


Transformers for Time Series Forecasting

Kashif Rasul (PhD)

ISF: Deep Learning for Forecasting
25.06.2023

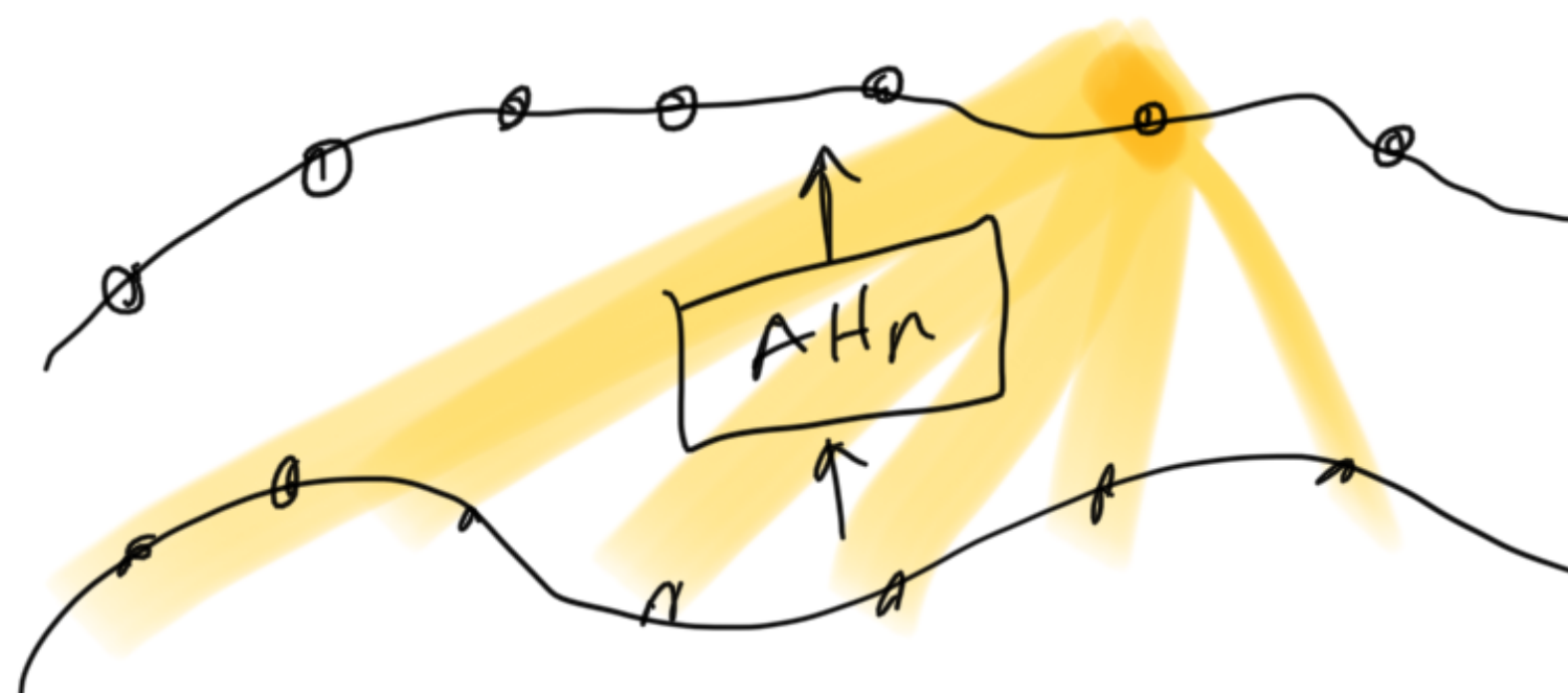
