Summary Paper : Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting

Sruthy Annie Santhosh, 312213
Universitat Hildesheim
Hildesheim
Germany
santhosh@uni-hildesheim.de

Abstract

Time series forecasting plays a pivotal role in many areas like demand, weather and financial predictions. These fields contain highly correlated time series which can further help produce optimized predictions. This leads to an area of high dimensional time series forecasting, where each of the correlated time series denotes one of the dimensions. Models need to capture the global patterns and combine them with the local patterns pertaining to individual time series to yield better results. Since most of the existing research focussed on low dimensional time series forecasting, in the paper(1), the authors have proposed a hybrid model known as DeepGLO, which combines matrix factorization model having Temporal convolutional network regularization with another local temporal convolutional network to **think globally and act locally**(1). This model also does not require apriori normalization and through experiment was found to show over 25% improvement in WAPE over the other existing methods on high dimensional datasets.

1 About the Paper

This is a summary report on the paper ' Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting' (1) which was authored by Rajat Sen, Hsiang-Fu Yu, and Inderjit Dhillon while working in Amazon at the time. It was published on October 29, 2019 in the NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems ,arXiv:1905.03806v2 [stat.ML]. It currently has 163 citations on Google Scholar.

2 Introduction

Time series forecasting is a widely popular method used in various sectors of life. It is used in demand forecasting for different items, for financial predictions in the stock market and even for weather predictions(2; 3; 4). In many cases, there exists different time series that are highly correlated to each other. For example, in an e-commerce site like Amazon for predicting the demand of one product, it can be useful to get demand predictions for other items belonging to the same category. This leads to the concept of forecasting for high dimensional time series. Here each dimension indicates one of the correlated time series. The number of correlated time series present can be in millions(1).

Traditionally research has been focused on individual or small number of time series. Such methods are not scalable to the order of millions and do not use the shared temporal patterns for predictions.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

In the paper(1), the authors propose a model to forecast high dimensional time series by using both local and global temporal patterns.

3 Related Work

Some of the existing work in the field of Time series forecasting is being discussed in this section. One of the popular models used is Recurrent Neural network (RNN) model which does sequential modelling and has the ability to capture non-linear temporal patterns. But this model cannot capture long range dependencies due to the vanishing gradient problem. Vanishing gradient problems refers to the gradients diminishing exponentially as they move down the layers during training(5). LSTM (Long short term memory) model tackles this issue by using gating mechanisms and hence capturing long range dependencies(6). DeepAR is another popular model based on LSTM where future time steps are predicted as a function of the corresponding hidden states of LSTM(7). But this model cannot handle more than 100K dimensional time series.

Another popularly used model is the temporal convolutional network (TCN). This is a one dimensional fully convolutional network with dilation. The causal convolutions help maintain the time series factor while the dilations ensure a wider receptive field(8). There was also research done in the Matrix factorization field which led to the Temporal Regularized Matrix Factorisation model (TRMF). This model can handle forecasting of high dimensional time series with missing data. It uses Auto Regressive (AR) based temporal regularization for forecasting thus capturing global patterns(9). However TRMF can only model linear temporal patterns(1). In this paper, the authors have extended the TRMF model with a TCN based regulariser.

4 Contributions of the Paper

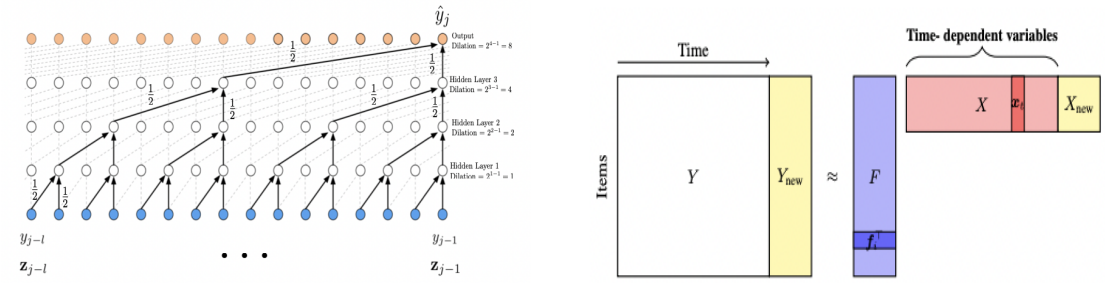
Though much research has been done on the field of forecasting high dimensional time series, there are still two main issues that have not been handled properly.

