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
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; Chung 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 \( G = (V, E) \), where \( V \) is the set of \( N \) nodes and \( E \subseteq V \times V \) 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 \( G \), 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-
RNN with Particle Flow for Probabilistic Spatio-temporal Forecasting

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.

The joint posterior distribution of the forecasts: $P_n(y_{F|P}|y_{1:F}) = \int \prod_{t=1}^{P} P_n(y_t|y_1, z, \lambda) P(z|y_1, \lambda) dz d\lambda$.

5. Methodology

5.1. State-space model

We postulate that $y_t \in \mathbb{R}^{N \times 1}$ is the observation from a Markovian state space model with hidden state $X_t \in \mathbb{R}^{N \times d_x}$. We denote by $x_t$ and $z_t$ the vectorizations of $X_t$ and $Z_t$, respectively. The state space model is:

$$x_t \sim p_1(\cdot | z_{t-1}, \rho),$$

$$x_t = g_{\Psi}(x_{t-1}, y_{t-1}, z_t, v_t), \text{ for } t > 1,$$

$$y_t = h_{\theta}(x_t, z_t, w_t), \text{ for } t \geq 1.$$  \hspace{1cm} (6)

Here $v_t \sim p_v(\cdot | x_{t-1}, \sigma)$ and $w_t \sim p_w(\cdot | x_t, \gamma)$ are the noises in the dynamic and measurement models respectively, $\rho, \sigma$ and $\gamma$ are the parameters of the distribution of the initial state $x_1$, process noise $v_t$ and measurement noise $w_t$ respectively. $g$ and $h$ denote the state transition and measurement functions, possibly linear or nonlinear, with parameters $\psi$ and $\phi$ respectively. The subscript $\Psi$ in $g$ and $h$ indicates that the functions are potentially dependent on the graph topology. We assume that $h_{\theta}(x_t, z_t, 0)$ is a $C^1$ function in $x_t$, i.e., $h_{\theta}(x_t, z_t, 0)$ is a differentiable function whose first derivative w.r.t. $x_t$ 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

The joint posterior distribution of the forecasts: $P_n(y_{F|P}|y_{1:F}) = \int \prod_{t=1}^{P} P_n(y_t|y_1, z, \lambda) P(z|y_1, \lambda) dz d\lambda$.

The state transition and measurement equations are:

$$x_t \sim p_1(\cdot | z_{t-1}, \rho),$$

$$x_t = g_{\Psi}(x_{t-1}, y_{t-1}, z_t, v_t), \text{ for } t > 1,$$

$$y_t = h_{\theta}(x_t, z_t, w_t), \text{ for } t \geq 1.$$  \hspace{1cm} (6)

Here $v_t \sim p_v(\cdot | x_{t-1}, \sigma)$ and $w_t \sim p_w(\cdot | x_t, \gamma)$ are the noises in the dynamic and measurement models respectively, $\rho, \sigma$ and $\gamma$ are the parameters of the distribution of the initial state $x_1$, process noise $v_t$ and measurement noise $w_t$ respectively. $g$ and $h$ denote the state transition and measurement functions, possibly linear or nonlinear, with parameters $\psi$ and $\phi$ respectively. The subscript $\Psi$ in $g$ and $h$ indicates that the functions are potentially dependent on the graph topology. We assume that $h_{\theta}(x_t, z_t, 0)$ is a $C^1$ function in $x_t$, i.e., $h_{\theta}(x_t, z_t, 0)$ is a differentiable function whose first derivative w.r.t. $x_t$ 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
variables \((y_t, z_t)\) to the latent variables \((v_t, w_t)\) and the graph \((G)\).

