arXiv:2303.04721v1 [cs.LG] 8 Mar 2023

快速偏移校正的存内训练

Malte J. Rasch [1] Fabio Carta [1] Omebayode Fagbohungbe [1] Tayfun Gokmen [1]

摘要

采用电阻式交叉阵列进行存内计算，有望以极高的效率加速深度学习工作负载。为了充分发挥存内计算的潜力，理想情况下，应同时加速大型深度神经网络（深度神经网络）的训练和推理。过去，人们提出了专门的存内训练算法，不仅能加速前向和反向传播，还能通过一些技巧在内存中并行更新权重。然而，最先进算法 Tiki-Taka 2.0（TTv2）仍需要近乎完美的偏移校正，并且容易受到潜在偏置的影响；这些偏置可能源于编程和估计误差，也可能由器件材料的长期不稳定性引起。本文提出并介绍了两种改进的存内计算算法：Chopped-TTv2（c-TTv2）和动态参考模拟梯度累积（AGAD）。这两种算法保持相同的运行时间复杂度，同时利用斩波器校正所有残余偏移。这些算法大幅放宽了对器件的要求，从而扩大了可用于此类快速存内深度神经网络训练的潜在材料范围。

1 引言

利用电阻式交叉阵列进行存内计算，有望大幅加速深度学习工作负载，因为通过电阻元件和基本物理定律（基尔霍夫定律和欧姆定律）实现普遍存在的矩阵向量乘法（MVM），即可实现高能效加速（Sebastian 等，2020；Burr 等人，2017；Haensch 等人，2019；Yang 等人，2013；Sze 等人，2017）。迄今为止，大多数原型芯片的研发工作都集中于加速预训练深度神经网络的推理阶段（Wan 等，2022；Khaddam-Aljameh 等，2021；Xue 等，2021；Fick 等，2022；Narayanan 等，2021；Ambrogio 等人，2018；Yao 等人，2020）。然而，就原始计算需求而言，训练阶段的开销通常比推理高出几个数量级，因此原则上对利用存内计算实现硬件加速的需求更大（Gokmen & Vlasov, 2016）。不过，利用非易失性存储器元件进行内存训练一直面临挑战，尤其是因为存储器件的开关行为不对称且并不理想，而且梯度更新对精度的要求高得多（讨论示例参见 Jain 等人，2019）。

为了加速深度神经网络训练中的随机梯度下降，不仅需要考虑前向传播，还需要考虑反向传播，以及梯度计算和更新。线性层的反向传播只需转置输入和输出即可轻松实现存内加速，但要在存内快速、高效地完成梯度累积和更新则更为困难。一种方法是牺牲速度和效率，在数字电路中计算梯度并进行累积，只对存内的前向传播和反向传播进行加速，正如（Nandakumar 等人，2018年；2020年）所提出的那样。另一方面，Gokmen 等人（Gokmen& Vlasov，2016 ）建议利用随机脉冲的重合，以高效并行的方式完全在存内执行外积更新操作。然而，在最初的朴素方案中，为了以所需精度计算并累积梯度（Agarwal 等人，2016；Gokmen& Vlasov，2016），需要使用对称性高、精度高的双向开关器件（就增量电导变化的可靠性而言）。不过，算法的进一步改进显著降低了对可靠电导状态数量和器件对称性的要求（Gokmen& Haensch，2020年；Gokmen，2021年）。

在最新版本中，所谓的 TTv2 学习算法（Gokmen，2021年）要求每个权重矩阵元素配备三个可调电导，即矩阵[1] $\check{A}$、 $\check{R}$和 $\check{W}$。前两个电导， $\check{A}$和 R，用于执行外积和（部分）

[1]IBM Research，位于美国纽约州约克敦高地。通讯请联系：Malte J. Rasch <malte.rasch@ibm.com>、Tayfun Gokmen <tgokmen@us.ibm.com>。

