



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Master's Thesis

Learning Basis Functions of Deep Spatio-Temporal  
Neural Networks with Covariance Loss

Ji-Woo Lee

Department of Computer Science and Engineering  
Computer Science and Engineering

Graduate School of UNIST

2020

# Learning Basis Functions of Deep Spatio-Temporal Neural Networks with Covariance Loss

Ji-Woo Lee

Department of Computer Science and Engineering  
Computer Science and Engineering

Graduate School of UNIST

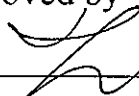
# Learning Basis Functions of Deep Spatio- Temporal Neural Networks with Covariance Loss

A thesis/dissertation  
submitted to the Graduate School of UNIST  
in partial fulfillment of the  
requirements for the degree of  
Master of Science

Jiwoo Lee

06/17/2020

Approved by



---

Advisor

Kwang In Kim

# Learning Basis Functions of Deep Spatio- Temporal Neural Networks with Covariance Loss

Jiwoo Lee

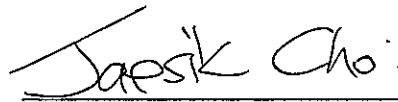
This certifies that the thesis/dissertation of Jiwoo Lee is approved.

06/17/2020



---

Advisor: Kwang In Kim



---

Committee Member : Jaesik Choi



---

Committee Member : Sungphil Kim

## Abstract

Gaussian prior on parameters of a neural network results in Gaussian prior on the functionality of the network. The goal of this paper is to guide the Gaussian prior of the network to converge true distributions of target variables. To achieve this goal, we suggest a novel loss function that additionally penalizes the learning of a neural network according to the discrepancy between covariance matrices target values and the last hidden layer of a neural network. Unlike to the conventional loss function that only captures the first moment of the target values, the proposed loss function further captures the second moment of the target values and leads to the better fit to the true distribution, which in turn, facilitates approaching to better prediction. To demonstrate the validity of the proposed loss function, We experimented with various datasets for different models., Spatio-Temporal Graph Convolution Network (STGCN) and Graph WaveNet (GWNENET) on popular benchmark datasets, PeMSD7 and METR-LA.

## Contents

<b>Contents</b>		<b>iv</b>
<b>List of Figures</b>		<b>vi</b>
<b>List of Tables</b>		<b>vii</b>
<b>I. Introduction</b>		<b>1</b>
<b>II. Background</b>		<b>3</b>
2.1 Deep Neural Networks . . . . .		3
2.1.1 Spatio-Temporal Graph Convolutional Network (STGCN) . . . . .		3
2.2 Gaussian Processes . . . . .		5
<b>III. Related works</b>		<b>8</b>
<b>IV. Spatio-Temporal Covariance loss</b>		<b>9</b>
4.1 Spatio-Temporal Covariance loss . . . . .		9
4.1.1 Learning the Basis Functions in STGCN . . . . .		9
4.1.2 Spatio-Temporal Covariance Loss . . . . .		10
4.1.3 Principles of Spatio-Temporal Covariance Loss Based Neural Networks . . . . .		11
<b>V. Experimental Evaluations</b>		<b>13</b>
5.1 Experimental Evaluations . . . . .		13
5.1.1 Experimental setups . . . . .		13
5.1.2 Learning Basis functions . . . . .		14
5.2 Effect of the basis functions on PeMSD7 dataset . . . . .		16
5.3 Effect of the basis functions on METR-LA dataset . . . . .		18
5.4 Noise robustness . . . . .		20
5.4.1 Case studies on the effect of noisy nodes . . . . .		21
5.5 Comparisons with L1 and L2 regularizations . . . . .		22

<b>VI. Conclusion</b>	<b>24</b>
<b>References</b>	<b>25</b>



## List of Figures

4.1	An architecture of a Spatio-Temporal Graph Convolutional Network with Covariance Loss. . . . .	10
5.1	<b>Learning variance of target variables:</b> with the proposed loss, diagonal elements of basis inner product ( $\sigma\phi_i\phi_i$ ) indicate variances of target variables. In contrast to STGCN (left), STGCN-Cov (right) successfully learns such basis functions. . . . .	14
5.2	<b>Learning covariance of target variables:</b> non-diagonal elements of basis inner product ( $\sigma^2\phi_i\phi_j$ ) mean covariance of target variables. Regardless of how the two different variables are related, independent (left, Node 52 and 129) or correlated (right, Node 1 and 7), STGCN-Cov successfully learns such basis functions. . . . .	14
5.3	Predictions and basis of STGCN (left) and STGCN-Cov (right) on Node 7 of PeMSD7 . . . . .	16
5.4	Prediction and basis of STGCN (left) and STGCN-Cov (right) on Node 110 of PeMSD7 . . . . .	17
5.5	Training and test dataset of node 110 . . . . .	17
5.6	Prediction and basis of GWNET (left) and GWNET-Cov (right) on Node 6 of METR-LA . . . . .	18
5.7	RMSE increase v.s. the number of noisy node . . . . .	20
5.8	Difference of predicted value (RMSE) as the noisy node increases . . . . .	20
5.9	The effect of noisy nodes on prediction results . . . . .	21
5.10	RMSE comparison of STGCN with L1, L2 and Covariance loss for prediction step 3. . . . .	23
5.11	RMSE comparison of STGCN with L1, L2 and Covariance loss for prediction step 6. . . . .	23
5.12	RMSE comparison of STGCN with L1, L2 and Covariance loss for prediction step 9. . . . .	23

## List of Tables

5.1	Accuracy Comparison for PeMSD7 dataset . . . . .	16
5.2	Accuracy Comparison for METR-LA dataset . . . . .	18

---

## List of Abbreviations

---

**DNN** Deep Neural Network. 3

**GWNET** Graph WaveNet. iii, 2

**MSE** mean squared error. 11

**RKHS** reproducing kernel Hilbert space. 8, 11

**SE** squared exponential. 6

**STGCN** Spatio-Temporal Graph Convolution Network. iii, 2

---

# Introduction

---

Goals of regression problems are to find relationships or functions from input features to continuous outputs (or target variables), such as weather forecasting or stock price prediction. In theory, Gaussian Processes (GPs) can represent arbitrary smooth functions which satisfy mean square differentiability and the continuity. [1]. GPs have a close relationship with neural networks. GPs can incorporate the neural networks as a kernel function approximation [2–4]. If Neural networks have infinite numbers of hidden units, can represent GPs [5]. These relationships motivate various compositions of GPs and neural networks. For example, a GP kernel is derived for multi-hidden-layer neural networks with general non linearity based on signal propagation theory [6]. A feature map of deep neural network is used to inference GP kernel for regression tasks [7].

In this paper, we aim at guiding a neural network to learn the underlying function with Gaussian prior. We can regard that a neural network for regression problem consists of a feature extraction unit that is a series of non-linear mapping and a linear regression unit that uses the feature as a input for final predictions. From the Bayesian view on neural networks, assuming Gaussian prior on parameters on the linear regression unit results in Gaussian prior on the output of the regression unit [5]. This indicates that the inner product of input of the regression unit should converge to covariance of sampled target variables and as the number of sample increases, the covariance will eventually converge to true covariance of target variables. This motivates us to devise a new loss function, *Spatio-Temporal Covariance loss* that measures the discrepancy

between covariance matrices of sampled target values and their corresponding output of the feature extraction unit. Another nature of exploiting the covariance matrix is that the learning of a network consider spatial and temporal dependencies among variables. To take spatial and temporal dependencies into the learning scheme, we employ state-of-art models that consider spatial and temporal locality, STGCN [8] and GWNET [9].

