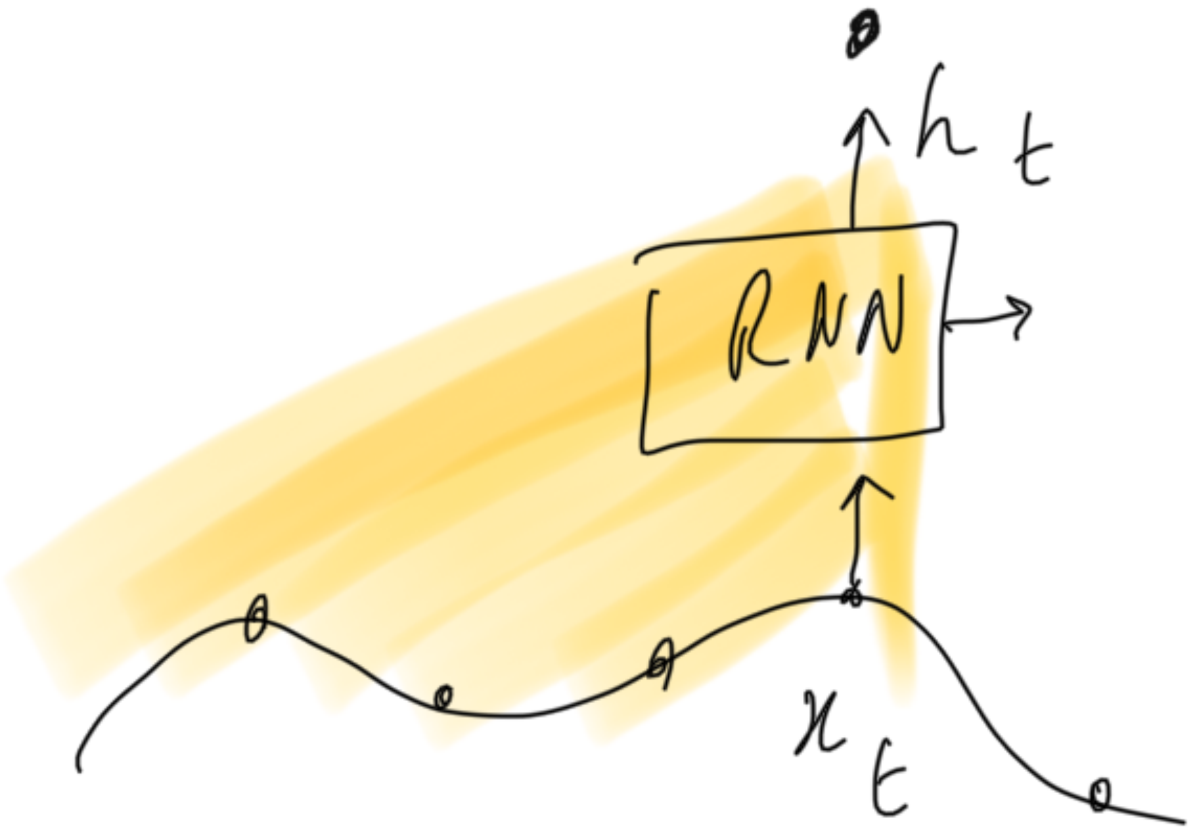
Transformers for Time Series Forecasting

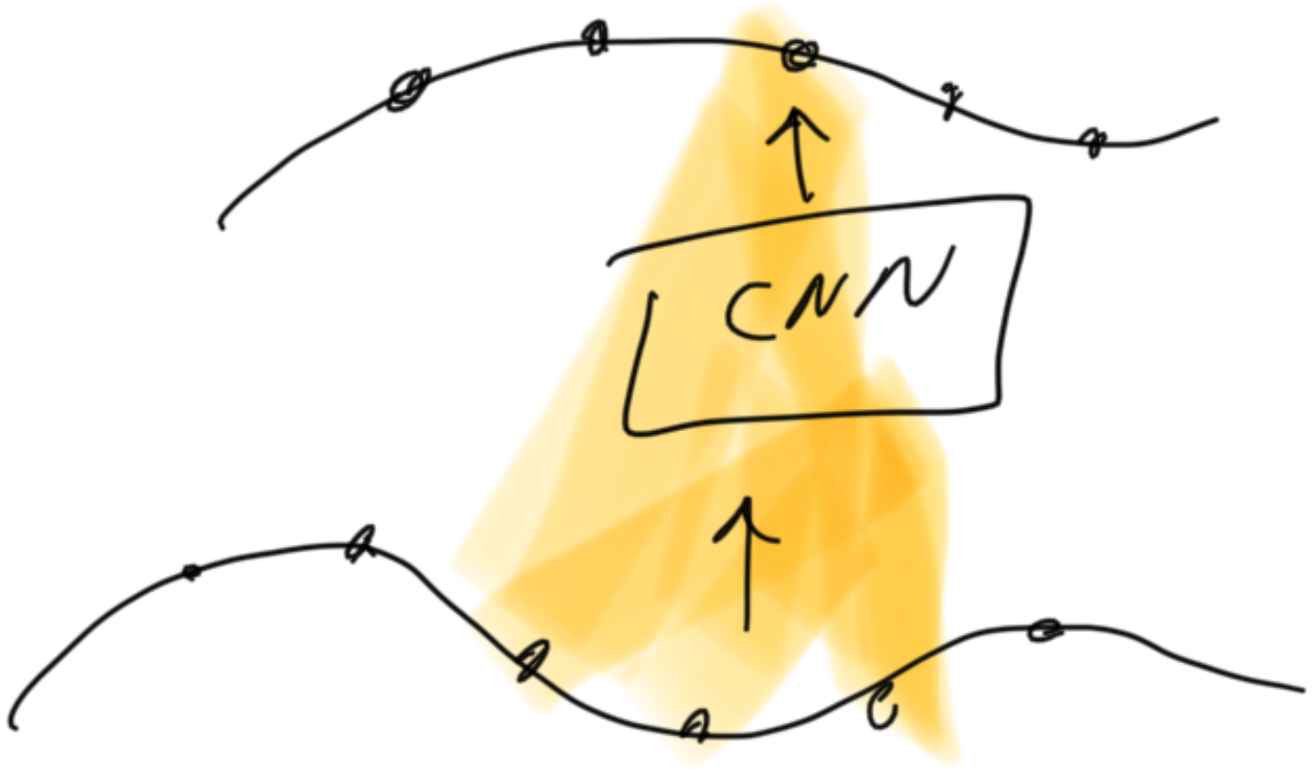
Kashif Rasul (PhD)