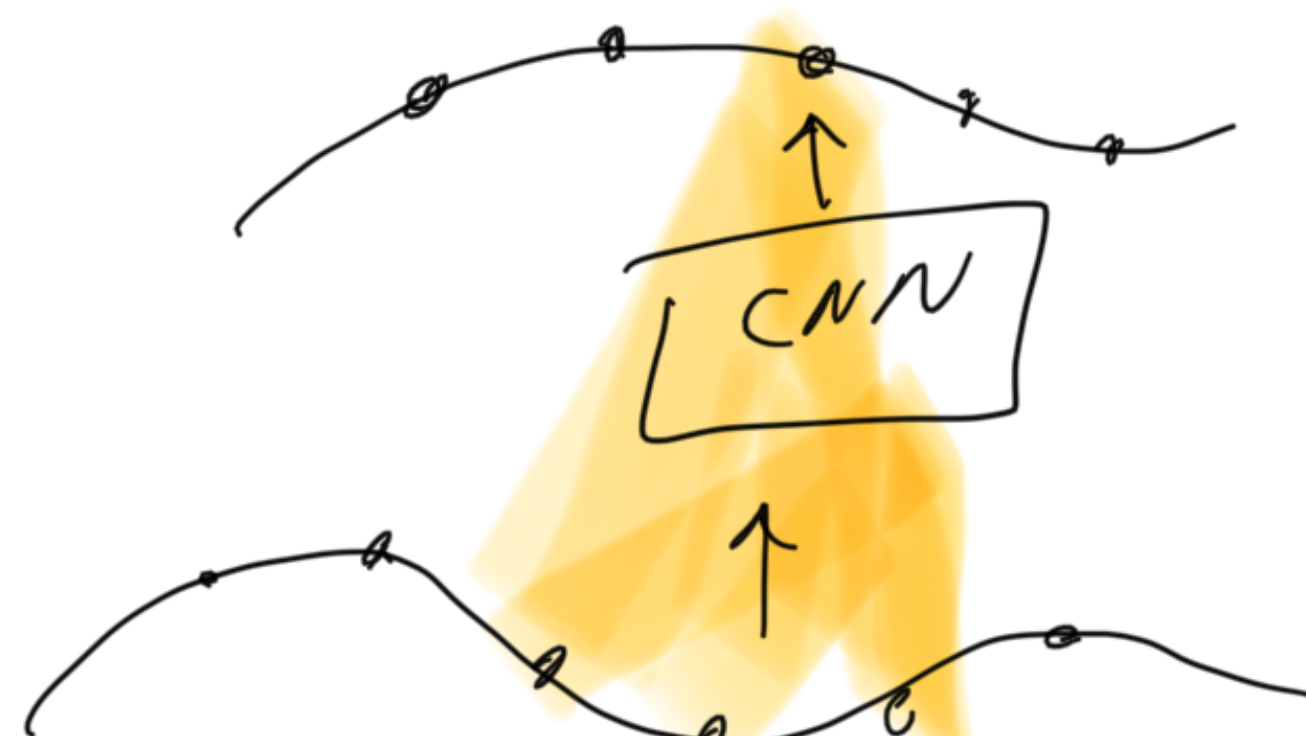
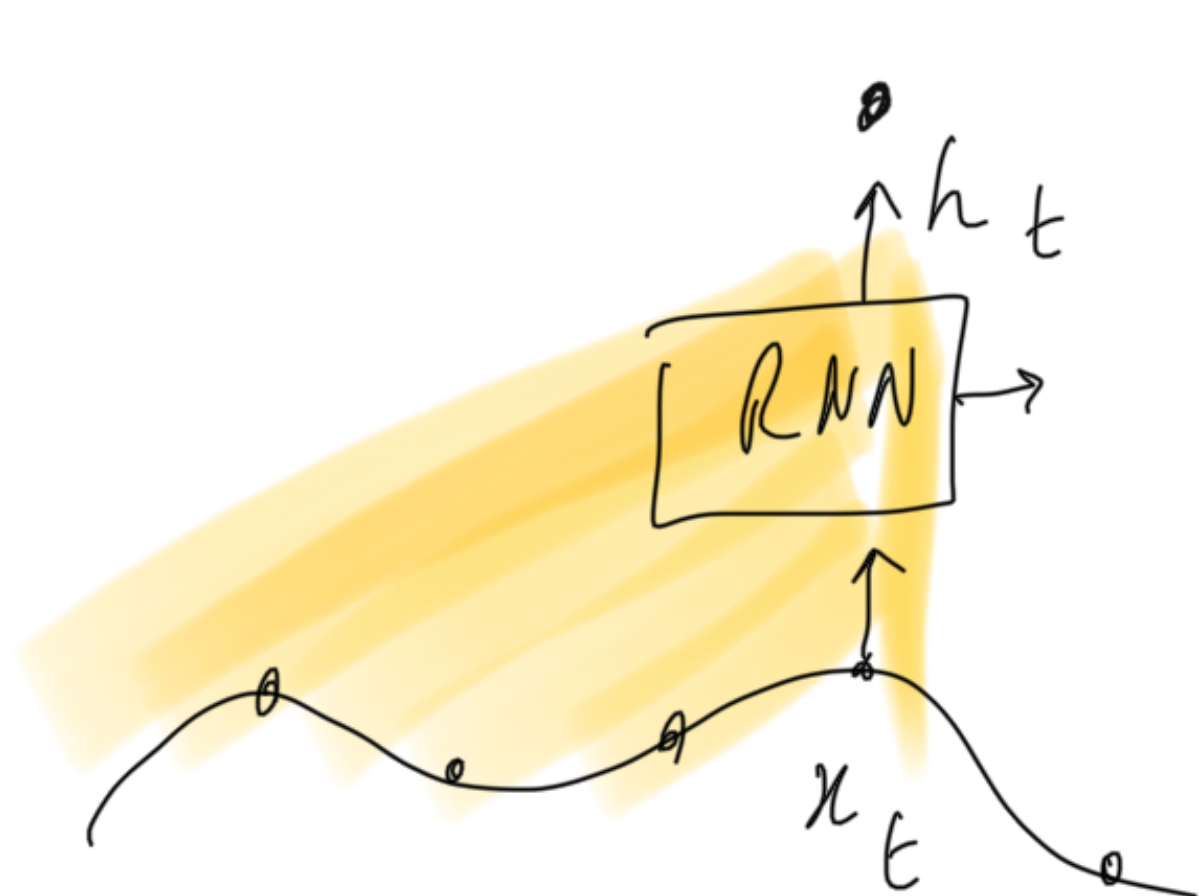
Neural Forecasting Models

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.