---

# Background

---

## 2.1 Deep Neural Networks

Deep Neural Network (DNN) is a non-linear mapping of  $\mathbf{x}$  to  $\mathbf{y}$  with stacked transformation functions. Typically, each layer of DNN non-linearly maps the input with matrix multiplication and activation function such as logistic sigmoid, tanh, and relu [10]. Then, we can represent equations of deep neural network from the first layer to the output layer as like this,

$$\begin{aligned}
 f_1(\mathbf{x}) &= g(W_1\mathbf{x} + b_1), \\
 \Phi(\mathbf{x}) &= g(W_{L-1}(f_{L-2}(\dots f_1(\mathbf{x})) + b_{L-1}), \\
 \mathbf{y} &\approx \Phi(\mathbf{x})^T W_L + b_L.
 \end{aligned}
 \tag{II.1}$$

in which,  $W_i$  and  $b_i$  indicate weight and bias for  $i^{th}$  layer map,  $f_i$ , while  $g$  means non-linear activation function.  $\Phi(\mathbf{x})$  represent the output of the last hidden layer and it is a set of features generated from  $\mathbf{x}$  that has a crucial role for learning the best mapping from  $\mathbf{x}$  to  $\mathbf{y}$  and thus, is referred as basis function in the rest of this paper.

### 2.1.1 Spatio-Temporal Graph Convolutional Network (STGCN)

STGCN [8] is a variant of Graph Convolutional Network (GCN) [11] that generalizes traditional convolution operation onto graph structured data by defining a graph convolution opera-

tor. The traditional convolution operator considers spatial locality specified by filters to extract meaningful features on grid-shaped data structures. The graph convolution operates similarly and spatial locality is specified by a *graph laplacian* matrix. The graph laplacian matrix specifies neighbor relations between nodes in a graph and how strong the relations are. The graph laplacian matrix, ( $L$ ) is defined as follows.

$$(Lf)(i) = \sum_{j \in N_i} W_{i,j}(f(i) - f(j)), \quad (\text{II.2})$$

where,  $N_i$  and  $W_{i,j}$  indicate neighbors of node  $i$  and weights between node  $i, j$ . As shown in the Equation II.2, with graph laplacian matrix, it is able to infer node  $i$ 's signal,  $f(i)$  by aggregating the neighbor nodes' signals,  $f(j)$ , which is analogous to the traditional convolution operation that consider spatial locality to extract meaningful features.

Another interesting nature of graph laplacian matrix is that the matrix is eigen-decomposable so that we have  $L = U\Lambda U^T$  if the matrix is symmetric, i.e. the graph is undirected. Thus, a feature extraction,  $h = f_\theta(L)x$ , becomes filtering at spectrum domain (a whole set of eigen-vectors),  $h = Uf_\theta(\Lambda)U^T x$  [12]. Notice that aforementioned graph convolutions are not fully utilizing locality (only 1-hop neighbor nodes are included in the estimation) or computationally expensive (additional matrix multiplications for  $U$  and  $U^T$  are required). One well-known solution for the limitations is to employ polynomial approximations such as Chebyshev polynomials [12] as follows,

$$f_\theta(L)\mathbf{x} \approx \sum_{k=0}^K \theta_k T^k(\bar{L})\mathbf{x}, \quad (\text{II.3})$$

where,  $\bar{L}$  indicates  $2L/\lambda_{max} - I_N$ . With  $k$ -order Chebyshev expansion, we can consider signals of  $k$ -hop neighbors for feature extractions.

GCN [11] introduces a first order approximation on chebyshev expansion (Equation II.3). Assuming  $K = 1$  and  $\lambda_{max} = 2$ , Eq.II.3 is simplified as

$$f_\theta(L)\mathbf{x} = \theta_0\mathbf{x} + \theta_1 D^{-1/2} A D^{-1/2} \mathbf{x}, \quad (\text{II.4})$$

where  $A$  is an adjacent matrix and  $D$  is a diagonal matrix of node degree. With the definition on normalized graph laplacian matrix,  $L = I_N - D^{-1/2} A D^{-1/2}$  and further assumption that  $\theta_0 \approx \theta_1$ , the authors of GCN introduces the well-known form of graph convolution operation,

$$H^{l+1} = f(\bar{A}H^l W^l), \quad (\text{II.5})$$

where  $\bar{A} = I_N - D^{-1/2}AD^{-1/2}$ ,  $H^l$  and  $W^l$  are input and weight of  $l^{th}$  layer while  $H^0 = \mathbf{x}$  and  $f$  denotes activation function. With the mathematically well established graph convolution operation, GCN has achieved successful performance, while being computationally efficient.

STGCN extends GCN by stacking 1-dimensional convolution layers and graph convolution layers to extract features considering temporal and spatial locality as shown in Figure 4.1. STGCN consists of 2 ST-conv blocks in which a 1-dimensional convolution layer with gated linear unit and two graph convolution layers are stacked as shown in Figure 4.1. In the 1-dimensional convolutional layer, filters extract features from each time series variable. After extracting temporal features, STGCN extracts spatial features with graph convolution operation by considering signals of neighboring nodes specified by graph laplacian matrix. The STGCN repeats the spatial and temporal feature extractions and thus, the output of last hidden layer (basis function in Figure 4.1) represents condensed spatio-temporal information of given dataset.

Finally, STGCN applies a fully convolutional operation on the basis functions. Here, the the number of features of the basis function is the same with the width of filters. This indicates that the last layer of STGCN considers all the features at the same time, which is equivalent to traditional linear regression models. With the characteristics of STGCN, our Spatio-Temporal Covariance loss function promotes STGCN to learn temporally and spatially condensed information of given data and thus we can fully utilize temporal and spatial covariance for the learning of STGCN.

## 2.2 Gaussian Processes

Here, we briefly review GPs and how the proposed loss utilizes the principle of GPs for a neural network. GPs are a samples of random variables such that any finite set have a joint Gaussian distribution [1]. GPs are effectively expressed by mean and covariance functions for a real process  $f(x)$ . As follows,  $f(x) \sim \mathcal{GP}(m(x), k(x, x'))$ , where  $m(x)$  and  $k(x, x')$  indicate mean and covariance functions respectively.

Specifying GP allows you to calculate the marginal likelihood of the given data and derive a predicted distribution for new points with the existing data .

Bayesian regression model  $f(\mathbf{x}) = \Phi^T W$  with prior  $W \sim \mathcal{N}(0, \Sigma_p)$ , We can obtain an example of GPs. Mean and covariance are as follows.

$$\begin{aligned} \mathbb{E}[f(\mathbf{x})] &= \Phi(\mathbf{x})^T \mathbb{E}[W] = 0 \\ \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] &= \Phi(\mathbf{x})^T \mathbb{E}[WW^T]\Phi(\mathbf{x}') = \Phi(\mathbf{x})^T \Sigma_p \Phi(\mathbf{x}'), \end{aligned} \tag{II.6}$$



where  $f(\mathbf{x})$  and  $f(\mathbf{x}')$  are jointly Gaussian with the zero mean and covariance given by  $\Phi(\mathbf{x})^T \Sigma_p \Phi(\mathbf{x}')$ . When we assume that  $\Sigma_p$  is an identity matrix,  $\Phi(\mathbf{x})^T \Phi(\mathbf{x}')$  becomes the covariance of  $f(\mathbf{x})$  and  $f(\mathbf{x}')$ . In GP, we typically represent the covariance with a kernel function. A kernel function can depict some distinctive characteristics of functions, such as variance, length scales, and periodicity. For instance, the squared exponential (SE) covariance function,  $\sigma^2 \exp(-|\mathbf{x}_p - \mathbf{x}_q|^2/2l)$ , specifies the covariance between pairs of random variables with two hyper-parameters,  $\sigma$  and  $l$  that control the variation and the smoothness of the functions, respectively. Note that the covariance between outputs are described as a function of inputs. For this particular covariance function, the covariance of outputs depends on how close the inputs are. The covariance of output decreases, as the distance of input,  $|\mathbf{x}_p - \mathbf{x}_q|^2$  increases. Indeed, for every positive definite covariance function  $k(\cdot, \cdot)$ , there exists an (possibly infinite) expansion in terms of basis functions (That fact can be demonstrated by Mercer's theorem). In the case of multivariate, GP can be extended such that,

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_D \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} K_{f_1, f_1} & \dots & K_{f_1, f_D} \\ K_{f_2, f_1} & \dots & K_{f_2, f_D} \\ \vdots & \dots & \vdots \\ K_{f_D, f_1} & \dots & K_{f_D, f_D} \end{bmatrix} \right) \quad (\text{II.7})$$

Note that multivariate GP considers relations between each variables for approximating the distributions of function. The proposed Spatio-Temporal Covariance loss function also exploits the covariance of variables to the learning of DNN to achieve more accurate model approximation.

