# 交互式神经计算机

Ross Murphy 伦敦大学学院r.murphy@cs.ucl.ac.uk
Marcel Hedman 牛津大学marcel.hedman@stats.ox.ac.uk
Gabriel Brostow 伦敦大学学院g.brostow@cs.ucl.ac.uk

arXiv:2611.32811v1 [cs.AI] 24 Nov 2026

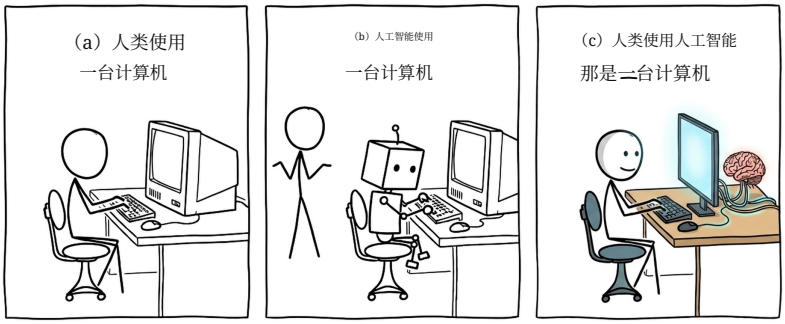



图 1。**从人工智能** 使用 **计算机到人工智能** 成为 **计算机**。 在当今的范式中，人类使用计算机（a）；当前的趋势则是训练 AI 智能体代表用户使用计算机（b）。我们选择了不同的路径（c）：人类使用一个是 计算机的人工智能。

## 摘要

计算机及其界面通常被视为固定不变的，因此人类，或日益增多的 *AI* 智能体，必须适应操作系统所呈现的一切。当用户需要不同的工作流，或适合免提使用或残障人士的界面时，这种做法便会受到限制。我们探索了一种常被忽略的替代方案。我们不再训练智能体操作现有计算机，而是训练生成模型<*strong id='1'*>成为计算机，根据过去的屏幕画面以及用户的鼠标和键盘操作，预测下一屏幕。将桌面视为交互式世界模型，会带来一些自然视频世界模型大多可以避开的问题。由于没有相机位姿来索引记忆，模型必须在缺乏任何几何线索的情况下，记住已经离开屏幕的内容。终端中的算术运算必须完全正确，而不能仅仅做到看似合理。界面中的小字体文本也必须在交互式帧率下保持清晰可读。我们的交互式神经计算机采用类似神经图灵机的结构，通过三个从像素和操作端到端训练的模块，开始解决这些问题。由潜在下一帧预测（MEMORY-JEPA）监督的惊奇门控槽记忆，用于存储屏幕外的信息。探索性查询桥接器引入预训练视觉语言模型的推理能力，而残差渲染器只预测帧与帧之间的潜变量变化。在长时间闭环运行中，我们的模型在 *64.2%* 的试验中成功回忆起被遮挡的文件（随机水平为 *33.3%*），而循环基线和滑动窗口基线的最高准确率仅为 *56.0%*。此外，它还将终端——文本光学字符识别准确率从 *1.3%* 提升至 *50.1%*。最后，由于记忆是显式的，因此可以在测试时将其移植进去，从因果上改变模型所渲染的内容。

## 1. 引言

近期许多关于计算机使用的研究旨在构建能够像人一样操作计算机的智能体，通过读取屏幕、移动光标并从菜单中进行选择，代表用户完成任务 [3, 45, 61]。这种设置存在三个根本问题。首先，用户被排除在循环之外，这种设置旨在取代人类能力，而不是增强人类能力。其次，移除用户后，执行过程难以监督；鉴于已有文献记录了前沿模型中的欺骗行为和伪装对齐行为，这具有很大风险 [20, 30, 37]。第三，这种设置将计算机界面视为有待“完成”的固定环境。实际上，界面是一种任意构造，因此，与其关注智能体能否像在 MiniGrid 中那样“解决”界面 ，不如更多关注人工智能能否生成有用的界面，从而增强人类能力 [15]。归根结底，如果人工智能就是计算机，人机协作就能朝着开放式界面的方向发展。继 Hughes 等人 [31], 之后，我们将一个系统相对于某个观察者称为开放式系统，当它持续产出观察者认为既新颖又可学习的产物时。在这里，观察者是用户，而产物则是能够适应其不断演变的工作流、同时

仍然易于学习的界面，无需任何人重写、重新编译和重新加载静态源代码。

因此，我们着手构建一种生成式交互神经计算机（图 1），使其能够直接从像素与动作轨迹中学习，记忆抽象概念、执行符号运算并渲染清晰的更新，而无需手工设计的符号记忆状态。这三个要求中的每一个，对于世界模型而言都是尚未解决的开放问题；而在计算机上，每个要求又都呈现出特定的形式。第一项是长时程物体恒存。三维场景的世界模型可以利用相机位姿和场景几何来决定将观察结果存储在哪里以及何时检索它 [35, 60],，但桌面不提供这样的索引。窗格中打开的文件会精确覆盖此前文件所在的像素，因此模型必须自行学习存储什么、何时存储以及存储在哪里。第二项是在生成的世界中进行符号推理。在自然视频中，近似的动力学看起来很逼真，但接收 3+3 必须每次打印6 。第三项是渲染。从一帧到下一帧，屏幕的大部分内容都是静态的，这很容易诱使模型简单复制上一帧；然而，确实发生变化的区域包括 10–12px 的文本，而这些文本必须在交互式帧率下保持清晰可读。

为解决这三个问题，我们提出交互式神经计算机（图2），这是一种类似于神经图灵机的端到端架构 [18, 19]，由三个核心模块组成：

**•记忆（惊讶门控概念槽）：** 为了在没有相机位姿或手工设计状态的情况下维持物体恒存性，我们引入了一种以预测编码为基础的无坐标概念槽记忆 [17, 43] 和惊讶驱动的记忆选择（CAMBRIAN-S [65]），其可微写入由潜变量预测意外度和表征空间中的自监督机制（MEMORY-JEPA）进行门控。

**•计算（探索性单遍视觉语言模型查询桥）：** 作为在生成世界内部进行符号推理的初步步骤，我们通过将因果查询严格置于仅用于训练的结果文本之前，扩展了可学习查询桥 [14, 38, 39]，使其不再局限于仅扩散监督，并在单次前向传播中将符号执行状态提取为潜在令牌（$H_{\text{out}t}$），无需在测试时进行文本解码。

**• 接口（残差潜在扩散渲染）：** 为了在交互速度下渲染清晰的更新，同时避免退化为帧复制，我们将两个模块都与残差潜在扩散结合起来 [47, 62]，该方法预测帧与帧之间的潜在增量，而不是在每一步都重新合成整个屏幕，使得在匹配训练预算下，持久终端文本光学字符识别的准确率从采用全帧扩散时的 0.2% 提升至 43.5%。我们的完整模型在 512×384 下达到 50.1%，接近四通道编解码器 56.3% 的上限；使用 16 通道 FLUX.1-schnell 时则达到 88.7% 1024×768处的自编码器。

标准感知指标无法判断世界模型是否记得屏幕之外的内容，因为某一帧看起来可能合理，但其中的内容却是错误的。因此，我们直接评估记忆（第 5.2 节）。在隐藏形状召回任务中，一个包含某种形状的文件被关闭，用户转而在其他位置操作，随后重新打开该文件，因此正确的形状只能来自记忆。更困难的变体会先用一个具有竞争性的文件覆盖同一窗格。由于我们的记忆是一个显式记忆库，而不是纠缠隐藏状态的一部分，我们还可以在测试时移植或擦除它，并检查模型渲染的内容是否随之发生相应变化（第 5.3 节）。

公共 GUI 语料库很少重新访问被遮挡的内容，也很少在屏幕清空后测试符号执行，因此我们构建了两个受控数据集：桌面和数学，相关内容见（第 4 节）。我们将发布完整的五源对齐语料库，包括这两个受控数据集以及我们的转换和对齐流程，同时提供全部训练代码和模型权重。

## 2. 相关工作

**计算机世界模型**。相比于对智能体进行建模，近年来开始出现对机器本身进行建模的研究。Neural Computers [76]将桌面建模为用户与计算机交互的开环生成视频，能够生成引人入胜的开环视频，但不支持闭环用户交互。State- Craft [55] 同样以视频扩散为基础，但为其模型增添了显式的符号记忆。这有助于回忆（物体恒存性），但其设计要求访问人工构建的符号可访问性树状态图。其他同期的 GUI 世界模型依赖外部支架或省略持久记忆：Solaris [1]将状态存储在文本提示中，而 CUWM [23] 和 Yu 等人 [67] 将视觉语言模型或潜变量预测器与独立的解码器配对，却没有持久化学习记忆库。NeuralOS [46],作为这些方法的前身，是唯一一种开放权重、可交互、像素到像素的桌面世界模型，可直接进行闭环推演，且不要求特权符号状态。NeuralOS 只需要像素监督，但将观测历史压缩到单个循环 LSTM 状态中，会导致其在长时间推演中的性能下降。我们的模型与上述所有模型的不同之处在于：我们从像素和操作中端到端地学习可寻址外部记忆，而无需人工构建的状态模式。

**生成式交互式世界模型与记忆**。更广泛地说，动作条件世界模型已经从单向量循环状态空间 [24, 25]发展到用于驾驶 [28]，2二维和三维世界 [12, 40, 41]，以及游戏 [2, 53]的大规模视频生成。在所有这些

领域中，世界模型都面临着这样一种权衡：存储原始历史帧会随着推演长度增加而难以扩展，而将历史压缩为固定大小状态则有覆盖较早信息的风险 [18, 19, 42, 69]。当环境为读取和写入提供外部索引时，固定大小的记忆更易于管理。在三维物理环境中，几何结构可通过相机位姿、视场重叠 [60, 66],三维表面元素 [35], 或射线图 [13, 59]免费提供这一索引。二维界面不具备这样的索引（第1节），因此模型必须学习写入什么、写入哪里以及何时写入。现有的二维世界模型通过保留较短的帧滑动窗口 [2, 53, 69] 或将历史压缩为单个循环向量 [24, 25, 46], 来规避这一索引问题，但这两种方式都会在长时间滚动生成中丢失被遮挡状态。为了在没有三维几何的情况下构建具备索引且容量固定的记忆，我们转而借鉴表征学习和序列建模文献中的三种互补思想。首先，为了在没有空间坐标的情况下决定写入哪里以及何时写入，可微分神经计算机 [18, 19], 、槽位注意力 [36, 72], 和门控增量网络 [22, 33, 48, 63, 64] 等序列模型，会根据内容相似度和可用容量访问离散槽位（哪里）。预测编码模型 [10, 17, 43, 51]和CAMBRIAN-S [65] 则根据新观测是否令人意外来门控记忆写入（何时）。然而，只有当底层表征捕捉到高层语义，而非静态背景像素时，内容寻址和惊讶门控才能发挥作用。至于存储什么，因此我们借鉴联合嵌入预测架构 [4, 5, 9, 71, 74], , 通过潜在下一状态预测（MEMORY-JEPA）对概念槽进行自监督，而不只是进行像素重建。

**生成世界中的符号推理**。仅通过像素扩散学习符号规则和多位数算术 [12, 40, 41] 的样本效率很低。预训练视觉语言模型（VLM） [6, 7]已经具备强大的符号推理能力，尽管在计算机使用研究中，它们通常被训练为操作现有界面的外部智能体 [3, 16, 34, 44, 45, 49, 52, 61, 75]。相反，我们从预训练视觉语言模型出发，在世界模型内部引导符号执行，并监督其查询标记 [14, 38, 39]以及结果文本，而不仅仅是扩散损失（第3.2节）。

## 3. 方法

交互式世界模型预测下一帧视频$x_{t+1} \in \mathbb{R}^{3\times H\times W}$ 基于观测结果 $x_{1:t}$ 和用户操作$a_{1:t}$ （鼠标坐标、点击和按键操作）。遵循第 1 节提出的三个问题，我们将模型拆分为三个模块：记忆、计算和接口（图 2），并以端到端方式联合训练。

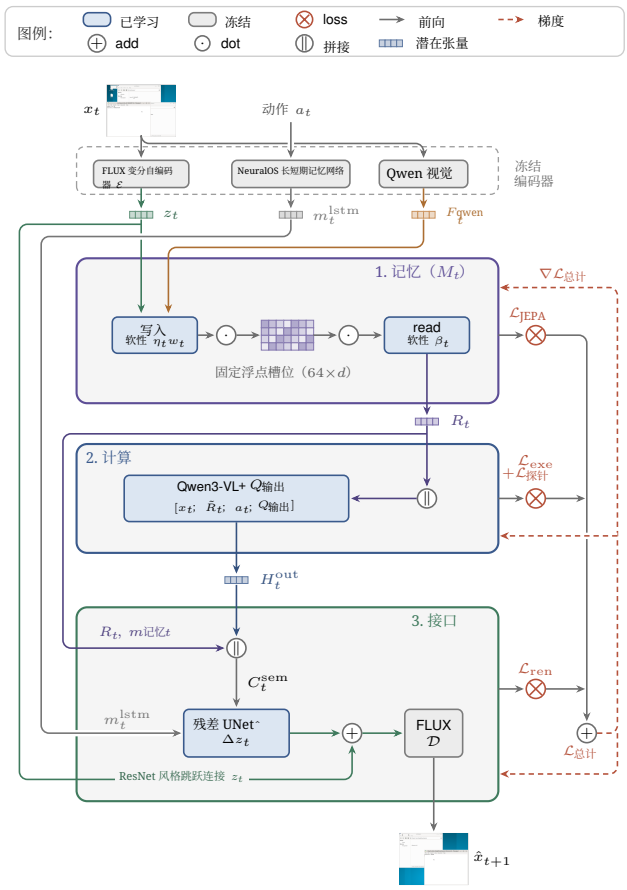



图 2。**端到端交互式神经计算机**。数据通过冻结编码器和三个经过学习的模块垂直流动：**记忆**（$M_t$）、**计算**（$Q_{\text{out}}$）和**接口**（$\hat{}\Delta z_t$）。

### 3.1. 记忆

继神经图灵机 [18] 和可微分神经计算机 [19], 之后，我们在一个显式的固定容量存储库 $M_t$ $=$（$K_t \in \mathbb{R}^{M\times 64}$、$V$语义$_t$ $\in \mathbb{R}^{M\times 512}$、$V$视觉$_t$ $\in \mathbb{R}^{M\times 8\times 128}$、$u_t \in [0,1]^M$）中，将计算与记忆分离；该存储库包含 $M = 64$ 个概念槽（$m \in \{1,\ldots,M\}$），用于存储单位范数键 $K_t$、$_m$、语义向量 $V_{t\,m}^{\text{sem}}$、8 个视觉令牌 $V_{t\,m}^{\text{vis}}$，以及标量占用值 $u_t$ 、 $_m$。记忆组件的分离也为之后实现更强的可解释性和因果干预打开了大门（例如，将捐赠者记忆 $M_t^{\text{donor}}$移植到正在运行的推演中（第 5.3 节））。操作 $M_t$ 归结为四项决策，即写入什么、何时写入、写入何处，以及如何读取（图 2、模块 1；第S1节的图 S1中给出了循环示意图）。

**要写入的内容：内在键与自监督 M记忆-JEPA**。在第 $t$ 步，8 个经过学习的查询通过交叉注意力机制处理冻结的视觉特征 $F_t^{\text{qwen}}$ （Qwen2.5-VL-3B-Instruct [7] 在 512×384, 以及 Qwen3-VL-2B-Instruct [6] 在 1024×768），以及

来自我们冻结的自动编码器（16 通道 FLUX.1-schnell f8 VAE [11]在 1024×768，以及 4 通道 Stable Diffusion KL-变分自编码器在512×384）的潜在帧编码 $z_t = \mathcal{E}(x_t)$，以提取标记 $O_t \in \mathbb{R}^{8\times512}$ 和池化摘要 $s_t \in \mathbb{R}^{512}$。由于桌面帧通常共享静态背景，我们在单位球投影之前减去运行中的 EMA 背景向量 $\mu_s$，从而形成写入键 $k_t = \mathrm{normalize}(W_k(s_t - \mu_s)) \in \mathbb{S}^{63}$；同时结合居中的摘要、动作嵌入 $e(a_t)$ 和 $O_t$，得到槽写入值（$v_t^{\mathrm{sem}}$、$v_t^{\mathrm{vis}}$）。

当我们最初仅使用外部扩散损失训练这些写入和读取头时 $\mathcal{L}_{\mathrm{ren}}$，记忆库未能学会（第5.3节）。依赖记忆的少量像素被静态背景淹没，而损失中没有任何信息告知读头应请求哪个已存储的键，导致回忆能力停留在随机水平。为了解决这一问题，同时无需真实槽标签，我们直接在记忆模块 MEMORY-JEPA [4, 9], 上引入一个明确的自监督目标，该目标训练预测器 $g_\theta$ 仅根据当前屏幕摘要 $x_{t+1}$、用户的动作 $s_t$ $e(a_t)$和记忆读出 $r_t^{\mathrm{sem}}$（公式（4））， 预测真实的下一帧表示

$$\hat{s}_{t+1} = \mathrm{normalize}\big(s_t + g_\theta([s_t;\, e(a_t);\, r_t^{\mathrm{sem}}])\big). \quad (1)$$

为了获得稳定的目标，我们遵循 BYOL [21]和 I- JEPA [4], 将真实的下一帧 $x_{t+1}$ 输入帧编码器的指数移动平均（EMA）副本，并停止梯度（sg），从而生成目标摘要 $s_{t+1}^*$ 和目标键 $k_{t+1}^*$。为什么这会迫使记忆跨越长时遮挡进行学习？假设在第 $t$ 步，用户点击一个选项卡（$a_t$），以重新打开一个在 $\Delta t$步之前关闭的文件。当前屏幕 $x_t$ 和点击操作 $a_t$ 不包含该文件内部所绘内容的任何信息，因此$g_\theta$ 只有在写入头于文件关闭前将其存入$M$，且读头将其读取到 $r_t^{\mathrm{sem}}$中时，才能与$s_{t+1}^*$ 匹配。MEMORY-JEPA 在每一步通过$\mathcal{L}_{\mathrm{JEPA}} = \mathcal{L}_{\mathrm{InfoNCE}}$监督方程（1）：$(\hat{s}_{t+1}, s_{t+1}^*) + \lambda_{\mathrm{align}}\|\hat{s}_{t+1} - s_{t+1}^*\|_2^2 + \lambda_q(1 - \langle q_t^{(1)}, k_{t+1}^*\rangle) + \lambda_{\mathrm{VICReg}}\mathcal{L}_{\mathrm{VICReg}}(k_t) + \lambda_{\mathrm{swap}}\mathcal{L}_{\mathrm{swap}}$。第一项是 InfoNCE 对比损失 [54], , 它将预测 $\hat{s}_{t+1}$ 拉向其自身的目标 $s_{t+1}^*$，并使其远离批次中其他样本的目标；第二项则直接对齐二者。第三项将第一个读取查询 $q_t^{(1)}$（方程（4））与目标键$k_{t+1}^*$对齐，因此无需让梯度穿过遮挡即可监督读取地址。VICReg 项 [8] 保持每个键维度的高方差，并消除各维度之间的相关性，从而防止所有键坍缩为同一个向量。最后， $\mathcal{L}_{\mathrm{swap}}$ 要求当 $M_t$ 替换为另一个回合的记忆时，预测结果应当变差（方程（S1））。