Firstly, in high dimensional time series, some of the individual time series would be orders of magnitude higher than the others. In such cases, each time series has to be normalized for training and then scaled back during predictions. The mode and parameters of normalization are difficult to choose and can lead to inconsistencies in the accuracy. Thus it is quite difficult to forecast time series having wide variations in scale between each individual time series.

Secondly, though deep models are trained on the entire dataset, during prediction the main focus is only on the local past data. But it can be useful to leverage the global properties during the prediction time also. For example, for the prediction of demand in sales of shirts, it can be useful to consider the past demand values of shirts as well as pants. One method to leverage this information was proposed in (10) but was not scalable for very high dimensional time series. Another method is the TRMF model which represents all the time series as a linear combination of k (very less than the total number) time series that models the global patterns during prediction. But TRMF is an AR based model and hence cannot capture non-linear temporal dependencies. Also it requires re-training to perform rolling predictions. Since it only concentrates on the global patterns during prediction, approximation errors occur(9).

In this paper, the authors have proposed a deep learning model that can think globally and act locally, using both local and global patterns during training and predictions. The model should also be able to handle wide variations in scale. Hence the proposed model has the following attributes(1) :

- A Simple Initialization scheme known as LeveledInit is introduced to handle the variations in scale for a Temporal convolutional network (TCN). This mitigates the need for apriori normalization for training.
- A matrix factorization model regularized by TCN (TCN- MF) is used to handle the global dependencies for prediction. It captures non-linear dependencies. TCN is used as the regularisor and predictions are made.
- A new hybrid model known as DEEPGLO is proposed which is a TCN model which has one of its inputs as the global predictions made by TCN-MF. This global prediction is being passed as a covariate to the TCN along with its individual input time series and covariates.



(a) Figure (a)

(b) Figure (b)

Figure 1: Figure (1a) :Illustration of TCN from paper(1). The model has filter size 2 and number of layers 4. It maps the input the input $y_{t-l:t-1}$ to the one-shifted output $\hat{y}_{t-l+1:t}$ .Figure (1b) :Illustration of Matrix Factorization model for multi dimensional time series. F denotes the features for each time series in the matrix Y , and X contains the temporal patterns (9)

Hence DEEPGLO model leverages both global and local dependencies for both training and prediction.

5 Problem Setting

The main problem being tackled in this paper is the forecasting of high dimensional time series. Representing it in a formal way, we have as input high dimensional datasets containing several correlated time series along with its covariates which can be global features like time of the day, month of the year etc or local covariates for each time series(1).

Input:

- the raw time series as a matrix Y= $[Y^{(tr)}Y^{(te)}]$, where $Y^{(tr)} \epsilon R^{nxt}$, $Y^{(te)} \epsilon R^{nx\tau}$, where n is the number of time series ,t denotes number of time points in training phase and τ denotes the window size for forecasting
- Covariates : $Z \epsilon R^{nxrx(t+\tau)}$

Output: The task's output is to provide prediction of future time steps for the input time series.

$\hat{Y}^{(te)} \epsilon R^{nx\tau}$

The performance of the model is measured by finding the error between the true and predicted values in the test range. The metrics used in this paper are Weighted Absolute Percent Error(WAPE), Squared Error, Mean Absolute Percent Error(MAPE) and Symmetric Mean Absolute Percent Error(SMAPE). Further details on the error metrics are given in the Appendix.

6 Base Model: Temporal Convolutional Network (TCN)

As mentioned in section 4 the newly proposed model is based on the Temporal Convolutional network. In this section, the TCN model is explained in detail.

TCN is a one dimensional fully convolutional network with dilation. Dilation is a method of expanding the kernel by inserting holes or skipping between adjacent elements so as to increase the receptive field (8). Hence a wider range can be taken as the input. For layer i, the dilation factor is 2^{i-1}. Hence by increasing the kernel size and the dilation, the dynamic look back range can be increased. For a filter size k, and dilation d, the look back range is 2(k-1)2^{d-1}. Zero padding is added to ensure that all layers have equal size(8). Usually the stride is assumed to be one. Another important feature of TCN are the causal convolutions. They ensure that the model does not violate the ordering of data , that is , the prediction will depend only on the previous values and not on any future values. Also the TCN has no gating mechanisms and consists of residual blocks for stabilization. Each of

