Recurrent Flow Networks: A Recurrent Latent Variable Model for Spatio-Temporal Density Modelling

Daniele Gammelli  
Technical University of Denmark  
daga@dtu.dk

Filipe Rodrigues  
Technical University of Denmark  
rodr@dtu.dk

Abstract

When modelling real-valued sequences, a typical approach in current RNN architectures is to use a Gaussian mixture model to describe the conditional output distribution. In this paper, we argue that mixture-based distributions could exhibit structural limitations when faced with highly complex data distributions such as for spatial densities. To address this issue, we introduce recurrent flow networks which combine deterministic and stochastic recurrent hidden states with conditional normalizing flows to form a probabilistic neural generative model capable of describing the kind of variability observed in highly structured spatio-temporal data. Inspired by the model’s factorization, we further devise a structured variational inference network to approximate the intractable posterior distribution by exploiting a spatial representation of the data. We empirically evaluate our model against other generative models for sequential data on three real-world datasets for the task of spatio-temporal transportation demand modelling. Results show how the added flexibility allows our model to generate distributions matching potentially complex urban topologies.

1 Introduction

Building well-specified probabilistic models for sequential data is a long-standing challenge of the statistical sciences and machine learning. Historically, dynamic Bayesian networks (DBNs), such as hidden Markov models (HMMs) and state space models (SSMs), have characterized a unifying probabilistic framework with illustrious successes in modelling time-dependent dynamics. Advances in deep learning architectures however, shifted this supremacy towards the field of Recurrent Neural Networks (RNNs). At a high level, both DBNs and RNNs can be framed as parametrisations of two core components: 1) a transition function characterising the time-dependent evolution of a learned internal representation, and 2) an emission function denoting a mapping from representation space to observation space.

Despite their attractive probabilistic interpretation, the biggest limitation preventing the widespread application of DBNs in the deep learning community, is that inference can be exact only for models typically characterized by either simple transition/emission functions (e.g. linear Gaussian models) or relatively simple internal representations. On the other hand, RNNs are able to learn long-term dependencies by parametrising a transition function of richly distributed deterministic hidden states. To do so, current RNNs typically rely on gated non-linearities such as long short-term memory (LSTMs) [14] cells and gated recurrent units (GRUs) [5], allowing the learned representation to act as internal memory for the model.

More recently, evidence has been gathered in favor of combinations bringing together the representative power of RNNs with the consistent handling of uncertainties given by probabilistic approaches.
The core concept underlying recent developments is the idea that, in current RNNs, the only source of variability is found in the conditional emission distribution (i.e. typically a unimodal distribution or a mixture of unimodal distributions), making these models inappropriate when modelling highly structured data. Most efforts have therefore concentrated in building models capable of effectively propagating uncertainty in the transition function of RNNs.

In this paper, we build on these recent advances by shifting the focus towards more flexible emission functions. We suggest that the traditional treatment of output variability through the parametrisation of unimodal (or mixtures of unimodal) distributions may act as a bottleneck in cases characterized by complex data distributions. We propose the use of Conditional Normalizing Flows (CNFs) \[31\] as a general approach to define arbitrarily expressive output probability distributions under temporal dynamics.

In their basic form, normalizing flows act by propagating a simple initial distribution through a series of bijective transformations to produce a richer, more multimodal distribution. In this paper, we are specifically interested in modelling complex sequential data and propose a stochastic version of RNNs capable of exploiting the flexibility of normalizing flows in the conditional output distribution. On one hand, we model the temporal variability in the data through a transition function combining stochastic and deterministic states, on the other, we propose to use this mixed hidden representation as a conditioning variable to capture the output variability with a CNF. We call this model a Recurrent Flow Network (RFN).

We evaluate the proposed RFNs against both deterministic and stochastic variants of RNNs on three challenging spatio-temporal density estimation tasks. In particular, we focus on the problem of modelling the spatial distribution of transportation demand for the cases of New York, U.S.A. and Copenhagen, Denmark. For the explored tasks, we show how the additional emission flexibility allows RFNs to outperform mixture-based density models in capturing complex spatio-temporal dependencies. To summarize, the main contributions of this paper are threefold:

- we propose a probabilistic model which is able to combine deterministic and stochastic temporal representations with the flexibility of normalizing flows in the conditional output distribution;
- we evaluate the model on the task of spatio-temporal demand modelling, where the ability to represent spatially complex distributions is of fundamental importance. We use real-world data relevant for many transportation applications \[32, 11\];
- we use recent advances in variational inference to devise an inference network able to approximate in a scalable manner the intractable posterior distribution over the latent states by exploiting the spatial structure of the data.

