CONTEXT-AWARE FORECASTING FOR MULTIVARIATE STATIONARY TIME-SERIES

Anonymous authors
Paper under double-blind review

ABSTRACT

The domain of time-series forecasting has been extensively studied because it is of fundamental importance in many real-life applications. Weather prediction, traffic flow forecasting or sales are compelling examples of sequential phenomena. Predictive models generally make use of the relations between past and future values. However, in the case of stationary time-series, observed values also drastically depend on a number of exogenous features that can be used to improve forecasting quality. In this work, we propose a change of paradigm which consists in learning such features in embeddings vectors within recurrent neural networks. We apply our framework to forecast smart cards tap-in logs in the Parisian subway network. Results show that context-embedded models perform quantitatively better in one-step ahead and multi-step ahead forecasting.

1 INTRODUCTION

Classical statistical forecasting methods rely on the existence of temporal correlation between past and future values. In particular, the auto-regressive component of ARIMA estimators (Box & Jenkins (1968)) models the relation between past and future as a linear regression. In the deep learning paradigm, Recurrent Neural Networks have long been used to tackle sequential problems. Increasingly complex models such as ConvLSTM (Shi et al. (2015)) or Graph Neural Networks (Wang et al. (2018)) are developed to model multivariate phenomena and allow a precise modeling of the temporal dynamics.

However, exogenous factors can greatly influence the observed values and are not taken into account by the mentioned models. For example, the type of road can drastically change traffic flow predictions, the period of the year will determine the values of sales time-series, and so on. In this work, we refer to these features as contextual information, or *context*. Such context is naturally used when dealing with stationary time-series to construct baselines based on the average of past values *given a context*. NARX models and their neural networks variations also make use of context by inputting it jointly with previous values of the forecast variable (Xie et al. (2009)).

Similar to how Graph NN learn relations between nodes, we propose for multivariate stationary time-series to *learn context* within a recurrent architecture and we introduce context-embedded RNN. For each contextual feature, we concatenate to the observed value an embedding that is to be learned jointly with the weights of the network. We do not deal with the case of continuous features but these could be transformed into categories. We tested our framework on public transportation tap-in logs one-step ahead and multi-step ahead forecasting, where we consider spatial context in the form of subway stations and temporal context through day of the week and time of the day.

To the best of our knowledge, there exists no good-quality public dataset containing subway logs at a satisfying granularity. We realized experiments on data provided by Ile-de-France Mobilités[1] (Parisian region public transportation agency) but we expect that the fast development of data collection in this domain will entail the availability of public datasets in a near future. On the other hand, all of the source code used to realize the experiments is available on `https://github.com/XXXX`.

Results of the experiments show that contextual models consistently outperform other recurrent models as well as the historical average baseline which is especially strong in the case of stationary

[1] https://www.iledefrance-mobilites.fr/

time-series. Contextual models perform particularly well for long-term forecasting. In summary, in this paper we propose a new paradigm for learning contextual information within RNNs, which quantitatively improves forecasting performance by allowing a fine-grained modeling of local dynamics.

The remainder of this paper is organized as follows: background in time-series forecasting and use of context is presented in Section 2; proposed models are introduced in Section 3 and are tested in prediction experiments in Section 4.

2 RELATED WORK

Time-series forecasting When it comes to time-series forecasting, the classical methods rely on ARMA models (Box & Jenkins (1968)) and their variants ARIMA for non-stationary series or SARIMA in the case of seasonality. However, RNNs have long been used for this task (Connor et al. (1994)) and perform well on a variety of applications (Zheng et al. (2017)). They are now employed to model more complex data. For instance, spatio-temporal data, which are similar to the application studied in this work, can be dealt with using a combination of CNN and RNN as in Shi et al. (2015). More generally, it is viewed as a graph problem in many works (Wang et al. (2018); Li et al. (2017); Cui et al. (2018); Ziat et al. (2017)). In particular, applied to traffic forecasting, Cui et al. (2018) learn weighted convolutional features representing the relations between each node of the graph. These features are then processed by a LSTM. While we could deal with the use case of transportation logs forecasting with such Graph NN, we choose to develop a more general framework where we learn peculiarities of each location instead of the relations between them.

