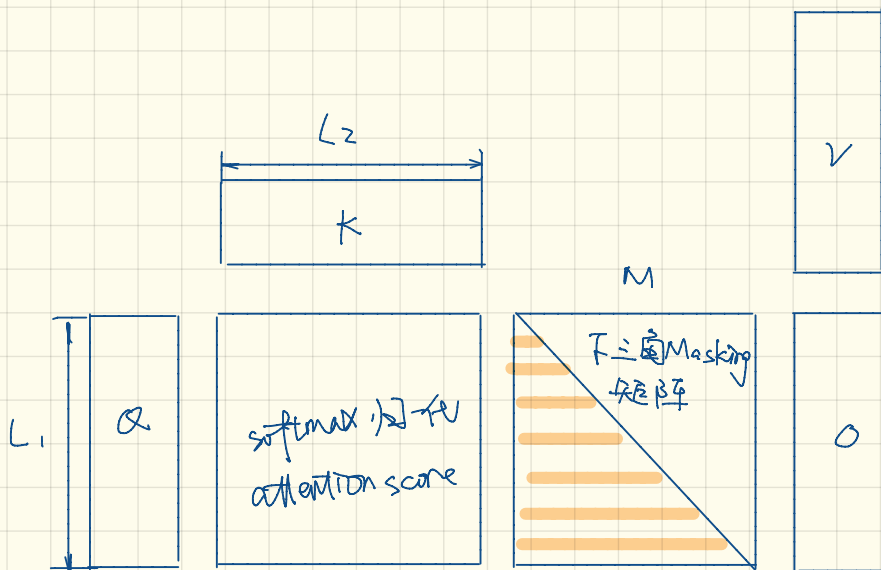


训练时 attention 以全并行方式进行计算, 也就是矩阵化公式

$$Q \ K \ V = xW_q, xW_k, xW_v$$

$$O = (\text{softmax}(QK^T) \odot M) V$$



↑
中间结果的时间和空间复杂度是
序列长度的二次方

全并行但 I/O 复杂度高

预测可以 recurrent-form 计算

$$q_t, k_t, v_t = x_t W_q, x_t W_k, x_t W_v$$

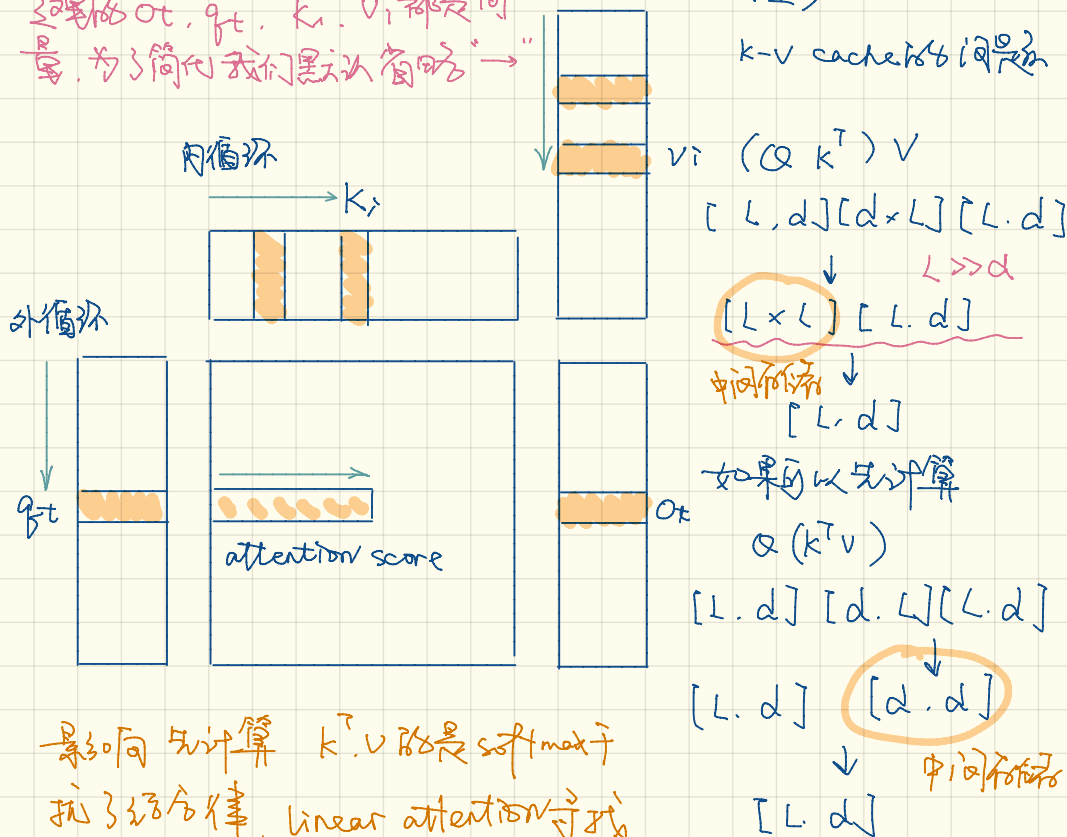
将这里的 softmax 计算看作是相似性计算

$$O_t = \frac{\sum_{i=1}^t \exp(q_t \cdot \underline{k_i^T}) \underline{v_i}}{\sum_{i=1}^t \exp(q_i \cdot \underline{k_i^T})}$$

随着外循环长度的增加如 $\{k_i\} \{v_i\}$ (都是向量) 的长度在增加。

k-v cache 的问题

这里的 q_t, k_t, v_i 都是向量, 为了简化我们默认省略 "→"



影响先计算 $k^T \cdot v$ 的是 softmax 违反了结合律, linear attention 寻找对 softmax 的改造, 使得可以

全并行和串行 RNN 式计算 (这里串行可以以较少的并行性 parallel scan 进行并行性增强)

$$Q_t = \frac{\sum_{i=1}^T \exp(q_t, k_i^T) v_i}{\sum_{i=1}^T \exp(q_t, k_i^T)}$$

