Flash-Genie：一种面向长序列蛋白质结构生成的高效内存框架

Jiayu Wu

2026 年 1 月

Abstract

本文基于 Genie [Lin and AlQuraishi, 2023] 提出了 Flash-Genie，一种新型蛋白质结构生成框架，旨在解决长序列蛋白质建模中的计算挑战。Flash-Genie 整合了多项关键创新：用于高效注意力计算的 Flash 不变点注意力（Flash-IPA）、用于稳定梯度的流形约束超连接（mHC）以及将内存复杂度从 $O(L^2)$ 降低到 $O(L \cdot R)$ 的分解成对特征。与传统方法相比，我们的框架能实现内存节省，同时在蛋白质结构预测任务上仍然有不错的表现。

1 引言

蛋白质结构预测随着深度学习方法的出现经历了革命性变革，例如 Genie [Lin and AlQuraishi, 2023] 通过等变扩散定向残基云来生成新颖、可设计和多样化的蛋白质结构。然而，这些方法在应用于长序列蛋白质时面临重大挑战。传统方法需要 $O(L^2 \cdot C)$ 的内存来存储成对特征，而三角运算则需要 $O(L^3)$ 的时间复杂度，这使得超过 1024 个残基的序列在现有硬件上计算难以进行。此外，Evoformer 中多层的深度网络在处理长序列时经常出现梯度爆炸或消失，标准注意力机制需要显式构建 $L \times L$ 注意力矩阵，导致二次的内存和计算成本。

Flash-Genie 通过系统的优化方法应对这些挑战。我们将 Flash Attention [Dao et al., 2022]、Flash-IPA [Liu et al., 2025] 和 mHC 架构 [Xie et al., 2026] 与分解成对特征 [Ho et al., 2019] 集成，以直接生成分解表示而不产生中间 L^2 张量。实现了三角运算分解，在保持类 Evoformer 处理表达能力的同时避免了 $O(L^3)$ 的内存复杂度。随后引入了渐进式训练调度器和分块损失计算，允许课程学习策略以实现长序列的稳定收敛。当序列长度超过实际限制时，采用稀疏 kNN 成对选择将内存从 $O(L^2)$ 减少到 $O(L \cdot k)$。在此之后，通过轴向注意力和模型压缩技术（包括层共享和瓶颈架构）进一步优化计算效率。最后，通过分布式数据并行、序列张量并行和梯度累积机制实现大规模训练。

2 方法论 (Methodology)

2.1 问题形式化

给定蛋白质序列 $S = (s_1, s_2, \ldots, s_L)$，其中 s_i 表示位置 i 处的氨基酸，目标是预测所有原子或主链原子的三维坐标 $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L)$。传统基于扩散的方法 [Lin and AlQuraishi, 2023] 将蛋白质结构空间中的生成过程建模为生成过程。核心挑战在于高效表示和处理所有残基对 (i, j) 之间的成对相互作用，这自然导致 $L \times L$ 成对张量 $\mathbf{P} \in \mathbb{R}^{L \times L \times C}$。

AlphaFold2[Jumper et al., 2021] 的 Evoformer 模块通过交替的注意力和转换操作处理单表示 $\mathbf{s} \in \mathbb{R}^{L \times C_s}$ 和成对表示 $\mathbf{p} \in \mathbb{R}^{L \times L \times C_p}$。关键的计算瓶颈是成对表示，由于三角运算需要 $O(L^2 \cdot C)$ 的内存和 $O(L^3)$ 的时间。对于 $L = 2048$ 和 $C = 128$，仅成对张量（FP32）就需要约 2.1 GB 的内存，不包括梯度或中间激活值。

