Multivariate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows

Kashif Rasul[1], Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann and Roland Vollgraf

Zalando Research @ ICLR 2021

[1]Email: kashif.rasul@zalando.de

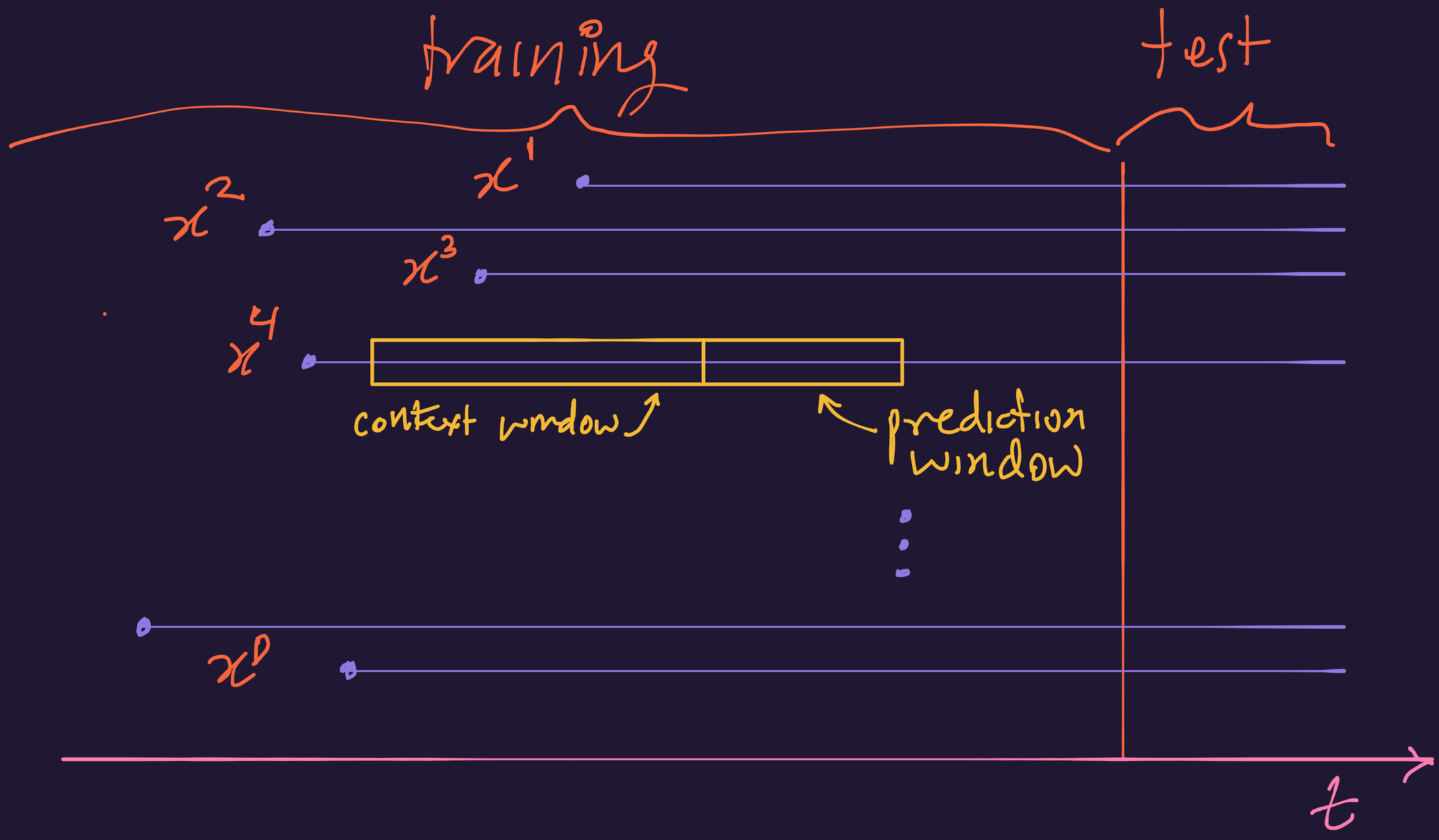
Introduction

- Time series forecasting ubiquitous in academia and industry
- Classical methods do not scale with large data
- Deep learning methods offer a popular alternative

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.

