arXiv:2602.12128v1 [cs.AI] 12 Feb 2026

HLA：Hadamard线性注意力

Hanno Ackermann HongCai Mohsen Ghafoorian Amirhossein Habibian

Qualcomm AI Research [1]

摘要

注意力机制是 Transformer 成功的重要原因。它依赖于计算词元之间的两两关系。为降低标准二次注意力的高昂计算成本，人们提出了线性注意力，作为一种高效的近似方法。它使用核函数，在计算两两相似度之前，分别对输入进行变换。这使得计算过程更加高效；然而，这相当于用低次有理函数近似 Softmax。

我们提出了 Hadamard 线性注意力（HLA）。与以往的线性注意力研究不同，HLA 中的非线性并非分别作用于查询和键，而是在计算出两两相似度之后才应用，这与标准 softmax 注意力类似。我们将证明，所提出的非线性相当于使用更高次有理函数来近似 Softmax。我们还推导出一种与标准线性注意力类似的高效计算方案，用于实现所提出的方法。与其他方法相比，应用所提出的算法无需耗时的张量重塑。我们将该方法应用于一个用于视频生成的大型扩散 Transformer模型，展示了其有效性；这一应用涉及数量极其庞大的词元。

1. 引言

Transformer架构在广泛的领域中都取得了显著成功，例如问答（Sima 等人，2024）、推理（Wei 等人，2022），甚至是三维重建等任务（Wang 等人，2024）。这一成功的重要原因是注意力机制能够促进输入序列或集合中各元素之间的信息交换。标准注意力机制的一个重要局限是，其复杂度相对于输入序列长度呈二次增长。尽管近期的进展（Dao 等人，2022；Dao，2023；Shah 等人，2024）已降低二次内存复杂度，但计算复杂度仍呈二次增长。线性注意力机制被提出作为可扩展的替代方案。然而，尽管基于线性注意力的Transformer（Katharopoulos 等人，2020；Schlag 等人，2021）效率更高，其性能仍逊于采用标准注意力的Transformer。

Softmax 注意力产生的注意力分数通常只有少数较大的值，而大多数输出都很小。研究已证实，这些矩阵的秩非常高（Zhang 等人，2025）。线性注意力的大多数研究都专注于通过可分离核近似 Softmax。这会导致低秩约束，使线性注意力无法精确近似 Softmax 注意力的输出（Fan 等人，2025a）。正如（Letourneau等人，2024）所指出的，这种约束相当于一个低次有理函数。已有研究提出在线性注意力模型中加入额外项，以缓解低秩约束（Fan 等人，2025a），但这会降低线性注意力的效率。另一种提升线性注意力性能的方法是门控 Delta 网络（Schlag 等人，2021年；Yang 等人，2025a）。该算法试图通过衰减函数抑制相关性较低的词元交互，从而优先保留输入序列中信息量更丰富的依赖关系。近期提出了混合注意力机制，通过结合基于 Softmax 的二次注意力与线性注意力来解决这一问题（Zhang 等人，2025）。尽管这是一个很有前景的方法，但它实际上仍具有二次复杂度。

在标准线性注意力模型中，非线性变换，即可分离核，会在计算两两交互之前应用，从而对完整注意力矩阵进行低秩近似。这与基于 softmax 的注意力形成对比，后者会在计算两两交互之后应用非线性。在这项工作中，我们为线性注意力引入了一种新颖的非线性，即 Hadamard 线性注意力（HLA）。与现有方法不同，所提出的非线性也可以在

[1]Qualcomm AI Research 是 Qualcomm Technologies, Inc. 的一项计划。联系邮箱：Hanno Ackermann < hackerman@qualcomm.com>。



图 *1.* 使用 Hadamard线性注意力生成的所提出的视频扩散 Transformer（**HLA-3F-R1-10**，更多详情请参见第 5.1 节）。

之后应用于两两交互的计算结果，从而使其行为更接近标准 softmax 注意力。我们为所提出的注意力机制推导出一种高效的计算方案，同时保留线性注意力的可扩展性优势。

所提出的算法的一项重要优势是可以直接应用于序列，这与其他算法不同（Zhang 等人，2025；Letourneau 等人，2024），后者需要先重塑张量。尽管这种重塑操作不会计入浮点运算次数（FLOPs）等指标，但在内存受限平台上可能会增加延迟[1]。此外，Hadamard线性注意力通过其注意力机制考虑整个序列，而（Letourneau 等人，2024）则只能通过其感受野获取有限范围的信息。

最后，我们提出一种门控注意力的修改：该方法已被提出，用于通过门控函数对注意力的输出重新加权，该门控函数使用输入的 Sigmoid 归一化函数（Qiu 等人，2025）。作者报告称，这提升了性能并缓解了注意力汇。我们未能确认性能提升，在我们的应用中，但提出了该技术的一种简化变体，可带来性能提升。

以下总结了我们的贡献：

- 我们提出了一种新颖的线性注意力算法——Hadamard 线性注意力（HLA），它会在计算完成对相似度后应用非线性变换。
- 与线性注意力相比，HLA 具有更高的多项式次数，因此模型的表达能力更强，也能取得更好的结果。
- 该过程基于 einsum 等标准且适合硬件实现的操作，既不需要 Softmax 这类开销高昂的操作，也不需要动态索引。
- 在视频生成这一具有挑战性的任务中，序列长度极长。与采用标准注意力的最先进视频扩散模型相比，我们的方法性能相当，同时计算量最多可减少 90%。
- 结果表明，这种新颖的注意力机制无需窗口化或 Delta 规则等技术，就能生成动态效果丰富的高质量视频。
- 此外，我们的模型训练仅需使用 8 个 H100。

[1]编程时，遇到错误消息 无法通过非连续张量进行反向传播，我们可能会碰到这个问题。哪些操作需要张量按连续方式排列，取决于特定硬件的能力。

2. 相关工作

人们提出了线性注意力，以解决标准的基于 softmax 的注意力所带来的二次内存需求和计算复杂度（Shen 等人，2024；Katharopoulos 等人，2020）。有研究指出，与使用标准注意力的算法相比，依赖线性注意力的算法通常表现较差。Schlag 等人，2021）指出，线性注意力等价于快速权重编程器（Schmidhuber，1992）。同一批作者还提出，通过 Delta 规则使用衰减因子，抑制在序列中保留无关信息。由于该算法的延迟会随序列长度增加，因此（Qiu 等人，2025）提出了该算法的分块版本。对于大语言模型训练，（Sun 等人，2023）也提出了类似的想法。

（Peng 等人，2021a；Choromanski 等人，2022；Qin 等人，2022；Meng 等人，2025）提出了如何在线性注意力的背景下更好地近似指数函数。线性注意力固有的低秩约束可能是其性能不佳的原因。文献（Fan 等人，2025a）提出了一种解决方案。此外，还有其他方法，例如基于最大编码率降低（Wu 等人，2024）或局部敏感哈希（Han 等人，2023）。

另一种降低标准注意力二次复杂度的方法是状态空间方法（SSMs）。它们通过一组递归作用于输入序列的一阶微分方程来建模时间变化。它们在长程任务中表现良好（Gu 等人，2022），并且在某些任务上的表现可与 Transformer 相当，甚至更好（Gu 与 Dao，2024）。文献（Dao 与 Gu，2024 ）揭示了线性注意力与状态空间模型之间的联系。采用文献（Schlag 等人，2021年）中的 Delta 规则也有益于状态空间模型（Yang 等人，2025a）。受状态空间模型思想的启发，文献（Egorov 等人，2025）在不牺牲性能的前提下改进了标准注意力机制。

线性注意力已被用于改进扩散网络，例如在Xie 等人，2024；Zhu 等人，2024）中用于图像。经 Delta 规则改进的线性注意力也已用于视频生成的扩散算法中（Chen 等人，2025；Team 等人，2025）。Ye 等人，2025的作者通过执行两次标准注意力并将两次结果相减，对其进行了改进。

近期，（Zhang 等人，2025）和（Ghafoorian 等人，2025）将 Softmax 注意力与线性注意力结合，用于视频扩散。这一思路是用 Softmax 对注意力矩阵中的较大值建模，而用线性注意力处理较小值。尽管这些混合方法表现良好，但需要动态索引[2]，且并非无 Softmax

[2]某些低功耗计算单元不支持直接选取张量中的一个或多个单独的值。

3. 注意力

3.1. Softmax 注意力

设有 N个词元 $\boldsymbol{X} = [\boldsymbol{x}_0^\top, \ldots, \boldsymbol{x}_N^\top] \in \mathbb{R}^{N\times d}$，其中标量 d 表示每个向量 $\boldsymbol{x}_i$ 包含的元素数量。这些输入词元通过与可学习矩阵 Q、 K 和 V右乘，映射为查询 W_q、 W_k、 W_v和 QK^T，其乘积经*Softmax*归一化。（Vaswani 等人，2017）所采用的注意力机制使用它们来计算值的凸组合

$$\operatorname{softmax}\left(\frac{1}{\sqrt{d}}QK^\top\right)V. \tag{1}$$

此处说明的是自注意力，它比较来自同一序列的词元 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$。所谓的交叉注意力则使用来自两个不同输入序列的词元。

3.2. 线性注意力

标准注意力机制已在许多工作中取得巨大成功，但其计算和内存复杂度相对于注意力分数的计算呈二次增长。尽管（Dao 等人，2022年；Dao，2023）已提出解决二次内存复杂度问题的方法，但这些算法仍需要 $\mathcal{O}$（N^2）次计算操作。

线性注意力已被提出（Schlag 等人，2021年；Katharopoulos 等人，2020；Schlag 等人，2021年）作为同时解决这两个问题的方法。其思路是使用可分离核来近似 exp 函数

$$\exp\langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle \approx K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)\rangle. \tag{2}$$

