《多模态大模型》

Lecture 7
新一代AI架构

王广润
中山大学
人机物智能融合实验室（HCP Lab）
wanggrun@gmail.com

本节内容

CONTENTS

Marvin Minsky & Seymour Papert, Perceptrons,1969

- **Minsky and Papert mathematically proved that a single-layer perceptron could only solve linearly separable problems, they completely failed at the XOR problem.**
- **"And we suspect that the extension to multi-layer systems will be similarly sterile."**
- **The Negative Impact: The "AI Winter"**
  - ✓Minsky was a towering figure in the AI community, and Perceptrons was viewed as a definitive, rigorous mathematical takedown of neural networks.
  - ✓ Government agencies (like DARPA) and universities almost completely stopped funding neural network research

Weierstrass Approximation Theorem

假设 f 是定义在闭区间 $[a,b]$ 上的连续实值函数。那么，对于任意给定的正数 $\epsilon > 0$，都存在一个多项式 $P(x)$，使得对于区间 $[a,b]$ 中的所有 x，都满足：

$$|f(x) - P(x)| < \epsilon$$

$$\lim_{n\to\infty} \max_{x\in[a,b]} |f(x) - P_n(x)| = 0$$

$$B_n(f)(x) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right) \binom{n}{k} x^k (1-x)^{n-k}$$

$$P_{n,k}(x)$$

Weierstrass Approximation Theorem

1. **概率和为1**：$\sum_{k=0}^{n} P_{n,k}(x) = 1$
2. **期望值**：$\sum_{k=0}^{n} \frac{k}{n} P_{n,k}(x) = x$
3. **方差相关**：$\sum_{k=0}^{n} \left(\frac{k}{n} - x\right)^2 P_{n,k}(x) = \frac{x(1-x)}{n} \leq \frac{1}{4n}$ （因为 $x(1-x)$ 在 $x = 1/2$ 时取得最大值 $1/4$）

1. **一致连续性**：对于任意给定的 $\epsilon > 0$，存在一个 $\delta > 0$，使得只要 $|x - y| < \delta$，就有 $|f(x) - f(y)| < \frac{\epsilon}{2}$。
2. **有界性**：存在一个常数 $M > 0$，使得对于所有的 $x \in [0, 1]$，都有 $|f(x)| \leq M$。这意味着任意两点间的函数值之差 $|f(x) - f(y)| \leq 2M$。

Weierstrass Approximation Theorem

$$|B_n(f)(x) - f(x)| = \left| \sum_{k=0}^{n} \left[f\left(\frac{k}{n}\right) - f(x) \right] P_{n,k}(x) \right| \leq \sum_{k=0}^{n} \left| f\left(\frac{k}{n}\right) - f(x) \right| P_{n,k}(x)$$

$$|B_n(f)(x) - f(x)| \leq \sum_{\text{近}} \left| f\left(\frac{k}{n}\right) - f(x) \right| P_{n,k}(x) + \sum_{\text{远}} \left| f\left(\frac{k}{n}\right) - f(x) \right| P_{n,k}(x)$$

$$\left|\tfrac{k}{n} - x\right| < \delta \qquad \left|\tfrac{k}{n} - x\right| \geq \delta$$

$$\sum_{\text{近}} \left| f\left(\frac{k}{n}\right) - f(x) \right| P_{n,k}(x) < \frac{\epsilon}{2} \sum_{\text{近}} P_{n,k}(x) \leq \frac{\epsilon}{2} \cdot 1 = \frac{\epsilon}{2}$$

Weierstrass Approximation Theorem

$$\sum_{\text{远}} \left| f\left(\frac{k}{n}\right) - f(x) \right| P_{n,k}(x) \leq 2M \sum_{\text{远}} P_{n,k}(x) \leq 2M \sum_{\text{远}} \frac{\left(\frac{k}{n} - x\right)^2}{\delta^2} P_{n,k}(x)$$

远的定义

$$\leq \frac{2M}{\delta^2} \sum_{k=0}^{n} \left(\frac{k}{n} - x\right)^2 P_{n,k}(x)$$

$$\leq \frac{2M}{\delta^2} \cdot \frac{1}{4n} = \frac{M}{2n\delta^2}$$

前述工具

George Cybenko，1989，Universal Approximation Theorem

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(w_j^T x + b_j)$$

即对于任意连续函数 $f(x)$ 和任意误差 $\epsilon > 0$，总存在一个 $G(x)$ 使得 $|G(x) - f(x)| < \epsilon$。

George Cybenko，1989，Universal Approximation Theorem

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(w_j^T x + b_j)$$

即对于任意连续函数 f(x) 和任意误差 \epsilon > 0，总存在一个 G(x) 使得：

|G(x) - f(x)| < \epsilon。

证明

- 把“函数”看作是“无限维度的向量”：如果我们在一段音频信号（也就是一个连续函数 $f(x)$）上进行采样：如果无限密集地采样下去？这个连续函数 $f(x)$ 就变成了一个无限维度的向量

- 假设神经网络 $NN(x)$ 有缺陷，有个目标信号 $f(x)$ 它死活拟合不了。目标信号 $f(x)$ 和神经网络的输出之间，一定存在一个非零的误差信号 $e(x)$

- 如果神经网络完全没法表达这个 $e(x)$，这就意味着：不管神经网络的权重怎么调，调出什么形状，它都和 $e(x)$ 绝对正交

- 但是！ Cybenko 证明了，Sigmoid 函数极其特殊，世界上根本不存在任何一个非零的信号，能跟所有相位的 Sigmoid 函数相乘的积分都为 0。Sigmoid 就像一个全能雷达，没有任何信号能对它保持完全“隐身”。

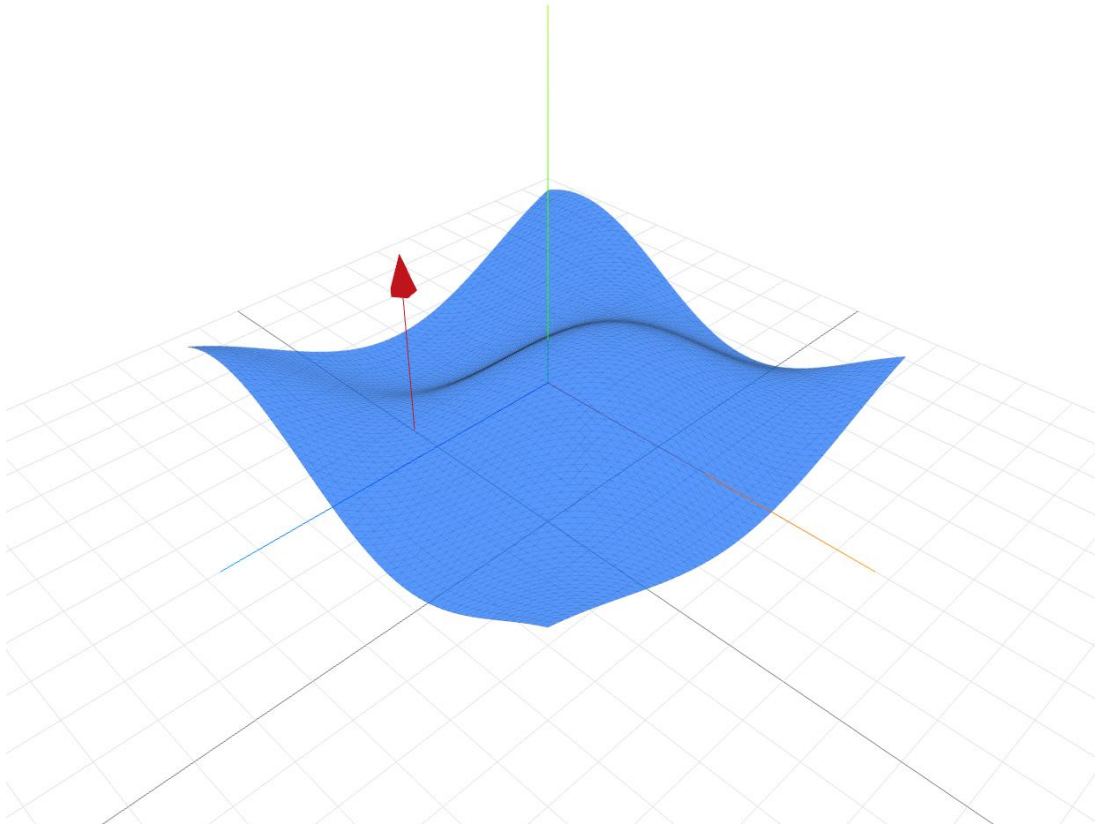
为什么正交？

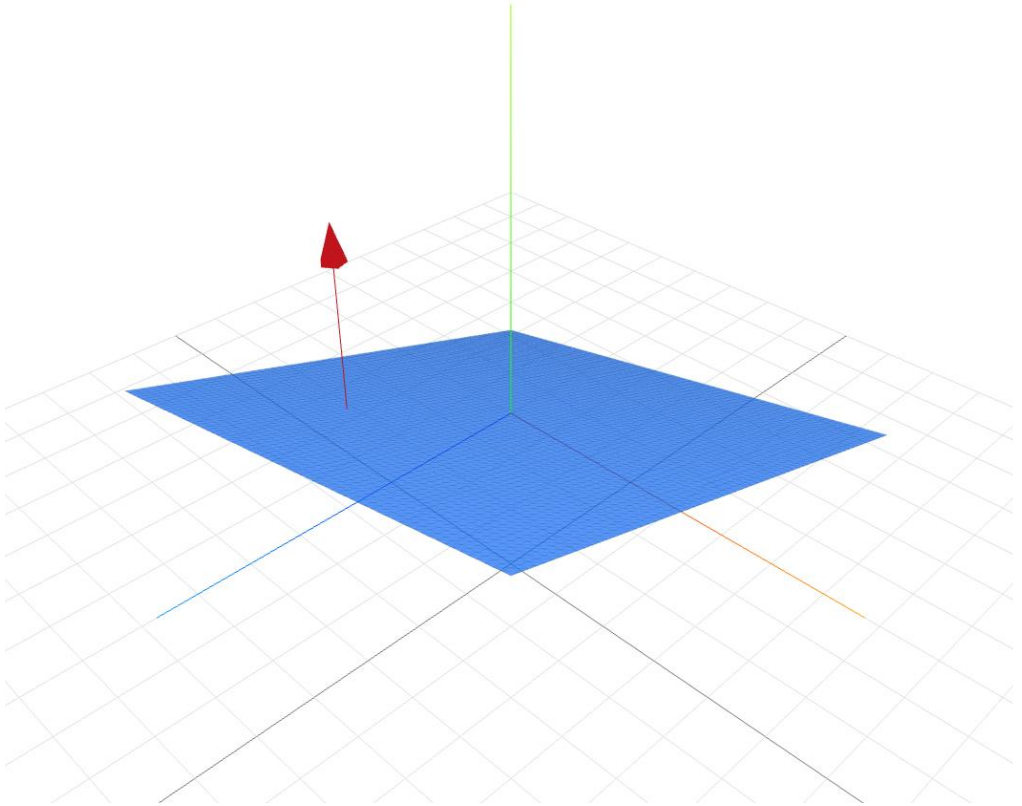
- 工具：只能用在“线性空间”（平坦的桌面）上，对弯曲的流形是无效的
- Cybenko 做了一个非常巧妙的操作：他把神经网络拆成了两半来看

$$f(x) = \sum_{i=1}^{N} v_i \sigma(w_i x + b_i)$$

- ✓ 内层的非线性字典：Cybenko 假设我们把所有可能的 w 和 b 的组合全部穷举出来，算出一万亿个、甚至无数个固定形状的 Sigmoid 函数。他把这些函数当成一个巨大的“积木库”或者“字典”。
- ✓ 外层的线性组合：既然内部的形状已经固定了，那么最外层的输出权重 v_i 就仅仅是在做线性加权求和

为什么正交？





为什么正交？

- 误差 $e(x) = f(x) - NN_{best}(x)$。在几何上，它就是连接“桌面上的阴影”和“空中的点”的那根绝对垂直的线段

- 根垂直的误差支柱 $e(x)$，与桌面上的任何一条线、任何一个方向都是垂直。

不存在非0信号跟所有Sigmoid 函数相乘积分都为0

- 假设存在一个非零的“幽灵误差信号” e(x)，它对所有频率 w 和相位 b 的 Sigmoid 函数都绝对正交。即：

$$\int_{-\infty}^{\infty} e(x) \cdot \sigma(wx + b)dx = 0 \quad (\text{对任意 } w, b \text{ 都成立})$$

- 令 $w \to \infty$

- Sigmoid 函数就退化成了工科生最熟悉的单位阶跃信号： H(x - \theta)

$$\int_{-\infty}^{\infty} e(x) \cdot H(x - \theta)dx = 0$$

$$\int_{\theta}^{\infty} e(x)dx = 0$$

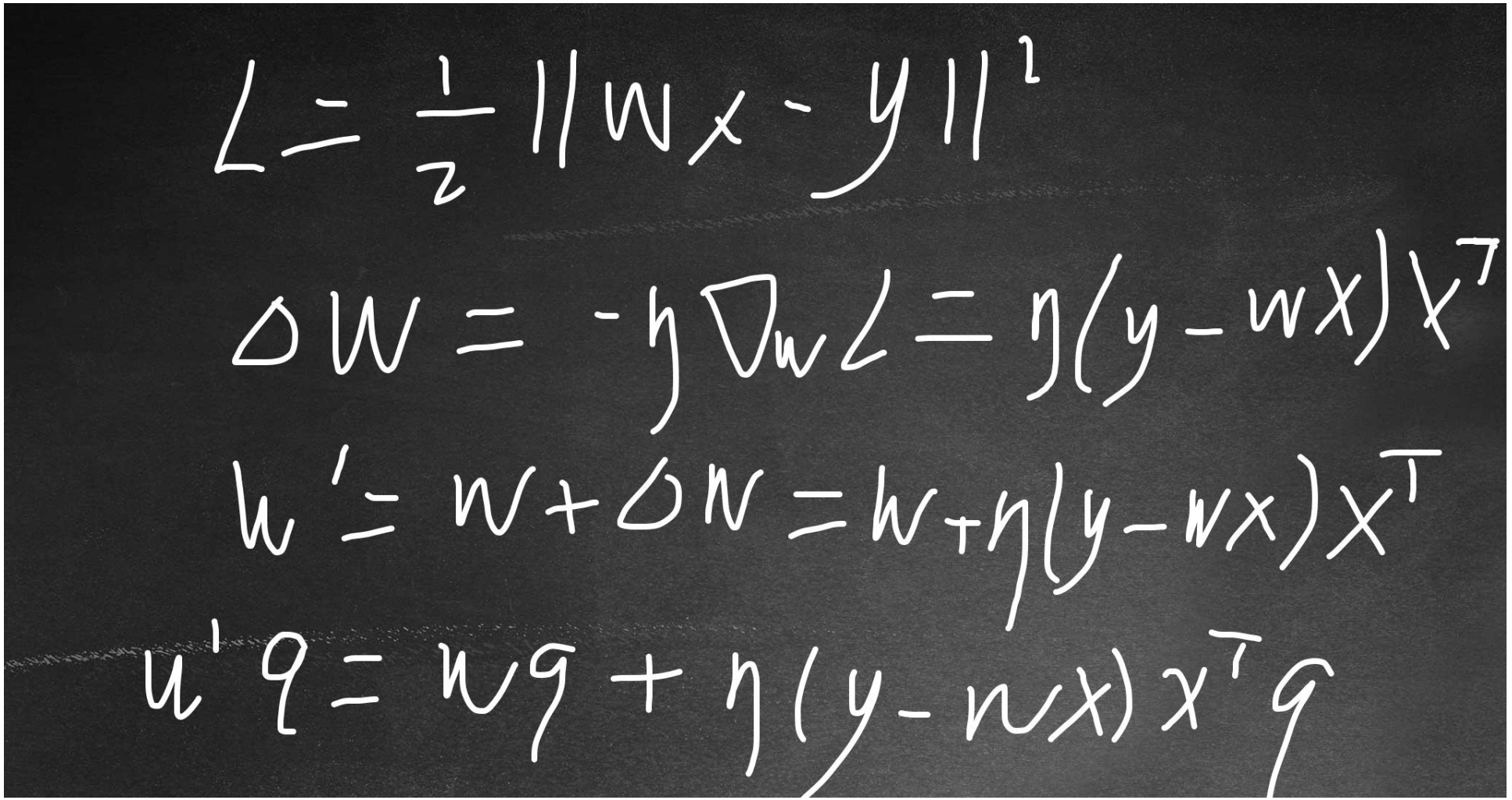
不存在非0信号跟所有Sigmoid 函数相乘积分都为0

$$S(\theta) = \int_{\theta}^{\infty} e(x)dx \equiv 0$$

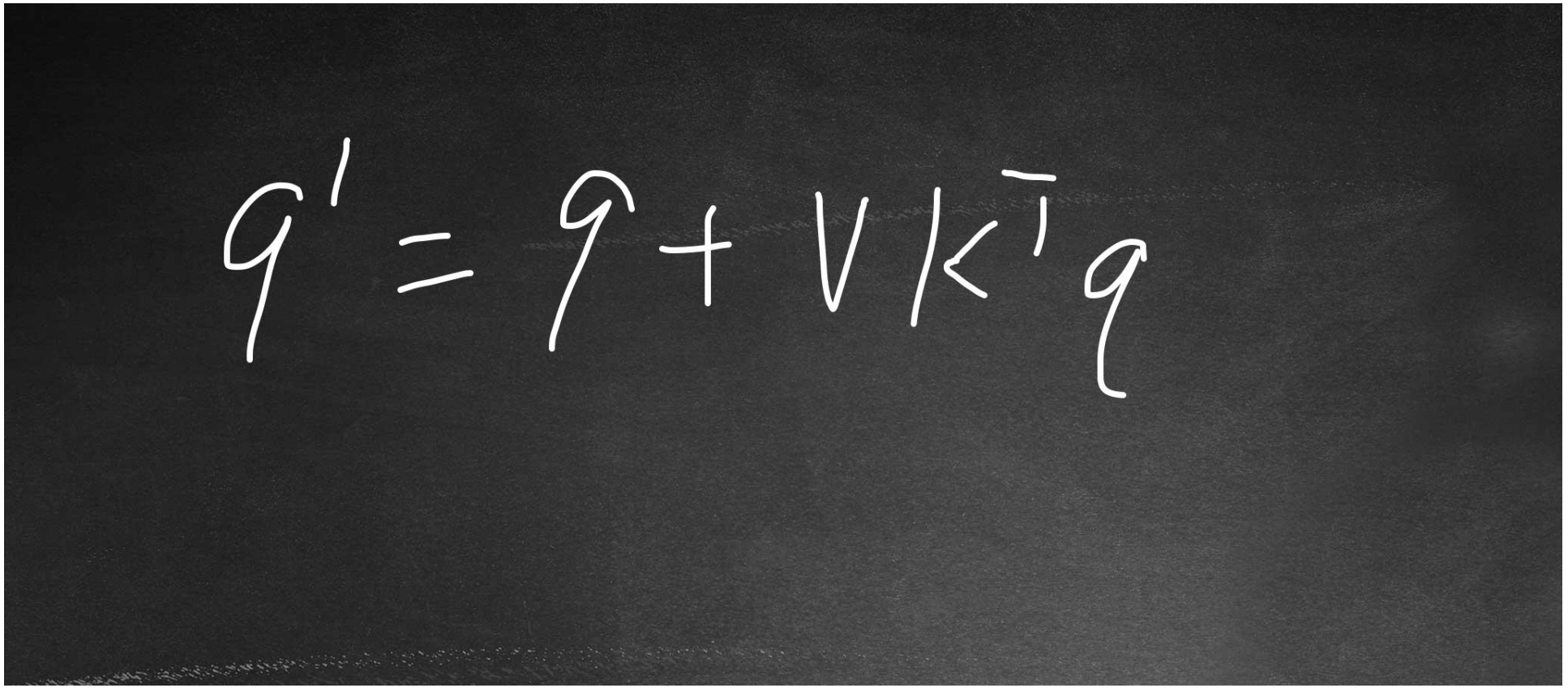
$$\frac{d}{d\theta}S(\theta) = -e(\theta) \equiv 0$$

$$e(\theta) = 0$$

in-context learning和prompt learning是隐式的梯度下降



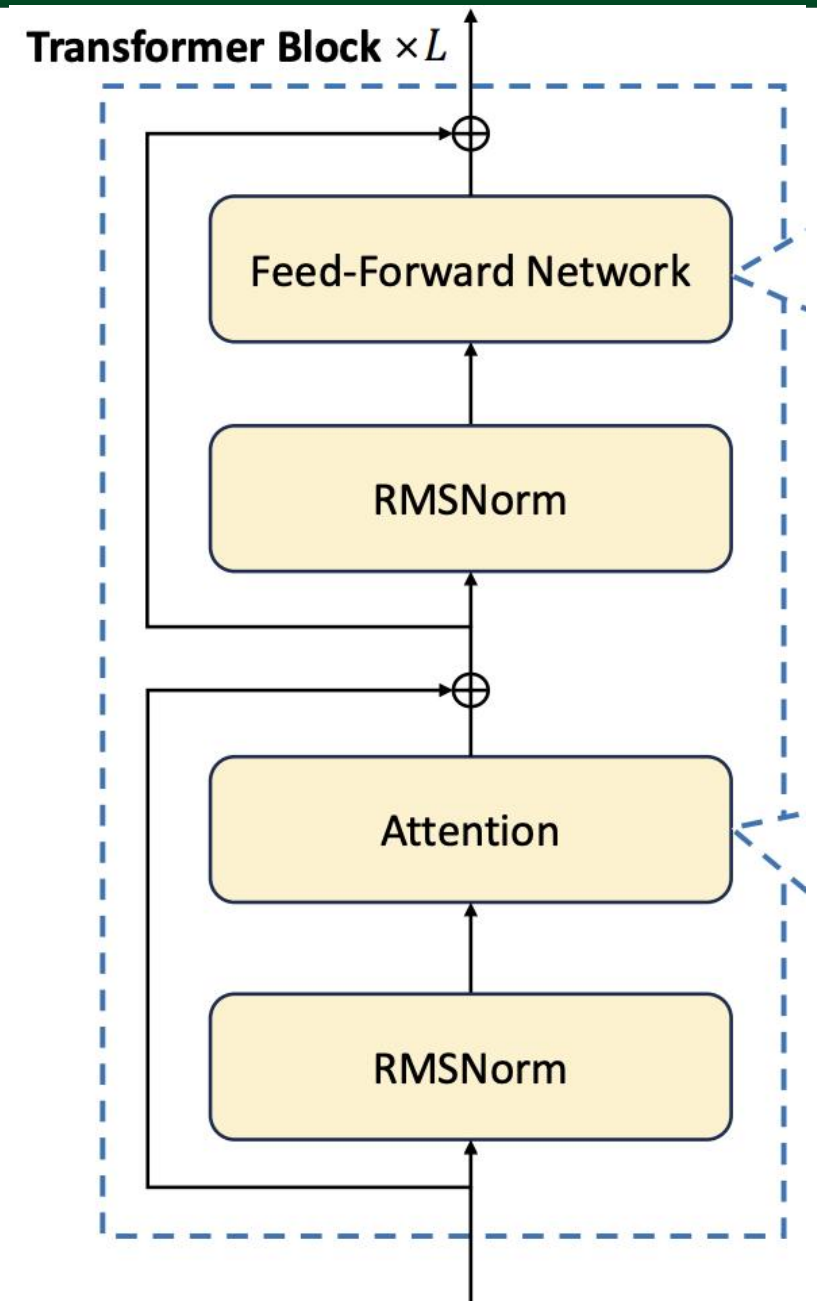
in-context learning和prompt learning是隐式的梯度下降



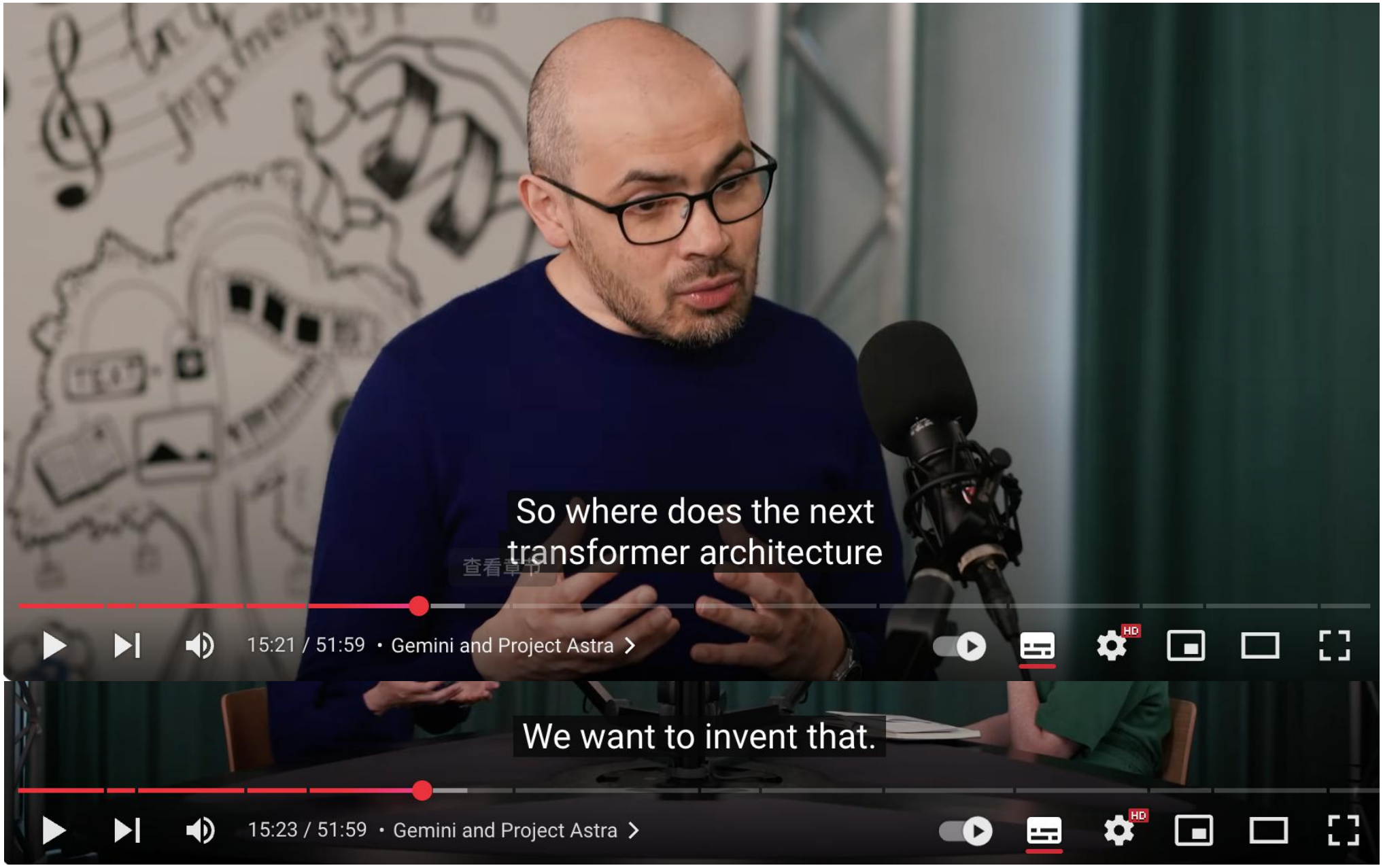
本节内容

CONTENTS

Transformer



新架构


So where does the next
transformer architecture
查看章节
15:21 / 51:59 • Gemini and Project Astra
We want to invent that.
15:23 / 51:59 • Gemini and Project Astra

Is attention all you need?

- https://www.isattentionallyouneed.com/

Is Attention All You Need?

Current Status: Yes

Time Remaining: 259d 15h 20m 19s

258天

Donation of equity

Proposition:

On January 1, 2027, a Transformer-like model will continue to hold the state-of-the-art position in most benchmarked tasks in natural language processing.

For the Motion

Jonathan Frankle
@jefrankle
Harvard Professor
Chief Scientist Mosaic ML

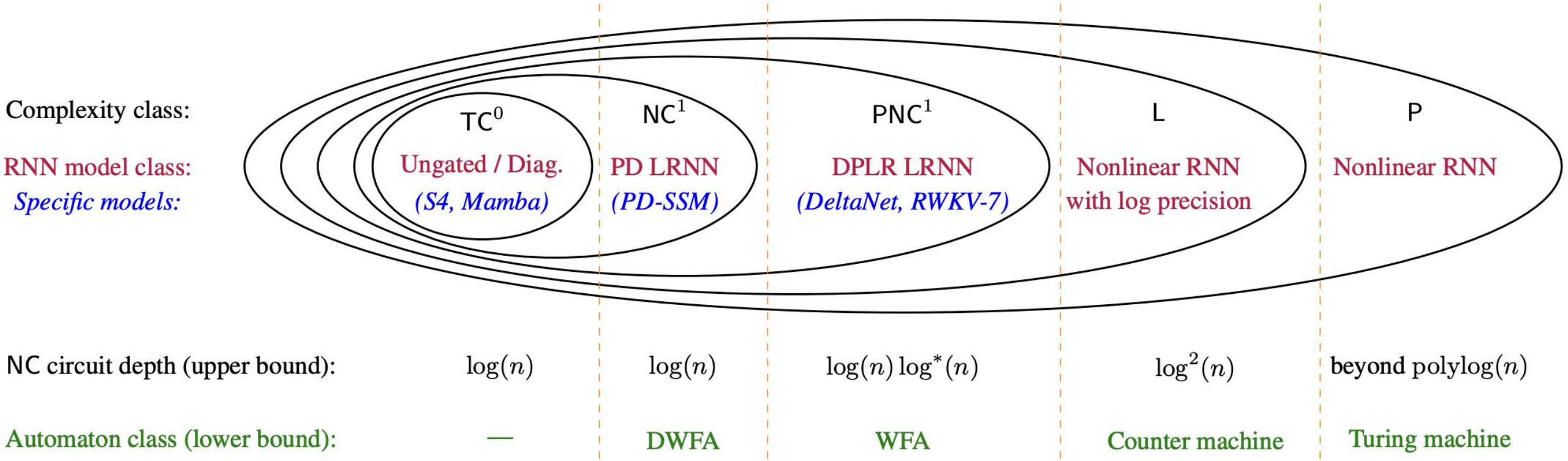
Against the Motion

Sasha Rush
@srush_nlp
Cornell Professor
Research Scientist Hugging Face 🤗

Wager

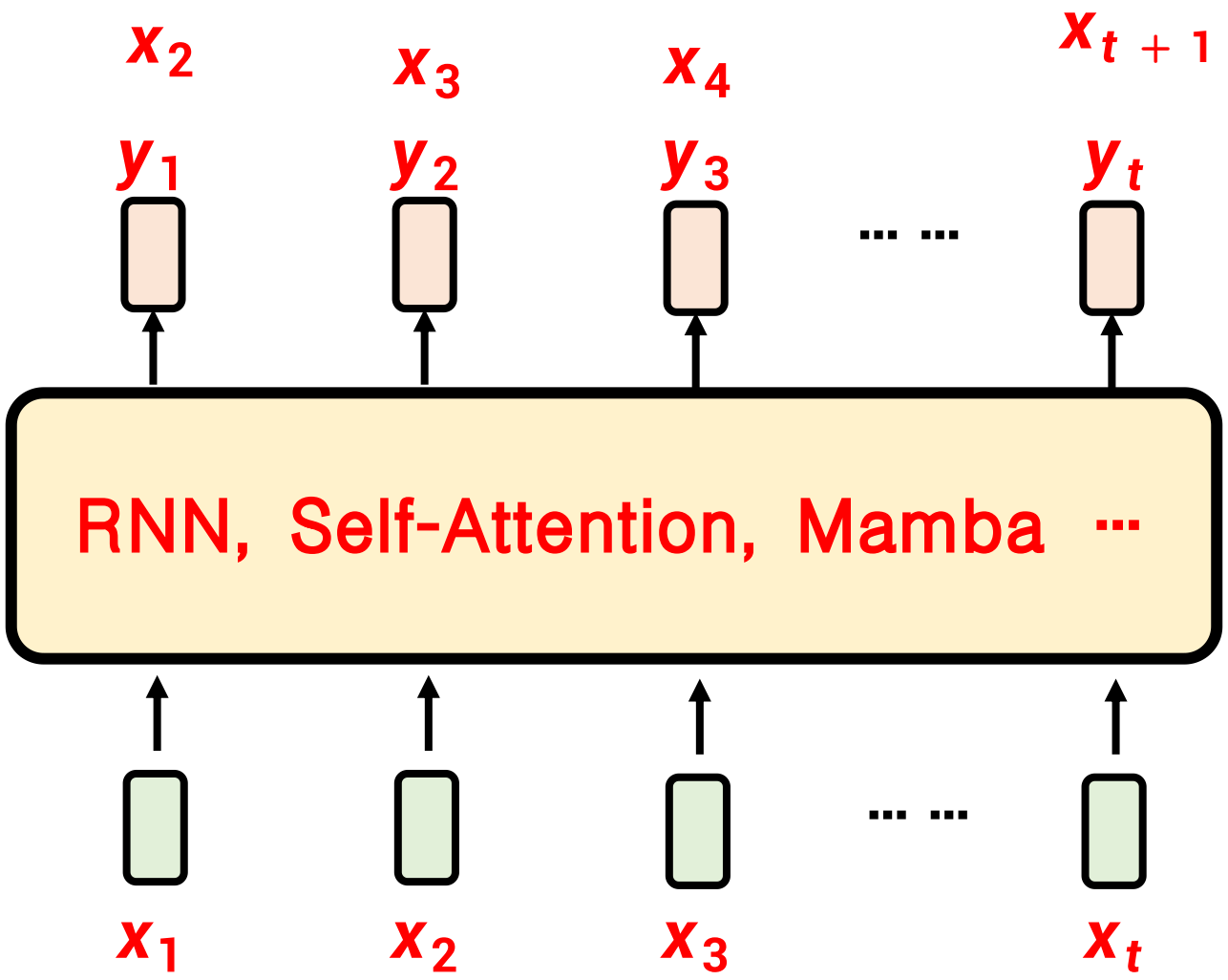
The wager is for donation of equity in Mosaic ML or Hugging Face to a charity of the winner's choice. Details to come.

