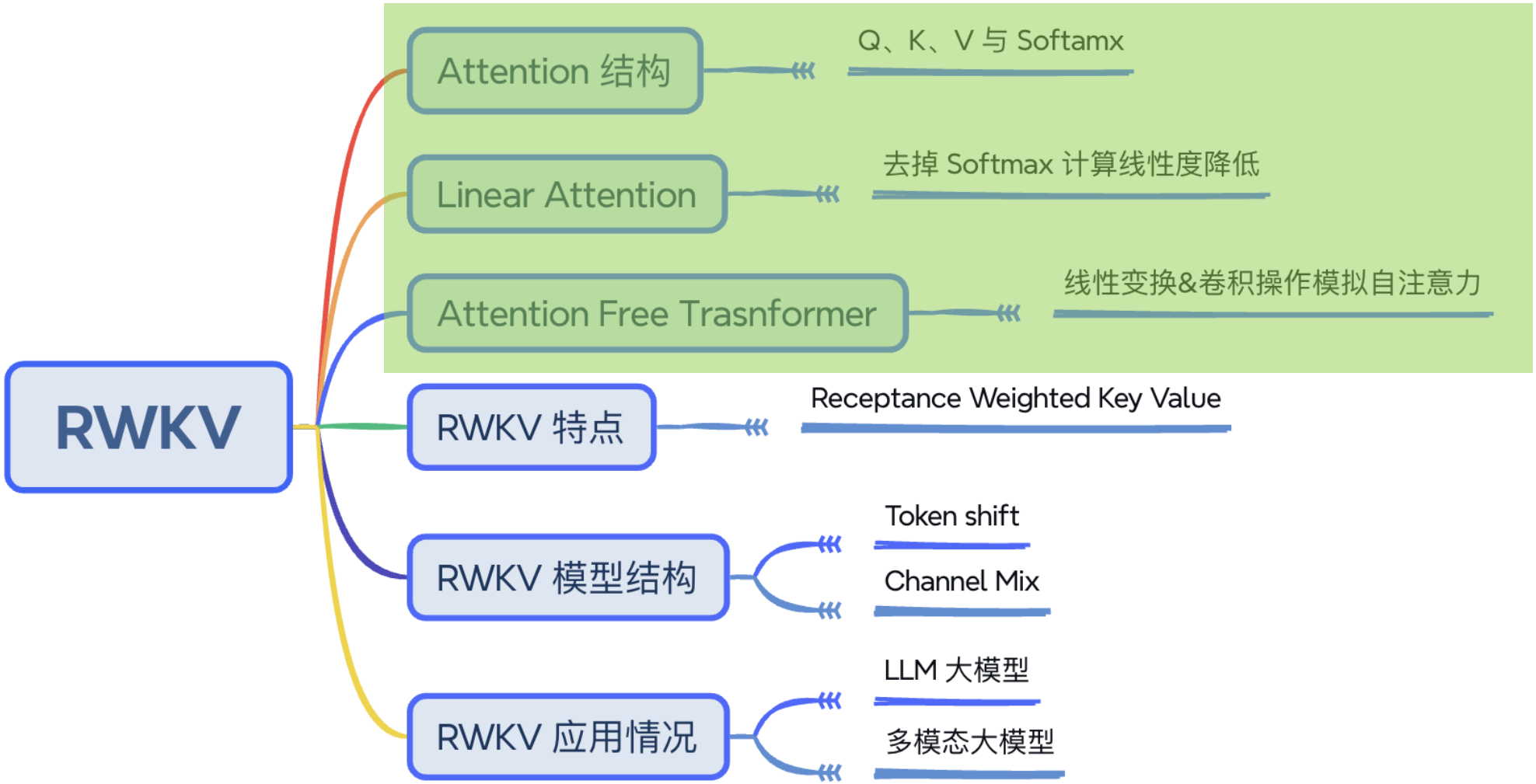
大模型系列 – 大模型算法

Linear Attention

ZOMI

关于本内容

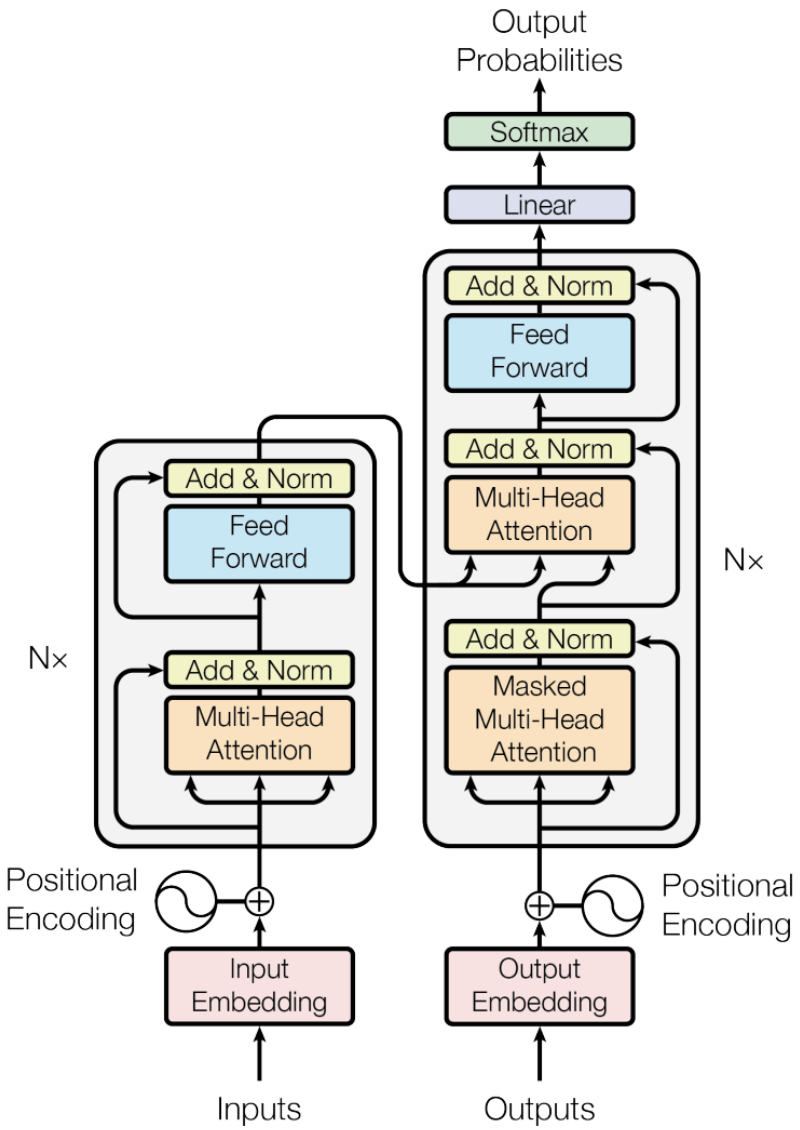

RWKV
Attention 结构
Q、K、V 与 Softamx
Linear Attention
去掉 Softmax 计算线性度降低
Attention Free Trasnformer
线性变换&卷积操作模拟自注意力
RWKV 特点
Receptance Weighted Key Value
RWKV 模型结构
Token shift
Channel Mix
RWKV 应用情况
LLM 大模型
多模态大模型