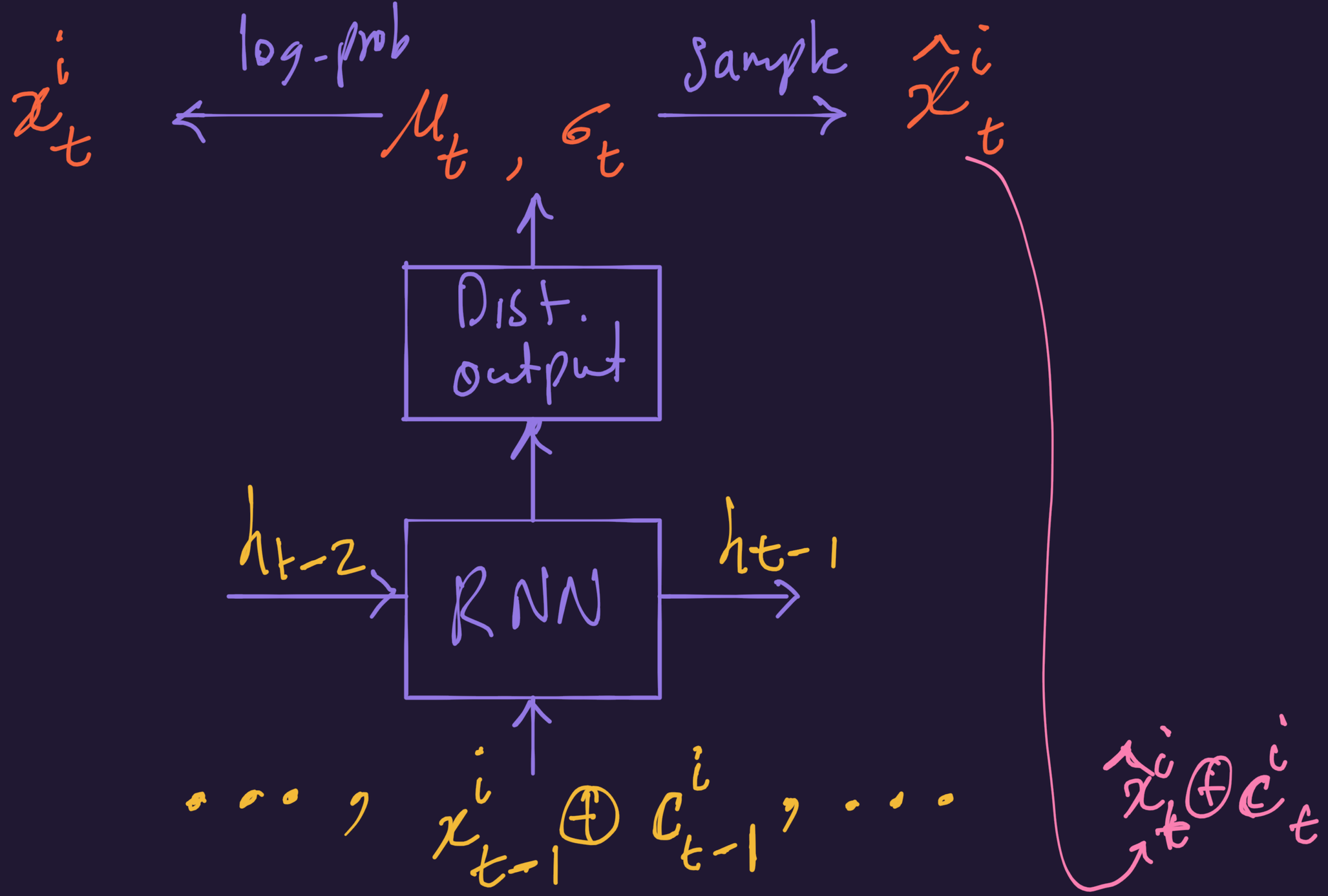
This work proposes a general deep learning based probabilistic multivariate time series forecasting model.


training
test
x^1
x^2
x^3
x^4
x^D
context window
prediction window
t

Univariate Problem

- $x_t^i \in \mathbb{R}$ where $i \in \{1, \ldots, D\}$ and t time index
- Context window $[1, t_0)$ and prediction window $[t_0, T]$
- Covariates for each series at t given by vector $\mathbf{c}_t^i \in \mathbb{R}^C$
- Univariate probablistic problem:

$$p_{\mathcal{X}}(x_{t_0:T}^i | x_{1:t_0-1}^i, \mathbf{c}_{1:T}^i) \approx \Pi_{t=t_0}^T p_\theta(x_t^i | x_{1:t-1}^i, \mathbf{c}_{1:T}^i)$$

