Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 31(4): 2252–2264.
DOI: 10.3934/era.2023115
Received: 27 October 2022
Revised: 06 January 2023
Accepted: 28 January 2023
Published: 23 February 2023

Research article

Deep evidential learning in diffusion convolutional recurrent neural network

Zhiyuan Feng[1,†], Kai Qi[1,†], Bin Shi[1], Hao Mei[2], Qinghua Zheng[1] and Hua Wei[2,*]

[1] Xi 'an Jiaotong University, 28 Xianning West Road, Beilin District, Xi 'an, China
[2] New Jersey Institute of Technology, New Jersey, USA

† The authors contributed equally to this work.

* **Correspondence:** Email: hua.wei@njit.edu.

Abstract: Graph neural networks (GNNs) is applied successfully in many graph tasks, but there still exists a limitation that many of GNNs model do not consider uncertainty quantification of its output predictions. For uncertainty quantification, there are mainly two types of methods which are frequentist and Bayesian. But both methods need to sampling to gradually approximate the real distribution, in contrast, evidential deep learning formulates learning as an evidence acquisition process, which could get uncertainty quantification by placing evidential priors over the original Gaussian likelihood function and training the NN to infer the hyperparameters of the evidential distribution without sampling. So evidential deep learning (EDL) has its own advantage in measuring uncertainty. We apply it with diffusion convolutional recurrent neural network (DCRNN), and do the experiment in spatiotemporal forecasting task in a real-world traffic dataset. And we choose mean interval scores (MIS), a good metric for uncertainty quantification. We summarized the advantages of each method.

Keywords: uncertainty quantification; evidential deep learning; graph neural network; spatiotemporal forecasting; regression task

1. Introduction

In recent years, GNNs have facilitated excellent performance in many graph analysis tasks, such as graph node classification and link prediction, by extracting advanced features of nodes from their topological neighborhood [1–3]. Although graph neural networks has achieved some success in its tasks on graph, it still has two major limitations: 1) some graph structures have the phenomena of *over-smoothing* and *over-fitting*, GNN cannot get very high-level features of the graph [4]; 2) currently, majority of GNNs implementations do not provide uncertainty quantification of their output predictions [5].

For uncertainty quantification, methods are statistically divided into two types: frequentist methods and Bayesian methods. The biggest difference between them is whether to allow the use of a prior probability distribution. For freuqentist methods, although it is computationally cheaper for a single confidence interval, re-training for different intervals is required. For Bayesian methods, there are several limitations which including the intractability of directly inferring the posterior distribution of the weights by given data, the computational expense of sampling during inference and how to choose a rensonable prior.

On the contrary, EDL formulates learning as an evidence acquisition process [6, 7]. For EDL, it can learn a higer-order evidential distribution in every training examples. EDL samples from this distribution yields instances of lower-order likelihood function which the data was drawn. It is different from Bayesian, because it doesn't need to place prior on network weights. Evidential methods place priors directly over the likelihood function. By training a neural network to output the hyparparameters of the higher-order evidential distribution, we can learn the grounded representation of uncertainty without the need for sampling.

Since Bayesian and frequentist methods already have achieved good performance on the tasks of uncertainty quantification in graph neural network, we try to apply EDL to graph neural network to provide the uncertainty quantification of its output predictions. For graph-based data, we use the DCRNN [8] as deep learning model. DCRNN is a holistic method which is trained by maximizing the likelihood of generating the target future time series by backpropagation through time. DCRNN has the ability to capture spatiotemporal dependencies among time series and can be applied to various spatiotemporal forecasting problems. We experiment our method, EDL with DCRNN in a real-world spatiotemporal forecasting task and used MIS as the metric to make the uncertainty quantification.

Our contributions include:

- We apply EDL to graph neural network, and use DCRNN as deep learning model, using it to learning uncertainty quantification (UQ) in the graph neural network task.
- We choose MIS [9] as a metric of uncertainty measurement, and compare it with baselines mentioned in [10]which includes boostrap, quantile regression and MIS regression.
- We do the experiment in METR-LA dataset [11] and Covid-19 incident dataset [12]. Then we analyze the advantages of each method.

2. Materials and method

2.1. Related work

Spatiotemporal forecasting is one of the important tasks in graph neural network. Considering spatiotemporal correlation for a given time series information will significantly improve the prediction performance. There are already many works on graph, which can be mainly divided into two types, applying to grid data and graph data. For spatiotemporal prediction on graph, graph convolutional neural networks (GCNs) [13] were first introduced, which connected the spectral graph theory and deep neural networks, then ChebNet [14] was proposed which improves GCN with fast localized convolutions filters and later combined with recurrent neural networks (RNN) [15] for spatialtemporal forecasting. However the following methods all require the graph to be undirected to calculate meaningful. Then diffusion-convolutional neural network (DCNN) [16] which defines convolution as a diffusion process across each node in a graph-structured input makes some transformations to vertex domain. GraphCNN [17] generalize convolution to graph by convolving every node with its p nearest neighbors. On the basis of the previous two models, DCRNN [8] keeps temporal dynamics into consideration, which models the sensor network as a weighted directed graph, and defines the proposed convolution using bidirectional graph random walk, as well as introducing scheduled sampling to bridge with long-term temporal dependency.