2 Background

2.1 Recurrent Neural Networks and Mixture Density Outputs

Recurrent neural networks are widely used to model variable-length sequences \(x = (x_1, x_2, \ldots, x_T)\), possibly influenced by external covariates \(u = (u_1, u_2, \ldots, u_T)\). The core assumption underlying these models is that all observations \(x_{1:t}\) up to time \(t\) can be summarized by a learned deterministic representation \(h_t\). At any timestep \(t\), an RNN recursively updates its hidden state \(h_t \in \mathbb{R}^p\) by computing:

\[
h_t = h_{\theta_h}(u_t, h_{t-1}), \tag{1}
\]

where \(t\) is a deterministic non-linear transition function parametrised by \(\theta_h\), such as an LSTM cell or a GRU. The sequence is then modelled by defining a factorization of the joint probability distribution as the following product of conditional probabilities:

\[
p(x_1, x_2, \ldots x_T) = \prod_{t=1}^{T} p(x_t | x_{<t})
\]

\[
p(x_t | x_{<t}) = e_{\theta_e}(h_t), \tag{2}
\]

where \(e\) is typically a non-linear emission function with parameters \(\theta_e\).
When modelling complex real-valued sequences, a common choice is to represent the emission function with a mixture density network (MDN), as in [12]. The idea behind MDNs is to use the output of a neural network to parametrise a Gaussian mixture model. In the context of RNNs, a subset of the outputs at time \( t \) is used to define the vector of mixture proportions \( \pi_t \), while the remaining outputs are used to define the means \( \mu_t \) and covariances \( \Sigma_t \) for the corresponding mixture components. Under this framework, the probability of \( x_t \) is defined as follows:

\[
p_{\pi_t, \mu_t, \Sigma_t}(x_t|x_{<t}) = \sum_k K \pi_{k, t} \mathcal{N}(x_t|\mu_{k, t}, \Sigma_{k, t}),
\]

where \( K \) is the assumed number of components characterising the mixture.

### 2.2 Stochastic Recurrent Neural Networks

As introduced in [10], a stochastic recurrent neural network (SRNN) represents a specific architecture combining deterministic RNNs with fully stochastic SSM layers. At a high level, SRNNs build a hierarchical internal representation by stacking a SSM transition \( t_{\theta_h}(z_{t-1}, h_t) \) on top of a RNN \( t_{\theta_u}(h_{t-1}, u_t) \). The emission function is further defined by skip-connections mapping both deterministic \( (h_t) \) and stochastic \( (z_t) \) states to observation space \( (x_t) \). Assuming that the starting hidden states \( h_0, z_0 \) and inputs \( u_{1:T} \) are given, the model is defined by the following factorization:

\[
p(x_{1:T}, z_{1:T}, h_{1:T}| u_{1:T}, z_0, h_0) = \prod_{t=1}^T p(x_t|z_t, h_t)p(z_t|z_{t-1}, h_t)p(h_t|h_{t-1}, u_t)
\]

where \( e \) and \( t \) represent again the emission and transition functions and where parameters \( \theta_h, \theta_z, \theta_b \) are jointly optimized at inference time.