1 Attention 结构的问题

Transformer Attention

BERT

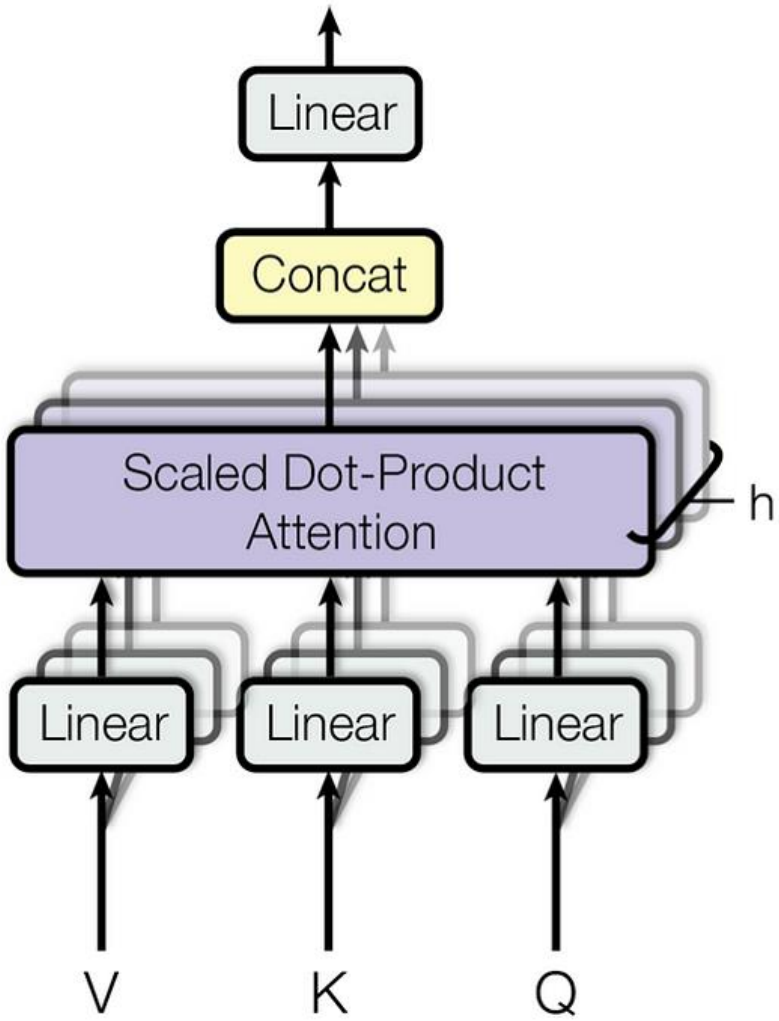
Encoder

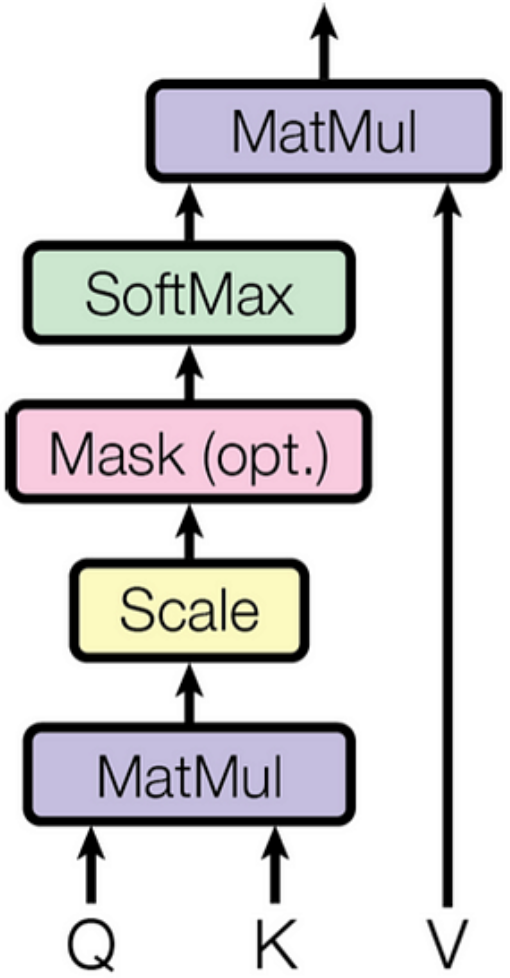


GPT

Decoder

Dot-Product Attention

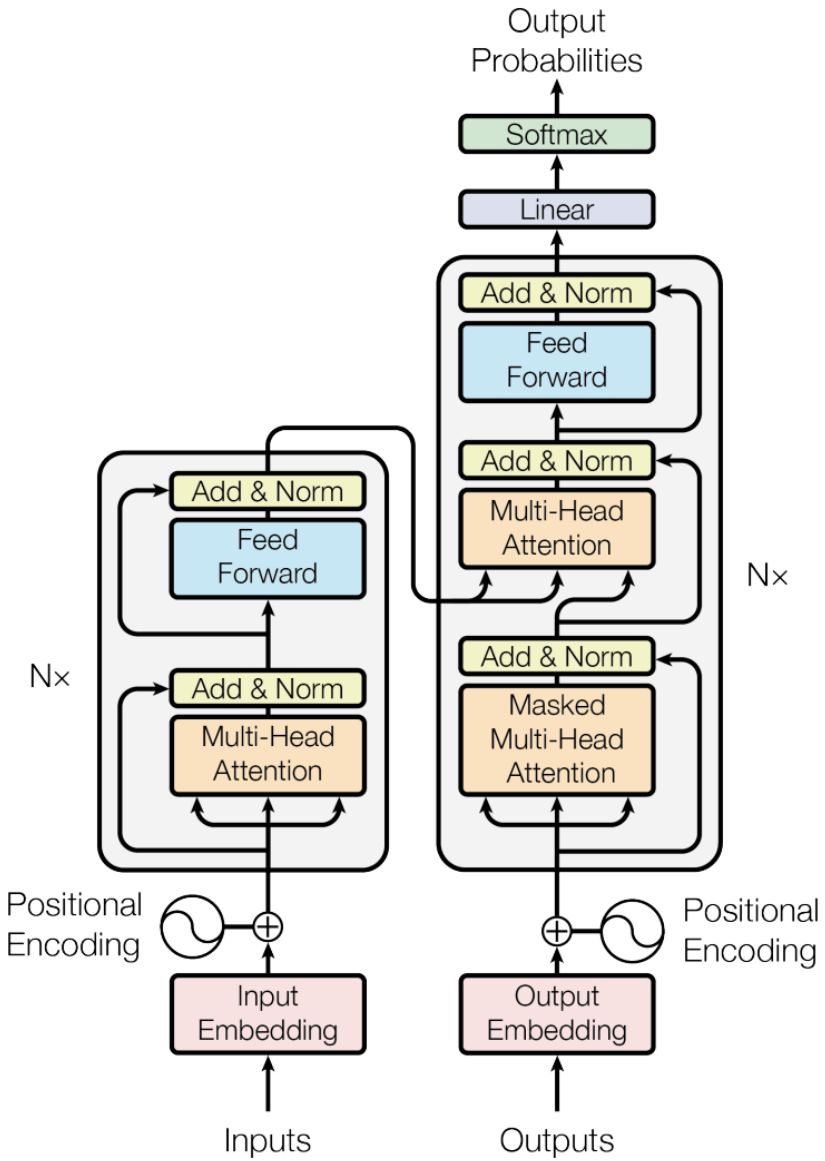




Transformer 结构

- $x \in \mathbb{R}^{N\times F}$，$N$ 个 F 维的特征向量。Transformer 即一个函数 $T: \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times F}$，由 L 个 Transformer 层 $T_1(\cdot), \ldots, T_L(\cdot)$ 组成：