由于核具有可分离性，线性注意力改变了矩阵乘法的顺序

$$\phi(Q)\left(\phi(K)^\top \cdot V\right) = \phi(Q)\cdot C \tag{3}$$

从而避免显式计算注意力分数，使复杂度降低至 $\mathcal{O}$（N），因为 ϕ（K）$^\top \cdot V$构成了 $d\times d$ 上下文矩阵。矩阵 C 在公式 3 中通常称为上下文。（Fan 等人，2025b）指出，由于根据公式 d将查询映射到 3 维空间，而不是根据公式N将其映射到更具表达力的 1 维空间，因此线性注意力机制的性能可能会下降：

$$\operatorname{rank}\left(\phi(Q)\cdot C\right) \le \min\left(\operatorname{rank}\left(\phi(Q)\right), \operatorname{rank}\left(C\right)\right) \tag{4}$$

随机傅里叶特征由（Peng 等人，2021b）提出，用于对方程2建模。正交随机傅里叶特征，以及对Softmax 的三角函数和双曲函数近似，被（Choromanski

等人，2020年）采用。一些作者遵循（Peng 等人，2021b）提出的思路，将输入特征投影到比输入维度数量更多的维度上。这样可以缓解甚至消除其他算法可能面临的低秩约束。

4. Hadamard线性注意力

4.1. 定义

可分离核函数是线性注意力的基础，能够实现计算和内存高效的注意力过程。由于非线性函数是在计算两两交互之前应用的，线性注意力模型对两两交互进行了低秩近似（Zhang 等人，2025）。这不同于基于 *Softmax* 的注意力，后者将非线性函数应用于两两交互。

本文的贡献在于提出了一种新型线性注意力机制，以及一种在线性注意力中全新的非线性函数。与现有研究不同，这种新型非线性函数可以在计算两两交互之后应用。因此，所提出的注意力机制更类似于标准的 *Softmax* 注意力。尽管看起来这种模型无法采用高效的计算过程来计算注意力，但我们将证明，所提出的注意力机制能够实现这一点，与线性注意力类似。

设 $\phi_q\colon\ d \to d_\phi$ 和 $\phi_{k_f}\colon\ d \to d_\phi$ 为非线性变换，其中 $f = 1, \ldots, F$。我们将注意力算子定义如下

$$A = \left(\phi_q(Q) \cdot \phi_{k_1}(K)^\top\right) \odot \left(\phi_q(Q) \cdot \phi_{k_2}(K)^\top\right) \odot \cdots \tag{5}$$

其中，符号 $\odot$ 表示逐元素乘积，也称为 Hadamard 积。方程 5 $\prod_{f=1}^{F} \phi_q\ q_i^\top \cdot$ 使 $\phi_{k_f}\ k_j$ 用逐元素乘积 ( ) ( )，而非将 *Softmax* 作为非线性函数。注意力更新可通过将 A 右乘值矩阵 V来计算。网络 ϕ_q和 ϕ_{k_f} 的定义请参见补充材料。

与标准注意力相比，Hadamard线性注意力的一大优势是，低秩约束的限制性较弱

$$\begin{aligned}\operatorname{rank}&\left(\left(\phi_q(Q) \cdot \phi_{k_1}(K)^\top\right) \odot \cdots \odot \left(\phi_q(Q) \cdot \phi_{k_F}(K)^\top\right)\right) \\ &\leq \prod_{f=1}^{F} \operatorname{rank}\left(\phi_q(Q) \cdot \phi_{k_f}(K)^\top\right). \end{aligned} \tag{6}$$

由于低秩约束带来的瓶颈得到了缓解，与标准线性注意力相比，基于 HLA 的网络在每个注意力层中的信息损失更少，因此性能可能更好。

4.2. 高效注意力

直接通过方程 $A \cdot V$计算5 的复杂度为二次。现在，我们推导一种高效的注意力计算策略，这将揭示一个与标准线性注意力类似的过程。在下文中，不带任何下标的求和符号表示$\sum$对参数的所有元素求和。

引理 4.1。 给定 $F = 2$ 方程 5中的 *Hadamard* 积因子，我们可以将涉及以下向量的乘积表示为d维向量 q， r^1 和 r^2 如下

$$\left(q^\top \cdot r^1\right) \cdot \left(q^\top \cdot r^2\right) = \sum \underbrace{\left(q \cdot q^\top\right)}_{d \times d} \odot \underbrace{\left(r^1 \cdot {r^2}^\top\right)}_{d \times d} \tag{7}$$

其中，求和遍历所有长度为$(q)^2$ 的元素。

对于 $F > 2$，涉及 d维向量q 和 r^f 的关系可推广为

$$\prod_{f=1}^{F}\left(q^\top \cdot r^f\right) = \sum \underbrace{\left(q \otimes \cdots \otimes q\right)}_{d \times \cdots \times d} \odot \underbrace{\left(r^1 \otimes \cdots \otimes r^F\right)}_{d \times \cdots \times d} \tag{8}$$

其中，符号 $\otimes$ 表示张量外积，求和遍历所有元素。

证明。 给定向量 $a, b_1, \ldots, b_n$属于 $\mathbb{R}^d$，令 $A = a \otimes a \otimes \cdots \otimes a$和 $B = b_1 \otimes \cdots \otimes b_n$。令$[\mathcal{T}]_{i_1,\ldots,i_n}$为算子，它选取张量$\mathcal{T}$中位置 $i_1, \ldots, i_n$处的元素。那么，张量 $\mathcal{A} \odot \mathcal{B}$ 中的该元素等于

$$\begin{aligned}[\mathcal{A} \odot \mathcal{B}]_{i_1,\ldots,i_n} &= \mathcal{A}_{i_1,\ldots,i_n} \cdot \mathcal{B}_{i_1,\ldots,i_n} \\ &= \left(a_{i_1} \cdots a_{i_n}\right) \cdot \left({b_1}^{i_1} \cdots {b_n}^{i_n}\right). \end{aligned} \tag{9}$$

对张量的所有元素求和，这对应于等式右侧的求和8，我们可以写成

$$\begin{aligned}&\sum_{i_1,\ldots,i_n} a_{i_1} \cdots a_{i_n} \cdot b_1^{i_1} \cdots b_n^{i_n} \\ &\quad= \left(\sum_{i_1} a_{i_1} b_1^{i_1}\right) \cdots \left(\sum_{i_n} a_{i_n} b_n^{i_n}\right) \\ &\quad= \left(a^\top b_1\right) \cdots \left(a^\top b_n\right) \end{aligned} \tag{10}$$

这对应于等式 8的左侧 □

令 $\mathcal{T}_q$为一个 $N \times d_\phi \times \cdots d_\phi$维张量，其中包含向量$\phi_q(q_i)$与自身的$N$ F重张量外积，即第 i个切片 $i = 1,$ ....., N等价于 $[\mathcal{T}_q]_i = \phi_q(q_i) \otimes \cdots \otimes \phi_q(q_i)$。类似地，定义 $\mathcal{T}_k$为一个 $N \times d_\phi \times \cdots d_\phi$维张量，其中包含向量 $\phi_{k_f}(k_j)$的N F重张量外积。

定理4.2。 由方程 5 定义的注意力可等价地表示为以下两者的乘积： $\mathcal{T}_q$ 和一个$d_\phi \times \cdots \times d_\phi \times d$ 维上下文张量 $\mathcal{C}$。

证明。 将值矩阵 v_j 的第 j 列与注意力分数矩阵 A_i 的第 i 行的乘积表示为

$$\begin{aligned} A_i \cdot v_j &\overset{\text{L4.1}}{=} \begin{bmatrix} \sum \mathcal{T}_{q_i} \odot \mathcal{T}_{k1} & \cdots & \sum \mathcal{T}_{q_i} \odot \mathcal{T}_{kN} \end{bmatrix} \cdot v_j \\ &= \sum \mathcal{T}_{q_i} \odot \left(\sum_{i=1}^{N} v_{ij} \underbrace{\left(\phi_{k_1}(k_i) \otimes \cdots \otimes \phi_{k_F}(k_i)\right)}_{d_\phi \times \cdots \times d_\phi} \right) \\ &= \sum \mathcal{T}_{q_i} \odot \left(\mathcal{T}_k \otimes_1 v_j\right). \end{aligned} \tag{11}$$

回顾一下， $\otimes$ 表示 $f = 1, \ldots, F$ 个向量 $\phi_{k_f}(k_i)$ 的外张量积。符号 $\otimes_1$ 表示沿第一维进行收缩，即类似于标准矩阵乘法的乘法与求和。换言之，将张量 $\mathcal{T}_k$ 中大小为 $d_\phi \times \cdots d_\phi$ 的 $i = 1, \ldots, N$ 个切片分别乘以标量 v_{ij}。然后将所得结果与 $\mathcal{T}_{q_i}$ 逐元素相乘并求和。

对于完整矩阵 V，我们通过以下方式计算 $d_\phi \times \cdots \times d_\phi \times d$ 维张量

$$\mathcal{C} = \mathcal{T}_k \otimes_1 V. \tag{12}$$

因此，乘积 $A \cdot V$ 可表示为

$$A \cdot V = \mathcal{T}_q \otimes \mathcal{C}. \tag{13}$$

收缩针对张量 $\mathcal{T}_q$ 除第一维（序列维）之外的所有维度，以及张量 $\mathcal{C}$ 除最后一维之外的所有维度；最后一维对应于矩阵 V 的列数。 □

请注意，尽管方程 13看起来与普通线性注意力的公式相似，但它通过 $\mathcal{T}_k$在 $\mathcal{C}$ 中包含了键的高阶项。换句话说，线性注意力计算的是关于键呈线性的注意力分数，而 HLA 计算的是关于键呈 F 次的注意力分数。

最后，我们需要确保矩阵 A 的每一行之和为 1。如果要求 $\phi_{q,\ k_1,\ k_2,\ k_3}(\cdot) \geq 0$，我们可以采用与标准线性注意力类似的方法。将方程 5中的矩阵每一行归一化为和为 1 的因子 η_i 由张量 $\mathcal{T}_q \odot \mathcal{T}_k$在除第一维之外的所有维度上的收缩给出

$$\eta = \sum_{l,m,\ldots} \left[\mathcal{T}_q \odot \mathcal{T}_k\right]_{l,m,\ldots}. \tag{14}$$

