POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

DEEP NEURAL NETWORKS FOR TIME SERIES FORECASTING BEYOND TRANSFORMERS

Laurea Magistrale in Computer Science Engineering - Ingegneria Informatica

Author: Riccardo Ughi

Advisor: Prof. Matteo Matteucci

Co-advisors: Eng. Eugenio Lomurno,

Academic year: 2021-2022

---

1. Abstract

Human culture has always wondered how to know the future in advance. In recent years, a specific subject of Machine Learning studies precisely this, trying to grasp the future from the study of Time Series with the use of neural networks. More precisely, it aims to find neural network models applicable to Time Series that are able to calculate the trend of an unknown phenomenon from other known parts of it.
In this context, Time Series are sequential data, i.e., sequences of interrelated information in the time domain. It is clear that if it were possible to calculate Time Series values before knowing them through real-world observation, it would mean being able to predict the future, and this would certainly have great appeal and obvious practical interest. Time Series forecasting sets itself precisely this goal, that of calculating the future.
There are numerous active lines of research dealing with this using different techniques. In particular, numerous studies develop models derived from the Transformer, a deep learning tool which was originally invented for linguistic analysis. Several researches foresee a growing complication of the models, in a sort of emulation of what happened in other areas, such as the classification of images and Neural Language Processing (NLP) in which a growing complexity of Deep Learning models can correspond to a better Time Series forecasting sets itself precisely this goal, that of calculating the future!. We will show that this route is incorrect and that simplified models have a higher prediction capability for all the datasets examined. The models studied in this work are representative of situations with different degrees of regularity, but for all of them the conclusion is always that the complicated models perform no better than the simplified ones. We have also developed models capable of making predictions for extremely long periods, up to 5376 values in some cases with quite good results.

2. Introduction

It can be assumed that various phenomena occurring in nature or caused by human technology can be represented by numerical Time Series. In order to predict these phenomena, it is necessary to find models capable of receiving sequences of data as input and calculating numerical outputs that represent the phenomena at successive times. The ambition is to predict the unknown future from the known past.
As a preliminary remark, it is noted that any model that has the ability to predict the future

must receive from the known past the information necessary to know it.
There are many active machine learning research fields that study new models or refine existing ones. The Deep Learning models we are dealing with are certainly an important part of all existing ones. A significant proportion of the more recent ones use models derived from Transformer [1], which has proven to work quite well in computer vision, linguistic analysis and more. Many researchers have therefore found it natural to try to extend its use to other areas and in particular to Time Series in order to predict future trends. The related techniques involve a training phase in which a parametric function is identified that represents the evolution of the phenomenon as a function of time. This function must minimize the differences between the function and reality with sufficient approximation even when the input is not part of those used during training.
This work aims to simplify the techniques currently in use to see if this can lead to improvements in terms of both complexity and performance.

3. Releted works

The performance improvement of Deep Learning models applied to Time Series, understood as the ability to predict more exact and longer unknown time sequences, is an active research field. There has been in very recent times and within a short time many researches which have reported an improvement in the state of the art. Different variants of the Transformer [1] have affected different research groups. For example: FEDformer, Autoformer, Informer, Pyraformer, LogTrans and Reformer.
The Transformer was born for NLP problems and in its original context it can be used to generate texts, translate, and so on. To perform these functions, the Transformer in NLP problems normally has a final layer softmax which is used to obtain a classification that, through a softmax activation function, identifies a dictionary word as the most probable. In the case of Time Series, however, the output is a vector or matrix of continuous values that never represents a classification. Consequently in Time Series, the output is never a one-hot vector. Instead, the ultimate goal in Time Series problems is always to find a vector of numerical values that are never a probability distributionwithout final softmax layer.
Regarding complexity, Transformer type architectures based on the original model have quadratic complexity with respect to the length of the attention level input matrix. This is a significant obstacle because as the length of the input increases, spatial and temporal complexities that are practically inaccessible are quickly reached.
Also for this reason most of the research works have in mind the objective of reducing the complexity from the quadratic one to a lower level.
In this work, an attempt will be made to modify the Transformer to fit the Time Series. However, in contrast to the aforementioned work, a radical simplification of complexity will be undertaken through the elimination of repeated Encoder and Decoder layers and a reduction in the number of parameters. It will be shown that with the elimination of the number of modules and a drastic reduction in the number of parameters, the results will improve instead of deteriorating as might be expected.