Contextual information Jointly with complex architectures, contextual features can be used to improve forecasting performance. In an early work, Van Der Voort et al. (1996) develop the KARIMA algorithm. It uses a Kohonen neural network to cluster data based on present and past observations, but also time-step and day of the week. Then an ARIMA is used to predict the next value. More recently Xu et al. (2018) and Ding et al. (2016) use additional temporal features in LSTM and gradient boosting decision trees respectively. In general, predictive models with exogenous features belong to the class of NARX models such as Guzman et al. (2017) which forecast groundwater level based on precipitation or Koschwitz et al. (2018) where building heat load depends on many features. A different method is adopted by Liu et al. (2016) in the prediction of the next location problem. They replace the weight matrix multiplied by the input of a RNN by transition matrices representing spatial and temporal information. In this work, we choose to let the neural network learn its representation of the contextual features.

Public transportation data We apply our models on public transportation data, a domain which has not been as extensively studied as traffic forecasting because of the late apparition of data. Yu et al. (2010) combine SVM and Kalman filters to predict bus arrival times while Ceapa et al. (2012) only consider historical average for tap-in and tap-out forecasting. Closer to our work, Toqué et al. (2017) use LSTM networks for tap-in data in the Parisian area. However, they do not use context in the proposed models, whether spatial context because they study a small business zone, or temporal context.

3 MODELS

3.1 NOTATIONS

We describe notations for the considered transportation problem but the developed ideas can be extended to other context-dependent forecasting problems. A particularity of the data is the discontinuity caused by the closure of the subway network every night (as mentioned in Ceapa et al. (2012); Roos et al. (2016)). Therefore the observations for each day form a multivariate time-series containing the number of passengers entering the transportation network. Data is processed in the form of a 3D tensor $\mathbf{X} \in \mathbb{R}^{N \times S \times T}$, with N the number of days, S the number subway stations and T the number of time-steps. In particular, for a station s, $\boldsymbol{X}^s = \mathbf{X}_{:,s,:} \in \mathbb{R}^{N \times T}$ contains all the values for a specific location. We also denote $\boldsymbol{x}^s = \mathbf{X}_{d,s,:} \in \mathbb{R}^T$ the vector of values for a day d and station s and $\boldsymbol{x}_t = \mathbf{X}_{d,:,t} \in \mathbb{R}^S$ the values for a day d at time t.

In the recurrent models, the hidden state at time t of size h will be noted $\boldsymbol{h}_t \in \mathbb{R}^h$, or $\boldsymbol{h}_t^s$ when it represents a single location s. We will also introduce embeddings vectors for spatial location $\boldsymbol{z}^s$, day of the week $\boldsymbol{z}^d$ and time-step $\boldsymbol{z}^t$ whose sizes are respectively λ_s, λ_d and λ_t.

3.2 RECURRENT MODELS

Recurrent neural networks are a natural choice for forecasting time-series as they encode a latent state for each time step which is used to predict the next value. These architectures can model the dynamics of time-series and have a memory allowing the use of several observations in the past. In particular, they may be able to adapt themselves to anomalous behaviors, making them more robust.

We propose three recurrent architectures, introducing three different ways of dealing with spatial context. Two of them model it implicitly while the third one explicitly learns it. Each architecture is composed of a recurrent encoder E transforming the observations into hidden latent states. These states are then decoded into predictions using a linear layer D. Each of the models can then be completed with temporal context.

Univariate RNN First of all, we consider each station separately. That is, we explicitly train S distinct RNNs over as many matrices of samples $\boldsymbol{X}^s \in \mathbb{R}^{N \times T}$. In this case the input dimension of each RNN is 1, i.e. we compute $p(x_{t+1}^s | x_t^s, ..., x_0^s)$. The underlying assumption is that every station has a different dynamics and therefore requires a singular RNN.

At time step t, for a station s the observation $x_t^s \in \mathbb{R}$ and hidden state $\boldsymbol{h}_t^s \in \mathbb{R}^h$ are encoded via a singular encoder E^s into a hidden state representing only this station, then decoded into a single-valued prediction $\hat{x}_{t+1}^s$. The process is described in Equation 1 and Figure 1.

$$\forall (s,t) \in \{1,..,S\} \times \{1,..,T-1\}, \quad \hat{x}_{t+1}^s = D^s(\boldsymbol{h}_{t+1}^s) = D^s(E^s(x_t^s, \boldsymbol{h}_t^s)) \tag{1}$$

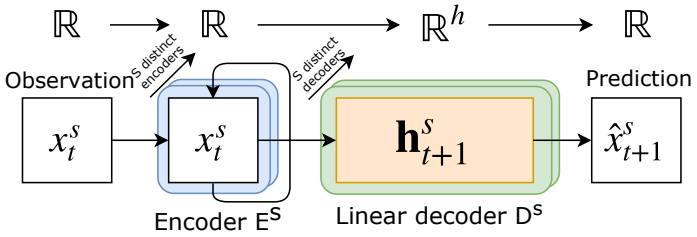


Figure 1: Computing one station's s predictions using the univariate architecture. Vector $\boldsymbol{x}^s$ of observations is processed by a dedicated recurrent encoder E^s which computes a hidden state $\boldsymbol{h}_{t+1}^s$ for each t which is decoded into the prediction by D^s.

