稀疏分布式记忆是一种持续学习器

Trenton Bricken*系统、生物合成与定量生物学 哈佛大学

Xander Davies计算机科学 哈佛学院

Deepak Singh计算机科学 哈佛学院

Dmitry Krotov麻省理工学院—IBM 沃森人工智能实验室
IBM 研究院

Gabriel Kreiman生物物理学与神经科学项目 哈佛医学院

arXiv:2303.11934v1 [cs.NE] 20 Mar 2023

摘要

持续学习是人工神经网络面临的一项难题，而其生物学对应物却擅长解决这一问题。我们基于使用稀疏分布式记忆（SDM）将核心神经回路与强大的 Transformer 模型相连接的研究，构建了一种改良的多层感知机（MLP），使其成为强大的持续学习器。我们发现，从生物学中借鉴并转化到这一 MLP 变体中的每个组成部分，对持续学习都不可或缺。我们提出的解决方案无需记忆回放或任务信息，并引入了训练稀疏网络的新方法，这些方法可能具有广泛的适用性。

1 引言

生物网络往往能够出色地持续学习新任务，而这仍是人工神经网络面临的一项棘手难题。在此，我们使用稀疏分布式记忆（SDM），将类小脑神经回路的特征融入多层感知机（MLP）；这些特征广泛存在于人类、果蝇和电鱼等多种生物体中（Modi 等人，2020年；Xie 等，2022年）。这些改造产生了一种新的 MLP 变体（称为 SDMLP），它采用 Top-K[1] 激活函数（仅使每层中 k 兴奋程度最高的神经元保持激活），不含偏置项，并对其权重和数据同时施加 L^2 归一化与非负约束。所有这些源自 SDM 的组成部分，对于使我们的模型避免灾难性遗忘都不可或缺。

训练 SDMLP 时，我们遇到了一些挑战。我们借助更多神经生物学知识来解决这些问题，从而提升持续学习性能。第一个问题是“死亡神经元”：它们对任何输入都不会激活，而这是由 Top-K 激活函数导致的（Makhzani & Frey，2014年；Ahmad & Scheinkman，2019）。参与学习的神经元越少，新的持续学习任务就越容易覆盖其中更多神经元，从而加剧灾难性遗忘。我们的解决方案模仿了“GABA 开关”现象：在发育早期，实现 Top-K 的抑制性中间神经元会兴奋 而非抑制 （Gozel & Gerstner，2021）。

第二个问题与使用动量的优化器有关：在训练高度稀疏的网络时，动量会变得“过时”。这指的是优化器仍使用过时的移动平均值来更新不活跃的神经元，导致神经元死亡，再次损害持续学习。据我们所知，我们是首个正式识别这一问题的研究者。理论上，该问题会影响任何稀疏模型，包括近期的专家混合模型（Fedus 等人，2021年；Shazeer 等人，2017年）。

我们的 SDMLP 是一种强大的持续学习器，尤其是在结合互补方法时。将我们的方法与弹性权重巩固（EWC）（Kirkpatrick 等，2017）结合，据我们所知，我们在 CIFAR-10 上取得了最先进的性能

*联系邮箱：trentonbricken@g.harvard.edu

[1]在相关文献中也称为“k 胜者全得”。

在不允许记忆回放的类别增量设置中。另一种独立于我们这项工作开发的 SDM 变体，似乎在 CIFAR-100、MNIST 和 FashionMNIST 上达到了最先进的水平，而我们的 SDMLP 紧随其后（Shen 等，2021）。

令人振奋的是，我们的持续学习成果是“有机”的，因为它源自底层模型架构，无需任何任务标签、任务边界或记忆回放。从抽象层面来看，SDM 通过两个核心模型组件，学习负责持续学习的子网络。首先，Top-K 激活函数使输入激活程度最高的 k 个神经元专门适应该输入，从而形成专门化的子网络。其次，L^2 归一化与不含偏置项相结合，将神经元约束在数据流形上，确保所有神经元都能平等地参与学习。结合这两个组件后，新的学习任务只会激活并训练一小部分神经元，使网络其余部分保持不变，从而记住先前任务而不被覆盖

作为本文的路线图，我们首先讨论相关工作（第 2 节），随后简要介绍 SDM（第 3 节），再将其转化为我们的多层感知机（第 4 节）。接下来，我们展示实验结果，将 SDMLP 的有机持续学习能力与相关基准进行比较（第 5 节）。最后，我们讨论本研究的局限性、更广泛意义上的稀疏模型，以及 SDM 如何将多层感知机与 Transformer 注意力联系起来（第 6 节）。

2 相关工作

持续学习 - 持续学习技术大致可分为三类：架构方法（Goodfellow 等，2014；2013）、正则化（Smith 等人，2022；Kirkpatrick 等，2017；Zenke 等，2017；Aljundi 等，2018）和复习（Lange 等人，2021；Hsu 等，2018）。其中许多方法利用稀疏子网络的形成来实现持续学习（Abbasi 等人，2022；Aljundi 等人，2019b；Ramasesh 等人，2022；Iyer 等人，2022；Xu 和 Zhu，2018；Mallya 和 Lazebnik，2018；Schwarz 等人，2021；Smith 等人，2022；Le 等人，2019；Le 和 Venkatesh，2022）。[2] 然而，与我们的“有机”方法相比，这些方法采用复杂的算法并增加额外的内存消耗，以明确保护对先前任务至关重要的模型权重不被覆盖。[3]

将 Top-K 激活应用于持续学习的研究包括（Aljundi 等人，2019b；Gozel & Gerstner，2021；Iyer 等人，2022）。Srivastava 等人（2013）采用了局部 Top-K，将每一层的神经元划分为互不相交的子集，并分别在每个子集中应用 Top-K。不过，这种方法仅用于一个简单的任务增量双分割 MNIST 任务，且没有采用我们发现对取得优异性能至关重要的任何额外 SDM 修改（表 2）。

Top-K 激活函数还被更广泛地应用于深度学习（Makhzani & Frey，2014；Ahmad & Scheinkman，2019；Sengupta 等，2018；Gozel & Gerstner，2021；Aljundi 等人，2019b）。Top-K 不仅与许多利用抑制性中间神经元的脑区连接方式相吻合，而且研究结果还表明，它在持续学习之外也有诸多优势，包括：更强的可解释性（Makhzani & Frey，2014；Krotov & Hopfield，2019；Grinberg 等人，2019）、对抗攻击鲁棒性（Paiton 等人，2020；Krotov & Hopfield，2018；Iyer 等人，2022）、高效的稀疏计算（Ahmad & Scheinkman，2019）、对数据流形的铺设（Sengupta 等，2018），以及通过局部赫布学习规则实现（Gozel & Gerstner，2021；Sengupta 等，2018；Krotov & Hopfield，2019；Ryali 等人，2020；Liang 等人，2020）。

FlyModel - 与我们最相关的方法是一个果蝇蘑菇体回路模型（Shen 等，2021）。该模型被称为“FlyModel”，它在无意中实现了 SDM 算法，具体而言，是采用我们也使用并将在下文论证的 Top-K 激活函数的超平面变体（Jaeckel，1989a；Keeler，1988）。FlyModel 展现出出色的持续学习性能，在不同任务上与我们的 SDMLP 交替成为最佳模型，也印证了我们论文标题所言：SDM 是一种持续学习器。

尽管这两种模型都源自 SDM，我们的工作不仅扩展了 SDM 的理论，使其能够成功学习数据流形（附录 A.4），还调和了 SDM 与

[2]即使没有稀疏性或 Top-K 激活函数，预训练模型仍然可能形成子网络，从而提升持续学习性能；Ramasesh 等人（2022年）也发现了这一点。[3]记忆回放方法通过决定回放哪些记忆，间接确定并保护权重。

多层感知机之间的差异，使 SDM 能够在深度学习框架中进行训练（包括不使用固定的神经元权重以及采用反向传播）。这两项贡献都具有创新性，或可启发持续学习以外的未来研究。例如，学习数据流形能够保留数据中的相似性，并产生更具专门性、可解释的神经元（例如，附录 H 中的图 26）。

在深度学习框架中进行训练，还使我们能够将 SDMLP 与其他基于梯度的方法结合起来，例如 FlyModel 不兼容的弹性权重巩固（EWC）（Kirkpatrick 等，2017）。此外，展示如何将多层感知机实现为小脑回路，也为如何利用神经科学中的理念（如 GABA 开关）来促进深度学习提供了范例。

3 稀疏分布式记忆背景

稀疏分布式记忆（SDM）是一种联想记忆模型，旨在解决如何将模式（记忆）存储在大脑中的问题（Kanerva，1988；1993），并且与 Hopfield 网络、小脑回路和 Transformer 注意力密切相关（Kanerva，1988；Bricken 和 Pehlevan，2021；Hopfield，1982；1984；Krotov & Hopfield，2016；Tyulmankov 等人，2021；Millidge 等人，2022）。我们简要介绍 SDM 及其相关符号，以便说明 SDM 与多层感知机的关系。关于 SDM 与小脑的关系概述，请参见附录 A.2。我们采用连续型 SDM，其中所有神经元和模式都位于 L^2 单位范数超球面上，并以余弦相似度作为距离度量（Bricken 和 Pehlevan，2021）。

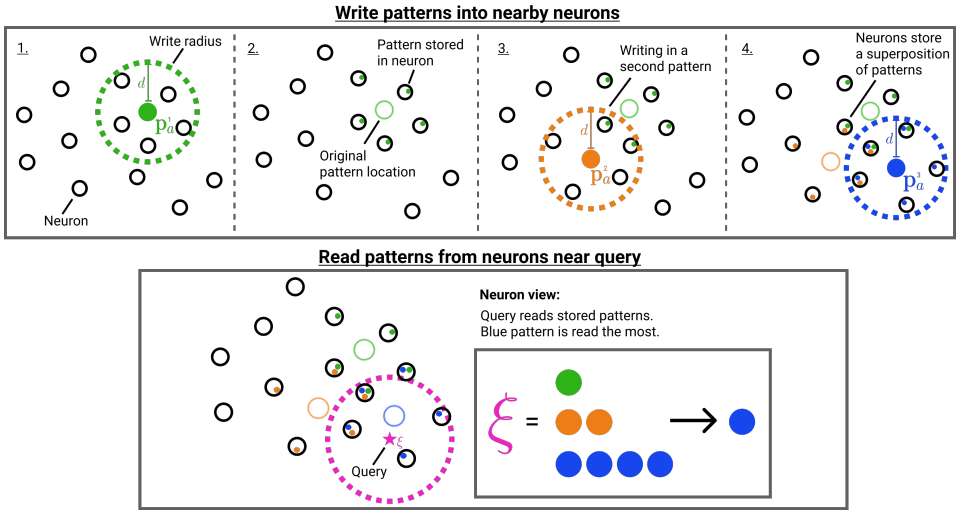


图 1：**SDM 写入和读取操作的图示概要。上排 -** 三个模式被写入附近的神经元。1. 第一次写入操作；2. 模式存储在神经元中，并显示原始模式地址；3. 写入第二个模式；4. 写入第三个模式，神经元存储多个模式的叠加。**下排 -** SDM 读取操作。查询从附近的神经元中读取，插图显示每个模式被读取的次数。蓝色模式被读取的次数最多，因为其原始模式位置（以空心圆表示）离查询最近。经许可改编自 Bricken 和 Pehlevan（2021年）的图。

SDM 在 r 个神经元的地址上进行随机初始化，这些地址位于 n 维空间中的 L^2 单位超球面上。这些神经元的地址各占据地址矩阵$X_a \in (L^2)^{n\times r}$中的一列，其中 L^2 是对单位范数超球面上所有 n 维向量的简写。每个神经元还有一个存储向量，用于存储以矩阵$X_v \in \mathbb{R}^{o\times r}$表示的模式，其中 o 是输出维度。模式也有地址，这些地址受其编码所确定的 n 维 L^2 超球面约束；编码可以很简单，例如将图像展平为向量，也可以很复杂，例如使用深度学习模型进行预处理。

通过激活余弦相似度阈值 c 范围内所有邻近的神经元，并将激活神经元的存储向量逐元素求和，从而存储模式。根据

具体任务，模式可以将自身写入存储向量（例如，在重建任务中），也可以写入另一种模式，该模式的维度可能不同（例如，在分类任务中写入其独热标签）。这一写入操作及随后的读取操作总结于图 1 中。

由于在大多数情况下神经元数量少于模式数量，同一个神经元会被多个不同模式激活。对此，我们通过前述逐元素求和操作，将模式值以叠加的形式存储。每个模式在这种叠加中的保真度取决于向量正交性和维度 n。

用 m 表示模式数量，用矩阵 $P_a \in (L^2)^{n\times m}$ 表示模式地址，用矩阵 $P_v \in \mathbb{R}^{o\times m}$ 表示模式要写入的值，则 SDM 写入操作为：

$$X_v = P_v b\big(P_a^T X_a\big), \qquad b(e) = \begin{cases} 1, & \text{if } e \geq c \\ 0, & \text{else,} \end{cases} \tag{1}$$

其中，$b(e)$ 对其输入进行逐元素二值化，以确定哪些模式地址和神经元地址之间的余弦相似度在阈值 c 以内。

将模式写入神经元后，我们通过输入查询 $\boldsymbol{\xi}$从系统中读取数据，该查询同样会激活附近的神经元。每个被激活的神经元都会输出其存储向量，并将这些向量逐元素求和，得到最终输出$\mathbf{y}$。输出$\mathbf{y}$ 可被视为更新后的查询，并可选择再次进行 L^2 归一化，作为后处理步骤：

$$\mathbf{y} = X_v b\big(X_a^T \boldsymbol{\xi}\big). \tag{2}$$

直观来说，SDM 的查询会向地址最接近的模式所对应的值更新。这是因为，与其他竞争模式相比，地址最接近的模式会将其值写入更多查询读取的神经元中。例如，在图 1 中，蓝色模式的地址离查询最近，这意味着在查询读取的附近神经元中，蓝色模式出现得最多。

若将现代 Hopfield 网络（Krotov & Hopfield，2016；Ramsauer 等人，2020；Krotov & Hopfield，2020）的循环动态限制为单步更新，它们便与 SDM 密切相关（Tyulmankov 等人，2021；Bricken 和 Pehlevan，2021；Millidge 等人，2022）。

4 将 SDM 转化为用于持续学习的多层感知机

一个单隐藏层多层感知机将输入 $\boldsymbol{\xi}$ 转换为输出 $\mathbf{y}$。采用与 SDM 兼容的记法，并用波浪号表示未归一化的连续向量，我们可以将多层感知机写为：

$$\mathbf{y} = \tilde{X}_v f(\tilde{X}_a^T \tilde{\boldsymbol{\xi}} + \tilde{\mathbf{b}}_a) + \tilde{\mathbf{b}}_v, \tag{3}$$

其中，$\tilde{X}_a \in \mathbb{R}^{n\times r}$ 和 $\tilde{X}_v \in \mathbb{R}^{o\times r}$ 分别是与我们的 SDM 神经元地址和值相对应的权重矩阵。同时，$\mathbf{b}_a \in \mathbb{R}^r$、$\mathbf{b}_v \in \mathbb{R}^o$ 是偏置参数，而 $f(\cdot)$ 是多层感知机所使用的激活函数，例如 ReLU（Glorot 等，2011）。

使用这种面向多层感知机的 SDM 记法，可以轻易看出 SDM 读取公式 2 与单隐藏层多层感知机的公式 3 之间的相似性；这种相似性最早由 Kanerva（1993）确立。然而，SDM 固定随机的神经元地址使其无法有效地对真实世界数据建模。借鉴 Keeler（1988）的思想，我们通过一种生物学上合理的抑制性中间神经元解决这一问题，并用 Top-K 激活函数对其进行近似。附录 A.3 说明了如何修改 SDM，实际使用的 Top-K 激活函数将在公式 4 中很快给出。这项修改使 SDM 能够兼容没有固定权重的多层感知机。至于公式 1 中的 SDM 写入操作，它与使用反向传播训练的多层感知机有关，具体如附录 A.6 所述。

死亡神经元问题 - Top-K 激活函数的一个问题是会产生死亡神经元（Ahmad & Scheinkman，2019；Rumelhart & Zipser，1985；Makhzani & Frey，2014；Fedus 等人，2021年）。随机初始化的神经元中，只有一部分会最接近数据流形，并进入最活跃的 k 个神经元之列，其余神经元则永远不会被激活。在持续学习情境中，引入新任务时，模型可用于学习的神经元更少，因此必须覆盖更多曾用于先前任务的神经元，从而导致灾难性遗忘。

与其使用数据流形的分解来预初始化神经元权重（Rumel- hart & Zipser, 1985; McInnes & Healy, 2018; Strang, 1993），我们不如确保训练开始时所有神经元都处于活跃状态，

从而使它们更新并映射到流形上。这种方法以前也曾用过，但采用的方式在生物学上并不合理（见附录 B.1）（Makhzani & Frey, 2014; Ahmad & Scheinkman, 2019; van den Oord 等人, 2017）。我们则借助此前引入的抑制性中间神经元，扩展了 Gozel & Gerstner（2021）提出的优雅解决方案。神经发生后，当神经元树突随机初始化时，抑制性神经递质 GABA 会激活神经元（见附录 B.2）。这意味着，最初，同一个用于通过 Top-K 抑制来实现竞争的抑制性中间神经元，会通过激活每个神经元来促成协作，让它们都能学习。因此，每个神经元都会先收敛到数据流形上；此时，GABA 会抑制神经元，Top-K 竞争随之开始，迫使每个神经元专门化，并铺满数据流形。

我们在附录 B.3 中给出了完整的 GABA 开关实现，它在生物学上最为合理。然而，借助使所有激活值均为正的正权重约束，我们发现了一种更简单、性能相同的近似方法：将我们的 k 值从该层的最大神经元数量线性退火至所需的 k。Makhzani & Frey（2014）也采用了类似方法，但没有生物学依据；他们将非最活跃的 k 个神经元的值全部置零，并保持其余 k 不变。受抑制性中间神经元的启发，我们改为从所有仍处于激活状态的神经元中减去第 $k+1$ 个激活值。这一改动显著提升了持续学习性能（见表 2 和附录 B.3）。形式化地，令抑制前的神经元激活值为$\mathbf{a} \coloneqq X_a^T \boldsymbol{\xi}$，抑制后的神经元激活值为$\mathbf{a}^*$：

$$\begin{aligned} \mathbf{a}^* &= [\mathbf{a} - I]_+ \\ I &= \text{descending-sort}([\mathbf{a}]_+)_{(k_t+1)} \\ k_t &= \max\left(k_{\text{target}}, \lfloor k_{\max} - E_t(k_{\max} - k_{\text{target}})/s \rfloor\right), \end{aligned} \tag{4}$$

其中，$[\cdot]_+$ 表示 ReLU 操作，$\lfloor \cdot \rfloor$ 表示向下取整运算，以确保 k_t 为整数；descending- sort(·)按降序排列神经元激活值，以找出第 $k+1$ 大的激活值。E_t 是表示当前训练轮次的整数，下标 t 表示 k_t 和 E_t 会随时间变化。超参数 s 设定了 k 从初始值 $k_{\max}$变化到目标值 k_{target}所需的训练轮次数。当 $k_t = k_{\text{target}}$ 时，我们近似实现的 GABA 开关会抑制每个神经元。附录 A.1 中的算法框总结了 SDMLP 各组成部分的作用。

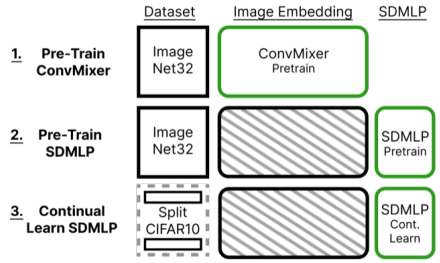
陈旧动量问题 - 我们发现，即使使用 GABA 开关，一些优化器仍会导致大量神经元逐渐失活，从而损害持续学习。为探究原因，我们提出了“陈旧动量”问题：在稀疏激活情形下，采用某种形式动量的优化器（尤其是 Adam 和 RMSProp）无法准确计算先前梯度的移动平均（Kingma 和 Ba，2015；Geoffrey Hinton，2012）。这些动量优化器不仅会更新不在 Top-K 中的非激活神经元，还会在神经元静默一段时间后重新激活时，使梯度幅值激增。我们在附录 D 中进一步解释并研究陈旧动量，并采用不带动量的随机梯度下降作为解决方案。

其他修改 - SDM 与多层感知机之间还有五项较小的差异（详见附录 A.5）：(i) 使用频率编码，以避免不可微的二值激活；(ii) 使用L^2 对输入和权重进行归一化，作为对比编码和异突触性的近似（Sterling & Laughlin, 2015；Rumelhart & Zipser，1985；Tononi & Cirelli，2014）；(iii) 移除偏置项，因为它们假设存在小脑颗粒细胞所不具备的紧张性基线放电率（Powell 等，2015；Giovannucci 等，2017）；(iv) 仅使用符合戴尔定律（Dale，1935）的正（兴奋性）权重；(v) 使用反向传播，作为实现赫布学习规则的一种更高效（尽管可能不符合生物学特性（Lillicrap 等，2020））的方法（Krotov 和 Hopfield，2019；Sengupta 等，2018；Gozel & Gerstner，2021）。

5 SDM 避免灾难性遗忘

这里我们展示，所提出的 SDMLP 架构能够实现强大且自然的持续学习。由于我们只修改模型架构，因此我们的方法与其他持续学习策略高度兼容，例如重要权重的正则化（Kirkpatrick 等，2017；Aljundi 等，2018；Zenke 等，2017）和记忆回放（Zhang 等人，2021；Hsu 等，2018），而大脑很可能也会以某种方式运用这两种策略。我们展示了将 SDMLP 与权重正则化结合能够带来正向的累加效果，但未探索记忆回放。

实验设置 - 为了尽可能贴近现实中的持续学习设置，我们在类别增量设置下使用拆分 CIFAR10，并在 ImageNet 上进行预训练（Russakovsky 等，2015）。这会将 CIFAR10 划分为互不相交的子集，每个子集包含两个类别。例如，第一个数据划分包含类别 5 和 2，第二个划分包含类别 7 和 9，依此类推。CIFAR 比 MNIST 更复杂，也能体现真实世界的图像统计特性。类别增量设置比任务增量学习更困难，因为模型需要对所有 CIFAR 类别进行预测，而不只是区分当前任务中的两个类别（Hsu 等，2018；Farquhar 和 Gal，2018）。在 ImageNet 上预训练能够学习通用的图像统计特性；此时 GABA 开关会启动，使神经元能够专门化并分布于数据流形上。在正文中，我们展示了相关结果：ImageNet32 和 CIFAR 数据集被压缩为 256 维的潜在嵌入，这些嵌入取自一个在 ImageNet32 上预训练的冻结 ConvMixer 的最后一层（图 2 中训练流程的第 #1 步）（Trockman 和 Kolter，2022；Russakovsky 等，2015）。[4]



方法	神经元	k	验证准确率
SDMLP	10K	10	0.71
FlyModel	10K	32	0.82
MAS	10K	NA	0.67
SI	10K	NA	0.44
ReLU	10K	NA	0.21
EWC	10K	不适用	0.65
SDMLP+MAS	10K	10	0.84
SDMLP+EWC	10K	10	**0.86**
预言机	10K	NA	0.93

图 2：**左侧**为三步训练流程。绿色轮廓表示当前正在训练的模块。ConvMixer 上的灰色斜线表示其被冻结的阶段。表 1：**右侧**为拆分 CIFAR10 最终验证准确率。我们突出显示了 10K 神经元设置下表现最佳的 SDMLP、基线和**整体最佳模型**。预言机在完整的 CIFAR10 数据集上训练。所有结果均为 5 个随机任务划分的平均值。

使用图像嵌入，我们在 ImageNet 上预训练模型（图 2 的步骤 #2）。随后重置学习类别标签的 X_v，并在拆分 CIFAR10 上开始持续学习，在五个数据划分中的每个划分上对每个模型训练 2,000 个训练轮次（图 2 的步骤 #3）。我们使用如此多的训练轮次，是为了促使模型遗忘先前任务。我们在当前数据划分以及此前见过的每个数据划分上测试模型性能，以评估遗忘情况。我们使用五个不同的随机种子将 CIFAR 数据集划分为互不重叠的集合，以确保结果不受数据划分顺序及各划分所含类别的影响。

我们使用预处理后的图像嵌入，因为单隐藏层多层感知机难以直接从图像像素中学习。这种预处理也代表了视觉处理通路：它会压缩供大脑更深层脑区使用的图像（Pisano 等人，2020年；Li 等人，2020年；Yamins 等人，2014年）。不过，我们进行了两项消融实验。它们会改变验证准确率，但不会改变模型的排名顺序或得出的结论：(i) 移除 ConvMixer 嵌入，改为直接在图像像素上训练（移除图 2 中的步骤 #1，附录 E.1）；(ii) 在不进行任何 ImageNet32 预训练的情况下测试持续学习（移除图 2 中的步骤 #2，附录 E.2）。我们还针对 CIFAR100（附录 F.2）、MNIST（附录 F.3）和 FashionMNIST（附录 F.4）进行了实验。

模型参数和基线方法 -除非另有说明，所有模型均为多层感知机，且在单个隐藏层中包含 1,000 个神经元。使用 Top-K 激活函数时，我们设置 $k_{\text{target}} = 10$，并同时展示 $k_{\text{target}} = 1$。我们测试了其他 k 值，并在附录 C.2 中说明如何选择最佳的 k_{target} 值。由于所考察的 k 值高度稀疏——因此可节省浮点运算次数和内存消耗——我们还评估了包含 10,000 个神经元的设置，尤其是该设置提升了 SDM 和 FlyModel 的持续学习能力。

