# Final Project of Pattern Recognition

## Deep Learning for Multivariate Time Series Forecasting

**1811018 Guan Song Wang**

Dept. Software Engineering, Tongji University, Shanghai, CN

## Introduction

The topic we choose for my final course project is multivariate time series forecasting. Multivariate time series (MTS) data are common in daily life ranging from household electricity consumption, meteorological observation, currency exchange rate, solar power production, and even music notes can all be considered as time series data. There may exist complex dynamic interdependencies between different series that are significant but difficult to capture and analyze. As for forecasting, the task is to predict future time series based on observed data in the past. For example, precipitation in the next days, weeks or months can be forecast according to historical measurement. The further ahead we try to forecast, the harder it is.

When it comes to MTS forecasting using deep learning, it often faces a major research challenge, that is, how to capture and leverage the dynamics dependencies among multiple variables as well as the mixture of short-term and long-term repeating patterns. It is commonly recognized that deep learning is suitable for capturing complex dependencies. And it has also received an increasing attention in time series analysis. For instance, naive RNN models [1], ARIMA models combining with Multilayer Perceptron (MLP) [2,3,4], vanilla RNN with Dynamic Boltzmann Machines [5] and CNN models [6] have been studied in this direction.

To the best of our knowledge, Long- and Short-term Time-series Network (LSTNet, 2018) [7] is the first model that is designed specifically for MTS forecasting with up to hundreds of evolving variables. In LSTNet, CNN is utilized to capture short-term patterns, whereas LSTM or GRU is responsible for memorizing relatively long-term patterns. LSTNet adds a recurrent-skip layer or a typical attention to deal with gradient vanishing problem coming with long-term RNN. And traditional autoregression is also part of the entire model that helps to tackle the scale insensitive problem of neural networks.

Finally, the main contributions of this project are summarized as follows:

* We reproduce the LSTNet by implementing the model with pytorch and repeating the main experiments.
* Several original deep CNN models and RNN models are designed to compare with LSTNet.
* Based on the multi-head attention mechanism in [8], we introduce a temporal attention usage only model with autoregression part for MTS forecasting.
* We rigorously evaluate my approaches and LSTNet on 4 benchmark tasks and 2 of them achieve state-of-the-art prediction performance.

## Related Work

The most renowned model for linear univariate time series forecasting is the autoregressive integrated moving average (ARIMA) [9], which encompasses other autoregressive time series models, including autoregression (AR), moving average (MA), and autoregressive moving average (ARMA). Additionally, linear support vector regression (SVR) [10] treats forecasting problem as typical regression problem with time-varying parameters. However, these models are rarely used in high dimensional multivariate time series forecasting due to their high computational cost.

To forecast MTS data, vector autoregression (VAR), which is a generalization of AR-based models, is proposed [11,12]. VAR models extend AR models to the multivariate setting but ignores the dependencies between output variables. Nevertheless, neither AR-based models nor VAR-based models are capable of capturing non-linearity. For that reason, substantial effort has been made for non-linear models for time series forecasting based on kernel methods [13], ensembles [14] and Gaussian Processes (GP) [15].

GP is a non-parametric method for modeling distributions over a continuous domain of functions. This contrasts with models defined by a parameterized class of functions such as VARs and SVRs. It can be applied to multivariate time series forecasting task as suggested in [16], However, the power of GP comes with the price of high computation complexity. A straightforward implementation of Gaussian Process for multivariate time-series forecasting has cubic complexity over the number of observations. What's more, GP and VAR apply predetermined non-linearity and may fail to recognize different forms of non-linearity for different MTS.

Recently, deep neural networks have received great amount of attention due to their adaptable abilities in capturing non-linear interdependencies. Two variants of RNN, namely long short-term memory (LSTM) [17] and gated recurrent unit (GRU) [18], have shown promising results in several NLP tasks and have also be employed on MTS forecasting. As mentioned before, previous work in this area starts from using naive RNN [1], to hybrid models that combined ARIMA and Multilayer Perceptron [2,3,4], and to the latest Dynamic Boltzmann Machine with RNN [5]. Although these models can be applied to MTS, they mainly target univariate or bivariate time series.

