Multivariate Time Series Forecasting

Suhas S

May 2022

1 Introduction

Time series forecasting is an interesting problem with industrial applications in stock price predictions, retail demand forecasting and traffic prediction. In particular, forecasting high dimensional data using highly correlated time series has proven to be difficult. Traditional time series forecasting methods operate only on individual time series. These methods include AR, ARIMA and exponential smoothing [3]. However these models fail to capture even simple relationships between time series. In recent times, the M5 competition [5] shed light on the ability of Deep Learning techniques to do well in the forecasting domain. We explore the problem of *"Multivariate Time Series Forecasting"* between time series that have very strong explicit correlations (e.g. traffic data) in this draft.

2 Problem Statement

Suppose we have a collection of N univariate time series $\{Z_{i,1:T}\}_{i=1}^{N}$, where $Z_{i,t}$ denotes the value of the i^{th} series at time step t. In time series forecasting, we usually want to predict the next h time steps $\{Z_{i,T+1:T+h}\}_{i=1}^{N}$. In this draft, we use $\hat{Z}_{i,t}$ to denote our estimate of $Z_{i,t}$. Two popular metrics for measuring the quality of predictions are:

- MAPE:

$$L(\hat{Z}, Z) = \frac{1}{C} \sum_{i=1}^{N} \sum_{t=1}^{t'} \frac{|Z_{i,t} - \hat{Z_{i,t}}|}{|Z_{i,t}|} \mathcal{I}(Z_{i,t} \neq 0)$$

  where $C = \sum_{i=1}^{N} \sum_{t=1}^{t'} \mathcal{I}(Z_{i,t} \neq 0)$

- WAPE:

$$L(\hat{Z}, Z) = \frac{1}{C} \frac{\sum_{i=1}^{N} \sum_{t=1}^{t'} |Z_{i,t} - \hat{Z_{i,t}}|}{\sum_{i=1}^{N} \sum_{t=1}^{t'} |Z_{i,t}|}$$

Most of the existing techniques for multivariate time series forecasting learn a single global model Φ which takes in the i^{th} timeseries $(Z_{i,1}, \ldots, Z_{i,T})$ as input and outputs its future predictions $\hat{Z}_{i,T+1}, \ldots, \hat{Z}_{i,T+h}$ [8,9].

$$(\hat{Z}_{i,T+1}, \ldots \hat{Z}_{i,T+h}) = \Phi(Z_{i,1}, \ldots, Z_{i,T})$$

Note that Φ is common to all the N time series. In such models, Φ is the only way information is shared across different time series. These models can potentially have two issues:

1. they may not be able to capture correlations across time series
2. a single global model may not be able to specialize well to individual time series

3 Related Work

Deep neural networks (DNNs) have increasingly been used in multi-horizon forecasting, demonstrating strong performance improvements over traditional time series models [5, 7]. While many architectures have focused on variants of recurrent neural network (RNN) architectures [6, 8, 10], recent improvements have also used attention-based methods to enhance the selection of relevant time steps in the past – including Transformer-based models [1]. Much of the focus in these papers is capturing long term dependencies within the same time series using attention based mechanisms.

There have also been a number of Graph Convolution Network [12] based methods that are proposed mainly for traffic data prediction [4, 13]. These methods aim to learn the dependencies between different nodes in a network and use them during inference. They make certain assumptions about the structure of the laplacian matrix. With these methods, it is unclear how to translate these methods to unstructured data.

4 Attention based methods

To address the issue on capturing cross-correlations, we consider the following model. In this model, instead of passing just the i^{th} time series data as input to Φ, we pass the data from other timeseries as well:

$$\hat{Z}_{i,t} = \Phi(Z_{i,1:t-1}, \{Z_{j,1:t-1}\}_{j \neq i})$$

Note that Φ is common to all the N time series. Using representer theorems we can show that such a Φ should have the following form for some functions Φ' , Φ''

$$\Phi(Z_{i,1:t-1}, \{Z_{j,1:t-1}\}_{j \neq i}) = \Phi'\left(Z_{i,1:t-1}, \sum_{j \neq i} \Phi''(Z_{i,1:t-1}, Z_{j,1:t-1})\right) \tag{1}$$

Remark 1: *Note that one could also consider models of the form $\hat{Z}_{i,t} = \Phi_i(Z_{i,1:t-1}, \{Z_{j,1:t-1}\}_{j \neq i})$, where Φ_i is unique to the i^{th} series. The classical VAR model falls in this category. One disadvantage with these models is that they have too many parameters and require a lot of data for good performance*

Remark 2: *One could consider a straightforward multivariate extension of the univariate RNN, TCN used in the works of [8, 9] . In these approaches, a single global LSTM/RNN/TCN receives and predicts all target dimensions at once. Similar to VAR, these approaches have too many parameters and require a lot of data for good performance.*

We believe that existing models might not be enough to capture relationships between highly correlated time series. For example, we would like to do well on the following kind of data - See Figure 1 .