4. Models

This work initially examines a first Transformer model adapted to the Time Series. Models are then proposed that simplify this first Transformer-based model by eliminating the Attention Modules, first contained in the Decoder, then also in the Encoder. The models then become much simpler FNN-type neural networks. Furthermore, the results of a Persistent model will be shown as a comparison.
The pre-processing phase which is necessary to all the models examined during this work will be described in detail below.

4.1. Standardization

Neural networks converge much faster if the input data is normalized. All input series were normalized by subtracting the population mean from the population standard deviation:

$$x_i = \frac{x_i^d - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

This simple calculation reduces the ranges of features.

4.2. Windows Input

The input during training is a sequence of numerical vectors often called Windows Inputs. These are Times Series data sequences characterised by

a hyper-parameter that defines their length, which is fixed. These windows overlap each other with a single time interval to make the best use of the available data.
It is absolutely necessary to use Windows Input because the input Transformers models always have vectors or numeric matrices of a fixed size that are processed one at a time. It is the very logic of the Attention layer that works this way. Its purpose in fact is to calculate the relationships between the different parts of the input vector by marking the most important ones with corresponding higher values of the output matrix. Consequently, the Transformer cannot work continuously by adding input values one at a time, but works 'in blocks' so to speak, i.e. by adding vectors or matrices made up of several values.
Data windows are computed together with the target vectors during the pre-processing phase.

4.3. Positional embedding

All Transformer models, both those applied to NLP problems and those applied to Time Series, must apply a positional embedding to inject into the model the information relating to the order in which the values are sequenced. This process is critically important with Time Series because it is clear that the order of values is just as important as the number that represents them and it is worthwhile to delve a little deeper into the subject. In the case of Transformer models applied to Time Series, there is an absolute necessity for the models interpreting them to be able to take the position of the values into account. In fact, in NLP ploblems, the inversion of some parts of the text does not always change the translation or meaning. In NPL, therefore, there is greater flexibility and a certain tolerance towards errors in the reconstruction of the sequence.
In the case of the present work and unlike the classic Transformer and other works, the choice of positional coding is an algorithm obtained from Time2Vec [2] which was devised by its authors with the aim of providing an embedding function positional applicable to different models.
Time2Vec is a learnable vector representation (or embedding) for time as a vector representation.

The i^{th} of the Time Vector of time τ is:

$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i\tau_i + \omega_i & if\ i = 0 \\ F = (\omega_i\tau + \phi_i)\ if\ 1 \leq i \leq k \end{cases}$$

where F (according with authors *sin* in the best in order to capture periodic patterns) is a periodic activation function, ω and ϕ are the frequency and the phase-shift and are learnable parameters.
Time2Vec is inspired by positional-encoding used in Transformer with embedding layer, but:

- represents continuous time instead of discrete time
- is learnable
- is able to capture periodic behaviors which is not the goal in positional encoding
- vector is a model input and is concatenated rather than added to a vector

4.4. Output

In all the models examined in this work, including ones based on Transformer, the output is a numeric vector that directly constitutes the forecast and has a variable length between 24 and 5376, therefore with a very long forecast length. Note that in previous works, probably also due to the complexity of the models and the limitations of the hardware available to the researchers, the prediction has a much shorter length, up to 720 values.
The last layer suitable for calculating a vector (or a matrix) of the values that make up the output is a Fully Connected Layer.

4.5. Optimization

The output vector is computed from a final linear parameterized layer that directly outputs the prediction vector. During the training phase this vector is compared with the target vector which is exactly the same size. The two vectors are applied to the formula to calculate the Mean Absolute Error (MAE):

$$\sum_{i=1}^{D} |x_i - y_i|$$

Two errors are commonly compared, the MAE and the Mean Squared Error (MSE). However, even looking at the Time Series works mentioned in this document, MSE does not seem to add useful information in order to find better performing models. Consequently, to streamline the tables of results, the choice was made to use only the MAE. The MAE compared to the MSE has the advantage of being linear with respect to errors with a consequent more intuitive understanding.

5. Model architectures

In this subsection a description of the peculiar chosen Artificial Neural Network is reported.

- **Encoder + Decoder:** first model derived directly from the Transformer. It is simplified compared to the original Transformer with a drastic decrease of the parameters. The final layer (which in the original Transformer is a softmax type layer) is a Fully Connected Neural Network.
- **Encoder:** this second model is the same as the previous one but composed only of the Encoder part.The purpose of the model is to check whether the decoder module adds information capacity to the model and is therefore useful for improving the results. From an interpretative point of view, this is not obvious because in Transformer models applied to Time Series no activation phase is necessary.
- **Multilayers:** the third model is a Fully Connected Neural Network with 3 layers. Each layer has a 'relu' activation function and a final regularization of type L1 and L2. This model with a further simplification compared to the previous model, does not have the Attention module.
- **One layer:** it is a neural network with only one layer with a further simplification compared to the previous model. The purpose of this model is to test whether a single linear layer can perform against more complex networks with multiple levels or modules of attention.
- **Persistent:** it is obviously not a Neural Network because predictions are made simply by assuming that the past can repeat itself. The forecast consists of exactly the same number of closest past values. It is used only as a basic comparison.
- **Liner regression:** the Linear Regression (without penalty) was calculated as a comparison term for each prediction. This is an extremely simple model that differs from Deep Networks.