[4]这个版本的 ImageNet 已重新缩放为 32 × 32 的尺寸，并经过了 300 个训练轮次（Russakovsky 等，2015年）。

我们实现的最简单基线方法包括 ReLU 激活函数、L2 正则化和 Dropout；它们都同样自然地不使用任何任务信息（Goodfellow 等，2014）。我们还与广受欢迎的基于正则化的方法进行比较：弹性权重巩固（EWC）、记忆感知突触（MAS）和突触智能（SI）（Kirkpatrick 等，2017；Aljundi 等，2018；Zenke 等，2017；Smith 等，2022）。这些方法会推断出对每个任务都至关重要的模型权重，并在损失中使用正则化项来惩罚对这些权重的更新。为了跟踪每个参数，这至少需要使内存消耗翻倍；此外，还需要提供任务边界，并为损失项指定一个系数。[5]

在类别增量学习设定中，已有研究表明这些正则化方法会发生灾难性遗忘（Hsu 等，2018；Gurbuz 和 Dovrolis，2022）。然而，我们发现这些结果并不准确，通过仔细进行超参数调优可以取得更好的表现。这包括为 EWC 和突触智能的 softmax 输出添加一个 β 系数，以缓解梯度消失问题（见附录 G.1）。据我们所知，以往文献中尚未使用过这种简单的修改，但我们采用它来提高基线方法的挑战性。

我们还将结果与强调生物学合理性和稀疏性的相关工作进行比较。FlyModel 表现出色，而且不需要任务信息（Shen 等人，2021年）。该模型所用的具体训练参数见附录 G.2。此外，我们还测试了主动树突（Iyer 等人，2022年）和 NISPA（神经启发的稳定性-可塑性自适应）（Gurbuz 和 Dovrolis，2022年）。不过，这两种模型都采用了较简单的任务增量设置，在类别增量设置下泛化能力不佳；尤其是主动树突，即便是在最简单的拆分 MNIST 数据集上也表现如此（附录 F.3）。[6] 任务增量方法无法应用于类别增量设置，这一问题此前已有记载（Farquhar 和 Gal，2018年）。这些算法也不够自然，需要任务标签，且实现复杂得多，生物学合理性也较低。我们还考虑了其他一些基线方法，但发现它们在类别增量设置下都缺乏已有结果，而且往往没有公开的代码库。[7]

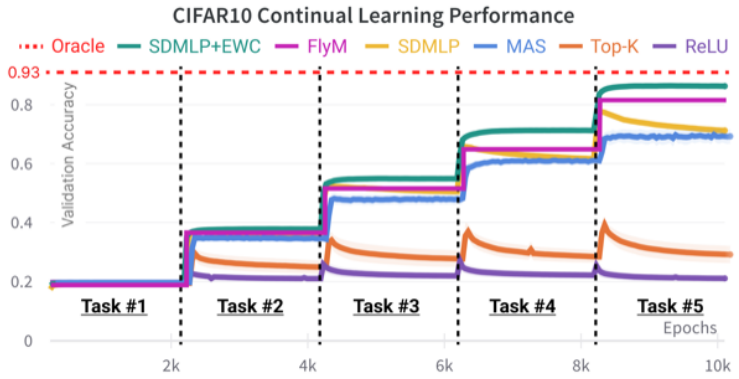


图 3：**SDM 的表现优于正则化方法，而将它们结合起来能带来正向增益**。我们绘制了拆分 CIFAR10 上表现最佳的方法和基线方法在各任务上的验证准确率。SDMLP+EWC（绿色曲线）表现最好，其次是 FlyModel（洋红色），然后是 SDMLP（黄色），MAS（蓝色）紧随其后。未采用 SDM 修改的 Top-K（橙色）和 ReLU（紫色）基线方法表现不佳。按照 Shen 等人（2021）的做法，FlyModel 在每个任务上只训练了一个训练轮次；不过，为了便于比较方法，我们在图中将其在每个任务上的验证准确率曲线作了视觉延伸。附录 F.1 展示了与 ReLU 基线方法的灾难性遗忘相比，SDMLP 如何逐渐遗忘各个任务。我们取 5 个随机种子的平均值，并用误差条表示均值标准误；但由于方差很小，误差条不易看清。

[5]我们承认，可以通过无监督方式推断任务边界，但这需要进一步增加复杂性；我们预计，使用真实任务边界能够取得最佳性能（Aljundi 等，2019a）。[6]由于我们没有测试记忆回放方法，因此忽略了针对类别增量设置开发的 NISPA 修改版（Gurbuz 和 Dovrolis，2022）。[7]我们的所有代码和训练参数均可在公开的代码库中找到：https:// github.com/TrentBrick/SDMContinualLearner。

CIFAR10 结果 - 我们在图 3 及其对应的表 1 中展示了各方法在持续学习任务中的最佳结果。SDMLP 自然地优于所有正则化基线，其最佳验证准确率为 71%，而 MAS 为 67%。在 1K 个神经元的设置下（见附录 F.1 中的表 5），SDMLP 与 $k = 1$ FlyModel 并列，准确率均为 70%。在 10K 个神经元的设置下，FlyModel 表现更佳，准确率达到 82%，而 SDMLP 为 71%。不过，SDMLP 与 EWC 结合后表现最强，准确率达到 86%；与在 CIFAR10 上同时训练所得的 93% 预言机相比，几乎没有遗忘任何信息。[8]

SDMLP 与基于正则化的方法互为补充，并作用于不同的抽象层次。SDMLP 学习哪些神经元子集至关重要，而正则化方法则学习哪些权重子集至关重要。由于 FlyModel 将固定权重与 Top-K 结合起来，因此可以将其视为 SDMLP 与正则化方法结合后更为刚性的形式。FlyModel 也受益于稀疏权重，这使神经元彼此更加正交，能够针对不同任务作出独特响应。我们尝试将这种权重稀疏性融入 SDMLP，使神经元能够学习数据流形，同时受到更多约束，仅对应于数据的某个子集；理论上，这样可以减少任务间的串扰和遗忘。有趣的是，我们最初尝试随机剪枝权重，或仅剪除 ImageNet 预训练模型中权重最小的部分，结果都导致了灾难性遗忘。我们将对权重稀疏性的更深入探索留待未来工作。

消融实验 - 我们对 SDMLP 的各个组件进行消融实验，以确定哪些组件有助于持续学习，结果见表 2。附录 H 提供了关于各项消融实验为何失败的深入分析；此处我们重点介绍最重要的两点：首先，Top-K 激活函数可确保对于任何给定输入，更新的神经元不超过 k 个，从而限制可能遗忘信息的神经元数量。一小部分神经元会自然地专门适应新的数据类别，并持续朝其方向更新，从而保护网络其余部分不被覆盖。附录 C.2 中的结果支持这一理论：k 越小，发生的灾难性遗忘就越少。我们还通过展示附录 H 中学习每项任务时激活的神经元数量，进一步证明了 k 的重要性。
其次，L^2 归一化约束以及不含隐藏层偏置项这一设计，对于使所有神经元保持在 L^2 超球面上至关重要，从而确保它们都能平等地参与。若缺少这两项约束，但仍使用 Top-K 激活函数，我们发现一小部分神经元会变得“贪婪”，拥有更大的权重范数或偏置项，并始终胜过其他神经元。正如表 2 的消融实验所示，这会导致每个新任务都只有少数神经元更新，从而灾难性地遗忘先前的任务。图 6 展示了流形铺砌的证据：图中是在预先使用 ImageNet32 嵌入进行训练的 SDM权重（橙色）上拟合的 UMAP（McInnes & Healy，2018）图。我们将嵌入后的 CIFAR10 训练数据（蓝色）投影到图中，以展示预训练使 SDM 子网络覆盖了 ImageNet32 中包含的通用图像统计特性流形。这种流形铺砌与附录 H 中展示的其他方法形成鲜明对比，并使 SDM 能够避免灾难性遗忘。作为流形铺砌的进一步证据，我们在同一附录中展示，直接使用 CIFAR10 图像进行训练时，SDM权重通常具有很强的可解释性。

作为对公式 2 所定义的 SDMLP 的一项补充消融实验，我们发现，在输出层引入偏置项会破坏持续学习。这是因为模型会为当前任务中的类别赋予较大的偏置值，而相较之下，先前任务中的类别所得偏置值较小，从而呈现出灾难性遗忘的表象。有趣的是，这种效应仅出现在 SDM 中；通过移除输出层偏置项来修改其他所有基线方法，都不会影响它们的持续学习表现。

6 讨论

着手将 SDM 实现为多层感知机时，我们引入了若干修改，最终得到一个能够持续学习的模型。这些修改受到生物学的启发，融入了小脑神经生物学的相关组成部分，而这些组成部分共同构成强大持续学习能力的必要条件。我们针对死神经元问题提出的新解决方案，以及对陈旧动量问题的识别，是取得这些成果的关键，也可能进一步推广到其他稀疏神经网络架构。

[8]我们发现，对于组合式 SDMLP+EWC，SGDM 的表现略优于 SGD，因此本文采用 SGDM；而对于所有其他方法，SGD 的表现最佳。

Name	验证准确率
SGD	0.63
SGDM	0.54
Adam	0.23
RMSProp	0.20
线性减法	0.63
线性掩码	0.38
无线性退火	0.35
负权重	0.57
无 L^2 归一化	0.20
隐藏层偏置项	0.20
输出层偏置项	0.20

表 2：**SDMLP 消融实验**。在 CIFAR10 嵌入和 $k = 10$ 上，对 SDMLP 的消融版本进行验证所得的准确率，使用 1,000 个神经元（与表 1 中相同的任务）。详情请参阅正文。

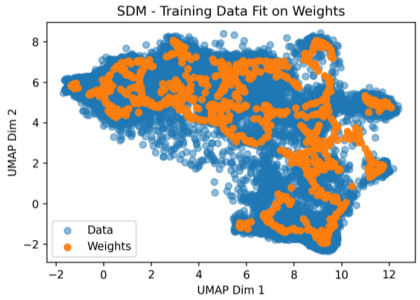


图 6：**SDM 学习数据流形**。在 ImageNet32 上预训练的 CIFAR10 嵌入（蓝色）和 SDM权重（橙色）的 UMAP 表示。其他方法中的权重与数据之间缺乏重叠（参见附录 H）。

鉴于 SDM 与 Transformer 注意力之间的联系（Bricken 和 Pehlevan，2021），仅需少量修改便能将 SDM 写成多层感知机，这一点颇有意义。这一联系也与相关研究相吻合：研究表明，Transformer多层感知机层能够执行类似联想记忆的操作，并通过在较后层呈现高达 90% 的激活稀疏性来近似 Top-K（Geva 等，2020；Sukhbaatar 等，2019；Nelson 等，2022）。从 SDM 的视角审视注意力和多层感知机，可以看出二者各自的权衡：注意力作用于模型当前感受野中的模式。这提高了写入和读取操作的保真度，因为模式无需依赖那些无法很好地近似其原始地址位置、且会以噪声叠加形式存储值的神经元。然而，在多层感知机中，模式存储于神经元并从中读取，此时需要以增加噪声为代价，才能存储超出当前感受野的模式。这有助于从模式的叠加中学习统计规律，形成有用的原型，并可能提升泛化能力（Irie 等，2022）。

局限性 - 我们面临的最大局限，仍是确保 SDM 能够成功避免死神经元问题，并铺满数据流形，从而实现持续学习。设置 k_{target} 和 s衰减率仍然很困难，而且取决于数据流形（附录 C.2）。目前，要避免死亡神经元，就需要在静态数据流形上训练 SDM，无论该流形是图像像素还是固定嵌入。早期实验尝试在 ConvMixer 中交错放置 SDM 模块进行联合训练，或将其放在网络末端进行训练，结果都出现了大量死亡神经元，且无法持续学习。我们认为，这在很大程度上是因为流形会随时间不断变化。

将神经元限制在数据流形上，并且只允许高度稀疏的子集激活，在非持续学习场景中也不利于最大限度地提高分类准确率。增大 k可以缓解这一问题，但会牺牲持续学习能力。然而，硬件加速器利用稀疏性的能力，应能让网络变得更宽（增大 r），而非减小 k，从而有助于缓解这一问题（Wang, 2020；Gale 等，2020）。最后，尽管我们改进后的 SDM 版本在小脑回路中为五种不同的细胞类型分配了功能，但仍有更多回路结构有待映射，例如篮状细胞和小脑深部核团（Sezener 等，2021）。

结论 - 我们证明，当 SDM 能够以符合小脑神经生物学的方式对相关数据流形进行建模时，它就是一种强大的持续学习器。这一结果令人瞩目，因为 SDMLP 的每个组成部分——包括 L^2 归一化、GABA 开关和无动量优化——都是取得这些持续学习成果的必要条件。除了为持续学习设立一个新的“有机”基线之外，我们还通过解决神经元死亡和过时动量问题，帮助确立了在深度学习场景中训练稀疏模型的方法。更广泛地说，这项工作将 SDM 的神经生物学映射扩展到小脑回路，并将其与多层感知机联系起来，从而加深了其与大脑和深度学习之间的联系。

致谢

感谢 Beren Millidge 博士、Joe Choo-Choy、Miles Turpin、Blake Bordelon、Stephen Casper、Mengmi Zhang 博士、Tomaso Poggio 博士和 Pentti Kanerva 博士提供宝贵的启发、讨论和反馈。我们也要感谢帮助促成本研究的开源软件贡献者，其中包括但不限于：NumPy、Pandas、SciPy、Matplotlib、PyTorch 和 Anaconda。本研究得到了 NIH R01EY026025 号资助、NSF CCF- 1231216 号资助以及 NSF 研究生研究奖学金计划的支持。

作者贡献

- Trenton Bricken 提出了项目构想和理论，完成了所有实验，并在合作者的建议下撰写了论文。
- Xander Davies 与 T.B 共同发现了陈旧动量，开展了关于死亡神经元的早期研究，审阅了相关工作，并参与了讨论。
- Deepak Singh 审阅了鲁棒性和 Top-K 相关工作，并参与了讨论。
- Dmitry Krotov 为持续学习实验、Top-K 学习理论以及与联想记忆模型的关系提供了建议。
- Gabriel Kreiman 指导了该项目，并在理论和实验方面全程提供建议。

参考文献

稀疏 Adam。https://pytorch.org/docs/stable/generated/torch.optim.SparseAdam.html。访问日期：2022-05-18。

Ali Abbasi、Parsa Nooralinejad、Vladimir Braverman、Hamed Pirsiavash 和 Soheil Kolouri。《神经激活零空间中的稀疏性与异质 dropout：用于持续学习》。*ArXiv*，abs/2203.06514，2022年。

Diego Aguirre。用于深度学习架构的一组新型权重初始化技术。201 9.

S. Ahmad 和 Luiz Scheinkman。我们怎会如此稠密？使用高度稀疏表征的益处。*ArXiv*，abs/1903.11257，2019。

Rahaf Aljundi、Francesca Babiloni、Mohamed Elhoseiny、Marcus Rohrbach 和 Tinne Tuytelaars。记忆感知突触：学习该忘记什么（以及不该忘记什么）。发表于*ECCV*，2018年。

Rahaf Aljundi、Klaas Kelchtermans 和 Tinne Tuytelaars。无任务持续学习。*2019*年*IEEE/CVF*计算机视觉与模式识别会议（*CVPR*），第11246–11255页，2019a。

Rahaf Aljundi、Marcus Rohrbach 和 TinneTuytelaars。无私的序列学习。*ArXiv*，abs/1806.05421，2019b。

David Attwell 和 Simon B. Laughlin。《大脑灰质信号传递的能量预算》。《脑血流与代谢杂志》，21：1133 – 1145，2001年。

Jimmy Lei Ba、Jamie Ryan Kiros 和 Geoffrey E. Hinton。《层归一化》，2016。

Trenton Bricken 和 Cengiz Pehlevan。《注意力近似稀疏分布式记忆》。*NeurIPS*，2021年。

Michael Chang、Sid Kaushik、S. Matthew Weinberg、Thomas L. Griffiths 和 Sergey Levine。《去中心化强化学习：通过本地经济交易进行全局决策》。发表于*ICML*，2020a。

Oscar Chang、Lampros Flokas 和 Hod Lipson。面向超网络的原理性权重初始化。载于*ICLR*，2020b。

Aakanksha Chowdhery、Sharan Narang、Jacob Devlin、Maarten Bosma、Gaurav Mishra、Adam Roberts、Paul Barham、Hyung Won Chung、Charles Sutton、Sebastian Gehrmann、Parker Schuh、Kensen Shi、Sasha Tsvyashchenko、Joshua Maynez、Abhishek Baindoor Rao、Parker Barnes、Yi Tay、Noam M. Shazeer、Vinodkumar Prabhakaran、Emily Reif、Nan Du、Benton C. Hutchinson、Reiner Pope、James Bradbury、Jacob Austin、Michael Isard、Guy Gur-Ari、Pengcheng Yin、Toju Duke、Anselm Levskaya、Sanjay Ghemawat、Sunipa Dev、Henryk Michalewski、Xavier 初始化 Garc´ıa、Vedant Misra、Kevin Robinson、Liam Fedus、Denny Zhou、Daphne Ippolito、David Luan、Hyeon- taek Lim、Barret Zoph、Alexander Spiridonov、Ryan Sepassi、David Dohan、Shivani Agrawal、Mark Omernick、Andrew M. Dai、Thanumalayan Sankaranarayana Pillai、Marie Pellat、Aitor Lewkowycz、Erica Oliveira Moreira、Rewon Child、Oleksandr Polozov、Katherine Lee、Zongwei Zhou、Xuezhi Wang、Brennan Saeta、Mark Diaz、Orhan Firat、Michele Catasta、Jason Wei、Kath- leen S. Meier-Hellstern、Douglas Eck、Jeff Dean、Slav Petrov 和 Noah Fiedel。《PaLM：利用路径扩展语言建模》。*ArXiv*，abs/2204.02311，2022年。

J. A. Connor、H Y Tseng 和 P. E. Hockberger。在外植体培养中，去极化和神经递质诱导的大鼠小脑颗粒细胞内钙2+ 变化。《神经科学学会官方期刊：神经科学杂志》，1987。

Henry Hallett Dale。《药理学与神经末梢》（Walter Ernest Dixon纪念讲座）：（治疗学与药理学分会）。《皇家医学会会刊》，28 3: 319–32，1935年。Peter Dayan 和 L. F. Abbott。《理论神经科学：神经系统的计算与数学建模》。2001年。Matasha Dhar、Adam W. Hantman 和 Hiroshi Nishiyama。体内小脑颗粒细胞树突存活的发育模式与结构因素。《科学报告》，8，2018年。Sebastian Farquhar 和 Yarin Gal。迈向稳健的持续学习评估。*ArXiv*，abs/1805.09733，2018年。W. Fedus、Barret Zoph 和 Noam Shazeer。Switch Transformer：借助简单高效的稀疏性扩展至万亿参数模型。*ArXiv*，abs/2101.03961，2021年。Jonathan Frankle 和 Michael Carbin。彩票假说：寻找稀疏且可训练的神经网络。*arXiv:* 学习，2019。Trevor Gale、Matei A. Zaharia、Cliff Young 和 Erich Elsen。用于深度学习的稀疏 GPU 内核。发表于*SC*，2020年。Karunesh Ganguly、Alejandro F. Schinder、Scott T. Wong 和 Mu-Ming Poo。GABA本身会促进神经元GABA能反应从兴奋向抑制的发育性转换。《细胞》，105:521–532，2001年。Kevin Swerksy Geoffrey Hinton、Nitish Srivastava。机器学习神经网络课程第6a讲：小批量梯度下降概述。Coursera深度学习课程幻灯片，2012年。Mor Geva、R. Schuster、Jonathan Berant 和 Omer Levy。Transformer前馈层是键值记忆。*ArXiv*，abs/2012.14913，2020年。Andrea Giovannucci、Aleksandra Badura、Ben Deverett、Farzaneh Najafi、Talmo D. Pereira、Zhenyu Gao、Ilker Ozden、Alexander D Kloth、Eftychios A. Pnevmatikakis、Liam Paninski、Chris I. De Zeeuw、Javier F. Medina 和 Samuel S.-H. Wang。小脑颗粒细胞在运动学习过程中获得广泛的预测性反馈信号。《自然-神经科学》，20:727–734，2017年。Xavier Glorot 和 Yoshua Bengio。理解深度前馈神经网络训练的难点。发表于*AISTATS*，2010年。Xavier Glorot、Antoine Bordes 和 Yoshua Bengio。深度稀疏整流神经网络。发表于*AISTATS*，2011年。Ian J. Goodfellow、David Warde-Farley、Mehdi Mirza、Aaron C. Courville 和 Yoshua Bengio。Maxout网络。发表于*ICML*，2013。

Ian J. Goodfellow、Mehdi Mirza、Xia Da、Aaron C. Courville 和 Yoshua Bengio。对基于梯度的神经网络灾难性遗忘的实证研究。*CoRR*，abs/1312.6211，2014年。Ian J. Goodfellow、Yoshua Bengio 和 Aaron C. Courville。《深度学习》。《自然》，521:436–444，2015年。Olivia Gozel 和 Wulfram Gerstner。成年齿状回神经发生的功能模型。*eLife*，10，2021年。Leopold Grinberg、John J. Hopfield 和 Dmitry Krotov。用于图像分析的局部无监督学习。*ArXiv*，abs/1908.08993，2019。Charles G. Gross。“祖母细胞”的谱系。《神经科学家》，8:512 – 518，2002年。Mustafa Burak Gurbuz 和 Constantine Dovrolis。NISPA：面向稀疏网络持续学习的神经启发式稳定性—可塑性适应。*ArXiv*，abs/2206.09117，2022年。Stefanie Heigele、S´ebastien Sultan、Nicolas Toni 和 Josef Bischofberger。GABA能对新生海马颗粒细胞动作电位发放的双向控制。《自然-神经科学》，19: 263–270，2016。Dan Hendrycks 和 Kevin Gimpel。高斯误差线性单元（GELU）。*arXiv:* 学习，2016。

J J Hopfield。具有涌现集体计算能力的神经网络与物理系统。美国国家科学院院刊，79(8):2554–2558，1982。ISSN 0027-8424。doi: 10.1073/pnas.79.8.2554。URL https://www.pnas.org/content/79/8/2554。

John J Hopfield。具有分级响应的神经元拥有与双态神经元相似的集体计算特性。美国国家科学院院刊，81(10):3088–3092，1984。Eriola Hoxha、F. Tempia、P. Lippiello 和 M. C. Miniaci。平行纤维-浦肯野细胞突触的调节、可塑性与病理生理学。《突触神经科学前沿》，8，2016。Yen-Chang Hsu、Yen-Cheng Liu 和 Zsolt Kira。重新评估持续学习场景：分类方法及采用强基线的理由。*ArXiv*，abs/1810.12488，2018。AI Impacts。人类的神经元发放率，2022年5月。URL https://aiimpacts.org/ rate-of-neuron-firing/。Sergey Ioffe 和 Christian Szegedy。批归一化：通过减少内部协变量偏移加速深度网络训练。*ArXiv*，abs/1502.03167，2015。Kazuki Irie、R’obert Csord’as 和 J¨urgen Schmidhuber。重新探讨神经网络的对偶形式：通过注意力聚光灯将测试时预测与训练模式联系起来。收录于*ICML*，2022。Abhiram Iyer、Karan Grewal、Akash Velu、Lucas O. Souza、Jeremy Forest 和 Subutai Ahmad。避免灾难：主动树突使动态环境中的多任务学习成为可能。收录于《神经机器人学前沿》，2022。

Louis A. Jaeckel。稀疏分布式记忆的一类设计。1989a。

Louis A. Jaeckel。稀疏分布式记忆的另一种设计。1989b。

P. Kanerva。稀疏分布式记忆及相关模型。1993。

Pentti Kanerva。稀疏分布式记忆。MIT 出版社，1988。

James D. Keeler。Kanerva 的 SDM 与 Hopfield 型神经网络的比较。《认知科学》，12(3):299 – 329，1988。ISSN 0364-0213。doi: https://doi.org/ 10.1016/0364-0213(88)90026-2。URL http://www.sciencedirect.com/science/ article/pii/0364021388900262。

