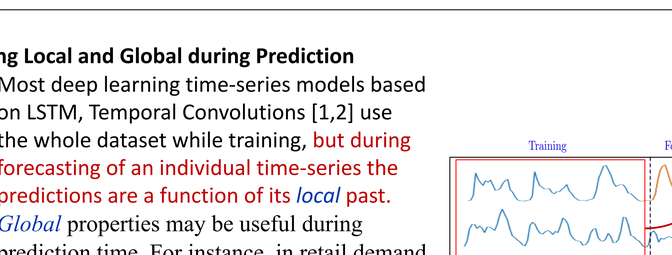
Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting

Rajat Sen[&], Hsiang-Fu Yu[&] and Inderjit Dhillon[#,&]

[&]Amazon, [#]University of Texas at Austin

High-Dimensional Time-Series Forecasting

- Applications like item demand forecasting, web-traffic forecasting can have millions of correlated time-series evolving together e.g forecasting future demand of items for a retailer like Amazon

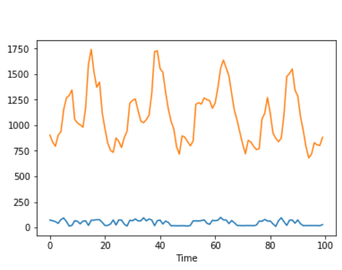


- Additionally there can be covariates at each time point, which are either common to all time-series (time of day, day of week etc.) or are unique to each time-series.

Key Challenges in Deep Temporal Models

Issues with Normalization

- The individual time-series may have vastly different scales. For instances, in demand forecasting some popular items may have orders of magnitude more demand, than niche items.
- This creates issues in training deep models based on LSTM or Temporal Convolution Networks (TCN). The datasets need to be normalized and the choice of scaling parameters can have impact on performance.
- For instance, [1] uses standard normalization, while [2] scales all the time-series with the value in their first time-index.



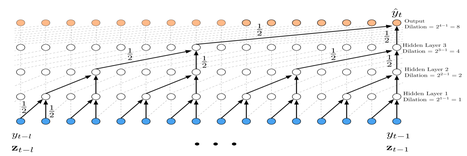
Two time-series from the electricity dataset [4] with very different scales

Using Local and Global during Prediction

- Most deep learning time-series models based on LSTM, Temporal Convolutions [1,2] use the whole dataset while training, but during forecasting of an individual time-series the predictions are a function of its *local* past.
- *Global* properties may be useful during prediction time. For instance, in retail demand forecasting, past values of similar items can be leveraged while predicting the future for a certain item.
- TRMF [3] can express all the time-series as linear combinations of basis time-series. These basis time-series can capture global patterns during prediction. However, TRMF can only model linear temporal dependencies.

LeveledInit: Handling Diverse Scales with TCN

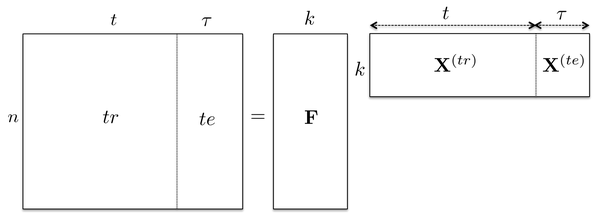
- In order to handle datasets, with a lot of variation in scale among different time-series, we propose as simple initialization scheme for Temporal Convolution Networks (TCN).
- **LeveledInit:** We recommend setting the filter weights equal to $1/k$, where k is the filter size, in all layers except the input layer. In the input layer, all weights in the channel that takes in the original time-series are set to $1/k$, while all other weights are initialized to 0.
- We show that under some configurations of the TCN (filter size, dilation) LeveledInit leads to the network initially predicting the average of the past values in the dynamic range of the network, and then the network learns to predict variations around this mean value. The variations around the mean are relatively scale free and can alleviate the problems with scale.



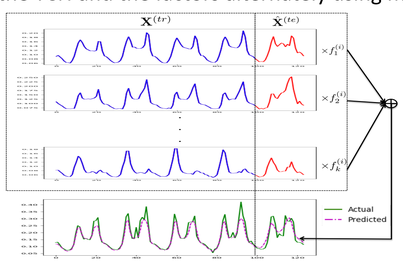
The base image has been borrowed from [4]

TCN-MF: Our Global Model

- Similar to [3], the idea is to decompose the training matrix $\mathbf{Y}^{(tr)}$ into factors $\mathbf{F}$ and $\mathbf{X}^{(tr)}$ such that $\mathbf{X}^{(tr)}$ has temporal features. In other words $\mathbf{X}^{(tr)}$ consists of k basis time-series which can be predicted into the future, where the rank $k \ll n$.
- In order to compute non-linear temporal dependencies we use a TCN (trained concurrently) to regularize $\mathbf{X}^{(tr)}$ during the factorization. This encourages temporal structure in $\mathbf{X}^{(tr)}$.