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 \(g\):

\[
x_t = AGCGRU^{(L)}_{G,v}\left(x_{t-1}, y_{t-1}, z_t\right) + v_t, \quad (7)
\]

\[
y_t = W_y x_t + w_t. \quad (8)
\]

In this model, we have \(p(v_t) = \mathcal{N}(0, \sigma^2 I)\), i.e., the latent variables for the dynamics are independent. The initial state distribution is also chosen to be isotropic Gaussian, i.e., \(p_1(x_1, z_1, \rho) = \mathcal{N}(0, \rho^2 I)\). The parameters \(\rho\) and \(\sigma\) are learnable variance parameters. The observation model \(q\) incorporates a linear projection matrix \(W_y\). The latent variable \(w_t\) for the emission model is modelled as Gaussian with variance dependent on \(x_t\) via a learnable softplus function:

\[
p_w(w_t| x_t) = \mathcal{N}\left(0, \text{diag}\left(\text{softmax}(C_x x_t)\right)^2\right). \quad (9)
\]

5.2. Inference

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

Inference involves an iterative process. We randomly initialize the parameters of the model to obtain \(\Theta_0\). Subsequently, at the \(k\)-th iteration of the algorithm (processing the \(k\)-th training batch), we first draw samples \(\{y_{t_0+1:t_0+P+1}^{i}\}^{i=1}_{i=N_p}\) from the distribution \(p_{\Theta_{k-1}}(y_{t_0+1:t_0+P+1}^{i}|y_{1:t_0}^{i}, z_{1:t_0+P+1}^{i})\). With this set of samples, we can subsequently apply a gradient descent procedure to obtain the updated model parameters \(\Theta_k\).

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., \(p_{\Theta}(y_{t_0+1:t_0+P+1}|y_{1:t_0}^{i}, z_{1:t_0+P+1}^{i})\). For conciseness we drop the time-offset \(t_0\).

\[
p_{\Theta}(y_{t_0+1:t_0+P+1}|y_{1:t_0}^{i}, z_{1:t_0+P+1}^{i}) = \int \prod_{t=t_0+1}^{t_0+P+1} p_{\Theta}(\hat{y}_t|y_{1:t_0}^{i}, z_{1:t_0+P+1}^{i}) dy_{t_0+1:t_0+P+1}.
\]

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 \(1 \leq t \leq P\), 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:

\[
p_{\Theta}(x_t|y_{1:t}, z_{1:t}) \approx \frac{1}{N_p} \sum_{j=1}^{N_p} \delta(x_t - x_t^{j}). \quad (11)
\]

Here \(\{x_t^{j}\}^{j=1}_{j=N_p}\) are approximately distributed according to the posterior distribution of \(x_t\). The generation of each sample \(x_t^{j}\) involves an associated sampling of the latent variables \(v_{1:t}\) and implies a sampling of \(w_{1:t}\), but these samples are not required since the proposed model only needs \(x_{1:t}\) to construct the forecast.

**Step 2:** For \(P+1 \leq t \leq P+Q\), we iterate between the following two steps:

a. We sample \(x_t^{j}\) from \(p_{\psi,\sigma}(x_t^{j}|x_{t-1}^{j}, y_{t-1}^{j}, z_{t}^{j})\) (for \(t > P+1\)) or from \(p_{\psi,\sigma}(x_t^{j}|x_{t-1}^{j}, y_{t-1}^{j}, z_{t}^{j})\) (for \(t = P+1\)) for \(1 \leq j \leq N_p\). This amounts to a state transition at time \(t\) to obtain the current state \(x_t^{j}\) from the previous state \(x_{t-1}^{j}\), using eq. (5).

b. We sample \(y_t^{j}\) from \(p_{\Theta,\gamma}(y_t^{j}|x_t^{j}, z_t^{j})\) for \(1 \leq j \leq N_p\), i.e., we use \(x_t^{j}\) in the measurement model, specified by eq. (6), to sample \(y_t^{j}\).