Diederik P. Kingma 和 Jimmy Ba。Adam：一种随机优化方法。*CoRR*, abs/1412.6980, 2015年。

James Kirkpatrick、Razvan Pascanu、Neil C. Rabinowitz、Joel Veness、Guillaume Desjardins、An- drei A. Rusu、Kieran Milan、John Quan、Tiago Ramalho、Agnieszka Grabska-Barwinska、Demis Hassabis、Claudia Clopath、Dharshan Kumaran 和 Raia Hadsell。克服神经网络中的灾难性遗忘。美国国家科学院院刊, 114:3521 – 3526, 2017年。D. Krotov 和 J. Hopfield。神经生物学与机器学习中的大型联想记忆问题。*ArXiv*, abs/2008.06996, 2020年。Dmitry Krotov 和 John J Hopfield。用于模式识别的稠密联想记忆。神经信息处理系统进展, 29, 2016。Dmitry Krotov 和 John J. Hopfield。稠密联想记忆对对抗性输入具有鲁棒性。神经计算, 30:3151–3167, 2018年。Dmitry Krotov 和 John J. Hopfield。通过竞争性隐单元进行无监督学习。美国国家科学院院刊, 116:7723 – 7731, 2019。Matthias De Lange、Rahaf Aljundi、Marc Masana、Sarah Parisot、Xu Jia、Ale Leonardis、Gregory G. Slabaugh 和 Tinne Tuytelaars。持续学习综述：在分类任务中克服遗忘。*IEEE*模式分析与机器智能汇刊, PP, 2021。Frederic Lanore、N. Alex Cayco-Gajic、Harsha Gurnani、Diccon Coyle 和 Robin Angus Silver。小脑颗粒细胞轴突支持高维表征。《自然-神经科学》, 24: 1142 – 1150, 2021 年。Hung Le 和 Svetha Venkatesh。Neurocoder：利用存储的神经程序实现通用计算。载于国际机器学习大会, 2022。Hung Le、T. Tran 和 Svetha Venkatesh。神经存储程序记忆。*ArXiv*, abs/1906.08862, 2019。Jinzhi Li、Brenna Mahoney、Miles Solomon Jacob 和 Sophie Jeanne C´ecile Caron。视觉输入进入黑腹果蝇的蘑菇体。*bioRxiv*, 2020年。Yuchen Liang、Chaitanya Ryali、Benjamin Hoover、Leopold Grinberg、Saket Navlakha、Mo- hammed J Zaki 和 Dmitry Krotov。果蝇能学会词嵌入吗？载于国际学习表征会议, 2020年。Timothy P. Lillicrap、A. Santoro、Luke Marris、C. Akerman 和 G. Hinton。反向传播与大脑。《自然综述：神经科学》, 21:335–346, 2020年。Andrew C. Lin、Alexei M. Bygrave、Alix de Calignon、Tzumin Lee 和 Gero Miesenb¨ock。蘑菇体中的稀疏、去相关气味编码增强了习得性气味辨别能力。《自然-神经科学》, 17:559 – 568, 2014年。Ashok Litwin-Kumar、K. D. Harris、R. Axel、H. Sompolinsky 和 L. Abbott。突触连接度的最优水平。神经元, 93:1153–1164.e7, 2017年。Alireza Makhzani 和 Brendan J. Frey。k稀疏自编码器。*CoRR*, abs/1312.5663, 2014年。Arun Mallya 和 Svetlana Lazebnik。Packnet：通过迭代剪枝将多个任务添加到单个网络中。*2018 IEEE/CVF*计算机视觉与模式识别会议, pp. 7765– 7773, 2018年。D. Marr。小脑皮层理论。《生理学杂志》, 202, 1969年。Leland McInnes 和 John Healy。UMAP：用于降维的均匀流形近似与投影。*ArXiv*, abs/1802.03426, 2018年。Beren Millidge、Tommaso Salvatori、Yuhang Song、Thomas Lukasiewicz 和 Rafał Bogacz。通用 Hopfield 网络：单次联想记忆模型的通用框架。*ArXiv*, abs/2202.04557, 2022年。

M. Modi、Yichun Shuai 和 G. Turner。《果蝇蘑菇体：从架构到学习回路中的算法》。《神经科学年度评论》，2020年。

Elhage Nelson、Hume Tristan、Olsson Catherine、Nanda Neel、Henighan Tom、Johnston Scott、ElShowk Sheer、Joseph Nicholas、DasSarma Nova、Mann Ben、Hernandez Danny、Askell Amanda、Ndousse Kamal、Jones、、Drain Dawn、Chen Anna、Bai Yuntao、Ganguli Deep、Lovitt Liane、Hatfield-Dodds Zac、Kernion Jackson、Conerly Tom、Kravec Shauna、Fort Stanislav、Kada-vath Saurav、Jacobson Josh、Tran-Johnson Eli、Kaplan Jared、Clark Jack、Brown Tom、McCandlish Sam、Amodei Dario 和 Olah Christopher。Softmax 线性单元。*Transformer Circuits Thread*，2022年。https://transformer-circuits.pub/2022/solu/index.html。

Dylan M. Paiton、Charles G Frye、Sheng Y. Lundquist、Joel Bowen、Ryan Zarcone 和 Bruno A. Olshausen。稀疏编码网络的选择性与鲁棒性。《视觉杂志》，20，2020年。

DabalPedamonti。深度神经网络在 MNIST 分类任务中非线性激活函数的比较。*ArXiv*，abs/1804.02763，2018年。

Thomas John Pisano、Zahra M. Dhanerawala、Mikhail Kislin、Dariya Bakshinskaya、Esteban A. En- gel、Ethan J. Hansen、Austin T. Hoag、Junuk Lee、Nina L. de Oude、Kannan Umadevi Venkataraju、Jessica L. Verpeut、Freek E. Hoebeek、Ben D. Richardson、Henk-Jan Boele 和 Samuel S.-H. Wang。小脑通路向感觉、运动及联想前脑的同源组织。*bioRxiv*，2020年。

Kate Powell、Alexandre Mathy、Ian Duguid 和 Michael H¨ausser。单个小脑颗粒细胞中的运动突触表征。*eLife*，4，2015年。

D Purves、GJ Augustine 和 D Fitzpatrick。《style id='1'>神经元通常会释放多种神经递质。Sinauer Associates，第二版，2001年。

Rodrigo QuianQuiroga、Leila Reddy、G. Kreiman、Christof Koch 和 Itzhak Fried。人脑中由单个神经元形成的不变视觉表征。《自然》，435:1102–1107，2005年。

Vinay V. Ramasesh、Aitor Lewkowycz 和 Ethan Dyer。模型和预训练规模对神经网络灾难性遗忘的影响。2022年。

Hubert Ramsauer、Bernhard Sch¨afl、Johannes Lehner、Philipp Seidl、Michael Widrich、Lukas Gru- ber、Markus Holzleitner、Milena Pavlovi´c、Geir Kjetil Sandve、Victor Greiff、David Kreil、Michael Kopp、G¨unter Klambauer、Johannes Brandstetter 和 Sepp Hochreiter。Hopfield 网络就是你所需要的一切，2020年。

Stephen Roller、Sainbayar Sukhbaatar、Arthur D. Szlam 和 Jason Weston。用于大型稀疏模型的哈希层。发表于*NeurIPS*，2021年。

David E. Rumelhart 和 David Zipser。通过竞争学习发现特征。1985年。

Olga Russakovsky、Jia Deng、Hao Su、Jonathan Krause、Sanjeev Satheesh、Sean Ma、Zhiheng Huang、Andrej Karpathy、Aditya Khosla、Michael S. Bernstein、Alexander C. Berg 和 Li Fei-Fei。ImageNet 大规模视觉识别挑战。国际计算机视觉期刊，115：211–252，2015年。

Chaitanya Ryali、John Hopfield、Leopold Grinberg 和 Dmitry Krotov。用于无监督相似性搜索的仿生哈希。发表于国际机器学习大会，第 8295–8306 页。PMLR，2020年。

Jonathan Schwarz、Siddhant M. Jayakumar、Razvan Pascanu、Peter E. Latham 和 Yee Whye Teh。功率传播：一种促使权重稀疏化的重参数化方法。发表于*NeurIPS*，2021年。

Anirvan M. Sengupta、M. Tepper、C. Pehlevan、A. Genkin 和 D. Chklovskii。流形铺砌式局部感受野在相似性保持神经网络中是最优的。*bioRxiv*，2018年。

BiswaSengupta、Martin BStemmler、Simon B.Laughlin 和Jeremy E.Niven。脊椎动物和无脊椎动物的不同神经元类型，其动作电位能量效率各不相同。《公共科学图书馆计算生物学》，6，2010年。

Eren Sezener、Agnieszka Grabska-Barwinska、Dimitar Kostadinov、Maxime Beau、Sanjukta Krishnagopal、David Budden、Marcus Hutter、Joel Veness、Matthew M. Botvinick、Claudia Clopath、Michael H¨ausser 和 Peter E. Latham。生物神经回路的一种快速高效学习规则。*bioRxiv*，2021年。

Noam M. Shazeer、Azalia Mirhoseini、Krzysztof Maziarz、Andy Davis、Quoc V. Le、Geoffrey E. Hinton 和 Jeff Dean。超大型神经网络：稀疏门控混合专家层。*ArXiv*，abs/1701.06538，2017年。

Yang Shen、Sanjoy Dasgupta 和 Saket Navlakha。果蝇带来的持续学习算法见解。*ArXiv*，abs/2107.07617，2021年。

JamesSmith、JunjiaoTian、Yen-Chang Hsu 和 Zsolt Kira。深入探究无回放持续学习。*ArXiv*，abs/2203.17269，2022年。

Rupesh Kumar Srivastava、Jonathan Masci、Sohrob Kazerounian、Faustino J. Gomez 和 J¨urgen Schmidhuber。竞争以实现计算。在*NIPS*，2013。

Peter Sterling 和 Simon Laughlin。《神经设计原理》。2015年。

Gilbert Strang。《线性代数导论》。1993。

Sainbayar Sukhbaatar、E. Grave、Guillaume Lample、H. J´egou 和 Armand Joulin。用持久记忆增强自注意力。*ArXiv*，abs/1907.01470，2019。

Giulio Tononi 和 Chiara Cirelli。睡眠与可塑性的代价：从突触和细胞稳态到记忆巩固与整合。神经元，81：12–34，2014年。

Asher Trockman 和 J. Zico Kolter。补丁就是你所需要的一切吗？*ArXiv*，abs/2201.09792，202 2.

Danil Tyulmankov、Ching Fang、Annapurna Vadaparty 和 Guangyu Robert Yang。生物键值记忆网络。神经信息处理系统进展，34，2021年。

A¨aron van den Oord、Oriol Vinyals 和 K. Kavukcuoglu。神经离散表征学习。载于*NIPS*，2017年。

Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N. Gomez、Lukasz Kaiser 和 Illia Polosukhin。Attention Is All You Need，2017年。

Ziheng Wang。Sparsert：为深度学习推理加速 GPU 上的非结构化稀疏性。*ACM*并行架构与编译技术国际会议论文集，2020年。

Marjorie Xie、Samuel P. Muscinelli、Kameron Decker Harris 和 Ashok Litwin-Kumar。适用于小脑学习的任务依赖型最优表征。*bioRxiv*，2022年。

Ju Xu 和 Zhanxing Zhu。强化持续学习。载于*NeurIPS*，2018年。

Daniel Yamins、Ha Hong、Charles F. Cadieu、Ethan A. Solomon、Darren Seibert 和 James J. DiCarlo。性能优化的层级模型可预测高级视觉皮层中的神经反应。美国国家科学院院刊，111：8619 – 8624，2014年。

Chris I. DeZeeuw. 小脑上行和下行微区的双向学习。《自然综述：神经科学》，22:92–110，2020年。

Friedemann Zenke、Ben Poole 和 SuryaGanguli。通过突触智能实现持续学习。机器学习研究论文集，70:3987–3995，2017年。

Mengmi Zhang、Rohil Badkundri、Morgan B. Talbot 和 Gabriel Kreiman。使用增强记忆进行假设驱动的流式学习。*ArXiv*，abs/2104.02206，2021年。

附录

目录

A SDM **16**

A.1 SDM 训练算法.16 A.2 SDM的生物学合理性.17 A.3 使用 Top-K 激活函数学习 SDM 神经元地址 . . . . . 20 A.4 SDM 最初为何需要固定神经元地址.20 A.5 对 SDM 的其他修改.21 A.6 SDM写入操作与多层感知机反向传播的关系 . . . . . 23 A.7 频率编码激活使 SDM 保持对 Transformer 注意力的近似 23

B GABASwitch 26

B.1 关于死神经元问题的不符合生物学原理的解决方案 26 B.2 GABA开关生物学.26 B.3 GABA开关实践考量.28

C Top-K 31

C.1 抑制性中间神经元近似实现 Top-K . 31 C.2 优化的 Top-K.33

D 陈旧动量 **35**

E 训练机制消融实验 40

E.1 直接在图像像素上训练.40 E.2 无预训练.42

F 额外数据集 **43**

F.1 CIFAR10 补充图表.43 F.2 CIFAR100.43 F.3 拆分 MNIST.46 F.4 拆分 FashionMNIST.47

G 基线实现 48

G.1 弹性权重巩固和突触智能的 Beta 系数 . . . 48 G.2 FlyModel 参数.49

H 持续学习能力差异研究 49

A SDM

A.1 SDM 训练算法

此处我们完整呈现了 SDMLP 使用的训练循环。

算法 1SDMLP 训练算法

输入： 模型权重：(X_a，X_v)，任务：(T_1，.....，T_n)，Top-K 参数：$k_{\max}, k_{\text{target}}$，训练步数：$s$
输出： 训练后的模型权重：(X_a，X_v)
$X_v \leftarrow$LayerInit(X_v) // 重置神经元输出权重
对 *i in (1,. . . ,n)* **执行**
  $\mathbf{X}$，$\mathbf{Y} \leftarrow T_i$ // 获取当前任务的数据和类别标签
  对 *j in (1,. . . ,s)* **执行**
    $\mathbf{x}$，$\mathbf{y} \sim \mathbf{X}$，$\mathbf{Y}$ // 从任务中采样一个数据点
    $\mathbf{x} \leftarrow \mathbf{x}/||\mathbf{x}||$ // 对数据进行 L2 归一化
    $\mathbf{a} \leftarrow X_a^T \mathbf{x}$ // 获取神经元激活值
    $\mathbf{a} \leftarrow$Top-K(a) // 应用 Top-K 公式 4
    $\hat{\mathbf{y}} \leftarrow X_v^T \mathbf{a}$ // 获取模型预测
    $\nabla_\theta X_a, \nabla_\theta X_v \leftarrow$ 计算梯度($\hat{\mathbf{y}}, \mathbf{y}$) // 计算损失和梯度
    $X_a, X_v \leftarrow$ 梯度步进($\nabla_\theta X_a, \nabla_\theta X_v$) // 更新模型权重
    $X_a, X_v \leftarrow [X_a]_+, [X_v]_+$ // 将所有权重限制为正值
    $X_a \leftarrow X_a/||X_a||$ // 对神经元地址进行 L2 归一化
  end
end
返回 X_a, X_v

A.2 SDM 的生物学合理性

此处我们总结了 SDM 的生物学基础。[9] 关于我们对 SDM 所做的新修改（使其能够学习数据流形并作为深度学习模型实现）的生物学合理性，请参见附录 A.5。关于 GABA 开关的生物学合理性，请参见附录 B.2。

图 5 将 SDM 的符号表示和操作映射到小脑回路上。苔藓纤维通过其放电活动表示输入查询 $\boldsymbol{\xi}$ 以及模式地址 $\mathbf{p}_a$。每个颗粒细胞代表一个 SDM 神经元 $\mathbf{x}$，其树突构成神经元地址向量 $\mathbf{x}_a$，而它与浦肯野细胞的突触后连接则构成存储向量 $\mathbf{x}_v$。每个颗粒细胞的轴突（称为平行纤维）垂直分支，与数千个浦肯野细胞以及许多颗粒细胞形成连接（Hoxha等，2016）。每个连接的强度表示一个计数器，用于记录神经元存储向量中特定元素的值（$\mathbf{x}_v$）$_i$，其中 i 为该向量元素的索引。

因此，在 SDM 读取操作期间，这个浦肯野细胞可以对所有已激活神经元存储向量的索引 i 求和，并利用其放电率表示$\mathbf{y}_i$的值。在 SDM 写入操作期间，攀缘纤维用于根据该索引位置上的模式值（$\mathbf{p}_v$）$_i$更新每个活跃神经元的浦肯野细胞计数器。SDM 在生物回路中的实现要求最高的一点，是浦肯野细胞、攀缘纤维和平行纤维之间存在的三方接口及特异性，而小脑恰好显著满足了这一要求。这使颗粒细胞能够拥有可被精确写入和读取的存储向量。我们还提供了图 6，以从另一个角度展示这种生物学映射。

虽然我们改进版的 SDM 为小脑回路中的五种不同细胞类型分配了功能，但仍有更多回路结构有待映射，例如篮状细胞和小脑深部核团（Sezener 等，2021）。此外，还需要更多神经科学证据来确认小脑的运作方式符合 SDM 的预测，例如颗粒细胞以稀疏方式放电（Kanerva，1988；Bricken 和 Pehlevan，2021；Giovannucci 等，2017）。在我们的模型范围内，生物学合理性最薄弱之处在于 Top-K 对高尔基抑制性中间神经元（附录 C.1 ）的近似程度，以及 L^2 归一化的实现方式（附录 A.5）。

[9]关于小脑如何实现 SDM 的更详尽讨论，可参见 SDM 专著第9章（Kanerva，1988）和（Kanerva，1993）。

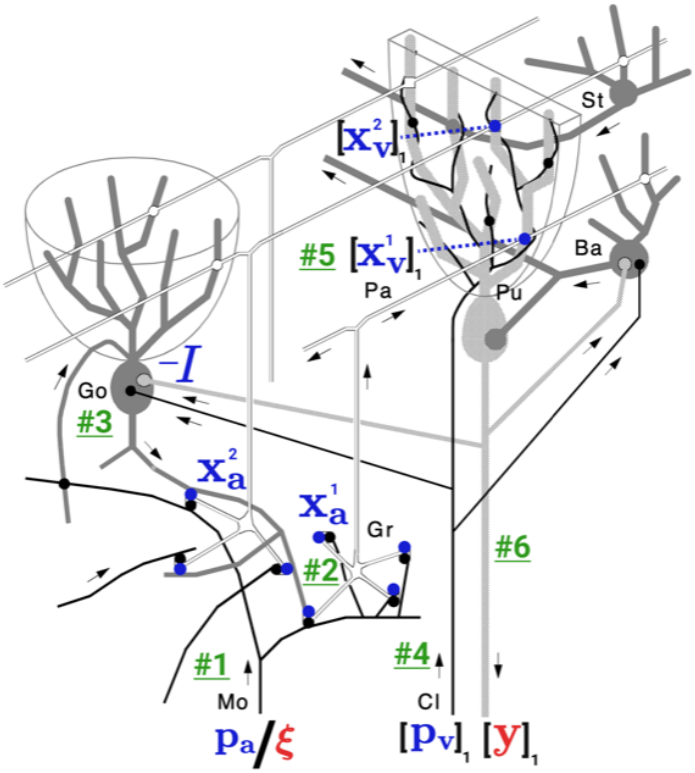


图 5：**SDM 映射到小脑**。 操作如下：#1. 模式$\mathbf{p}_a$（蓝色）或查询 ξ（红色）通过苔藓纤维（Mo）的放电进入。#2. 苔藓纤维激活颗粒细胞（Gr），Gr 的树突表示$\mathbf{x}_a$。这里展示了两个不同的颗粒细胞及其地址$\mathbf{x}_a^1$ 和$\mathbf{x}_a^2$。这些颗粒细胞将其轴突（称为平行纤维（Pa））向上延伸，并横跨浦肯野细胞（Pu）；它们的突触连接编码各自的存储向量。与浦肯野细胞的每个接触都编码存储向量的一个元素。这里展示了其中一个元素，即第一个浦肯野细胞对应的元素，带有下标 1, $[\mathbf{x}_v^1]_1$ 和 $[\mathbf{x}_v^2]_1$。稍后会再谈到这些。#3. 高尔基细胞（Go）这一抑制性中间神经元接收多种神经元类型的输入，包括颗粒细胞；它以 $-I$ 的值抑制这些颗粒细胞，从而实现近似的 Top-K。#4. 当且仅当我们要写入模式$\mathbf{p}_v$时，信号才会经由攀缘纤维（Cl）传入。攀缘纤维缠绕单个浦肯野细胞，产生强动作电位，从而诱发长时程增强/抑制（LTP/LTD）。这里展示了写入浦肯野细胞的攀缘纤维，该细胞表示存储向量的第一个元素，带有 $[\mathbf{p}_v]_1$。#5. 正在放电的颗粒细胞会按其突触连接（代表其存储向量）的强度激发浦肯野细胞。在写入时，这会更新这些突触。#6. 如果我们从系统读取，浦肯野细胞会整合所有激活神经元的存储向量所传来的信号，并判定是否放电，从而实现一个输出 $[\mathbf{y}]_1$ 的非线性激活函数（此处为向量的第一个元素）。此前未提及的神经元类型，从右上方开始顺时针依次为：St=星形细胞（抑制性中间神经元）和 Ba=篮状细胞（抑制性中间神经元）。原图最初未叠加 SDM 标注，刊于 (Kanerva，1993) 的图 3.11；经 Pentti Kanerva 许可转载，所有权利均归其所有。

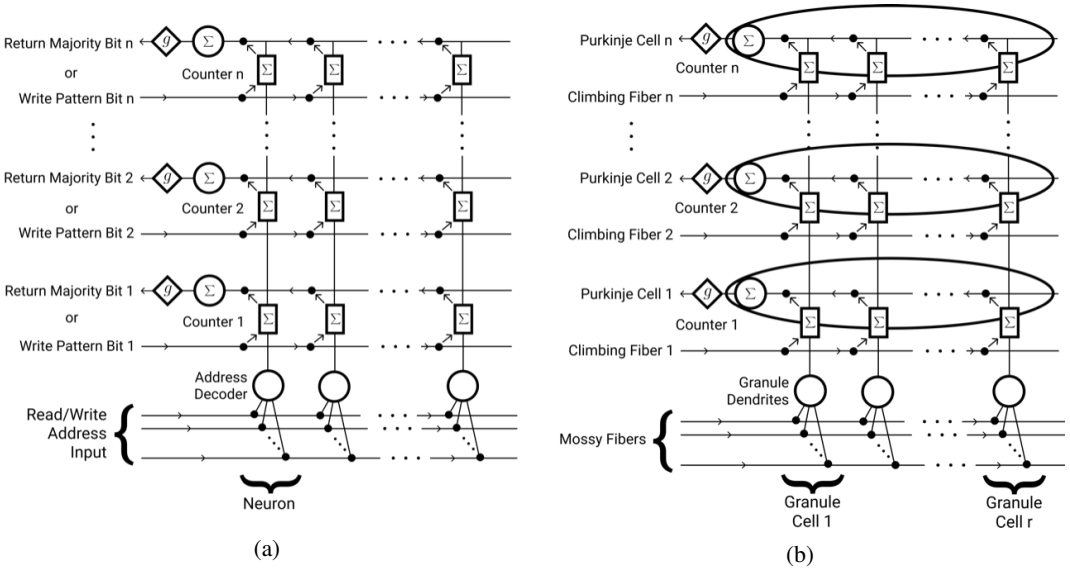


图 6：**从 SDM 到小脑回路映射的另一种视角。（a）**实现 SDM 所需的计算回路。**（b）**SDM 的生物学实现，其中（a）中的标签已替换为相应的生物学组成部分。浦肯野细胞以椭圆表示，以呈现其与颗粒细胞的突触接触。架构特征的说明请参见正文。

A.3 使用 Top-K 激活函数学习 SDM 神经元地址

SDM 建立在这样的假设之上：其神经元地址 X_a 是随机初始化并固定不变的（Kanerva, 1988）。这是出于生物学合理性以及分析可处理性的考虑，以便确定模型的最大记忆容量和收敛动力学（见附录 A.4）。然而，这些随机且固定的神经元地址通常与向量空间中某个低维流形上的现实世界模式的余弦相似度 c 不足。因此，这些神经元往往永远不会被写入或读取。

Keeler（1988）阐述了如何在保留 SDM 理论结果的同时，使神经元地址能够学习数据流形。方法是将固定的余弦相似度激活阈值 c （它会激活数量可变的神经元）替换为可变阈值，从而激活固定数量的 k 个神经元。直观而言，在神经元地址和模式地址均为随机的情况下，固定的 c 预期会使某个模式或查询激活一定数量的 k 个神经元，从而保持神经元利用率恒定。然而，若地址并非随机，仍采用相同的固定 c，则被激活和利用的神经元数量会有所变化。利用过度的神经元会在其存储向量叠加中存储过多模式值，从而损害模式保真度。相反，我们可以通过调整 c，使始终只有 k 个神经元被激活，从而在随机地址设定下实现同样恒定的神经元利用率。

如何以符合生物学合理性的方式动态调整 c，使活跃的神经元数量保持恒定为 k？ 可以通过抑制性中间神经元形成负反馈回路：活跃的神经元越多，中间神经元就越活跃，抑制作用也越强，从而在收敛时仅保留k 处于活跃状态。[10] 在大脑皮层处理的各层中，诱导稀疏性的中间神经元广泛存在；与 SDM 尤其相关的是小脑高尔基中间神经元（Marr，1969年；Keeler，1988；Paiton 等人，2020年）。

A.4 SDM 最初为何需要固定神经元地址

为了符合生物学合理性，SDM 的设计遵循戴尔定律，即突触连接可以是兴奋性的，也可以是抑制性的，但不能在两者之间转换（Dale，1935年）。使用最初的二进制向量表述，神经元可以通过以下步骤判断自身是否足够接近输入模式或查询（即是否处于某个汉明距离阈值范围内），从而被激活：

> 1. 将神经元的二进制地址转换为双极权重 $\{+1, -1\}$，分别对应兴奋性或抑制性。2. 激活传入查询的地址中值为 1 的位置所对应的权重，并将这些权重求和。这表示查询向量和神经元地址向量中有多少个 1 相匹配——如果地址相同，则会激活一个兴奋性权重，得到 +1；如果地址不同，则会激活一个负权重，得到 -1。3. 根据神经元地址中 1 的总数，重新缩放二进制汉明距离阈值，以确定神经元是否被激活。有关此概述之外的详情，请参阅（Kanerva，1988；1993）。

形式上，二进制神经元地址可以转换为权重 $w_i = \pm 1$，如下所示：

$$w_i = \big(2 * [\mathbf{x}_a^\tau]_i\big) - 1, \forall i \in \{1, ..., n\}. \tag{5}$$

