NEURAL IMPRINT · WHITEPAPER V2

NI 下一代：持续学习的愿景、哲学与技术路径

白皮书 v2 · 质子梯度（AtomGradient） · 2026 年 9 月 17 日

文稿日期 2026-09-17 版本 v2 语言 中文 · English

这份白皮书阐述质子梯度对持续学习的研究立场、已经建立的算法与工程基础，以及下一代 Neural Imprint 的演进方向。

我们把持续学习视为一个贯穿算法、模型与设备运行的实际问题。对智能的理解决定研究方向，算法设计给出学习机制，工程与实验让这些机制接受现实检验。

质子梯度（北京）科技有限公司 · AtomGradient
在线版本与最新更正：https://atomgradient.github.io/whitepapers/neural-imprint/

目录

1. NI 的目标始终是持续学习

Neural Imprint（NI）是质子梯度研发的持续学习算法与算法架构。围绕 NI，我们配套研发了覆盖模型优化、推理运行时、学习状态管理、工具接入、开发接口与跨设备协作的整套技术栈。我们要让运行在用户设备上的模型，能够从新的记录、交互和行动反馈中积累认识，把学到的变化用于下一次任务。

第一代已经在手机上实现了这样的基础：模型先理解用户特征与工具说明，保存处理这些信息时形成的内部计算状态，后续会话直接带着这些状态工作。随着新记录与用户校正积累，现有流程已经支持增量学习，更新并启用新的学习状态。下一代将在这一基础上，研究更细粒度的更新、更长期的能力保持，以及不同设备之间的学习延续。

对使用者而言，模型可以在后续会话中沿用已形成的用户特征和工具用法，减少反复说明的负担；遇到需要精确记录的问题，则可以调用已接入的工具查询。

NI 从第一代起，目标就是持续学习。我们沿着同一个目标，把已经建立的机制推进为更完整的学习循环。

记忆、表示、状态承载、工具使用和推理调整，共同服务于这个学习循环。我们关心的核心问题是：模型是否因为经历过，而更会面对下一次；新的证据出现后，又是否能够继续修正自己。

1.1 让部署后的智能体持续成长

基础模型已经让智能体具备了很强的语言理解、感知与任务处理能力。但在一次任务中表现良好，与在长期使用中不断成长，仍是两件不同的事。

模型今天可以理解一条新信息，明天却可能仍需要我们重新说明；今天接受了纠正，下次面对类似情境时却未必改变。这种理解与积累之间的断点，正是持续学习要解决的问题。

测试时持续学习（Test-Time Continual Learning，TTCL）关注的，正是智能体在部署后持续获取、巩固和完善能力。我们的立场是：成长应当贯穿模型的使用周期，而不止发生在交付之前。这里的“测试时”主要指实际使用阶段。

我们的目标，是让这种成长发生在用户的设备和环境中。模型从观察、交互、行动后果与纠正中形成新认识，保留仍然有用的能力，并把学到的变化用于尚未经历的情境。日常更新应当建立在已有学习成果之上，以可承受的增量代价持续发生。

这要求我们把理解、预测、行动和长期可靠性作为一个整体来研究。持续学习的价值，在于经历能持续转化为面对未来的能力。

1.2 临时适应与持续成长，差别在哪里?

我们对当前大模型最不满意的地方，是一次交互中形成的理解，常常难以积累为以后仍然拥有的能力。上下文学习表明，模型能够利用当前提供的新示例、新定义来适应任务；持续学习要让这种适应进一步延续。

真正的区别在于：这一次理解了，下一次是否还需要从头提醒？今天得到的纠正，能否在相关情境中继续起作用？如果情况改变，模型能否知道旧认识已经不再适用?

我们用“学习连续性”描述这一目标：把当前任务中的适应积累为跨任务可使用、可检验、也可修正的变化。

1.3 贯穿全文的研究问题

> NI 怎样以可持续的增量机制，让智能体把不断到来的经历组织、压缩和巩固为可使用、可修正、可迁移的能力，并在有限资源与异构环境中保持学习连续性?

我们从学习目标出发选择载体。学到的变化可以由模型参数、内部状态或二者共同承载；不同时间尺度也可以采用不同更新方式。判断一种机制的价值，要看变化如何发生、为什么有用，以及怎样延续。

工具提供观察、行动和权威事实；NI 进一步关注，模型怎样从使用工具的经历中形成可复用的认识与策略。

1.4 智能走向数据

我们曾经提出一个问题：到底是智能走向数据，还是数据走向智能?

质子梯度的回答是：智能走向数据。

这决定了我们在哪里发展学习能力：让模型进入数据产生的设备和环境，在个人经历发生的地方完成理解、回应与积累。

与此相连，我们始终秉承一个研究立场：**设备就是智能体。** 我们以设备本身的计算能力为基础，把模型、感知入口、可执行的动作与持续保留的学习状态组织在一起，让观察、理解、行动与学习发生在设备所处的环境中。

设备由此成为经验发生、能力运行和学习成果积累的主体。本地计算让设备能够处理自己的观察、调用自己的工具，并依据新的反馈更新认识。持续学习也因此有了具体的承载者：一台与使用者长期相处、在使用中继续成长的设备。

智能走向数据，也决定了模型与人的关系。真实生活不是一次提交完毕的数据集。习惯会变化，环境会变化，工具会变化，一个人对同一件事的理解也会变化。如果智能能够在这些变化发生的地方持续学习，它才有机会参与一段长期关系，而不仅是处理一连串彼此隔离的请求。

这还意味着控制权。哪些经历被学习，哪些认识应当修正，哪些状态可以保存、删除或迁移，应当由使用者掌握。学习状态与原始数据一样，需要受到明确的隐私保护。

同样的原则贯穿跨设备学习。我们希望学习能在不同设备、算力和感知条件下延续，原始数据始终保留在使用者的设备上；学习状态的流动遵循明确的兼容性与授权边界。

选择智能走向数据，就要从研究之初面对真实设备的内存、功耗、带宽和运行条件。这些约束直接参与学习机制的设计。

2. “贴纸片”的比喻，怎样变成一个学习问题?

2.1 纸片是经历的入口

我们曾经用一个多维空间来想象这件事：Agent 处在其中，每一面墙上都可以贴纸片。不断到来的记录就是纸片；同一件事可能与不同的墙有关，多张纸片也可能共享同一处表示。以后面对一个问题，Agent 能找到相关内容，把它重新表达出来。

这里的“纸片”对应一次观察或一段记录，“墙”对应模型组织信息的内部空间。“贴”是让新经历改变内部状态，“找”是用当前问题唤起相关信息；“叠贴”则是在共同结构中容纳不同经历。

我们选择纸片这个比喻，是从信息的可呈现形式出发。文字可以是一条记录，图像可以是一张画面，视频可以按帧观察，音频可以转录成文本。它把连续到来的信息变成了可以处理的单元。

纸片的呈现是二维的，它承载的关系却可以是多维的。一条记录可能同时关联时间、对象、目标、情境和结果。视频按帧观察时，要保留动作次序；音频转为文字时，也要考虑语气、停顿等信息的损失。学习的对象是相互联系的经历，而不只是孤立的数据单元。

“墙”可以启发我们思考表示空间：同一条经历能否沿着不同的关系被理解和调用，而不必只能放进某个固定抽屉？我们更关心模型能否学会这种组织方式，而不是开发者事先为每一种情况写好分类规则。

2.2 “穿透式叠贴”意味着什么?