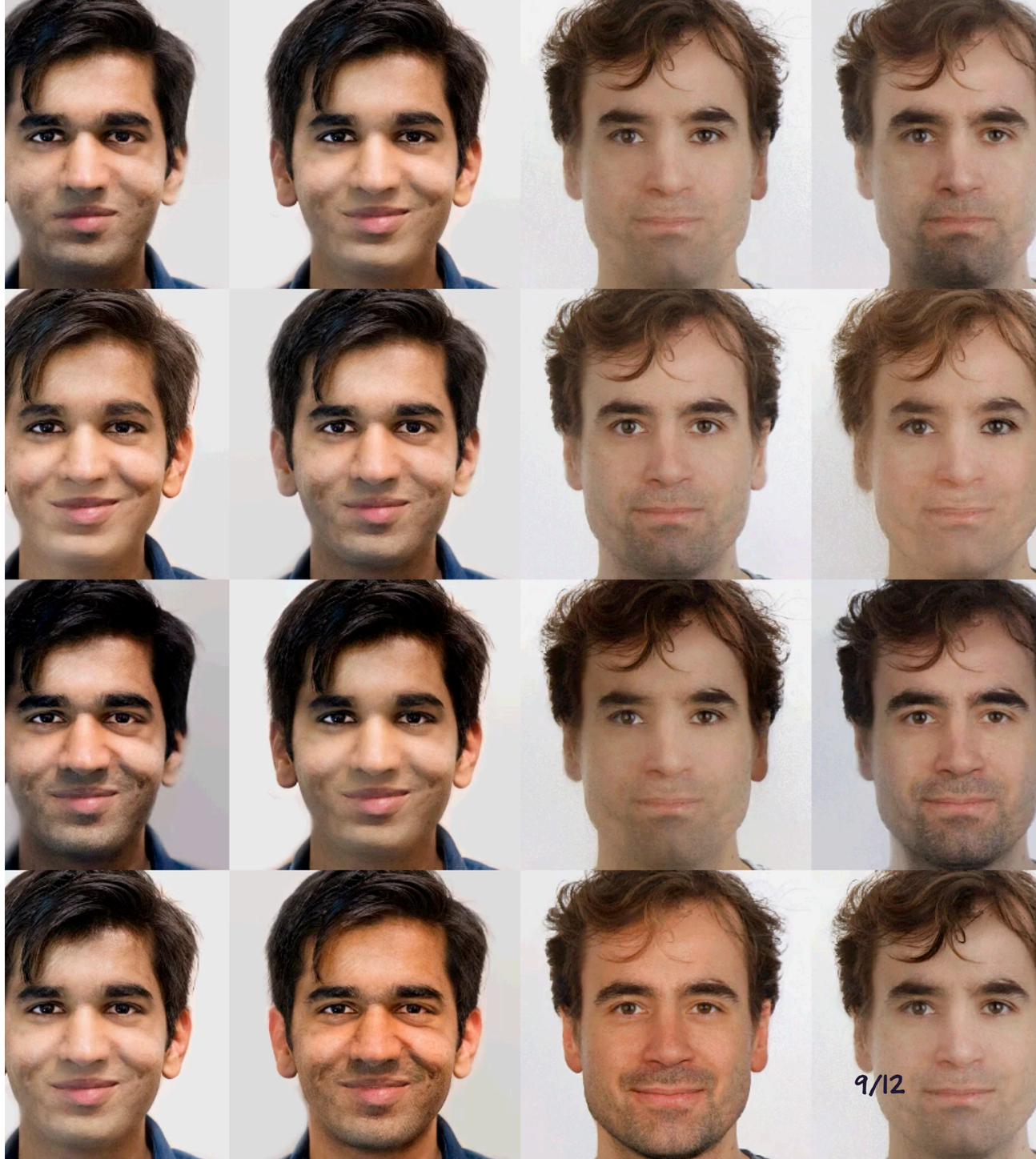

log-prob
Sample
Dist. output
RNN
x_t^i
μ_t, σ_t
$\hat{x}_t^i$
h_{t-2}
h_{t-1}
$x_{t-1}^i \oplus c_{t-1}^i$
$\hat{x}_t^i \oplus c_t^i$

Multivariate Forecast?