版权 2023，归作者所有。

[1]我们用 $\check{X}$ 表示编码在交叉阵列电导中的权重矩阵 X，以区别于存储在数字存储器中的矩阵。

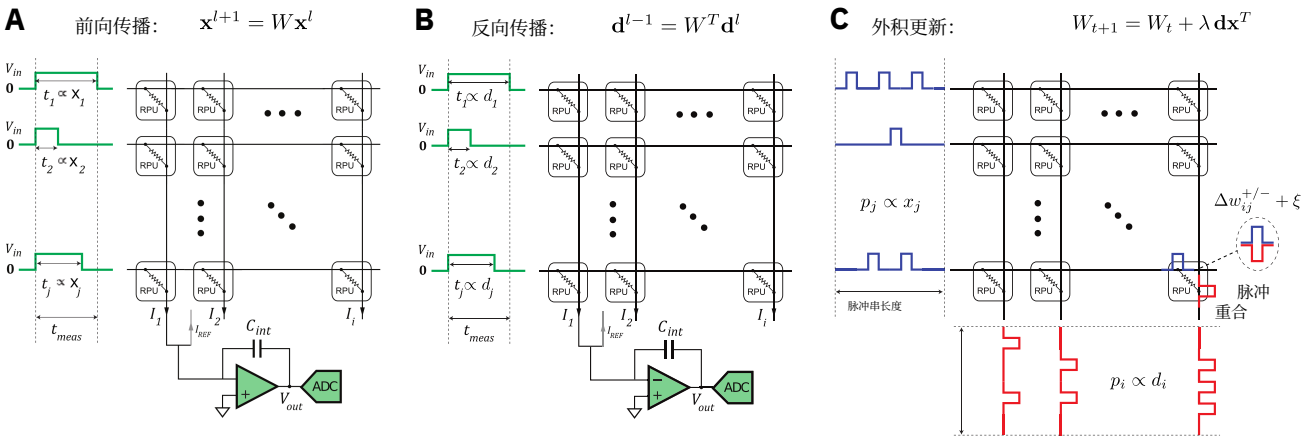


图 1. 根据（Gokmen & Vlasov，2016），用于前向和反向传播及外积更新的存内矩阵向量乘法（MVM）的物理实现。

梯度累积，而 W 则用作线性层权重 W 的表示，因此用于前向和反向传播。在功能层面，该算法与现代随机梯度下降方法相似；这些方法会引入动量项（例如 ADAM（Kingma & Ba，2014）），因为在这里，梯度也会先被计算出来，并以泄漏方式累积到一个独立矩阵中，然后再加到权重矩阵上。不过，模拟友好的 TTv2 算法会针对每一行（或列）异步计算并传输累积梯度，以提升运行时性能。此外，至关重要的是，存储器件的器件非对称性会导致近期累积梯度发生依赖于输入的衰减，这与动量项通常采用的恒定衰减率不同；后者难以在存储器中高效实现（另请参见（Gokmen，2021年；Onen 等人，2022））。

更具体地说，对于每次更新，首先利用随机脉冲序列的重合，在 A 上计算外积梯度更新。然后，以按行（或按列）的方式读取 A，并将读出的累积梯度进一步滤入数字存储（隐藏）矩阵 H，以平滑梯度计算本身以及在噪声材料上使用随机脉冲进行物理写入过程所引入的高频噪声。只有当累积梯度达到阈值时，才会向矩阵 W 中相应的权重元素发送单个写入脉冲。

虽然该 TTv2 算法通过对最近的梯度进行低通滤波，大幅改善了材料规格，但它依赖于这样一个假设：器件在其电导范围内具有预先定义且稳定的对称点（SP）（Onen 等人，2022）。SP 定义为这样一个电导值：正向更新和负向更新平均而言会使电导产生相同的净变化。由于假定器件存在不对称性，SP 对随机输入而言是一个稳定不动点，因此 A 上的累积梯度会在接近收敛时自动衰减。然而，要通过算法使其衰减至零，必须为每个器件确定 SP 对应的零值；这可以通过使用参考阵列 R去除偏置来实现。由此，参考电导 R 用于存储 A 中与之对应器件的 SP 值；访问 A时，不直接读取 A，而是读取其差值 $A - R$；训练过程中则仅更新 A。

在开始时将偏置阵列 R 编程为 A 的对称点（SP），在实践中可能存在问题，因为任何剩余偏置 $A - R + \Delta$ 都会被持续读取并累加到权重矩阵 W上。例如，对不动点的估计或将对称点写入 R 时不够准确，或者 A的对称点随时间长期变化，都可能导致这种相对于不动点的恒定偏置 Δ。此外，由时间变化 $A \neq R$ 引起的偏移瞬态 可能导致显著的梯度累积并写入 W，进而可能造成振荡和非最优学习行为。

