ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

Autoregressive Convolutional Recurrent Neural Network for Univariate and Multivariate Time Series Prediction

Matteo Maggiolo and Gerasimos Spanakis

Department of Data Science and Knowledge Engineering, Maastricht University
6200MD, Maastricht, the Netherlands

Abstract. Time Series forecasting (univariate and multivariate) is a problem of high complexity due the different patterns that have to be detected in the input, ranging from high to low frequencies ones. In this paper we propose a new model for timeseries prediction that utilizes convolutional layers for feature extraction, a recurrent encoder and a linear autoregressive component. We motivate the model and we test and compare it against a baseline of widely used existing architectures for univariate and multivariate timeseries. The proposed model appears to outperform the baselines in almost every case of the multivariate timeseries datasets, in some cases even with 50% improvement which shows the strengths of such a hybrid architecture in complex timeseries.

1 Background & Introduction

Time Series (TS) modelling and forecasting has been the target of research for more than ninety years now, however the first contributions using neural networks happened in the 1980s when Recurrent Neural Networks (RNN) [1] were introduced. New advanced variants of a base RNN were proposed in the following years, namely the Long-Short Term Memory (LSTM) network [2] in the 1990s and the Gated Recurrent Unit (GRU) network [3] in 2014. These models have shown to be able to model dependencies deep in time, and therefore are very useful to model long-term, low frequency patterns in the data. In the mean time, Convolutional Neural Networks (CNN) [4], initially applied to image classification were also applied to TS analysis, takes advantage of the high correlation between neighbouring time steps, since it makes the dimensional continuity assumption: features and values that are close in the input dimension (in this case, it is time) are more correlated with respect to inputs at very distant positions. Work has been done to fully exploit this type of networks in the field of TS analysis, for example by feeding in different transformation of the input [5]. The continuity assumption makes this type of network very good at modelling short-term, high frequency patterns in the input signal.

Different approaches (in different fields) have tried to combine the two neural models (RNN and CNN) into a hybrid model (e.g. [6]). In this direction we also contribute with a newly proposed model but also by comparing it with different baselines, namely the popular ARIMA (Auto-Regressive Integrated Moving Averages) models , Support Vector Machines (SVM) , baseline LSTM and GRU, as well as models from recent literature (and are presented in the experiments).

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

2 Proposed Model

Our proposed model is inspired by the observations about RNNs and CNNs made in the Introduction and is suitable for both the univariate and multivariate TS forecasting problems. The model is composed of three parts and an overview can be seen in Figure 1: (a) a multi-scale, convolutional part to extract features from the input TS, (b) a recurrent part with three GRU units to encode the sequence, followed by a linear transformation to obtain the output and (c) an autoregressive part. These parts are analyzed in detail below.

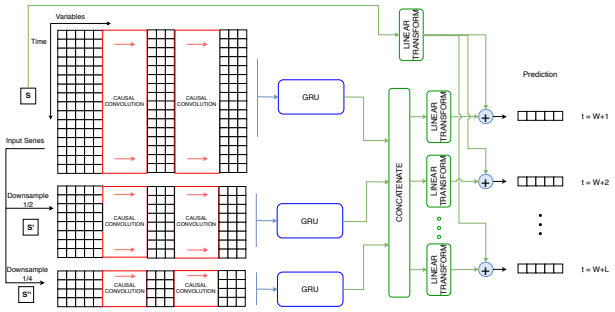

Fig. 1: A representation of the proposed model architecture.

First, the input TS is downsampled twice by a factor of 1/2 and 1/4 (averaging the values), generating a total of three input series of different lengths. This procedure will enable the convolutional layer to model a larger frequency band of the signal. In formulae, if the input TS is $S \in R^{T \times v}$, we have the three series:

$$S = (\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, ..., \mathbf{s}_T)$$
$$S' = (\mathbf{s}'_1, ..., \mathbf{s}'_{T/2}) = \left(\frac{\mathbf{s}_1+\mathbf{s}_2}{2}, \frac{\mathbf{s}_3+\mathbf{s}_4}{2}, ...\right)$$
$$S'' = (\mathbf{s}''_1, ..., \mathbf{s}''_{T/4}) = \left(\frac{\mathbf{s}_1+\mathbf{s}_2+\mathbf{s}_3+\mathbf{s}_4}{4}, ...\right)$$