### 2.3 Normalizing Flows for probabilistic modelling

Normalizing flows represent a flexible approach to define rich probability distributions over continuous random variables. At their core, flow-based models define a joint distribution over a \( D \)-dimensional vector \( x \) by applying a transformation \( T \) to a real vector \( b \) sampled from a (usually simple) base distribution \( p_b(b) \):

\[
x = T(b) \quad \text{where} \quad b \sim p_b(b).
\]

In order for the density of \( x \) to be well-defined, some important properties need to be satisfied. In particular, the transformation \( T \) must be invertible and both \( T \) and \( T^{-1} \) must be differentiable. Such a transformation is known as a diffeomorphism (i.e. a bijection having invertible inverse). If these properties are satisfied, the model distribution on \( x \) can be obtained by the change of variable formula:

\[
p_x(x) = p_b(b)|\det J_T(b)|^{-1} \quad \text{and} \quad \log(p_x(x)) = \log(p_b(b)) + \log(|\det J_T(b)|^{-1}),
\]

where \( b = T^{-1}(x) \) and the Jacobian \( J_T(b) \) is the \( D \times D \) matrix of all partial derivatives of \( T \). In practice, the transformation \( T \) and the base distribution \( p_b(b) \) can have parameters of their own (e.g. \( p_b(b) \) could be a multivariate normal with mean and covariance also parametrised by any flexible function). The fundamental property which makes normalizing flows so attractive, is that invertible and differentiable transformations are composable. That is, given two transformations \( T_1 \) and \( T_2 \), their composition \( T_2 \circ T_1 \) is also invertible and differentiable, with inverse and Jacobian determinant given by:

\[
(T_2 \circ T_1)^{-1} = T_1^{-1} \circ T_2^{-1} \quad \text{and} \quad \det J_{T_2 \circ T_1}(b) = \det J_{T_2}(T_1(b)) \cdot \det J_{T_1}(b).
\]

\(^1\)Not to be confused with the time-horizon \( T \) from e.g. Eq. [2]. In general, the distinction should be clear from the context.
As a result, this framework allows to construct arbitrarily complex transformations by composing multiple stages of simpler transformations, without sacrificing the ability of exactly calculating the (log) density \( p_x(x) \).

In [7], the authors introduce a bijective function of particular interest for this paper. This transformation, known as an affine coupling layer, exploits the simple observation that the determinant of a triangular matrix can be efficiently computed as the product of its diagonal terms. Concretely, given a \( D \)-dimensional input vector \( x \) and \( d < D \), this property is exploited by defining the output \( b \) of an affine coupling layer as follows:

\[
\begin{align*}
    b_{1:d} &= x_{1:d} \\
    b_{d+1:D} &= x_{d+1:D} \circ \exp(s(x_{1:d})) + t(x_{1:d}),
\end{align*}
\]

where \( s \) and \( t \) are arbitrarily complex scale and translation functions from \( \mathbb{R}^d \rightarrow \mathbb{R}^{D-d} \) and \( \circ \) is the element-wise or Hadamard product. Since the forward computation defined in Eq. (9) and Eq. (10) leaves the first \( d \) components unchanged, these transformations are usually combined by composing coupling layers in an alternating pattern, so that components unchanged in one layer are effectively updated in the next (for a more in-depth treatment of normalizing flows, the reader is referred to [24][20]).

3 Recurrent Flow Networks

In this section, we define the generative model \( p_\theta \) and inference network \( q_\psi \) characterising the RFN for the purpose of sequence modelling. RFNs explicitly model temporal dependencies by combining deterministic and stochastic layers. The resulting intractability of the posterior distribution over the latent states \( z_{1:T} \), as in the case of VAEs [19][27], is further approached by learning a tractable approximation through amortised variational inference. The schematic view of the RFN is shown in Fig.1.

**Generative model** As in [10], the transition function of the RFN interlocks an SSM with an RNN:

\[
\begin{align*}
    h_t &= t_{\theta_h}(h_{t-1}, \varphi^\text{extr}(u_t)) \\
    z_t &\sim \mathcal{N}(\mu_{0,t}, \text{diag}(\sigma_{0,t}^2)), \quad \text{where } [\mu_{0,t}, \sigma_{0,t}] = t_{\theta_0}(z_{t-1}, h_t),
\end{align*}
\]

where \( \mu_{0,t} \) and \( \sigma_{0,t} \) represent the parameters of the conditional prior distribution over the stochastic hidden states \( z_{1:T} \). In our implementation, \( t_{\theta_h} \) and \( t_{\theta_0} \) are respectively an LSTM cell and a deep feed-forward neural network, with parameters \( \theta_h \) and \( \theta_0 \). In Eq. (11), \( \varphi^\text{extr} \) can also be a neural network extracting features from \( u_t \). Unlike the SRNN, the learned representations (i.e. \( z_{1:T}, h_{1:T} \)) are used as conditioners for a CNF parametrising the output distribution. That is, for every time-step \( t \), we learn a complex distribution \( p(x_t|z_t, h_t) \) by defining the conditional base distribution \( p(b_t|z_t, h_t) \) and conditional coupling layers characterising the transformation \( T_\psi \) as follows:

