A. Deep Factor Models with Random Effects

In this section, we provide additional details on the models discussed in Section 3.1. In what follows, we overload the notation and use $\boldsymbol{g}_t(\cdot) : \mathbb{R}^d \to \mathbb{R}^K$ to denote the vector function consisting of the global factors.

NAME	DESCRIPTION	LOCAL	LIKELIHOOD (GAUSSIAN CASE)
DF-RNN	Zero-mean Gaussian noise process given by RNN	$r_{i,t} \sim \mathcal{N}(0, \sigma_{i,t}^2)$	$p(\boldsymbol{z}_i) = \prod_t \mathcal{N}(z_{i,t} - f_{i,t} \vert 0, \sigma_{i,t}^2)$
DF-LDS	State-space models	$r_{i,t} \sim \text{LDS}_{i,t}$ (cf. Eqn. (6))	$p(\boldsymbol{z}_i)$ given by Kalman Filter
DF-GP	Zero-mean Gaussian Process	$r_{i,t} \sim \text{GP}_{i,t}$ (cf. Eqn. (7))	$p(\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{z}_i - \boldsymbol{f}_i \vert \boldsymbol{0}, \mathbf{K}_i + \boldsymbol{\sigma}_i^2\mathbf{I})$

Table A.1. Summary table of Deep Factor Models with Random Effects. The likelihood column is under the assumption of Gaussian noise.

A.1. DF-RNN: Gaussian noise process as the local model

The local model in DF-RNN is zero-mean i.i.d. Gaussian, where the variance $\sigma_{i,t}^2$ is generated by a noise RNN process. The generative model is given as:

$$\begin{aligned}
\text{random effect} : \quad & \varepsilon_{i,t} \sim \mathcal{N}(0, \sigma_{i,t}^2), \quad \sigma_{i,t} := \text{RNN}(x_{i,t}),\\
\text{fixed effect} : \quad & f_{i,t} = \boldsymbol{w}_i^\top \boldsymbol{g}_t(x_{i,t}),\\
\text{emission} : \quad & z_{i,t} \sim p(\cdot | u_{i,t}),\ u_{i,t} = f_{i,t} + r_{i,t}.
\end{aligned}$$

Although the noise is i.i.d., there is correlated noise from the latent RNN process. For the non-Gaussian likelihood, the latent innovation terms $\boldsymbol{\varepsilon}$ in the latent function $\boldsymbol{u}$ requires inference. To do so, we use Algorithm 1. The variational lower bound in Eqn. (10) is straightforward to calculate, since the marginal likelihood $\log p(\widetilde{\boldsymbol{u}})$ is the log-likelihood of a multivariate Gaussian with mean $w^\top g_t$ and a covariance matrix with ε_t^2 on the diagonal entries. DF-RNN can be seen as a type of deep Gaussian latent variable model.

A.2. DF-LDS: LDS as the local model

The generative model for DF-LDS is given as:

$$\begin{aligned}
\text{random effect} : \quad & \boldsymbol{h}_{i,t} = \mathbf{F}_{i,t}\boldsymbol{h}_{i,t-1} + \boldsymbol{q}_{i,t}\epsilon_{i,t}, \quad \epsilon_{i,t} \sim \mathcal{N}(0,1),\\
& r_{i,t} = \boldsymbol{a}_{i,t}^\top \boldsymbol{h}_{i,t}\\
\text{fixed effect} : \quad & f_{i,t} = \boldsymbol{w}_i^\top \boldsymbol{g}_t(x_{i,t}),\\
\text{emission} : \quad & z_{i,t} \sim p(\cdot | u_{i,t}),\ u_{i,t} = f_{i,t} + r_{i,t}.
\end{aligned}$$

We consider the level-trend ISSM model with damping,

$$\boldsymbol{a}_{i,t} = \begin{bmatrix} \delta_i \\ \gamma_i \end{bmatrix}, \quad \mathbf{F}_{i,t} = \begin{bmatrix} \delta_i & \gamma_i \\ 0 & \gamma_i \end{bmatrix}, \quad \text{and} \quad \boldsymbol{q}_{i,t} = \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix},$$

where δ_i, γ_i control the damping effect, and the parameters $\alpha_i > 0$ and $\beta_i > 0$ contain the innovation strength for the level and trend (slope), respectively. The initial latent state $h_{i,0}$ is assumed to follow an isotropic Gaussian distribution. We choose a simple level-trend ISSM because we expect the global factors with attention to explain the majority of the time series' dynamics, such as trend and seasonality. This avoids having to hand design the latent states of the ISSM.