完整流程的概要见补充材料中的算法1。有关该算法的 PyTorch 风格代码，请参见补充材料中的图 11 和 12 。

4.3. 多项式次数和复杂度

引理4.3。注意力输出 $A \cdot V$（其中注意力矩阵 A 由公式5定义）是输入变量的有理函数。分子的次数为 $2^F + 1$，而分母的次数为 2^F。

标准线性注意力等价于一个有理函数，其分子的次数为 3，分母的次数为 2 （Le- tourneau 等人，2024）。根据方程 5，该结论成立。

这一显著更高的次数使所提出的算法能够更好地执行注意力机制。实验结果将显示 Hadamard 积中$F = 2$ 因子与 $F = 3$ 因子之间甚至存在差异。

所提出算法的计算复杂度相对于序列长度呈线性增长，而相对于方程 5 $\mathcal{O}\left(d_\phi^F\right)$ 中 Hadamard 积的因子数量则呈多项式增长。

4.4. 因果注意力、衰减因子与顺序更新

由于我们要求 $\phi_{q,k_1,k_2,k_3}(\cdot) \geq 0$ 满足归一化条件，因此，例如，我们可以通过将一个 $n \times n$ 矩阵 M 左乘到键 $\phi_{k_f}(K)$上，轻松地实现因果注意力。对于因果注意力，M 的元素取为 $\{0, 1\}$。如（Schlag 等人，2021年）所提出，掩码可以包含衰减因子。

顺序更新 所提出的算法可以轻松修改，以实现高效的顺序更新。在第一个词元处，张量 $\mathcal{T}^1_{k_1} = \phi_{k_1}(k_1) \otimes \cdots \otimes \phi_{k_F}(k_1)$ 与 $V = v_1^\top$ 的每个 d 元素相乘 d 次

$$\mathcal{C}^1 = \mathcal{T}_{k_1} \boxtimes v_1, \tag{15}$$

其中， $\boxtimes$ $\mathcal{C}^1$ $\in$ 表示克罗内克积，且。 $\mathbb{R}^{d_\phi \times \cdots d_\phi \times d}$ $\mathcal{T}^1_q = \phi_q$ q_1 $\otimes$ 此时的查询张量（）

$\cdots \otimes \phi_q(q_1)$与 $\mathcal{C}^1$中形状相同的每个d 切片逐元素相乘。将每个 d 张量相加，得到注意力的 d维结果

$$o^1 = \sum_{d_\phi^F} \mathcal{T}^1_q \odot \mathcal{C}^1. \tag{16}$$

对于在时间 $i > 1$顺序到达的任何新数据，我们只需通过加上以下内容来更新上下文张量

$$\mathcal{C}^i = \mathcal{C}^{i-1} + \mathcal{T}_{k_i} \boxtimes v_i, \tag{17}$$

并计算 $\mathcal{T}^i_q$ 与 $\mathcal{C}^i$之间的乘积。换句话说，上下文张量充当顺序注意力的状态。

张量 $\mathcal{T}_{k_i}$的大小为 d_ϕ^F，因此计算乘积 $\mathcal{T}_{k_i} \boxtimes v_i$需要 $d_\phi^F \cdot d$ 次浮点乘法。将其

结果加到 $\mathcal{C}^{i-1}$ 上需要相同数量的浮点加法。如果硬件支持融合乘加，操作总数不会增加。将上下文张量与 $\mathcal{T}_q^i$ 在时间 $N=i$ 可用的所有查询相乘，需要 $N \cdot d_\phi^F \cdot d$ 次运算。采用衰减机制（Schlag 等人，2021年；Yang 等人，2025a），可将运算总数减少到仅 $3d_\phi^F d + Ndd_\phi^F$ 次。

4.5. 值调制

尽管 Hadamard 线性注意力大幅缓解了标准线性注意力所受的低秩约束，但我们仍观察到高频信号分量有所退化。

为了解决这一问题，我们尝试了跳跃连接和 Sigmoid 门控调制。然而，这两种方法都未能取得令人满意的结果。

需要注意的是，逐元素乘法门控等价于频域中的（循环）卷积。由于 Sigmoid 函数会将输入映射到区间 $[0,1],$，它不会去除任何频带，但会显著衰减这些频带。因此，使用经 Sigmoid 函数归一化的信号进行门控，相当于与一个分量极小的核进行卷积，这限制了恢复高频信息的能力。

为避免 Sigmoid 导致的衰减，我们移除这一激活函数，并应用

$$T = T + \phi_{v_1}(T) \odot \phi_{v_2}(V) \tag{18}$$

其中，$T = A \cdot V$ 表示应用输出映射之前的注意力结果。由于与 ϕ_{v_2}（V）对应的核具有较大的支撑集，该卷积可以恢复丢失的高频信息。网络 ϕ_{v_1} 和 ϕ_{v_2} 的定义请参见补充材料。

5. 实验

5.1. 实现细节

数据我们使用 OpenSoraPlan（Lin 等人，2024）所用数据的 35 万条子集，对我们的模型变体进行微调。我们还使用了约 10 万个由 Wan2.1 14B 生成的合成视频。

模型我们将所提出的注意力机制集成到用于视频扩散的 Wan2.1 1.3B 模型中（等人，2025）。我们使用两种分辨率：第一种是原始模型使用的两种分辨率中较小的一种（81x480x832）；第二种是更低的分辨率，更适合快速视频生成（81x320x480）。两者对应的序列长度分别为 32760 和 12600 个词元。我们用**R1**和**R2**表示这两种分辨率。

我们使用 Hadamard线性注意力 的两种变体进行评估。第一种在公式中采用三个因子 5，而第二种只使用两个因子。对于第一种，我们使用较小的多层感知机 $\phi_q, \phi_{k_j},\ j=\{1,2,3\}$ 和 $\phi_{v_l},\ l=\{1,2\}$。对于第二种，我们使用稍大一些的多层感知机。

这两种 HLA 变体被纳入 Wan 的四种变体中：**HLA-2F-R1-21** 在公式5中使用两个因子，分辨率为 81x480x832，并在 30 个 Transformer 块中的 21 个块上应用 Hadmard 线性注意力。**HLA-3F-R1-21** 与**HLA-2F-R1-21** 相同，区别在于它在公式5中使用三个因子。**HLA-3F-R1-10**与**HLA-3F-R1-21** 相同，区别在于它仅在 10 个 Transformer 块上应用 HLA。最后，**HLA-3F-R2-15**在 30 个 Transformer 块中的 15 个块上使用三个因子计算 Hadamard 积，分辨率为 81x320x480。有关模型定义的更多信息，请参阅补充材料。

训练细节 我们采用两种训练方案。首先，只训练经过修改的注意力层，同时固定所有其他层。随后进入第二阶段，对整个 Transformer 进行训练。这两个阶段使用 (Lin 等人，2024) 中的子集。 **HLA-2F-R1-21**、**HLA-3F-R1-10**和**HLA-3F-R2-15**均采用此流程进行训练。对于**HLA-3F-R1-21**，我们直接对整个 Transformer 进行训练。在第三阶段，我们继续训练模型，但此时加入合成数据。

所有模型和所有阶段均采用学习率 1×10^{-4}，并使用 8 个 H100 GPU。最后一个阶段的批次大小设为 16，其余阶段设为 32。除第一阶段外，其余阶段均进行 1000 步预热。除最后一个阶段外，训练步数均为 100k。最后一个阶段中，我们对**HLA-2F-R1-21** 训练 50k 步，对**HLA-3F-R1-10** 训练 90k 步，对**HLA-3F-R2-15**训练 110k 步，对**HLA-3F-R1-21** 训练 140k 步。

评估 *VBench*（Huang 等人，2024）分数用于评估视觉质量，并报告各模型变体的浮点运算次数（FLOPs），以衡量计算复杂度。我们使用 Wan2.1 的两个检查点进行评估（等人，2025），其中一个是原作者提供的 480p 版本，另一个是我们训练的 320p 版本。我们不与近期采用窗口化、Delta 规则，或将标准线性注意力与 Softmax 注意力相结合的混合方法进行比较。这些技术可以集成到 HLA 中；或者，HLA 可以取代混合方法中的标准线性注意力。

5.2. 主要结果

我们在表中比较了 **HLA-3F-R1-10** 与若干参数数量相近的最先进模型的性能 1。为进行比较，我们使用了 *VBench*工具（Huang 等人，2024）。数值越高表示结果越好。可以看出，采用所提出注意力机制的模型表现略逊一筹。与其基础模型 Wan2.1-1.3B 相比，它的计算

	总计↑	质量↑	语义↑
参数量不超过 20 亿的模型			
(Peng 等人，2025)	79.76	81.35	73.39
(Wu 等人，2025)	81.14	83.47	71.84
(HaCohen 等人，2024)	80.00	82.30	70.79
(Yang 等人，2025b)	81.55	82.48	77.81
(Isobe 等人，2025)	81.35	83.73	71.84
(Huang 等人，2025)	81.91	83.36	76.10
(Jin 等人，2025)	81.72	84.74	69.62
(Wan 等人，2025)	83.31	85.23	75.65
Wan2.1-1.3B-HLA ∼ 23% 更少的总计算量			
HLA-3F-R1-10	81.42	83.22	74.20

表*1*。 与其他视频扩散模型在 VBench 上的比较。我们报告了各论文中公布的数据。虽然所提出的方法分数略低，但其所需的万亿次浮点运算不到 Wan2.1-1.3B 的 80%。

复杂度更低（参见表2）。

5.3. 消融实验

表 2 展示了不同模型变体的比较。这些变体的区别在于：在公式 5 中的 Hadamard 积使用两个还是三个因子（分别标记为 **2F** 或 **3F**）、训练时采用的视频分辨率（**R1** 或 **R2**），以及所使用的哈达玛线性注意力层数量（**10**、**15** 或 **21**）。可以看出，随着 HLA 层数量增加，VBench 分数略有下降；而计算复杂度则有所降低。与基线相比，**HLA-3F-R1-21** 生成单个视频所需的时间明显更短。 **HLA-3F-R1-21**所需的浮点运算次数（FLOPs）几乎只有基线的一半。我们还发现，尽管二因子版本通过使用更大的多层感知机实现了相近的 FLOPs，三因子 HLA 的性能仍优于二因子版本。计算时间的降幅不及浮点运算次数降幅那么大的原因，在于计算单元的数量及其支持大规模并行化的能力。可以预期，能力较弱的加速器（例如移动应用中广泛使用的加速器）将从所提出的算法中获益更多。这表明，采用更高阶的 HLA 是提升模型容量的更有效方式。