\[
\begin{align*}
    \text{Conditional Prior: } b_t &\sim \mathcal{N}(\mu_{b,t}, \text{diag}(\sigma_{b,t}^2)), \quad \text{where } [\mu_{b,t}, \sigma_{b,t}] = f_\psi(z_t, h_t) \\
    \text{Conditional Coupling: } b_{t,1:d} &= x_{t,1:d} \circ \exp(s_\psi(x_{t,1:d}, z_t, h_t)) + t_\psi(x_{t,1:d}, z_t, h_t)
\end{align*}
\]

where \( \mu_{b,t} \) and \( \sigma_{b,t} \) represent the parameters of the conditional base distribution (determined by a learnable function \( f_\psi \)), while \( s_\psi \) and \( t_\psi \) denote the conditional scale and translation functions characterising the coupling layers in the CNF. In our implementation, \( f_\psi, s_\psi \) and \( t_\psi \) are parametrised by deep neural networks. Together, Eq. (13) and Eq. (14) define the emission function \( e_{\theta_e}(z_t, h_t) \), enabling the generative model to result in the factorization in Eq. (4).

**Inference** The variational approximation defining the RFN directly depends on \( z_{t-1}, h_t \) and \( x_t \) as follows:

\[
\begin{align*}
    z_t|x_t \sim \mathcal{N}(\mu_{z,t}, \text{diag}(\sigma_{z,t}^2)), \quad \text{where } [\mu_{z,t}, \sigma_{z,t}] = \varphi^\text{enc}(z_{t-1}, h_t, x_t),
\end{align*}
\]

where \( \varphi^\text{enc} \) is an encoder network defining the parameters of the approximate posterior distribution \( \mu_{z,t} \) and \( \sigma_{z,t} \). Given the above structure, the generative and inference models are tied through the RNN hidden state \( h_t \), resulting in the factorization given by:

\[
q_\psi(z_{1:T}|x_{1:T}) = \prod_{t=1}^{T} q_\psi(z_t|z_{t-1}, h_t, x_t).
\]
In addition to the explicit dependence of the approximate posterior on $x_t$ and $h_t$, the inference network defined in Eq. (15) also exhibits an implicit dependence on $x_{1:t}$ and $h_{1:t}$ through $z_{t-1}$. This implicit dependency on all information from the past can be considered as resembling a filtering approach from the state-space model literature [8]. Denoting $\theta$ and $\phi$ as the set of model and variational parameters respectively, variational inference offers a scheme for jointly optimising parameters $\theta$ and computing an approximation to the posterior distribution by maximising the following step-wise evidence lower bound (i.e. ELBO):

$$L(\theta, \phi) = \mathbb{E}_{q_\phi(z_{1:T} | x_{1:T})} \left[ \sum_{t=1}^{T} \log p_\theta(x_t | z_t, h_t) + \log p_\theta(h_t | h_{t-1}, u_t) - KL(q_\phi(z_t | z_{t-1}, h_t, x_t) || p_\theta(z_t | z_{t-1}, h_t)) \right].$$

(17)

The generative and inference models are therefore learned jointly in $\{\theta, \phi\}$ space, so that the variational approximation $q_\phi(z_t | z_{t-1}, h_t, x_t)$ is effectively tracking a moving posterior $p_\theta(z_t | z_{t-1}, h_t)$.

