RNN with Particle Flow for Probabilistic Spatio-temporal Forecasting

Soumyasundar Pal[† 1] **Liheng Ma**[† 1] **Yingxue Zhang**[2] **Mark Coates**[1]

Abstract

Spatio-temporal forecasting has numerous applications in analyzing wireless, traffic, and financial networks. Many classical statistical models often fall short in handling the complexity and high non-linearity present in time-series data. Recent advances in deep learning allow for better modelling of spatial and temporal dependencies. While most of these models focus on obtaining accurate point forecasts, they do not characterize the prediction uncertainty. In this work, we consider the time-series data as a random realization from a nonlinear state-space model and target Bayesian inference of the hidden states for probabilistic forecasting. We use particle flow as the tool for approximating the posterior distribution of the states, as it is shown to be highly effective in complex, high-dimensional settings. Thorough experimentation on several real world time-series datasets demonstrates that our approach provides better characterization of uncertainty while maintaining comparable accuracy to the state-of-the-art point forecasting methods.

1. Introduction

Spatio-temporal forecasting has many applications in intelligent traffic management, computational biology and finance, wireless networks and demand forecasting. Inspired by the surge of novel learning methods for graph structured data, many deep learning based spatio-temporal forecasting techniques have been proposed recently (Li et al., 2018; Bai et al., 2020). In addition to the temporal patterns present in the data, these approaches can effectively learn and exploit spatial relationships among the time-series using various Graph Neural Networks (GNNs) (Defferrard et al., 2016; Kipf & Welling, 2017). Recent works establish that graph-based spatio-temporal models outperform the graph-agnostic baselines (Li et al., 2018; Wu et al., 2020). In spite of their accuracy in providing point forecasts, these models have a serious drawback as they cannot gauge the uncertainty in their predictions. When decisions are made based on forecasts, the availability of a confidence or prediction interval can be vital.

There are numerous probabilistic forecasting techniques for multivariate time-series, for example, DeepAR (Salinas et al., 2020), DeepState (Rangapuram et al., 2018), DeepFactors (Wang et al., 2019), and the normalizing flow-based algorithms in (Kurle et al., 2020; Rasul et al., 2021). Although these algorithms can characterize uncertainty via confidence intervals, they are not designed to incorporate side-knowledge provided in the form of a graph.

In this work, we model multivariate time-series as random realizations from a nonlinear state-space model, and target Bayesian inference of the hidden states for probabilistic forecasting. The general framework we propose can be applied to univariate or multivariate forecasting problems, can incorporate additional covariates, can process an observed graph, and can be combined with data-adaptive graph learning procedures. For the concrete example algorithm deployed in experiments, we build the dynamics of the state-space model using graph convolutional recurrent architectures. We develop an inference procedure that employs particle flow, an alternative to particle filters, that can conduct more effective inference for high-dimensional states.

The novel contributions in this paper are as follows:
1) we propose a graph-aware stochastic recurrent network architecture and inference procedure that combine graph convolutional learning, a probabilistic state-space model, and particle flow;
2) we demonstrate via experiments on graph-based traffic datasets that a specific instantiation of the proposed framework can provide point forecasts that are as accurate as the state-of-the-art deep learning based spatio-temporal models. The prediction error is also comparable to the existing deep learning based techniques for benchmark non-graph multivariate time-series datasets;
3) we show that the proposed method provides a superior characterization of the prediction uncertainty compared to

[1]Department of Electrical and Computer Eningeering, McGill University, Montréal, QC, Canada. [2] Huawei Noah's Ark Lab, Montréal Research Center, Montréal, QC, Canada. Correspondence to: Soumyasundar Pal <soumyasundar.pal@mail.mcgill.ca>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

† Part of the research performed as an intern at Huawei Noah's Ark Lab, Montréal Research Center.

existing probabilistic multivariate time-series forecasting methods, both for datasets where a graph is available and for settings where no graph is available.

2. Related Work

Our work is related to (i) multivariate and spatio-temporal forecasting using deep learning and graph neural networks; (ii) stochastic/probabilistic modelling, prediction and forecasting for multivariate time-series; and (iii) neural (ordinary) differential equations. Recently, neural network-based techniques have started to offer the best predictive performance for multivariate time-series prediction (Bao et al., 2017; Qin et al., 2017; Lai et al., 2018; Guo & Lin, 2018; Chang et al., 2018; Li et al., 2019; Sen et al., 2019; Oreshkin et al., 2020; Smyl, 2020). In some settings, a graph is available that specifies spatial or causal relationships between the time-series. Numerous algorithms have been proposed that combine GNNs with temporal neural network architectures (Li et al., 2018; Yu et al., 2018; Huang et al., 2019; Bai et al., 2019; Chen et al., 2019a; Guo et al., 2019; Wu et al., 2019; Yu et al., 2019; Zhao et al., 2019; Bai et al., 2020; Huang et al., 2020; Park et al., 2020; Shi et al., 2020; Song et al., 2020; Xu et al., 2020; Wu et al., 2020; Zheng et al., 2020; Oreshkin et al., 2021). Algorithms that take into account the graph provide superior forecasts, if the graph is accurate and the indicated relationships have predictive power. However, none of these algorithms is capable of characterizing the uncertainty of the provided predictions; all are constructed as deterministic algorithms.

Recently, powerful multivariate forecasting algorithms that are capable of providing uncertainty characterization have been proposed. These include DeepAR (Salinas et al., 2020), DeepState (Rangapuram et al., 2018), the Multi-horizon Quantile RNN (MQRNN) (Wen et al., 2017), the Gaussian copula process approach of (Salinas et al., 2019), and DeepFactors (Wang et al., 2019). Normalizing flow has also been combined with temporal NN architectures (Kumar et al., 2019; de Bézenac et al., 2020; Gammelli & Rodrigues, 2020; Rasul et al., 2021). Various flavours of stochastic recurrent networks have also been introduced (Boulanger-Lewandowski et al., 2012; Bayer & Osendorfer, 2014; Chung et al., 2015; Fraccaro et al., 2016; 2017; Karl et al., 2016; Mattos et al., 2016; Doerr et al., 2018). In most cases, variational inference is applied to learn model parameters, although sequential Monte Carlo has also been employed (Gu et al., 2015; Le et al., 2017; Maddison et al., 2017; Zheng et al., 2017; Karkus et al., 2018; Naesseth et al., 2018; Ma et al., 2020). These are related to methods that determine the parameters of sequential Monte Carlo models via optimizing Monte Carlo objectives (Maddison et al., 2017; Naesseth et al., 2018; Le et al., 2017).

Our proposed method is different from this body of work in two important ways. First, we design a probabilistic state-space modeling framework that can incorporate information about predictive relationships that is provided in the form of a graph. Second, our inference procedure employs particle flow, which avoids the need for some of the approximations required by a variational inference framework and is much better suited to high-dimensional states than particle filtering. Our particle flow method has connections to normalizing flows (Kobyzev et al., 2020) and to neural ordinary differential equations (Chen et al., 2018; 2019b). In particular, Chen et al. (2019b) address a Bayesian inference task by solving a differential equation to transport particles from the prior to the posterior distribution. However, such flow-based methods were first introduced by Daum & Huang (2007) in the sequential inference research literature.

3. Problem Statement

We address the task of discrete-time multivariate time-series prediction, with the goal of forecasting multiple time-steps ahead. We assume that there is access to a historical dataset for training, but after training the model must perform prediction based on a limited window of historical data. Let $\mathbf{y}_t \in \mathbb{R}^{N\times 1}$ be an observed multivariate signal at time t and $\mathbf{Z}_t \in \mathbb{R}^{N\times d_z}$ be an associated set of covariates. The i-th element of $\mathbf{y}_t$ is the observation associated with time-series i at time-step t.

We also allow for the possibility that there is access to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of N nodes and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ denotes the set of edges. In this case, each node corresponds to one time-series. The edges indicate probable predictive relationships between the variables, i.e., the presence of an edge (i, j) suggests that the historical data for time-series i is likely to be useful in predicting time-series j. The graph may be directed or undirected.

The goal is to construct a model that is capable of processing, for some time offset t_0, the data $\mathbf{Y}_{t_0+1:t_0+P}$, $\mathbf{Z}_{t_0+1:t_0+P+Q}$ and (possibly) the graph $\mathcal{G}$, to estimate $\mathbf{Y}_{t_0+P+1:t_0+P+Q}$. The prediction algorithm should produce both point estimates and prediction intervals. The performance metrics for the point estimates include mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean squared error (RMSE). For the prediction intervals, the performance metrics include the Continuous Ranked Probability Score (CRPS) (Gneiting & Raftery, 2007), and the P10, P50, and P90 Quantile Losses (QL) (Salinas et al., 2020; Wang et al., 2019). Expressions for these performance metrics are provided in the supplementary material.

