Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting

Kashif Rasul[*] • Calvin Seward • Ingmar Schuster • Roland Vollgraf

Zalando Research @ ICML 2021

[*] kashif.rasul@zalando.de

Neural Probabilistic Forecasts

- Probabilistic forecasts account for uncertainty
- Individual time series can be statistically dependant
- Model needs to account for this via **multivariate forecasts**
    - e.g. in traffic flow a disruption will ripple to nearby streets etc.
    - can resort to low-rank approximation or diagonal distribution
    - can use normalizing flows or GANs

This work proposes a denoising diffusion based model for probabilistic multivariate time series forecasting.

Diffusion Models (Sohl-Dickstein et al. 2015)

- A class of latent variable models for $\mathbf{x}^0 \in \mathbb{R}^D$ with $\mathbf{x}^0 \sim q(\mathbf{x}^0)$:

$$p_\theta(\mathbf{x}^0) := \int p_\theta(\mathbf{x}^{0:N})\, d\mathbf{x}^{1:N}$$

- N latents: $\mathbf{x}^1, \ldots, \mathbf{x}^N \in \mathbb{R}^D$
- **Fixed** diffusion (forward) process is a Markov chain that adds noise via given $\beta_1, \ldots, \beta_N$, with $\beta_n \in [0, 1]$:

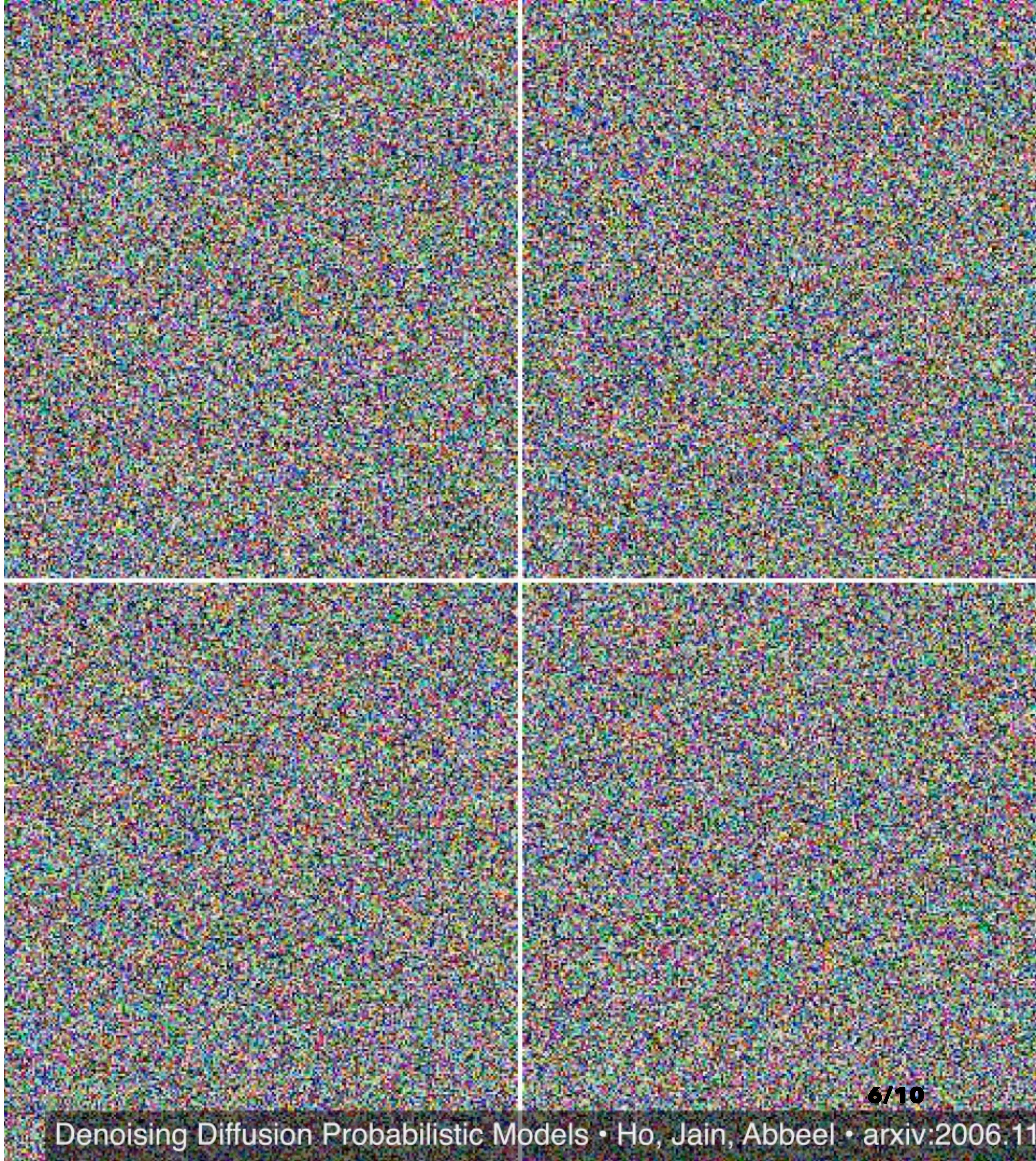
$$q(\mathbf{x}^n | \mathbf{x}^{n-1}) := \mathcal{N}(\mathbf{x}^n; \sqrt{1-\beta_n}\mathbf{x}^{n-1}, \beta_n \mathbf{I})$$