- Can use the same model with vector valued input $\mathbf{x}_t \in \mathbb{R}^D$
- Covariates now associated with the whole vector rather than individual entities
- Need a full joint distribution model for each t
- E.g. multivariate Gaussian but:
  - increase parameters by $O(D^2)$
  - log-prob will be $O(D^3)$

Normalizing Flows (Rezende & Mohamed, 2015)

- NF are a sequence of invertible mappings from $\mathbb{R}^D \mapsto \mathbb{R}^D$
- Idea: transform data distribution to some simple distribution and max. that log-prob or sample from simple and go backwards
- Has two properties:
  1. inverse is easy to evaluate
  2. Jacobian determinant is $O(D)$

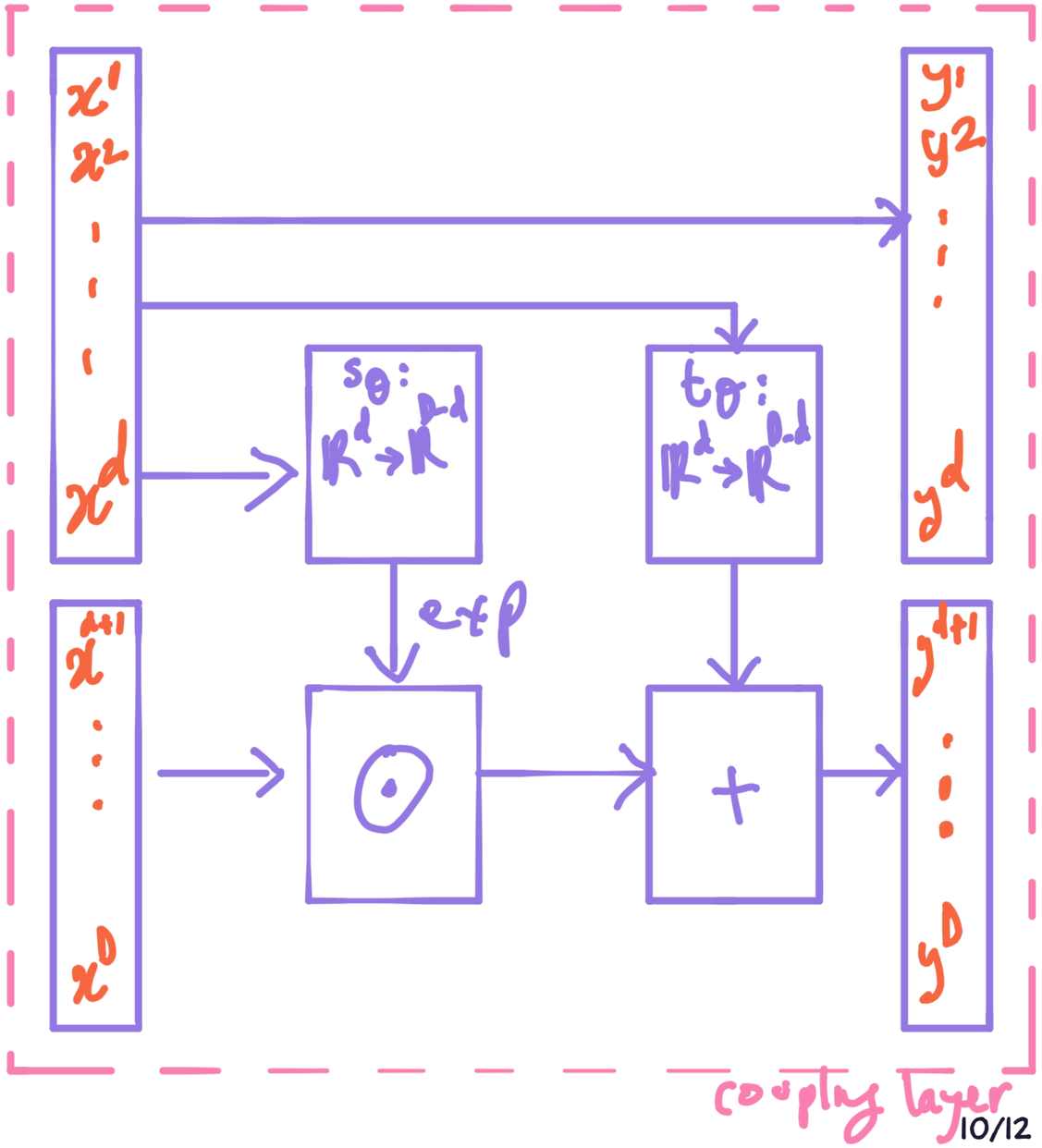


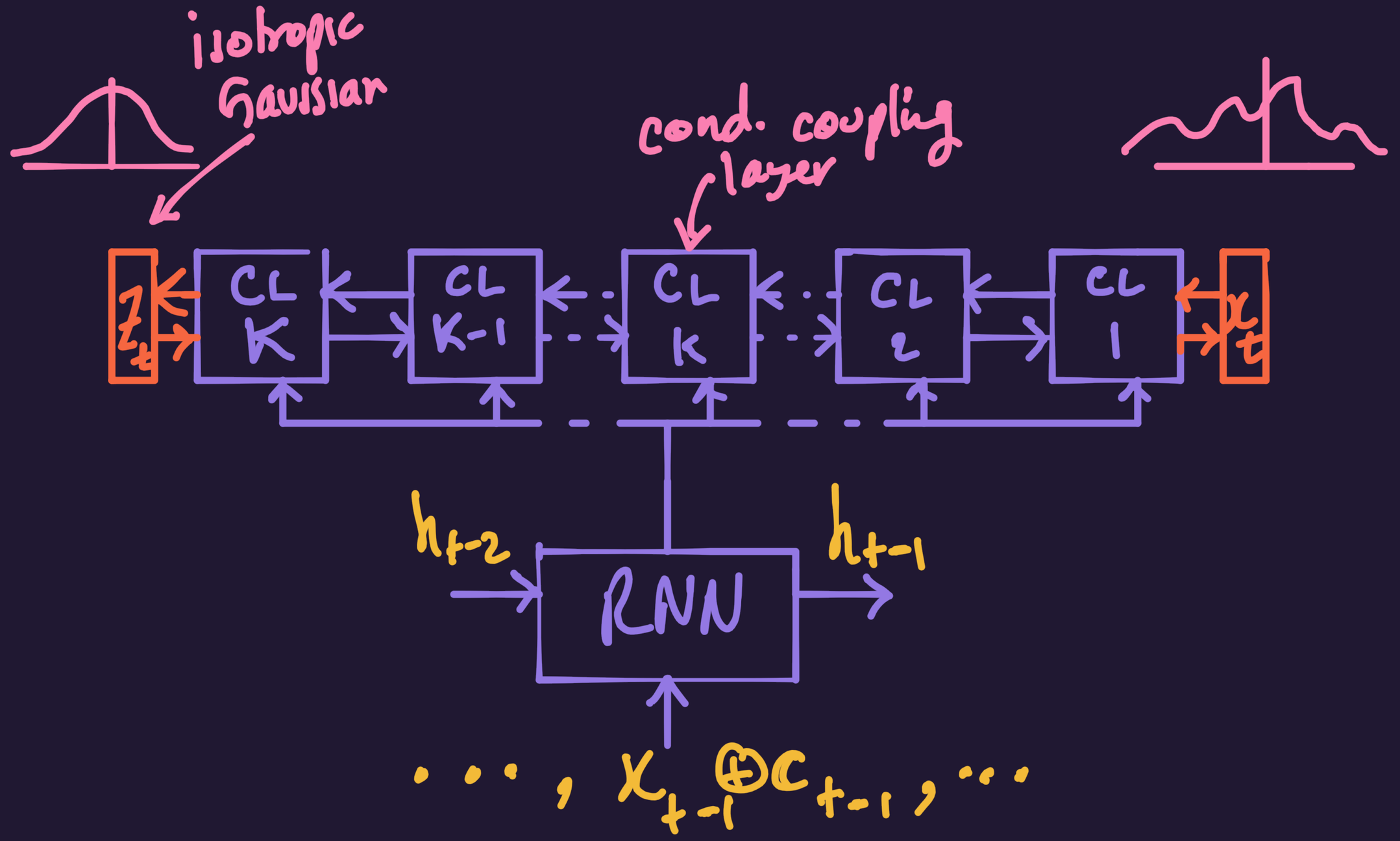
Real NVP (Dinh et al., 2017)

- Introduces a bijection: Coupling Layer
- For some d and two neural networks $s_\theta : \mathbb{R}^d \to \mathbb{R}^{D-d}$ and $t_\theta : \mathbb{R}^d \to \mathbb{R}^{D-d}$ we have:

$$\begin{cases} \mathbf{y}^{1:d} = \mathbf{x}^{1:d} \\ \mathbf{y}^{d+1:D} = \mathbf{x}^{d+1:D} \odot \exp(s_\theta(\mathbf{x}^{1:d})) + t_\theta(\mathbf{x}^{1:d}) \end{cases}$$

- Jacobian is block triangular: det. is sum over diagonal
- **Conditioning:** concat vectors to inputs of s_θ and t_θ




isotropic Gaussian
cond. coupling layer
CL K
CL K-1
CL k
CL 2
CL 1
RNN
h_{t-2}
h_{t-1}
$x_{t-1} \oplus c_{t-1}$

Summary

- Experiments on 6 real-world datasets: establish SOTA results (with max. $D = 2,000$)
- Extensive comparison against competitive modern probablistic multivariate methods
- Can replace RNN with self-attention based Transformer (Vaswani et al., 2017) module
- PyTorch implementation Github: `zalandoresearch/pytorch-ts`