Uncertainty quantification has a long history [18–20] and and there are also many works have done that training the neural networks to model probability distribution [21–24]. Now, there are mainly two types of uncertainty quantification, Bayesians and frequentists. The Bayesians assume that the data is obey a prior distribution, which uses sampling to approximate real distribution [18]. Relevant methods including: MC dropout [25], SG MCMC [26, 27] etc. But frequentists doesn't think there is a prior distribution. It approximate real distribution only by sampling. Relevant methods including: Bootstrap [28], MIS regression [10], quantile regression [29] etc.

2.2. Preliminaries

2.2.1. Problem definition

The task of spatiotemporal forecasting on graph is to predict the feature information of specific steps from the historical time series of nodes on a given graph and the spatial correlation between them. For directed graph $\mathcal{G}$, the vertex set and edge set of the graph are needed to represent the base structure, the number of nodes is set as N. And the spatial correlation between nodes is essential, a weighted adjacency matrix $W \in \mathbb{R}^{N\times N}$ is used to denote this. Denote the feature information of graph G at t time steps as $X \in \mathbb{R}^{N\times D}$, where D is defined as the feature number of each node. Through the spatiotemporal correlation matrix W and the time series X, we construct a function F to achieve the spatiotemporal forecasting of feature information of the graph in the future T time steps:

$$\left[X^{(1)}, \cdots, X^{(t)}; W\right] \xrightarrow{F(\cdot)} \left[X^{(t+1)}, \cdots, X^{(t+T)}\right] \tag{2.1}$$

Compared with the traditional time series prediction and spatial interpolation, the spatiotemporal forecasting models spatial and temporal dependencies in two dimensions and makes prediction, deep neural network usually constructs CNN to simulate the spatial correlation for the standard grid data and RNN to simulate the time correlation. While for map data, graph convolution clearly adapt better to graph structures. To achieve more accurate prediction, the performance of DCRNN is very impressive.

DCRNN defines graph convolution in the following form:

$$X_{:,d} \star \mathcal{G} f_\theta = \sum_{k=0}^{K-1} \left(\theta_{k,1}\left(D_O^{-1}W\right)^k + \theta_{k,2}\left(D_I^{-1}W^\top\right)^k\right) X_{:,d} \tag{2.2}$$

where $d \in \{1, \cdots, D\}$ denote the features of the node, $\theta \in \mathbb{R}^{K\times 2}$ denote the parameters for the filter, and D_O, D_I denote the out degree and in degree matrices respectively. Graph convolution realizes spatial dependency, while temporal dependency is usually realized by RNNs [30]. In particular, DCRNN [8] makes the transformation of GRU [30] to complete this task, which called DCGRU [8]:

$$r^{(t)} = \sigma\left(\Theta_r \star_{\mathcal{G}} \left[X^{(t)}, H^{(t-1)}\right] + b_r\right) \tag{2.3}$$

$$u^{(t)} = \sigma\left(\Theta_u \star \mathcal{G}\left[X^{(t)}, H^{(t-1)}\right] + b_u\right) \tag{2.4}$$

$$C^{(t)} = \tanh\left(\Theta_C \star \mathcal{G}\left[X^{(t)}, \left(r^{(t)} \odot H^{(t-1)}\right)\right] + b_c\right) \tag{2.5}$$

$$H^{(t)} = u^{(t)} \odot H^{(t-1)} + \left(1 - u^{(t)}\right) \odot C^{(t)} \tag{2.6}$$

where $\star\mathcal{G}$ denotes the graph convolution, $H^{(t)}$, $X^{(t)}$represent the output and input at time t, Θ_r, Θ_u, $\Theta_C$mean the parameters for the corresponding filters. Evidently, DCGRU can be trained as well as GRU using backpropagation through time.

2.2.2. Evaluation metrics

In the previous section, we described the network architecture and implementation methods for spatio-temporal graph prediction. In this section, we discuss the measurement of accuracy and uncertainty. In point estimation, the accuracy of the model's prediction is the most important criterion, and mean absolute error (MAE), root mean square error (RMSE) are used in our experiment. However, when considering probabilistic prediction, since machine learning model only generates the optimal solution according to its training data, such inference will produce uncertainty of data and model parameters, which will bring unknown risks to major decisions of the algorithm. Therefore, the uncertainty quantification methods include frequentist methods and Bayesian methods are necessary. And for different uncertainty qualification methods, MIS [10] is seemed as the brilliant metric for evaluation.