本文提出了两种基于 TTv2 的新算法，可在存在任意类型的参考不稳定性或残余偏置时改善梯度计算。这两种算法都引入了一种借鉴自放大器电路设计的技术，称为斩波器（Enz 和 Temes，1996年）。斩波器是一种广为人知的技术，通过随机改变信号的符号（即斩波），消除累积系统造成但信号中不存在的偏置或残差；从累加器读取数据时，再对信号进行校正。在第一种算法中，我们在内存中计算外积更新之前，随机翻转激活值向量或误差向量的符号。 读取 $A - R$ 后，再次对读数进行符号校正。这样，任何残余偏置

算法1 使用动态调整的并行内存更新随机脉冲。

输入数据向量 x_i，大小 n，梯度向量 d_j，大小 m，权重矩阵 $\breve{w}_{ij}$，对称点处的平均脉冲大小 δ_w，学习率 η，最大脉冲数 $l_{\max}$

输出 已更新的模拟权重矩阵 $\breve{w}_{ij}$

设置 $m_x \leftarrow$ 最大值$_i\,|x_i|$ 和 $m_d \leftarrow$ 最大值$_j\,|d_j|$

设置 $\kappa \leftarrow \frac{\eta m_x m_d}{\delta_w}$

设置 $l \leftarrow$ 最小值$(l_{\max}, \lceil\kappa\rceil)$ 和 $\tilde{m}_d \leftarrow m_d$ 最小值$(\frac{l_{\max}}{\kappa}, 1)$

设置 $a \leftarrow \sqrt{\frac{\eta\, m_x}{l\, \tilde{m}_d \delta_w}}$ 和 $b \leftarrow \sqrt{\frac{\eta\, \tilde{m}_d}{l_{\max}\, m_x \delta_w}}$

对于 $k = 1$ 到 l 执行

  将所有 $q_i \leftarrow 0$ 设为所有 $p_j \leftarrow 0$

  从 $i = 1$ 到 m 循环

    生成随机数 $\xi \in \mathcal{U}(0, 1)$

    如果 $a|d_i| < \xi$ 则

      将 $q_i \leftarrow$ 设为 sgn(d_i)

    结束条件判断

  结束循环

  对于 $j = 1$ 到 n 执行

    生成随机数 $\xi \in \mathcal{U}(0, 1)$

    如果 $b|x_j| < \xi$ 则

      将 $p_j \leftarrow$ 设为sgn(x_j)

    结束条件判断

  结束循环

  {发送脉冲向量 p 和 q 以更新 $\breve{w}_{ij}$:}

  对于 $i = 1$ 到 m，执行

    对于 $j = 1$ 到 n，执行

      如果 $p_j \neq 0$且 $q_i \neq 0$，则

        设置 $\breve{w}_{ij} \leftarrow \breve{w}_{ij} + dF(\,\breve{w}_{ij}|\boldsymbol{\theta}, \text{sgn}(q_i p_j))$

      结束条件判断

    结束循环

  结束循环

结束循环

都能通过滤波阶段平均消除，而不会显著增加算法的复杂度。在第二种新算法中，我们还提出动态估计近期参考点，并在运行过程中实时更新，从而进一步改善瞬态的消除效果，实质上彻底消除了˘对可调参考阵列 R 的需求。

2 背景：内存中优化器

2.1 模拟矩阵-向量乘法

早期已有研究提出使用电阻式交叉阵列在存储器内计算矩阵向量乘法（Steinbuch，1961年），并且已有多款原型芯片被报道用于加速推理期间深度神经网络的矩阵向量乘法（Wan 等人，2022年；Khaddam-Aljameh 等人，2021年；Xue 等人，2021年；Fick 等人，2022年；Narayanan 等人，2021年）。原则上，在所有这些方案中，线性层的权重存储在由可调电导组成的交叉阵列中，输入例如编码为电压脉冲，并分别利用欧姆定律和基尔霍夫定律将权重与输入相乘并累加乘积（图1A；另请参见（Sebastian 等人，2020年）以了解更多详情）。在许多设计中，所得电流或电荷通过高度并行的模数转换器（ADC）转换回数字信号。

正如（Gokmen& Vlasov，2016年）所提出的，要实现完全内存中的模拟训练，还必须实现一个转置矩阵向量乘法，以进行反向传播；这可以通过相应地转置输入和输出实现（见图1B）。

