终于铺垫了这么多，可以总结一下做了这么久工作的核心网络——Mamba。尽管Mamba被某些学者认为是MambaOUT，但我认为就Mamba在医学影像处理领域的作用来看，处理医学高分辨率图像时能够取得和Transformer相似的效果的同时，保持线性的时间复杂度。特别在分辨复杂医学结构时，Mamba通过上下文提取全局特征，通常表现出比CNN更加优秀的结构理解能力。

Day8: Mamba

Mamba是一种新的选择性结构状态空间模型，在长序列建模任务中表现出色。

- Mamba通过全局感受野和动态加权，缓解了卷积神经网络的建模约束。
- Mamba提供了类似于Transformers的高级建模能力，而不会产生通常与Transformer相关的二次计算复杂性。

1. Mamba的背景

Mamba 集中了循环神经网络（RNN）的循环框架、Transformer 的并行计算和注意力机制、状态空间模型（SSM）的线性特性。因此，为了透彻地理解 Mamba，就必需先理解这三种架构。

- Transformer的二次复杂度：计算复杂度和序列长度的平方N^2成正比（矩阵乘法）
- RNN在序列中的每个时间步需要两个输入，即时间步t的输入x_t和前一个时间步$t-1$的隐藏状态h_{t-1}，以生成t时的隐藏状态h_t，最终预测输出y_t。但虽然每个隐藏状态都是所有先前隐藏状态的聚合，然随着时间的推移，RNN 往往会忘记某一部分信息，且RNN没法并行训练，相当于推理快但训练慢。
- 也就是Mamba所优化的基线状态空间模型SSM，详细可以参考之前总结的SSM-1和SSM-2。