MIS is defined to evaluate the merits of the prediction interval, α denotes confidence level, Z are real-valued random variables, the U and L are the upper and lower boundary of confidence interval, respectively, the score function contains three penalty terms, the first penalty term is the gap between the two bounds, the second is the situation that predicted values higher than upper bound, the third is predictions under the confidence lower bound, these three are mainly estimations of uncertainty. For large samples, MIS is defined by expectations:

$$\mathrm{MIS}_\infty(U, L; \alpha) = (U - L) + \frac{2}{\alpha}(\mathbb{E}[Z - U \mid Z > U] + \mathbb{E}[L - Z \mid Z < L]) \tag{2.7}$$

2.3. Evidential deep learning

2.3.1. Evidential regression network

Evidential regression network [31] considers the target value y which is a sample drawn from a normal distribution, but does not know the parameters of the normal distribution μ, σ. These two parameters are drawn from a high order evidential distribution, that is normal-inverse-gamma (NIG) distribution, which is the conjugate prior to the normal distribution:

$$\gamma \sim \mathcal{N}\left(\mu, \sigma^2\right), \quad \mu \sim \mathcal{N}\left(\gamma, \frac{\sigma^2}{\nu}\right), \quad \sigma^2 \sim \mathrm{Gamma}^{-1}(\alpha, \beta) \tag{2.8}$$

The NIG distribution can be parameterized by $\mathbf{m} = \gamma, \nu, \alpha, \beta$, where $\gamma \in \mathbb{R}, \nu > 0, \alpha > 1, \beta > 0$. And Gamma^{-1} is the inverse-gamma distribution.

Through these four parameters, we can get

$$\underbrace{\mathbb{E}[\mu] = \gamma,}_{\text{prediction}} \quad \underbrace{\mathbb{E}\left[\sigma^2\right] = \frac{\beta}{\alpha - 1},}_{\text{aleatoric}} \quad \underbrace{\operatorname{Var}[\mu] = \frac{\beta}{v(\alpha - 1)}}_{\text{epistemic}}. \tag{2.9}$$

2.3.2. Training the evidential regression network with the marginal likelihood

The evidential regression network can learn the trainable parameters θ by maximizing a marginal likelihood with unknown Gaussian parameters, μ, σ.

$$p(\underbrace{\mu, \sigma^2}_{\theta} \mid \underbrace{\gamma, v, \alpha, \beta}_{m}) = \frac{\beta^\alpha \sqrt{v}}{\Gamma(\alpha)\sqrt{2\pi\sigma^2}} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left\{-\frac{2\beta + v(\gamma - \mu)^2}{2\sigma^2}\right\} \tag{2.10}$$

Through Bayesian probability theory, we can get the following formula.

$$p\left(y_i \mid \boldsymbol{m}\right) = \frac{p\left(y_i \mid \theta, \boldsymbol{m}\right) p(\theta \mid \boldsymbol{m})}{p\left(\theta \mid y_i, \boldsymbol{m}\right)} \tag{2.11}$$

$$p\left(y_i \mid \boldsymbol{m}\right) = \int_{\sigma^2=0}^{\infty} \int_{\mu=-\infty}^{\infty} p\left(y_i \mid \mu, \sigma^2\right) p\left(\mu, \sigma^2 \mid \boldsymbol{m}\right) \mathrm{d}\mu \mathrm{d}\sigma^2 \tag{2.12}$$

From the formula given above, we can know that for the model prediction y, it is follows the student-t distribution.

$$p\left(y_i \mid m\right) = \operatorname{St}\left(y_i; \gamma, \frac{\beta(1 + v)}{v\alpha}, 2\alpha\right) \tag{2.13}$$

In order to maximizing the model fit, we can minimize the negative log marginal likelihood (NLL) loss during the training procedure of evidential regression network. The definition of NLL loss function is:

$$L_{NLL}(y, \mathbf{m}) = -\log(p(y \mid \mathbf{m})) \tag{2.14}$$

Replace p in the above equation with the probability density function of student-t distribution, we can get the following optimization formula:

$$\begin{aligned} \arg\min_\theta \left(L_{NLL}(y, \mathbf{m})\right) = \arg\min_\theta & \tfrac{1}{2} \log\left(\tfrac{\pi}{v}\right) - \alpha \log \Omega \\ & + \left(\alpha + \tfrac{1}{2}\right) \log\left((y - \gamma)^2 v + \Omega\right) + \log\left(\tfrac{\Gamma(\alpha)}{\Gamma\left(\alpha + \frac{1}{2}\right)}\right) \end{aligned} \tag{2.15}$$

Where $\Omega = 2\beta(1 + v)$.