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

\[
p_{\Theta}(y_{t_0+1:t_0+P+1}|y_{1:t_0}^{i}, z_{1:t_0+P+1}^{i}) \approx \prod_{t=P+1}^{P+Q} \frac{1}{N_p} \sum_{j=1}^{N_p} \delta(y_t^{j} - y_t^{j}). \quad (12)
\]
Each $y_j^{P+Q}$ is approximately distributed according to the joint posterior distribution of $y_{P+1:P+Q}$. 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

1. **Input:** $y_{1:P}, z_{1:P}$, and $\Theta$
2. **Output:** $\{y_{P+1:P+Q}\}_{j=1}^{N_p}$
3. **Initialization:** Sample $\eta_0^j \sim p_1(x_1, z_1, \rho)$, $j = 1 : N_p$
4. for $t = 1, 2, ..., P$
5. if $t > 1$
6. Sample $\eta_0^j \sim p_{t-1}(x_{t-1} | x_{t-2}, y_{t-1}, z_t)$, $j = 1 : N_p$ as: $\eta_0^j = g_{t-1}(x_{t-1}^j, y_{t-1}^j, z_t)$.
7. end if
8. Use particle flow (details in supplementary material) to obtain $\{\eta_t^j\}_{j=1}^{N_p}$ from $\{\eta_0^j\}_{j=1}^{N_p}$, and $y_t$
9. Set $x_t^j = \eta_t^j$
10. end for
11. for $t = P + 1, P + 2, ..., P + Q$
12. if $t = P + 1$
13. Sample $x_{P+1}^j \sim p_{t+1}(x_{P+1}^j | x_{P+1}^j, y_{P+1}, z_{P+1})$, $j = 1 : N_p$ as: $x_{P+1}^j = g_{P+1}(x_{P+1}^j, y_{P+1}^j, z_{P+1})$.
14. else
15. Sample $x_t^j \sim p_{t-1}(x_{t-1}^j | x_{t-2}^j, y_{t-1}^j, z_t)$, $j = 1 : N_p$ as: $x_t^j = g_{t-1}(x_{t-1}^j, y_{t-1}^j, z_t)$.
16. end if
17. Sample $y_t^j \sim p_{t}(y_t^j | x_t^j, z_t)$, $j = 1 : N_p$ as: $y_t^j = h_{t}(x_t^j, z_t)$.
18. end for
19. Form the Monte Carlo estimate using eq. (12).

### 5.2.2. Parameter Update

With the predictive samples $\{y_{P+1:P+Q}^{j}\}_{j=1}^{N_p}$, 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 $\hat{y}_{P+1:P+Q}^{(m)}$ is obtained based on a statistic such as the mean or median of the samples $\{y_{P+1:P+Q}^{j}\}_{j=1}^{N_p}$. The MAE loss function on a dataset indexed by $D$ can then be expressed as:

$$L_{\text{MAE}}(\Theta, D) = \frac{1}{|Q|} \sum_{m \in D} \sum_{t=P+1}^{P+Q} \|y_t^{(m)} - \hat{y}_t^{(m)}\|_1.$$  

(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:

$$L_{\text{prob}}(\Theta, D) = \frac{1}{|D|} \sum_{m \in D} \log p_\theta(y_{P+1:P+Q}^{(m)} | y_{1:P}, z_{1:P+Q}^{(m)})$$

where we approximate the posterior probability as:  

$$p_\theta(y_{P+1:P+Q} | y_{1:P}, z_{1:P+Q}) = \prod_{t=P+1}^{P+Q} \left[ 1 \sum_{j=1}^{N_p} p_{\theta, \gamma}(y_t | x_t^j, z_t) \right].$$

(14)

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 $p_\theta(y_{P+1:P+Q} | y_{1:P}, z_{1:P+Q})$. By Jensen’s inequality, the summation over the log terms in (14) is thus a lower bound for the desired $\mathbb{E}[\log p_\theta(y_{P+1:P+Q} | y_{1:P}, z_{1:P+Q})]$. This lower bound improves 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.

5.1. 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

In Table 1, we summarize the traffic datasets used in our experiments. Each dataset is defined by its number of road segments, total number of records, road type, and the number of sensors. We also report the number of road segments that are considered for training and testing.