这个设想最有意思的地方，是多条经历不必各占一个完全独立的格子。它们可能共享某些结构，同时保留各自重要的条件。

共享表示带来压缩的可能，也带来相互干扰的风险。两次表面相似的经历，可能因为一个关键条件不同而需要相反的行动；两次表面不同的经历，又可能体现同一条规律。怎样保留这种区别，比单纯增加存储空间更接近学习问题。

因此，“叠贴”的关键是有选择地共享：写入时识别可以复用的结构，保留决定结果的条件；读取时理解当前线索指向哪一组关系。

把内部表示重新表达为语言，是让已保留的信息重新可用。模型还应清楚区分回忆、推断与未知：哪些来自具体经历，哪些是根据规律作出的判断，哪些仍然缺少依据。

2.3 记住经历、理解规律与形成能力

这里包含三个相互联系的层次：**记住经历**，是知道某次发生了什么；**理解规律**，是知道某些条件与后果之间可能有什么联系；**形成能力**，则是在新情境中利用这些认识，选择和完成合适的行动。

要把这条路径讲清楚，需要区分情景记忆、语义记忆，以及我们日常所说的“事实记忆”。Tulving 对情景记忆与语义记忆的研究，帮助我们区分对具体经历的回忆与对知识的掌握。本文借用这一功能区分来设计模型的学习任务，关注能否保留和使用经历中的信息。

概念	它回答什么问题?	在智能体中的例子
情景记忆	曾经在什么情境下经历了什么?	记得某次尝试的目标、当时条件、采取的行动及其后果，并能在相关线索下回忆这段经历
语义记忆	已经知道什么，可以怎样理解它?	掌握工具的含义、某类环境的规则或已经形成的个人偏好认识，使用时无需重现最初获得它的那次经历
事实记忆	保留了哪些事实性内容?	一次事件发生的时间、某个对象的属性或一个已经学会的定义；事实内容可以出现在情景记忆中，也可以成为语义知识

这里并列解释三个常用词，但它们不是三个互斥的抽屉。情景与语义区分信息怎样被组织和使用；“事实”描述保留的内容。语义记忆包括事实与概念，情景记忆也会包含事件事实。我们进一步用“精确事实回忆”指逐项回答原始记录细节的任务，把它与从经历中归纳认识分开讨论。

NI 关心的是：哪些细节决定了经历的意义，哪些认识值得在以后继续使用。对于需要准确金额、日期、原文或最新状态的问题，模型可以通过工具访问权威记录；学习则保留有助于理解情境、预测后果和选择行动的结构。“不把全部原始事实背进模型”，因此与保留有用的情景和语义知识相容。

例如，Agent 在游戏里的一次失败，可以被保留为带有条件和后果的情景；比较相关经历之后，模型可能形成关于环境的规律；到了一个从未见过的新局面，它再结合当前条件判断旧规律是否适用。这最后一步属于适应性的推理：利用已有认识处理新问题，并在证据变化时修正判断。

记忆为推理提供材料，推理让材料在当前问题中发挥作用，学习再把新的反馈积累下来。**持续学习的目标，是让这一循环不断形成能力。** 情景不必全部逐字保留，规律也应保有适用条件，模型才能既从过去获益，又不被过去束缚。

2.4 学习是压缩，但不是压得越小越好

我们一直认为，理解学习离不开压缩。人不可能把所有事实的每个细节都保留下来，却能够通过经历形成习惯、判断方法和技能。我们希望模型也能保留对未来有用的结构，而不只是累积过去的表面形式。

我们所说的压缩，是保留意义与作用的压缩：舍弃无关差异，保留会影响理解、预测和行动的信息。它的目标是提高经验的可用性，而非单纯追求最小体积。信息瓶颈理论关心的正是这种取舍：用有限的内部表示保留多少与目标有关的信息，并压缩其余细节。

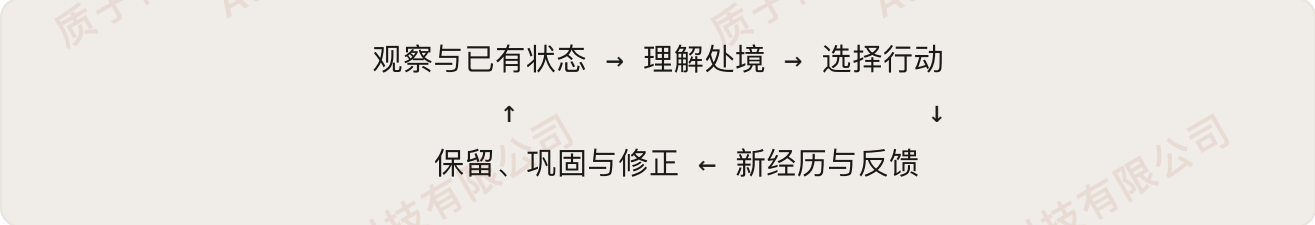
持续学习更困难的一点，在于未来的问题并不完全已知。今天看似无关的细节，明天可能决定判断。我们需要研究的，不只是怎样概括，还包括概括的适用范围、不确定性，以及发现例外之后怎样重新组织认识。

有选择地遗忘，是有限资源中的必要取舍；灾难性遗忘，则是学习新内容时失控地丢掉仍然重要的能力。我们的目标是在接纳新经验的同时，保住仍然有效的理解与技能。

3. 下一代 NI 的核心：让经历改变可使用的状态

3.1 观察、行动与更新应当构成循环

我们以连续的反馈循环组织学习：已有能力与当前经历共同决定怎样更新；更新后的模型与状态，又参与下一次理解和行动。



这里的经历不只是收到了一段文字。它还可以包括做过什么、在什么条件下做、发生了什么，以及原来的预测哪里不对。

Agent 可以主动观察，为减少不确定性而探索，也可以回顾一次失败。行动改变之后能看到的经历，经历又应改变下一次行动。

这个循环还需要识别两种不同变化：一种是在跟踪当前处境，另一种是形成跨经历保留的认识。知道当前所处的位置，与学会“在这种条件下怎样行动”，都需要内部状态，却不是同一种学习成果。

3.2 增量学习：在已经学会的基础上继续

我们对持续学习有一个直接的要求：每一批新经历，都应当在已有学习成果之上继续积累。

增量学习强调利用已有成果吸收新信息；持续学习还要面对长期保持、纠错、迁移和环境变化；在线学习则更强调数据陆续到来时的处理方式。这些概念在研究中有交叉，重要的是说明更新的对象、时机和代价。

第一代 NI 已经支持增量学习。端侧应用会累计新增记录和用户校正，在满足更新条件时提示再次学习，由使用者决定何时启动。

启动后，在逐条提取记录表示时，系统按内容识别没有变化的记录，复用它们已有的内部表示，只为新增或变化的内容重新计算表示。随后，RPP 结合新旧表示重新归纳个人特征，生成、保存并启用新版学习状态，供后续推理使用。这条从新数据进入到学习产物更新、再到推理加载使用的流程，已经在真机上运行验证。

来源：质子梯度内部研究与测试记录。

现有机制把增量提取表示与阶段性整体归纳结合起来：未变化的记录无需重复提取表示，个人特征分析与状态生成则结合新旧资料一起完成。下一代将在这条已经运行的学习路径上，进一步研究更细粒度的更新，让日常学习成本更紧密地随新增信息量变化，并加强长期巩固与旧能力保持。

回顾和整理仍然重要。人也会复盘、归纳和重新理解旧经历。我们主张把日常增量更新与阶段性巩固结合起来：前者保证学习的连续性，后者让已有经历形成更稳定、更可复用的结构。

3.3 模型就是产品：学习产物由模型承载

NI 的设计目标明确：持续学习的产物是模型。经历造成的变化应被模型内化，保留在参数或学习状态中，并参与后续理解、预测与行动。

模型就是产品，意味着产品价值随着模型的学习而积累。 使用者长期积累的价值，不只来自交付时的通用能力，也来自模型后来形成的理解、判断与行动方式。我们希望这些成长成为模型自身可以携带的部分。

把所有记录放进一个数据库，再每次检索出来交给模型，是有用的技术路线，但它回答的主要是“怎样取回外部内容”。NI 要进一步回答的是：这些经历如何改变模型理解和使用信息的方式？

因此，迁移的单位应当是一个完整的已学习模型：既包括执行计算的模型参数，也包括承载已学认识、参与后续计算的内部学习状态。如果学习改变的是权重，就保留更新后的权重；如果变化由快速权重或其他神经记忆状态承载，这些状态就是学习产物的一部分。它们保存为一个还是多个文件，是实现与交付方式的问题。

