基于极微模型与表达涌现的新一代 AI 底座：一项未经验证的构想

投稿方向：AI 大模型理论构想论文
作者：郭富（ORCID: 0009-0000-8142-6457）
第二作者：郭付信
日期：2026-11-26

摘　要

大语言模型时代，神经网络仍沿袭一条近乎固执的路线：以千亿级静态权重为核心，以自回归方式逐个吐出 token。权重在训练结束后被"定死"，上下文在拼接完成后被"焊死"，任何对历史内容的中途修改都可能引发键值缓存（KV cache）的连锁失效。本文提出一种新一代 AI 底座构想，极微模型（Micro-Model Unit，MMU）。极微模型是结构同一、参数极小的通用计算单元，通常只做两件事：承接上游表达，产生本级表达并移交下游。字、词、句乃至整段话在底座中处于平权地位，其具体意义不由静态权重预装，而在表达链的传导过程中随上下文涌现。在此基础上，本文给出三项范式转变：其一，计算对象由"整片神经网络"变为"表达序列"，文本以整体排序（数组）的形式参与数组与数组之间的计算；其二，输入输出采用全双工模式，上下文在内存中常驻为活的数组，可在任意位置热插入、热删除极微模型，整段文本的意义排序随之重排，而无需重建缓存或人工干预；其三，图像可近似为固定的颜色数组，与文本、数字、音频统一为纯数字表达，识图极微模型可随插随拔，上下文瞬间连通。在传播方式上，表达链并非单向流而是头尾双向矫正，任意位置皆可成为上游，下一个字同时影响上一个字的表达；内存中自始至终只存放极微小模型及其承载的表达，所谓上下文修改，改变的只是表达而非内容本身。本文依次给出架构定义、计算范式、复杂度分析与八个思想实验，并坦诚讨论训练目标缺失、链式传播延迟、容量上限等局限。本文提出的是一个未经验证的猜想，不含实验数据与实验验证，结论有待后续工程工作检验。

关键词：关键词：极微模型；表达涌现；全双工计算；热插拔上下文；统一数组表示；双向矫正；底座架构

目　录

1 绪论 5
2 相关工作 6
3 极微模型底座 8
3.1 设计动机 8
3.2 定义 8
3.3 表达链与意义涌现 9
3.4 与传统前向计算的对照 10
3.5 极微模型的形态谱系与意义网络 10
3.6 双向传播与头尾矫正 11
4 全双工数组计算与热插拔上下文 12
4.1 整体排序表示：数组对数组的计算 12
4.2 全双工模式 12
4.3 热插入、热删除与缓存不失效 12
4.4 解释还原 13
4.5 表达接入校验与被动自排查 14
4.6 分布式、复用与成本 14
4.7 内容治理：可停用的表达槽位 15
4.8 模糊输入与模糊输出 16
4.9 意义的范围与分段语义 16
4.10 内存中存什么：上下文的不存在 17
5 多模态统一表达 18
5.1 图像即颜色数组 18
5.2 识图极微模型的热插拔 18
5.3 数字、文本、音频的统一 18
5.4 输出端的重新展示 18
5.5 视图的无限细分与整合 19
5.6 语言无关与免再训练 19
5.7 意义的聚合与细分 19
5.8 从表达到可读文字：生成机制的纯粹猜想 21
6 复杂度分析与思想实验 23
6.1 计算模型分析 24
6.2 思想实验 A：截稿前改稿 24

6.3 思想实验 B：边听边改的对话 24
6.4 思想实验 C：图文混排的即插即用 24
6.5 思想实验 D：一个字的平权 25
6.6 思想实验 E：三个人同时改一句话 25
6.7 思想实验 F：排错工况 25
6.8 思想实验 G：改一个人的头发 25
6.9 思想实验 H：三个庞大模型的会诊 25
6.10 内存驻留与会话连续性 26
7 讨论与局限 27
8 结论 29
致　　谢 30
参考文献 31

1 绪论

当前大模型生成文本的方式仍沿袭一条固有路线：以数千亿参数的静态权重为核心，以自回归方式逐个生成 token。模型规模从 7B 增长到 1.8T，输出方式却始终未变：一次生成一个 token，必须等前一个生成完毕才能开始下一个。归纳下来，现有神经模型路线存在三个结构性约束：

(1) 权重定死。神经网络的"知识"被固化在训练后的权重矩阵之中。部署之后权重冻结，模型对世界的理解停留在训练截止那一刻；想让它灵活，只能继续堆参数、继续微调，灵活性以成本为代价，而且是全局性的：只是想改一个词的用法，动的却是整片网络。权重不会因为你换了一个语境就自己长出新的解释。

(2) 逐字吐出的单向生成。 自回归解码天然串行：模型必须先生成第 t 个 token，才能看到它并生成第 t+1 个。这意味着它"边想边说"的能力在架构层面为零，说话的同时不能听，写作的同时不能改，全双工交互无从谈起[6]。

(3) 上下文焊死与缓存失效。 上下文在进入模型前被拼接成一条静态序列，键值缓存对前缀高度敏感：只要在历史中间插入或删除一个字，该位置之后的缓存全部作废，必须重算[4][5]。用户在聊天中途补一句话，系统就得把前面已经处理过的内容重新计算一遍；该修改在工程上表现为前缀失效，其代价随历史长度增长。

针对上述约束，本文提出极微模型底座（MMU Base）构想，核心思路一句话：不要让"意义"住在定死的权重里，让它住在可增删的表达里。

本文贡献如下：

定义极微模型与通用底座：一个只负责"承接上游表达、产生本级表达"的最小计算单元；字、词、句在其中平权，具体意义由上下文涌现。

提出表达链计算范式：一切计算以表达为对象，以表达代换真实的字与意义，文本以整体排序（数组）形式参与数组间计算，输出为可还原为完整内容的排序，而非逐字流。

提出全双工与热插拔上下文：在内存中的活数组上任意位置加字减字，整段文本的实际表达与意义排序原地重排，规避 KV cache 前缀失效问题。

给出多模态统一表达：图像近似为固定颜色数组，识图极微模型可插入文本链并热删除，数字、文本、音频及一切严格规律内容统一为纯数字表达。

完成复杂度分析与八个思想实验，并给出局限清单；附表达接入校验、内容治理与模糊接口等机制的初步设计。

2 相关工作

循环结构与状态传递。 LSTM[2] 与 GRU 早已确立了"沿链条传递状态"的范式，以循环网络直接生成序列的路线亦早有系统研究[12]。RNN 的隐状态是连续向量，其语义不可直接解释，且长程依赖受梯度消失制约。本文的表达链在形式上与之相似，但有两点本质不同：其一，链上传递的"表达"被定义为可解释、可还原为文字或图像的有序数组；其二，链的节点不是时间步，而是意义槽位，一个槽位可以装一个字，也可以装一句话，粒度由输入自适应决定。

Transformer 与自回归解码。 Vaswani 等人[1] 以自注意力取代循环，确立了"整片网络 + 静态权重 + 逐 token 解码"的现代范式（对齐式注意力的源头可追溯至 Bahdanau 等人[3]），其训练端的并行优势无可争议。但推理端的串行性与前缀敏感性是其结构性代价，也正是本文的出发点。