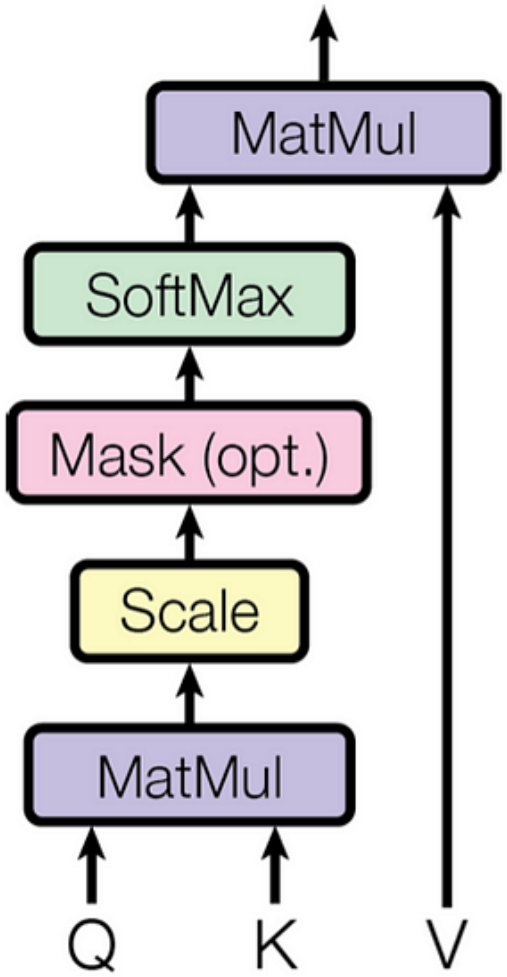
$$T_l = f_l(A_l(x) + x)$$



Transformer 结构

- $A_l(\cdot)$ 代表自注意力函数，即 Self Attentio n。输入序列 x 由三个矩阵 $W_Q \in \mathbb{R}^{F\times D}$，$W_{\mathrm{K}} \in \mathbb{R}^{F\times D}$，$W_v \in \mathbb{R}^{F\times M}$ 映射到 Q, K, V，$A_l(x) = V'$:

$$Q = xW_Q, K = xW_k, V = xW_v$$

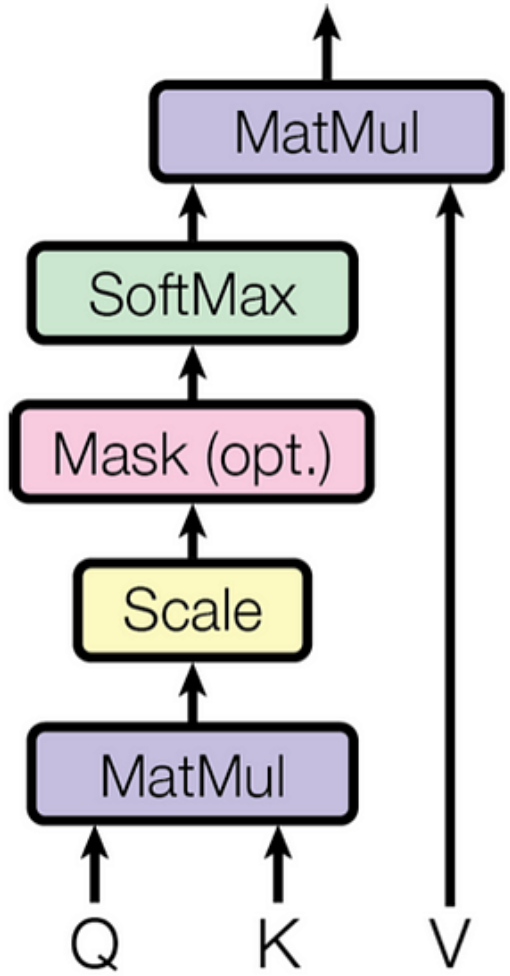


Transformer 结构

- $Softmax$ 函数按行应用与 QK^T， Q, K, V 分别表示 queries，keys 和 values：

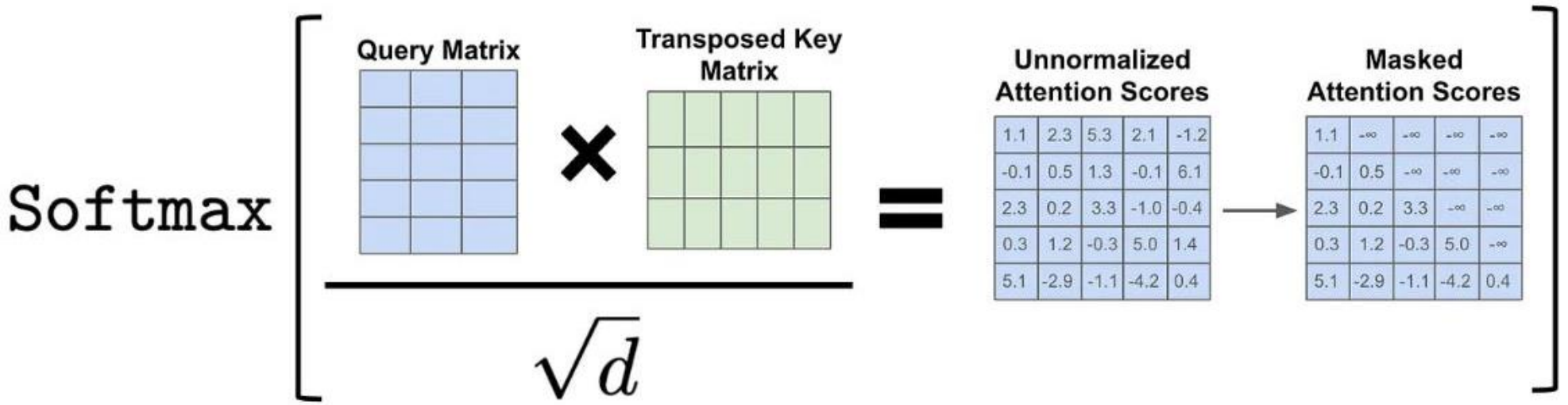
$$A_l(x) = V' = softmax\left(\frac{QK^T}{\sqrt{D}}\right)V$$

- 其中相似性是由 Q 和 K 点积的指数表示。



Attention 的计算

- 尽管基于 Attention 的 Transformer 有着良好并行性能，但空间和时间复杂度都是 $O(n^2)$，n 是序列长度，所以当 n 比较大时 Transformer 模型计算量非常夸张。



2 Linear Attention

Linear Attention

- 制约 Attention 性能关键因素，其实是定义里的 Softmax， QK^T 这一步得到一个 $n \times n$ 的矩阵，就是这一步决定了 Attention 的复杂度为 $O(n^2)$；
- 如果没有 Softmax，那么就是三个矩阵连乘 QK^TV ，而矩阵乘法是满足结合率，所以可以先算 K^TV，得到一个 $d \times d$ 的矩阵，然后再用 Q 左乘，由于 $d \ll n$ ，所以去掉 Softmax 的复杂度是 $O(n)$ 。