其中，τ 表示特定的神经元地址，查询向量为 $\boldsymbol{\xi}$，用于判断神经元是否应被激活的函数为 $a(\cdot, \cdot)$，该函数返回一个二进制动作电位。我们可以写作：

$$a(\boldsymbol{\xi}, \mathbf{x}_a^\tau) = I\bigg[\Big(\sum_{i=1}^{n} w_i \xi_i\Big) > d\bigg] = I\bigg[\Big(\sum_{i=1}^{n} \Big((2 * [\mathbf{x}_a^\tau]_i) - 1\Big)\xi_i\Big) > d\bigg],$$

其中，d 是神经元触发的动作电位阈值，$I[\cdot]$ 是指示函数。地址解码的取值区间为地址中 1 的个数，$[-\sum_i^n [\mathbf{x}_a^\tau]_i,\ \sum_i^n [\mathbf{x}_a^\tau]_i]$。通过调整我们的 d 值，以考虑可能的匹配数量，我们可以为每个神经元实现汉明距离阈值。

[10] 附录 C.1 讨论了 Top-K 对抑制性中间神经元的近似效果。

人们曾考虑改变神经元权重的大小，但根据戴尔定律，禁止改变其符号。Kanerva（1988）指出，改变神经元权重的大小会产生一个加权和，在计算神经元是否被激活时，地址中某些位的匹配比其他位更重要。然而，关键特征是权重的符号，它决定了匹配还是不匹配。戴尔定律禁止的正是这种符号变化，因此通过使用无法表示权重大小细微差异的二进制向量，神经元地址的符号和具体取值都被固定下来（Kanerva，1988；1993）。

然而，原始 SDM 方法实际上仍然违反了戴尔定律，因为它将二进制神经元地址映射为双极形式，而这会影响突触前输入神经元。考虑输入向量（模式或查询均可）的第 3 个元素处于激活状态（值为 1）的情况。代表该值的活跃神经元，与 SDM 神经元地址之间不会同时存在兴奋性和抑制性传出连接。在小脑映射中，苔藓纤维代表模式/查询输入，并释放谷氨酸。这意味着它们对颗粒细胞始终具有兴奋性，因此神经元地址中的 0 应对应于没有权重（没有树突连接），而不是负权重（抑制性连接）。

原始 SDM 公式的另一个问题是，神经元地址是稠密向量；而在小脑中，它们所映射到的颗粒细胞并非如此，每个颗粒细胞只有 ~ 4 条树突（Litwin-Kumar 等人，2017年）（关于一种使 SDM 权重稀疏的解决方案，参见 Jaeckel，1989a）。

综合来看，只需让二进制神经元地址保持稀疏并维持二进制形式，而不是转变为双极形式，就能解决这些问题。这样一来，我们便可以允许神经元地址采用正实数值，同时允许权重强度发生变化，并遵循 Dale 原则。在本文中，我们采用正实数值，但允许权重为稠密形式，将高程度的权重稀疏性留待未来探索（Jaeckel，1989a;b）。

A.5 对 SDM 的其他修改

The 为使 SDM 能够在深度学习框架中实现而进行的五项修改，以及 t
diff使其区别于普通多层感知机的修改，将在此处详细说明。这些修改包括：

1. 使用连续而非二进制神经元激活 2. L^2 对输入和
权重进行归一化 3. 不使用偏置项 4. 仅使用正权重
5. 反向传播

1. 连续激活 - SDM 最初使用赫维赛德阶跃函数，将神经元建模为具有二进制动作电位。然而，这种函数不可微，而我们希望使用反向传播来训练模型。解决方法是考察神经元在一段时间间隔内的动作电位；神经科学文献将此称为频率编码（Dayan 与 Abbott，2001年）。我们认为这与 SDM 相容：地址与输入模式或查询越接近的神经元，会受到越强的刺激，并因更多树突连接受到兴奋性神经递质的刺激而发出更多动作电位。

我们可以用一个实数正数 $a \in [0, t]$来表示神经元激活的期望放电率，其中 t 是最大放电率。这看起来可能像修正双曲正切函数：由于存在不应期，进一步增加刺激所带来的收益会逐渐减少（Glorot 等，2011）。然而，由于我们的神经元已经受到 L^2 归一化的约束，其余弦相似度范围为 $[-1, 1]$，因此我们只使用 ReLU 激活函数。
在原始 SDM 公式 1 和 2 中实现加权读写操作时，我们会将二值化函数 $b(\cdot)$ 替换为一个与输入和神经元地址之间的距离成正比的权重系数。我们在附录 A.7 中表明，这一修改对 SDM 为不同模式分配权重的方式影响极小。这意味着它对记忆容量的影响也应当极小，并且其关系仍近似呈指数形式，从而保持与 Transformer 注意力的关联（Bricken 和 Pehlevan，2021）。

2. L^2 **归一化 -**SDM 需要一种有效的距离度量，以判断神经元与模式/查询之间是否足够接近，从而产生激活。我们遵循 Bricken 和 Pehlevan（2021）的方法，将原始的汉明距离替换为余弦相似度，使 SDM 连续化，从而具备可微性。

在 SDM 与小脑的映射关系中（见附录 A.2），苔藓纤维表示输入模式/查询，颗粒细胞树突表示神经元地址。这些特定细胞类型是否以及如何实现 L^2 归一化，仍有待通过实验确定。不过，对比度编码和异突触性都是普遍存在的机制，分别可以近似实现苔藓纤维激活和颗粒细胞树突的 L^2 归一化（Sterling & Laughlin, 2015；Rumelhart & Zipser, 1985；Tononi & Cirelli，2014）。从深度学习的角度来看，层归一化和批归一化都被广泛使用，也可以被视为 L^2 归一化的近似方法（Ioffe & Szegedy，2015；Ba 等，2016）。

3. 无偏置项 -为了使神经元激活表示余弦相似度，我们必须移除偏置项。我们也移除输出层的偏置项，使输出仅表示已激活神经元存储向量的总和。对比 SDM 方程 2 和 MLP 方程 3，可以清楚地看到这两个偏置项均已移除。

偏置项在生物学上可视为代表神经元激活阈值（若为负值）或基线持续放电率（若为正值）。在 SDM 的小脑映射中，代表神经元的颗粒细胞没有紧张性基线放电率，这意味着不应允许正偏置项（Powell 等，2015；Giovannucci 等，2017）。代表输出层的浦肯野细胞放电机制复杂得多，因此无论保留还是移除偏置项，都难以从生物学角度论证（Zeeuw，2020）；不过，我们遵循 SDM 方程，也将其移除。

从颗粒细胞中移除正偏置项符合神经生物学原理，而移除对应激活阈值的负偏置项则较难得到充分论证。不过，就学习动态以及 Top-K 是否仍可近似而言，(Gozel & Gerstner, 2021) 的常微分方程（概述见 C.1）在使用激活阈值时，仍能维持约 k 个神经元放电。这是因为放电的神经元越少，抑制性中间神经元的激活程度就越低，从而使更多神经元保持活跃。因此，与 SDM 方程 2 一致，我们作出简化假设，不允许负偏置项，从而彻底移除偏置项。

SDM 不含偏置项与采用正权重相结合，还有一个额外好处：默认情况下，所有神经元都会具有正激活。这使得可以使用 k 退火，而不必像原本那样使用 GABA 开关，在学习数据流形时向每个神经元注入正电流（附录 B.3 中也有讨论）。所有神经元都处于激活状态，也能保证始终至少有 k 个激活神经元可用于 Top-K。

值得注意的是，最先进模型中正重新出现了移除偏置项的做法，例如拥有 5400 亿参数的 PaLM Transformer 语言模型（Chowdhery 等，2022）；该研究指出，移除偏置项能够带来更稳定的训练。

4. 正权重 -苔藓纤维与颗粒细胞之间的连接具有兴奋性，因此权重应为正。允许权重既有正值也有负值，会违反戴尔定律（Dale，1935）。如上一节所述，仅使用正权重且不设偏置项，两者结合还能确保所有神经元激活均为正，从而可以使用更简单的 k 退火算法。

巧合的是，如表 2 的消融实验所示，仅使用正权重也能带来更好的持续学习性能。最后一个好处是能够生成稀疏权重，但这一优势也有其局限性。在使用我们的 ConvMixer 嵌入训练的模型中，我们发现权重稀疏性仅为 $\sim$20%。与此同时，在尝试将 ConvMixer 与 SDM 联合训练时，过多权重被设为 0，导致训练失败。

5. 反向传播 -用于训练多层感知机的反向传播梯度下降很可能不符合生物学实际（Lillicrap 等，2020；Goodfellow 等，2015）。然而，我们认为，存在一些与抑制性中间神经元和流形铺砌相关的局部赫布学习规则，是可行的替代方案（Sengupta 等，2018；Krotov 和 Hopfield，2019；Gozel & Gerstner，

2021）。此外，小脑通过攀缘纤维[11], 接收监督学习信号；已知攀缘纤维能够编码异联想关系，例如对音调产生眨眼反应。这种神经回路在（Sezener 等，2021）中进行了探讨，并为小脑训练提供了另一种反向传播替代方案。

A.6 SDM写入操作与多层感知机反向传播的关系

原始 SDM 写入操作直接使用模式值$\mathbf{x}_v = \mathbf{x}_v + \alpha\mathbf{p}_v$ 更新神经元值向量，其中 α 根据每个神经元被激活的程度对模式加权。与此同时，反向传播写入操作会使用模型输出与真实类别独热编码之间的误差（采用交叉熵损失）来更新$\mathbf{x}_v$ 。模型训练期间，所有输入都被视为写入操作。训练结束后，在推理期间，所有输入都被视为执行 SDM 读取操作的查询。$\mathbf{p}_a$ 对应一张 CIFAR 图像，而$\mathbf{p}_v$ 是其编码的独热标签。
这种差异与 SDM 相容，原因有以下几点：
首先，对于$\mathbf{x}_v$ ，能够产生零误差的最优解是原始 SDM 写入操作所使用的独热编码。

其次，只有在神经元地址固定且$\mathbf{x}_v$是唯一被学习的内容时，原始 SDM 写入操作才适用。否则，随着神经元更新其地址$\mathbf{x}_a$，它们被激活的模式以及它们应存储在$\mathbf{x}_v$中的$\mathbf{p}_v$ 都会发生变化。采用反向传播方法，根据当前激活$\mathbf{x}_v$ 的模式持续更新它，是一种可行的解决方案。

最后，从生物学角度来看，反向传播所用的误差信号，更接近于 SDM 所映射到的小脑回路更新 $\mathbf{x}_v$ 的方式（小脑上行和下行微区中的双向学习，De Zeeuw，2021）。虽然有人认为，深度网络多层之间的反向传播在生物学上并不合理，但由于输出层的这项更新与误差计算直接相关，因此这种做法是可行的。
总之，SDM 会显式地将 $\mathbf{p}_v$ 写入，而采用反向传播的多层感知机会计算 $\mathbf{p}_v$ 与网络输出之间的差值；不过，这种方法与相同的解决方案兼容，在同时学习神经元地址时效果更好，而且在生物学上可能更合理。

A.7 频率编码激活维持 SDM 对 Transformer 注意力的近似

Bricken 和 Pehlevan（2021）指出，Transformer 的注意力操作与 SDM 的读取操作（Vaswani 等，2017）非常接近。这是因为 SDM 中分配给每个模式的权重近似呈指数形式，因此产生了 Softmax 规则。

不过，这一结果是在 SDM 读取和写入时采用神经元二值激活、而非根据距离对每个模式加权的情况下推导得出的。这里我们表明，采用线性或指数模式加权时，SDM 仍然是注意力的良好近似。因此，先前工作的结果在 SDM 以多层感知机形式书写的情形下仍然成立，这揭示了多层感知机与注意力之间一种有趣的关系（见第 6 节）。
在说明如何利用圆交集实现加权系数及这种加权的效果之前，先概述 (Bricken 和 Pehlevan，2021) 推导的圆交集大小算法。

与原始 SDM 表述一样，我们使用 n 维二进制向量，汉明激活阈值为 d ，查询 $\boldsymbol{\xi}$ 与模式 $\mathbf{p}_a$ 的地址之间的距离为$d_v := d(\boldsymbol{\xi},\mathbf{p}_a)$。我们可以将 $\boldsymbol{\xi}$ 和$\mathbf{p}_a$ 的元素分成两个互不相交的集合： $n - d_v$ 个元素

[11]攀缘纤维信号究竟编码的是误差还是待学习的目标，目前仍是一个开放问题；不过，我们在这里使用“监督式”一词，将这两种情况都涵盖在内，并以此区别于没有任何“教师”信号的无监督学习。

其中这些元素的值相同，以及 d_v 个值不同的元素[12]：

$$\begin{aligned}\boldsymbol{\xi} &= [\overbrace{1,\ldots,1}^{n-d_v} \mid \overbrace{0,\ldots,0}^{d_v} \mid \overbrace{1,\ldots,1}^{d_v} \mid \overbrace{0,\ldots,0}^{n-d_v}] \\ \mathbf{p}_a &= [\underbrace{1,\ldots,1}_{a+z} \mid \underbrace{0,\ldots,0}_{b+c} \mid \underbrace{0,\ldots,0}_{b+c} \mid \underbrace{1,\ldots,1}_{a+z}]\end{aligned} \tag{6}$$

现在设想第三个向量，它代表一个潜在的神经元地址。该神经元有四种可能的分组，其向量元素可以归入其中：

- a- 同时赞同 两者 $\boldsymbol{\xi}$ 和 $\mathbf{p}_a$
- z - 同时不赞同两者 $\boldsymbol{\xi}$ 和 $\mathbf{p}_a$
- b - 赞同 $\boldsymbol{\xi}$ ，不赞同 $\mathbf{p}_a$
- c - 赞同$\mathbf{p}_a$ ，不赞同 $\boldsymbol{\xi}$

我们希望约束 a、 z、 b 和 c 的取值，使神经元地址位于 $\boldsymbol{\xi}$ 和$\mathbf{p}_a$之间的圆交集内。由此得到以下约束：

$$\begin{aligned} a+b+c+z &= n \\ a+b &\geq n-d \\ a+c &\geq n-d \\ a+z &= n-d_v \\ b+c &= d_v \end{aligned}$$

采用（Vaswani 等，2017）的记法，读圆和写圆交集中的神经元总数可表示为：

$$\sum_{a=n-d-\lfloor \frac{d_v}{2} \rfloor}^{n-d_v} \sum_{c=\max(0,n-d-a)}^{d_v-(n-d-a)} w_{\text{Type}}(a,c,d_v,n) \left(\binom{n-d_v}{a} \cdot \binom{d_v}{c} \right),$$

其中，引入权重系数$w_{\text{Type}}()$，其取值可以是：

$$w_{\text{Binary}}(a,\ c,\ d_v,\ n) = 1 \tag{7}$$

$$w_{\text{Linear}}(a,c,d_v,n) = \frac{a+c}{n} \cdot \frac{a+(d_v-c)}{n} \tag{8}$$

$$w_{\text{Exp}}(a,c,d_v,n) = \exp\Big(-\beta\big(n-(a+c)\big)\Big) \cdot \exp\Big(-\beta\big(n-(a+(d_v-c))\big)\Big), \tag{9}$$

其中，w_{Binary} 是对所有内容均赋予 1 的原始权重； w_{Linear} 会从完美向量匹配时的 1 线性衰减至向量间距达到允许的最大距离时的 0；w_{Exp} 则是指数衰减权重，使用 β 系数作为其衰减率。

正如我们接下来将展示的，线性加权会对模式与查询之间正中位置的模式赋予最大权重，并以该点为中心逐渐对称地降低权重。与此同时，指数加权相互抵消，对所有内容施加恒定的权重重新缩放。令 x 表示神经元到其所对应读写向量的距离，则有： $x \coloneqq (n-z)-(a+c) = b-z$。由于我们想了解对模式与查询之间所有可能距离处的神经元所适用的加权系数，不失一般性，对于接下来的分析，我们可以设定 $z=0$：

$$\begin{aligned} w_{\text{Linear}}(a,c,d_v,n) &= \frac{n-x}{n} \cdot \frac{n-((d_v+z)-x)}{n} \\ &= \frac{n-x}{n} \cdot \frac{n-(\mathcal{C}-x)}{n} \\ &= \frac{n^2-x^2-n\mathcal{C}+x\mathcal{C}}{n^2} \\ &= Z + \frac{-x(x-\mathcal{C})}{n^2} \end{aligned} \tag{10}$$

[12]这一公式最初受（Jaeckel，1989b;a）的启发。

其中，$\mathcal{C} = (d_v + z)$是一个常数，$Z = \frac{n^2 - n\mathcal{C}}{n^2}$也是如此。对公式 10 求一阶和二阶导数可知，读写向量正中间位置的距离处所获得的权重最大，$\mathcal{C}/2$。这种线性加权会对不同距离处的神经元赋予不同权重。因此，叠加存储的模式也会得到不同的加权。不过，正如图 7 所示，从经验上看，这对 SDM 指数近似的影响可以忽略不计。我们推测这由两个因素造成：（i）权重值之间的差异并不特别大；（ii）获得最大权重的神经元数量最多。因此，随着向量距离变化，加权与圆交集中神经元数量近似呈指数衰减的现象相关。

至于指数项，它会抵消为一个取决于 β选择的常数项：

$$\begin{aligned} w_{\text{Exp}}(a, c, d_v, n) &= \exp\big(-\beta x\big) \cdot \exp\big(-\beta(\mathcal{C} - x)\big) \\ &= \exp\big(-\beta \mathcal{C}\big). \end{aligned} \tag{11}$$

随后，softmax 操作中的归一化项会消除这个对所有模式的加权所作的常数项调整，因此不会影响输出。

我们在图 7 中，通过整个研究中使用的 d^* 和 $n = 64$ 维标准 Transformer 注意力设置（Bricken 和 Pehlevan，2021），对三个最优 SDM 汉明距离的结果进行了实证验证。

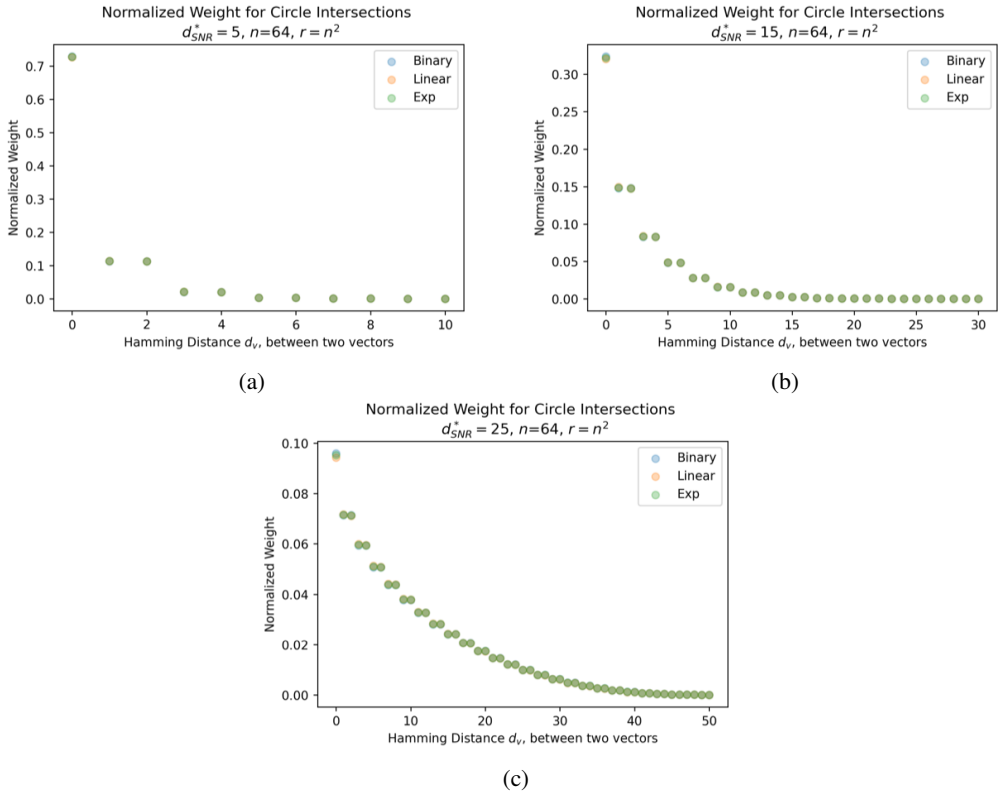


图 7：**对神经元读写操作加权对 SDM 的影响微乎其微**。正如预期，指数加权（绿色）与二元加权（蓝色）的结果相同，二者完全重合。线性加权（橙色）造成的差异微乎其微，由于高度重合，几乎看不出来。

B GABA 开关

B.1 死神经元问题的生物学上不合理解决方案

死神经元问题是指某些神经元从未被任何输入数据激活，因而造成资源浪费。使用 Top-K 激活函数时，这一问题尤为突出，特别是在初始化阶段：少数恰好已经存在且最接近数据流形的神经元可能总是被激活并进入 Top-K。这意味着其他神经元永远不会被激活，因此也永远不会收到权重更新，无法学习数据流形并变得有用。

解决死神经元问题的第一类策略，是将神经元初始化在数据流形上或其附近，例如使用主成分分析或奇异值分解（Rumelhart & Zipser，1985；McInnes & Healy，2018）。从生物学角度看，这意味着数据流形位置的先验信息被编码在我们的基因组中，并预先决定了神经元树突的结构。然而，树突的形成往往具有高度动态性和随机性（Dhar 等，2018；Purves 等，2001）。

此外，这种解决方案要求基因组存储大量信息，并会限制生物体在发育过程中学习新数据流形的适应能力，例如迁移到新的生态位。在深度学习中，常见的做法是不直接使用数据来初始化权重，而是采用 Xavier 初始化（Glorot & Bengio，2010）等经过优化的策略；这些策略经验证效果良好，即使对于稀疏的 ReLU 激活函数也是如此。然而，初始化策略仍在不断发展，这表明这种方法依然是启发式的，无法在不同数据集、网络和训练技术之间普遍适用（Aguirre，2019；Chang 等，2020b；Goodfellow 等，2015）。

第二类解决死神经元问题的方法，确保所有神经元无论初始状态如何，都会激活并更新，以学习数据流形。这是人们经常避免使用 ReLU 的一个重要理由（Goodfellow 等，2015；Hendrycks & Gimpel，2016），尽管在实践中，ReLU 往往似乎并不会显著损害性能（Pedamonti，2018；Trockman 和 Kolter，2022）。(Ahmad & Scheinkman，2019) 提出的 Top-K 模型、VQ-VAE（van den Oord 等人，2017）以及 Switch Transformer（Fedus 等人，2021）都以不符合生物学原理的方式处理了各自的死神经元问题。使用 Top-K 激活函数时，(Ahmad & Scheinkman，2019) 的研究人为地提高了每个神经元进入 Top-K 的排名，且不活跃的神经元越多，排名提升得越高。这种解决方案做出了不符合生物学原理的假设：神经元会根据自身的不活跃程度而放电，并普遍经历反赫布式可塑性。与此同时，VQ-VAE 和 Switch Transformer 都在各自的损失函数中使用了相关项，分别提高“死亡”代码向量和专家模块的利用率。

B.2 GABA 开关生物学

图 8 总结了 GABA 开关在生物学上的工作原理（Gozel & Gerstner，2021）。在发育早期，神经元会受到 GABA 的兴奋，之后由于细胞内 CL^- 浓度发生变化，GABA 转而抑制神经元。神经发生后，神经元会表达 NKCC1，将 CL^-转运进细胞，使 CL^- 反转电位高于静息电位（Gozel & Gerstner，2021；Heigele 等人，2016）。存在 GABA 时，CL^- 会流向其反转电位，从而导致去极化。随着时间推移，这种 GABA 激活会间接增加 KCC2 的表达，而 KCC2 会将 CL^- 泵出细胞（Ganguly 等人，2001；Connor 等人，1987）。这会使 CL^- 反转电位低于静息电位，并在 GABA 激活期间引起超极化。
这一生物学机制表明，我们在公式 12 中的 GABA 开关实现可以做得更加复杂。与其统计神经元单纯处于激活状态的次数（二元结果），不如让开关根据神经元实际经历的激活量或梯度更新的幅度进行切换。

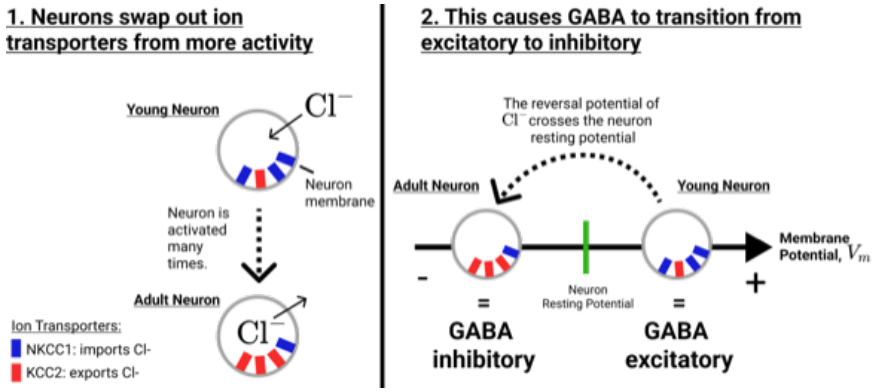


图 8：**GABA 开关生物学概述**。 此处展示了年轻与成年神经元内部的离子转运体，以及它们随时间发生的变化。图的左侧显示，随着神经元被激活的次数增多，其离子转运体的浓度组成会从 NKCC1（蓝色）转变为 KCC2（红色）。这会使神经元从摄取氯离子（Cl-）转变为排出氯离子。图的右侧展示了这一转变的结果：氯离子的反转电位从高于神经元静息电位（使 GABA 具有兴奋性）变为低于静息电位（使 GABA 具有抑制性）。