2.2 分解成对特征

分解成对特征模块将 $O(L^2)$ 成对张量分解为低秩因子。给定秩 R，成对表示重构为：

$$\mathbf{p}_{ij} = \sum_{r=1}^{R} (\mathbf{f}_{L,i,r} \odot \mathbf{f}_{R,j,r}) \tag{1}$$

$$\mathbf{f}_{L,i,r} = \mathbf{f}_{L,i,r}^{(s)} + \mathbf{f}_{L,i,r}^{(\text{rel})} + \mathbf{f}_{L,i,r}^{(\text{tmpl})} \tag{2}$$

$$\mathbf{f}_{R,j,r} = \mathbf{f}_{R,j,r}^{(s)} + \mathbf{f}_{R,j,r}^{(\text{rel})} + \mathbf{f}_{R,j,r}^{(\text{tmpl})} \tag{3}$$

其中 $\odot$ 表示逐元素乘法。内存复杂度从 $O(L^2 \cdot C)$ 减少到 $O(L \cdot R \cdot C)$。
分解相对位置编码计算为：

$$\mathbf{h}_i = [\text{Emb}_{\text{abs}}(i); \text{Emb}_{\text{bin}}(\text{clamp}(i - L/2))] \tag{4}$$

$$\mathbf{f}_{L,i,r}^{(\text{rel})} = \mathbf{W}_L^{(\text{pos})} \mathbf{h}_i + \mathbf{b}_{\text{rel},r} \tag{5}$$

$$\mathbf{f}_{R,j,r}^{(\text{rel})} = \mathbf{W}_R^{(\text{pos})} \mathbf{h}_j - \mathbf{b}_{\text{rel},r} \tag{6}$$

分解模板特征使用注意力池化：

$$\mathbf{v}_{\text{diag},i} = \mathbf{T}_{ii} \tag{7}$$

$$\mathbf{v}_{\text{row},i} = \sum_k \text{Softmax}(\mathbf{w}_q^T \mathbf{T}_{ik}) \cdot \mathbf{T}_{ik} \tag{8}$$

$$\mathbf{v}_{\text{col},i} = \sum_k \text{Softmax}(\mathbf{w}_q^T \mathbf{T}_{ki}) \cdot \mathbf{T}_{ki} \tag{9}$$

$$\mathbf{f}_{L,i,r}^{(\text{tmpl})} = (\mathbf{W}_U \mathbf{u}_{L,i}) \cdot \sigma_r \tag{10}$$

$$\mathbf{f}_{R,j,r}^{(\text{tmpl})} = \mathbf{W}_V \mathbf{u}_{R,j} \tag{11}$$

其中 σ_r 是学习到的奇异值标量。
分解三角乘法更新计算为：

$$A_{left} = \text{Linear}_a(\text{LN}(Z_{left})) \odot \sigma(\text{Linear}_{g\_a}(\text{LN}(Z_{left}))) \tag{12}$$

$$B_{right} = \text{Linear}_b(\text{LN}(Z_{right})) \odot \sigma(\text{Linear}_{g\_b}(\text{LN}(Z_{right}))) \tag{13}$$

$$\tilde{A} = A_{left} W_{mix\_a}^T, \quad \tilde{B} = B_{right} W_{mix\_b}^T \tag{14}$$

$$\bar{B} = \frac{1}{L} \sum_{k=1}^{L} \tilde{B}_k \tag{15}$$

$$U_{left} = \tilde{A} \odot \bar{B} \tag{16}$$

分块三角注意力聚合分解表示：

$$Z_{pair} = \text{LN}\left(\sum_{r=1}^{R} Z_{left,r} + \sum_{r=1}^{R} Z_{right,r} \right) \tag{17}$$

$$Q, K, V = Z_{pair} W_Q, Z_{pair} W_K, Z_{pair} W_V \tag{18}$$

$$\text{Scores}^{(m)} = \frac{Q_{[i:i+S]} K^T}{\sqrt{d_h}} + B_{bias[i:i+S]} \tag{19}$$