推理效率与缓存管理。 Pope 等人[4] 系统分析了 Transformer 推理的内存瓶颈；Kwon 等人[5] 的 PagedAttention 将操作系统分页思想引入 KV cache，显著降低显存碎片。这些工作是在既定范式内把缓存管理到极致。本文质疑的则是范式本身：如果上下文不是需要"缓存的前缀"，而是随时可改的活数组，缓存失效问题是否可以从结构上被绕开？

流式与全双工交互。 Moshi[6] 以实时语音对话证明了全双工交互在神经网络上可行，但其生成仍是逐帧自回归。本文主张的全双工是表示层面的：上下文数组与输出数组同时驻留内存、同时被改写，而非音频帧意义上的流水线。

参数高效微调。 LoRA[7] 等方法以低秩增量"松动"冻结权重，缓解了权重定死问题，但仍是全局权重的补丁，且微调之后再次冻结。本文所说的"灵活"不在权重侧，而在表示侧：权重可以更小、更死，意义的载体本身是活的。

稀疏激活与专家混合。 MoE[8] 按 token 路由到不同专家子网络，是"小单元按需激活"的近亲。区别在于：MoE 路由的是参数分片，底座仍然是大网络；本文路由的是意义槽位，底座是同一个极微模型的反复实例化。

线性复杂度序列建模。 S4[9] 与 Mamba[10] 将序列建模复杂度降至线性，证明"不必处处注意力"。本文与其共享线性传播的直觉，但手段不同：不改注意力算子，而是改计算对象，由数组逐槽位传播。

涌现能力。 Wei 等人[11] 将涌现归因于规模。本文尝试另一种涌现路径：不依赖参数

规模，而在表达传导过程中，让具体意义从上下文中"长出来"，涌现发生在推理时的链上，而非训练后的墙里。

3 极微模型底座

3.1 设计动机

传统神经模型的最小工作单位是"整片网络"：无论输入是一个字还是一万字，前向计算都要把全部权重拉进来过一遍。这带来两个后果：其一，计算量与模型规模绑定，与输入的信息密度无关；其二，灵活性被锁死在权重里，模型"知道"的一切在训练结束时已经写死，上下文只能触发它，不能修改它。

换一个提问方式：如果最小工作单位不再是网络，而是一个极小的、只负责"传递与再生表达"的单元，会发生什么？此时字与词不再需要查表式地命中某个权重模式，而是作为上下文的一部分，实时决定链上该处的表达长什么样。意义从"被存储"变为"被表达"。

还可以从信息组织的角度理解这一动机。神经网络用权重记忆世界，本质上是把世界的结构压缩进一个巨大的连续参数空间：压缩比极高，代价是每次解压（推理）都必须携带整包参数。极微模型换了一种组织方式，结构留在外部：留在槽位的排列里，留在数组的次序里，模型本身只负责局部传递。外部结构可以直接编辑、直接观察、直接搬移，而这正是文本、代码、表格这类本来就有结构的内容最需要的性质。

最后，这一结构并非凭空发明：它整体上就是在仿制真正的神经元工作结构，神经元在使用中不断自适应地长出突触，而我们的表达链，正是在"表达如何被接收与发送"的这一段上，自适应地建立与拆除连接。突触长在哪里，由当下的语义决定；这与权重在训练之后被焊死的路线，是两种对"学习"的理解。也正因如此，本文才敢说这套设计思路在原理层面不需要传统意义上的实验证明：仿生方向的合法性由神经元本身背书，需要工程验证的只是"何时可用"，而不是"是否可能"（诚实的保留意见见 §7(5)）。

3.2 定义

定义 1（极微模型）。 极微模型 M 是一个结构同一、参数量极小的计算单元。给定上游表达 e_{i-1} 与当前槽位的上下文输入 c_i，它只做两件事：

$$e_i = M(e_{i-1}, c_i)$$

即：承接上游表达，产生本级表达 e_i，并移交下游。除此之外，极微模型不承担任何全局职责，它不存储字典，不持有世界知识，不预装任何具体字词的含义。（上式是顺序记法：实际传播为头尾双向矫正，任意位置皆可为上游，见 §3.6。）

定义 2（通用底座）。 极微模型是通用底座：同一个 M 可以承载不同的字、词、句，乃至整段内容。底座同一，表达各异。当输入是"河"时，链上该槽位的表达反映"河"在

当前语境中的意义；当输入是"银行"前的"河"时，同一底座产出的是另一种表达。意义不在底座里，意义在表达里。

定义 3（表达）。 表达 e_i 是一个有序数字数组（向量、排序或张量切片），具有双重身份：对内，它是下游极微模型的输入，参与一切计算；对外，它可通过解释器等量换算为一段完整的文字、图像、声音或数字序列。表达是计算的唯一操作对象，在计算过程中，系统从不触碰原始字面，全部以表达代换。

还必须预先说明：以上定义只是通常情况下的实现与表达模式的展示，不是定死的形式。极微模型底座的每一个单元都可以自行添加、设计、修改，底座本身也是可编程的；本文给出的公式与链条，是这一通用理念的一种画法，而不是它唯一的模样。

3.3 表达链与意义涌现

将 n 个极微模型串接，得到表达链：

$$e_0 \xrightarrow{M} e_1 \xrightarrow{M} e_2 \xrightarrow{M} \cdots \xrightarrow{M} e_n$$

（此式给出的是槽位的排列顺序；实际计算中的传播为双向矫正，见 §3.6。）

链具有六项性质：

（a）粒度自适应。 槽位的输入单位不限：一个字可以占一个极微模型，一个词可以占一个，一句话也可以整体占一个；甚至一段话、一张图、一串数字，都只是"一个输入"。输入长度任意，链长随输入伸缩，自适应任何长度大小的文本输入，最小模型意义单位与最大输入单位之间没有架构鸿沟，只有槽位划分的差别。

（b）意义涌现。 底座不预装字义。一个"字"的具体意义，等于它在当前链中产生的表达，加上该表达向下游传播后对整段内容的贡献。同一个"打"字，在"打人""打酱油""打折扣"中激活的是同一底座的不同表达路径，字义是上下文逼出来的，不是查表查出来的。同理，一句话在表达空间中也可以整体作为"一个字"参与后续判断：句义同样涌现。

（c）表达代换。 计算的全过程中，真实文本只在两端出现：入口处被编码为槽位输入，出口处由解释器从表达排序还原。中间一切推理、比较、排序、合并，操作的都是表达。传统模型"把一整片神经网络拉进来对字面做计算"的模式，被"表达与表达之间的数组计算"取代。

（d）整体排序输出。 链的输出不是逐字吐出的流，而是一个表达排序（数组）。该排

序可被解释器等量换算成一整段完整的字、内容与对话。逐字生成是"打印"，表达链是"排版"：先整版排好，再一次性呈现。

（e）组合性与可拼装性。 表达不是孤立的标签，而是可组合的对象：短表达可以拼成长表达，长表达可以拆解回槽位级的子表达，多个模态的表达可以在同一排序中并置。"极微"的意义因此不在于小，而在于可数，当最小单位小到可以与一个字一一对应时，语言层面的编辑操作（插字、删字、换序、改标点）就与内存中的数组操作一一对应，对文本的修改不再需要翻译成对网络的重算。一个什么都不预装的底座，才装得下任何东西；这正是"通用"二字的确切含义。