B.3 GABA 开关的实践考量

本节关于何时以及如何使用 GABA 开关来避免死亡神经元、提升持续学习效果，得出了三点结论：

- 如果使用正权重约束，那么退火 k 的性能与完整的 GABA 开关相同。如果允许使用负权重，则应使用 GABA 开关。
- 与掩蔽相比，减法操作更有利于持续学习。不过，在非持续学习场景中则恰恰相反。我们对这一现象的原因进行了初步分析。
- 为 GABA 开关选择 s 的值时，应考虑数据流形的复杂度、优化器以及学习率。若不确定，增大 s 也无妨，只是需要更多训练轮次。

我们首先给出完整的 GABA 开关算法，该算法是在本文中开发并用于 SDMLP 的。随后，我们概述为何当权重和输入被限制为仅取正值时，k 退火近似仍同样有效。接下来，我们讨论为何通过相减来实现 Top-K，而非采用掩蔽，能够带来更好的持续学习性能。最后，我们探讨其他注意事项，说明如何通过数据流形、学习率和超参数 s之间的相互作用来避免死亡神经元。

GABA 开关算法

形式上，GABA 开关算法的实现如下：

$$\begin{aligned} a_i^* &= [a_i - \lambda_i I]_+ \\ I &= \text{descending-sort}([\mathbf{a}]_+)_{(k_{\text{target}}+1)} \\ \lambda_i &= \min\Big(1, \max\big(-1, -1 + 2C_i/s\big)\Big) \end{aligned} \tag{12}$$

该算法比正文公式 4 中介绍的 K 退火算法稍复杂。C_i 是每个神经元的计数器，用于记录自训练开始以来该神经元被激活的次数。λ_i 随 C_i线性增加，从 -1 变为 1；$[\cdot]_+$ 表示 ReLU 操作。s 是一个超参数，用于确定该神经元需要被激活多少次，才能从受到 GABA 的兴奋切换为受到抑制。在使用 ConvMixer 嵌入进行训练时，我们发现 $s = 250{,}000$ 足以确保所有神经元都能移至数据流形上，因此没有神经元死亡。举例来说，假设 GABA 开关触发前所有神经元都已激活，那么取此 s 值时，在 CIFAR10 上训练仅 2.5 个训练轮次后便会发生切换（共有 50,000 个训练样本）。然而，该值很大程度上取决于学习率以及流形的复杂程度；例如，在原始 CIFAR10 图像上训练时，为确保没有死亡神经元，我们将 $s = 5{,}000{,}000$ 设为该值，因此 GABA 开关要经过 100 个训练轮次才会触发。不过，在这种情况下，我们没有尝试寻找 s的下限。

为什么 k 在权重为正时，退火的效果与 GABA 开关同样好

由于戴尔定律，我们使用正权重和正输入来实现 SDM。经验上，这也能提升持续学习的性能，并引入权重稀疏性[13] ；如果能利用非结构化稀疏性，还可提高计算效率（Wang, 2020）。

使用正权重时，我们发现采用公式 4 线性退火并不会改变死亡神经元的数量或持续学习性能。ImageNet32 ConvMixer 嵌入和原始像素数据集都是如此。然而，允许使用负权重时，情况便不再如此，只有完整的 GABA 开关算法能够避免神经元死亡。我们推测，这是因为当 GABA 具有兴奋性时，完整算法会向原本可能具有负活动、因而不会触发并无法获得梯度更新的神经元注入正激活。与此同时，当权重和输入均为正时，每个神经元的激活始终为正，因此除非强制采用 Top-K，否则所有神经元都会处于激活状态。因此，正权重和正输入

[13]我们发现，检查的两个模型（分别在 ImageNet32 ConvMixer 嵌入表示上训练，以及直接在像素上训练）中，有 $\sim 20\%$ 的权重值小于 0.01，这意味着它们很可能可以被剪枝。不过，关于稀疏性的进一步研究，包括引入 L^1 损失等，将留待未来工作。



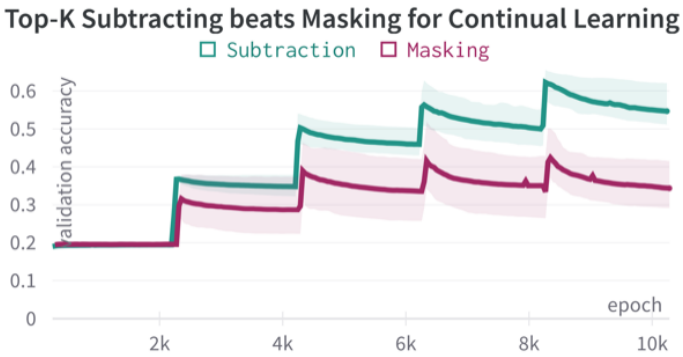
图 9：**线性 k 退火需要正权重**。除允许使用负权重时的线性退火（绿色）外，所有曲线都在图表底部重合。GABA 开关会激活那些可能因负权重而失活的神经元。这确保所有神经元都会更新到数据流形上，而不会成为死亡神经元。然而，公式 4 中提出的线性 k 退火不会激活这些神经元，导致它们成为死亡神经元。训练直接在 CIFAR10 流形上进行，这比在 ConvMixer 嵌入上训练更难。

确保每个神经元都能获得梯度更新，并移动到数据流形上。在图 9 中，我们展示了直接使用 CIFAR10 像素进行训练的结果；与 ConvMixer 嵌入相比，这一流形更难学习。

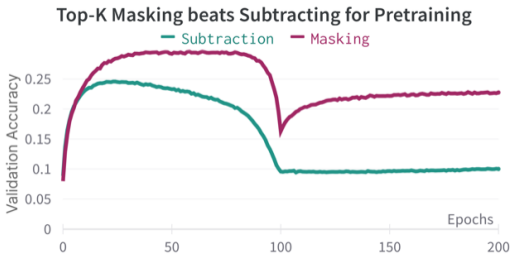
请注意，虽然正权重允许进行线性 k 退火，但也可能导致所有权重都被设为 0，从而产生死亡神经元并导致模型训练失败。

为什么减法比掩蔽更适用于持续学习

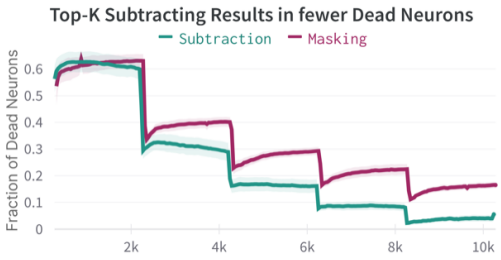
从实证结果来看，预训练期间，掩蔽的验证准确率远高于减法（图 10b）。然而，正如表 2 和图 10a 中的消融实验所示，在持续学习期间，减法的效果更好。为了弄清减法为何更适用于持续学习，我们在附录 H 中分析了两种方法的学习动态，并在此讨论。至于掩蔽为何在预训练期间效果更好，我们将其留待未来研究，因为在此我们并不关心如何最大化预训练任务的验证准确率。



(a)



(b)



(c)

图 10：（**a**）在持续学习中，减法优于掩蔽。然而，如（**b**）所示，在预训练中减法效果较差。在预训练期间，两种方法都不会产生死亡神经元。（**c**）在持续学习期间，随着每个任务的进行，采用减法时死亡神经元更少，这或许可以解释其对先前任务更好的记忆保留。由于这些图表并不拥挤，误差条显示的是五种随机持续学习数据划分中的最小值和最大值范围，而不是均值标准误。

如图 10c 所示，在持续学习期间，采用减法的 Top-K 会使用更多神经元。附录 H 的图 23 也支持这一结果：持续被激活的神经元更少，而且这些神经元具有更强的多义性，会被用于多个任务。

受到附录 D 中陈旧动量结果的启发，我们想知道，掩蔽所保留的较大激活值，在使用 SGDM 作为基于动量的优化器时，是否会导致更多死亡神经元。然而，使用随机梯度下降并未改变结果；事后看来，我们的梯度小于 1，这意味着附录 D 中的结果不受梯度值影响。

这意味着，减法对应的梯度实际上必须更小，从而导致更新速度更慢。乍看之下，减法操作只是通过一个常数项降低了仍保持激活的 k 个神经元的活动程度。这会缩放每个梯度的大小，效果等同于修改学习率。

然而，情况更为复杂。 k 个神经元的活动程度及其学习率取决于第 $k+1$ 个神经元的活动程度。如果这个神经元的活动程度与 k 子集中的神经元几乎相同，那么它们的活动程度都会非常接近零。与此同时，如果第 $k+1$ 个神经元的活动程度远低于 k 子集中的神经元，那么它们的活动程度将保持较高，并获得更大的梯度更新。这两种情况分别对应于输入位于数据流形上密集铺展区域和稀疏铺展区域。
我们可以将这种情况视为对贝叶斯情形的近似：在这种情形下，我们用高斯混合分布来定义数据流形上的后验分布。某个区域内神经元的数量与该位置分布的方差成反比，因此会影响当前数据点的似然对分布的更新幅度。换言之，减法操作引入了动态学习率。

在持续学习过程中，当任务发生变化并呈现数据流形的一个新子区域时，我们希望距离该子区域最近的神经元且仅有这些神经元进行更新，并向该子区域移动。在随机优化的设置中，距离新子区域不一定最近的神经元也可能被更新并向其移动，从而导致分配给新任务的神经元过多。我们推测，减法操作通过降低学习率并使其动态变化，有助于避免这一问题。不过，我们认为动态学习效应的影响比随机性更大，因为即使将批量大小从 128 增至 10,000，减法对于灾难性遗忘仍然更具鲁棒性。

降低且动态变化的学习率很可能是一把双刃剑：虽然它更擅长记忆，但学习新信息的能力也更差。无论是在预训练期间，还是在每个数据划分内，减法操作都导致更差的训练和验证准确率。至于为何在单个任务的预训练过程中减法操作表现更差，我们留待未来研究探讨。

顺带一提，当 $k=1$ 时，由抑制性中间神经元实现的 Top-K 减法操作，与 (Chang 等人，2020a) 中采用的维克里第二价格拍卖相同；该拍卖将神经元建模为各自拥有独立效用函数的个体代理。获胜神经元的活动量相当于它的“出价”。它的活动量会减去活动量第二高的神经元的活动量（也就是支付第二高的价格），而剩余活动量则决定了它受到下一次梯度更新影响的程度。

GABA 开关的其他考量——数据流形、优化器与学习率 **s**

请注意，对于 ConvMixer 嵌入，死亡神经元并非那么棘手的问题；无论是否采用正权重，GABA 开关或 k 退火都能在短短几个训练轮次内避免出现死亡神经元。这是因为该流形更容易学习：向量的维度更低（256 对比 3072），而且数据流形中的相关性也更有意义。

图 11 展示了学习率过低会导致随机梯度下降产生死亡神经元。死亡神经元的数量与学习率成反比，而在 GABA 开关之后，死亡神经元数量骤增并很快趋于平稳，这两点都支持学习率过低这一理论。这意味着，在 GABA 开关发生之前，神经元无法足够快地更新到数据流形上。死亡神经元数量骤增并趋于平稳的现象，与图 15 所示的 Adam 和 RMSProp 优化器形成对比；由于过时动量问题，这些优化器中的神经元会随着时间推移持续死亡（Kingma 和 Ba，2015；Geoffrey Hinton，2012）。与此同时，SGDM 并

不会同等程度地受到陈旧动量问题的影响，而且对学习率的选择要稳健得多，因此是预训练的不错选择。不过，在持续学习设置中，它仍会在一定程度上受到过时动量问题的影响，如表 2 的消融实验所示。

为避免使用随机梯度下降时出现死亡神经元，可以调整学习率、提高 GABA 开关值 s，或者使用能够避免过时动量问题的稀疏优化器（不过，这些稀疏优化器的效果仍有待未来工作验证）。还要注意，所选学习率会受到 L^2 归一化的影响。



图 11：**GABA 开关需要考虑学习率**。我们直接在 CIFAR10 像素上训练 SDM，以测试其学习流形并避免死亡神经元的能力。GABA 开关在训练轮次 ∼150 时触发，每条线代表不同的学习率 “lr”。根据附录 D 中概述的原因，随机梯度下降应产生最少的死亡神经元。不过，学习率需要根据 GABA 开关阈值 s 仔细设定。如果学习率过低，神经元将无法及时更新至数据流形上，反而会死亡。

C Top-K

C.1 抑制性中间神经元近似 Top-K

从宏观上看，抑制性中间神经元的作用是调节兴奋性神经元的放电，引入稀疏放电，并仅保留最活跃的神经元。[14] 这意味着，在任意时刻，大脑中放电的神经元数量很可能非常稀少。估算表明，“人脑中的一个神经元平均每秒传递约 0.1–2 个尖峰。”（Impacts，2022年）；而动作电位和不应期发生的时间间隔约为 10 毫秒（Sterling & Laughlin, 2015）。假设神经元在这 10 毫秒的时间间隔内随机放电，则粗略估算可得，在该时间间隔内放电的神经元最多占 10ms/100ms = 10% 。这与（Attwell & Laughlin，2001年）预测的 15% 十分吻合。从代谢成本的角度来看，稀疏放电也很合理，因为动作电位消耗大量能量，占大脑能量的 ∼20%（Sterling & Laughlin, 2015；Attwell & Laughlin，2001年；Sengupta 等，2010年）。

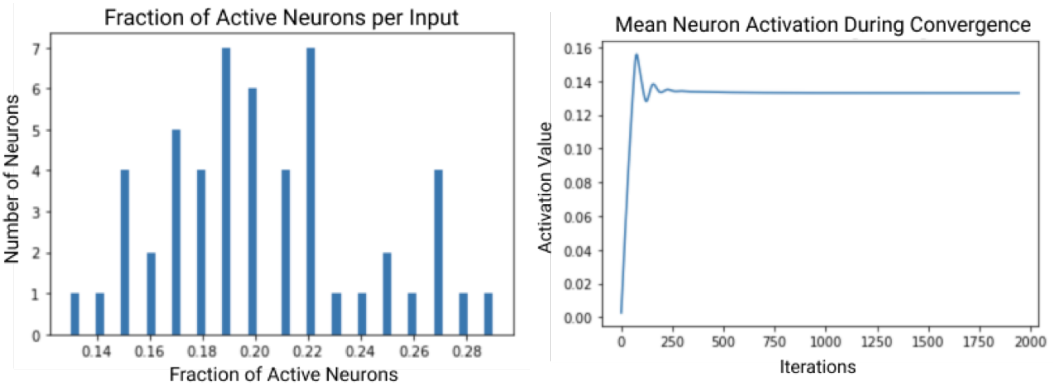
然而在实践中，假设抑制性神经元能够针对任意给定的输入，始终恰好让 k 个神经元保持活跃是不现实的，因为它的传入和传出突触连接并不均一。抑制性中间神经元会将所有神经元的激活加总，并根据它们的突触前强度进行加权，然后输出一个按突触后连接强度缩放的抑制值。对输入进行加权求和会抹去神经元放电数量及其放电频率的信息。不同的突触后连接会对抑制作用于各个神经元的效果赋予不同权重。

考虑到这些问题，我们仍有几个理由乐观地认为，Top-K 可以得到近似实现。首先，视网膜中的抑制性水平细胞会计算数千个邻近视锥光感受器的平均激活值，并抑制它们，从而使它们能够编码对比度（Sterling & Laughlin, 2015，第13章）。水平细胞不仅为其缝隙连接精细调节权重以计算这一平均值，还会动态调整其突触前和突触后连接强度，以便在不同光照条件下准确计算平均活动。在弱光环境下，水平细胞会对更多神经元求平均，以降低其均值估计的方差（Sterling & Laughlin, 2015，第252页）。如果一般的抑制性中间神经元在校准突触连接方面能够接近水平细胞所展现的精细程度，那么中间神经元就有可能计算出应当施加多大抑制，从而可靠地实现 Top-K。

其次，从小脑样结构中 Top-K 的生物学证据来看，有强有力的证据表明，在果蝇蘑菇体中，Top-K 由其高尔基中间神经元类似物加以近似（Lin 等，2014）。

[14] 抑制性中间神经元也可以相互抑制，从而产生去抑制，但这种机制仍可用于实现更复杂的兴奋性神经元调节。

在哺乳动物中，相关证据更为复杂，一些论文发现颗粒细胞激活是稠密的（Giovannucci 等，2017）。然而，其他实验研究和理论表明，哺乳动物小脑中的颗粒细胞稀疏性可能取决于任务复杂度（Lanore 等，2021；Xie 等，2022）。任务复杂度越低，表征就可以越稠密，因为需要以独特方式编码的刺激更少。与此同时，高复杂度任务需要更多的正交编码，因此需要更稀疏的表征，但这会导致保真度下降（Xie 等，2022）。归根结底，颗粒细胞放电的密度仍是一个尚无定论的问题；要解决这一问题，需要延迟更低的电压指示剂，并能在更多样化的任务中同时记录更多神经元（Lanore 等，2021）。



(a) (b)

图12：**抑制性中间神经元近似 Top-K**。我们复现了 (Gozel & Gerstner，2021) 的赫布学习规则。**(a)** 我们展示了在 50 个不同的 MNIST 输入中，1000 个神经元里有 $k \approx 200$ 个处于激活状态的近似情况。**(b)** 我们展示了一个示例：将公式 13 中的常微分方程运行至收敛后，神经元活动的平均值。

最后，我们尝试对抑制性中间神经元近似 Top-K 的能力进行建模，并简化假设为其突触前和突触后权重固定且均匀。首先，我们使用 (Gozel & Gerstner，2021) 的赫布学习动力学，分析仍保持激活的 k 个神经元。随后，我们考察了学习过程中的 SDMLP，以及兴奋性激活总和与公式 4 中使用的第 $k+1$ 个抑制值之间的关系，以确保仅有 k 个神经元仍保持激活。

介绍 GABA 开关的研究（Gozel & Gerstner，2021年）采用了带有抑制性中间神经元的赫布学习规则。向网络呈现一个 MNIST 数字供其学习，网络遵循以下常微分方程来确定兴奋性神经元和抑制性神经元的活动：

$$
\begin{aligned}
\text{兴奋性}^0 &= W^T_{\text{inp}}\mathbf{x} \\
\text{抑制性}^0 &= 0 \\
\\
\mathbf{p}_a^t &= \mathbf{e}^0 + W_{IE}\mathbf{i}^{t-1} \\
\mathbf{p}_i^t &= \mathbf{e}^0 + W_{EI}\mathbf{e}^{t-1} \\
\\
\mathbf{g}_a &= \frac{1}{\tau_a}\big(-\mathbf{e}^{t-1} + \tanh\left([(\mathbf{p}_a^t - \mathbf{b})/L]_+\right)\big) \\
\mathbf{g}_i &= \frac{1}{\tau_i}\big(-\mathbf{i}^{t-1} + [(\mathbf{p}_i^t - b_i)]_+\big) \\
\\
\mathbf{e}^t &= \mathbf{e}^{t-1} + \Delta\mathbf{g}_a \\
\mathbf{i}^t &= \mathbf{i}^{t-1} + \Delta\mathbf{g}_i,
\end{aligned}
\tag{13}
$$

其中，$\mathbf{e}$是描述兴奋性神经元活动的向量，$\mathbf{i}$ 表示抑制性神经元，Δ 是学习率。兴奋与抑制时间常数分别为 τ_a 和 τ_i。L 是平滑项，$\mathbf{b}$ 是为每个神经元学习得到的放电阈值。W_{IE} 和 W_{EI} 分别是从兴奋性神经元到抑制性神经元以及反向的权重矩阵；它们是固定的，且权重均一。W_{inp} 是对输入 $\mathbf{x}$作出响应的权重。b_i 是抑制性中间神经元的放电阈值，它是一个固定标量，可视为产生了一个带噪声的近似 k 值。训练过程中，我们采用赫布学习规则（未给出）训练 W_{inp}，并运行所示常微分方程，使网络收敛到约有 k 个神经元保持激活的状态。

这些动力学过程是简化的，因为兴奋性神经元与抑制性神经元之间的权重固定且均匀。W_{EI} 初始化时，每个兴奋性神经元以 90% 的概率与每个抑制性神经元相连，连接强度为 1。W_{IE} 以相同的连接概率初始化，但连接强度为 $1/(0.9r_I)$，其中 r_I 是抑制性神经元的数量。这样，连接抑制性神经元和兴奋性神经元的突触权重之和在期望上为 1。
暂且不考虑这些简化假设，当我们运行模型并分析仍保持激活的神经元数量时，如图 12a 所示，该数量约为一个常数 k 个神经元。给定抑制激活阈值 b_i，每个输入下约有 20% 的神经元保持激活，即 $k \approx 200$。图 12b 还展示了兴奋性神经元平均活动随时间的变化，以强调抑制性中间神经元无需一次性猜出如何正确抑制除 k 个神经元以外的所有神经元。相反，这是一个由式 13 的常微分方程定义的动态过程。

我们还考察了 SDMLP 中抑制性神经元用于实现 Top-K 的第 $k+1$ 高活动值，以及该值与进入抑制性中间神经元的神经元激活值无权重总和之间的关系。在 GABA 开关之后，当神经元铺满数据流形并形成子网络时，这种关系大体呈线性：第 $k+1$ 高活动值等于抑制发生前所有神经元激活值的总和除以神经元总数。这同样是基于所有输入和输出权重均相同且等于 1 这一假设得出的简化结果。不过，如果该中间神经元能够像水平细胞一样精确地调整其突触前与突触后强度（Sterling & Laughlin, 2015），那么这些结果支持这样一种可能性：中间神经元可以通过一个简单函数对其输入总和作出响应，从而近似实现 Top-K。

C.2 优化的 Top-K

k 值是我们模型中必须选定的一个参数，并且会显著影响性能。我们展示了在多个不同数据流形上，对预训练和持续学习中的 k 进行消融的实证结果。

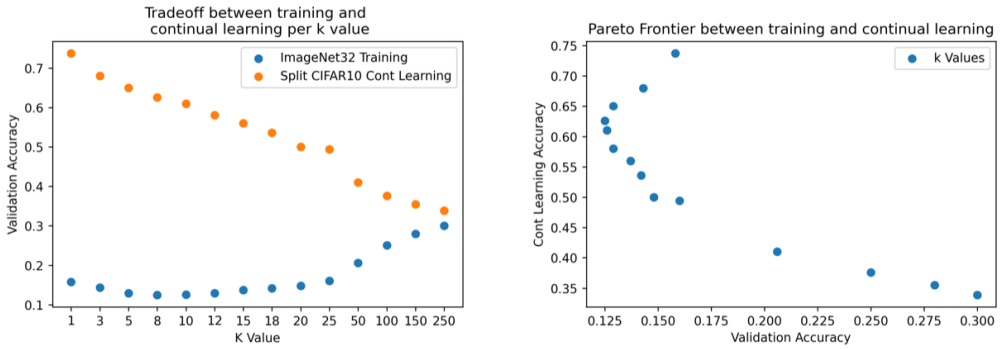
总结我们的发现：

- 更大的 k 有利于预训练，但不利于持续学习，反之亦然。
- k 取决于数据流形的复杂度和神经元数量，r。
- SDM 会大致铺满所提供的数据流形。将 SDM 与卷积神经网络等其他模型组件联合训练，有助于提升预训练效果。然而，在这种联合训练设置下，SDM 难以学习非静态流形。

预训练或持续学习的最佳 k 值?