不同的新架构

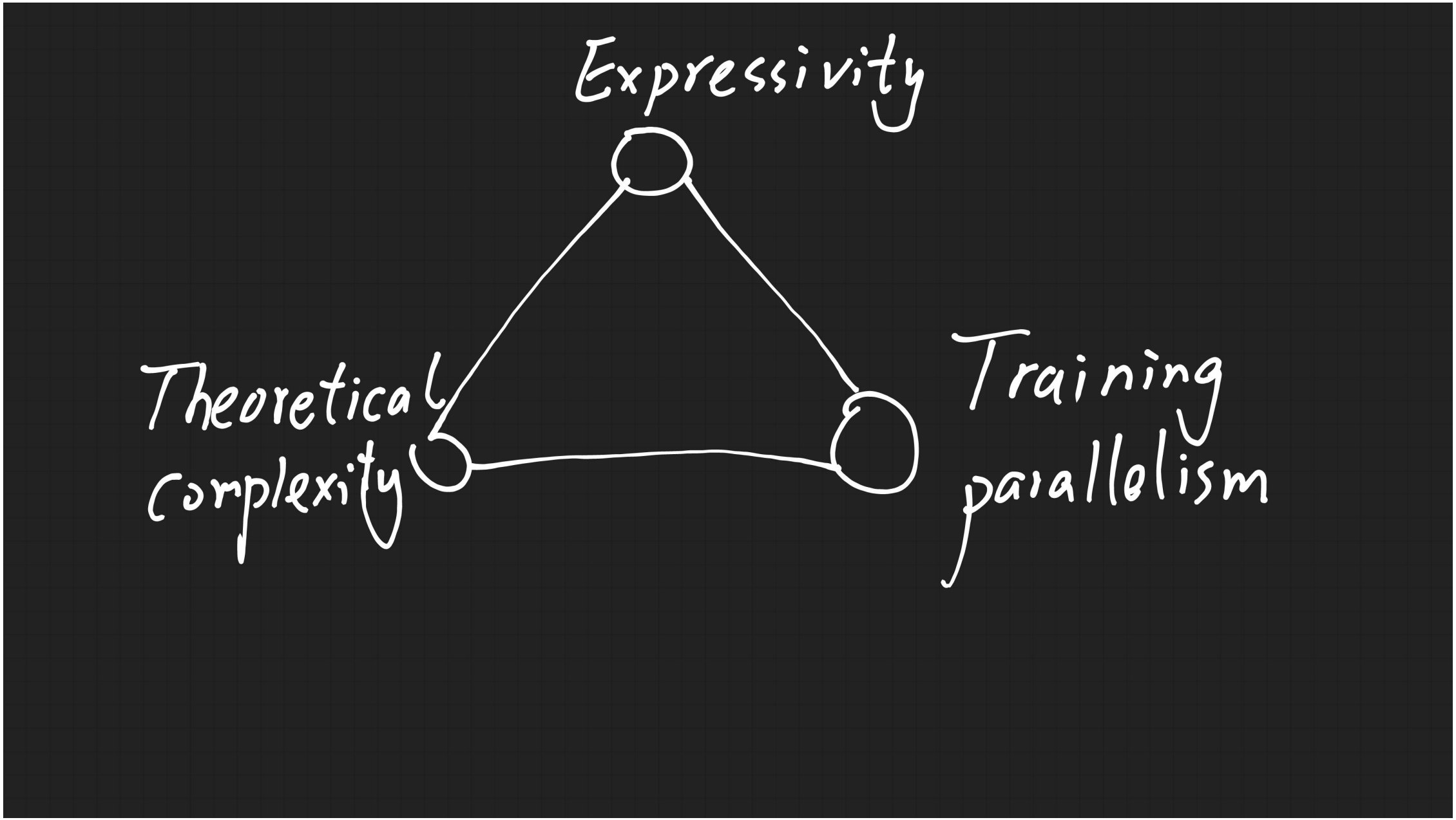


https://arxiv.org/pdf/2603.03612

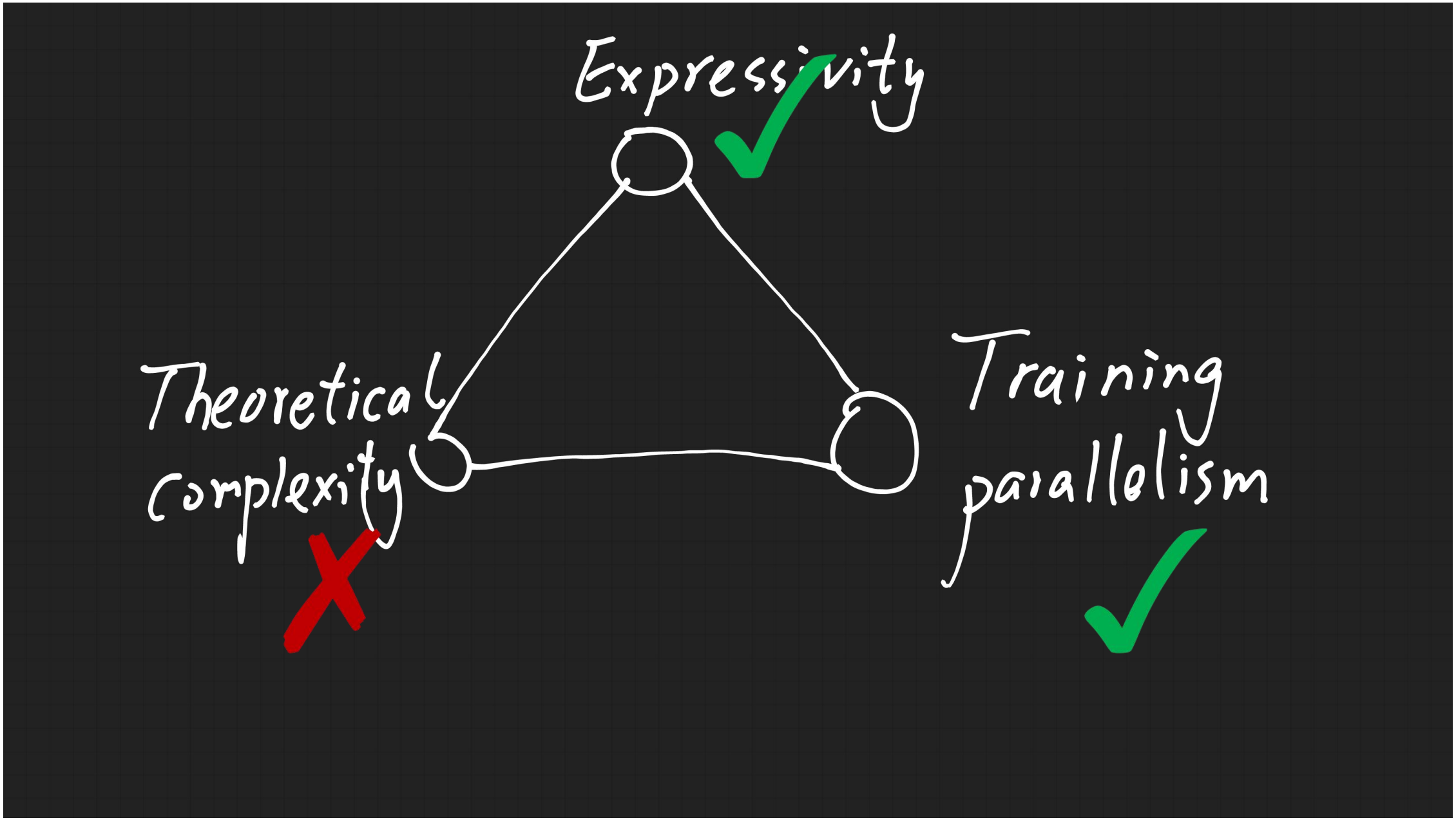
要解的问题



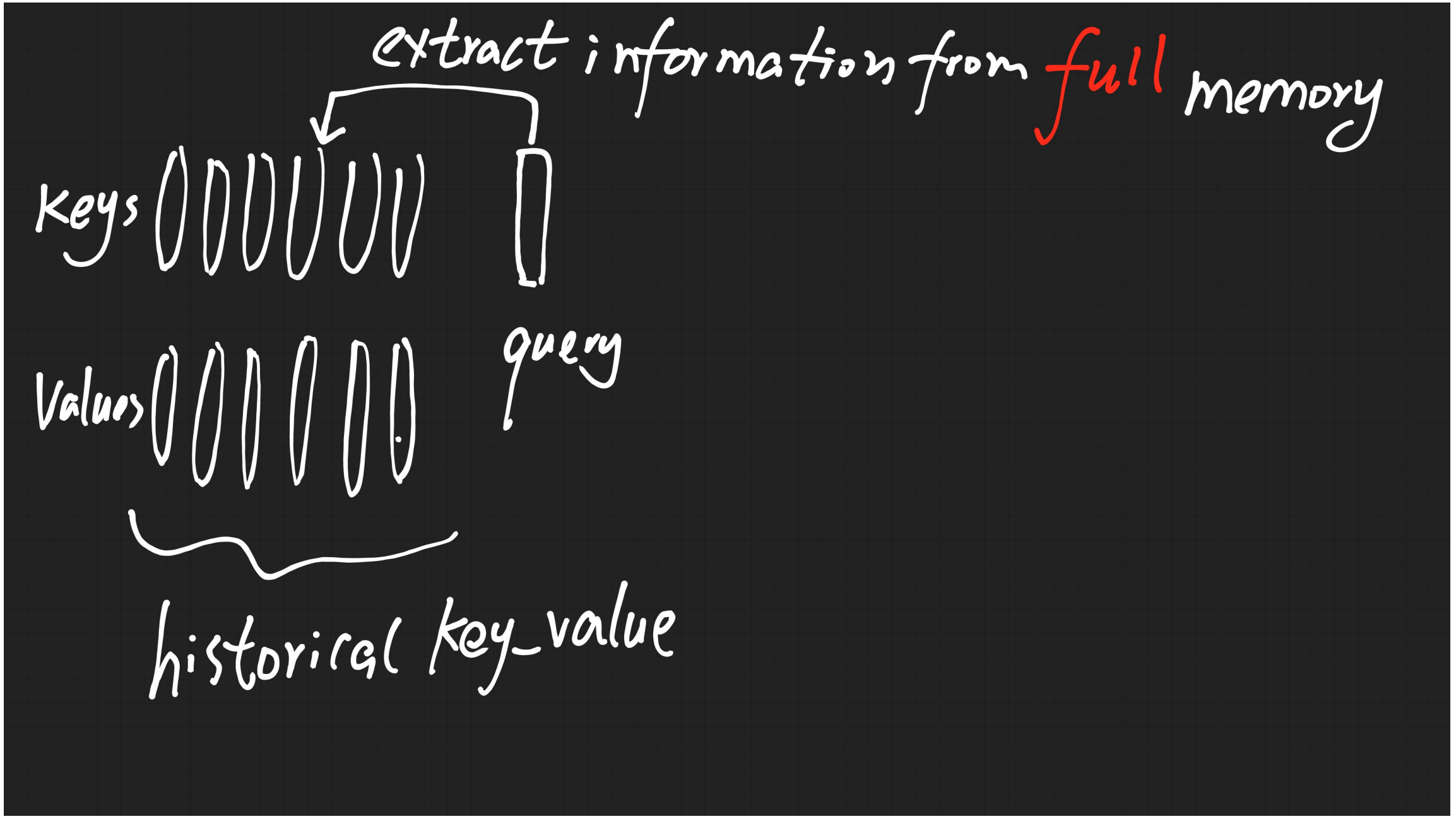
不可能三角


Expressivity
Theoretical complexity
Training parallelism

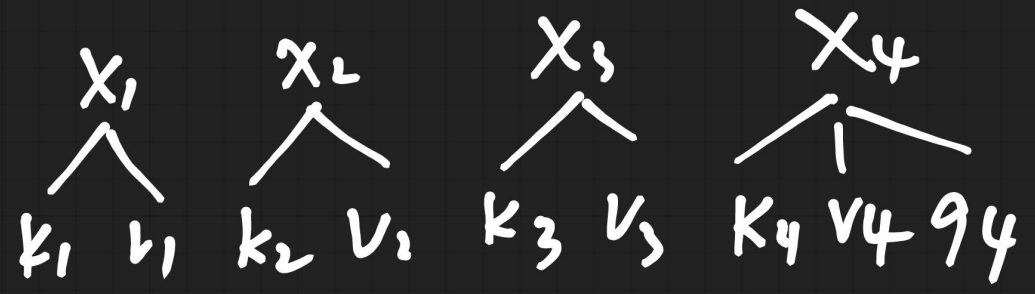
Transformer



Expressivity

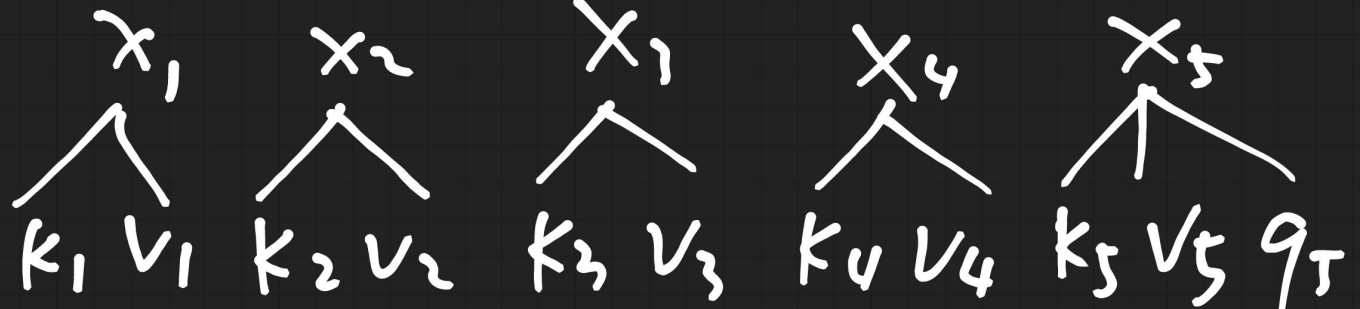


Computation Complexity



x_5
$\vdots$

$$y_4 = \overline{\overline{q_4 k_1}} v_1 + \overline{\overline{q_4 k_2}} v_2 + \overline{\overline{q_4 k_3}} v_3 + \overline{\overline{q_4 k_4}} v_4$$

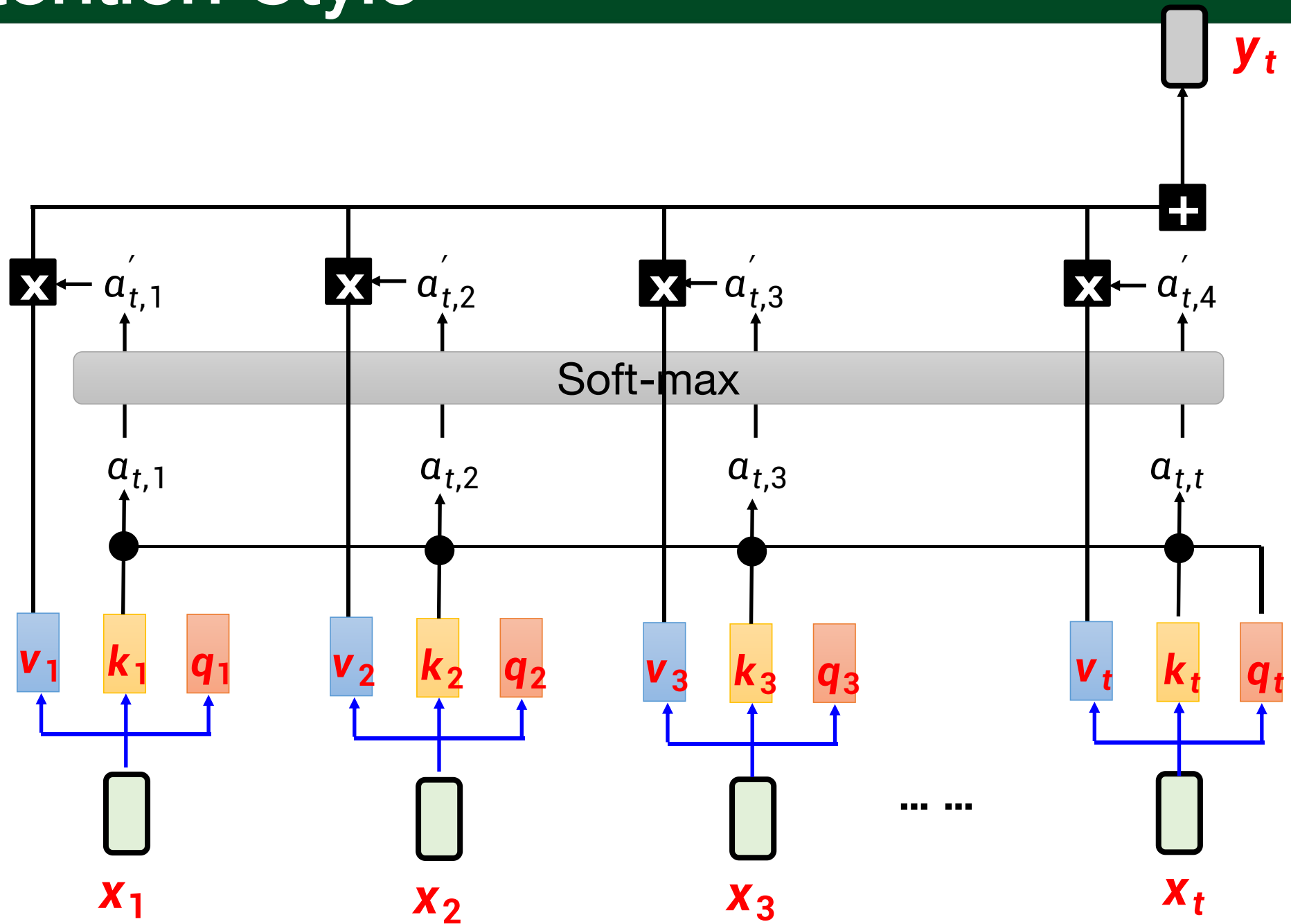


$$y_5 = \overline{\overline{q_5 k_1}} v_1 + \overline{\overline{q_5 k_2}} v_2 + \overline{\overline{q_5 k_3}} v_3 + \overline{\overline{q_5 k_4}} v_4 + \overline{\overline{q_5 k_5}} v_5$$

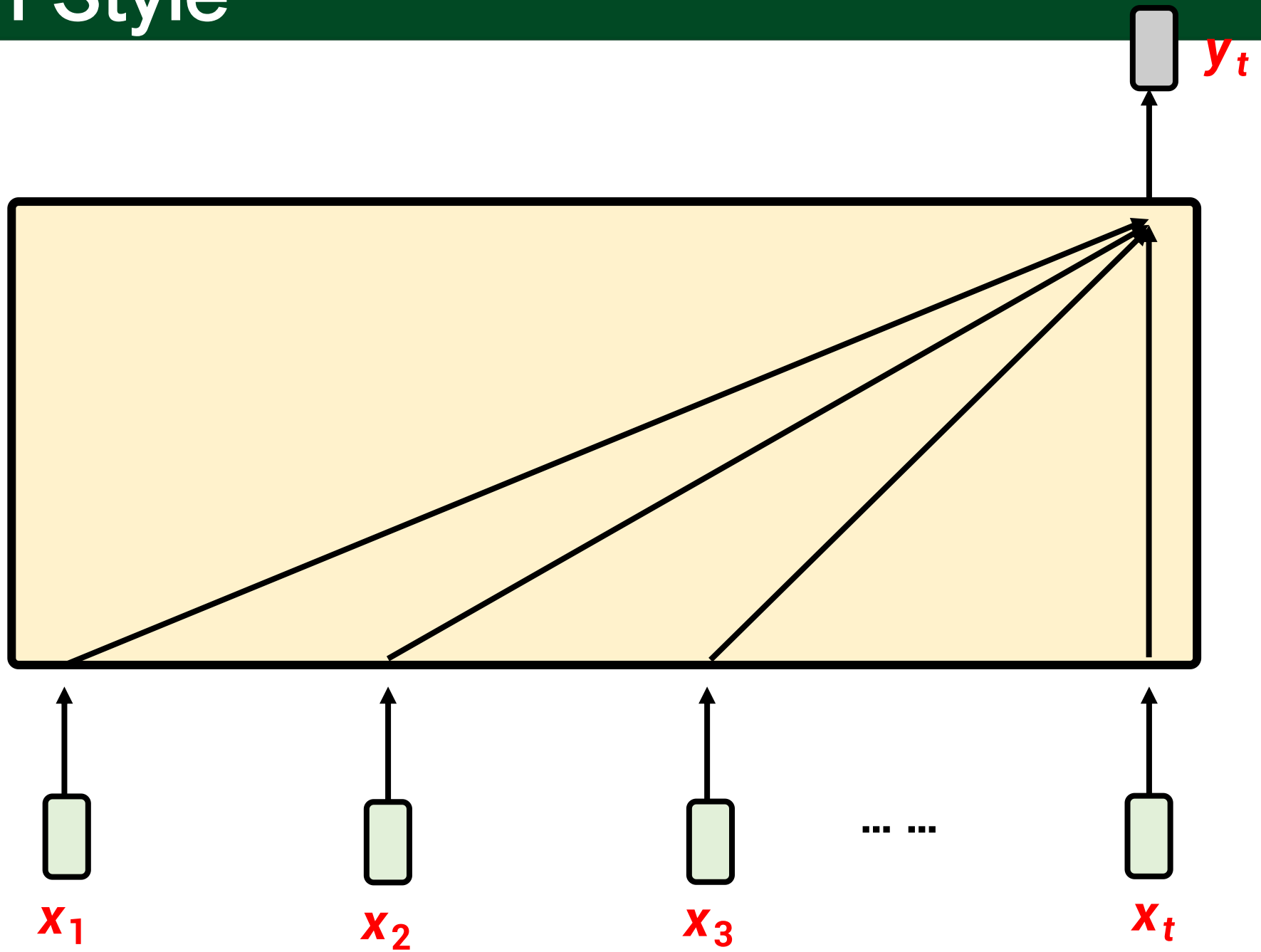
$\vdots$
x_6

test: $O(n)$

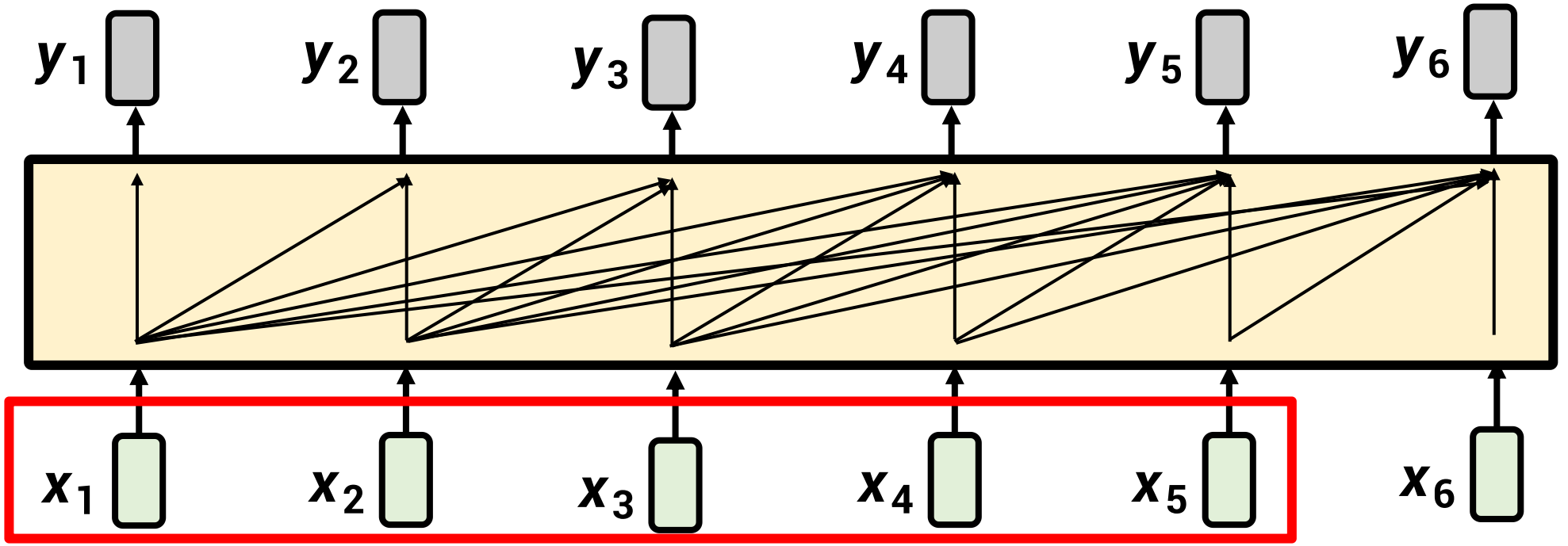
Attention Style



Attention Style



Attention: Theoretical Complexity



输入越长，运算量越来越大

Training Parallelism

$x_1 \to x_2$

$(x_1, x_2) \to x_3$

$(x_1, x_2, x_3) \to x_4$

$$y_1 = sm(q_1 k_1) v_1$$

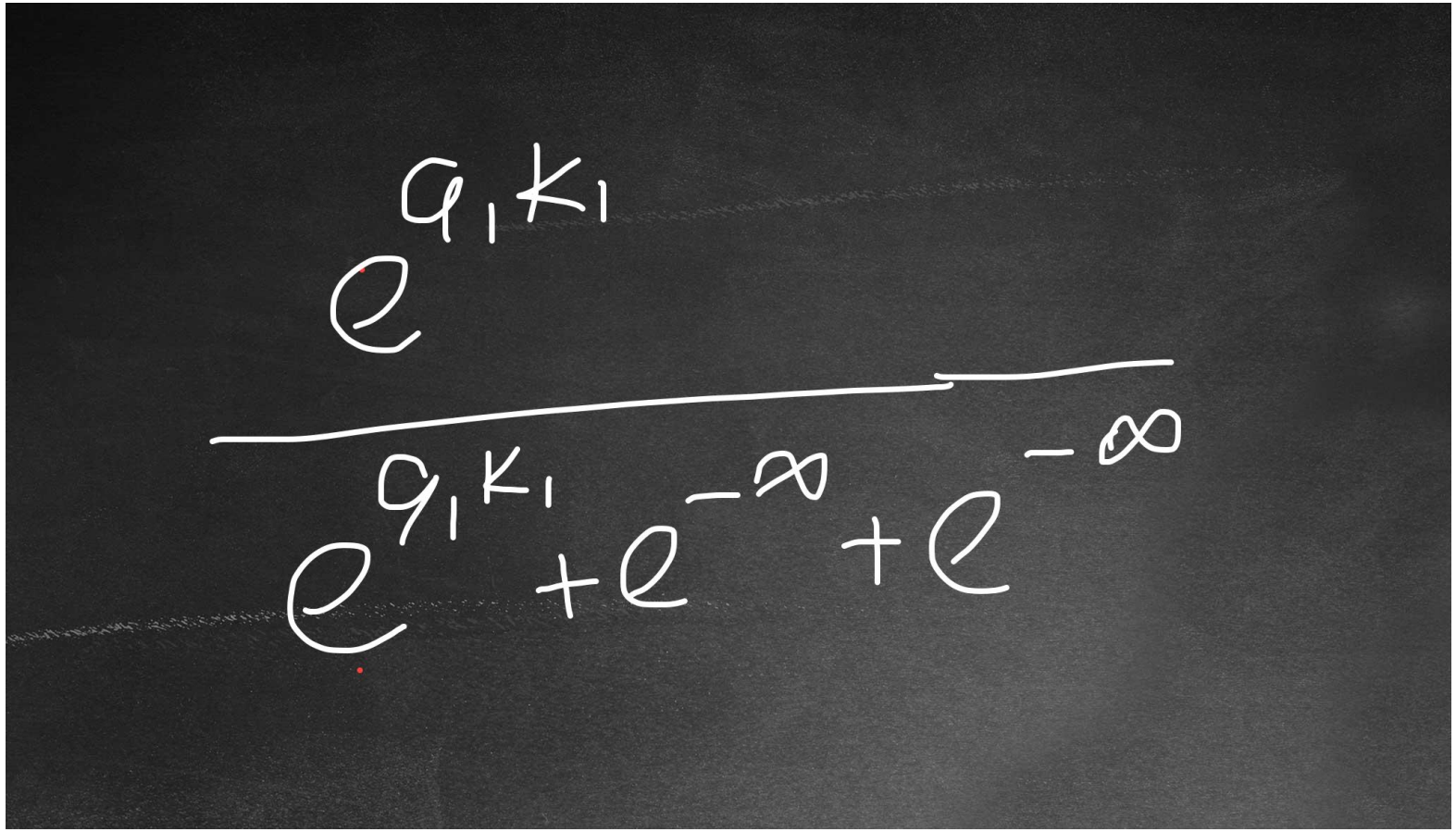
$$y_2 = sm(q_2 k_1) v_1 + sm(q_2 k_2) v_2$$

$$y_3 = sm(q_3 k_1) v_1 + sm(q_3 k_2) v_2 + sm(q_3 k_3) v_3$$

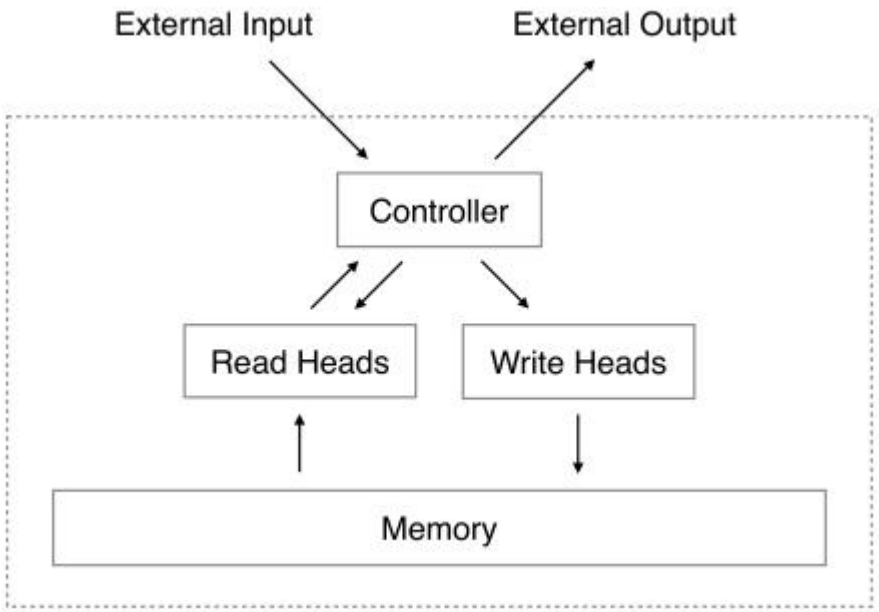
train: $O(n^2)$ but parallel

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} s(q_1 k_1) & 0 & 0 \\ s(q_2 k_1) & s(q_2 k_2) & 0 \\ s(q_3 k_1) & s(q_3 k_2) & s(q_3 k_3) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = s_m \begin{bmatrix} q_1 k_1 & -\infty & -\infty \\ q_2 k_1 & q_2 k_2 & -\infty \\ q_3 k_1 & q_3 k_2 & q_3 k_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = sm(QK)V$$

SoftMax



Attention 的概念很早就有了



Joe went to the garden then Fred picked up the milk; Joe moved to the bathroom and Fred dropped the milk, and then Dan moved to the living_room.
Where is Dan? A: living room I believe
Where is Joe? A: the bathroom
Fred moved to the bedroom and Joe went to the kitchen then Joe took the milk there and Dan journeyed to the bedroom; Joe discarded the milk.
Where is the milk now ? A: the milk is in the kitchen
Where is Dan now? A: I think he is in the bedroom
Joe took the milk there, after that Mike travelled to the office, then Joe went to the living_room, next Dan went back to the kitchen and Joe travelled to the office.
Where is Joe now? A: I think Joe is in the office

Neural Turing Machine

https://arxiv.org/abs/1410.5401

Memory Networks

https://arxiv.org/pdf/1410.3916

Attention

Attention Is All You Need

不是发明 Attention ， 而是拿掉 Attention 以外的东西

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

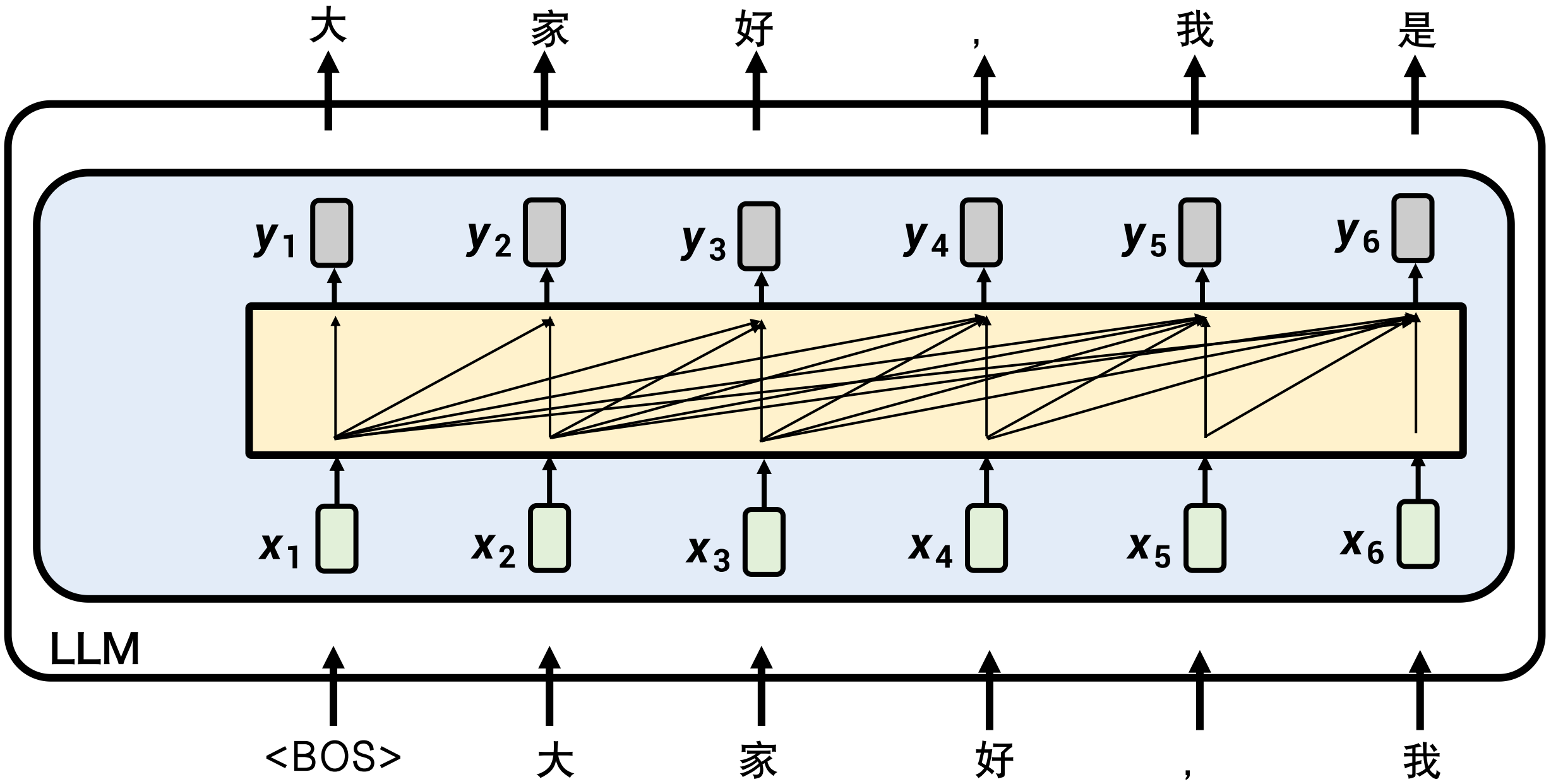
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