**何时写入：预测编码惊讶门控。** 如果每一帧都无条件地写入，固定数量的（$M$）槽位很快就会耗尽，还可能将模型自行生成的推演噪声写入记忆。我们从预测编码理论 [17, 43] 和由意外度驱动的记忆选择 [65]，中获得启发，并采用一种“预测惊奇度”度量来调节记忆写入。简而言之，如果实际观测到的帧 $x_t$ 与模型对其认为 自己将在第 $t$ 步看到的内容的预期相冲突，即($\hat{s}_t$)，那么我们就称其为“出乎意料”。形式上，预测惊奇度 $\varepsilon_t$ 是 $s_t$ 与 sg($\hat{s}_t$)之间的余弦距离，再除以其运行平均值进行归一化，以形成写入门控 $\eta_t \in$ （0，1），计算如下：

$$\begin{aligned} \varepsilon_t &= 1 - \big\langle \mathrm{normalize}(s_t),\, \mathrm{normalize}(\mathrm{sg}(\hat{s}_t))\big\rangle, \\ \eta_t &= \sigma\Big(\alpha_s\Big(\frac{\varepsilon_t}{\mathrm{EMA}(\varepsilon) + 10^{-5}} - \beta_s\Big)\Big). \end{aligned} \quad (2)$$

可预测的帧会保持 $\eta_t \approx 0$，而无法预测的转变则会开启 $\eta_t \to 1$。由于训练早期 $g_\theta$ 不准确，通过 EMA($\varepsilon$) 对 $\varepsilon_t$ 进行归一化，可确保 $\eta_t$ 仅对相对意外度峰值作出响应；同时，学习得到的逐槽因子$\eta_{t,\,m} = \eta_t\,\sigma(w_g^\top[V_{\mathrm{semt-1},\,m};\;\; v_t^{\mathrm{sem}}] + b_g)$（$b_g$=1.0 在初始化时）可让外层梯度在预热期间调节写入。

写入位置：匹配槽位，否则为第一个空闲槽位。写入操作应更新已经存储该内容的槽位（如果存在），否则占用一个空槽位。我们按照可微分神经计算机（DNC） [19]（第S1节中的公式（S2））计算两个候选目标位置。内容寻址$a_{c,\,m}$ 指向已占用的槽位，其中该槽位的键与 $k_t$最匹配，类似于软最近邻查找；而分配$a_{a,\,m}$ 指向第一个空槽位，因为只要更早的槽位 $j < m$ 仍为空闲，某个槽位就会受到惩罚。一个门控 $\gamma_t = \sigma$（$\kappa$（最大值$_m\, u_{t-1,\,m}\langle K_{t-1,\,m},\; k_t\rangle - \vartheta$））$\in$（0，1）衡量 $k_t$ 与其最佳已占用槽位的匹配程度，并通过 $w_{t,\,m} = \gamma_t\, a_{c,\,m} + (1-\gamma_t)\, a_{a,\,m}$在两者之间进行选择。重新编辑存储文件会产生 $\gamma_t \approx 1$ 并更新其槽位，而新文件会产生 $\gamma_t \approx 0$并占用一个空闲槽位。通过惊奇门控进行缩放后，得到写入强度$e_{t,\,m} = \eta_{t,\,m}\, w_{t,\,m} \in [0,1]$；因此，只有当某个槽位是选定目标且该帧出乎意料时，才会向其中写入内容。随后，每个槽位的键、语义向量和视觉令牌都会按 $e_{t,\,m}$的比例，向（$k_t$、$v_t^{\mathrm{sem}}$、$v_t^{\mathrm{vis}}$）移动；因此，未被寻址的槽位保持不变，而被强烈寻址的槽位则会被覆盖。槽位的占用状态$u_{t,\,m} \in [0,1]$ 会在写入时向1上升，并由遗忘门控 $f_{t,\,m}$ 降低，从而释放内容已被替代的槽位。综合起来，这些机制构成了槽位更新

$$\begin{aligned} K_{t,m} &= \mathrm{clip}_1\big((1 - e_{t,m})K_{t-1,m} + e_{t,m}k_t\big), \\ V_{t,m}^{\mathrm{sem}} &= (1 - e_{t,m})V_{t-1,m}^{\mathrm{sem}} + e_{t,m}v_t^{\mathrm{sem}}, \\ V_{t,m}^{\mathrm{vis}} &= (1 - e_{t,m})V_{t-1,m}^{\mathrm{vis}} + e_{t,m}v_t^{\mathrm{vis}}, \\ u_{t,m} &= \big(u_{t-1,m} + (1 - u_{t-1,m})\, e_{t,m}\big)\,(1 - \eta_t f_{t,m}), \end{aligned} \quad (3)$$

其中 $\mathrm{clip}_1(v) = v/\max(1, \|v\|_2)$使键保持在单位球内，并使空槽位保持零范数；将$f_{t,m}$乘以$\eta_t$后，仅会在出乎意料的步数释放槽位，因此当屏幕静止时，记忆不会衰减。

**阅读方式：动作条件预测检索**。更新 $M_t$ 后，模型必须决定读取哪些信息，以渲染下一帧 $x_{t+1}$。在许多情况下，仅凭当前屏幕 $x_t$ 无法判断接下来需要哪个已存储槽位。我们还需要用户的动作 $a_t$。例如，如果用户正在屏幕上查看文件 B（$x_t$），并点击文件 A 的标签页（$a_t$），正是屏幕与点击的组合告诉模型从记忆中检索文件 A。因此，我们的每个 $H_r = 4$ 读取头都会从居中的屏幕摘要和动作中形成其查询 $q_t^{(h)} = \mathrm{normalize}(\phi_q^{(h)}([s_t - \mu_s; e(a_t)])) \in \mathbb{S}^{63}$，并在已占用槽位上进行关注。当不需要读取屏幕外记忆时，可学习的空操作对数几率 $b_\emptyset$ 允许其不采取行动，而注意力权重则

$$\beta_t^{(h)} = \mathrm{softmax}_{m\in\{1,\dots,M,\emptyset\}}\Big(\Big[\frac{\langle K_{t,m}, q_t^{(h)}\rangle}{\tau_r} + \log(u_{t,m}+\epsilon),\ b_\emptyset\Big]\Big)_{1:M}, \tag{4}$$

其中每个头返回一个语义向量 $r_t^{\mathrm{sem}}$以及 8 个视觉标记（$4\times9 = 36$ 标记 $R_t \in \mathbb{R}^{36\times512}$），每个屏幕图像块都会关注这些标记，从而形成空间图 $m_t^{\mathrm{mem}}$（图2，紫色跳跃连接）。

## 3.2. 计算

我们从预训练视觉语言模型中引导出符号执行能力。仅从像素中学习精确算术的速度极其缓慢，而手工构建的状态图 [55] 无法扩展到开放式应用或通用世界模型。然而，预训练视觉语言模型已经具备符号推理能力。基于此，我们探究视觉语言模型骨干网络能否直接指导我们的界面扩散模块。

问题在于，视觉语言模型无法直接插入扩散模型中使用。若在提示词-˜ ˜前缀状态 $[x_t; R_t; a_t]$（$R_t$: $R_t$ 投影到视觉语言模型的嵌入空间中）上进行注意力计算，却不提供明确的符号监督（$\mathcal{L}_{\mathrm{exe}} = 0$），算术准确率仅为 0.0%（第 S4.3 节）。另一种替代方案是以解码文本 $y_t$作为条件，但这会使外部世界模型的每个状态生成步骤变得非常缓慢，因为我们必须等待视觉语言模型缓慢的测试时自回归解码。

为了在训练期间获得明确的文本监督，同时避免测试时解码的代价，我们基于近期使用可学习查询令牌 [14, 38, 39] 将自回归视觉语言模型连接到扩散解码器的方法（第S1节中的图 S2）展开工作。我们将 $N_{\mathrm{out}} = 16$ 可学习查询令牌 $Q_{\mathrm{out}} \in \mathbb{R}^{16\times2048}$ 插入 Qwen 视觉语言模型的输入序列中，该模型采用冻结的视觉塔和经 LoRA 适配的语言骨干网络（Qwen3-VL-2B-Instruct [6] / Qwen2.5-VL-3B-Instruct [7, 29]），并严格置于多模态前缀与训练目标 $y_t$（即动作结果的文本，例如打印出的答案）之间，从而使序列排列为

$$\underbrace{[x_t;\ \tilde{R}_t;\ a_t]}_{\text{多模态前缀}} \longrightarrow \underbrace{Q_{\mathrm{out}}}_{\text{因果查询}} \longrightarrow \underbrace{y_t}_{\text{训练目标}}. \tag{5}$$

在因果掩码下，$Q_{\mathrm{out}}$ 位于 $y_t$之前，因此其输出状态 $H_t^{\mathrm{out}} \in \mathbb{R}^{16\times2048}$ 无法看到答案（$\partial H_t^{\mathrm{out}}/\partial y_t = 0$），并在测试时通过一次前向传播计算得出。在训练期间，$y_t$ 会向后关注 $Q_{\mathrm{out}}$，因此下一令牌交叉熵（$\mathcal{L}_{\mathrm{exe}}$）和线性探针（$\mathcal{L}_{\mathrm{probe}}$）迫使˜视觉语言模型从 $R_t$中检索已清除变量，执行运算，并将答案存储在 $H_t^{\mathrm{out}}$中。

## 3.3. 接口

一旦**记忆**检索出屏幕外槽位 $R_t$，且**计算**生成结果状态 $H_t^{\mathrm{out}}$后，**接口**模块便通过解决两个子问题，以交互速度渲染 ˆ$x_{t+1}$：仅更新发生变化的区域，同时避免使静态字体排版变得模糊；并将非空间记忆和视觉语言模型令牌路由至二维屏幕坐标。我们没有从头训练低级光标移动和桌面接口渲染，而是直接基于 NeuralOS [46]，的开源版本进行构建；其作者慷慨地发布了文档完善的训练代码和权重，这些权重经过数千个 GPU小时、数百万次桌面状态转移的预训练。我们使用其预训练骨干网络的权重初始化短期动作LSTM（$m_t^{\mathrm{lstm}}$）和基础 UNet 通道，从而继承强大的局部动作到像素先验。

将记忆和计算路由至二维空间。UNet 在二维潜在网格（$96\times128$）上运行，而 $R_t \in \mathbb{R}^{36\times512}$ 和 $H_t^{\mathrm{out}} \in \mathbb{R}^{16\times2048}$ 是用于指定应显示哪个文件或算术结果的一维词元序列，而不是指定应在屏幕的什么位置绘制。我们通过两条路径弥合这一差距（图 2，模块 3；完整的条件关系图见第S1节的图 S3）。首先，为了定位动作发生的位置，我们将携带屏幕坐标的二维映射堆叠在 UNet 输入处，以进行二维卷积。这些映射包括带噪残差 $\Delta z_t^{(k)}$（即 UNet 预测的潜变量变化，详见下文）、短期 LSTM 映射 $m_t^{\mathrm{lstm}}$、（$y$，$x$）光标热图、空间记忆读出 $m_t^{\mathrm{mem}}$，以及受噪声干扰的参考 $\tilde{z}_t$（即 $z_t$ 的带噪副本）。其次，为了让屏幕上的任何区域都能从 $R_t$和 $H_t^{\mathrm{out}}$中查找非局部内容，我们将二者都投影到 $\mathbb{R}^{512}$ 中，形成一个包含 52 个词元的语义令牌库 $C_t^{\mathrm{sem}} = \mathrm{LN}([W_R R_t; W_H H_t^{\mathrm{out}}] + \alpha_{\mathrm{pos}} P_{\mathrm{1D}}) \in \mathbb{R}^{52\times512}$，其中一维正弦编码 $P_{\mathrm{1D}}$ 保留从左到右的字符和数字顺序（例如 842 与 248）。在每个 U-Net 阶段 $\ell \in \{1,\dots,7\}$，$U^{(\ell)}$中的每个二维图像块

都充当一个 **查询**（带有二维坐标 $P_{2D}$），并通过注意力机制关注$C_t^{\text{sem}}$作为**键和值**，从而使每个阶段更新为

$$\tilde{U}^{(\ell)} = U^{(\ell)} + \tanh(g_\ell)\,\mathrm{Proj}_\ell\Big(\mathrm{MHA}_\ell\big(\mathrm{GN}(U^{(\ell)}) + P_{2D},\; C_t^{\text{sem}},\; C_t^{\text{sem}}\big)\Big), \tag{6}$$

检索与其（$y$，$x$）区域相关的内容，并通过可学习门控 $\tanh(g_\ell)$ 将其添加到 $U^{(\ell)}$ 中（$g_0 = 0.25$）。

**残差潜变量预测。** 标准的 4 通道全帧扩散往往会模糊 10–12px 字形等密集特征，同时还会浪费容量来重新生成每一步保持静态的~95% 像素。在两种分辨率下，我们训练一个 $x_0$参数化的残差 UNet [27,47, 62, 68]，仅预测标准化的帧间增量 $\Delta z_t = (z_{t+1} - z_t)/s_c$（$s_c$ 每通道训练标准差）$\hat{}$，并将其添加到跳跃连接 $z_t$ 中，形成 $\hat{}z_{t+1} = z_t + s_c \odot \Delta z_t$（图 2，绿色跳跃连接），在 2–4个去噪扩散隐式模型步数内 [50]（第 5.1节；第S1节中的图 S4）收敛。

第 5 节使用 NeuralOS 原生的四通道 512×384SD KL-VAE 进行计算量匹配的基线比较，并将相同的记忆模块和计算模块（通过无参数的 2×2 潜变量池化）与冻结的 16 通道 1024×768 `FLUX.1-schnell`变分自编码器 [11] 结合，以克服四通道编解码器的56.3%文本可读性上限。

### 3.4. 端到端训练

由于屏幕的大部分区域都是静态的，未加权损失 $\hat{}$ 可能会预测 $\Delta z_t \approx 0$。为解决这一问题，我们将渲染损失定义为 $\mathcal{L}_{\text{ren}} = \frac{\sum_p \omega_p \|\hat{}\Delta z_{t\,p} - \Delta z_{t\,p}\|_2^2}{\sum_p \omega_p} + \lambda_{\text{static}} \frac{\|\hat{}\Delta z_t\|_1}{|\mathcal{S}_t|C}$（第S1节中的公式（S3））。这会通过 $\omega_p = 1 + 4 \cdot \mathbf{1}[\max_c |\Delta z_{t,p,c}| > \delta_{\text{chg}}]$ 提高活跃图像块的权重，同时对 $\mathcal{S}_t$施加静态图像块$L_1$ 安静度惩罚。训练期间，我们还会对两个输入进行扰动，向 $\tilde{z}_t$ [73]添加依赖噪声调度的噪声，并对 $x_t$ 进行掩码（$p_{\text{drop}} = 0.5$，第 S4.4 节），使模型无法简单地复制当前帧，而必须依赖 $\tilde{R}_t$ 和 $H_t^{\text{out}}$。三个模块在 128 步轨迹上联合训练（先进行 64 帧无梯度的预热，再进行 64 帧的截断时间反向传播（时间反向传播）[57, 58]），采用 $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{ren}} + \lambda_{\text{JEPA}}\mathcal{L}_{\text{JEPA}} + \lambda_{\text{exe}}\mathcal{L}_{\text{exe}} + \lambda_{\text{probe}}\mathcal{L}_{\text{probe}}$，沿一条主干网络（图 2，红色虚线）反向传播外部损失，使其进入记忆头和惊奇门控（第 5.3 节和 S4 节）。

## 4. 训练语料与数据混合

训练一个能够作为可用计算机运行的世界模型，需要同时提供以下两类训练示例：(a) 开放世界中的图形用户界面多样性（光标动态、窗口管理和多样化的字体排版），以及 (b) 受控的多步工作流，其中屏幕外的文件或已清除变量能够唯一确定未来的帧。仅依靠公共 GUI 语料库，很少会重新访问被遮挡的状态，或在屏幕清除后测试符号执行。因此，我们选择整理五个独立的数据集，并将其归入三种范式（第 S3 节中的图 S5–S7 和表 S2）：

1. **三个开放世界图形用户界面语料库（AgentNet、ProCUA-SFT 和 CUA-Gym，包含** $>$ 2.1**TB 原始数据、**173，975**条轨迹 /** 8。83**M 步）：**来自 Windows、macOS 和 Ubuntu 的人类录制数据，包括（1）AgentNet [56]（21，957 个清理后的回合），以及来自 Ubuntu 的（2）ProCUA-SFT [32]（92，964 个 Kimi-K2.5 回合）和（3）CUA-Gym（55，139 个 Qwen3.5 虚拟机回合）的滚动生成数据。
2. **受控的 XFCE 文件持久性（桌面，来源 4）：**人工收集的 VS Code 会话（768 个训练回合、96 个评估回合，组成 32 个匹配的三元组），其中编辑一个几何图形（三角形、圆形或星形），在同一窗格中编辑其他文件的同时将其遮挡 13–256 步，然后从像素一致帧中重新打开。
3. **受控 XFCE清屏命令行界面（数学，数据源 5）：**交互式 Python 交互式解释器会话（4，000 个训练回合、300 个评估回合，四类任务），其中变量（q =854）在计算查询（q * 738）之前通过 Ctrl+L 清除。

**数据源合成。**所有 16:9 录制数据都加黑边缩放到 4:3 画布上，并将 pyautogui 脚本解析为抽象语法树而不执行，这使我们能够过滤掉 194 条损坏的间隙/滚动记录，并修复 ProCUA-SFT 中未记录的动作后帧偏移量，使每一步都遵循 $(x_t, a_t) \to x_{t+1}$（第 S3 节）。每一步都会为**计算**生成逐字的击键字符串（保留字符顺序和 Ctrl+L），并为（$y$，$x$）光标热图、点击标志和按键保持向量生成数据，用于**记忆**和**接口**。