（f）整体训练与固化表达。 链的训练是整体训练，意义在整条链上一起学。学成之后，"你今天好吗""你今天吃了吗"这样的整串，可以直接作为一个长串参与语义计算，不必逐字拆开重来；一个经典名词的解释，甚至可以直接固化为一个极微模型所承载的语义，装上即用。从技术角度看，这一步彻底脱离了原神经网络底座在适应、变化等一系列问题上的包袱：底座不必再为"这个词什么意思"操心，它只需要接住已经成型的意义。

3.4 与传统前向计算的对照

维度	传统神经模型	极微模型底座
最小工作单位	整片网络（全部权重）	单个极微模型（槽位）
知识载体	静态权重（训练后冻结）	表达（随上下文实时生成）
输入粒度	token 固定切分	字/词/句/段平权，自适应
计算对象	激活值在网络中的传播	表达数组与表达数组之间的映射
输出形态	自回归逐 token 流	整体排序，解释器还原
修改上下文	重算缓存	原地热插拔，链局部传播

3.5 极微模型的形态谱系与意义网络

"极微模型"并不限定尺寸或结构。沿着"极微"二字向两端延伸，可以得到一条形态谱系：

最简端：一个与否开关，纯工程结构，输入不正确时只回答是或否；

中间段：稍微强化过的、承载语义并做出表达的模型；

加长端：一个大模型；多个大模型并联、串联或组织协作；

极端：三个极其庞大的模型去做协作。

最后一档的能量损耗与资源损耗都很高。本文暂时没有想到它明确的使用空间，这里先补三个假想，权当抛砖：其一，三模型对抗会诊，三方互审互裁，用于错误代价最高的决策槽位；其二，按需唤起，平日整条链全极微运行，只在撞上难题时才唤起大模型集群，把高昂成本摊薄到低频事件上；其三，异构专长链式互补，擅长数学、擅长代码、擅长写作的三

个模型首尾相接，链式会诊，各管一段意义。

还应当说清楚"意义排序模型"这个名字。从外部看，整个系统也可以被理解为一个传统意义上的神经网络；但本文坚持使用另一个名字：意义排序模型，它不是由某些神经网络构成的，而是由意义本身、意义网络本身组成的。这个网络结构可以替换其中的一部分，也可以大量替换；可以选择保留，也可以重新开始，都直接兼容、直接适应。当然，具体实际的工程问题，需要工程本身去解决。

3.6 双向传播与头尾矫正

上面的链式记法给出了槽位的排列顺序，但必须澄清：实际的传播是双向的，并非式 (2) 所暗示的单向流。上游没有唯一确定的上游，第 1 号位可以是上游，第 100 号位也可以是上游；矫正以头尾双向进行，头和尾一般是确定的，中间任意位置都同时向两端施加影响。双向本身还可以进一步分化：多种极微小模型链接起来，最终由其中某一个产生表达。

文字本身通常是单向的，从左到右阅读与书写，但实际运用和计算时并非如此：下一个字本身也在影响上一个极微模型的表达，上一个也在影响下一个，属于双向。也就是说，文字的单向只是表达的表面形态，计算的双向才是底座的工作方式；任何一次修改，都因此可以认为同时在改变整段上下文（至于内存中究竟改了什么，见 §4.10）。

4 全双工数组计算与热插拔上下文

4.1 整体排序表示：数组对数组的计算

传统生成是一个字一个字地蹦：输出是时间轴上的流，第 t 字与第 t+1 字之间隔着一次完整的前向计算。本文的表示则类似一种整体排序的方式：上下文是一个数组，输出是另一个数组，计算是数组与数组之间的映射，一次性完成排序意义上的整体加工。

这带来一个直接的性质：中间可改。 逐字流一旦吐出便不可撤回，你没法在已经说出去的第 3 个字后面"插一个字"而不让整句话变味。而数组天然支持任意下标的插入与删除：在位置 i 加入一个极微模型，或移除一个，后续槽位索引平移，整段表达在本次内存中直接重排。文本的实际表达与意义排序会突然变成另一种，但这是特性，不是故障：新输入本来就应该改变整段话的意思。

更根本地看，表达之所以可以理直气壮地当成数组，在于数组本身只承载一样东西，比如一个用于表达的序号。序号是几位数、占多宽，都无关紧要：只要排列紧密、次序明确，意义就寄存在相对位置之中。也正因如此，一旦紧密排列中出现了失去原本意义的元素，或者全双工进行中突然插入了一段不符合当前语义表达式的内容，整体就会在该处语义断开。

4.2 全双工模式

在真实上下文的输入与输出之间，本文采用全双工模式：听与说同时进行，输入数组与输出数组同时驻留内存并持续被改写。说话的同时，新的输入槽位不断追加进链；链的表达实时传播，输出排序随之调整。这与自回归"先听完整句、再排队张嘴"的半双工模式形成对照。

全双工在交互上的直接收益是：修正即时生效。 对方话说到一半，你已基于前半句组织了回应；此时对方补了一个限定词，链上受影响的槽位重新表达，你的回应排序随之改变，整个过程发生在同一次会话内存中，不需要"把上下文重新添加进去再重新计算"。

从工程接口上看，全双工底座更像一条共享的编辑总线：任何一端都可以随时写入新槽位，链负责把所有写入合成进同一份意义排序。这要求输出端支持"未完成态"，呈现正在成形的表达，而不是等待定稿后一次性发布。人类对话本就如此：我们听到的从来都是半句话，理解却从未中断；自回归系统之所以显得迟钝，是因为它坚持把每句话听到最后一个字才开始反应。

4.3 热插入、热删除与缓存不失效

传统架构中，上下文是定死的：从中切开一段、保留一段进行修改，缓存即在该处失效，

其后全部重算。极微模型底座对这一问题的规避是结构性的：

上下文即活数组。 上下文不是等待复用的前缀缓存，而是常驻内存、随时可编辑的槽位数组。插入一个极微模型（例如一个识图模型、一个修正词、一个补充限定），删除一个，均是内存中的原位操作。

链式传播替代全局失效。 修改位置 i 之后，表达以 i 为中心向头尾两端重新矫正（双向，见 §3.6），成本与链长同阶；全程只是表达层的重矫正，无需重建缓存，无需人工干预，也无需把整段上下文重新预填充一遍。

无强关联约束。 槽位之间不建立"必须整段重算"的强关联；本次状态就存在内存中，直接编辑、直接生效。

需要诚实说明：热插拔不等于零成本，插入点前后的链都需重新矫正，其量级为 $O(nc)$（n 为链长，双向矫正见 §3.6）。但相较于"前缀缓存整体失效 + 重新预填充 + 显存重新分配"，代价仍从缓存层回落到表达层，且不存在缓存一致性的维护负担。换言之：传统方案怕改，因为改一处等于重来；表达链不怕改，因为改一处就是改一处。