我们希望，当这个模型进入另一台兼容设备时，已经形成的情景理解、知识与适应性推理能力可以一同延续，无需另外携带原始经历档案或外置检索库来重新建立这些认识。进一步把学习成果巩固进统一的模型参数，是值得探索的一条路径；选择载体时，我们关注可用能力、持续更新与设备代价之间的关系。

我们以功能判断内化：经历是否改变了模型后续理解、预测与行动的方式，这种改变能否持续使用并接受新的修正。设备继续提供计算、感知和动作接口，工具继续提供当前信息与权威事实；模型则带着已经学到的认识进入新的环境。

3.4 不只是恢复使用，还要恢复学习

长期成长意味着模型重启后，既能继续使用已有成果，也能继续沿着原来的机制学习。

恢复学习需要同时保存可读取的成果与继续更新所依赖的信息。例如，一个关联映射可以支持读取，而它后续的更新还可能依赖运行中累计的统计数值或其他辅助状态。学习连续性要求把这些依赖纳入设计。

这也给迁移提出了两层要求：先让已学能力随模型恢复，再让后续成长从同一位置继续。下一代需要研究，哪些信息必须随模型保留，哪些可以由已巩固的状态替代，以及怎样吸收新经历而不反复读取完整历史。

学习过程中，原始资料与历史表示仍可按使用者的选择留在设备上，用于复盘、纠错或阶段性整理；它们与迁移时必须携带的学习产物需要分别说明。我们要逐步降低的是能力使用和日常更新对原始历史的依赖，使模型能够以自身已经形成的认识继续成长。

恢复之后，模型应当从上一次成长的位置继续出发：使用旧有认识，接纳新的证据，并让新旧状态保持一致地演进。

4. 围绕 NI，我们已经构建的技术栈

持续学习要求算法、模型和设备相互配合。我们从学习目标出发，自上而下定义所需能力，再自下而上构建整套技术栈。NI 提供持续学习的算法与算法架构，配套工程则把这些机制带到真实设备，支撑它们的运行、更新、验证与使用。

这套技术栈连接了以下几项工作：

方向	我们已经建立的能力	对持续学习的作用
经验表示与学习机制	RPP、端侧增量学习、方向构建与激活干预接口	从记录与纠正中形成可使用的认识，让变化进入后续推理
模型分析与优化	激活分析、剪枝与量化、优化前后对照，以及模型与应用导出	根据设备条件研究模型的容量与计算取舍，把结果带到实际使用
推理运行时与资源管理	DSR、FrogJump、缓存量化、设备预算、执行调度与多模态状态管理	组织哪些状态被保留、哪些计算被执行，为长期交互提供运行基础

方向	我们已经建立的能力	对持续学习的作用
学习状态生命周期	学习产物的生成、保存、启用、恢复、移除与兼容校验	让学习成果能够跨会话使用，更新与恢复保持一致
工具接入与开发接口	工具契约承载、调用执行，以及把模型、工具与学习流程接入设备应用的接口	把模型的理解连接到设备能够获取的信息与执行的动作
跨设备协作与状态管理	设备间通信、兼容性校验、学习状态传递与恢复机制	让已有学习成果在授权、兼容的设备之间延续使用

来源：质子梯度内部研究。

这些工作由同一个目标牵引：设备要能长期使用模型，也要能接纳新的经历、更新学习成果，并在后续任务中使用它们。下面先说明运行条件与核心机制，再用代表性实测展示这些工程工作的具体作用。

4.1 本地持续学习，需要能够长期运行的模型

智能走向数据，首先要求模型在数据所在的设备上完成感知、推理与交互。持续学习依赖反复发生的观察、行动和反馈，因此，模型在真实设备上的持续运行能力，是我们研究的出发点。

端侧的困难集中在几个相互牵制的环节：模型权重、会话状态和临时计算共享有限内存；新的图像与语音不断进入；不同芯片又有不同的执行与数据搬运成本。处理这些问题，需要理解模型的计算结构，并据此设计状态管理、内存分配与执行调度。

推理运行时，是负责安排模型计算、内存使用与状态管理的软件。我们把模型优化与运行时协同设计，已经在连续推理、会话状态保留和非苹果平台的语音执行优化上取得具体进展。沿着这些成果，下一代 NI 将进一步研究学习更新的计算与状态需求，让模型能够在持续交互中积累新的能力。

4.2 模型优化与运行时的核心机制

我们把模型运行拆成几个需要协同解决的问题：哪些历史状态值得保留，哪些计算可以省下，有限内存怎样分配，感知切换时怎样延续上下文，以及优化后的模型怎样进入设备使用。围绕这些问题，我们已经形成了以下机制与工具。

DSR（Dual-Sparse Retention，双重稀疏 token 驻留）：在有限缓存中安排历史信息的保留。

长会话中的注意力缓存，会随着输入与输出不断增长。注意力缓存也称 KV 缓存，保存模型后续计算会查询的内部表示。DSR 将有限空间分配给开头的少量位置、近期内容，以及历史中注意力得分较高的内容，并结合层的位置与场景配置安排保留预算。

这里的重要性由模型运行中的注意力得分估计，得分会随后续计算更新。DSR 据此选择保留的状态，定期整理缓存，让历史的保留进入明确的资源预算。它把“有限内存留给什么”变成运行时可以执行的规则，是我们支持长时间交互的一项核心机制。

FrogJump：按模型结构组织更精简的计算路径。

这一机制面向由递归层与注意力层共同组成的混合架构：递归层持续整合先前信息，注意力层则从保留的表示中读取所需内容。

FrogJump 根据已适配的模型层结构生成跳层计划，执行时略过选定的中间递归计算层，同时保留完整注意力层及紧邻其前的递归层。这样，模型可以沿着一条计算量更少的路径处理输入。

我们已经实现跳层计划、推理执行接入，以及对应的结构检查与测试。当前路径在适配模型的非思考模式下显式启用。FrogJump 的价值在于把计算取舍落实为可配置的执行方式；输出质量、任务表现与实际运行成本，是评价这类路径时需要一起观察的结果。

缓存量化与混合状态管理：同时考虑保留多少、怎样保存。

DSR 处理状态的选择，缓存量化处理状态的数值表示。量化用更少的位数保存数值，降低缓存占用，同时引入需要评估的数值误差。我们把量化格式、缓存预算与实际计算路径一起设计，让存储方式能够配合模型执行。

混合架构还要求分别管理不同性质的状态：注意力缓存保留供后续查询的表示，递归状态则把先前信息整合进固定形状的数值。我们的运行时分别处理它们的分配、更新与恢复，使不同状态的增长方式能够被独立管理。

设备预算与执行调度：让计算安排适应实际运行条件。

我们根据设备可用内存、模型占用、带宽信息与执行能力规划资源，为临时计算预留空间，再分配缓存预算和处理输入时的批次规模。随着上下文增长，执行调度还可以调整每批提交的计算量，控制瞬时压力。

这项工作把模型结构与设备条件连接起来。同一个模型进入不同设备时，缓存规模、计算批次和提交节奏可以采用不同安排，为持续交互提供与硬件条件相匹配的运行方式。

多模态状态连续性：切换计算任务时，保留需要延续的状态。

视觉处理与语言解码轮流使用资源时，运行时分别管理权重和会话状态；语音分块生成时，则延续输出所需的内部状态。围绕这些差异，我们实现了分阶段资源安排与状态保留路径，让新的观察或下一段输出能够接续已有计算。

模型分析、优化与交付：把研究中的改动带到实际使用。

我们还建立了激活分析、剪枝与量化、优化后重新加载和质量检查，以及模型与应用导出的工作流程。激活分析帮助观察计算中的使用情况，剪枝与量化提供不同的容量和数值表示方案，对照检查则帮助判断改动的影响。

配套开发接口把模型运行、工具执行和学习状态管理接入设备应用。这使模型优化、学习机制与设备运行能够相互衔接：研究得到的改动可以被加载、检验和使用，设备中的运行结果又能回到下一轮研究。

