Spatiotemporal Residual Regularization with Kronecker Product Structure for Traffic Forecasting

Seongjin Choi, Nicolas Saunier, Martin Trepanier, Lijun Sun

Abstract

Existing deep learning-based traffic forecasting models are often trained with MSE as the loss function, which is equivalent to assuming the residual/error to follow independent Gaussian distribution for simplicity of modeling. However, this assumption does not hold especially in traffic forecasting tasks, where the residuals are correlated in both spatial and temporal dimensions. For a multistep forecasting model, we would also expect the variance to increase with the number of steps. In this study, we propose a SpatioTemporal Residual Regularization based on the assumption that the residuals follow a zero-mean multivariate Gaussian distribution with a learnable spatiotemporal covariance matrix. This approach benefits from directly considering correlated spatiotemporal residuals. However, it suffers from scalability issues since the spatiotemporal covariance is often large. For model scalability, we model the spatiotemporal covariance as a sum of Kronecker products of spatial and temporal residual covariance, which significantly reduces the number of parameters and computation complexity. The performance of the proposed method is tested on a traffic speed forecasting task, and the results show that the proposed method improves the model performance by properly dealing with correlated residuals.

1 Introduction

The unprecedented availability of traffic data and significant advancements in data-driven algorithms have sparked considerable interest and developments in traffic forecasting. In particular, state-of-the-art deep learning models such as DCRNN [1], STGCN [2], Graph-Wavenet [3], and Traffic Transformer [4], have the best performance for traffic speed forecasting. Deep learning approaches have shown a clear advantage of capturing complex non-linear relationships in traffic forecasting. Graph Neural Networks (GNNs), Recurrent Neural Networks (RNNs), Gated Convolutional Neural Networks (Gated-CNNs), and Transformer are generally employed to capture spatiotemporal patterns in traffic data [5, 6, 7, 8].

Conventional deep learning models for traffic forecasting use Mean Squared Error (MSE) or Mean Absolute Error (MAE) as a loss function to train the neural network. This convention of using MSE or MAE is based on the assumption that residuals (the deviation between the prediction and the actual value) follow either *uncorrelated* independent Gaussian or Laplacian distributions. On the other hand, recent studies present evidence that the assumption is not always true. Sun et al. [9] claimed that in most cases prediction errors of time series prediction models are actually autocorrelated due to the temporality of the data. In this

2022 NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems.

work, authors proposed a method to adjust the autocorrelated errors. Also, Kim et al. [10] analyzed the errors in the traffic forecasting models and found that the errors are correlated in both spatial and temporal dimensions. The authors thus proposed a simple residual estimation module to predict the expected residuals.

In this study, we propose a novel approach to deal with *correlated* spatiotemporal residuals. We assume that the distribution of the spatiotemporal residuals follows a multivariate Gaussian distribution, and train the model and the spatiotemporal covariance matrix by maximizing the likelihood. This approach benefits from directly considering correlated spatiotemporal residuals. However it suffers from scalability issue because the spatiotemporal covariance matrix is too large. As a result, we design a decomposition technique to efficiently deal with the issue. The spatiotemporal covariance is reparameterized as a sum of Kronecker products of spatial and temporal covariance matrices, and, as a result, the computation of the likelihood function is much simpler. We name this approach *SpatioTemporal Residual Covariance Regularization (STRR)*, and use it as an auxiliary loss to train the model with conventional MSE or MAE loss.

2 Methodology

2.1 Problem Formulation

In this section, we briefly give out the formulation of the problem of interest. Given a time stamp t, we denote the average speeds over a time interval (from t to $t + \Delta t$) of the n-th sensor as v_t^n. The speed observations in the whole network at the given time stamp are denoted as an N-dimensional vector, $x_t = \left[v_t^1, v_t^2, \cdots, v_t^N\right]^\top$. Given the K historical observations, $X_t = [x_{t-K+1}, \cdots, x_t] \in \mathbb{R}^{N \times K}$, we aim to predict the speed of N sensors over multiple timesteps T (i.e. spatiotemporal speed matrix), $Y_t \in \mathbb{R}^{N \times T}$.

2.2 Spatiotemproal Residual Regularization with Kronecker Product Structure

In this study, we propose a SpatioTemporal Residual Regularization to improve the performance of conventional deep learning models for short-term traffic forecasting. Unlike conventional models which assume that the residuals, the difference between target value and predicted value, follow independent Gaussian distributions, we assume that the residuals follow a multivariate zero-mean Gaussian distribution with a learnable spatiotemporal covariance matrix (Σ). From a given input X_t, we predict the mean $M_t \in \mathbb{R}^{N \times T}$ as well as residual $R_t \in \mathbb{R}^{N \times T}$ with

$$Y_t = M_t + R_t, \;\; \mathrm{vec}(R_t) \sim \mathcal{N}(0, \Sigma), \tag{1}$$

where $\mathrm{vec}(R_t) \in \mathbb{R}^{NT}$ is a vectorized form of R_t. By marginalizing R_t, the above specification corresponds to $\mathrm{vec}(Y_t) \sim \mathcal{N}(M_t, \Sigma)$.