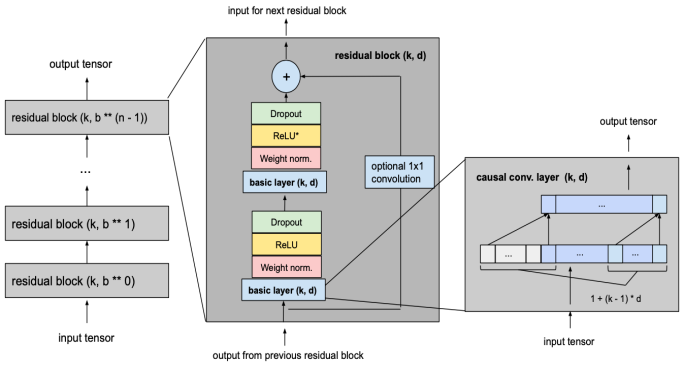


Figure 2: The layers in TCN model. Each residual block contains different layers for stabilising the model(8).

the residual blocks consists of two convolutional layers with RELU function for non-linearity and weighted normalization to ensure that the performance does not vanish. Dropout layers are also added to avoid overfitting(8).

The TCN model takes the past values of a time-series Y_J ,where J =j-l,j-l+1,⋯,j-1 along with their past covariates Z_J as the input and predicts the one-step look ahead predicted value $\hat{Y}_{J+1}$. This is denoted as T (·|Θ), where Θ is the parameter weights of TCN(1). Usually TCN requires apriori normalization and we have already seen the drawbacks of this kind of normalization in section 4 .

7 LeveledInit: Handling Diverse Scales with TCN

One of the main contributions of the paper was the introduction of a new initialization scheme for TCN so as to mitigate the issues regarding the wide variations in scale. LeveledInit Initialization does not require the TCN to be initialized using apriori normalization. The network parameters are initialized in such a way that the future prediction $\hat{y}_j$ is a mean of the values in the past l time points $y_{j-l:j-1}$ where l is the look back range of the TCN. During training, the TCN will learn to make predictions around that mean value, without depending on the scale(1). The authors had stated a proposition for this method which is as follows:

Considering a TCN with one channel per layer, a filter size k = 2 , the number of layers = d, no covariates and all the activation functions are ReLUs. When all the biases are set to 0 and the weights set to 1/k, the prediction for time point j

$$\hat{y}_j = \mu(Y_J)$$

where J = j-l,.....j-1 and l = $2(k-1)2^{d-1}$. This proposition is an extension of the fact that the activation value in an internal layer is the mean of the corresponding k inputs from its previous layers. This scheme can also be extended to include covariates, multiple channels per layer and varying filter sizes. This scheme also matches the performance of the Deep Leveled Network which is being explained in detail in the Appendix section.

8 Global Component : Temporal Convolution Network regularized Matrix Factorization (TCN-MF)

This global component expresses all the time series as a linear combination of k basis time series where k is very much less than n. The authors have used a low rank matrix factorisation model that is regularized by a TCN. The training matrix Y^{tr} is factorized into low rank factors F ϵ R^{nxk} and $X^{(tr)}$ ϵ R^{kxt} . If $X^{(tr)}$ preserves temporal structures then the future values $X^{(te)}$ can be predicted by a time-series forecasting model and thus the test period predictions can be made as $FX^{(te)}$(9). Hence X contains k basis time series that models the global temporal patterns in the whole data set, thereby

Algorithm 1 Mini-batch Training for TCN with LeveledInit

Require: learning rate η, horizontal batch size b_t, vertical batch size b_n, and `maxiters`

Initialize $\mathcal{T}(\cdot|\Theta)$ according to LeveledInit
for `iter` $= 1, \cdots,$ `maxiters` **do**
  for each batch with indices $\mathcal{I}$ and $\mathcal{J}$ in an epoch **do**
    $\mathcal{I} = \{i_1, ..., i_{b_n}\}$ and $\mathcal{J} = \{j+1, j+2, ..., j+b_t\}$
    $\hat{\mathbf{Y}} \leftarrow \mathcal{T}(\mathbf{Y}[\mathcal{I}, \mathcal{J}], \mathbf{Z}[\mathcal{I}, :, \mathcal{J}]|\Theta)$
    $\Theta \leftarrow \Theta - \eta \frac{\partial}{\Theta} \mathcal{L}(\mathbf{Y}[\mathcal{I}, \mathcal{J}+1], \hat{\mathbf{Y}})$
  end for
end for

Algorithm 2 Temporal Matrix Factorization Regularized by TCN (TCN-MF)

Require: $\texttt{iters}_{\texttt{init}}$, $\texttt{iters}_{\texttt{train}}$, $\texttt{iters}_{\texttt{alt}}$.