2.3.3. Minimizing evidence deep learning on errors

Evidential deep learning also proposed a method to solve the problem that how to regularize training by applying an incorrect evidence penalty, or high uncertainty prior, which is in order to minimize evidence on incorrect predictions. In order to address this challenge, [31] formulate a new evidence regularizer as following, which scaled on the i-th prediction's error:

$$\mathcal{L}_i^{\mathrm{R}}(\boldsymbol{w}) = \left|y_i - \mathbb{E}\left[\mu_i\right]\right| \cdot \Phi = \left|y_i - \gamma\right| \cdot (2v + \alpha) \tag{2.16}$$

This loss function can make a penalty when there is an error in the model prediction and scales with the total evidentce of our inferred posterior. So it can realize minimizing the evidence on errors.

2.3.4. Final loss function

So the final loss function is:

$$\mathcal{L}_i(\boldsymbol{w}) = \mathcal{L}_i^{\mathrm{NLL}}(\boldsymbol{w}) + \lambda \mathcal{L}_i^{\mathrm{R}}(\boldsymbol{w}) \tag{2.17}$$

The λ is a regularization coefficient.

2.3.5. Multi-task-based evidential network

The basic goal of deep learning is to achieve good prediction accuracy, not just uncertainty. Therefore, in order to improve the prediction accuracy of evidential deep learning and it also can't interrupt its ability to estimate uncertainty. Lipschitz MSE loss function is proposed by [32], which as an additional loss function in original EDL loss function and it can mitigate gradient conflicts with the NLL loss that can improve the model prediction accuracy without disturb the ability of uncertainty estimation.

The Lipschitz MSE loss [33] is defined as follows:

$$L_{mse}^{*}(y_i, \gamma_i) = \begin{cases} (y_i - \gamma_i)^2, & \text{If } \lambda_i^2 < U_{\nu,\alpha} \\ 2\sqrt{U_{\nu,\alpha}}\,|y_i - \gamma_i| - U_{\nu,\alpha}, & \text{If } \lambda_i^2 \geq U_{\nu,\alpha} \end{cases} \tag{2.18}$$

where $U_{\nu,\alpha} \overset{\text{def}}{=} \min_{i\in[1:N]}\left(\min\left(U_{\nu_i}, U_{\alpha_i}\right)\right)$, and γ_i is model prediction and y_i is the ground truth.

The final loss function is obtained by combining this loss function with the original edl loss function.

$$\mathcal{L}_{total} = \frac{1}{N}\sum_{i=1}^{N} L_{NLL}(y_i, \mathbf{m}_i) + L_{mse}^{*}(y_i, \gamma_i) + cL_R(y_i, \mathbf{m}_i) \tag{2.19}$$

2.3.6. Mean square error evidential network

This method comes from the [32] and is also a method that can be used to experiment for contrast. Compared with Lipschitz modified MSE loss function, directly adding MSE loss function does not solve gradient loss, but theoretically, it can optimize the prediction accuracy of EDL to a certain extent. The final loss function of this method is the original loss function combined with MSE loss function.

3. Results and discussion

We hold experiments with these methods on dataset METR-LA traffic and Covid-19 incident: Point, EDL, MT-EDL, MSE-EDL, Bootstrap, quantile regression, MIS regression. And we choose MAE and RMSE for predictive accuracy metrics, MIS for uncertainty estimation metric.

3.1. Dataset

The METR-LA [11] dataset, which full name is METR-LA road network traffic. Compared with other traffic data sets, it has a more complex traffic environment. It has 4 months vehicle spped information from loop detector sensors in Los Angeles county highwat system. The task of this dataset is to forecast the traffic speed for 207 sensors simultaneously. For the distance between two sensors, we use a thresholded Gaussian kernel [34] recomputing the pairwise road network distances between sensor:

$$W_{ij} = \exp\left(-\frac{(dist(vi, vj))^2}{\sigma^2}\right) \tag{3.1}$$

The Covid-19 dataset contains two parts, one is the reported deaths from Johns Hopkins University, the other is the death predictions from a mechanistic, stochastic, and spatial metapopulation epidemic model called global epidemic and mobility model (GLEAM) [12]. The dataset records information for 50 US states from May 24th to Sep 12th, 2020. We use the residual between the reported death and the corresponding GLEAM predictions to train the model.

3.2. Implementation details

Table 1. All data is taken from the best training model.