This kind of relationship can be expected in traffic data. Consider two nodes A and B on a road with traffic flowing from A to B. If traffic flow increases at A, we expect traffic flow to increase at B after a fixed lag interval.

One way to instantiate the model in Equation 1 is to rely on transformer based architectures. Taking inspiration from attention based models, we propose a model where the value matrix $\mathbf{V}$

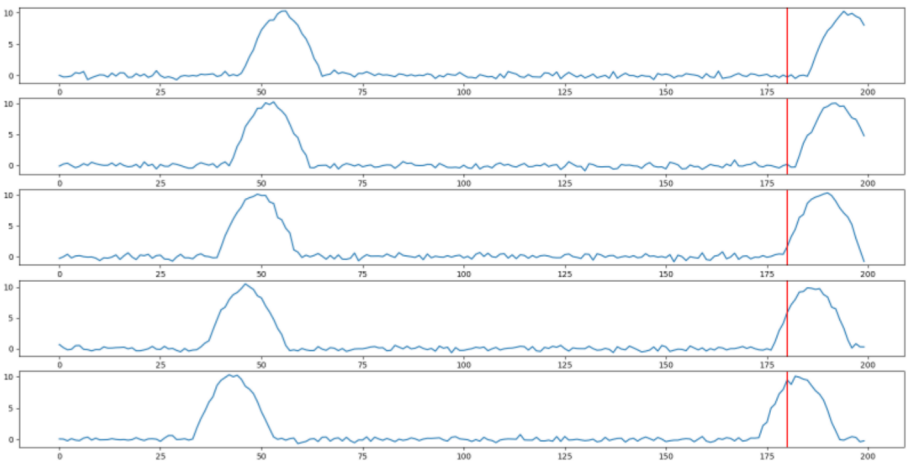

Figure 1: Toy Dataset : The first time series experiences the spike after the fifth time series does. Thus we want to make use of that information while making predictions for the first one. *Our prediction window is from the red line onwards.*

comes from the output of a prediction function Γ_1. Let $v_i = \Gamma_1(Z_{i,1:T}) \in \mathbb{R}^d$ be one row in the value matrix $\mathbf{V}$. The queries and keys are a combination of fixed embeddings $q_{i,f}, k_{i,f} \in \mathbb{R}^k$ (free parameters associated with a time series) as well as dynamic embeddings $q_{i,d} = \Gamma_2(Z_{i,1:T}) \in \mathbb{R}^l$. We concatenate them to get

$$q_i = \{q_{i,f}, q_{i,d}\}$$
$$k_i = \{k_{i,f}, k_{i,d}\}$$

We now linearly combine v_i as follows to get a d-dimensional vector s_i using the query, key values.

$$s_i = \sum_{j=1}^{N} \langle q_i, k_j \rangle v_j$$

$\langle q_i, k_j \rangle$ estimates the correlation between i^{th} and j^{th} time series. The strength of this correlation determines the amount of influence j^{th} time series has on the i^{th} series. We now, use a MLP to get a scalar output from s_i

$$y_i = \Gamma_{3,i}(s_i)$$

The simplified version of the model is shown in Figure 2

Remark 4 (Fixed Embeddings) : A part of the query/key for a time series comprises of a a fixed embedding vector $e_i \in \mathbb{R}^E$ for each time series i. This is akin to learning word embeddings in NLP. One disadvantage of this technique though is that the learned embeddings do not change with time and can potentially lead to poor performance for time-varying processes.

Synthetic Dataset 1				
Model	n	T	WAPE	MAPE
TCN	5	400	51.09	83.15
TCN + Attention	5	400	**25.10**	**26.91**
Synthetic Dataset 2				
Model	n	T	WAPE	MAPE
TCN	2	1000	62.12	88.78
TCN + Attention	2	1000	**27.23**	**31.67**

Table 1: Attention based model results on Toy datasets. Prediction window = 20.