去掉 Softmax 的 Attention 复杂度可以降到最理想线性级别 $O(n)$！

这就对计算的极致追求： Linear Attention！

Linear Attention

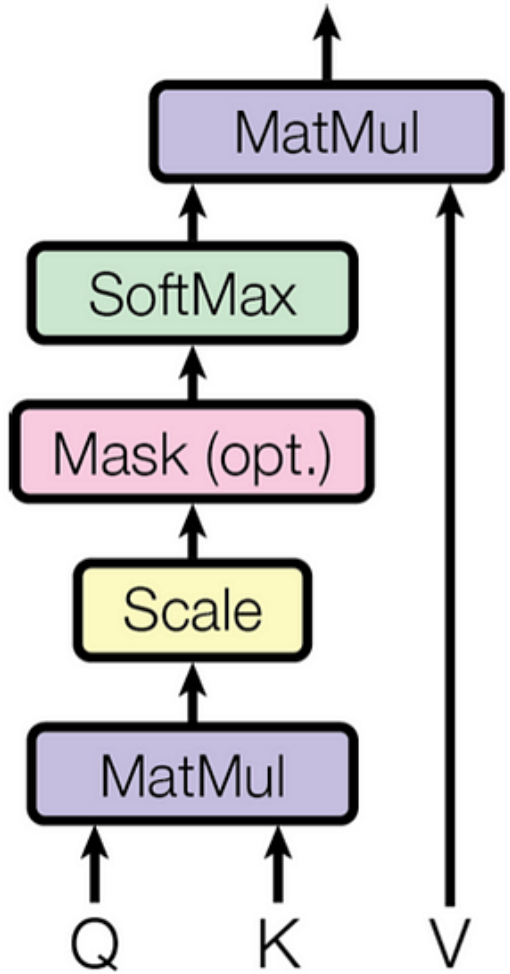
- 之前工作证明了多头自注意力 self Attention 只要有足够注意力头数就可以表示任意的卷积层。但是，Linear Attention 论文反向表明，用自回归目标训练自注意力层可以被看作是一个 RNN，可以显著加快自回归 Transformer 模型推理时间。
- 将 self-attention 表示为核特征图的线性点积，并利用矩阵乘积的结合率将复杂度从 $O(n^2)$ 降低到 $O(n)$，其中 n 为序列长度。

Linear Attention

- 给定一个下标 i，返回一个矩阵的第 行作为一个向量，对应任意相似性函数，可以写出一个广义的注意力方程：

$$V_i' = \frac{\sum_{j=1}^{N} sim(Q_i, K_j)V_j}{\sum_{j=1}^{N} sim(Q_i, K_j)}$$

- 其中，相似性函数 $\mathrm{sim}(q,k)$ 替代为 $\exp(QK^T/\sqrt{D})$ 数学等价。



Linear Attention

- 为了保留Attention相似的分布特性，需要对 $sim(\cdot)$ 施加一个非负的约束，保证 $\text{sim}(q_i, k_i) \geq 0$ 恒成立。
- 如果直接去掉Softmax，那么就是 $\text{sim}(q_i, k_i) = {q_i}^T k_i$，问题是内积无法保证非负性，所以这还不是一个合理的选择。

Linear Attention

- 如果 q_i, k_i 的每个元素都是非负的，那么内积自然也就是非负的。为了完成这点，可以给 q_i, k_i 各自加个特征表示核函数 ϕ ，则可以将广义的 self Attention 重写为：

$$V_i' = \frac{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j) V_j}{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)}$$

- 根据矩阵乘法的结合律，进一步简化：

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)\, V_j^T}{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)}$$

Linear Attention

- 当分子写成向量形式时，可以简化为：

$$(\phi(Q)\phi(K)^T)V = \phi(Q)\,(\phi(K)^T V)$$

- 特征映射 $\phi(\cdot)$ 逐行应用于矩阵 Q, K。最终 Linear transformer 时间复杂度、空间复杂度都是 $O(n)$ 的，因为可以一次计算 $\sum_{j=1}^{N} \phi(K_j)\, V_j^T$ 和 $\sum_{j=1}^{N} \phi(K_j)$ 并且在每个查询中重复使用。

Linear Attention

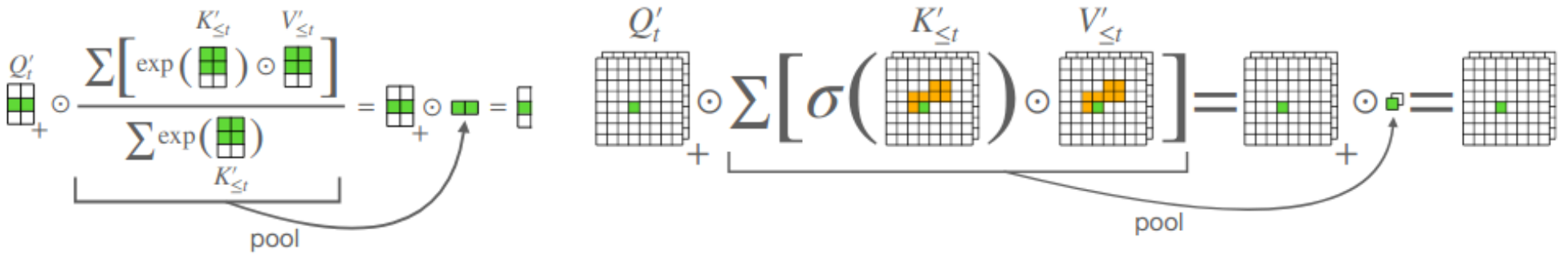
- Paper Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention 选择的是值域非负的激活函数：

$$\phi(\cdot) = elu(x) + 1$$

Attention Free Transformer

Attention Free Transformer

- Attention Free Transformer (AFT) 是 Apple 提出的一种新型的神经网络模型，它在传统 Transformer 基础上，消除了点积自注意力，使内存复杂度从 $O(Td^2)$ 变成 $O(Td)$，其中 T 是序列长度，d 是嵌入的维数。

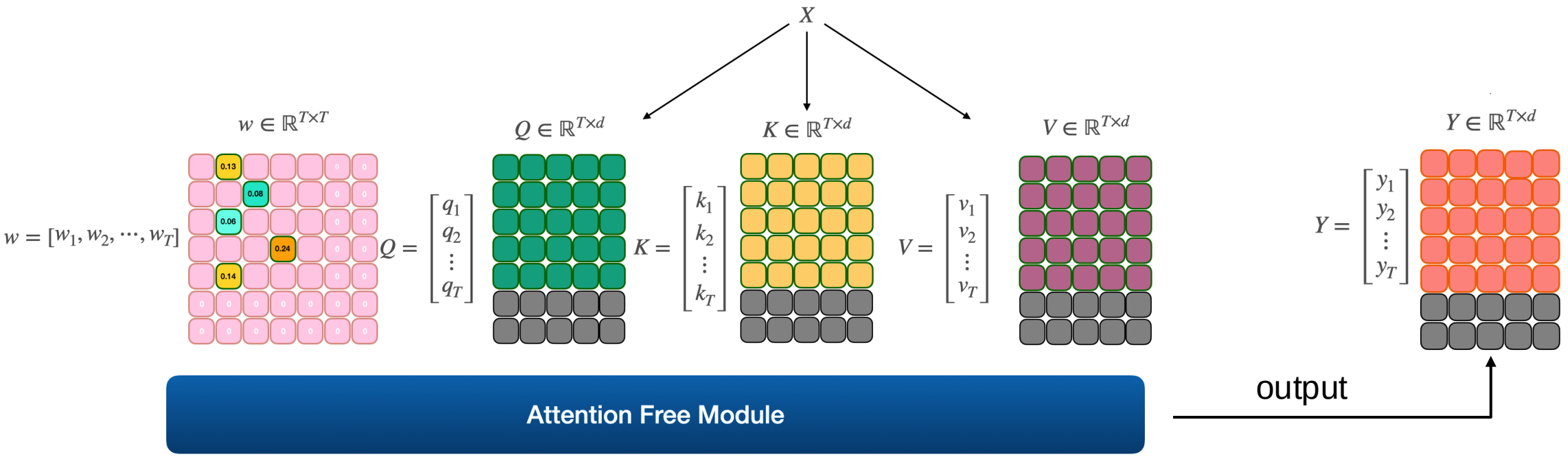


(a) Causal attention free operation. (b) Local causal attention free operation.