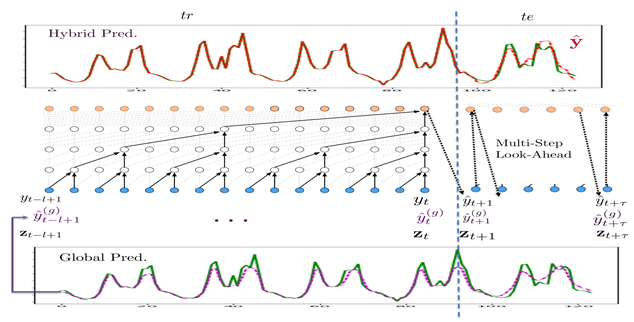
- Temporal regularization by TCN: $\mathcal{R}(\mathbf{X}^{(tr)}|\mathcal{T}_{\mathcal{X}}(\cdot)) := \frac{1}{|\mathcal{J}|}\mathcal{L}(\mathbf{X}[:,\mathcal{J}],\mathbf{X}[:,\mathcal{J}-1])$ where $\mathcal{J} = \{l,...,t\}$ and $\mathcal{L}(.,.)$ is the squared loss.
- Here, $\mathcal{T}_{\mathcal{X}}(\cdot)$ is a TCN that is trained concurrently. Thus this regularization ensures $\mathbf{X}^{(tr)}$ is such that , there exists a TCN, that can predict $\mathbf{X}^{(tr)}$ based on past time-points.
- Thus the global loss being minimized is: $\mathcal{L}(\mathbf{Y}^{(tr)}, \mathbf{F}\mathbf{X}^{(tr)}) + \lambda_{\mathcal{T}}\mathcal{R}(\mathbf{X}^{(tr)}|\mathcal{T}_{\mathcal{X}}(\cdot))$
- We train the TCN and the factors alternately using mini-batch SGD (more details in paper)



In this figure, we show some of the basis time-series extracted from the traffic dataset, which can be combined linearly to yield individual original time-series. It can be seen that the basis series are highly temporal and can be predicted in the test range using the network $\mathcal{T}_{\mathcal{X}}(\cdot)$

DeepGLO: Deep Global/Local Model

- Our final hybrid model, is a TCN, where the output of the global model is fed as a covariate, along with other covariates.
- Thus, DeepGLO can combine local per time-series properties along with the global predictions from the TCN-MF model.



The base image has been borrowed from [4]

Empirical Results

- We compare DeepGLO with other state of the art time-series models on rolling validation tasks on four public datasets depicted below.

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

The four datasets are electricity, traffic, wiki and pems. std($\{\mu\}$) denotes the standard deviation among the means of all the time series in the data-set and the next column shows the same for St. deviation.

	Algorithm	electricity $n = 370$ Normalized	electricity $n = 370$ Unnormalized	traffic $n = 963$ Normalized	traffic $n = 963$ Unnormalized	wiki $n = 115,084$ Normalized	wiki $n = 115,084$ 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

Comparison of algorithms on normalized and unnormalized versions of data-sets on rolling prediction tasks The error metrics reported are WAPE/MAPE/SMAPE. TRMF is retrained before every prediction window, during the rolling predictions. All other models are trained once on the initial training set and used for further prediction for all the rolling windows. Prophet could not be scaled to the wiki dataset, even though it was parallelized on a 32 core machine. Note that for DeepAR normalized setting corresponds to scaler=True and unnormalized setting is scaler=False, in the GluonTS implementation. In the bottom table DeepGLO is compared with the models in [5] on the pems dataset.

References

1. Valentin Flunkert, David Salinas, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks. arXiv preprint arXiv:1704.04110, 2017.
2. Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. arXiv preprint arXiv:1703.04691, 2017.
3. Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. Temporal regularized matrix factorization for highdimensional time series prediction. In Advances in neural information processing systems, pages 847–855, 2016.
4. A¨aron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. CoRR abs/1609.03499, 2016.
5. Yu, Bing, Haoteng Yin, and Zhanxing Zhu. "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting." *arXiv preprint arXiv:1709.04875* (2017).