METR-LA Dataset								
Horizen	Metric	Point	EDL	MT-EDL	MSE-EDL	Bootstrap	Quantile	MIS
15 min	MAE	2.9474	2.8922	2.8881	2.8816	3.1754	**2.4091**	**2.5639**
	RMSE	7.0494	7.2698	7.1418	8.7941	6.6432	**4.6284**	**4.7128**
	MIS	—	49.7711	54.9948	50.2000	39.5424	**20.8003**	**20.9707**
	Interval width	—	29.9560	26.3881	24.3664	5.0731	14.8636	14.8769
30 min	MAE	3.6837	3.4196	3.4009	3.4659	3.5616	**2.6825**	**2.9072**
	RMSE	8.4398	8.8610	8.7028	8.7368	7.2318	**5.4017**	**5.4382**
	MIS	—	63.5237	66.6163	64.2761	40.1293	**23.3577**	**23.3783**
	Interval width	—	28.6619	26.5174	27.8069	6.7621	16.0293	16.3625
60 min	MAE	4.7259	4.2664	4.1318	4.2277	4.3779	**3.0272**	**3.3969**
	RMSE	10.4362	10.6668	10.5993	10.8145	8.6117	**6.3312**	**6.3390**
	MIS	—	85.3857	81.0241	84.1179	40.0738	**26.7827**	**26.7918**
	Interval width	—	41.8989	41.7899	43.7021	11.1766	16.2501	17.0685
Covid-19 incident dataset								
Category	Metric	Point	EDL	MT-EDL	MSE-EDL	Bootstrap	Quantile	MIS
Week1	MAE	41.1191	41.1401	40.9577	41.6395	**39.9859**	40.8062	**40.2865**
	RMSE	75.4122	74.9475	74.4459	**73.9358**	**72.3700**	76.1414	76.4073
	MIS	—	467.5646	469.7116	**438.8426**	1524.6378	454.4100	**394.4060**
	Interval width	—	348.2935	354.7570	326.9575	4.0168	263.6418	333.4511
Week2	MAE	39.3436	38.0071	37.8595	38.4856	**37.6186**	38.1539	**37.6126**
	RMSE	72.8770	72.6179	**72.6117**	72.8098	**72.4540**	72.8411	72.9681
	MIS	—	**489.4117**	**491.3482**	492.4358	1460.4114	542.4969	525.1689
	Interval width	—	390.3598	353.2775	382.5128	1460.4114	542.4969	525.1689
Week3	MAE	**31.8655**	32.4491	32.1018	**31.5814**	33.4619	32.2205	32.2050
	RMSE	60.1569	**57.2531**	**56.8395**	57.6512	62.3746	61.4664	61.1469
	MIS	—	446.1847	441.5700	419.2901	1272.3398	**398.4482**	**394.4420**
	Interval width	—	326.4607	342.8699	329.5548	3.5607	234.8078	394.3024
Week4	MAE	**50.4617**	50.5435	50.7066	50.5536	**49.8641**	50.6653	50.7764
	RMSE	109.3619	**106.7713**	107.2596	**106.5863**	108.1830	110.4269	109.8570
	MIS	—	**740.0419**	757.4979	**746.7488**	1954.7770	766.9339	777.5025
	Interval width	—	342.1250	357.4564	361.2385	2.1519	377.3929	447.0502

Table 2. Computation cost of different methods.

Method	EDL	Bootstrap	Quantile	MIS
Computation	1	25	1	1

We adapt diffusion convolutional recurrent neural network with 6 methods of uncertainty measurement. For the confidence level selected for the experiment is 0.95. The following are the specific parameters of the experiment.

Point: Our parameter settings are the same as those in DCRNN [8]. And the point method only considers the accuracy of prediction.

EDL: We use the network parameters as same as the point estimation, but the difference is that DCRNN outputs four parameters needed to calculate EDL loss: $\mathbf{m} = \gamma, \nu, \alpha, \beta$, and the loss function is changed to EDL loss [31]. As for hyperparameter, we use $1E^{-4}$ as the evidential regularization factor.

MT-EDL: There is an additional Lipschitz modified MSE loss [32] as shown in Eq (2.20). Combined with the loss function of EDL throught a trainable parameter.

MSE-EDL: Just like MT-EDL, the Lipschitz modified MSE loss is replaced by MSE loss.

Bootstrap: The parameters of these methods are as same as [10]. We randomly drop half training data while keeping the original validation and testing data. We obtain 25 samples for constructing mean prediction and confidence intervals.

Quantile, MIS: We apply the quantile loss function with the corresponding quantile (0.025, 0.5, 0.975). The design of the MIS regression is very similar to that of the quantile, but combine MAE with MIS. Other parameters are set the same as point estimation.

How to calculate MIS: For EDL, MT-EDL, MSE-EDL, we can calculate the mis by Eqs (8) and (11).

3.3. Analysis of experimental results

From the definition of MIS, we know that the smaller of MIS, the better of uncertainty estimate and the higher of the model's credibility. From Table 1, we can see that in METR-LA Dataset, the quantile and MIS methods get the best performance in both accuracy and uncertainty estimation. For EDL, MT-EDL and MSE-EDL, there is no big gap between these three methods. MT-EDL is slightly better than EDL in prediction accuracy because of the addition of the Lipschitz modified MSE loss function and the resolution of gradient conflict with NLL loss function. MSE-EDL has added MSE loss function, but it does not solve the problem of gradient conflict, so it is not better than EDL in prediction accuracy. Such result is also consistent with the results of the paper [32].