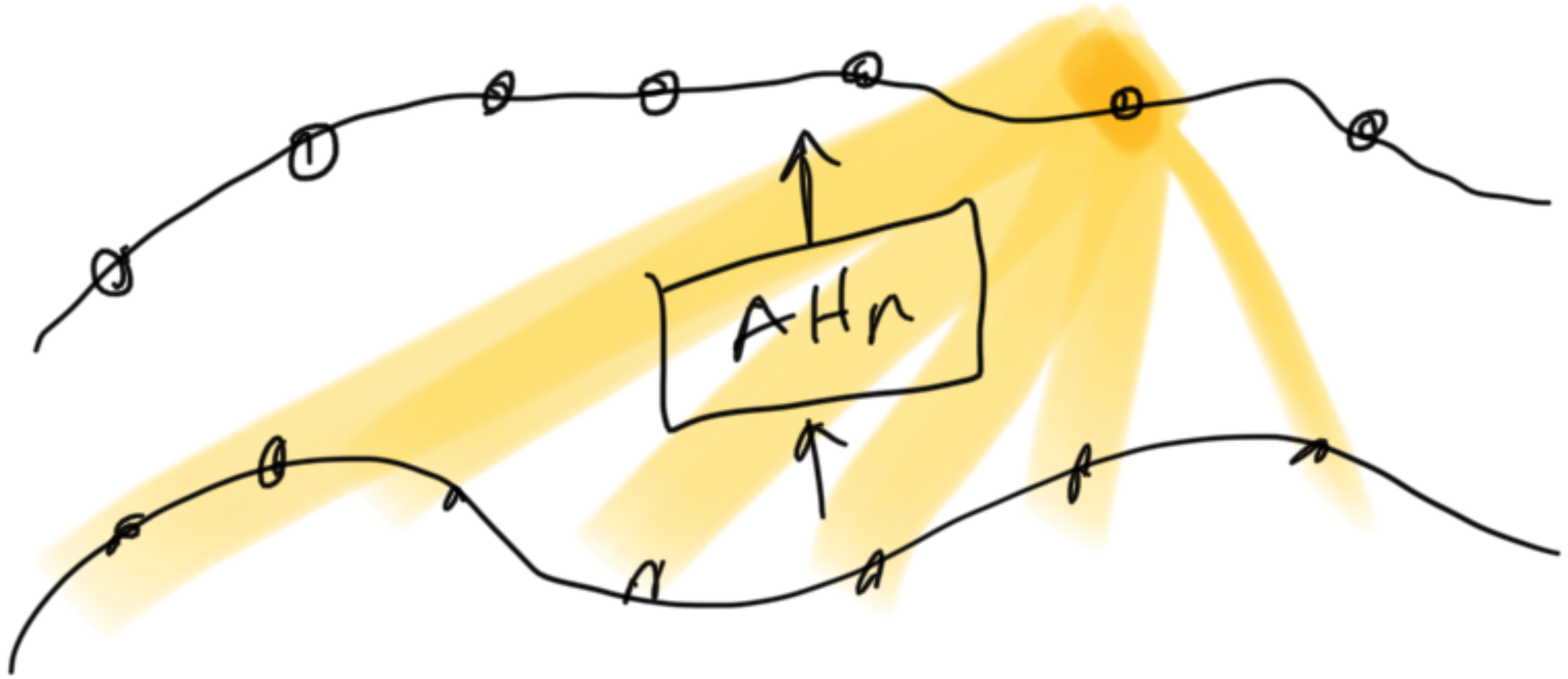
ISF: Deep Learning for Forecasting
25.06.2023