GPs assume that the probability of  $f(\mathbf{x})$  and  $f(\mathbf{x}')$  being present at the same time follows joint Gaussian distributions. GPs specify relationships between  $f(\mathbf{x})$  and  $f(\mathbf{x}')$  by defining the relationships between  $\mathbf{x}$  and  $\mathbf{x}'$  and thus perform relatively well for predicting smooth functions even though given dataset is small [1]. The performance of GPs highly depends on the carefully chosen or manually designed kernel function that defines the relationships between  $\mathbf{x}$  and  $\mathbf{x}'$  and may be deteriorated for predicting severely fluctuating variables. On the contrary, neural networks are robust to such fluctuating variables and learn relationships between  $\mathbf{x}$  and  $\mathbf{y}$  automatically. However, challenges of applying neural networks in real-world problems lie in that large dataset is required for accurate predictions.

The performance of GP inference is highly depends on the kernel function. However, choosing or designing an appropriate kernel function for a given dataset is challenging. This motivates us to exploit neural networks to automatically learn the kernel function. According to Equation.II.1, a neural network consists of a feature mapping unit that generates  $\Phi(x)$  and a linear regression unit for predictions. Assuming Gaussian prior on the parameters of the linear regression unit,

we can expect the output to have Gaussian prior whose mean and covariance are zero and  $\Phi(x)^T \Sigma_P \Phi(x')$  as in Equation II.6. By penalizing the learning of the network according to the discrepancy between  $\Phi(x)^T \Sigma_P \Phi(x')$  and the covariance matrix of target values that we expect to converge to that of true distributions of target variables as the number of samples increases, we guide the network to learn the true kernel function. Another benefit from exploiting kernel matrix is that by nature, the network reflects spatial and temporal dependencies among variables. The kernel matrix of GPs indicates hidden temporal and spatial dependencies among variables. This characteristics can be fully utilized with the models that consider the dependencies for feature extraction.

## CHAPTER III

---

# Related works

---

Relationship between GPs are introduced by [5]. A fully-connected neural network is equivalent to GPs if the network consists of a single-layer with infinite number of hidden units with an i.i.d. prior over its parameters. The work proved that a one-hidden layer neural network equipped with infinite number of hidden units becomes a non-parametric GPs model by placing independent zero-mean Gaussian priors to all the weights of the network and integrating them out. [2] proves the exact equivalence between the infinitely wide deep network and GP. They also derive the GPs kernel for multi-hidden-layer neural networks with general non-linearity based on signal propagation theory in [6], called as NNGP. A scalable GPs model for regression by applying a DNN as the feature-mapping function is proposed in [7]. The deep neural network is pre-trained in an unsupervised way, as like a stacked denoising auto-encoder. After that, the last hidden layer of the pre-trained deep network can be interpreted as a Bayesian linear regression. DNN-based Gaussian Process (DNN-GP), The resulting model, can represent much more meaningful features of the data although it has the finite-dimensional but deep-layered feature-mapping function. Unlike to the standard GPs, DNN-GP avoids kernel matrix inversion, so it scales with the size of the training set.

Recently, a regularization of DNN with the norm of a reproducing kernel Hilbert space (RKHS) is proposed in [13]. A study about deep learning model that learns a differential compositional GP kernel by combining several primitive kernels using a neural network has also been proposed in [14].

---

# Spatio-Temporal Covariance loss

---

## 4.1 Spatio-Temporal Covariance loss

This section briefly explains STGCN [8] and give implementation details of the proposed Spatio-Temporal Covariance loss coupled with STGCN.

### 4.1.1 Learning the Basis Functions in STGCN

Figure 4.1 shows STGCN extracting spatial and temporal dependencies among variables with our proposed Spatio-Temporal covariance loss function. STGCN consists of 2 ST-conv blocks in which a 1-D convolution layer with gated linear unit and two graph convolution layers are stacked. In a 1-D convolution layer, filters extract temporal dependencies from a time series variable while a graph convolution layer extracts spatial dependencies by considering information of neighboring time series variables specified by an adjacent matrix. Thus, the output of last hidden layer (i.e. basis function in Figure 4.1) represents condensed spatio-temporal information of a given dataset. Finally, STGCN applies fully connected operation which is equivalent to a linear regression.

[Basis function] Given the output of the last hidden layer  $\Phi(\mathbf{x})$ ,  $\Phi(\mathbf{x})$  and  $f(\mathbf{x})$  is in a relation of a linear regression as  $f(\mathbf{x}) = \Phi(\mathbf{x})^T \mathbf{w}$ . Thus,  $\Phi(\mathbf{x})$  is equivalent to the basis function expansion for the linear regression. Assuming  $\mathbf{w} \sim \mathcal{N}(0, \sigma^2 I)$ , the outputs of STGCN,  $f(\mathbf{x})$

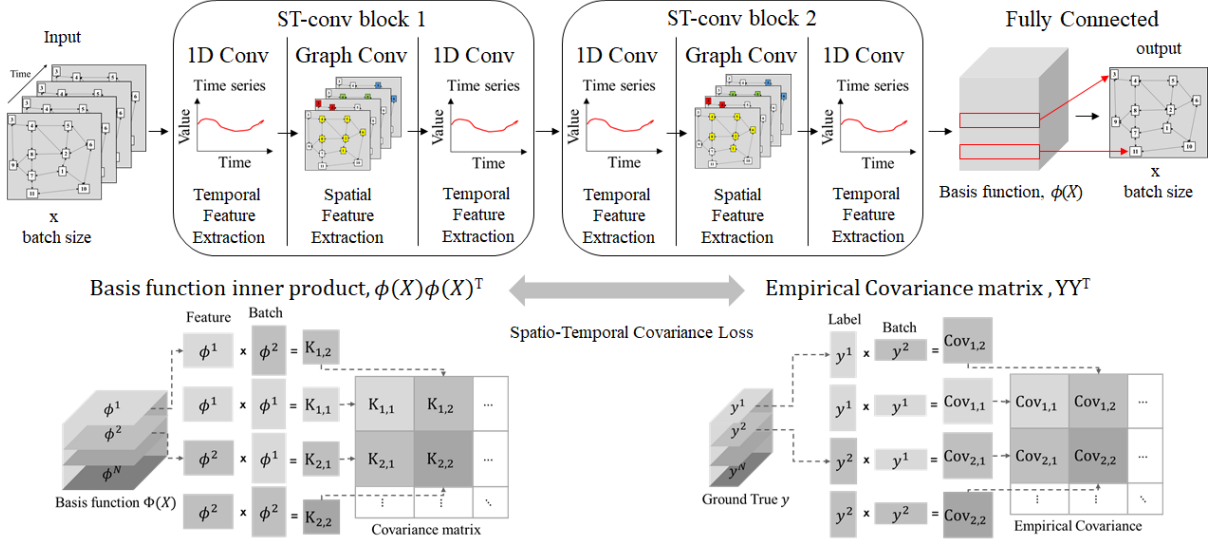


Figure 4.1: An architecture of a Spatio-Temporal Graph Convolutional Network with Covariance Loss.

follows Gaussian distributions whose mean and covariance are

$$\mathbb{E}[f(\mathbf{x})] = \Phi(\mathbf{x})^T \mathbb{E}[\mathbf{w}] = 0, \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] = \Phi(\mathbf{x})^T \mathbb{E}[\mathbf{w}\mathbf{w}^T] \Phi(\mathbf{x}') = \sigma^2 \Phi(\mathbf{x})^T \Phi(\mathbf{x}'). \quad (\text{IV.1})$$

Thus,  $f(\mathbf{x})$  and  $f(\mathbf{x}')$  are jointly Gaussian with the zero mean and covariance matrix given by  $\sigma^2 \Phi(\mathbf{x})^T \Phi(\mathbf{x}')$ .