运行配置同时遵循状态兼容要求。长会话可以采用有预算的缓存管理；恢复已捕获的完整学习状态时，则使用与状态生成相兼容的配置，保持其布局与执行方式的一致性。

来源：质子梯度内部实现与研究记录。

这些机制共同决定模型如何使用有限的计算与存储资源；下面的学习机制进一步决定，新的经历怎样进入可保留、可使用的模型状态。

4.3 RPP

RPP 用来从多条记录中提取相对稳定的个人特征线索。它读入模型处理这些记录时产生的内部数值，输出一组用于个人特征分析的数值方向，帮助辨认哪些结构在不同记录中反复出现。

具体而言，RPP 先以预设的通用特征方向为参照，分离它们在记录中对应的成分，再分析剩余结构。这里的通用方向，是描述一般特征、供不同用户共同使用的一组参照。之后，再通过加权主成分分析与稳定性检查，形成用于个人特征提取的方向。

这里的“激活”，可以理解为模型在处理输入时形成的内部数值表示；“方向”描述的是这些表示中某一类变化。“主成分分析”寻找较显著的变化结构，“稳定性检查”则观察这些结构是否容易随样本变化而改变。

RPP 的意义在于，它让我们能够在模型的表示空间里观察个人相关结构，而不只是对原文作一次表面摘要。下一代可以继承这项基础，也可以研究表示、分析目标与后续使用方式怎样继续演进。

在现有流程中，这些方向帮助选出有代表性的记录，再由模型结合这些记录概括个人特征。得到的文字描述与工具说明一起，构成下一节用于生成可复用状态的输入。

来源：质子梯度内部研究。

沿着这一基础，下一代将进一步研究时间、条件与行动后果的表示：使分析得到的结构既能被识别，也能参与后续判断与预测。这是从个人相关结构走向更丰富学习能力的重要连接点。

4.4 学习状态如何进入推理?

模型先读入一段描述用户特征和可用工具的文字，我们将读完后形成的内部计算状态保存下来。以后开始新的会话，可以直接从这份状态继续处理当前问题，复用先前已经完成的计算。

第一代 NI 已经实现了这些状态的生成、更新、保存、恢复与移除。技术上，先读入的那段文字称为“前缀”；其中的“工具契约”，就是工具的功能、参数和使用方式说明。

这种方式使先前形成的内部计算状态可以跨会话复用，后续输入能够在已有状态的基础上参与计算。再次学习时，更新后的个人特征与工具说明一起生成新版状态，替换此前用于推理的版本；后续会话便可恢复新版状态继续工作。下一代将进一步研究更细粒度的状态更新，让新认识的写入、旧认识的修正与长期巩固更紧密地配合。

不同模型结构提供了不同的保留与整合机制。有的状态随输入序列增长，支持后续注意力查询；有的通过递归更新，把之前的信息整合进固定形状的状态。理解这些差异，有助于为不同的容量、读取与更新需求选择合适的承载方式。

4.5 工具学习与个人状态参与推理

工具学习是第一代 NI 已有的基础。工具契约描述工具能做什么、接受哪些参数、怎样使用；模型处理这些契约并形成状态，恢复后可以根据当前问题生成工具调用，由运行时执行。

例如，用户询问某个月花了多少钱，模型可以调用已接入的消费记录查询工具获取数据。

在这条路径中，模型在构建阶段理解工具契约；真实工具完成实现与注册后，使用阶段便可复用已形成的状态，无需为每次请求重复注入同一份工具说明。

我们已在设备上验证个人特征回答与对应工具调用的实际流程，验证内容包括状态恢复、基于当前问题生成调用，以及由运行时执行工具。

来源：质子梯度内部测试记录。

个人状态让模型能够依据已形成的偏好与规律调整回答，更接近前面所说的语义记忆。以此为起点，下一代将进一步研究具体经历及其时间、情境和后果的保留，并让这些认识支持条件变化后的判断。

下一代还要从“知道工具契约”走向“从使用经历中完善策略”：什么时候值得调用，失败以后怎样调整，协议变化后怎样重新适应。成功调用一次工具，是这条路上的基础，不是终点。

4.6 Activation Steering 与 Directional Steering

Activation Steering 是在模型生成回答时，调节它内部正在使用的数值，从而影响接下来怎样回答。这些运行中的数值称为“激活”；这种方式可以在保持模型权重不变的情况下使用。

Anthropic 的金门大桥实验是一个直观例子：增强与金门大桥相关的特征后，模型会在原本不相关的话题中也提到它。这展示了内部激活对生成结果的直接影响，也凸显了干预强度与适用情境的重要性。

我们的现有实现包含两条方向处理路径：一条利用 RPP 产生的方向观察当前输入与已有结构的关系，并为干预准备所需的数值数据；另一条从纠正前后的激活差出发，构建用于影响后续计算的方向，称为 **Directional Steering**。

两条路径都围绕内部激活，区别在于方向怎样获得、什么时候使用、如何接入计算。方向是一组协调的数值调整量；Directional Steering 是激活干预的一种具体实现。

目前，RPP 方向在现有应用路径中用于观察输入与个人相关结构的关系；纠正方向的实际干预，已在专项运行中验证加载、重新加载与移除。这构成了我们继续研究方向选择、情境匹配与干预效果的基础。

来源：质子梯度内部研究。

下一代将重点研究方向与学习目标的对应、干预的情境选择，以及连续更新中的相互影响。

4.7 代表性工程成果与实测

我们从上述工作中选取三个有具体运行记录的案例，分别展示长时间运行、多模态状态连续性与执行效率。长会话案例结合了缓存量化、内存预算、动态调度与有上限的缓存保留；多图和语音案例分别展示状态管理与计算组织的改进。

本节数据来源：质子梯度内部测试记录。

控制长会话中的缓存与计算开销，让大模型在手机上持续运行。

长会话会不断增加缓存和计算压力。模型中的一部分层把历史整合成固定大小的状态，另一部分层则保留供后续查询的注意力缓存。我们分别管理这两类状态，以更紧凑的数值格式保存缓存，设定内存预算，并随上下文增长调整每批计算的规模。这样，运行时能够在设备内存约束下持续推进生成。

在两台手机的连续运行测试中，90 亿参数模型在同一会话内分别完成了 200 轮问答，累计各生成 204,800 token，峰值内存均约为 5.5 GB。Token 是模型处理文字的基本单位；这里观察的是手机能否连续完成一轮又一轮的生成。

测试条件：Qwen3.5-9B-4bit，iPhone Air 与 iPhone 17 Pro，每轮固定生成 1,024 token，持续供电并主动散热。

这项工作把长会话中的状态增长与执行压力纳入了明确预算，使连续交互拥有可实际运行的设备基础。

把权重加载与会话状态分开管理，让新图像接入已有上下文。

图像处理需要额外内存。直接释放整个语言模型会话，会同时丢掉已经形成的状态，之后还要重新计算历史。我们将模型权重的生命周期与会话状态分离：暂时卸载解码器权重，保留注意力与递归状态；视觉编码完成后，再恢复权重，只处理新图像及新增文字。

在多图连续会话对照中，第二张图送入后，模型开始输出回答的等待时间从 8.15 秒降到 6.64 秒，也就是少等约 1.5 秒，同时保留此前的会话状态。保留状态带来的额外峰值内存约为 190 MB。

测试条件：iPhone Air、Qwen3.5-9B 多图工作负载；状态保留机制为默认关闭、测试时启用的实验路径。

这项改动解决了感知切换时的历史重算问题。设备可以为新的观察腾出计算资源，同时延续已经形成的会话状态。这也是我们面向持续学习设计运行时的重要原则：资源可以重新安排，状态应当有独立的保留与恢复机制。

重组语音预测器的执行结构，提高非苹果平台上的计算效率。

语音预测器负责逐步生成合成声音所需的编码信息，每一步只需要对应的输出部分，也就是“输出头”。我们把各步骤共用的计算部分——共享主干——与输出头拆成独立的执行单元，在保留原有权重数值和主干状态更新结构的同时，只运行当前所需的固定权重输出头，并将主干输出直接传入该输出头。

在某款非苹果平台的旧款芯片上，预测器固定 15 步计算的耗时中位数从 40.05 毫秒降到 32.28 毫秒。另行进行的三条流式输入对照中，优化前后的生成音频逐字节一致。