内循环

用 kernel 替代 $\exp(q, k)$

$\exp(\vec{q}, \vec{k}) = \langle \phi(\vec{x}), \phi(\vec{y}) \rangle$. 让 Q_t 的计算被简化.

$$Q_t = \frac{\sum_{i=1}^T \phi(q_t) \phi(k_i)^T v_i}{\sum_{i=1}^T \phi(q_t) \phi(k_i)^T}$$

$$= \frac{\phi(q_t) \sum_{i=1}^T \phi(k_i) v_i}{\sum_{i=1}^T \phi(k_i)} \triangleq S_t$$

$$\triangleq Z_t$$

$$\begin{cases} Q_t = \frac{\phi(q_t) S_t}{\phi(q_t) Z_t} \\ S_t = S_{t-1} + \phi(k_t)^T v_t \\ Z_t = Z_{t-1} + \phi(k_t) \end{cases}$$

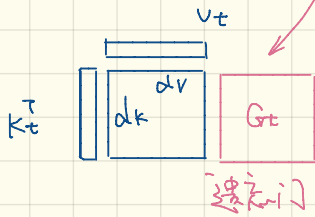
这是在 attention score 的计算中是一个归一化项, 一些研究验证了可以使用线性 kernel (例如将 ϕ 设置为 identity 函数) 使用未归一化的 score. 于是这个推广版本的 attention 可以被简化

这是 linear attention 的基本框架

$$\begin{cases} Q_t = \phi(q_t) S_t \\ S_t = S_{t-1} + \phi(k_t)^T v_t \end{cases}$$