2.2 内存中的外积更新

虽然使用内存中的矩阵向量乘法加速随机梯度下降的前向传播和反向传播很有前景，但若要实现完整的内存训练方案，还必须考虑加速内存中的梯度计算和权重更新：这是深度神经网络训练中剩余的、复杂度为 $\mathcal{O}(n^2)$的运算（其中 $n \times n$ 是“模拟”矩阵的大小）（Gokmen& Vlasov，2016年）。

对于线性层权重矩阵 W 的梯度累积（即计算 $\mathbf{y} = W\mathbf{x}$），需要计算 $W \leftarrow W + \lambda \mathbf{d}\mathbf{x}^T$ 的外积更新。虽然可以通过数字方式完成，并可能利用稀疏性（例如，见（Nandakumar 等人，2020年）），但仍需要进行 $\mathcal{O}(n^2)$ 次数字运算，因此会限制内存训练所能实现的整体加速比。为使外积更新也能在内存中完全并行地执行，Gokmen 等人（Gokmen& Vlasov，2016年）建议使用随机脉冲序列及其重合（如图1 C所示）。

近年来，精确更新算法经历了多次改进（Gokmen 等人，2017年；2018年），不过，我们在此建议使用算法 1 中一个进一步改进的版本。具体而言，在我们的版本中，脉冲序列的长度会动态调整。请注意，为简化起见，我们在此假设可以发送负脉冲。实际操作中，负脉冲和正脉冲会分阶段依次发送。

尽管（Gokmen& Vlasov，2016；Gokmen 等人，2017年；2018年）研究了使用算法 1 直接在内存中实现梯度更新时的噪声特性，但结果表明，这要求存储器件元件具有非常对称的开关特性。因此，尤其是面对当今材料中观察到的不对称性时，这种“内存中的普通随机梯度下降”算法的要求颇具挑战性，我们将在下一节讨论这一点。

2.3 器件材料模型

当施加足够大的电压脉冲时，双向开关的器件材料（如电阻式随机存取存储器（Zahooret al.，2020年）通常会表现出显著的不对称性：两个方向的开关行为存在差异，并逐渐饱和至最小或最大电导值。在先前的研究（Fusi & Abbott，2007；Frascaroli etal.，2018年）中，研究表明，“软边界”模型能够很好地定性描述开关行为。根据该模型，权重变化 $w \leftarrow w + dF(w|\boldsymbol{\theta},\ D)$ 在向上（$D = +$ 或向下 $D = -$）方向施加单个更新脉冲时，由下式给出

$$\begin{aligned} dF(w|\boldsymbol{\theta},+) &= \alpha_+ \frac{b_{\max} - w}{b_{\max}}(1 + \sigma_{\text{c-to-c}}\xi) \\ dF(w|\boldsymbol{\theta},-) &= -\alpha_- \frac{b_{\min} - w}{b_{\min}}(1 + \sigma_{\text{c-to-c}}\xi) \end{aligned} \tag{1}$$

这里我们用占位符 $\boldsymbol{\theta}$ 表示超参数。为捕捉器件间变异性，我们首先抽取$b_{\max} = \max(1 + \sigma_b\xi_1,0)$ 和 $b_{\min} = \min(-1 + \sigma_b\xi_2,0)$，其中 $\xi_i \in \mathcal{N}(0,1)$是随机数，且每个器件对应的随机数都不同。斜率参数定义如下：

$$\begin{aligned} \alpha_+ &= \delta_w\ (\gamma + \rho) \\ \alpha_- &= \delta_w\ (\gamma - \rho) \end{aligned} \tag{2}$$

其中 $\gamma = e^{\sigma_{\text{d-to-d}}\xi_3}$，且 $\rho = \sigma_\pm\xi_4$，因此 $\sigma_{\text{d-to-d}}$是描述器件间斜率变化的超参数，而$\sigma_\pm$ 则是描述上行与下行方向斜率差异的另一项器件间变异性。材料参数 δ_w 决定了权重为零时单个脉冲的平均更新响应。我们将器件状态数定义为平均权重范围除以 δ_w，即$n_{\text{states}} = \frac{2}{\delta_w}$。

2.3.1 对称点

很容易看出，对于器件模型方程1，正向更新时，更新幅度会随权重接近上界而线性减小 $b_{\max}$；同样，负向更新时，更新幅度会随权重接近下界 $b_{\min}$而线性减小，因此权重会饱和于 $b_{\min}$ 和 $b_{\max}$。因此，存在一个权重值，使得向上和向下更新的平均幅度相等，这个值因此称为对称点（SP）（Gokmen 和 Haensch，2020年；Onen 等人，2022）。