For Covid-19 incident dataset, we find that the bootstrap method relatively achieved better prediction accuracy. However, for uncertainty quantification, bootstrap method is the worst. It can also be seen that the ability of uncertainty quantification of most methods is similar in Covid-19 dataset on the whole, in both cases of week2 and week4, EDL performed better in quantifying uncertainty. Moreover, as can be seen from Table 2, bootstrap needs much higher computation cost than other methods.

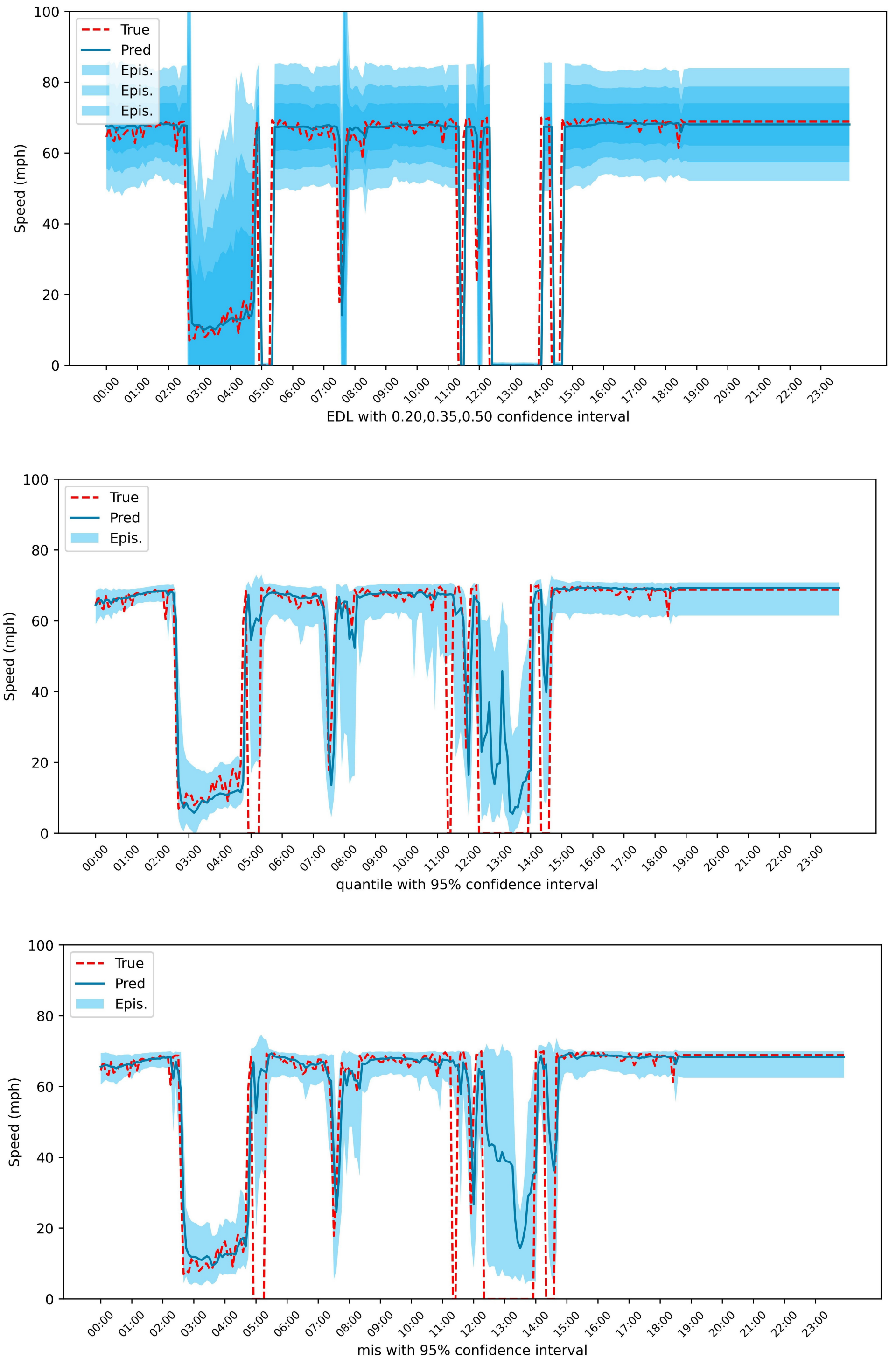

Figure 1. 1 hour ahead traffic forecasting over the 24 hour time span of sensor 717472 on June 30, 2012.

3.4. Forecasts visualization

To illustrate the results of the discussed methods more clearly, we also visualize the prediction results including the confidence interval. As can be seen from Figure 1, the quantile and MIS methods tend to generate narrow confidence interval. While for EDL, it tends to have a larger confidence interval. It is also worth noting that due to the nature of the methods, quantile and MIS can only obtain fixed confidence interval, e.g., 95% confidence interval. If different confidence intervals are needed, we have to retrain the models. While for EDL, it can obtain a whole distribution. As can also be seen from Figure 1, EDL can get any confidence interval according to our needs without retraining the model.