架. SSM/RWKV 这样的新型 RNN 从另一个理论视角得到这种形式

$$S_t = S_{t-1} + K_t^T V_t, \quad O_t = g(S_t)$$



Linear attention 在学习效果建模上有一个缺点，类似于 LSTM 中遗忘门的衰减机制对学习效果帮助很大，这里没有 decay
 ← 这是一个外积操作，Linear attention 是一个隐状态是 $> D$ 的并行 RNN

$$K_t^T V_t = (x_t W_k)^T (W_v x_t)$$

同学习参数

$$O = ((K^T) \odot M) V$$

但由于 M 的存在破坏了矩阵乘法的结合性。
 O 的计算还是需要从左到右，因此依然是 O^2 空间复杂度

Gated Linear Unit

RetNet 引入静态衰减

Mamba / Mamba-2 data-dependent 的 decay 并行难度高

GLA 的基本形式: $S_t = G_t \odot S_{t-1} + K_t^T V_t$

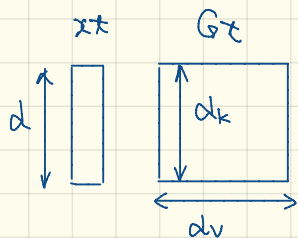
这个公式可以统一许多模型的计算过程

为隐状态引入遗忘门，随时间衰减

$$G_t = [d_t \times d_v] \in [0, 1]$$

在算法设计上要着重考虑如何得到 G_t

$x_t \rightarrow G_t$ 求解一个参数的映射



如果通过某种映射矩阵得到 G_t , 需要 $d * d_k * d_v$ 的参数量. 这是低效的

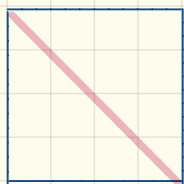
在 Mamba 中, 使用向量 α 与矩阵 A 之间的运算得到 G_t . 缺点是没法使用 tensor core