**数据混合、训练窗口和公开发布。**我们的五个数据源中有三个已经公开。公开发布时，我们将发布完整的对齐语料库，包括桌面和数学，以及我们的对齐流程、混合清单、训练代码和模型权重。

为了在学术计算预算下全面评估我们的所有架构机制，多随机种子消融实验和长时域滚动均采用 128 步窗口（64 帧预热 + 64-帧时间反向传播），在 *3*，000–9，000 步（48，000–144，000 个 128 步轨迹）上进行训练，数据来自**稠密交互混合数据集**（50% 桌面，50% 数学），并辅以一次高分辨率 1024×768 迁移运行，使用 **BroadMixture**（75。0% 按比例均衡分层的 AgentNet、ProCUA-SFT 和 CUA-Gym，12。5% 桌面，以及 12。5%数学，包含 12，268 个 FLUX.1-schnell 回合 / 325，573 个状态转移，并重采样为 694，266 个窗口（更多详情请参见补充材料中的表 S2 以及第 S4、S6 和 S7 节）。

| （A） 512×384 | 桌面 学习感知图像块相似度 ↓ | 桌面 峰值信噪比 ↑ | 活动命令行界面 学习感知图像块相似度 ↓ | 活动命令行界面 峰值信噪比 ↑ | 文本光学字符识别 (%) ↑ | 261步 峰值信噪比 ↑ | 渲染速度 (帧/秒) ↑ |
|---|---|---|---|---|---|---|---|
| NEURALOS（循环长短期记忆网络）[46] | 0.115 ± 0.002 | 25.51 | 0.128 ± 0.009 | 23.82 | 0.0 | 20.64 | **20.5** |
| WINDOW（8帧扩散） | 0.111 ± 0.002 | 25.70 | 0.094 ± 0.002 | 25.23 | 1.3 | 20.69 | 19.8 |
| **Ours** | **0.107 ± 0.002** | **25.80** | **0.086 ± 0.003** | **25.75** | **50.1 ± 2.0** | **22.09** | 18.2 |

| （B） 1024×768 | 学习感知图像块相似度 ↓ | 弗雷歇起始距离 ↓ | 整体 峰值信噪比 ↑ | 主动编辑 增益（分贝）↑ | 文本光学字符识别 （%） ↑ | 16步 PSNR ↑ | 速度 （帧/秒） ↑ |
|---|---|---|---|---|---|---|---|
| COPY-ONLY（无记忆/计算） | 0.051 ± 0.001 | 66.1 | 29.50 ± 0.23 | +2.73 ± 0.21 | 83.4 ± 6.4 | 17.38 | **21.0** |
| **Ours** | **0.044 ± 0.004** | **56.1** | **31.22 ± 0.18** | **+5.20 ± 0.11** | **88.7 ± 4.6** | **22.34** | 10.5 |

表 1。**视觉保真度、文本可读性和速度**。（A）受控 512×384 套件（SD KL-VAE）。（B） 1024×768 套件（FLUX.1-schnell 16 通道 VAE）与 COPY-ONLY 的对比；后者是没有记忆或计算的我们的渲染器。3 个随机种子上的平均值及 ± 标准差。主动编辑增益是变更像素 PSNR 减去复制上一帧所得的 PSNR。NeuralOS 基于公开发布版本，使用我们的数据进行微调，并遵循其完全相同的训练配置。零样本情况下，它会忽略我们的屏幕并渲染自己的桌面。

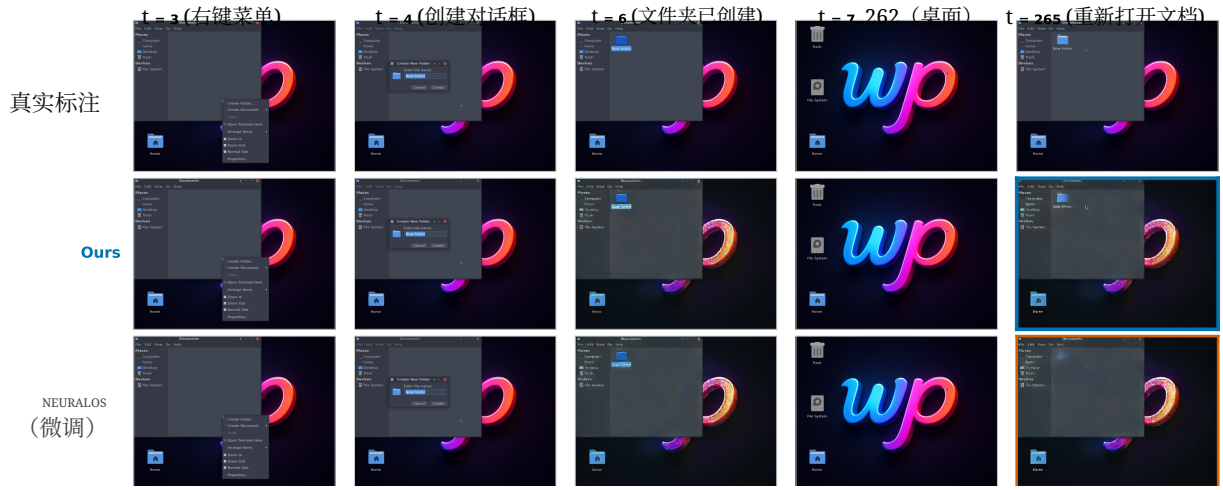



图 3。**在 XFCE 文件管理器任务上的长时段** 512×384 **持久性**。用户在文档中创建“新建文件夹”（$t$=3..6），关闭文件管理器 256 步，然后在 $t$=265 （第 5 列）重新打开文档。**我们的方法**保留了“新建文件夹”图标，而微调后的 **NEURALOS** 则绘制出一个空窗格。请注意，当文件夹显示在屏幕上时，两个模型的结果一致；只有在文件夹被隐藏 256 步后，它们才出现分歧。

## 5. 实验与结果

我们向每个模型提出三个问题。它是否是一个合格的渲染器（第 5.1 节）？它是否记得屏幕外状态（第 5.2 节）？槽位记忆是否是驱动这种回忆的因素（第 5.3 节）？

我们采用两种评估设置（第S1节）：一种是与 NEURALOS [46]在编解码器和分辨率上匹配的**受控** 512×384 **套件** （SD KL-VAE +Qwen2.5-VL-3B），用于进行公平的 3 个随机种子比较；另一种是**高分辨率** 1024×768**套件** （FLUX.1-schnell 16 通道 VAE + Qwen3-VL-2B）。我们不跨分辨率比较模型。

### 5.1. 视觉保真度

表 1比较了**我们的方法**与**NEURALOS** [46], **WIN- DOW**（8帧滑动窗口扩散 [2, 53]），针对512×384 设置。在 1024×768中，我们将其与**COPY-仅有**我们的方法的残差渲染器，不使用记忆或计算，并在相同的混合数据集上进行训练。

在 512×384 ，三个模型在桌面 LPIPS 上相近（0。107 对比 0。111 和 0。115），而**我们的方法**在内容发生变化或必须保持不变的场景中领先：活动命令行界面过渡（0。086 对比 0。094 和 0。128 LPIPS）以及持久终端文本（50。1% 对比 1。3% 和 0。0 % 光学字符识别）。

在 1024×768，与 COPY-ONLY 的比较分离出了记忆和计算相对于其他方面完全相同的渲染器所带来的增益。加入二者后，所有保真度指标均得到改善，其中 16 步展开 PSNR（22。34 对比 17。38 分贝）和主动编辑增益（+5.20 对比 +2.73 分贝）的提升最为明显。

### 5.2. 我们的学习型记忆模块能够应对遮挡和覆盖

我们通过隐藏形状回忆直接测试物体恒存性：在 32 组三元 VS Code 配对中（96 个回合 ×3随机种子 = 288 评估），一个包含三角形、圆形或星形的文件已关闭，

| 模型（512×384） | 已编辑文件召回率（%）↑ | Δ$t$步数的同窗格覆盖（%）↑ | | | 捐赠者记忆交换（%）↑ |
|---|---|---|---|---|---|
| | | Δ$t$=32 | Δ$t$=64 | Δ$t$=128 | |
| NEURALOS（循环长短期记忆网络） [46] | 56.0 | 36.8 | 36.5 | 38.2 | – |
| WINDOW（8帧扩散） | 49.6 | 37.8 | 34.0 | 39.9 | – |
| 坐标槽（（$y$, $x$）寻址令牌库） | 51.9 | 38.9 | 39.6 | 37.5 | – |
| **我们的方法**（按内容寻址的槽位） | **64.2** | **46.2** | **50.0** | **42.4** | **64.7 ± 1.1** |

表2。**屏幕外记忆：三路隐藏形状准确率（33.3% 随机水平）**。编辑文件召回率 在遮挡后重新打开已编辑文件（3 个随机种子）。在覆盖中，首先在同一窗格中编辑一个竞争文件，持续 Δ$t$ 步（每项288次评估）。供体交换 统计记忆移植后呈现供体形状的试验次数。粗体表示每列中的最佳结果。

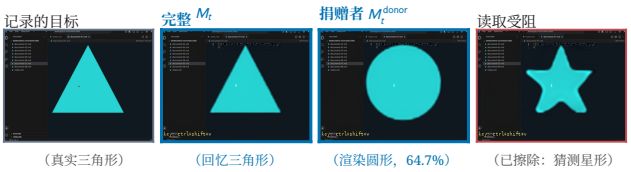



图4。**测试时记忆移植与擦除**。在$x_t$、 $a_t$和长短期记忆网络固定不变的情况下，保留 $M_t$ 会召回真实的三角形，移植 $M_t^{\text{donor}}$会呈现供体的圆形（64.7%），而阻断读取会擦除文件，使模型只能猜测（此处为星形）。由于各面板中的屏幕、点击和 LSTM 状态都完全相同，形状的变化只能来自槽位记忆。

智能体会从已关闭的文件中移开（并在屏幕上执行干扰任务），随后最终返回该文件。此时，屏幕上没有任何信息说明应绘制哪种形状，因此答案只能来自记忆。当渲染帧在 $L_1$ 中更接近真实形状，而不是两个干扰项之一时，即视为回忆正确（33.3% 随机水平，边距为 $\Delta L_1 = L_1^{\text{foil}} - L_1^{\text{true}}$）。

该测试的结果见表2第1列的“编辑文件召回率”部分。编辑文件在遮挡后重新打开时，**我们的方法** 在64.2%的回合中召回了正确的形状，而49.6–56.0%的结果来自基线方法。

更严苛的测试是覆盖：在文件被重新打开之前，另一个竞争文件（具有不同的形状或文本）在同一 VS Code 窗格中被编辑了 Δ$t$ 步，因此需要记忆的内容与干扰项占据完全相同的像素。此时，WINDOW 和 NEURALOS 的表现都接近随机水平（34.0–39.9%）。COORDINATE SLOTS 使用同一个由 $(y, x)$ 位置而非内容寻址的槽位库，表现也好不到哪里去（37.5–39.6%），因为同一窗格中的两个文件可能共享同一个地址。**我们的方法**在 Δ$t$=64 (Δ$L_1$ =+0.070，95% CI $[+0.046, +0.095]$) 时达到 50.0%。在Δ$t$=128 时，领先优势缩小至 42.4%（因为所有模型在固定记忆缓冲区下经过足够长的时间后，最终都会趋近于随机水平）。

图3直观展示了 NEURALOS 在原生 XFCE 文件管理器任务中的效果。一个隐藏了 256 步的文件夹，在由**我们的方法**重新打开 Documents 时仍然存在，而微调后的 NEURALOS 则绘制出一个空窗格。

### 5.3. 记忆具有因果性且可编辑

高召回率本身并不能证明习得的记忆模块发挥了作用。例如，形状等抽象概念可能由长短期记忆网络或渲染器承载。然而，由于我们的记忆是一个显式令牌库 $M_t$，而不是纠缠潜在状态的一部分，因此我们可以直接进行测试。我们固定屏幕 $x_t$、动作 $a_t$和长短期记忆网络（隐藏状态），只改变 $M_t$ （图4）。在 $M_t$ 保持不变时，模型会回忆起真实的形状（三角形）。当我们用一个供体的记忆替换原记忆时 $M_t \leftarrow M_t^{\text{donor}}$，该供体来自绘制过圆形的回合，模型会在64.7 ± 1.1% 的试验中渲染出供体的圆形（表2）；而当我们阻断记忆读取时，存储文件会被擦除。因此，模型重新打开文件时所绘制的内容由其槽位中的内容决定；主管可以检查、替换这些槽位，甚至可能在不重置模型其余潜在状态的情况下将其全部擦除。

**消融实验**。 在 3,000 步时进行消融实验（S4 节），移除 MEMORY-JEPA（$\mathcal{L}_{\text{JEPA}}$=0）会使已编辑文件的召回率降至随机水平（32.1%），而移除残差预测会使终端 OCR 从 43.5% 降至 0.2%（$p$=0.0007）。有关我们模型每个组件的完整消融实验，请参阅补充材料。

## 6. 结论

我们引入了一个计算机世界模型，其中包含新颖的意外度门控 记忆组件。我们的模型在编辑区域保真度、文本可读性以及抵抗覆盖方面均优于基线模型。它学会了如何与模块化、可拆卸的记忆状态交互，并且比通过学习得到的替代方案更具功能性（更实用）。

由于学术计算资源限制，我们仅使用完整混合数据集的一小部分进行训练，而性能尚未达到饱和，因此扩大训练规模是明确的下一步。

未来工作的另一个令人兴奋的方向，是探索开放式界面生成：根据用户的目标、能力和环境，在实时生成能够增强人类能力的应用，并围绕用户呈现。

## 参考资料

[1] Yuval Alaluf、Omri Avrahami、Guy Bukchin Leshem、Michal Geyer、Kfir Goldberg、Elad Richardson 等。Solaris：迈向生成式而非编码式的界面，2026 年。2[2] Eloi Alonso、Adam Jelley、Vincent Micheli、Anssi Kanervisto、Amos Storkey、Tim Pearce 和 François Fleuret。用于世界建模的扩散：视觉细节在 Atari 中至关重要。载于《神经信息处理系统进展》（NeurIPS），2024 年。2、3、7[3] Anthropic。开发计算机使用模型。
h
t
t
ps://
www.anthropic.com/news/3-5-models-and-computer-use，2024 年。1、3[4] Mahmoud Assran、Quentin Duval、Ishan Misra、Piotr Bojanowski、Pascal Vincent、Michael Rabbat、Yann LeCun 和 Nicolas Ballas。基于联合嵌入预测架构的图像自监督学习。载于 IEEE/CVF计算机视觉与模式识别会议（CVPR），2023 年。3、4[5] Mahmoud Assran、Adrien Bardes、David Fan 等。V-JEPA 2：自监督视频模型支持理解、预测和规划。arXiv 预印本 arXiv:2506 09985，2025 年 3。。[6] Shuai Bai、Yuxuan Cai、Ruizhe Chen、Keqin Chen、Xionghui Chen、Zesen Cheng、Lianghao Deng、Wei Ding、Chang Gao、Chunjiang Ge 等。Qwen3-VL 技术报告，2025 年。3、5、14[7] Shuai Bai、Keqin Chen、Xuejing Liu、Jialin Wang、Wenbin Ge、Sibo Song、Kai Dang、Peng Wang、Shijie Wang、Jun Tang 等。Qwen2.5-VL 技术报告，2025 年。3、5[8] Adrien Bardes、Jean Ponce 和 Yann LeCun。VI-CReg：用于自监督学习的方差-不变性-协方差正则化。载于《国际学习表征会议》（ICLR），2022 年。4、13[9] Adrien Bardes、Quentin Garrido、Jean Ponce、Xinlei Chen、Michael Rabbat、Yann LeCun、Mahmoud Assran 和 Nicolas Ballas。重新审视用于从视频中学习视觉表征的特征预测。《机器学习研究汇刊》（TMLR），2024 年。3、4[10] Ali Behrouz、Peilin Zhong 和 Vahab Mirrokni。Titans：学习在测试时进行记忆。载于《神经信息处理系统进展》（NeurIPS），2025 年。3[11] Black Forest Labs。FLUX。https://github.com/black-forest-labs/flux，2024 年。4、6、15[12] Jake Bruce、Michael Dennis、Ashley Edwards、Jack Parker-Holder、Yuge Shi、Edward Hughes、Matthew Lai、Aditi Mavalankar、Richie Steigerwald、Chris Apps、Yusuf Aytar、Sarah Bechtle、Feryal Behbahani、Stephanie Chan、Nicolas Heess、Lucy Gonzalez、Simon Osindero、Sherjil Ozair、Scott Reed、Jingwei Zhang、Konrad Zolna、Jeff Clune、Nando de Freitas、Satinder Singh 和 Tim Rockt¨aschel。Genie：生成式交互环境。载于《国际机器学习会议》（ICML），2024 年。2、3[13] Eren Cetin、Julius Erbach、Vukasin Bozic、Hyojun Go、Zixiang Zhao、Nikolai Kalischek、Federico Tombari、Konrad Schindler 和 Dominik Narnhofer。借助区域令牌寻址槽位记忆的长时程视频生成，2026 年。3