考虑到Transformer的二次复杂度和RNN无法并行计算的特点，诞生了可以并行推理计算且

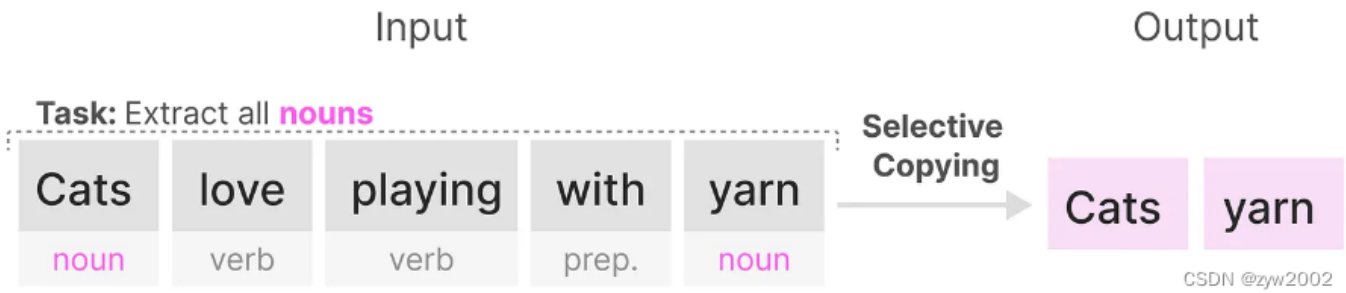
执行与序列长度线性扩展推理的Mamba。

S4的缺点

Mamba的原文标题是Mamba将SSM或者S4优化成S6。主要是SSM / S4的线性时间不变性决定其是静态的，因为无法选择性的关注指定的输入。

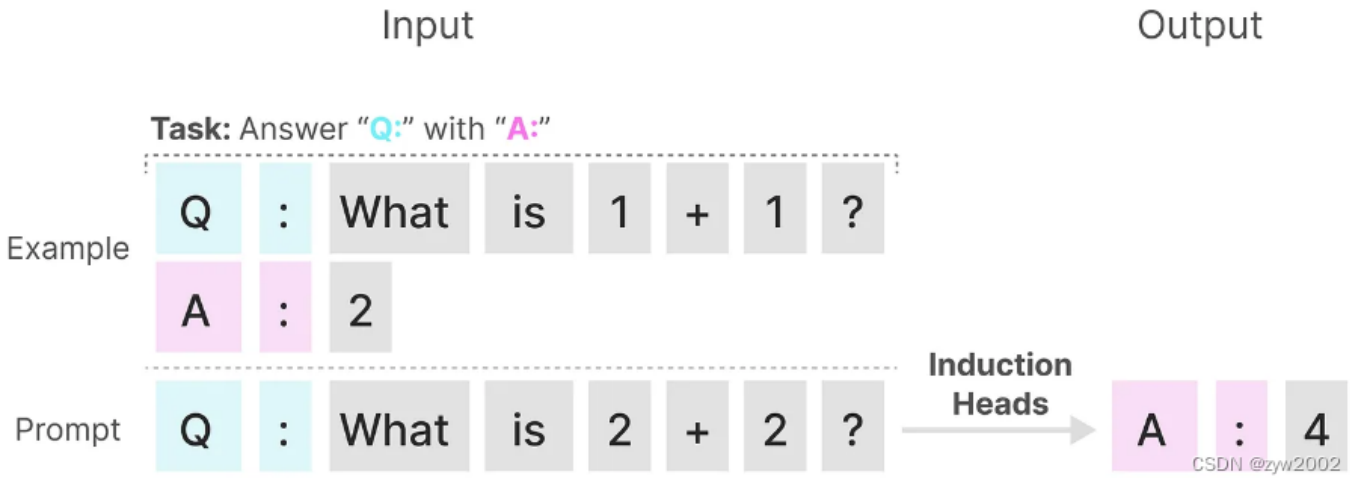
比如

- 在选择性复制任务中，SSM的目标是复制输入的一部分并按顺序输出：



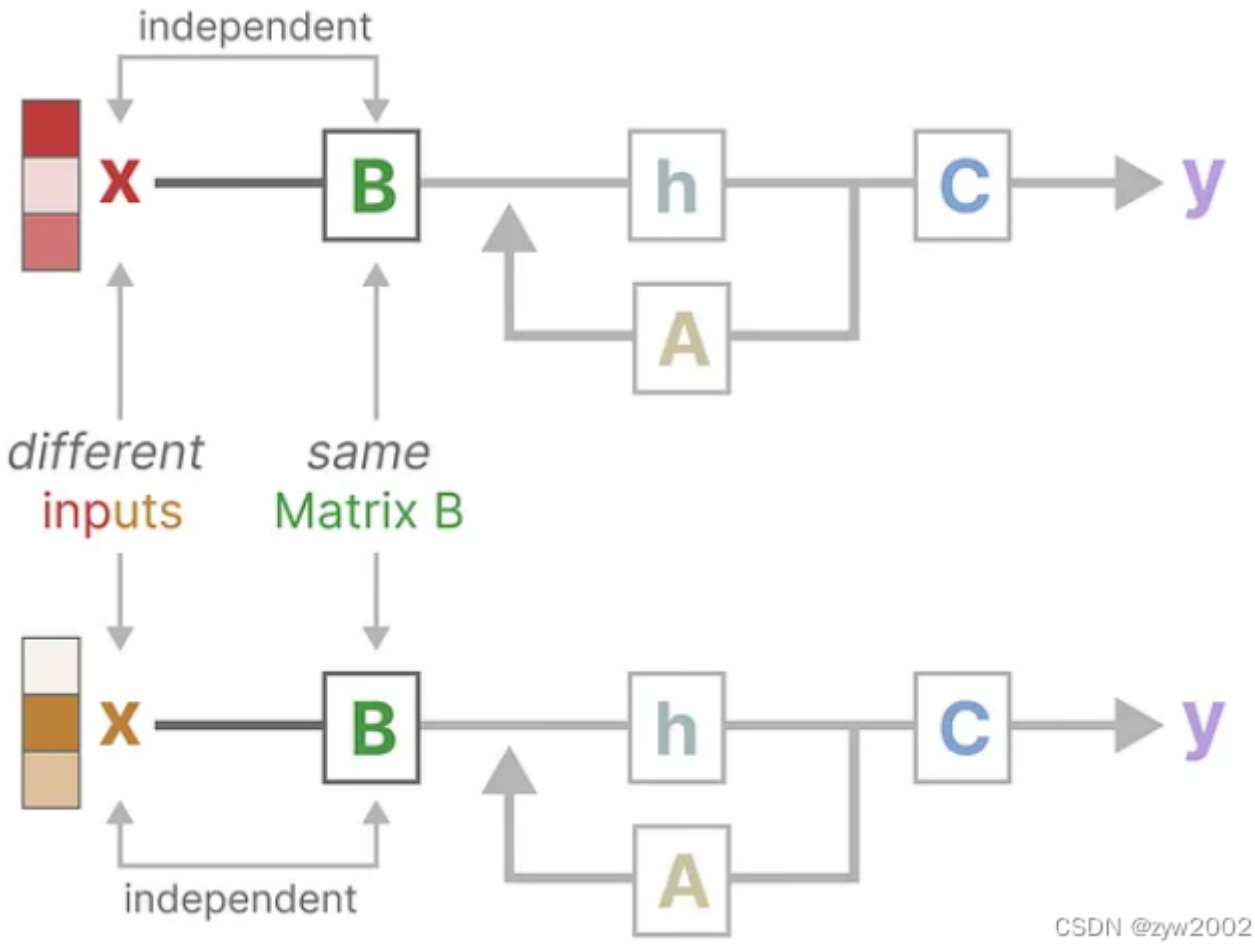
然而，(循环/卷积)SSM在这项任务中表现不佳，因为它是线性时间不变（Linear Time Invariant）的，矩阵A、B和C对于SSM生成的每个token都是相同的。因此，SSM无法进行内容感知推理（content-aware reasoning），因为它将每个token视为固定的A、B和C矩阵的结果。这是一个问题，因为我们希望SSM对输入(提示)进行推理。

- 目标是重现在输入中发现的模式：



本质上是在执行one-shot prompting，试图"教"模型在每个"Q:"之后提供一个"A:"响应。然而，由于ssm是时不变的，它无法选择从历史中回忆之前的哪个标记。