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

Att n

$$\begin{aligned} vec_i &\in \mathbb{R}^D \\ \vec{vec}_i &\in \mathbb{R}^D \end{aligned}$$

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

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

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

scaling \rightarrow

Similarity score

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

prob. weights

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

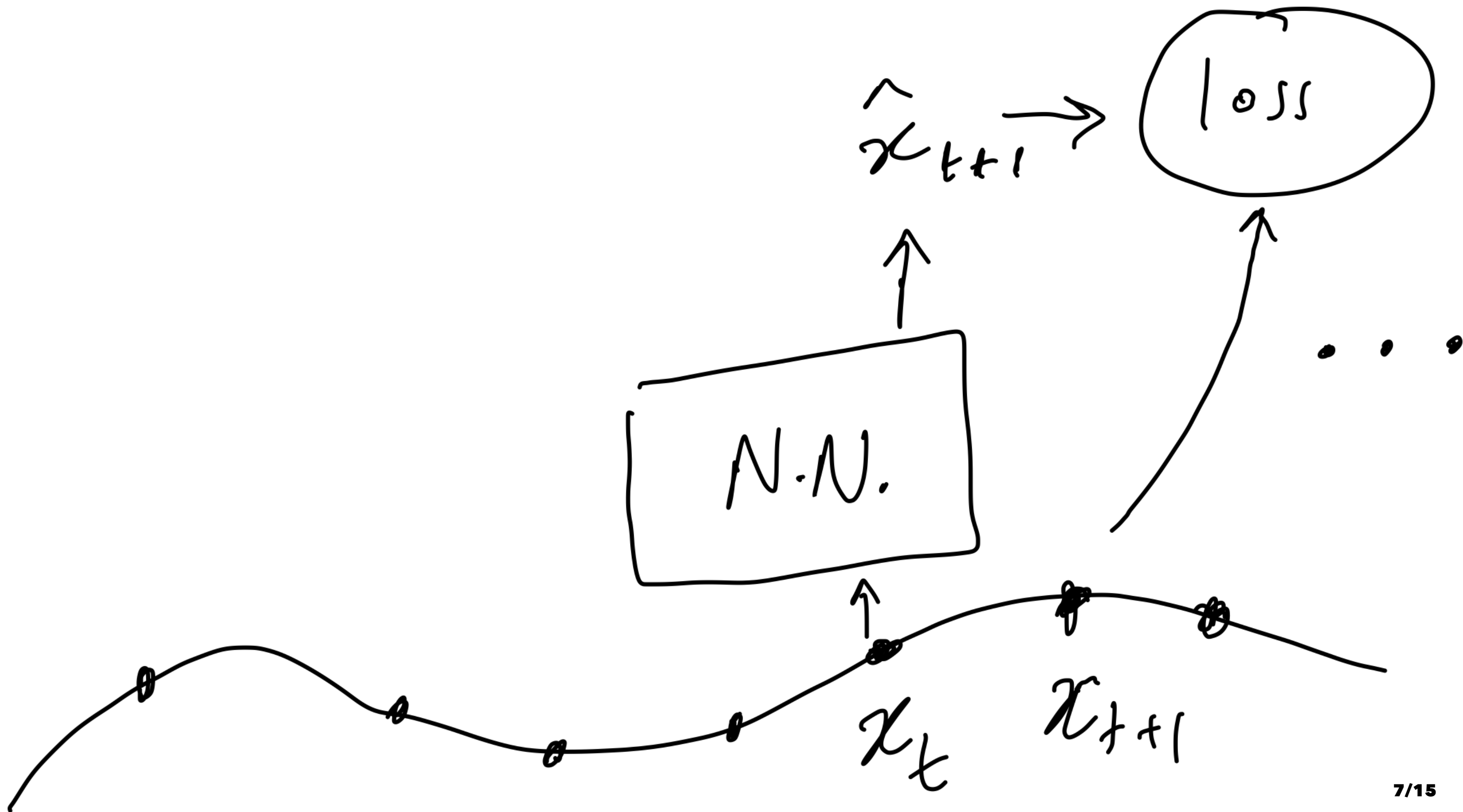
$$X = \{\vec{x}_1, \dots, \vec{x}_N\} \in \mathbb{R}^{N \times D}$$