4 Experiments

In this paper, we are interested in modelling the spatio-temporal demand distribution for different transportation services. The complex spatial structure (in latitude-longitude space), together with the inherent temporal dynamics characterising the demand distribution, make this problem particularly relevant from both a methodological and applied standpoint. Being able to model and accurately forecast the need for transportation could allow service providers and institutions to guarantee more efficient systems, ultimately leading to reduced traffic congestion and lower emissions. We evaluate the proposed RFN on three transportation datasets:

- **NYC Taxi (NYC-P/D):** This dataset is released by the New York City Taxi and Limousine Commission. We focused on aggregating the taxi demand in 2-hour bins for the month of March 2016 containing 249,637 trip geo-coordinates. We further differentiated the task of modelling pick-ups (i.e. where the demand is) and drop-offs (i.e. where people want to go). In what follows, we denote the two datasets as NYC-P and NYC-D respectively.

- **Copenhagen Bike-Share (CPH-BS):** This dataset contains geo-coordinates from users accessing the smartphone app of Donkey Republic, one of the major bike sharing services in Copenhagen, Denmark. As for the case of New York, we aggregated the geo-coordinates in 2-hour bins for the month of August, resulting in 87,740 app accesses.

\footnote{Code available at \url{https://github.com/DanieleGammelli/recurrent-flow-nets}}
For both New York and Copenhagen experiments we processed the data so to discard corrupted geo-coordinates outside the area of interest. For the taxi experiments, we discarded coordinates related to trips either shorter than 30s or longer than 3h, while in the bike-sharing dataset, we ensured to keep only one app access from the same user in a window of 5 minutes. In both cases we divide the data temporally into train/validation/test splits using a ratio of 0.5/0.25/0.25.

Training: We train each model using stochastic gradient ascent on the evidence lower bound \( \mathcal{L}(\theta, \phi) \) defined in Eq. (17) using the Adam optimizer [17], with a starting learning rate of 0.003 being reduced by a factor of 0.1 every 100 epochs without loss improvement (in our implementation, we used the ReduceLROnPlateau scheduler in PyTorch with patience=100). As in [29], we found that annealing the KL term in Eq. (17) (using a scalar multiplier linearly increasing from 0 to 1 over the course of training) yielded better results. The final model was selected with an early-stopping procedure based on the validation performance. Training using a NVIDIA GeForce RTX 2080 Ti took around 6 hours for CPH-BS and around 9 hours for NYC-P/D.

Models: We compare the RFN with RNN, VRNN [6] and SRNN [10] models using two different MDN-based emission distributions. In particular, we compare against a GMM output parametrised by Gaussians with either diagonal (MDN-Diag) or full (MDN-Full) covariance matrix. Based on a random search, we use 50 and 30 mixtures for MDN-Diag and MDN-Full respectively. For every model, we select a single layer of 128 LSTM cells. The feature extractor \( \varphi_x^{\text{enc}} \) in Eq. (11) has three layers of 128 hidden units using rectified linear activations [23]. For the VRNN, SRNN and RFN we also define a 128-dimensional latent state \( z_{1:T} \). Both the transition function \( t_\theta \) from Eq. (12) and the inference network \( \varphi_z^{\text{enc}} \) in Eq. (15) use a single layer of 128 hidden units. For the mixture-based models, the MDN emission is further defined by two layers of 64 hidden units where we use a softplus activation to ensure the positivity of the variance vector in the MDN-Diag case and a Cholesky decomposition of the full covariance matrix in MDN-Full. The emission function in the RFN is defined as in Eq. (13) and Eq. (14), where \( f_\psi, s_\psi, \text{ and } t_\psi \) are neural networks with two layers of 128 hidden units. The conditional function is further defined as an alternation of 35 layers of the triplet [Affine coupling layer, Batch Normalization [15], Permutation], where the permutation ensures that all dimensions are processed by the affine coupling layers and where the batch normalization ensures better propagation of the training signal, as shown in [7]. In our experiments we define \( u_i = x_{t-1} \), although \( u_i \) could potentially be used to introduce relevant information for the problem at hand (e.g. weather or special event data in the case of spatio-temporal transportation demand estimation).

All models were implemented using PyTorch [25] and the universal probabilistic programming language Pyro [2]. To reduce computational cost, we use a single sample to approximate the intractable expectations in the ELBO.