### 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

In Table 1, we summarize the traffic datasets used in our experiments. Each dataset is defined by its number of road segments, total number of records, road type, and the number of sensors. We also report the number of road segments that are considered for training and testing.

6.2. Parameter Selection

We choose the hyperparameters for the model based on grid search over a limited set of values. The hyperparameters include the number of particles, the number of layers in the RNN, and the learning rate of the optimizer.

6.3. Evaluation Metrics

We evaluate the performance of our model using several standard metrics, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Squared Logarithmic Error (MSLE). We also report the Pearson correlation coefficient and the coefficient of determination ($R^2$).

6.4. Results

Table 2 presents the results of our experiments on the traffic datasets. We compare our model against several state-of-the-art methods, including RNN, GNN, and GCRN. The results indicate that our model outperforms the baselines in terms of all the metrics.

Figure 3 shows the boxplot of ranks for 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.
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.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MAE (15/30/45/60 min)</th>
<th>PeMSD3</th>
<th>PeMSD4</th>
<th>PeMSD7</th>
<th>PeMSD8</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCRNN</td>
<td>14.42/15.87/17.10/18.29</td>
<td>1.38/1.78/2.06/2.29</td>
<td>2.23/3.06/3.67/4.18</td>
<td>1.16/1.49/1.70/1.87</td>
<td></td>
</tr>
<tr>
<td>STGCN</td>
<td>15.22/17.54/19.74/21.59</td>
<td>1.42/1.85/2.14/2.39</td>
<td>2.21/2.96/3.47/3.90</td>
<td>1.22/1.56/1.79/1.98</td>
<td></td>
</tr>
<tr>
<td>GWN</td>
<td>14.63/16.56/18.34/20.08</td>
<td>1.37/1.76/2.03/2.24</td>
<td>2.23/3.03/3.56/3.98</td>
<td><strong>1.11/1.40/1.59/1.73</strong></td>
<td></td>
</tr>
<tr>
<td>GMAN</td>
<td>14.73/15.44/16.15/16.96</td>
<td><strong>1.38/1.61/1.76/1.88</strong></td>
<td>2.40/2.76/2.98/3.16</td>
<td><strong>1.23/1.36/1.46/1.55</strong></td>
<td></td>
</tr>
<tr>
<td>AGCRN</td>
<td>14.20/15.34/16.18/17.38</td>
<td>1.41/1.67/1.84/2.01</td>
<td>2.19/2.81/3.15/3.42</td>
<td>1.16/1.39/1.53/1.67</td>
<td></td>
</tr>
<tr>
<td>LSGCN</td>
<td>14.28/16.08/17.79/19.23</td>
<td>1.40/1.78/2.03/2.20</td>
<td>2.23/2.99/3.50/3.95</td>
<td>1.21/1.54/1.75/1.89</td>
<td></td>
</tr>
<tr>
<td>AGCGRU+flow</td>
<td></td>
<td>1.16/1.39/1.53/1.67</td>
<td>2.22/2.85/3.18/3.36</td>
<td>1.18/1.47/1.62/1.72</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MAE (15/30/45/60 min)</th>
<th>PeMSD3</th>
<th>PeMSD4</th>
<th>PeMSD7</th>
<th>PeMSD8</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGCGRU+flow</td>
<td><strong>13.79/14.64/15.58/16.06</strong></td>
<td><strong>1.35/1.63/1.78/1.88</strong></td>
<td><strong>2.15/2.70/2.99/3.19</strong></td>
<td><strong>1.13/1.37/1.49/1.57</strong></td>
<td></td>
</tr>
<tr>
<td>FC-AGCGRU</td>
<td>13.96/15.37/16.52/17.45</td>
<td>1.37/1.74/2.00/2.20</td>
<td>2.21/2.99/3.56/4.05</td>
<td>1.16/1.48/1.70/1.87</td>
<td></td>
</tr>
<tr>
<td>DCGRU+flow</td>
<td>14.48/15.67/16.52/17.36</td>
<td><strong>1.38/1.71/1.92/2.08</strong></td>
<td><strong>2.19/2.87/3.29/3.61</strong></td>
<td><strong>1.17/1.44/1.58/1.70</strong></td>
<td></td>
</tr>
<tr>
<td>FC-DCGRU</td>
<td>14.42/15.87/17.10/18.29</td>
<td>1.38/1.78/2.06/2.29</td>
<td>2.23/3.06/3.67/4.18</td>
<td>1.16/1.49/1.70/1.87</td>
<td></td>
</tr>
<tr>
<td>GRU+flow</td>
<td>14.40/16.10/17.63/19.18</td>
<td>1.37/1.76/2.02/2.23</td>
<td>2.24/3.02/3.55/3.96</td>
<td>1.12/1.41/1.59/1.74</td>
<td></td>
</tr>
<tr>
<td>FC-GRU</td>
<td>15.82/18.37/20.61/22.93</td>
<td>1.46/1.91/2.25/2.54</td>
<td>2.41/3.40/4.17/4.84</td>
<td>1.20/1.56/1.81/2.02</td>
<td></td>
</tr>
</tbody>
</table>