Reverse Process

- Joint $p_\theta(\mathbf{x}^{0:N}) := p(\mathbf{x}^N)\Pi_{n=N}^{1} p_\theta(\mathbf{x}^{n-1}|\mathbf{x}^n)$ is a **learned** Markov chain
- Start from $p(\mathbf{x}^N) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Each transition is given by Gaussian with shared parameters: $p_\theta(\mathbf{x}^{n-1}|\mathbf{x}^n) := \mathcal{N}(\mathbf{x}^{n-1}; \mu_\theta(\mathbf{x}^n, n), \sigma_\theta(\mathbf{x}^n, n)\mathbf{I})$
    - Where $\mu_\theta: \mathbb{R}^D \times \mathbb{N} \to \mathbb{R}^D$ and $\sigma_\theta: \mathbb{R}^D \times \mathbb{N} \to \mathbb{R}^+$ are neural networks

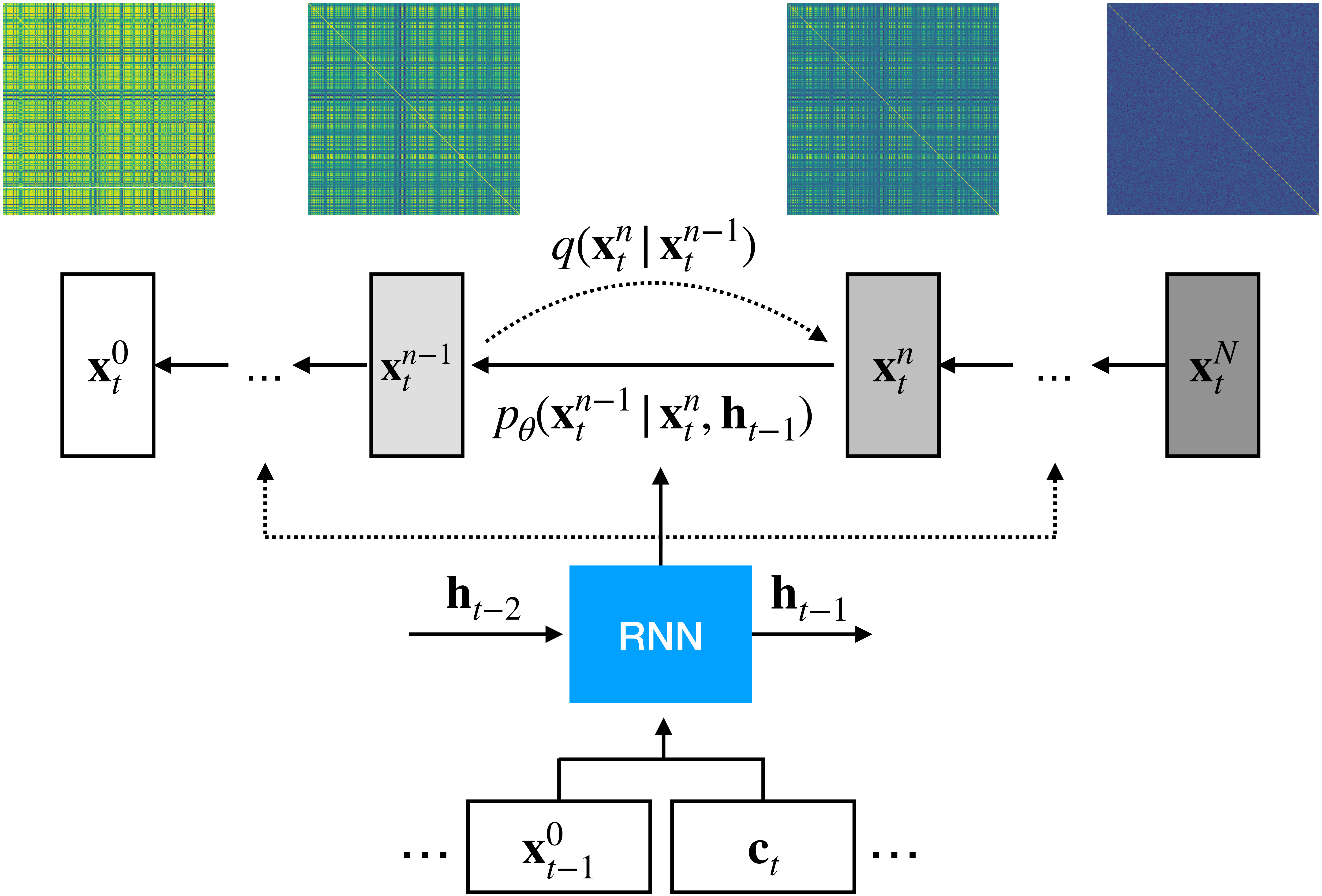
High level intuition

- Sample from distribution by gradual reversal of noising process
- Starting with white noise the learned model produces slightly less noisy signal
- Repeating this N times gives us a data sample
- **Training** can be expressed as a regression problem!
- Can learn complex distribtions (Ho et al. 2020)



TimeGrad

- We could learn $p_\theta(\mathbf{x}_t^0|\mathbf{h}_{t-1})$ using a conditional Denosing Diffusion Model for each t
- $\mathbf{h}_t$ state of an RNN from the history of the time series and covariate
- Again: minimizing $-\log p_\theta(\mathbf{x}_t^0|\mathbf{h}_{t-1})$ corresponds to a regression problem for each time step!


$q(\mathbf{x}_t^n \mid \mathbf{x}_t^{n-1})$
$\mathbf{x}_t^0$
…
$\mathbf{x}_t^{n-1}$
$\mathbf{x}_t^n$