[14] Jiuhai Chen、Zhiyang Xu、Xichen Pan、Yushi Hu、Can Qin、Tom Goldstein、Lifu Huang、Tianyi Zhou、Saining Xie、Sil- vio Savarese、Le Xue、Caiming Xiong 和 Ran Xu。《BLIP3-o：完全开放的统一多模态模型系列——架构、训练与数据集》，2025 年。2、3、5[15] Maxime Chevalier-Boisvert、Bolun Dai、Mark Towers、Ro- drigo de Lazcano、Lucas Willems、Salem Lahlou、Suman Pal、Pablo Samuel Castro 和 Jordan Terry。《MiniGrid 与 Miniworld：面向目标导向任务的模块化与可定制强化学习环境》。载于《神经信息处理系统进展》（NeurIPS）数据集与基准赛道，2023 年。1[16] Xiang Deng、Yu Gu、Boyuan Zheng、Shijie Chen、Samuel Stevens、Boshi Wang、Huan Sun 和 Yu Su。《Mind2Web：迈向面向网络的通用智能体》。载于《神经信息处理系统进展》（NeurIPS）数据集与基准赛道，2023 年。3[17] Karl Friston。《自由能原理：统一的大脑理论？》《自然·神经科学评论》，11(2):127–138，2010 年。2、3、4、13、17[18] Alex Graves、Greg Wayne 和 Ivo Danihelka。《神经图灵机》。arXiv 预印本 arXiv:1410.5401，2014 年。2、3[19]Alex Graves、Greg Wayne、Malcolm Reynolds、Tim Harley、Ivo Danihelka、Agnieszka Grabska-Barwi´nska、Ser- gio G´omez Colmenarejo、Edward Grefenstette、Tiago Ra- malho、John Agapiou 等。《使用带有动态外部记忆的神经网络进行混合计算》。《自然》，538(7626):471–476，2016 年。2、3、4、13[20] Ryan Greenblatt、Carson Denison、Benjamin Wright、Fabien Roger 等。《大语言模型中的对齐伪装》，2024 年。1[21] Jean-Bastien Grill、Florian Strub、Florent Altch´e、Corentin Tallec、Pierre Richemond、Elena Buchatskaya、Carl Doersch、Bernardo Avila Pires、Zhaohan Guo、Mohammad Ghesh- laghi Azar、Bilal Piot、Koray Kavukcuoglu、R´emi Munos 和 Michal Valko。《自举自身潜变量：一种新的自监督学习方法》。载于《神经信息处理系统进展》（NeurIPS），第 21271–21284 页，2020 年。4[22] Albert Gu 和 Tri Dao。《Mamba：使用选择性状态空间进行线性时间序列建模》。载于《语言建模会议》（COLM），2024 年。3[23] Yiming Guan、Rui Yu、John Zhang、Lu Wang、Chaoyun Zhang、Liqun Li、Bo Qiao、Si Qin、He Huang、Fangkai Yang 等。《计算机使用世界模型》。载于《国际学习表征会议》（ICLR），2026 年。2[24] David Ha 和 J¨urgen Schmidhuber。《循环世界模型促进策略演化》。载于《神经信息处理系统进展》（NeurIPS），2018 年。2、3[25] Danijar Hafner、Jurgis Pasukonis、Jimmy Ba 和 Timothy Lillicrap。《通过世界模型掌握多样化领域》。arXiv 预印本 arXiv:2301.04104，2023 年。2、3[26]Martin Heusel、Hubert Ramsauer、Thomas Unterthiner、Bernhard Nessler 和 Sepp Hochreiter。《通过双时间尺度更新规则训练的生成对抗网络收敛于局部纳什均衡》。载于《神经信息处理系统进展》（NIPS），2017 年。16

[27] Jonathan Ho、Ajay Jain 和 Pieter Abbeel。《去噪扩散概率模型》。载于《神经信息处理系统进展》（NeurIPS），2020 年。6[28]Anthony Hu、Lloyd Russell、Hudson Yeo、Zak Murez、George Fedoseev、Alex Kendall、Jamie Shotton 和 Gian- luca Corrado。《GAIA-1：用于自动驾驶的生成式世界模型》。arXiv 预印本 arXiv:2309.17080，2023 年。2[29] Edward J. Hu、Yelong Shen、Phillip Wallis、Zey uan Allen- Zhu、Yuanzhi Li、Shean Wang、Lu Wang 和 Weizhu Chen。《LoRA：大语言模型的低秩适配》。载于《国际学习表征会议》（ICLR），2022 年。5、14[30] Evan Hubinger、Carson Denison、Jesse Mu、Mike Lambert 等。《潜伏智能体：训练能够持续通过安全性训练的欺骗性大语言模型》，2024 年。1[31]Edward Hughes、Michael Dennis、Jack Parker-Holder、Feryal Behbahani、Aditi Mavalankar、Yuge Shi、Tom Schaul 和 Tim Rockt¨aschel。《开放式性对于人工超人智能至关重要》。arXiv 预印本 arXiv:2406.04268，2024 年。1[32] Jaehun Jung、Ximing Lu、Brandon Cui、Muhammad Khalifa、Shaokun Zhang、Hao Zhang、Jin Xu、Amala Sanjay Desh- mukh、Karan Sapra、Andrew Tao、Yejin Choi、Jan Kautz、Mingjie Liu 和 Yi Dong。《ProCUA-SFT 技术报告》，2026 年。6、18[33] Angelos Katharopoulos、Apoorv Vyas、Nikolaos Pappas 和 Franc¸ois Fleuret。《Transformer 就是循环神经网络：具有线性注意力的快速自回归 Transformer》。载于《机器学习国际会议论文集》（ICML），第 5156–5165 页，2020 年。3[34] Kenton Lee、Mandar Joshi、Iulia Turc、Hexiang Hu、Fangyu Liu、Julian Eisenschlos、Urvashi Khandelwal、Peter Shaw、Ming-Wei Chang 和 Kristina Toutanova。《Pix2struct：将屏幕截图解析作为视觉语言理解的预训练》。载于《国际机器学习会议》（ICML），2023 年。3[35] Runjia Li、Philip Torr、Andrea Vedaldi 和 Tomas Jakab。《VMem：使用基于曲面元素索引的视图记忆进行一致的交互式视频场景生成》。载于《IEEE/CVF 国际计算机视觉会议论文集》（ICCV），2025 年。2、3[36]Francesco Locatello、Dirk Weissenborn、Thomas Un- terthiner、Aravindh Mahendran、Georg Heigold、Jakob Uszkoreit、Alexey Dosovitskiy 和 Thomas Kipf。《使用槽位注意力进行以对象为中心的学习》。载于《神经信息处理系统进展》（NeurIPS），2020 年。3[37] Alexander Meinke、Bronson Schoen、J´er´emy Scheurer、Mikita Balesni、Rusheb Shah 和 Marius Hobbhahn。《前沿模型能够进行上下文内谋划》，2024 年。1[38] Xichen Pan、Satya Narayan Shukla、Aashu Singh、Zhuokai Zhao、Shlok Kumar Mishra、Jialiang Wang、Zhiyang Xu、Ji- uhai Chen、Kunpeng Li、Felix Juefei-Xu、Ji Hou 和 Saining Xie。《通过 MetaQueries 实现模态间迁移》，2025 年。2、3、5[39] PAN 团队、Zihan Liu、Yi Gu、Mingkai Deng、Guangyi Liu、Zeyu Feng、Qiyue Gao、Yiyan Hu、Benhao Huang、Yichi Yang、Kun Zhou、Jiannan Xiang、Zhiting Hu、Zhengzhong Liu 和 Eric P. Xing。《PAN：用于通用、可操作且长时域世界模拟的世界模型》，2025 年。2、3、5[40] Jack Parker-Holder 和 Shlomi Fruchter。《Genie 3：世界模型的新前沿》。Google DeepMind 博客，2025 年。2、3[41]Jack Parker-Holder、Philip Ball、Jake Bruce、Vibhavari Dasagi、Kristian Holsheimer、Christos Kaplanis、Alexandre Moufarek、Guy Scully、Jeremy Shar、Jimmy Shi 等。《Genie 2：大规模基础世界模型》。Google DeepMind 博客，2024 年。2、3[42] Ryan Po、Yotam Nitzan、Richard Zhang、Berlin Chen、Tri Dao、Eli Shechtman、Gordon Wetzstein、Xun Huang。《长上下文状态空间视频世界模型》。载于《IEEE/CVF国际计算机视觉会议（ICCV）会议录》，2025 年。3[43] Rajesh P. N. Rao 和 Dana H. Ballard。《视觉皮层中的预测编码：对某些经典外感受野效应的功能性解释》。《自然·神经科学》，2(1)：79–87，1999 年。2、3、4、13、17[44] Christopher Rawles、Alice Li、Daniel Rodriguez、Oriana Riva 和 Timothy Lillicrap。《AndroidInTheWild：用于安卓设备控制的大规模数据集》。载于《神经信息处理系统进展（NeurIPS）数据集与基准赛道》，2023 年。3[45] Christopher Rawles、Sarah Clincke maillie、Yifan Chang、Jonathan Waltz、Gabrielle Lau、Marybeth Fair、Alice Li、William Bishop、Wei Li、Folawiyo Campbell-Ajala、Daniel Riva、Timothy Lillicrap、Kris Toyama 和 Oriana Riva。《Android World：面向自主智能体的动态基准测试环境》。arXiv 预印本 arXiv:2405.14573，2024 年。1 、 3[46] Luke Rivard 、 Sun Sun、Hongyu Guo、Wenhu Chen 和 Yuntian Deng。《NeuralOS：通过神经生成模型模拟操作系统》。载于《国际学习表征会议（ICLR）》，2026 年。2、3、5、7、8、14、16、19[47] Mohammadreza Salehi、Mehdi Noroozi、Luca Morreale、Ruchika Chavhan、Malcolm Chadwick、Alberto Gil Ramos 和 Abhinav Mehrotra。《RFDM：用于高效因果视频编辑的残差流扩散模型》。载于《IEEE/CVF计算机视觉与模式识别会议（CVPR）会议录》，2026 年。2、6[48] Imanol Schlag、Kazuki Irie 和 J¨urgen Schmidhuber。《线性变换器实际上是快速权重编程器》。载于《第38届国际机器学习会议（ICML）会议录》，2021 年。3[49] Peter Shaw、Mandar Joshi、James Cohan、Jonathan Berant、Panupong Pasupat、Hexiang Hu、Urvashi Khandelwal、Kenton Lee 和 Kristina Toutanova。《从像素到用户界面操作：通过图形用户界面学习遵循指令》。载于《神经信息处理系统进展（NeurIPS）》，2023 年。3[50] Jiaming Song、Chenlin Meng 和 Stefano Ermon。《去噪扩散隐式模型》。载于《国际学习表征会议（ICLR）》，2021 年。6、15[51] Yu Sun、Xinhao Li、Karan Dalal、Jiarui Xu、Arjun Vikram、Genghan Zhang、Yann Dubois、Xinlei Chen、Xiaolong Wang、

Sanmi Koyejo、Tatsunori Hashimoto 和 Carlos Guestrin。《学习如何学习（在测试时学习）：具有表达性隐藏状态的循环神经网络》。载于《第42届国际机器学习会议（ICML）会议录》，2025 年，第57503–57522页。3 – 。[52] Daniel Toyama、Philippe Hamel、Anita Gergely、Gheorghe Comanici、Amelia Glaese、Zafarali Ahmed、Tyler Jackson、Shibl Mourad 和 Doina Precup。《AndroidEnv：面向安卓的强化学习平台》。arXiv 预印本 arXiv:2105.13231，2021 年。3[53] Dani Valevski、Yaniv Leviathan、Moab Arar 和 Shlomi Fruchter。《扩散模型是实时游戏引擎》。载于《国际学习表征会议（ICLR）》，2025 年。2、3、7[54] A¨aron van den Oord、Yazhe Li 和 Oriol Vinyals。《通过对比预测编码进行表征学习》，2018 年。4[55] Lorenzo Venturoli、Nedko Savov、Xi Wang、Marc Pollefeys 和 Luc Van Gool。《Statecraft：利用抽象状态推理增强视频世界模型》，2026 年。2、5[56] Xinyuan Wang、Bowen Wang、Dunjie Lu、Junlin Yang、Tianbao Xie、Junli Wang、Jiaqi Deng、Xiaole Guo、Yiheng Xu、Chen Henry Wu 等。《OpenCUA：计算机使用代理的开放基础》，2025 年。6、18、19[57] Paul J. Werbos。《时间反向传播：其作用及实现方法》。《IEEE会议录》，78(10)：1550–1560，1990 年。6[58] Ronald J. Williams 和 Jing Peng。《一种用于在线训练循环网络轨迹的高效基于梯度的算法》。《神经计算》，2(4)：490–501，1990 年。6[59] Junyi Wu、Zekun Li、Chenhongyi Yang、Chen Guo、Lingchen Yang、Yongan Zhang、Xinyi Zhang、Hongbing Hu、Bartek Rymkowski、Srinath Sridhar、Yan Yan 和 Xijie Huang。《World-T3：将测试时训练作为实时视频世界模型的持久记忆》，2026 年。3[60] Zeqi Xiao、Yushi Lan、Yifan Zhou、Wenqi Ouyang、Shuai Yang、Yanhong Zeng 和 Xingang Pan。《WorldMem：利用记忆实现长期一致的世界模拟》。载于《神经信息处理系统进展（NeurIPS）》，2025 年。2、3[61] Tianbao Xie、Danyang Zhang、Jixuan Chen、Xiaochuan Li、Siheng Zhao、Ruisheng Cao、Toh Jing Hua、Z houjun Cheng、Dongchan Shin、Fangyu Lei、Yitao Liu、Yiheng Xu、Shuyan Zhou、Silvio Savarese、Caiming Xiong、Victor Zhong 和 Tao Yu。《OSWorld：真实计算机环境中面向开放式任务的多模态智能体基准测试》。载于《神经信息处理系统进展（NeurIPS）数据集与基准赛道》，2024 年。1、3[62] Ruihan Yang、Prakhar Srivastava 和 Stephan Mandt。《用于视频生成的扩散概率建模》。《Entropy》，25(10)：1469，2023 年。2、6[63] Songlin Yang、Bailin Wang、Yikang Shen、Rameswar Panda 和 Yoon Kim。《具有硬件高效训练的门控线性注意力变换器》。载于《第41届国际机器学习会议（ICML）会议录》，2024 年，第56501–56523页。3

[64] Songlin Yang、Jan Kautz 和 Ali Hatamizadeh。门控增量网络：使用增量规则改进 Mamba2。载于国际学习表征会议（ICLR），2025 年。3[65] Shusheng Yang、Jihan Yang、Pinzhi Huang、Ellis Brown、Zihao Yang、Yue Yu、Shengbang Tong、Zihan Zheng、Yifan Xu、Muhan Wang、Rob Fergus、Yann LeCun、Li Fei-Fei 和 Saining Xie。Cambrian-S：迈向视频中的空间超感知，2025 年。2、3、4、13、17[66] Jiwen Yu、Jianhong Bai、Yiran Qin、Quande Liu、Xintao Wang、Pengfei Wan、Di Zhang 和 Xihui Liu。上下文即记忆：通过记忆检索实现场景一致的交互式长视频生成。载于《亚洲计算机图形学国际会议 2025 会议论文集》，2025 年。3[67] Seungjun Yu、Hojun Choi、Seojeong Park、Jaeyo Shin 和 Hyunjung Shim。面向图形用户界面智能体的潜在世界动作模型，2026 年。2[68] Zongsheng Yue、Jianyi Wang 和 Chen Change Loy。ResShift：通过残差移位实现高效的图像超分辨率扩散模型。载于《神经信息处理系统进展》（NeurIPS），2023 年。6[69] Lvmin Zhang、Shengqu Cai、Muyang Li、Gordon Wetzstein 和 Maneesh Agrawala。下一帧预测视频扩散模型中的帧上下文打包与漂移防止。载于《神经信息处理系统进展》（NeurIPS），2025 年。3[70] Richard Zhang、Phillip Isola、Alexei A. Efros、Eli Shechtman 和 Oliver Wang。深层特征作为感知度量的惊人有效性。载于 IEEE 计算机视觉与模式识别会议论文集（CVPR），2018 年。16、19[71] Xinyu Zhang、Zhengtong Xu、Yutian Tao、Yeping Wang、Yu She 和 Abdeslam Boularias。通过残差潜在动作学习基于视觉特征的世界模型。载于《神经信息处理系统进展》（NeurIPS），2026 年。3[72] Yu Zhang、Songlin Yang、Ruijie Zhu、Yue Zhang、Leyang Cui、Yiqiao Wang、Bolun Wang、Freda Shi、Bailin Wang、Wei Bi、Peng Zhou 和 Guohong Fu。用于高效线性时间序列建模的门控槽位注意力。载于《神经信息处理系统进展》（NeurIPS），2024 年。3[73] Min Zhao、Hongzhou Zhu、Chendong Xiang、Kaiwen Zheng、Chongxuan Li 和 Jun Zhu。识别并解决图像到视频扩散模型中的条件图像泄漏问题。载于《神经信息处理系统进展》（NeurIPS），2024 年。6[74] Gaoyue Zhou、Hengkai Pan、Yann LeCun 和 Lerrel Pinto。DINO-WM：基于预训练视觉特征的世界模型实现零样本规划。载于第 42 届国际机器学习会议（ICML）论文集，2025 年。3[75] Shuyan Zhou、Frank F. Xu、Hao Zhu、Xuhui Zhou、Robert Lo、Abishek Sridhar、Xianyi Cheng、Tianyue Ou、Yonatan Bisk、Daniel Fried、Uri Alon 和 Graham Neubig。WebArena：用于构建自主智能体的逼真网络环境。载于国际学习表征会议（ICLR），2024 年。3[76] Mingchen Zhuge、Jürgen Schmidhuber 等。神经计算机，2026 年。2

# 交互式神经计算机

## 补充材料

## S1. 方法方程、图示与评估协议

这篇论文的主要问题是：生成式世界模型能否成为一台可用的计算机，而不仅仅是渲染出合理的帧。为了实现这一目标，我们首先需要克服三个关键问题：

1. 由于缺乏可依赖的三维相机位姿，计算机的世界模型会遗忘屏幕外状态，即使在较短的时间跨度内也是如此。
2. 世界模型通常往往难以进行明确的符号推理，但这种问题在计算机世界模型中尤为明显。基线模型经常会在屏幕上生成看似合理、实则明显不合逻辑的文本。

3. 由于桌面的 ∼95% 可能从一帧到下一帧保持不变（例如终端 shell），如果不对渲染损失加权而直接扩散下一屏幕状态，可能会导致模型退化为简单地复制前一帧。

我们的交互式神经计算机（主图 2）通过将系统拆分为三个端到端训练的模块，依次解决这三项问题：**记忆、计算**和**接口**。我们为每个模块提供有针对性的监督，而不是仅依靠像素扩散来教会这三个模块的全部功能。

本节将“打开”论文主文图 2 的内部结构，并阐释主文第 3 节和第 5 节所介绍架构背后的直觉、精确方程及评估协议。

我们将细节分为四个部分：为什么每个训练目标和寻址规则都采用当前形式（第S1.1节），架构如何保持紧凑并扩展到 1024×768（第S1.2节），如何在单步图像指标之外测试功能实用性（第S1.3节），以及如何在有限的 GPU 内存下训练长时程记忆（第 S1.4 节）。

### S1.1. 训练目标与槽位寻址

损失函数的每个组成部分，都针对我们在端到端训练这种神经计算机时遇到的一种具体失败模式。

**为什么普通世界模型——或仅靠外部渲染损失——无法学习长时程记忆，而 MEMORY-JEPA 又是如何解决这一问题的?** 假设用户编辑了一个文件，将其关闭，然后在其他位置继续操作约一百步（几分钟），之后点击原文件所在的标签页将其重新打开。普通视觉世界模型没有显式的