As a special case, we recover SSM with non-Gaussian likelihood, and our inference algorithm gives a new approach to perform inference on such models, which we call *Variational LDS*.

Variational LDS: DF-LDS with no global factors Variational LDS is a state-space model (SSM) with an arbitrary likelihood as its emission function. In this subsection, we supply additional details of the inference of the variational LDS using the synthetic example in Section 5.1. The data is generated as,

$$h_{t+1} = Ah_t + \epsilon_h, \quad A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix},$$

$$\epsilon_h \sim \mathcal{N}(0, \alpha^2 \mathbf{I}_2), \quad v_{t+1} = e_1^T h_{t+1} + \epsilon_v, \quad \epsilon_v \sim \mathcal{N}(0, \sigma^2).$$

Our observations are generated as:

$$z_t = \text{Poisson}[\lambda(v_t)], \qquad \lambda(v_t) = \log[1 + \exp(v_t)],$$

resulting in the popular Poisson LDS.

Our goal is perform inference over the latent variables, $\{h, \boldsymbol{u}\}$, where $h = h_{1:T}$ and $\boldsymbol{u} = v_{1:T}$, given the observations $z := z_{1:T}$. As in Section 3.2.2, we use a variational auto-encoder (VAE) with structural approximation, and a variational posterior to approximate the true posterior $p(h, \boldsymbol{u}|z)$ as $q_\phi(h, \boldsymbol{u}|z) = q_\phi(\boldsymbol{u}|z)p(h|\boldsymbol{u})$. As in DF-LDS, the first term is generated by the recognition network, and the second term chosen to match the conditional posterior. For the training procedure, we resort to Algorithm 1 with no global factors, i.e., $\widetilde{r}$ is the sample generated by the recognition network.

There are different neural networks structures to choose from for the variational encoder $q_\phi(h|\boldsymbol{z})$. We choose the bi-directional LSTM since it considers information from both the past and the future. This is similar to the backwards message in the Kalman smoother. Similar preference can also be found in (Krishnan et al., 2015), although their model structure differs. The recognition network is only used in the training phase to better approximate the posterior distribution of the latent function values, and so there is no information leak for the desired forecasting case.

A.3. DF-GP: Gaussian Process as the local model

With DF-GP, the random effects is assumed to be drawn from a zero-mean Gaussian process.

$$\begin{aligned} \text{random effect :} \quad & r_i(\cdot) \sim \text{GP}(0, \mathcal{K}_i(\cdot, \cdot)), \\ \text{fixed effect :} \quad & f_{i,t} = \boldsymbol{w}_i^\top \boldsymbol{g}_t(x_{i,t}), \\ \text{latent function values :} \quad & u_{i,t} = f_{i,t} + r_{i,t}, \quad r_{i,t} = r_i(\boldsymbol{x}_{i,t}), \\ \text{emission :} \quad & z_{i,t} \sim p(\cdot|u_{i,t})). \end{aligned}$$

Simiarily with Variational LDS, with the proposed inference algorithm, we get a new approach for doing approximate inference for GP with non-Gaussian likelihood. We leave the comparison with classical approaches (Rasmussen & Williams, 2006) such as Laplace approximation, Expectation Propagation, Variational Inference to future work.

B. Detailed Experimental Analysis

In this appendix, we provide further results in both the synthetic and empirical cases.

B.1. Synthetic Experiments

The local model and observations are generated using the variational-LDS example in Appendix A.2. We want to test the ability of the algorithm to recover the underlying latent factors. To do so, we add global factors that are generated from Fourier series of different orders, to the synthetic dataset described in Section 5.1. For each time series, the coefficients of the linear combination are sampled from a uniform distribution.