其中 S 是块大小。

流形约束超连接（mHC）架构通过在扩展的" 超空间"$\mathcal{Z} \in \mathbb{R}^{L\times n\times C}$ 中维护表示来解决梯度流稳定性问题：

$$\bar{x} = \text{RMSNorm}(\text{Flatten}(\mathcal{Z})) \tag{20}$$
$$\mathbf{H}_{pre} = \sigma(\mathbf{W}_{pre}\bar{x} + b_{pre}) \in \mathbb{R}^{1\times n} \tag{21}$$
$$\mathbf{H}_{post} = 2 \cdot \sigma(\mathbf{W}_{post}\bar{x} + b_{post}) \in \mathbb{R}^{n\times 1} \tag{22}$$
$$\mathbf{A} = \mathbf{W}_{res}\bar{x} + b_{res} \in \mathbb{R}^{n\times n} \tag{23}$$
$$\mathbf{H}_{res} = \text{SinkhornKnopp}(\mathbf{A}) \tag{24}$$

Sinkhorn-Knopp 迭代确保 $\mathbf{H}_{res}$ 是双随机的（行和列和等于 1），保证谱半径为 1，从而通过深度网络实现稳定梯度流。更新机制如下：

$$s_{contracted} = \mathbf{H}_{pre} \cdot \mathcal{S} \tag{25}$$
$$\Delta s = \text{IPA}(s_{contracted}, p, T, \text{mask}) \tag{26}$$
$$\Delta\mathcal{S} = \mathbf{H}_{post} \otimes \Delta s \tag{27}$$
$$\mathcal{S}^{(l+1)} = \mathbf{H}_{res} \cdot \mathcal{S}^{(l)} + \Delta\mathcal{S} \tag{28}$$

2.3 渐进式训练和混合精度

课程学习调度器在训练期间动态调整序列长度：

$$p = \text{clamp}\left(\frac{t - T_{warmup}}{T_{growth}}, 0, 1\right) \tag{29}$$
$$\alpha = \text{schedule}(p) \quad \text{（线性、余弦或指数）} \tag{30}$$
$$L_{curr}(t) = \begin{cases} L_{min} & t < T_{warmup} \\ \lfloor L_{min} + \alpha \cdot (L_{max} - L_{min}) \rfloor & T_{warmup} \le t < T_{total} \\ L_{max} & \text{否则} \end{cases} \tag{31}$$

分块损失计算避免构建完整的 $L \times L$ 距离矩阵：

$$d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2, \quad \hat{d}_{ij} = \|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\|_2 \tag{32}$$
$$E_{ij} = \min\left(|\hat{d}_{ij} - d_{ij}|, \tau\right) \tag{33}$$
$$\mathcal{L}_{dist} = \frac{\sum_k \sum_{i\in\text{chunk}_k} \sum_{j=1}^{L} M_{ij} E_{ij}}{\sum_{i,j} M_{ij} + \epsilon} \tag{34}$$

内存复杂度从 $O(L^2)$ 减少到 $O(S \cdot L)$，其中 S 是块大小。混合精度训练通过 FP16/BF16 支持和动态损失缩放提供约 50% 的内存节省和 2-3 倍训练加速。

mHC 正则化损失通过多种机制稳定深度网络训练。残差平衡损失约束残差和主分支贡献之间的比率：

$$\rho = \frac{\|\mathbf{F}(\mathbf{x})\|_M^2}{\|\mathbf{x}_{in}\|_M^2 + \|\mathbf{F}(\mathbf{x})\|_M^2 + \delta} \tag{35}$$
$$\mathcal{L}_{balance} = (\rho - \gamma)^2 \tag{36}$$

梯度范数保持损失维持稳定的梯度流：

$$r = \frac{\|\mathbf{x}_{out}\|_M}{\|\mathbf{x}_{in}\|_M + \delta} \tag{37}$$

$$\mathcal{L}_{norm} = \text{ReLU}(|r - 1| - \tau)^2 \tag{38}$$

双随机惩罚强制权重矩阵的双随机属性：

$$\tilde{W} = \exp(W) \tag{39}$$

$$\mathcal{L}_{DS} = \frac{1}{N}\sum_i(\sum_j \tilde{W}_{ij} - 1)^2 + \frac{1}{N}\sum_j(\sum_i \tilde{W}_{ij} - 1)^2 \tag{40}$$