在 SDM 设置中，稀疏性带来了一个根本性的权衡：在持续学习（CL）和非持续学习（NCL）设置下的表现难以兼顾。这种权衡之所以存在，是因为稀疏性会限制模型的表征能力，因此预计会降低 NCL 的准确率。同时，模型越稀疏，就越能形成不会被未来持续学习任务覆盖的子网络。此外，SDM 会铺满所提供的数据流形，而根据流形的不同，这一特性可能与最大化分类准确率的目标相关性较弱。

我们在图 13 中看到了这种权衡。该图总结了在 ImageNet32 上对 ConvMixer 的嵌入进行预训练并在拆分 CIFAR10 上测试时，消融 k 的结果。随着 k 值降低，持续学习准确率有所提高，而预训练准确率则有所下降。然而，原始 ImageNet32 训练集上的性能为何呈抛物线变化尚不清楚；当 $k < 8$时，性能略有提升。虽然我们



(a) (b)

图 13： k 值如何影响模型学习 ImageNet32 预训练任务和 Split CIFAR10 持续学习任务的能力的两种视角。**(a)** 最终 NCL ImageNet 预训练（蓝色）和最终持续学习 Split-CIFAR10（橙色）在所有任务上的验证准确率随 k 值的变化。**(b)**ImageNet 预训练准确率（横轴）与 CIFAR10 持续学习（纵轴）之间的帕累托前沿。每个蓝点代表不同的 k 值。该图清楚地显示了 $k < 8$更优的预训练表现。

在这里并不关心 NCL 预训练任务，而是将其作为一种让神经元在真实世界图像构成的任意流形上实现专门化的方法，但仍值得指出，性能有所下降。在单次训练过程中，随着 GABA 开关切换且 k 逐渐退火，性能下降更明显地表明稀疏性导致了这种性能下滑。图 14a 展示了较小的 k 值下性能下降的情况。
为突出最优 k 值如何随神经元数量和数据流形而变化，我们描述了更多结果：

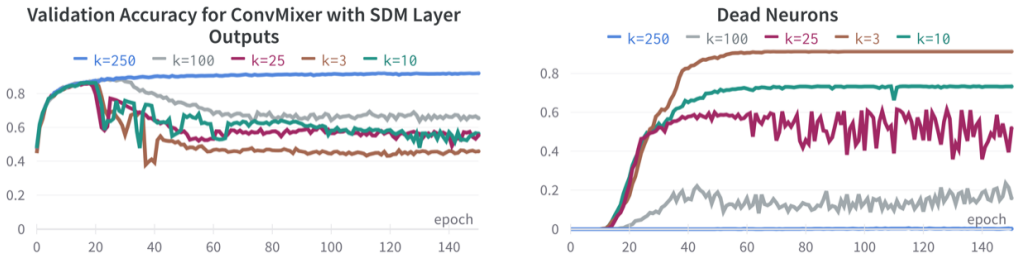
对于相同的 ImageNet32 预训练和 Split CIFAR10 测试 ConvMixer 嵌入，如果使用 10K 个神经元而不是 1K 个，那么预训练表现会像图 13 中那样线性下降，但 $k = 5$ 优于所有 $k \leq 50$，在持续学习方面也表现最佳。与此同时，$k = 1$的预训练表现最差，而持续学习表现排名第二。

再次采用相同的设置，但直接在像素上操作而不使用 ConvMixer 嵌入时，$k = 10$ 最适合预训练，而 $k = 1$ 在持续学习方面表现最佳，如附录 E.1 的图 18 所示。

在上述所有情况下，我们都为 SDM 提供了一个静态流形供其学习。与性能相当的 ReLU 多层感知机相比，这总会导致更差的预训练准确率，即使使用较大的 k 值也是如此。这是因为 ReLU 多层感知机不受数据流形约束，它具有偏置项，且没有 L^2 归一化。因此，它能够学习权重和偏置，从而更好地最大化 NCL 验证准确率。

如果我们对 ConvMixer 和 SDM 模块进行联合训练，那么 ConvMixer 就能学会为 SDM 创建一个可供其铺排、且能最大化验证准确率的流形，其表现可与 ReLU 多层感知机相当。在一项测试中，我们在 NCL 设置下使用整个 CIFAR10 数据集进行训练时，只要满足 $k \geq 250$，就会出现这一结果；SDM 层中有 1,000 还是 10,000 个神经元则无关紧要。图 14a 展示了这一结果，它表明 ReLU 网络的最后一层只需至少有 250 个神经元，就能通过反向传播传递梯度并实现较高的预测准确率。有证据表明，人工神经网络在训练开始时存在过度参数化，并且在功能上是稀疏的；网络剪枝的研究结果为此提供了支持，其中最著名的是彩票假说（Frankle & Carbin，2019）。

然而，这一结果再次取决于流形：即便使用 $k = 2$，500 个或 10,000 个神经元在 ImageNet32 上训练，性能仍不如 ReLU 网络。我们认为，这是因为 ImageNet32 的数据流形复杂得多，包含 $\sim$ 1.2M 张图像，分属 1,000 个不同类别；相比之下，CIFAR10 只有 10 个类别、50,000 张图像。这意味着，即使 a k 为 2500，也太小了，会损害模型的表征能力。



(a) (b)

图 14：**过小的 k 取值会损害网络训练**。在 CIFAR10 上训练末端带有 SDM 层的 Conv Mixers。所有网络在最初 20 个训练轮次中的表现相同，此时 Top-K 已基本完全实现（GABA 开关约在第 10 个训练轮次时触发）。**(a)**所有模型的验证准确率。 k 值越小，准确率越低。**(b)** 对任何验证输入都未激活的神经元比例（这是训练数据中可能未激活的死亡神经元实际数量的上限）。同时训练网络的卷积神经网络部分与 SDM 神经元，会导致许多神经元死亡。我们认为这是因为流形仍在更新。随着更多神经元死亡，模型的表征能力会下降。

联合训练的另一项难点，是确定 GABAswitch 的切换时机，以避免出现死亡神经元。这是因为 SDM 学习的流形会随时间动态变化，我们认为这也解释了图 14b 所示与 k 成反比的神经元死亡现象。

我们将最优 k 值以及联合训练与使用冻结的预训练模型之间的比较留待未来工作进一步研究。我们认为，要解决稀疏性限制表征能力所带来的权衡，最佳方法是加宽网络层，而不是减少可激活神经元的数量，即 k。这种做法是可行的，因为稀疏激活的计算成本很低——计算 k 个最活跃的神经元需要将输入与每个神经元相乘（这可以并行完成），然后进行排序；而固定数量的 $k << r$ 个神经元会产生输出并接收梯度更新。使用稀疏权重也能降低计算哪些神经元处于激活状态的成本（Jaeckel，1989a；Ahmad & Scheinkman，2019）。

运用 SDM理论推导最优 k 值

在结束本节之前，我们想提醒读者，SDM 在采用一些简化假设的情况下，能够针对几种不同的最优性条件解析推导出最优 k 值：(i) 优化每个模式的信噪比；(ii) 在满足特定检索概率的前提下，优化可存储的模式数量上限；(iii) 在仍能正确收敛的前提下，最大化查询与其目标模式之间的距离。相关推导和进一步讨论见（Kanerva，1993；Bricken 和 Pehlevan，2021）。

虽然这些解析结果确实存在且值得关注，但它们都致力于最大化所存储模式的信息内容，而这与重建任务最为相关。在本文中，我们关注两个与最大化信息内容关联不大的不同目标函数：(i) 分类准确率：无需尝试对数据流形建模，也能取得更好的性能；(ii) 持续学习：我们既关注数据流形的建模，也希望形成独特的子网络，以输出正确的分类标签。此外，SDM 的解析结果假设模式是随机的。对于存在相关性的真实世界数据集，我们需要找到量化数据流形复杂度的方法；而对这一问题及其解决方案进行全面探究，超出了本文的讨论范围。

D 陈旧动量

假设某个神经元在三个训练批次中被激活，接着在之后的 10 个批次中处于非激活状态，最后又被激活两次。在第一组激活期间没有问题：神经元会利用最新的梯度信息更新其权重和动量项。在非激活期间，问题开始出现，因为处于非激活状态的神经元仍会更新其权重，而且使用的

动量项会越来越过时。当该神经元再次被激活时，问题仍会持续，因为动量项衰减得很慢，会显著增强指向过时方向的梯度。

这些陈旧动量在 Top-K 竞争激活设置下尤其有害：在这种设置中，神经元必须至少对一个输入成为最 k 活跃的神经元，才能继续接收梯度更新，否则就会永久失活。使用不含任何动量项的随机梯度下降（SGD）可以消除这一问题。带动量的 SGD 效果不如不带动量的 SGD，但仍优于 Adam 或 RMSProp （ Kingma 和 Ba，2015； Geoffrey Hinton，2012； Goodfellow 等，2015）。

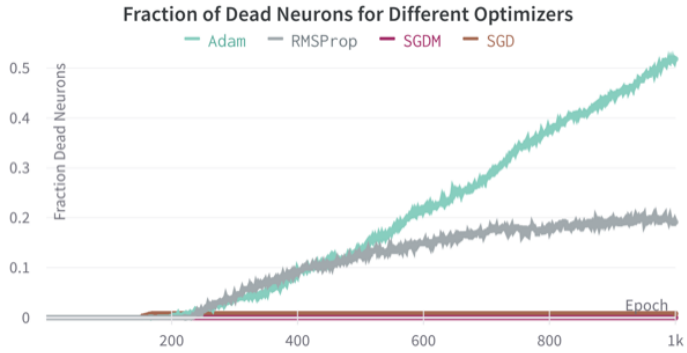


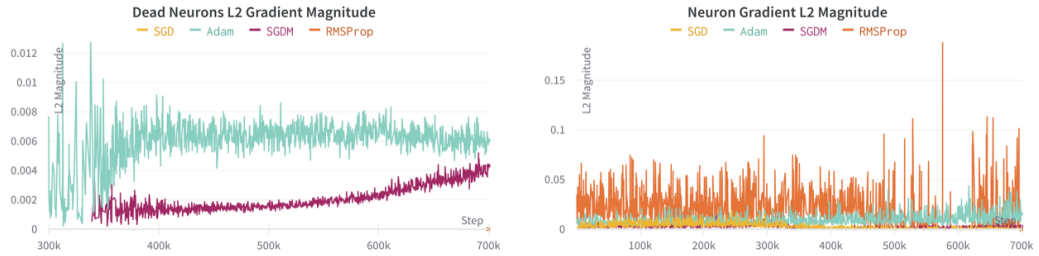
图 15：GABA 开关在第 ~100 个训练轮次时触发，到第 ~200 个训练轮次时 Top-K 已完全实现。此时，RMS 和 Adam 优化器开始不断使神经元失活，导致它们在任何训练或验证数据上都不再被激活。这些结果是在 CIFAR10 图像上直接训练得到的。关于在 ConvMixer 嵌入上训练的讨论，请参阅正文。

我们在直接使用 CIFAR10 图像训练模型时发现了陈旧动量效应；与 ConvMixer 嵌入相比，这些图像的数据流形复杂得多。图 15 显示，GABA 开关触发后，当 Top-K 在第 200 个训练轮次左右完全实现时，Adam 和 RMSProp 仍会不断使神经元失活（Kingma 和 Ba，2015；Geoffrey Hinton，2012）。这些神经元之所以被认为已经失活，是因为它们不会被任何训练或验证数据激活。我们的结果在不同优化器的各种学习率下都很稳健。
虽然使用 Adam 和 RMSProp 在 ImageNet32 嵌入上进行预训练时不会出现这些死亡神经元，但如表 2 的消融实验所示，陈旧动量问题仍会导致错误的梯度更新，从而损害持续学习。这是因为覆盖数据流形并能够保留不同任务记忆的神经元更少。

值得注意的是，使用 Adam 或 RMSProp 训练的 ReLU 网络直接在 CIFAR10 像素上训练时，也会使多达 95% 的神经元失活，而训练准确率和验证准确率仅受到轻微影响。对于该网络，SGDM 和随机梯度下降同样不会使任何神经元失活。
我们记录了训练过程中神经元权重的梯度，以确认即使神经元处于死亡状态，仍会收到梯度更新，如图 16a 所示。这张图起初令人困惑，因为尽管 Adam 和 RMSProp 在图 15 中使最多的神经元失活，但在图 16a 中，持续更新死亡神经元的却是 Adam 和 SGDM（Kingma 和 Ba，2015；Geoffrey Hinton，2012）。观察图 16b 中 GABA 开关切换后应用于单个神经元的每一次梯度更新，可以得到不同的视角。此处，RMSProp 和 Adam 产生的方差最高，梯度幅值也最大（以其 L^2 范数衡量）。这些梯度尖峰尤其会出现在神经元死亡一段时间后再次被激活的情况下。

为了进一步研究，我们在一个玩具环境中实现了每种优化器，跟踪一个接收稀疏梯度更新的单一权重，以观察移动平均的响应。[15] 我们得以重现 Adam 和 RMSProp 在静默一段时间后收到前几次

[15]关于这个玩具实验的更多细节，以及如何复现该实验，可在我们 GitHub 仓库 notebooks/ 目录下名为 StaleGradients.ipynb 的 Jupyter Notebook 中找到。



(a) (b)

图 16：**训练期间观察到的经验性陈旧动量。(a)** 我们选取一个批次中所有死亡神经元，并记录它们的权重在更新过程中经历的平均 L^2 幅度。Adam（青绿色）对死亡神经元的更新幅度最大。请注意，这些幅度仍比活跃神经元的平均幅度小 5–10 倍。SGDM（紫色）也会更新死亡神经元，但程度较轻。随机梯度下降（黄色）和 RMSProp（橙色）未显示，因为它们不会对死亡神经元进行任何更新。我们将 x 轴的起点设为训练步 300k，因为此时死亡神经元的数量已足够多，可以有意义地计算其梯度更新。**(b)** 我们追踪训练过程中应用于十个随机神经元的所有梯度更新，并选取一个具有代表性的神经元在此展示。我们从 GABA 开关在训练轮次约为 100 时发生的时刻开始追踪梯度更新，此后的所有训练步均显示在 x 轴上。RMSProp（橙色）的梯度更新方差和幅度最大，其次是 Adam（青绿色）。

梯度更新时出现的幅度大得多的梯度尖峰。这些结果如图 17a 所示，其中展示了每种优化器的 $\Delta_{\text{Optimizer}}$ 项，与学习率无关。我们将式 14 中各自的项初始化为零，先引入两次梯度更新，然后在注入梯度更新前保持一段静默期。

图 17a 展示了四种不同的梯度注入周期，并表明 RMSProp 的响应最大，其次是 Adam，然后是 SGDM，重现了图 16b 中观察到的梯度尖峰。图 17b 通过仅绘制两个梯度注入周期下 Adam 和 RMSProp 的分子与分母，解释了这些较大的梯度。特别要注意，对于 Adam，其分子 m_t（深橙色）下降得相当快，衰减为 $\beta = 0.9$；然而，其分母 $\sqrt{v_t}$（浅橙色）的衰减慢得多，为 $\beta_2 = 0.999$，因此会长时间保持较大，导致新的梯度输入幅度激增。

最后，图 17c 量化了当我们改变梯度注入之间的时间间隔时，这三种优化器的梯度更新幅度与实际梯度之间的差距有多大。

总而言之，我们认为，与更新当前已死亡的神经元相比，静默期后发生的梯度爆炸更可能是 Adam 和 RMSProp 像图 15 所示那样导致神经元死亡的原因。这些优化器中的分母（见公式 16 和 17）难辞其咎，因为它们是变化极其缓慢的移动平均，会放大稀疏梯度。

SGDM、Adam 和 RMSProp 的更新规则如下，其中 λ 表示学习率，θ_t 表示时间点 t的参数，g_t 是梯度，而 $\gamma = 0.9$ $\beta_1 = 0.9$、$\beta_2 = 0.999$、$\alpha = 0.99$是各算法的超参数，其取值为标准值。

$$\theta_t = \theta_{t-1} - \lambda \Delta_{\text{Optimizer}} \tag{14}$$

$$\Delta_{\text{SGD}} = g_t$$

$$\Delta_{\text{SGDM}} = \gamma \theta_{t-1} + g_t \tag{15}$$

$$\begin{aligned} \Delta_{\text{Adam}} &= \frac{\hat{m}_t}{\sqrt{\hat{v}_t}} \\ \hat{m}_t &= \frac{m_t}{1-\beta_1} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2} \\ m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{16}$$

$$\begin{aligned} \Delta_{\text{RMSProp}} &= \frac{g_t}{\sqrt{v_t}} \\ v_t &= \alpha v_{t-1} + (1-\alpha) g_t^2 \end{aligned} \tag{17}$$

请注意，对于任何梯度更新 $g_t < 1$，这些结果都具有鲁棒性；否则，SGDM 的梯度尖峰会比 Adam 更大。不过，在我们的训练机制中，这种情况从未发生，因为我们的权重和输入经过了 L^2归一化，因此权重或神经元激活值都不会大于 1。如果不是这样，我们预测 SGDM 的梯度会出现足够大的跳变，从而也导致死亡神经元。

实践要点下面我们说明为何随机梯度下降是最佳方案，但需要仔细调整学习率和 GABA 开关的超参数 s。稀疏优化器（Spa）是另一种替代方案，值得在未来的工作中进一步研究。

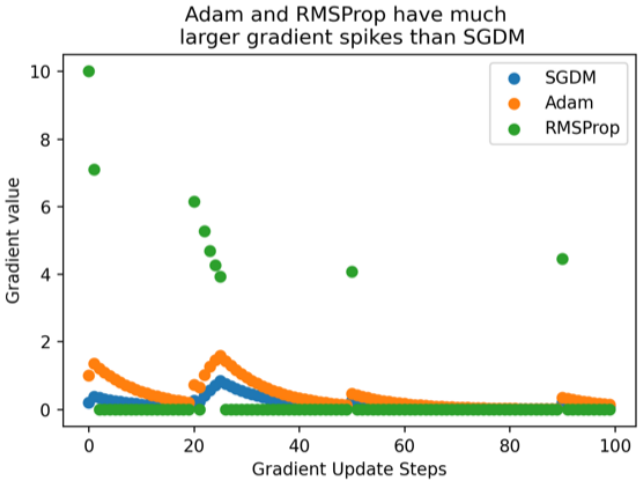
由于随机梯度下降没有动量项，因此不会受到陈旧动量的影响，是最有原则的解决方案。然而，由于缺少动量项，随机梯度下降对学习率的选择也最为敏感；学习率过高或过低，分别可能导致收敛缓慢或效果不佳。在我们的设置中，GABA 开关的兴奋期使学习率过低尤为棘手，因为神经元可能无法获得足够的梯度更新，因而无法移动到数据流形上，最终成为死亡神经元。在附录 B.3 的图 11 中，我们展示了学习率过低会导致死亡神经元：在 GABA 开关触发之前，它们始终无法学会流形。

鉴于随机梯度下降需要进行上述超参数调优，我们建议使用 SGDM 对 SDMLP 进行预训练，并在持续学习中使用随机梯度下降。如图 15 所示，采用 SGDM 预训练不会产生死亡神经元，而且对学习率的选择更具鲁棒性。例如，在这个使用 SGDM 处理 CIFAR10 像素的实例中，将学习率从 0.01 调整到 0.09，既没有影响收敛速度，也没有导致更多神经元失活。与此同时，正如表 2 的消融实验所示，在持续学习设置中，随机梯度下降是最佳选择。

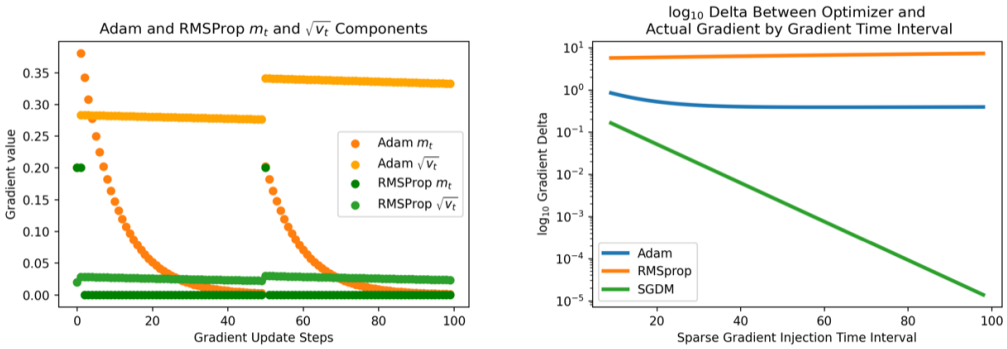
另一种选择是稀疏优化器，例如“稀疏 Adam”，它不会对任何已死亡的神经元应用梯度更新（Spa）。然而，如果不以一种新颖的方式更新动量项，理论上仍会导致我们认为更严重的问题：重新激活神经元的梯度爆炸。在 PyTorch 中使用稀疏 Adam 的唯一方法是实现稀疏权重层；我们将基于稀疏动量的优化器的实证测试留待未来工作。

本节缺少许多引文，因为我们不知道现有文献中有其他采用稀疏激活机制的模型——无论是使用 Top-K 还是专家混合（Fedus 等人，2021年；Roller 等人，2021年；Shazeer 等人，2017年）——讨论了过时动量问题以及基于动量的优化器所存在的问题。[16] 我们也未能找到关于稀疏 Adam 的学术引文

[16]有趣的是，Shazeer 等人（2017年）通过设置 $\beta_1 = 0$ 并对 v_t 取平均来修改 Adam，以减少其参数数量；这两种做法可能都无意中帮助避免了陈旧动量。



(a)



(b) (c)

图 17：**Adam、RMSProp 和 SGDM 如何引入陈旧动量的玩具实验。(a)**在四个不同时间点注入 0.2 的梯度更新，这些时间点由较大的 RMSProp 绿色圆点标出。我们先应用两次梯度更新，然后是四次，接着各一次。请注意，RMSProp 会放大梯度幅值，但在神经元不活跃时不会应用任何梯度更新。与此同时，Adam（橙色）和 SGDM（蓝色）会持续应用梯度更新，并随时间缓慢衰减。还请注意，Adam 的梯度幅值大于 SGDM 的梯度幅值。**(b)**我们在两个单独的时间点注入梯度，并观察 Adam 和 RMSProp 的分子（深色）与分母（浅色）如何变化。分母衰减得非常缓慢。**(c)** 我们观察实际梯度值 0.2 与不同优化器所应用的梯度值之间的差值，如何随距上次梯度更新的间隔而变化。请注意，y 轴为 $\log_{10}$。RMSProp 的梯度放大了 $10/0.2 = 50$ 倍，而 Adam 放大了 $1/0.2 = 5$ 倍。请注意，不含分母项的 SGDM 是唯一一种在较长时间后会停止放大梯度值的优化器。所有结果均适用于梯度值为 < 1的情况。

实现（该实现由 PyTorch 提供），也未能找到关于实现它的动机的学术引文。值得注意的是，在 TensorFlow 中，这种算法被称为“LazyAdam”，并被宣传为有助于提高计算效率，但没有提及死亡神经元。这并不令人意外，因为死神经元问题只会在稀疏区间中出现；即便如此，如果只关注验证准确率，该问题也可能不会被察觉。例如，在 CIFAR10 像素上使用 ReLU 网络和 Adam 进行训练，会导致 95% 的神经元死亡，而对验证准确率影响甚微。

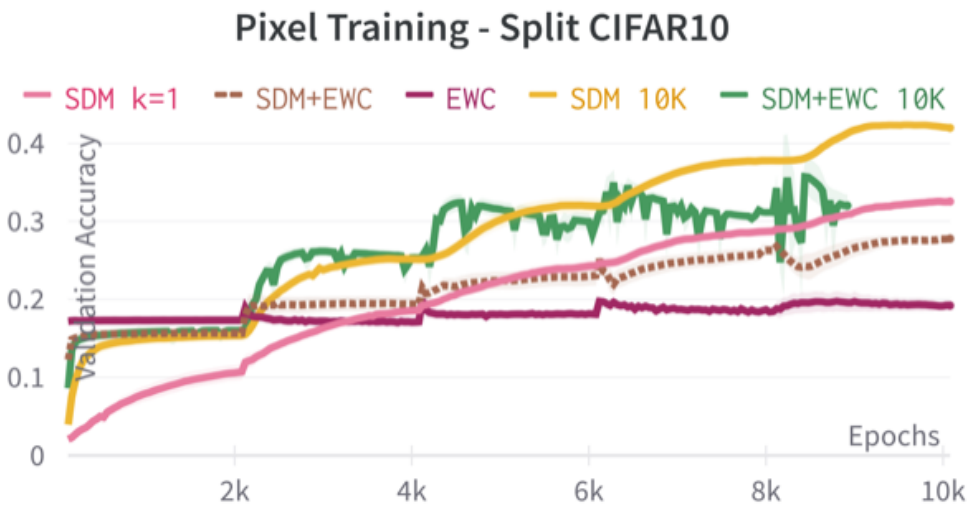


图 18：**直接在 CIFAR10 上训练时，持续学习算法的排名大致保持不变**。 这里，拥有 10K 个神经元的 SDMLP（黄色）实际上优于 SDMLP+EWC（绿色），后者是正文图 3(a) 中表现最好的模型。不过，我们没有针对 EWC 损失系数或 β 直接在像素上训练的参数进行调优。拥有 10K 个神经元的 SDMLP+EWC 未能在分配的 15 小时图形处理器运行时间内完成训练，但显然 SDM 的表现优于它。结果基于 Split CIFAR 10 的 5 个随机种子划分。

E 训练方案消融实验

E.1 直接在图像像素上训练

我们移除 ConvMixer 预处理，直接在 ImageNet32 像素上训练模型，然后在 Split CIFAR10 像素上测试其持续学习能力。此举旨在回应这样的担忧：ConvMixer 预处理改变了 CIFAR10 数据流形，使持续学习任务变得过于简单。图 18 和表 3 展示了结果。参照表 1，我们只测试表现最好的模型，并选择 EWC 而非 MAS，因为总体而言它产生的模型最佳，且其他方面的表现相近。我们没有测试 FlyModel，因为它假定输入是经过预处理的潜在嵌入，而非维度为 3072 的 CIFAR10 展平图像向量。所有训练设置和参数均与表 1 中的测试保持一致。

有趣的是，所有 EWC 方法的表现都不如之前。这很可能是因为我们没有针对直接在像素上训练调整损失系数或 β 参数。因此，表现最好的模型现在是拥有 10K 个神经元的 SDM（黄色），而不是 SDMLP+EWC（绿色）。还请注意，尽管带有 $k = 1$ 的 SDMLP（粉色）表现更差，但它未能在 2,000 个训练轮次内学会每个任务。例如，在 2,000 个训练轮次结束时，其验证准确率为 $\sim$10%，而其他方法在任务内达到的最高准确率为 $\sim$20%。这意味着拥有 $k = 1$ 的 SDM 若经过更多训练轮次或采用更高的学习率，表现可能会更好。

我们使用相同的 5 个随机种子，并为每个任务训练 2,000 个训练轮次。由于图像尺寸为 3x32x32=3072像素，而非 256 个 ConvMixer 嵌入，训练耗时更长。最复杂的模型 SDMLP+EWC（含 10K 个神经元）未能在分配给每次运行的 15 小时图形处理器时间内完成训练，但运行至最后一个学习任务，因此可以推断其性能。

表 3：像素训练 - 拆分 CIFAR10 验证准确率

方法	神经元	k	验证准确率
SDMLP	1K	1	0.33
SDMLP	10K	10	**0.42**
EWC	1K	不适用	0.19
SDMLP+EWC	1K	10	0.28
SDMLP+EWC*	10K	10	0.32
预言机	1K	NA	0.53
预言机	10K	NA	0.53