Multivariate RNN In this model we consider that each sample represents a single day over the whole network and is a multi-dimensional series $\mathbf{X}_d \in \mathbb{R}^{S \times T}$. This representation assumes a correlation between the values of the different stations at each time t. In this setting we compute $p(\boldsymbol{x}_{t+1} | \boldsymbol{x}_t, ..., \boldsymbol{x}_0)$. This is similar to spatio-temporal models, but here the relations are not specified and the network must discover them during training.

At time t the vector sample $\boldsymbol{x}_t \in \mathbb{R}^S$ represents the *observed state* of the subway network, which is combined with the current hidden state $\boldsymbol{h}_t \in \mathbb{R}^h$ by the recurrent encoder to compute the next hidden state. During this stage, the recurrent encoder E has used several layers to combine past and current information into a synthetic state which is decoded back to S predictions by D (see Equation 2 and Figure 2). At the end of the day, this architecture captures the dynamics and the correlation of the entire network. Spatial context is not explicitly specified but included in the weights of the network.

$$\forall t \in \{1,..,T-1\}, \quad \hat{\boldsymbol{x}}_{t+1} = D(\boldsymbol{h}_{t+1}) = D(E(\boldsymbol{x}_t, \boldsymbol{h}_t)) \tag{2}$$

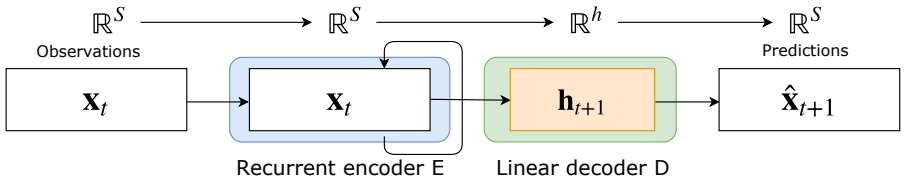


Figure 2: Computing predictions using the multivariate architecture. Vector $\boldsymbol{x}$ of observations is processed by a recurrent encoder which computes a hidden state $\boldsymbol{h}_{t+1}$ for each t. This state, which represents all the subway stations is finally decoded into S predictions.

This second model offers a natural way to deal with multivariate series. However, because of the large number of relations to learn between stations compared to the scarcity of samples, it may face overfitting and perform poorly.

Spatial RNN Finally, we propose a hybrid architecture which mixes the univariate and multivariate ones. As with the Univariate RNN we consider $N * S$ samples $\boldsymbol{x}^s \in \mathbb{R}^T$ that are encoded into a singular hidden state. However, there is a single couple (E, D) shared across all the stations - as in the Multivariate RNN - that allows to take into account the correlations between the stations and greatly reduces the number of weights.

This time, spatial context is explicitly learned in the form of a matrix of spatial embeddings $\boldsymbol{Z}^S \in \mathbb{R}^{S \times \lambda_s}$, hence the name *Spatial RNN*. For each station s, the corresponding embedding $\boldsymbol{z}^s$ is concatenated to the observation as in Equation 3 where c is the concatenation operation.

At time step t, for a station s, the observation $x_t^s \in \mathbb{R}$ is concatenated to the embedding $\boldsymbol{z}^s \in \mathbb{R}^{\lambda_s}$. The resulting vector and hidden state $\boldsymbol{h}_t^s \in \mathbb{R}^h$ are encoded via the common encoder E into a hidden state representing only this station. This state is then decoded into a single-valued prediction $\hat{x}_{t+1}^s$. See Figure 3 for an illustration.

$$\forall (s,t) \in \{1,..,S\} \times \{1,..,T-1\}, \quad \hat{x}_{t+1}^s = D(\boldsymbol{h}_{t+1}^s) = D(E(c(x_t^s, \boldsymbol{z}^s), \boldsymbol{h}_t^s)) \tag{3}$$

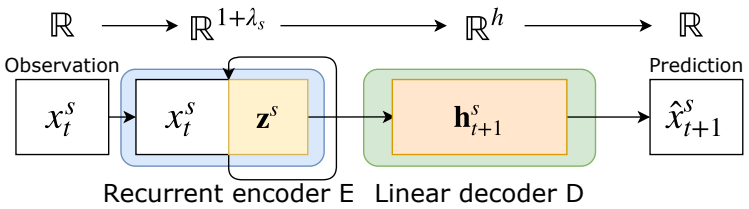


Figure 3: Computing one station's s predictions using the spatial architecture. Observations $\boldsymbol{x}^s \in \mathbb{R}^T$ are concatenated with a vector of embeddings $\boldsymbol{z}^s \in \mathbb{R}^{\lambda_s}$ and then processed by a recurrent encoder which computes a hidden state $\boldsymbol{h}_{t+1}^s \in \mathbb{R}^h$ for each t. This state is then decoded into a single prediction.