具体让我们以矩阵B为例来说明这一点。无论输入x是什么，矩阵B都是完全相同的，因此与x无关。



同样，无论输入是什么，A和C也保持固定。这表明了我们迄今为止看到的SSM都是**静态**的。相比之下，这些任务对transformer来说相对容易，因为它们根据输入序列动态改变注意力。他们可以选择性地"看"或"关注"序列的不同部分。

SSM在这些任务上的糟糕表现说明了time-invariant SSM的潜在问题，矩阵A、B和C的静态性质导致了其无法进行内容感知（content-awareness）。

2. Mamba

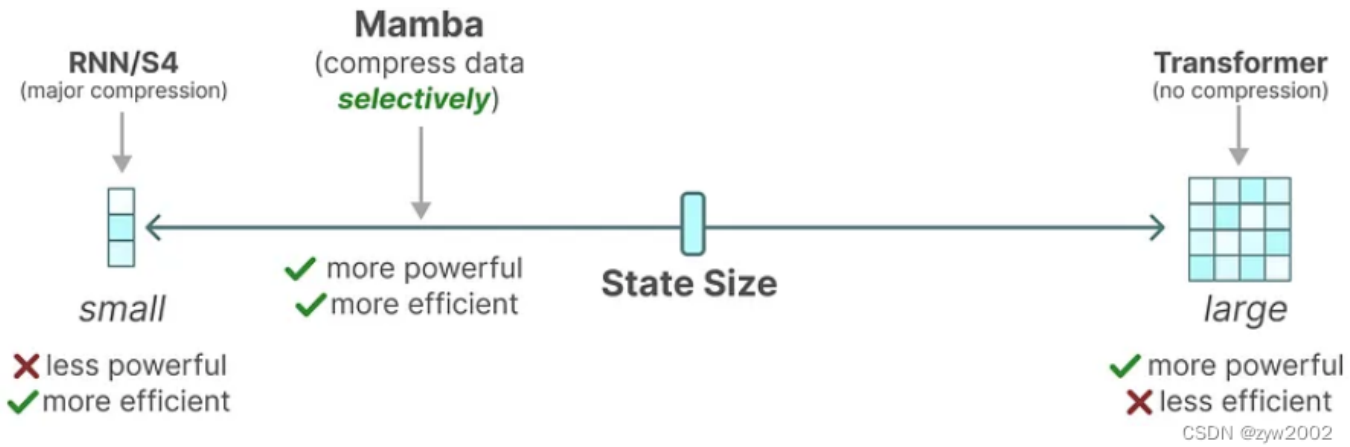
Mamba主要改进体现在以下两点：

- 选择性扫描算法(selective scan algorithm），允许模型过滤相关或者不相关的信息
- 硬件感知的算法(hardware-aware algorithm)，允许通过并行扫描(parallel scan)、核融合(kernel fusion)和重计算(recomputation)有效地存储(中间)结果。

(1) 选择性的保留信息

SSM的循环表示创建了一个非常高效的小状态，因为它主要是利用了HIPPO压缩了整个历史状态。然而，与没有压缩历史状态(通过attention map)的Transformer相比，它的性能要差的多。

Mamba 致力于保留一个小的且有用的状态信息， 兼顾性能和效率。

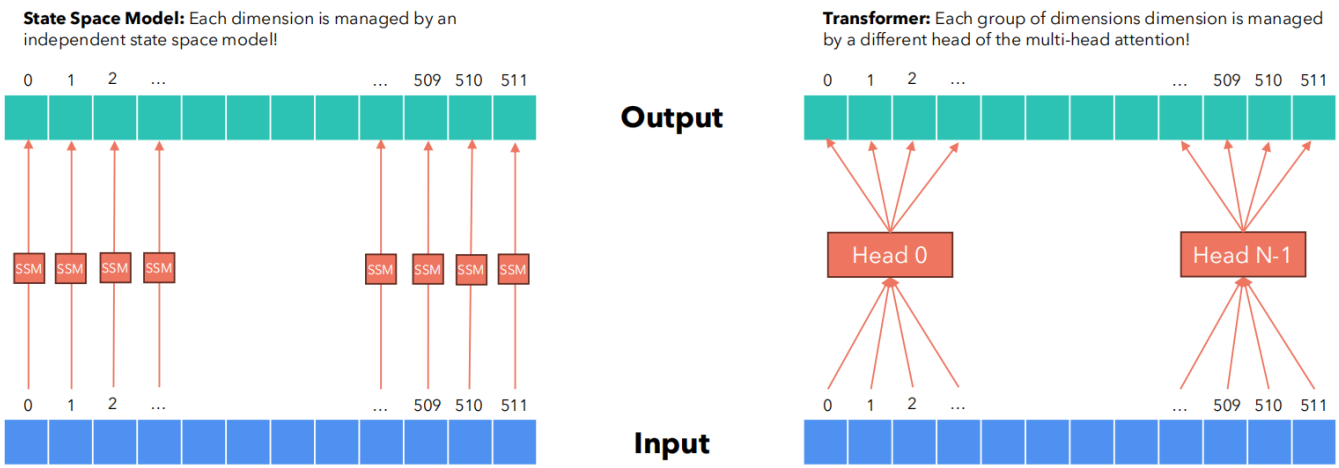


如上所述，它通过有**选择地将数据压缩到状态**中来实现这一点。（当有一个输入句子时，通常会有一些信息，比如标点，没有太多意义。这些无意义的信息就可以被忽略掉。）为了有选择地压缩信息，我们需要**参数依赖于输入**。