#### 4.1.2 Spatio-Temporal Covariance Loss

For a given dataset,  $D = \{(\mathbf{x}_i, \mathbf{y}_i) | i = 1, \dots, T\}$ ,  $\mathbf{x} \in R^N$ ,  $\mathbf{y} \in R^N$ , where  $T$  and  $N$  denote the size of the dataset  $D$  and the number of nodes in a graph, respectively and the prediction goal  $\mathbf{y}_i = \mathbf{x}_{i+\tau}$  while  $\tau$  is time interval for prediction. [Spatio-Temporal Covariance] For given two time series variables,  $\mathbf{v}$  and  $\mathbf{w}$ , we define the Spatio-Temporal Covariance  $\tilde{\Sigma}_{\mathbf{v}, \mathbf{w}}$  such that  $\tilde{\Sigma}_{\mathbf{v}, \mathbf{w}} = (\mathbf{v} - \bar{\mathbf{v}})(\mathbf{w} - \bar{\mathbf{w}})^T$ .  $\bar{\mathbf{v}}$  and  $\bar{\mathbf{w}}$  denote means of the  $\mathbf{v}$  and  $\mathbf{w}$ . When  $\mathbf{v} = \mathbf{w}$ , the Spatio-Temporal Covariance represents temporal dependencies in the variable. In contrast, when  $\mathbf{v} \neq \mathbf{w}$ , the Spatio-Temporal Covariance measures spatial dependencies of  $\mathbf{v}$  and  $\mathbf{w}$ . To utilize the Spatio-Temporal Covariance while training STGCN, we further define Spatio-Temporal Covariance Loss function. (Spatio-Temporal Covariance Loss) For a batch of given  $D$ , we define the Spatio-Temporal Covariance loss as

$$\frac{1}{m^2} \sum_{i=1}^N \sum_{j=1}^N (\tilde{\Sigma}_{\mathbf{y}^i, \mathbf{y}^j} - \sigma^2 \Phi^i(\mathbf{x}) \Phi^j(\mathbf{x})^T)^2. \quad (\text{IV.2})$$

$m$  is the number of input units in a dataset while  $N$  is the number of nodes in a graph.  $\sigma$  denotes the variance of the weights of the last hidden layer.  $\Phi^i(\mathbf{x})$  indicates basis functions on  $\mathbf{x}$  of STGCN for  $i^{th}$  node in a graph. More specifically,  $\Phi^i(\mathbf{x}) = [\Phi_1^i(\mathbf{x}), \Phi_2^i(\mathbf{x}), \dots, \Phi_h^i(\mathbf{x})]^T$  where  $h$  denotes the number of nodes in the last hidden layer. Since  $\mathbf{y}^i = \mathbf{w}\Phi^i(\mathbf{x})$ , if  $\Phi^i(\mathbf{x})$  and  $\Phi^j(\mathbf{x})$  are related,  $\mathbf{y}^i$  and  $\mathbf{y}^j$  should be related as well with a scale factor  $1/\sigma$  according to the Equation IV.1. Thus, with the proposed Spatio-Temporal Covariance Loss, we measure the degrees of their relations with Spatio-Temporal Covariance of  $\mathbf{y}^i$  and  $\mathbf{y}^j$  and inner product of  $\Phi^i(\mathbf{x})$  and  $\Phi^j(\mathbf{x})$  as shown in Figure 4.1 and additionally penalize the model by comparing the relations as given in Equation IV.2. Finally, we add the Spatio-Temporal Covariance Loss into the general mean squared error (MSE) loss of STGCN with the hyper-parameter  $\lambda$  which regulates the importance of the Spatio-Temporal Covariance Loss such that,

$$\frac{1}{m}(\mathbf{y} - \hat{\mathbf{y}})^2 + \lambda \left( \frac{1}{m^2} (\tilde{\Sigma}_{\mathbf{y}} - \sigma^2 \Phi \Phi^T)^2 \right), \quad (\text{IV.3})$$

where  $\hat{\mathbf{y}}$  is the prediction output from the model.

### 4.1.3 Principles of Spatio-Temporal Covariance Loss Based Neural Networks

The proposed Spatio-Temporal Covariance loss function aims to find a basis function with properties similar to those in RKHS of the covariance kernel function of  $\mathbf{y}$ . Therefore, we wish to guide the output of last hidden layer which is represent basis functions of STGCN with the following principles. Assuming that  $\mathbf{y} \sim \mathcal{GP}(\mu, \Sigma)$ , training STGCN with MSE is equivalent to guide  $\hat{\mathbf{y}}$  to  $\mu$  as it reduces  $\frac{1}{m}(\mathbf{y} - \hat{\mathbf{y}})^2$ . By contrast, we expect that the basis functions can represent the variance and covariance of  $\mathbf{y}$  with our proposed loss which measures the difference between the Spatio-Temporal Covariance  $\tilde{\Sigma}_{\mathbf{y}}$  and  $\Phi \Phi^T$ .

Ever since the relationship between GPs and DNNs are introduced [5], combining GPs with DNNs has been attracting huge interest. The exact equivalence between infinitely wide deep networks and GPs is defined [2]. They also derive the GPs kernel for multi-hidden-layer neural networks with general non-linearity based on signal propagation theory [6], called as NNGP. A scalable GPs model for regression is proposed by applying a DNN as the feature-mapping function [7]. A regularization of DNN with the norm of a reproductive kernel hilbert space (RKHS) is presented [13]. Unlike previous studies, our proposed loss function operates on CNN based networks and does not require any derivation from the covariance functions or matrices of the GPs. Also note that we do not set prior of the given data pair  $(\mathbf{x}, \mathbf{y})$  to find the basis functions. Rather than changing DNN architecture or model itself, the covariance loss function

can be attached to the conventional loss functions with the hyper-parameter during training phase.

---

# Experimental Evaluations

---

## 5.1 Experimental Evaluations

This section presents an extensive set of experiments to show the validity and efficiency of the proposed Spatio-Temporal Covariance loss function. For experiments, we compare prediction accuracy of various models on benchmark datasets including PeMSD7 and METR-LA. To measure and evaluate the accuracy of different models, we employ masked mean absolute error (MAE), mean absolute percent error (MAPE) and root mean square error (RMSE) in which only valid data points are involved for the accuracy measures.

### 5.1.1 Experimental setups

Our predictive objective is to minimize the loss function in Eq. IV.3. We conduct grid search to find the best parameters for the proposed loss function. All of the input units consist of 12 data points that cover consecutive 60 minutes to forecast traffic conditions in next 15 ~ 60 minutes. For experiments, we employ the following two popular benchmark datasets, PeMSD7 and METR-LA. PeMSD7 is a highway traffic dataset from California which collects velocity of cars in every 30 seconds and finally aggregated into every 5-minute interval from the raw data. PeMSD7 consists of 12,612 velocity data of randomly selected 228 sensor stations measured in the weekdays of May and June of 2012 [8]. METR-LA contains records of statistics on traffic



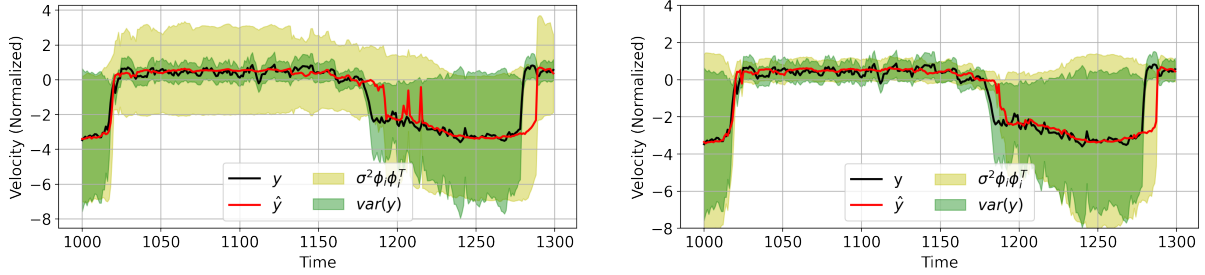


Figure 5.1: **Learning variance of target variables:** with the proposed loss, diagonal elements of basis inner product ( $\sigma^2 \phi_i \phi_i^T$ ) indicate variances of target variables. In contrast to STGCN (left), STGCN-Cov (right) successfully learns such basis functions.