Then, we can train a given deep learning model and the spatiotemporal covariance matrix by using the following loss function.

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{MSE} + \rho \mathcal{L}_{STRR}, \\ \mathcal{L}_{MSE} &= \|Y_t - M_t\|_2^2, \\ \mathcal{L}_{STRR} &= -\frac{1}{2}\log|\Sigma| - \frac{1}{2}\mathrm{vec}(R_t)^\top \Sigma^{-1} \mathrm{vec}(R_t), \end{aligned} \tag{2}$$

where $\mathcal{L}_{MSE}$ is the Mean Squared Error, $\mathcal{L}_{STRR}$ is the SpatioTemporal Residual Regularization loss (i.e. the negative log likelihood of the residuals), and ρ is the regularization parameter.

Directly training such a large spatiotemporal covariance matrix ($\Sigma \in \mathbb{R}^{NT \times NT}$) is infeasible if either N or T is large since computing the negative log-likelihood includes inverse operation and determinant operation of the covariance matrix. For example, PEMS-BAY data, which is one of the widely used traffic speed dataset, consists of observations from 325 sensors, and the prediction horizon is usually set as 12 timestep with 5 minute interval. This would result in $N = 325$ and $T = 12$, which makes $\Sigma \in \mathbb{R}^{3900 \times 3900}$.

Instead of inefficiently training such a large matrix, we propose a decomposition method to address the scalability issue. First, we decompose the residual (R_t) to a sum of unit residuals, i.e., $R_t = \sum_{k=1}^{K} E_t^k$. Then we reparameterize the spatiotemporal covariance matrix as a sum of Kronecker products of spatial and temporal covariance matrices. For convenience, we use the precision matrix (inverse of covariance matrix) for notations:

$$\begin{aligned} &Y_t = M_t + R_t = M_t + \sum_{k=1}^{K} E_t^k \\ &\mathrm{vec}(E_t^k) \sim \mathcal{N}(0, \Sigma^k), \ \text{ for } k = 1, \ldots, K, \\ &\Sigma^k = \Sigma_S^k \otimes \Sigma_T^k = \left(\Lambda_S^k\right)^{-1} \otimes \left(\Lambda_T^k\right)^{-1} \end{aligned} \tag{3}$$

where $\Sigma^k \in \mathbb{R}^{NT \times NT}$ is the spatiotemporal covariance matrix for the unit residual $\mathrm{vec}(E_t^k)$. We denote Σ_S^k and Σ_T^k as spatial and temporal covariance for variable E_t^k, respectively, and Λ_S^k and Λ_T^k as spatial and temporal precision (inverse of covariance), respectively. In training the neural network, we let the E_t^k for $k = 1, \ldots, K-1$ to be the model output and then define the last residual, or the K-th residual, to be the final deviation $E_t^K = Y_t - M_t - \sum_{k=1}^{K-1} E_t^k$.

This representation is equivalent to assuming the unit residual (E_t^k) follows a matrix normal distribution as follows:

$$E_t^k \sim \mathcal{MN}(0, \Sigma_S^k, \Sigma_T^k) = \mathcal{MN}(0, \left(\Lambda_S^k\right)^{-1}, \left(\Lambda_T^k\right)^{-1}), \quad \forall t \tag{4}$$

Then the STRR loss can be calculated as follows:

$$\mathcal{L}_{STRR} = \sum_{k=1}^{K} \left(N \log \left|\Lambda_T^k\right|^{\frac{1}{2}} + T \log \left|\Lambda_S^k\right|^{\frac{1}{2}} - \frac{1}{2} \operatorname{Tr}\left[\Lambda_T^k \left(E_t^k\right)^\top \Lambda_S^k E_t^k\right] \right) \tag{5}$$

For ease of computation, we parameterize both spatial and temporal precision matrices using Cholesky factorization as follows:

$$\Lambda_S^k = (L_S^k)\left(L_S^k\right)^\top, \quad \Lambda_T^k = (L_T^k)\left(L_T^k\right)^\top \tag{6}$$

where L_S^k and L_T^k are lower triangular matrices of size $N \times N$ and $T \times T$, respectively. Also, due to associative property of scalar multiplication of the Kronecker product (i.e. $A \otimes B = (\nu A) \otimes (\frac{1}{\nu} B)$), we fix the first entry of L_T^k (element at first row and first column) to be 1. As a result, we have $\left(\frac{N(N+1)}{2}\right)$ parameters for L_S^k and $\left(\frac{T(T+1)}{2} - 1\right)$ parameters for L_T^k [1].