4. Background: Particle Flow

Particle flow is an alternative to particle filtering for Bayesian filtering (and prediction) in a state-space model. The filtering task is to approximate the posterior distribu-

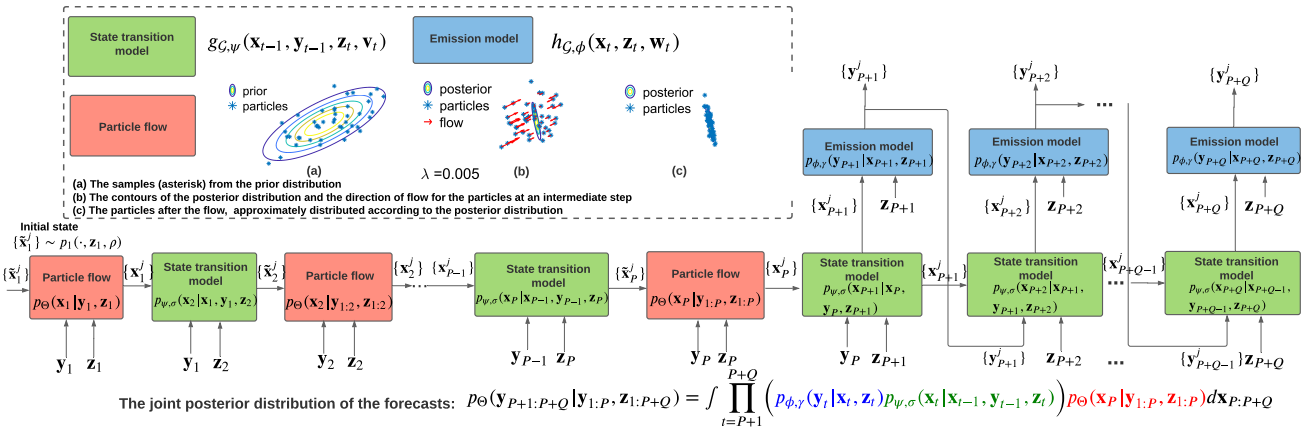

Figure 1. Probabilistic forecasting from the state-space model using particle flow. Migration of particles from a 2-d Gaussian prior to a 2-d Gaussian posterior distribution is illustrated as an example.

tion of the state trajectory $p_\Theta(\mathbf{x}_t|\mathbf{y}_{1:t})$ recursively, where $\mathbf{x}_t$ denotes the state at time t and $\mathbf{y}_{1:t}$ are observations from times 1 to t. A particle filter (Gordon et al., 1993; Doucet & Johansen, 2009) maintains a population of N_p samples (particles) and associated weights $\{\mathbf{x}_t^j, w_t^j\}_{j=1}^{N_p}$ that it uses to approximate the marginal posterior distribution of $\mathbf{x}_t$:

$$p_\Theta(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \frac{1}{N_p}\sum_{j=1}^{N_p} w_t^{(j)}\delta(\mathbf{x}_t - \mathbf{x}_t^j)\,. \tag{1}$$

Here, $\delta(\cdot)$ denotes the Dirac-delta function. Particles are propagated by the application of importance sampling using a proposal distribution; the weights are updated accordingly. When the disparity in the weights becomes too great, resampling is applied, with particles being sampled proportionally to their weights and the weights being reset to 1. Constructing well-matched proposal distributions to the posterior distribution in high-dimensional state-spaces is extremely challenging. A mismatch between the proposal and the posterior leads to weight degeneracy after resampling, which results in poor performance of particle filters in high-dimensional problems (Bengtsson et al., 2008; Snyder et al., 2008; Beskos et al., 2014). Instead of sampling, particle flow filters offer a significantly better solution by transporting particles continuously from the prior to the posterior (Daum & Huang, 2007; Ding & Coates, 2012).

For a given time step t, particle flow algorithms solve differential equations to gradually migrate particles from the predictive distribution so that they represent the posterior distribution after the flow. A particle flow can be modelled by a background stochastic process η_λ in a pseudo-time interval $\lambda \in [0, 1]$, such that the distribution of η_0 is the predictive distribution $p_\Theta(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ and the distribution of η_1 is the posterior distribution $p_\Theta(\mathbf{x}_t|\mathbf{y}_{1:t})$.

One approach (Daum et al., 2010), is to use an ordinary differential equation (ODE) with zero diffusion to govern the flow of η_λ:

$$\frac{d\eta_\lambda}{d\lambda} = \varphi(\eta_\lambda, \lambda)\ . \tag{2}$$

For linear Gaussian state-space models, the flow can be expressed in the form:

$$\varphi(\eta_\lambda, \lambda) = A(\lambda)\eta_\lambda + b(\lambda)\ , \tag{3}$$

and we can derive analytical expressions for $A(\lambda)$ and $b(\lambda)$ (see supplementary material for details). For non-linear and non-Gaussian models, we employ Gaussian approximations and repeated local linearizations.

5. Methodology

5.1. State-space model

We postulate that $\mathbf{y}_t \in \mathbb{R}^{N\times 1}$ is the observation from a Markovian state space model with hidden state $\mathbf{X}_t \in \mathbb{R}^{N\times d_x}$. We denote by $\mathbf{x}_t$ and $\mathbf{z}_t$ the vectorizations of $\mathbf{X}_t$ and $\mathbf{Z}_t$, respectively. The state space model is:

$$\mathbf{x}_1 \sim p_1(\cdot, \mathbf{z}_1, \rho)\,, \tag{4}$$

$$\mathbf{x}_t = g_{\mathcal{G},\psi}(\mathbf{x}_{t-1}, \mathbf{y}_{t-1}, \mathbf{z}_t, \mathbf{v}_t),\ \text{for } t > 1\,, \tag{5}$$

$$\mathbf{y}_t = h_{\mathcal{G},\phi}(\mathbf{x}_t, \mathbf{z}_t, \mathbf{w}_t),\ \text{for } t \geqslant 1\,. \tag{6}$$

Here $\mathbf{v}_t \sim p_v(\cdot|\mathbf{x}_{t-1}, \sigma)$ and $\mathbf{w}_t \sim p_w(\cdot|\mathbf{x}_t, \gamma)$ are the noises in the dynamic and measurement models respectively. ρ, σ and γ are the parameters of the distribution of the initial state $\mathbf{x}_1$, process noise $\mathbf{v}_t$ and measurement noise $\mathbf{w}_t$ respectively. g and h denote the state transition and measurement functions, possibly linear or nonlinear, with parameters ψ and ϕ respectively. The subscript $\mathcal{G}$ in g and h indicates that the functions are potentially dependent on the graph topology. We assume that $h_{\mathcal{G},\phi}(\mathbf{x}_t, \mathbf{z}_t, 0)$ is a $\mathcal{C}^1$ function in $\mathbf{x}_t$, i.e., $h_{\mathcal{G},\phi}(\mathbf{x}_t, \mathbf{z}_t, \mathbf{0})$ is a differentiable function whose first derivative w.r.t. $\mathbf{x}_t$ is continuous. The complete set of the unknown parameters is formed as: $\Theta = \{\rho, \psi, \sigma, \phi, \gamma\}$. Figure 2 depicts the graphical model relating the observed

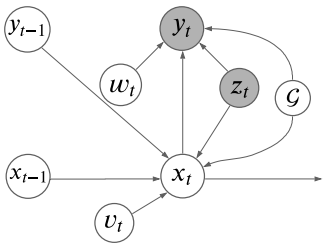

Figure 2. The graphical model representation of the state-space model in Section 5.1

variables ($\mathbf{y}_t$ and $\mathbf{z}_t$) to the latent variables (v_t, w_t) and the graph ($\mathcal{G}$).