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.
We train for 100 epochs using the Adam optimizer, with AGCRN (Bai et al., 2020), LSGCN (Huang et al., 2020). The detailed comparison of our approach with all of these models is not publicly available. From these models for our datasets, even with considerable hyperparameters are fixed to the same values as presented in Appendix C.4 in (Oreshkin et al., 2020). We set the number of exponentially spaced particles \( N_p = 10 \) 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.

6.4. 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 \( d_r = 64 \). We treat \( \rho \) and \( \sigma \) 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 stopping 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.

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...
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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PeMSD3</th>
<th>PeMSD4</th>
<th>PeMSD7</th>
<th>PeMSD8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>CRPS (15/30/45/60 min)</td>
<td>CRPS (15/30/45/60 min)</td>
<td>CRPS (15/30/45/60 min)</td>
<td>CRPS (15/30/45/60 min)</td>
</tr>
<tr>
<td>DeepAR</td>
<td>11.41/13.11/14.62/16.27</td>
<td>11.31/15.2/18.2/20.07</td>
<td>19.2/2.78/3.44/3.99</td>
<td>0.94/1.24/1.46/1.64</td>
</tr>
<tr>
<td>DeepFactors</td>
<td>14.16/15.87/17.59/18.99</td>
<td>1.52/18.4/20.7/2.26</td>
<td>2.35/3.00/3.48/3.87</td>
<td>1.26/1.51/1.69/1.83</td>
</tr>
<tr>
<td>GRU+flow</td>
<td>11.23/12.70/13.98/15.25</td>
<td>1.14/15.0/1.75/1.95</td>
<td>1.88/2.61/3.09/3.46</td>
<td>0.95/1.23/1.42/1.57</td>
</tr>
<tr>
<td>DCGRU+flow</td>
<td>11.21/12.14/12.87/13.64</td>
<td>1.13/13.4/1.63/1.79</td>
<td>1.85/2.51/2.95/3.27</td>
<td>0.94/1.18/1.35/1.47</td>
</tr>
<tr>
<td>AGCGRU+flow</td>
<td>10.53/11.39/12.03/12.47</td>
<td>1.08/13.2/1.46/1.56</td>
<td>1.73/2.18/2.43/2.58</td>
<td>0.90/1.10/1.20/1.28</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PeMSD3</th>
<th>PeMSD4</th>
<th>PeMSD7</th>
<th>PeMSD8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>MAE (15/30/45/60 min)</td>
<td>MAE (15/30/45/60 min)</td>
<td>MAE (15/30/45/60 min)</td>
<td>MAE (15/30/45/60 min)</td>
</tr>
<tr>
<td>AGCGRU+flow</td>
<td>13.79/14.84/15.58/16.06</td>
<td>1.35/1.63/1.78/1.88</td>
<td>2.15/2.70/2.99/3.19</td>
<td>1.13/1.37/1.49/1.57</td>
</tr>
<tr>
<td>AGCGRU+BPF</td>
<td>14.19/15.13/15.85/16.35</td>
<td>1.36/1.65/1.80/1.90</td>
<td>2.19/2.73/2.99/3.17</td>
<td>1.18/1.41/1.52/1.59</td>
</tr>
<tr>
<td>Algorithm</td>
<td>CRPS (15/30/45/60 min)</td>
<td>CRPS (15/30/45/60 min)</td>
<td>CRPS (15/30/45/60 min)</td>
<td>CRPS (15/30/45/60 min)</td>
</tr>
<tr>
<td>AGCGRU+flow</td>
<td>10.53/11.39/12.03/12.47</td>
<td>1.08/13.2/1.46/1.56</td>
<td>1.73/2.18/2.43/2.58</td>
<td>0.90/1.10/1.20/1.28</td>
</tr>
<tr>
<td>AGCGRU+BPF</td>
<td>11.32/11.94/12.55/12.92</td>
<td>1.10/13.2/1.45/1.54</td>
<td>1.79/2.24/2.49/2.66</td>
<td>0.96/1.13/1.22/1.28</td>
</tr>
</tbody>
</table>

