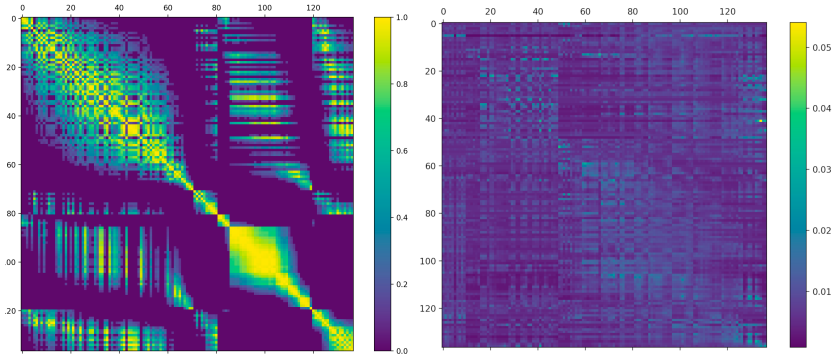
[3] Servicio Meteorológico Nacional de Argentina, `http://www.smn.gob.ar/`
[4] Centro de Investigaciones sobre Clima y Resiliencia, `https://www.cr2.cl/`
[5] Instituto Uruguayo de Meteorología, `https://www.inumet.gub.uy/`
[6] Centro Meteorológico Nacional, `https://www.meteorologia.gov.py/`
[7] Instituto Nacional de Meteorología, `https://portal.inmet.gov.br/`; the centers of the 5 countries were accessed on June 2023
[8] ClusterUY: Centro Nacional de Supercomputación (National Supercomputing Center), Uruguay, `https://www.cluster.uy/` (Accessed: 2022-10-06).

Fig. 2: Location of the 137 weather stations in southern South America (different color per country).

stations and applying a thresholded Gaussian Kernel [14]. If $\text{dist}(s_i, s_j) \leq \kappa$ then $a_{ij} = \exp\big(\text{dist}(s_i, s_j)^2/\sigma^2\big)$, otherwise $a_{ij} = 0$. The effect of the threshold is to get some sparsity in the matrix (see the result in Figure 3a for our N weather stations).

We tried the three possible configurations, training the model with $\mathcal{D}_{\text{train}}$ and evaluating the results with the RMSE of $\mathcal{D}_{\text{val}}$. The results are shown in Table 1. As we can see, the best choice is *input adjacency* + *adaptive*. This is what appears in the final result. In this case, the auto-adaptive matrix that learns the spatial connections hidden in our problem, $\tilde{\mathbf{A}}_{adp}$, can be seen in Figure 3b.

We also looked for good values of other hyper-parameters using the same procedure (training with $\mathcal{D}_{\text{train}}$ and evaluating with $\mathcal{D}_{\text{val}}$). This allowed for some fine-tuning of the following hyper-parameters: `learning rate`, `epochs`, `weight decay`, and `dropout`. The best configurations used in our final model, obtained after about 5 days of computations, were: `learning rate = 0.005, epochs = 168, = 0.0001` and `dropout = 0.3`.



(a) *input adjacency.* (b) *input adjacency + adaptive.*

Fig. 3: Spatial relationships obtained by Graph WaveNet in two configurations: (a) *input adjacency* configuration, (b) *input adjacency + adaptive* configuration; we show matrix $\tilde{\mathbf{A}}_{adp}$ learned at the end of the training process.

Configuration	**Matrices used**	**RMSE in $\mathcal{D}_{\mathbf{val}}$**
input adjacency	$[\mathbf{P}_f, \mathbf{P}_b]$	2.6713°C
self-adaptive	$[\tilde{\mathbf{A}}_{adp}\ ]$	2.6160°C
input adjacency + adaptive	$[\mathbf{P}_f, \mathbf{P}_b, \tilde{\mathbf{A}}_{adp}]$	2.6159°C

Table 1: Error in our final model (Graph WaveNet) with the three different adjacency matrices

5.2 XGBoost hyper-parameter definition

In XGBoost, we applied a similar procedure with, as we have seen, a much larger set of hyper-parameters. In this case, as the number of possible combinations is huge, we helped the search procedure with the Optuna[9] tool. After 14.400 trials and 108 hours of computation, we obtained the following hyper-parameters configuration: `n_estimators` = 125, `subsample` = 1.0, `max_depth` = 9, `eta` = 0.07, `gamma` = 0.26, `alpha` = 0.54, `lambda` = 0.93, `colsample_bytree` = 1.0, `min_child_weight` = 0.16, `grow_policy` = "lossguide".