求解方程 1 以求得对称点 $\breve{w}^*$ 通过要求上行和下行脉冲的大小相等，可得[2]

$$\breve{w}^* = \frac{\alpha_+ - \alpha_-}{\frac{\alpha_+}{b_{\max}} - \frac{\alpha_-}{b_{\min}}} = \frac{2\rho}{\frac{\gamma+\rho}{b_{\max}} - \frac{\gamma-\rho}{b_{\min}}} \tag{3}$$

[2]此处我们仅假设非退化情形，即 $b_{\max} > 0$,$b_{\min} < 0$, $\alpha_+ > 0$, and $\alpha_- > 0$

总之，我们使用的器件模型已按特定方式定义，使得当 $\sigma_\pm = 0$ 时，对称点位于零处。然而，如果 $\sigma_\pm > 0$，对称点就不在零处。

趋向 SP 时的瞬态 如果对器件施加随机的向上和向下脉冲（不存在单向偏置），器件将达到其 SP。如果正脉冲总是紧随负脉冲之后，则权重变化（暂且假设 $\sigma_b = \sigma_{\text{c-to-c}} = \sigma_\pm = \sigma_{\text{d-to-d}} = 0$）可写为：

$$\begin{aligned} \Delta w &= dF(w|\boldsymbol{\theta},+) + dF(w + dF(w|\boldsymbol{\theta},+)|\boldsymbol{\theta},-) \\ &\approx dF(w|\boldsymbol{\theta},+) + dF(w|\boldsymbol{\theta},-) \\ &= \delta_w(1 - w) - \delta_w(1 + w) \\ &= -2\delta_w w \end{aligned} \tag{4}$$

这表明，对于重复的向上和向下脉冲对，权重将以 $\tau = 2\delta_w$ 的（近似）衰减速率呈指数衰减至 $\breve{w}^* = 0$处的不动点。

将参考器件设为对称点 为消除衰减至对称点所产生的偏移量，以下讨论的某些算法要求将参考矩阵 R 设为对称点（Gokmen，2021年；Onen 等人，2022）。我们通过设定（使用公式 3）来对此进行建模

$$\breve{r}_{ij} = \breve{a}^*_{ij} + \mu_r + \sigma_r\xi_{ij} \tag{5}$$

其中 $\xi_{ij} \in \mathcal{N}(0,\ 1)$。因此 $o_r = \mu_r + \sigma_r\xi$ 表示在扣除对称点后参考器件上残留的偏移量。残留偏移量可能是由于对的对称点估计不准确， A 也可能是设置

R时产生的编程误差。实际中如何编程 R 至 A 的对称点，讨论见（Kim 等人，2019年）。

2.4 TTv2 算法

为了降低内存中随机梯度下降对器件的要求，(Gokmen 和 Haensch，2020年) 提出先在内存中快速计算外积更新，从而将近期的梯度历史累积到一个单独的模拟忆阻交叉阵列 A中，再通过依次进行向量读取（单个 列或行），将近期累积梯度缓慢传输到 A 模拟权重矩阵 W上，以抵消器件不对称性造成的梯度矩阵信息损失 A。这一方法的创新之处在于，A 上的累积梯度会因器件不对称性而以依赖于输入的方式衰减：如上所述，随机（无偏）输入会使 A 上的电导衰减至对称点（SP）。因此，若减去对称点，当不存在净梯度时（例如处于损失景观的极小值处），梯度矩阵便会按设计衰减至零。