对于长序列（$L > 1024$），自适应损失函数动态调整容差和权重：

$$\lambda = \sqrt{L/L_{base}} \tag{41}$$

$$\tau_{long} = \max(0.1, \frac{0.2}{\lambda}) \tag{42}$$

$$\mathcal{L}_{adaptive} = \lambda \cdot \text{ReLU}\left(\left|\frac{\|\epsilon_{pred}\|}{\|\epsilon_{target}\|} - 1\right| - \tau_{long}\right)^2 \tag{43}$$

2.4 稀疏 kNN 成对选择

对于超长序列，稀疏 kNN 成对选择将内存从 $O(L^2)$ 减少到 $O(L \cdot k)$。混合选择策略结合了基于坐标和基于序列的邻居：

$$d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2 \tag{44}$$

$$\mathcal{N}_i^{coord} = \underset{j}{\text{argtopk}}(d_{i,:}, k_{coord}, \text{largest=False}) \tag{45}$$

$$\mathcal{N}_i^{seq} = \{j \mid \max(0, i - k_{seq}/2) \le j \le \min(L - 1, i + k_{seq}/2)\} \tag{46}$$

$$\mathcal{N}_i^{hybrid} = \mathcal{N}_i^{coord} \cup \mathcal{N}_i^{seq} \tag{47}$$

内存复杂度比较：

$$\text{密集成对：} \quad O(L^2 \cdot C) \tag{48}$$

$$\text{稀疏成对（}k = 32\text{）：} \quad O(L \cdot k \cdot C) \approx \frac{1}{120}O(L^2 \cdot C) \quad (L = 4096) \tag{49}$$

强制性局部成对确保二级结构信息的覆盖：

$$\mathcal{N}_i^{local} = \{j \mid |i - j| \le w\} \tag{50}$$

$$\mathcal{N}_i^{final} = \mathcal{N}_i^{knn} \cup \mathcal{N}_i^{local} \tag{51}$$

其中 w 是局部窗口大小。

2.5 轴向注意力和模型压缩

轴向注意力将二维注意力分解为顺序的一维操作：

$$X_{row\_view} \in \mathbb{R}^{(B\cdot L)\times L\times C} \tag{52}$$

$$X^{(1)} = X + \text{Attention}_{row}(X_{row\_view}) \tag{53}$$

$$X_{transposed} = (X^{(1)})^T \tag{54}$$

$$X^{(2)} = X^{(1)} + (\text{Attention}_{col}(X_{transposed}))^T \tag{55}$$

内存复杂度从 $O(L^2)$ 减少到分块实现的 $O(L)$，计算复杂度从 $O(L^3)$ 减少到 $O(L^2)$。
通过层共享的模型压缩显著减少参数。通用层共享重复应用相同的层 f_θ：

$$x^{(l)} = f_\theta(x^{(l-1)}), \quad l = 1, \ldots, L \tag{56}$$

$$\text{参数} \propto O(1 \times |\theta|) \tag{57}$$

交替层共享使用两组参数：

$$x^{(l+1)} = \begin{cases} f_{\theta_{even}}(x^{(l)}) & l \equiv 0 \pmod 2 \\ f_{\theta_{odd}}(x^{(l)}) & l \equiv 1 \pmod 2 \end{cases} \tag{58}$$