5.3 Comparisons

We compared the performances of XGBoost and Graph WaveNet, together with a Baseline predictor that simply uses the last observed value to predict the next one (at $t+1$). This Baseline is simply used as a reference.

Let us look first at some prediction examples. In Figure 4 we observe future temperatures and their forecast, at two geographical points: the city of Artigas, in

[9] Optuna - A hyperparameter optimization framework. `https://optuna.org/`. (Accessed: 2022-10-06)

the North of Uruguay (lon: -56.51, lat: -30.38), and that of Encruzilhada-do-Sul, in the South of Brazil (lon: -52.51, lat: -30.53). The two stations were selected at similar latitudes. We can see that for $S = 1$ the forecasting works very well in the three models, with a slightly better value for Graph WaveNet. However, when we increase S, the prediction horizon, XGBoost has a more accurate forecast.

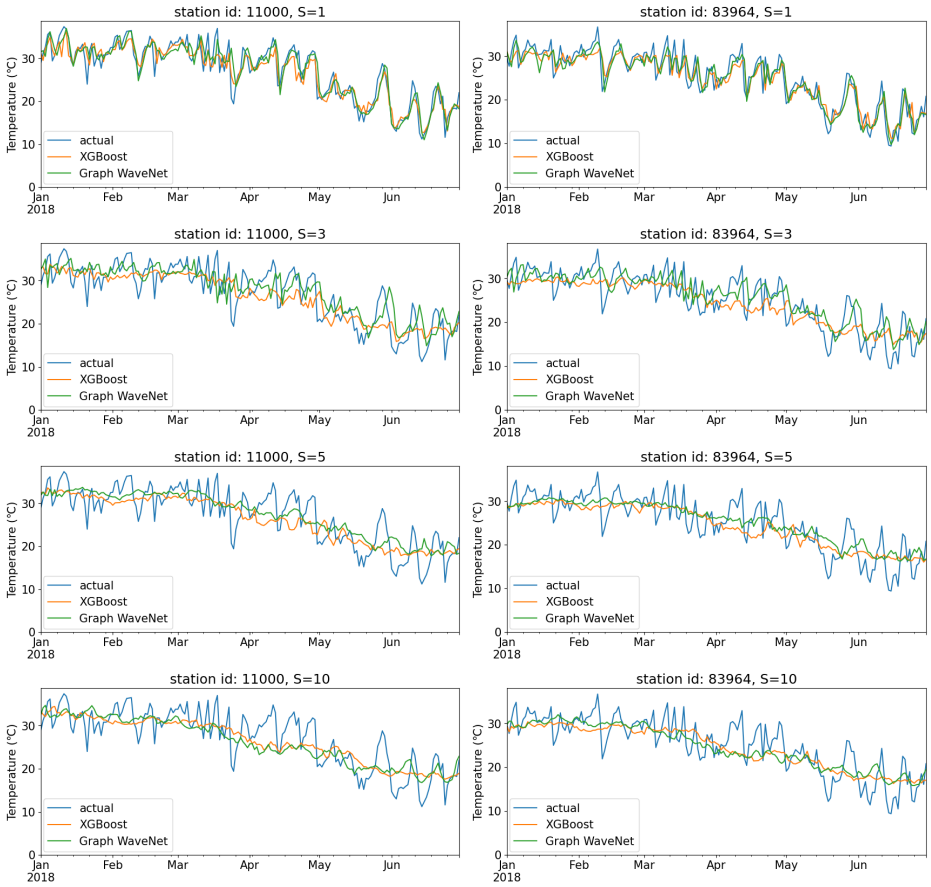


Fig. 4: Forecasting evaluation of Graph WaveNet and XGBoost models for the four studied forecast horizons, $S = 1, 3, 5$ and 10, on two typical stations: (left) 11000 Artigas, Uruguay (lon: -56.51, lat: -30.38), (right) 83964 Encruzilhada-do-Sul, Brazil (lon: -52.51, lat: -30.53). We removed the Baseline from the graphics for a cleaned view.

Table 2 summarizes the main results. It shows that Graph WaveNet is the best predictor for the next day, but XGBoost behaves better when we increase

the prediction horizon S. Both performs better than the Baseline in all the considered horizons.

Tool	S = 1	S = 3	S = 5	S = 10
XGBoost	2.93 °C	3.69 °C	3.84 °C	4.00 °C
Graph WaveNet	2.54 °C	3.82 °C	4.30 °C	4.06 °C
Baseline	3.51 °C	5.20 °C	5.25 °C	5.28 °C

Table 2: Performance comparison between *Graph WaveNet*, *XGBoost* and *Baseline* as a function of the number of days in the future; the table shows the RMSE values.