The matrix factorization (MF) techniques of [9, 11] are closely related to the fixed embedding based technique. To see this, let us first look at MF techniques. These techniques assume that there are E latent time series which generate the N observed time series. In particular, they assume that the observed vector $Z_t = [Z_{1,t}, ...Z_{N,t}]$ can be written as

$$Z_t = WU_t$$

where $W \in \mathbb{R}^{N \times K}$ is a fixed weight matrix, and $U_t = [U_{1,t}, \dots U_{K,t}]$ is the vector of latent time series at time step t. [11] assume that U_t follows an AR(p) process and [9] model U_t using a TCN (i.e., they fit a single TCN to all the K time series). Now, consider the previously described fixed embedding based technique. The vector $\hat{Z}_t = [\hat{Z}_{1,t}, \dots, \hat{Z}_{N,t}]$ can be written as

$$\hat{Z}_t = VV^T\Gamma_1(Z_{1:t-1}),$$

where $\Gamma_1(Z_{1:t-1}) = [\Gamma_1(Z_{1,1:t-1}), \dots, \Gamma_1(Z_{N,1:t-1})]$ and $V \in \mathbb{R}^{N \times E}$ is the embedding matrix whose rows correspond to the embeddings of each time series. This formulation is very similar to MF techniques, with the main difference being the assumption of latent time series in MF techniques.

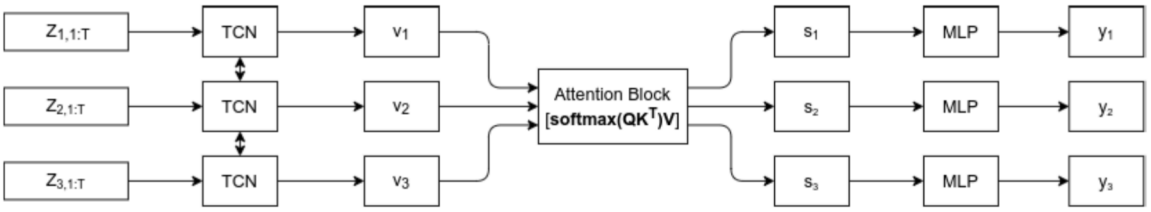

Figure 2: Attention based model

We see in Figure 3 that a TCN fails to capture this kind of relationship since the output $Z_{i,T:T+h}$ depends only on the $Z_{i,T-l:T}$ during inference. However our model is able to use information from the other time series during prediction.

4.1 Experimental Results

Although our model performs well on synthetic datasets, we are not able to outperform baseline models on real world datasets. Refer to Table 2.

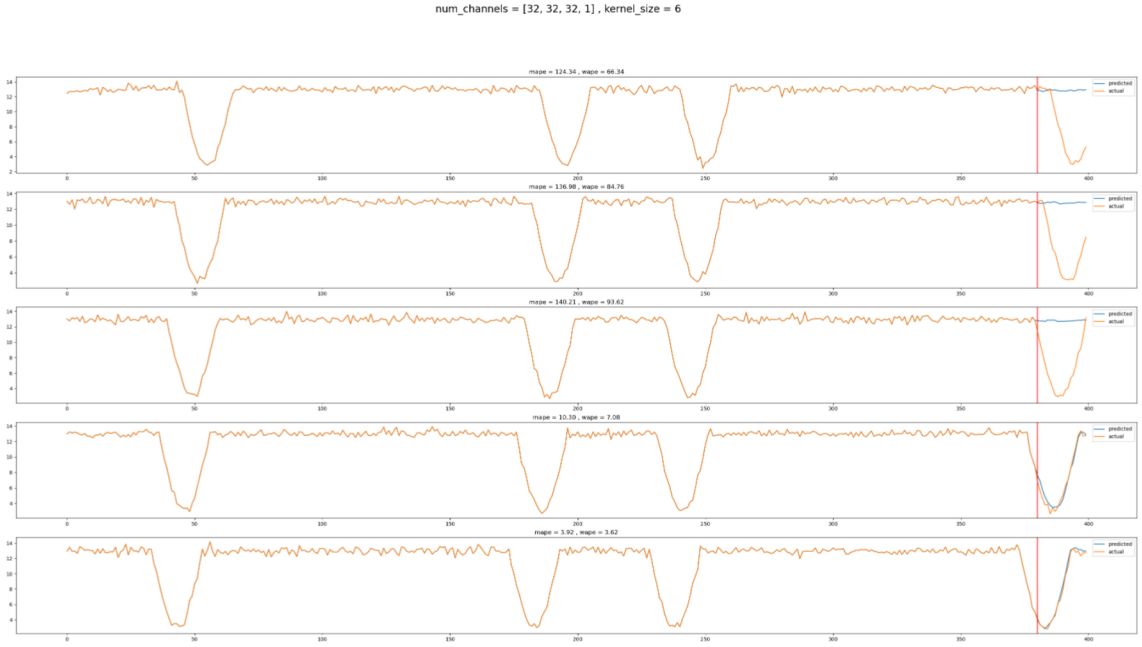

Figure 3: Synthetic Dataset 1 - Baseline TCN (Blue is predicted). Fails on Time series 1-3

For Fred-MD we use prediction window of 18, with 10 windows. For Electricity Hourly, we predict the Global Active Power over the next 20 hours. Since Electricity Hourly has values close to 0, we ignore MAPE for it.