分块层共享将层分为 K 个块：

$$x^{(k,m+1)} = f_{\theta_k}(x^{(k,m)}), \quad m \in [0, M-1] \tag{59}$$

$$\text{参数} \propto O(K \times |\theta|) \tag{60}$$

瓶颈层通过维度投影减少计算成本：

$$x_{bot} = xW_{down} + b_{down}, \quad W_{down} \in \mathbb{R}^{d_{in}\times(d_{in}/r)} \tag{61}$$

$$h = \text{Operation}(x_{bot}) \tag{62}$$

$$y = hW_{up} + b_{up}, \quad W_{up} \in \mathbb{R}^{(d_{in}/r)\times d_{in}} \tag{63}$$

$$\text{输出} = \text{LayerNorm}(x + y) \tag{64}$$

深度可分离层进一步减少参数：

$$Y_{c,i} = \sum_k W^{depth}_{c,k} \cdot X_{c,i+k} \tag{65}$$

$$Z_{j,i} = \sum_c W^{point}_{j,c} \cdot Y_{c,i} \tag{66}$$

与标准卷积相比，深度可分离层将参数从 $C_{out} \times C_{in} \times K$ 减少到 $C_{in} \times K + C_{out} \times C_{in}$。
压缩比计算为：

$$\text{比率} = \frac{L \times |\theta|_{base}}{|\Theta|_{shared}} \tag{67}$$

对于 $L = 12$ 层，通用共享实现 12 倍压缩，交替共享实现 6 倍，分块共享（大小 =4）实现 4 倍压缩。

2.6 阶段 5：分布式训练

分布式数据并行（DDP）通过同步梯度更新实现多 GPU 训练。序列张量并行沿序列维度分割计算，允许不同 GPU 处理蛋白质序列的不同部分。梯度累积通过在多个微批次上累积梯度来实现大于物理 GPU 内存的有效批量大小：

$$\text{梯度}_{total} = \sum_{m=1}^{M} \text{梯度}_{micro\_batch}^{(m)} \tag{68}$$

这种方法实现稳定的大批量训练，同时保持类似于单批次优化的梯度统计。
有效批量大小随 GPU 数量线性缩放：

$$\text{eff}_{bs} = \text{eff}_{bs\_per\_GPU} \times n_{GPUs} \tag{69}$$

多 GPU 训练在保持统计等价于单 GPU 训练的同时减少壁钟时间，这通过正确同步模型参数和梯度来实现。

2.7 mHC 集成

mHC 成对变换层通过收缩-变换-扩展循环处理成对表示：

$$\tilde{Z}_{ij} = H_{pre,ij} \cdot \mathcal{Z}_{ij} = \sum_{k=1}^{n} H_{pre,ij}^{(k)} \cdot z_{ij}^{(k)} \tag{70}$$

$$\Delta Z_{ij} = F(\tilde{Z}_{ij}) \quad \text{（标准 Evoformer 操作）} \tag{71}$$

$$\Delta \mathcal{Z}_{ij} = H_{post,ij} \otimes \Delta Z_{ij} \tag{72}$$

$$\mathcal{Z}_{ij}^{(l+1)} = H_{res,ij} \cdot \mathcal{Z}_{ij}^{(l)} + \Delta \mathcal{Z}_{ij} \tag{73}$$

mHC 结构网络将超连接应用于单表示：

$$s_{contracted,i} = \mathbf{H}_{pre,i} \cdot \mathcal{S}_i \tag{74}$$

$$\Delta s_{ipa} = \text{LayerNorm}(\text{Dropout}(\text{IPA}(s_{contracted}, p, T, \text{mask}))) \tag{75}$$

$$\Delta s_{total} = \text{Transition}(\Delta s_{ipa}) \tag{76}$$

$$\Delta \mathcal{S}_i = \mathbf{H}_{post,i} \otimes \Delta s_{total} \tag{77}$$

$$\mathcal{S}_i^{(l+1)} = \mathbf{H}_{res,i} \cdot \mathcal{S}_i^{(l)} + \Delta \mathcal{S}_i \tag{78}$$