Spatial representation: For the task of spatio-temporal density estimation, \( x_{1:T} \) takes the form of a set of variable-length samples from the target distribution \( p(x_{1:T}) \). That is, for every time-step \( t \), \( x_t \) is a vector of geo-coordinates representing a corresponding number of taxi trips (NYC-P/D) or smartphone app accesses (CPH-BS). We propose to process the data into a representation enabling the models to effectively handle data in a single batch computation. As shown in Fig. 2, we choose to represent \( x_t \) as a \( k \times k \) normalized 2-dimensional histogram (in our implementation we set \( k = 64 \)). Given its ability to preserve the spatial structure of the data, we believe this representation to be well suited for spatio-temporal density estimation tasks. More precisely, the proposed representation is obtained by applying the following three-step procedure: 1) select data \( x_t \), 2) build a 2-dimensional histogram computing the counts \( c_{ij}, i, j = 1, \ldots, k \) of the geo-coordinates falling in every cell of the \( k \times k \) grid and 3) normalize the histogram such that \( \sum_{i,j} c_{ij} = 1 \). By fixing \( u_i = x_{t-1} \), this enables the definition of a sequence generation problem over spatial densities. In practice, we found the above spatial representation to be both practical in dealing with variable-length geo-coordinate vectors, as well as effective, yielding better results. To the authors’ best knowledge, this spatial approximation of the target distribution has never been used for the task of spatio-temporal density modelling.

\[^1\]In our implementation, we used a variation of the pre-processing from [https://github.com/hughsalimbeni/bayesian_benchmarks/blob/master/bayesian_benchmarks/data.py](https://github.com/hughsalimbeni/bayesian_benchmarks/blob/master/bayesian_benchmarks/data.py)
Table 1: Test log-likelihood for each task. For the non-deterministic models (VRNN, SRNN, RFN) the approximation on the marginal log-likelihood is given with the $\approx$ sign.

| Models     | NYC-P  | NYC-D  | CPH-BS |
|------------|--------|--------|--------|
| RNN-MDN-Diag | 163582 | 143765 | 49124  |
| RNN-MDN-Full | 164016 | 146676 | 50109  |
| VRNN-MDN-Diag | $\approx$ 161345 | $\approx$ 139964 | $\approx$ 49231 |
| VRNN-MDN-Full | $\approx$ 162549 | $\approx$ 143671 | $\approx$ 49664 |
| SRNN-MDN-Diag | $\approx$ 164830 | $\approx$ 143719 | $\approx$ 49331 |
| SRNN-MDN-Full | $\approx$ 164976 | $\approx$ 147400 | $\approx$ 49810 |
| RFN        | $\approx$ 168734 | $\approx$ 148291 | $\approx$ 51100 |

Results: In Table 1 we compare test log-likelihoods on the tasks of spatio-temporal demand modelling for the cases of New York and Copenhagen. We report exact log-likelihoods for both RNN-MDN-Diag and RNN-MDN-Full, while in the case of VRNNs, SRNNs and RFNs, given their inherent stochasticity, we report the importance sampling approximation to the marginal log-likelihood as stated in [28] using 30 samples. We see from Table 1 that RFN outperforms competing methods yielding higher log-likelihood. The results support our claim that more flexible output distributions are advantageous when modelling potentially complex and structured temporal data distributions.

In Fig. 2, we show a visualization of the predicted spatial densities from three of the implemented models at specific times of the day. The heatmap was generated by computing the approximation of the marginal log-likelihood, under the respective model, on a $110 \times 110$ grid within the considered geographical boundaries. The final plot is further obtained by mapping the computed log-likelihoods back into latitude-longitude space. Opposed to GMM-based densities, the figures show how the RFN exploits the flexibility of conditional normalizing flows to generate sharper distributions capable of better approximating complex shapes such as geographical landforms or urban topologies.

5 Related Work

A number of works have concentrated on defining more flexible emission functions for sequential models [26, 22, 4]. As we have in this paper, these works argue that simpler output models may turn out limiting when dealing with structured and potentially high dimensional data distributions (e.g. images, videos). The performance of these models is highly dependent on the specific architecture defined in the conditional output distribution, as well as on how stochasticity is propagated in the transition function. In this section we highlight how RFNs differ from some of these works.