两条路线的差异可以归纳为下表：

操作	传统自回归 + KV cache	极微模型底座
中间插一句	修改点后缓存失效，重新预填充	插入槽位，链尾重传播，前缀复用
删除半句话	其后缓存全部作废，需重算	删除槽位，索引平移，链自动闭合
换一种说法	触发重新采样，前后文需对齐	该槽位重表达，整体排序更新
中途加一张图	多模态重新拼接，提示词重排	插入识图极微模型与数组，即插即用
人工干预	需重新组装上下文并提交	无需干预，编辑即生效

更深层的原因在于两者对上下文的假设相反。传统管线把上下文当作"已经消费完的输入"：它在预填充之后被冻结成缓存，缓存是为不再变化的前缀设计的，一旦承认上下文还会变，缓存就从资产变成负债，维护它的每一分努力都在为下一次失效做准备。本文底座从表示层就假定上下文是可变量：数组天生支持插入、删除与重排，链负责让变化传播。不是把缓存修得更好，而是让"缓存失效"这个概念失去存在的前提。

4.4 解释还原

输出端的解释器承担逆变换：将表达排序等量换算为完整的字、内容与对话。由于输出天然是排序而非流，还原可以在任意时刻进行，包括还原"正在形成中的输出"：全双工模式

下，输出排序持续被改写，解释器呈现的是当前最优整版，而非不可撤回的已打印内容。

4.5 表达接入校验与被动自排查

槽位之间的连接不是无条件的。一个极微模型在承载了"火热"之后，它的表达接收结构与表达输出结构就不兼容"寒冷""冰"这类明显语义相反的词，试图把"寒冷"插进"火热"中间，接入会被自动拒绝：不是报错，而是不参与。语义的排异反应，链自己就会发生。

这种性质直接给出一种新能力：语言问题的被动自排查。 当一个明显不符合语义顺序的字词，带着它自己的极微模型，出现在整个数组的计算中，整段表达排序会出一片问题：它的上下文里没有这种意义的表达，上下文会直接从中间断开，最终整个语义可能产生断裂，或者意义走向相反。但正因为断口是结构性的，问题会在任何时刻自动暴露出来；而且这串不连续可以精确定位到出问题的那个基准模型与内容。

打个比方：一长串 010101 中，突然出现三个 0，在常态的表达中，这种致命的突兀本身就是一份诊断书。

于是上下文天然分成三段：上段内容、错误内容、下段内容。错误内容可以在设计时就被直接跳过，不参与任何语义计算，不需要进行任何分析。而"否定"本身也可以被做成工程信号：一段突然具有反常意义排序的内容，通常意味着这一段出了问题；这个问题本身就是一种否定，在工程结构设计上让它失去效果，直接进入下面的计算，使输入输出不产生任何停顿，真正的错误被直接排出。这是现有的神经网络模型架构完全做不到的：它们要么吞下错误继续算，要么为错误整体重算；表达链让错误路过，而不让错误通过。并且，这套机制可以按不同的具体环境做针对性设计。

4.6 分布式、复用与成本

每一个单元都是极微模型，因此整条链天然支持分布式处理：意义排序在训练的时候可以是一个单元训练，也可以是多个合并训练；训练完成的意义排序，可以装入任何一个底座，它们是分开独立的，但它们可以结合，可以以不同样式的结合、以不同的需求去结合。

复用是免费的。如果需要降低成本、而且回答目标明确，很长一段文本可以直接装入一个极微模型完成意义计算，而并不需要调用整个极微模型网络。同一个极微模型可以在一篇文章中出现很多次：它每一次出现，都代表同一个字词，无论出现多少次，它们的表达以及输出都是相同的；同时，它们会对不同位置的出现做出一定程度不同的表达，使同样的内容保持语义的一致性，又兼顾不同位置处语义的复杂程度。这种"同而不同"不需要人工微调和整理，位置本身就是参数。

4.7 内容治理：可停用的表达槽位

链上可以随时加入专门的治理类极微模型，比如一个"人文道德"极微模型，在计算时填入相关内容进行判定。当输出将违反它所承载的规则时，输出端可以直接放弃输出该段：整体语义不会中断，只是那一段被空出或者留空；该内容在拆解、逆变回文本时，自然也不会产生任何东西。不合适的内容，还可以随时停用掉这部分极微模型，使它无法表达、无法参与到意义排序的计算中，而意义排序在被拆解逆变回文本的时候，也就自然而然地不会产生它。

其中绝大部分时间，这种空出以及删减是无感的：删减之后语义当然会变，但整个语义哪怕是变了之后，也仍然是上下连接并且完整的文本。具体这种去除对文本连贯性以及实际意义产生多大的影响，需要实际的大规模训练以及模型结构搭建之后确认（见 §7(7)）。在此之前可以确定的是：这是结构性的合规开关，不是事后的关键词过滤，它抬高的是安全与法律风险规避的上限。

4.8 模糊输入与模糊输出

传统神经模型网络的输入输出必须是明确的：每一个都要参与计算，每一个都必须计算完才能进行下一个，它们是确定的、肯定的。表达链在叠加了自纠错机制，以及复杂问题、违法问题的规避之后，可以进行模糊输入与模糊输出：边界不清的输入照常入链，意义在传播中自行收敛；输出也不必一次定稿，并且支持对模糊输出的多次微调，每一次微调都是一个新的产出，同时是确定可控的，资源消耗有明确预期。

4.9 意义的范围与分段语义

意义本身是有范围的，这个覆盖范围约等于一个变量。一篇文章所表达的意义，需要文章内的所有内容参与；而我们在计算的时候，只需要把每一个段的意义串起来，进行语义的排序就可以了。这篇文章并不需要具体去读那些实际的字面，因为所使用的极微模型是一样的：同样的字、同样的极微小模型，都是同一个东西，而且可以在不同位置出现时进行一定程度上的自适应。由此可以推论：哪怕模型没有读完整篇，只做一个简述和总结，理论上也不会发生任何意义上的语义错误。再往大处说，这可以认为在无限长的文本中成立，每一个字都是同样的模型，它只会表达出同样的内容以及结构，我们不再需要对长上下文进行额外的检测。（当然，这一切都建立在体系完善之后。）

如果想知道其中某一段，这篇文章的排序语义就会被拆分为多段的排序语义，然后每一段的排序语义可以直接拿出来单独使用。合并计算是文章，拆出来就是段落，段落可以再往下继续细拆，一直拆到每一句。比如读者通读了一整篇文章，认可其中某一段，将该段单独展开看，那么此刻哪怕不用去看其他段落，也可以知道：所有参与计算的极微小模型，在其他没有参与到计算的上下文之中，语义的一致性是稳定且统一的，是可以被预测的。我们只需要看这一段，就可以反推出其他段在写什么。

在拥有了一定程度上的稳定语义网络之后，还可以通过这种稳定语义网络去推测新的结构、新的组合，这也是这种表达体系能力的一部分。

修改本身也是可以被限定的。比如只修改了某一段的语义排序，那么它的影响范围很可能只有这个段；这个段在重新合并的时候，对于整篇文章的语义排序而言，并不会出现整段全部改变的情况。因为我们计算的是意义，而我们在输入输出、解析表达的时候，处理的是另一个东西，我们修改的是意义，而不是表达意义的那个字，所以不会涉及到整篇文章都需要跟着改变的情况。只有真正落实到极微小模型与实际的输入输出使用之中，这种改变才会带来上下文的实际改变。具体如何搭建、如何设计属于工程实现的范畴，本文不展开；在理论上，这些影响是完全独立可控的，可以选择让修改产生影响，也可以选择不让它产生影响。