In the presence of Gaussian LDS noise ($\boldsymbol{u}_t$ observed), Figure B.1 shows that the algorithm does not precisely recover the global factors. The distance between the subspaces spanned by the true and estimated factors reveals that they are reasonably close. Even with Poisson observations in Figure B.2, the method is still able to roughly recover the factors.

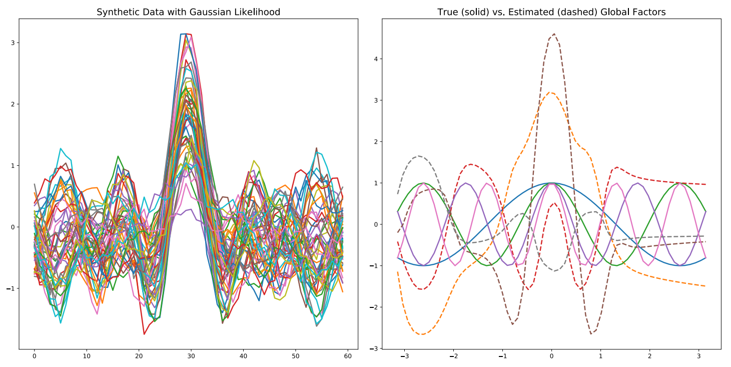

Figure B.1. Synthetic data with Gaussian noise and true (solid) vs. estimated (dashed) global factors.

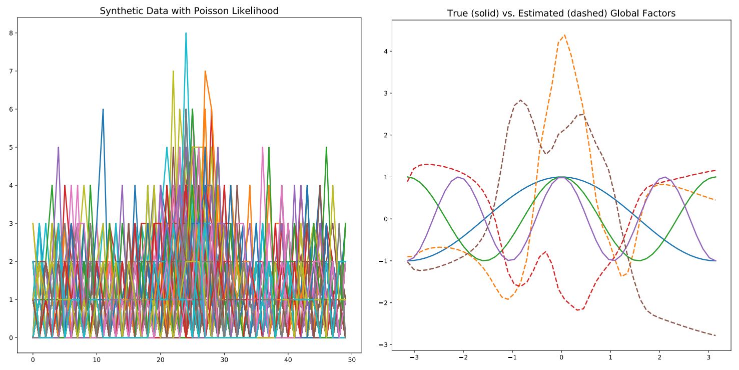

Figure B.2. Synthetic data with Poisson noise and true (solid) vs. estimated (dashed) global factors.

B.2. Empirical Studies

Table B.1 summarizes the datasets that are used to test our Deep Factor models.

NAME	SUPPORT	GRANULARITY	NO. TS	COMMENT
`electricity`	$\mathbb{R}^+$	hourly	370	electricity consumption of different customers
`traffic`	$[0, 1]$	hourly	963	occupancy rate of SF bay area freeways
`taxi`	$\mathbb{N}^+$	hourly	140	No. of taxi pick ups in different blocks of NYC
`uber`	$\mathbb{N}^+$	hourly	114	No. of Uber pick ups in different blocks of NYC

Table B.1. Summary of the datasets used to test the models.

We first compare our two methods, DF-RNN and DF-LDS with level-trend ISSM, on the `electricity` dataset with forecast horizon 24. Table B.2 shows the averaged results over 50 trials. We see that both methods have similar accuracy with DF-RNN being slightly preferrable. Due to its simplicity and computational advantages, we choose DF-RNN as the Deep Factor model to compare against the other baseline methods. Regarding DF-GP, although it shows a good performance in our perliminary results, still extra exploration is needed for which we left to future work.

METHOD	P50QL	P90QL	RMSE
DF-RNN	0.144 ± 0.019	0.101 ± 0.044	888.285 ± 278.798
DF-LDS	0.153 ± 0.016	0.135 ± 0.052	1012.551 ± 240.118

Table B.2. Comparison of DF-RNN and DF-LDS with level-trend ISSM on `electricity` over 50 runs.

We provide the result of one trial for Prophet in Table 2 since its classical methods have less variability between experiments and are more computationally expensive. The acronyms DA, P, MR, DF in these tables stand for DeepAR, Prophet, MQ-RNN and DeepFactor, respectively. Figure B.5 illustrates example forecasts from our model.