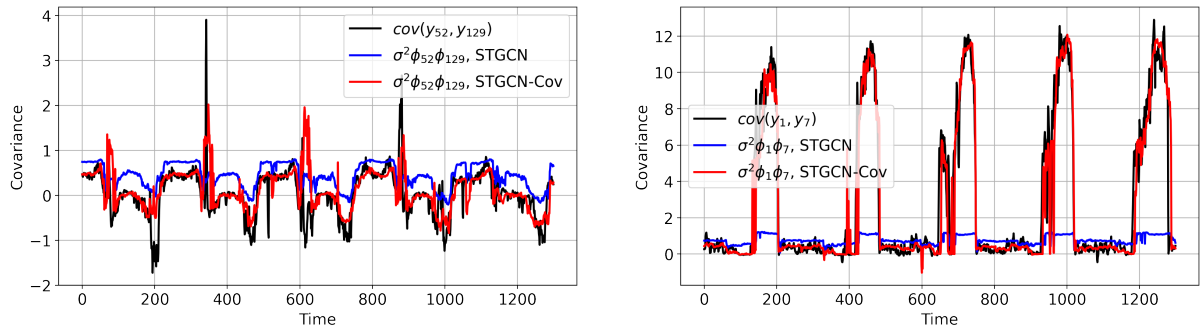


Figure 5.2: **Learning covariance of target variables:** non-diagonal elements of basis inner product ( $\sigma^2 \phi_i \phi_j$ ) mean covariance of target variables. Regardless of how the two different variables are related, independent (left, Node 52 and 129) or correlated (right, Node 1 and 7), STGCN-Cov successfully learns such basis functions.

speed for highway of Los Angeles County ranging from Mar 1st 2012 to Jan 30th 2012. The dataset consists of 207 sensors and about 35,000 records for each sensor.

### 5.1.2 Learning Basis functions

The goal of the proposed loss function is to guide a neural network to automatically learn true kernel matrix of target variables, which consists of variances and covariances of the target variables. To achieve the goal, it is important to learn basis functions whose inner product is equivalent to the variances and covariances of target variables. In this section, we demonstrate the validity of the proposed loss function by comparing the inner product of basis functions with target variables true variances and covariances. For this set of experiments, we apply the proposed loss function to STGCN with the PeMSD7 dataset and analyze the characteristics of the basis functions.

In Fig.5.1, we present a diagonal element of basis inner product ( $\sigma^2 \phi_i \phi_i^T$ ) and a variance of a corresponding target variable as a function of time on STGCN and STGCN-Cov. Even though

we can estimate mean function of  $\mathbf{y}$  by revealing the linear relation between  $\mathbf{x}$  and  $\mathbf{y}$ , we assume  $\mathbb{E}[\mathbf{y}] = 0$  for simplicity. In contrast to STGCN that learns a basis function whose inner product is irrelevant to the variance of target variable, STGCN-Cov successfully learns a basis function whose inner product are equivalent to the variance of target variable. In Fig.5.2, we present a covariance of two different target variables and a non-diagonal element of basis inner product corresponding to the target variables as a function of time. As shown in the figure, STGCN-Cov successfully learns basis functions whose inner products are equivalent to the covariance of the target variables as well regardless of how much the two target variables are correlated.

Table 5.1: Accuracy Comparison for PeMSD7 dataset

Model	MAE (15/30/45 min)	MAPE (15/30/45 min)	RMSE (15/30/45 min)
HA	4.01	10.61	7.2
ARIMA	5.55 / 5.86 / 6.27	12.92 / 13.94 / 15.20	9.00 / 9.13 / 9.38
FC-LSTM	3.57 / 3.94 / 4.16	8.60 / 9.55 / 10.10	6.20 / 7.03 / 7.51
STGCN(Cheb)	2.25 / 3.03 / <b>3.57</b>	5.26 / <b>7.33</b> / <b>8.69</b>	4.04 / 5.70 / 6.77
STGCN(1st)	2.26 / 3.09 / 3.79	5.24 / 7.39 / 9.12	4.07 / 5.77 / 7.03
STGCN-Cov	<b>2.22</b> / <b>3.02</b> / 3.6	<b>5.21</b> / 7.37 / 8.88	<b>4.00</b> / <b>5.61</b> / <b>6.68</b>

## 5.2 Effect of the basis functions on PeMSD7 dataset

For experiments on PeMSD7, we employ STGCN and the hyper-parameters presented in [8] to show the accuracy improvements clearly. More specifically, the number of frames in one input unit is 12 and a batch dataset consists of randomly sampled 32 input units. The kernel sizes of 1-D and graph convolution operations are set to be 3. For accuracy comparison, we employ the following baseline: Historical Average (HA), ARIMA<sub>kal</sub> [15], FC-LSTM [16].

Table 5.1 shows prediction results of traffic speed over the next 15, 30 and 45 minutes. From the comparison, we observe that neural network based algorithms obtained more accurate results than the other algorithms, and especially, algorithms that simultaneously utilize temporal and spatial locality, such as STGCN and STGCN-Cov, are superior to the traditional deep learning models. In terms of RMSE, STGCN coupled with the proposed loss function, STGCN-Cov continuously outperforms the other representatives in every predictions but in case of MAE and MAPE, STGCN (Cheb) shows better performance in middle and long term predictions.

In Fig. 5.3, we plot prediction results and basis functions (dashed color lines) as a function of time. In the figure, we can see that STGCN-Cov outperforms STGCN in all of the evaluation metrics. Furthermore, STGCN-Cov detects the beginning of rush hour earlier than STGCN and huge prediction failures of STGCN in rush hour doesn't occur in STGCN-Cov. The basis

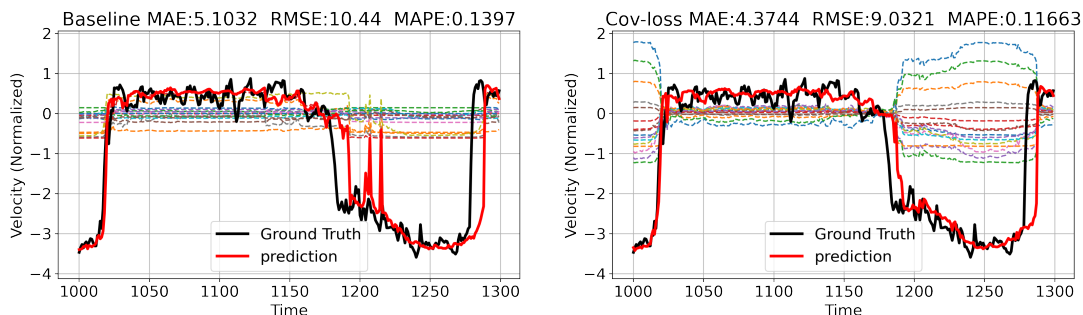


Figure 5.3: Predictions and basis of STGCN (left) and STGCN-Cov (right) on Node 7 of PeMSD7

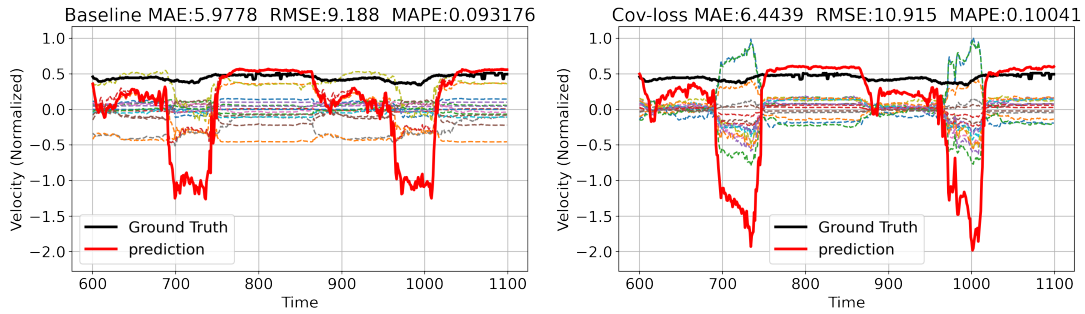


Figure 5.4: Prediction and basis of STGCN (left) and STGCN-Cov (right) on Node 110 of PeMSD7