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

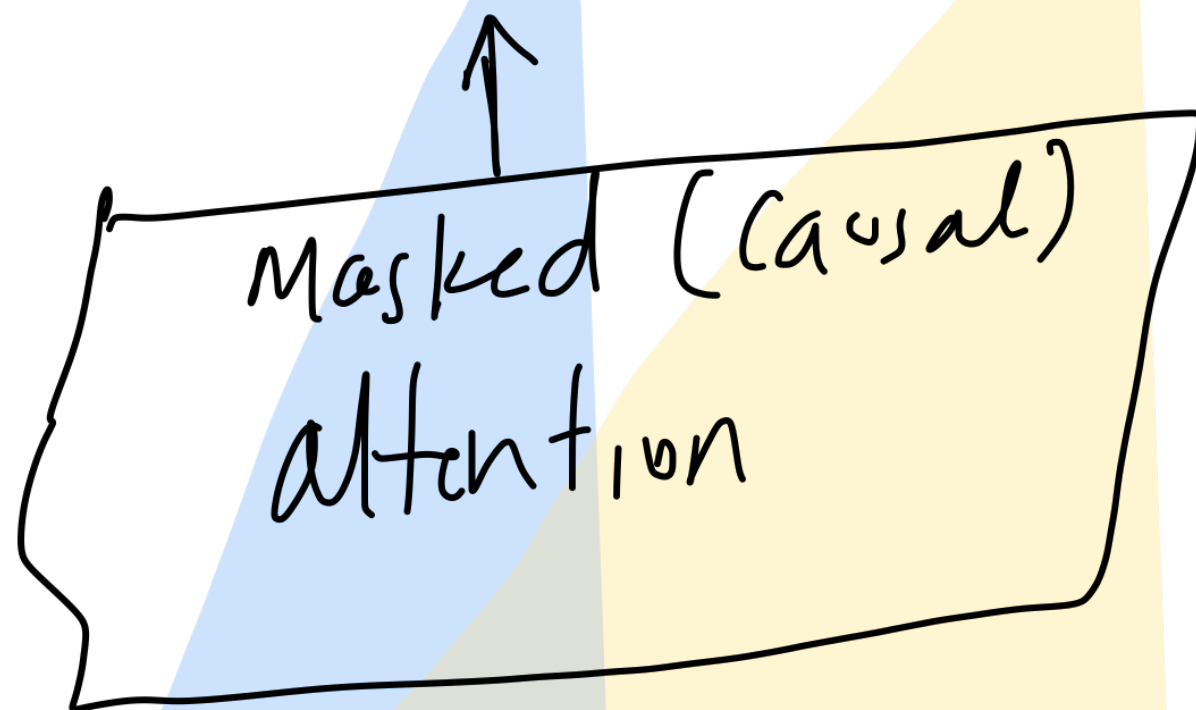
$$E = \frac{Q(X) \cdot K(X)^T}{\sqrt{D}}$$