Finally, Equation 5 can be simplified as follows:

$$\mathcal{L}_{STRR} = \sum_{k=1}^{K} \left(N \sum_{m=1}^{T} \log[L_T^k]_{m,m} + T \sum_{n=1}^{N} \log[L_S^k]_{n,n} - \frac{1}{2} \|Q_t^k\|_F^2 \right), \tag{7}$$

where $Q_t^k = \left(L_S^k\right)^\top E_t^k \left(L_T^k\right)$ (see Appendix A for details).

3 Results

We conduct experiments to verify the performance of the proposed model. The dataset we use is called **PEMS-BAY**, which is one of the most widely-used traffic datasets in traffic forecasting. It was collected from the California Transportation Agencies (CalTrans) Performance Measurement System (PeMS) [11]. It is collected from 325 sensors installed in the Bay Area with observations of 6 months of data ranging from Jan 1st 2017 to May 31th 2017 [1]. Also, we collected a different set of data from the same site with observations

[1]Directly using Equation 1 would have $\left(\frac{NT(NT+1)}{2}\right)$ parameters.

Table 1: Summary of results

Data	Model	RMSE				MAPE				MAE			
		15min	30min	45min	60min	15min	30min	45min	60min	15min	30min	45min	60min
PEMS-BAY (2017)	GWN[3]	2.74	3.70	-	4.52	2.73	3.67	-	4.63	1.30	1.63	-	1.95
	GWN*	2.37	**3.13**	**3.58**	3.91	2.19	3.01	3.62	4.11	1.17	1.55	1.83	2.08
	+STRR(1)	2.38	3.14	**3.58**	3.90	2.19	3.05	3.62	4.08	1.15	1.51	1.75	1.95
	+STRR(3)	**2.36**	3.14	3.59	**3.85**	2.11	2.93	3.50	4.04	**1.12**	**1.47**	1.71	1.91
	+STRR(5)	2.37	3.21	3.69	4.03	2.14	2.94	3.52	3.96	1.14	1.49	1.74	1.94
	+STRR(7)	2.38	3.18	3.61	3.96	**2.10**	2.91	3.48	3.91	**1.12**	1.48	1.72	1.90
	+STRR(10)	**2.36**	3.17	3.63	3.94	2.11	**2.90**	**3.46**	**3.88**	**1.12**	**1.47**	**1.70**	**1.89**
PEMS-BAY (2022)	GWN*	2.33	3.05	3.44	3.72	2.21	3.05	3.62	4.05	1.16	1.50	1.74	1.93
	+STRR(1)	2.23	2.94	3.32	3.58	2.13	3.03	3.52	3.91	1.08	1.40	1.58	1.73
	+STRR(3)	**2.20**	**2.89**	**3.27**	3.58	**2.06**	**2.83**	3.34	3.94	**1.06**	**1.35**	1.53	1.73
	+STRR(5)	2.21	**2.89**	3.28	3.53	2.09	**2.83**	3.32	3.68	1.07	**1.35**	1.53	1.67
	+STRR(7)	2.21	2.90	**3.27**	3.53	2.07	**2.83**	3.32	3.67	1.07	**1.35**	1.53	1.67
	+STRR(10)	2.21	2.90	**3.27**	**3.52**	2.07	**2.83**	**3.31**	**3.66**	1.07	**1.35**	**1.52**	**1.66**

of 6 months ranging from Jan 1st 2022 to May 31th 2022. For convenience, we denote the original PEMS-BAY data as **PEMS-BAY(2017)** and the new one as **PEMS-BAY(2022)**[2].

We choose Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and Mean Absolute Error (MAE) of speed prediction at 15, 30, 45, 60 minutes ahead as key metrics to evaluate prediction accuracy.

The proposed method can work as an add-on to any type of traffic forecasting model based on deep learning. We use Graph Wavenet (**GWN**) [3] as our baseline model. The original model in Wu et al. [3] aims to predict 12 step with 5 minute interval with Mean Absolute Error loss. For simplicity, we used 4 step prediction with 15 minute interval with Mean Squared Error as shown in Equation 2, and GWN model with this setting is denoted as **GWN***. In addition, we use different settings for R, the number of unit residuals, and we denote STRR model as **STRR(r)** where $r \in \{1, 3, 5, 7, 10\}$.