Figure 1: AFT blocks require only element-wise and pooling operations.

Attention Free Transformer

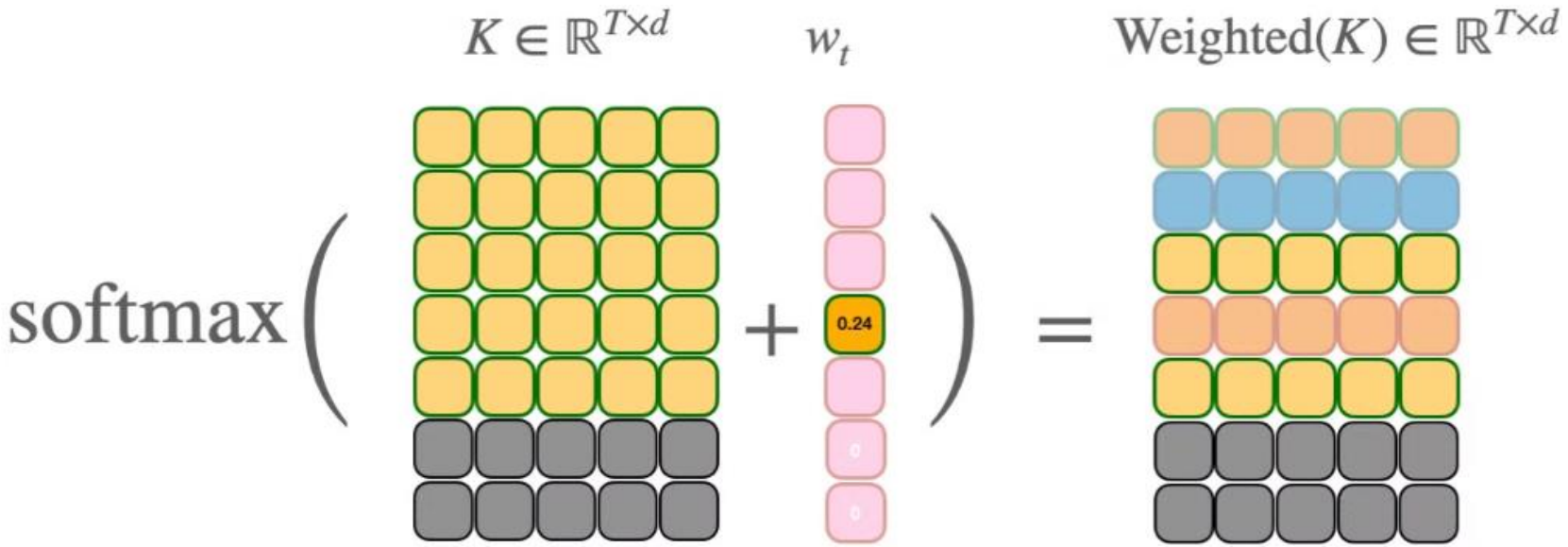
- 输入 X 经过三个 linear Transformer 得到 Q K V 3个矩阵， 维度为 $x \in \mathbb{R}^{T \times d}$ 。AFT 引入了一个新的可训练参数矩阵 $w \in \mathbb{R}^{T \times T}$，论文将其称之为可学习的一对一位置偏置（learned pair-wise position biases）。



Attention Free Transformer：Step1

- 求偏置的权重 $weight(K^{(t)})$，从 w 取 $t=i$ 的向量，和 K 做点乘后以列方向计算 softmax。该步骤的计算复杂度为 $O(Td)$：

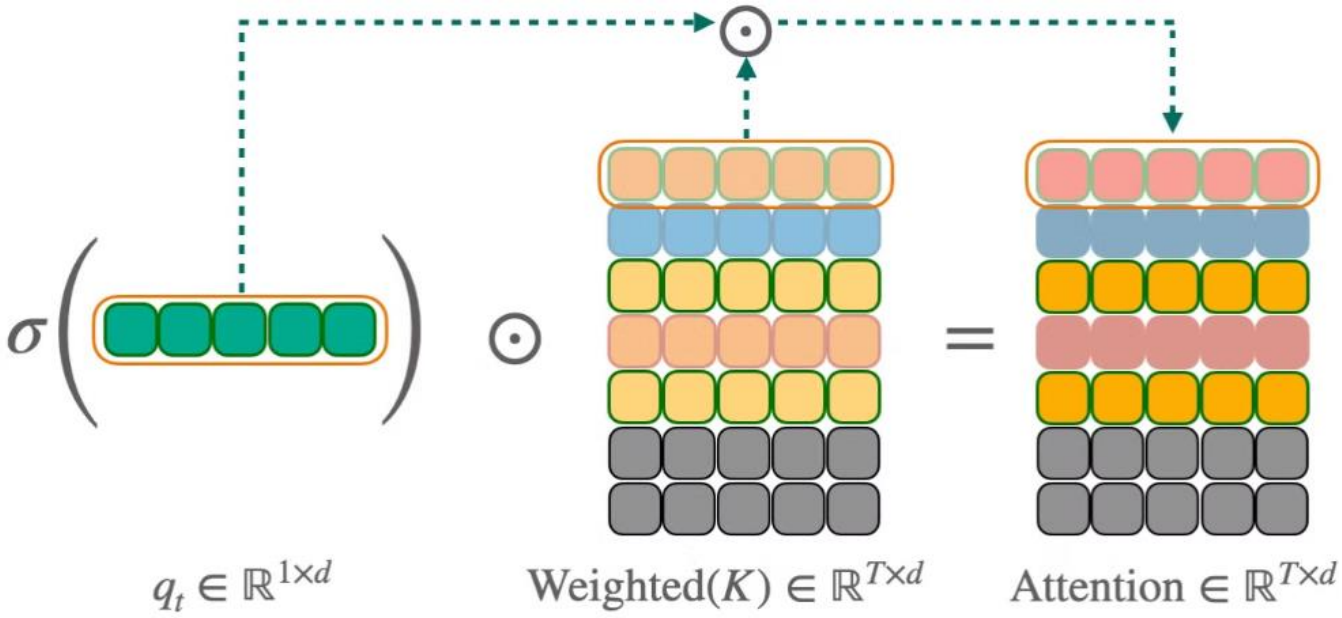
$$\text{Weight}(K^{(t)}) = softmax(K + w_t) = \frac{exp(K + w_t)}{\sum_{i=1}^{T} \exp(k_i + w_{ti})}$$



Attention Free Transformer：Step2

- 求 $Attention^{(t)}$ 矩阵。将 q_t 用 sigmoid 变换后，点乘 $weight\left(K^{(t)}\right)$ 。该步骤的计算复杂度为 $O(Td)$:

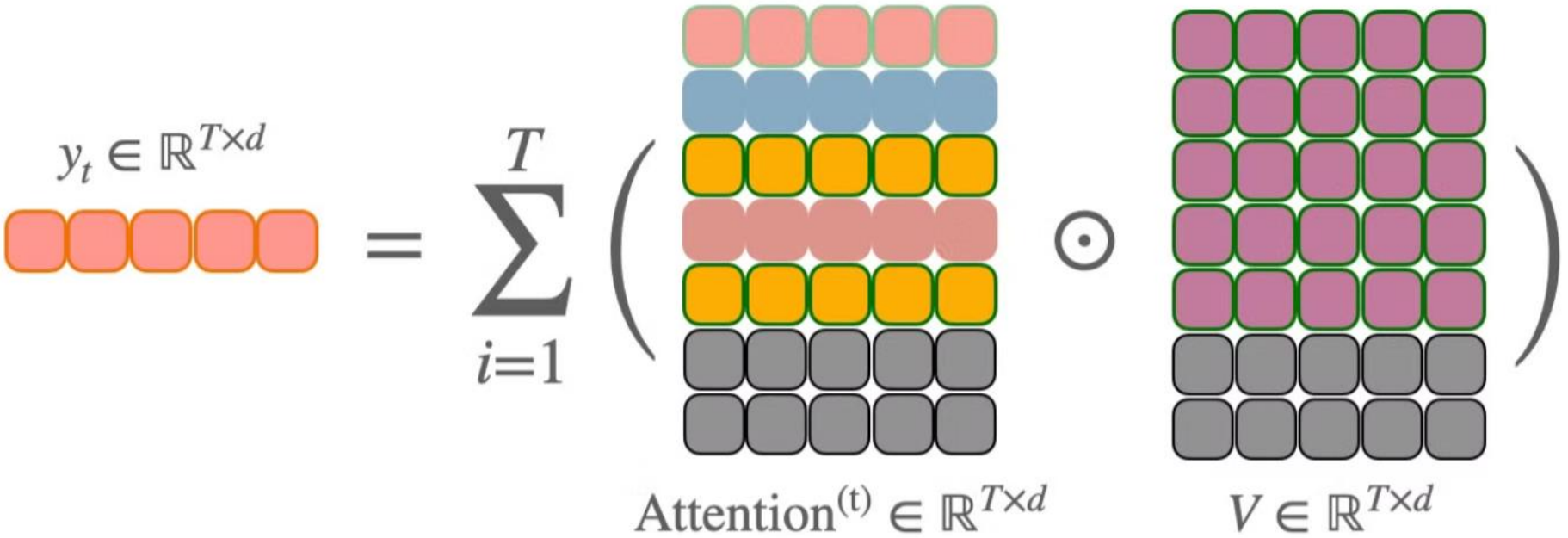
$$Attention^t = \sigma(q_t) \odot \text{Weight}(K^t) = \frac{\sigma(q_t) \odot exp(K + w_t)}{\sum_{i=1}^{T} \exp(k_i + w_{ti})}$$



Attention Free Transformer：Step3

- 计算 y_t。该步骤的计算复杂度为 $O(Td)$：

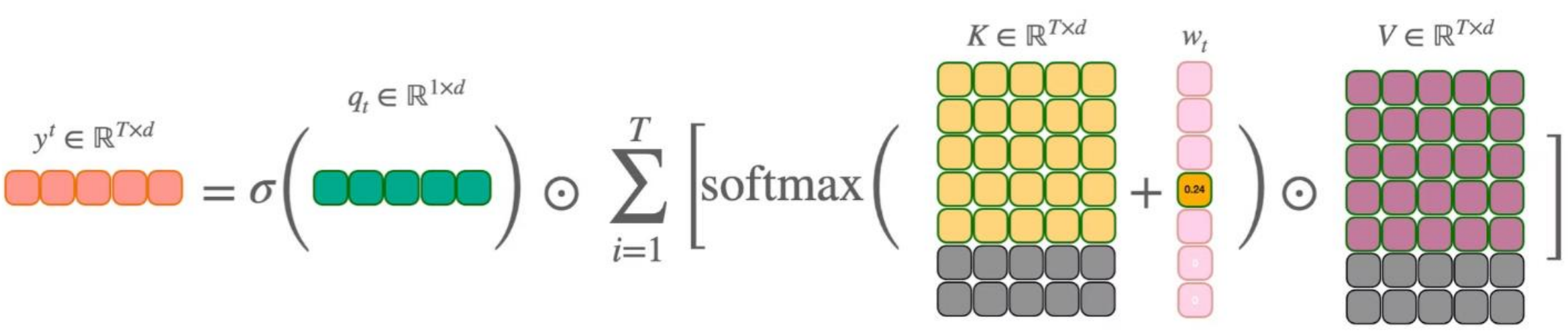
$$y_t = \sum_{i=1}^{T} (Attention_i^T \odot v_i) = \sum_{i=1}^{T} \frac{\sigma(q_t) \odot exp(k_i + w_t)}{\sum_{i=1}^{T} \exp(k_i + w_{ti})} \odot v_i$$



Attention Free Transformer：Step4

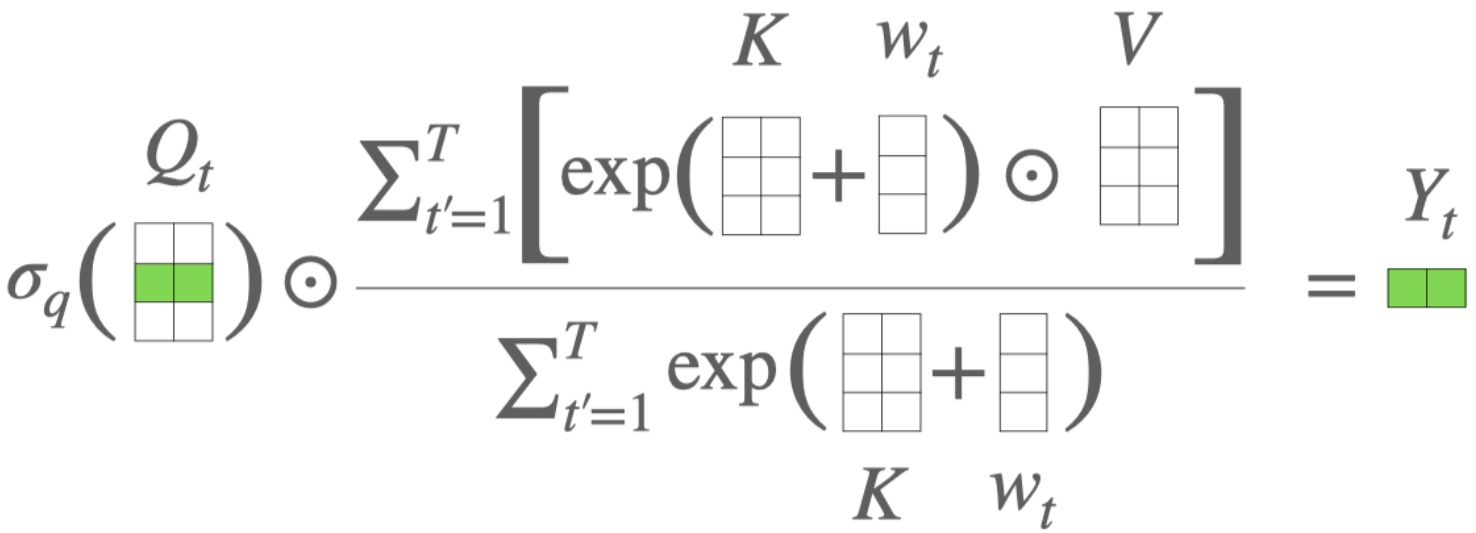
- 对 Step3 使用交换律得到：

$$y_t = \sum_{i=1}^{T} (Attention_i^T \odot v_i) = \sigma(q_t) \odot \frac{\sum_{i=1}^{T} exp(k_i + w_t) \odot v_i}{\sum_{i=1}^{T} \exp(k_i + w_{ti})}$$