- SSM(S4)

```
Algorithm 1 SSM (S4)
1: x : (B, L, D)
2: y : (B, L, D)
3: A : (D, N) ← Parameter          ▷ Represents structured N×N matrix
4: B : (D, N) ← Parameter
5: C : (D, N) ← Parameter
6: Δ : (D) ← τΔ(Parameter)
7: Ā, B̄ : (D, N) ← discretize(Δ, A, B)
8: y ← SSM(Ā, B̄, C)(x)         ▷ Time-invariant: recurrence or convolution
9: return y
```

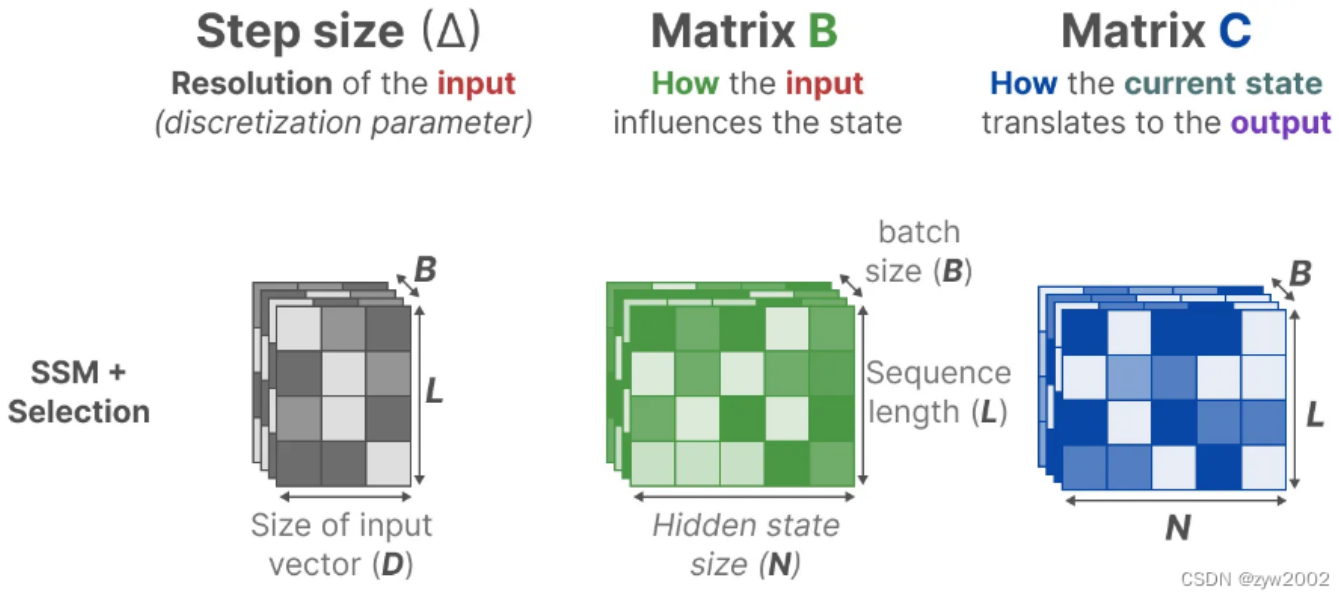
输入张量 `x` 和输出张量 `y` 的形状为 `(B, L, D)`，其中 `B` 是批次大小， `L` 是序列长度， `D` 是特征维度。Mamba通过给每一个维度 `D` 都设置一个独立的SSM，且每个维度之间相互独立。



类似于Transformer的多头注意力处理，每个token默认设为8个注意力头。

Mamba通过合并输入的序列长度和批次大小，使矩阵B和C，甚至步长Δ依赖于输入。（矩阵A保持不变，因为我们希望状态本身保持静态，但它被影响的方式`通过B和C`是动态的。）

- SSM + S6



这意味着对于每个输入标记，我们现在有不同的B和C矩阵，这解决了内容感知的问题。这样就可以依赖于输入，选择什么保持在隐藏状态，什么要忽略。

```
Algorithm 2 SSM + Selection (S6)
1: x : (B, L, D)
2: y : (B, L, D)
3: A : (D, N) ← Parameter          ▷ Represents structured N×N matrix
4: B : (B, L, N) ← sB(x)
5: C : (B, L, N) ← sC(x)
6: Δ : (B, L, D) ← τΔ(Parameter + sΔ(x))
7: Ā, B̄ : (B, L, D, N) ← discretize(Δ, A, B)
8: y ← SSM(Ā, B̄, C)(x)         ▷ Time-varying: recurrence (scan) only
9: return y
```

Δ的大小由原来的D变成了(B,L,D)，意味着对于一个 batch 里的每个 token (总共有 $B \times L$ 个)都有一个独特的Δ。且每个位置$B, C,$ Δ都不相同，可以解决内容感知问题。具体的计算如下：