Neural Forecasting Models

1. Some aspect that learns a representation over history:
   - RNN
   - MLP, CNN
   - Attention/Transformer 👈🏻
   - GNNs etc.
2. Some aspect that models the emissions appropriately:
   - Point emissions
   - Probabilistic etc.


h_t
RNN
x_t


CNN


Attn

$$Y = \{\bar{vec}_1, \bar{vec}_2, \ldots \bar{vec}_N\}$$

$$\uparrow$$

$$\boxed{\text{Attn}}$$

$$\uparrow$$

$$X = \{vec_1, vec_2, \ldots vec_N\}$$

$vec_i \in \mathbb{R}^D$

$\bar{vec}_i \in \mathbb{R}^D$

$$X = \{\vec{x}_1, \vec{x}_2 \cdots, \vec{x}_N\} \in \mathbb{R}^{N \times D}$$

$$E = \frac{X \cdot X^T}{\sqrt{(D)}} \in \mathbb{R}^{N \times N}$$

Similarity score

scaling →

$$A = \text{softmax}(E, dim=1) \in \mathbb{R}^{N \times N}$$

prob. weights

$$Y = A \cdot X \in \mathbb{R}^{N \times D}$$

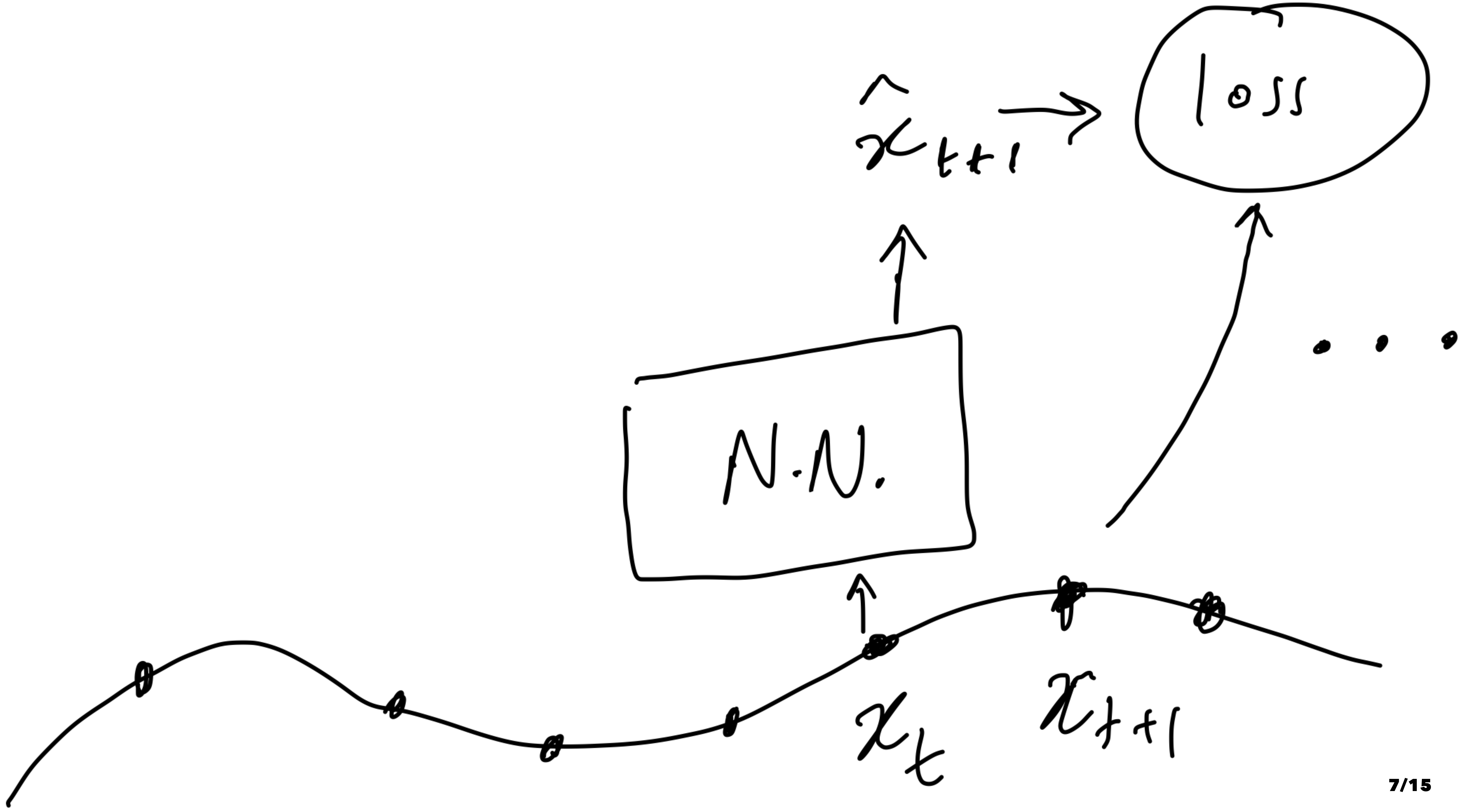
$$X = \{\vec{x}_1, \dots, \vec{x}_N\} \in \mathbb{R}^{N \times D}$$