https://arxiv.org/abs/1706.03762

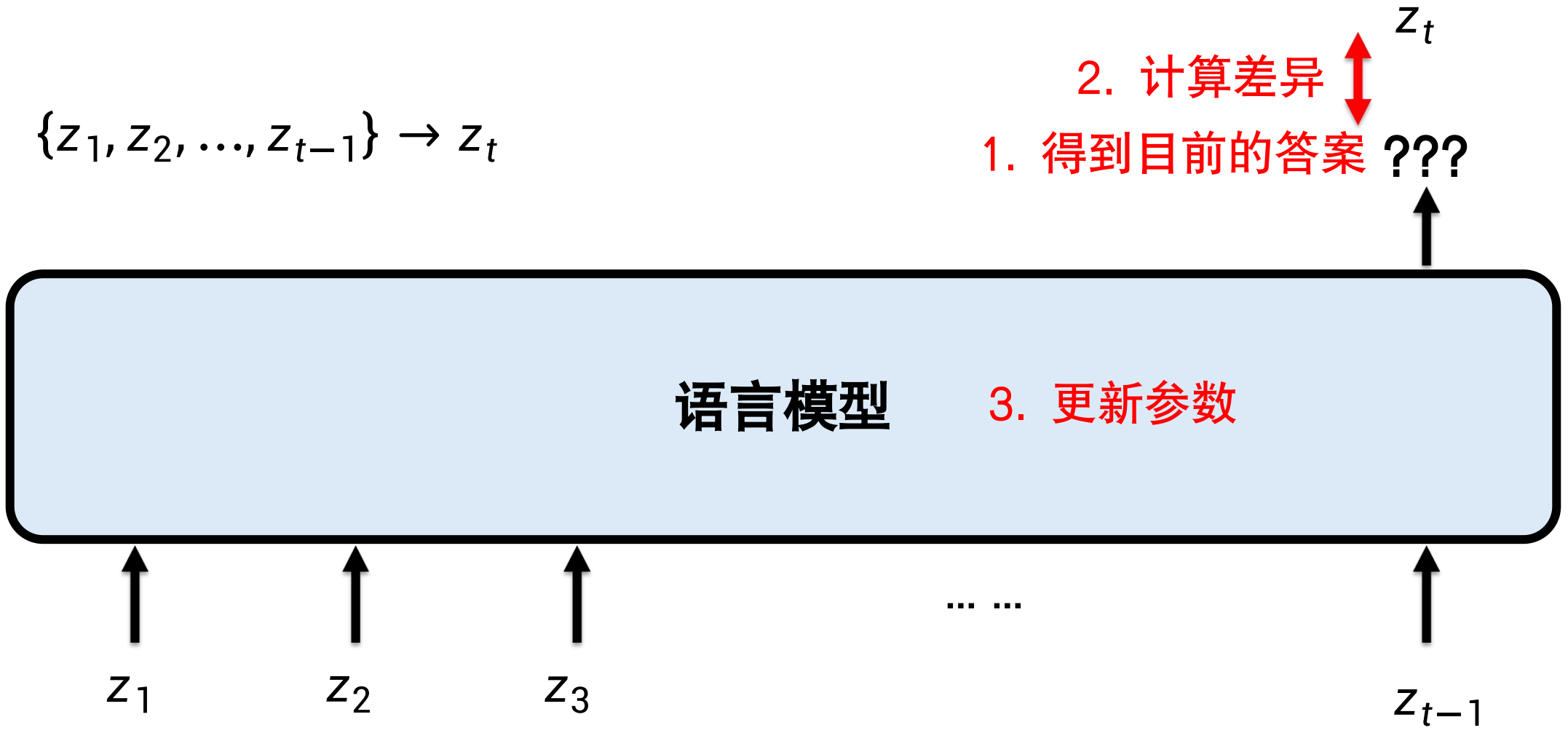
Attention



语言模型的训练

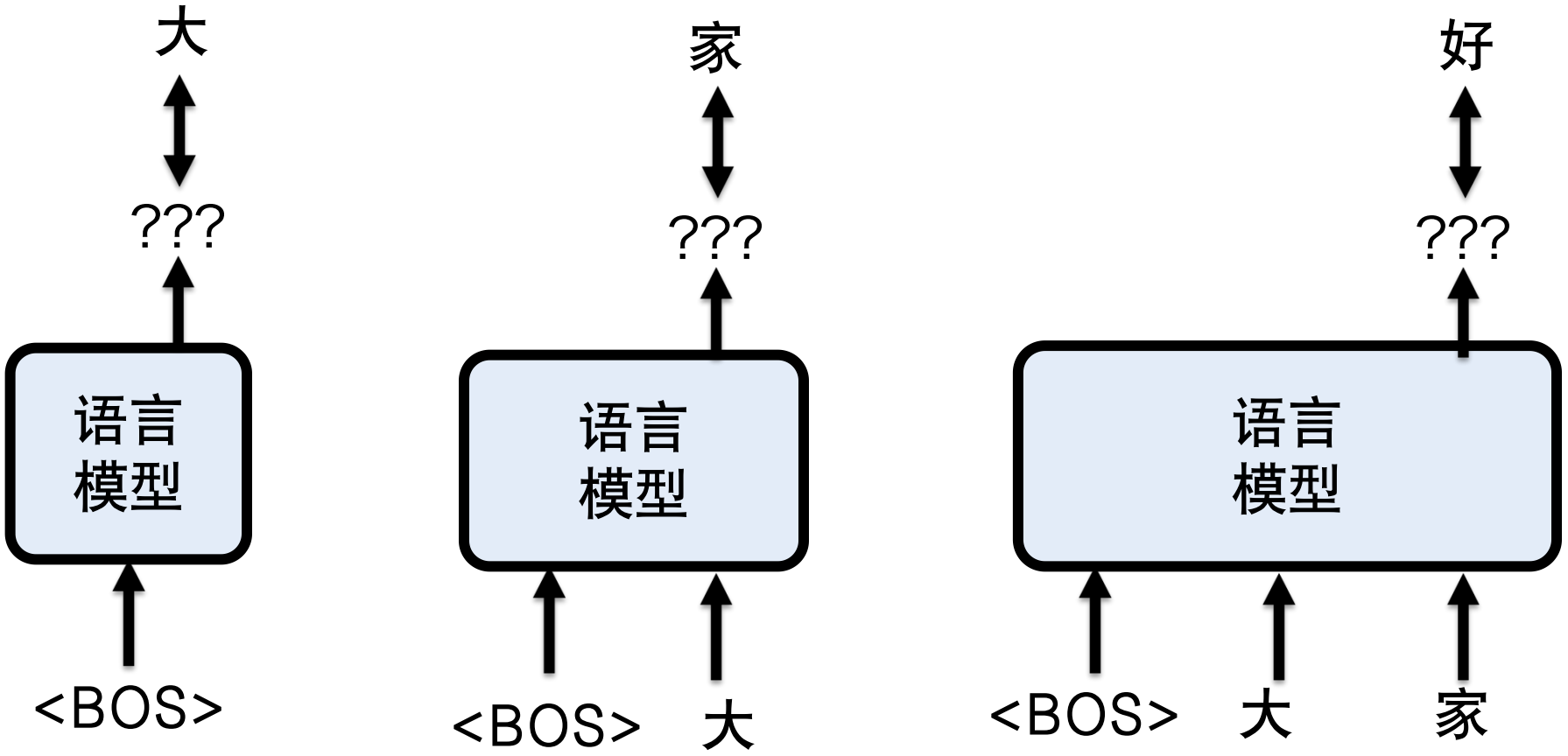
❑更新参数前要先算出自己的答案

$\{z_1, z_2, \ldots, z_{t-1}\} \rightarrow z_t$

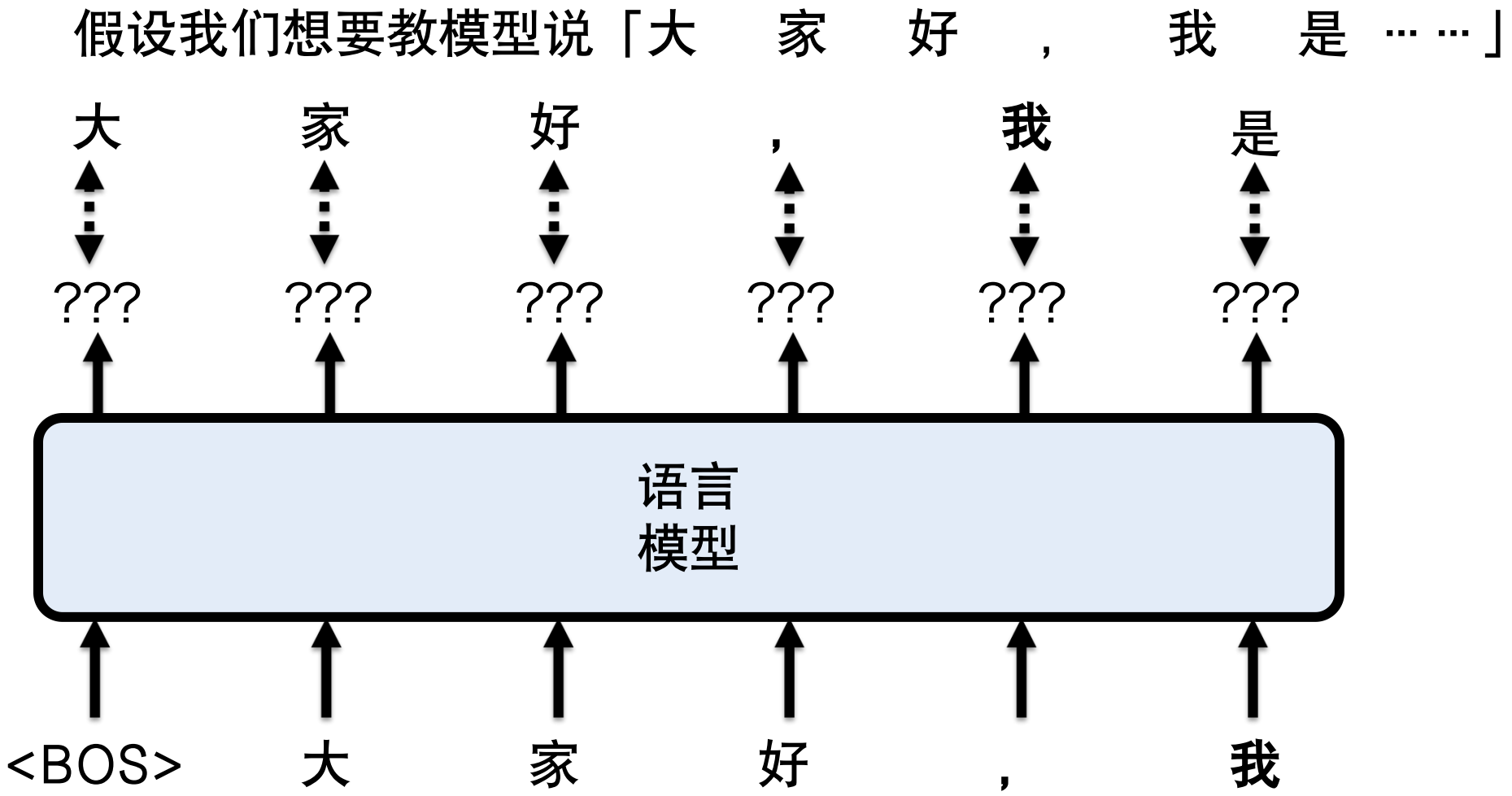


语言模型的训练（找出参数）

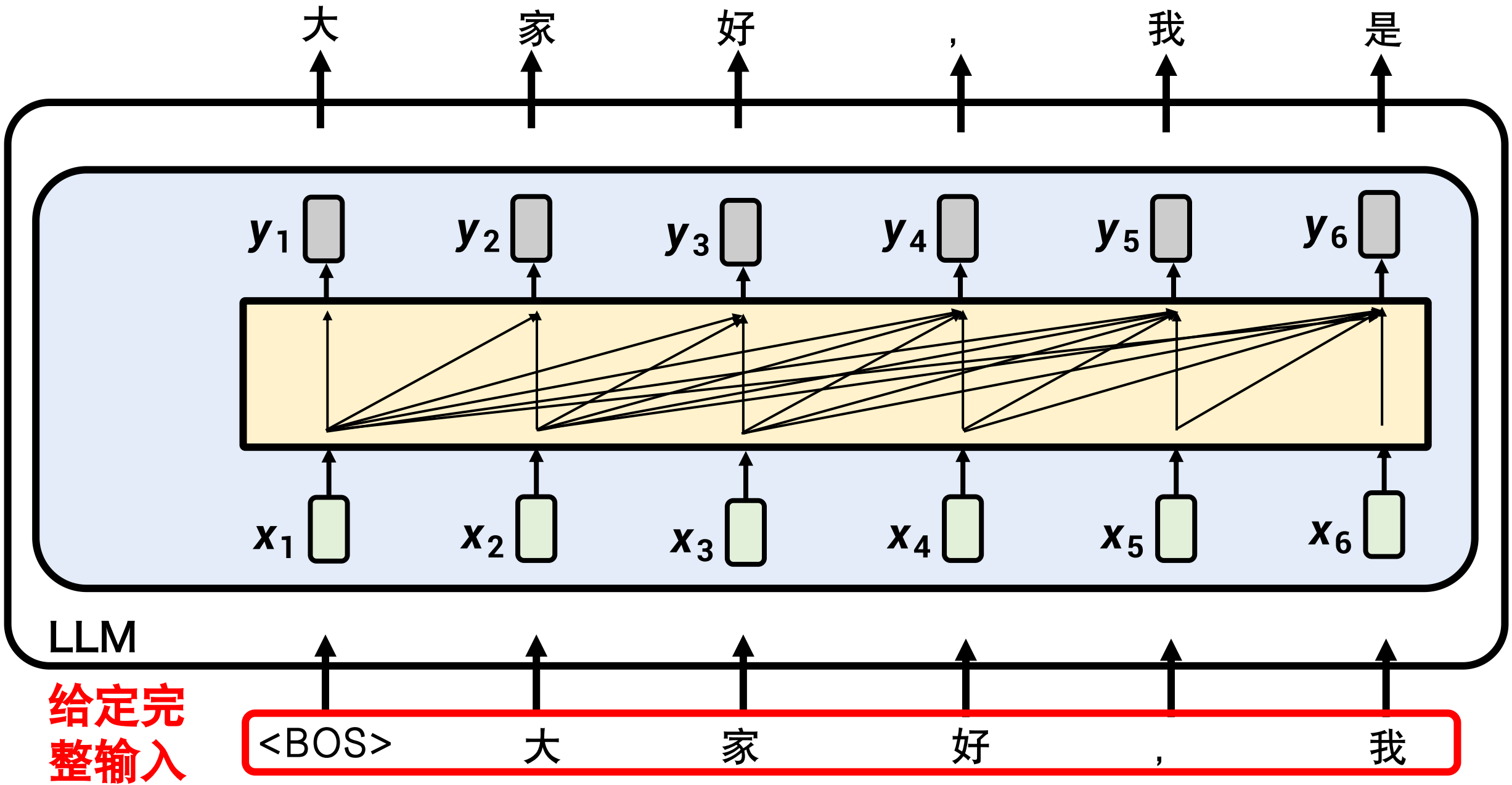
假设我们想要教模型说「大　家　好　，　我　是 …… 」



语言模型的训练（找出参数）



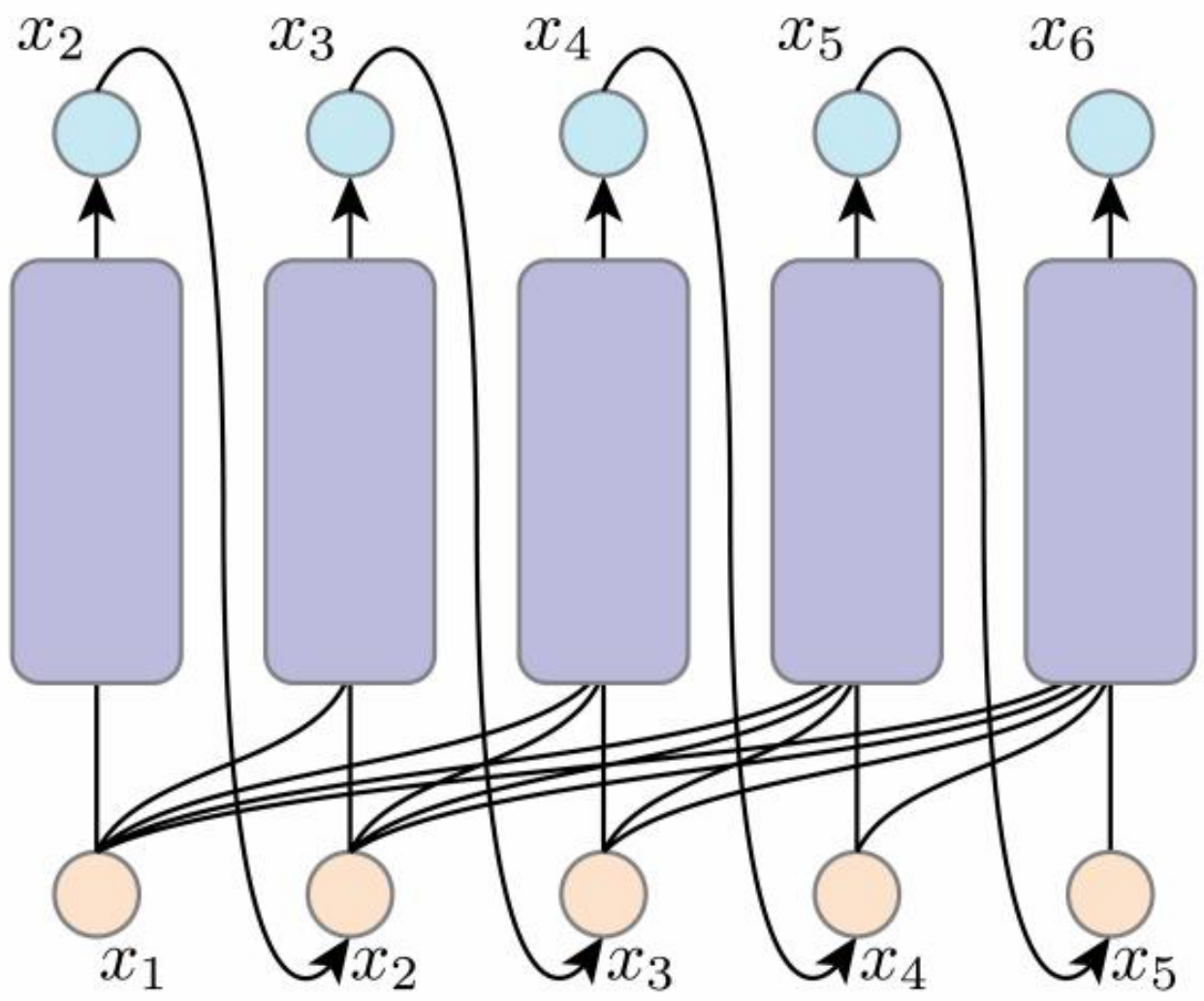
Teacher Forcing



Inference: Autoregressive

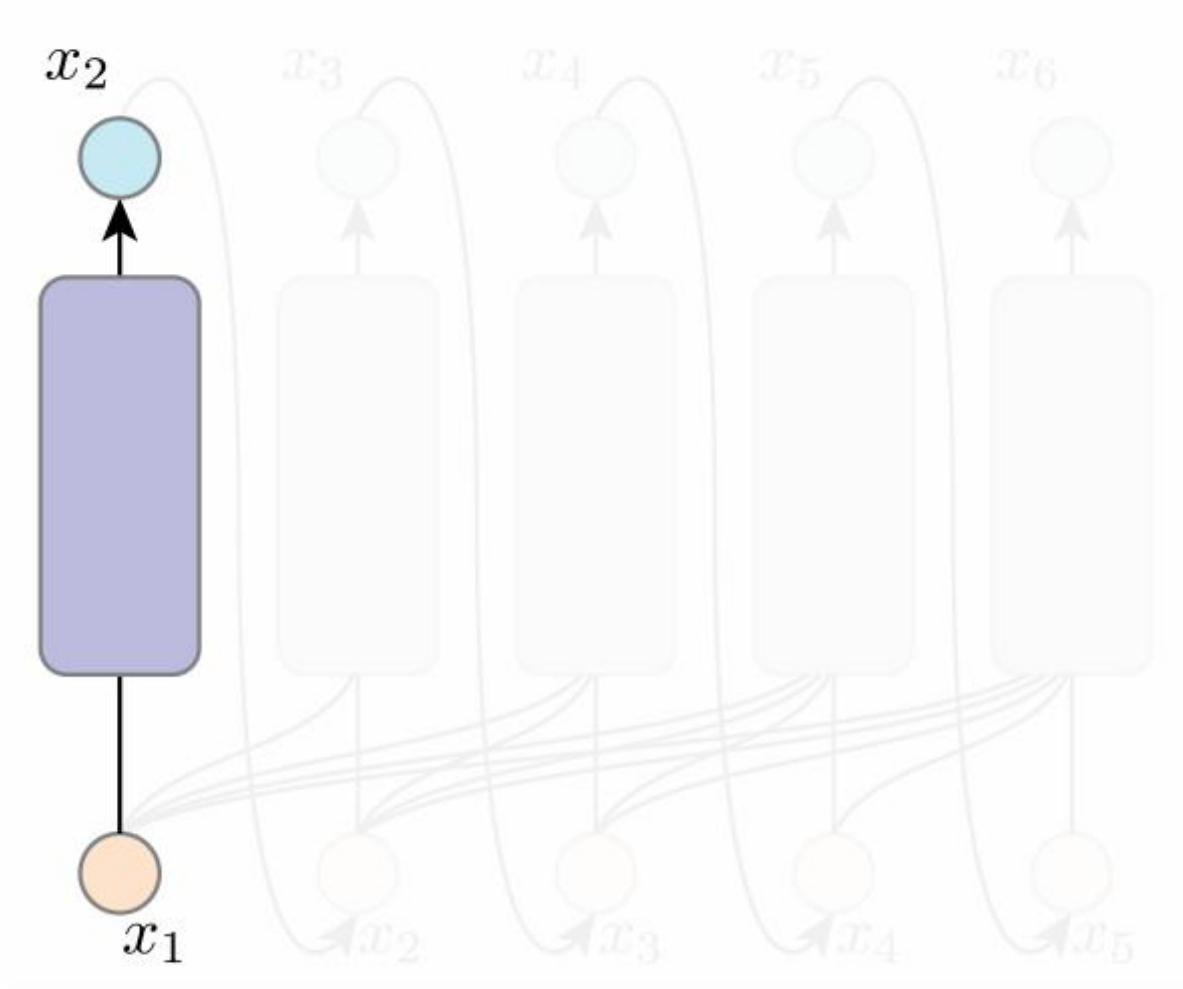
This figure implements this formulation:

$$p(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} p(x_i \mid x_1, x_2, ..., x_{i-1})$$



Inference: Autoregressive

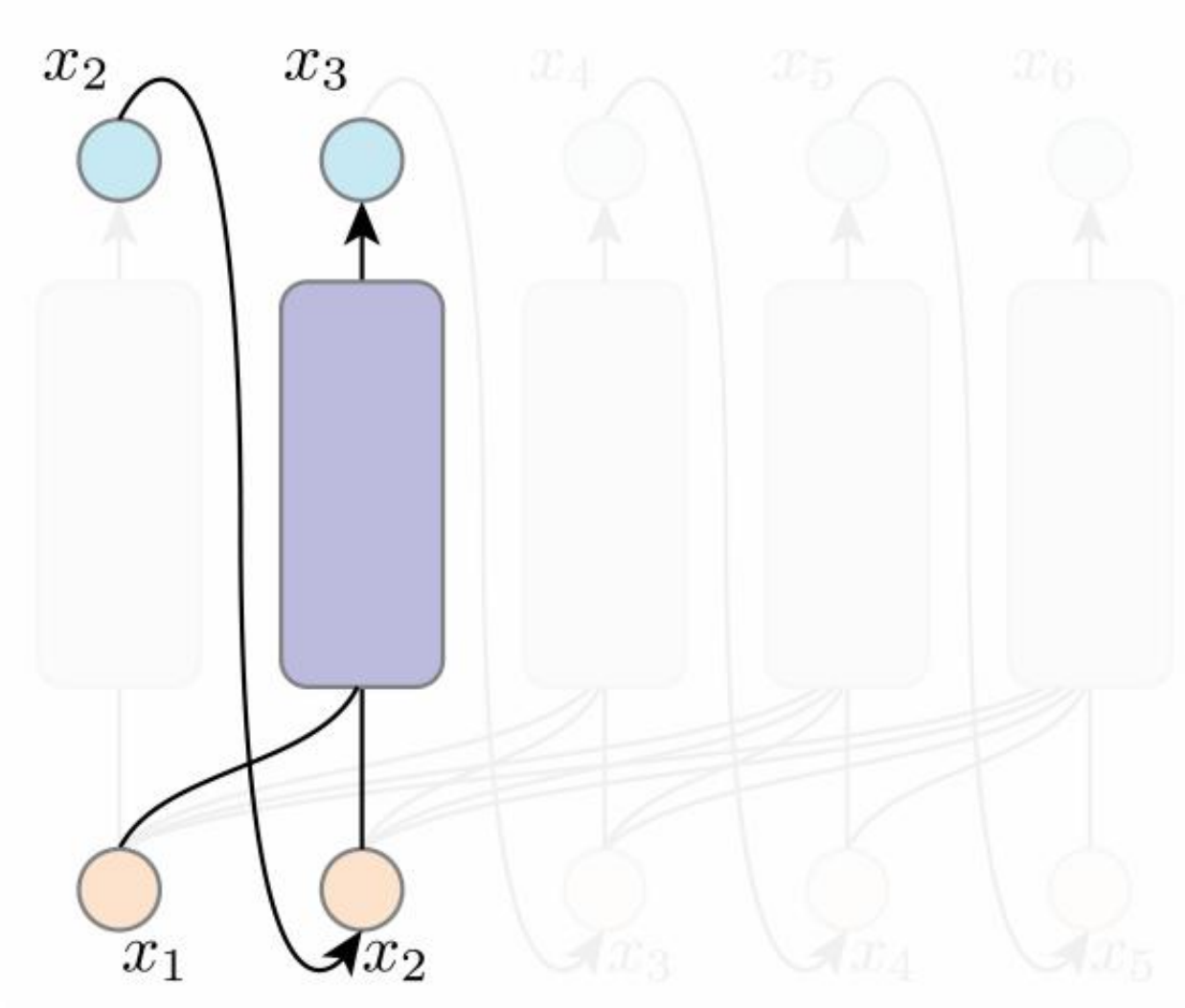
- This net models $p(x_2 \mid x_1)$
- 1 input
- 1 output



Inference: Autoregressive

- This net models $p(x_3 \mid x_{1,2})$
- **2 inputs**
- 1 output

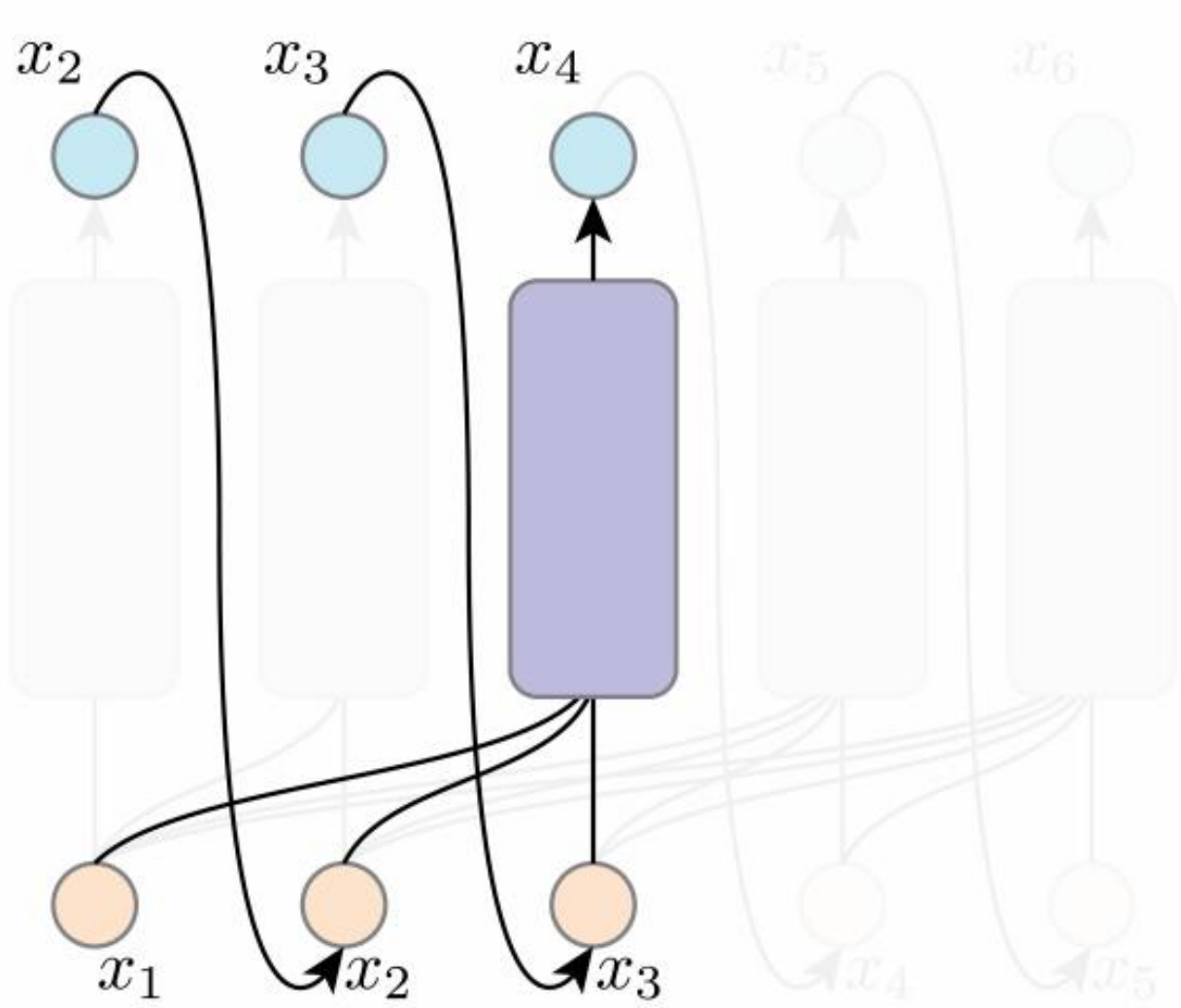
- inputs: outputs from previous steps



Inference: Autoregressive

- This net models $p(x_4 \mid x_{1,2,3})$
- **3 inputs**
- 1 output

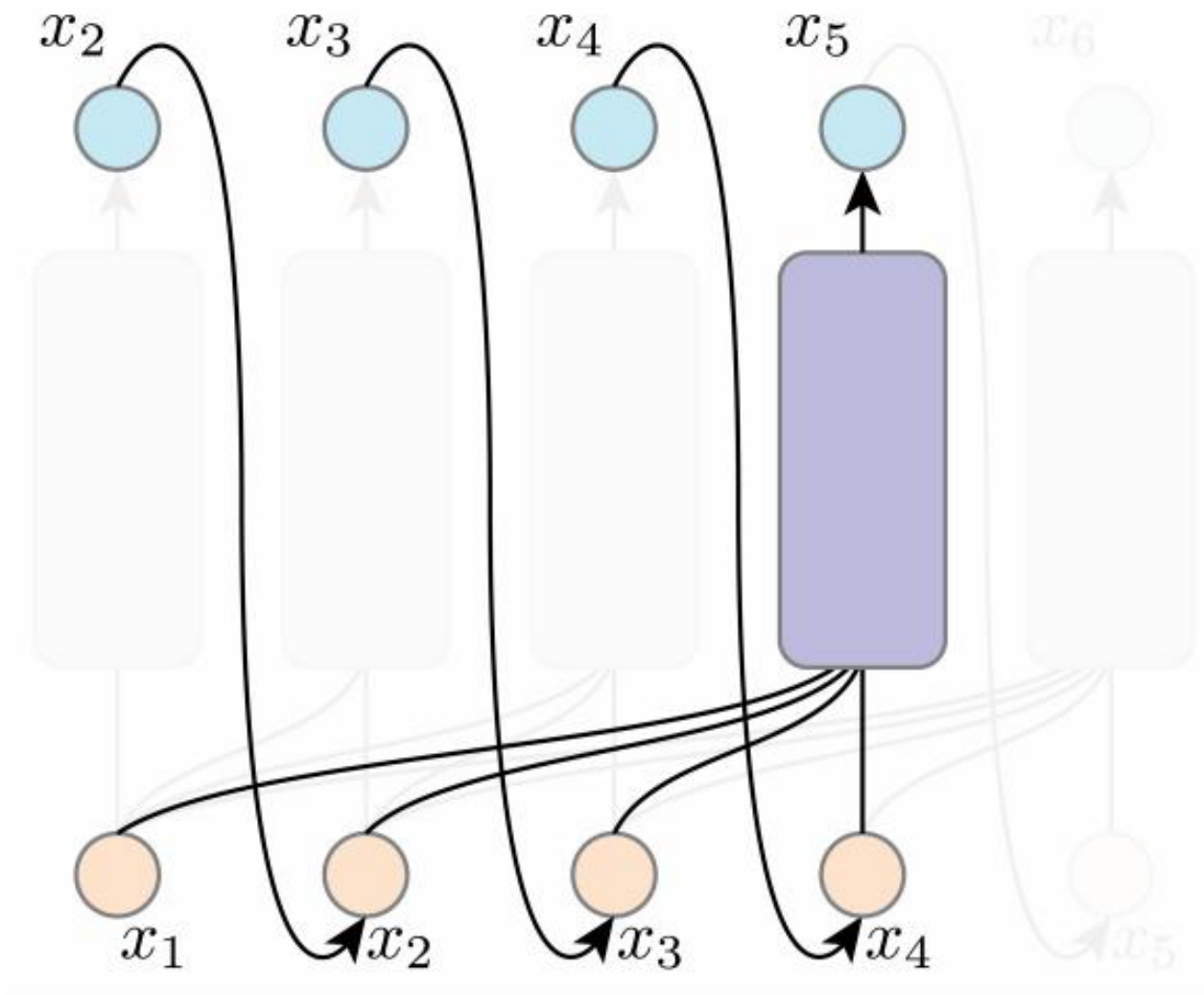
- inputs: outputs from previous steps



Inference: Autoregressive

- This net models $p(x_5 \mid x_{1,2,3,4})$
- **4 inputs**
- 1 output

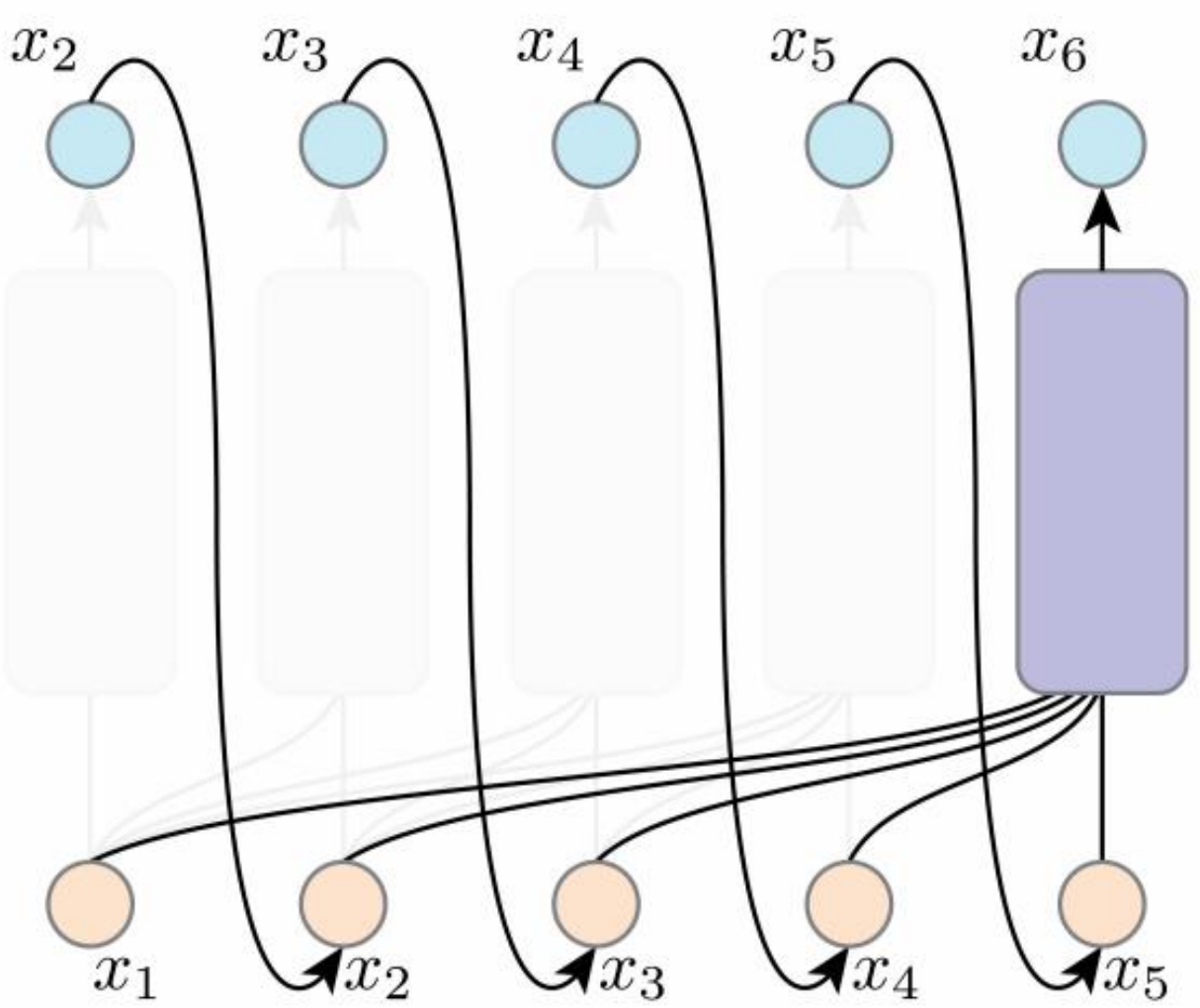
- inputs: outputs from previous steps



Inference: Autoregressive

- This net models $p(x_6 \mid x_{1,2,3,4,5})$
- **5 inputs**
- 1 output

- inputs: outputs from previous steps

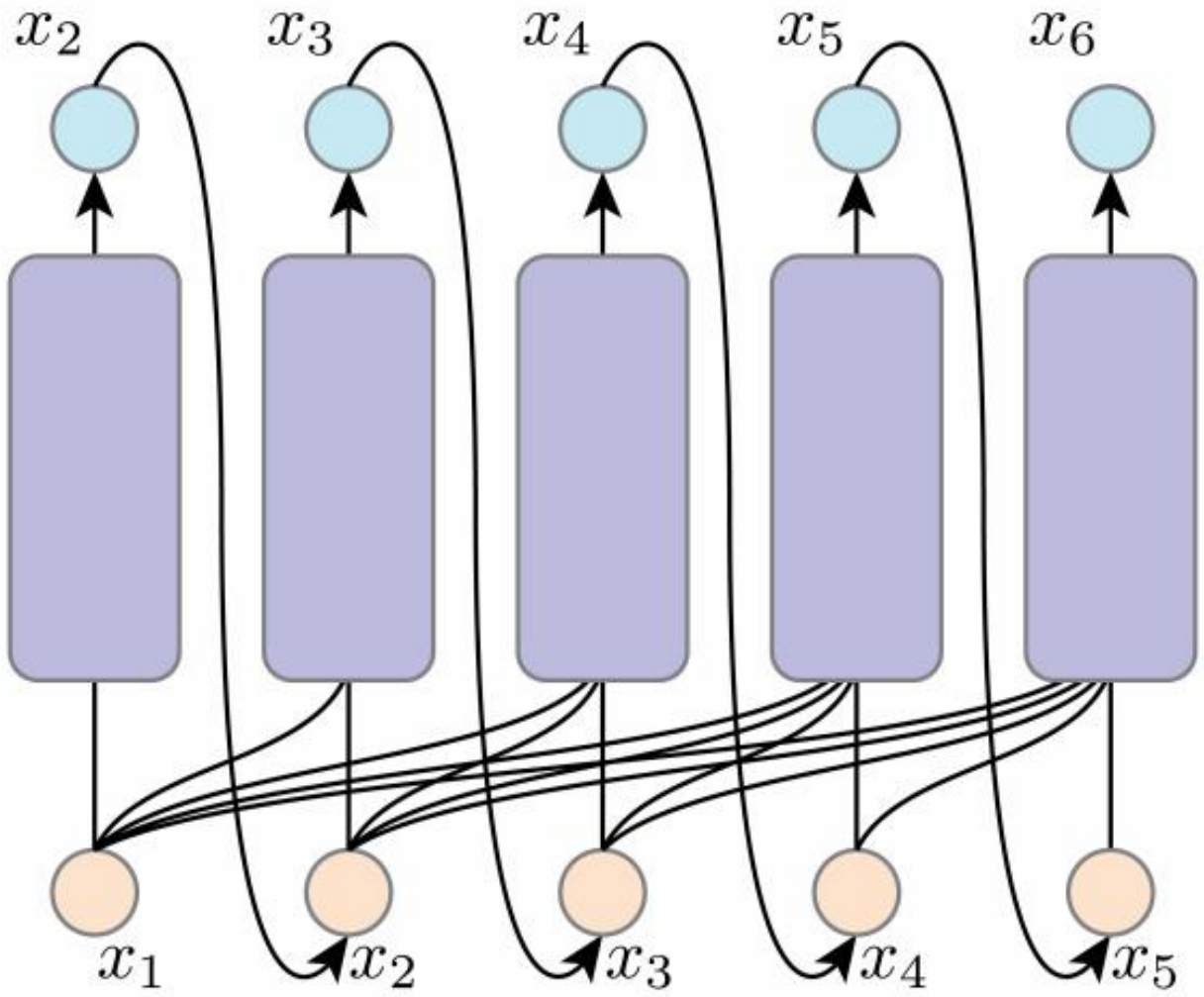


Inference: Autoregressive

Note:

- This is a **recursive** process

- but **not** necessarily done by RNN
- can be done by **any** architecture (e.g., CNN or Transformers)



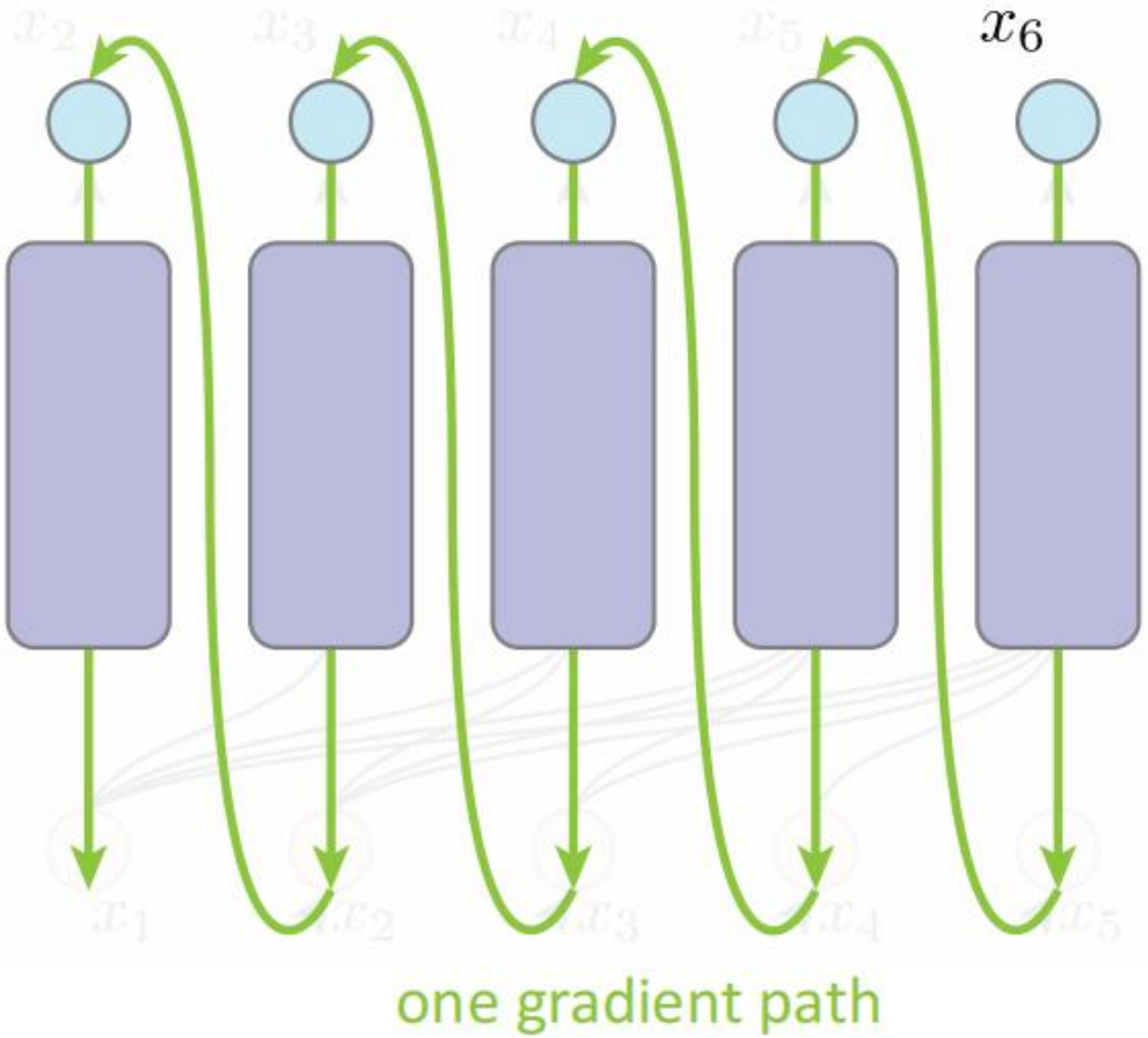
What if we backprop through this graph

Consider one gradient path of x_6:

- go through all previous outputs, ...
- all previous sampling ops, ...
- all previous networks

(e.g., each is a full Transformer)

It's **infeasible** to **train** the AR model following its **inference** graph.



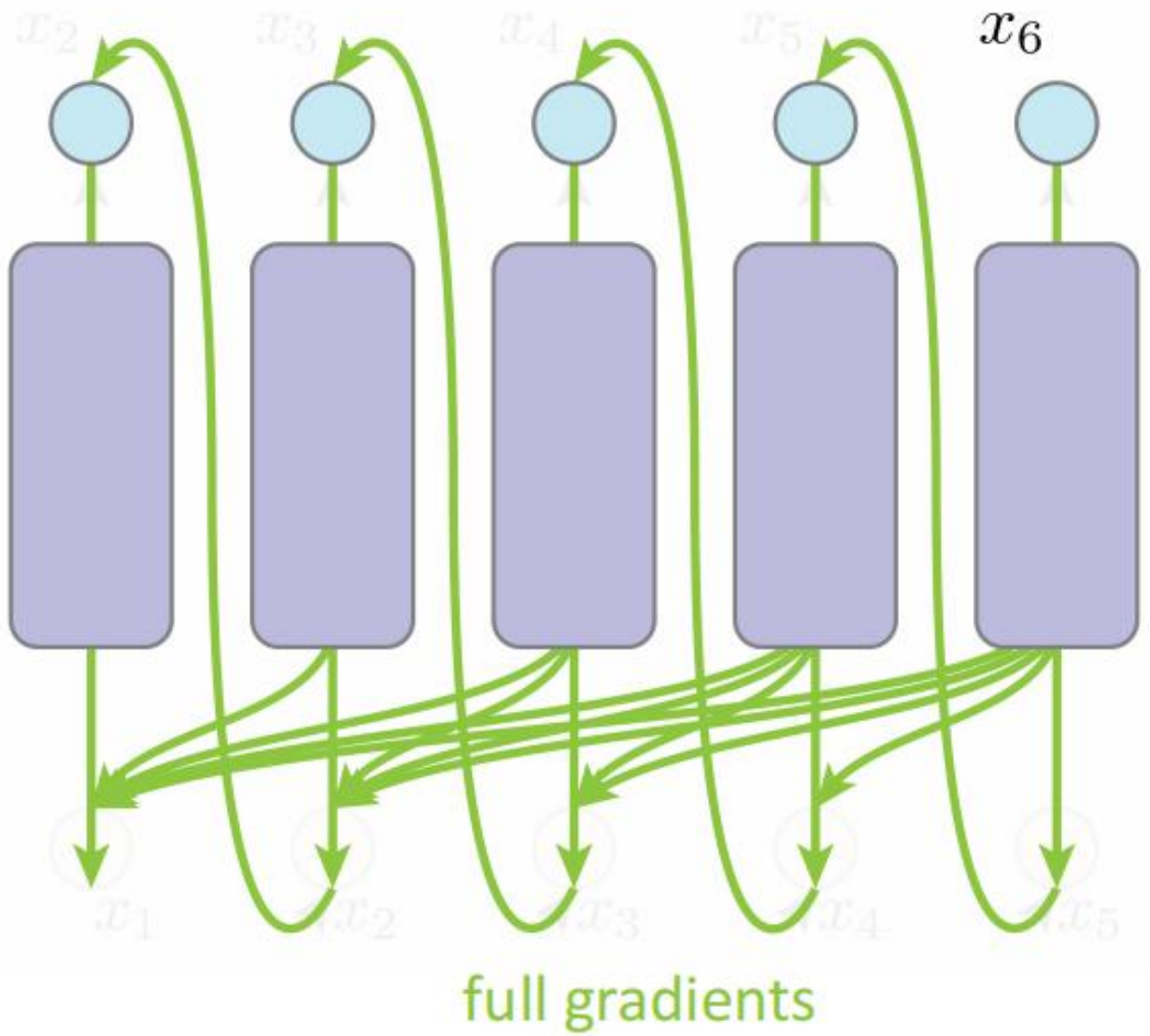
What if we backprop through this graph

Consider one gradient path of x_6:

- go through all previous outputs, ...
- all previous sampling ops, ...
- all previous networks

(e.g., each is a full Transformer)

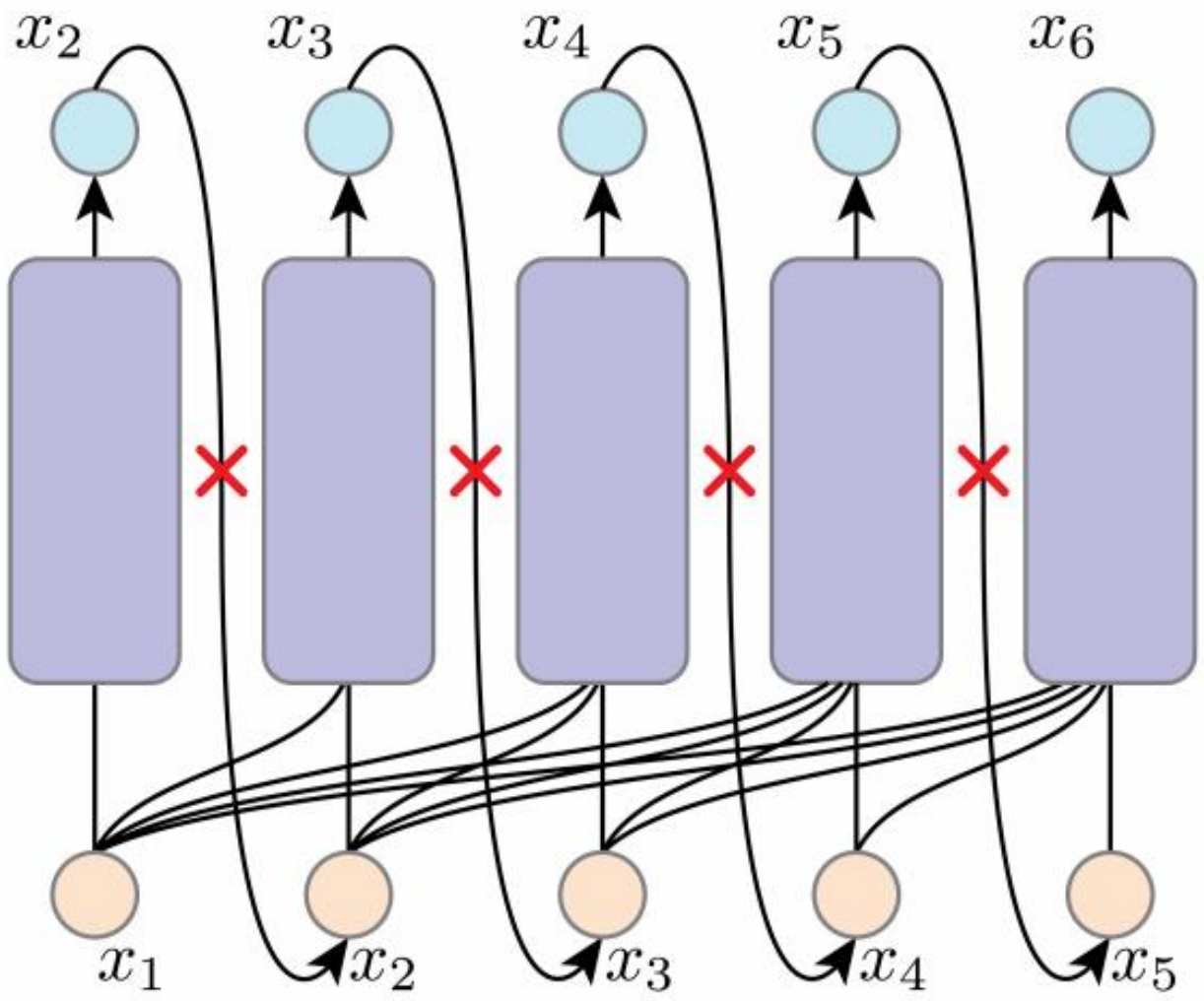
It's **infeasible** to **train** the AR model following its **inference** graph.



Training: Teacher-Forcing

Teacher-forcing

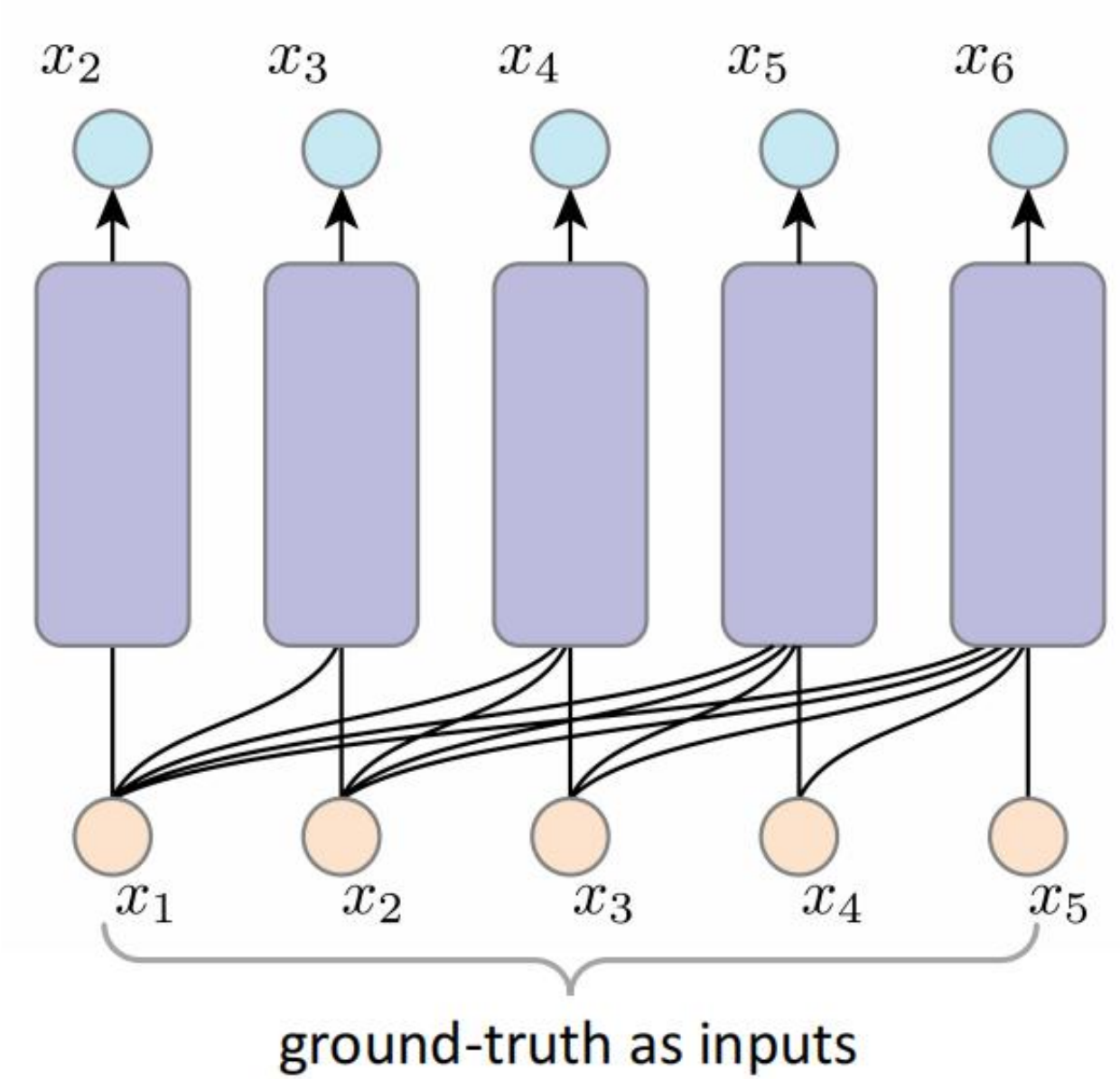
- Inputs are not from previous outputs
- Inputs are from ground-truth data



Training: Teacher-Forcing

Teacher-forcing

- Inputs are not from previous outputs
- Inputs are from ground-truth data



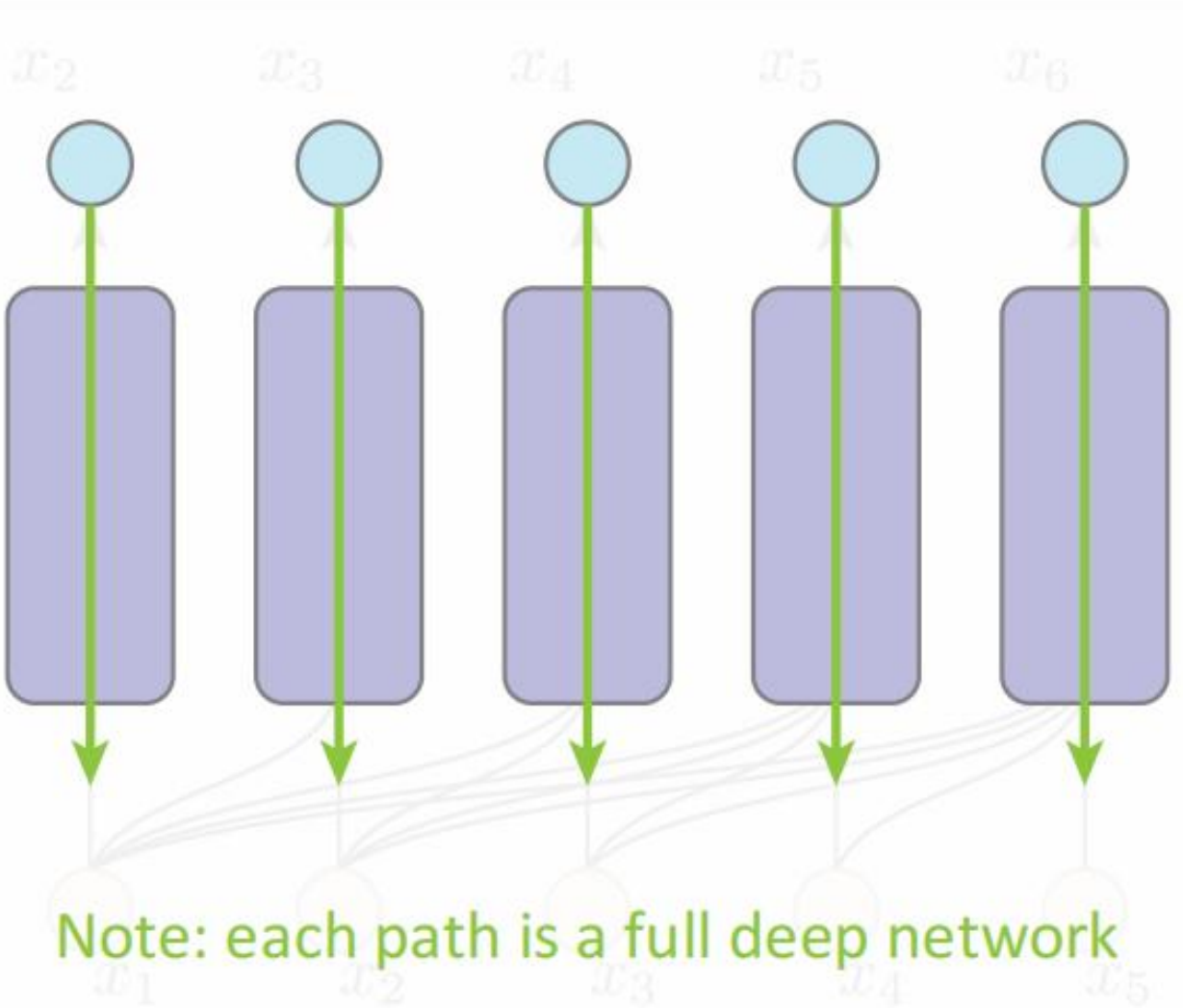
Training: Teacher-Forcing

Teacher-forcing

- Inputs are not from previous outputs
- Inputs are from ground-truth data

Pros:

- backprop path is much shorter



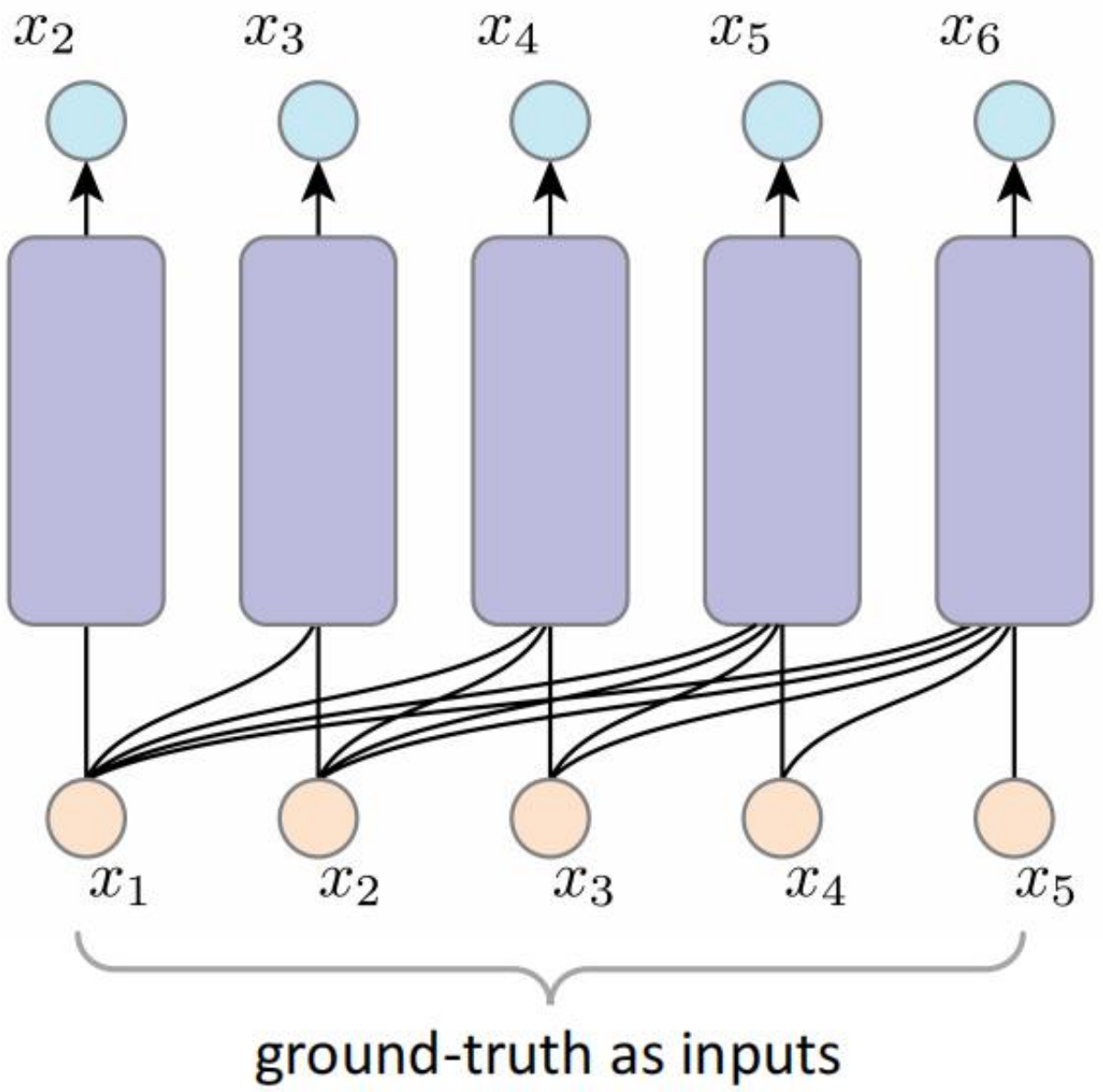
Training: Teacher-Forcing

Teacher-forcing

- Inputs are not from previous outputs
- Inputs are from ground-truth data

Pros:

- backprop path is much shorter
- ground-truth inputs can ease training



Training: Teacher-Forcing

Teacher-forcing

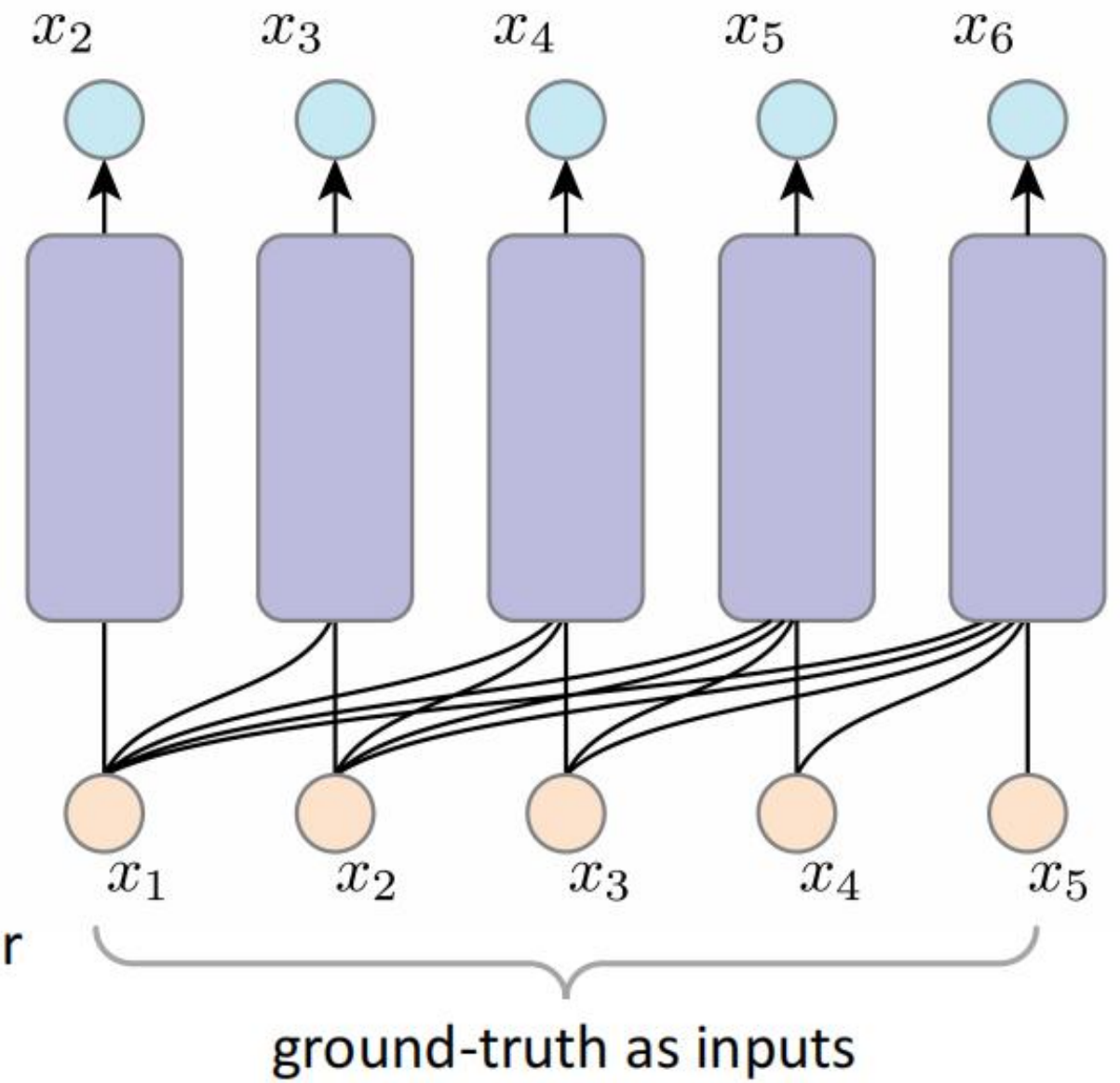
- Inputs are not from previous outputs
- Inputs are from ground-truth data

Pros:

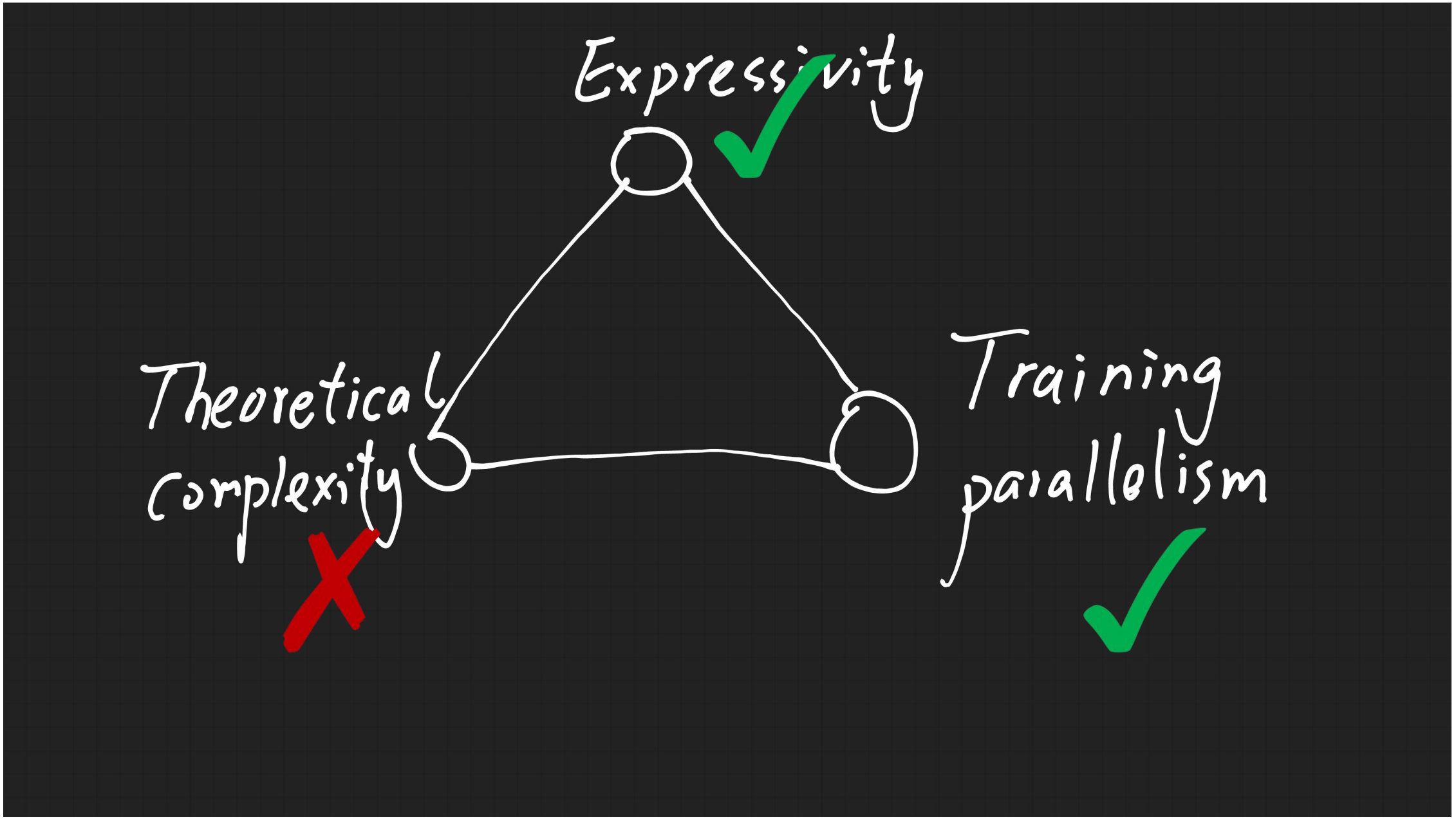
- backprop path is much shorter
- ground-truth inputs can ease training

Cons:

- inconsistent training/inference
- distribution shift: can't see its own error



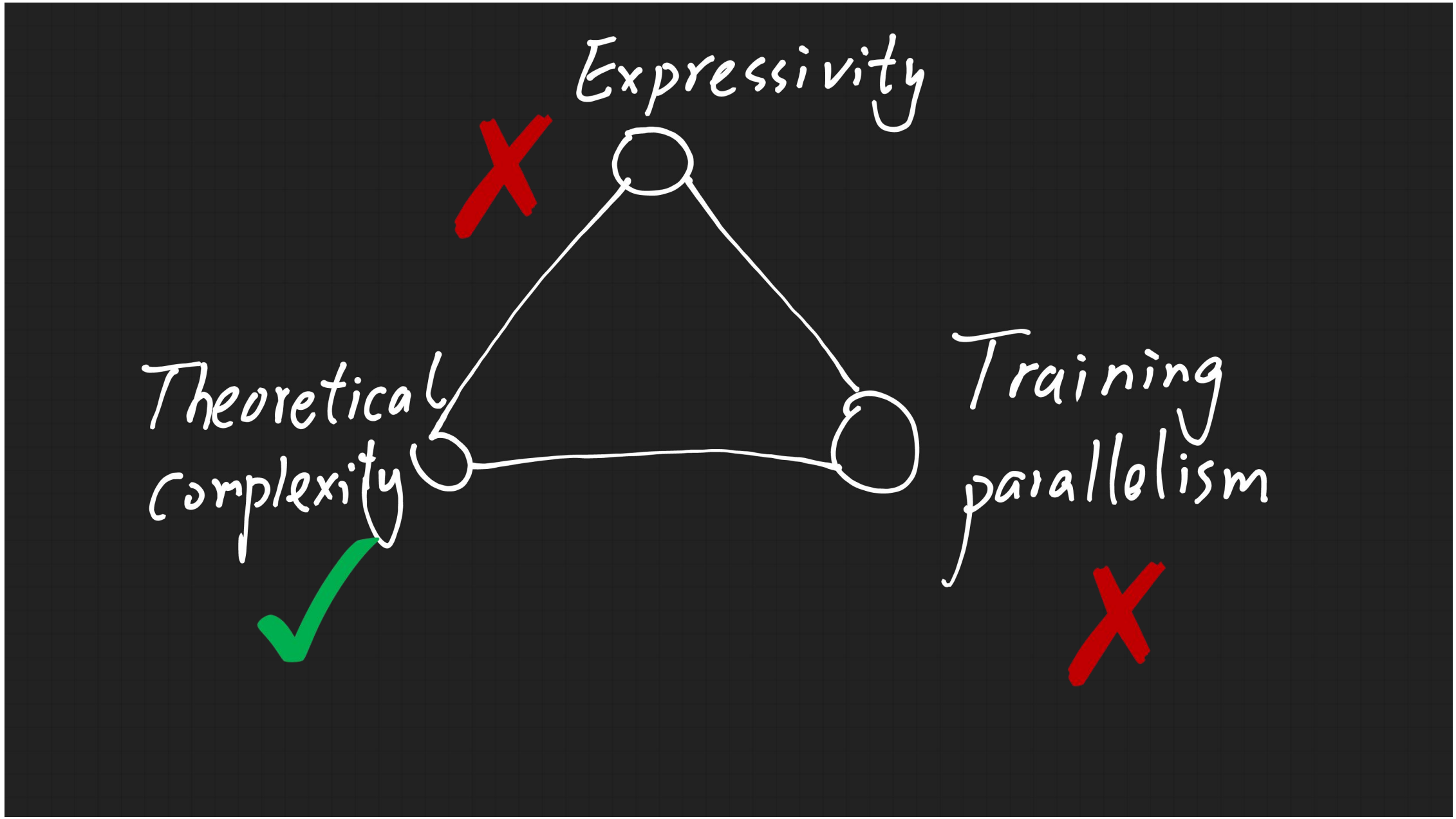
不可能三角：Transformer是这样



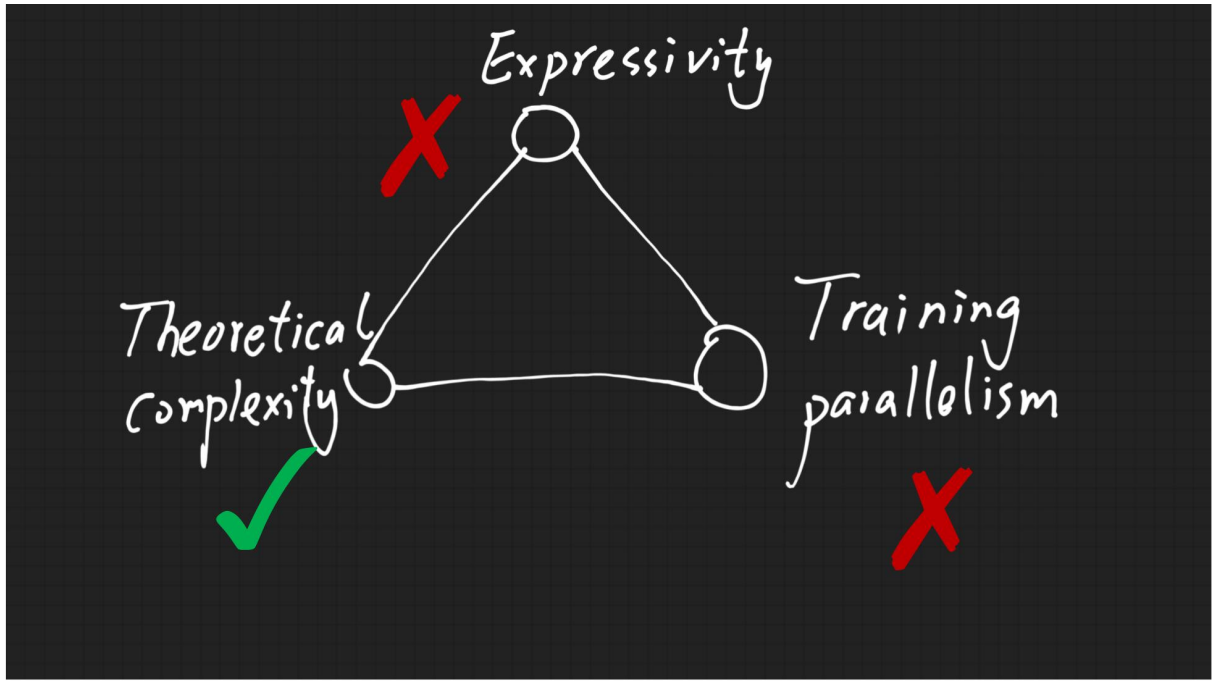
本节内容

CONTENTS

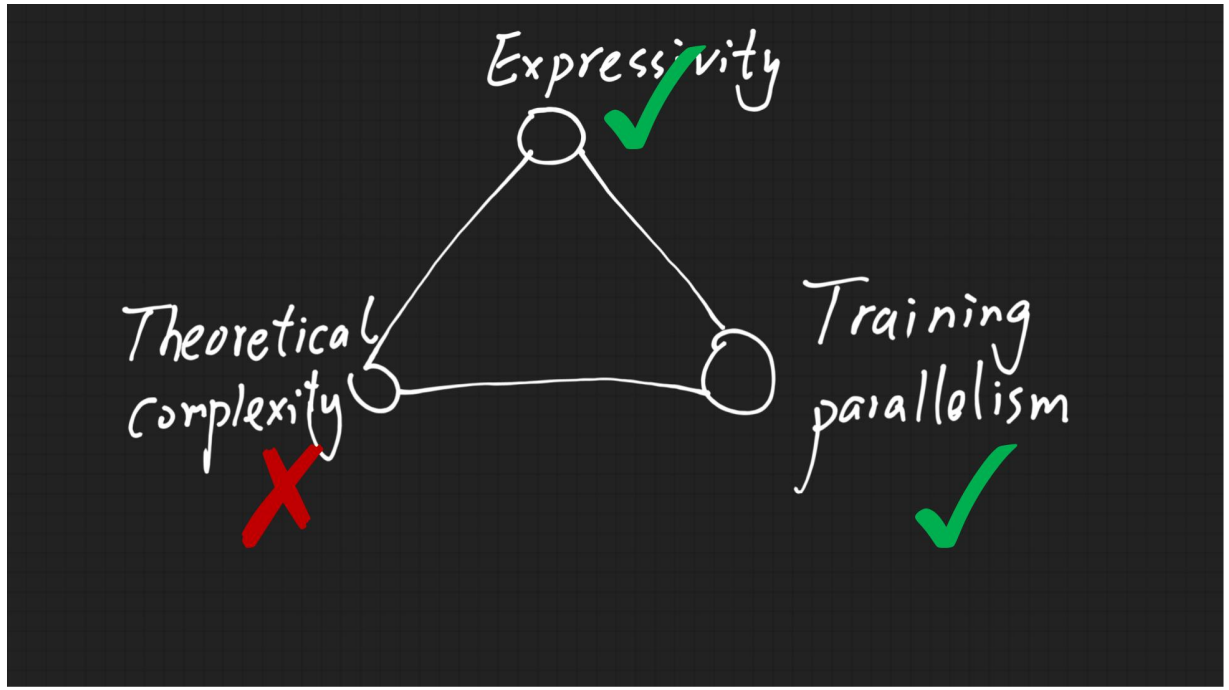
原始RNN


Expressivity
Theoretical complexity
Training parallelism

原始RNN

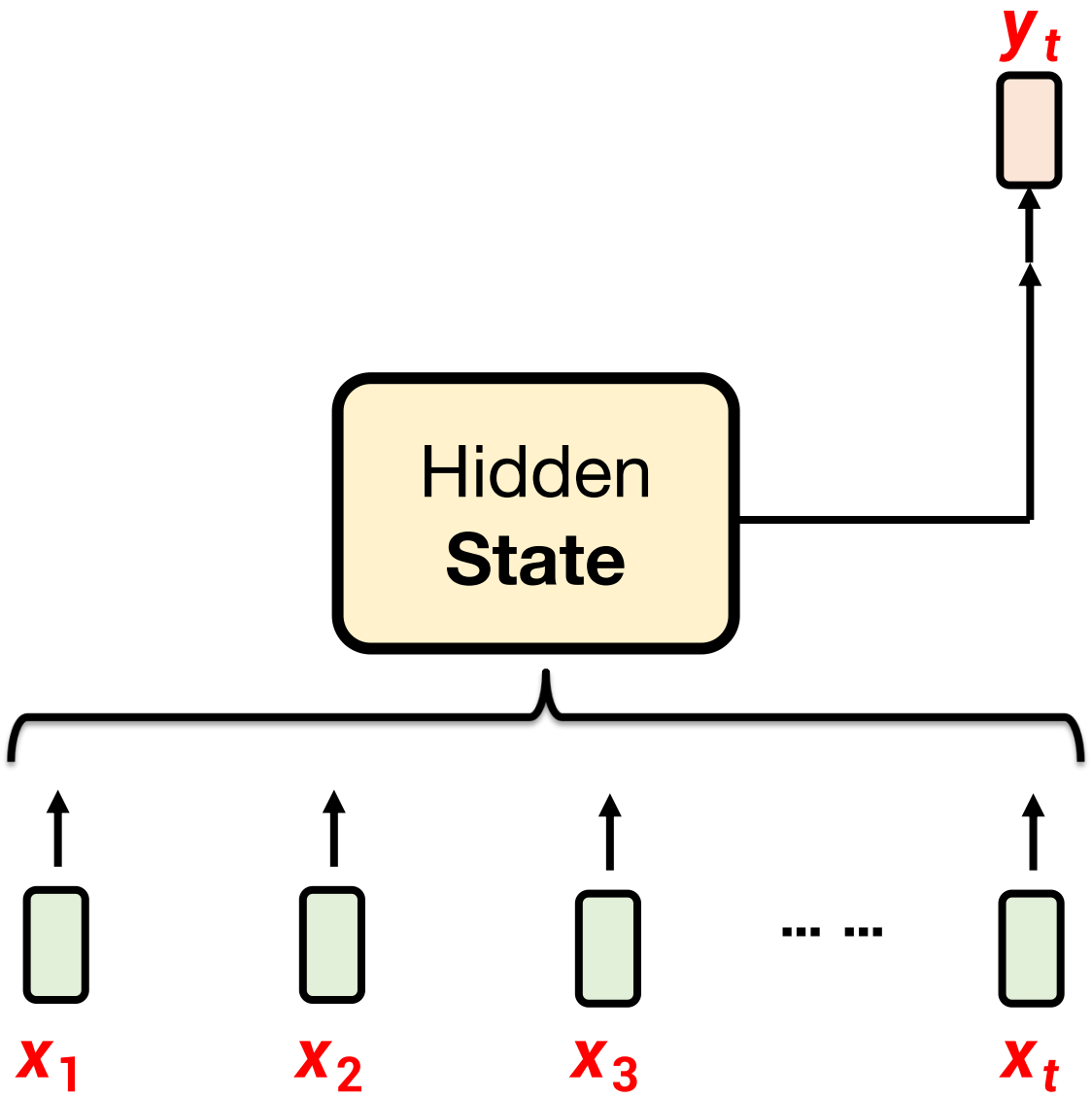


原始RNN



Transformer

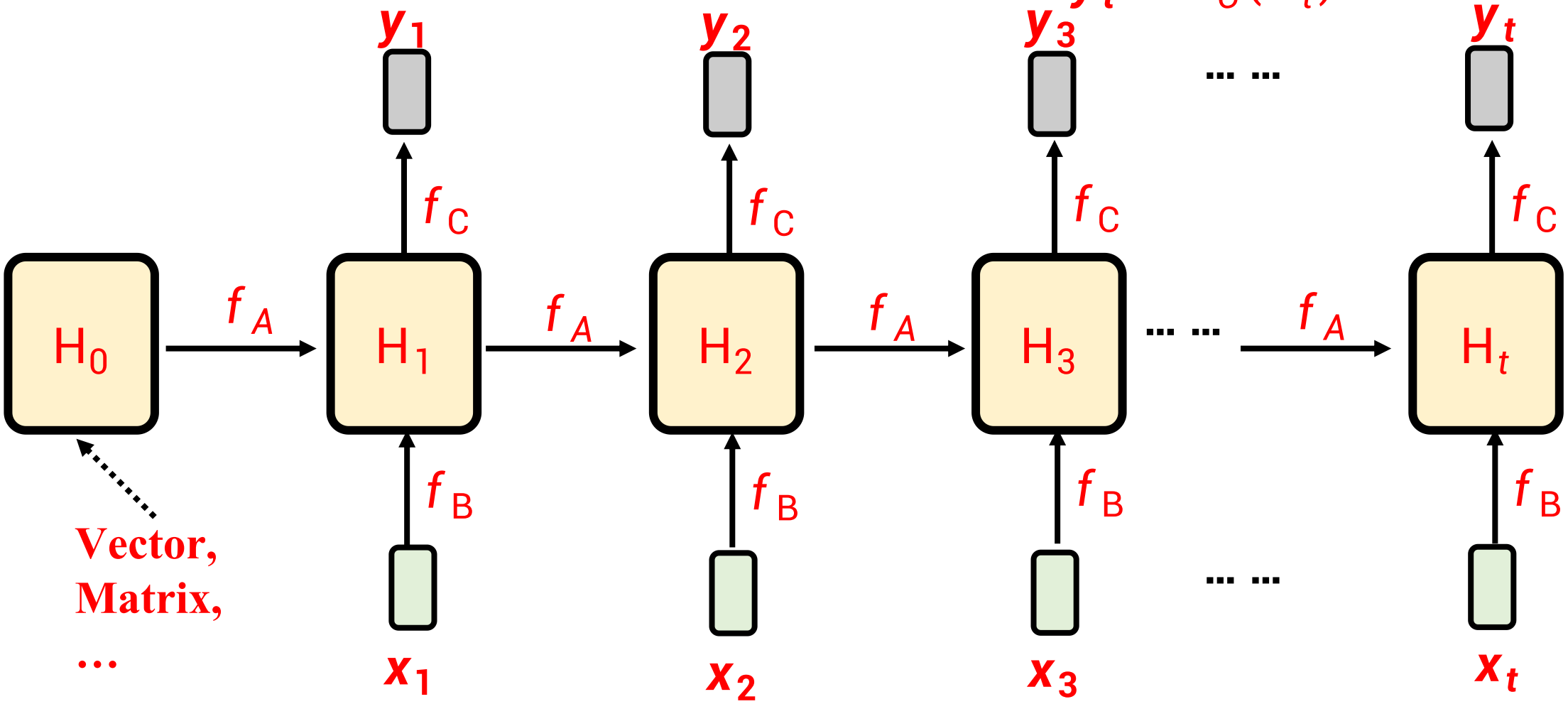
RNN-Style



RNN-Style

$$H_t = f_A(H_{t-1}) + f_B(x_t)$$

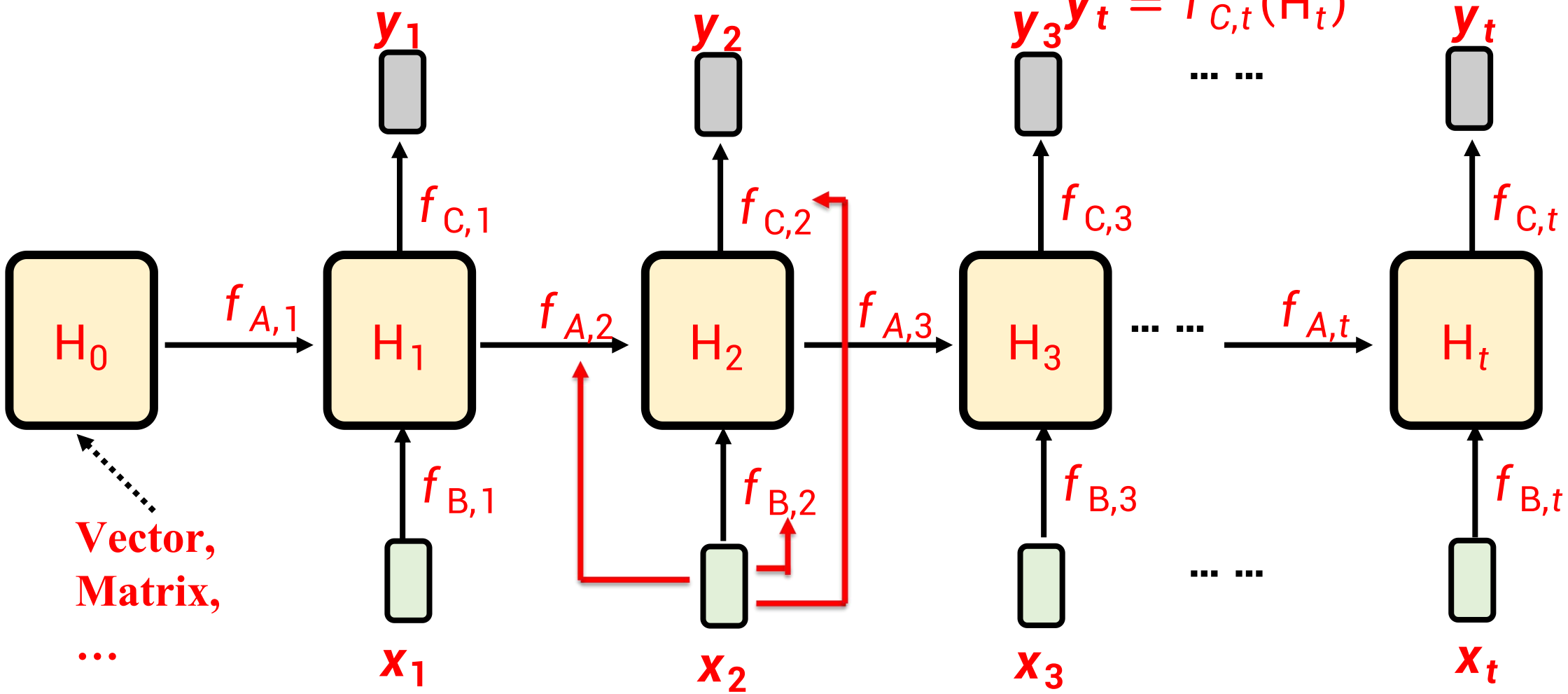
$$y_t = f_C(H_t)$$



RNN-Style

$$H_t = f_{A,t}(H_{t-1}) + f_{B,t}(\boldsymbol{x}_t)$$

$$\boldsymbol{y}_t = f_{C,t}(H_t)$$

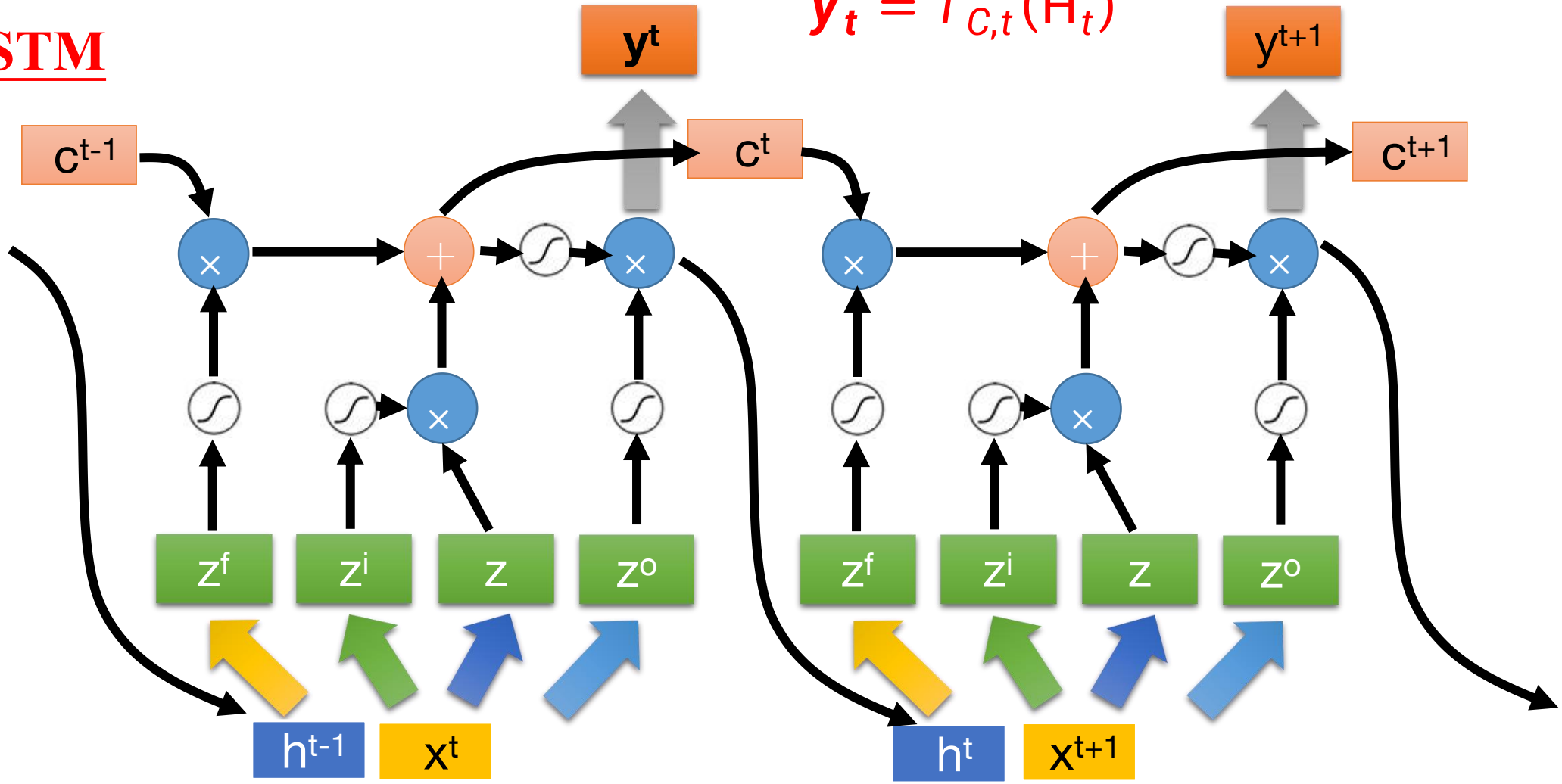


RNN-Style

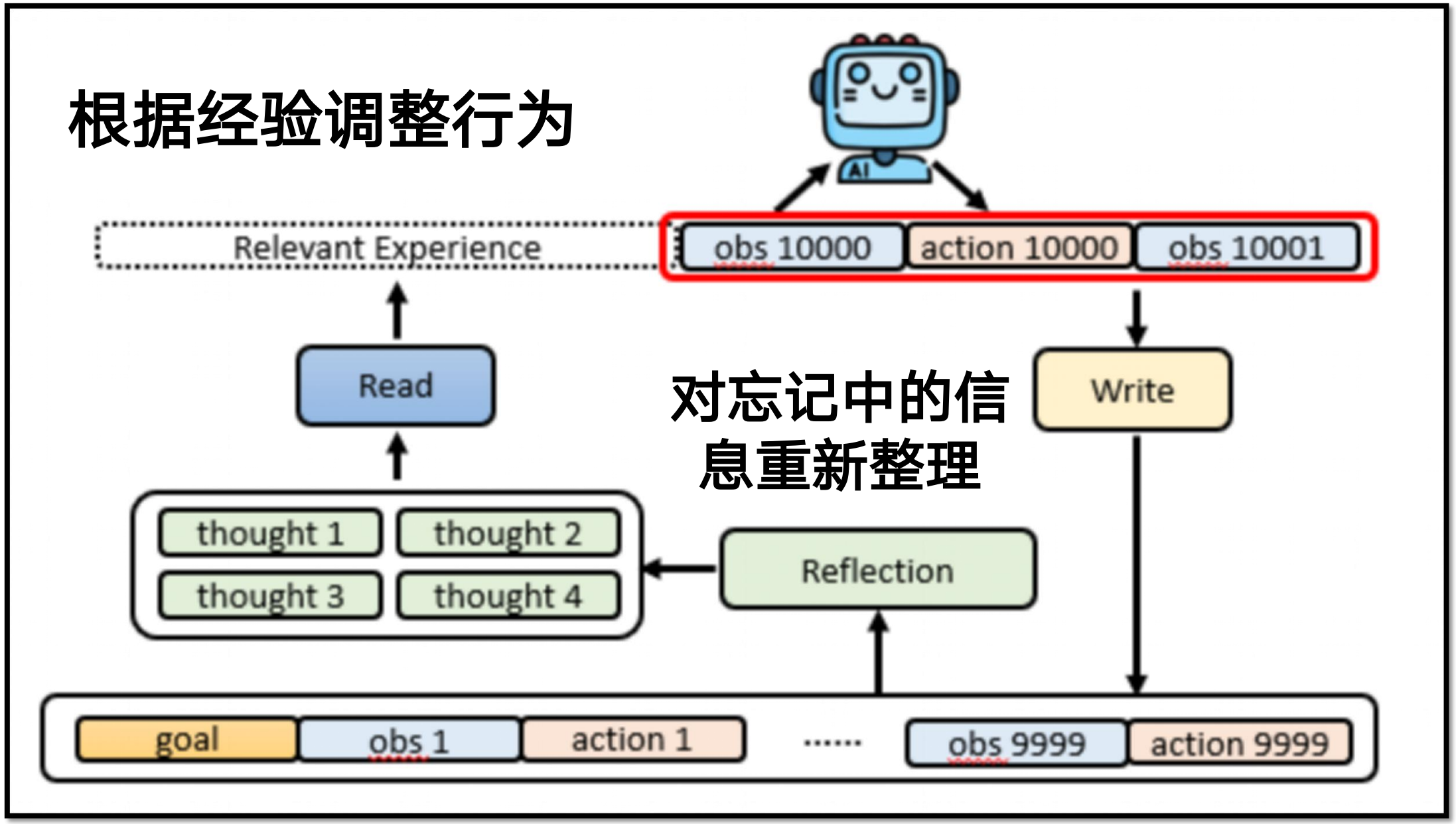
$$H_t = f_{A,t}(H_{t-1}) + f_{B,t}(\boldsymbol{x}_t)$$

$$\boldsymbol{y}_t = f_{C,t}(H_t)$$

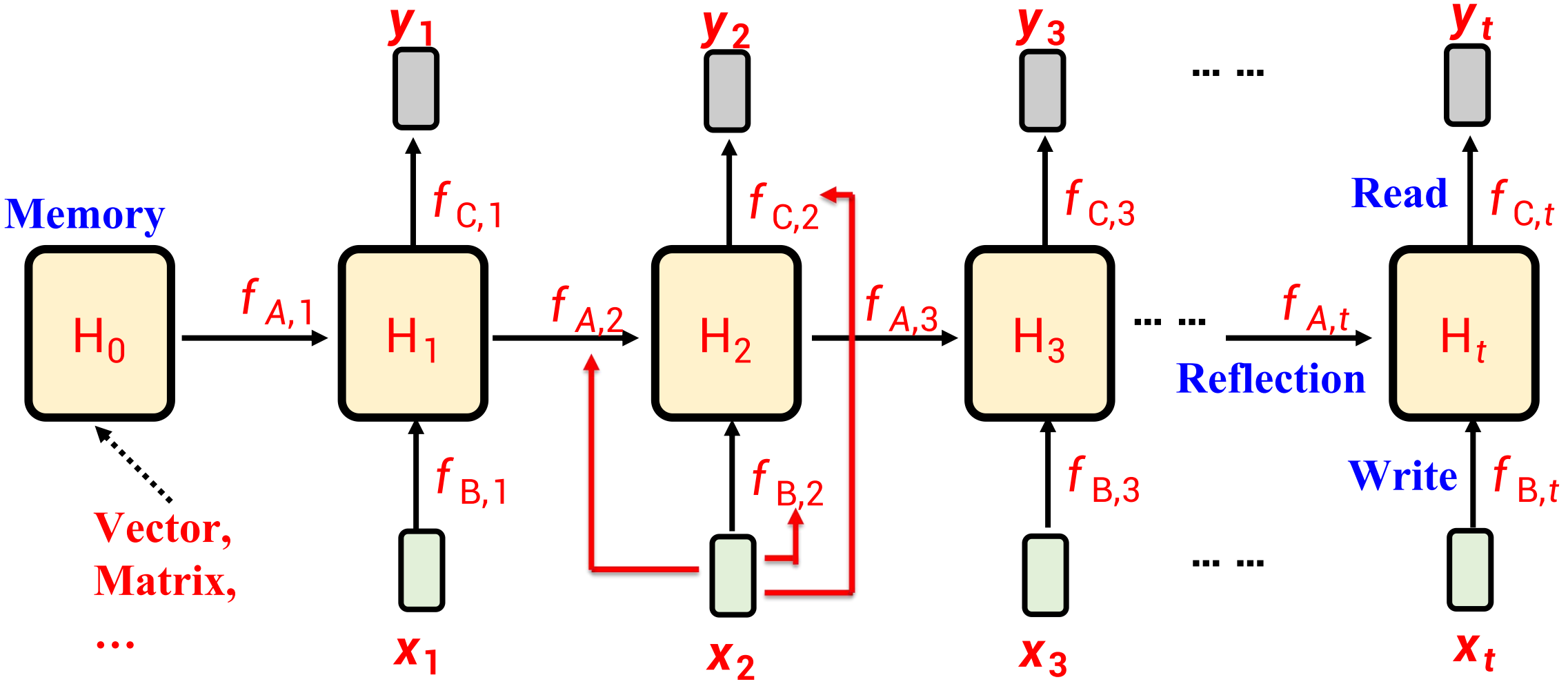
LSTM



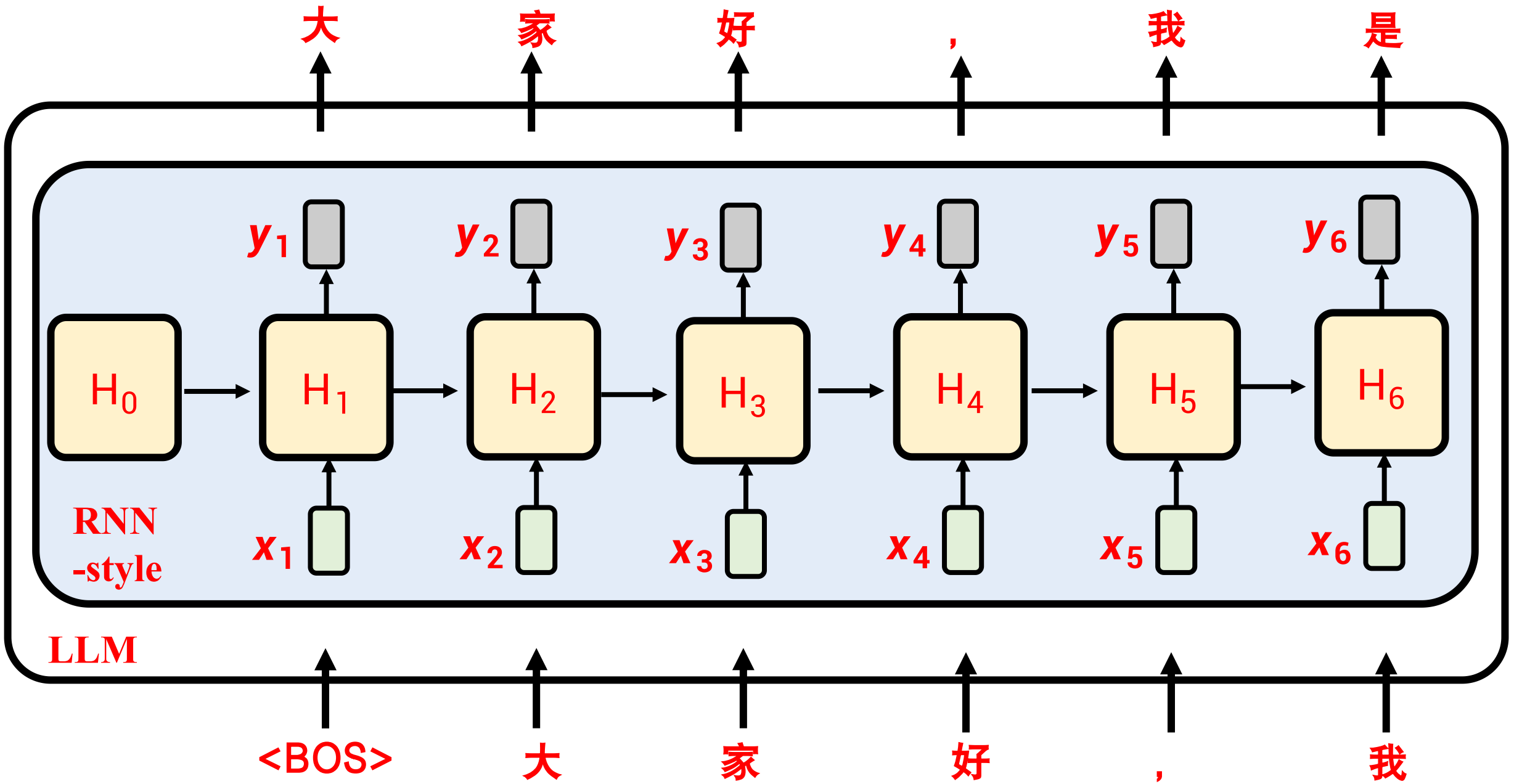
RNN-Style vs. AI Agent’s Memory



RNN-Style vs. AI Agent's Memory

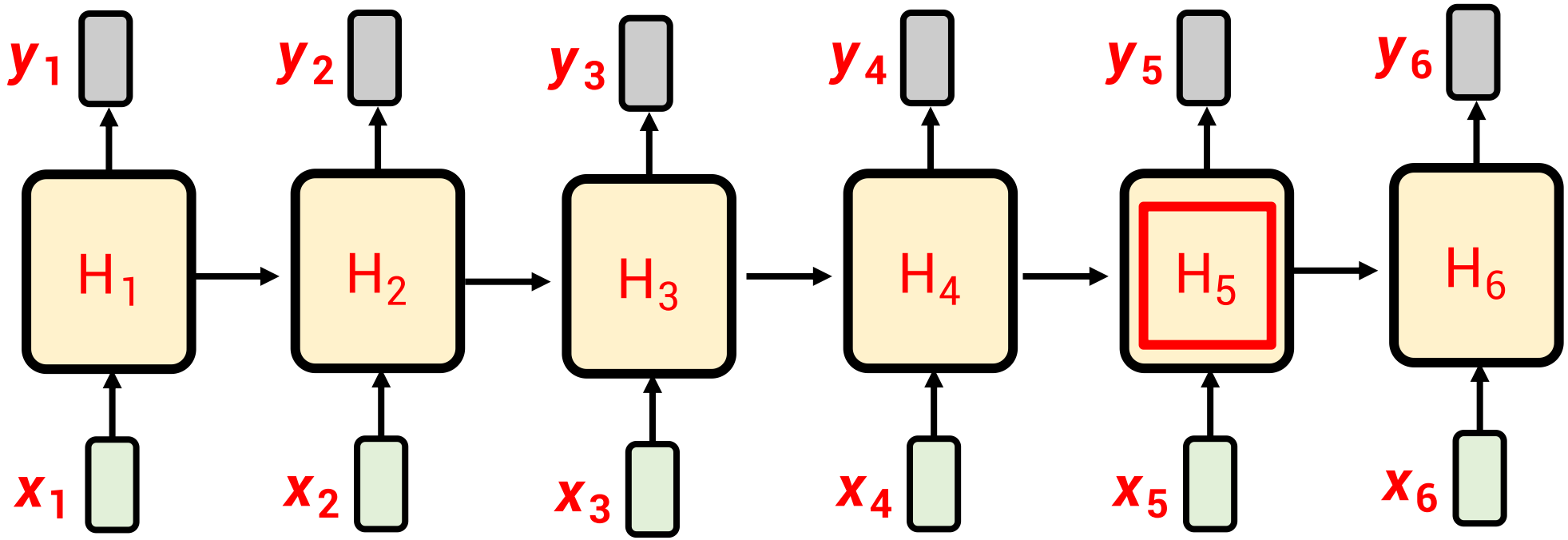


RNN-Style


大
家
好
，
我
是
y_1
y_2
y_3
y_4
y_5
y_6
H_0
H_1
H_2
H_3
H_4
H_5
H_6
RNN
-style
x_1
x_2
x_3
x_4
x_5
x_6
LLM
<BOS>
大
家
好
，
我

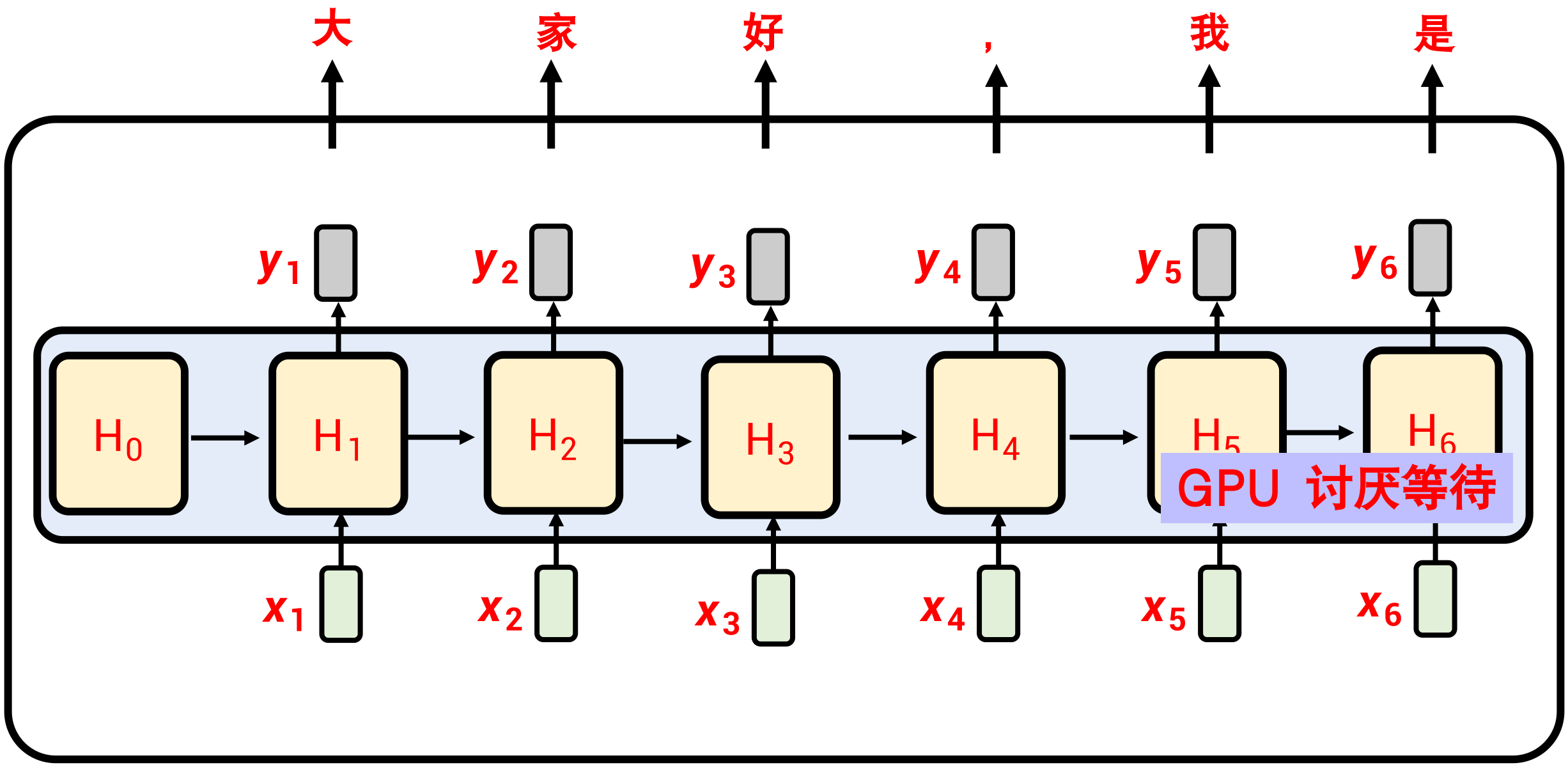
RNN Style (Expressivity)

每一步运算量
都一样



RNN 没办法记大量信息?