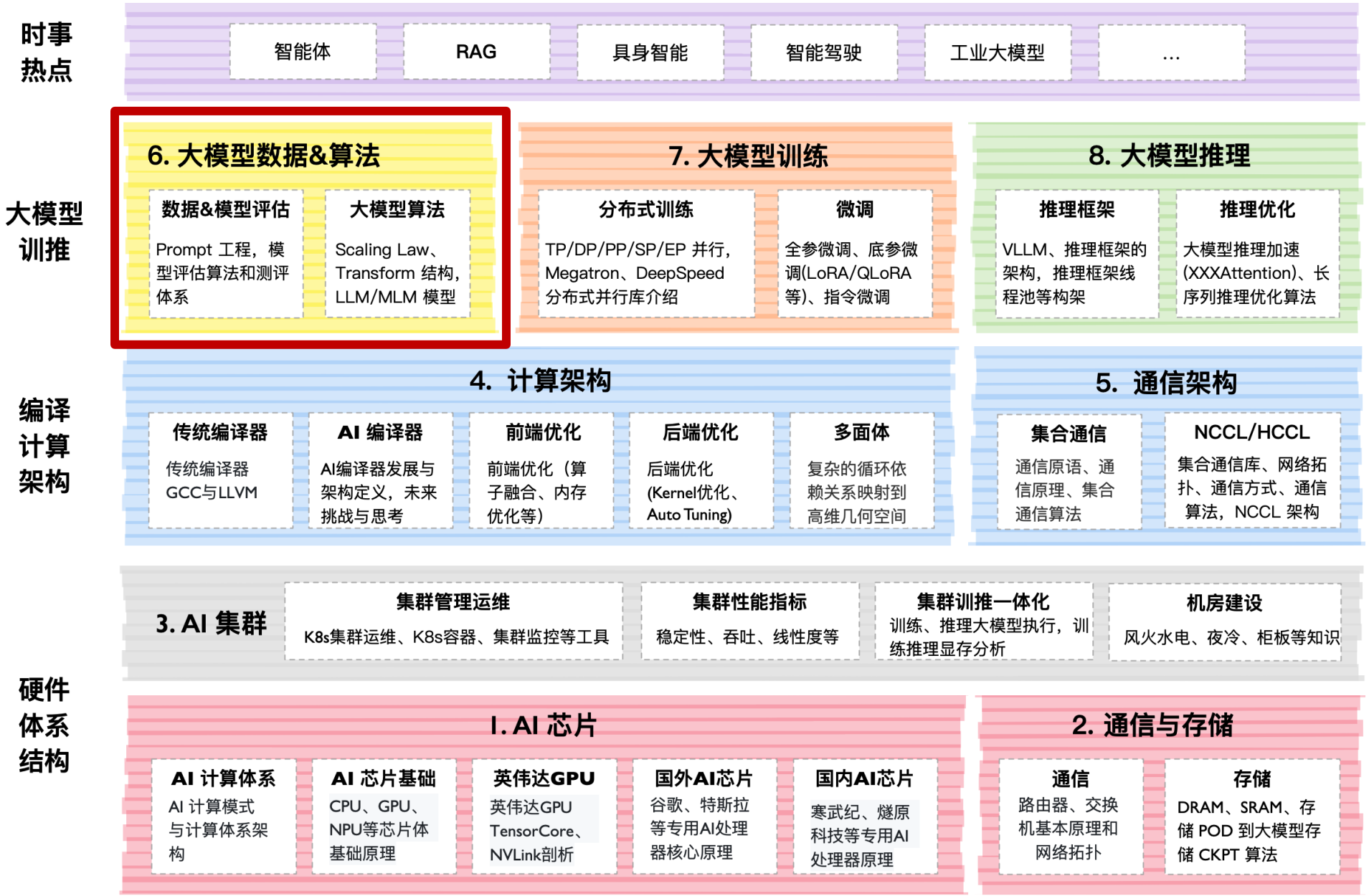


Attention Free Transformer

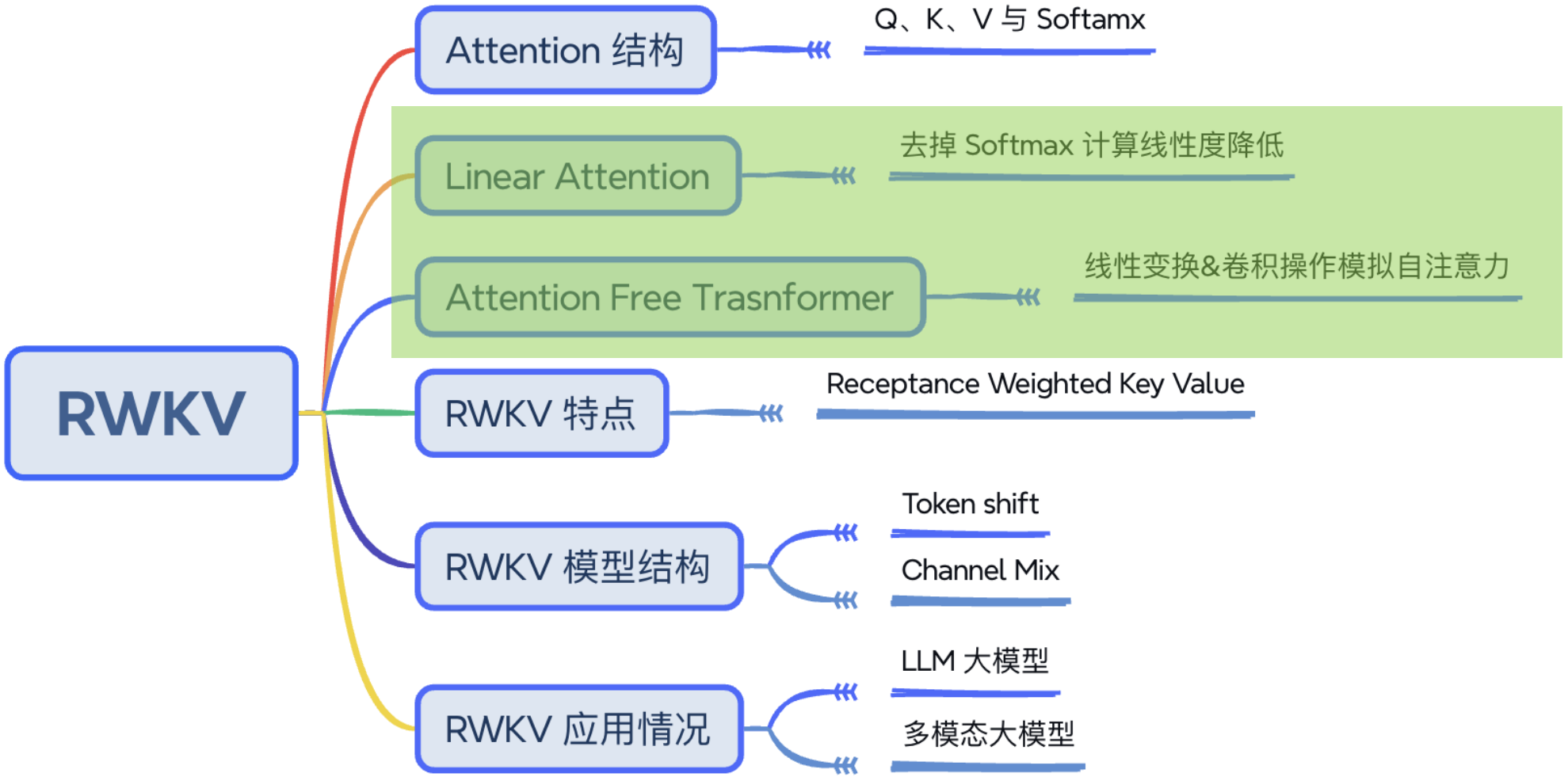
- 从 AFT-full 计算过程拆解后的内容来看 AFT-full 本身并不复杂，它最关键的改进是将 Transformer 的矩阵乘法修改为了按元素相乘：