最具竞争力的模型在 ImageNet32 上预训练，并在不使用任何 ConvMixer 嵌入的情况下于拆分 CIFAR10 上进行测试。SDMLP+EWC（含 10K 个神经元）旁的 * 表示其未完成训练。结果为 5 次随机任务划分的平均值。

E.2 不进行预训练

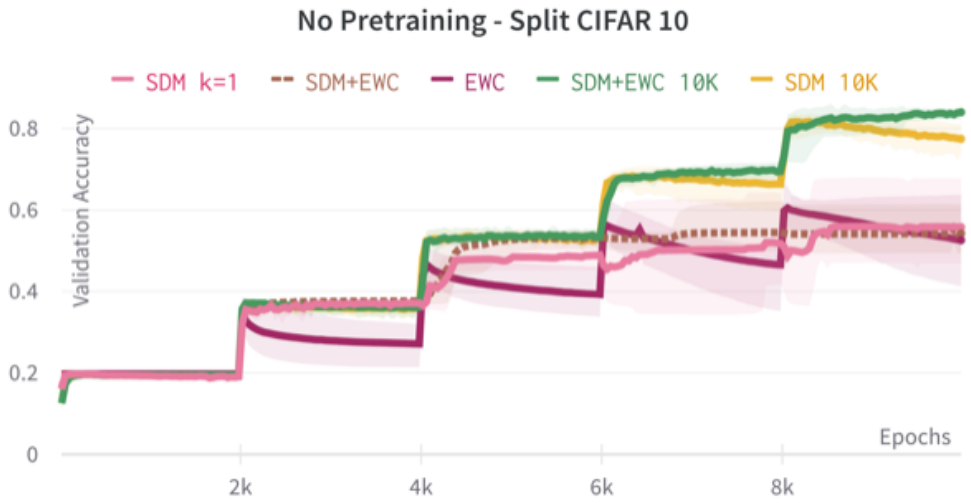


图 19：**不使用 ImageNet32 预训练时，持续学习算法的排名顺序保持不变**。实线表示不同的算法，虚线表示某个算法的变体。拥有 10K 个神经元的 SDMLP+EWC（绿色）表现最佳，其次是拥有 10K 个神经元的 SDMLP（黄色）。所有拥有 1K 个神经元的算法表现大致相同，包括 EWC（紫色）、SDMLP+EWC（棕色虚线）和 SDMLP k=1（粉色）。结果基于 Split CIFAR 10 的 5 个随机种子划分。TopK 和 ReLU 基准表现都较差，因此未使用全部随机种子运行，也未在结果中展示。

图 19 展示了最具竞争力的持续学习算法在未经 ImageNet32 预训练、并在 ConvMixer 嵌入的 Split CIFAR10 上进行测试时的结果。最终验证结果见表 4。

我们在每个任务上训练 2,000 个训练轮次，总计训练 10,000 个训练轮次。GABA 开关在第一个任务尚未完成时便已触发，此时 $s = 5,000,000$，GABA 在第 ~ 50 个训练轮次切换，而 Top-K 到第 ~ 100 个训练轮次时已完全实现。其性能低于使用 ImageNet32 预训练的情况，例如，表现最好的 SDMLP+EWC 达到 84%，而表 1 中为 86%。最值得注意的是，缺少足够容量的 1K 神经元模型表现更差，无法在没有预训练帮助的情况下学习新任务的数据流形。

表 4：无预训练——嵌入式拆分 CIFAR10 验证准确率

方法	神经元	k	验证准确率
SDMLP	1K	1	0.56
SDMLP	10K	10	0.77
FlyModel	1K	32	0.69
FlyModel	10K	32	0.82
EWC	1K	不适用	0.52
SDMLP+EWC	1K	10	0.54
SDMLP+EWC	10K	10	**0.84**
预言机	1K	NA	0.93
预言机	10K	NA	0.93

F 额外数据集

F.1 CIFAR10 附图和表格

表 5 展示了 1K 神经元设置下拆分 CIFAR10 的结果。图 20 展示了与基线 ReLU 模型相比，SDMLP 忘记先前任务的程度。该图为图 3 中的持续学习核心结果提供了更多背景信息。

表 5：拆分 CIFAR10 - 1K 个神经元 - 验证准确率

方法	神经元	k	验证准确率
SDMLP	1K	1	0.70
SDMLP	1K	10	0.63
Top-K	1K	10	0.29
FlyModel	1K	64	0.70
MAS	1K	不适用	0.69
EWC	1K	不适用	0.67
SI*	1K	NA	0.34
NISPA	∼1K	NA	0.19
L2	1K	NA	0.23
Dropout	1K	NA	0.21
SDMLP+MAS	1K	10	0.83
SDMLP+EWC	1K	10	**0.83**
预言机	1K	NA	0.93

拆分 CIFAR10 最终验证准确率 - 我们突出显示了 1K 神经元设置下表现最佳的 SDMLP、基线和**整体表现最佳者**。预言机使用完整的 CIFAR10 数据集进行训练。SI 后带有 *，表示其部分运行失败。NISPA 使用一个包含三个隐藏层、每层 400 个隐单元的网络，但其参数数量与一个包含 1K 个神经元的隐藏层相近。所有结果均为 5 个随机任务划分的平均值。

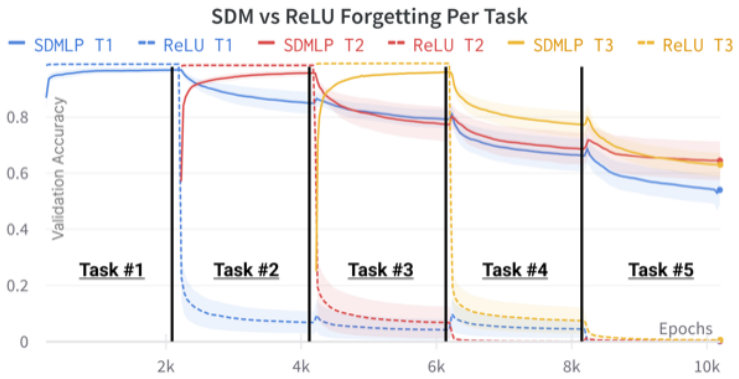


图 20：**SDMLP 会逐渐遗忘先前任务**。顶部的实线显示了 SDMLP 随时间推移对先前任务的遗忘情况，与会发生灾难性遗忘的 ReLU（底部的虚线）形成对比。现在，验证准确率是在各任务内计算的；为避免图表过于杂乱，我们只展示前三个任务的学习曲线。两张图均采用 5 个随机种子的平均值，误差条表示均值标准误。

F.2 CIFAR100

为了更准确地评估模型的实际记忆容量，我们使用相同的 ImageNet32 预训练，然后在 CIFAR100 的 50 个划分上测试持续学习，结果如图 21 和表 6 所示。我们允许

每个任务训练 500 轮次，总计训练 25,000 轮次。我们使用了三个随机种子（而非五个），采用为 CIFAR10 找到的相同超参数，并且只测试了表 1 中表现最佳的模型。

在 CIFAR100 上，包含 FlyModel 和 SDMLP+EWC 的 SDM模型在性能上明显优于正则化基线方法。在 1K 个神经元的设置下，SDMLP+EWC 表现最佳，准确率为 42%。与此同时，在 10K 设置下，FlyModel 以 58% 的准确率领先，SDMLP+EWC 则以 51% 紧随其后，而预言机的准确率为 72%。次优的基线方法是 MAS，准确率为 40%。在 CIFAR10 和 CIFAR100 上，FlyModel 从 1K 个神经元增至 10K 个神经元后性能大幅提升，这凸显了维度对于执行随机投影的固定神经元地址的重要性。有趣的是，观察各模型在任务1上的验证准确率（这反映了模型随时间推移对该任务的遗忘程度）可以发现，SDMLP+EWC 的表现优于 FlyModel；然而，FlyModel 对后续任务的保留效果更好。

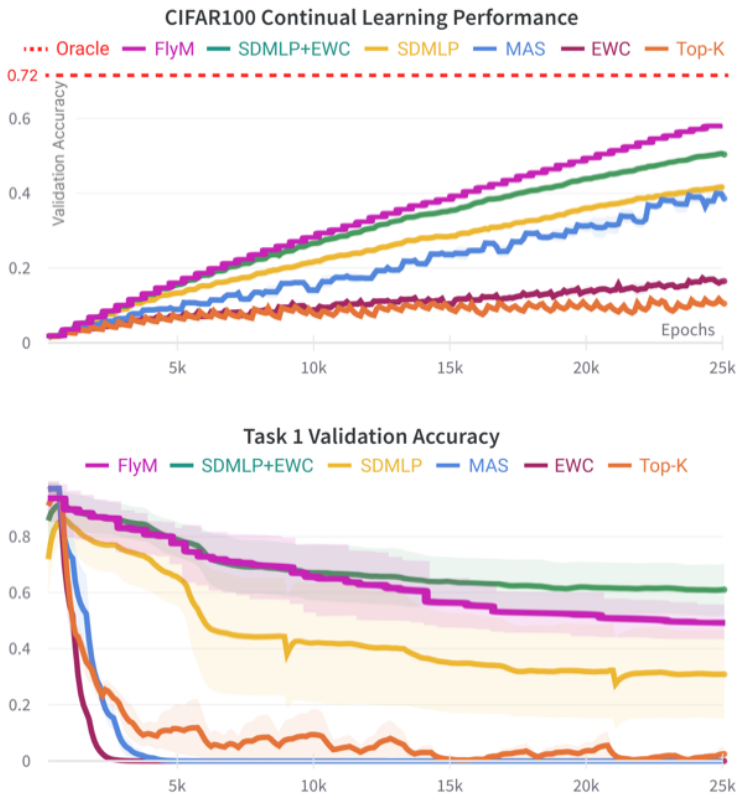


图21：**上图——对于 CIFAR100，持续学习算法的排名顺序依然相似**。带神经元的 FlyModel（洋红色）现在优于 SDMLP+EWC（绿色）。SDM 算法表现次佳（黄色），MAS（蓝色）紧随其后，而 EWC（紫色）和 Top-K（橙色）的表现较差。**下图——SDM方法能够有效抵御遗忘**。图中展示了每种方法在另外 49 个任务的整个训练过程中，CIFAR100 第一个任务及其在该任务上的验证准确率。SDMLP+EWC 最能抵御遗忘（绿色），其次是 FlyModel（洋红色）和 SDMLP（黄色）。有趣的是，对于第一个任务，SDMLP+EWC 的遗忘程度比 FlyModel 更低。Top-K 保留了一些记忆，但数量不多；而 MAS 和 EWC 都会随着时间推移，在 ~ 5 个任务（2,500 个训练轮次）中发生灾难性遗忘。

表 6：拆分 CIFAR100 验证准确率

方法	神经元	k	验证准确率
SDMLP	1K	1	0.32
SDMLP	1K	10	0.39
Top-K	1K	10	0.11
FlyModel	1K	32	0.36
MAS	1K	NA	0.24
EWC	1K	不适用	0.12
SDMLP+EWC	1K	10	**0.42**

方法	神经元	k	验证准确率
SDMLP	10K	10	0.43
FlyModel	10K	32	**0.58**
MAS	10K	NA	0.40
EWC	10K	NA	0.16
SDMLP+EWC	10K	10	0.51
预言机	10K	NA	0.72

我们将 1K 和 10K 神经元设置中表现最佳的模型加粗显示。预言机使用完整的 CIFAR10 数据集进行训练。结果为 3 次随机任务划分的平均值。

F.3 拆分 MNIST

为确保内容完整，并为其他方法提供更简单的基准，我们使用拆分 MNIST 数据集，同时保留类别增量设置。结果见表 7，大致反映了各方法的总体性能排名：拥有 10K 个神经元的 FlyModel 表现最佳，SDMLP+EWC 位居第二，其次是单独使用的 SDMLP，然后是参数重要性正则化方法。我们对 EWC 和 MAS 的正则化系数及 β 参数，以及 FlyModel 的参数，进行了大量超参数调优（附录 G.2）。

我们不使用任何预训练，而是直接在 MNIST 像素上进行训练。每个任务训练 500 轮次，也就是说总共训练 2,500 轮次。我们仅使用三个随机种子来初始化模型权重，但始终采用朴素划分：任务1包含数字 0 和 1，任务2包含数字 2 和 3，以此类推。

作为我们的基准之一，我们使用了主动树突（Iyer 等人，2022）。我们使用了其提供的代码，但发现它无法泛化到论文中采用的更简单的置换 MNIST 任务增量基准之外。

NISPA 在表 7 的神经元数量列中标有 *，因为它按照原始实现使用了三个隐藏层，每层 400 个神经元。这与单层中的 1K 个神经元所对应的参数数量大致相同（忽略偏置项时分别为 637600 和 794000）。[17]

表 7：拆分 MNIST 验证准确率

方法	神经元	k	验证准确率
SDMLP	1K	1	0.69
SDMLP	1K	10	0.53
SDMLP	10K	10	0.53
FlyModel	1K	64	0.77
FlyModel	10K	32	**0.91**
EWC	1K	不适用	0.61
EWC	10K	不适用	0.67
MAS	1K	不适用	0.49
MAS	10K	NA	0.58
SI	1K	NA	0.36
Top-K	1K	10	0.22
NISPA	*	NA	0.40
主动树突	1K	NA	0.20
ReLU	1K	NA	0.21
SDMLP+EWC	1K	10	0.83
SDMLP+EWC	10K	10	0.86
预言机	1K	NA	0.98
预言机	10K	NA	0.99

直接在 MNIST 像素上训练且未进行任何预训练的模型。我们将表现最佳的方法加粗。我们使用三个随机种子运行这些实验，并针对每个划分训练 500 轮（这远高于其他基线方法（Hsu 等，2018））。

有趣的是，包含 1K 个神经元的 SDMLP 表现优于包含 10K 个神经元的 SDMLP。我们还发现，在训练结束时，SDMMLP+EWC 模型仍在学习最后一个任务。然而，增加训练轮数并未使最后一个任务学得更好，这表明模型可能已没有新的神经元可用于避免覆盖旧知识，或者在与 SDM 结合时应降低 EWC 的正则化系数。[18]

[17]此处 NISPA 的权重稀疏性为 80%，但这有助于其持续学习；在统计参数数量时，我们也没有考虑 Top-K 模型的激活稀疏性。此外，由于权重需要迭代增长和剪枝，训练时还需要执行更多浮点运算次数。[18]在附录 F.4 的 FashionMNIST 结果中也观察到了这一现象。

F.4 拆分 FashionMNIST

我们还在 FashionMNIST 数据集上评估了表现最好的模型和基线方法。该数据集包含十个不同类别的服饰项目，以灰度 28x28 图像呈现。与 MNIST 相比，这项任务稍难一些：我们的预言机在该任务上的准确率为 90%，而在 MNIST 上为 99%。表 8 展示了 3 个随机种子对应结果的平均值，这些随机种子用于初始化模型权重。结果的排名顺序与拆分 MNIST 的表 7 一致。

表 8：拆分 FashionMNIST 验证准确率

方法	神经元	k	验证准确率
SDMLP	1K	1	0.73
SDMLP	1K	10	0.53
SDMLP	10K	10	0.52
FlyModel	1K	64	0.67
FlyModel	10K	64	**0.76**
EWC	1K	不适用	0.68
EWC	10K	不适用	0.72
MAS	1K	NA	0.33
MAS	10K	NA	0.34
Top-K	1K	10	0.23
ReLU	1K	NA	0.21
SDMLP+EWC	1K	10	0.74
SDMLP+EWC	10K	10	0.72
预言机	1K	NA	0.90
预言机	10K	NA	0.90

直接使用 FashionMNIST 像素训练且未进行任何预训练的模型。我们将表现最佳的方法加粗。

G 基线实现

G.1 弹性权重巩固和突触智能的 Beta 系数

这篇论文（Hsu 等，2018）考察了多种基线持续学习算法，并采用了不同的持续学习设置。其中包括我们所使用的最贴近现实的类别增量设置。论文还开源了这些不同算法的实现，并发现 EWC、MAS 和突触智能等正则化方法（Kirkpatrick 等，2017；Aljundi 等，2018；Zenke 等，2017）都会在拆分 MNIST 上发生灾难性遗忘。

然而，在我们的实验中，通过更仔细地调整损失函数中正则化项的损失系数，我们能够提升性能。调整 EWC 和 SI 的损失系数并未提升它们的性能。但当我们考察 EWC 和 SI 的学习动态时，发现由于它们在每个任务划分上都达到了 100% 的准确率，因此损失中没有可用来推断各权重对于未来任务的重要性的梯度。为了向模型提供梯度信息，我们修改了交叉熵损失，引入一个$\beta < 1$ 系数，使模型对其预测不那么确信。形式上：

$$p_i = \frac{\exp\left(\beta l_i\right)}{\sum_{i=1}^{o} \exp\left(\beta l_i\right)}, \tag{18}$$

其中，$l_i \in \mathbb{R}$ 是每个 o 输出类别对应的实值输出（logits），由 i索引。经过 softmax 分布后，会得到一个概率分布，其中 $\sum_{i=1}^{o} p_i$ =1。在 (Hsu 等，2018) 的原始算法中，$\beta = 1$ 并非需要调整的超参数，而且该 $\beta = 1$ 足够大，因此模型对正确类别的输出概率为 ~ 1，其他类别的概率均为 0，导致损失为 0，且没有梯度。通过设置 $\beta < 1$，我们迫使模型降低对正确类别预测的确信程度，从而产生损失和梯度信息，以推断哪些参数很重要。请注意，此 β 项仅在推断参数重要性时使用，而不用于训练模型。通过结合正则化损失系数调整β值，我们得以避免灾难性遗忘，并再次超越 (Hsu 等，2018) 中报告的 EWC 和 SI 基线。虽然这又是一个必须调整的超参数，且我们所知的原始算法 (Kirkpatrick 等，2017；Zenke 等，2017) 或其他文献均未提及这一修改，但这是一个简单直接的修改，能够提升性能，同时保留算法的核心。因此，我们认为，这是一项恰当的修改，可以建立更有意义的基线。

表 9：拆分 MNIST 上改进的正则化基线

Name	损失系数	β	新验证准确率	原始
EWC（Kirkpatrick 等，2017）	200	0.005	63.23	19.80
突触智能（Zenke 等，2017）	1500	0.005	35.76	19.67
MAS（Aljundi 等，2018）	0.5	NA	24.99	19.52
L2（Goodfellow 等，2014）	10	NA	36.77	22.52

在拆分 MNIST 上测试一个包含 1,000 个神经元的单隐藏层多层感知机，每个拆分训练 10 个训练轮次；手动尝试若干不同的超参数后，得到的最终验证准确率如表 9 所示。[19]我们展示了所用的超参数，并列出 Hsu 等（2018）中表 2 的类别增量设置结果以供比较。

请注意，这里的训练轮次数少于附录 F.3 中对完整拆分 MNIST 的分析。这是为了将我们的结果与其他每个任务仅使用少量训练轮次的基线方法进行比较（Hsu 等，2018）。比较表 9 与 F.3 中的表 7，就会发现，增加训练轮次确实会影响 EWC 和 MAS 的性能。

[19]对于表 1 所示的预训练设置下使用拆分 CIFAR10 进行的完整实验，我们进行了更广泛的贝叶斯超参数搜索，以最大限度地提升每个基线方法的性能。

G.2 FlyModel 参数

FlyModel 的独特之处在于，它只训练一个训练轮次，且不使用反向传播。作者提出了一种方法，可以让模型在每个任务上训练多个训练轮次；然而，这会引入突触衰减，可能通过遗忘降低性能。Shen 等人（2021）从未实现这一方法，因此我们在此展示该算法在每个任务上训练一个训练轮次时的最强版本。

我们以（Shen 等，2021）中列出的参数作为起点。由于我们的 ConvMixer 嵌入维度为 256，是原论文所用 512 的一半，因此我们将肯扬细胞的数量从 20,000 减至 10,000，恰好与其他实验中考察的神经元数量相符。我们还将投射神经元连接的数量在 64 到 3 之间进行调整，发现无论在 10,000 还是 1,000 个神经元的设置下，32 的表现都更好。我们对学习率进行了实验，发现 0.005 在 MNIST 和 CIFAR10 实验中效果最佳。对于 CIFAR100，只有在使用 10,000 个肯扬细胞的模型时，学习率为 0.2 才能取得更好的表现。文中呈现的是各次运行在不同随机种子下取得的最佳结果。

H 探究持续学习能力的差异

我们探究了为何 SDM 能抵御灾难性遗忘，而基线模型以及消融特定特征的 SDMLP 模型却都以失败告终。

总结我们的发现：

- ReLU——神经元从未学会铺满数据流形，而且几乎每个输入都会激活每个神经元。
- Top-K——缺乏 L^2 归一化且使用偏置项，意味着神经元不会铺满数据流形。这意味着它们不仅会被更多任务激活，还会遗忘先前的任务。此外，死亡神经元更多，导致记忆容量下降。
- SDM Top-K 掩码——掩蔽会导致更多死亡神经元，而存活的神经元则具有更强的多义性，因此无法形成能够避免被未来任务覆盖的子网络。
- SDM 无 L^2 范数——少数神经元的权重范数极大，且在多达 50% 的所有输入中处于激活状态。这些神经元在所有任务中都处于激活状态，导致记忆覆盖和灾难性遗忘。

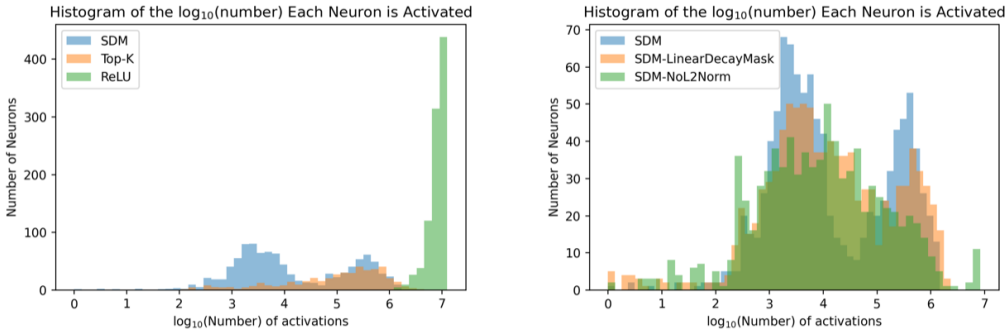
在 Split CIFAR10 持续学习过程中，我们每隔 100 个训练轮次就保存一次模型检查点；每个数据划分训练 300 个训练轮次，总计 1,500 个训练轮次。我们还会追踪每个神经元在整个训练过程中处于激活状态的次数，并将其可视化于图 22 所示的对数坐标图中[10]。

图 22a 对比了 SDM、Top-K 和 ReLU，可以明显看出，在没有 Top-K 激活函数的情况下，所有 ReLU 神经元（绿色）都会被多次激活。虽然 Top-K 的激活次数较少（橙色），但其神经元不像 SDM（蓝色）那样呈现双峰分布；我们认为，这种分布对应于仅针对特定任务激活的独特子网络。

图 22b 展示了 SDMLP 的消融实验：这些实验采用 Top-K 掩码而非减法，且不进行 L^2 归一化。请注意，无L^2 范数（绿色）对应的神经元中有一小簇，这些就是在所有任务中始终位于 Top-K 中的贪婪神经元。另请注意，Top-K 掩码（橙色）略微偏向较大的激活值。

在以下所有分析中，我们都不展示死亡神经元。死亡神经元实际上只会对 Top-K 构成问题：Top-K 中有 55% 的死亡神经元，而未进行 L^2 归一化的 SDM 中则有 18%。

为了考察每个神经元针对特定任务和数据类别的专门化程度，我们取拆分 CIFAR10 持续学习后的最终模型，将所有 CIFAR10 训练数据输入这些模型，并记录每个输入对应的 Top-K 神经元。对于每个神经元，我们统计它在 10 个输入类别中各类别下被激活的次数，并据此构建该神经元所响应类别的概率分布。我们将该分布的熵作为衡量每个神经元多义性程度的指标。在图 23 中，我们绘制了每个神经元的熵与其进入 Top-K 的时间百分比之间的关系。我们还结合这两个指标，根据每个神经元进入 Top-K 的时间比例对其熵加权，从而计算激活神经元的平均熵，并



(a) (b)

图 22：**SDM 神经元比 Top-K 或 ReLU 更具专门性**。我们统计每个神经元在持续学习过程中被激活的次数，并以直方图呈现，纵轴采用对数10 刻度。**（a）**我们将 SDM（蓝色）与 Top-K（橙色）和 ReLU（绿色）基线方法进行比较。**（a）**我们将 SDM（蓝色）与两种消融实验进行比较：采用 Top-K 掩码的 SDM（橙色）和未进行 L^2归一化的 SDM（绿色）。

在表 10 中列出这一结果，并同时列出各方法的持续学习性能。激活神经元的平均熵与持续学习之间存在明显的负相关。[20]

表 10：按 Top-K 出现情况加权的平均神经元熵

Name	加权平均熵	验证准确率
SDMLP	0.99	0.54
SDMLP 线性掩码	1.36	0.35
SDMLP 无 L^2 归一化	1.96	0.20
Top-K	1.48	0.29
ReLU	2.25	0.21