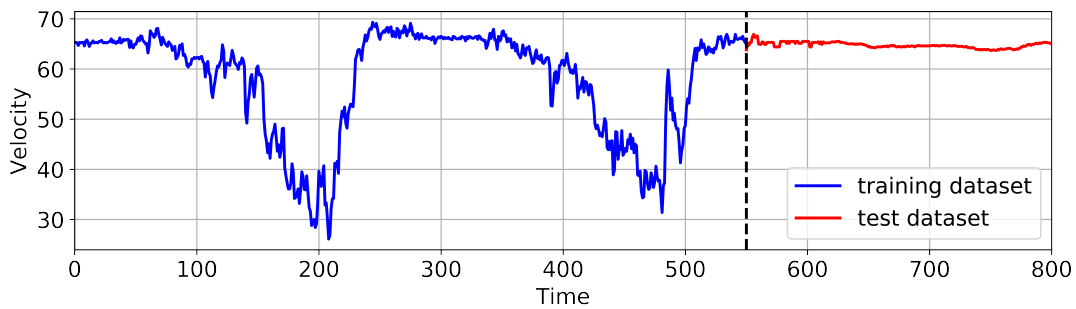


Figure 5.5: Training and test dataset of node 110

functions of STGCN in Fig. 5.3 are results of reflecting temporal and spatial trend extracted from training dataset on test dataset and thus the basis functions are simply formed in such a way that for time  $t$ , the summation of the basis functions is close to the prediction. In contrast, the proposed loss function further limits the basis functions to consider temporal and spatial dependencies among themselves and thus each basis function has a shape similar to prediction results but only differs in its scale as shown in the right chart of Fig. 5.3. This further indicates that with the STGCN-Cov, the distance of the best parameters for conflicting predictions of STGCN on time  $t$  and  $t'$  becomes closer than that of STGCN. This fact results in performance gains in final prediction results.

Next, we analyze prediction results of STGCN-Cov to see where the poor MAE and MAPE come from. Fig.5.4 shows prediction results and corresponding basis functions that report less accurate results than STGCN. We found that the huge miss-predictions in the figure result from dependencies extracted from training dataset that is significantly different from those of test dataset as shown in Fig.5.5. Considering predictions of 1-d FCN that only captures time dependencies report quite similar prediction errors with STGCN/STGCN-Cov, temporal dependencies in training dataset is the most significant reason for the huge prediction error. We can observe that the prediction results of STGCN-Cov are even worse than those of STGCN

Table 5.2: Accuracy Comparison for METR-LA dataset

Model	MAE (15/30/60 min)	MAPE (15/30/60 min)	RMSE (15/30/60 min)
HA	4.16	13.0	7.8
ARIMA	3.99 / 5.15 / 6.90	9.6 / 12.7 / 17.40	8.12 / 10.45 / 13.23
FC-LSTM	3.44 / 3.77 / <b>4.37</b>	9.60 / 10.9 / 13.2	6.30 / 7.23 / 8.69
STGCN	2.88 / 3.47 / 4.59	7.6 / 9.6 / 12.7	5.74 / 7.24 / 9.40
GWNET	2.74 / 3.41 / 4.77	6.90 / 9.08 / 12.69	5.31 / 6.85 / 9.41
GWNET-Cov	<b>2.71</b> / <b>3.32</b> / 4.39	<b>6.82</b> / <b>8.8</b> / <b>11.85</b>	<b>5.13</b> / <b>6.56</b> / <b>8.6</b>

from the Fig.5.4. The figure shows that the proposed loss function makes a model learn temporal and spatial dependencies more strongly and thus, in cases where data distributions change significantly, may deteriorate the overall accuracies severely.

### 5.3 Effect of the basis functions on METR-LA dataset

In this set of experiments, we employ Graph wavenet and parameters presented in [9]. To train the model, we use historical window of size 60 minutes and batch size of 64. To connect the model with the proposed loss function we switch the activation, relu to tanh. For accuracy comparison, we employ the following baseline: Historical Average (HA), ARIMA<sub>kal</sub> [15], FC-LSTM [16] and STGCN [8]. Graph wavenet is originally designed to predict all the time steps independently not recursively. However, this strategy is inappropriate with the proposed loss since it forces the learning of basis function to consider conflicting kernel functions simultaneously. For the reason, we modify GWNET and GWNET-Cov to predict in a recursive manner as all the other algorithms do.

Table 5.2 shows prediction results of traffic condition over the next 15, 30 and 60 minutes. From the table, we can see that the proposed loss function coupled with GWNET (GWNET-

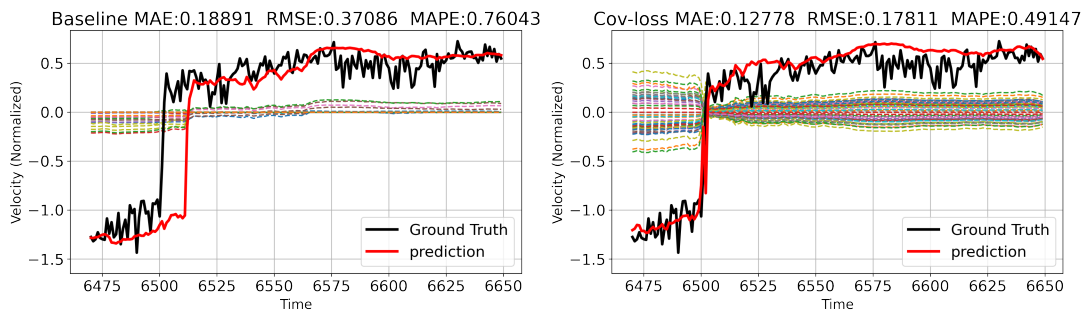


Figure 5.6: Prediction and basis of GWNET (left) and GWNET-Cov (right) on Node 6 of METR-LA

Cov) outperforms all the others in every evaluation metric except for the MAE on long-term prediction.

In Fig. 5.6, we compare prediction results and basis functions of GWNET and GWNET-Cov. We can see that GWNET-Cov successfully learns basis functions that can represent temporal and spatial dependencies of target variables. In addition, GWNET-Cov detects the ending of the rush hour much earlier than GWNET as shown in the figure.

## 5.4 Noise robustness

We observe that with the proposed loss function, a neural network learns temporal and spatial dependencies more strongly. This nature leads to more robust prediction results against noisy input. Figure 5.7 shows RMSE increase of STGCN and STGCN-Cov on PeMSD7 dataset as a function of the number of noisy nodes. We can see that STGCN-Cov is more robust than STGCN when the number of noisy node is small due to the spatial dependencies. However, as the number of node increases, the performance of STGCN-Cov follows global dependencies. Figure 5.8 depicts difference of predict value as the noisy node increases at various nodes (7, 11, 52, 113). When the number of noisy nodes is small, difference is small due to spatio dependencies, but the number of noisy node becomes large, the shape of the kernel distribution changes and predicted value also changes more sensitively (Node 7,11). And there are nodes that are affected by the spatio change of the kernel more faster (Node 52) or slower (Node 113).