where we assumed that our window size $T = 0 \bmod 4$. Then we apply two layers of causal convolution with the Rectifier Linear Unit (ReLU) as activation function. For filter $k = 1, ..., N_f$ at the first layer and for filter filter $j = 1, ..., N_f$ at the second layer we will have: $q_k = RELU(W_k^{(1)} * S + b_k^{(1)})$ and $g_j = RELU(W_j^{(2)} * Q + b_j^{(2)})$. Similarly we can define the q and g for the other two series (namely q'_k, q''_k, g'_j, g''_j). where W and b denote the filter weights and biases, which are different for each layer and for each TS resolution, and $*$ represents a causal convolution, as described in [7]. The outputs of the multi-scale convolution are the sequences G, G' and G'' which have different lengths and different receptive fields over the input sequence, so as to capture multiple frequencies of patterns. We decided to use downsampling to preprocess the TS, instead of using a convolution with larger receptive field and stride, to reduce the number of parameters of the CNN layers. Furthermore, this constrains the convolutional kernels in the downsampled streams of the architecture to only model the low frequency patterns in the data.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

Next, we encode the matrices utilizing three GRU unit [3]. We compute the GRU's hidden states as: $h_i = GRU(h_{i-1}, g_i)$ and similarly for h'_i and h''_i. Finally, the output prediction of the nonlinear part at time t is computed with a linear transformation of the concatenation of last recurrent hidden states, that is $o_t = W_t^{rec}[h_T, h'_{T/2}, h''_{T/4}] + b_t^{rec}$, where W_t^{rec} and b_t^{rec} are the weights/biases to learn. Note that these are different for every output time step t.

Finally, since the network components are nonlinear, as denoted in [6], the outputs are not sensitive to changes in the scale of the inputs, and this might give rise to problems in the case of non-stationary scale changes in the input sequences. Therefore, we subdivide the model prediction in a non-linear part (as explained in the two previous sections), plus a linear regression on the inputs. The linear prediction is given by $l^j = \mathbf{s}^j\ W^{lin} + b^{lin}$, where $W^{lin} \in R^{T \times L}$, $b^{lin} \in R^L$, L is the length of the model's prediction, $\mathbf{s}^j$ is a column vector representing the input series for the variable j. The weights and biases for the linear shortcut are shared among the different variables. Furthermore, we reduce the regression input window to 5 previous steps in all cases.

The column vector l^j is therefore the linear prediction for the variable j. The final prediction for the whole model is then the sum of the linear and non-linear predictions, that is $\hat{s}^j_{T+t} = o^j_t + l^j_t$. Afterwards the overall model is trained by minimizing the mean square loss function, i.e. $L = \frac{1}{N}\sum_i (Y_i - \hat{Y}_i)^2$.

3 Experiments

We are testing performance for both univariate and multivariate TS forecasting, thus we are using 4 different datasets, two univariate (Daily values for Melbourne's minimum temperature[1] and The Zurich Sunspot dataset[2]) and two multi-variate (Energy production for 10 different photovoltaic power plants in California [3] and SML2010 dataset[4], containing internal and external measurements in a domotic house).

For all datasets, we apply per-variable normalization with $\mu = 0$ and $\sigma = 1$. For all datasets we also apply a gaussian filter of size 5 and std 2 (for each variable), in order to smooth out some of the noise in the data. The model comparison is performed using k-fold cross-validation with $k = 5$. As per metrics, we compare the model prediction's accuracy on the test set using three different metrics, the Mean Squared Error (MSE), the Mean Absolute Error (MAE) and the Dynamic Time Warping (DTW) (using FastDTW [8]).

Three types of experiments were performed with the aforementioned models. First all the models were trained and compared to predict only one step into the future, then a subset of suitable models were trained to predict more than one output step out of time, and subsequently compared them against each other and

[1] https://datamarket.com/data/set/2324/daily-minimum-temperatures-in-melbourne-australia-1981-1990

[2] https://datamarket.com/data/set/22t4/monthly-sunspot-number-zurich-1749-1983

[3] https://www.nrel.gov/grid/solar-power-data.html

[4] https://archive.ics.uci.edu/ml/datasets/SML2010

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

themselves with also different prediction lengths. The last experiments test the need for the autoregressive shortcut, and they were performed using synthetic data that presents varying levels of trend and periodicity.