## Framework

* 1. LSTNet

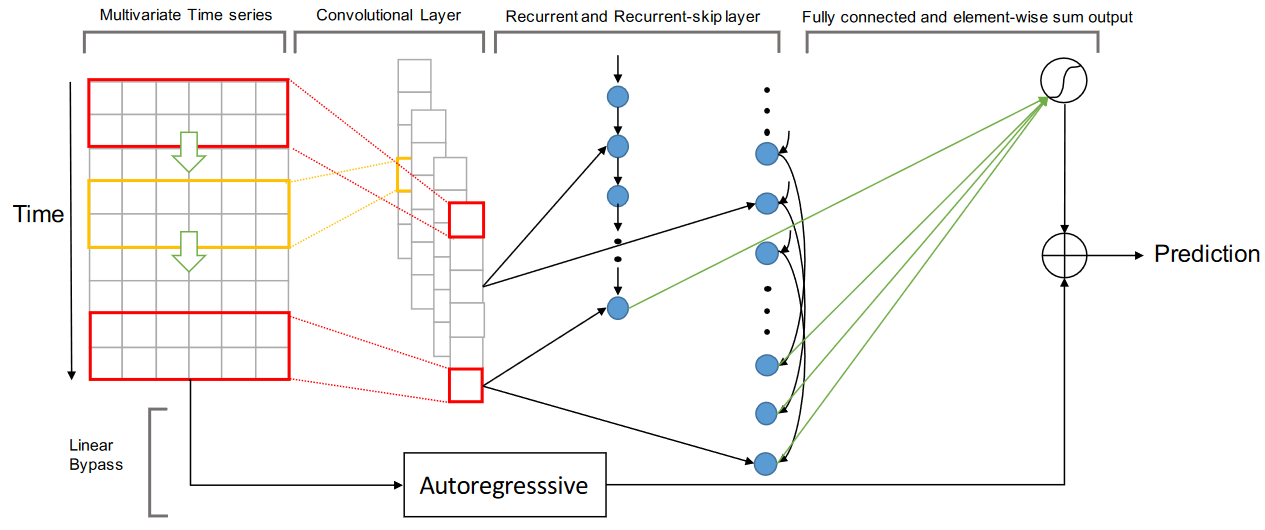


Figure 1: An overview architecture of the LSTNet.

The first layer of LSTNet is a convolutional network without pooling, which aims to extract short-term patterns in the time dimension as well as local dependencies between variables. The output of the convolutional layer is simultaneously fed into the Recurrent component and Recurrent-skip component. The Recurrent component is a recurrent layer with the Gated Recurrent Unit (GRU).

The author designs a recurrent structure with temporal skip connections to extend the temporal span of the information flow and hence to ease the optimization process. Specifically, skip-links are added between the current hidden cell and the hidden cells in the same phase in adjacent periods. In simple words, the input hidden states of time step T in GRU which previously comes from time step T-1 now comes from time step T-p. And p is the number of hidden cells skipped through. The p hyper-parameter need to be tuned for different datasets. **However, the determination of p can be difficult sometimes as some data series do not have an obvious periodic in time.**

The forth component of LSTNet is an autoregressive part. Due to the non-linear nature of the Convolutional and Recurrent components, one major drawback of the neural network model is that the scale of outputs is not sensitive to the scale of inputs.

The LSTNet adopts the classical Autoregressive (AR) model as the linear component. It can be thought of as a simple windowed weighted average of pervious historical data. The final prediction of LSTNet is then obtained by integrating the outputs of the neural network part and the AR component.

* 1. Temporal Attention Layer

The original paper does not provide details on its implementation of temporal attention layer. I choose to use the multi-head attention module consisting of parallel self-attention layers. Self-attention, also referred as intra-attention, is designed to capture dependencies of a single sequence. Self-attention has been used successfully in a variety of NLP tasks including reading comprehension.

The attention fucntion can be defined as mapping a query **q** and a set of key-value pairs **{k, v}** to an output **o.** Mathematically, the attention computation can be expressed as follows:

(1)

Where **Q, K, V** are the matrices formed by query, key and value vectors respectively, and **d** is the dimension of the key vectors. This mechanism is often referred to as the scalar dot-product attention. Since we use only self-attention, **Q, K, V** all correspond to input embeddings of the multivariate times series.