/* *Model Initialization* */
Initialize $\mathcal{T}_X(\cdot)$ by LeveledInit
Initialize $\mathbf{F}$ and $\mathbf{X}^{(\text{tr})}$ by Alg 3 for $\texttt{iters}_{\texttt{init}}$ iterations.
/* *Alternate training cycles* */
for `iter` $= 1, 2, ..., \texttt{iters}_{\texttt{alt}}$ **do**
  Update $\mathbf{F}$ and $\mathbf{X}^{(\text{tr})}$ by Alg 3 for $\texttt{iters}_{\texttt{train}}$ iterations
  Update $\mathcal{T}_X(\cdot)$ by Alg 1 on $\mathbf{X}^{(\text{tr})}$ for $\texttt{iters}_{\texttt{train}}$ iterations, with *no covariates*.
end for

Figure 3: Algorithms 1 and 2 as in paper (1)

capturing the global pattern. By using the matrix factorisation method, this model is also suitable for high dimensional time series with missing data forecasting(9).

The temporal structures are encouraged in the model by using a TCN as the regulariser. In TRMF, the regularisor added was AR based (9) which captured linear temporal patterns and avoided overfitting. If a TCN can capture the temporal patterns in Y^{tr}, then it can encourage temporal structure in $X^{(tr)}$ also. It can also model non-linear dependencies. If the TCN model is $T_x(.)$, J = 2,....t and $L_2()$ is the squared error function, then temporal structure can be captured by adding the following regularization model to the objective function as follows(1):

$$R(X^{(tr)}|T_x(.))) = \frac{1}{|J|} L_2(\mathbf{X}[:, J]_{,x} (\mathbf{X}[:, J-1]))$$

The overall loss function of the TCN-MF model with λ_T as the regularization parameter for TCN component is

$$L_G(\mathbf{Y}^{(tr)}, F, \mathbf{X}^{(tr)}, T_x) = L_2(Y^{(tr)}, FX^{(tr)}) + \lambda_T R(X^{(tr)}|T_x(.)))$$

Training : The model is trained by mini-batch SGD with two steps being performed alternatively to minimize the above loss function.

1. Minimizing $L_G(F, X^{(tr)}, T_x)$ on factors F and $X^{(tr)}$ keeping $T_x(.)$ fixed (Algorithm 3).
2. Train the network $T_x(.)$ on matrix $X^{(tr)}$ (Algorithm 1)

The model $T_x(.)$ is initialized using the LevelledInit scheme and then the factors F and $X^{(tr)}$ are trained on the same for a number of iterations and then for a number of iterations F , $X^{(tr)}$ and $T_x(.)$ are updated. Algorithm 2 describes the overall training process.

Prediction : The trained network $T_x(.)$ provides standard multi-step look ahead predictions. With the past data points of the basis time series $x_{j-l:j-1}$, the prediction for the next step is given by the network. This new prediction is added to the past values and passed through the model to provide the next prediction as so on. Thus the basis time series in the test range is predicted as $\hat{X}^{te}$. Finally the global predictions are found as $Y^{te} = F\hat{X}^{te}$. TCN-MF can also perform rolling predictions without re-training which the TRMF model could not do (1) .

9 DeepGLO : A Deep Global Local Forecaster

In this section, the newly proposed hybrid model is being discussed. In DeepGLO model, the output of the global component TCN-MF is fed as one of the inputs to a TCN model $T_y(.|\Theta_y)$. Hence the model can utilize both the global temporal patterns as well as local time series data to make predictions. The TCN model has an input size of r+2 dimensions (1).

1. Past data points of the individual local time series

Algorithm 3 Training the Low-rank factors $\mathbf{F}, \mathbf{X}^{(\text{tr})}$ given a fixed network $\mathcal{T}_X(\cdot)$, for one epoch

Require: learning rate η, a TCN $\mathcal{T}_X(\cdot)$.