RNN无法并行


大
家
好
，
我
是
y1
y2
y3
y4
y5
y6
H0
H1
H2
H3
H4
H5
H6
GPU 讨厌等待
x1
x2
x3
x4
x5
x6

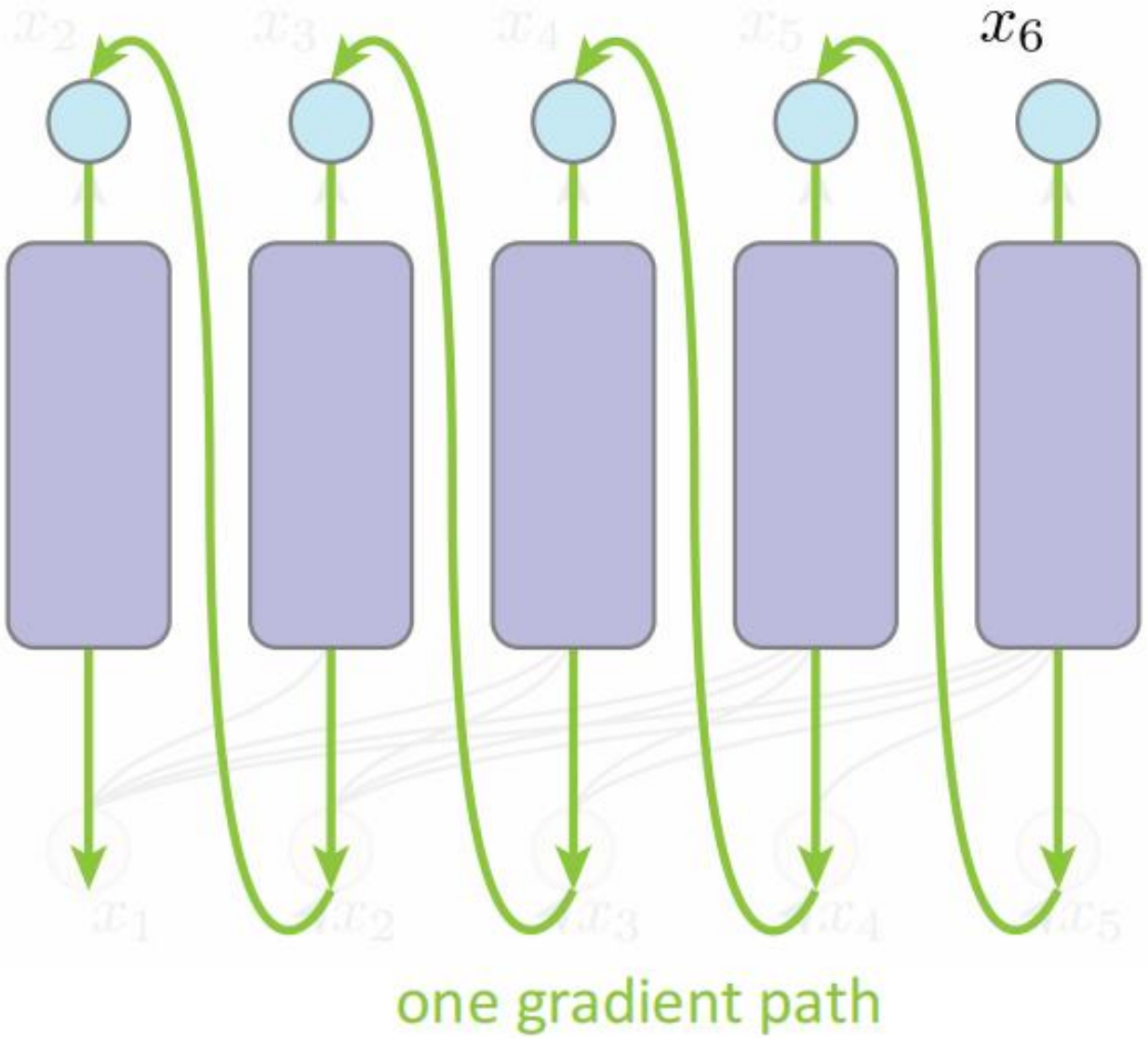
What if we backprop through this graph

Consider one gradient path of x_6:

- go through all previous outputs, ...
- all previous sampling ops, ...
- all previous networks

(e.g., each is a full Transformer)

It's **infeasible** to **train** the AR model following its **inference** graph.



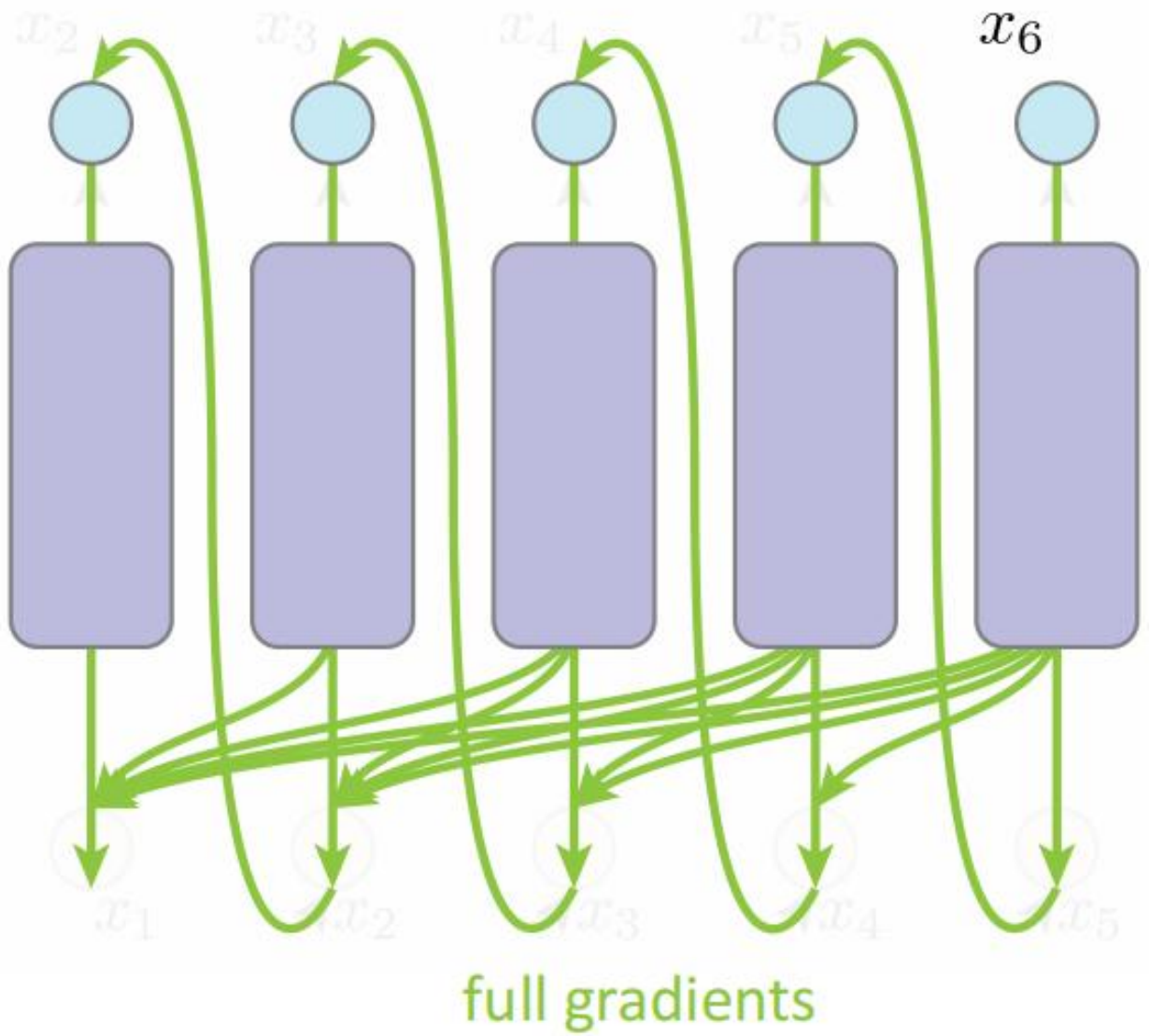
What if we backprop through this graph

Consider one gradient path of x_6:

- go through all previous outputs, ...
- all previous sampling ops, ...
- all previous networks

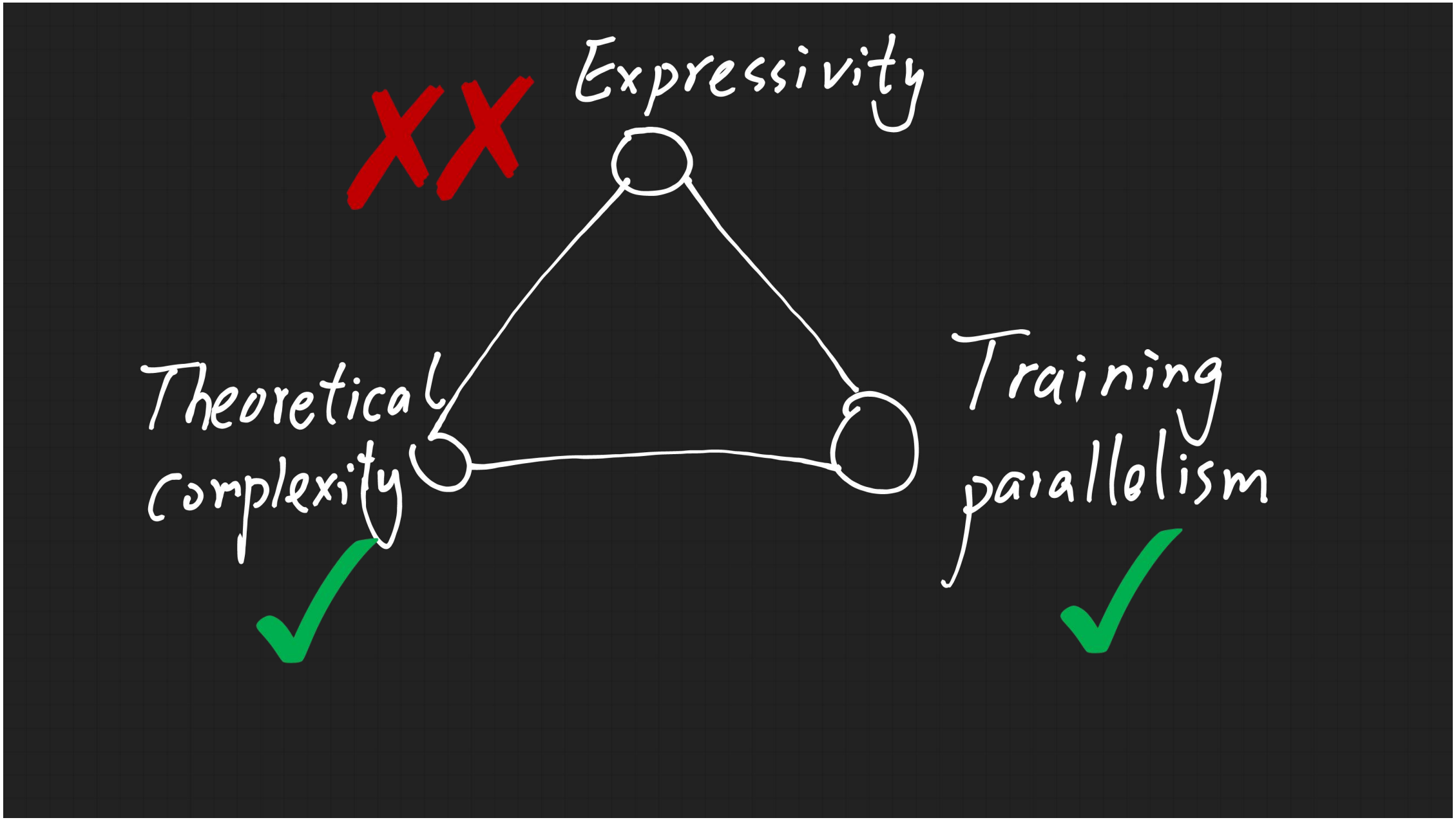
(e.g., each is a full Transformer)

It's **infeasible** to **train** the AR model following its **inference** graph.

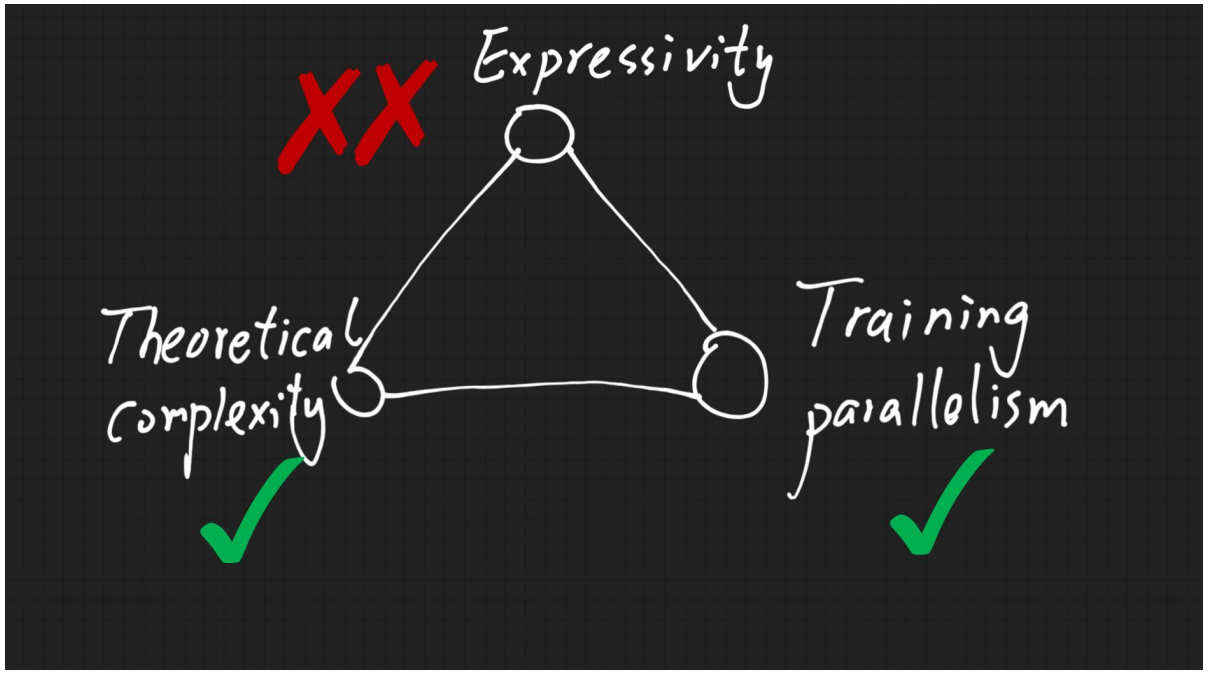


代表1： Linear Attention

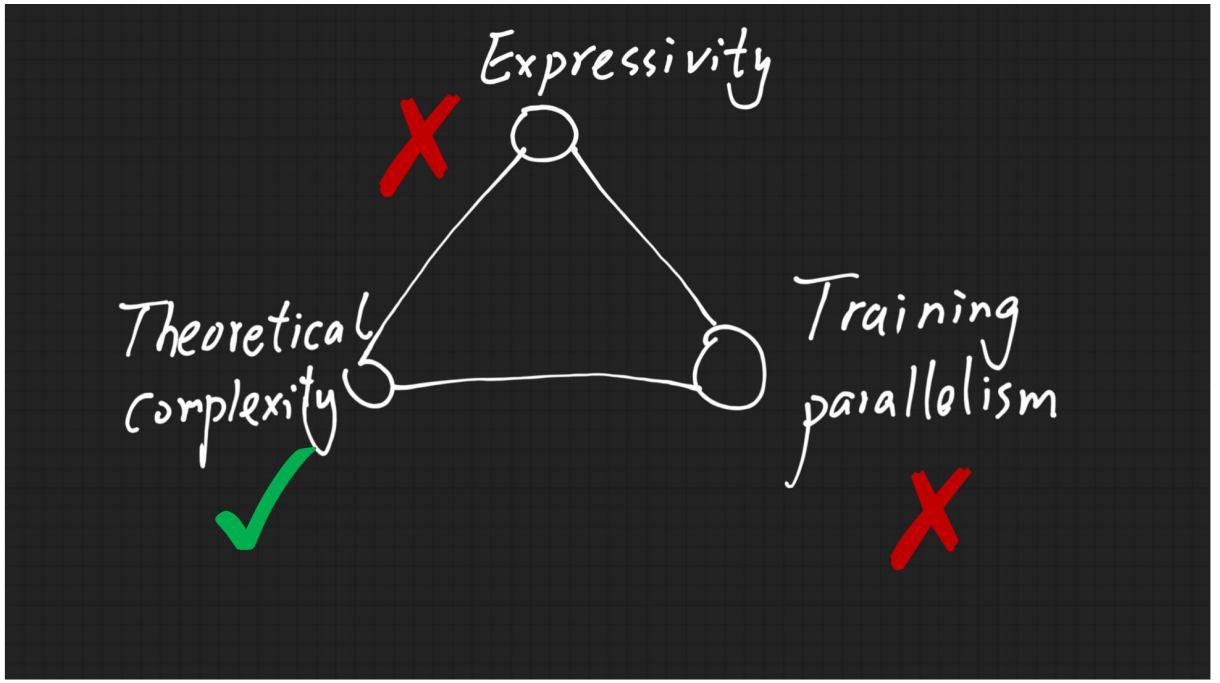
Linear Attention



Linear Attention



Linear Attention
Linear RNN



non-Linear RNN

RNN 有没有训练时并行的可能性

$$H_t = f_{A,t}(H_{t-1}) + f_{B,t}(\boldsymbol{x_t})$$

$$\boldsymbol{y_t} = f_{C,t}(H_t)$$

$$f_{A,1}(H_0) = O$$

$$H_1 = f_{A,1}(H_0) + f_{B,1}(\boldsymbol{x_1}) \quad = f_{B,1}(\boldsymbol{x_1})$$

$$H_2 = f_{A,2}(H_1) + f_{B,2}(\boldsymbol{x_2}) \quad = f_{A,2}(f_{B,1}(\boldsymbol{x_1})) + f_{B,2}(\boldsymbol{x_2})$$

$$H_3 = f_{A,3}(H_2) + f_{B,3}(\boldsymbol{x_3}) \quad = f_{A,3}\left(f_{A,2}(f_{B,1}(\boldsymbol{x_1})) + f_{B,2}(\boldsymbol{x_2})\right) + f_{B,3}(\boldsymbol{x_3})$$

$$\vdots$$

$$H_t = f_{A,t}(H_{t-1}) + f_{B,t}(\boldsymbol{x_t}) \quad = \underbrace{f_{A,t}(f_{A,t-1}\ldots f_{A,3}(f_{A,2}}(f_{B,1}(\boldsymbol{x_1})\ldots) \qquad \ldots + f_{B,t}(\boldsymbol{x_t})$$

RNN 有没有训练时并行的可能性

$$f_{A,1}(H_0) = O$$

$$H_t = H_{t-1} + f_{B,t}(\boldsymbol{x_t})$$

$$\boldsymbol{y_t} = f_{C,t}(H_t)$$

H_t **is a** $d \times d$ **matric**

$$f_{B,t}(\boldsymbol{x_t}) = D_t$$

$$\begin{cases} H_1 = H_0 + f_{B,1}(\boldsymbol{x_1}) & = f_{B,1}(\boldsymbol{x_1}) \\ H_2 = H_1 + f_{B,2}(\boldsymbol{x_2}) & = f_{B,1}(\boldsymbol{x_1}) + f_{B,2}(\boldsymbol{x_2}) \\ H_3 = H_2 + f_{B,3}(\boldsymbol{x_3}) & = f_{B,1}(\boldsymbol{x_1}) + f_{B,2}(\boldsymbol{x_2}) + f_{B,3}(\boldsymbol{x_3}) \\ \vdots & \\ H_t = H_{t-1} + f_{B,t}(\boldsymbol{x_t}) & = f_{B,1}(\boldsymbol{x_1}) + f_{B,2}(\boldsymbol{x_2}) + f_{B,3}(\boldsymbol{x_3}) \quad \cdots\cdots + f_{B,t}(\boldsymbol{x_t}) \end{cases}$$

RNN 有没有训练时并行的可能性

$$f_{A,1}(\mathrm{H}_0) = O$$

$$\begin{cases} \mathrm{H}_1 = D_1 & \boldsymbol{y_1} = D_1\boldsymbol{q_1} \\ \mathrm{H}_2 = D_1 + D_2 & \boldsymbol{y_2} = D_1\boldsymbol{q_2} + D_2\boldsymbol{q_2} \\ \mathrm{H}_3 = D_1 + D_2 + D_3 & \boldsymbol{y_3} = D_1\boldsymbol{q_3} + D_2\boldsymbol{q_3} + D_3\boldsymbol{q_3} \\ \vdots & \\ \mathrm{H}_t = D_1 + D_2 + \ldots + D_t & \boldsymbol{y_t} = D_1\boldsymbol{q_t} + D_2\boldsymbol{q_t} + \ldots + D_t\boldsymbol{q_t} \end{cases}$$

$$\mathrm{H}_t = \mathrm{H}_{t-1} + f_{B,t}(\boldsymbol{x_t})$$

$$\boldsymbol{y_t} = f_{C,t}(\mathrm{H}_t)$$

H_t **is a** $d \times d$ **matric**

$$f_{B,t}(\boldsymbol{x_t}) = D_t$$

$$f_{C,t}(\mathrm{H}_t) = \mathrm{H}_t\boldsymbol{q_t}$$

$$\boldsymbol{q_t} = W_Q\boldsymbol{x_t}$$

RNN 有没有训练时并行的可能性

$$f_{A,1}(\mathrm{H}_0) = O$$

$$\begin{cases} \boldsymbol{y_1} = D_1\boldsymbol{q_1} \\ \boldsymbol{y_2} = D_1\boldsymbol{q_2} + D_2\boldsymbol{q_2} \\ \boldsymbol{y_3} = D_1\boldsymbol{q_3} + D_2\boldsymbol{q_3} + D_3\boldsymbol{q_3} \\ \vdots \\ \boldsymbol{y_t} = D_1\boldsymbol{q_t} + D_2\boldsymbol{q_t} + \ldots + D_t\boldsymbol{q_t} \end{cases}$$

$$\mathrm{H}_t = \mathrm{H}_{t-1} + f_{B,t}(\boldsymbol{x_t})$$

$$\boldsymbol{y_t} = f_{C,t}(\mathrm{H}_t)$$

H_t **is a** $d \times d$ **matric**

$$f_{B,t}(\boldsymbol{x_t}) = D_t$$

$$D_t = \boldsymbol{v_t}\boldsymbol{k_t}^T \qquad \boldsymbol{v_t} = W_v\boldsymbol{x_t}$$

$$\boldsymbol{k_t} = W_k\boldsymbol{x_t}$$

$$f_{C,t}(\mathrm{H}_t) = \mathrm{H}_t\boldsymbol{q_t}$$

$$\boldsymbol{q_t} = W_Q\boldsymbol{x_t}$$

RNN 有没有训练时并行的可能性

$$f_{A,1}(\mathrm{H}_0) = O$$

$$\begin{cases} \boldsymbol{y}_1 = \boldsymbol{v}_1 \boldsymbol{k}_1{}^T \boldsymbol{q}_1 \\ \boldsymbol{y}_2 = \boldsymbol{v}_1 \boldsymbol{k}_1{}^T \boldsymbol{q}_2 + \boldsymbol{v}_2 \boldsymbol{k}_2{}^T \boldsymbol{q}_2 \\ \boldsymbol{y}_3 = \boldsymbol{v}_1 \boldsymbol{k}_1{}^T \boldsymbol{q}_3 + \boldsymbol{v}_2 \boldsymbol{k}_2{}^T \boldsymbol{q}_3 + \boldsymbol{v}_3 \boldsymbol{k}_3{}^T \boldsymbol{q}_3 \\ \vdots \\ \boldsymbol{y}_t = \boldsymbol{v}_1 \boldsymbol{k}_1{}^T \boldsymbol{q}_t + \boldsymbol{v}_2 \boldsymbol{k}_2{}^T \boldsymbol{q}_t + \ldots + \boldsymbol{v}_t \boldsymbol{k}_t{}^T \boldsymbol{q}_t \end{cases}$$

$$\mathrm{H}_t = \mathrm{H}_{t-1} + f_{B,t}(\boldsymbol{x}_t)$$

$$\boldsymbol{y}_t = f_{C,t}(\mathrm{H}_t)$$

H_t **is a** $d \times d$ **matric**

$$f_{B,t}(\boldsymbol{x}_t) = D_t$$

$$D_t = \boldsymbol{v}_t \boldsymbol{k}_t{}^T \qquad \boldsymbol{v}_t = W_v \boldsymbol{x}_t \qquad \boldsymbol{k}_t = W_k \boldsymbol{x}_t$$

$$f_{C,t}(\mathrm{H}_t) = \mathrm{H}_t \boldsymbol{q}_t$$

$$\boldsymbol{q}_t = W_Q \boldsymbol{x}_t$$

RNN 有没有训练时并行的可能性

$$f_{A,1}(\mathrm{H}_0) = O$$

$$\boldsymbol{y_t} = \boldsymbol{v_1}\boldsymbol{k_1}^T\boldsymbol{q_t} + \boldsymbol{v_2}\boldsymbol{k_2}^T\boldsymbol{q_t} + \ldots + \boldsymbol{v_t}\boldsymbol{k_t}^T\boldsymbol{q_t}$$

$$= \boldsymbol{v_1}a_{t,1} + \boldsymbol{v_2}a_{t,2} + \ldots + \boldsymbol{v_t}a_{t,t}$$

$$= a_{t,1}\boldsymbol{v_1} + a_{t,2}\boldsymbol{v_2} + \ldots + a_{t,t}\boldsymbol{v_t}$$

这不就是 Self-attention! (少了 softmax)

叫做 Linear Attention

$$\mathrm{H}_t = \mathrm{H}_{t-1} + f_{B,t}(\boldsymbol{x_t})$$

$$\boldsymbol{y_t} = f_{C,t}(\mathrm{H}_t)$$

H_t **is a** $d \times d$ **matric**

$$f_{B,t}(\boldsymbol{x_t}) = D_t$$

$$D_t = \boldsymbol{v_t}\boldsymbol{k_t}^T$$

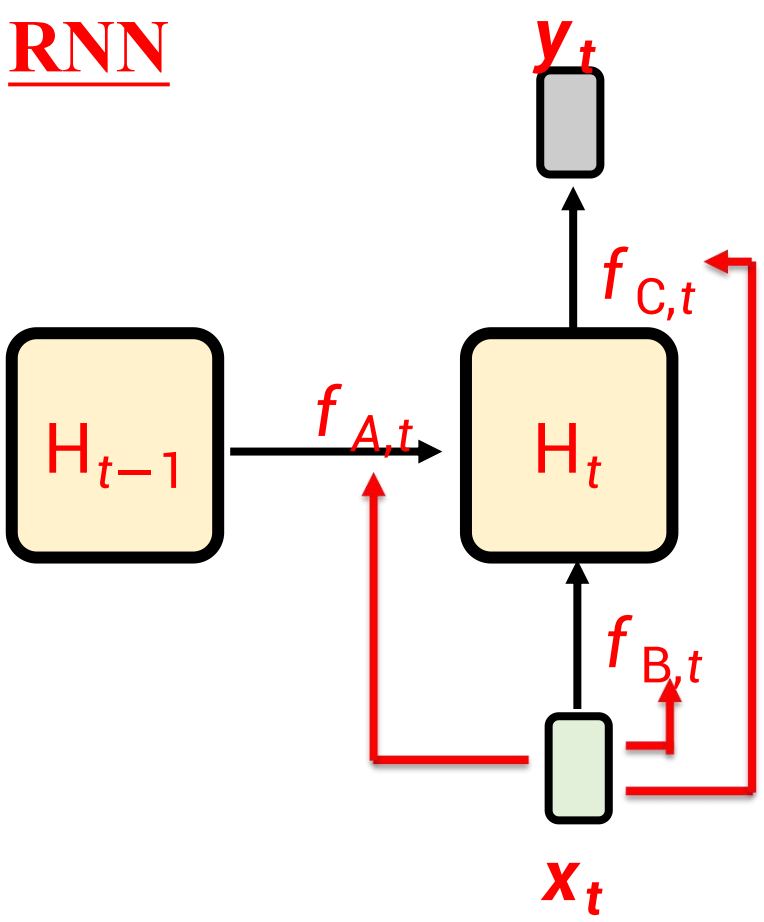
$$\boldsymbol{v_t} = W_v\boldsymbol{x_t}$$