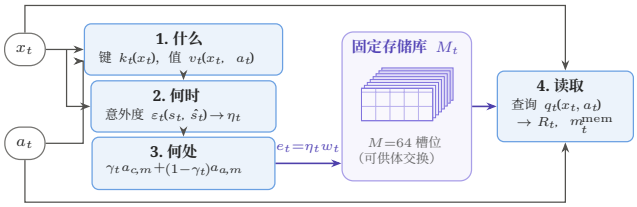



图 S1。 **放大查看主图 2 中的模块 1（记忆）：概念槽记忆将写入和读取拆分为四个经过学习的决策。** 在主图 2 的紫色记忆框内，步骤 $t$ 从 $x_t$、$a_t$）中提取键和值（内容），通过预测惊奇度 $\varepsilon_t$ 经由 $\eta_t$ 门控写入，使静态帧不会覆盖令牌库（何时），通过 $w_t$ 将内容匹配与空槽分配结合起来，把更新写入 64槽令牌库 $M_t$（其中，公式（S2））， 并发出动作条件查询，以将屏幕外状态读取到 $R_t$ 和 $m_t^{\mathrm{mem}}$（读取）。

屏外存储：它依赖近期帧的短滑动窗口或单个循环隐藏状态，并且完全通过重建像素进行训练。经过长时间间隔后，已关闭的文件会被挤出有限上下文；即使该文件重新出现，能够揭示其内容的少量像素也会被数千个未发生变化的背景像素淹没。

为了在较长时间范围内记住抽象概念，我们加入了一种固定容量槽记忆——类似于神经硬盘（图 S1）——配备经过学习的读取头和写入头，用于决定存储什么以及检索什么。训练仍然完全基于无标签帧-动作轨迹，不使用真实槽标签。然而，如果这些读取头和写入头 仅 从外部渲染损失 $\mathcal{L}_{\mathrm{ren}}$（通过 UNet 反向传播）中接收梯度，它们就会遭遇同样的梯度淹没问题。

我们的 MEMORY-JEPA 架构通过一个直接在表征空间中基于记忆控制器训练的自监督预测器，同时解决了这两种失效——淹没效应和拥挤效应。

首先，预测器不再通过 UNet 重建数十万个像素，而是仅根据当前屏幕、用户的动作以及读头从记忆中读取的内容，预测一个概括下一屏幕的单一 512 维语义向量（由 Qwen-VL 视觉特征和变分自编码器潜变量池化得到）。通过 $\mathcal{L}_{\mathrm{InfoNCE}}$ 与批次中的其他屏幕进行对比训练时，这项损失要求模型判断即将出现的是哪个屏幕，而不是在数千个未发生变化的背景像素上取平均；此外，由于当前屏幕和点击动作都无法揭示已关闭文件中的内容，预测器只有在写入头此前已将该内容写入记忆，并且

读头将其读取出来后，才能匹配正确的目标。

其次，由于预测器在每一步都会生成对下一屏幕的实时预期，它为我们提供了一种有原则的预测惊奇度（正文第 3.1 节；图S1，步骤 2），其灵感来自预测编码 [17, 43] 和 CAMBRIAN-S [65]。因此，我们可以根据意外度的大小，对写入槽位库的每一次操作进行门控：可预测的静态帧会使门控保持关闭，从而避免重复的桌面帧挤占令牌库；只有未预测到的状态转移（例如编辑或关闭文件）才会打开门控，以提交一次写入。

这种 JEPA式架构可能出现两类潜在的坍塌问题，而我们对两者都进行了防护。首先，考虑表示坍塌：如果编码器和预测器只是将每个屏幕映射到同一个常数向量，预测误差就会降为零，却没有学到任何有关桌面的信息。为堵住这一漏洞，我们采用编码器的指数移动平均（EMA）副本——停止梯度——根据真实下一帧 $x_{t+1}$计算出一个稳定的目标摘要 $s^*_{t+1} \in \mathbb{R}^{512}$ 。随后，我们训练控制器的预测值$\hat{s}_{t+1} \in \mathbb{R}^{512}$ ，通过对比方式（$\mathcal{L}_{\text{InfoNCE}}$）和直接方式（平方$L_2$ 距离）使其匹配$s^*_{t+1}$ 两者。其次是记忆写入坍塌：即使屏幕摘要仍然包含丰富信息，控制器也可能将每个屏幕都归入同一个写入键 $k_t \in \mathbb{S}^{63}$ （ $\mathbb{R}^{64}$ 中的一个单位向量），或者读头的首次查询 $q_t^{(1)} \in \mathbb{S}^{63}$ 无法发现哪个已存储的键与即将出现的屏幕相匹配。我们通过添加 $\mathcal{L}_{\text{VICReg}}$ [8] 来使写入键分布于单位球面，并将 $q_t^{(1)}$ 直接与 EMA 编码器针对下一帧的目标键 $k^*_{t+1} \in \mathbb{S}^{63}$ 对齐，从而堵住这一漏洞——在不要求梯度跨越遮挡持续传播的情况下，对读取地址进行监督。

然而，最后一个捷径仍然存在。在许多计算机界面状态转移中，预测器可以简单地忽略 记忆库 $M_t$ ，仅依赖当前屏幕——在这种情况下，它通常也能表现得相当不错；鉴于计算机界面在相邻时间步之间往往十分静态，许多实例中尤其如此。为阻止这种情况，我们通过将 $M_t$ 替换为另一个回合的令牌库来构造反事实预测 $\hat{s}^{\text{swap}}_{t+1}$ ，并要求交换后的预测差一个边距 $\rho = 0.1$而更差：

$$\mathcal{L}_{\text{swap}} = \max\big(0,\ \rho + \|\hat{s}_{t+1} - s^*_{t+1}\|_2^2 - \|\text{sg}(\hat{s}^{\text{swap}}_{t+1}) - s^*_{t+1}\|_2^2\big), \tag{S1a}$$

三个向量均经过归一化，并在屏幕发生变化且 $k^*_{t+1}$ 与某个已占用槽位的键的余弦相似度高于0.7的帧上取平均。

将五个组件整合起来——对比式预测、直接 $L_2$ 对齐、读查询监督、键分散正则化以及反事实交换边距——给出了完整的 MEMORY-JEPA 损失：

$$\begin{aligned}\mathcal{L}_{\text{JEPA}} = {} & \mathcal{L}_{\text{InfoNCE}}\big(\text{normalize}(\hat{s}_{t+1}),\ \text{normalize}(s^*_{t+1})\big) \\ & + \lambda_{\text{align}} \big\| \text{normalize}(\hat{s}_{t+1}) - \text{normalize}(s^*_{t+1}) \big\|_2^2 \\ & + \lambda_q \big(1 - \langle q_t^{(1)}, k^*_{t+1} \rangle\big) \\ & + \lambda_{\text{VICReg}}\, \mathcal{L}_{\text{VICReg}}(k_t) + \lambda_{\text{swap}}\, \mathcal{L}_{\text{swap}}.\end{aligned} \tag{S1b}$$

移除 $\mathcal{L}_{\text{JEPA}}$ 会使已编辑文件召回率降回随机水平（第S4.1节）。

**记忆如何决定是更新现有槽位，还是占用一个空闲槽位？**不同于三维物理世界模型，计算机界面没有可用于记忆寻址的相机位姿。屏幕坐标也不是可行的替代方案，因为新应用程序窗口经常会在已关闭窗口的相同坐标上打开——这会简单地覆盖旧文件。第三种方案是定长队列，但它最终会填满并驱逐长期存在的状态。

我们通过遵循可微神经计算机（DNC） [19]来应对这些问题。一次写入会在两个目的地之间进行选择（图S1，步骤 3）：它要么更新已占用槽位中所存内容与当前屏幕最匹配的槽位，要么占用第一个空闲槽位。

每个槽位 $m \in \{1, \ldots, M\}$ 存储一个键 $K_{t-1,\,m} \in \mathbb{R}^{64}$ 以及一个占用状态 $u_{t-1,\,m} \in [0, 1]$二者均初始化为零。门控 $\gamma_t \in (0,\ 1)$（正文第 3 节 3.1）衡量当前写入键 $k_t$ 与最佳已占用槽位的匹配程度，并将内容分布 $a_{c,\,m}$与分配分布 $a_{a,\,m}$ 按 $w_{t,\,m} = \gamma_t a_{c,\,m} + (1 - \gamma_t) a_{a,\,m}$进行混合——在重新编辑已知文件时更新现有槽位（$\gamma_t \approx 1$），并为新文件占用空闲槽位（$\gamma_t \approx 0$）。对于内容匹配，我们根据每个槽位与 $k_t$ 的相似度进行评分，并加入对数占用偏置，使得一旦任一槽位被占用，空闲槽位便获得可忽略的权重。对于分配，我们不再维护 DNC 的离散使用率排序空闲列表，而是以可微方式从左到右占用空闲槽位：偏好占用状态较低的槽位，并根据每个槽位 $m$ 之前仍低于空闲阈值 $u_{\text{free}}$的槽位数量 $j < m$ 对其进行惩罚：

$$\begin{aligned} a_{c,m} &= \text{softmax}_m\Big(\frac{\langle K_{t-1,m}, k_t \rangle}{\tau_w} + \log(u_{t-1,m} + \epsilon)\Big), \\ a_{a,m} &= \text{softmax}_m\Big(\frac{-u_{t-1,m}}{\tau_a} \\ &\qquad - \lambda_{\text{prec}} \sum_{j=1}^{m-1} \sigma\Big(\frac{u_{\text{free}} - u_{t-1,j}}{\tau_a}\Big)\Big). \end{aligned} \tag{S2}$$

其中，$\tau_w$ 和 $\tau_a$ 是Softmax温度，$\lambda_{\text{prec}}$ 用于对从左到右优先级惩罚加权，$\sigma$ 是逻辑Sigmoid函数，而 $\epsilon$=$10^{-6}$ 用于防止出现log0。

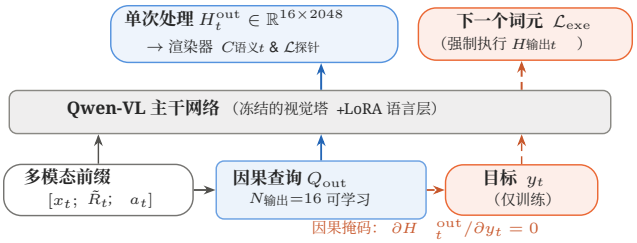



图 S2。**主图 2 中模块 2（计算）的放大图：前缀后因果查询提取单次处理的视觉语言模型执行结果，无需测试时解码。**在主图 2 的蓝色计算框内，16 个可学习查询 $Q_{\text{out}}$ 严格位于˜多模态前缀 $[x_t$、 $R_t$、 $a_t]$ 与仅用于训练的结果文本 $y_t$之间。在因果掩码下， $Q_{\text{out}}$ 读取前缀而看不到 $y_t$（$\partial H_t^{\text{out}}/\partial y_t = 0$），因此 $y_t$ 在测试时被丢弃；训练期间，针对 $y_t$ （橙色虚线）的下一词元损失 $\mathcal{L}_{\text{exe}}$ 迫使视觉语言模型将执行出的答案压缩到 $H_t^{\text{out}}$中。

**为什么标准扩散损失在大部分静止的桌面上会失效？** 在状态转换之间，大多数桌面像素几乎不会移动。在每一步都重新合成整帧会浪费容量来重绘静态背景，而且往往会使界面文本变得模糊。因此，仅预测帧间变化似乎是个好主意。这样每个时间步只需预测状态中的增量，就能避免处理静态文本。然而，这种格式又引入了另一个陷阱：未加权损失会被未发生变化的图像块主导，促使渲染器在各处都预测零变化，从而冻结屏幕。

我们通过潜变量空间中的主动加权残差损失解决这一张力（图 S3）。令 $\Delta z_t = (z_{t+1} - z_t)/s_c$表示按其每通道尺度ˆ $s_c$标准化后的潜在增量，并令 $\Delta z_t$ 表示残差 UNet 的预测。若一个潜在补丁 $p$ 在 $C$个通道上的最大变化低于静默阈值 $\delta_{\text{stat}}$，我们就将其标记为静态，从而形成静态集合 $\mathcal{S}_t = \{p : \max_c |\Delta z_{t,\ p,\ c}| < \delta_{\text{stat}}\}$。相反，变化超过 $\delta_{\text{chg}}$ 的补丁会获得额外权重$\omega_p = 1 + \omega_{\text{chg}} \cdot \mathbf{1}[\max_c |\Delta z_{t,\ p,\ c}| > \delta_{\text{chg}}]$。随后，渲染损失在活跃编辑与静态静默性之间进行平衡：

$$\mathcal{L}_{\text{ren}} = \frac{\sum_p \omega_p \|\hat{\Delta} z_{t,p} - \Delta z_{t,p}\|_2^2}{\sum_p \omega_p} + \frac{\lambda_{\text{static}}}{|\mathcal{S}_t|\, C} \sum_{p \in \mathcal{S}_t} \|\hat{\Delta} z_{t,p}\|_1. \tag{S3}$$

第一项确保变化区域不会被静态背景淹没，而第二项则对静态补丁施加 $L_1$静默惩罚，使未变化区域不会在滚动生成过程中累积漂移。将这一残差目标替换为全帧预测会使终端 OCR（S4.4节）失效；所有损失权重和阈值列于表 S1。

**我们如何利用视觉语言模型的符号推理能力，而无需为测试时的文本解码付费？**近似在自然视频中，动力学看起来很有说服力，但在终端中输入 3+3必须每次都产生 6。因此，我们需要让模型理解符号计算。然而，仅从像素中学习精确算术的速度慢得令人难以接受。因此，我们想知道能否在这里借助预训练视觉语言模型：大多数预训练视觉语言模型已经知道如何进行计算；剩下的唯一问题是，我们能否将视觉语言模型的答案传递给渲染器。沿着这条路径探索时，我们遇到了两个问题。直接从视觉语言模型中读取提示特征，而不提供文本监督，并不能揭示答案。等待视觉语言模型自回归地生成答案会准确得多；然而，这会显著降低推理速度，因为我们必须在每个时间步等待视觉语言模型完成一次完整的自回归滚动生成。

我们的前缀后因果查询桥接（图 S2，放大查看主图 2 中的模块 2）通过一个简单的排序技巧解决了这一权衡。在训练期间，我们将 $N_{\text{out}}$=16 可学习查询令牌 $Q_{\text{out}}$ 紧接在多模态前缀 $[x_t$之后追加˜； $R_t$； $a_t]$ 并严格置于目标结果字符串 $y_t$之前。在因果注意力下， $Q_{\text{out}}$读取前缀时看不到 $y_t$ （$\partial H_t^{\text{out}}/\partial y_t = 0$），因此在测试时，我们只需丢弃 $y_t$ ，并通过一次前向传播计算查询隐状态 $H_t^{\text{out}} \in \mathbb{R}^{16\times 2048}$ 。然而，在训练期间， $y_t$ 会回看并关注 $Q_{\text{out}}$：对 $y_t$ 计算的下一标记交叉熵 $\mathcal{L}_{\text{exe}}$ 迫使视觉语言模型检索已清除的˜变量，这些变量来自 $R_t$，执行该操作，并将答案打包到 $H_t^{\text{out}}$中。最后，使用交叉熵训练的辅助线性头 $\mathcal{L}_{\text{probe}}$ 通过冻结的语言模型输出头，从 $H_t^{\text{out}}$预测 $y_t$的前 16 个标记，使答案能够被渲染器以线性方式读取。移除两项执行损失后，清屏算术准确率降至 0.0%（第 S4.3 节）。

## S1.2. 架构开销与分辨率桥接

实用的神经计算机必须在会话不断变长的同时保持内存开销恒定，并扩展到高分辨率字体排版，而不会导致参数量或延迟大幅增加。

**记忆和计算会增加多少开销，用户操作又是如何编码的？**由于屏幕外状态被压缩到包含 64 个槽位的固定存储库中，而不是存入不断增长的帧缓冲区或键值缓存，因此内存成本与推演长度无关：槽位记忆增加了 7.78M 个可训练参数，并在每个回合中保持恒定的 0.41MB FP32 状态。总而言之，固定记忆组件只增加了极少的开销，为进一步扩展其规模留下了充足空间。

在计算方面，我们冻结视觉语言模型骨干网络，仅使用 LoRA [29]（$r$=16；17.4M 个参数，用于 Qwen3-VL-2B-Instruct [6]）调整其语言层。在渲染方面，预训练的短期动作长短期记忆网络 [46]

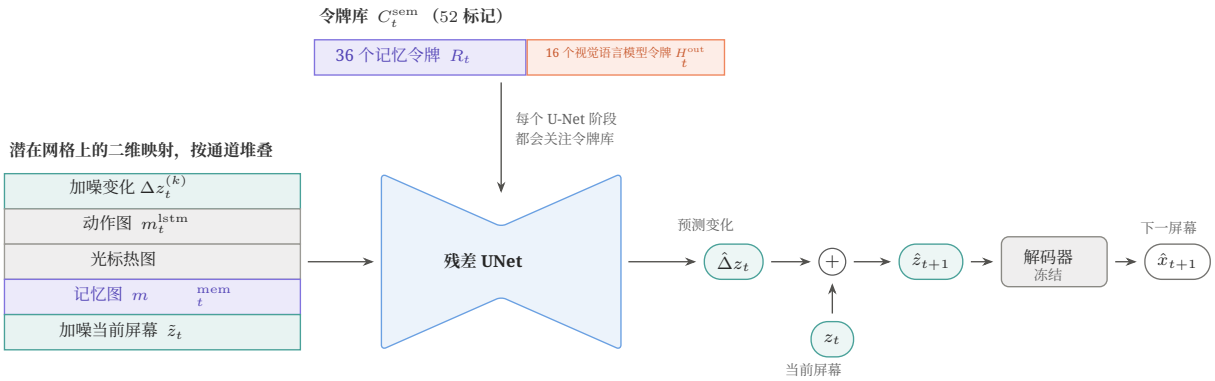



图 S3。放大查看主图 2 中的模块 3（接口）：记忆和计算以与其他输入堆叠的二维地图形式，以及以每个 UNet 阶段都会进行注意力处理的标记形式，到达渲染器。左侧为堆叠地图。五张地图共享潜在网格，并在 UNet 输入处作为通道拼接：加噪变化 $\Delta z_t^{(k)}$，即需要进行去噪的量（扩散步 $k$时下一屏幕与当前屏幕潜变量之间的差值）；动作图 $m_t^{\text{lstm}}$，其中预训练的长短期记忆网络会总结最近的鼠标和键盘输入；以光标为峰值的热力图；记忆图 $m_t^{\text{mem}}$，用于存储每个屏幕位置从槽位记忆中读取的内容；以及当前屏幕的加噪副本 $\tilde{z}_t$，其经过破坏处理，使模型无法直接复制它。顶部为令牌库。部分内容没有屏幕位置：36 个记忆令牌 $R_t$（4 个读取头，每个返回 1 个语义令牌和 8 个视觉令牌）以及 16 个视觉语言模型令牌 $H_t^{\text{out}}$，后者携带计算得到的结果。二者都被投影到共同的宽度，形成包含 52 个令牌的令牌库 $C_t^{\text{sem}}$。在其 7 个阶段中的每个阶段，UNet 的每个位置都会对令牌库进行注意力处理，并通过一个学习门控加入结果；该门控初始值较小，因此预训练渲染器在初始化时不会受到干扰。右侧为输出。UNet 仅预测潜变量中的变化 $\hat{}\,\Delta z_t$。将其加到当前潜变量 $z_t$ 上即可得到下一潜变量，冻结的解码器会将其转换为下一屏幕$\hat{x}_{t+1}$；因此，不发生变化的区域会原封不动地保留。UNet 使用 $\mathcal{L}_{\text{ren}}$ 进行训练（公式 (S3)）。示意图。