主干更新使用收缩的特征：

$$s_{for\_bb} = \text{Contract}(\mathcal{S}^{(l+1)}) \tag{79}$$

$$T^{(l+1)} = T^{(l)} \circ \text{BackboneUpdate}(s_{for\_bb}) \tag{80}$$

mHC Flash 结构网络结合 mHC 与 Flash-IPA 以实现最佳内存效率：

$$s_{in,i} = \mathbf{H}_{pre,i} \cdot \mathcal{S}_i \tag{81}$$

$$b_{ij} \approx \text{Linear}(Z_{fac1,i} \odot Z_{fac2,j}) \tag{82}$$

$$s_{ipa} = \text{FlashAttention}(Q, K, V, \text{BiasFactors}) \tag{83}$$

$$\Delta s = \text{Transition}(\text{LayerNorm}(\text{Dropout}(s_{ipa}))) \tag{84}$$

$$\mathcal{S}_i^{(l+1)} = \mathbf{H}_{res,i} \cdot \mathcal{S}_i^{(l)} + \mathbf{H}_{post,i} \otimes \Delta s_i \tag{85}$$

$$T^{(l+1)} = T^{(l)} \circ \text{BackboneUpdate}(s_{ipa}) \tag{86}$$

内存复杂度比较：

$$\text{标准 IPA：} \quad O(L^2 \cdot H) \tag{87}$$

$$\text{Flash-IPA：} \quad O(L \cdot H) \tag{88}$$

mHC 和 Flash-IPA 的结合通过双随机混合实现梯度稳定性，通过 Flash Attention 算法实现线性内存缩放。

3 局限性与未来工作

尽管 Flash-Genie 在内存效率方面取得了显著改进，但一些局限性仍然存在。首先，稀疏 kNN 成对选择策略虽然将内存复杂度降低到 $O(L \cdot k \cdot C)$，但当邻居数量 k 较小时，可能无法完全捕获残基的蛋白质中的非常长程依赖关系。其次，mHC 中的动态投影矩阵和分解表示引入了额外的计算开销，对于二次缩放不构成问题的短序列，这可能抵消一些内存节省。另外，尽管 mHC 提供了出色的梯度稳定性，但成对特征的内存复杂度仍然保持在 $O(L^2 \cdot n \cdot C)$，限制了 n 的实际使用大选，单表示可使用 $n = 4$，而成对表示的实际应用可能只能使用 $n = 2$。从数据集的方向来看，训练数据需求仍然很大，渐进式训练课程的有效性取决于多样化长序列训练样本的可用性。更重要的 FlashAttention 实现通常将头维度严格限制为 256，当超过 256 时，GPU 甚至无法在一个线程块（Thread Block）的共享内存（SRAM）中放下最小的有效计算单元，导致算法无法运行或效率极低，这限制了分解秩，并可能限制密集成对表示的近似质量。最后，用于双随机矩阵的 Sinkhorn-Knopp 迭代在每层引入了额外的计算成本，尽管具有有益的梯度属性，但对于非常深的网络来说可能变得显著。

未来，我们计划探索更先进的稀疏注意力模式，以更好地捕获长程依赖关系，同时保持线性内存复杂度。我们还将研究更高效的 Sinkhorn-Knopp 近似，以减少迭代开销。此外，我们将探索动态 mHC 参数 n 根据序列长度自动调整的可能性，以在不同序列长度上实现最佳效率-性能权衡。最后，我们将把框架扩展到支持多链蛋白质复合物的建模。

References

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers, 2019. URL https://arxiv.org/abs/1912.12180.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, July 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. URL http://dx.doi.org/10.1038/s41586-021-03819-2.

Yeqing Lin and Mohammed AlQuraishi. Generating novel, designable, and diverse protein structures by equivariantly diffusing oriented residue clouds, 2023. URL https://arxiv.org/abs/2301.12485.

Andrew Liu, Axel Elaldi, Nicholas T Franklin, Nathan Russell, Gurinder S Atwal, Yih-En A Ban, and Olivia Viessmann. Flash invariant point attention, 2025. URL https://arxiv.org/abs/2505.11580.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mhc: Manifold-constrained hyper-connections, 2026. URL https://arxiv.org/abs/2512.24880.