此外，由于将梯度从 A 传输到 W只需进行 $\mathcal{O}(n)$ 次数字运算（每次矩阵更新最多传输一个向量）

算法 2 Chopped-TTv2（c-TTv2）算法。

输入学习率 λ、缓冲区尺度 γ_0、传输周期 n_s，
斩波器概率 ρ
输出收敛的模拟权重 $\breve{w}_{ij}$
设置 $\gamma \leftarrow \gamma_0 \frac{\delta_{w_{\max}}}{n n_s}$
将计数器 s 和 k 设为 0，并设置数字矩阵 $h_{ij} \leftarrow 0$
调用编程参考 R，获取 A 的对称点：$\breve{r}_{ij} \leftarrow \breve{a}^*_{ij}$
重复
调用开始小批量 {的随机梯度下降迭代（使用 $\breve{w}_{ij}$ 作为权重），直到需要更新 $\breve{w}_{ij}$}
对于要更新到的所有输入 x 和 d $\breve{W}$ **do**
调用 算法 1 更新 $(c_j x_j)$ 并将 d 映射到 $\breve{A}$
设置 $s \leftarrow s+1$
如果 $s = n_s$ **then**
设置 $k \leftarrow k+1$ 模 n 和 $s \leftarrow 0$ 以及所有 $p_i \leftarrow 0$
调用模拟矩阵-向量乘法：y $\leftarrow (\breve{A} - \breve{R})\, \mathrm{e}_k^{(n)}$
对于 $i = 1$ 到 m 执行
设为 $h_{ik} \leftarrow h_{ik} + c_k \frac{\lambda}{\gamma} y_i$
如果 $|h_{ik}| > 1$，则
设为 $p_i \leftarrow \mathrm{sgn}(h_{ik})$
设为 $h_{ik} \leftarrow 0$
结束 if
结束 for
{使用脉冲向量 p_i 更新列 k $\breve{W}$:}
对于 $i = 1$ 到 m，执行
如果 $p_i \neq 0$，则
设置 $\breve{w}_{ik} \leftarrow \breve{w}_{ik} + dF(\breve{w}_{ik} | \boldsymbol{\theta}, \mathrm{sgn}(p_i))$
结束条件判断
结束循环
{按如下方式翻转斩波器。}
生成随机数 $\xi \in \mathcal{U}(0, 1)$
如果 $\rho < \xi$ 则
设置 $c_k \leftarrow -c_k$
结束如果
结束如果
结束循环
调用完成随机梯度下降迭代
直到收敛

传输读写 W 所引入的数字运算和额外计算并非瓶颈；而如果在数字域中执行完整的梯度更新，则需要进行 $\mathcal{O}(n^2)$ 次运算。

在（Gokmen，2021年）中，该算法通过使用一个数字矩阵对传输的梯度进行额外滤波而得到改进 H，该矩阵还会累积最近读取的梯度（来自 A），并且只有达到阈值后，才会发送一个脉冲，以更新权重矩阵中相应的器件。W这种额外的平均虽然引入了更多数字计算，但仍保留了 $\mathcal{O}$（n）特性，因为每次传输至多只对一个向量进行操作。研究表明

（Gokmen，2021年）证明，滤波阶段大幅降低了对器件的要求，尤其是对有效器件状态数量的要求：只需15种状态，就足以将长短期记忆网络训练到可接受的精度。降低后的器件要求使当前的阻变存储器器件也有望用于内存训练（Gong 等人，2022年）。这种 Tiki-Taka 2.0（TTv2）算法是我们的基准比较方法[3]。

3 改进的存内优化器

我们在此介绍新的算法贡献，以及一种根据材料规格和权重矩阵调整学习率的新方法。

3.1 c-TTv2 算法

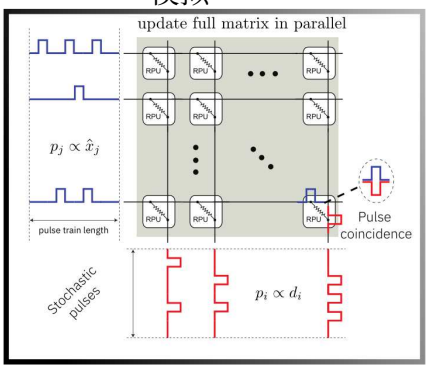
虽然模拟参考矩阵 R 具有以下优势：能高效地减去对称点 A，但这种设计选择也带来了独特的挑战。具体而言，R 的编程可能并不精确，或者对称点可能随时间缓慢变化，从而残留一些偏移瞬态。然而，任何偏移量都会持续累积在 H 上，并被写入 W，因而可能无意中使权重矩阵产生偏置。此外，衰减至其对称点的速度会随器件状态数增加而变慢（见式 4），且取决于输入。后者可能导致 A 经过许多次更新仍未衰减，进而使 W 产生偏置（如下所示）。尽管损失的反馈最终会改变梯度并校正 W，学习动态仍可能受到影响。