6. Training

The proposed models are tested on the Time Series shown in this section. Autocorrelations were calculated to represent the degree of periodicity present in the dataset. It will be seen that the more autocorrelation indicates a regular periodicity, the more accurate the predictions are, with particular reference to long-term predictions. Conversely, the lack of periodicity will lead to less accurate forecasts.
This work considere 13 dataset with a different degree of periodicity (ETT three distribution of electricity, DAX index, Bitcoin values, Venezia see level, Airlines passengers, Temperature of Milan, Traffic San Francisco freeways, hourly electricity consumption, exchange rates of 8 countries, ratio of patients seen with influenza-like illness and the total number of the patients, 21 indicators of weather).
The training phases were performed by applying two different methods depending on whether it was a multidimensional or one-dimensional study, according to the methodology illustrated below:

- If the input is a multidimensional matrix, the backpropagation step is performed for each column of information for each epoch. This means that as many backpropagation steps are performed for each epoch as there are columns in the input matrix. Each step works on a specific column, changing the value of the weights each time to try to get closer to the optimal values.
- In the case of one-dimensional dataset the backpropagation phase is performed for each dataset.

Calculations are performed on 50 epochs and the baseline result is considered to be the one in which the forecast on the valuation set is best.

7. Results

The section just concluded was used to choose the models to compare with those at the state of the art (Table 1). This section will present experiments related to the presented models and a representative selection of the previously described datasets. Models in details are:

- **Encoder + Decoder:** The results of the simplified Transformer are comparable to those of the simple neural network, but with a much higher spatial and temporal complexity that makes them much less efficient. A similar conclusion that considers Transformer-based models of little use for analyzing Time Series was already reached in [3]. They test and compare FEDformer, Autoformer, Informer, Pyraformer, LogTrans and Reformer. Each of these models has considerable complexity, although some authors explicitly state that they have worked to reduce it. The realities and

Table 1: Multivariate NLinear MAE - Bold shows the best results for each row

Dataset	Forecast	Persistent	One Layer	Encoder	Multi	Encoder + Decoder	NLinear	FEDFormer	Autoformer	Informer	LogTrans
ETTh1	96	0.48	0.392	0.39	**0.388**	0.415	0.394	0.419	0.459	0.713	0.74
	192	0.53	0.423	0.433	0.424	0.444	**0.415**	0.448	0.482	0.792	0.824
	336	0.571	0.456	0.463	0.464	0.457	**0.427**	0.465	0.496	0.809	0.932
	720	0.718	0.528	0.542	0.54	0.537	**0.453**	0.507	0.512	0.865	0.852
ETTh2	96	0.428	0.348	0.361	0.38	0.349	**0.338**	0.388	0.397	1.525	1.197
	192	0.509	0.4	0.445	0.448	0.773	**0.381**	0.439	0.452	1.931	1.635
	336	0.56	**0.381**	0.514	0.537	0.469	0.4	0.487	0.486	1.835	1.604
	720	0.682	0.455	0.702	0.685	0.744	**0.436**	0.474	0.511	1.625	1.54
ETTm1	96	0.389	0.335	**0.333**	0.334	0.345	0.348	0.419	0.475	0.571	0.546
	192	0.421	**0.358**	0.389	0.366	0.382	0.375	0.441	0.496	0.669	0.7
	336	0.845	**0.38**	0.415	0.403	0.433	0.388	0.459	0.537	0.871	0.832
	720	0.851	**0.414**	0.425	0.419	0.455	0.422	0.49	0.561	0.823	0.82
Electricity	96	0.477	0.217	0.218	**0.21**	0.221	0.237	0.308	0.317	0.368	0.357
	192	0.372	0.242	0.236	**0.233**	0.242	0.248	0.315	0.334	0.386	0.368
	336	0.435	0.277	0.296	0.297	0.311	**0.265**	0.329	0.338	0.394	0.38
	720	0.561	0.383	0.363	0.371	0.397	**0.297**	0.355	0.361	0.439	0.376
Exchange	96	0.513	0.225	0.916	0.326	0.603	**0.208**	0.278	0.323	0.752	0.812
	192	0.628	0.333	0.76	0.602	0.632	**0.3**	0.38	0.369	0.895	0.851
	336	0.694	0.814	0.686	0.736	0.75	**0.415**	0.5	0.524	1.036	1.081
	720	1.07	**0.499**	0.808	0.796	0.86	0.78	0.841	0.941	1.31	1.127
Weather	96	0.889	0.321	0.501	0.486	0.509	**0.232**	0.296	0.336	0.384	0.49
	192	0.956	0.394	0.572	0.541	0.544	**0.269**	0.336	0.367	0.544	0.589
	336	1.09	0.452	0.571	0.566	0.566	**0.301**	0.38	0.395	0.523	0.652
	720	0.714	0.479	0.541	0.551	0.559	**0.348**	0.428	0.428	0.741	0.675