Learnable: $Q_\theta(\vec{x}_i)$; $K_\theta(\vec{x}_i)$; $V_\theta(\vec{x}_i)$: $\mathbb{R}^D \to \mathbb{R}^D$

$$E = \frac{Q(X) \cdot K(X)^T}{\sqrt{(D)}}$$

$$A = \text{softmax}(E; \text{dim}=1)$$

$$Y = A \cdot V(X) \in \mathbb{R}^{N \times D}$$


loss
$\hat{x}_{t+1}$
N.N.
x_t
x_{t+1}
...

$$Y = \{\vec{vec}_1, \vec{vec}_2, \dots, \vec{vec}_i, \dots, \vec{vec}_N\}$$

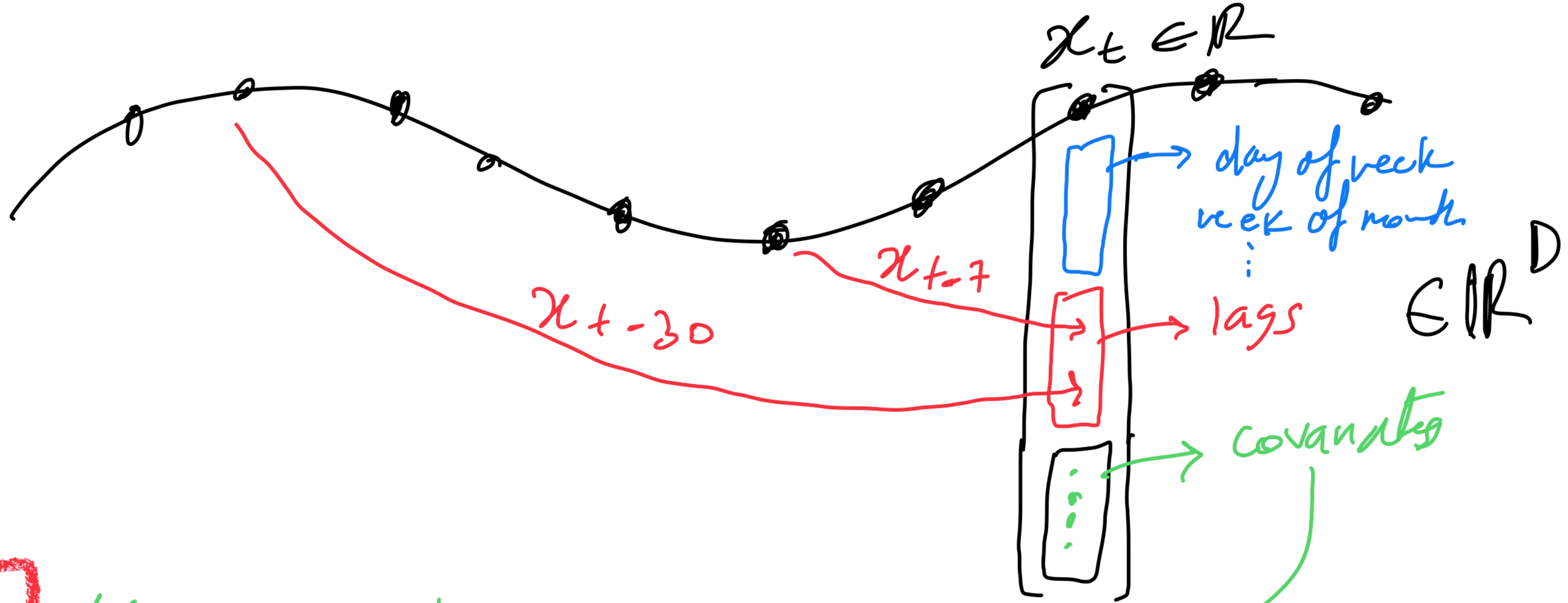
Masked (causal) attention

$$M = \begin{bmatrix} & -\infty & -\infty & -\infty \\ & & -\infty & -\infty \\ & 1 & & -\infty \\ \vdots & & & \end{bmatrix} \in \mathbb{R}^{N \times N}$$