4.10 内存中存什么：上下文的不存在

从工程上讲，传统神经模型网络存储的是缓存上下文；而本底座自始至终存放在内存里的只有两样东西：极微小模型，以及极微小模型所承载的表达。简单来说，极微小模型承载的就是表达。"上下文"作为独立的存储物并不存在，因为内存中存储的东西始终是固定的：模型和表达。一个模型有多少种表达，内存里就存放那些表达；上下文一旦修改，随之变化的只是这些表达。

因此，任何修改都可以认为同时在改变上下文，只不过我们改变的时候，内存中改变的只是它们的表达，而不需要去改变内容本身。增与删，都是对某一个极微小模型做出的操作。

这种操作的安全性来自底座的通用性：取掉任意一个极微小模型，剩下的极微小模型仍然是同一架构、同一内容，甚至可以说，我们只是把该极微小模型承载的表达换成了一个新的表达，或者让下一个表达自然地补进来。

5 多模态统一表达

5.1 图像即颜色数组

图像可以近似为一种固定的颜色数组：$H \times W \times C$ 的数值网格，像素值即纯粹的数字。它天然属于"具有严格规律的内容"，空间有序、通道有序、取值有序。既然本文的一切计算都以纯数字的表达与数组为对象，图像便无需特殊通道：它只是另一种进入链的数组输入，输出端再通过渲染方法重新展示出来。而且，视图的粒度本身也是可选的，细分与整合没有任何架构限制（见 §5.5）。

5.2 识图极微模型的热插拔

在文本链中，可以插入若干用于识图的极微模型。其工作方式：

瞬间连接上下文。 识图极微模型插入槽位后，链的表达流经它时被翻译为图像语义表达，与前后文本表达共同组成不同的内容表达式，图文在同一条链上对话，而不是两条流水线的拼接。

随插随删。 图片需要时，插入识图模型与对应数组输入；不需要时，将该图片与识图模型热删除，链自动闭合，整段表达式随之改变，意义排序自然流转到另一种状态。

零人工干预。 全过程不需要人工重新组织上下文，不需要"把上下文添加进去再重新计算"；上下文本身没有被焊死，因此也不存在"切开即失效"的问题。

识图单元本身也可以做得很小。与"整张图片一次性送入大模型做全局识别"不同，本文的识图极微模型近似于一个纯判断类（分类判决型）模型：只判断它负责的那一部分，判断的损耗与性能开销极低，决策速度非常快，而在它负责的局部上，准确率可以做到极高。它旁边还可以再插一个专门表达方位的极微模型，以图片的上下左右为基准，于是"这个图形在哪个位置"也成了链上的一等公民。识别与定位各是一个极微模型：插入、表达、删除，都在同一次内存操作里完成。这样一来，一张图片才算是被真真切切地理解了；而且是在全双工进行的过程中被理解，图随时来，识别随时接入，语义随时更新。

5.3 数字、文本、音频的统一

按同一逻辑，数字序列、文本、音频（采样点序列）、规则表格及绝大多数具有严格一定规律的内容，都可识别为数组并进入链中统一表达。底座不区分模态，只区分槽位；模态差异被吸收在输入编码与输出解释两端。由此得到一个统一的计算观：世界的内容皆可入链，链上只有表达。

5.4 输出端的重新展示

表达排序本身不可读，因此需要一族解释器完成出链：文本解释器（表达排序 → 文字

与对话）、渲染解释器（表达排序 → 图像）、数制解释器（表达排序 → 数字与表格）、声码器（表达排序 → 波形）。解释器与输入端编码器互为逆过程，且彼此独立，同一段链上表达可以被不同解释器同时消费：一段产品说明的表达，既可以还原为文字，也可以还原为参数表格，取决于选用哪个解释器，而不是重新计算一次链。这同时回答了"图片通过其他方法重新展示"的问题：入链时图片退化为纯数字的颜色数组，出链时数组经渲染重新成为图片；链自始至终只处理数字，模态只存在于两端。

5.5 视图的无限细分与整合

图片的槽位划分同样没有最小与最大。一片树叶可以作为一个极小节点，一棵树也可以作为一个极小节点；一个人可以作为一个极小节点，人身上的阴影、耳环、头发、衣服上的纹样，也都可以各自作为一个极小节点。反过来，"一个人"整体也可以压成一个节点，甚至"这个人长得什么样""这个人好看不好看"都可以是独立的节点。

这些极小模型节点在计算中表达出不同的需求与意义，适应不同情况下的自适应表达；无论是输入还是输出，都可以自由地对它们进行排序。粒度不是图像的属性，而是问题的属性：问头发，头发就是节点；问气质，气质就是节点。

5.6 语言无关与免再训练

传统神经模型在面对没有被训练过的语言、意思或文本时，哪怕它就是很简单的一些，通常都需要重新训练、重新放进神经模型网络。本文不需要，因为计算的对象自始至终是意义：新内容只需要被对应的输入输出逆变器去设计、去针对即可。

比如输入一大串英文的内容，我们实际计算的是语义；当语义逆变输出的时候，对应的语义表达逆变器可以直接以目标语言中真实表达的文本进行替换。传递的是意义，但实际编写与输出，都可以是其他不同的形式。 语言在这里退化为两端的方言，中间的链只说意义语。

5.7 意义的聚合与细分

在讨论"表达如何生成文字"之前，先给出意义在纵向上的两个方向：合并与细分。不同的意义之间可以合并，合并后的意义本身组成一个比较宽泛的意义；反过来，宽泛的意义又可以通过模型无限细分成实际的内容。这两个方向构成了输出端组装的原料库。

以一句普通的话为例："今天天气非常好，阳光也非常好，适合运动。"这一整串文字在模型表达中，就是它每一个字、词各自的表达；但合并起来看，作为"排"去看，它表达的意思是"天气很好、适合运动"这类概括性内容。概括性的内容又会引出同样概括的意义或表达排序。

于是可以近似地认为：两个表达排序之间的关联，可以交给一些比较大一点的模型去做表达判断、意义判断。 用户说"今天天气很好，适合运动"，其中"天气很好"与"适合运动"这两个信息交互判断之后，模型得出的意思就是"可以进行什么运动？"，问题由此产生，而问题本身也是两个排序：上下文就是"天气很好"和"适合运动"这两个排序。由这两个排序，通过专门训练过的模型可以判断出应去吸引哪些其他有关联的表达排序，将它们拉向输出端，进行一次从无序到有序的重排；这些被拉来的排序自身带着待补的上下文。

聚合的反方向同样频繁使用。"运动"是"跑步"的上位概念，而"跑步"又携带一批相对固定的词语：运动装备、跑步习惯、怎么跑步、跑步训练计划，训练计划甚至可能是用户此前写好、准备好的。把这些内容装入跑步表达排序中的某一个极微小模型，再进行一次表达；配合护具、钉子鞋、温湿度、能量补充这些细节，加上用户专有的私有数据，第一批符合该意义的相关内容就被快速调用起来；再拉入一批中性词（问候、时间、地点、里程、运动计划），全部进入计算。