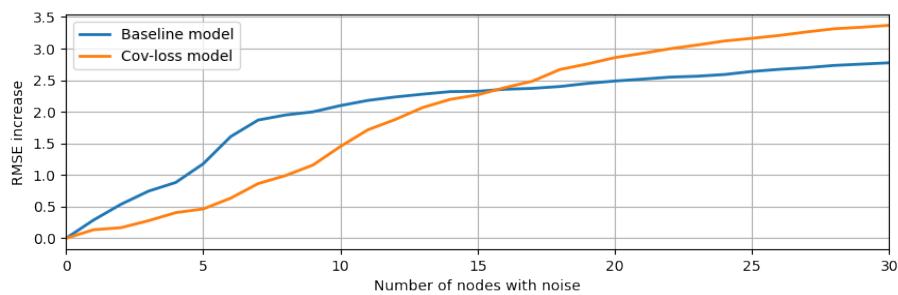


Figure 5.7: RMSE increase v.s. the number of noisy node

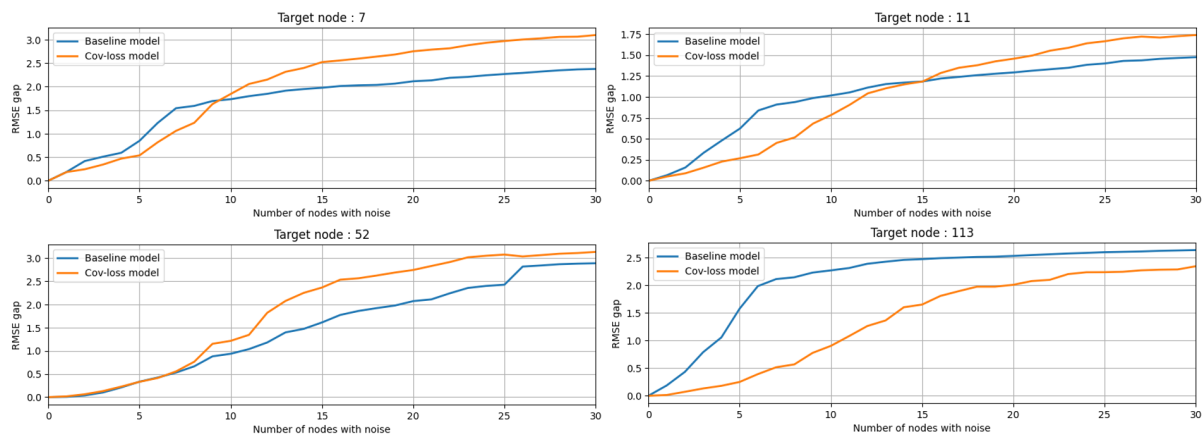


Figure 5.8: Difference of predicted value (RMSE) as the noisy node increases

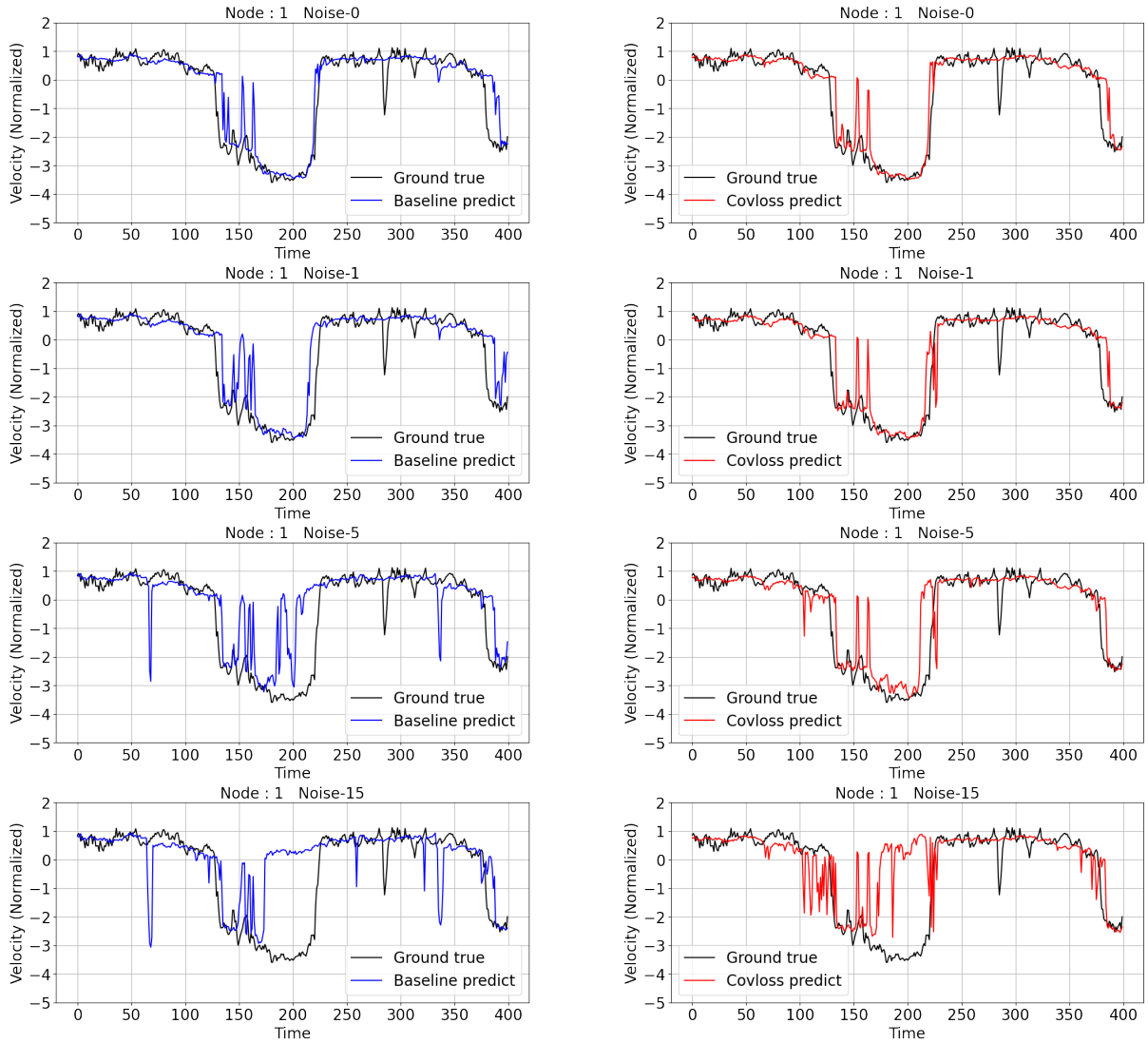


Figure 5.9: The effect of noisy nodes on prediction results

### 5.4.1 Case studies on the effect of noisy nodes

Figure 5.9 depicts prediction accuracy of Node 1 in PeMSD7 dataset of STGCN (left) and STGCN-Cov (right) as the number of noisy nodes increases from 0 to 15. For this set of experiments, we deliberately add noise to  $k$  nodes that have higher covariances with Node 1. Noise-1 and Noise-5 mean that the number of nodes including noise is 1 and 5, respectively. As shown in the figure, the prediction results of STGCN deteriorate severely when the number of noisy nodes is more than 5, however, STGCN-Cov still shows accurate predictions. This is because with the proposed loss function, a neural network learns spatial dependencies more strongly.



## 5.5 Comparisons with L1 and L2 regularizations

This section compares RMSE of STGCN when applying L1, L2 and the proposed covariance loss. Figures 5.10, 5.11 and 5.12 show RMSE of STGCN (baseline), STGCN-Cov (cov- $\lambda$ ), STGCN-L1 (L1- $\lambda$ ), STGCN-L2 (L2- $\lambda$ ) on PeMSD7 dataset where  $\lambda$  indicates the importance of the additional loss terms. In every case, the proposed Spatio-Temporal Covariance loss outperforms all the others. In particular, the existing regularizers (L1, L2) showed lower performance than the baseline as the prediction step was farther away, but the cov-loss regularizer showed good performance even at the farthest prediction step (step 9). This proves that cov-loss performed better than other regularizers for long term prediction by learning the kernel distribution of timeseries data.