测试条件：相同输入，原路径与优化路径交替运行 10 轮；上述耗时统计预测器计算，包含调用与等待。

在完整语音管线上，我们还实现了有状态的分块生成，让设备一边合成、一边提供可播放的音频。在一轮 10 个会话的本地语音测试中，从服务收到文本到发出第一块音频，中位等待约为 0.37 秒。另一次独立的 30 分钟语音合成测试完成了 357 句，失败数为 0。

这项优化保留了既有权重，通过改变执行组织方式降低计算耗时。它展示了我们在异构设备上推进模型优化的具体方法：结合模型结构与平台执行特性，组织真正需要发生的计算。

从长会话的资源管理，到多模态切换的状态延续，再到语音生成的执行重组，我们已经围绕模型长期驻留和持续交互建立了具体的工程能力。它们共同支撑 NI 的研究方向：让学习所依赖的观察、推理与状态积累，能够在用户的设备上持续发生。

4.8 从已有基础走向下一代

我们已经有运行条件、表示分析、端侧增量学习、工具契约承载、个人状态参与推理、方向处理与状态恢复等具体基础。新增记录与校正能够进入学习流程，更新后的学习产物能够保存、启用并参与后续推理。

下一代将从这条已经运行的学习循环继续演进，拓展表示、更新与巩固的方法，让可积累的内容从个人特征与工具用法，进一步扩展到情境理解、行动后果与可迁移的策略。

5. 语言、世界模型与 Agent，为什么会在这里汇合?

5.1 不只知道发生过什么，还要理解可能发生什么

语言模型擅长处理语言中的结构、知识和表达。世界模型则把我们的注意力引向另一个问题：在当前处境下采取某种行动，环境可能怎样变化?

经典 World Models 将感知表示、时序预测和控制联系起来；Dreamer 则进一步展示了利用内部预测展开想象轨迹、学习行为的路线。我们看重的是其中的原则：表示应当服务于后果预测与决策。

对 NI 来说，这带来一个重要方向：经历的价值不只在于以后能回答“发生过什么”，还在于它是否帮助我们判断“在这种条件下，接下来可能怎样”。

后果预测要求表示保留行动、时序和环境条件之间的关系。我们希望模型能够识别，一次结果究竟依赖哪些条件；在内部推演之后，再用真实反馈校正预测，减少连续推演中的误差积累。

5.2 设备就是智能体：模型在设备中进入环境

在我们的研究中，Agent 是模型进入目标、观察、行动与反馈循环的形态。设备提供本地计算、感知与交互入口，以及实际可执行的动作；模型提供理解、预测与选择行动的能力；NI 让经历形成的变化能够保留、更新并继续使用。三者结合，让设备在自己的环境中作为智能体运行。

这也解释了我们为什么把计算能力放在智能体概念的基础位置：设备要能够在本地处理信息、运行模型并管理学习状态，才能在持续交互中承担判断与学习。手机、笔记本和其他异构设备可以具有不同的算力、感知条件与动作范围；我们的研究方向，是让它们依据各自条件承担不同角色，并在使用者授权下协作、延续已有学习成果。

这样的智能体可以主动获取信息，调用工具改变环境，也可以回忆过去的尝试。

这使学习面临的问题发生变化。被动阅读时，接下来看到什么主要由外部决定；主动交互时，Agent 的选择会影响未来数据。它可以反复做熟悉的事情，也可以承担适当成本去确认一个不确定判断。

因此，自主学习不只涉及“新内容如何写入”，还包括“下一步应该观察什么”。学会提出有价值的问题、辨认反馈是否可信、发现自己并不知道什么，都可能成为能力的一部分。

5.3 在游戏中学习：经历怎样变成能力？

我们喜欢用游戏来理解这一点。Agent 进入陌生环境，尝试行动，观察结果，逐渐发现哪些条件重要，形成可复用的判断方法。它并不是把每一局的记录重新背一遍，而是在经历中改变面对下一局的方式。

在这个场景中，学习表现为持续的变化：先前经历影响后来的策略，成功的方法得到巩固，失败的认识得到修正，环境变化时能够重新调整。我们将围绕这些变化观察和验证能力的积累。

完成当前任务，与通过当前任务变得更有能力，是两个相关但不同的目标。 持续学习要把第二个目标也纳入系统设计和评价。

5.4 能不能统一成一个模型？

统一语言表达、环境预测与行动选择，是我们看重的研究方向。这些功能可以探索共享的内部表示，让模型对世界的理解同时服务于表达、推演与行动。

WorldVLA 用一个逐步生成的统一模型处理动作生成与未来视觉预测，为这种功能统一提供了具体参照。我们关注的下一步，是怎样把持续更新与经验巩固纳入这样的统一结构。

我们更深一层的期待是：模型说出的解释、对未来的判断和采取的行动，能够基于相互一致的认识。共享表示有望减少转换与割裂，也要求我们处理不同功能之间的相互影响，包括错误怎样被发现、限制和修正。

因此，我们将统一结构与持续更新作为两个相互联系的研究问题：前者连接理解、预测与行动，后者让这些能力随着新经历继续成长。

5.5 它们各自承担什么？

语言与多模态能力帮助模型理解和表达；世界建模帮助它预测后果；NI 研究经历怎样持续改变可使用的模型状态；Agent 将这些能力放入真实交互；验证环境则检验这种变化究竟有何作用。

这些职责可以由不同架构组织，也可以逐步统一到同一模型中。关键关系始终清楚：模型在内部形成变化，Agent 在环境中获得经历，评价通过独立任务检验结果。

6. 技术方法与哲学启发：学习机制应该怎样设计？

我们从具体的学习问题出发设计方法：经历怎样被表示、怎样引起更新、怎样被再次使用，以及整个过程怎样持续。

6.1 先让表示保留真正重要的区别

如果两个后果完全不同的事件被表示成几乎一样的状态，后续再精巧的读取机制也可能混淆它们；如果同一件事的不同表达相距很远，写入时和提问时又可能对不上。

因此，表示不只是把输入变成一串数字。它决定哪些差异被保留、哪些关系容易被利用。对持续学习而言，这种表示应同时服务回忆、判断与后果预测，而不只是让相似的文字靠近。

一个值得研究的方向，是让表示在新的反馈中继续调整。但调整也会带来困难：如果表示空间发生变化，过去已经写入的状态是否还能读懂？我们不仅需要学会写入，还要保持新旧表示之间的可用关系。

6.2 关联记忆：给“叠贴”一套读写规则

关联记忆按线索寻找内容。例如，当前提问提供线索，模型据此调用先前经历中相关的信息。快速变化的权重或状态，让新的线索与内容在模型内部形成关联，读取时再用当前线索激活相关信息。

简单叠加会产生干扰，基于预测误差的写入则可以修正已有映射，而不是只增加一层痕迹。这类研究把“墙”和“叠贴”从比喻推进到了可分析的更新规则。

自然语言的读写还需要解决线索对齐：同一段经历可能以不同说法被提问，相似说法也可能包含相反条件。地址怎样形成、怎样容忍表达变化、怎样保留关键差异，是关联记忆进入真实交互时的核心问题。

下一代将把容量、读取可靠性与不确定性表达一起研究，使已经写入的结构能够在相关情境中被准确使用。

6.3 先学会怎样读写，再持续接受新经历：TTT 的启发

“先教会怎样贴、怎样找、怎样读懂；以后再不断给它新的纸片”，是一个重要的研究思路。它把通用的学习机制，与每次具体经历引起的变化区分开来。

Larimar 将编码、分布式记忆和解码连接起来，研究快速知识更新；MemoryLLM 则研究可自更新的内部记忆池。对我们而言，它们最有价值的启发是：读写之间的配合本身可以学习。先形成吸收新经历的机制，再让使用中的经历持续更新模型状态，是一条值得深入探索的路径。