为进一步减轻残余偏置及低频噪声和偏置源，我们在此建议通过引入斩波器来改进算法。斩波稳定是放大器电路设计中常用的偏移校正方法（Enz 和 Temes，1996年）。斩波器在梯度累积之前调制输入信号，随后在读取累积梯度时对其进行解调。具体而言，我们引入斩波器 $c_j \in \{-1, 1\}$，用于翻转每个激活值 x_j 的符号，然后在上进行梯度累积 A。读取 k的第列 A 并将其传输到 H 时（如 TTv2 中所示），我们应用对应的斩波器 c_k 以恢复信号的正确符号。随后随机翻转斩波器，详细算法见算法 2（另见图 2）。

这样，由不对称性引起的任何低频分量，以及任何残留的偏移量和瞬态在 A 上都不会被斩波器调制，因此会通过符号翻转抵消。我们将这种算法称为 Chopped-TTv2 (c-TTv2) 随机梯度下降。

[3]请注意，通过设置 Alg. 2 和图 2，可以推导出 TTv2 的详细实现 $\rho = 0$，从而将斩波器固定为 $c_k = 1$，如第 3 节所述。

模拟 $\breve{A}$



使用斩波器更新梯度

$$\hat{x}_j = c_j x_j$$
$$\breve{A} \leftarrow \breve{A} + \eta\, \mathbf{d}\hat{\mathbf{x}}^T$$

数字 H

读取和传输时：

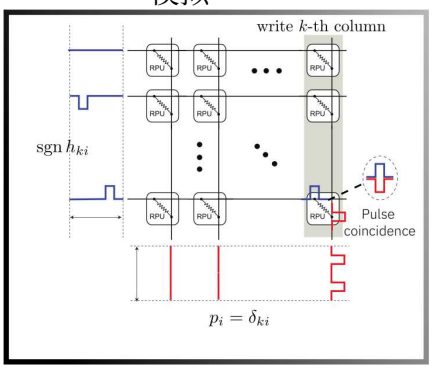
$$k \leftarrow (k+1) \bmod d_{\text{in}}$$
$$\boldsymbol{\omega} \leftarrow \left(\breve{A} - \breve{R}\right) \mathbf{e}_k$$
$$\boldsymbol{\mu}_k \leftarrow (1-\beta)\boldsymbol{\mu}_k + \beta\boldsymbol{\omega}$$
$$\mathbf{h}_k \leftarrow \mathbf{h}_k + c_k \tfrac{\lambda}{\gamma}\left(\boldsymbol{\omega} - \boldsymbol{\mu}_k^{\text{past}}\right)$$

当斩波器翻转时：

$$c_k \leftarrow -c_k$$
$$\boldsymbol{\mu}_k^{\text{past}} \leftarrow \boldsymbol{\mu}_k$$
$$\boldsymbol{\mu}_k \leftarrow 0$$

读取一列/行并进行数字处理
计算一行/列

模拟 $\breve{W}$



达到阈值时写入单个脉冲：

$$\breve{W} \leftarrow \breve{W} + \lambda_W \left(\operatorname{sgn} \mathbf{h}_k\right) \mathbf{e}_k^T$$
$$h_{ik} \leftarrow \begin{cases} 0, & \text{if } |h_{ik}| > 1 \\ h_{ik}, & \text{otherwise.} \end{cases}$$

图2. 改进的存内训练算法：c-TTv2 和 AGAD。蓝色表示 AGAD 独有且额外增加的方程。AGAD 不需要参考减法（红色部分）。移除蓝色方程并修正斩波器后，即可恢复 TTv2。

3.2 AGAD 算法

虽然斩波器与低通滤波大幅提升了对任何残留偏移量的抵抗力（见第 4 节），但如果偏移量过大，所需的滤波可能会过慢，从而降低算法的响应速度。可以如下理解这一点。假设一个恒定梯度 g 被计算并作用于梯度矩阵中的单个元素，$\breve{a}$。进一步假设有两个斩波器相位，每个相位有 k 次读取，参考偏移为 o_r。那么，隐藏矩阵 h 中相应的元素将发生以下变化

$$h_{t+n} - h_t = \lambda k\,(o_r + g) \tag{6}$$
$$h_{t+2n} - h_t = \lambda k\,(o_r + g - o_r + g) = 2\lambda k g \tag{7}$$