$$G_t = \alpha_t^T$$

$$S_t = (\alpha_t^T) \odot S_{t-1} + K_t^T V_t$$

$$= \text{Diag}(\alpha_t) S_{t-1} + K_t^T V_t$$

这个公式的通用性很好
并且对硬件友好的并行实现



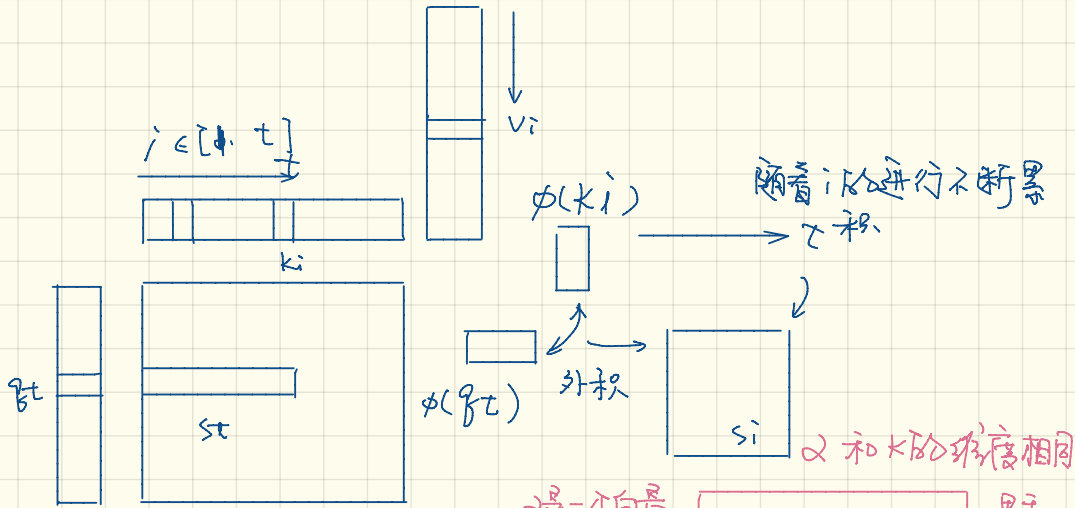
α_t



S_{t-1}

$K_t^T V_t$

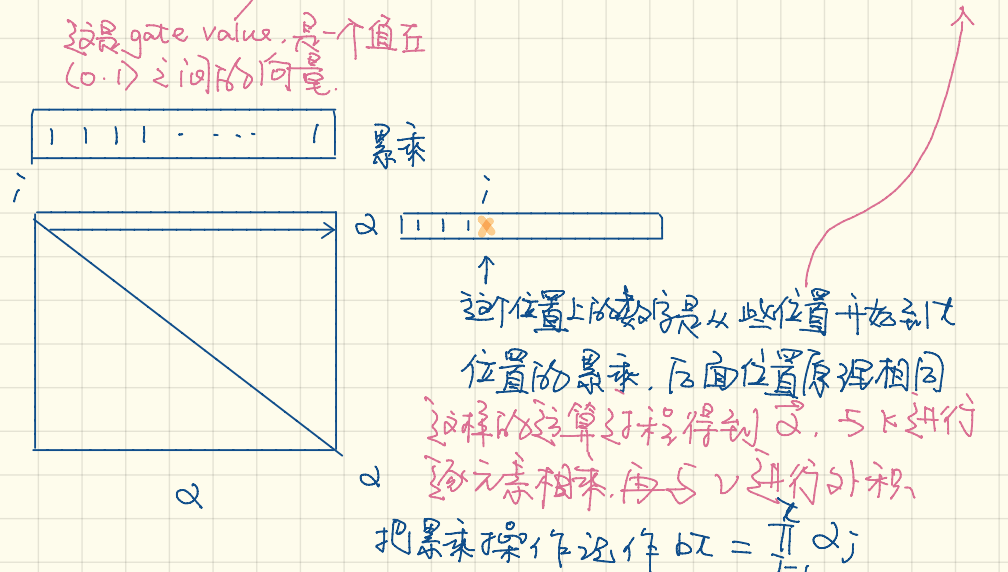
通过 x_t 上的低秩线性层 + sigmoid 得到 α_t



$$S_t = \sum_{i=1}^t \left(\phi(k_i)^T v_i \right)$$

$$= \sum_{i=1}^t \left(\left(\begin{pmatrix} 1 \\ \vdots \\ \alpha_j^T \vdots 1 \end{pmatrix} \odot k_i^T \right) v_i \right)$$

这是一个向量 α 累乘
 通过某种低秩方法 得到
 这一步是GVA设计出来的 kernel函数
 单位向量
 这是gate value, 是一个值在 (0,1) 之间的向量.

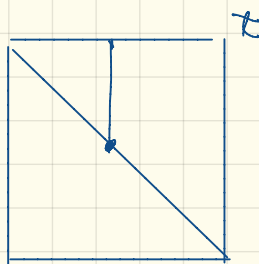
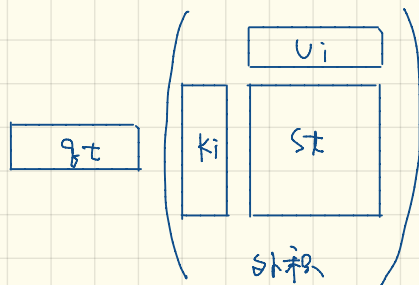
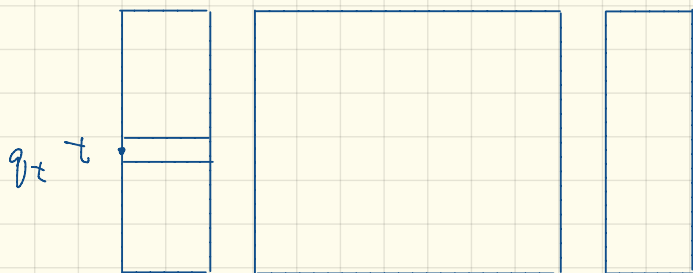
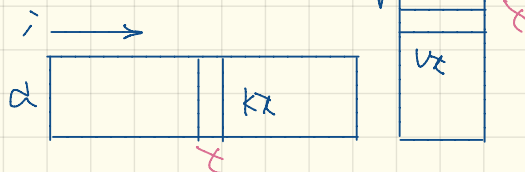