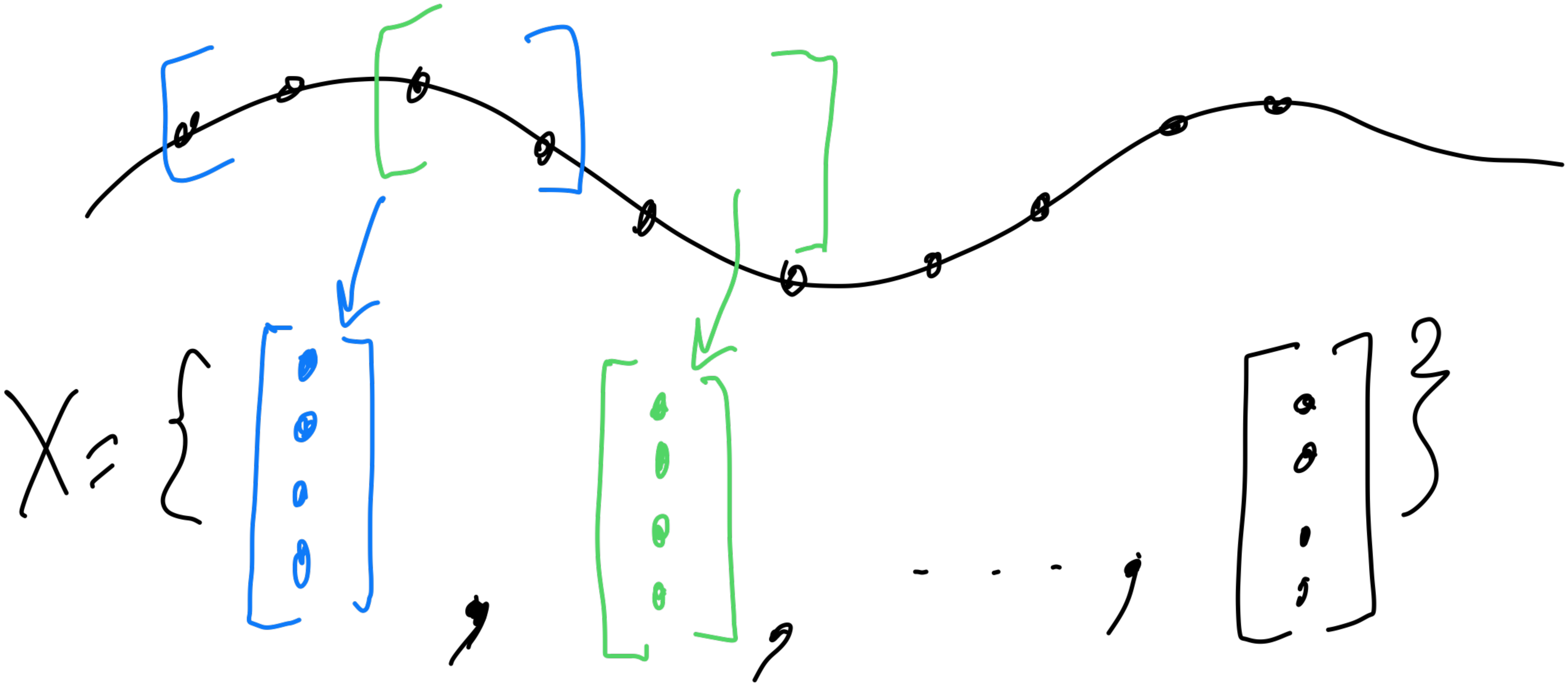
$$A = \mathrm{softmax}(E \odot M, d=1)$$