Table 1 shows the summary of results. Generally speaking, applying STRR loss for training improves the model performance in most cases for both datasets. Especially, it is notable that applying STRR improves the performance of MAPE, since MAPE penalizes large errors in congested traffic states (when speed is low). Most of the correlated residuals occur when the traffic state is in congested state [12], while the residuals in free flow state usually follow independent Gaussian distribution.

4 Conclusion

In this study, we propose SpatioTemporal Residual Regularization to deal with correlated spatiotemporal residuals. The proposed method designs the distribution of residual as a multivariate Gaussian distribution with a covariance that can be represented as a sum of Kronecker products of spatial and temporal covariance matrices. The proposed method can be used as an add-on module to any type of traffic forecasting model based on deep learning. In this study, to evaluate the performance of proposed methodology, we apply STRR to Graph Wavenet, which is one of the state-of-the-art used traffic forecasting model based on deep learning. The results show that the proposed method can improve the prediction accuracy. Finally, it is worth mentioning that the proposed method can be applied to other types of spatiotemporal forecasting such as electricity and energy consumption forecasting and wind speed forecasting.

[2]https://github.com/benchoi93/PeMS-BAY-2022

References

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[2] 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.

[3] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121*, 2019.

[4] Ling Cai, Krzysztof Janowicz, Gengchen Mai, Bo Yan, and Rui Zhu. Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. *Transactions in GIS*, 24(3):736–755, 2020.

[5] Li Li, Xiaonan Su, Yi Zhang, Yuetong Lin, and Zhiheng Li. Trend modeling for traffic time series analysis: An integrated study. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3430–3439, 2015.

[6] 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.

[7] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 21(11):4883–4894, 2019.

[8] Fuqiang Liu, Jiawei Wang, Jingbo Tian, Dingyi Zhuang, Luis Miranda-Moreno, and Lijun Sun. A universal framework of spatiotemporal bias block for long-term traffic forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[9] Fan-Keng Sun, Chris Lang, and Duane Boning. Adjusting for autocorrelated errors in neural networks for time series. *Advances in Neural Information Processing Systems*, 34: 29806–29819, 2021.

[10] Daejin Kim, Youngin Cho, Dongmin Kim, Cheonbok Park, and Jaegul Choo. Residual correction in real-time traffic forecasting. *arXiv preprint arXiv:2209.05406*, 2022.

[11] 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.

[12] Guopeng Li, Victor L Knoop, and Hans van Lint. Estimate the limit of predictability in short-term traffic forecasting: An entropy-based approach. *Transportation Research Part C: Emerging Technologies*, 138:103607, 2022.

Appendix A Parameterization

For ease of computation, we parameterize both spatial and temporal precision matrices using Cholesky factorization. In this way, we can avoid computing inverse operation during the training, and efficiently compute the determinant by summing the diagonal entries as shown in Equation 8. Also, the trace term can be simplified into computing the square of Frobenius Norm of $Q_t^k = \left(L_S^k\right)^\top E_t^k \left(L_T^k\right)$ as shown in Equation 9.

$$\begin{aligned}
&\Lambda_S^k = (L_S^k)\left(L_S^k\right)^\top, \quad \Lambda_T^k = (L_T^k)\left(L_T^k\right)^\top, \\
&\log\left|\Lambda_S^k\right|^{\frac{1}{2}} = \sum_{n=1}^{N} \log[L_S^k]_{n,n}, \\
&\log\left|\Lambda_T^k\right|^{\frac{1}{2}} = \sum_{m=1}^{T} \log[L_T^k]_{m,m}.
\end{aligned} \tag{8}$$

$$\begin{aligned}
\mathrm{Tr}\left[\Lambda_S^k \left(E_t^k\right)^\top \Lambda_T^k E_t^k\right] &= \mathrm{Tr}\left[(L_T^k)\left(L_T^k\right)^\top \left(E_t^k\right)^\top (L_S^k)\left(L_S^k\right)^\top E_t^k\right] \\
&= \mathrm{Tr}\left[\left(L_S^k\right)^\top E_t^k (L_T^k)\left(L_T^k\right)^\top \left(E_t^k\right)^\top (L_S^k)\right] \\
&= \mathrm{Tr}\left[\left(\left(L_S^k\right)^\top E_t^k (L_T^k)\right)\left(\left(L_S^k\right)^\top E_t^k (L_T^k)\right)^\top\right] \\
&= \mathrm{Tr}\left[\left(Q_t^k\right)\left(Q_t^k\right)^\top\right] \\
&= \|Q_t^k\|_F^2,
\end{aligned} \tag{9}$$