In addition to directly learning spatial context, this architecture offers the possibility to scale to a network of thousands or tens of thousands of stations because the number of recurrent weights is fixed. More generally, learning embeddings greatly helps to reduce dimensionality when dealing with a large number of contextual cases, compared to NARX models.

3.3 Temporal context

We proposed three different ways to deal with spatial context, one of them being to learn it. A promising way to improve performance is to introduce temporal context in the models. We consider two distinct time scales for temporal context, that are the days of the week and the time of the day. Indeed, the number of logs during one day at a specific time step is expected to be the same from one week to another.

We wish to see if the model is able to learn and discover meaningful representations of such temporal entities. Therefore, for each recurrent architectures we add the possibility to concatenate temporal embeddings to the observations. It is noteworthy that the temporal embeddings are *shared across every networks* i.e. there is one set of embeddings for the entire Univariate architecture, and not one different set per station.

Similarly to the way we dealt with spatial context, we could design multivariate and univariate architectures for days and time-steps. However we lack data to learn such models and the overfitting risk would be especially high for the day of the week scale.

Day embeddings We first introduce embeddings corresponding to the day of the week, via a matrix $(\boldsymbol{z}^d)_{d=\{1,..,7\}} \in \mathbb{R}^{7\times\lambda_d}$ containing 7 different embeddings.

Because we focus on fully-contextual models we only present in Equation 4 the prediction in the Spatial case, but temporal embeddings can be used for the other architectures as well.

$$\forall(d,s,t) \in \{1,..,7\}\times\{1,..,S\}\times\{1,..,T-1\}, \quad \hat{x}^s_{d,t+1} = D(\boldsymbol{h}^s_{t+1}) = D(E(c(x^s_{d,t}, \boldsymbol{z}^d, \boldsymbol{z}^s), \boldsymbol{h}_t)) \tag{4}$$

Time-step embeddings Similarly, the number of logs is very dependent on the time of the day, with notable morning and evening peak hours separated by off-peak time. Therefore we learn a matrix of embeddings $(\boldsymbol{z}^t)_{t=\{1,..,T-1\}} \in \mathbb{R}^{T-1\times\lambda_t}$. Prediction in the Spatial case is presented in Equation 5.

$$\forall(s,t) \in \{1,..,S\}\times\{1,..,T-1\}, \quad \hat{x}^s_{t+1} = D(\boldsymbol{h}^s_{t+1}) = D(E(c(x^s_t, \boldsymbol{z}^t, \boldsymbol{z}^s), \boldsymbol{h}_t)) \tag{5}$$

These embeddings can be learned using each of the architectures presented before and the two types of temporal embeddings can obviously be combined. An illustration for the Spatial model with day and time embeddings is presented in Figure 4.