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

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



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

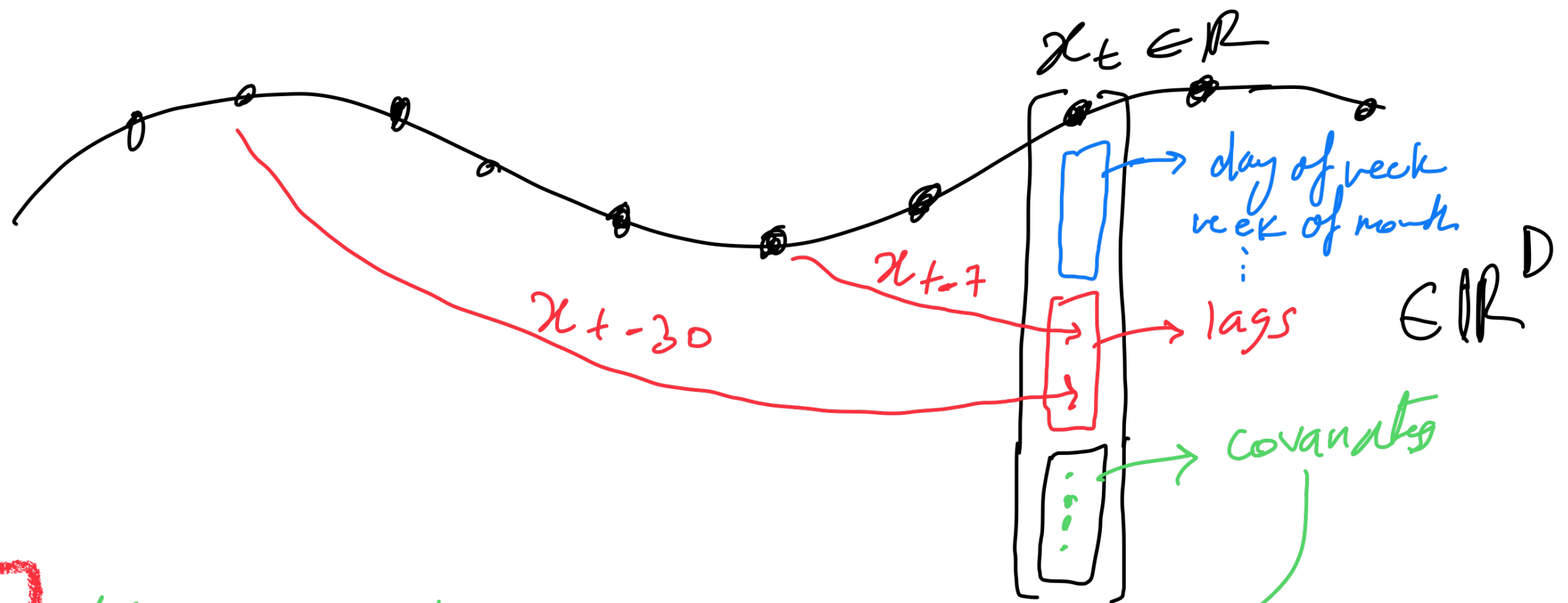


$$X = \{\vec{vec}_1, \vec{vec}_2, \dots, \vec{vec}_N\}$$

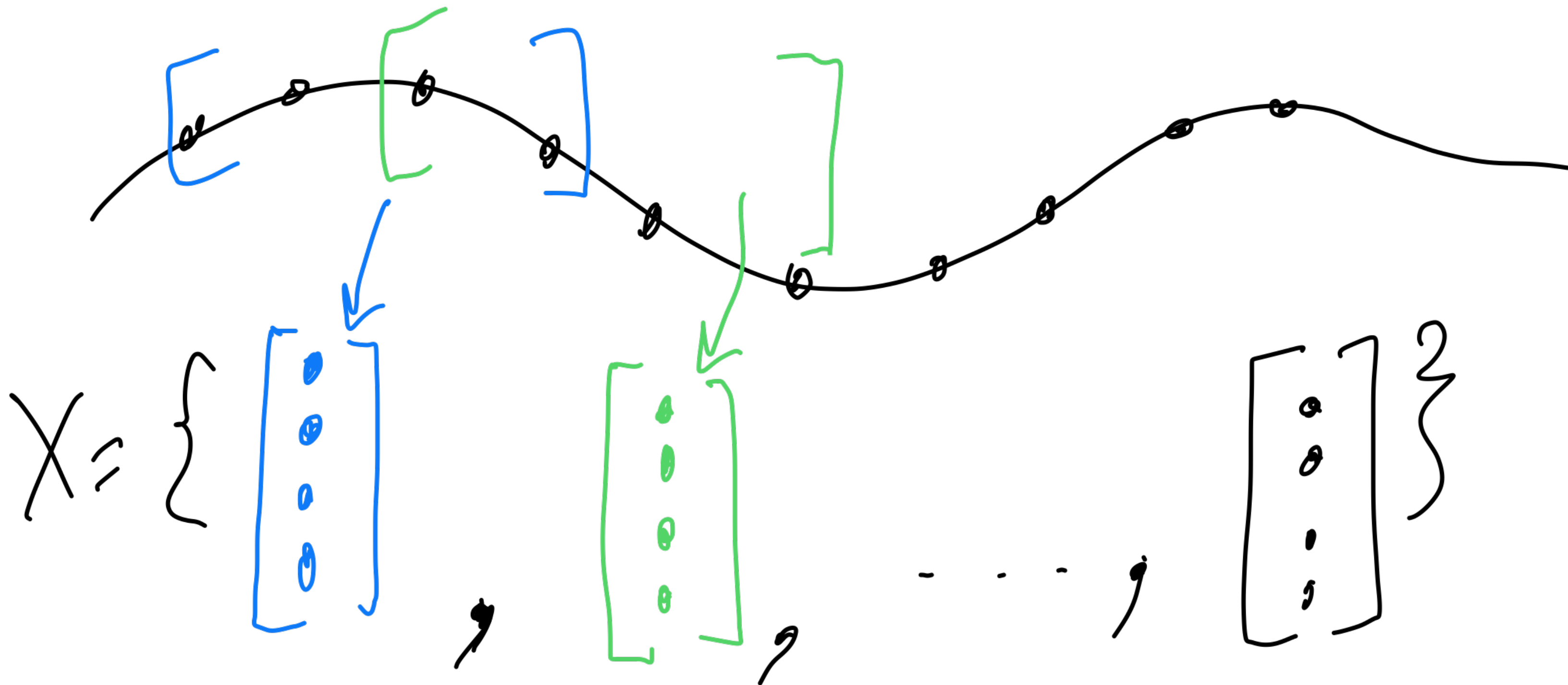
$$M = \begin{bmatrix} -\infty & -\infty & -\infty \\ -\infty & -\infty & \\ -\infty & & 1 \\ \vdots & & & \ddots \end{bmatrix} \in \mathbb{R}^{N \times N}$$