Table 3 shows the results for the univariate methods. Here it is possible to see that, for the Temperature dataset, the ARIMA model is significantly better against the baseline and the proposed model. The proposed model still ranks second, but not significantly better than the Simple LSTM model. For the Sunspot dataset, on the other hand, the original LSTNet [6] performs best on average but the uncertainty on the measures suggest that further experiments should be performed to better validate these rankings. The proposed structure still ranks third on average, but it is very close in performance with the first two.

Name	One step forecast loss (Univariate)			
Dataset	Temperature		Sunspot	
Loss	MSE($\times 10^2$)	MAE($\times 10$)	MSE($\times 10^2$)	MAE($\times 10$)
Simple LSTM	1.362±0.126	0.9197±0.0400	0.564±0.024	0.5425±0.1076
Deep GRU	2.003±0.220	1.1077±0.0554	2.238±0.271	0.9060±0.0348
SVM	1.771±0.338	1.0321±0.0837	2.773±0.455	0.7133±0.0332
ARIMA	**1.190±0.022**	**0.8659±0.0102**	0.492±0.081	0.5230±0.0468
LSTNet	1.447±0.090	0.9530±0.0548	**0.477±0.087**	**0.4980±0.0505**
Our Model	1.317±0.083	0.9019±0.0290	0.501±0.126	0.5194±0.0653

Table 1: Results on the 1-step TS prediction task on the univariate datasets

On the other hand, for the task of multivariate prediction (results can be find in Table 3), the proposed model is able to outperform all the other models used for comparison, comprised the LSTNet, which follows up in the second place, with a 40% higher average error (MAE) on the Energy dataset, and a 20% higher average error on the SML2010 dataset. These results confirm our hypothesis that for a simple prediction (just one step ahead) univariate time-series problem, traditional models (such as ARIMA) perform very well but when multiple variables are introduced then the ability of CNNs and RNNs to capture their complex relations proves necessary for improving performance.

Name	One step forecast loss (Multivariate)			
Dataset	Energy		SML 2010	
Loss	MSE($\times 10^2$)	MAE($\times 10$)	MSE($\times 10^2$)	MAE($\times 10$)
Simple LSTM	0.691±0.025	0.5734±0.0046	1.070±0.090	0.5676±0.0762
Deep GRU	8.563±2.239	2.0517±0.2572	2.588±0.296	1.0106±0.0751
RidgeReg	0.626±0.175	0.5203±0.0717	0.517±0.281	0.3343±0.0505
LSTNet	0.164±0.044	0.2551±0.0313	0.105±0.035	0.1269±0.0178
Our Model	**0.101±0.037**	**0.1824±0.0374**	**0.085±0.037**	**0.1061±0.0223**

Table 2: Results on the one step TS prediction task on the multivariate datasets

Finally, for the task of multiple steps prediction, results are summarized in Table 3 (for univariate datasets) and Figure 2 (for multivariate). The loss displayed for both cases is Dynamic Time Warping on the output time steps. Looking at the results of univariate datasets, it is possible to see that in all cases but one the simple LSTM model is clearly better than others but also the loss values are very close and not significantly different.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

The story is different for the multivariate datasets, where we are able to better distinguish the models' performances, thus we opted for a figure to better highlight the differences. In both datasets the results are similar: the suggested model significantly outperforms the baseline, and achieves a very low overall loss on the predictions. Its loss is significantly smaller than LSTM which gets the second place (more than 50% improvement) for all multiple time steps in the SML2010 dataset, and for the three steps prediction in the Energy dataset. Furthermore, as the number of predicted time steps increases, LSTNet becomes more unstable over some of the training folds, and its loss rises, possibly because of problems with our implementation. Its loss, however, when it managed to converge to the minimum on other training folds, was still higher than that of the proposed architecture. Further experiments are required to better pinpoint the comparison between our model and the LSTNet, and to clearly assess the better performance of the former.