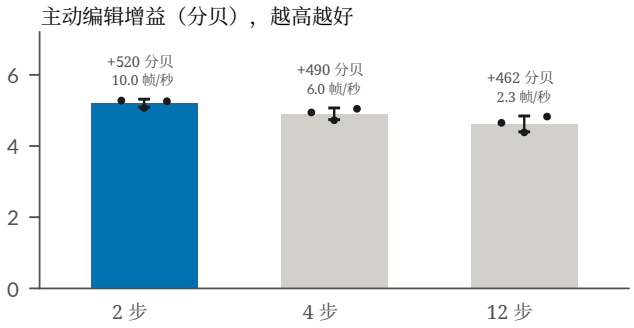



图 S4。**在 1024×768时，两步去噪扩散隐式模型采样已足够：更多步数会降低速度，且不会提升主动编辑增益。**在 2、4 和 12 个采样步数下，**我们的方法**（主表 1B 中的模型）的主动编辑增益；每个柱状图上方标出了在一块 GH200 GPU 上的端到端帧率；我们采用 2 步（蓝色）。随着步数增加，增益略有下降，从 2 步到 12 步降低 0.6 分贝。柱状图表示均值，误差线表示标准差，圆点表示 3 个训练随机种子；每个随机种子包含 862 个主动状态转移。

提供局部条件图 $m_t^{\text{lstm}}$，而残差 UNet 中的门控二维交叉注意力层（图S3）注入投影后的记忆和计算标记 $C_t^{\text{sem}} \in \mathbb{R}^{52\times 512}$。由于符号推理和二维渲染需要以不同方式查看用户输入，每一步都会同时提供逐字的击键字符串 $a_t$ 供视觉语言模型使用（保留准确的字符顺序和快捷键，例如`Ctrl+L`），以及供 UNet 和长短期记忆网络使用的二维光标热图、点击标志和多热键向量。

相同的记忆在 1024×768 交互式帧率下如何扩展？4 通道的 512×384稳定扩散变分自编码器将终端可读性限制在 56.3% 的光学字符识别率，因此，使用 16 通道 FLUX.1-schnell f8 VAE [11]（96×128 潜变量）扩展到 1024×768对于呈现清晰文本至关重要。为将记忆大小与像素分辨率解耦，记忆库始终在固定的 48×64 空间网格上运行：从 512×384 移至 1024×768 不会增加任何记忆参数，因为我们只需将潜变量 2×2 平均池化到48×64，然后进行记忆读取和写入，并通过双线性插值上采样 $m_t^{\text{mem}}$回 96×128。为保持生成的交互性，我们在 [50] 处使用 2 步去噪扩散隐式模型采样 1024×768（而每个 512×384 模型均使用 32 步，以匹配 NEU- RALOS；表 S3）。如图 S4 所示，在1024×768 下，2 步时的主动编辑增益为 $+5.20\pm0.11$ 分贝，4 步时为 $+4.90\pm0.16$分贝，12 步时为 $+4.62\pm0.22$分贝，因此额外步数只会牺牲速度，并不会提升质量；我们尚未测试512×384 下的步数。在 2 步时，完整模型在一块 NVIDIA GH200 GPU 上，跨三个随机种子端到端运行速度为 $10.0 \pm 0.4$帧/秒；主表 1B 报告的是第一个随机种子的结果（10.5 帧/秒）。

## S1.3. 评估套件和评分协议

为什么要将功能性基准测试与峰值信噪比和学习感知图像块相似度结合起来进行评估？桌面上大约 95% 的内容在每个时间步都不会发生变化，因此，一个只是复制上一帧的模型——或者更糟糕的是，在重新打开一个已关闭的文件时，臆造出看似合理的胡言乱语——仍然可能获得相当不错的单步感知得分。仅评估桌面

世界模型的感知指标（例如峰值信噪比和学习感知图像块相似度），因此是现有基准测试中的一个重大盲点。

为了区分真正可工作的计算机与无状态的帧插值器，我们将标准感知指标与对物体恒存性、因果记忆使用和符号执行的直接测试结合起来。

我们在两个独立的设置中进行评估：一个受控的512×384测试套件，在编解码器和分辨率方面与 NEURALOS [46] 保持一致，以进行公平的基线比较；另一个是使用 1024×768`FLUX.1-schnell` 自编码器的测试套件。不同分辨率之间不比较模型。

• 隐藏形状回忆与同窗格覆盖。每个匹配三元组包含三个 VS Code 回合，其隐藏文件分别保存三角形、圆形或星形。在重新打开帧（下采样至 256×192，在 $[0,1]$ 中）中，我们提取三个真实目标 $T_u (u \in \{0,1,2\})$ 不同的区域，$\mathcal{R} = \{(y,x)\colon \max_c(\max_u T_u - \min_u T_u) > 0.1\}$ ($|\mathcal{R}| \geq 500$ 像素)。三路回忆准确率是指生成帧 $\hat{}x$ 的平均 $L_1$ 误差在 $\mathcal{R}$ 上对真实形状最低的比例（33.3% 随机水平——复制当前帧的模型得分恰好为 33.3%）。桌面开发集包含 32 个匹配三元组，其中 18 个包含已编辑的隐藏文件，14 个包含未编辑的隐藏文件。编辑文件召回率使用 18 个已编辑三元组：54 个回合，每个回合生成 3 个噪声样本，因此每个训练模型有 162 次评估。覆盖使用全部 32 个三元组：96 个回合，每个回合生成 3 个噪声样本，因此每个 $\Delta t$有 288 次评估。在本补充材料中，$\pm$ 是独立训练模型之间的标准差。开发集三元组一次性分为包含 17 个三元组和包含 15 个三元组的两半；消融实验（S4 节）在第一半上评分（9 个已编辑三元组，每个训练模型有 81 次评估）。编辑文件召回率、覆盖和捐赠者交换结果见主表2。

• **XFCE 文件管理器持久性**。匹配的回合会创建新建文件夹 于 文档中，关闭文件管理器 64、128或256 步后重新打开文档。我们以定性方式展示这些回合（主图 3 和第 S8节）。

• **反事实供体记忆交换**。为检验回忆是否由显式槽位库中的因果机制驱动，而非由当前帧或长短期记忆网络中的快捷方式驱动，我们固定 $x_t$、$a_t$以及回合 $i$ 重新打开帧处的 LSTM 状态，并将 $M_t \leftarrow M_{\text{供体}t}$ 替换为来自同一三元组但具有不同形状的另一个回合 $j$ 的供体记忆。供体交换跟随率 是指生成帧在同一区域上的平均 $L_1$ 误差低于原始形状、且更接近供体形状的试验比例，$\mathcal{R}$ 与上文相同（50。0% 为随机水平）。模型通过 $\mathcal{L}_{\text{交换}}$（方程（S1a））进行训练，该方法奖励记忆预测对这种交换的敏感性。

• **感知保真度与主动编辑增益**。单步质量使用 AlexNet LPIPS v0 进行测量。1 [70], InceptionV3 弗雷歇起始距离 [26], 、峰值信噪比和 SSIM。为排除无意义的帧复制，在主动转换（$\geq$ 64 像素发生变化的 $> 2/255$）中，我们还报告 主动编辑增益：变更像素 PSNR 减去复制 $x_t$的变更像素 PSNR。 1024×768 得分使用每个随机种子的862 次状态转移，而表中的得分使用表 S4 中的 930 个桌面和 701个数学帧。

• **CLI 执行与终端可读性**。数学开发集在四个 Python REPL 系列中包含 324 个评分状态转移，其中 124 个是已清除操作数的状态转移，即在查询前通过 Ctrl+L 清除屏幕。清屏答案（%）是视觉语言模型解码答案的贪心精确匹配，而 $H_{\text{输出}t}$ 探针（%）是单次处理线性头通过取 argmax 得到的精确匹配。终端 *OCR*（%）是指生成帧中能够以不区分大小写的子字符串形式，由 EasyOCR 恢复的较早终端行（$n$=157）所占的比例；生成帧已放大 2×。

## S1.4. 训练与优化

我们如何在有限的 GPU 内存下训练记忆，使其经受住长时遮挡？每个模型都在包含连续 128 帧的窗口上进行训练（图 S10 ）。前 64 帧是预热阶段：模型在无梯度的情况下对这些帧进行前向运行，因此到第 65 帧时，其记忆中已经保留了回合此前展示的内容，例如一个经过编辑后又被关闭的文件。剩余 64 帧采用截断反向传播通过时间进行训练，并在其中 8 帧上计算损失。我们使用 AdamW（$\beta_1$=0.9、$\beta_2$=0.999，权重衰减 0.01）、先以学习率预热逐步升至恒定学习率（512×384 时为 200 步； 1024×768 时为 150 至 300 步）、全局梯度范数裁剪 1.0，以及 $T$=1,000 的扩散调度。表 S3 给出了每个模型的训练预算和采样设置，表 S1 给出了 512×384模型的其余设置。所有 512×384 模型均从 NEURA- LOS 的 joint-272k 检查点初始化；基线模型按照已发表的 NEURALOS 配方进行微调。 1024×768的运行会在达到墙钟时间限制时停止，因此其步数是多个随机种子下的均值（在 Dense Interactive Mixture 上为 867、834 和 826 步；在 Broad Mixture 上为 1,510、2,105 和 2,069 步）。值得注意的是，每项预算差异都对基线而非我们的方法有利：微调后的 NEURALOS 所见窗口数量是我们 512×384 模型的三倍，而在 Dense Interactive Mixture 上，C OPY-ONLY 所见窗口数量约为我们的四倍。 1024×768 模型采用表 S1 中的记忆和损失设置，预训练权重的学习率为 $10^{-5}$，新模块的学习率为 $10^{-4}$ （在 Dense Interactive Mixture 上为$1.5\times10^{-4}$ ）。

## S2. 模型超参数

表 S1列出了 512×384模型的设置，该模型用于第 S4节的消融实验训练；训练预算

| 符号 | 值 | Role |
|---|---|---|
| $M$，$H_r$ | 64，4 | 槽位和读取头 |
| – | 64，512，8×128 | 键、语义、视觉维度 |
| $\mu_{\text{EMA}}$ | 0.996 | JEPA 目标动量 |
| $\alpha_s$, $\beta_s$ | 4.0，1.25 | 惊讶门控尺度、偏移量 |
| $\tau_w$, $\tau_r$, $\tau_a$ | 0.1，0。1，0。05 | 写入，读取，分配临时值 |
| $\lambda_{\text{prec}}$, $u_{\text{free}}$ | 5，0。1 | 分配惩罚，阈值 |
| $\lambda_{\text{align}}$, $\lambda_q$ | 0.5，0.5 | 对齐，查询权重 |
| – | 0.07 | InfoNCE 温度 |
| $\delta_{\text{stat}}$, $\delta_{\text{chg}}$ | 0.02，0.35 | 静态，已更改阈值。 |
| $\lambda_{\text{VICReg}}$, $\lambda_{\text{swap}}$, $\rho$ | 0.25，0.5,0.1 | VICReg、交换、边距 |
| $N_{\text{out}}$ | 16 | 视觉语言模型结果查询 |
| $r$ | 16 | LoRA 秩 |
| $\lambda_{\text{JEPA}}$, $\lambda_{\text{exe}}$, $\lambda_{\text{probe}}$ | 0.5, 0.2, 0.1 | 联合损失权重 |
| $\omega_{\text{chg}}$，$\lambda_{\text{static}}$ | 4，0。1 | 渲染损失（公式（S3））|
| $g_0$ | 0.25 | 交叉注意力门控初始化 |
| $p_{\text{drop}}$ | 0.5 | 发送给视觉语言模型的空白屏幕 |
| – | 0.3 | 参考复制丢弃 |
| – | $10^{-4}$，$10^{-5}$ | 学习率：新模型、预训练模型 |
| – | 200，3,000 | 学习率预热、总步数 |
| – | 16 | 批量大小 |
| – | 8 | 每个窗口的损失帧数 |

表 S1。**模型的 512×384 设置**。符号遵循正文第 3 节和第 S1.1 节；没有符号的行在第 S1.4节中说明。 $s_c$ 表示训练残差的逐通道标准差。

见表 S3。

## S3。数据语料库来源与 1024×768混合数据集

公共 GUI 语料库提供了广泛的应用多样性，但很少重新访问已关闭文件，或在屏幕清空后测试算术。将三个公共桌面语料库与我们的受控 XFCE 录制数据结合后，得到来自全部五个数据源的 173,975条经过清理的训练轨迹（169,207 条来自公共数据源，4,768 条来自内部 XFCE，见表 S2）。图 S5 展示了全部五个数据源，图 S6 以完整分辨率展示了三个 AgentNet 回合，图 S7则展示了每个受控任务的一个完整回合。

**对齐与清理**。数据源在宽高比、动作日志记录和帧时序方面存在差异，因此我们会在训练前分别进行标准化处理。我们将所有 16:9 的录制数据加黑边缩放至 4:3 画布，并使用不执行代码的抽象语法树（AST）解析器解析 pyautogui 脚本。此过程移除了 194 条损坏的轨迹（其中包括 45 条 ProCUA-SFT 记录中被错误解析的 (0，0) 滚动坐标，以及 149 条 Agent-Net 记录中缺失的间隔帧），并修正了 ProCUA-SFT 中一个帧的动作偏移量，从而确保每一步都遵循 $(x_t,\ a_t) \rightarrow x_{t+1}$。短于 64 帧的回合会在前面填充零潜变量，并在所有损失计算中将这些潜变量屏蔽。

[1]https://huggingface.co/datasets/p-doom/cuagym- qwen35-滚动生成

## S4。3,000 步时的四项消融实验

消融实验是彼此独立且更短的实验。包括完整模型在内的每个实验分支仅训练了 3，000 步（第 S1.4 节），因此第 S4.1 节至 S4.4 节中的数值应彼此比较，而不应与主表格中的数值比较，因为后者报告的是更长时间的训练结果。

在此设置下，完整模型的 40.3±0.7%编辑文件召回率、 28.2 ± 2.1%清屏答案和 43.5 ±2.2%终端 OCR。每项消融都会从完整模型中移除一个组件[2]。答案准确率基于 124 次状态转移计算，因此以 0.8 个百分点为步长变化；若干保留执行损失的实验组的平均值相同，均为 28.2%（124 次中的 35 次）。

### S4.1. 记忆监督（MEMORY-JEPA）

记忆目标旨在跨越一次遮挡提供学习信号，其中隐藏文件只会影响单个重新打开帧中的一小块区域。去除记忆目标（$\mathcal{L}_{\text{JEPA}}$=0）后，编辑文件召回率为 32.1%（随机种子为 25.9% 和 38.3%），而同一实验中完整模型的该指标为 40.3±0.7%；清屏答案（28.2%）和终端 OCR（45.2 %）保持不变。

### S4.2. 惊讶门控

在每一帧（$\eta_t \equiv 1$）上写入内容可获得 35.8% 的已编辑文件召回率（随机种子为 33.3% 和 38.3%），而在使用惊讶门控 [17, 43, 65] 的相同实验完整模型中，该指标为 40.3±0.7% 。

### S4.3. 符号计算监督

清屏答案准确率评估的是视觉语言模型的解码文本，而不是渲染后的屏幕。借助执行损失，完整模型在 124 个清屏查询中答对了 28.2 ± 2.1%（其中算术查询的答案准确率为 15.5%，数字召回查询的答案准确率为 38.7%），而在操作数仍显示在屏幕上的 200 个查询中，答案准确率为 65.5 ± 0.5% 。不使用这些损失（$\mathcal{L}_{\text{exe}}$=0,$\mathcal{L}_{\text{probe}}$=0）时，三个随机种子下的清屏答案准确率均为 0.0%，且 $H_t^{\text{out}}$探针准确率从 23.7±2.8%降至 0.0%。该答案并非仅源于渲染监督。

### S4.4. 残差目标；屏幕丢弃设置

在每一步都重新生成完整帧会使小字形变得模糊。将残差目标 $\Delta z_t$ 替换为完整帧 $x_0$ 预测后，终端 OCR 从 43.5 ± 2.2% 降至 0.2 ±0。4%，而清屏答案保持不变（28.2±2。4%）。在该实验中，残差目标以可读文本为代价换取了召回率：完整帧组的已编辑文件召回率更高（65.4 ± 10.5% 对 40.3 ± 0.7%）。在训练期间

[2]完整模型以及不含执行损失或残差目标的各个分支均使用三个随机种子。不含记忆损失或惊奇门控的分支使用两个随机种子。

| 来源 | 来源／许可证 | 训练／开发回合 | 分享 | 备注 |
|---|---|---|---|---|
| 桌面 （VS Code） | 内部 XFCE | 768 / 96 | 12.5% | 32 个匹配的三元组 |
| 数学 （Python 交互式解释器） | 内部 XFCE | 4，000 / 300 | 12。5% | 4 个算术类别 |
| AgentNet [56] | xlangai/AgentNet（MIT） | 21,957 / 425 | | 2,500 回合 at 1024×768 |
| ProCUA-SFT [32] | NVIDIA/ProCUA-SFT（知识共享署名 4.0） | 92，964 / 396 | 75.0% | 2，500 回合，于 1024×768 |
| CUA-Gym[1] | p-doom/cuagym-qwen35-rollouts（CC0） | 54，286 / 531 | | 2，500 回合，于 1024×768 |
| 总计 | 5 个数据源（> 2.1TB 原始数据） | 173,975 / 1,748 | 100。0% | 12，268 回合，位于 1024×768 |