…
$\mathbf{x}_t^N$
$p_\theta(\mathbf{x}_t^{n-1} \mid \mathbf{x}_t^n, \mathbf{h}_{t-1})$
$\mathbf{h}_{t-2}$
RNN
$\mathbf{h}_{t-1}$
…
$\mathbf{x}_{t-1}^0$
$\mathbf{c}_t$
…

Method	Exchange	Solar	Electricity	Traffic	Taxi	Wikipedia
VES	**0.005**$\pm$0.000	0.9$\pm$0.003	0.88$\pm$0.0035	0.35$\pm$0.0023	-	-
VAR	**0.005**$\pm$0.000	0.83$\pm$0.006	0.039$\pm$0.0005	0.29$\pm$0.005	0.292$\pm$0.000	3.4$\pm$0.003
VAR-Lasso	0.012$\pm$0.0002	0.51$\pm$0.006	0.025$\pm$0.0002	0.15$\pm$0.002	-	3.1$\pm$0.004
GARCH	0.023$\pm$0.000	0.88$\pm$0.002	0.19$\pm$0.001	0.37$\pm$0.0016	-	-
KVAE	0.014$\pm$0.002	0.34$\pm$0.025	0.051$\pm$0.019	0.1$\pm$0.005	-	0.095$\pm$0.012
Vec-LSTM ind-scaling	0.008$\pm$0.001	0.391$\pm$0.017	0.025$\pm$0.001	0.087$\pm$0.041	0.506$\pm$0.005	0.133$\pm$0.002
Vec-LSTM lowrank-Copula	0.007$\pm$0.000	0.319$\pm$0.011	0.064$\pm$0.008	0.103$\pm$0.006	0.326$\pm$0.007	0.241$\pm$0.033
GP scaling	0.009$\pm$0.000	0.368$\pm$0.012	0.022$\pm$0.000	0.079$\pm$0.000	0.183$\pm$0.395	1.483$\pm$1.034
GP Copula	0.007$\pm$0.000	0.337$\pm$0.024	0.0245$\pm$0.002	0.078$\pm$0.002	0.208$\pm$0.183	0.086$\pm$0.004
Transformer MAF	**0.005**$\pm$0.003	0.301$\pm$0.014	0.0207$\pm$0.000	0.056$\pm$0.001	0.179$\pm$0.002	0.063$\pm$0.003
TimeGrad	0.006$\pm$0.001	**0.287**$\pm$0.02	**0.0206**$\pm$0.001	**0.044**$\pm$0.006	**0.114**$\pm$0.02	**0.0485**$\pm$0.002

Thank you
&
See you at the poster!

Github: `zalandoresearch/pytorch-ts`