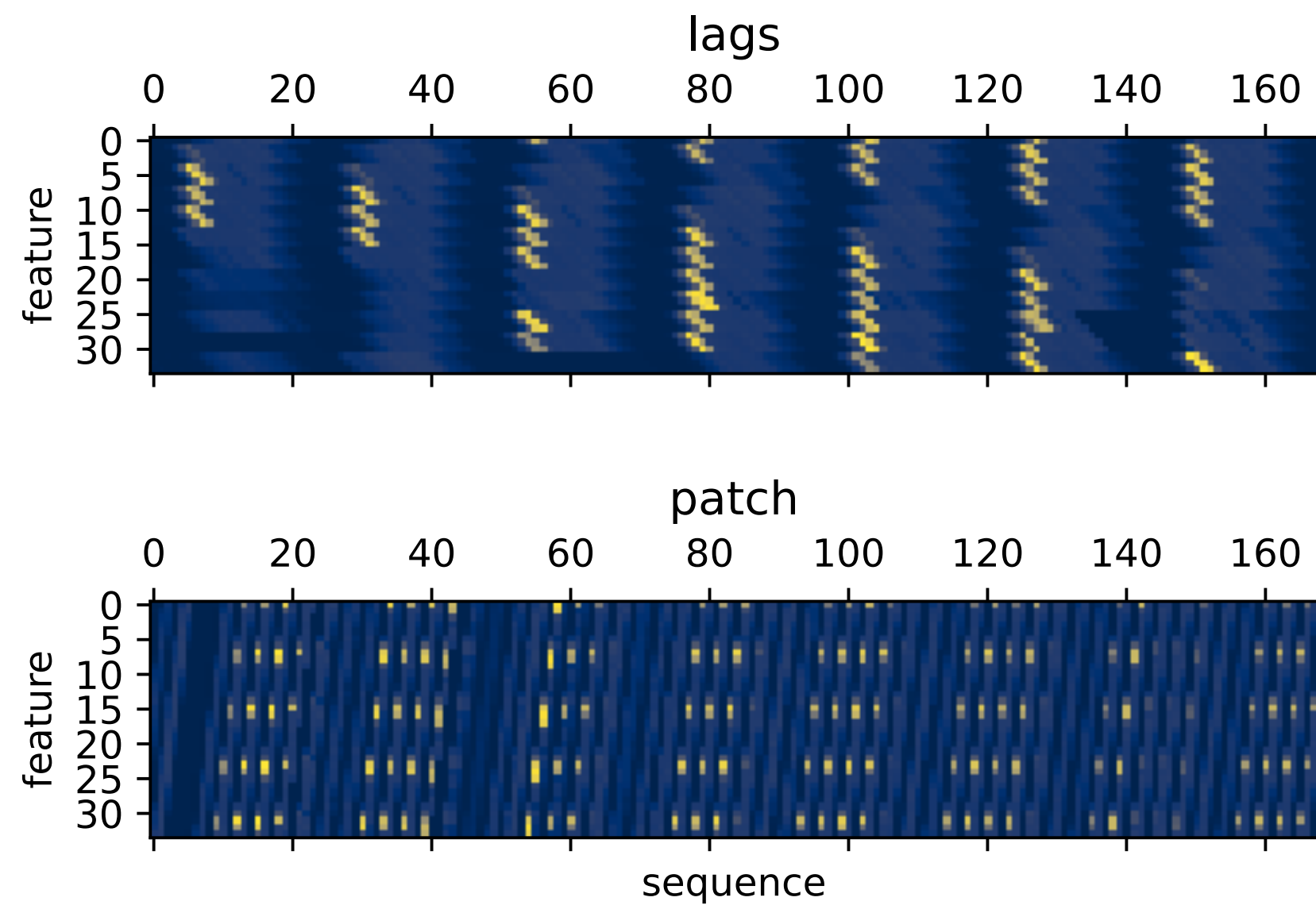
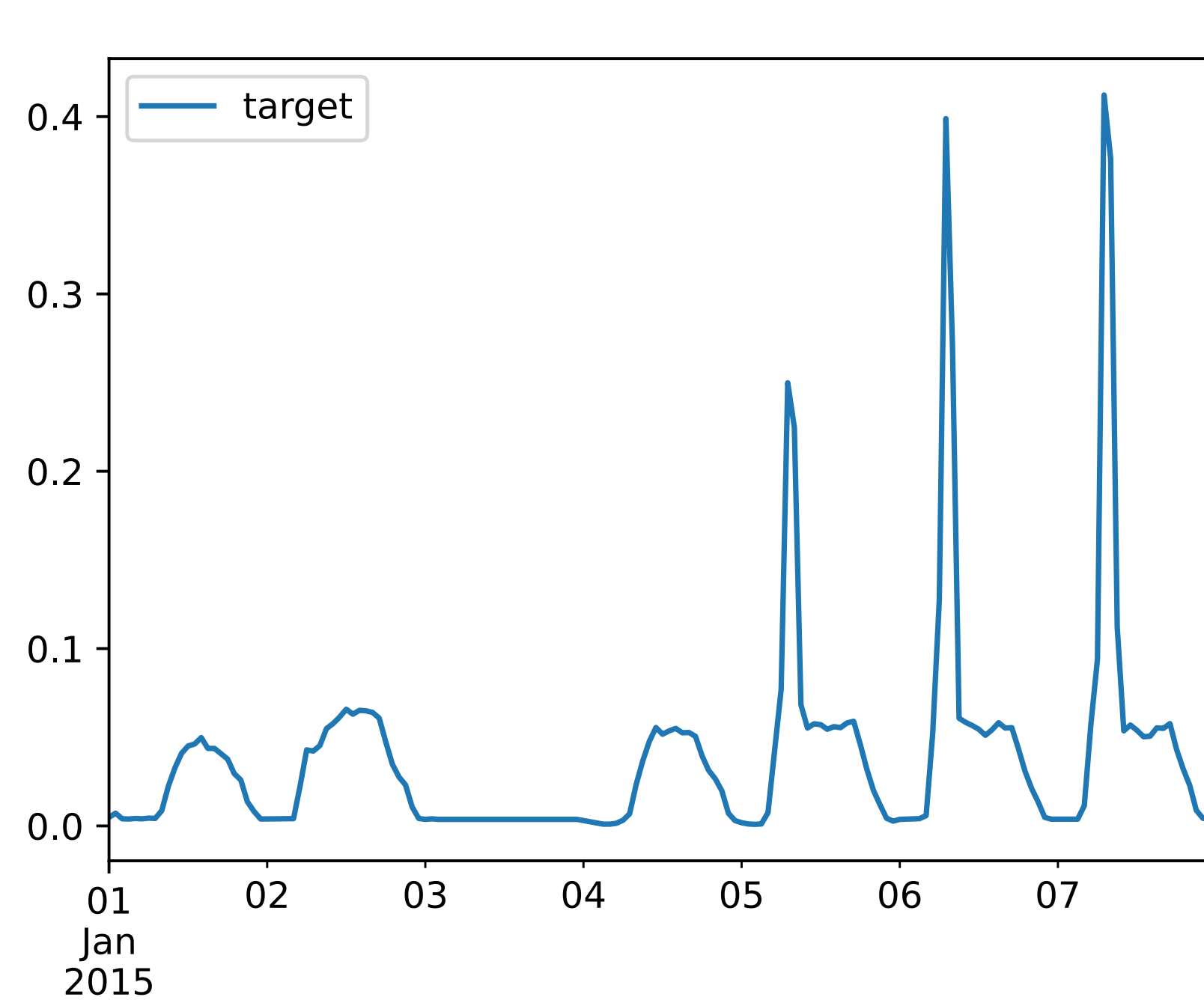
$$A = \text{softmax}(E \cdot M, d=1)$$