models of Transformer remain quite complex. However, the results are worse than the simple model proposed here.

- **Multilayers:** With this model in which the Attention modules are removed altogether, the results do not change substantially. Therefore, the results show that it is the entire Attention module that does not add predictive capability for all the datasets examined. This is quite evident when looking at the results and is certainly surprising.
- **Encoder:** This model is the same as before, adapted for not having the decoder. It can be seen by examining the results that models without the Decoder perform very similarly to those without. Removing the Decoder module obviously does not change the quality of the results. The interpretative conclusion is that the Decoder module obviously does not add its own ability to improve the information.
- **One layer:** Looking at the results, this simple single-layer neural network often obtains slightly better results than more complex models. The difference is often minimal and may mean that the models are quite similar in their results. This is certainly surprising. From an interpretative point of view, the fact that simple neural networks perform as well as, or even better than, more complex neural networks may lead to the uncomfortable conclusion that neural networks are unable to interpret the complexity of Time Series datasets. In fact, if the opposite were true, a higher complexity of the input dataset should corre-

spond to a higher complexity of the model for good prediction, as happens in image analysis or with NLP and image recognition problems. But for Time Series this does not happen. Instead, the exact opposite happens, namely that extremely simple networks perform better, and this regardless of the complexity of the dataset. The work of increasing the complexity of models for Time Series to improve results apparently does not work. This observation leaves a future line of research open and it is clear that there is much work to be done to find an eventual solution in this field.

- **Persistent:** It can also be observed that the Persistent forecasting model which has no hyperparameters and which acts as a reference has, in some cases, results comparable to the trailed models studied here, but better than the results obtained by the Informer. The approximation with which the Persistent model approaches the exact values is not uniform and again depends on the regularity of the dataset. The conclusion that the Persist model does better than the Informer model (and LogTrans) is surprising considering the efforts made to make such complex models work.
- **Linear:** The linear model clearly fails and its predictions are worse than Deep learning.

8. Conclusion

This work has set itself the goal of verifying the actual ability of certain Neural Network models to predict the trend of Historical Series even for very long-term forecasts. Starting from a very simplified Transformer-based model with respect to the original Transformer, even more simplified models are introduced.

The first conclusion of some relevance concerns the impact on the goodness-of-fit results of the presence or absence of the Attention modules. The conclusion is that models containing Attention modules applied to Time Series do not seem to add predictive power and their presence. does not bring benefits. However, it is also predictably observed that the model is able to adapt to their presence making the Attention module almost, but not quite, irrelevant.

Twelve different datasets were studied. These datasets have a very different periodicity from each other. They range from the dataset representing the tide level in Venice to the model that has a clear and constant periodicity over tens of years to datasets such as the DAX that do not actually have any periodicity that can be measured by the auto-correlation index. For all of them, however, the conclusions regarding the Transformer's inability to do better than simple linear models are valid.

Several models have been studied that starting from a Transformer model progressively eliminated parts and ended up defining an extremely simple linear model consisting of a single layer

The rather unexpected conclusion is that all these models have a similar predictive capacity. The differences between the results obtained are not so significant. However, a slightly better predictive ability, on average, can be noted in the simpler models.

Furthermore, it has been shown that for data sets with regular periodicity, the simple model consisting of a single linear layer retains the predictive capability even for very long periods, and we have demonstrated this for long periods of up to 5376 values. The conclusion is that simple neural networks are able to repeat periodic patterns and are able to anticipate the future, even with the limitations we have tried to highlight.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. arXiv:1706.03762, 2017.

[2] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, and Pascal Poupart andMarcus Brubaker. Time2vec: Learning a vector representation of time. 2019.

[3] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*, 2022.