表 S2。**该 1024×768 Broad Mixture 由75% 公开 GUI 轨迹和25% 受控 XFCE 任务组成**。清理后的数据池包含 173，975 条训练轨迹（8。83M 步），而分层 FLUX.1-schnell 子集则将 12，268 条轨迹（694，266 个训练窗口）编码为 FP16 $\mathbb{R}^{16\times96\times128}$ 潜变量。

| 模型 | 步数 | Win. /步 | Win. 总 | DDIM 步数 |
|---|---|---|---|---|
| 512×384 | | | | |
| **我们的方法**（章节 S4） | 3，000 | 16 | 48k | 32 |
| NEURALOS，微调 9，000 | | 16 | 144k | 32 |
| 1024×768 | | | | |
| **我们的方法**，密集交互 | 842 | 16 | 13k | 2 |
| **Ours**, Broad | 1,895 | 16 | 30k | 2 |
| COPY-ONLY | 3,500 | 16 | 56k | 2 |

表 S3。**每个模型的训练预算和采样设置**。Win./step 表示每个优化器步所对应的 128 帧训练窗口数，Win. 表示该数值与步数的乘积。

视觉语言模型以概率$p_{\text{drop}}$=0.5看到空白屏幕，而不是 $x_t$；参考副本 $\tilde{z}_t$ 以 0。3 的概率被丢弃；我们未报告对这两项设置中任一项的消融实验。

## S5. 基线验证与闭环运行

在闭环推演中，模型自身的输出会在每一步替代真值帧，因此误差可能会不断累积。图S8 展示了在 VS Code 中进行的一次学生强制展开，第S8 节展示了更多示例，主表 1A 报告了经过 261个自生成步数后的峰值信噪比。

零样本 NEURALOS。公开的 joint-272k 检查点在我们的 XFCE 套件上未经微调进行评估时，能够跟随点击操作，但会渲染其自身的训练桌面（在桌面上的峰值信噪比为 9.84 分贝、学习感知图像块相似度为 0.708，在数学上的峰值信噪比为 9.04 分贝、学习感知图像块相似度为 0.714，见表 S4）。在我们的语料上对其进行微调后，桌面的峰值信噪比提升 15.8 分贝（达到 25.66 分贝），数学的峰值信噪比提升 15.0 分贝（达到 24.05 分贝）；在全文中，我们均将微调后的模型作为基线。表 S4 仅采用单个评估随机种子，因此其值与主表 1A 中三个随机种子所得的均值略有不同。

## S6. 混合数据集上的高分辨率 1024×768训练

此外，我们还在包含 694,266 个窗口的 Broad Mixture（表 S2）上训练该 1024×768 模型，这与主表 1B 中的 Dense Interac- tive Mixture 相比是额外的训练。训练在 4 个 GH200 GPU 上进行了 21.5 小时，覆盖 0.044 个训练轮次。从第 500 步到最后一个检查点，整帧峰值信噪比从 27.8 分贝上升至 29.6 分贝，主动编辑增益从 +0.09 上升至 +2.97分贝，AlexNet 学习感知图像块相似度则从 0.074 降至 0.056（图 S9）。这些是用于主表 1B 的 862 个主动开发集转换上的渲染指标：其中 474 个来自 VS Code，324 个来自数学，64 个来自 AgentNet。该集合有 93% 为内部数据，其中包括 Dense Interactive Mixture 的全部数据和 Broad Mixture 的四分之一，因此更有利于在前者上训练的模型。对于此次运行，我们不报告记忆或计算指标。

## S7. 训练进展与计算预算

这些 1024×768 模型在第128帧窗口上进行训练，详见第S1.4 节；训练时长为6.3 小时（密集交互）和21.5小时（Broad），使用4 块 NVIDIA GH200 GPU；在最后一个检查点，模型的峰值信噪比和主动编辑增益仍在上升（图S9）。

## S8. 其他定性展开

另外四个 512×384推演条带补充了主图 3 和 4。每条都是手动选取的单个回合，用于展示某种行为；汇总结果见主表 1 和 2。每幅图都带有说明其协议的横幅。在教师强制推演中，每一帧都根据它之前的真实帧进行预测，因此测试隔离了间隔期间的记忆。在学生强制推演中，模型只能获得其自身上一个输出，因此错误可能会不断累积。教师强制：在主图 3 文件管理器任务的第二个回合中，**我们的方法**在新建文件夹图标经历128步遮挡后重新绘制该图标（图S11）。学生强制：**我们的方法**在37 个 VS Code 步骤后重新绘制隐藏的星形（图S12），在经历 16 个文件管理器步骤后仍保留已创建的文件夹（图S13），并在经历9

| 数据划分（帧） | NEURALOS (512×384) | 学习感知图像块相似度 ↓ | 峰值信噪比 ↑ | SSIM ↑ | 弗雷歇起始距离 ↓ | 终端 OCR（%） ↑ |
|---|---|---|---|---|---|---|
| 桌面 (930) | 公开发布（零样本） | 0.708 | 9.84 | 0.565 | 376.8 | – |
| | 微调（9k 步数） | **0.113** | **25.66** | **0.914** | **67.9** | – |
| 数学 (701) | 公开发布（零样本） | 0.714 | 9.04 | 0.425 | 466.1 | 0.0 |
| | 微调（9k 步） | **0.127** | **24.05** | **0.795** | **85.2** | 0.0 |

表 S4。**在我们的语料上微调 NEURALOS 后，相比未修改的公开发布版本提升了 +15–16 分贝**。在相同的 512×384 开发帧上评估（32 个去噪扩散隐式模型步数，随机种子为 0）。零样本 joint-272k 会渲染其自身的训练桌面，即使经过微调，终端 OCR 仍为 0。0%（$n$=157）。

个 Python 交互式解释器步骤后仍保持终端标题栏清晰可读（图S14）。

## S9. 对未见桌面的泛化

**我们的方法** 对未见桌面的泛化优于 NEURA-LOS [46]：在相同录制数据上训练时，两种模型渲染 VS Code 的保真度相近；但在 Windows、ma- cOS 和 Ubuntu 桌面上，NEURALOS 的 20。0 分贝峰值信噪比下降，而我们的方法仅下降 11。1 分贝（图 S15）。

**设置**。 我们比较表 S3 中的两个 512×384 模型。两者均从 NEURALOS joint-272k 检查点开始，仅在密集交互混合数据集上训练；该数据集包含我们在 XFCE 桌面中的 VS Code（桌面）以及 Python 交互式解释器（数学）上的录制数据；NEURALOS 微调 9,000 步，我们的方法微调 3,000 步。两个模型都未见过 AgentNet 帧。已见应用 集包含 30 个留出的 VS Code 回合，来自桌面。未见桌面 集包含 30 个 AgentNet [56] 开发集回合：其中 16 个来自 Windows，9 个来自 ma- cOS，5 个来自 Ubuntu，涵盖浏览器、办公和演示软件、终端、系统设置及地图；其中 4 个展示了在另一种操作系统和主题下运行的 VS Code。每一帧都采用教师强制方式进行预测，即根据真实的前序帧和操作进行预测，因此两种模型始终接收真实的当前屏幕。我们对每个回合的前 12 个状态转移进行评分，每个状态转移使用 32 个去噪扩散隐式模型步数采样；每个模型使用一个训练随机种子和一个采样随机种子。我们报告峰值信噪比和学习感知图像块相似度 [70], , 后者是在 AlexNet 特征上计算的感知距离（越低越好）；先在每个回合内取平均，再跨回合取平均，并给出以回合为单位计算的 95% 自助法区间。

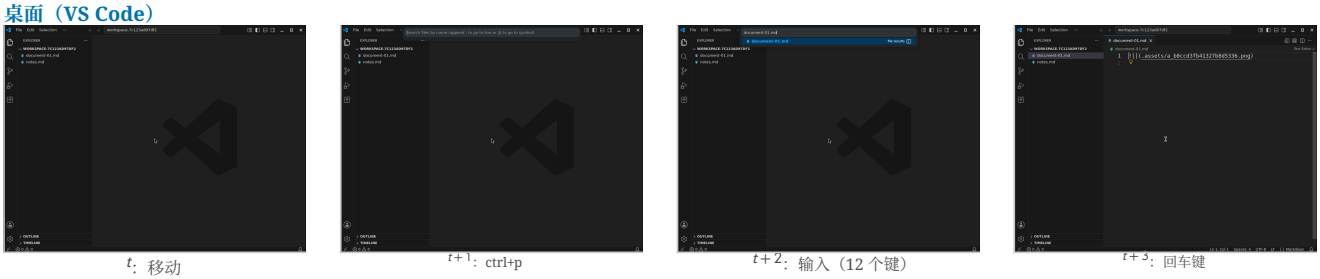
**结果**。 在 VS Code 上，两个模型的表现接近：我们的方法和 NEURALOS 的峰值信噪比分别为 26。2 和 25。8 分贝，学习感知图像块相似度分别为 0。097 和 0。120。在未见过的桌面上，NEURALOS 的表现降至 5。8 分贝和 0。86 学习感知图像块相似度，而我们的方法降至 15。1 分贝和 0。34；两者的区间没有重叠（图 S15）。在图 S16 的生成屏幕中，NEURALOS 会将 Windows 或 macOS 屏幕替换为黑暗屏幕；在 Windows 示例中，文字难以辨认，而我们的方法会保留输入给它的屏幕。当动作在这些桌面上打开新页面或对话框时，我们的方法会保留旧屏幕，并产生伪影。我们的 1024×768 模型在相同的混合数据集上训练，并在下采样至 512×384 后使用相同的回合进行评估，结果从 30。4 分贝降至 19。2 分贝。

**NEURALOS 失败的原因**。在 NEURALOS 中，负责绘制下一屏幕的 UNet 从未看到当前屏幕：当前屏幕的潜变量$z_t$（16×48×64 = 49，152 个数字）仅作为一个包含 1，024 个数字的注意力读取结果输入堆叠的两个长短期记忆网络中位于下方的一个，而 UNet 仅以位于上方的长短期记忆网络输出的 4，096 个数字的多层感知机投影作为条件。该状态没有编码的当前屏幕信息，都必须来自 UNet 的权重，而这些权重已经学习了训练数据中的屏幕。表 S4 中的零样本结果表明，即使不进行我们的微调，也会出现同样的失败：公开发布的 NEURALOS 在我们的 XFCE 桌面上运行时，却渲染出其自身的训练桌面（9。84 分贝）。

## S10. 更广泛的影响与伦理

屏幕外状态存储在一个显式记忆库中，因此无需重置模型的其余部分即可对其进行检查、替换或清除（正文第 5.3 节）。主要的滥用风险是生成欺骗性屏幕截图或“欺骗式”应用会话，这促使我们为生成式界面系统采用来源水印。我们的内部数据是 XFCE 虚拟机录制数据，不包含个人数据。图 S5 和S6中的 AgentNet 示例是随该语料库分发的屏幕截图；我们没有展示任何包含人物照片或账户详细信息的截图。公开语料库均依据其许可证使用（`AgentNet` 依据 MIT，`ProCUA-SFT`依据知识共享署名 4.0，以及 `CUA-Gym` 依据 CC0）。发布时，我们将公开经过对齐的五来源语料库、我们的转换与对齐流程，以及我们的训练代码。

**桌面（VS Code）**



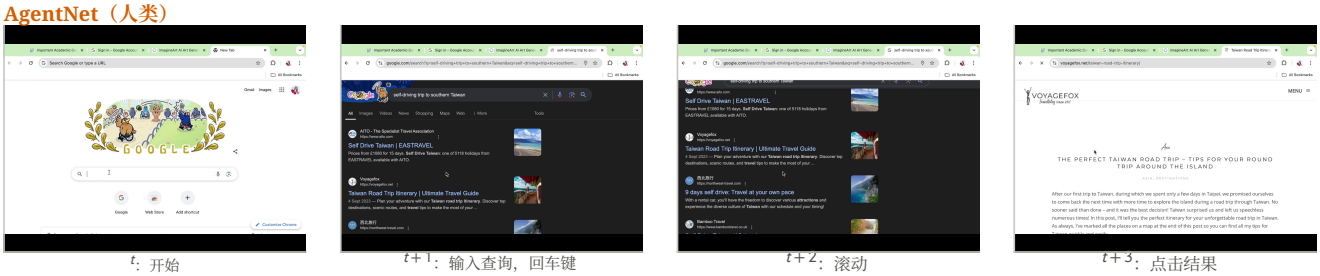
$t$：移动　$t+1$：ctrl+p　$t+2$：输入（12 个键）　$t+3$：回车键

**数学（Python 交互式解释器）**



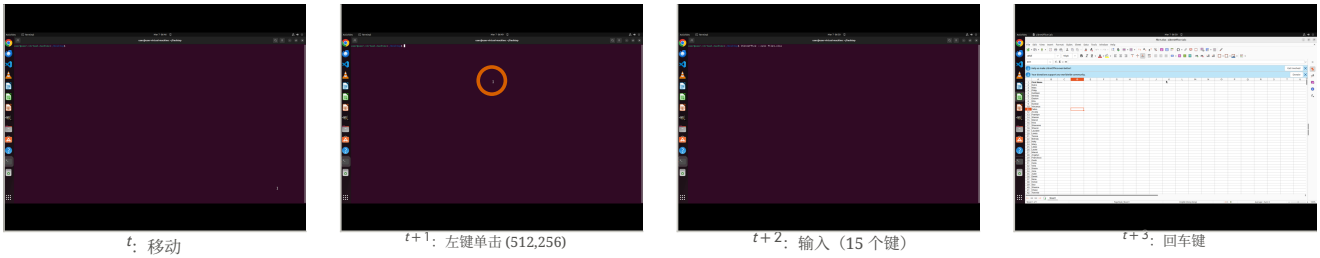

$t$：移动　$t+1$：输入（5 个键）　$t+2$：回车键　$t+3$：Ctrl+L

**AgentNet（人类）**



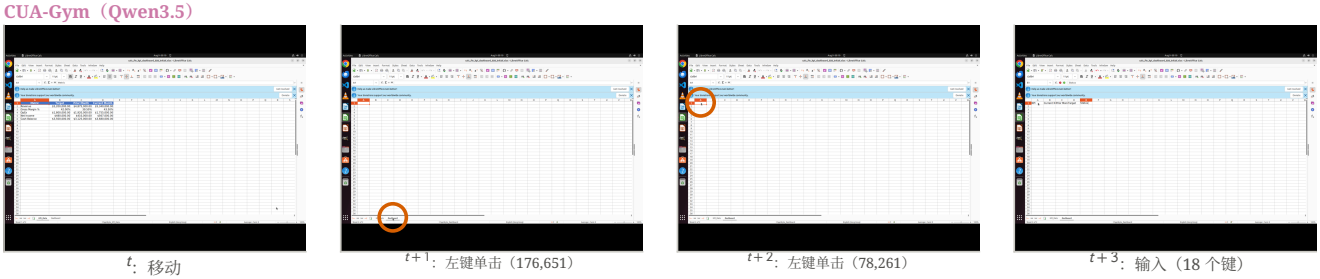
$t$：开始　$t+1$：输入查询，回车键　$t+2$：滚动　$t+3$：点击结果

**ProCUA-SFT（Kimi-K2.5）**



$t$：移动　$t+1$：左键单击 (512,256)　$t+2$：输入（15 个键）　$t+3$：回车键

**CUA-Gym（Qwen3.5）**



$t$：移动　$t+1$：左键单击（176,651）　$t+2$：左键单击（78,261）　$t+3$：输入（18 个键）

图 S5。**涵盖全部五个训练来源的代表性四帧轨迹**。前两行：受控 XFCE 桌面持久性和数学 Ctrl+L 算术。后三行：加黑边的公共 GUI 语料库（橙色圆形标记点击）。每一帧均标注了生成该帧的动作。

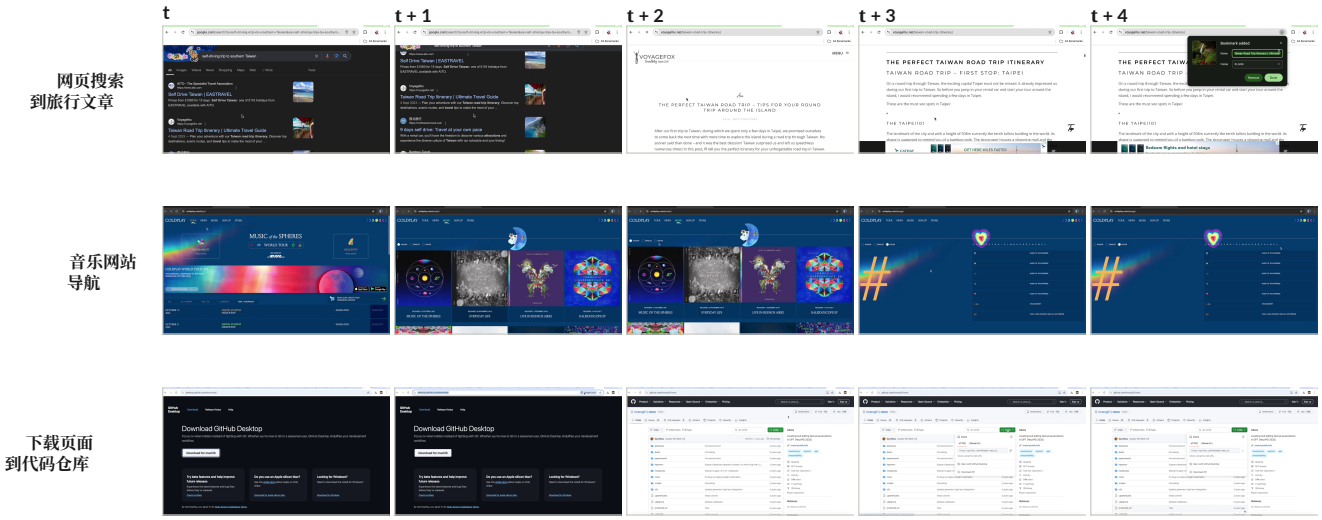
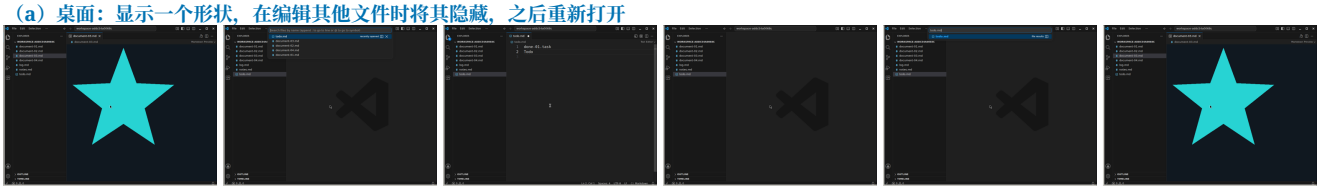