1: **for** each batch with indices $\mathcal{I}$ and $\mathcal{J}$ in an epoch **do**
2: $\mathcal{I} = \{i_1, ..., i_{b_n}\}$ and $\mathcal{J} = \{j+1, j+2, ..., j+b_t\}$
3: $\mathbf{X}[:, \mathcal{J}] \leftarrow \mathbf{X}[:, \mathcal{J}] - \eta \frac{\partial}{\partial \mathbf{X}[:,\mathcal{J}]} \mathcal{L}_G(\mathbf{Y}[\mathcal{I}, \mathcal{J}], \mathbf{F}[\mathcal{I}, :], \mathbf{X}[:, \mathcal{J}], \mathcal{T}_X)$
4: $\mathbf{F}[\mathcal{I}, :] \leftarrow \mathbf{F}[\mathcal{I}, :] - \eta \frac{\partial}{\partial \mathbf{F}[\mathcal{I},:]} \mathcal{L}_G(\mathbf{Y}[\mathcal{I}, \mathcal{J}], \mathbf{F}[\mathcal{I}, :], \mathbf{X}[:, \mathcal{J}], \mathcal{T}_X)$
5: **end for**

Algorithm 4 DeepGLO- Deep Global Local Forecaster

1: Obtain global $\mathbf{F}$, $\mathbf{X}^{(\text{tr})}$ and $\mathcal{T}_X(\cdot)$ by Alg 2.
2: Initialize $\mathcal{T}_Y(\cdot)$ with number of inputs $r+2$ and LeveledInit.
3: /* *Training hybrid model* */
4: Let $\hat{\mathbf{Y}}^{(g)}$ be the global model prediction in the training range.
5: Create covariates $\mathbf{Z}' \in \mathbb{R}^{n \times (r+1) \times t}$ s.t $\mathbf{Z}'[:, 1, :] = \hat{\mathbf{Y}}^{(g)}$ and $\mathbf{Z}'[:, 2 : r+1, :] = \mathbf{Z}[:, :, 1 : t]$.
6: Train $\mathcal{T}_Y(\cdot)$ using Algorithm 1 with time-series $\mathbf{Y}^{(tr)}$ and covariates $\mathbf{Z}'$.

Figure 4: Algorithms 3 and 4 as in paper (1)

2. r inputs for the original r dimensional covariates
3. The remaining 1 dimension for the output of the global component which will be fed as one of the input covariates.

The steps for training this model are shown in Algorithm 4.

(a) (b)

Figure 5: Figure a, shows some of the basis time series which are extracted from the traffic dataset. These can be combined to get the input time series. We also see that they are temporal in nature. In Figure b, The model DeepGLO is illustrated.The TCN in the figure $T_y(\cdot)$, whose inputs are the original time series ,it's covariates and the output of the global model as covariates. This model combines the local properties with the output of the global model for prediction(1).

Hence by inputting the global prediction as one of the input covariates, the TCN model has both global and local properties of the dataset for making the final predictions. DeepGLO can perform multi-step look ahead and rolling predictions.

10 Datasets and Experiment

An experiment was conducted to check the performance of the DeepGLO model on four real world datasets.

- To perform rolling validation of next 7 days on the Electricity dataset which contains hourly load of 370 houses(11).
- To perform rolling validation of next 7 days on the traffic dataset which contains the hourly traffic on 963 roads in San Francisco(12).
- To perform rolling validation of next 86 days on the Wiki dataset which has the daily web traffic on 115,084 articles in Wikipedia(13).
- To perform rolling validation for next 1440 points on the PeMSD7(M) dataset which contains the traffic data on District 7, California, consisting of 228 time series (14).

There is wide variation in scales in each dataset as shown by the table in figure 6:

The different models being considered in this experiment are(1) :

Data	n	t	τ_w	n_w	std($\{\mu(\mathbf{y}_i)\}$)	std($\{\text{std}(\mathbf{y}_i)\}$)
electricity	370	25,968	24	7	$1.19e4$	$7.99e3$
traffic	963	10,392	24	7	$1.08e{-2}$	$1.25e{-2}$
wiki	115,084	747	14	4	$4.85e4$	$1.26e4$
PeMSD7(M)	228	11,232	9	160	3.97	4.42

Figure 6: The table depicts the statistics of the data-sets. τ_w denotes the number of time points in the window, n_w denotes the number of windows, $std(\mu(y_i))$ denotes the standard deviation of the means of the time series in the datasets and $std(std(y_i))$ denotes the standard deviation among the standard deviations of all the time time series in a dataset. Here we can see that wiki and traffic datasets have huge variations in scale(1).