5.4. 复杂度

表 3比较了四种不同 HLA 变体与标准二次注意力的计算复杂度。与第 5.1 节一样，我们为具有 2 个因子的 HLA 使用了稍大的多层感知机。根据分辨率和算法变体的不同，HLA 所需的浮点运算次数（FLOPs）比二次注意力少 20% 到 90%。图 7 展示了 2 因子和 3 因子 HLA 与标准注意力在不同序列长度下所需浮点运算次数的对比。

方法	词元	分数	万亿次浮点运算	Time
Wan2.1（四倍）	32760	83.3	283.03	1:36
Wan2.1（四卡）	12600	81.0	62.13	0:45
HLA-3F-R1-10	32760	81.42	218.59	1:26
HLA-3F-R2-15	12600	79.78	48.37	0:26
HLA-2F-R1-21	32760	79.02	147.16	1:13
HLA-3F-R1-21	32760	80.54	147.71	1:16

表*2*。 多个模型变体生成单个视频的 VBench 分数和耗时。TFLOPs 指模型单次前向传播的浮点运算次数。所有测量均在一块H100 GPU上完成。

注意力类型	分辨率	词元	万亿次浮点运算
Wan2.1（四倍）	320x480	12600	1.21
Wan2.1（四卡）	480x832	32760	7.21
HLA 2 个因子	320x480	12600	0.97
HLA 2 个因子	480x832	32760	2.52
HLA 3 个因子	320x480	12600	0.30
HLA 3 个因子	480x832	32760	0.77

表 *3.* 方程中 Hadamard 积含有 2 个或 3 个因子的模型的计算复杂度 5 适用于 12600 或 32760 个词元。

5.5. 定性示例

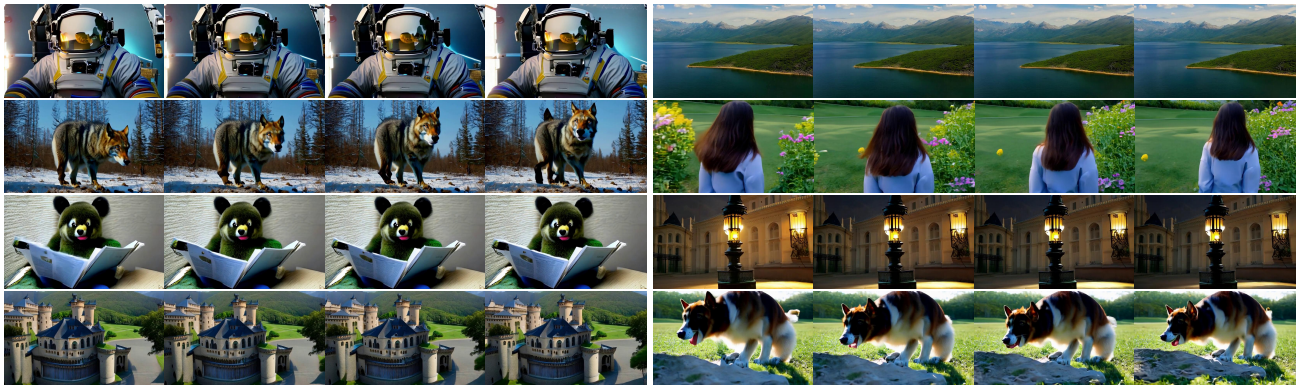
图 1 和2展示了使用所提出的注意力机制生成的视频定性示例。这些结果表明，HLA 更强的非线性确实足以弥合标准线性注意力与二次复杂度 Softmax 注意力之间的性能差距。更多示例和视频请参阅补充材料。

6. 结论

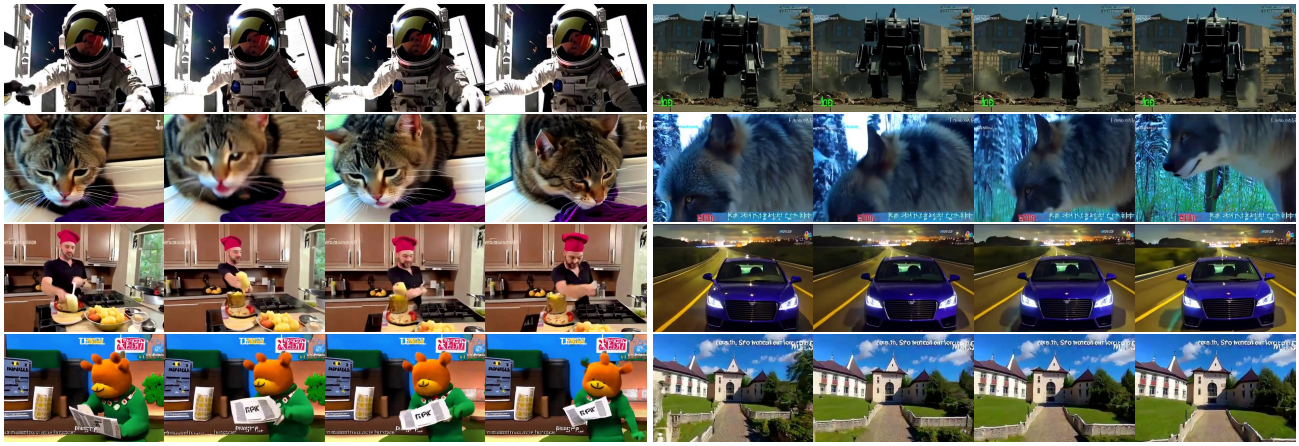
我们提出了 Hadamard 线性注意力（HLA），这是一种计算注意力的新算法。与传统线性注意力算法不同，HLA 在计算两两相似度后应用非线性变换。我们推导出一种高效的公式，将复杂度降至 $\mathcal{O}(N)$。HLA 依赖标准张量运算，无需进行任何耗时的张量重塑，即可直接处理输入序列。我们在具有挑战性的视频扩散任务上验证了其良好效果，该任务涉及极长的序列。对于常见的序列长度，我们提出的 HLA 所需的浮点运算次数仅约为基线所需的 $1/10$。HLA 可以替代混合算法（如Zhang 等人，2025）和（Ghafoorian 等人，2025）中的标准线性注意力，从而提升性能。同样，也可以通过窗口化或 delta 规则扩展 HLA，以改进该算法。



*(a)*结果 **HLA-3F-R1-10**



*(b)*结果 **HLA-3F-R1-21**



*(c)*结果 **HLA-3F-R2-15**

图*2.* HLA-3F-R1-10生成的视频示例 （上方视频）和**HLA-3F-R1-21** （下方视频）。可以看出，采用所提出的注意力机制的方法确实能够生成高质量的视频。更多示例请参见补充材料。

参考文献

Chen, J., Zhao, Y., Yu, J., Chu, R., Chen, J., Yang, S., Wang, X., Pan, Y., Zhou, D., Ling, H., Liu, H., Yi, H., Zhang, H., Li, M., Chen, Y., Cai, H., Fidler, S., Luo, P., Han, S., and Xie, E. Sana-video：采用块线性扩散Transformer实现高效视频生成，2025。链接：https://arxiv.org/abs/2509.24695。

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mo- hiuddin, A., Kaiser, L., Belanger, D., Colwell, L., and Weller, A. 通过Performers重新思考注意力。*arXiv*预印本 *arXiv:2009.14794*，2020。链接：https://arxiv.org/abs/2009.14794。

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., and Weller, A. 通过Performers重新思考注意力，2022。链接 https://arxiv.org/abs/2009.14794。Dao, T. Flashattention-2：以更高并行性和更优工作划分加速注意力。*arXiv*预印本 *arXiv:2307.08691*，2023。链接 https://arxiv.org/abs/2307.08691。Dao, T. 和 Gu, A. Transformer即状态空间模型：通过结构化状态空间对偶实现通用模型与高效算法，2024。链接：https://arxiv.org/abs/2405.21060。

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., 和 R´e, C. Flashattention：借助I/O感知实现快速且节省内存的精确注意力。神经信息处理系统进展，35:16344–16359，2022。Egorov, E., Ackermann, H., Nagel, M., 和 Cai, H. Myosotis：用于注意力类层的结构化计算，2025。链接：https://arxiv.org/abs/2509.20503。等，T. W. 开放且先进的大规模视频生成模型，2025。链接：https://arxiv.org/abs/2503.20314。

Fan, Q., Huang, H., 和 He, R. 打破线性注意力的低秩困境，2025a。链接：https://arxiv.org/abs/2411.07635。Fan, Q., Huang, H., 和 He, R. 打破线性注意力的低秩困境，2025b。Ghafoorian, M., Korzhenkov, D., 和 Habibian, A. 注意力手术：将视频扩散 Transformer 线性化的高效方案，2025。链接：https://arxiv.org/abs/2509.24899。

Gu, A. 和 Dao, T. Mamba：采用选择性状态空间的线性时间序列建模，2024。URL https://arxiv.org/abs/2312.00752。Gu, A.、Goel, K. 和 R´e, C. 使用结构化状态空间高效建模长序列，2022。HaCohen, Y.、Chiprut, N.、Brazowski, B.、Shalem, D.、Moshe, D.、Richardson, E.、Levin, E.、Shiran, G.、Zabari, N.、Gordon, O.、Panet, P.、Weissbuch, S.、Kulikov, V.、Bitterman, Y.、Melumian, Z. 和 Bibi, O. Ltx-video：实时视频潜扩散，2024。URL https://arxiv.org/abs/2501.00103。Han, I.、Jayaram, R.、Karbasi, A.、Mirrokni, V.、Woodruff, D. P. 和 Zandieh, A. Hyperattention：近线性时间内的长上下文注意力，2023。URLhttps://arxiv.org/abs/2310.05869。