With the proposed formulation, we can modify recurrent graph convolutional architectures when designing the function g. When a meaningful graph is available, such architectures significantly outperform models that ignore the graph. For example, we conduct experiments by incorporating into our general model the Adaptive Graph Convolutional Gated Recurrent Units (AGCGRU) presented in (Bai et al., 2020). The AGCGRU combines (i) a module that adapts the provided graph based on observed data, (ii) graph convolution to capture spatial relations, and (iii) a GRU to capture evolution in time. The example model used for experiments thus employs an L-layer AGCRU with additive Gaussian noise to model the system dynamics g:

$$\mathbf{x}_t = AGCGRU^{(L)}_{\mathcal{G},\psi}(\mathbf{x}_{t-1}, \mathbf{y}_{t-1}, \mathbf{z}_t) + \mathbf{v}_t\,, \tag{7}$$

$$\mathbf{y}_t = \mathbf{W}_\phi \mathbf{x}_t + \mathbf{w}_t\,. \tag{8}$$

In this model, we have $p_v(\mathbf{v}_t) = \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, i.e., the latent variables for the dynamics are independent. The initial state distribution is also chosen to be isotropic Gaussian, i.e., $p_1(\mathbf{x}_1, \mathbf{z}_1, \rho) = \mathcal{N}(\mathbf{0}, \rho^2\mathbf{I})$. The parameters ρ and σ are learnable variance parameters. The observation model g incorporates a linear projection matrix $\mathbf{W}_\phi$. The latent variable $\boldsymbol{w}_t$ for the emission model is modelled as Gaussian with variance dependent on $\mathbf{x}_t$ via a learnable softplus function:

$$p_w(\mathbf{w}_t|\mathbf{x}_t) = \mathcal{N}\Big(\mathbf{0}, \mathrm{diag}\big(\mathrm{softplus}(\boldsymbol{C}_\gamma \boldsymbol{x}_t)\big)^2\Big)\,. \tag{9}$$

5.2. Inference

We assume that a dataset $\mathcal{D}_{trn}$ is available for training. Although this data may be derived from a single time-series, because our task is to predict $\mathbf{y}_{t_0+P+1:t_0+P+Q}$ using a limited historical window $\mathbf{y}_{t_0+1:t_0+P}$, we splice the time-series and thus construct multiple training examples, denoted by $(\mathbf{y}^{(m)}_{1:P}, \mathbf{y}^{(m)}_{P+1:P+Q})$. In the training set, all of these observations are available; in the test set $\mathbf{y}_{P+1:P+Q}$ are not. In addition, the associated covariates $\mathbf{z}_{1:P+Q}$ are known for both training and test sets.

Inference involves an iterative process. We randomly initialize the parameters of the model to obtain Θ_0. Subsequently, at the k-th iteration of the algorithm (processing the k-th training batch), we first draw samples $\{\mathbf{y}^i_{P+1:P+Q}\}_{i=1}^{N_p}$ from the distribution $p_{\Theta_{k-1}}(\mathbf{y}_{P+1:P+Q}|\mathbf{y}_{1:P}, \mathbf{z}_{1:P+Q})$. With this set of samples, we can subsequently apply a gradient descent procedure to obtain the updated model parameters Θ_k. We discuss each of these steps in turn as follows.

5.2.1. Sampling

In a Bayesian setting with *known* model parameters $\Theta = \Theta_{k-1}$, we would aim to form a prediction by approximating the posterior distribution of the forecasts, i.e., $p_\Theta(\mathbf{y}_{P+1:P+Q}|\mathbf{y}_{1:P})$. (For conciseness we drop the time-offset t_0).

$$p_\Theta(\mathbf{y}_{P+1:P+Q}|\mathbf{y}_{1:P}, \mathbf{z}_{1:P+Q}) = \int \prod_{t=P+1}^{P+Q} \Big(p_{\phi,\gamma}(\mathbf{y}_t|\mathbf{x}_t, \mathbf{z}_t)$$

$$p_{\psi,\sigma}(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{t-1}, \mathbf{z}_t)\Big) p_\Theta(\mathbf{x}_P|\mathbf{y}_{1:P}, \mathbf{z}_{1:P}) d\mathbf{x}_{P:P+Q}\,. \tag{10}$$

Since the integral in eq. (10) is analytically intractable for a general nonlinear state-space model, we take a Monte Carlo approach as follows:

Step 1: For $1 \leqslant t \leqslant P$, we apply a particle flow algorithm (details in Sec. 4) with N_p particles for the state-space model specified by eqs. (4), (5) and (6) to recursively approximate the posterior distribution of the states:

$$p_\Theta(\mathbf{x}_t|\mathbf{y}_{1:t}, \mathbf{z}_{1:t}) \approx \frac{1}{N_p}\sum_{j=1}^{N_p} \delta(\mathbf{x}_t - \mathbf{x}^i_t)\,. \tag{11}$$

Here $\{\mathbf{x}^j_t\}_{j=1}^{N_p}$ are approximately distributed according to the posterior distribution of $\mathbf{x}_t$. The generation of each sample $\mathbf{x}^j_t$ involves an associated sampling of the latent variables $\mathbf{v}_{1:t}$ and implies a sampling of $\mathbf{w}_{1:t}$, but these samples are not required since the proposed model only needs $\mathbf{x}_P$ to construct the forecast,

Step 2: For $P+1 \leqslant t \leqslant P+Q$, we iterate between the following two steps:

a. We sample $\mathbf{x}^j_t$ from $p_{\psi,\sigma}(\mathbf{x}_t|\mathbf{x}^j_{t-1}, \mathbf{y}^j_{t-1}, \mathbf{z}_t)$ (for $t > P+1$) or from $p_{\psi,\sigma}(\mathbf{x}_t|\mathbf{x}^j_{t-1}, \mathbf{y}_{t-1}, \mathbf{z}_t)$ (for $t = P+1$) for $1 \leqslant j \leqslant N_p$. This amounts to a state transition at time t to obtain the current state $\mathbf{x}^j_t$ from the previous state $\mathbf{x}^j_{t-1}$, using eq. (5).

b. We sample $\mathbf{y}^j_t$ from $p_{\phi,\gamma}(\mathbf{y}_t|\mathbf{x}^j_t, \mathbf{z}_t)$ for $1 \leqslant j \leqslant N_p$, i.e., we use $\mathbf{x}^j_t$ in the measurement model, specified by eq. (6), to sample $\mathbf{y}^j_t$.

A Monte Carlo (MC) approximation of the integral in eq. (10) is then formed as:

$$p_\Theta(\mathbf{y}_{P+1:P+Q}|\mathbf{y}_{1:P}, \mathbf{z}_{1:P+Q}) \approx \prod_{t=P+1}^{P+Q} \frac{1}{N_p}\sum_{j=1}^{N_p} \delta(\mathbf{y}_t - \mathbf{y}^j_t)\,. \tag{12}$$

Each $\mathbf{y}^j_{P+1:P+Q}$ is approximately distributed according to the joint posterior distribution of $\mathbf{y}_{P+1:P+Q}$. The resulting algorithm is summarized in Algorithm 1. A block diagram of the probabilistic forecasting procedure is shown in Figure 1.

Algorithm 1 Sequence to sequence prediction

Input: $\mathbf{y}_{1:P}, \mathbf{z}_{1:P+Q}$, and Θ
Output: $\{\mathbf{y}^j_{P+1:P+Q}\}_{j=1}^{N_p}$
Initialization: Sample $\eta_0^j \sim p_1(\mathbf{x}_1, \mathbf{z}_1, \rho)$, $j = 1 : N_p$.
for $t = 1, 2, ..., P$ **do**
  if $t > 1$ **then**
    Sample $\eta_0^j \sim p_{\psi,\sigma}(\mathbf{x}_t|\mathbf{x}^j_{t-1}, \mathbf{y}_{t-1}, \mathbf{z}_t)$, $j = 1 : N_p$ as: $\eta_0^j = g_{\mathcal{G},\psi}(\mathbf{x}^j_{t-1}, \mathbf{y}_{t-1}, \mathbf{z}_t, \mathbf{v}_t)$.
  end if
  Use particle flow (details in supplementary material) to obtain $\{\eta_1^j\}_{j=1}^{N_p}$ from $\{\eta_0^j\}_{j=1}^{N_p}$, $\mathbf{z}_t$, and $\mathbf{y}_t$.
  Set $\mathbf{x}_t^j = \eta_1^j$.
end for
for $t = P+1, P+2, ..., P+Q$ **do**
  if $t = P+1$ **then**
    Sample $\mathbf{x}^j_{P+1} \sim p_{\psi,\sigma}(\mathbf{x}_{P+1}|\mathbf{x}^j_P, \mathbf{y}_P, \mathbf{z}_{P+1})$, $j = 1 : N_p$ as: $\mathbf{x}^j_{P+1} = g_{\mathcal{G},\psi}(\mathbf{x}^j_P, \mathbf{y}_P, \mathbf{z}_{P+1}, \mathbf{v}_{P+1})$.
  else
    Sample $\mathbf{x}_t^j \sim p_{\psi,\sigma}(\mathbf{x}_t|\mathbf{x}^j_{t-1}, \mathbf{y}^j_{t-1}, \mathbf{z}_t)$, $j = 1 : N_p$ as: $\mathbf{x}_t^j = g_{\mathcal{G},\psi}(\mathbf{x}^j_{t-1}, \mathbf{y}^j_{t-1}, \mathbf{z}_t, \mathbf{v}_t)$.
  end if
  Sample $\mathbf{y}_t^j \sim p_{\phi,\gamma}(\mathbf{y}_t|\mathbf{x}_t^j, \mathbf{z}_t)$, $j = 1 : N_p$ as: $\mathbf{y}_t^j = h_{\mathcal{G},\phi}(\mathbf{x}_t^j, \mathbf{z}_t, \mathbf{w}_t)$.
end for
Form the Monte Carlo estimate using eq. (12).