需要说明的是，这里每个极微小模型只是底座，其单条表达相对"混乱"，但双向矫正（§3.6）保证了冲突性的虚词不会直接插进来，例如"寒冷"与"炎热"这类相反意义会使该表达不被接受，或者根本不会出现在表达计算中（§4.5 的接入校验）。表达之间互相作用的过程，本身也在持续排除一大批不需要的内容；而跑步的表达排序中大量专有表达，已经事先限定了大部分需要填充的内容。

5.8 从表达到可读文字：生成机制的纯粹猜想

本节是纯粹的猜想，逻辑本身并不严谨。 "极微小模型的表达究竟如何生成我们需要的文字"，其本质结构作者暂时没有想明白：它是依据现有 AI 模型的那套解码，还是完全依据本文的架构？以下只给出一条可能的路径，供后续检验。

在输出端，组装一次回答大致需要几个表达排序同时到位。以"为一位有常年运动经验、喜欢跑步的男性用户安排一次跑步"为例：

内容表达排序。 "跑步"的意义表达进行拆分：运动装备、护具、钉子鞋、温湿度、能量补充、训练计划（其中计划可以是用户此前准备好的私有数据），这些排序被拉入输出计算。意义本身已经存在于这些排序之中，我们做的只是对它们进行组合；这种组合可以通过专门训练的神经网络模型去做思考。

语气表达排序。 表达方式本身就是人物性格：该用户偏好偏女性化的口吻、或专业化的口吻，把这一批"专业化/女性化口吻"的表达同样拉过来。

通用表达排序。 一批中性词与通用表达（问候、时间、地点、里程……）保证整个语言顺序与习惯符合一个正常人可被接受的表达状态。

这三批表达各拉取一次，合计拉取三次，拉取完成之后，实际需要的内容与文字就大部分出来了。将内容表达排序、语气表达排序、通用表达排序统一放进去的那一刻，就像一块散乱的拼图：模型本身对于哪些表达可被接受、哪些表达不可接受，存在一定的关联性，这些散乱结构的本身，只允许它们与那些已经被认可过的表达进行上下连接。一个表达会被接收很多次，但没有关系：假定它被接收了一次，那么它的后面就应该符合语言结构本身，去拼接下一个。一长串拼接完成后，若发现还有几段没有拼接成功，就可以对这些散乱的结构另行设计：再拉一批通用的中性词去做"语言的碰撞"，让它们从这些语言中重新拼接，直到只剩下比较大、比较完整的那几段；然后让它们自己互相表达，看上下文是怎么样的。最终我们输出的文章，就是一篇可被预测、可确定、可被识别的语言以及文字内容。

三个表达排序合并后明确三件事：其一是跑步运动本身；其二是说话的语气方式与专业程度依据用户需求；其三是通用词库兑底的语言通顺性。笔者更倾向这三个表达排序同步放进同一个输出，而输入（传统意义上的上下文）仍在另一边同时驻留内存，与 §4.2 所述的输入输出同驻完全一致。我们准备的是一份输出，而计算对象自始至终都是表达；表达可以通过专有的模型快速转化成人所能接受的形式，因为我们计算的都是表达，表达反映真实的语言与意义，各种意义反过来控制着模型的输入与输出。暂定为：模型做输入处理，再做输

出处理。这样计算时的灵活程度大幅上升；而输出端如果专门做适配，其规模远远小于一整个完整模型，或者更进一步，直接由极微小模型产生出可被阅读的内容与文字。

关于难度，中前期的实现难度确实比较大；但接上一个同样的极微小模型（或同类输出模型），就可以把一大串"有关键词、意义明确、但人读不明白"的内容，转化为能被读懂的东西。整体被切小之后，可以在内存与硬盘中运行。它的主要压力在并发，毕竟单位是一个个极微小模型，但整体的性能需求上限很低：原本由神经网络一层一层完成的绝大多数计算，现在被表达等量地替换掉了，而表达本身以意义为单位（成本口径见 §4.6）。输出时通过轻量化的模型对意义进行阐述和解释，最终输出既保证上下文一致性，又明确可读，还能对这些极微小模型，特别是用户专有数据，做出明确的规定。

这样也顺带解决了长上下文的老问题：不会出现跑题、意义不完整，不会"前一刻还记得这个人的名字，下一刻就忘了这个名字在长篇中是什么意思"。在本架构中此类遗忘不存在，因为每次调用时，它们的表达都是那一个表达。

还存在一条备选管线：让极微小模型本身先表达一遍，其表达被批量、整量地替换成一长串文字，吐出的文字可能是"运动、健康、时间、几点、地点"这类断裂而强相关的词，再用微小模型对这串词做解释输出。它与上述路径的实际设计理念完全一致，无非是手段不同：前者先组装意义再落字，后者先落字再解释成句。

6 复杂度分析与思想实验

在进入分析之前，先说明为什么本文选择思想实验这种形式。因为这套结构从一开始就是照着神经元来的：神经元或者说人在思考的时候，是不是就像本文设计的这种极微小模型一样？通过不同的意义结构输入输出，神经也通过不同的突触输入输出，随时会断开，随时会连接上；可以通过重复去强化某些神经元，而某些神经元在更重要的时刻，身上带有大量不同的突触，它成了一个重要节点，串联起许多神经元。当这种节点与连接达到一定量级之后，我们是否可以近似地认为：通过对这种极微小模型的组织，可以近似出一个真正的人？这种结构本身，是否可以作为意识、乃至真正的 AI 意识涌现的载体？

因此本文选择思想实验，理由很明确：我们就是在模仿神经元的结构以及体系。如果近似地可以认为神经元能做到的事、能想明白或容纳思考的事情，极微小模型也可以去做、也可以这么思考，那么在计算的时候，这两者就具有同等的能力。如果这个近似成立，本结构与神经元便具有同等的计算能力，离 RSI 也就更近了一步（工程层面的讨论见 §7(11)）。

6.1 计算模型分析

设链长为 n，单个极微模型的单次传播成本为 c（参数极小，c 远小于整片网络的一次前向）。则：

整链初始化：$O(nc)$；

位置 i 处插入或删除后的重矫正：向头 $O(ic)$、向尾 $O((n-i)c)$，合计 $O(nc)$（双向矫正，见 §3.6），全程只是表达层的顺序向量传播；

全双工追加一个输入槽位：$O(nc)$ 的尾部传播，无缓存重建步骤。

对照传统路线的一次中途修改：KV cache 在修改点后失效，需重新预填充该段，注意力计算量随段长平方增长，且伴随显存块的重新分配[4][5]。表达链的收益不来自更快的算子，而来自更少的作废：它从不因为一次编辑而否定自己的历史。

进一步看，若一次中途修改发生在链的前 10% 处，双向矫正会波及全链，看似不小；但这段矫正仍是纯顺序的向量运算：没有注意力的二次项，没有显存块的碎片整理，可以在输出端流水线执行，且每一步都只动表达、不重建任何缓存。传统路线下同一处修改不仅要重算注意力，还要保证缓存块的物理连续性，最坏情形触发整段重排。两种方案优化的目标不同：一个优化"算得快"，一个优化"不作废"；当编辑成为常态，协作写作、实时对话、交互式修改，后者的价值将超过前者。