S_t 最初引入是为了能先计算 $K^T U$, 这一步得到的中间结果大小是 $d \times d$, 这对应于 $C \times C$

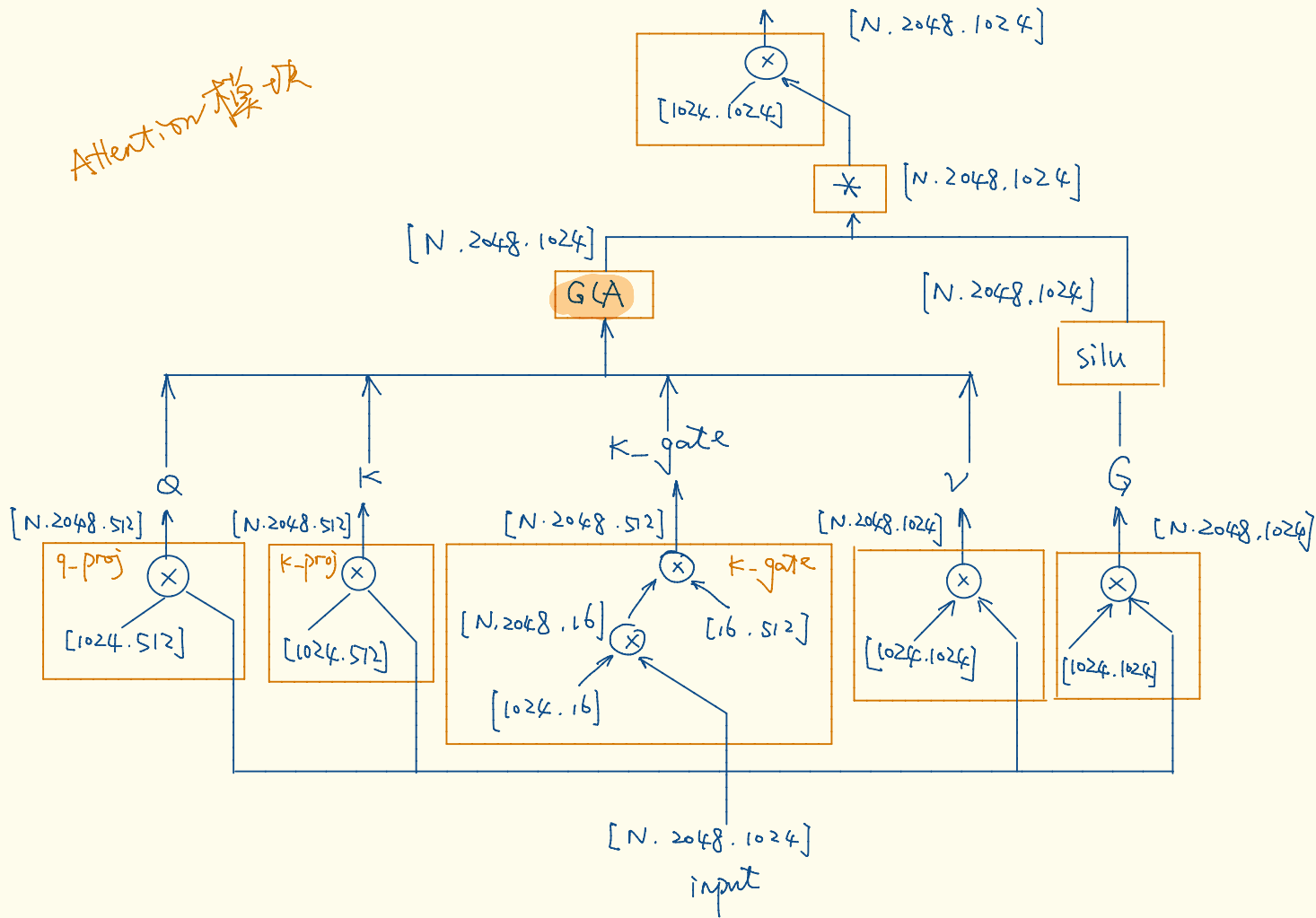
$$S_t = \sum_{i=1}^t \phi(K_i^T) U_i =$$

$$= \sum_{i=1}^t \left(\left(\prod_{j=i+1}^t \alpha_j^T \right) \odot K_i^T U_i \right)_i$$