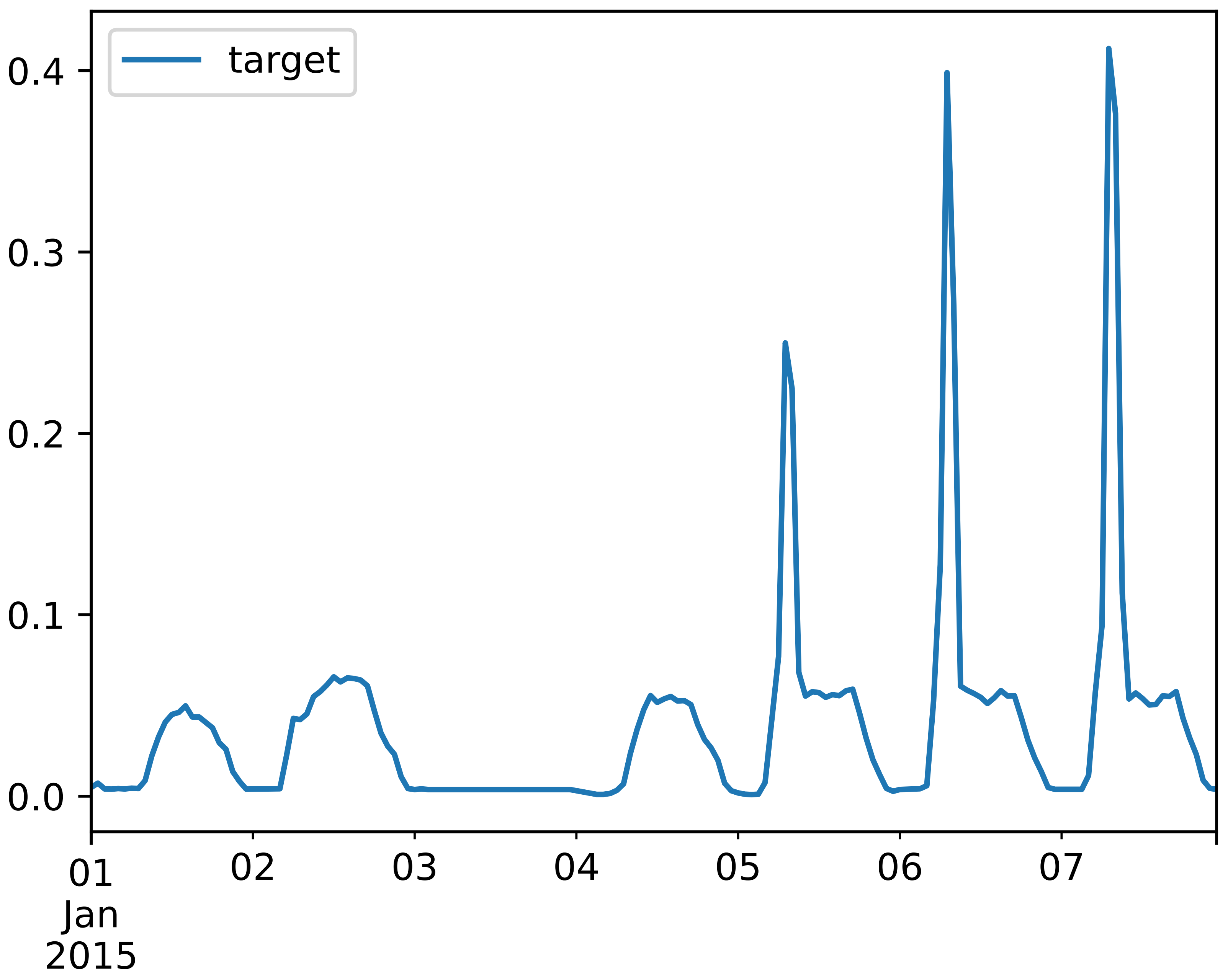
$$Y = A \cdot V$$

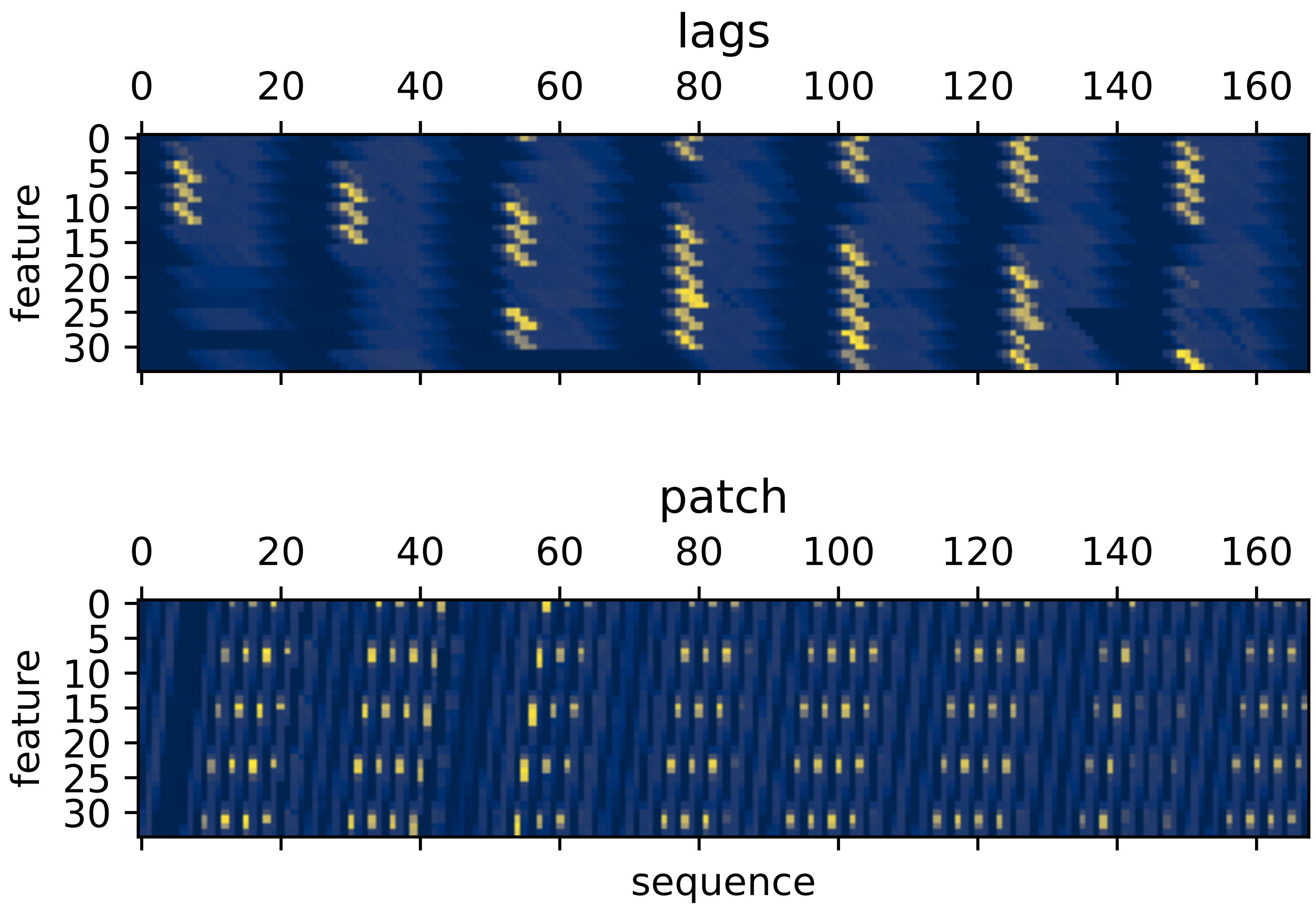
$$X = \{vec_1, vec_2, \dots, vec_i, \dots, vec_N\}$$