Huang, J.、Zhang, G.、Jie, Z.、Jiao, S.、Qian, Y.、Chen, L.、Wei, Y. 和 Ma, L. M4v：用于文生视频生成的多模态 Mamba，2025。URL https://arxiv.org/abs/2506.10915。Huang, Z.、He, Y.、Yu, J.、Zhang, F.、Si, C.、Jiang, Y.、Zhang, Y.、Wu, T.、Jin, Q.、Chanpaisit, N.、Wang, Y.、Chen, X.、Wang, L.、Lin, D.、Qiao, Y. 和 Liu, Z. Vbench：视频生成模型综合基准套件。发表于计算机视觉与模式识别（*CVPR*），2024。Isobe, T.、Cui, H.、Zhou, D.、Ge, M.、Li, D. 和 Barsoum, E. Amd-hummingbird：迈向高效文生视频模型，2025。URL https://arxiv.org/abs/2503.18559。Jin, Y.、Sun, Z.、Li, N.、Xu, K.、Xu, K.、Jiang, H.、Zhuang, N.、Huang, Q.、Song, Y.、Mu, Y. 和 Lin, Z. 用于高效视频生成建模的金字塔流匹配，2025。URL https://arxiv.org/abs/2410.05954。

Katharopoulos, A.、Vyas, A.、Pappas, N. 和 Fleuret, F. Transformer 就是循环神经网络：采用线性注意力的快速自回归 Transformer。发表于第*37*届国际机器学习会议（*ICML*）论文集，第5156–5165页，2020。Letourneau, P.-D.、Singh, M. K.、Cheng, H.-P.、Han, S.、Shi, Y.、Jones, D.、Langston, M. H.、Cai, H. 和 Porikli, F. Padre：一种统一的多项式注意力即插即用替代方案，用于高效视觉 Transformer，2024。URLhttps://arxiv.org/abs/2407.11306。Lin, B.、Ge, Y.、Cheng, X.、Li, Z.、Zhu, B.、Wang, S.、He, X.、Ye, Y.、Yuan, S.、Chen, L.、Jia, T.、Zhang, J.、Tang, Z.、Pang, Y.、She, B.、Yan, C.、Hu, Z.、Dong, X.、Chen, L.、Pan, Z.、Zhou, X.、Dong, S.、Tian, Y. 和 Yuan, L. Open-sora 计划：开源大型视频生成模型，2024。URLhttps://arxiv.org/abs/2412.00131。

Meng, W.、Luo, Y.、Li, X.、Jiang, D. 和 Zhang, Z. Polaformer：面向视觉变换器的极性感知线性注意力。发表于国际学习表征会议（*ICLR*），2025。URLhttps://openreview.net/forum?id=kN6MFmKUSK。

Peng, H.、Pappas, N.、Yogatama, D.、Schwartz, R.、Smith, N. A. 和 Kong, L. 随机特征注意力。载于国际学习表征会议（*ICLR*），2021年a。

Peng, H.、Pappas, N.、Yogatama, D.、Schwartz, R.、Smith, N. A. 和 Kong, L. 随机特征注意力。 *CoRR*, abs/2103.02143，2021年b。网址：https://arxiv.org/abs/2103.02143。Peng, X.、Zheng, Z.、Shen, C.、Young, T.、Guo, X.、Wang, B.、Xu, H.、Liu, H.、Jiang, M.、Li, W.、Wang, Y.、Ye, A.、Ren, G.、Ma, Q.、Liang, W.、Lian, X.、Wu, X.、Zhong, Y.、Li, Z.、Gong, C.、Lei, G.、Cheng, L.、Zhang, L.、Li, M.、Zhang, R.、Hu, S.、Huang, S.、Wang, X.、Zhao, Y.、Wang, Y.、Wei, Z. 和 You, Y. Open-Sora 2.0：以20万美元训练商用级视频生成模型，2025。网址：https://arxiv.org/abs/2503.09642。Qin, Z.、Sun, W.、Deng, H.、Li, D.、Wei, Y.、Lv, B.、Yan, J.、Kong, L. 和 Zhong, Y. cosformer：重新思考注意力机制中的 Softmax。载于国际学习表征会议（*ICLR*），2022。Qiu, Z.、Wang, Z.、Zheng, B.、Huang, Z.、Wen, K.、Yang, S.、Men, R.、Yu, L.、Huang, F.、Huang, S.、Liu, D.、Zhou, J. 和 Lin, J. 面向大型语言模型的门控注意力：非线性、稀疏性与无注意力汇，2025。网址：https://arxiv.org/abs/2505.06708。Schlag, I.、Irie, K. 和 Schmidhuber, J. 线性变换器其实是快速权重编程器。载于第*37*届国际机器学习会议论文集，《机器学习研究论文集》。PMLR，2021。Schmidhuber, J. 学习控制快速权重记忆：动态循环网络的替代方案。《神经计算》，4（1）：131–139，1992。doi：10.1162/neco. 1992.4.1.131。Shah, J.、Bikshandi, G.、Zhang, Y.、Thakkar, V.、Ramani, P. 和 Dao, T. FlashAttention-3：利用异步和低精度实现快速且准确的注意力，2024。网址：https://arxiv.org/abs/2407.08608。Shen, Z.、Zhang, M.、Zhao, H.、Yi, S. 和 Li, H. 高效注意力：具有线性复杂度的注意力。载于冬季计算机视觉应用会议（*WACV*），2024。

Sima, C., Renz, K., Chitta, K., Chen, L., Zhang, H., Xie, C., Beißwenger, J., Luo, P., Geiger, A., 和 Li, H. Drivelm：通过图视觉问答进行驾驶。在欧洲计算机视觉会议（*ECCV*），2024。Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., 和 Wei, F. 保持网络：大语言模型中 Transformer 的继任者，2023。网址 https://arxiv.org/abs/2307.08621。Team, K., Zhang, Y., Lin, Z., Yao, X., Hu, J., Meng, F., Liu, C., Men, X., Yang, S., Li, Z., Li, W., Lu, E., Liu, W., Chen, Y., Xu, W., Yu, L., Wang, Y., Fan, Y., Zhong, L., Yuan, E., Zhang, D., Zhang, Y., Liu, T. Y., Wang, H., Fang, S., He, W., Liu, S., Li, Y., Su, J., Qiu, J., Pang, B., Yan, J., Jiang, Z., Huang, W., Yin, B., You, J., Wei, C., Wang, Z., Hong, C., Chen, Y., Chen, G., Wang, Y., Zheng, H., Wang, F., Liu, Y., Dong, M., Zhang, Z., Pan, S., Wu, W., Wu, Y., Guan, L., Tao, J., Fu, G., Xu, X., Wang, Y., Lai, G., Wu, Y., Zhou, X., Yang, Z., 和 Du, Y. Kimi Linear：一种富有表现力的高效注意力架构，2025。网址 https://arxiv.org/abs/2510.26692。Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., 和 Polosukhin, I. 注意力就是你所需要的一切。在神经信息处理系统进展，第 5998–6008 页，2017。Wan, T., Wang, A., Ai, B., Wen, B., Mao, C., Xie, C.-W., Chen, D., Yu, F., Zhao, H., Yang, J., Zeng, J., Wang, J., Zhang, J., Zhou, J., Wang, J., Chen, J., Zhu, K., Zhao, K., Yan, K., Huang, L., Feng, M., Zhang, N., Li, P., Wu, P., Chu, R., Feng, R., Zhang, S., Sun, S., Fang, T., Wang, T., Gui, T., Weng, T., Shen, T., Lin, W., Wang, W., Wang, W., Zhou, W., Wang, W., Shen, W., Yu, W., Shi, X., Huang, X., Xu, X., Kou, Y., Lv, Y., Li, Y., Liu, Y., Wang, Y., Zhang, Y., Huang, Y., Li, Y., Wu, Y., Liu, Y., Pan, Y., Zheng, Y., Hong, Y., Shi, Y., Feng, Y., Jiang, Z., Han, Z., Wu, Z.-F., 和 Liu, Z. Wan：开放式先进大规模视频生成模型，2025。网址 https://arxiv.org/abs/2503.20314。Wang, S., Leroy, V., Cabon, Y., Chidlovskii, B., 和 Revaud, J. Dust3r：轻松实现几何三维视觉，2024。网址 https://arxiv.org/abs/2312.14132。Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., 和 Zhou, D. 思维链提示能够激发大语言模型的推理能力。在神经信息处理系统进展（*NeurIPS*），2022。网址 https://openreview.net/forum?id=_VjQlMeSB_J。Wu, Y., Zhang, Z., Li, Y., Xu, Y., Kag, A., Sui, Y., Coskun, H., Ma, K., Lebedev, A., Hu, J., Metaxas, D., Wang, Y., Tulyakov, S., 和 Ren, J. Snapgen-v：生成

在移动设备上五秒内生成一段五秒视频，2025。网址https://arxiv.org/abs/2412.10494。

Wu, Z., Ding, T., Lu, Y., Pai, D., Zhang, J., Wang, W., Yu, Y., Ma, Y., and Haeffele, B. D. Token统计Transformer：通过变分率降低实现线性时间注意力，2024年。网址 https://arxiv.org/abs/2412.17810。

Xie, E., Chen, J., Chen, J., Cai, H., Tang, H., Lin, Y., Zhang, Z., Li, M., Zhu, L., Lu, Y., and Han, S. Sana：采用线性扩散Transformer的高分辨率图像高效合成，2024年。网址https://arxiv.org/abs/2410.10629。

Yang, S., Kautz, J., and Hatamizadeh, A. 门控Delta网络：利用Delta规则改进Mamba2，2025a年。网址 https://arxiv.org/abs/2412.06464。

Yang, Z., Teng, J., Zheng, W., Ding, M., Huang, S., Xu, J., Yang, Y., Hong, W., Zhang, X., Feng, G., Yin, D., Zhang, Y., Wang, W., Cheng, Y., Xu, B., Gu, X., Dong, Y., and Tang, J. Cogvideox：采用专家 Transformer 的文生视频扩散模型，2025b。网址 https://arxiv.org/abs/2408.06072。