联系



Attention 模块



$$s_t = G_t \odot s_{t-1} + k_t^T v_t \in \mathbb{R}^{d_k \times d_v} \rightarrow \text{对这一公式进行 "unfold"}$$
 这是一个典型的 RNN 形式

设计目标有两个

- ① 不用 hidden state, 减小 I/O 复杂度
- ② 全并行计算

将上述公式 unfold

$$s_1 = G_1 \odot s_0 + k_1^T v_1$$

$$s_2 = G_2 \odot s_1 + k_2^T v_2 = G_2 \odot (G_1 \odot s_0 + k_1^T v_1) + k_2^T v_2$$

$$= G_2 \odot G_1 \odot s_0 + G_2 k_1^T v_1 + k_2^T v_2$$

$$s_3 = G_3 \odot s_2 + k_3^T v_3 = G_3 \odot (G_2 \odot G_1 \odot s_0 + G_2 k_1^T v_1 + k_2^T v_2) + k_3^T v_3$$

$$= G_3 \odot G_2 \odot G_1 \odot s_0 + G_3 G_2 k_1^T v_1 + G_3 k_2^T v_2 + k_3^T v_3$$

$$s_t = G_t \odot s_{t-1} + k_t^T v_t = \sum_{i=1}^t \left(\left(\prod_{j=i+1}^t G_j \right) \odot (k_i^T v_i) \right) \rightarrow \text{这里需要 } t > 1$$
 Linear attention 的展开形式

$$G_t \odot G_{t+1} = (\alpha_t^T \beta_t) \odot (\alpha_{t+1}^T \beta_{t+1}) = (\alpha_t \odot \alpha_{t+1})^T (\beta_t \odot \beta_{t+1}) \leftarrow \text{这一步利用了外积的特性}$$

$$\prod_{j=i+1}^t G_j = \left(\prod_{j=i+1}^t \alpha_j \right)^T \left(\prod_{j=i+1}^t \beta_j \right)$$

$$A_t = \prod_{j=1}^t \alpha_j$$

$$= \left(\frac{A_t}{A_i} \right)^T \left(\frac{B_t}{B_i} \right) \frac{B_t}{B_i}$$

$$s_t = \sum_{i=1}^t \left(\frac{A_t}{A_i} \right)^T \left(\frac{B_t}{B_i} \right) \odot (k_i^T v_i)$$

到这一步, 所有的计算都是 element-wise 的

继续将 S_t 代入 O_t

$1 \sim T$ 步 gating rate 反馈

$$A_t = \prod_{j=1}^t \alpha_j \in \mathbb{R}^{1 \times d_k} \quad [\text{---}]$$

$$B_t := \prod_{j=1}^t \beta_j \in \mathbb{R}^{d_v \times 1} \quad [\text{---}]$$

$$O_t = Q_t^T S_{t-1}$$

$$= Q_t^T \sum_{i=1}^T \left(\frac{A_t}{A_i} \right)^T \left(\frac{B_t}{B_i} \right) \odot (K_i^T V_i)$$

$$= \sum_{i=1}^T Q_t^T \left(\frac{A_t}{A_i} \right)^T \left(\frac{B_t}{B_i} \right) \odot (K_i^T V_i)$$

$$= \left(\sum_{i=1}^T \left(Q_t^T \odot \frac{A_t}{A_i} \right)^T \left(\frac{K_i}{A_i} \right)^T \left(\frac{V_i}{B_i} \right) \right) \odot B_t \in \mathbb{R}^{d_v \times 1}$$

这是一个矩阵乘法
(外积矩阵乘)

这是一个外积矩阵
乘法

$$\tilde{O} = (\tilde{Q} \tilde{K}^T \odot M) \tilde{V} \odot B$$

$$= (\tilde{Q} \tilde{K}^T \odot M) \tilde{V} \odot B$$

$$\tilde{Q} = Q \odot A \quad \tilde{K} = K / A \quad \tilde{V} = V / B \rightarrow \text{全并行形式}$$