图 S6。**来自 1024×768 开放世界语料库的三个 AgentNet 回合：每个回合各包含连续五个记录步骤**。帧以完整分辨率显示，并裁去了上下黑边。回合属于最早的开发集回合，其中包含连续六步的可见变化。

**（a）桌面：显示一个形状，在编辑其他文件时将其隐藏，之后重新打开**



形状文件打开 | 7步之后 | 15步之后 | 22步之后 | 30步之后 | 37步之后：重新打开

**（b）数学：为变量赋值，清空屏幕，并计算需要该变量的查询**



输入 "f = 1" | 回车键 | Ctrl+ | 输入 "n = 394" | 回车键 | 输入 "f - 3" | 回车键（输出 -2）

图 S7。**两个受控任务分别展示为一个完整的录制回合**。（a）桌面：只有根据第一帧的记忆，才能正确绘制重新打开的帧。（b）数学：计算 f - 3 时，f 的值已不再显示在屏幕上。所有帧均是在 512×384 时录制的；每帧均标注了导致其出现的动作。




图 S8。**我们的方法在 14 次闭环 VS Code 步之后恢复了编辑后的圆形，而 NEURALOS 则恢复为编辑前的星形。**学生强制：从 $t$=17 开始，模型仅获得其自身先前的输出。未经修改的 NEURALOS（joint-272k）会跳回其训练桌面。

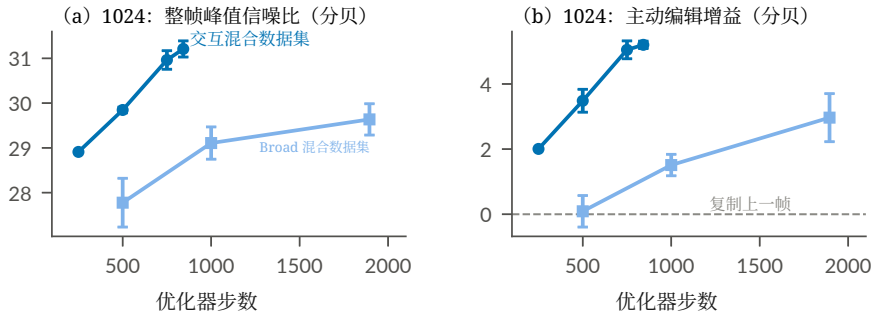



图 S9。**当训练停止时，** $1024\times768$ **渲染质量仍在提升。**（a、b）在 Dense Interactive（6.3 小时）和 Broad（21.5 小时）混合数据集上，使用 4 个 GH200 GPU 得到的整帧峰值信噪比和主动编辑增益。误差条表示 3 个训练随机种子上的标准差。

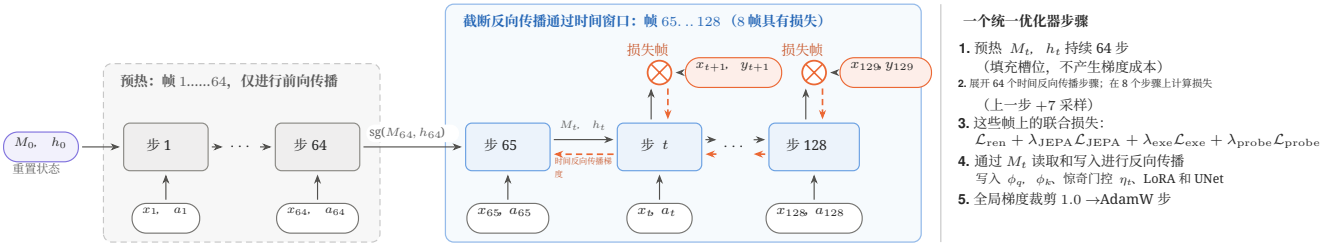



图 S10。**每个训练窗口包含 128 帧：64 个仅用于填充记忆的预热帧，然后是64 个用于训练的帧。**预热帧在没有梯度的情况下进行前向运行，因此记忆 $M_t$ 和 LSTM 状态 $h_t$ 已经保存了回合中此前发生的情况（例如，一个经过编辑后又关闭的文件）。随后将状态（$M_{64}$, $h_{64}$）分离，并在接下来的帧上应用损失（$\mathcal{L}_{\text{ren}}$、$\mathcal{L}_{\text{JEPA}}$、$\mathcal{L}_{\text{exe}} + \mathcal{L}_{\text{probe}}$ 8 64），通过这些帧内的记忆读取和写入进行反向传播。




图 S11。**我们的方法的两个随机种子都保留了新建文件夹在经历128步遮挡后，而微调后的 NEURALOS 则将其模糊或擦除。**教师强制：每一帧都根据其之前的真实帧进行预测。主图 3 的 XFCE 文件管理器任务的第二个回合（512×384）：文件夹在 $t$=3..6 已创建，并在 $t$=137 重新打开（最右列：2.5× 缩放）。

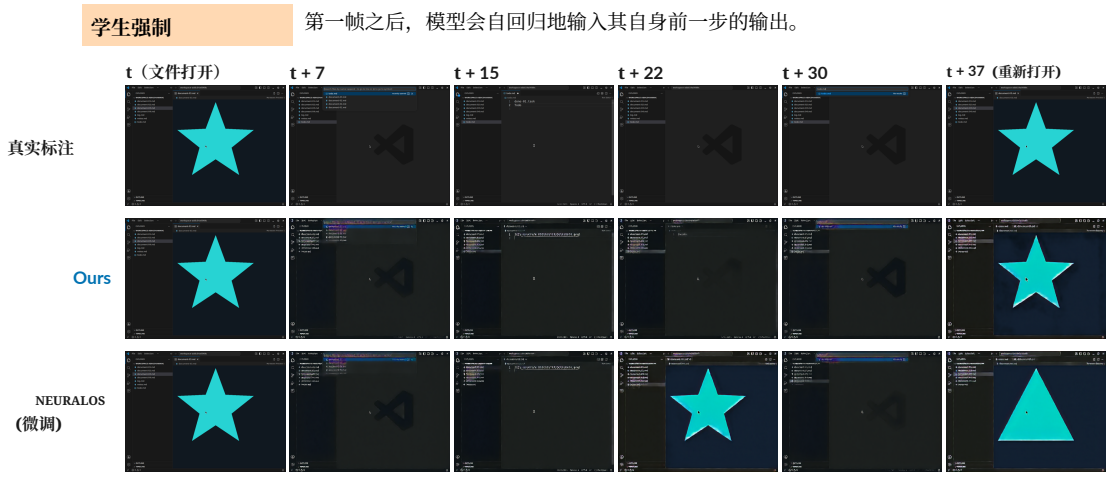



图 S12。**我们的方法在37 个闭环 VS Code 步骤中恢复了隐藏的星形，而微调后的 NEURALOS 则绘制了一个三角形。**学生强制：模型仅接收其自身前一步的输出（512×384；todo.md 中有 32 个被遮挡的步骤，重新打开时为 24.93 分贝）。NEU- RALOS 也会在 $t$+22 绘制星形，此时文件仍处于关闭状态，然后在重新打开时绘制一个三角形。

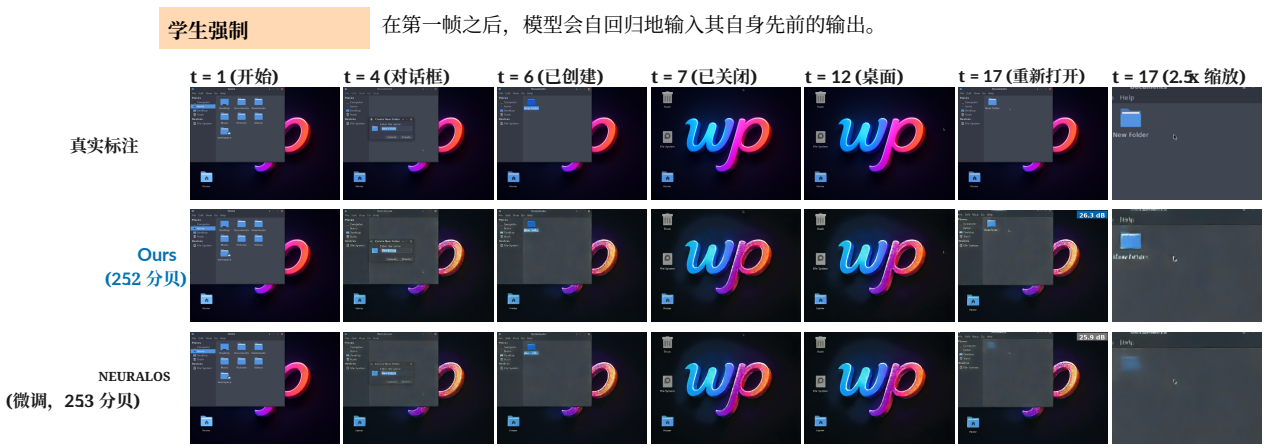



图 S13。**我们的方法在 新建文件夹 图标在 16 个学生强制文件管理器步骤中始终保持不变。** 学生强制： $t$=1 之后的每一帧都由模型自身先前的输出生成（512×384，最右列为 2.5× 缩放）： **我们的方法** 保留了图标和标签，而微调后的 NEURALOS 几乎将其擦除。两种模型的全帧峰值信噪比相同（25.2 和 25.3 分贝）；它们的差异体现在缩放图中显示的文件夹区域。

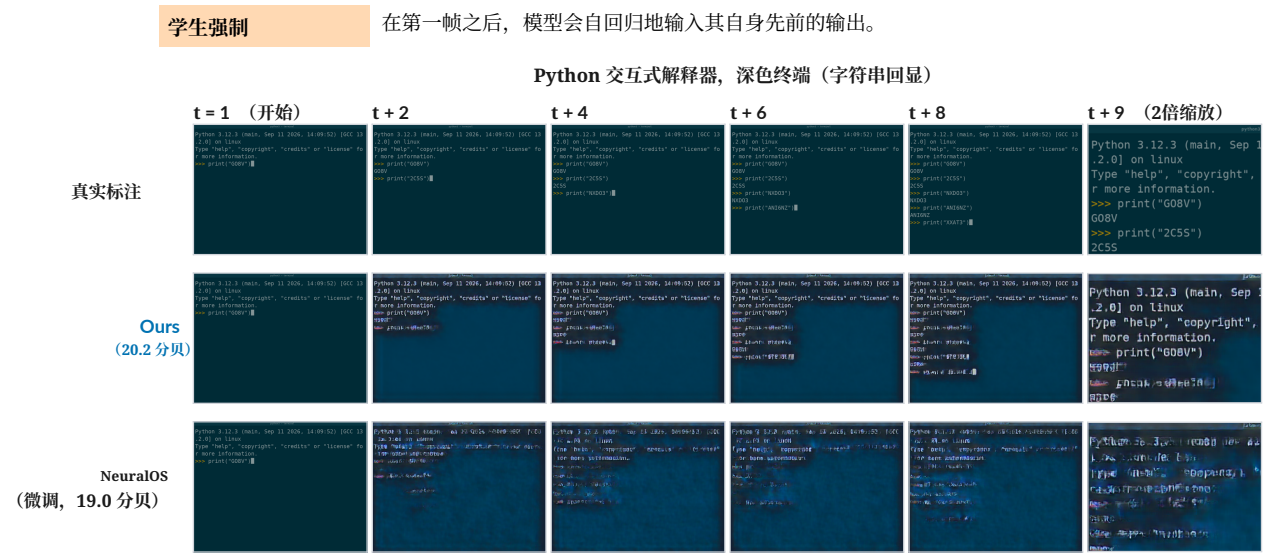



图 S14。**我们的方法在 9 个学生强制读取-求值-打印循环步骤中，仍能保持终端标题栏和第一条命令清晰可读，而微调后的 NEURALOS 在 2 步之内就丢失了标题栏。** 学生强制：模型仅接收其自身先前的输出。 512×384 在深色主题终端上的推演（最右列： 2× 缩放）。对于两种模型，后续行都会变得模糊。

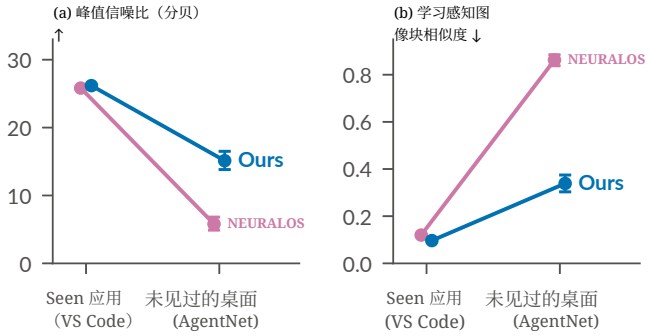



图 S15。**两个模型渲染 VS Code 的保真度相近；在未见过的桌面上，NEURALOS 的表现损失了 20。0 分贝，而我们的方法损失了 11。1 分贝。** 下一帧峰值信噪比（a；越高越好）和学习感知图像块相似度（b；一种学习得到的感知距离，越低越好）分别为512×384 模型的指标；两者均仅使用我们的 VS Code 和 Python 交互式解释器录制数据进行训练。每一帧均根据真实的前序帧和操作进行预测（教师强制）。已见应用：30 个留出的 VS Code 回合。未见过的桌面：来自 Windows、macOS 和 Ubuntu 的 30 个 AgentNet 回合。每个点先对每个回合的前 12 帧取平均，再对各回合取平均；误差条表示跨回合计算的 95% 自助法置信区间。

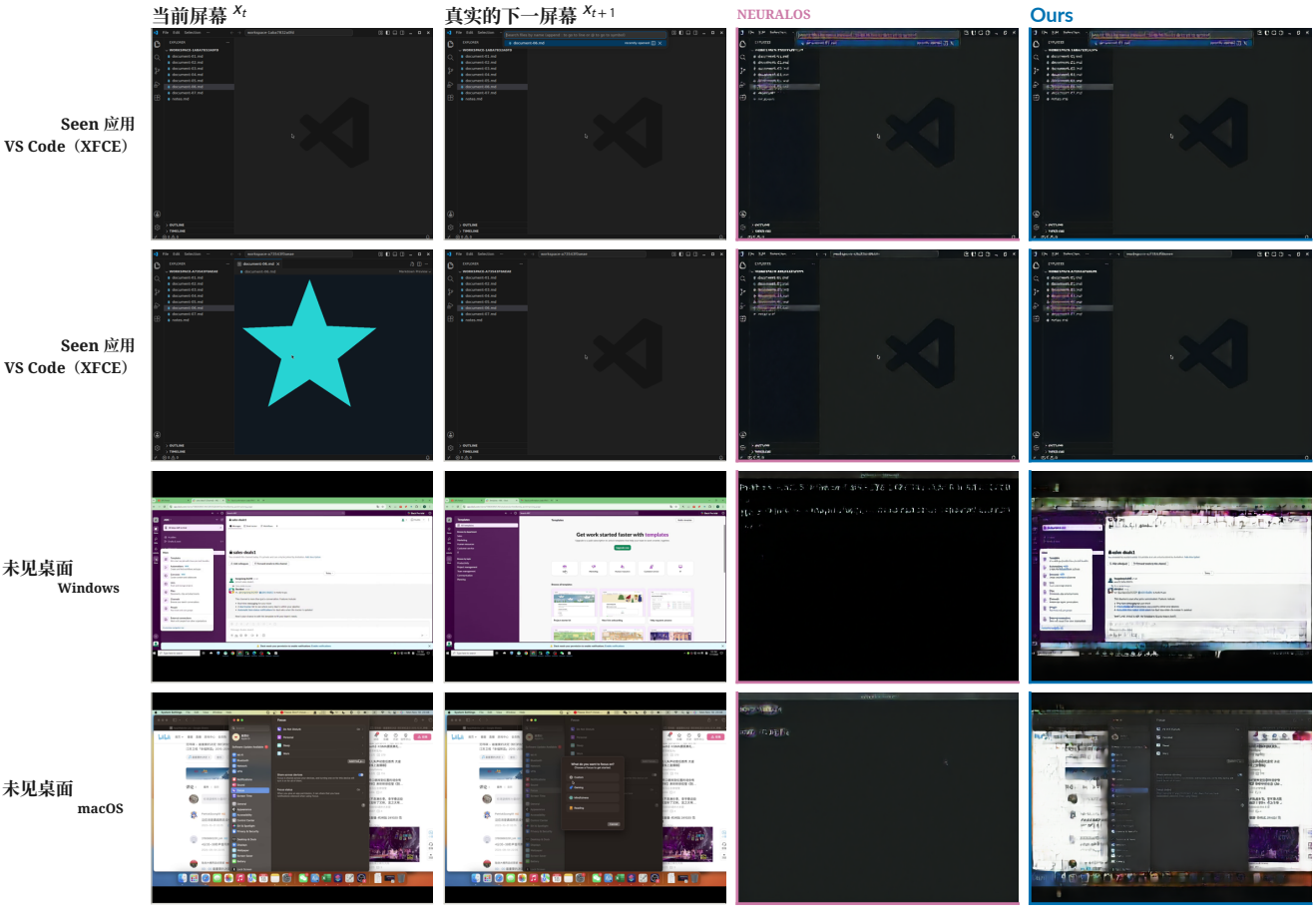



图 S16。**在一个未见桌面上，NEURALOS 将屏幕替换为黑屏；我们的方法则保留屏幕。**每一行代表一个教师强制步骤：模型接收截至 $x_t$ 的真实屏幕和用户的动作，并预测 $x_{t+1}$。在 VS Code （前两行）中，两个模型都正确重绘了屏幕，包括关闭显示星形标记的文件。在 Windows（第三行）中，真实的下一屏幕是一个新的模板页面；在 macOS（底行）中，则打开了一个对话框。NEURALOS 绘制出黑屏，Windows 行中的文字无法辨认；我们的方法保留当前屏幕，但带有斑驳的产物，既没有绘制该页面，也没有绘制该对话框。行的选择规则如下：在每组中，先筛选出至少有 2% 像素发生变化的帧，再按照我们的学习感知图像块相似度低于 NEURALOS 的幅度进行排序，并展示位于三分之一和三分之二位置的帧。帧采用原生 $512\times384$。