In VideoFlow [22] and in [26], the authors similarly use normalizing flows to parametrise the emission function for the tasks of video generation and multi-variate time series forecasting, respectively. In VideoFlow, the latent states representing the temporal evolution of the system are defined by the conditional base distribution of a normalizing flow. This differs from our work where we explicitly model the temporal dynamics through a combination of latent variables and fully deterministic recurrent hidden states. We found this mixed hidden representation able to yield better performance in practice. Normalizing flows in RFNs are therefore exclusively used to model output variability through the emission function, rather than directly parametrising the recurrent hidden states. Moreover, the architecture of VideoFlow is inspired by Glow [18], and so specifically tailored for image generation tasks (e.g. through the proposed 3D multi-scale latent variables and the 3D dilated Convolutional Residual Network). Similarly to our work, in [26] the authors also propose to use conditional
affine coupling layers in order to model the output variability. The RFNs differ clearly through the combination of stochastic and deterministic recurrent hidden states in the transition function.

In [4], the authors address the task of video generation by defining a hierarchical version of the VRNN and a Conv-LSTM decoder. While the latter also combines stochastic and deterministic states, as for the case of VideoFlow, the emission function is specifically focused on image modelling tasks.

6 Conclusion

This work addresses the problem of spatio-temporal density modelling by proposing the use of conditional normalizing flows as a general approach to parametrise the output distribution of recurrent latent variable models. We approximate the intractable posterior distribution over the latent states by devising an inference network able to exploit the spatio-temporal structure of the data distribution. We also propose to use a spatial representation of data to effectively represent samples from densities in geo-coordinate space within temporal models. Our experiments focus on real-world data for the task of transportation demand density modelling. We empirically show that the flexibility of normalizing flows enables RFNs to generate rich output distributions capable of describing potentially complex geographical surfaces.

In future work, similarly to [22], we plan to apply RFNs for the task of video generation. We believe the combination of deterministic and stochastic hidden representations could enable more reliable long-term predictions compared to fully stochastic states. We also plan to explore the role of multi-head attention mechanisms [30] for the efficient and effective learning of both long-term and short-term dependencies between the currently Markovian stochastic states.
References

[1] Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks, 2014.

[2] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. Journal of Machine Learning Research, 2018.

[3] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription, 2012.

[4] Lluis Castrejon, Nicolas Ballas, and Aaron Courville. Improved conditional vrnns for video prediction. In The IEEE International Conference on Computer Vision (ICCV), October 2019.

[5] Junyoung Chung, Çağlar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555, 2014.

[6] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. CoRR, abs/1506.02216, 2015.

[7] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. CoRR, abs/1605.08803, 2016.

[8] James Durbin and Siem Jan Koopman. Time Series Analysis by State Space Methods. Oxford University Press, 2 edition, 2012.

[9] Otto Fabius and Joost R. van Amersfoort. Variational recurrent auto-encoders, 2014.

[10] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers, 2016.

[11] Xu Geng, Yaguang Li, Leye Wang, Lingyu Zhang, Qiang Yang, Jieping Ye, and Yan Liu. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. Proceedings of the AAAI Conference on Artificial Intelligence, 33:3656–3663, 07 2019.

[12] Alex Graves. Generating sequences with recurrent neural networks. CoRR, abs/1308.0850, 2013.

[13] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels, 2018.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Comput., 9(8):1735–1780, November 1997.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[16] Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data, 2016.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[18] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions, 2018.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[20] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. IEEE Transactions on Pattern Analysis and Machine Intelligence, page 1–1, 2020.

[21] Rahul G. Krishnan, Uri Shalit, and David Sontag. Deep kalman filters, 2015.
[22] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A flow-based generative model for video. CoRR, abs/1903.01434, 2019.

[23] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML’10, page 807–814, Madison, WI, USA, 2010. Omnipress.

[24] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference, 2019.

[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[26] Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. Multi-variate probabilistic time series forecasting via conditioned normalizing flows, 2020.

[27] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models, 2014.

[28] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models, 2014.

[29] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders, 2016.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. CoRR, abs/1706.03762, 2017.

[31] Christina Winkler, Daniel Worrall, Emiel Hoogeboom, and Max Welling. Learning likelihoods with conditional normalizing flows, 2019.

[32] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. Proceedings of the AAAI Conference on Artificial Intelligence, 33:5668–5675, Jul 2019.