Ye, T., Dong, L.,Xia, Y., Sun, Y., Zhu, Y., Huang, G., and Wei, F. 差分 Transformer，2025。网址 https://arxiv.org/abs/2410.05258。

Zhang, J., Wang, H., Jiang, K., Yang, S., Zheng, K., Xi, H., Wang, Z., Zhu, H., Zhao, M., Stoica, I., Gonzalez, J. E., Zhu, J., and Chen, J. SLA：通过可微调的稀疏线性注意力，突破扩散 Transformer 中稀疏性的限制，2025。网址 https://arxiv.org/abs/2509.24006。Zhu, L., Huang, Z., Liao, B., Liew, J. H., Yan, H., Feng, J., and Wang, X. DIG：采用门控线性注意力的可扩展高效扩散模型，2024。网址 https://arxiv.org/abs/2405.18428。

A. 模型定义

A.1. 定义 ϕ

我们为网络 ϕ_q,$\phi_{k_{1,2,3}}$ 和 $\phi_{v_{1,2}}$ （见图3）使用参数较少的模型。这些模型由两个线性层组成，中间使用 GELU 激活函数；对于 ϕ_q 和 $\phi_{k_{1,2,3}}$，还可选择使用 RELU 激活函数。训练过程中我们发现，如果 $\phi_{v_{1,2}}$ 不使用层归一化，模型会出现不稳定。

对于第一个线性层，我们将头维度（$1536/12 = 128$）用作输入和输出。对于 $\phi_{v_{1,\ 2}}$，第二个线性层的输入和输出数量相同。否则，如果是 3 因子 HLA，则使用 6 个输出通道；如果是 2 因子 HLA，则使用 12 个通道。

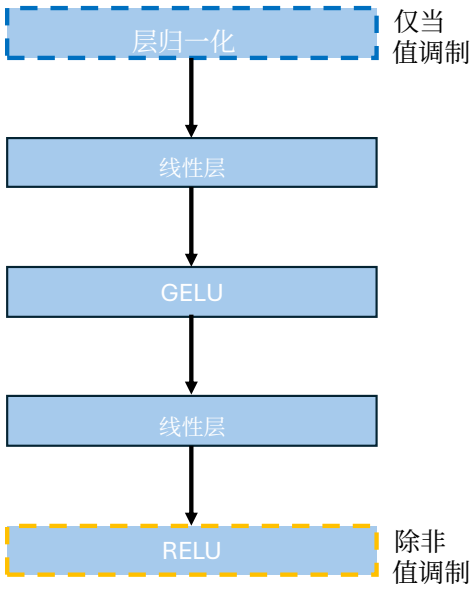


图 3. 网络的定义$\phi_{q,k_{1,2,3},v}$

对于 ϕ_{v_1}，我们最初尝试使用 Sigmoid 激活，但没有效果。

A.2. 模型变体

我们计算了教师模型每个自注意力模块的最大注意力分数，并对多个批次取平均。对于 **HLA-3F-R1-21**，我们选取了分数最低的 21 个模块（1,2,3,4,5,6,7,8, 11,12,13,14,15,16,17,18, 22,23,24,25,26）。对于**HLA-3F-R2-15**，我们在最后 6 个模块中不使用 HLA，这些模块为（1,2,3,4,5,6,7,8, 11,12,13,14,15,16,17）。对于**HLA-3F-R1-10**，我们只在奇数层使用 HLA（1,3,5,7, 11,13,15,17, 23,25）。

A.3. 伪代码

算法 1 给出了整个注意力过程的伪代码。计算双因子或三因子 HLA 的子程序实现所用的类 PyTorch 代码可见图 11 和12。

B. 值调制

为了展示我们提出的值调制模块所带来的积极效果，我们使用 **HLA-3F-R1-21** 和 50 个采样步生成了一段视频。生成序列的第一张图像如图 5所示。

表 4 展示了不同采样步数（列）和不同 Transformer 块（行）下，值调制前的注意力更新。我们重塑张量，以撤销将其拆分为多个注意力头的操作。1536 个通道被投影到 3

算法 1 Hadamard线性注意力

输入： 输入词元 X
输出： 注意力输出

步骤 1：生成查询、键和值q，k，$v \leftarrow$ 到 q(X)，到 k(X)，到 v(X) # 重塑张量以支持多头注意力

步骤 2：应用旋转嵌入q，$k \leftarrow$ 应用 RoPE(q)，应用 RoPE(k)

步骤 3：应用缩放因子
$q \leftarrow q$ 缩放因子

步骤 4：HLA特定变换
$q \leftarrow$HLA 转换为 q(q)
k_1，$k_2 \leftarrow$ HLA 转换为 k1(k)，HLA 转换为 k2(k)
如果使用 HLA 的 3 个因子**则**
$k_3 \leftarrow$HLA 转换为 k3(k)
结束 if

步骤 5：计算 HLA
如果使用 HLA 的 2 个因子**则**
根据图4中的算法（补充材料）
注意力 $\leftarrow$ 2 因子 HLA(q，k_1，k_2，v)
结束如果
如果使用 HLA 的 3 个因子**则**
依据补充材料中的图 5 算法
注意力 $\leftarrow$ 3 因子 hla(q，k_1，k_2，k_3，v)
结束 if

返回到 out(attn)

通道由一个随机投影矩阵生成，其矩阵元素从均匀分布 $\in [0, 1]$ 中选取。随后，我们对张量进行归一化，使所有元素都处于 $[0, 1]$ 中。可以看出，中间的 Transformer 块 11 和 18 中的自注意力对整体结构贡献很大。

表5 展示了值调制的贡献。可以看出，该模块在块 2 和 3 的步数 30、40 和 40 时就已对结构有所贡献。块 4 和 11 显然主要在步数 10 、 20 、 30 和 40 时增加显著结构。块 26 则在所有较高采样步中增加高频信息。

C. VBench 分数

我们在图4中给出了各类别的 VBench 分数。深蓝色线条对应**HLA-3F-R1-10**，浅蓝色线条对应**HLA-3F-R1-21** ，红色线条对应**HLA-3F-R2-15**。

图6展示了包含合成数据的最后阶段在不同训练迭代次数下的 VBench 分数。颜色表示与图4中相同的变体。所有三种方法的分数均有所提升，这表明我们的合成数据带来了积极影响。

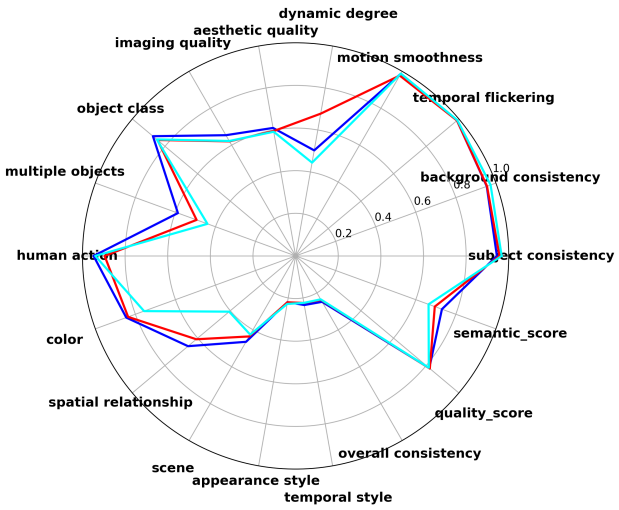


图 *4.* 各类别的 VBench 分数。深蓝色线条对应**HLA-3F-R1-10**，红色线条对应**HLA-3F-R1-21** ，浅蓝色线条对应**HLA- 3F-R2-15**。



图 *5.* 由**HLA-3F-R1-21** 经 50 步生成的视频的第一帧。

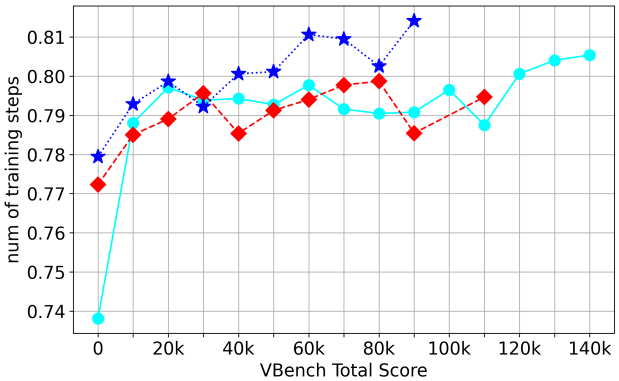


图 *6.* 随迭代次数变化的 VBench 分数。VBench 分数使用 40% 的提示词计算。蓝色线条表示**HLA-3F-R1-10**，红色线条表示**HLA-3F-R2-15**，浅蓝色线条表示**HLA-3F-R1-21**。