In our experiments, we do not tune the hyper-parameters of the methods. With tuning and a much bigger model, DeepAR would achieve higher accuracy in our preliminary experiments. For example, with the SageMaker HPO, we achieve the DeepAR's best performance of 0.118 MAPE and 0.042 P90 Loss on the `electricity` dataset for horizon 72. The hyper-parameters are selected to be a two-layer LSTM each with 40 units, a learning rate of 5e-3, 2000 epochs and an early stopping criteria of 200 epochs. As we can see, this metric is on par with our results in Table 2 and DF achieves such accuracy with much less number of parameters.

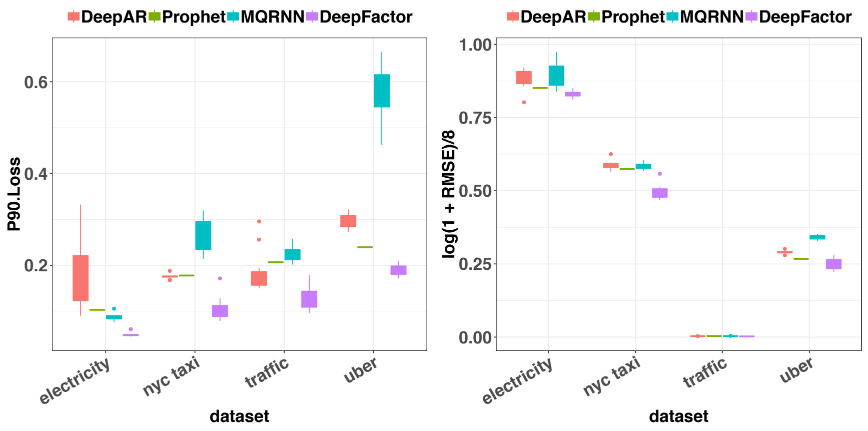

Figure B.3. P90QL and $\log(1 + \text{RMSE})$ normalized by 8 results for the forecast horizon 72 in Tables 2 and B.3, respectively. Purple denotes the proposed method.

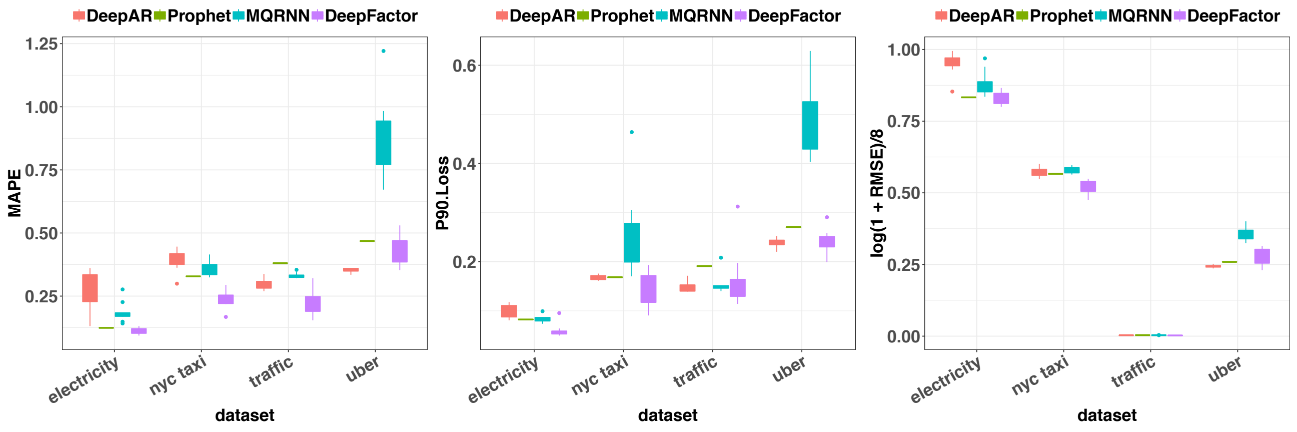

Figure B.4. P50QL (MAPE) P90QL and $\log(1 + \text{RMSE})/8.0$ (to make them in the same scale) results for forecast horizon 24 in Tables 2 and B.3, respectively. Purple denotes the proposed method.