- **B**：使用选择函数 $s_B(x) = LinearN(x)$，使 B 成为 $(B \times L \times N)$的输入依赖参数。
- **C**：使用选择函数 $s_C(x) = LinearN(x)$生成输入依赖参数C。
  - 对于BC的线性操作，把D维的输入向量经过一个线性层影射到N维，类似于$64 \times 3(N \times D)$变成$1000 \times 64(N \times D)$。
- **Δ**：通过组合函数 $T_\Delta(Parameter + s_\Delta(x))$获得随输入变化的 Δ。
  - 更进一步解释：当步长较小（即Δ较小）时，模型更倾向于忽略特定的单词，而更多地依赖前一个上下文。这意味着模型更注重前面的单词对当前单词的影响，而忽略了较远距离的单词。相反，当步长较大（即Δ较大）时，模型更多地关注当前输入单词而不是上下文。这意味着模型更多地考虑当前输入单词对上下文的影响，而不是依赖于前一个上下文来决定当前单词的特征。

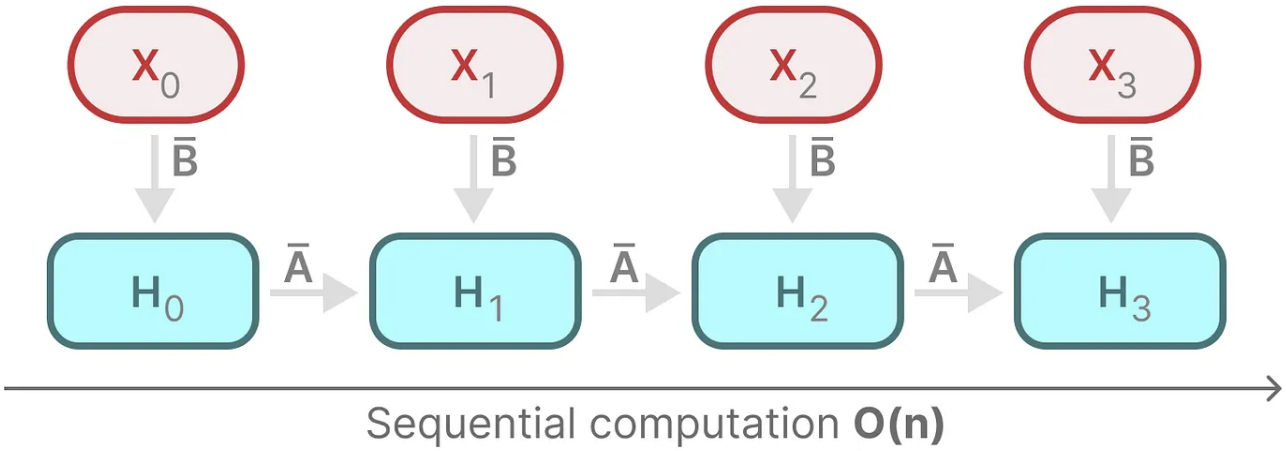
Δ类似于遗忘门，B和C类似于RNN的memory。总之，Mamba通过合并输入的序列长度和批量大小来使矩阵B和C，甚至步长Δ取决于输入(其意味着对于每个输入*token*，现在有不同的*B*和*C*矩阵，可以解决内容感知问题)，从而达到选择性地选择将哪些内容保留在隐藏状态以及忽略哪些内容的目标。

（2）硬件感知的设计

如之前所述，由于A B C这些矩阵现在是动态的了，因此无法使用卷积表示来计算它们(CNN需要固定的内核)，因此，我们只能使用循环表示，如此也就而失去了卷积提供的并行训练能力。