1. DeepGLO : the newly proposed model
2. Local TCN: TCN network with LevelledInit initialization
3. LSTM: predictions are based on hidden states(6).
4. DeepAR : model discussed in paper (7)
5. TCN : model discussed in paper (8)
6. Prophet : Forecasting model from Facebook (15).
7. TRMF : model discussed in paper (9)
8. SVD + TCN : used as a baseline for global only approach. Here SVD and TCN models are combined together. The data factorization is done by SVD and then training done by TCN.
9. STGCN : Spatio temporal model which captures global features by using weighted graphs. It requires external inputs. (7; 16; 17).

The network and other hyper-parameters used in these models are mentioned in the Appendix.

For all the datasets and all the models, the experiment was carried out in normalized and unnormalized environments. This helps to determine the impact of scaling on the prediction accuracy. The error metrics used are MAPE, WAPE and SMAPE. All the models are initially trained on the train set and then predictions are done by rolling predictions method. For TRMF, re-training is done. For the PeMSD7(M) dataset , MAE/MAPE/RMSE comparison is done. All the models are trained for a maximum of 300 epochs (early stoppage also allowed).

11 Results

	Algorithm	electricity $n = 370$		traffic $n = 963$		wiki $n = 115,084$	
		Normalized	Unnormalized	Normalized	Unnormalized	Normalized	Unnormalized
Proposed	DeepGLO	0.133/0.453/0.162	**0.082**/0.341/**0.121**	0.166/ 0.210 /0.179	**0.148**/**0.168**/**0.142**	0.569/3.335/1.036	0.237/0.441/0.395
	Local TCN (LeveledInit)	0.143/0.356/0.207	0.092/**0.237**/0.126	0.157/**0.201**/0.156	0.169/0.177/0.169	**0.243**/**0.545**/**0.431**	**0.212**/**0.316**/**0.296**
	Global TCN-MF	0.144/0.485/0.174	0.106/0.525/0.188	0.336/0.415/0.451	0.226/0.284/0.247	1.19/8.46/1.56	0.433/1.59/0.686
Local-Only	LSTM	0.109/0.264/0.154	0.896/0.672/0.768	0.276/0.389/0.361	0.270/0.357/0.263	0.427/2.170/0.590	0.789/0.686/0.493
	DeepAR	**0.086**/**0.259**/ **0.141**	0.994/0.818/1.85	**0.140**/**0.201**/ **0.114**	0.211/0.331/0.267	0.429/2.980/0.424	0.993/8.120/1.475
	TCN (no LeveledInit).	0.147/0.476/0.156	0.423/0.769/0.523	0.204/0.284/0.236	0.239/0.425/0.281	0.336/1.322/0.497	0.511/0.884/0.509
	Prophet	0.197/0.393/0.221	0.221/0.586/0.524	0.313/0.600/0.420	0.303/0.559/0.403	-	-
Global-Only	TRMF (retrained)	0.104/0.280/0.151	0.105/0.431/0.183	0.159/0.226/ 0.181	0.210/ 0.322/ 0.275	0.309/0.847/0.451	0.320/0.938/0.503
	SVD+TCN	0.219/0.437/0.238	0.368/0.779/0.346	0.468/0.841/0.580	0.329/0.687/0.340	0.697/3.51/0.886	0.639/2.000/0.893

Algorithm	PeMSD7(M) (MAE/MAPE/RMSE)
DeepGLO (Unnormalized)	**3.53**/ **0.079** / **6.49**
DeepGLO (Normalized)	4.53/ 0.103 / 6.91
STGCN(Cheb)	3.57/0.087/6.77
STGCN(1^{st})	3.79/0.091/7.03

Figure 7: The tables compare the performance of the models on the datasets in both normalized and unnorimalized settings while performing rolling prediction taks. The error metrics depicted are WAPE/MAPE/SMAPE. For TRMF model, re training is done for each prediction window. Rest of the models are trained initially on the trainset and then used for prediction for all rolling windows. Prophet model was not able to handle the wiki dataset as it was too huge. The normalized column for DeepAR model indicates that the model was trained with scalar=true and unnormalized indicates scalar=false. The second table shows the performance on PeMSD7 dataset.