5.2.2. Parameter Update

With the predictive samples $\{\mathbf{y}^j_{P+1:P+Q}\}_{j=1}^{N_p}$, we can update the model parameters via Stochastic Gradient Descent (SGD) to obtain $\Theta = \Theta_k$.

If our focus is on obtaining a point estimate, then we can perform optimization on the training set with respect to a loss function derived from Mean Absolute Error (MAE) or Mean Square Error (MSE). The point forecast $\hat{\mathbf{y}}^{(m)}_{P+1:P+Q}$ is obtained based on a statistic such as the mean or median of the samples $\{\mathbf{y}^{j,(m)}_{P+1:P+Q}\}_{j=1}^{N_p}$. The MAE loss function on a dataset indexed by $\mathcal{D}$ can then be expressed as:

$$\mathcal{L}_{\text{MAE}}(\Theta, \mathcal{D}) = \frac{1}{NQ|\mathcal{D}|} \sum_{m \in \mathcal{D}} \sum_{t=P+1}^{P+Q} ||\mathbf{y}_t^{(m)} - \hat{\mathbf{y}}_t^{(m)}||_1 \,. \tag{13}$$

In an alternate approach, we could consider the maximization of the marginal log-likelihood over the training set. In that case, a suitable loss function is:

$$\mathcal{L}_{\text{prob}}(\Theta, \mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{m \in \mathcal{D}} \log p_\Theta(\mathbf{y}^{(m)}_{P+1:P+Q}|\mathbf{y}^{(m)}_{1:P}, \mathbf{z}^{(m)}_{1:P+Q}) \,, \tag{14}$$

where we approximate the posterior probability as:

$$\widehat{p}_\Theta(\mathbf{y}_{P+1:P+Q}|\mathbf{y}_{1:P}, \mathbf{z}_{1:P+Q}) = \prod_{t=P+1}^{P+Q} \left[\frac{1}{N_p} \sum_{j=1}^{N_p} p_{\phi,\gamma}(\mathbf{y}_t|\mathbf{x}_t^j, \mathbf{z}_t) \right],$$

using eq. (10). This loss formulation is similar to the MC variational objectives in (Maddison et al., 2017; Naesseth et al., 2018; Le et al., 2017). If we use the particle flow particle filter (Li & Coates, 2017), then the sampled particles and the propagated forecasts form an unbiased approximation of the distribution $p_\Theta(\mathbf{y}_{P+1:P+Q}|\mathbf{y}_{1:P}, \mathbf{z}_{1:P+Q})$. By Jensen's inequality, the summation over the log terms in (14) is thus a lower bound for the desired $\mathbb{E}[\log p_\Theta(\mathbf{y}_{P+1:P+Q}|\mathbf{y}_{1:P}, \mathbf{z}_{1:P+Q})]$ that converges as $N_p \to \infty$.

In each training mini-batch, for each training example, we perform a forward pass through the model using Algorithm 1 to obtain approximate forecast posteriors and then update all the model parameters using SGD via backpropagation.

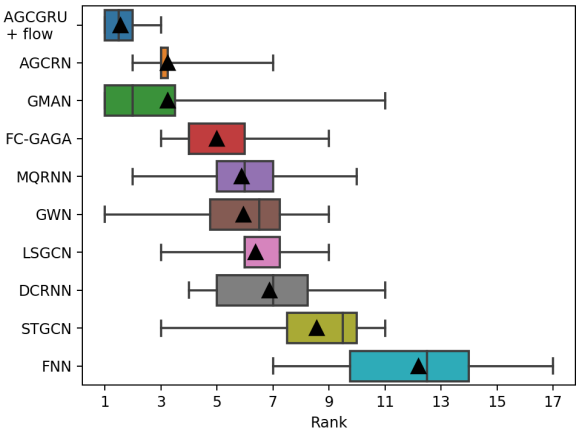

Figure 3. Boxplot of ranks of the top 10 algorithms across the four traffic datasets. The means of the ranks are shown by the black triangles; whiskers extend to the minimum and maximum ranks.

6. Experiments

We perform experiments on four graph-based and four non-graph based public datasets to evaluate proposed methods.

6.1. Datasets

We evaluate our proposed algorithm on four publicly available traffic datasets, namely PeMSD3, PeMSD4, PeMSD7 and PeMSD8. These are obtained from the Caltrans Performance Measurement System (PeMS) (Chen et al., 2000) and have been used in multiple previous works (Yu et al., 2018; Guo et al., 2019; Song et al., 2020; Bai et al., 2020; Huang et al., 2020). Each of these datasets consists of the traffic speed records, collected from loop detectors, and aggregated over 5 minute intervals, resulting in 288 data points per detector per day. In non-graph setting, we use Electricity (Dua

Table 1. Average MAE for PeMSD3, PeMSD4, PeMSD7, and PeMSD8 for 15/30/45/60 minutes horizons. The best and the second best results in each column are shown in bold and marked with underline respectively. Lower numbers are better.

Algorithm	MAE (15/ 30/ 45/ 60 min)			
	PeMSD3	PeMSD4	PeMSD7	PeMSD8
DCRNN	14.42/15.87/17.10/18.29	1.38/1.78/2.06/2.29	2.23/3.06/3.67/4.18	1.16/1.49/1.70/1.87
STGCN	15.22/17.54/19.74/21.59	1.42/1.85/2.14/2.39	2.21/2.96/3.47/3.90	1.22/1.56/1.79/1.98
GWN	14.63/16.56/18.34/20.08	1.37/1.76/2.03/2.24	2.23/3.03/3.56/3.98	**1.11**/1.40/1.59/1.73
GMAN	14.73/15.44/16.15/16.96	1.38/**1.61/1.76/1.88**	2.40/2.76/**2.98/3.16**	1.23/**1.36/1.46/1.55**
AGCRN	14.20/15.34/16.28/17.38	1.41/1.67/1.84/2.01	2.19/2.81/3.15/3.42	1.16/1.39/1.53/1.67
LSGCN	14.28/16.08/17.77/19.23	1.40/1.78/2.03/2.20	2.23/2.99/3.50/3.95	1.21/1.54/1.75/1.89
FC-GAGA	14.68/15.85/16.40/17.04	1.43/1.78/1.95/2.06	2.22/2.85/3.18/3.36	1.18/1.47/1.62/1.72
DeepAR	15.84/18.15/20.30/22.64	1.51/2.01/2.38/2.68	2.53/3.61/4.48/5.20	1.25/1.61/1.87/2.10
DeepFactors	17.53/20.17/22.78/24.87	1.54/2.01/2.34/2.61	2.51/3.47/4.17/4.71	1.26/1.63/1.88/2.07
MQRNN	14.60/16.55/18.34/20.12	1.37/1.76/2.03/2.25	2.22/3.03/3.58/4.00	1.13/1.43/1.62/1.77
AGCGRU+flow	**13.79/14.84/15.58/16.06**	**1.35**/1.63/1.78/**1.88**	**2.15/2.70**/2.99/3.19	1.13/1.37/1.49/1.57

Table 2. Average MAE for PeMSD3, PeMSD4, PeMSD7, and PeMSD8 for 15/30/45/60 minutes horizons for the proposed flow based approach and deterministic encoder-decoder models. Lower numbers are better.