表 *4.* 值调制前的注意力输出。列对应生成过程中的不同采样步，行表示 Transformer 块的索引。

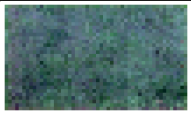
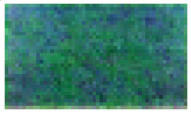
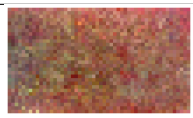
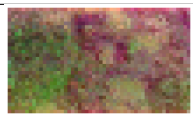
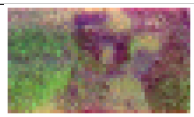
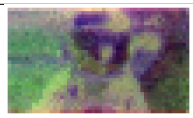
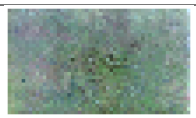
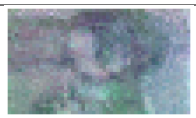
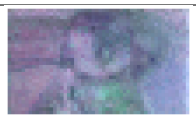
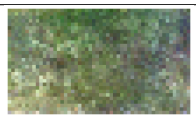
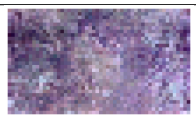
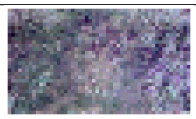
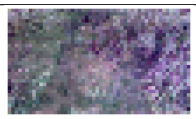
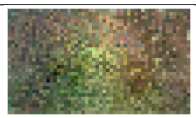
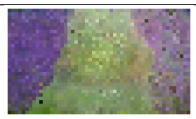
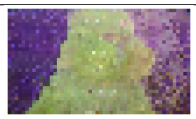
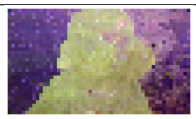
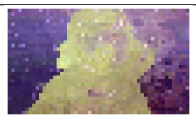
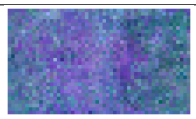
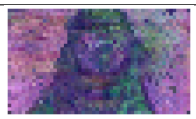
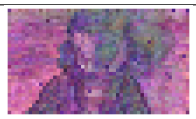
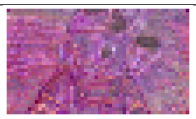
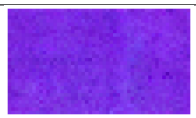
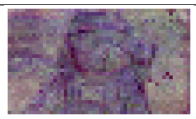
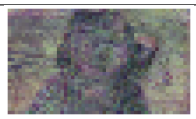
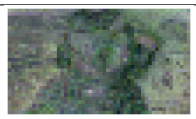
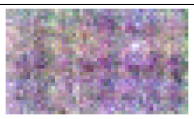
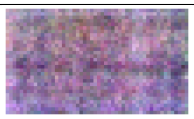
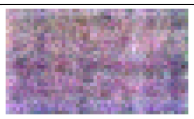
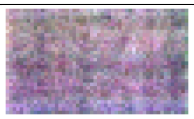
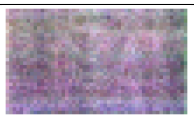
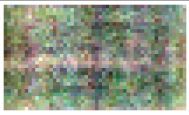
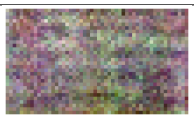
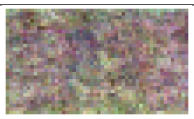
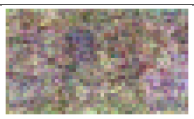
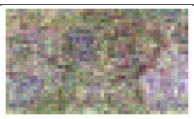
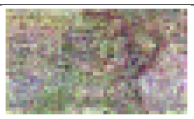
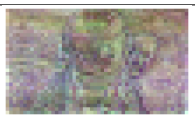
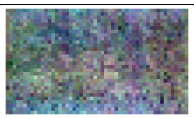
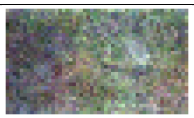
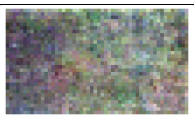
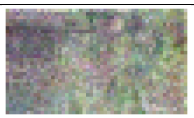
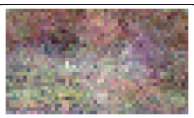
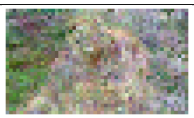
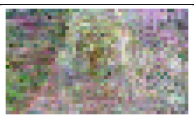
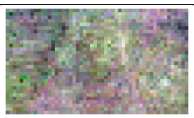
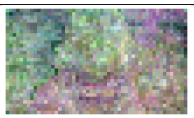
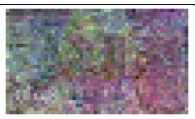
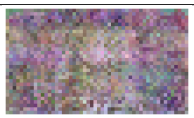
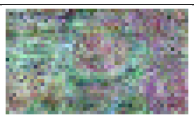
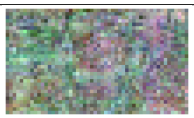
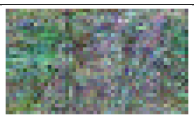
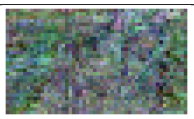
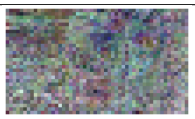
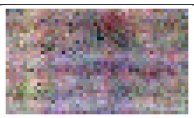
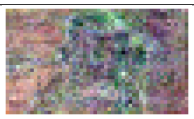
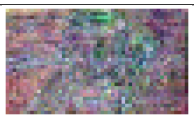
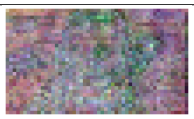
	第 1 步	第 10 步	第 20 步	第 30 步	第 40 步	第 50 步
1						
2						
3						
4						
11						
18						
26						

图 7. 标准二次注意力与 2 因子或 3 因子 HLA 在不同序列长度下的计算复杂度比较。

表*5.* 值调制对注意力更新的贡献。列对应生成过程中的不同采样步，行表示 Transformer 块的索引。

	第1步	第10步	第20步	第30步	第40步	第 50 步
1						
2						
3						
4						
11						
18						
26						