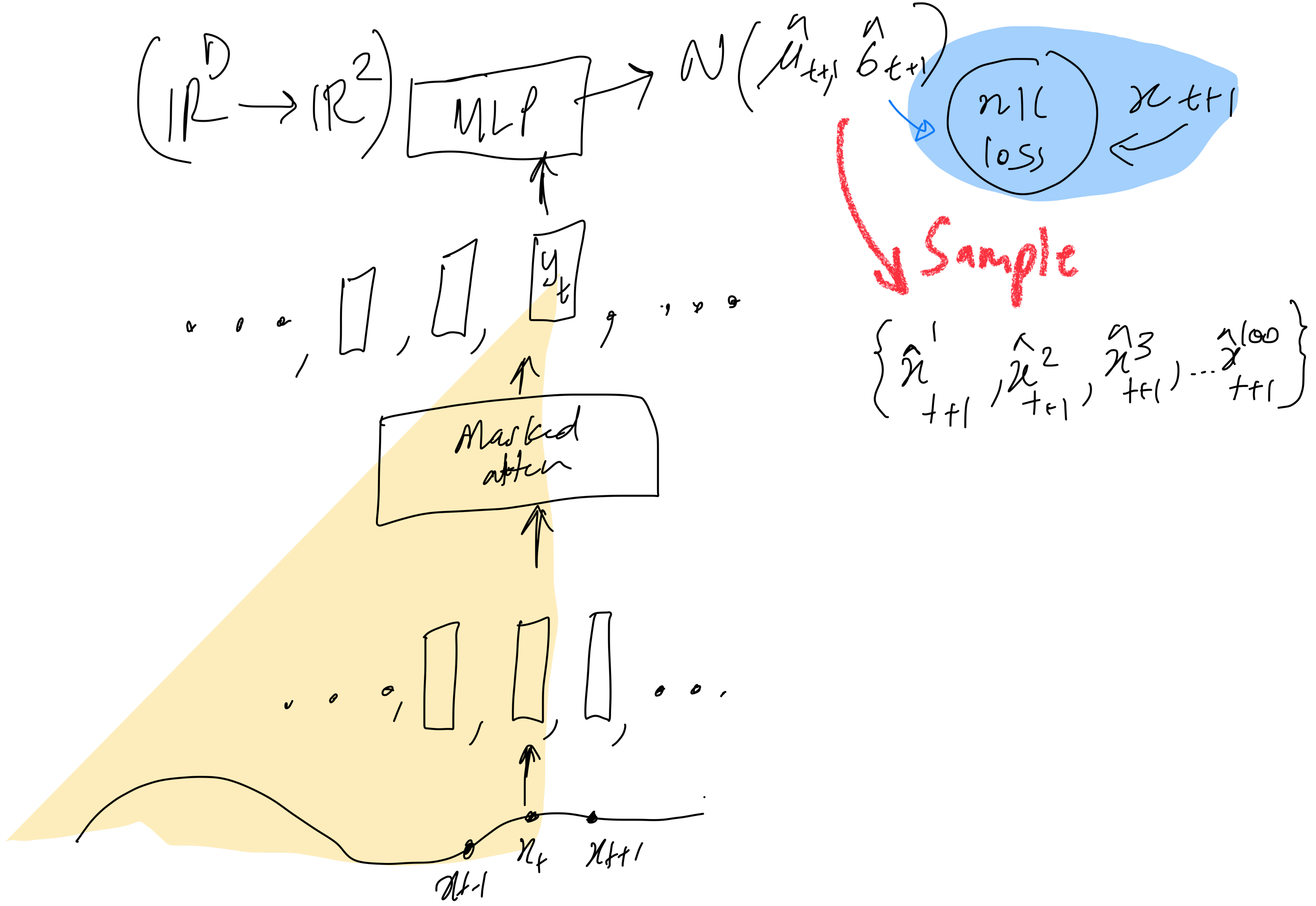


	real	cat.
static	μ, σ	item_id
dynamic	weather, price	?, ?