这条路径的关键是表示、写入与读取的共同设计。我们将结合设备资源、长期更新和数据本地的要求，检验这种配合的效果，并分别衡量形成学习机制与日常使用它的代价。

测试时训练（Test-Time Training，TTT）把学习更新带进模型的使用过程。 模型接触新的输入后，根据一个学习目标调整可更新的参数，再利用更新后的模型处理任务。原始 TTT 研究利用输入本身构造自监督任务，在没有当前样本人工标签的情况下更新特征提取器，以适应与原训练数据不同的输入。这里的训练确实改变了参数；额外生成几轮回答或延长推理时间，则属于另一类计算安排。相关讨论也可参看这份 TTT 分享。

TTT 的更新可以有不同的保留方式：原始工作中的标准版本为每个输入从同一起点适应，完成预测后丢弃这次更新；在线版本则沿用前一个输入更新后的参数。这个区别与 NI 的研究问题直接相连：当前任务学到的变化，怎样成为下一次仍然可以使用和修正的认识？TTT 提供使用时更新的方法，测试时持续学习进一步关注这种变化的保持、巩固与长期积累。

TTT Layers 则把这种思想放进了模型内部的序列处理结构。 《Learning to (Learn at Test Time)》让隐藏状态本身成为一个可训练的小模型：新的输入到来时，以自监督学习更新它的权重，再用更新后的状态处理信息。外层训练学习表示、读写相关变换及其他模型参数，内层更新把当前序列的信息压缩进快速变化的权重。

这为“先学会怎样贴，再不断接收纸片”提供了具体参照：一套较稳定的学习机制，可以驱动使用过程中的快速变化。该研究按输入序列组织快速状态；把这类机制用于 NI 时，我们还要设计跨会话保留、不同经历之间的干扰控制，以及快速状态如何巩固为长期能力。这些问题会与下一节的多时间尺度设计一起研究。

对 NI 而言，TTT 的吸引力在于把“学习”变成运行过程可以调用的机制。我们可以研究更新信号怎样来自预测误差、输入结构或行动反馈，也可以研究哪些部分适合快速调整、哪些认识需要更多证据再改变。自监督更新是一条具体路径，学习机制的形成与使用并不必然依赖强化学习奖励。

“学会学习”落实到三个具体问题：系统如何利用新证据决定更新什么，如何辨别异常与规律，以及什么时候应该回顾或修正已经形成的认识。

6.4 多时间尺度：新鲜经历与稳定能力不必同速变化

刚发生的事情，往往需要快速进入可用状态；长期形成的判断，则不应该因为一个偶然样本就被推翻。如果所有内容采用同样的写入力度、保持时间和遗忘速度，系统很难同时具备敏捷性与稳定性。

多时间尺度是我们下一代研究的重要方向。快速状态处理眼前变化，较慢的状态保留经过更多证据支持的结构；阶段性的整合再决定哪些经历值得成为更稳定的能力。

Titans 将运行期间可更新的记忆、遗忘机制与不同信息路径结合，为这种设计提供了参照。我们关注这些机制如何分工，以及额外计算、状态容量和长期保持之间的取舍。

巩固发生在不同时间尺度之间的有效协作中：新经历进入快速状态，有价值的结构逐步稳定下来，旧认识则在新证据下得到重新组织。信息怎样交换、哪些条件得到保留，将决定这种协作的实际效果。

对应到前面的记忆区分，模型既可以保留某次经历的重要情境，也可以从多次经历中形成更通用的认识。这些功能与更新速度需要分别设计：有些具体经历值得长期保留，有些一般性判断则需要快速修正。我们将依据内容的作用与证据强度安排更新和巩固。

6.5 写入、遗忘与回忆，都是取舍

令人意外的经历值得注意：当结果与预测差距很大时，原有认识可能需要调整。更新机制还要区分有意义的新证据与随机噪声，才能把有限学习资源用在真正重要的变化上。

我们将新颖性、可信度、任务价值和长期影响共同纳入研究。频繁发生的事情可能重要，少见但后果严重的例外也可能重要；最近出现的认识可能更新，长期积累的认识也不该被轻易抹去。

更新门决定写入多少，遗忘门决定保留或衰减什么，读取机制决定当前使用什么。我们需要让这些控制与内容的重要性、可信度和适用情境形成联系，在有限容量中协调写入、保持与调用。

回忆同样需要选择。持续学习的价值不是在每次回答前重新审阅全部人生，而是能够在当前情境中调用有用的结构，并在发现不足时知道需要补充什么。

6.6 芯片给我们的哲学启发：通用机制怎样面对未来任务？

一块芯片不需要预先知道未来所有程序。只要程序满足指令与执行条件，它就能利用既有机制进行计算。这给我们的启发是：**稳定的基础机制，可以承载后来才出现的内容和任务。**

对应到持续学习，我们不可能提前准备用户未来的全部经历。我们要研究的是，模型是否能获得一套吸收新经历的机制，而不必为每一个新领域、每一种纠正和每一位用户另写一套语义规则。

芯片通常执行外部给出的程序；智能体还需要从不完整、有噪声的反馈中发现规律，甚至判断自己提出的问题是否恰当。这使学习机制承担了更进一步的任务：既要接纳新的内容，也要判断新认识是否可靠。

这个类比带给我们的核心判断是：**稳定的基础机制，可以拥有不断拓展的知识与能力边界。** 我们要寻找的，正是能接纳后来经历、检验新认识并持续形成能力的学习机制。

6.7 异构集成的启发：分工与联系，而不只是增加层数

三维芯片集成将不同功能和工艺的部分，通过紧密互联组合起来。TSMC SoIC 所体现的分工与连接思路，对我们理解模型内部的协作也有启发。

借助这个工程类比，我们关注三个问题：不同类型的状态能否采用不同表示？不同更新速度的机制怎样交换有用信息？为了增加容量而引入的连接，是否带来了可接受的读取与通信成本？

容量不必只靠加深主干获得；它也可能来自新的模块、子空间或不同的组织方式。但新增容量只有能被学习机制写入、被推理过程使用，才会成为实际能力。

对我们来说，最值得保留的启发是系统性的取舍：不能只问能装多少，还要问怎样访问、怎样更新、怎样协调，以及为了这些能力付出了多少代价。

7. 真正需要攻克的难题是什么?

7.1 压缩要服务尚未遇到的任务

只为当前问题设计的表示，可能在下一次问题中丢掉关键线索。我们需要在保留内容的广度、组织结构的深度与资源成本之间取得平衡，让当前形成的表示能够服务更广泛的未来问题。

所以，下一代 NI 面对的不是单一压缩率，而是一条取舍曲线：在明确资源预算下，能保留多少有用结构，支持多大范围的未来任务，遗漏又发生在哪里。

这也意味着“重要”不是永久标签。一条看似次要的经历，可能因为环境改变而获得新的意义。如何保留重新解释过去的可能性，是比摘要长度更深的问题。

7.2 状态变化怎样成为能力变化?

最难的未必是让一个矩阵发生变化，而是让模型正确使用这种变化。

个性化表达、准确回忆、任务效率与陌生条件下的判断能力，是不同的改进维度。持续学习研究需要识别，哪些变化来自经历中的规律，哪些依赖具体内容的保留，以及这些变化能否继续积累。