The results of the experiment can be observed from the two tables in figure 7. We can see that DeepGLO is one of the top two models for all metrics and all datasets. DeepGLO is better than Local TCN(LeveledInit) and global TCN-MF method as it uses both local and global properties. The local TCN with LeveledInit model performs the best on the wiki dataset with DeepGLO being the close second. We can see that there is more than 25 % improvement in accuracy with respect to the other models. Also DeepGLO performs the best in unnormalized settings as it is relatively scale free. Models like TCN, LSTM, DeepAR showcase poor performance in the same. DeepGLO also performs better than STGCN which requires an external input. Hence from the experiment the performance of DeepGLO was proved to be the best in comparison. The code for this experiment done by the authors have also been uploaded in github at `https://github.com/rajatsen91/deepglo`

12 Discussion and Future scope

This section details some of the drawbacks I have found in the paper (1) and some future areas of research possible. Though matrix factorization exploits the global information in the feature space, it is ineffective to capture complicated global temporal patterns. Many of the correlated time series would have such patterns which can be important for forecasting. Another drawback of this proposed model is that since the parameters are shared among all time series, they lack the capacity of modeling highly heterogeneous local temporal patterns. Also, while this model deals with the variations in scale, it does not help in changing the magnitude of scale as the LeveledInit scheme provides initialisation around the mean of past values. Hence the magnitude remains the same. The proposed DeepGLO model is based on only Y series (raw time series) , rather than on both the Y series and the latent factors X, F (which is obtained on matrix factorisation). Hence the information available in the latent factors are not being used.

Further research has been done in this field to mitigate some of the above mentioned issues. In paper (18), they use a Data Adaptive Graph Generation (DAGG) module to model the inter-dependencies among different time series. In another paper (19) , Spectral Temporal Graph Neural Network (StemGNN) model is proposed to capture temporal patterns and multivariate dependencies jointly in the spectral domain. But this area still remains open to further research.

13 Conclusion

From the results of the experiment, the newly proposed hybrid model called DeepGLO that thinks globally and acts locally shows high improvement in forecasting for high dimensional time series data sets. This model also utilizes both global and local temporal patterns during prediction and mitigates the issues that arise due to wide variations in scale between individual time series.

References

[1] Sen, Rajat, 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).

[2] Chris Chatfield. Time-series forecasting. Chapman and Hall/CRC, 2000.

[3] Kyoung-jae Kim. Financial time series forecasting using support vector machines. Neurocomputing, 55(1-2):307–319, 2003.

[4] Matthias W Seeger, David Salinas, and Valentin Flunkert. Bayesian intermittent demand forecasting for large inventories. In Advances in Neural Information Processing Systems, pages 4646–4654, 2016.

[5] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. Neural networks, 6(6):801–806, 1993.

[6] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[7] Valentin Flunkert, David Salinas, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks. arXiv preprint arXiv:1704.04110, 2017.

[8] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271, 2018.

[9] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. Temporal regularized matrix factorization for high- dimensional time series prediction. In Advances in neural information processing systems, pages 847–855, 2016.

[10] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In The 41st International ACM SIGIR Conference on Research Development in Information Retrieval, pages 95–104. ACM, 2018.

[11] Artur Trindade. Electricityloaddiagrams20112014 dataset. `https://archive.ics.uci.edu/ml/datasets/ElectricityL` 2011. [Online; accessed 07-Jan-2019].

[12] Marco Cuturi. Fast global alignment kernels. In Proceedings of the 28th international conference on machine learning (ICML-11), pages 929–936, 2011.

[13] Kaggle. Wikipedia web traffic. `https://www.kaggle.com/c/web-traffic-time-series-forecasting/data`, 2017. [Online; accessed 07-Jan-2019].

[14] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. Freeway performance measurement system: mining loop detector data. Transportation Research Record, 1748(1):96–102, 2001.

[15] Facebook. Fbprophet. https://research.fb.com/prophet-forecasting-at-scale/, 2017. [Online; accessed 07-Jan-2019].

[16] A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Türkmen, and Y. Wang. GluonTS: Probabilistic Time Series Modeling in Python. arXiv preprint arXiv:1906.05264, 2019.

[17] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875, 2017.

[18] Bai, Lei, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. "Adaptive graph convolutional recurrent network for traffic forecasting." Advances in neural information processing systems 33 (2020): 17804-17815.