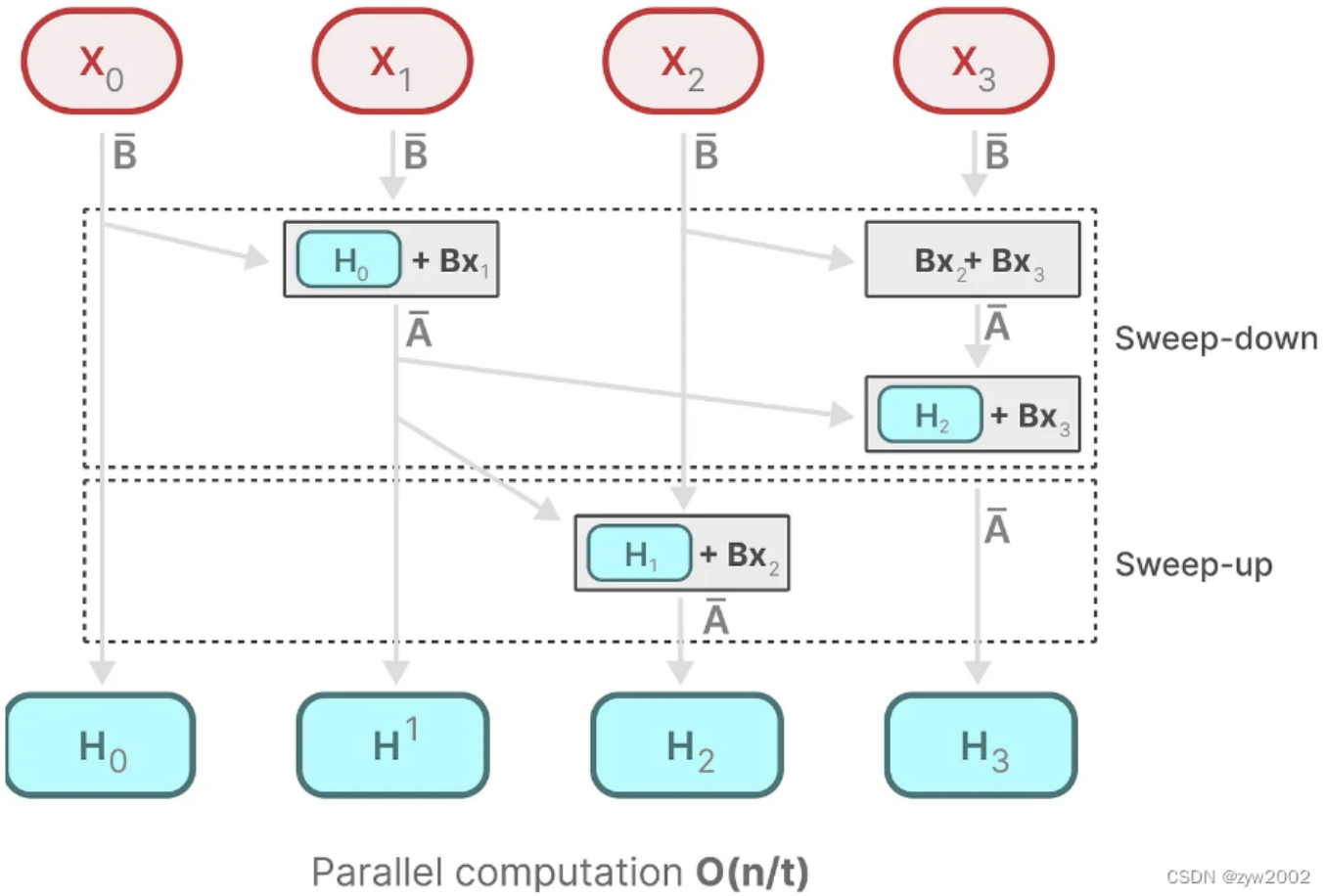
并行扫描算法是一种允许在保持循环计算特性的同时，对序列数据进行并行处理的技术。这种方法可以在处理序列时，对序列的各个部分同时进行计算——而不是一个接一个地处理，从而实现并行化。

一般S4的计算是这样的：



只有在获取到前一个状态的情况下才能计算当前的每个状态。此时利用for循环迭代，时间复杂度是O(N)。

我们可以分段计算序列并迭代地组合它们，即动态矩阵B和C以及并行扫描算法一起创建：选择性扫描算法(selective scan algorithm)



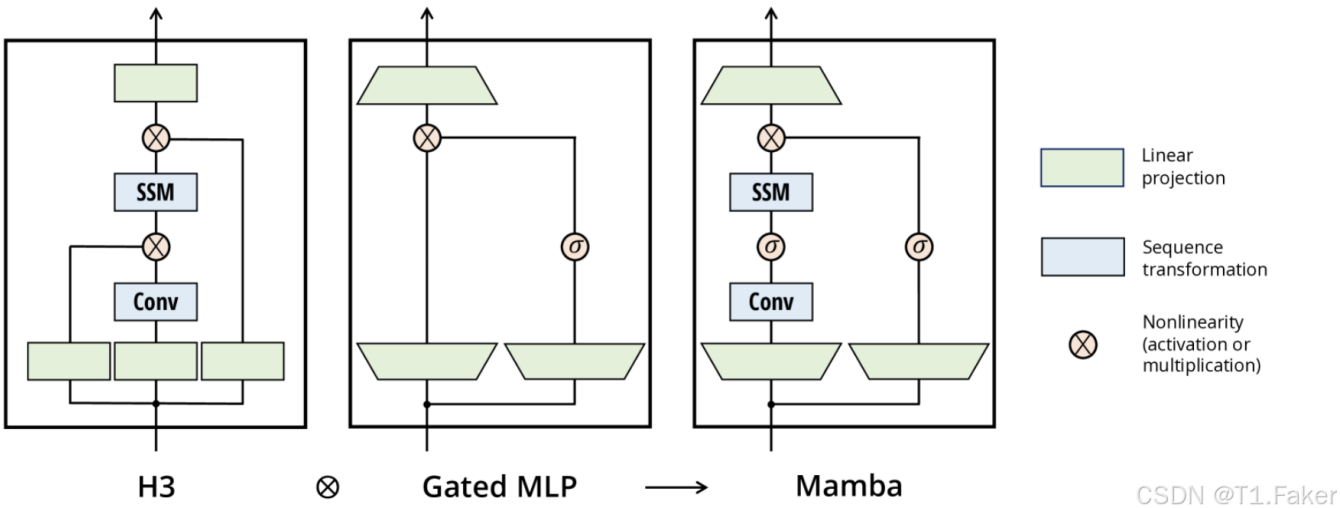
简单来说就是H_1的计算不需要等待H_0计算完毕。

拿图上的H_3举例，可以展开为$H_3 = \bar{A} \cdot (\bar{A} \cdot (\bar{A}H_0 + BX_1) + BX_2) + BX_3$。此时，每个状态都可以并行计算。