Implicitly, self-attention creates a graph structure for the sequence, where edges indicate the temporal dependencies. Instead of computing a single attention graph, we can actually create multiple attention graphs each of which is defined by different parameters. Each of these attention graphs can be interpreted to encode different types of edges and hence can provide complementary information about different types of dependencies. Hence, we use “multi-head attention” similar to [8], where 8 heads are used to create multiple attention graphs and the resulting weighted representations are concatenated and linearly projected to obtain the final representation.

We stack the multi-head attention module N (default =8) times and perform the actual prediction task using linear function with representations obtained at the final attention module. Unlike transduction tasks, we do not make predictions at each time step in all case.

* 1. CNN & RNN model

To explore different deep learning model for MTS forecasting, we design a simple CNN model and a stacked GRU network.

The first model is a multi-channel CNN model. We will do this by providing each one-dimensional time series to the model as a separate channel of input. The CNN will then use a separate kernel and read each input sequence onto a separate set of filter maps, essentially learning features from each input time series variable. The model overview is as Figure 2.

As for the stacked GRU model, multiple hidden GRU layers can be stacked one on top of another in what is referred to as a stacked GRU model. We use the final hidden states **ht** of the top GRU layer and a linear map function to get the final prediction.

The CNN and RNN model designed here are combined with an autoregressive highway module to deal with the time series data scale problem.

* 1. Program Structure & Functionality

The project files can be divided into 7 parts (Figure 3): shell scripts, main module, utils, saved models, train\_eval method, data files and pytorch models.

* The shell scripts will call the main module with preset hyper-parameters. Four scripts denote four examples for each dataset.

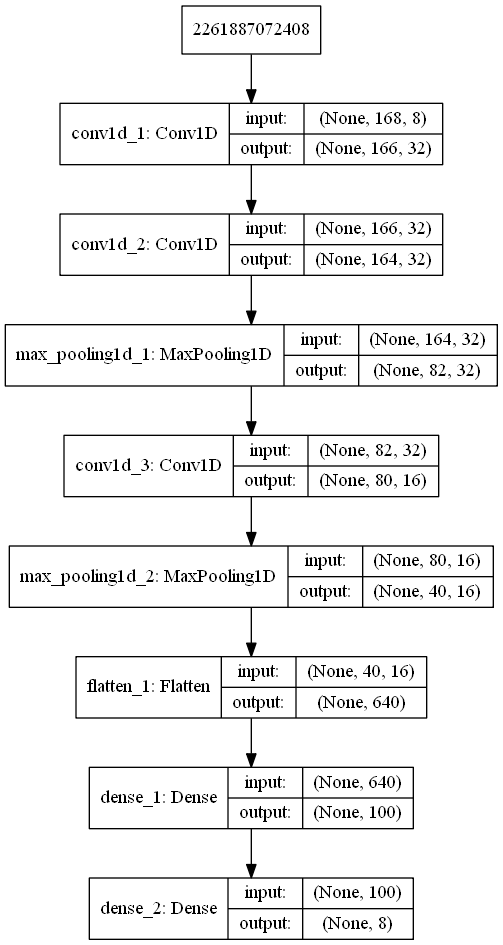


Figure 2: A brief architecture of the CNN model for MTS forecasting.

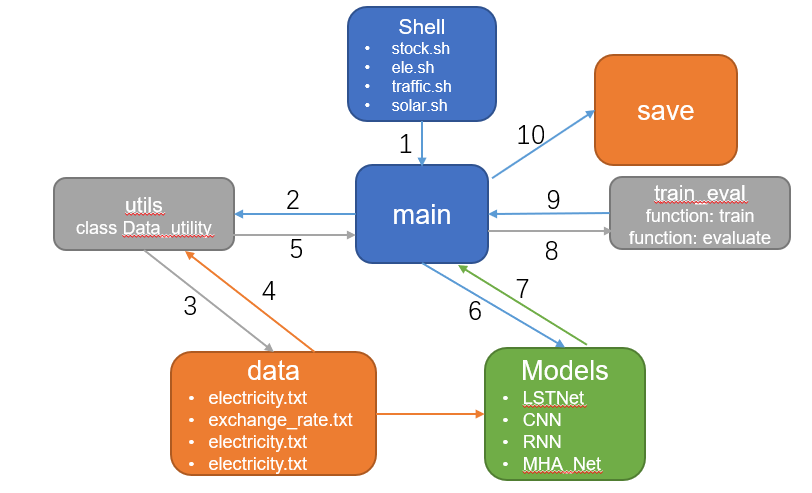


Figure 3: Program structure of this project.