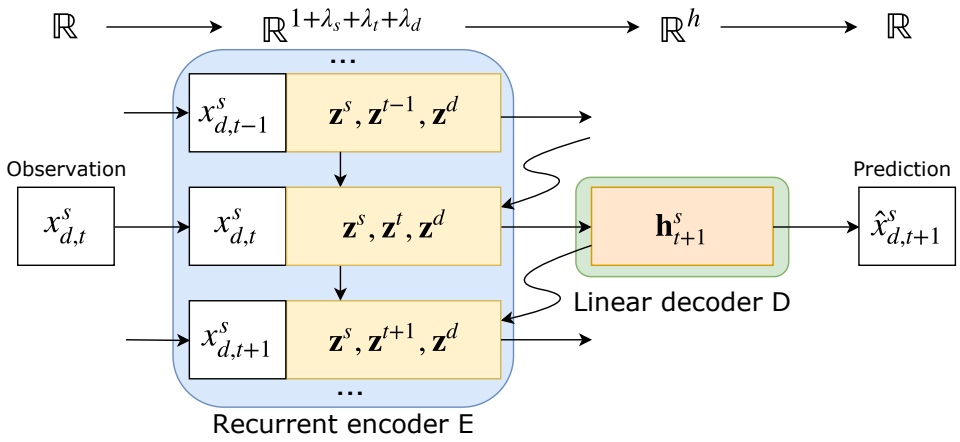


Figure 4: Computing predictions for a particular station s using the spatial architecture with temporal context. Given a day d, at each step t, the observed value x^s_t is concatenated with three embeddings representing the station, the day and the time, respectively $\boldsymbol{z}^s \in \mathbb{R}^{\lambda_s}$, $\boldsymbol{z}^d \in \mathbb{R}^{\lambda_d}$ and $\boldsymbol{z}^t \in \mathbb{R}^{\lambda_t}$. The obtained vector is processed by a recurrent encoder E (common to all stations) to compute a hidden state $\boldsymbol{h}^s_{t+1}$. Finally this vector is decoded into a single prediction $\hat{x}^s_{t+1}$

4 Experiments

4.1 Experimental settings

We train our models on a data set provided by Ile-de-France Mobilites (the transport agency of the Parisian region). It contains 256,028,548 logs $(user, station, time)$ between October and Decem-

ber 2015 across 300 subway stations. We aggregate the logs by windows of 15 minutes, in order to reduce inherent noise in the data and allow tractable computation.

From the data set we remove 3 stations which were undergoing planned works during the period. We also pull out 15 days with disturbed traffic pattern and that can be considered as *anomalies*.Finally we have $S = 297$ stations and $N = 77$ days. Those days are split into 70% train and 30% test samples. Splits are stratified with regards to the day of week, meaning that we try to have as many Sundays in train and test splits. In addition, 15% of the train split is kept for validation. In the end, there are 45 days in train split, 8 in validation and 24 in test.

Scaling The transportation network is comprised of few hubs and many less crowded stations. The existence of utterly different scales complicates the prediction problem, especially in the multivariate setting. To tackle this problem we rescale the data set, considering each station separately. Note that this also simplifies gradient computation. In more details we apply a two-step procedure:

- First, for a station s we calculate the 99.9th percentile and replace all superior or equal values by this bound. This removes the local outliers we missed when skipping some days.
- Then the values of s are scaled between -1 and 1 by min-max scaling. Treating each station one bye one prevents more important stations to squeeze the values of minor ones.

For these two steps, the percentile and the scaling values are computed on the train set and then applied on the other sets.

In this work we use vanilla RNN as well as Gated Recurrent Units networks Cho et al. (2014) for the encoder. Models are trained with pytorch (Paszke et al. (2017)) on GPU using the well-known optimizer Adam (Kingma & Ba (2014)) with a learning rate of 0.0001 and Mean Squared Error (MSE).

To select the best hyperparameters and epoch during training we monitor root MSE applied on *descaled* predictions of the validation set. Hyperparameters are presented in Table 1 and we use $\lambda_s = 80$, $\lambda_d = 4$ and $\lambda_t = 30$ for embeddings' sizes. Experiments are run with 5 different random seeds to compute standard deviation of the error.

Table 1: Hyperparameters used in the different architectures

	Batch size	Layers	Hidden size
Multivariate	2	1	400
Univariate	1	1	100
Spatial	128	2	200

4.2 BASELINE

A strong baseline is constructed by averaging previous values given their context. Dealing with a similar application of tap-in logs forecasting, Roos et al. (2016) propose a Bayesian network but performs slightly worse than the average baseline. Indeed, the considered series are very stationary and heavily depend on the context.

The baseline model is a tensor of predictions of size $7 \times S \times T$, where the first dimension corresponds to each day of the week. For a specific day d, station s and time-step t, the average baseline is equal to $\frac{\sum_{i=1}^{N} \mathbf{1}_{\mathrm{D(i)=d}} \mathbf{X}_{i,s,t}}{\sum_{i=1}^{N} \mathbf{1}_{\mathrm{D(i)=d}}}$, D being a look-up table from date stamp to day.

This model is only based on domain expert knowledge and contextual information. Unlike machine learning models, it cannot adapt to anomalous behaviors but it is context aware.

4.3 QUANTITATIVE RESULTS

RMSE of the different architectures before the addition of temporal context are presented in Table 2. All recurrent models, using RNN or GRU, significantly outperform the baseline. In particular, we check in Figure 5 that the models learn more than the average behavior by plotting predictions

during November 4th. An anomaly seems to occur during the day, disturbing the baseline while recurrent models precisely fit to the unusually low traffic. This means that the proposed models learned the dynamics of the time-series and are robust to unseen values.