我们将通过控制上下文、工具和计算预算，比较有无学习更新时的行为差异。把贡献归因到具体机制，才能知道下一步应该加强哪一种表示、更新或读取方式。

7.3 从相关性走向对行动后果的可靠判断

从事件之间的关联走向行动后果的判断，需要识别环境条件、反馈时序与潜在影响因素。同一种行动在不同处境下可能产生不同结果，学习应当保留这种条件性。

Agent 因此需要处理归因：一次成功或失败，应该改变对哪一步、哪种条件的认识？过早归纳可能让它形成错误规律；把每一次例外都当作新规则，也可能让它无所适从。

世界建模可以参与这个过程：先预测行动后果，再与真实反馈对照，通过新的观察与尝试修正认识。我们把这种可检验的预测—行动关系，视为理解进一步转化为能力的关键。

7.4 在接受新经验与保留旧能力之间保持平衡

可塑性让模型接受新信息，稳定性让它保留仍然有效的认识。过度稳定会拒绝真实变化，过度可塑又可能被一次异常经历带偏。

我们主张让学习机制保留不确定性、区分适用条件、允许暂时并存的解释，并在更多证据出现后决定怎样整合。这使模型既能接受变化，也能抵抗偶然噪声。

遗忘管理信息的保留与衰减，纠错修正已有认识，删除落实使用者对信息去留的决定。三者需要各自清楚的语义，并共同进入学习状态的生命周期设计。

7.5 端侧约束必须进入算法本身

一份供推理读取的学习状态文件可能很小，更新它却需要较多计算；一个读取很快的机制，可能需要保存大量辅助状态。我们将编码、更新、暂存和恢复的成本一起纳入算法设计。

我们需要协调日常推理与学习的资源使用：轻量更新及时响应，较重整理安排在合适时机，持续交互保持可用。计算时机、状态容量与更新收益，应当与学习目标一起设计。

不同设备的内存、功耗和使用节奏，会形成不同的算法取舍。我们已有的运行时成果，使这些问题能够进入真实环境中的测量与比较，让算法设计建立在实际运行条件之上。

7.6 跨设备与异构环境：迁移的究竟是什么？

我们希望已经形成的学习成果能够伴随使用者跨设备延续。实现这个目标，需要处理三类迁移问题。

迁移对象	真正需要回答的问题
学习机制	同一原则能否在不同算力、内存和运行环境下工作？
学习状态	已有状态能否在兼容条件下恢复；表示不同的时候怎样转换？
已学能力	感知形式、工具和动作空间改变之后，哪些认识仍然有用？

我们已经有兼容状态传输与恢复的实现基础，为同模型兼容条件下的学习连续性提供了实际通路。

来源：质子梯度内部研究。

下一步要进一步研究不同模型表示、感知形式与动作空间之间的衔接。

“模型就是产品”使迁移有了明确的研究目标：学习产物进入兼容设备后，能够继续使用已学认识，并从新经历中接着学习。我们将分别观察能力能否保留、更新能否延续，以及为实现这两件事必须携带哪些状态。模型对旧经历的依赖，应当逐步体现在自身形成的认识中。

内部状态依赖模型的表示方式。不同模型的层结构、数值坐标和动作语义可能不同，迁移因此需要处理表示对齐与能力重用。例如，将已有规划经验用于另一类设备，需要重新衔接其感知与执行方式。

异构迁移的核心，是识别哪些认识能够共享，哪些细节需要适应新的载体。我们希望延续的是已经形成的理解与能力，并让它们在新的环境中继续生长。

8. 来自 AgentOdyssey 的对照与参考

8.1 它提供了哪些可观察的对照?

AgentOdyssey 是一个用程序生成游戏环境的测试平台，用来观察智能体在长期交互中怎样学习。Agent 要接触新情况、选择行动、获得后果，再处理依赖此前经历的问题。它的价值在于让学习的过程及作用能够被观察。

它使我们能够更具体地追问：Agent 获得了哪些环境知识？是否保留了重要经历？探索是否带来有价值的信息？这些变化有没有帮助它推进任务?

官方评价包含任务表现、交互前后的世界知识问答、情景记忆以及探索等维度。这比只看最终是否通关，更适合分析经历产生的作用。

我们将这些维度联系起来观察：回忆为判断提供什么，探索带来了哪些新认识，环境知识又怎样改变行动。这样才能看清经历从被保留到被使用的完整过程。

8.2 与外置检索、上下文堆叠等做法的取舍

更宽的方法范围，为我们提供了丰富的比较对象。我们关注每种做法解决的问题、承担的代价，以及它能为持续学习提供什么。

做法	它擅长提供什么	对持续学习研究的启发
把历史继续放进上下文	让模型在当前计算中直接看到更多经历	比较直接使用历史与内部积累的效果、时延和资源增长

做法	它擅长提供什么	对持续学习研究的启发
在外部保存并按需取回材料	保留可检查、可更新的原始内容与来源	比较准确取回内容与内化规律各自的价值，而不是只比较回答是否相同
把历史整理成文字摘要	用较少文本传递此前的重要信息	检查语言概括保留了什么，以及反复整理时怎样失真
改变模型内部可用状态	让经历进入后续计算的条件或机制	检查变化能否持续、是否被正确使用，以及更新和恢复的代价

这些路线可以互相组合。NI 聚焦经历的内化与持续学习，精确事实则可以由工具访问权威来源。我们通过比较识别各类机制的长处、资源代价与适用条件，再决定怎样组织它们。

比较可以采用相同起点与相近资源预算，以识别机制的贡献；也可以报告不同预算下的效果曲线，以判断实际价值。这两种视角共同帮助我们作出有依据的取舍。

8.3 把这些参考用于 NI 的长期学习研究

借鉴 AgentOdyssey 的交互任务与评价维度，我们将在自己的实验中分别观察环境内适应、长期保持和跨环境积累，再把这些问题带到真实设备上检验。

例如，在游戏之间重置状态，适合观察单个环境内怎样获得新认识；跨环境保留状态，则有助于研究旧经验的保持、迁移与干扰。协议的选择应当对应我们真正要回答的学习问题。

AgentOdyssey 提供的任务组织与对照方法，帮助我们分析一段经历带来了什么知识，这些知识又怎样改变后续行动。对 NI 而言，这些参考最终要服务于真实设备上的长期学习：让模型在日常使用中持续积累认识、完善判断与行动。

9. 怎样证明持续学习发生了？

9.1 不只看结果，还要看变化来自哪里

下一阶段的实验将围绕明确的学习问题展开，用对应的观察和对照识别机制的作用。

问题	需要观察的证据
经历是否进入了可用状态？	在不重新提供同一段历史的条件下，声明的学习状态能否支持后续任务
更新是否真的起作用？	比较允许更新与禁止更新，并控制额外上下文、工具和计算预算
是否超出原题复述？	检查新的表达、条件组合和独立任务，而不仅是重复旧问题
是否保持与修正了能力？	连续加入经历，检查旧能力、冲突证据与无关任务

问题	需要观察的证据
是否形成了学习连续性?	恢复后不仅能读取旧状态，还能继续接收经历并有效更新
是否有实际运行价值?	同时记录完整状态、更新成本、推理代价与运行可靠性

缓存复用、回忆、归纳和能力整合可以出现在同一系统中。我们将分别测量它们的作用，并观察它们怎样共同支持持续成长。

验证也将连接不同层次：数学分析帮助理解容量与干扰，自然语言任务检验表示与读取，环境交互检验行动变化，设备实验检验长期运行的可行性。

围绕模型的独立承载与迁移，我们还将检验：在不重新提供旧经历原文的条件下，恢复完整学习产物之后，模型能否回忆相关情景、使用已有知识，并在接触新反馈后继续修正。这能把“携带了哪些状态”与“延续了哪些能力”联系起来。

9.2 让实验顺序跟着科学问题走

一个合理顺序是，先看表示、更新和读取是否能形成最小闭环，再看连续经历中的保持与修正，然后进入更复杂的行动、迁移和设备条件。

这种顺序便于定位问题。读取失败时，可以先检查表示与线索对齐；长期退化时，可以进一步分析容量、干扰和更新策略；状态恢复后，则检验新经历能否继续被正确吸收。

世界模型的评价同样连接预测与行动：既测量预测的准确性，也测量这些预测给决策带来的作用。我们追求的是让模型更会面对未来。

9.3 学习经历与独立检验要分开

情景记忆评价关心经历是否被保留，泛化评价关心已有认识能否用于未见任务。我们分别报告这两类能力，让每项结果都有清楚的含义。

独立测试的答案、环境隐藏信息与评分依据应当与学习过程隔离；用于选择方案的任务，与最终检验效果的任务也要分开。这样的设计使结果能够反映真正的学习作用。

真实数据的语义目标与评价依据，需要由独立、可追溯的评估流程提供。我们以一致的评价条件检验不同方法，并如实呈现成功与失败。

真实记录有助于研究日常信息的分布，交互环境则补充行动与反馈。我们将覆盖不同用户和场景，逐步检验适用范围；数据本地与使用者授权始终是前提。

9.4 资源账本同时覆盖“使用”与“继续学习”

资源评价至少需要区分三件事：学习过程要保留什么，推理时要挂载什么，为下一次更新还需要什么。

评价还将计入输入上下文、临时内存、编码计算和状态恢复成本。外部索引、更新辅助态等依赖，也属于对应方法的完整资源开销。

我们以能力与完整代价之间的关系评估方案，判断一种机制能否从实验延续到长期使用。

10. 我们的研究选择与开放取舍

我们的方向是明确的：NI 始终以持续学习为目标，智能走向数据，学习成果由模型及其学习状态承载，成长应当能够在真实设备与变化环境中延续。

实现方式保持开放。我们根据表示、更新与巩固的实际效果选择方法，以持续获得能力和长期运行价值衡量研究贡献。

10.1 哪些变化应该成为模型的一部分?