DS	H	RMSE			
		DA	P	MR	DF
E	72	1194.421± 295.603	902.724	1416.992 ± 510.705	**770.437** ± **68.151**
	24	2100.927 ± 535.733	**783.598**	1250.233 ± 509.159	786.667 ± 144.459
N	72	108.963 ± 15.905	97.706	106.588 ± 9.988	**53.523** ± **13.096**
	24	98.545 ± 13.561	91.573	102.721 ± 9.640	**63.059** ± **11.876**
T	72	0.027 ± 0.001	0.032	0.029 ± 0.003	**0.019** ± **0.002**
	24	0.025 ± 0.001	0.028	0.028 ± 0.001	**0.021** ± **0.003**
U	72	9.218 ± 0.486	7.488	14.373 ± 0.997	**6.393** ± **1.185**
	24	**6.043** ± **0.295**	6.948	16.585 ± 3.452	8.044 ± 2.307

Table B.3. RMSE: Results for short-term (3-day forecast, 72-hour) and near-term (24-hour forecast) scenario with one week of training data.

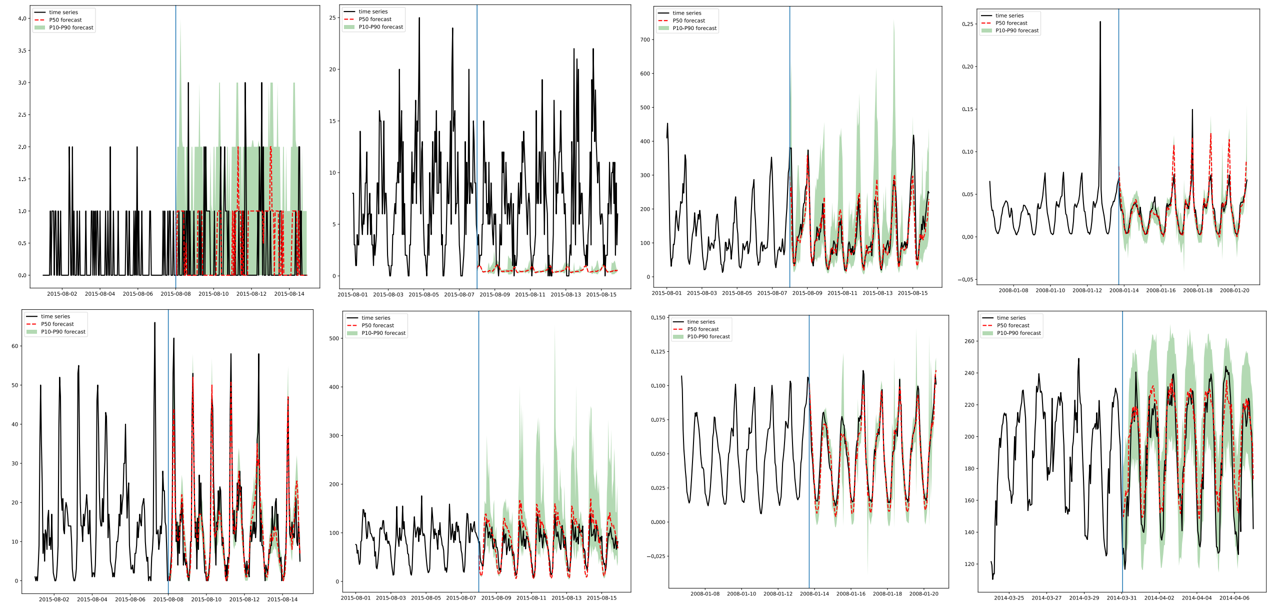

Figure B.5. Example forecast for (top row) `uber` (left) and `taxi` (center left) in the cold-start situation, and regular setting (center right) and `traffic`. For the `taxi` cold-start example, our model fails to produce a reasonable forecast, possibly due to insufficient information of corresponding latitude and longitude. Bottom role: another set of example forecasts for `uber, taxi, traffic` and `electricity`.