Table 2: Forecasting RMSE of subway logs on 15 minutes windows using RNN/GRU variants of our proposed architectures against an average baseline

	RNN	GRU	Baseline
Multivariate	28.31 ± 0.09	27.83 ± 0.12	
Univariate	26.15 ± 0.08	26.73 ± 0.18	31.98
Spatial	24.98 ± 0.05	24.96 ± 0.05	

Besides, the Spatial model is especially strong compared to the Univariate one. Spatial embeddings efficiently replace the very costly architecture based on S different RNN and use much less parameters. Indeed, the Spatial GRU contains 434,961 parameters, versus 9,207,297 for the Univariate GRU. These first results strengthen the hypothesis that context can be efficiently learned.

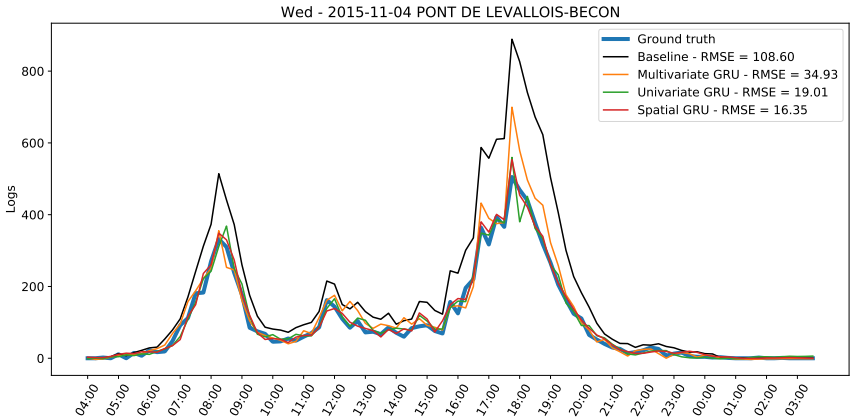


Figure 5: Predicted logs at the station Pont de Levallois-Becon on Wednesday 11/04/2015. November 4th is not a particular day by itself but an anomaly seems to have happened. The baseline mispredicts while our recurrent models correctly fit with the ground truth.

4.4 Improvement with temporal context

We assumed that it would be beneficial to combine the dynamics of recurrent models with the temporal contextual information used in the baseline. To that end we learned day and time embeddings within the previous models and present the results in Table 3. Since RNN and GRU performed similarly we chose to display only GRU results. The first column corresponds to the previous results.

Table 3: Forecasting RMSE of subway logs on 15 minutes windows using GRU models with different temporal embeddings

	No context	Day	Time	Day and time	Baseline
Multivariate	27.83 ± 0.12	27.99 ± 0.11	27.36 ± 0.12	27.24 ± 0.07	
Univariate	26.73 ± 0.18	25.16 ± 0.09	26.76 ± 0.14	24.93 ± 0.12	31.98
Spatial	24.96 ± 0.05	24.73 ± 0.07	24.96 ± 0.08	24.77 ± 0.13	

With the exception of day embeddings for Multivariate and Univariate GRU, the addition of temporal context benefits all models. Interestingly, the combination of time and day embeddings for these two architectures is better than time embeddings alone. On the opposite, the Spatial model benefits more from day embeddings.

4.5 INFERRING THE FUTURE

In the previous experiments we focused on predicting one value from each observation. However, we would like our model to deliver predictions in a window wider than 15 minutes. Therefore, for each step t, after the model has generated prediction $t + 1$, we feed it with this prediction in order to get value at $t + 2$, etc. Obviously the errors made at a previous step will propagate and the prediction will degrade, resulting in a increase in loss. In Figure 6 we plot this loss evolution against the number of time-steps predicted. We find that the addition of temporal embeddings noticeably improves the quality of predictions until a further horizon. While vanilla models perform similarly or worse than the baseline after one hour of predictions, augmented models adopt a concave curve deteriorate much slower. In particular, the addition of temporal embedding to the Spatial model allows to double the horizon during which it beats the baseline.

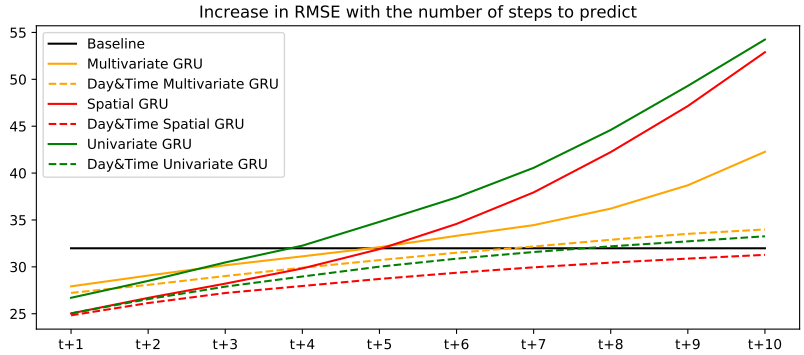


Figure 6: Evolution of RMSE values when the predictive window increases for GRU models. The values at t+1 can be found in the previous tables. Dashed lines correspond to models augmented with day and time embeddings.