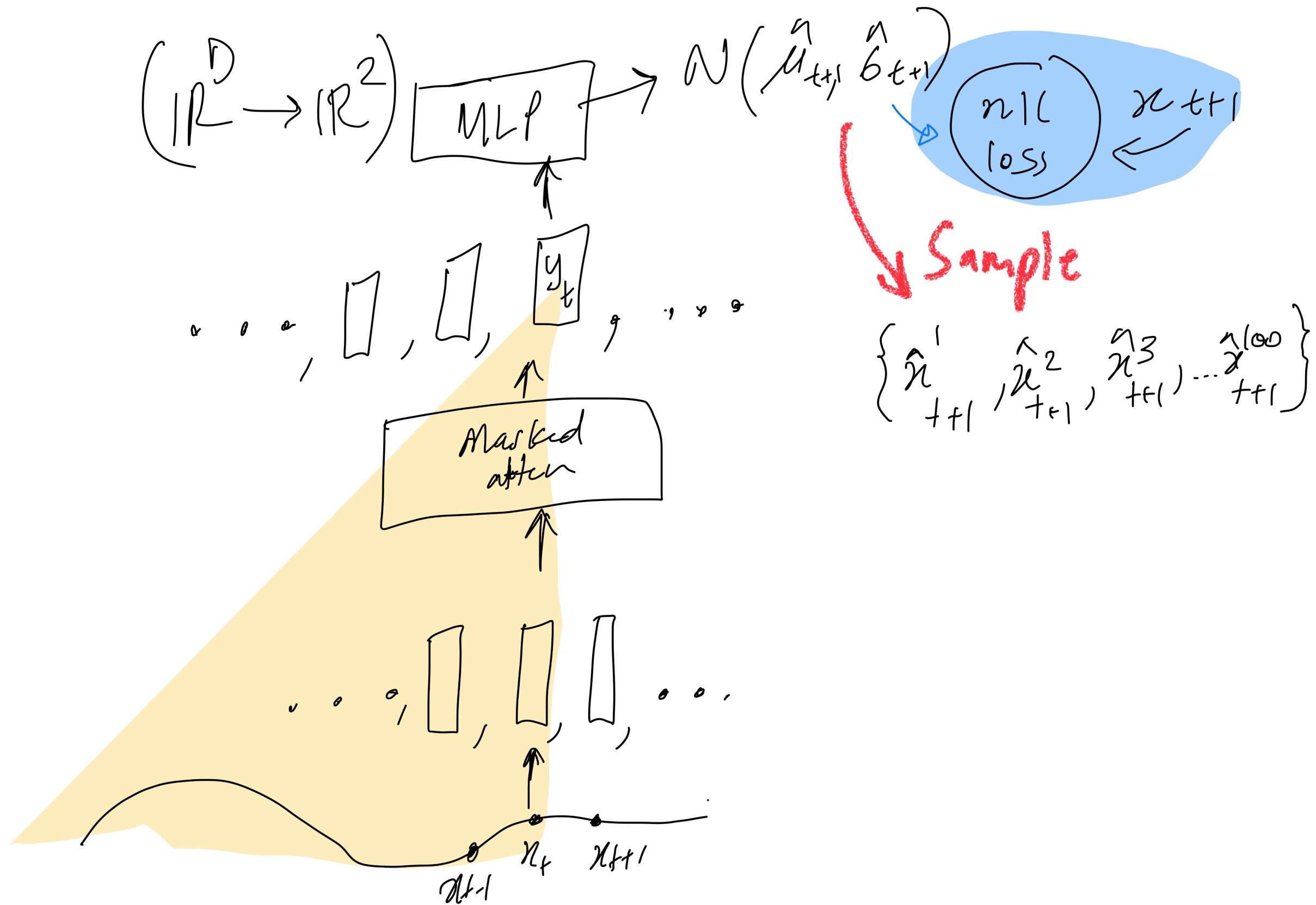
$$Y = A \cdot X$$

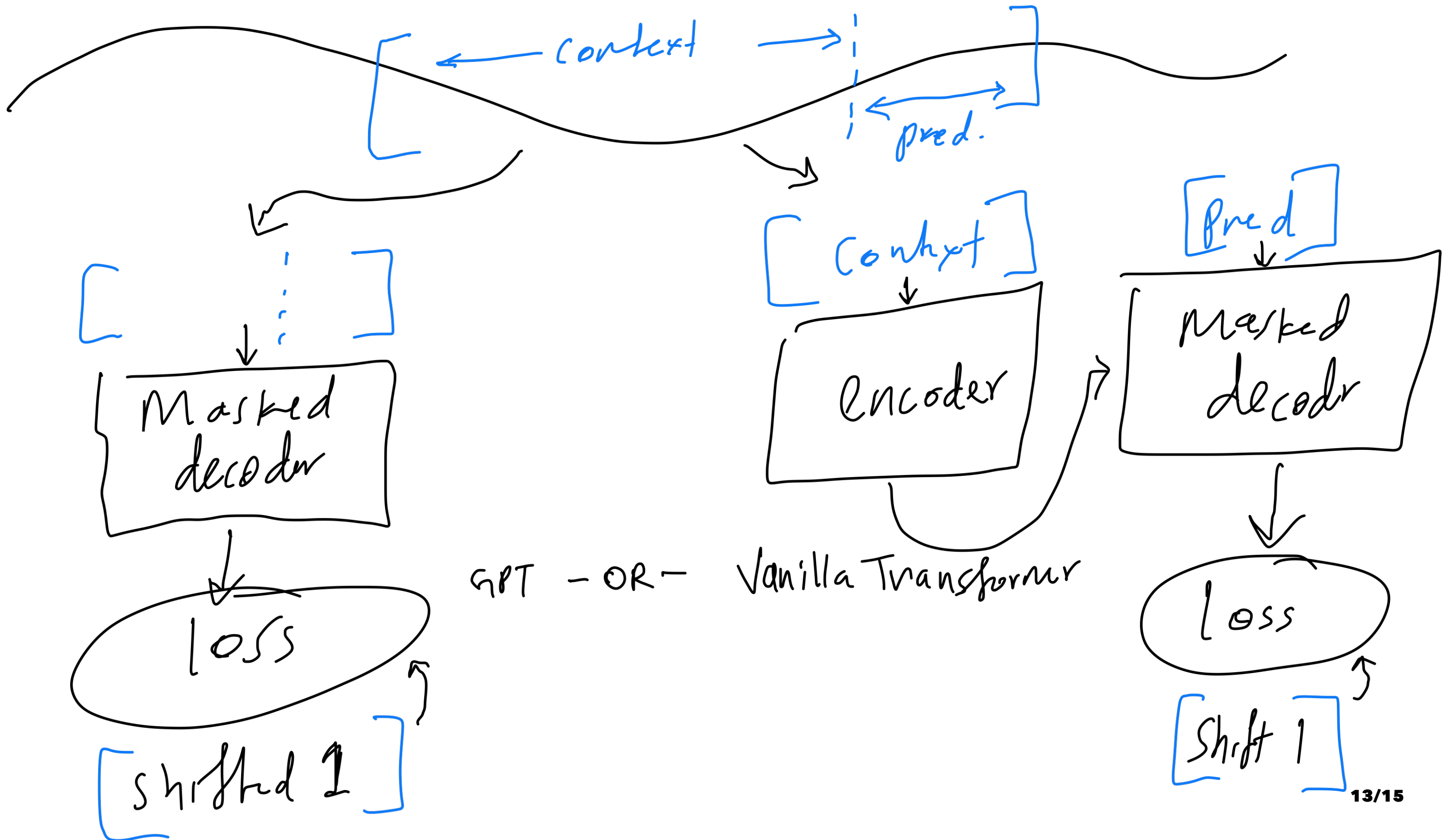


Static	μ, σ	item-id
dynamic	weather, price	?, ?
		real
		cat.









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](https://github.com/kashif/pytorch-transformer-ts)