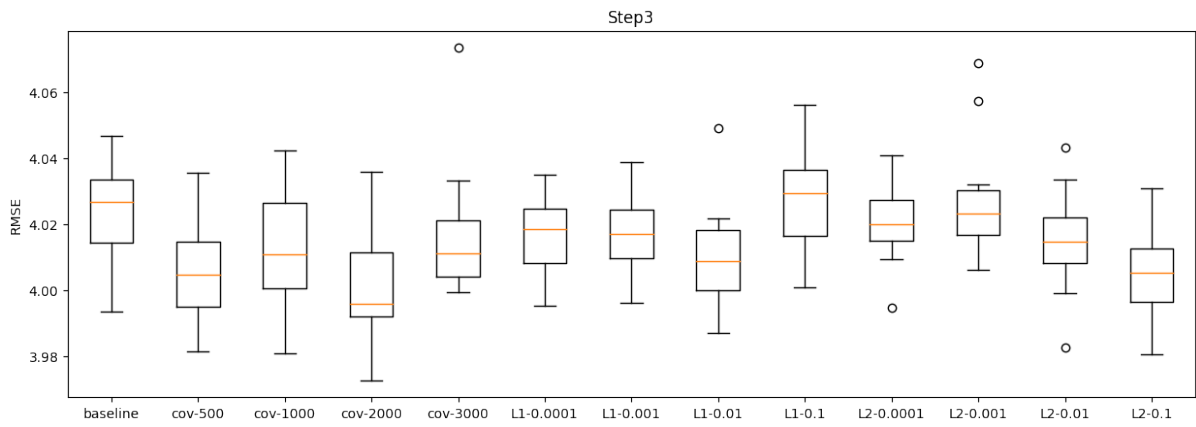


Figure 5.10: RMSE comparison of STGCN with L1, L2 and Covariance loss for prediction step 3.

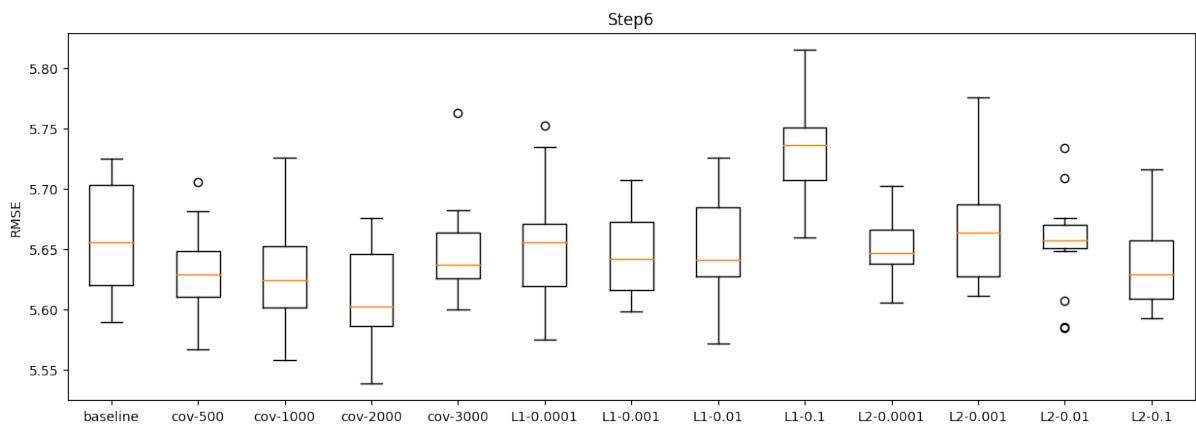


Figure 5.11: RMSE comparison of STGCN with L1, L2 and Covariance loss for prediction step 6.

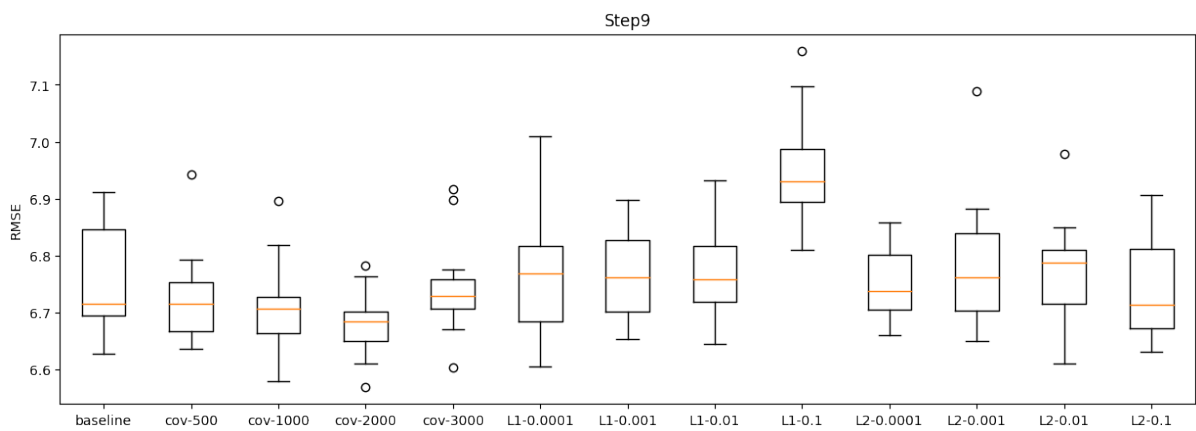


Figure 5.12: RMSE comparison of STGCN with L1, L2 and Covariance loss for prediction step 9.

## CHAPTER VI

---

# Conclusion

---

In this paper, we propose a novel loss function, Spatio-Temporal Covariance Loss function that guides the learning of a neural network to learn the underlying kernel function of target values, which are spatial and temporal dependencies of multivariate time series. With the proposed Spatio-Temporal Covariance loss, neural networks learn basis functions whose inner product is equivalent to kernel matrix. Our extensive sets of experimental evaluations have shown that the proposed loss function yields more accurate results in terms of prediction for real-world multivariate time series dataset.

---

## References

---

- [1] Christopher KI Williams and Carl Edward Rasmussen, *Gaussian processes for machine learning*, vol. 2, MIT Press Cambridge, MA, 2006. 1, 5, 6
- [2] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein, “Deep neural networks as gaussian processes,” *arXiv preprint arXiv:1711.00165*, 2017. 1, 8, 11
- [3] Tamir Hazan and Tommi Jaakkola, “Steps toward deep kernel methods from infinite neural networks,” *arXiv preprint arXiv:1508.05133*, 2015. 1
- [4] Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth, “Manifold gaussian processes for regression,” in *Proceedings of the International Joint Conference on Neural Networks*. IEEE, 2016, pp. 3338–3345. 1
- [5] Radford M Neal, *Bayesian learning for neural networks*, Ph.D. thesis, Citeseer, 1995. 1, 8, 11
- [6] Youngmin Cho and Lawrence K Saul, “Kernel methods for deep learning,” in *Proceedings of the Advances in Neural Information Processing Systems*, 2009, pp. 342–350. 1, 8, 11
- [7] Wenbing Huang, Deli Zhao, Fuchun Sun, Huaping Liu, and Edward Chang, “Scalable gaussian process regression using deep neural networks,” in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015. 1, 8, 11

## REFERENCES

---

- [8] 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. 2, 3, 9, 13, 16, 18
- [9] 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. 2, 18
- [10] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323. 3
- [11] Thomas N Kipf and Max Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016. 3, 4
- [12] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in *Proceedings of the Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852. 4
- [13] Alberto Bietti, Grégoire Mialon, Dexiong Chen, and Julien Mairal, “A kernel perspective for regularizing deep neural networks,” in *Proceedings of the International Conference on Machine Learning*, 2019, pp. 664–674. 8, 11
- [14] Shengyang Sun, Guodong Zhang, Chaoqi Wang, Wenyuan Zeng, Jiaman Li, and Roger Grosse, “Differentiable compositional kernel learning for gaussian processes,” *arXiv preprint arXiv:1806.04326*, 2018. 8
- [15] James D Hamilton, *Time series analysis*, vol. 2, Princeton New Jersey, 1994. 16, 18
- [16] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, “Sequence to sequence learning with neural networks,” in *Proceedings of the Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112. 16, 18