Algorithm	MAE (15/ 30/ 45/ 60 min)			
	PeMSD3	PeMSD4	PeMSD7	PeMSD8
AGCGRU+flow	**13.79/14.84/15.58/16.06**	**1.35/1.63/1.78/1.88**	**2.15/2.70/2.99/3.19**	**1.13/1.37/1.49/1.57**
FC-AGCGRU	13.96/15.37/16.52/17.45	1.37/1.74/2.00/2.20	2.21/2.99/3.56/4.05	1.16/1.48/1.70/1.87
DCGRU+flow	14.48/**15.67/16.52/17.36**	**1.38/1.71/1.92/2.08**	**2.19/2.87/3.29/3.61**	1.17/**1.44/1.58/1.70**
FC-DCGRU	**14.42**/15.87/17.10/18.29	**1.38**/1.78/2.06/2.29	2.23/3.06/3.67/4.18	**1.16**/1.49/1.70/1.87
GRU+flow	**14.40/16.10/17.63/19.18**	**1.37/1.76/2.02/2.23**	**2.24/3.02/3.55/3.96**	**1.12/1.41/1.59/1.74**
FC-GRU	15.82/18.37/20.61/22.93	1.46/1.91/2.25/2.54	2.41/3.40/4.17/4.84	1.20/1.56/1.81/2.02

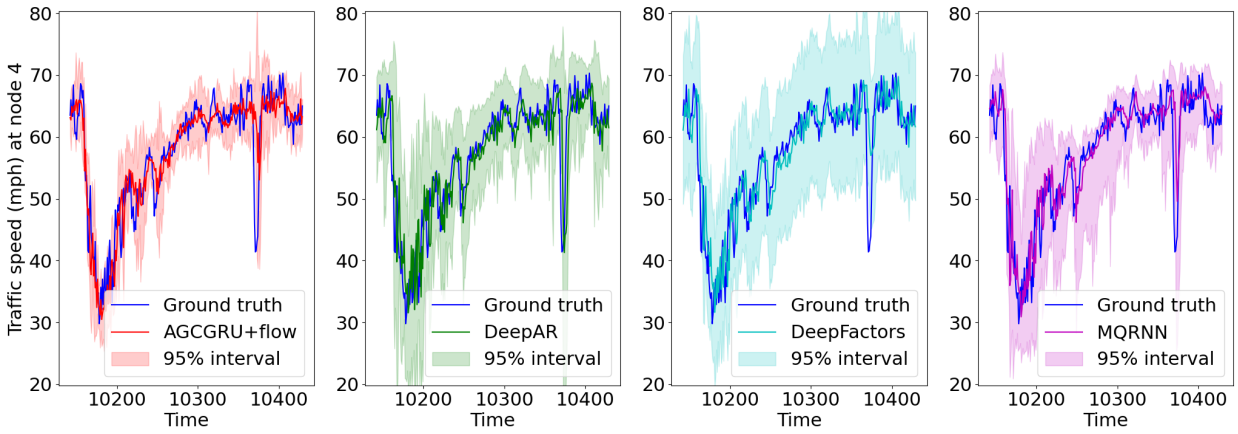

Figure 4. 15 minutes ahead predictions from the probabilistic forecasting algorithms with confidence intervals at node 4 of PeMSD7 dataset for the first day in the test set. The proposed AGCGRU+flow algorithm provides tighter confidence interval than its competitors, which leads to lower quantile error.

& Graff, 2017) (hourly time-series of the electricity consumption), Traffic (Dua & Graff, 2017) (hourly occupancy rate, of different car lanes in San Francisco), Taxi (Salinas et al., 2019), and Wikipedia (Salinas et al., 2019) (count of clicks to different web links) datasets. The detailed statistics of these datasets are summarized in the supplementary material.

6.2. Preprocessing

For the PeMS datasets, missing values are filled by the last known value in the same series. The training, validation and test split is set at 70/10/20% chronologically and standard normalization of the data is used as in (Li et al., 2018). We use one hour of historical data ($P = 12$) to predict the traffic for the next hour ($Q = 12$). Graphs associated with the datasets are constructed using the procedure in (Huang et al., 2020).

6.3. Baselines

To demonstrate the effectiveness of our model, we compare to the following forecasting methods. A detailed description of each baseline is provided in the supplementary material.

Spatio-temporal point forecast models: DCRNN (Li et al., 2018), STGCN (Yu et al., 2018), ASTGCN (Guo et al., 2019), GWN (Wu et al., 2019), GMAN (Zheng et al., 2020), AGCRN (Bai et al., 2020), LSGCN (Huang et al., 2020).
Deep-learning based point forecasting methods: DeepGLO (Sen et al., 2019), N-BEATS (Oreshkin et al., 2020), and FC-GAGA (Oreshkin et al., 2021).
Deep-learning based probabilistic forecasting methods: DeepAR (Salinas et al., 2020), DeepFactors (Wang et al., 2019), and MQRNN (Wen et al., 2017).

The detailed comparison of our approach with all of these models is provided in the supplementary material for space constraints. Here, we show results of a subset, focusing on those with the most competitive performance (Figure 3).

6.4. Hyperparameters and training setup

For our model, we use an $L = 2$ layer AGCGRU (Bai et al., 2020) as the state-transition function. The dimension of the learnable node embedding is $d_e = 10$, and the number of RNN units is $d_x = 64$. We treat ρ and σ as fixed hyperparameters and set $\rho = 1$ and $\sigma = 0$ (no process noise). We train for 100 epochs using the Adam optimizer, with a batch size of 64. The initial learning rate is set to 0.01 and we follow a decaying schedule as in (Li et al., 2018). Hyperparameters associated with scheduled sampling (Bengio et al., 2015), gradient clipping, and early stoppng are borrowed from (Li et al., 2018). We set the number of particles $N_p = 1$ during training and $N_p = 10$ for validation and testing. The number of exponentially spaced discrete steps (Li & Coates, 2017) for integrating the flow is $N_\lambda = 29$. For each dataset, we conduct two separate experiments minimizing the training MAE (results are used to report MAE, MAPE, RMSE, and P50QL) and the training negative log posterior probability (results are used to report CRPS, P10QL, and P90QL). We also experiment with alternative state transition functions, including the DCGRU (Li et al., 2018) and GRU (Chung et al., 2014). For these, the hyperparameters are fixed to the same values as presented above.

6.5. Results and Discussion

Comparison with baselines : Results for the point forecasting task are summarized in Table 1. We observe that most of the spatio-temporal models perform better than graph agnostic baselines in most cases. Moreover, the proposed AGCGRU+flow algorithm achieves on par or better performance with the best-performing spatio-temporal models, such as GWN, GMAN and AGCRN. We present a comparison of the average rankings across datasets in Figure 3. Our proposed method achieves the best average ranking and significantly outperforms the baseline methods. Table 3 summarizes the results for probabilistic forecasting. We observe that in most cases, the proposed flow based algorithms outperform the competitors. MQRNN also shows impressive performance in predicting the forecast quantiles, as it is explicitly trained to minimise the quantile losses. In particular, comparison of GRU+flow with the DeepAR model reveals that even without a sophisticated RNN architecture, the particle flow based approach shows better characterization of prediction uncertainty in most cases. Figure 4 provides a qualitative comparison of the uncertainty characterization, showing example confidence intervals for 15-minute ahead prediction for the PeMSD7 dataset. We see that the proposed algorithm provides considerably tighter intervals, while still achieving coverage of the observed values.

Generalization of particle flow inference across architectures : Table 2 shows that in comparison to deterministic encoder-decoder based sequence to sequence prediction models, the proposed flow based approaches perform better in almost all cases for three different architectures of the RNN. In each case, both of the encoder-decoder model and our approach use a 2-layer architecture with 64 RNN units.

Comparison to the particle filter : Table 4 demonstrates the effectiveness of particle flow (Daum & Huang, 2007), comparing it to a Bootstrap Particle Filter (BPF) (Gordon et al., 1993) with the same number of particles. The use of the bootstrap particle filter leads to a computationally faster algorithm (requiring approximately 60% of the training time of the particle flow-based method).

Comparison to ensembles : We compare the proposed approach with an ensemble of competitive deterministic forecasting techniques. We choose the size of the ensemble so that the algorithms have an approximately equal execution time. We use AGCRN and GMAN to form the ensembles, as they are the best point-forecast baseline algorithms. From Table 5, we observe that the proposed AGCGRU+flow achieves lower average CRPS compared to the ensembles in all cases.

Point forecasting results on non-graph datasets : We evaluate our proposed flow-based RNN on the Electricity and Traffic datasets, following the setting described in Appendix C.4 in (Oreshkin et al., 2020). We augment the results table in (Oreshkin et al., 2020) with the results from an FC-GRU (a fully connected GRU encoder-decoder) and GRU+flow. We use a 2 layer GRU with 64 RNN units in both cases. We follow the preprocessing steps in (Oreshkin et al., 2020). In the literature, four different data splits have

Some of the recent spatio-temporal models such as (Chen et al., 2020; Zhang et al., 2020; Park et al., 2020) do not have publicly available code. Although the codes for (Wu et al., 2020; Song et al., 2020; Pan et al., 2019) are available, these works use different datasets for evaluation. We could not obtain sensible results from these models for our datasets, even with considerable hyperparameter tuning. The code for (Kurle et al., 2020; de Bézenac et al., 2020) is not publicly available.

Table 3. Average CRPS, P10QL, P50QL, and P90QL for PeMSD3, PeMSD4, PeMSD7, and PeMSD8 for 15/30/45/60 minutes horizons. The best and the second best results in each column are shown in bold and marked with underline respectively. Lower numbers are better.