5 Domain Adaptation

To address the issue on specialising to individual time series, we consider the following approaches:

- The first approach is inspired from domain adaptation. Here, we first learn a single global model Φ using the entire data. Next, we fix all the layers except the last layer and fine-tune this layer to the i^{th} time series by relying on data from that particular series and performing a few gradient descent steps on Φ.

- The second approach is to have a local/private model that is specific to each time series to ensure we specialize to all the time series. There are two ways to implement this idea:

  $$\hat{Z}_{i,t} = \Phi(Z_{i,1:t-1}) + \Phi_i(Z_{i,1:t-1})$$

  For the purpose of good generalization, we need to restrict the complexity of $\{\Phi_i\}_{i=1}^N$.

  $$\hat{Z}_{i,t} = \Phi(Z_{i,1:t-1}; \theta + \theta_i)$$

  Here, we assume Φ is parameterized by θ and use a private component θ_i for each series i. For good generalization, we need to restrict the complexity of θ_i (say, we restrict θ_i to lie in an l_p ball).

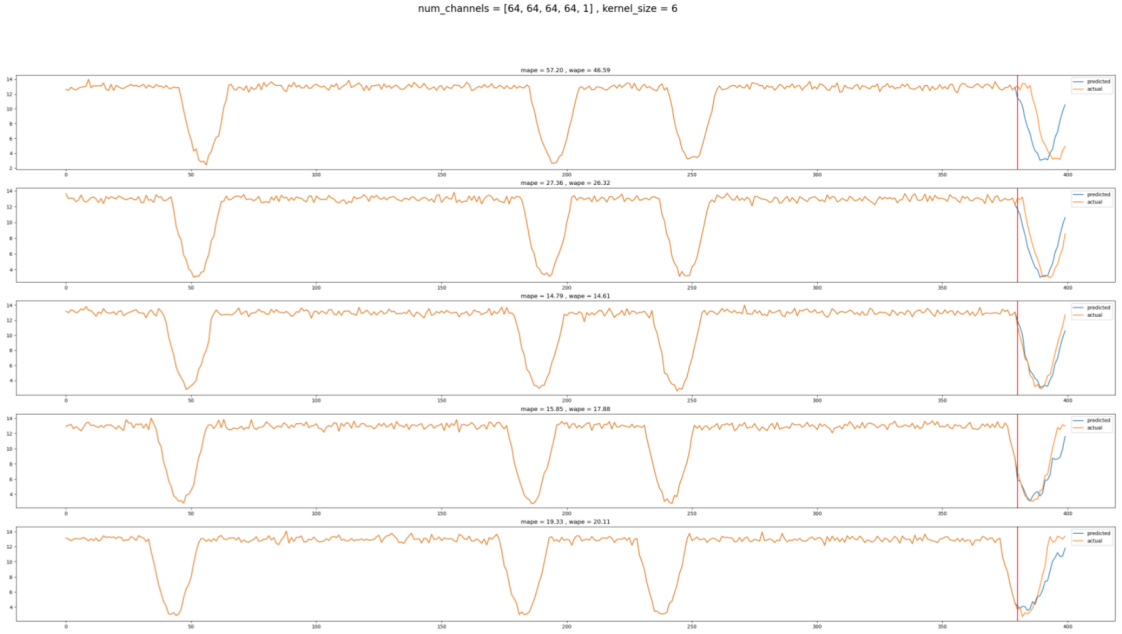

Figure 4: Synthetic Dataset 1 - Attention-based TCN (Blue is predicted)

- We can be a little bit more sophisticated and rely on HyperNetworks [2]. Here, we train a model Γ that takes the i^{th} timeseries as input, and outputs the parameters of the model (e.g. state space models, AR, RNN, TCN, Transformer) used for modeling that timeseries. For example, lets say we are using state space models (SSM) for modeling each individual time series $Z_{i,1:T}$. Then $\Gamma(Z_{i,1:T})$ will be the parameters of the SSM used to model $Z_{i,1:T}$.