此外，为了让传统的SSM在现代GPU上也能高效计算，Mamba中也使用了Flash Attention技术。简而言之，利用内存的不同层级结构处理SSM的状态，减少高带宽但慢速的HBM内存反复读写这个瓶颈

（3）Mamba的架构

Mamba 架构在设计上结合了选择性状态空间模型（SSM）和多层感知机（MLP）模块，构建出一种高效处理长序列的模型，尤其适用于减少计算成本并提升性能稳定性。该架构通过门控机制和选择性扫描操作，优化了对长序列的处理，且在多种应用中展现了良好的效果，如下图所示：

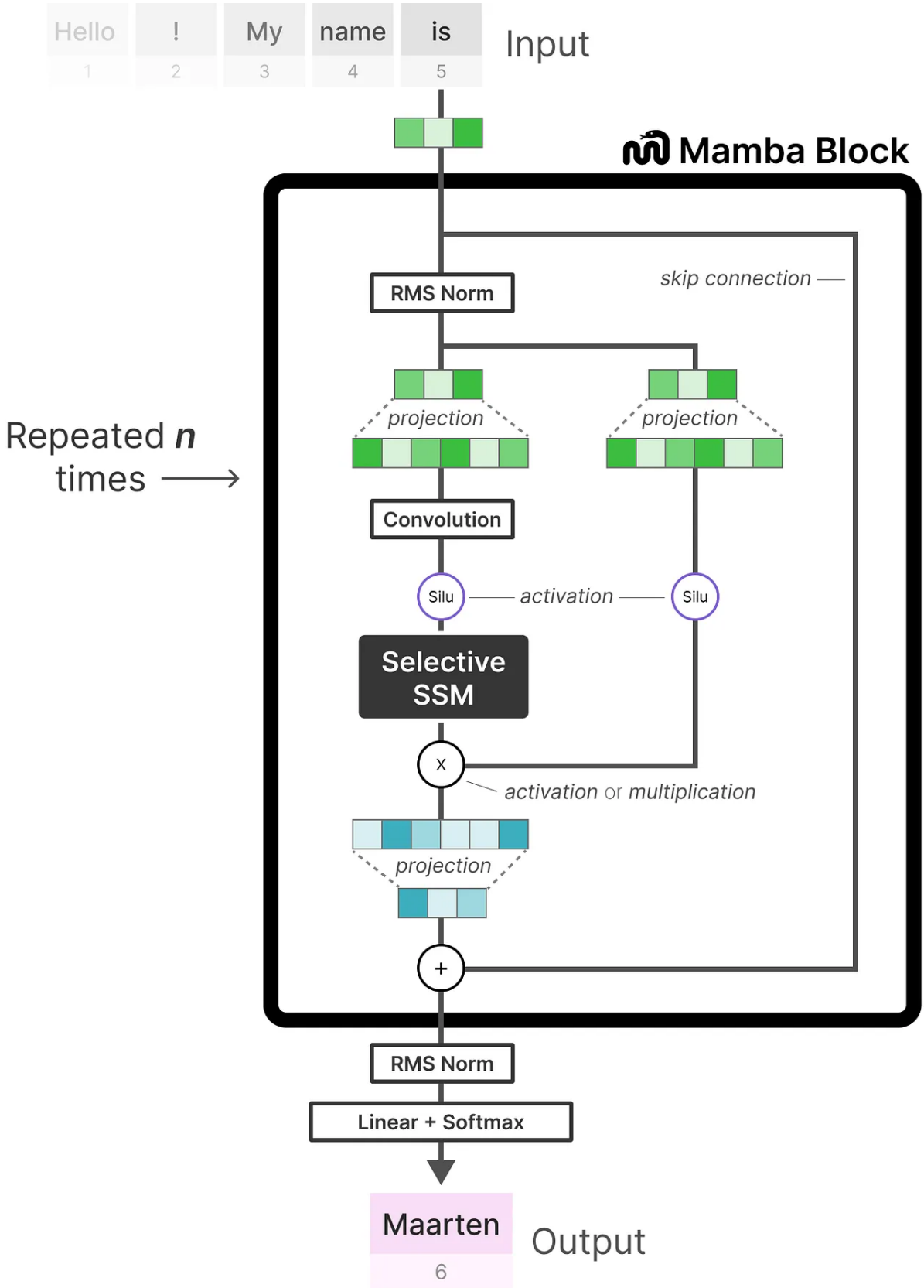


- H3 架构基础 H3 架构主要由状态空间模型（SSM）和卷积层组成，具备高效处理长序列的特点。这里的 SSM 是一种递归模块，旨在维持序列的状态，通过输入动态更新模型状态。
    - SSM 层（State-Space Model Layer）：这个模块通过状态空间形式来描述模型行为，用于跟踪输入序列的状态并进行递归预测。每个时间步都会更新状态，使模型能够持续记忆前序信息。
    - 卷积层（Convolution Layer）：卷积层作用于输入序列，将其转换成适合 SSM 处理的状态序列。通过卷积层的应用，H3 可以将序列的局部特征提取为 SSM 所需的输入，有助于提升序列间依赖信息的表达能力。
- 门控 MLP 模块