Dataset	PeMSD3	PeMSD4	PeMSD7	PeMSD8
Algorithm	CRPS (15/ 30/ 45/ 60 min)			
DeepAR	11.41/13.11/14.62/16.27	1.13/1.52/1.82/2.07	1.92/2.78/3.44/3.99	0.94/1.24/1.46/1.64
DeepFactors	14.16/15.87/17.59/18.99	1.52/1.84/2.07/2.26	2.35/3.00/3.48/3.87	1.26/1.51/1.69/1.83
GRU+flow	11.23/12.70/13.98/15.25	1.14/1.50/1.75/1.95	1.88/2.61/3.09/3.46	0.95/1.23/1.42/1.57
DCGRU+flow	11.21/12.14/12.87/13.64	1.13/1.43/1.63/1.79	1.85/2.51/2.95/3.27	0.94/1.18/1.35/1.47
AGCGRU+flow	**10.53/11.39/12.03/12.47**	**1.08/1.32/1.46/1.56**	**1.73/2.18/2.43/2.58**	**0.90/1.10/1.20/1.28**
Algorithm	P10QL(%) (15/ 30/ 45/ 60 min)			
DeepAR	4.11/4.69/5.21/5.69	1.37/1.96/2.45/2.86	2.56/3.90/4.92/5.78	1.14/1.59/1.93/2.24
DeepFactors	5.85/6.33/6.91/7.51	2.13/2.61/3.01/3.34	3.49/4.53/5.46/6.26	1.77/2.17/2.49/2.76
MQRNN	4.03/4.60/5.13/5.68	**0.95/1.18/1.31/1.40**	**1.70/2.20/2.47/2.66**	**0.77/0.94/1.04/1.10**
GRU+flow	4.19/4.71/5.14/5.55	1.36/1.87/2.25/2.56	2.50/3.57/4.29/4.85	1.12/1.52/1.80/2.04
DCGRU+flow	4.28/4.69/4.99/5.28	1.33/1.75/2.06/2.30	2.41/3.35/3.97/4.43	1.10/1.43/1.67/1.87
AGCGRU+flow	**4.01/4.44/4.76/4.97**	1.28/1.62/1.82/1.97	2.27/2.97/3.36/3.60	1.10/1.43/1.61/1.73
Algorithm	P50QL(%) (15/ 30/ 45/ 60 min)			
DeepAR	9.11/10.44/11.68/13.03	2.37/3.15/3.73/4.20	4.35/6.21/7.70/8.95	1.97/2.52/2.94/3.30
DeepFactors	10.08/11.60/13.11/14.31	2.42/3.15/3.68/4.10	4.31/5.97/7.16/8.10	1.97/2.55/2.95/3.25
MQRNN	8.40/9.52/10.55/11.58	2.15/2.77/3.19/3.53	3.82/5.21/6.16/6.88	1.77/2.24/2.54/2.77
GRU+flow	8.28/9.26/10.15/11.04	2.16/2.76/3.17/3.50	3.84/5.19/6.10/6.81	**1.76**/2.21/2.49/2.72
DCGRU+flow	8.33/9.01/9.50/9.99	2.16/2.69/3.01/3.26	3.77/4.94/5.66/6.20	1.83/2.25/2.49/2.66
AGCGRU+flow	**7.93/8.54/8.96/9.24**	**2.11/2.55/2.79/2.94**	**3.70/4.65/5.14/5.49**	1.78/**2.15/2.34/2.46**
Algorithm	P90QL(%) (15/ 30/ 45/ 60 min)			
DeepAR	4.40/5.13/5.70/6.40	1.10/1.45/1.67/1.84	2.13/3.03/3.65/4.08	0.93/1.22/1.40/1.53
DeepFactors	6.19/6.95/7.61/8.04	1.98/2.24/2.39/2.50	3.22/3.70/3.97/4.14	1.62/1.82/1.93/1.99
MQRNN	**3.75/4.27**/4.70/5.09	1.22/1.68/2.03/2.32	2.19/3.12/3.78/4.30	0.99/1.34/1.59/1.80
GRU+flow	4.33/4.94/5.48/6.04	1.11/1.43/1.63/1.77	2.02/2.74/3.16/3.44	0.93/1.18/1.33/1.44
DCGRU+flow	4.30/4.67/4.97/5.31	1.10/1.34/1.50/1.61	2.00/2.62/3.01/3.28	0.93/1.13/1.25/1.34
AGCGRU+flow	4.06/4.38/**4.63/4.82**	**1.05/1.26/1.37/1.45**	**1.83/2.25/2.48/2.62**	**0.87/1.01/1.09/1.14**

Table 4. Average MAE and average CRPS for PeMSD3, PeMSD4, PeMSD7, and PeMSD8 for 15/30/45/60 minutes horizons for AGCGRU+flow and AGCGRU+BPF. Lower numbers are better.

Dataset	PeMSD3	PeMSD4	PeMSD7	PeMSD8
Algorithm	MAE (15/ 30/ 45/ 60 min)			
AGCGRU+flow	**13.79/14.84/15.58/16.06**	**1.35/1.63/1.78/1.88**	**2.15/2.70/2.99**/3.19	**1.13/1.37/1.49/1.57**
AGCGRU+BPF	14.19/15.13/15.85/16.35	1.36/1.65/1.80/1.90	2.19/2.73/**2.99/3.17**	1.18/1.41/1.52/1.59
Algorithm	CRPS (15/ 30/ 45/ 60 min)			
AGCGRU+flow	**10.53/11.39/12.03/12.47**	**1.08/1.32**/1.46/1.56	**1.73/2.18/2.43/2.58**	**0.90/1.10/1.20/1.28**
AGCGRU+BPF	11.32/11.94/12.55/12.92	1.10/**1.32/1.45/1.54**	1.79/2.24/2.49/2.66	0.96/1.13/1.22/**1.28**

been used for the Electricity dataset, and three different splits have been used for the Traffic dataset. The evaluation metric is P50QL (Normalized Deviation).

In Table 6, we observe that the flow based approach performs comparably or better than the state-of-the-art N-BEATS algorithm for the Electricity dataset, even with a simple GRU as the state transition function. The better performance of the univariate N-BEATS compared to the multivariate models suggests that most time-series in these datasets do not provide valuable additional information for predicting other datasets. This is in contrast to the graph-based datasets, where the performance of N-BEATS was considerably worse than the multivariate algorithms. The proposed flow-based algorithm achieves prediction performance on the Traffic dataset that is comparable to N-BEATS except for one split with limited training data. Across all datasets and split settings, our flow-based approach significantly outperforms the FC-GRU. The proposed algorithm outperforms TRMF, DeepAR, DeepState and DeepGLO. It outperforms DeepFactors for the Electricity dataset, but is worse for the Traffic dataset (for the same split with limited available training data).

Probabilistic forecasting results on non-graph datasets : For comparison with state-of-the-art deep learning based probabilistic forecasting methods on standard non-graph time-series datasets, we evaluate the proposed GRU+flow algorithm following the setting in (Rasul et al., 2021). The results reported in Table 1 of (Rasul et al., 2021) are augmented with the results of the GRU+flow algorithm. We use a 2 layer GRU with 64 RNN units in each case. We follow the preprocessing steps as in (Salinas et al., 2019; Rasul et al., 2021). The evaluation metric is (normalized) CRPS_{sum} (defined in the supplementary material), which is obtained by first summing across the different time-series,

Table 5. Average CRPS for PeMSD3, PeMSD4, PeMSD7, and PeMSD8 for 15/30/45/60 minutes horizons for AGCRN-ensemble, GMAN-ensemble, and AGCGRU+flow. The best result in each column is shown in bold. Lower numbers are better.

Algorithm	CRPS (15/ 30/ 45/ 60 min)			
	PeMSD3	PeMSD4	PeMSD7	PeMSD8
AGCRN-ensemble	12.64/13.44/13.96/14.27	1.20/1.44/1.56/1.68	1.90/2.39/2.60/2.81	1.03/1.20/1.28/1.38
GMAN-ensemble	12.79/13.49/14.13/14.77	1.16/1.38/1.51/1.62	1.96/2.31/2.53/2.73	0.95/1.10/1.19/1.28
AGCGRU+flow	**10.53/11.39/12.03/12.47**	**1.08/1.32/1.46/1.56**	**1.73/2.18/2.43/2.58**	**0.90/1.10/1.20/1.28**

Table 6. Normalized Deviation on Electricity and Traffic datasets. The best and the second best results in each column are shown in bold and marked with underline respectively. Lower numbers are better.