* The main module will analysis parameters and obtain Data\_utility class and model class. Then it calls the training method for each epoch and evaluate method at certain intervals with Data\_utility and selected model. When fetching the loss value of objective criterion in each epoch, it prints the result to console.
* The Data\_utility class can normalize the input time series data and generate data batches for training and test phrase.
* The data file folder and save folder are container of raw time series data and saved pytorch models at best checkpoint.
* The train\_eval module contains our training and evaluation function as well as the method to build parameter optimizer.
* The model module consists of 4 implemented MTS forecasting models, CNN for deep CNN model, RNN for stacked GRU model, LSTNet for reproduced LSTNet and MHA\_Net for multi-head attention network combined with autoregression part.

## Evaluation

The four benchmark datasets we use are all public available, which are published by Lai et al.. Table 1 summarizes the corpus statistics. The descriptions of these datasets are as follows:

* Solar-Energy: the solar power production data from photovoltaic plants in Alabama State in 2006.
* Traffic: two years (2015-2016) of data provided by the California Department of Transportation that describes the road occupancy rate (between 0 and 1) on San Francisco Bay area freeways.
* Electricity: a collection of electricity consumption of 321 clients in kWh.
* Exchange Rate: the exchange rates of eight foreign countries (Australia, British, Canada, China, Japan, New Zealand, Singapore and Switzerland) from 1990 to 2016.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **L** | **D** | **S** |
| **Solar-Energy** | 52560 | 137 | 10 min |
| **Traffic** | 17544 | 862 | 1 hour |
| **Electricity** | 26304 | 321 | 1 hour |
| **Exchange Rate** | 7588 | 8 | 1 day |

Table 1: Dataset Statistics, where **T** is length of time series, **D** is number of variables, **S** is the sample rate.

On these benchmark MTS datasets, since we make comparison of our models with LSTNet, we follow the same evaluation metrics. The first metric is the root relative squared error (RSE), which is defined as:

And the other metric is empirical correlation coefficient (CORR):

Where y is the truth, and is the predict value., is the label of testing data. RSE is a normalized version of Root Mean Square Error (RMSE) that disregards data scale. For RSE, the lower is better, whereas for CORR, the higher is better.

## Performance

We compared our model against the following methods on the typical MTS datasets:

* AR: the standard autoregression model.
* LSTNet-Skip: the LSTNet with recurrent-skip layer.
* LSTNet-Attn: the LSTNet with attention layer.
* CNN: the deep CNN model.
* CNN-AR: the deep CNN model with AR layer.
* RNN: the stacked GRU model.
* RNN-AR: the stacked GRU model with AR layer.
* MHA\_Net: the multi-head attention model.
* MHA\_Net-AR: the multi-head attention model with AR layer.

AR, LRidge, LSVR, and GP are the traditional baseline methods (Lai’s work), LSTNet-Skip and LSTNet-Attn are state-of- the-art method based on deep neural networks.



Table 2. Results on typical MTS datasets using RSE (upper) and CORR (lower) as metric. Best performance in boldface, and worse performance is marked red. We report the mean and standard deviation of our model in ten runs. All numbers besides the results of our model is referenced from the paper of LSTNet paper.

On typical MTS datasets, we choose the best model on the validation set using RSE/CORR as metric to test on the testing set. The numerical results are tabulated in Table 2, where the metric of the upper table is RSE, and the metric of the lower table is CORR.

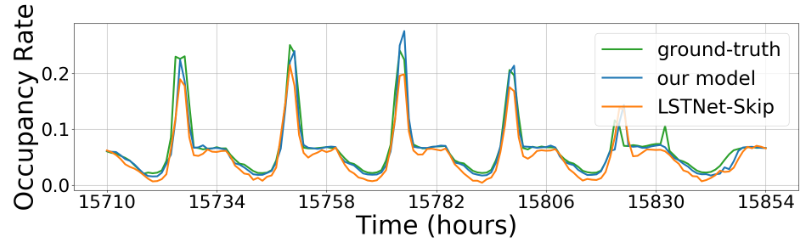


Figure 4. Side by side comparison of prediction between our MHA\_Net model and LSTNet-Skip on testing set of Traffic with 3-hour horizon. Our model clearly forecasts better around the flat line after the peak and around the valley.