  门控 MLP 模块（Gated MLP）是 Mamba 的另一个核心组件，用于增强网络的非线性表达能力，主要由多层感知机（MLP）结构和门控机制组成。
    - 多层感知机（MLP）：MLP 部分负责将输入映射到高维特征空间，通过多个全连接层（fully connected layers）加深网络的特征捕获能力。为了提升性能，该模块引入了 SwiGLU（Swish-Gated Linear Units）变体，增加了更复杂的激活模式。

    - 门控机制（Gating Mechanism）：通过引入门控结构，模块可以选择性地通过或过滤掉输入信息。Swish 或 SiLU 激活函数通常用于门控层，能够动态调节信息流动，提高信息的非线性表达效果。这种选择性过滤操作有助于模型在处理长序列时聚焦于关键输入，避免噪声干扰。

- 曼巴块的设计： Mamba 块是 H3 架构和门控 MLP 的结合体，通过将两者与额外的 SSM 层组合成一个模块，形成了 Mamba 独有的架构。
    - 层级结构：在 Mamba 模块中，SSM 层、卷积层、门控 MLP 层依次排列。每个模块间通过规范化（Normalization）和残差连接（Residual Connections）连接起来，提升了模型的稳定性和优化效果。
    - 去除乘法门和增加激活函数：相比 H3 模块，Mamba 模块中去除了第一个乘法门，用激活函数（如 SiLU）取而代之。这样能够在保留 H3 模块基本结构的前提下，提升 MLP 的处理灵活性。
    - 附加的 SSM 层：Mamba 模块增加了一层 SSM，在主分支中引入状态空间模型，使得模块具备更强的序列处理能力。这种设计在模块内部构建了多层次的状态跟踪能力，使得模型能有效管理和优化长序列中的重要上下文。

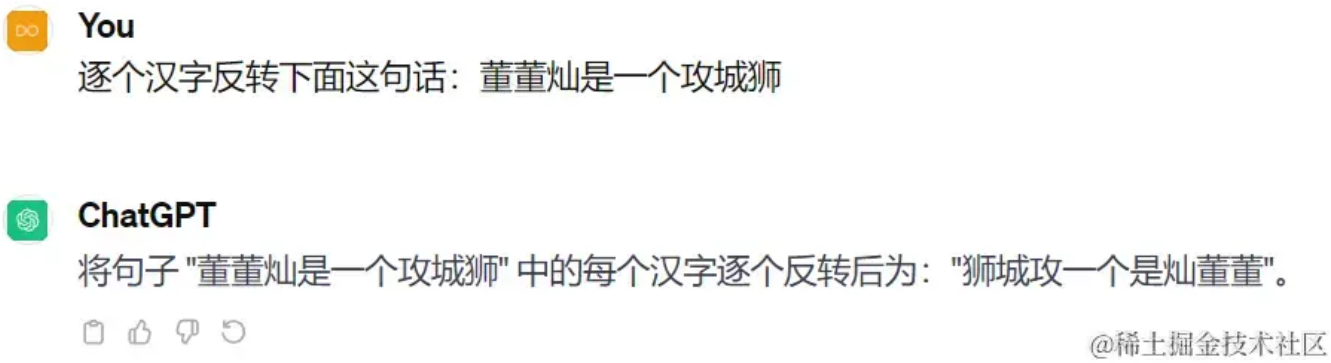


此外，关于mamba的整体架构，有两点值得强调下

Q：为何要做线性投影 经过线性投影后，输入嵌入的维度可能会增加，以便让模型能够处理更高维度的特征空间，从而捕获更细致、更复杂的特征。 Q：为什么SSM前面有个卷积？ 本质是对数据做进一步的预处理，更细节的原因在于： （1）SSM之前的CNN负责提取局部特征(因其擅长捕

捉局部的短距离特征)，而SSM则负责处理这些特征并捕捉序列数据中的长期依赖关系，两者算互为补充。 （2）CNN有助于建立token之间的局部上下文关系，从而防止独立的token计算。 毕竟如果每个 token 独立计算，那么模型就会丢失序列中 token 之间的上下文信息。通过先进行卷积操作，可以确保在进入 SSM 之前，序列中的每个 token 已经考虑了其邻居 token 的信息。这样，模型就不会单独地处理每个 token，而是在处理时考虑了整个局部上下文。

术语token：在自然语言处理（NLP）中，**token** 是文本处理的基本单位。它可以是一个单词、一个词组、一个标点符号、一个字符等，具体取决于文本处理的需求和方法。将文本划分为若干个 token 是文本处理的第一步，这个过程被称为 **tokenization**。例如：



可以看到，句子中的“一个”反转之后仍然是“一个”，而不是"个一"。这可能就是因为在GPT3.5模型处理时，“一个” 被当做了一个 token 来对待，而这又是一个基本单元，无法再进一步拆分完成反转。