Suppose we have $AR(p)$ data with $p = 5$ and θ as the AR parameters. Generate $Z_{i,1:T}$ using the parameters $\alpha_i = \theta + \theta_i$ where θ_i is a sparse vector and θ is common to all time series. Now try and fit an AR model $\Phi(.; \alpha_i)$ as follows :

$$L(\theta, \theta_1, \ldots, \theta_N) = \sum_{i=1}^{N} \left(L' \left(\Phi(Z_{i,T-l:T}; \theta + \theta_i), Z_{i,T+1} \right) + \lambda |\theta_i| \right)$$

Successfully showed parameter recovery with SGD for $AR(5)$ data and $n = 10$ time series and T number of time points in each time series. Refer to Table 3

The standard deviation of the noise used for generation of θ_i was 0.1. Parameter loss here is defined as

$$\sum_{i=1}^{N} ||\theta_{pred} - \theta_{actual}||$$

Fred-MD			
Model	WAPE	MAPE	RMSE
ARIMA	15.2	28.4	?
TCN	**10.1**	**17.3**	?
TCN + Attention	14.5	26.2	?
Electricity Hourly			
Model	WAPE	MAPE	RMSE
ARIMA	19.84	?	17.21
TCN	**12.82**	?	**10.31**
TCN + Attention	14.04	?	19.82

Table 2: Attention based model results on Real world Datasets

Synthetic AR Data			
Model	n-series	T	Parameter loss
AR	10	10000	6.12×10^{-1}
AR + Domain Adaptation	10	10000	$\mathbf{3.62 \times 10^{-3}}$
AR	10	1000	8.18×10^{-1}
AR + Domain Adaptation	10	1000	$\mathbf{6.41 \times 10^{-2}}$

Table 3: Domain Adaptation results on AR data using AR models

We tried using the same method with θ representing the TCN parameters on the Fred-MD Dataset. We consider a rolling window prediction task, with window length = 18 and number of windows = 10. We use TCNs with filters of the size [32,32,1]. Refer to Table 4

Since we didn't observe any improvement by directly adding θ and θ_i, we tried coming up with different ways of combining θ and θ_i.
We first wanted to investigate if sparsity was a good criteria for TCN parameters. For this we generated synthetic data (sine curves) with slowly varying frequencies and amplitudes. We then collected the individual parameters that the TCN converged to when trained using only that time series. Surprisingly, we noticed that the L1-norm betweeen the parameters of dissimilar time series were higher than similar time series.

6 Conclusion

The experiments conducted in this draft show that there is hope in trying to use attention based architectures to extract dependencies between time series. It shows improvement on reasonably realistic synthetic data over the baseline. However, it is still unclear how to scale this up to the high

Fred-MD Data				
Model	n-series	T	WAPE	MAPE
TCN	130	550	**10.12**	**17.33**
TCN + Domain Adaptation	130	550	13.26	23.21

Table 4: Domain adaptation results on Fred-MD Dataset

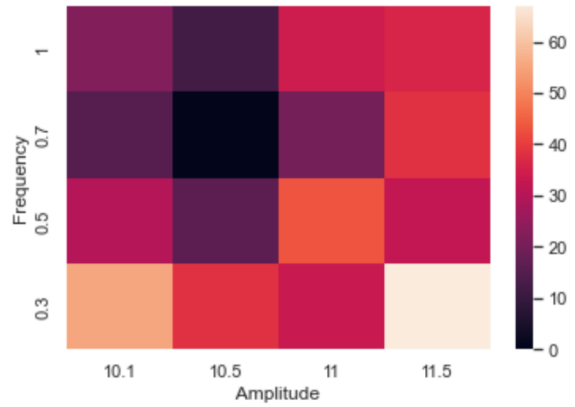

Figure 5: L1 norm distances from $\theta_{10.5,0.7}$ to the other converged θ

dimensional setting. Some ideas to explore could be regarding using sub-sampling while calculating the attention matrix. Further, it also sheds light on the use of domain adaptation based techniques in the field of time-series forecasting.

References

[1] Chenyou Fan, Yuze Zhang, Yi Pan, Xiaoyue Li, Chi Zhang, Rong Yuan, Di Wu, Wensheng Wang, Jian Pei, and Heng Huang. Multi-horizon time series forecasting with temporal attention learning. In *Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery & data mining*, pages 2527–2535, 2019.

[2] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[3] Rob Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

[4] Kyungeun Lee and Wonjong Rhee. Ddp-gcn: Multi-graph convolutional network for spatiotemporal traffic forecasting. *Transportation Research Part C: Emerging Technologies*, 134:103466, 2022.

[5] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m5 competition: Background, organization, and implementation. *International Journal of Forecasting*, 2021.

[6] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[7] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31, 2018.

[8] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[9] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[10] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

[11] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. Temporal regularized matrix factorization for high-dimensional time series prediction. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[12] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.

[13] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858, 2019.