Algorithm	Electricity				Traffic		
	2014-09-01	2014-03-31	2014-12-18	last 7 days	2008-06-15	2008-01-14	last 7 days
TRMF	0.160	n/a	0.104	0.255	0.200	n/a	0.187
DeepAR	0.070	0.272	0.086	n/a	0.170	0.296	0.140
DeepState	0.083	n/a	n/a	n/a	0.167	n/a	n/a
DeepFactors	n/a	0.112	n/a	n/a	n/a	**0.225**	n/a
DeepGLO	n/a	n/a	<u>0.082</u>	n/a	n/a	n/a	0.148
N-BEATS	**0.064**	**0.065**	n/a	<u>0.171</u>	**0.114**	<u>0.230</u>	**0.110**
FC-GRU	0.102	0.118	0.098	0.203	0.259	0.528	0.233
GRU+flow	<u>0.070</u>	<u>0.071</u>	**0.069**	**0.140**	<u>0.133</u>	0.322	<u>0.125</u>

Table 7. Average CRPS$_{sum}$ for Electricity, Traffic, Taxi, and Wikipedia datasets. The best and the second best results in each column are shown in bold and marked with underline respectively. Lower numbers are better

Dataset	Vec-LSTM ind-scaling	Vec-LSTM lowrank-Copula	GP scaling	GP Copula	LSTM Real-NVP	LSTM MAF	Transformer MAE	GRU+ flow
Electricity	0.025	0.064	0.022	0.024	0.024	<u>0.021</u>	<u>0.021</u>	**0.013**
Traffic	0.087	0.103	0.079	0.078	0.078	0.069	<u>0.056</u>	**0.028**
Taxi	0.506	0.326	0.183	0.208	0.175	<u>0.161</u>	0.179	**0.140**
Wikipedia	0.133	0.241	1.483	0.086	0.078	0.067	<u>0.063</u>	**0.054**

both for the ground-truth test data, and samples of forecasts, and then computing the (normalized) CRPS on the summed data. The results are summarized in Table 7. We observe that the proposed GRU+flow achieves the lowest CRPS$_{sum}$ for all datasets.

Computational complexity : For simplicity, we consider a GRU instead of a graph convolution based RNN and we only focus on one sequence instead of a batch. Our model has to perform both GRU computation and particle flow for the first P time steps and then apply the GRU and the linear projection for the next Q steps to generate the predictions. For an L-layer GRU with d_x RNN units and N-dimensional input, the complexity of the GRU operation for N_p particles is $\mathcal{O}((P+Q)N_pLNd_x^2)$ (Chung et al., 2014). The total complexity of the EDH particle flow (Choi et al., 2011) is $\mathcal{O}(PN_\lambda N^3)$ for computing the flow parameters and $\mathcal{O}(PN_pN_\lambda Nd_x^2)$ for applying the particle flow (more details in the supplementary material). The total complexity of the measurement model for N_p particles is $\mathcal{O}(QN_pNd_x^2)$. Since in most cases $N \gg d_x$ and $N \gg N_p$, the complexity of our algorithm for forecasting of one sequence is $\mathcal{O}(PN_\lambda N^3)$. Many of the other algorithms exhibit a similar $\mathcal{O}(N^3)$ complexity, e.g. TRMF, GMAN. We specify the execution time and memory usage in the supplementary material. Scaling the proposed methodology to extremely high dimensional settings is of significant importance and can be addressed in several ways. For spatio-temporal predictions using the graph-based recurrent architectures, this can be done if the graph can be partitioned meaningfully. For non-graph datasets, we can use the cross-correlation among different time-series to group them into several lower-dimensional problems. Alternatively, we can train a univariate model based on all the time-series as in (Rangapuram et al., 2018).

7. Conclusion

In this paper, we propose a state-space probabilistic modeling framework for multivariate time-series prediction that can process information provided in the form of a graph that specifies (probable) predictive or causal relationships. We develop a probabilistic forecasting algorithm based on the Bayesian inference of hidden states via particle flow. For spatio-temporal forecasting, we use GNN based architectures to instantiate the framework. Our method demonstrates comparable or better performance in point forecasting and considerably better performance in uncertainty characterization compared to existing techniques.

Acknowledgments

This work was partially supported by the Department of National Defence's Innovation for Defence Excellence and Security (IDEaS) program, Canada. We also acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [260250].

References

Bai, L., Yao, L., Kanhere, S., Wang, X., and Sheng, Q. STG2Seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting. In *Proc. Int. Joint Conf. Artificial Intell.*, pp. 1981–1987, 2019.

Bai, L., Yao, L., Li, C., Wang, X., and Wang, C. Adaptive graph convolutional recurrent network for traffic forecasting. In *Proc. Adv. Neural Info. Process. Systems*, 2020.

Bao, W., Yue, J., and Rao, Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS One*, 12(7), 2017.

Bayer, J. and Osendorfer, C. Learning stochastic recurrent networks. *arXiv e-prints: arXiv:1411.7610*, 2014.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proc. Adv. Neural Info. Process. Systems*, volume 1, pp. 1171–1179, 2015.

Bengtsson, T., Bickel, P., and Li, B. Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. In *Probability and Statistics: Essays in Honor of David A. Freedman*, volume 2, pp. 316–334. Institute of Mathematical Statistics, Beachwood, OH, USA, Apr. 2008.

Beskos, A., Crisan, D., and Jasra, A. On the stability of sequential Monte Carlo methods in high dimensions. *Ann. Appl. Prob.*, 24(4):1396–1445, 2014.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

Chang, Y.-Y., Sun, F.-Y., Wu, Y.-H., and Lin, S.-D. A memory-network based solution for multivariate time-series forecasting. *arXiv preprint arXiv:1809.02105*, 2018.

Chen, C., Petty, K., and Skabardonis, A. Freeway performance measurement system: Mining loop detector data. *Transport. Research Record*, 1748, Jan. 2000.

Chen, C., Li, K., Teo, S. G., Zou, X., Wang, K., Wang, J., and Zeng, Z. Gated residual recurrent graph neural networks for traffic prediction. In *Proc. AAAI Conf. Artificial Intell.*, pp. 485–492, 2019a.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. In *Proc. Adv. Neural Info. Process. Systems*, 2018.

Chen, W., Chen, L., Xie, Y., Cao, W., Gao, Y., and Feng, X. Multi-range attentive bicomponent graph convolutional network for traffic forecasting. In *Proc. AAAI Conf. Artificial Intell.*, pp. 3529–3536, 2020.

Chen, X., Dai, H., and Song, L. Particle flow Bayes' rule. In *Proc. Int. Conf. Machine Learning*, pp. 1022–1031, 2019b.

Choi, S., Willett, P., Daum, F., and Huang, J. Discussion and application of the homotopy filter. In *Proc. SPIE Conf. Signal Process., Sensor Fusion, Target Recog.*, pp. 805021, Orlando, FL, USA, May 2011.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proc. Workshop on Deep Learning, Adv. Neural Info. Process. Systems*, 2014.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. A recurrent latent variable model for sequential data. In *Proc. Adv. Neural Info. Process. Systems*, pp. 2980–2988, 2015.

Daum, F. and Huang, J. Nonlinear filters with log-homotopy. In *Proc. SPIE Signal and Data Process. Small Targets*, pp. 669918, San Diego, CA, USA, Sep. 2007.

Daum, F., Huang, J., and Noushin, A. Exact particle flow for nonlinear filters. In *Proc. SPIE Conf. Signal Process., Sensor Fusion, Target Recog.*, pp. 769704, Orlando, FL, USA, Apr. 2010.

de Bézenac, E., Rangapuram, S. S., Benidis, K., Bohlke-Schneider, M., Kurle, R., Stella, L., Hasson, H., Gallinari, P., and Januschowski, T. Normalizing kalman filters for multivariate time series analysis. In *Proc. Adv. Neural Info. Process. Systems*, volume 33, 2020.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proc. Adv. Neural Inf. Proc. Systems*, pp. 3844–3852, 2016.

Ding, T. and Coates, M. J. Implementation of the daum-huang exact-flow particle filter. In *Proc. IEEE Statist. Signal Process. Workshop (SSP)*, pp. 257–260, Ann Arbor, MI, USA, Aug. 2012.

Doerr, A., Daniel, C., Schiegg, M., Nguyen-Tuong, D., Schaal, S., Toussaint, M., and Trimpe, S. Probabilistic recurrent state-space models. *arXiv preprint arXiv:1801.10395*, 2018.

Doucet, A. and Johansen, A. M. A tutorial on particle filtering and smoothing: Fifteen years later. In *The Oxford Handbook of Nonlinear Filtering*, chapter 24, pp. 656–704. Oxford University Press, Oxford, UK, 2009.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. Sequential neural models with stochastic layers. *arXiv eprints, arXiv:1605.07571*, 2016.

Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Proc. Adv. Neural Info. Process. Systems*, 2017.

Gammelli, D. and Rodrigues, F. Recurrent flow networks: A recurrent latent variable model for spatio-temporal density modelling. *arXiv eprints, arXiv:2006.05256*, 2020.

Gneiting, T. and Raftery, A. E. Strictly proper scoring rules, prediction, and estimation. *J. Amer. Statist. Association*, 102(477):359–378, 2007.