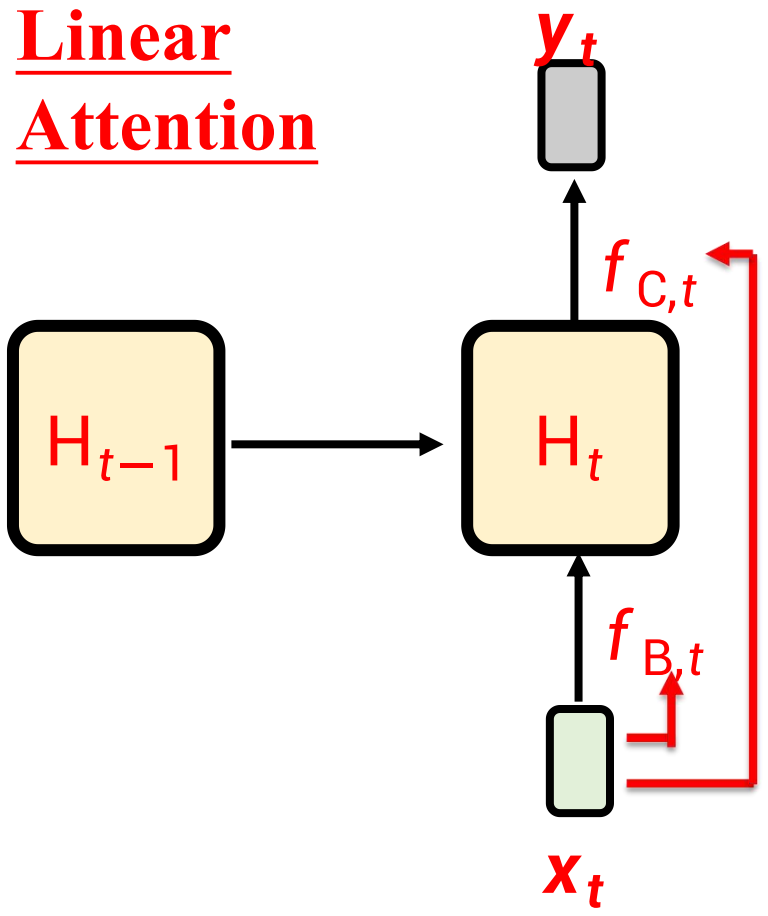
$$\boldsymbol{k_t} = W_k\boldsymbol{x_t}$$

$$f_{C,t}(\mathrm{H}_t) = \mathrm{H}_t\boldsymbol{q_t}$$

$$\boldsymbol{q_t} = W_Q\boldsymbol{x_t}$$

RNN vs Linear Attention





$$f_{C,t}(H_t) = H_t q_t$$

$$q_t = W_Q x_t$$

$$f_{B,t}(x_t) = v_t k_t^T$$

$$v_t = W_v x_t$$

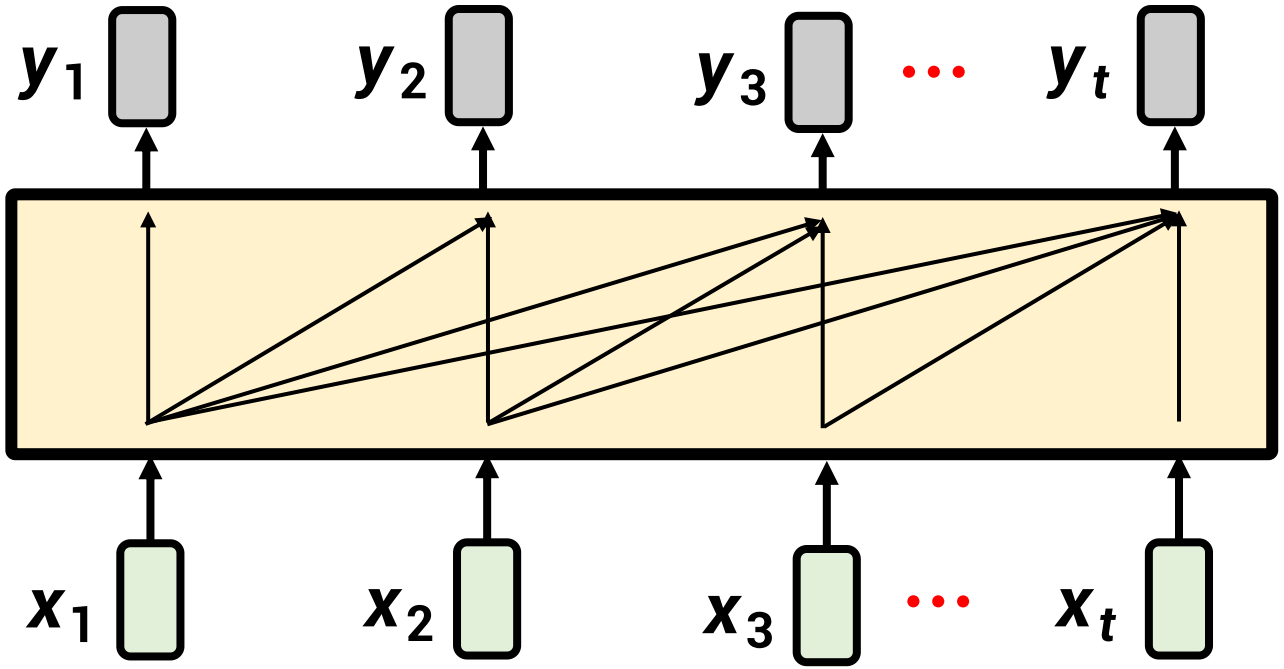
$$k_t = W_k x_t$$

- Linear Attention 就是广义 RNN 拿掉 “Reflection” $f_{A,t}$
- Linear Attention 就是 Self-attention 没有 Softmax

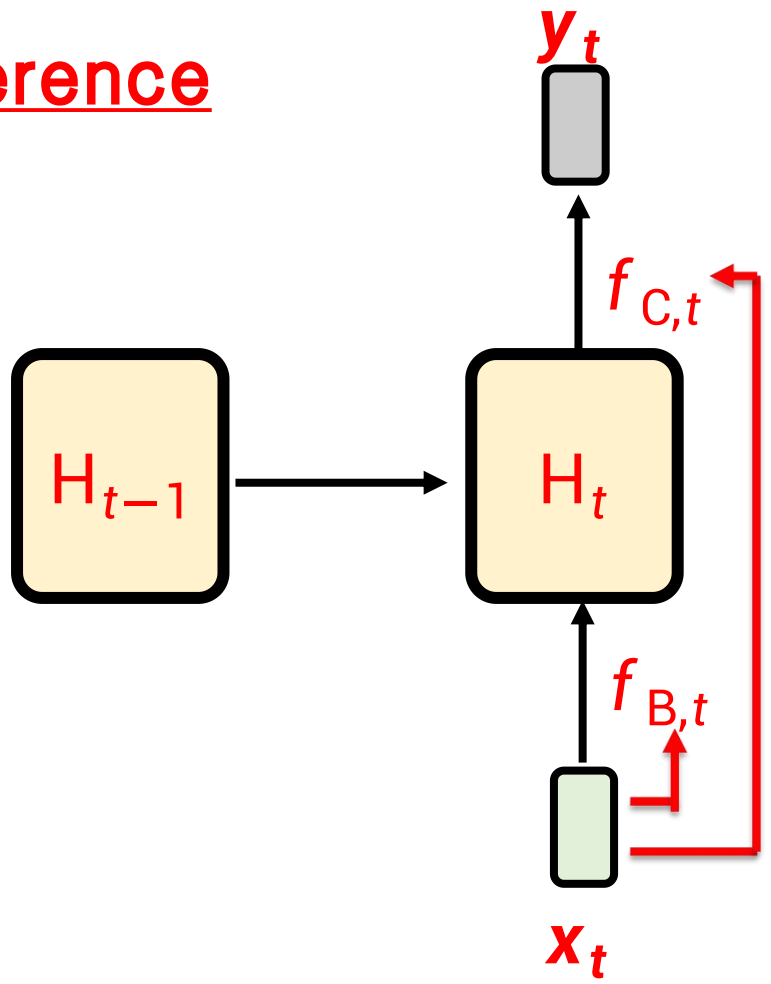
Linear Attention

Training 的时候像 Self-attention
Inference 的时候像 RNN

Training

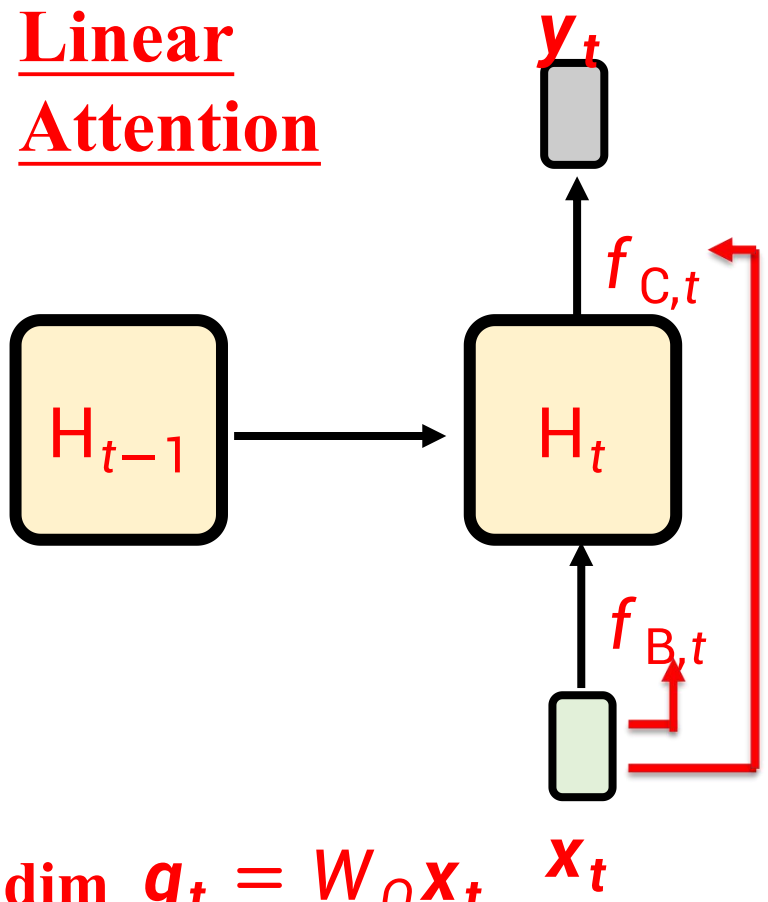


Inference



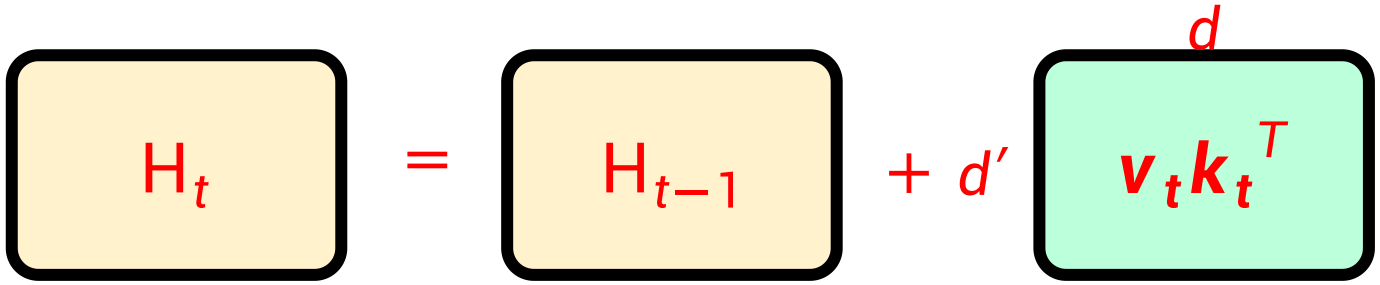
Linear Attention

Linear Attention

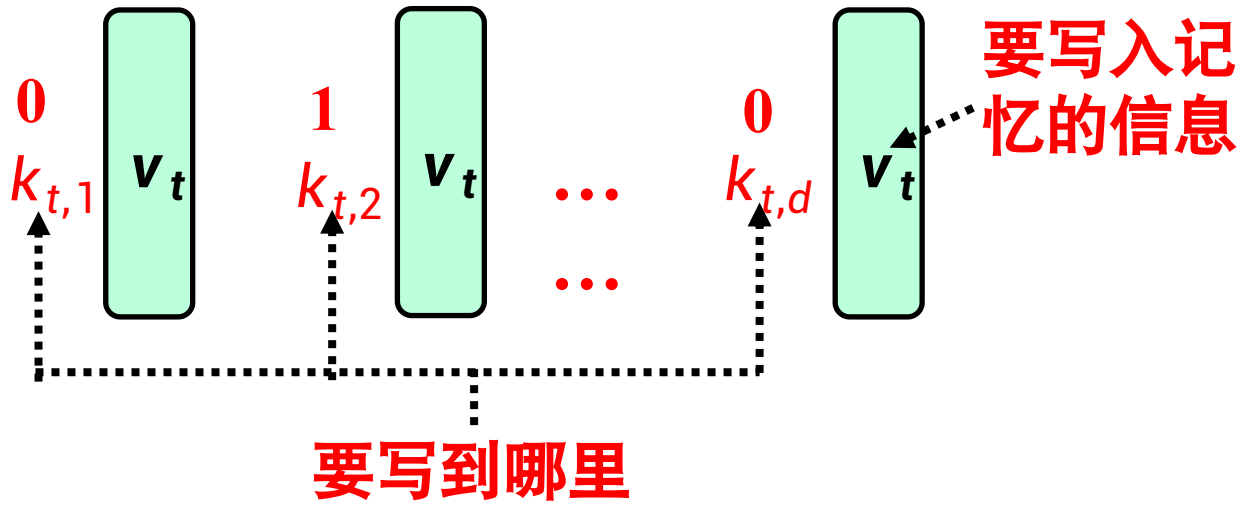


$$H_t = H_{t-1} + f_{B,t}(x_t) \qquad f_{B,t}(x_t) = v_t k_t^T$$

$$y_t = f_{C,t}(H_t) \qquad f_{C,t}(H_t) = H_t q_t$$



把 v_t 写入 H 的 2nd column



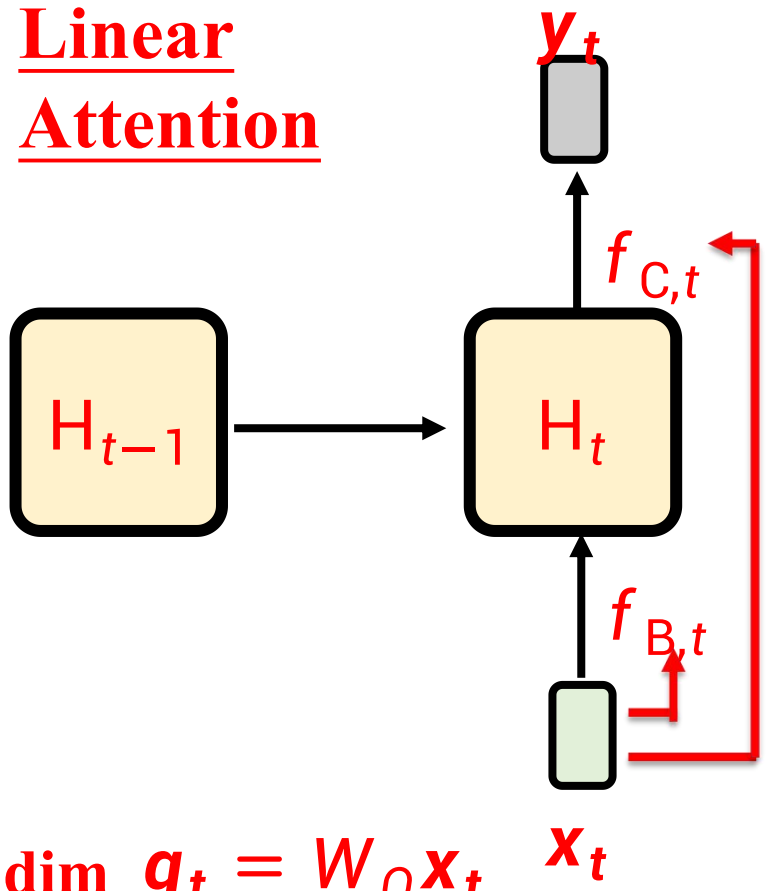
d dim $q_t = W_Q x_t$

d dim $k_t = W_k x_t$

d' dim $v_t = W_v x_t$

Linear Attention

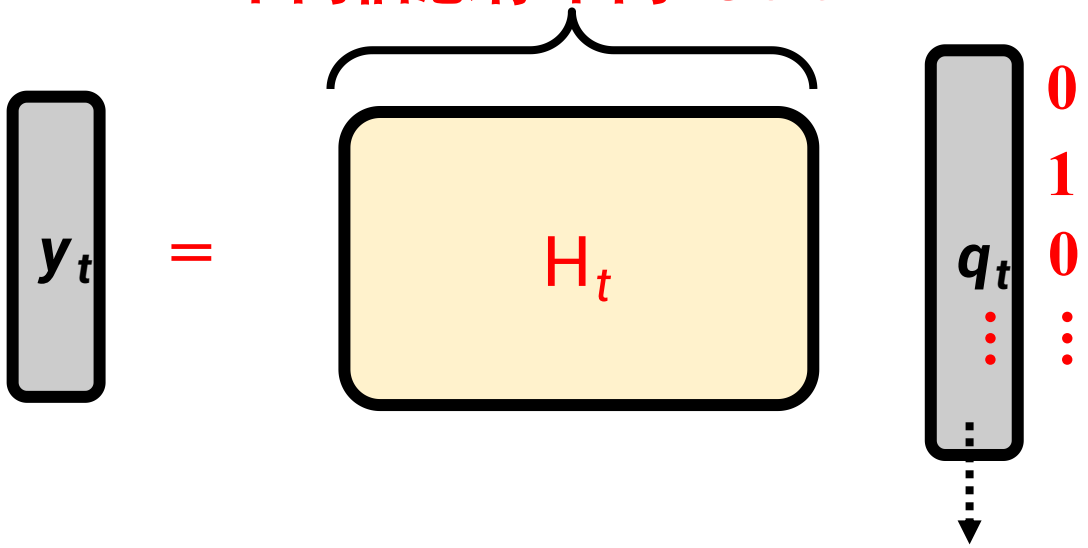
Linear Attention



$$H_t = H_{t-1} + f_{B,t}(x_t) \qquad f_{B,t}(x_t) = v_t k_t^T$$

$$y_t = f_{C,t}(H_t) \qquad f_{C,t}(H_t) = H_t q_t$$

不同信息存不同 Column



从哪一个 column 取多少信息

d dim $q_t = W_Q x_t$

d dim $k_t = W_k x_t$

d' dim $v_t = W_v x_t$

Transformer

$x_1 \to x_2$

$(x_1, x_2) \to x_3$

$(x_1, x_2, x_3) \to x_4$

$$y_1 = sm(q_1 k_1) v_1$$

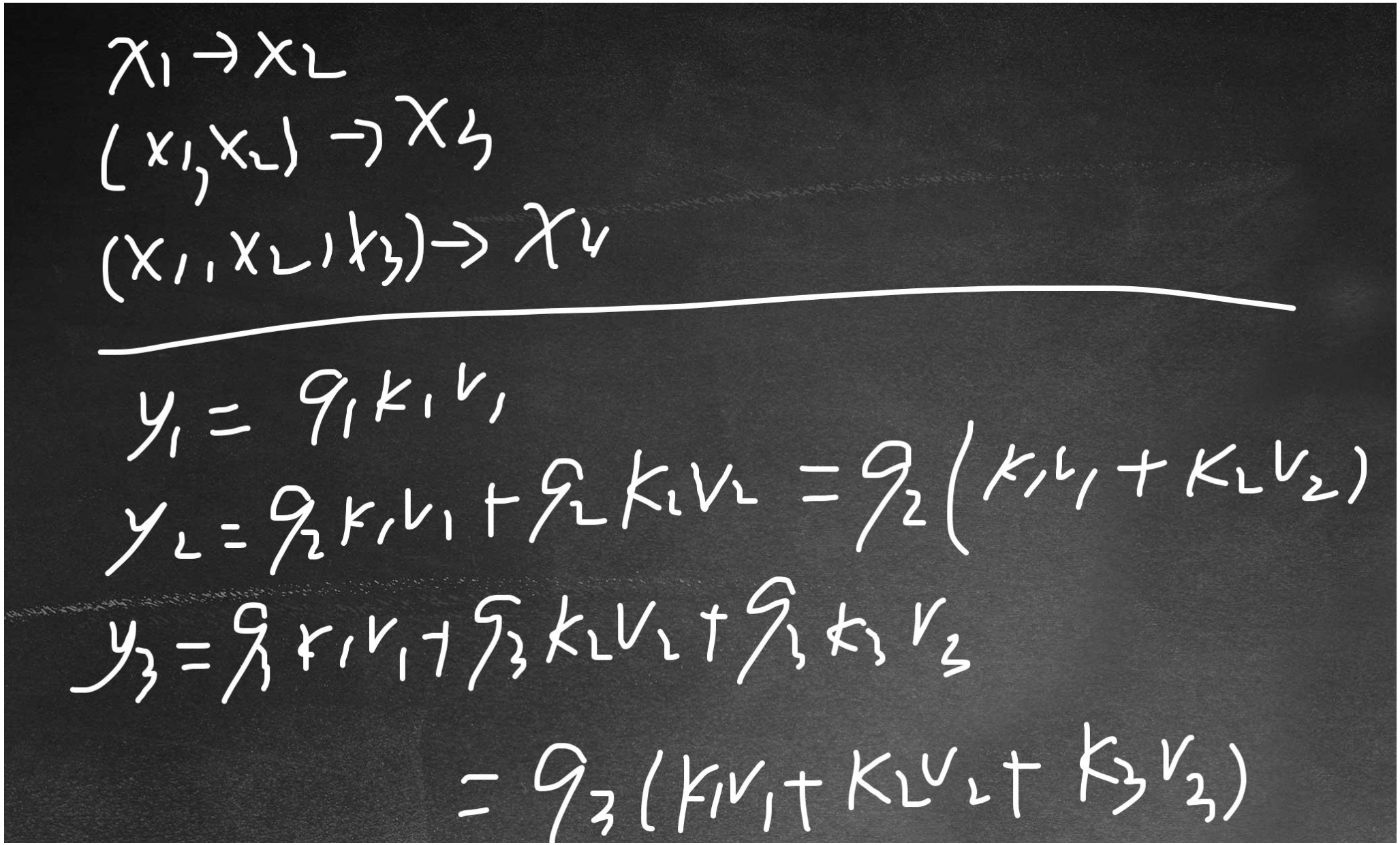
$$y_2 = sm(q_2 k_1) v_1 + sm(q_2 k_2) v_2$$

$$y_3 = sm(q_3 k_1) v_1 + sm(q_3 k_2) v_2 + sm(q_3 k_3) v_3$$

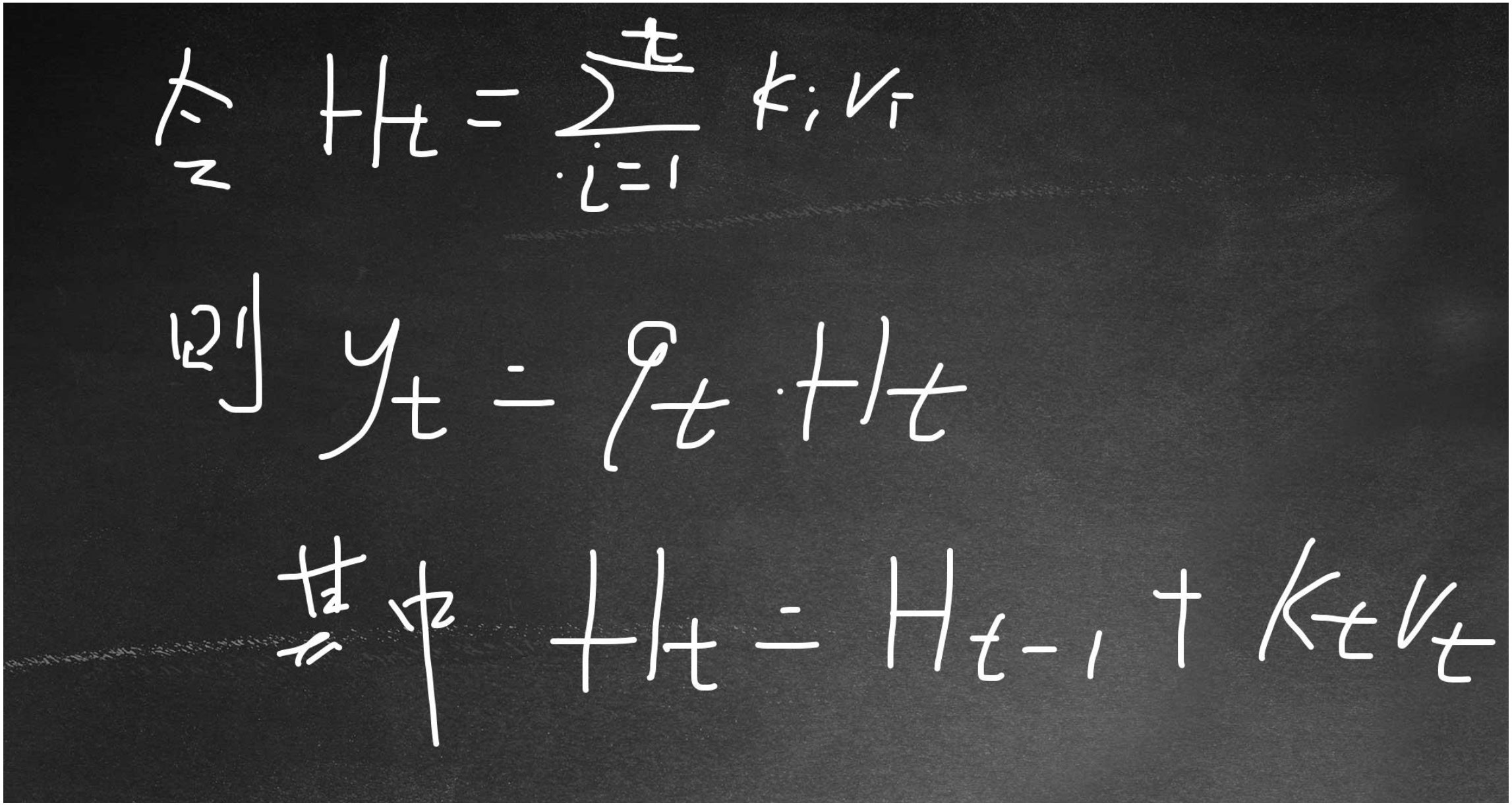
train: $O(n^2)$ but parallel

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} s(q_1 k_1) & 0 & 0 \\ s(q_2 k_1) & s(q_2 k_2) & 0 \\ s(q_3 k_1) & s(q_3 k_2) & s(q_3 k_3) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = sm \begin{bmatrix} q_1 k_1 & -\infty & -\infty \\ q_2 k_1 & q_2 k_2 & -\infty \\ q_3 k_1 & q_3 k_2 & q_3 k_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = sm(QK)V$$

Linear Attention

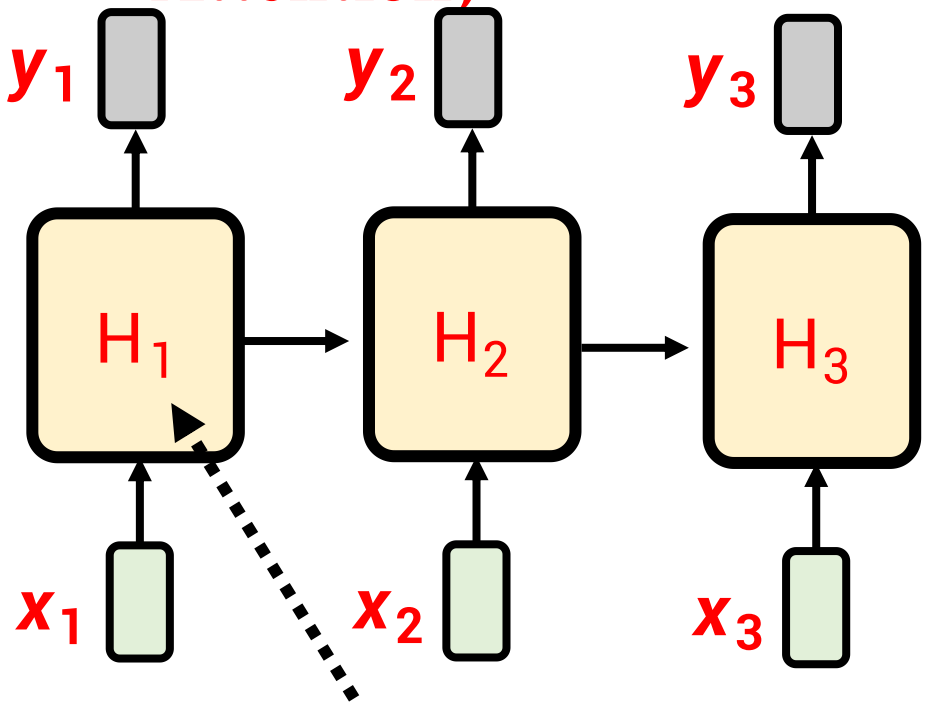


Linear Attention



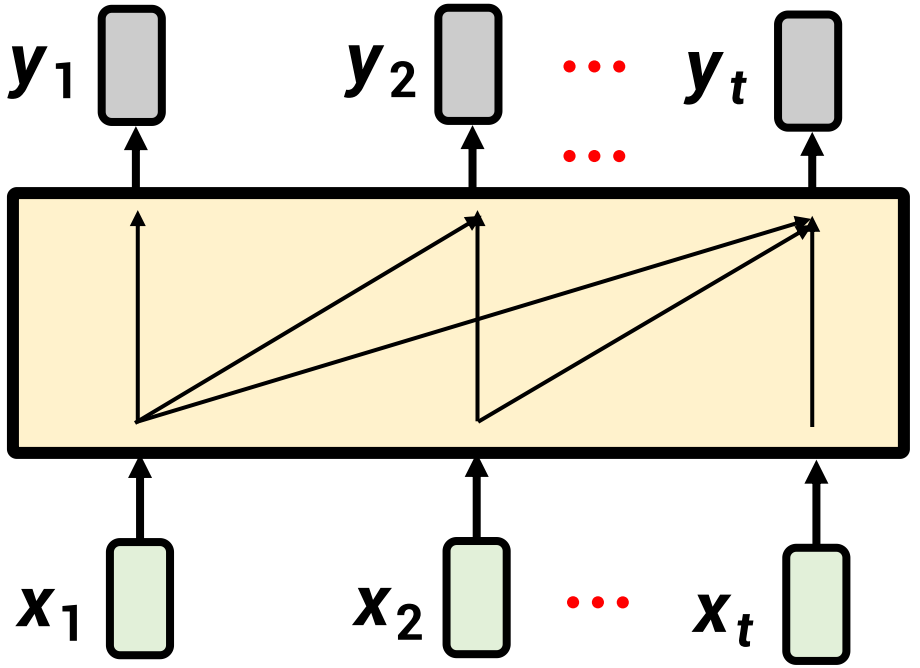
RNN (Linear Attention) 赢不过 Transformer

RNN (Linear Attention)



~~记忆太小~~ 记忆有限

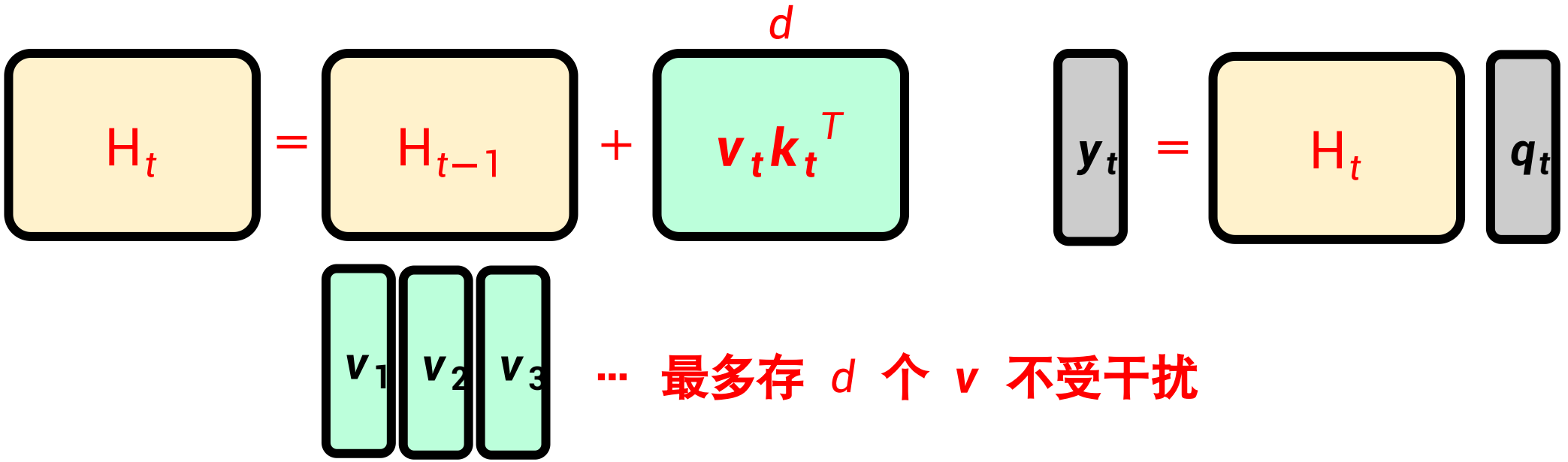
Transformer (Self-attention with softmax)



无限记忆?