图 23 和表 10 中的总结有效地展示了 SDM 如何通过子网络形成实现强大的持续学习性能。SDM（图 23a）的激活神经元平均熵最低，其次是采用 Top-K 掩码的 SDM（图 23b）。采用 Top-K 掩码的 SDM（图 23b）显示，最具多义性（熵最高）的神经元也是最活跃的，这会导致跨任务遗忘。图 23c 显著展示了右上方的 13 个“贪婪”神经元：它们不仅具有高度多义性，而且会对许多输入保持活跃（与 SDM 的 1.4% 相比，其 y 轴最高达到 6%）。图 23d 中的 Top-K 看起来与采用 Top-K 掩码的 SDM 相似，但每个神经元的平均多义性要高得多。最后，图 **??** 中的 ReLU 网络虽然神经元激活值分布非常均匀，y 轴范围也最低，仅为 0.16%，但所有神经元都具有高度多义性。

值得注意的是，图 23a 中的 SDM 并非每个神经元都能以相同的频率被激活并进入 Top-K，这意味着其神经元并未按照数据密度完美地铺满数据流形。不过，这可能是因为我们优化网络的目标是分类性能，而非重构损失。

[20]请注意，在这些实验中，我们使用的是 SGDM 优化器，而非随机梯度下降。这意味着 SDM 和线性掩码的验证准确率低于原本的水平。不过，我们认为由此得出的见解不受影响，而使用随机梯度下降只会减少死亡神经元的数量。

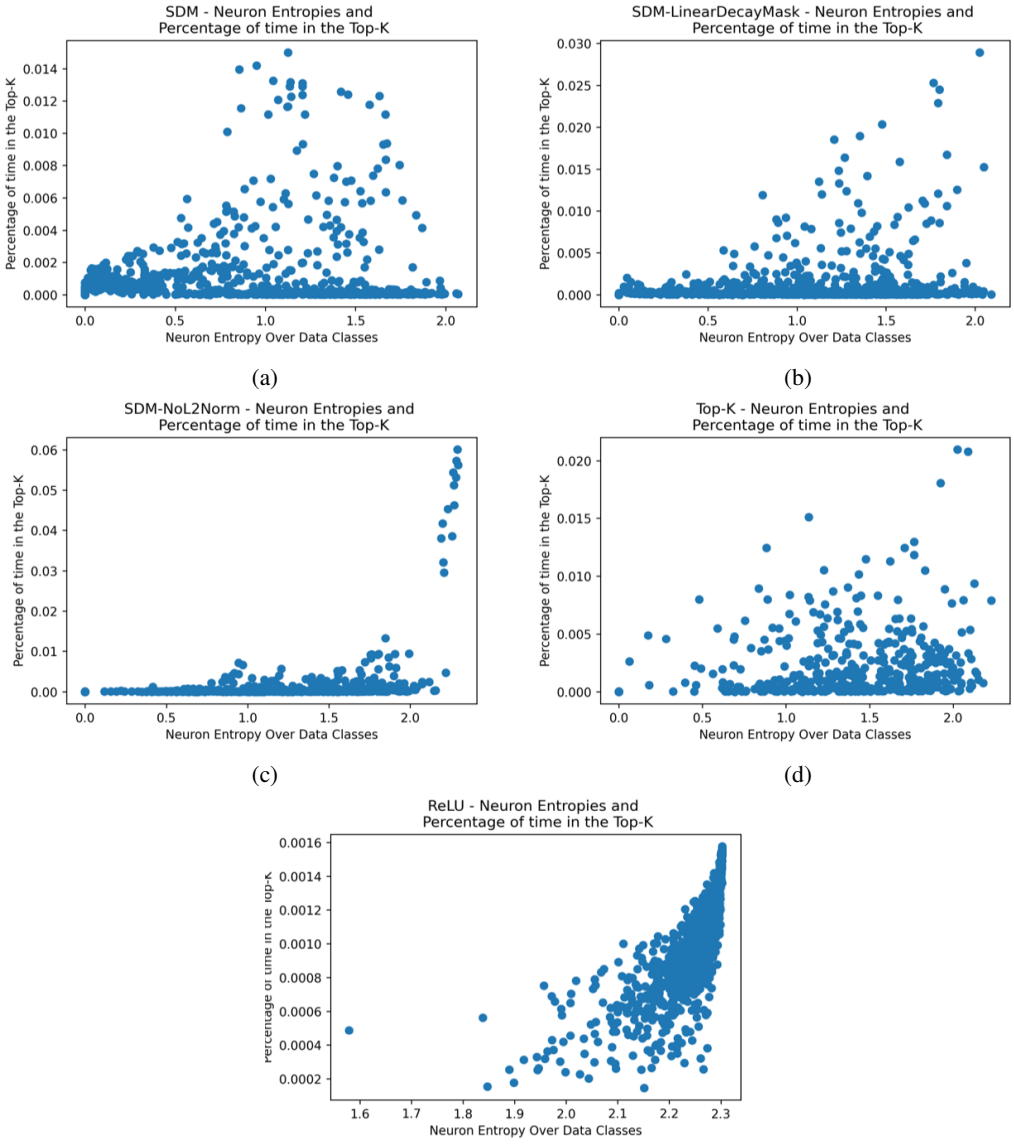


图 23：**SDM 神经元具有专门化特性，并以民主方式参与学习**。绘制每个神经元的熵（该神经元激活的数据类别分布）与其对所有输入处于 Top-K 中的时间百分比之间的关系。

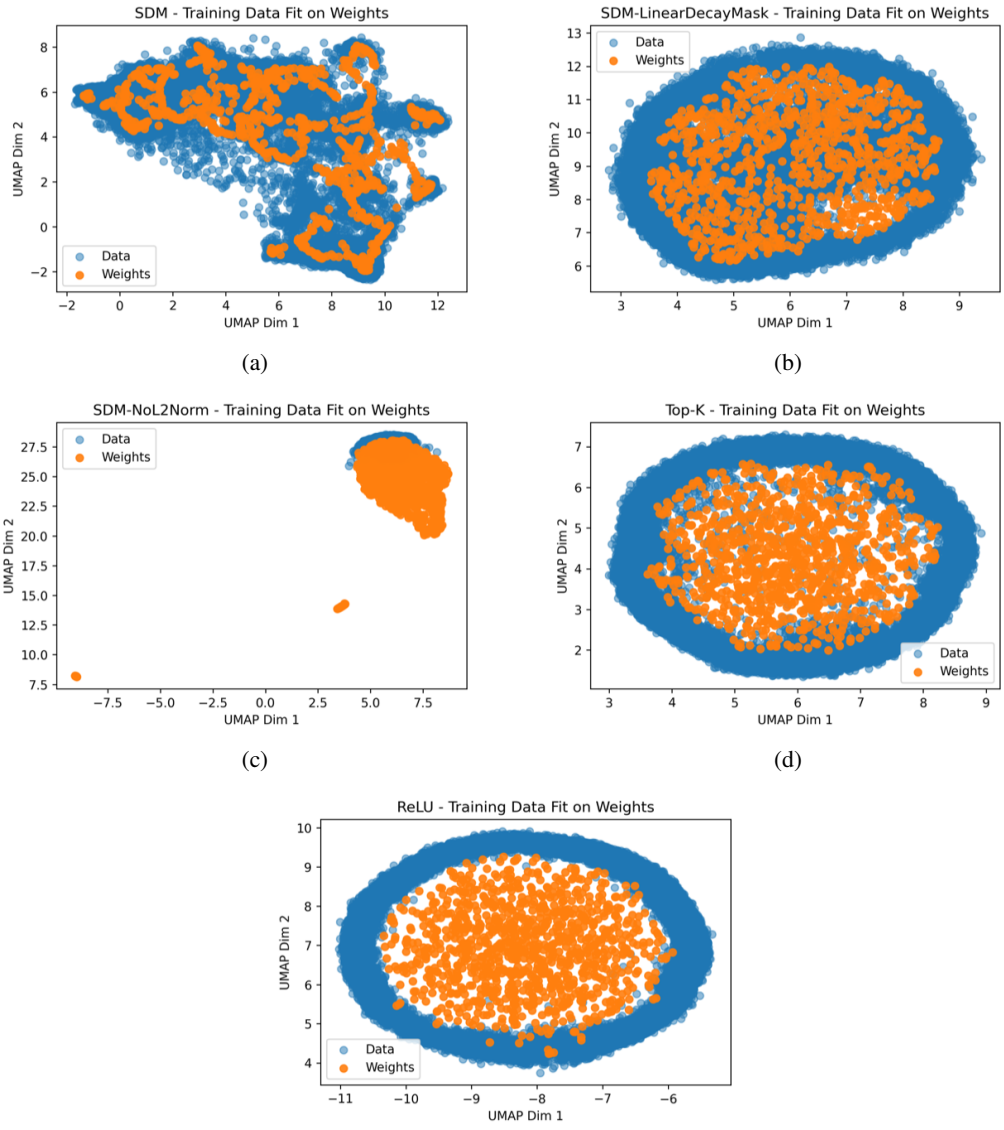
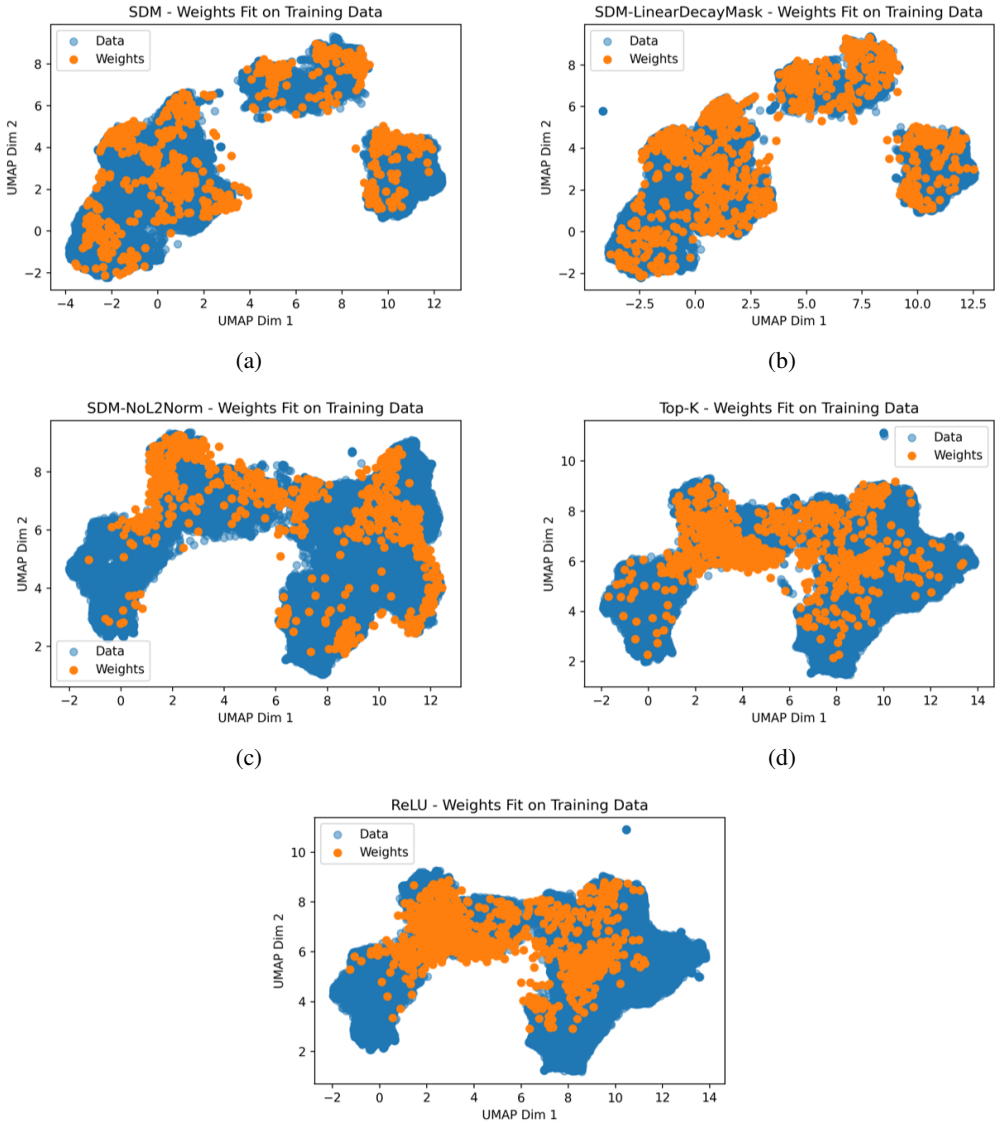


图 24：**将 CIFAR10 投影到 SDM 权重的 UMAP 嵌入上，能够最有效地铺满流形。(a)** 如正文图 6 所示，这是唯一能够铺满流形并使 UMAP 图呈现有意义结构的方法。



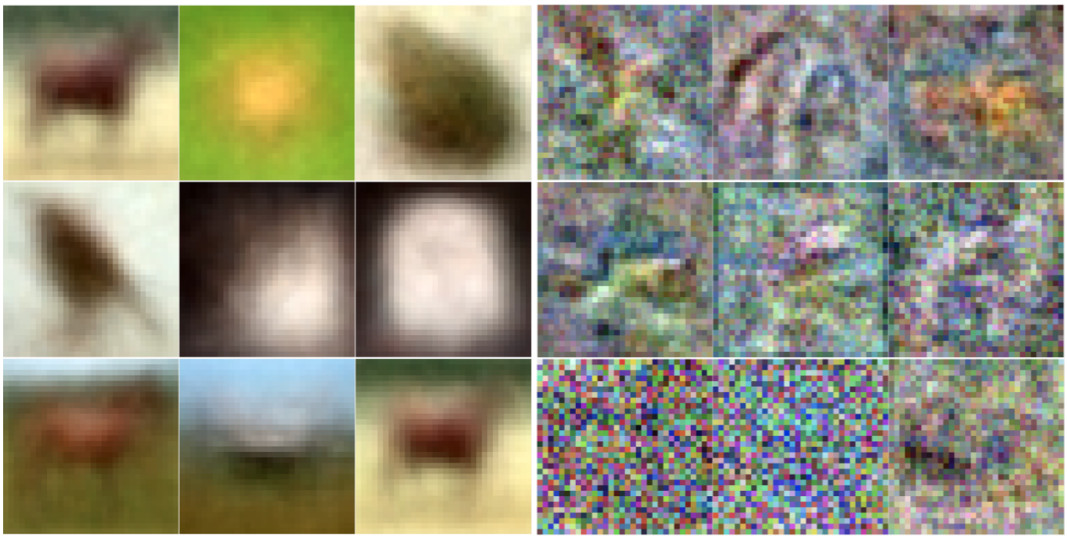
(a) (b) (c) (d) (e)

图 25：**SDM权重对 CIFAR10 UMAP 嵌入的铺展效果最佳**。顶行的两个 SDMLP 模型（（**a**）和（**b**））对流形的铺展效果最佳。这与前面的图 24 不同，在图 24 中，只有（**a**）中的 SDM 展现出良好的铺展效果。此处顶行的 CIFAR10 数据流形看起来有所不同，形成了三个彼此分离的团块，这是因为 L^2 归一化操作。

我们展示了一些额外的图表，用于将 SDM 的流形铺砌能力与其他方法进行对比。图 24 在 ImageNet32 上预训练的模型权重上拟合 UMAP 投影，然后将该投影用于 CIFAR10 数据。图 25 则相反：它在 ConvMixer 嵌入的 CIFAR10 训练数据上拟合 UMAP（McInnes & Healy，2018）投影，再将该投影用于训练后的模型权重。两幅图都表明，SDM 学习以最有效的方式铺砌数据流形；在两幅图中，只有完整的 SDMLP 能够铺砌流形。这表明，经过预训练后，SDM 的神经元已经学会区分一般图像统计特性所形成的流形，并形成有助于持续学习的子网络。[21]

我们强调，尽管存在流形铺砌和子网络形成，但 ImageNet32 预训练并不意味着 SDM 已经认识 CIFAR10 数据，或能完美铺砌 CIFAR10 流形。图 27 展示了一个 UMAP 图，其投影拟合于直接在 CIFAR10 像素上训练的 SDMLP，并用于投影相同的 CIFAR10 数据；请注意这里的流形铺砌紧密得多。基于像素的训练也进一步证明了流形铺砌的存在：在图 26 中，我们取十个随机 SDMLP 神经元的权重，并将其重塑为 3x32x32 的维度，从而显示这些神经元已专门适应数据中的特定类别。这与旁边展示的 ReLU 神经元权重形成鲜明对比。SDM 神经元不仅专门适应特定图像类别，甚至还专门适应类别中的特定样本，这类似于假想的“祖母神经元”（Gross，2002；Quiroga 等，2005）。该图展示了从一个在 CIFAR10 像素上训练的 SDMLP 中随机选取的神经元的训练后权重，并将其重塑为图像尺寸。图中可以看到一只鸟、一只青蛙，以及多只鹿和马。

请注意，如果直接将这些 L^2 归一化权重和我们 L^2 归一化后的 CIFAR10 图像可视化，它们看起来都会是黑色的，因为表示 RGB 颜色的像素值介于 0 和 1 之间，而我们的归一化会使它们变得更小。我们先减去最小像素值 $\mathbf{x} - \min(\mathbf{x})$，再乘以 $\mathbf{x} * 1/(\max(\mathbf{x}))$，将权重还原为像素值，这是标准做法。

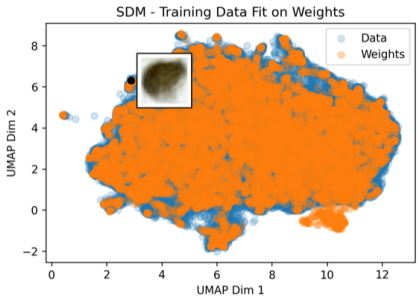


(a) (b)

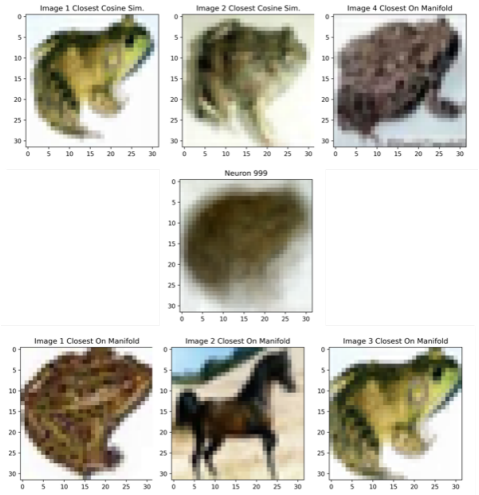
图 26：**SDM（a）与 ReLU（b）学习到的神经元权重**。我们直接在 CIFAR10 上训练模型，并将九个随机选取的神经元权重重塑为对应的图像尺寸进行可视化。SDM 使神经元的感受野更容易解释。

为了充分强调这些神经元及其可解释的感受野位于流形上有意义的部分，我们选取一个感受野对应青蛙的神经元，并展示它在流形上的位置。随后，我们扫描 CIFAR10 训练数据，选出能最大程度激活

[21]神经元各不相同且分散在数据流形上的这一特性，结合 Top-K 激活函数，确保不同输入类别会激活不同的神经元子网络。



(a)



(b)

图 27：**分析 SDM 神经元感受野与数据流形之间的关系。(a)**基于在 CIFAR10 像素上训练的神经元权重（橙色）和 CIFAR10 数据（蓝色）拟合的 UMAP 图。左上方的黑点表示产生插图所示青蛙图像的神经元权重的位置。**(b)** 中心显示神经元权重；上排为能最大程度激活该神经元的三张 CIFAR10 图像，显示神经元权重学到了一组相似青蛙图像的叠加。下排为根据 UMAP 嵌入上的欧几里得距离确定的三张最近图像。该嵌入的精确度较低，但仍显示流形在很大程度上能够捕捉相似类别，马除外。

该神经元的 3 张图像，并将它们绘制在图 27 的顶行。我们还查看了与神经元权重的 UMAP 嵌入欧几里得距离最小的图像，并展示距离最近的三张图像。这种 UMAP 方法确实包含一匹马，但由于 UMAP 嵌入会将 3072 维图像映射到 2 维，因此这种方法的精确度较低。

我们使用神经元的 UMAP 嵌入来可视化每个神经元被激活的次数，如图 28 所示。这里使用了图 22 直方图中展示的持续学习过程中的相同整体激活值，其中黄色表示激活次数最多，紫色表示最少。

最后，我们在图 29 中展示 SDM 的数据流形铺砌：在左侧图中，我们为每个 CIFAR10类别指定不同的颜色，然后展示各神经元的位置，并将最能激活该神经元的颜色分配给它。

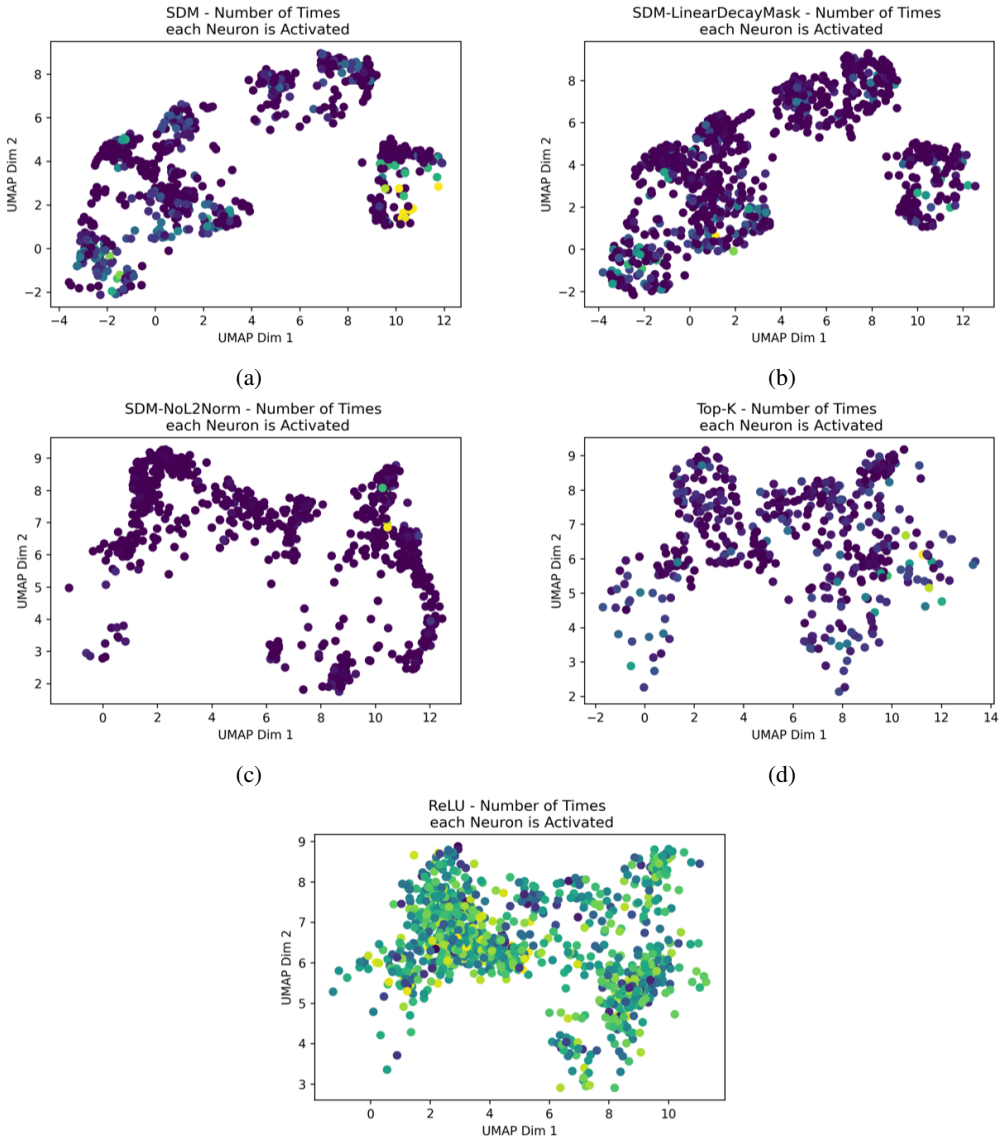
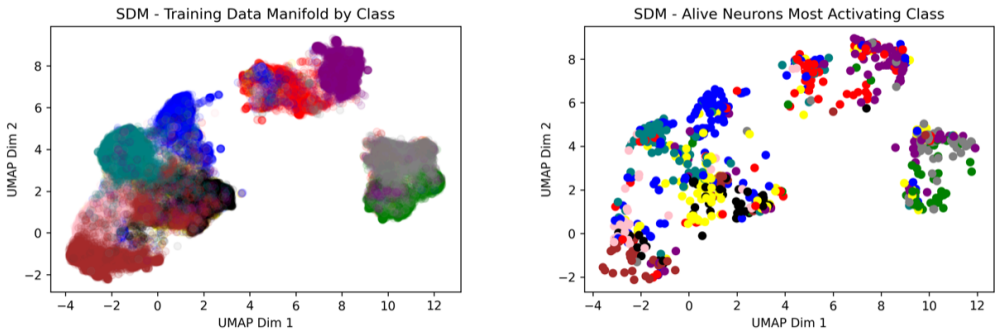


图 28：**按激活次数着色的神经元**。这幅图清楚地表明，不使用 L^2 norm 的 SDM **(c)** 只激活少数神经元来完成全部学习，导致灾难性遗忘。与此同时，ReLU**(e)** 几乎激活了所有神经元。所有使用 Top-K 的模型：SDM **(a)**、采用 Top-K 掩蔽的 SDM **(b)** 以及 Top-K **(d)** 都会激活部分神经元，因此若不进行本节介绍的额外分析，就更难区分它们的学习动态。



(a) (b)

图 29：**SDM 神经元会铺满它们响应最强的流形区域**。我们使用与图 25 相同的 UMAP 图，但在**(a)** 中按类别为数据着色，在**(b)**中则按最能激活各个神经元的类别为其着色。