3.5. Advantages of evidential deep learning

Based the above discussion, EDL shows the advantages among existing methods. Specifically, compared with bootstrap, EDL does not require multiple sampling to generate confidence intervals, so the calculation time cost is lower. Compared with quantile and MIS methods, EDL can provide multiple confidence intervals in same training time. Moreover, in the Covid-19 dataset, EDL performs well in both prediction accuracy and uncertainty quantification.

4. Conclusions, limitations and scope

In order to solve the limitation of uncertainty quantification on graph neural network by Bayesian and frequentist. We applied EDL [31] and the extension of EDL [32] to N and did the experiment on METR-LA dataset and Covid-19 dataset. The experimental results show the effectiveness and efficiency of EDL to make uncertainty estimation in graph neural network task.

Although EDL has several advantages over other uncertainty quantification methods, we still have to admit that its performance needs to be further improved. For example, compared with the quantile method and MIS regression method, the uncertainty quantification score MIS is relatively high in some cases. We consider this is mainly because EDL tends to get larger confidence interval and we leave the further optimization in the future works. Moreover, we also plan to evaluate the performance of EDL on other uncertainty scoring rules, such as expected calibration error (ECE) [35]. At last, the crucial use of uncertainty estimation is to judge when the output of the model is not reliable, and further to finish the detection of out-of-distribution (OOD) input samples. In the existing deep learning works, most of the OOD data are detected by training samples. Since EDL does not need sampling and can fit the data to high-order normal distribution, it can realize low uncertainty for in-distribution data and high uncertainty for OOD data. In the future, we also plan to study the application of EDL, such as OOD detection.

Acknowledgments

The work was supported in part by NSF award #2153311. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies. This research was partially supported by NSFC #62002282 and MOE Innovation Research Team No. IRT_17R86.

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, preprint, arXiv:1609.02907. https://doi.org/10.48550/arXiv.1609.02907
2. A. Hasanzadeh, E. Hajiramezanali, K. R. Narayanan , N. Duffield, M. Zhou, X. Qian, Semi-implicit graph variational auto-encoders, *Adv. Neural Inf. Process. Syst.*, **32** (2019).
3. E. Hajiramezanali, A. Hasanzadeh, K. R. Narayanan , N. Duffield, M. Zhou, X. Qian, Variational graph recurrent neural networks, *Adv. Neural Inf. Process. Syst.*, **32** (2019).
4. Q. Li, Z. Han, X. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in *Thirty-Second AAAI Conference on Atificial Intelligence*, 2018. https://doi.org/10.1609/aaai.v32i1.11604
5. A. Hasanzadeh, E. Hajiramezanali, S. Boluki, M. Zhou, N. Duffield, K. R. Narayanan, et al., Bayesian graph neural networks with adaptive connection sampling, in *International Conference on Machine Learning*, (2020), 4094–4104.
6. M. Sensoy, L. M. Kaplan, M. Kandemir, Evidential deep learning to quantify classification uncertainty, *Adv. Neural Inf. Process. Syst.*, **31** (2018).
7. A. Malinin, M. J. F. Gales, Predictive uncertainty estimation via prior networks, *Adv. Neural Inf. Process. Syst.*, **31** (2018).
8. Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, preprint, arXiv:1707.01926. https://doi.org/10.48550/arXiv.1707.01926
9. R. Askanazi, F. X. Diebold, F. Schorfheide, M. Shin, On the comparison of interval forecasts, *J. Time Ser. Anal.*, **39** (2018), 953–965. https://doi.org/10.1111/jtsa.12426
10. D. Wu, L. Gao, X. Xiong, M. Chinazzi, A. Vespignani, Y. Ma, et al., Quantifying uncertainty in deep spatiotemporal forecasting, preprint, arXiv:2105.11982. https://doi.org/10.48550/arXiv.2105.11982
11. H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, et al., Big data and its technical challenges, *Commun. ACM*, **57** (2014), 86–94. https://doi.org/10.1145/2611567
12. D. Balcan, V. Colizza, B. Gonçalves, A. Vespignani, Multiscale mobility networks and the spatial spreading of infectious diseases, in *Proceedings of the National Academy of Sciences*, **106** (2009), 21484–21489. https://doi.org/10.1073/pnas.0906910106
13. J. Bruna, W. Zaremba, A. D. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, preprint, arXiv:1312.6203. https://doi.org/10.48550/arXiv.1312.6203
14. M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, *Adv. Neural Inf. Process. Syst.*, **29** (2016).