RNN (Linear Attention) 赢不过 Transformer

RNN (Linear Attention)



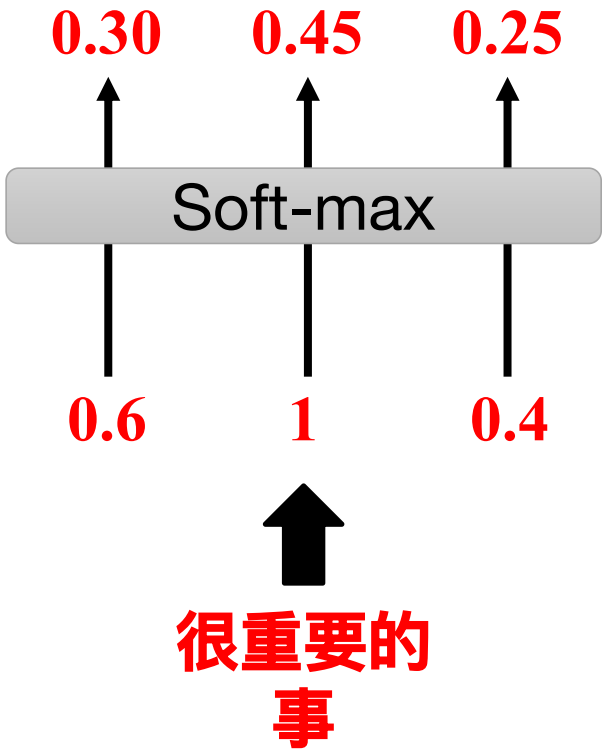
最多存 d 个 v 不受干扰

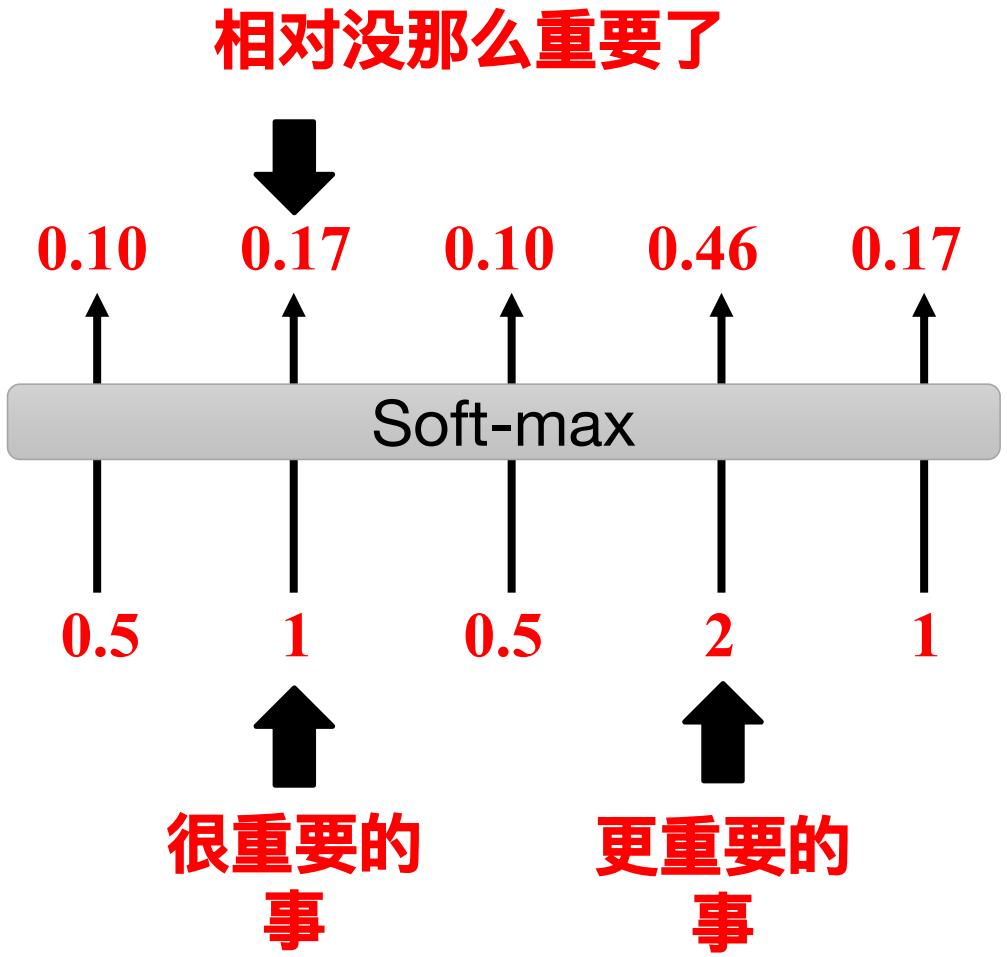
$k_1^T = [1 \quad 0 \quad \ldots]$ $k_2^T = [0 \quad 1 \quad \ldots]$ $k_3^T = [0 \quad 0 \quad 1 \ldots]$

RNN（Linear Attention）赢不过 Transformer

$$H_t = H_{t-1} + f_{B,t}(x_t)$$

Linear Attention 记忆永不改变





加上 Reflection：逐渐遗忘

Linear Attention

$$\mathrm{H}_t = \mathrm{H}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^T$$

$$\boldsymbol{y}_t = \mathrm{H}_t \boldsymbol{q}_t$$

$$\boldsymbol{v}_t = W_v \boldsymbol{x}_t$$

$$\boldsymbol{k}_t = W_k \boldsymbol{x}_t$$

$$\boldsymbol{q}_t = W_Q \boldsymbol{x}_t$$

Retention Network (RetNet)

$$\mathrm{H}_t = \gamma \mathrm{H}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^T$$

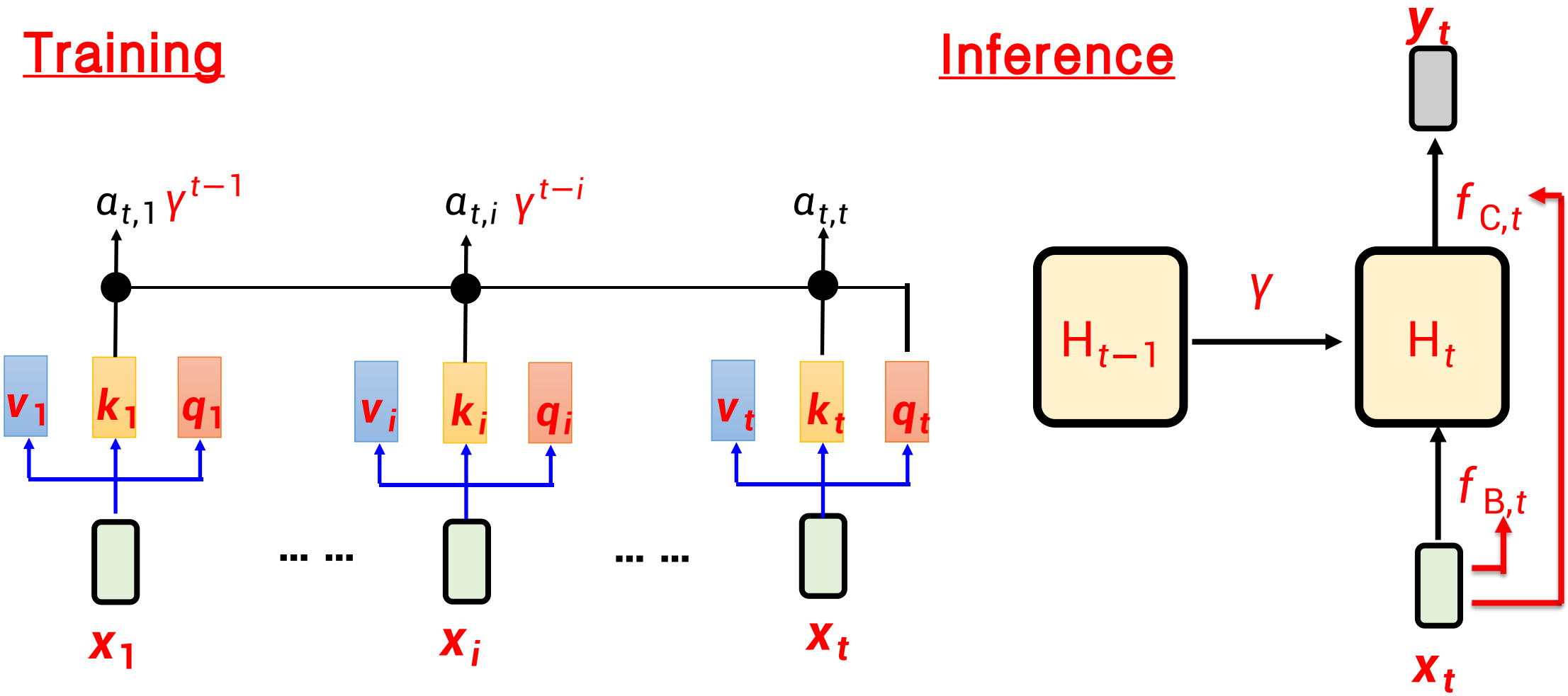
$$\boldsymbol{y}_t = \mathrm{H}_t \boldsymbol{q}_t$$

$$\boldsymbol{v}_t = W_v \boldsymbol{x}_t$$

$$\boldsymbol{k}_t = W_k \boldsymbol{x}_t$$

$$\boldsymbol{q}_t = W_Q \boldsymbol{x}_t$$

加上 Reflection：逐渐遗忘



加上 Reflection：根据情况遗忘

https://arxiv.org/abs/2405.05254

Retention Network (RetNet)

$$H_t = \gamma H_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^T$$

$$\boldsymbol{y}_t = H_t \boldsymbol{q}_t$$

$$\boldsymbol{v}_t = W_v \boldsymbol{x}_t$$

$$\boldsymbol{k}_t = W_k \boldsymbol{x}_t$$

$$\boldsymbol{q}_t = W_Q \boldsymbol{x}_t$$

Gated Retention

$$H_t = \gamma_t H_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^T$$

$$\boldsymbol{y}_t = H_t \boldsymbol{q}_t$$

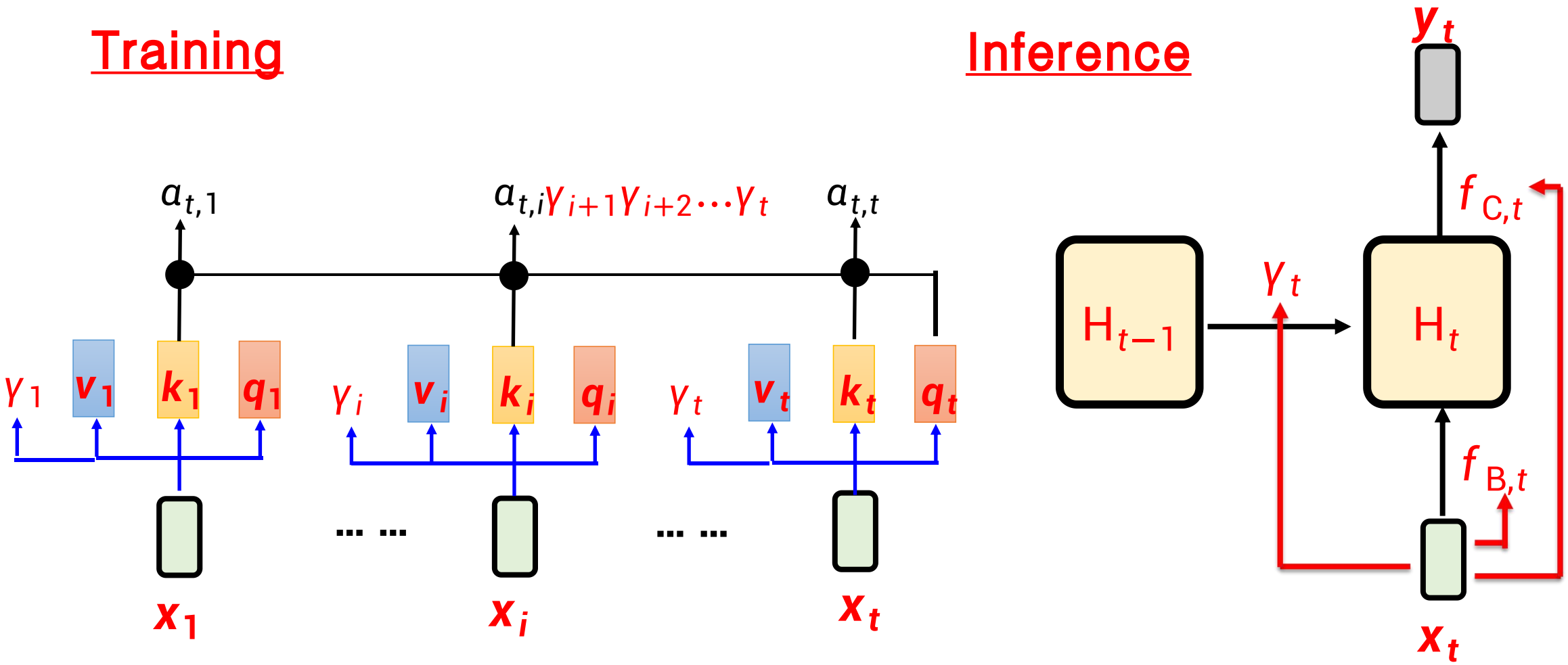
$$\boldsymbol{v}_t = W_v \boldsymbol{x}_t$$

$$\boldsymbol{k}_t = W_k \boldsymbol{x}_t$$

$$\boldsymbol{q}_t = W_Q \boldsymbol{x}_t$$

$$\gamma_t = sigmoid(W_\gamma \boldsymbol{x}_t)$$

加上 Reflection：逐渐遗忘



未卜先知

更复杂的 Reflection

$$H_t = G_t \odot H_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^T$$

$$G_t = \boldsymbol{e}_t \boldsymbol{s}_t^T$$

$$G_t = \mathbf{1} \boldsymbol{s}_t^T$$

$$\mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

$$\boldsymbol{s}_t^T = [0 \quad 1 \quad 0.1 \ \ldots\ldots]$$

$$\mathbf{1} \begin{bmatrix} s_{t,1} & s_{t,2} & s_{t,3} & \cdots & s_{t,d} \\ s_{t,1} & s_{t,2} & s_{t,3} & & s_{t,d} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ s_{t,1} & s_{t,2} & s_{t,3} & & s_{t,d} \end{bmatrix} \odot H_{t-1}$$

$\boldsymbol{s}_t^T$

0 1 0.1

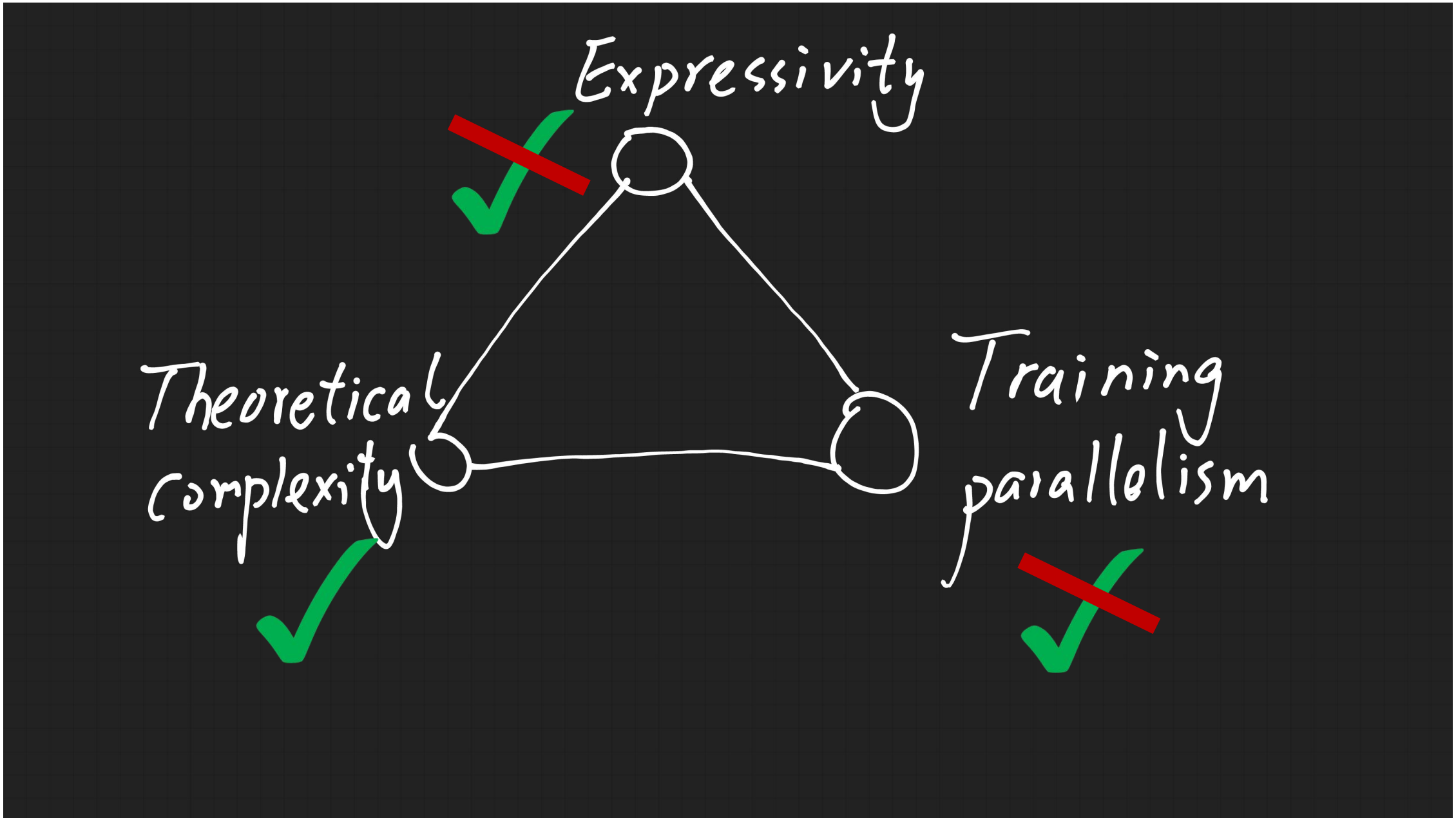
抹去 保留 减弱

更种各样的RNN

Model	Recurrence	Memory read-out
Linear Attention [48, 47]	$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
+ Kernel	$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{v}_t \phi(\boldsymbol{k}_t)^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \phi(\boldsymbol{q}_t)$
+ Normalization	$\mathbf{S}_t = \mathbf{S}_{t-1} + \boldsymbol{v}_t \phi(\boldsymbol{k}_t)^\top, \quad \boldsymbol{z}_t = \boldsymbol{z}_{t-1} + \phi(\boldsymbol{k}_t)$	$\boldsymbol{o}_t = \mathbf{S}_t \phi(\boldsymbol{q}_t) / (\boldsymbol{z}_t^\top \phi(\boldsymbol{q}_t))$
DeltaNet [101]	$\mathbf{S}_t = \mathbf{S}_{t-1}(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
Gated RFA [81]	$\mathbf{S}_t = g_t \mathbf{S}_{t-1} + (1 - g_t) \boldsymbol{v}_t \boldsymbol{k}_t^\top, \quad \boldsymbol{z}_t = g_t \boldsymbol{z}_{t-1} + (1 - g_t) \boldsymbol{k}_t$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t / (\boldsymbol{z}_t^\top \boldsymbol{q}_t)$
S4 [32, 106]	$\mathbf{S}_t = \mathbf{S}_{t-1} \odot \exp(-(\boldsymbol{\alpha} \mathbf{1}^\top) \odot \exp(\boldsymbol{A})) + \boldsymbol{B} \odot (\boldsymbol{v}_t \mathbf{1}^\top)$	$\boldsymbol{o}_t = (\mathbf{S}_t \odot \boldsymbol{C}) \mathbf{1} + \boldsymbol{d} \odot \boldsymbol{v}_t$
ABC [82]	$\mathbf{S}_t^k = \mathbf{S}_{t-1}^k + \boldsymbol{k}_t \boldsymbol{\phi}_t^\top, \quad \mathbf{S}_t^v = \mathbf{S}_{t-1}^v + \boldsymbol{v}_t \boldsymbol{\phi}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t^v \operatorname{softmax}(\mathbf{S}_t^k \boldsymbol{q}_t)$
DFW [63]	$\mathbf{S}_t = \mathbf{S}_{t-1} \odot (\boldsymbol{\beta}_t \boldsymbol{\alpha}_t^\top) + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
RetNet [108]	$\mathbf{S}_t = \gamma \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
Mamba [31]	$\mathbf{S}_t = \mathbf{S}_{t-1} \odot \exp(-(\boldsymbol{\alpha}_t \mathbf{1}^\top) \odot \exp(\boldsymbol{A})) + (\boldsymbol{\alpha}_t \odot \boldsymbol{v}_t) \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t + \boldsymbol{d} \odot \boldsymbol{v}_t$
GLA [124]	$\mathbf{S}_t = \mathbf{S}_{t-1} \odot (\mathbf{1} \boldsymbol{\alpha}_t^\top) + \boldsymbol{v}_t \boldsymbol{k}_t^\top = \mathbf{S}_{t-1} \operatorname{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
RWKV-6 [79]	$\mathbf{S}_t = \mathbf{S}_{t-1} \operatorname{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = (\mathbf{S}_{t-1} + (\boldsymbol{d} \odot \boldsymbol{v}_t) \boldsymbol{k}_t^\top) \boldsymbol{q}_t$
HGRN-2 [92]	$\mathbf{S}_t = \mathbf{S}_{t-1} \operatorname{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t (\mathbf{1} - \boldsymbol{\alpha}_t)^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
mLSTM [9]	$\mathbf{S}_t = f_t \mathbf{S}_{t-1} + i_t \boldsymbol{v}_t \boldsymbol{k}_t^\top, \quad \boldsymbol{z}_t = f_t \boldsymbol{z}_{t-1} + i_t \boldsymbol{k}_t$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t / \max\{1, \lvert \boldsymbol{z}_t^\top \boldsymbol{q}_t \rvert\}$
Mamba-2 [19]	$\mathbf{S}_t = \gamma_t \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$
GSA [131]	$\mathbf{S}_t^k = \mathbf{S}_{t-1}^k \operatorname{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{k}_t \boldsymbol{\phi}_t^\top, \quad \mathbf{S}_t^v = \mathbf{S}_{t-1}^v \operatorname{Diag}(\boldsymbol{\alpha}_t) + \boldsymbol{v}_t \boldsymbol{\phi}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t^v \operatorname{softmax}(\mathbf{S}_t^k \boldsymbol{q}_t)$
Gated DeltaNet [125]	$\mathbf{S}_t = \mathbf{S}_{t-1} \left(\alpha_t (\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top) \right) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \mathbf{S}_t \boldsymbol{q}_t$

https://arxiv.org/abs/2406.06484

DeltaNet


Expressivity
Theoretical complexity
Training parallelism

代表2：DeltaNet

DeltaNet

https://arxiv.org/abs/2406.06484

$$H_t = H_{t-1}(I - \beta_t k_t k_t^T) + \beta_t v_t k_t^T$$

$$H_t = H_{t-1} + v_t k_t^T$$ ← 也是 Gradient Descent，只是 L_t 不一样

$$H_t = H_{t-1} - v_{t,old} k_t^T + v_t k_t^T \qquad v_{t,old} = H_{t-1} k_t$$

$$H_t = H_{t-1} - \beta_t v_{t,old} k_t^T + \beta_t v_t k_t^T$$

$$H_t = H_{t-1} - \beta_t H_{t-1} k_t k_t^T + \beta_t v_t k_t^T$$

Gradient Descent

$$H_t = H_{t-1} - \beta_t (H_{t-1} k_t - v_t) k_t^T$$

H_t: Parameter after update; H_{t-1}: before update; β_t: learning rate; $(H_{t-1} k_t - v_t) k_t^T$: gradient

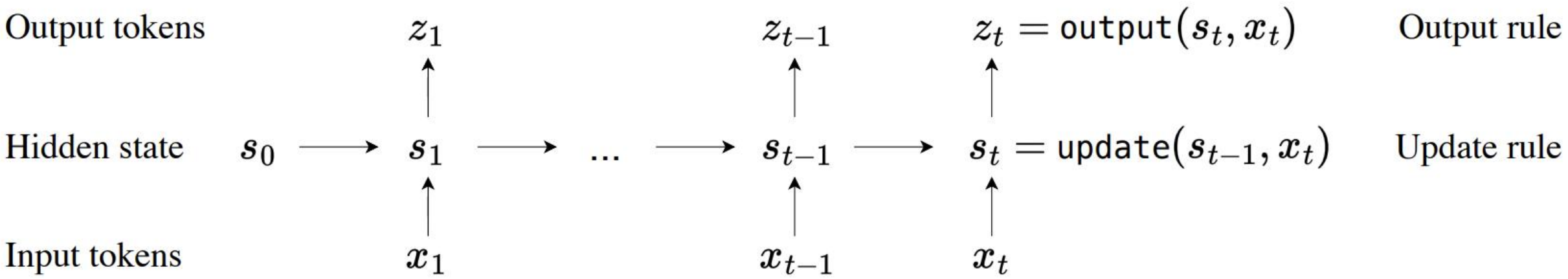
$$L_t(H) = \frac{1}{2} \|H k_t - v_t\|^2$$

$$\nabla L_t(H_{t-1})$$

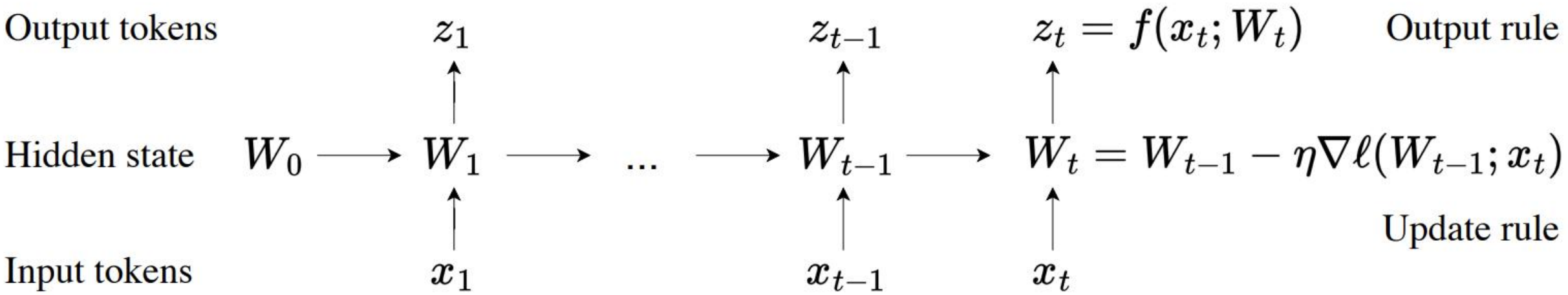
更新 H 使得用 k_t 抽取出的信息和 v_t 越接近越好

Titans: Learning to Memorize at Test Time https://arxiv.org/abs/2501.00663

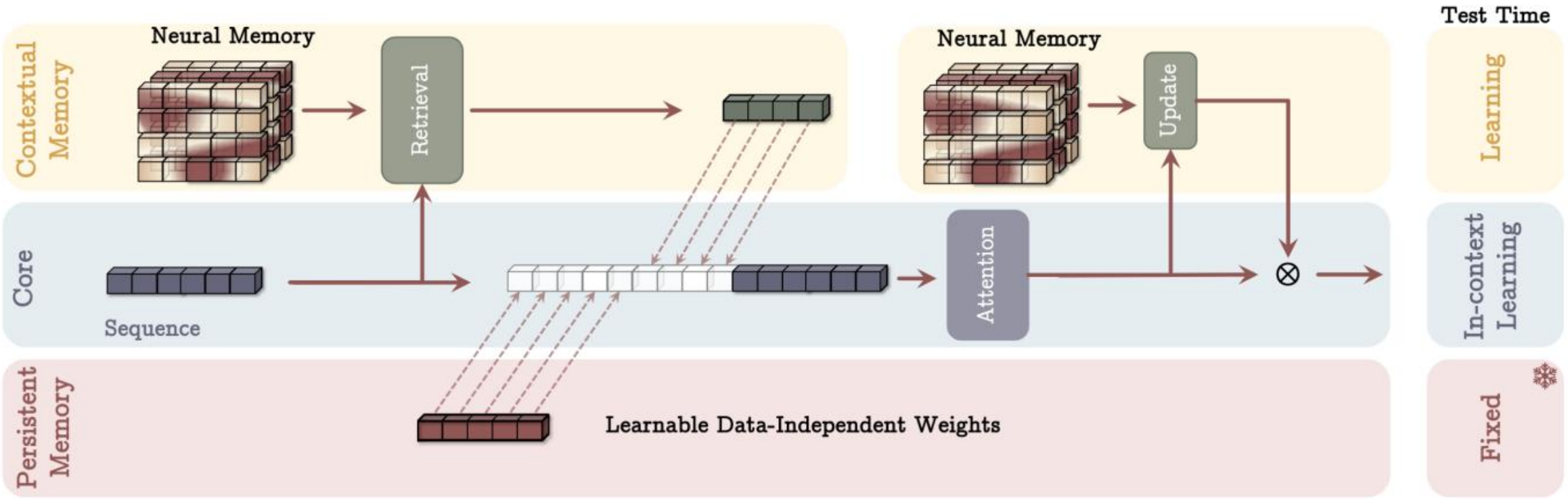
TTT



	Initial state	**Update rule**	**Output rule**	**Cost**
Naive RNN	$s_0 = \texttt{vector()}$	$s_t = \sigma(\theta_{ss} s_{t-1} + \theta_{sx} x_t)$	$z_t = \theta_{zs} s_t + \theta_{zx} x_t$	$O(1)$
Self-attention	$s_0 = \texttt{list()}$	$s_t = s_{t-1}.\texttt{append}(k_t, v_t)$	$z_t = V_t \texttt{softmax}\left(K_t^T q_t\right)$	$O(t)$
Naive TTT	$W_0 = f.\texttt{params()}$	$W_t = W_{t-1} - \eta \nabla \ell(W_{t-1}; x_t)$	$z_t = f(x_t; W_t)$	$O(1)$

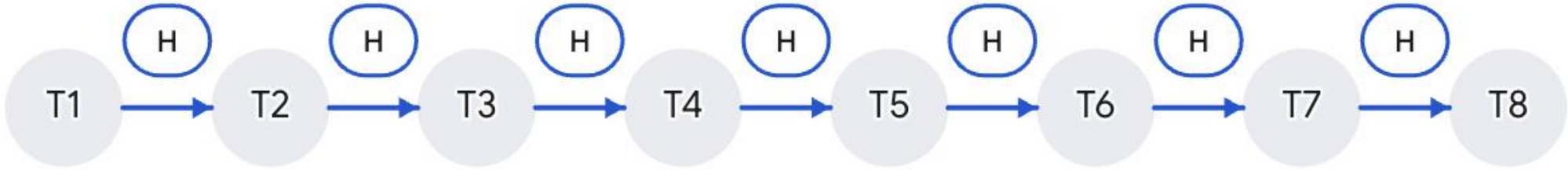

Output tokens
Hidden state
Input tokens
z_1
z_{t-1}
$z_t = f(x_t; W_t)$
Output rule
W_0
W_1
...
W_{t-1}
$W_t = W_{t-1} - \eta \nabla \ell(W_{t-1}; x_t)$
Update rule
x_1
x_{t-1}
x_t

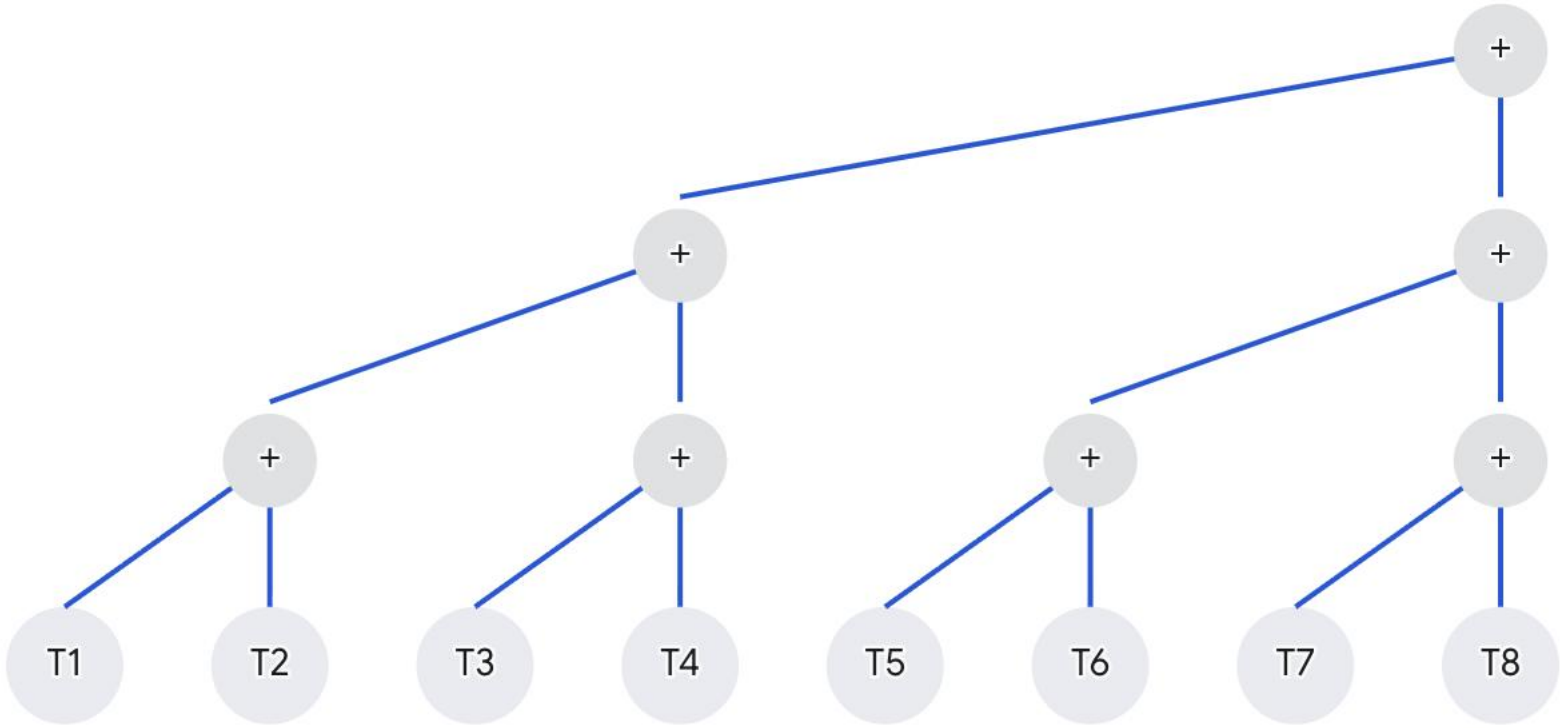
Titans


Test Time
Contextual Memory
Neural Memory
Retrieval
Neural Memory
Update
Learning
Core
Sequence
Attention
In-context Learning
Persistent Memory
Learnable Data-Independent Weights
Fixed

代表3：Mamba

MamBa





底层加速

- 虽然算法上快了，但 GPU 还有一个毛病：它的主显存容量大但是存取速度慢；而它每个计算核心旁边有一块超级小但超级快的缓存（SRAM）
- 以前的模型算一步，就要把中间结果写回慢速大显存，下一步再读出来，时间全浪费在“读写内存”上了
- Mamba 的 Scan 机制做了一个“核融合（Kernel Fusion）”：它把连续的输入全部一次性吞进极速的 SRAM 中，在 SRAM 里面用上面说的“锦标赛模式”疯狂合并计算，直到算出最终结果，才把结果写回慢速大显存

Mamba2; Mamba 3

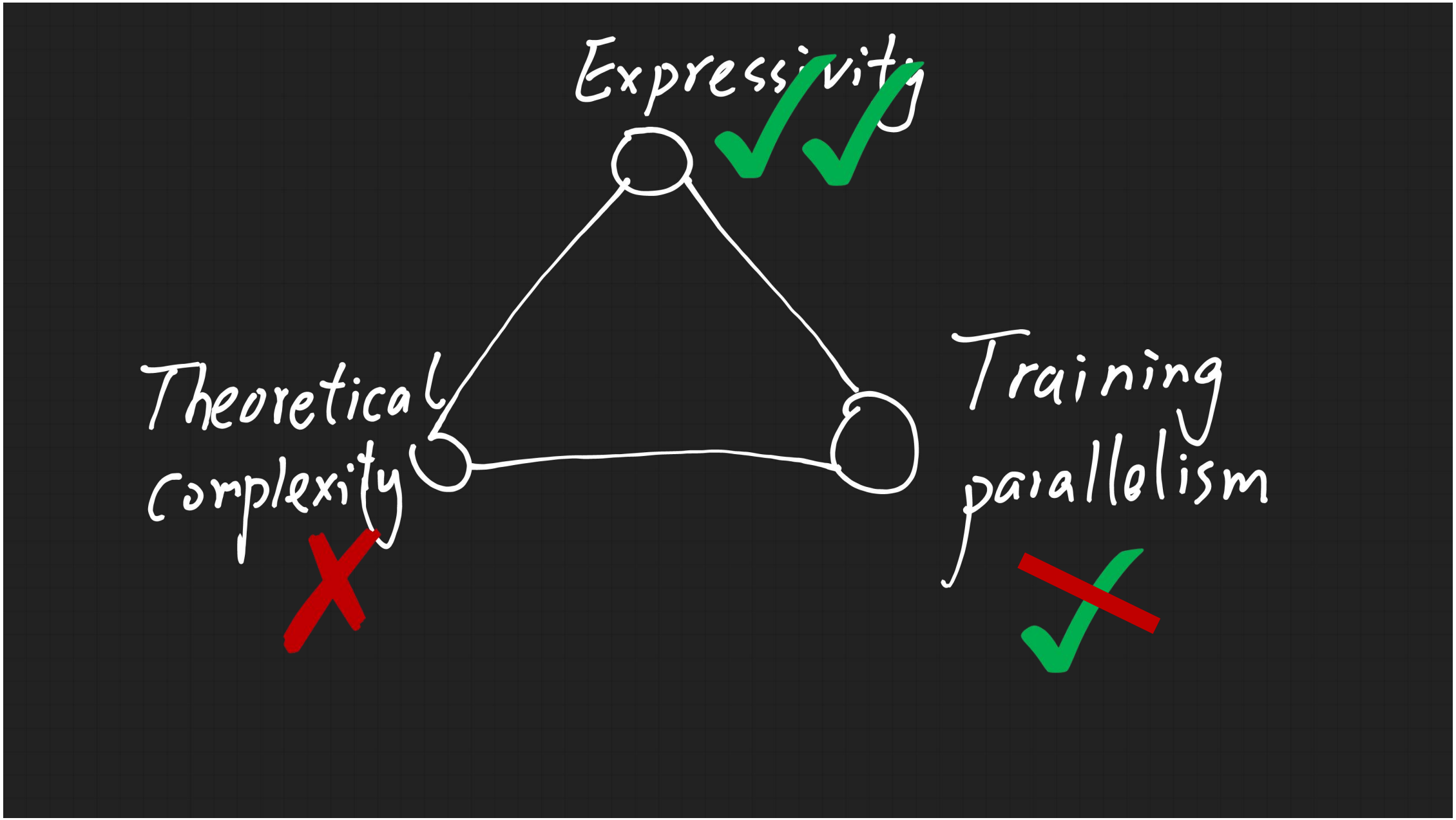
$$\mathbf{S}_t = \gamma_t \mathbf{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$$

Mamba团队自己抛弃自己了

本节内容

CONTENTS

如何提升Transformer的张力



双路记忆

$$E = -\frac{1}{\beta}\log\left(\sum_i \exp(\beta X_i^T \xi)\right) + \frac{1}{2}\xi^T\xi$$

$$\xi_{new} = \xi - \nabla_\xi E$$

$$\nabla_\xi E_1 = -\sum_{i=1}^{N} X_i \left(\frac{\exp(\beta X_i^T \xi)}{\sum_{j=1}^{N}\exp(\beta X_j^T \xi)}\right)$$

$$\nabla_\xi E_2 = \nabla_\xi\left(\frac{1}{2}\xi^T\xi\right) = \xi$$

$$\xi_{new} = X \cdot \mathrm{Softmax}(\beta X^T \xi)$$

双路记忆

$$\nabla_{X_i} E = -\left(\frac{\exp(\beta X_i^T \xi)}{\sum_{j=1}^{N} \exp(\beta X_j^T \xi)} \right) \xi$$

$$X_i^{(new)} = X_i - \eta \nabla_{X_i} E$$

$$X_i^{(new)} = X_i + \eta p_i \xi$$

Loss function

- **Fine-grained memory, next token prediction**
- **Abstract-level memory, contrastive learning**

问题和讨论