All numbers besides the results of our model is referenced from the paper of LSTNet [7]. From both tables, we can clearly see that our best model is MHA\_Net with AR. It outperforms all other methods on any dataset, horizon, and metric, but with only one exception. According to the results, our model consistently demonstrates its superiority on MTS forecasting.

Moreover, on Exchange Rate, where no repetitive pattern exists, our model is still the best overall, while the performance of LSTNet-Skip and LSTNet-Attn fall behind traditional methods, including AR, LRidge, LSVR and GP. We are defeated by LRidge on Exchange Rate with 6-days horizon. Because linear models are good enough at this dataset, deep learning is redundant. We also visualize and compare the prediction of our model and LSTNet-Skip in Figure 4 as an illustration.

But overall, our best model is more general and has stable and competitive results across different datasets. It achieved state-of-the-art performance on both periodic and non-periodic MTS datasets.

## Conclusion

In this project, we focus on MTS forecasting problem and propose several deep learning models which explore the potential capabilities of CNN, RNN and multi-head attention mechanism for this task. Our experiments strongly support temporal attention idea and show that our best model achieves the state-of-the-art results. We also reveal that the challenges remain in forecasting non-periodic and non-stable MTS data with deep learning approaches only.

However, there are still shortcomings of our best model and LSTNet: (1) The highway window of AR and other hyper-parameters of multi-head attention have to be predetermined by human, which have great variation among different time series data. (2) In real world dataset, we usually have rich attribute data series, some dimensions may carry image information. We should not treat each variable dimension equally, integrating them into deep learning model is another challenging problem.

## Reference

1. J. Connor, L. E. Atlas, and D. R. Martin. Recurrent networks and narma modeling. *Advances in Neural Information Processing Systems*, pages 301–308, 1991.
2. Jain and A. M. Kumar. Hybrid neural network models for hydrologic time series forecasting. *Applied Soft Computing*, 7(2):585–592, 2007.

.

1. G. P. Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.
2. G. E. Box and D. A. Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association*, 65(332):1509–1526, 1970.
3. S. Dasgupta and T. Osogami. Nonlinear dynamic boltzmann machines for timeseries prediction. *AAAI-17*. Extended research report available at goo. gl/Vd0wna, 2016.
4. J. B. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In Proceedings of the 24th *International Joint Conference on Artificial Intelligence (IJCAI)*, Buenos Aires, Argentina, pages 25–31, 2015.
5. Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. Modeling long- and short-term temporal patterns with deep neural networks. *SIGIR*, 95–104, 2018.
6. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. *Advances in Neural Information Processing Systems,* 5998-6008, 2017.
7. G. E. Box, G. M. Jenkins, G. C. R., and Ljung, G. M. Time series analysis: forecasting and control. *John Wiley & Sons*, 2015.
8. Cao, L.-J., and Tay., F. E. H. Support vector machine with adaptive parameters in financial time series forecasting. I*EEE Transactions on neural networks,* 1506–1518, 2003.
9. J. D. Hamilton. Time series analysis, volume 2. *Princeton university press Princeton*, 1994.
10. H. Lütkepohl. New introduction to multiple time series analysis. *Springer Science & Business Media*, 2005.
11. Chen, S.; Wang, X. X.; and Harris, C. J. Narxbased nonlinear system identification using orthogonal least squares basis hunting. *IEEE Transactions on Control Systems*, 78–84, 2008.
12. Bouchachia, A., and Bouchachia, S. Ensemble learning for time series prediction. *Proceedings of the 1st International Workshop on Nonlinear Dynamics and Synchronization*, 2008.
13. Frigola, R., and Rasmussen, C. E. Integrated preprocessing for bayesian nonlinear system identification with gaussian processes. *IEEE Conference on Decision and Control*, 552–560, 2014.
14. S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian processes for time-series modelling. *Phil. Trans. R. Soc. A*, 371(1984):20110550, 2013
15. Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural Computation,* 9(8):1735–1780, 1997.
16. Cho, K., van Merrienboer, B., G ¨ ulc¸ehre, C, Bahdanau, D., ¨ Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder–decoder for statistical machine translation. *In Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734, 2014.