Gordon, N., Salmond, D., and Smith, A. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. F Radar and Signal Process.*, 140:107–113, Apr. 1993.

Gu, S. S., Ghahramani, Z., and Turner, R. E. Neural adaptive sequential monte carlo. In *Proc. Adv. Neural Info. Process. Systems*, volume 28, pp. 2629–2637, 2015.

Guo, S., Lin, Y., Feng, N., Song, C., and Wan, H. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proc. AAAI Conf. Artificial Intell.*, 2019.

Guo, T. and Lin, T. Multi-variable LSTM neural network for autoregressive exogenous model. *arXiv preprint arXiv:1806.06384*, 2018.

Huang, R., Huang, C., Liu, Y., Dai, G., and Kong, W. LSGCN: Long short-term traffic prediction with graph convolutional networks. In *Proc. Int. Joint Conf. Artificial Intell.*, pp. 2355–2361, 7 2020.

Huang, Y., Weng, Y., Yu, S., and Chen, X. Diffusion convolutional recurrent neural network with rank influence learning for traffic forecasting. In *Proc. IEEE Int. Conf. Big Data Science And Engineering*, pp. 678–685, Aug 2019.

Karkus, P., Hsu, D., and Lee, W. S. Particle filter networks with application to visual localization. In *Proc. Conf. Robot Learning*, pp. 169–178, 2018.

Karl, M., Soelch, M., Bayer, J., and Van der Smagt, P. Deep variational Bayes filters: Unsupervised learning of state space models from raw data. In *Proc. Int. Conf. Learning Rep.*, 2016.

Kipf, T. and Welling, M. Semi-supervised classification with graph convolutional networks. In *Proc. Int. Conf. Learning Rep.*, 2017.

Kobyzev, I., Prince, S., and Brubaker, M. Normalizing flows: An introduction and review of current methods. *IEEE Trans. Pattern Analysis and Machine Intell.*, 2020.

Kumar, M., Babaeizadeh, M., Erhan, D., Finn, C., Levine, S., Dinh, L., and Kingma, D. Videoflow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434*, 2(5), 2019.

Kurle, R., Rangapuram, S. S., de Bézenac, E., Günnemann, S., and Gasthaus, J. Deep Rao-Blackwellised particle filters for time series forecasting. In *Proc. Adv. Neural Info. Process. Systems*, 2020.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long-and short-term temporal patterns with deep neural networks. In *Proc. ACM Int. Conf. Research & Development in Info. Retrieval*, pp. 95–104, 2018.

Le, T. A., Igl, M., Rainforth, T., Jin, T., and Wood, F. Auto-encoding sequential Monte Carlo. *arXiv preprint arXiv:1705.10306*, 2017.

Li, L., Yan, J., Yang, X., and Jin, Y. Learning interpretable deep state space model for probabilistic time series forecasting. In *Proc. Int. Joint Conf. on Artificial Intell.*, pp. 2901–2908, 2019.

Li, Y. and Coates, M. Particle filtering with invertible particle flow. *IEEE Trans. Signal Process.*, 65(15):4102–4116, Aug. 2017.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *Proc. Int. Conf. Learning Rep.*, 2018.

Ma, X., Karkus, P., Hsu, D., and Lee, W. S. Particle filter recurrent neural networks. In *Proc. AAAI Conf. Artificial Intell.*, pp. 5101–5108, 2020.

Maddison, C. J., Lawson, D., Tucker, G., Heess, N., Norouzi, M., Mnih, A., Doucet, A., and Teh, Y. W. Filtering variational objectives. In *Proc. Adv. Neural Info. Process. Systems*, 2017.

Mattos, C. L. C., Dai, Z., Damianou, A., Forth, J., Barreto, G. A., and Lawrence, N. D. Recurrent Gaussian processes. In *Proc. Int. Conf. Learning Rep.*, 2016.

Naesseth, C., Linderman, S., Ranganath, R., and Blei, D. Variational sequential Monte Carlo. In *Proc. Int. Conf. Artificial Intell. and Statist.*, pp. 968–977, 2018.

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *Proc. Int. Conf. Learning Rep.*, 2020.

Oreshkin, B. N., Amini, A., Coyle, L., and Coates, M. J. FC-GAGA: Fully Connected Gated Graph Architecture for spatio-temporal traffic forecasting. In *Proc. AAAI Conf. Artificial Intell.*, Jan. 2021.

Pan, Z., Liang, Y., Wang, W., Yu, Y., Zheng, Y., and Zhang, J. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. & Data Mining*, pp. 1720–1730, 2019.

Park, C., Lee, C., Bahng, H., won, T., Kim, K., Jin, S., Ko, S., and Choo, J. ST-GRAT: A spatio-temporal graph attention network for traffic forecasting. In *Proc. AAAI Conf. Artificial Intell.*, 2020.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., and Cottrell, G. W. A dual-stage attention-based recurrent neural network for time series prediction. In *Proc. Int. Joint Conf. Artificial Intell.*, 2017.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. Deep state space models for time series forecasting. In *Proc. Adv. Neural Info. Process. Systems*, 2018.

Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U., and Vollgraf, R. Multi-variate probabilistic time series forecasting via conditioned normalizing flows. In *Proc. Int. Conf. Learning Rep.*, 2021.

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., and Gasthaus, J. High-dimensional multivariate forecasting with low-rank Gaussian copula processes. In *Proc. Adv. Neural Info. Process. Systems*, 2019.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecasting*, 36(3):1181 – 1191, 2020.

Sen, R., Yu, H.-F., and Dhillon, I. S. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *Proc. Adv. Neural Info. Process. Systems*, volume 32, pp. 4837–4846, 2019.

Shi, X., Qi, H., Shen, Y., Wu, G., and Yin, B. A spatial-temporal attention approach for traffic prediction. *IEEE Trans. Intell. Transport. Systems*, pp. 1–10, Apr. 2020.

Smyl, S. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *Int. J. Forecasting*, 36(1):75 – 85, 2020.

Snyder, C., Bengtsson, T., Bickel, P., and Anderson, J. Obstacles to high-dimensional particle filtering. *Mon. Weather Rev.*, 136(12):4629–4640, Dec. 2008.

Song, C., Lin, Y., Guo, S., and Wan, H. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proc. AAAI Conf. Artificial Intell.*, pp. 914–921, Apr. 2020.

Wang, Y., Smola, A., Maddix, D., Gasthaus, J., Foster, D., and Januschowski, T. Deep factors for forecasting. In *Proc. Int. Conf. Machine Learning*, Long Beach, California, USA, Jun 2019.

Wen, R., Torkkola, K., Narayanaswamy, B., and Madeka, D. A Multi-Horizon Quantile Recurrent Forecaster. *arXiv e-prints, arXiv:1711.11053*, 2017.

Wu, Z., Pan, S., Long, G., Jiang, J., and Zhang, C. Graph wavenet for deep spatial-temporal graph modeling. In *Proc. Int. Joint Conf. Artificial Intell.*, pp. 1907–1913, 2019.

Wu, Z., Pan, S., Long, G., Jiang, J., and Zhang, C. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *Proc. Int. Joint Conf. Artificial Intell.*, 2019.

Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., and Zhang, C. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proc. ACM SIGKDD Conf. Knowl. Discov. & Data Mining*, pp. 753–763, 08 2020.

Xu, M., Dai, W., Liu, C., Gao, X., Lin, W., Qi, G.-J., and Xiong, H. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908*, 2020.

Yu, B., Yin, H., and Zhu, Z. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proc. Int. Joint Conf. Artificial Intell.*, 2018.

Yu, B., Li, M., Zhang, J., and Zhu, Z. 3d graph convolutional networks with temporal graphs: A spatial information free framework for traffic forecasting. *arXiv preprint arXiv:1903.00919*, 2019.

Zhang, Q., Chang, J., Meng, G., Xiang, S., and Pan, C. Spatio-temporal graph structure learning for traffic forecasting. In *Proc. AAAI Conf. Artificial Intell.*, pp. 1177–1185, Apr. 2020.

Zhao, R., Wang, K., Su, H., and Ji, Q. Bayesian graph convolution lstm for skeleton based action recognition. In *Pro. IEEE/CVF Int. Conf. Computer Vision*, Oct. 2019.

Zheng, C., Fan, X., Wang, C., and Qi, J. GMAN: A Graph Multi-Attention Network for Traffic Prediction. In *Proc. AAAI Conf. Artificial Intell.*, 2020.

Zheng, X., Zaheer, M., Ahmed, A., Wang, Y., Xing, E. P., and Smola, A. J. State space LSTM models with particle MCMC inference. *arXiv e-prints: arxiv 1711.11179*, 2017.