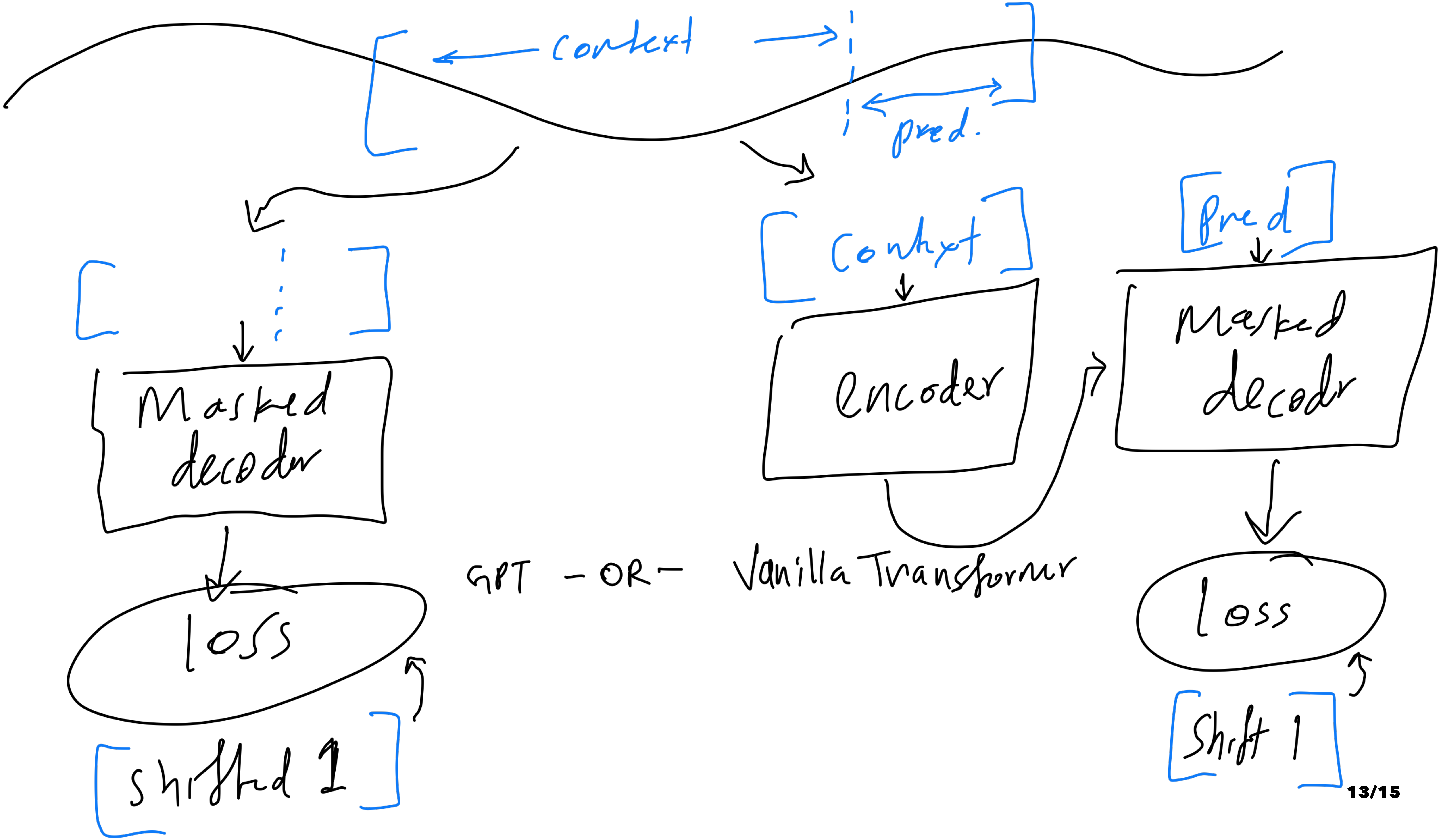

X =


target
0.4
0.3
0.2
0.1
0.0
01
Jan
2015
02
03
04
05
06
07


lags
patch
feature
sequence
0
20
40
60
80
100
120
140
160
0
5
10
15
20
25
30



$(\mathbb{R}^D \to \mathbb{R}^2)$
MLP
$\mathcal{N}(\hat{\mu}_{t+1}, \hat{\sigma}_{t+1})$
nll loss
x_{t+1}
Sample
$\{\hat{x}^1_{t+1}, \hat{x}^2_{t+1}, \hat{x}^3_{t+1}, \dots \hat{x}^{100}_{t+1}\}$
y_t
Masked atten
x_{t-1}
x_t
x_{t+1}


context
pred.
Masked decoder
loss
shifted 1
GPT -OR- Vanilla Transformer
context
encoder
pred
Masked decoder
loss
Shift 1

Pros/Cons

- 👍🏼 Can model temporal dependencies efficiently (number of parameters) without forgetting
- 👍🏼 Can handle nans naturally (similar to causal masking)
- 😿 Compute and memory is quadratic in number of input sequence size
- 😿 Auto-regressive inference can be restrictive for larger models due to need of many samples

Summary

- Transformers provide excellent inductive bias for forecasting
- Temporal covariates naturally serve as positional encoding
- Can potentially condition on any point in the past context window
- Fast to train and allows for auto-regressive sampling
- Naturally incorporate nans/missing data
- Code on Github: kashif/pytorch-transformer-ts