6 Conclusions

In this paper, we explore the behavior of two well known Machine Learning models in forecasting time series, in an area where the time series forecasting is known to be particularly hard. We selected the case of weather forecast in a short prediction horizon, and specifically predicting the maximal temperature in the next days, up to a limit of 10 days.

We added another factor to the work, the fact that the forecasting activity is to be done in a large geographic area, where we have the history of the measures in many geographic points (called stations), and the fact that at each point we know not only the mean temperature but also several other data. We expect that stations that are geographically close will exhibit correlations in their associated data (in general), etc. We selected two methods, one that behaves very well in many other areas (XGBoost) and another one belonging to the Graph Neural Network family (Graph WaveNet), meaning that it can integrate those spatial correlations or more generally dependencies between the measuring points. The former needs an intense feature engineering effort to compete with the latter, mainly for also exploiting the spatial dimension.

The results show first that the two forecasting systems behave similarly, in spite of the strong effort needed to prepare XGBoost to the job. They also show that Graph WaveNet starts in general closer to the target when the prediction horizon is small, and this ordering becomes the opposite very quickly, as the prediction horizon is increased.

This work can be extended in several ways. The use of the two tools can be improved in several ways, in particular, the hyper-parameters' configuration for Graph WaveNet.

Other tools also deserve exploration. Last, we can explore the same kind of problem with other metrics such as precipitations, or with mean or minimal temperatures instead of maximal ones.

References

1. C. Chatfield, *Time-Series Forecasting*. Cambridge: Chapman and Hall/CRC, 2001.

2. P. Bauer, A. Thorpe, and G. Brunet, "The quiet revolution of numerical weather prediction," *Nature*, vol. 525, no. 1, pp. 15–29, 2015.
3. C. W. J. Granger and R. Joyeux, "An introduction to long-memory time series models and fractional differencing," *Journal of Time Series Analysis*, vol. 1, no. 1, pp. 15–29, 1980. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9892.1980.tb00297.x
4. K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, and Q. Tian, "Pangu-weather: A 3d high-resolution model for fast and accurate global weather forecast," December 2022.
5. R. Lam, A. Sánchez-González, M. Willson, P. Wirnsberger, M. Fortunato, A. Pritzel, S. Ravuri, T. Ewalds, F. Alet, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, J. Stott, O. Vinyals, S. Mohamed, and P. Battaglia, "Graphcast: Learning skillful medium-range global weather forecasting," arXiv:2212.12794, November 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2212.12794
6. Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph Wavenet for Deep Spatial-Temporal Graph Modeling," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, ser. IJCAI'19. AAAI Press, 2019, p. 1907–1913.
7. T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785
8. S. Basterrech and G. Rubino, "Evolutionary Echo State Network: A neuroevolutionary framework for time series prediction," *Applied Soft Computing*, vol. 144, 2023.
9. D. Roy, "Forecasting the air temperature at a weather station using deep neural networks," *Procedia Computer Science*, vol. 178, pp. 38–46, 01 2020.
10. N. T. Stephan Rasp, "Data-Driven Medium-Range Weather Prediction With a Resnet Pretrained on Climate Simulations: A New Model for WeatherBench," *Journal of Advances in Modeling Earth Systems*, vol. 13, no. 2, feb 2021. [Online]. Available: https://doi.org/10.1029%2F2020ms002405
11. O. Bilgin, P. Maka, T. Vergutz, and S. Mehrkanoon, "TENT: tensorized encoder transformer for temperature forecasting," *CoRR*, vol. abs/2106.14742, 2021. [Online]. Available: https://arxiv.org/abs/2106.14742
12. M.-L. Lin, C. W. Tsai, and C.-K. Chen, "Daily maximum temperature forecasting in changing climate using a hybrid of multi-dimensional complementary ensemble empirical mode decomposition and radial basis function neural network," *Journal of Hydrology: Regional Studies*, vol. 38, p. 100923, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221458182100152X
13. T. B. McKee, N. J. Doesken, and J. Kleist, "Drought monitoring with multiple time scales, 9th conference, applied climatology," in *Conference on applied Climatology, Applied climatology, 9th Conference, Applied climatology*. The Society;, 1995, pp. 233–236. [Online]. Available: https://www.tib.eu/de/suchen/id/BLCP%3ACN008169111
14. D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "Signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular data domains," *CoRR*, vol. abs/1211.0053, 2012. [Online]. Available: http://arxiv.org/abs/1211.0053