[19] Cao, Defu, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong et al. "Spectral temporal graph neural network for multivariate time-series forecasting." Advances in neural information processing systems 33 (2020): 17766-17778.

A Appendix

A.1 Deep Leveled Network (DLN)

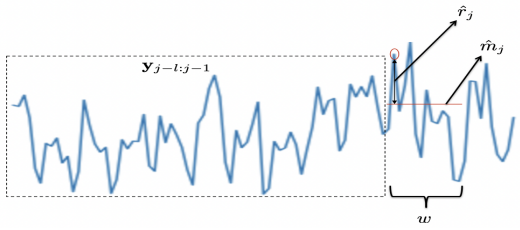


Figure 8: Deep Leveled Network(1)

In this paper, the authors have also proposed a levelling network where different data-sets having varying scales can be trained without the need for normalization. This Deep Leveled Network (DLN) consists of two temporally convoluted networks which are trained together. One of the networks (levelling component) will predict the rolling mean of future t time points with the given past. For a suitable window size, the rolling mean would remain consistent for a time series and can be predicted by the TCN. The other network (residual component) predicts the variations around the predicted mean value and hence this method is scale free and does not require normalization. Experiments where done using this network and the performance was found to be similar to the LeveledInit Scheme mentioned in Section 7.

A.2 Error metrics

As mentioned in Section 5, four error metrics are being used in the experiments conducted.

1) Weighted Absolute Percent Error (WAPE) - $L(\hat{Y}, Y) = \frac{\sum_{i=1}^{n'} \sum_{j=1}^{t'} |Y_{ij} - \hat{Y_{ij}}|}{\sum_{i=1}^{n'} \sum_{j=1}^{t'} |Y_{ij}|}$

2) Squared Error - $L(\hat{Y}, Y) = (\frac{1}{n\tau}) \left\| Y - \hat{Y} \right\|_F^2$

3) Mean Absolute Percent Error - $L_m(\hat{Y}, Y) = \frac{1}{Z_0} \sum_{i=1}^{n'} \sum_{j=1}^{t'} (|Y_{ij} - \hat{Y_{ij}}|/|Y_{ij}|) \mathbb{1}(|Y_{ij}| > 0)$

where $Z_0 = \sum_{i=1}^{n'} \sum_{j=1} \mathbb{1}(|Y_{ij}| > 0)$

4) Symmetric Mean Absolute Percent Error - $L_s(\hat{Y}, Y) = \frac{1}{Z_0} \sum_{i=1}^{n'} \sum_{j=1}^{t'} (2|Y_{ij} - \hat{Y_{ij}}|/|Y_{ij} + \hat{Y_{ij}}|) \mathbb{1}(|Y_{ij}| > 0)$

where $Z_0 = \sum_{i=1}^{n'} \sum_{j=1} \mathbb{1}(|Y_{ij}| > 0)$

A.3 Experimental Details

In this section, all the hyper parameters used in the models during experiment are mentioned. A TCN [a,b,c] indicates a network with 3 layers each having a, b and c number of filters in the corresponding layer. For an LSTM, the parameters (h,l) denotes the number of neurons in hidden layer(h) and number of hidden layers (l).

1. **DEEPGLO**: Kernel size - 7 , Network parameters -[32, 32, 32, 32, 32, 1] , 7 covariates. The rank k for electricity, traffic , wiki and PeMSD7(M) datasets are - [64,64,256,64].
2. **Local TCN (LeveledInit)**: Network params [32,32,32,32,32,1]
3. **LSTM**: Network parameters (45 - neurons in hidden layer , 3 hidden layers)
4. **DeepAR**: Default parameters in DeepAREstimator in GlutonTS implementation are being used here also.
5. **TCN**: Network parameters [32,32,32,32,32,1], kernel size - 7
6. **Prophet**: Parameters are selected automatically, growth is set as 'logistic', parallelisation using 32 cores is done.
7. **TRMF**: Rank of k [60,60,1024] for electricity, traffic and wiki datasets
8. **SVD + TCN**: Rank of k[60,60,500] for electricity, traffic and wiki datasets, kernel size of 7 and network parameters [32,32,32,32,1]
9. **STGCN**: Kernel size is 1 for STGCN(1st) and 3 for STGCN(Cheb), network parameters are [64,16,64]