4 总结与思考


时事热点
智能体
RAG
具身智能
智能驾驶
工业大模型
...
大模型训推
6. 大模型数据&算法
数据&模型评估
Prompt 工程，模型评估算法和测评体系
大模型算法
Scaling Law、Transform 结构，LLM/MLM 模型
7. 大模型训练
分布式训练
TP/DP/PP/SP/EP 并行，Megatron、DeepSpeed 分布式并行库介绍
微调
全参微调、底参微调(LoRA/QLoRA 等)、指令微调
8. 大模型推理
推理框架
VLLM、推理框架的架构，推理框架线程池等构架
推理优化
大模型推理加速(XXXAttention)、长序列推理优化算法
编译计算架构
4. 计算架构
传统编译器
传统编译器 GCC与LLVM
AI 编译器
AI编译器发展与架构定义，未来挑战与思考
前端优化
前端优化（算子融合、内存优化等）
后端优化
后端优化(Kernel优化、Auto Tuning)
多面体
复杂的循环依赖关系映射到高维几何空间
5. 通信架构
集合通信
通信原语、通信原理、集合通信算法
NCCL/HCCL
集合通信库、网络拓扑、通信方式、通信算法，NCCL 架构
硬件体系结构
3. AI 集群
集群管理运维
K8s集群运维、K8s容器、集群监控等工具
集群性能指标
稳定性、吞吐、线性度等
集群训推一体化
训练、推理大模型执行，训练推理显存分析
机房建设
风火水电、夜冷、柜板等知识
1. AI 芯片
AI 计算体系
AI 计算模式与计算体系架构
AI 芯片基础
CPU、GPU、NPU等芯片体基础原理
英伟达GPU
英伟达GPU TensorCore、NVLink剖析
国外AI芯片
谷歌、特斯拉等专用AI处理器核心原理
国内AI芯片
寒武纪、燧原科技等专用AI处理器原理
2. 通信与存储
通信
路由器、交换机基本原理和网络拓扑
存储
DRAM、SRAM、存储 POD 到大模型存储 CKPT 算法

关于本内容


RWKV
Attention 结构
Q、K、V 与 Softamx
Linear Attention
去掉 Softmax 计算线性度降低
Attention Free Trasnformer
线性变换&卷积操作模拟自注意力
RWKV 特点
Receptance Weighted Key Value
RWKV 模型结构
Token shift
Channel Mix
RWKV 应用情况
LLM 大模型
多模态大模型

改进 Softmax 降低计算复杂度从：

$$O(n^2) \text{ 到 } O(n)$$

Question

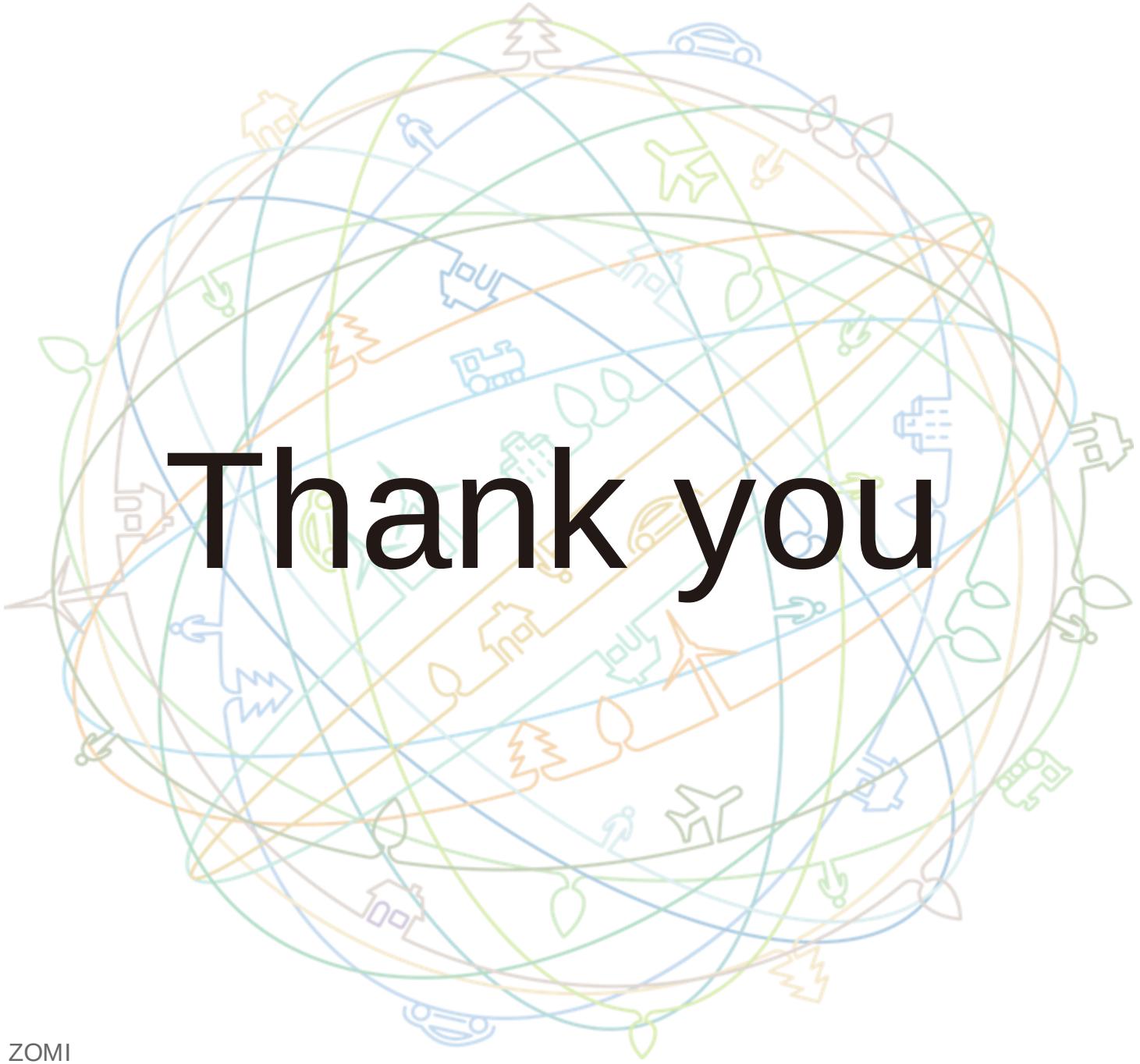
I. 带来什么好处吗?

模型的显示好处

1. 单 Token 推理时间恒定，总推理时间随序列长度线性增长
2. 内存占用恒定，不随序列长度而增加
3. 推理时间和内存占用随名参数量线性增长

对AI 系统的影响

1. 大模型硬件限制和部署成本下降，CPU、嵌入式设备都可以部署
2. 服务器部署大模型成本下降，将可能推动大模型进行架构迁移



Thank you

把AI系统带入每个开发者、每个家庭、每个组织，构建万物互联的智能世界

Bring AI System to every person, home and organization for a fully connected, intelligent world.

ZOMI

Course chenzomi12.github.io

GitHub github.com/chenzomi12/DeepLearningSystem

Reference 参考&引用

1. https://arxiv.org/abs/2006.16236
2. https://www.rwkv.com/
3. https://arxiv.org/abs/2105.14103
4. https://github.com/BlinkDL/RWKV-LM
5. https://wiki.rwkv.com/