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 (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.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>PeMSD3 (15/30/45/60 min)</th>
<th>PeMSD4 (15/30/45/60 min)</th>
<th>PeMSD7 (15/30/45/60 min)</th>
<th>PeMSD8 (15/30/45/60 min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGCRN-ensemble</td>
<td>12.64/13.44/13.96/14.27</td>
<td>1.20/1.44/1.56/1.68</td>
<td>1.90/2.39/2.60/2.81</td>
<td>1.90/2.01/2.18/2.38</td>
</tr>
<tr>
<td>GMAN-ensemble</td>
<td>12.79/13.49/14.13/14.77</td>
<td>1.16/1.38/1.51/1.62</td>
<td>1.96/2.31/2.53/2.73</td>
<td>0.95/1.01/1.19/1.28</td>
</tr>
<tr>
<td>AGCGRU+flow</td>
<td>10.53/11.39/12.03/12.47</td>
<td>1.08/1.32/1.46/1.56</td>
<td>1.73/2.18/2.43/2.58</td>
<td>0.90/1.10/1.20/1.28</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Electricity</th>
<th>Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRMF</td>
<td>0.160</td>
<td>n/a</td>
</tr>
<tr>
<td>DeepAR</td>
<td>0.070</td>
<td>0.272</td>
</tr>
<tr>
<td>DeepState</td>
<td>0.083</td>
<td>n/a</td>
</tr>
<tr>
<td>DeepFactors</td>
<td>n/a</td>
<td>0.112</td>
</tr>
<tr>
<td>DeepGLO</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>N-BEATS</td>
<td>0.064</td>
<td>0.065</td>
</tr>
<tr>
<td>PC-GRU</td>
<td>0.102</td>
<td>0.118</td>
</tr>
<tr>
<td>GRU+flow</td>
<td>0.070</td>
<td>0.071</td>
</tr>
</tbody>
</table>

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.

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

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 $d_s$ RNN units and $N$-dimensional input, the complexity of the GRU operation for $N_p$ particles is $O((P + Q)N_pLN_d_s^2)$ (Chung et al., 2014). The total complexity of the EDH particle flow (Choi et al., 2011) is $O(PN_d^3)$ for computing the flow parameters and $O(PN_pN_sN_d_s^2)$ for applying the particle flow (more details in the supplementary material). The total complexity of the measurement model for $N_p$ particles is $O(QN_pN_d_s^2)$. Since in most cases $N \gg d_s$ and $N \gg N_p$, the complexity of our algorithm for forecasting one sequence is $O(PN_sN^3)$. Many of the other algorithms exhibit a similar $O(N^3)$ 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