15. Y. Seo, M. Defferrard, P. Vandergheynst, X. Bresson, Structured sequence modeling with graph convolutional recurrent networks, in *Neural Information Processing: 25th International Conference*, (2018), 362–373. https://doi.org/10.1007/978-3-030-04167-0_33

16. S. J. Hanson, In advances in neural information processing systems, in *NIPS 1990*, 1990.

17. Y. Hechtlinger, P. Chakravarti, J. Qin, A generalization of convolutional neural networks to graph-structured data, preprint, arXiv:1704.08165. https://doi.org/10.48550/arXiv.1704.08165

18. A. Kendall, Y. Gal, What uncertainties do we need in Bayesian deep learning for computer vision, *Adv. Neural Inf. Process. Syst.*, **30** (2017).

19. P. Harris, H. Haris, Reliable prediction intervals with regression neural networks, *Neural Networks*, **24** (2011), 842–851. https://doi.org/10.1016/j.neunet.2011.05.008

20. I. Osband, C. Blundell, A. Pritzel, B. Van Roy, Deep exploration via bootstrapped DQN, *Adv. Neural Inf. Process. Syst.*, **29** (2016).

21. D. A. Nix, A. S. Weigend, Estimating the mean and variance of the target probability distribution, in *Proceedings of 1994 ieee international conference on neural networks*, (1994), 55–60. https://doi.org/10.1109/ICNN.1994.374138

22. F. Richter, Mixture density networks, in *Kombination Künstlicher Neuronaler Netze*, Deutscher Universitätsverlag, 2003.

23. I. Glitschenski, W. Schwarting, R. Sahoo, A. Amini, S. Karaman, Deep orientation uncertainty learning based on a bingham loss, in *International Conference on Learning Representations*, 2020.

24. D. P. Kingma, T. Salimans, M. Welling, Variational dropout and the local reparameterization trick, *Adv. Neural Inf. Process. Syst.*, **28** (2015).

25. Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in *International Conference on Machine Learning*, (2016), 1050–1059.

26. Y. A. Ma, T. Chen, E. Fox, A complete recipe for stochastic gradient MCMC, *Adv. Neural Inf. Process. Syst.*, **28** (2015).

27. M. Welling, Y. W. Teh, Bayesian learning via stochastic gradient langevin dynamics, in *Proceedings of the 28th international conference on machine learning*, (2011), 681–688.

28. J. B. Grill, F. Strub, F. Altch'e, C. Tallec, P. Richemond, E. Buchatskaya, et al., Bootstrap your own latent: A new approach to self-supervised learning, **33** (2020), 21271–21284.

29. F. Marilena, V. Domenico, *Quantile Regression*, Wiley series in Probability and Statistics, 2018.

30. J. Chung, C. Gulcehre, K. H. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, preprint, arXiv:1412.3555. https://doi.org/10.48550/arXiv.1412.3555

31. A. Alexander, S. Wilko, S. Ava, R. Daniela, Deep evidential regression, *Adv. Neural Inf. Process. Syst.*, **33** (2020), 14927–14937.

32. D. Oh, B. Shin, Improving evidential deep learning via multi-task learning, in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2022), 7895–7903. https://doi.org/10.1609/aaai.v36i7.20759

33. J. Qi, J. Du, S. M. Siniscalchi, X. Ma, C. Lee, On mean absolute error for deep neural network based vector-to-vector regression, *IEEE Signal Process. Lett.*, **27** (2020), 1485–1489. https://doi.org/10.1109/LSP.2020.3016837

34. D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, *IEEE Signal Process. Mag.*, **30** (2013), 83–98. https://doi.org/10.1109/MSP.2012.2235192

35. K. Kim, B. M. Ji, D. Yoon, S. Hwang, Self-knowledge distillation: A simple way for better generalization, preprint, arXiv:2006.12000.

36. H. Lin, Z. Gao, Y. Xu, L. Wu, L. Li, S. Z. Li, Conditional local convolution for spatio-temporal meteorological forecasting, in *Proceedings of the AAAI Conference on Artificial Intelligence*, **36** (2022), 7470–7478. https://doi.org/10.1609/aaai.v36i7.20711

37. X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, et al., Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting, in *Proceedings of the AAAI conference on artificial intelligence*, **33** (2019), 3656–3663. https://doi.org/10.1609/aaai.v33i01.33013656

38. J. E. Matheson, R. L. Winkler, Scoring rules for continuous probability distributions, *Manage. Sci.*, **22** (1976), 1087–1096. https://doi.org/10.1287/mnsc.22.10.1087

39. A. Vehtari, A. Gelman, J. Gabry, Practical bayesian model evaluation using leave-one-out cross-validation and WAIC, *Stat. Comput.*, **27** (2017), 1413–1432. https://doi.org/10.1007/s11222-016-9696-4

40. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskeve, R. Salakhutdino, Dropout: A simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.*, **15** (2014), 1929–1958.

41. J. Atwood, D. Towsley, Diffusion-convolutional neural networks, *Adv. Neural Inf. Process. Syst.*, **29** (2016).

42. A. Amini, W. Schwarting, A. P. Soleimany, D. Rus, Deep evidential regression, *Adv. Neural Inf. Process. Syst.*, **33** (2020), 14927–14937.