D. 定性示例

我们在图 8、9 和10中展示了三种变体的更多定性示例。如图所示，这三种方法都能生成高质量视频。另请参阅补充材料中的原始 mp4 文件。

E. 代码清单

我们在代码清单 11 和12中提供了类似 PyTorch 的代码。这些算法不需要耗时的张量重塑，只需进行基本的计算操作。

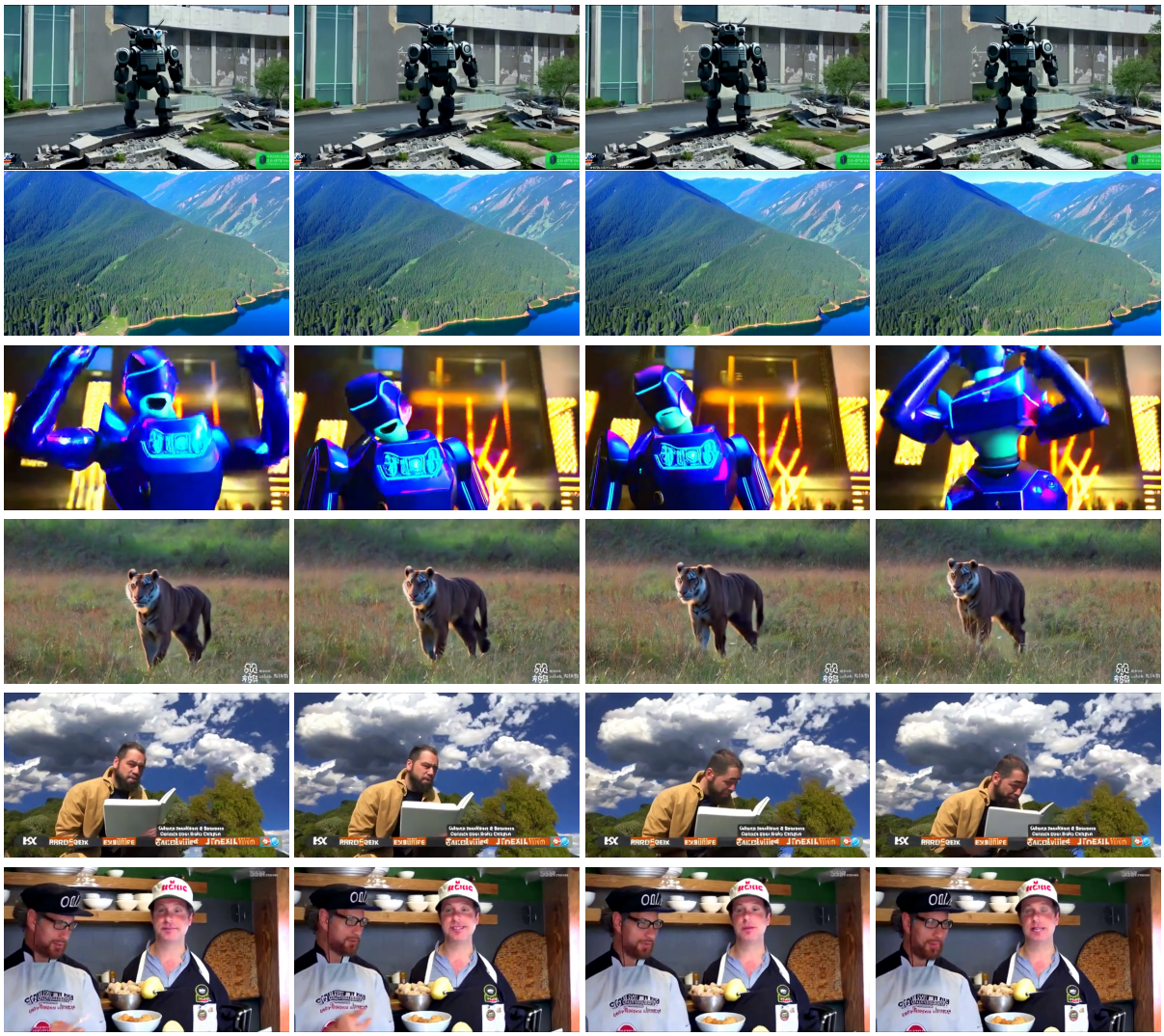


图*8*。 定性示例 **HLA-3F-R1-10**

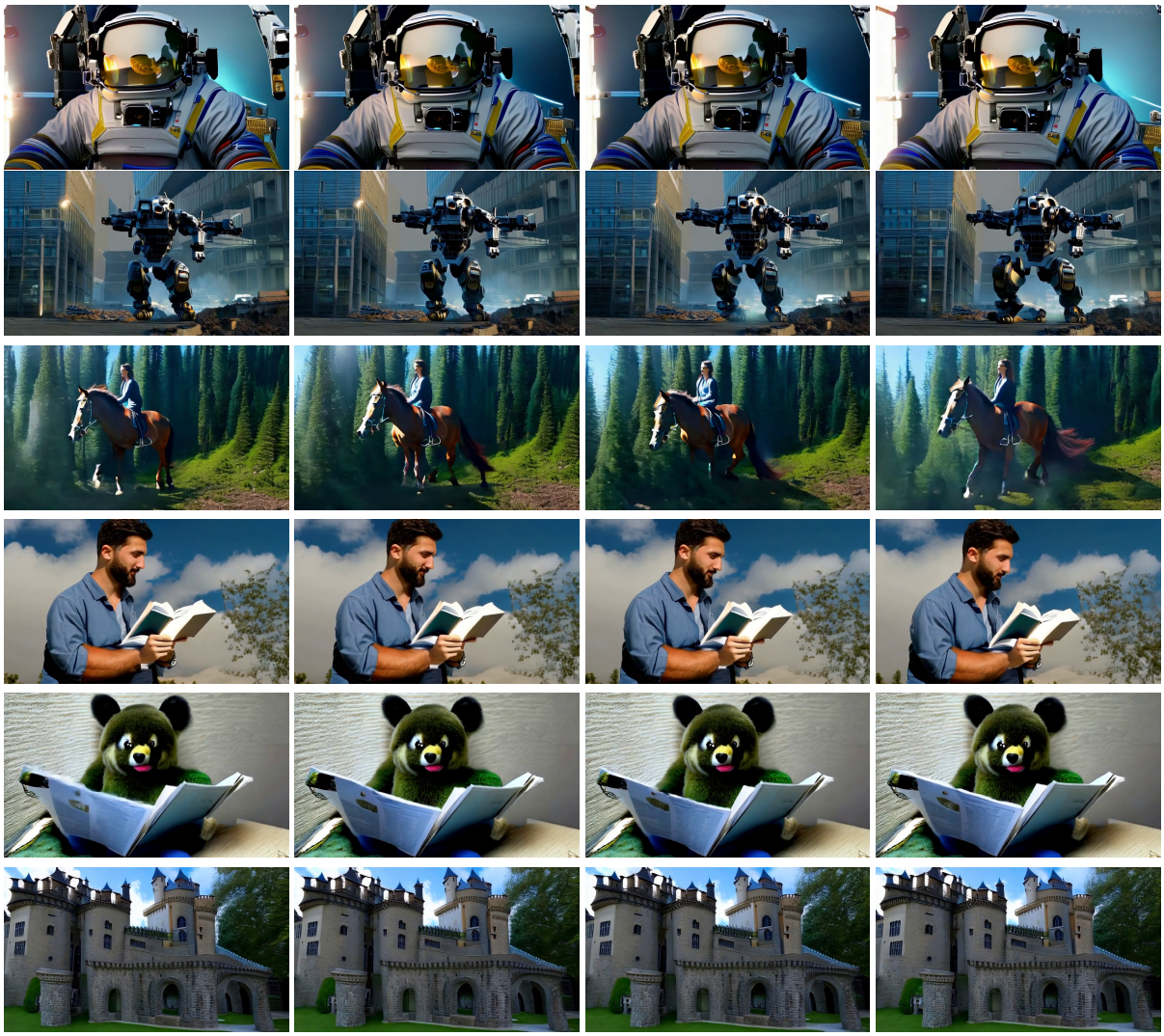


图*9.* 定性示例 **HLA-3F-R1-21**

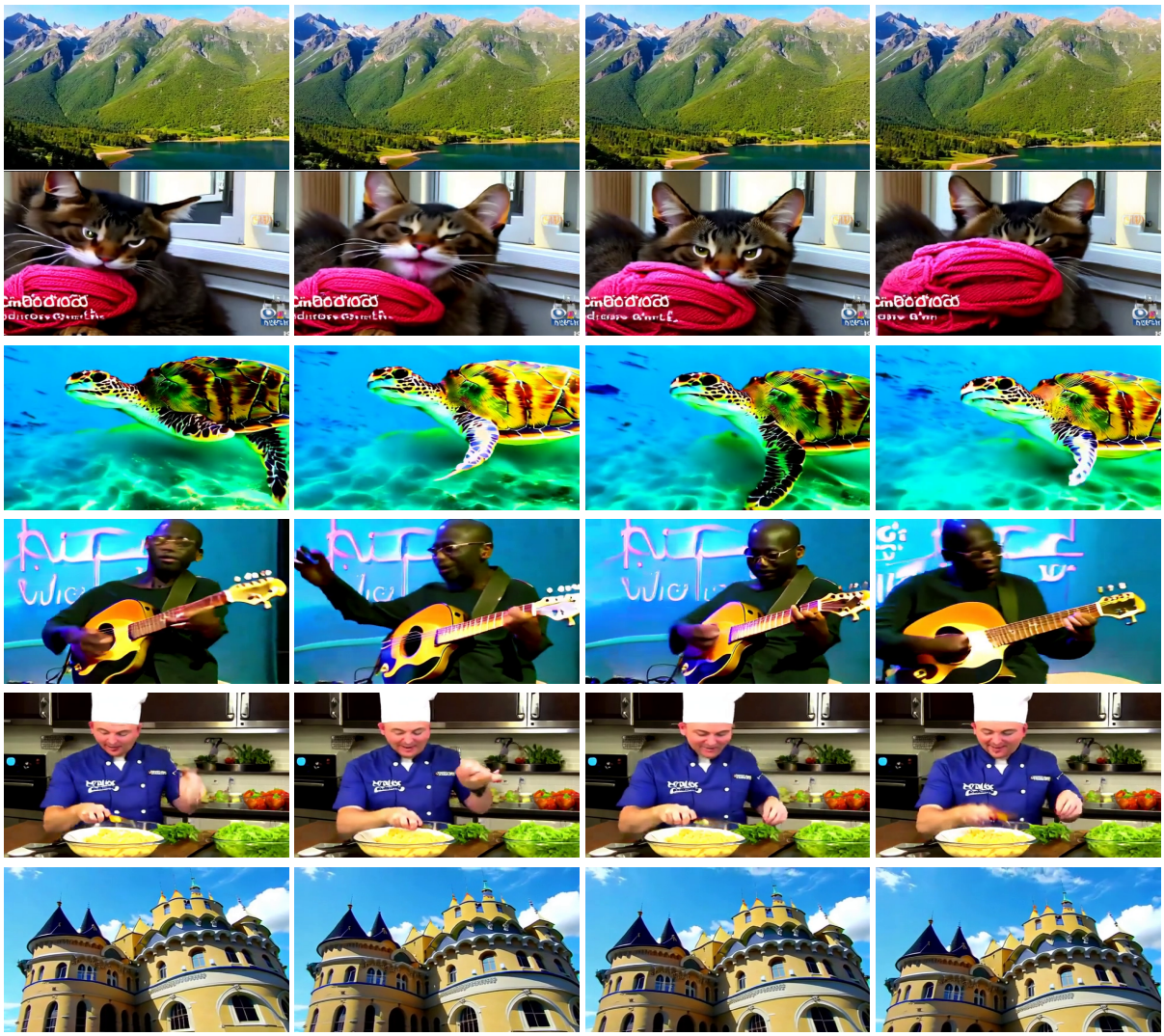


图*10.* 定性示例 **HLA-3F-R2-15**

```
定义 HLA_2_因子(查询,                键1,      键2,  值):
    # 等价于 ((Q@K1.t())                  * (Q@K2.t())) @ V
    #（为简洁起见，省略归一化）

    # 批次大小、头数、序列长度、d_phi
    bs,  nh、sl、   dp  = 查询.shape
                    d   = 值.shape[-1]

    # 计算查询与键 1、键 2 词元之间的外积
    k_外积_k   = torch.einsum('bhsd,bhse->bhsde',key1,                键2)
    上下文             = torch.einsum(
                            'bhmde,bhmg->bhdeg',
                            k_outerprod_k, 值
                        )
    q_outerprod_q   = torch.einsum('bhsd,bhse->bhsde',查询, 查询)

    # 归一化
    对k求和_     _      = k_外积_k.求和(维度=2,  保持维度=True)
    eta             = torch.sum(
                            q_外积_q.视图(bs,nh,sl,-1)
                            *
                            求和_对_k.视图(bs,nh,1,-1)    ,
                            dim=-1, keepdim=True
                        )
    q_外积_q =q_外积_q /(eta.视图(bs,nh,sl,1,1)+eps)

    # 内存高效计算
    q_外积_q   = q_外积_q.视图(bs,  nh,  sl,                dp**2)
    上下文             = 上下文.视图(bs,      nh,  dp**2,  d)

    # 线性注意力的输出（输出投影之前）
    注意力             = torch.einsum(
                            'bhsd，bhde->bhse' ,
                            q_外积_q,  上下文
                        )

    返回注意力
```

图 *11.* Hadamard线性注意力（含 2 个因子）的 PyTorch 风格代码。

```
定义 HLA_3_因子(查询,                键1,    键2,    键3, 值):
    # 等价于                  ((Q@K1.t())    * (Q@K2.t())    * (Q@K3.t()))    @ V
    #（为简化起见，省略归一化）

    # 批次大小、头数、序列长度、d_phi
    bs, nh, sl,    dp = 查询.shape
                   d  = 值.shape[-1]

    # 计算查询与键 1、键 2 词元之间的外积
    k 外积          = torch.einsum(
                          'bhsd ,bhse,bhsf->bhsdef'  ,
                          键1,    键2,    key3
                      )
     上下文         = torch.einsum(
                          'bhmdef,bhmg->bhdefg',
                          k_outerprods, 值
                      )
    q 外积          = torch.einsum(
                          'bhsd ,bhse,bhsf->bhsdef'  ,
                          查询, 查询, 查询
                      )

    # 归一化
    对 k 求和_    _ = k_外积。求和(维度=2, 保持维度=True)
    eta             = torch.sum(
                          q_外积。视图(bs,nh,sl,-1)
                          *
                          求和_关于_k。视图(bs,nh,1,-1)  ,
                          维度=-1, 保持维度=True
                      )
    q_外积          = q_外积 / (eta.视图(bs,nh,sl,1,1,1)+eps)

    # 内存高效计算
    q 外积          = q_外积。视图(bs,  nh,  sl,        dp**3)
     上下文         = 上下文。视图(bs,    nh,  dp**3,  d)

    # 线性注意力的输出（输出投影之前）
    注意力          = torch。einsum(
                          'bhsd ,bhde->bhse'  ,
                          q_outerprods, 上下文
                      )

    返回 注意力
```

图 *12.* Hadamard线性注意力（含 3 个因子）的 PyTorch 风格代码。