当前任务的临时条件、仍待确认的观察、已经稳定的规律，应当采用不同的处理方式。让重要认识逐步沉淀，是有限资源下持续学习的核心取舍。

学习机制应当解释哪些变化被保留、依据是什么，以及新证据出现后如何调整。选择值得长期积累的结构，是持续学习必须具备的能力。

10.2 容量增长与巩固怎样协调?

固定预算、按设备能力配置预算、适度扩展容量，各有代价。我们将容量管理与经验巩固结合起来研究，既提高已有状态的利用率，也检验适度扩展带来的收益与成本。

快速吸收、阶段性整合和更长期的能力保持，需要协调各自的节奏。下一代将研究这些过程之间的信息交换与调度，使日常更新和长期积累相互支持。

10.3 自主学习与使用者控制权如何共存?

环境反馈帮助模型判断什么值得注意，使用者决定学习的范围，以及状态保留和迁移的边界。学习的主动性与人的控制权，需要在同一设计中得到落实。

智能体应当在明确授权的范围内主动观察、更新与回顾，使用者能够理解、撤销和恢复相关变化。控制权贯穿学习过程。

10.4 不同设备可以有不同分工

设备之间的算力和使用节奏不同，日常适应与较重的整理未必必须在同一时刻、以同样方式完成。

这种分工将以数据本地、兼容性和使用者授权为前提，根据不同设备的资源与能力安排计算。我们将在具体设备条件下分别验证，使学习连续性与实际运行能力相匹配。

我们坚持明确的目标与可修正的方法：知道为什么做，也允许证据改变怎样做。

11. 结论：让模型因为经历而持续获得能力

NI 从第一代到下一代，始终围绕持续学习展开。纸片和墙面帮助我们思考经历怎样组织；压缩与遗忘帮助我们思考什么应该被保留；游戏与世界模型帮助我们思考行动、后果与认识怎样相互影响；芯片的通用机制则启发我们思考，怎样面对尚未出现的任务。

我们也已经为这些问题做了实际工作。围绕 NI 配套研发的技术栈，把模型优化、推理运行时、学习机制、工具接入与跨设备状态管理连接起来，使持续学习拥有可实际运行的设备基础。端侧增量学习已经将新增记录、个人特征更新与后续推理连接起来。下一代将沿着这些基础，继续研究更细粒度的更新、经验巩固、持续修正与跨设备学习连续性。

我们追求的，是一个能够在使用中不断形成理解、完善判断、改进行动，并在新证据面前修正自己的模型。记忆参与这一过程，持续获得能力贯穿始终。

设备是智能体的主体，模型是其中能够成长、能够迁移的智能核心，NI 负责让这种成长持续发生。“模型就是产品”表达了我们对这种积累的期待：长期相处形成的认识与能力，能够随着模型进入新的兼容设备，并在新的经历中继续发展。

智能走向数据，也应当意味着成长发生在经历所在的地方。

我们用一个朴素的问题检验这条路：在不反复提醒、不重新交付同一段历史的情况下，模型是否真的因为曾经经历过，而更会面对下一次？

附录 A：术语速查

这些解释服务于阅读，不要求读者先熟悉所有研究领域。

术语	在本文中的含义
Neural Imprint（NI）	质子梯度研发的持续学习算法与算法架构；围绕它配套研发的技术栈，支持模型在设备上运行、学习并持续成长
设备智能体	本文以具备计算能力的设备为主体，结合模型、感知与动作接口和学习状态，组织观察、行动、反馈与学习
持续学习	从连续经历中获取、保持并修正知识与能力，同时处理环境变化与旧能力保持
增量学习	在已有学习成果上吸收新信息，不默认每次从头重学
测试时持续学习（TTCL）	关注模型在部署、使用期间的持续适应与能力巩固，不只是交付前的训练
测试时训练（TTT）	在使用阶段根据学习目标更新可训练的参数；更新可以服务当前输入，也可以按设计向后保留
TTT Layers	将隐藏状态设计为可训练小模型的一类序列处理层，通过使用时的学习更新压缩当前序列信息
模型即产品	产品价值随模型学习而积累，已学认识与能力由完整的模型及其内部学习状态承载，并以可迁移、可继续成长为目标
模型权重（参数）	模型在计算中使用的一组数值；训练可以调整它们，推理时则用它们处理输入
推理运行时	负责安排模型计算、内存使用和状态管理的软件，让模型在具体设备上运行
内部状态	模型计算中可被读取或更新的数值信息；本文关注其中能够承载学习成果并持续参与计算的部分
激活与表示	模型处理输入时形成的内部数值；表示帮助模型组织输入中的内容与关系
潜在表示（latent）	模型内部表达内容与关系的形式；表示保留的信息决定后续能够读取或生成什么
RPP	在模型激活中分析个人相关结构的机制，涉及残差化、主成分分析与稳定性检查
DSR（Dual-Sparse Retention）	双重稀疏 token 驻留。我们的运行时中按注意力得分、近期内容及分层预算管理历史缓存保留的机制
FrogJump	在适配模型与运行模式下，按计划跳过选定计算层的推理执行机制
情景记忆	对具体经历及其情境的保留与回忆，关注当时条件、行动和后果；本文按功能讨论，不预设独立模块
语义记忆	对事实、概念与规律等知识的掌握，使用时无需重现最初获得它的那次经历

术语	在本文中的含义
事实记忆与精确事实回忆	前者泛指事实性内容的保留，可存在于情景或语义记忆中；后者在本文中特指准确回答原始记录细节的任务
适应性推理	结合已有认识与当前条件处理新问题，并依据新证据修正判断；记忆为它提供材料
世界模型	对环境状态及行动后果进行内部预测的模型，可用于推演，并通过真实反馈校正
关联记忆与快速权重	通过线索建立和读取关联，以及让内部映射随经历快速变化的机制
巩固	将新旧经历组织为较稳定、可复用的认识或能力，不限定只能发生在某一种载体中
可塑性与稳定性	接受新信息与保留仍有效认识的能力；二者需要协调
Activation Steering	在推理中调节内部激活，从而影响后续计算；不以修改模型权重为前提
Directional Steering	本文中指我们由纠正产生方向、再影响后续计算的具体路径，属于激活干预的一种实现
泛化与迁移	泛化关注未见输入上的有效性；迁移进一步关注不同任务、环境或载体之间的复用
异构	芯片、运行环境、模型表示、感知方式或动作空间存在差异；讨论时应说明是哪一种差异
Token	模型处理序列的单位，不固定等于一个汉字或一个词
首 token 等待时间	从对应请求开始到产生首个输出 token 的等待，具体计时范围随实验说明

如何引用

质子梯度（AtomGradient）. NI 下一代：持续学习的愿景、哲学与技术路径（白皮书 v2）. 2026-09-17. https://atomgradient.github.io/whitepapers/neural-imprint/

```
@techreport{atomgradient2026ni,
  title  = {NI 下一代：持续学习的愿景、哲学与技术路径},
  author = {{质子梯度 (AtomGradient)}},
  year   = {2026},
  month  = {09},
  type   = {白皮书 v2},
  url    = {https://atomgradient.github.io/whitepapers/neural-imprint/}
}
```

在线阅读：https://atomgradient.github.io/whitepapers/neural-imprint/ · 更新记录：github.com/AtomGradient/whitepapers · 官网：atomgradient.com