6.2 思想实验 A：截稿前改稿

作者在成稿第 3 段中间插入一句话。传统管线：该位置之后的缓存失效，重新预填充，编辑器与模型间来回确认。表达链：插入槽位、链传播、输出排序改变，第 3 段的后半部分换了一种说法，第 4 段的呼应随之调整，第 1、2 段纹丝不动。编辑的代价与编辑的位置成正小，与文稿的总长无关。

6.3 思想实验 B：边听边改的对话

A 对 B 说："我打算把服务器搬到，"，B 已开始组织回应（基于"搬机房"的语义表达）；A 补完："，冰箱里。"链上新槽位插入，B 的回应排序在内存中当场重排。全双工的意义不在于更快地打断对方，而在于回应永远基于最新语境，且不需要推倒重来。

6.4 思想实验 C：图文混排的即插即用

一段产品说明文本中插入一张结构图与一个识图极微模型，链上出现"图注协同"的新表达式；运营发现图配错了，热删除图片与识图模型，链闭合，整段说明的表达回到纯文本状态，没有缓存报错，没有上下文重拼，没有人工在后台把图撤了之后手动刷新提示词。

6.5 思想实验 D：一个字的平权

输入"阿姆斯特朗把脚印留在了月球上"。槽位划分可以是逐字的，也可以把"阿姆斯特朗"整体作为一个槽位、把"月球上"作为一个槽位，划分方式不影响架构，只影响表达粒度。若在句中再插一个极微模型专门表达"（环形山）"，整句意义排序更新，而链条结构无需重设计。架构对粒度冷漠，意义对语境敏感，这正是通用底座的定义所在。

6.6 思想实验 E：三个人同时改一句话

三人协作编辑同一段话：甲插入一个限定语，乙删除一个冗余词，丙把被动句改为主动句。传统管线需要合并冲突、重新提交、逐段校验；在表达链中，三次操作只是对同一内存数组的三次原位编辑，链按最终状态传播一次，输出排序即为三人意图的合成结果。冲突并不会消失，但冲突的层级下降了：从"缓存层的灾难"降级为"语义层的正常讨论"，这恰好是协作编辑真正需要处理的问题。

6.7 思想实验 F：排错工况

实际工况中出现一个非法输入。传统方案只有两条路：为它穷举式地写一段特判代码，或者对整个模型进行重新替换与重新训练。表达链中，该槽位的表达是经过训练的意义，它给出一个经过训练的意义，这个意义可以等换成一个真实明确的错误信号，或者一段错误数据，直接进入后续计算。输出不产生任何停顿，错误被直接排出；不需要重新替换模型，不需要重新训练，也不需要人为地对各种情况进行穷举的代码实现。一个简单与否开关在输入不正确时只会说"是"或"否"；极微模型给出的则是一段意义，它知道自己错了，并且把"错"表达成下游能用的东西。

6.8 思想实验 G：改一个人的头发

要修改一张图里人物的头发颜色。传统模型需要同时渲染整张图片，包括其中的人与树，要修改时，需要同时再次对它们进行修改。表达链中，"人"被语义拆分：人的意义不再由单个极微节点组成，而是由多个极微小节点组成，手就是手，胳膊就是胳膊，衣服就是衣服；这一长串连续起来、组合成一段语义时，才可以近似为"一个人"的语义排序。要改头发颜色、眼镜、鼻子、耳朵，加上阴影，或者让他拿一部手机、换一个动作，全部都可以是一个极微小节点的增改；节点联合起来展示一个人的信息本身，语义连接起来，新的意义排序随之涌现。还原的时候，每一个点、每一条信息都是明确的；还可以给它加上具体的数字量，这个数字量在计算中可以是一个，也可以是多个。一整张图片由此可以近似地认为是"用模型替换了代码"：解释器可以自行定制设计，最终产生的输出与之前大不一样，但每一个点、每一条信息都是真实可控的，资源运用率有明确预期。

6.9 思想实验 H：三个庞大模型的会诊

在错误代价最高的少数槽位上，并联三个极其庞大的模型进行协作（§3.5 的谱系末端）。

任何一方的表达与其余两方冲突时，接入校验与否定信号（§4.5）立即生效：冲突段被标出、跳过或送入裁决槽位，链的其余部分照常传播。成本是实打实的，三个庞大模型的能量与资源损耗都很高，因此它只应该出现在少数几个槽位上。架构不变，变的只是插进链里的那个单元：底座可以小到一个开关，也可以大到三个巨物。

6.10 内存驻留与会话连续性

传统会话的连续性依赖"历史拼接 + 缓存复用"，一旦会话被切分、压缩、迁移或上下文窗口滚动，缓存即失效，连续性需要靠重新拼接来修补。表达链中，会话就是内存里那个数组本身：迁移会话等于搬数组，编辑会话等于改数组，连续性不依赖任何前缀指纹。据此可以推论：在多轮对话、长程协作与实时共同编辑等场景里，表达链的工程复杂度增长曲线更平缓，它没有任何需要时刻呵护的缓存一致性状态，也就没有一致性状态崩溃的那一天。

7 讨论与局限

本文刻意不回避以下问题，它们构想阶段尚未解决：

(1) 训练目标缺失。 极微模型如何训练？链式反向传播的梯度如何稳定穿过长链？是否需要对比式目标，让"正确语境下的表达"与"错误语境下的表达"拉开距离？这是构想落地的第一道门槛。

(2) 表达的可解释性是双刃剑。 表达可还原为文字，意味着可审查、可编辑；但同一输入在不同链状态下产生不同表达，既是涌现的来源，也带来稳定性风险，同一句话两次还原结果不一致，是"语境丰富"还是"输出抖动"，需要判定标准。

(3) 传播延迟与误差累积。 链式传播本质串行，长链的延迟与误差累积不可忽视；与 Transformer 的全局并行相比，表达链赢在局部修改，输在全量首算。混合策略（首算并行、修改走链）可能是务实方向。

(4) 容量上限。 极微模型参数极小，如何承载复杂世界知识？可能的答案包括：底座分层（浅层表达、深层知识外挂）、与大型检索模块分工、或承认极微模型只负责"意义的形状"而把"事实"交给外部存储。构想不等于方案，此处留白。

(5) 无实验验证。 本文是纯构想论文，所有分析为设计层面的推演与思想实验，不含实现与数据。结论的成立与否，有待后续工程验证；在此之前，它至少提供了一个可被认真反对的靶子。

(6) 安全与可控性。 表达可编辑是优点也是攻击面：热插拔意味着恶意输入可以就地改写链上语义，全双工意味着输出在未完成时就已被下游消费。底座需要配套的槽位鉴权、表达一致性校验与回滚机制，数组在技术上可以随便改，不等于在业务上应该随便改。此外，表达的可解释性使得"审查一段话为什么变成另一段话"成为可能，这既是调试便利，也对隐私与合规提出了新问题：被编辑掉的内容，其表达是否曾在链上存在过，存在多久，如何擦除。