Name	Multi step DTW forecast loss (Univariate)					
Dataset	Temperature			Sunspot		
# Steps	3-steps	5-steps	7-steps	3-steps	5-steps	7-steps
Simple LSTM	**0.592±0.033**	**1.475±0.143**	2.679±0.303	**0.317±0.059**	**0.720±0.111**	**1.187±0.217**
Deep GRU	0.651±0.063	1.847±0.115	2.710±0.156	0.364±0.147	0.874±0.319	1.421±0.448
LSTNet	0.811±0.008	1.778±0.061	2.870±0.115	0.377±0.080	0.832±0.124	1.365±0.224
Our Model	0.679±0.038	1.672±0.133	**2.598±0.118**	0.359±0.095	0.859± 0.256	1.331±0.362

Table 3: Results on the multiple-steps prediction task on the univariate datasets

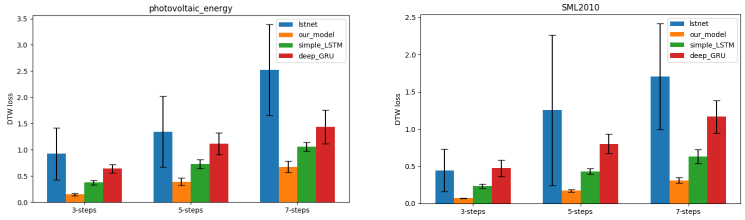

Fig. 2: Results of DTW on the multivariate datasets

Finally, we perform some tests to verify the effect of the linear autoregressive shortcut. Data was generated as a list of 80 time series, each having a length of 120 steps. In Figure 3 we see the prediction error (y axis shows the value-target) using a sliding window approach. The architecture with the autoregressive component (blue) is able to better capture the variable linear trend and periodicity of the data. Without the autoregressive component (orange), the error in the prediction of the oscillations accumulates very quickly over time, while in the presence of the component the information is almost perfectly retained even after many prediction steps. With regard to the predicted trend, the model without the shortcut is noticeably worse at keeping track of the trend's slope.

4 Conclusion and Future Work

In this paper we presented a neural architecture based on three components: a feature extractor, in the form of two down-sampling levels and two convolutional

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

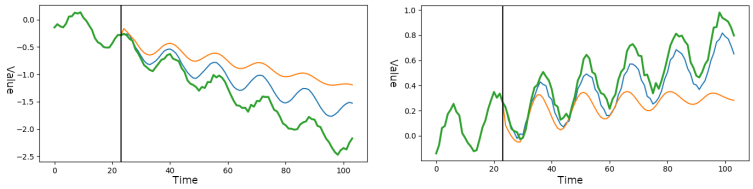

Fig. 3: Comparison between model with linear regression shortcut (blue), without (orange) and the original data (green). Both models are given all timesteps before the vertical black line and predict future time steps using a sliding window approach.

layers applied to each of the inputs, a recurrent encoder, that encodes the input series and the extracted features as a fixed dimensional vector, from which the prediction will be inferred linearly and a linear connection between the inputs and the outputs that shortcuts the problem of scale insensitivity.

Detailed experiments on the model showed that a combination of a CNN and a RNN is able to outperform most of the baseline models in specific settings. More specifically, we demonstrate the effectiveness of a feature extractor over a simple recurrent encoding, especially in multivariate TS. The architecture, however, was not as effective for the task of univariate forecasting, possibly because the datasets were simple enough for simpler models.

Further testing on the model and its hyperparameters is required to fully assess its capabilities, along with a detailed ablation study to confirm the effect of each component. It could be interesting to investigate the effect of the attention mechanism on the nonlinear structures, since it proved to be effective in sequence-to-sequence problems Attention modeling would give the model more freedom in which parts to look at and which parts to ignore, leading to interesting visualizations.

References

[1] J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *PNAS*, 1982.

[2] J. Schmidhuber S. Hochreiter. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[3] K. Cho et al. Learning phrase representations using rnn encoderâdecoder for statistical machine translation. *arXiv:1406.1078 [cs.CL]*, 2014.

[4] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[5] Y. Chen Z. Cui, W. Chen. Multi-scale convolutional neural networks for time series classification. *arXiv:1603.06995v4 [cs.CL]*, 2016.

[6] Y. Yang H. Liu G. Lai, W. Chang. Modeling long- and short-term temporal patterns with deep neural networks. *arXiv:1703.07015v2 [cs.CL]*, 2017.

[7] Oord, et Al. Wavenet: A generative model for raw audio. *arXiv:1609.03499v2*, 2016.

[8] P. Chan S. Salvador. Fastdtw: Toward accurate dynamic time warping in linear time and space. *KDD workshop on mining temporal and sequential data*, 2004.