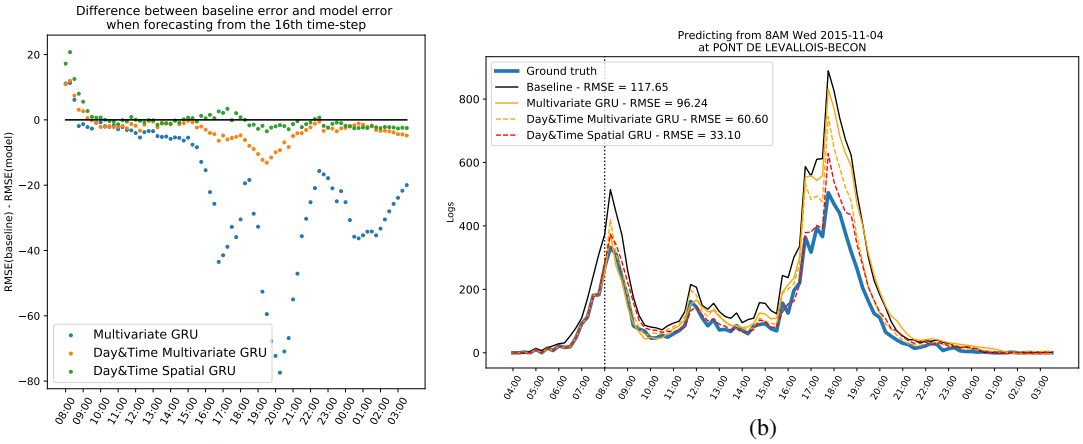
As a second evidence that temporal context is especially useful when predicting farther in the future, we input p observed values to the model to compute a starting hidden state h_p and then feed it only with its own predictions. Results of this experiment are presented in Figure 7 for $p = 16$, i.e. we input values until 8AM.

Figure 7a shows, for each time-step starting from 8AM, the difference between RMSE for the baseline and three recurrent models, averaged on the test set. We observe that the vanilla Multivariate model performs significantly worse than the baseline as the day progresses, especially during peak hours. On the other hand, temporal models tend to converge to the average model. Indeed, when predicting long term sequences, the historical mean is the best estimator in the least square sense. Therefore, spatial and temporal context allow the Day & Time Spatial GRU to predict as well as the baseline with very partial information. Besides, as seen in Figure 6, it is even better for around one hour after the last observed value was inputted.

In Figure 7b, the new protocol is applied to the same disrupted sample as in Figure 5 and in this particular case, the baseline is not a good estimator. On the opposite, contextual models are able to detect from the first 4 hours of information that the traffic is disrupted and that they should diverge from the baseline. Even in this unusual case, temporal context entails competitive long-term predictions.

5 CONCLUSION

In this paper we presented a novel idea for time-series forecasting with contextual features. It consists in learning contextual information that strongly conditions the observed phenomenom within a recurrent neural network. We applied this general idea to the concrete case of transportation logs forecasting in the subway and observed significant improvement of the prediction error when using spatial and temporal context. In particular, the proposed framework performs significantly better in one-step ahead prediction and remains competitive in long-term forecasting. In a very applicated perspective, robust recurrent models could be used in the case of anomalies to accurately predict traffic recovery and help users adapt their behavior.



(a) (b)

Figure 7: Predictions for the test set are computed using only he 16th first values of each day, i.e. until 8AM and we plot: (a) the average RMSE difference between the baseline and some proposed models for every time-step. 0 corresponds to the baseline performance & (b) the predicted logs for the same day and place than in Figure 5.

ACKNOWLEDGMENTS

REFERENCES

George EP Box and Gwilym M Jenkins. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 17(2):91–109, 1968.

Irina Ceapa, Chris Smith, and Licia Capra. Avoiding the crowds. *Proceedings of the ACM SIGKDD International Workshop on Urban Computing - UrbComp '12*, 2012. doi: 10.1145/2346496.2346518. URL http://dx.doi.org/10.1145/2346496.2346518.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

J.T. Connor, R.D. Martin, and L.E. Atlas. Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, 5(2):240–254, Mar 1994. ISSN 1045-9227. doi: 10.1109/72.279188. URL http://dx.doi.org/10.1109/72.279188.

Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. High-Order Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting. Feb 2018. URL http://arxiv.org/pdf/1802.07007v1.