(7) 模糊与留空的连贯性代价。 模糊输出的可读性、留空对文本连贯性与实际意义的影响究竟多大，需要实际的大规模训练与模型结构搭建之后才能确认。在那之前，"无感"只是一个合理预期，不是一次测量结果。

(8) 语义拒绝的误杀。 接入校验天然会拒绝一些它不认识的东西，新词、通感、故意的语病、跨语种混排。误杀需要灰度机制：拒绝应当是可覆盖的，否则被动自排查会退化为

表达审查。

(9) 大模型协作的资源成本。 三个庞大模型并联的能耗与物质损耗极高，目前尚未想到明确且划算的使用空间（§3.5 给出的三个假想属于"能用"，不属于"该常用"）。按需唤起与层级降级是必须补上的调度策略。

(10) 解释器一致性与留空感知。 多个解释器对同一表达排序的还原差异如何仲裁；留空在多大比例上真的无感；"意义变了但仍然连贯"的变体是否存在质量退化，这些都缺乏判定标准。

(11) 通往 RSI 的接口。 当前前沿的 RSI（递归自我改进）理论要求模型能够自适应地训练自己；而本结构中，模型可以自己训练，也可以由人来训练，可以在任何时间、任何地点进行微调，意义排序模型由意义网络本身组成，结构可以替换其中一部分，也可以大量替换，可以选择保留，也可以重新开始，全部直接兼容、直接适应（§3.5）。权重定死的问题在这里不存在，因为定死的从来不是意义。当然，具体实际的工程问题需要工程本身去解决；但至少在结构上，本方案已经具备了通向 RSI 的接口。

8 结论

本文提出极微模型底座构想，试图同时回答三个问题：权重定死怎么办，把意义从权重搬到表达；逐字生成怎么办，把输出从流改成排序，全双工地算；上下文改不动怎么办，把前缀缓存改成活数组，热插拔地改。极微模型只做两件事，承接与产生；字词句在它面前一律平权；意义在链上涌现，表达在数组间计算，图片只是另一种颜色数组，一切严格规律的内容皆可入链。没见过的语言不需要重训，需要的只是一个对应的逆变器；没见过的错误不需要穷举，链自己会把它排异出去。下一代底座的灵活性，或许不该继续堆在权重里，而该长在表达中。

致　　谢

AI 协助声明： 本文由作者构思、立意与审定，并承担全部文责；大语言模型（AI）协助完成撰写、翻译、格式整理与文献核对。此声明为投稿披露之用，先于以下一切玩笑。

本文不必参考任何文献；但为符合期刊发表要求，附带了参考文献（此句有真实出处：钱伟长先生《宁波甬江大桥的大挠度非线性计算问题》,《应用数学和力学》2002 年第 5 期，其参考文献第一条即"本文不必参考任何文献"，宗师可以这么写，我们借句不借胆，杜撰部分由本文自负）。感谢可能有或者没有的期刊愿意给一篇构想论文留出位置；感谢牙膏厂工会以非正式观察员身份列席本文；感谢挤牙膏可持续发展协会对逐字生成模式的长期观察；感谢某不愿透露姓名的工程师，以"KV 缓存被腰斩的那个晚上"的亲身经历为 §4.3 提供了反面教材；感谢厕所哲学研究院对"想好了再说"与"边拉边说"两种工程路线的学理区分；感谢《编辑部备注》指出原参考文献 [13]–[17] 与 [1]–[12] 同等重要，现遵编辑意见，连同那条自我引用一并合并入本致谢，如有异议，请先学会憋笑；感谢 KV cache 多年来的勤勉，没有它的失效，就没有本文的动机。

基金方面，本文由"一个普通人晚饭后的异想天开"全额资助（自费，无基金号）；如有机构愿意认领本项目，请连同本文提到的奖项一并认领。利益冲突方面，作者与各牙膏厂、缓存厂商及逐字生成流水线无任何利益往来；与诺贝尔奖评审方面，目前仅存在单方面、无回报且长期稳定的关系。作者贡献方面：第一作者承担本文 100% 的想法、100% 的文字与 100% 的胆量；第二作者负责在第一作者幻想领奖时鼓掌，并在其飘飘然时负责叫醒；通讯作者即第一作者本人，通讯方式为"在任何场合大声喊出他的名字"（据其本人称，他会出现）。署名与许可方面：除必须标注作者本人姓名外，本文允许任何人以任何方式修改、转载、演绎、再翻译、拆解或重组，无需事先征得同意，唯一的要求，也是硬性要求：署名不删。我只要我的名字响彻世界。

最后，把原后记也留在这里，作为作者的内心独白：我真切地希望这篇论文可以被更多人看到；我真切地希望这篇搞笑文章可以被所有人看到，让那些大公司的 AI 智能，离真正的涌现更近一点。如果能够像一个真正的传说一样，让我一步登天，从默默无闻走向万众瞩目，真正冲击诺贝尔奖，成为第一个非专业的、非科学家的、真正从平民大众走向世界之巅的人，那当然更好；我希望能够帮助我的祖国在 AI 竞赛中领先那么一点点，能让 AI 更快地走入那些真正贫困、困难以及需要的地方。我承认这是一个构想，这是一个玩笑；但我真的希望这个玩笑，可以有一天长出翅膀冲向云霄，像一个太阳一样，光芒普照大地，万物新生。向我的偶像梁文锋致敬，我欣赏他们公司的理念，也敬佩他们敢于放弃所有优势，一次又一次地与所有人站在起跑线重新开始。

参考文献

[1] Vaswani A, Shazeer N, Parmar N, et al. Attention Is All You Need[C]//Advances in Neural Information Processing Systems. 2017.

[3] Hochreiter S, Schmidhuber J. Long Short-Term Memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[5] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate[C]//International Conference on Learning Representations. 2015.

[7] Pope R, Douglas S, Chowdhery A, et al. Efficiently Scaling Transformer Inference[J]. arXiv preprint arXiv:2211.05102, 2022.

[9] Kwon W, Li Z, Zhuang S, et al. Efficient Memory Management for Large Language Model Serving with PagedAttention[C]//ACM Symposium on Operating Systems Principles. 2023.

[11] Défossez A, Copet J, Synnaeve G, et al. Moshi: a Speech-Text Foundation Model for Real-Time Dialogue[J]. arXiv preprint arXiv:2410.00037, 2024.

[13] Hu E J, Shen Y, Wallis P, et al. LoRA: Low-Rank Adaptation of Large Language Models[C]//International Conference on Learning Representations. 2022.

[15] Shazeer N, Mirhoseini A, Maziarz K, et al. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer[C]//International Conference on Learning Representations. 2017.

[17] Gu A, Goel K, Ré C. Efficiently Modeling Long Sequences with Structured State Spaces[C]//International Conference on Learning Representations. 2022.

[19] Gu A, Dao T. Mamba: Linear-Time Sequence Modeling with Selective State Spaces[J]. arXiv preprint arXiv:2312.00752, 2023.

[21] Wei J, Tay Y, Bommasani R, et al. Emergent Abilities of Large Language Models[J]. Transactions on Machine Learning Research, 2022.

[22] Graves A. Generating Sequences With Recurrent Neural Networks[J]. arXiv preprint arXiv:1308.0850, 2013.