Chuan Ding, Donggen Wang, Xiaolei Ma, and Haiying Li. Predicting Short-Term Subway Ridership and Prioritizing Its Influential Factors Using Gradient Boosting Decision Trees. *Sustainability*, 8(11):1100, Oct 2016. ISSN 2071-1050. doi: 10.3390/su8111100. URL http://dx.doi.org/10.3390/su8111100.

Sandra M. Guzman, Joel O. Paz, and Mary Love M. Tagert. The Use of NARX Neural Networks to Forecast Daily Groundwater Levels. *Water Resources Management*, 31(5):1591–1603, Mar 2017. ISSN 1573-1650. doi: 10.1007/s11269-017-1598-5. URL http://dx.doi.org/10.1007/s11269-017-1598-5.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

D. Koschwitz, J. Frisch, and C. van Treeck. Data-driven heating and cooling load predictions for non-residential buildings based on support vector machine regression and NARX Recurrent Neural Network: A comparative study on district scale. *Energy*, 165:134–142, Dec 2018. ISSN

0360-5442. doi: 10.1016/j.energy.2018.09.068. URL http://dx.doi.org/10.1016/j.energy.2018.09.068.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. Jul 2017. URL http://arxiv.org/pdf/1707.01926v3.

Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 194–200. AAAI Press, 2016. URL http://dl.acm.org/citation.cfm?id=3015812.3015841.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Jeremy Roos, Stephane Bonnevay, and Gerald Gavin. Short-Term Urban Rail Passenger Flow Forecasting: A Dynamic Bayesian Network Approach. *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec 2016. doi: 10.1109/icmla.2016.0187. URL http://dx.doi.org/10.1109/ICMLA.2016.0187.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. Jun 2015. URL http://arxiv.org/pdf/1506.04214v2.

Florian Toqué, Mostepha Khouadjia, Etienne Come, Martin Trepanier, and Latifa Oukhellou. Short & long term forecasting of multimodal transport passenger flows with machine learning methods. In *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, pp. 560–566. IEEE, 2017.

Mascha Van Der Voort, Mark Dougherty, and Susan Watson. Combining kohonen maps with arima time series models to forecast traffic flow. *Transportation Research Part C: Emerging Technologies*, 4(5):307–318, Oct 1996. ISSN 0968-090X. doi: 10.1016/s0968-090x(97)82903-8. URL http://dx.doi.org/10.1016/S0968-090X(97)82903-8.

Bao Wang, Xiyang Luo, Fangbo Zhang, Baichuan Yuan, Andrea L. Bertozzi, and P. Jeffrey Brantingham. Graph-Based Deep Modeling and Real Time Forecasting of Sparse Spatio-Temporal Data. Apr 2018. URL http://arxiv.org/pdf/1804.00684v1.

Hang Xie, Hao Tang, and Yu-He Liao. Time series prediction based on NARX neural networks: An advanced approach. *2009 International Conference on Machine Learning and Cybernetics*, Jul 2009. doi: 10.1109/icmlc.2009.5212326. URL http://dx.doi.org/10.1109/ICMLC.2009.5212326.

Jun Xu, Rouhollah Rahmatizadeh, Ladislau Boloni, and Damla Turgut. Real-Time Prediction of Taxi Demand Using Recurrent Neural Networks. *IEEE Transactions on Intelligent Transportation Systems*, 19(8):2572–2581, Aug 2018. ISSN 1558-0016. doi: 10.1109/tits.2017.2755684. URL http://dx.doi.org/10.1109/TITS.2017.2755684.

Bin Yu, Zhong-Zhen Yang, Kang Chen, and Bo Yu. Hybrid model for prediction of bus arrival times at next station. *Journal of Advanced Transportation*, 44(3):193–204, Jul 2010. ISSN 0197-6729. doi: 10.1002/atr.136. URL http://dx.doi.org/10.1002/atr.136.

Jian Zheng, Cencen Xu, Ziang Zhang, and Xiaohua Li. Electric load forecasting in smart grids using Long-Short-Term-Memory based Recurrent Neural Network. *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, Mar 2017. doi: 10.1109/ciss.2017.7926112. URL http://dx.doi.org/10.1109/CISS.2017.7926112.

Ali Ziat, Edouard Delasalles, Ludovic Denoyer, and Patrick Gallinari. Spatio-Temporal Neural Networks for Space-Time Series Forecasting and Relations Discovery. *2017 IEEE International Conference on Data Mining (ICDM)*, Nov 2017. doi: 10.1109/icdm.2017.80. URL http://dx.doi.org/10.1109/ICDM.2017.80.