因此，偏置会通过斩波器符号翻转而被消除，梯度也能正确恢复（式 7）。然而，在步骤（式 6 ）中，这种消除会带来问题：如果 $|o_r| \gg |g|$ 且$\lambda n|o_r| \approx 1$ （1 为阈值，见算法 2），那么增加偏置会错误地触发更新阈值。因此，对于较大的偏置，需要降低学习率 λ，以避免意外触发对 $\breve{w}$的更新，但这会减慢学习速度。

为改善这一问题，我们建议即时估计参考值：使用额外的数字计算，在斩波器符号翻转前存储 $\breve{a}$ 的值，并使用该值[4] 作为参考，而非使用$\breve{}$参考矩阵 R。这里的理由是，斩波器翻转与梯度信息的方向无关。因此，如果当前存在显著的平均梯度，$\breve{}$那么对 A 的更新方向就必须改变

[4]或者，一般而言，对近期的 $\breve{a}$ 取泄漏平均

斩波器翻转时。因此， 的近期值$\breve{a}$可以作为良好的参考点，用于无偏置地读取 $\breve{a}$ 上的信息。这种新算法称为带动态参考的模拟梯度累积（AGAD）。实现细节请参见算法 3和图 2 。

3.3 学习率缩放定律

在 TTv2 算法的原始表述中（Gokmen，2021年），向隐藏矩阵 λ_H 写入时所用的学习率 H 并未明确指定。我们在此建议采用以下缩放定律

$$\lambda_H = \frac{\lambda\, n_s\, n}{\gamma_0 \delta_w} \tag{8}$$

其中， n 是权重矩阵的列数，n_s 是在对 A进行一次$\breve{}$列读取之前执行的梯度更新次数， δ_w 是 A的对称点（SP）处平均更新响应的大小$\breve{}$。请注意，根据公式 8 H，阈值（即何时写入 W）$\breve{}$实际上会随梯度矩阵 A 中单个元素在读取发生前被更新的频率而缩放，并自然地除以平均更新大小的度量值 δ_w。这里， λ 是标准随机梯度下降（SGD）的学习率，该学习率可以按计划调整。因此，我们不调整 A 上的学习率$\breve{}$，而是根据整体随机梯度下降学习率调整写入 H 。超参数 γ_0 指定了累积长度；值越大，读取的梯度就会被平均得越久。不过请注意，调整 λ 也能达到相同效果，因此实际应用中只需调整其中一个即可。

同样，用于将梯度更新$\breve{}$应用于 A的学习率 η，必须针对当前的深度神经网络进行恰当设置。由于脉冲宽度有限，梯度幅度必须

算法 3 使用动态参考的模拟梯度累积（AGAD）算动态参考（AGAD）算法。

输入学习率 λ，缓冲区尺度 γ_0，传输周期 n_s，斩波频率 ρ，泄漏平均时间尺度 β

输出收敛的模拟权重 $\breve{w}_{ij}$

设置 $\gamma \leftarrow \gamma_0 \frac{\delta_w}{n n_s}$

将计数器 s, k 和 t 设为 0，并设置所有斩波器 $c_j \leftarrow 1$

将数字矩阵 h_{ij}, μ_{ij} 和 μ_{ij}^{past} 全部设为 0

重复

调用开始随机梯度下降迭代

针对所有要更新到.....的输入 x 和 d $\breve{W}$ **do**

调用算法1 以更新 $c_j x_j$ 和 d_i 到 $\breve{a}_{ij}$

设定 $s \leftarrow s+1$

如果 $s = n_s$ **then**

设定 $s \leftarrow 0$ 和 $t \leftarrow t+1$ 以及所有 $p_i \leftarrow 0$

设定 $k \leftarrow k+1$ 模 n

调用模拟矩阵-向量乘法: y $\leftarrow \breve{A}\,\mathbf{e}_k$

对于 $i = 1$ 到 m 执行

设置 $h_{ik} \leftarrow h_{ik} + c_k \frac{\lambda}{\gamma}\left(y_i - \mu_{ik}^{\text{past}}\right)$

设置 $\mu_{ik} \leftarrow (1-\beta)\,\mu_{ik} + \beta y_i$

如果 $|h_{ik}| > 1$则

设置 $p_i \leftarrow$ 为sgn(h_{ik})

设置 $h_{ik} \leftarrow 0$

结束 if

结束 for

{使用脉冲向量 p_i 更新列 k 的 $\breve{W}$:}

对于 $i = 1$ 到 m，执行

如果 $p_i \neq 0$，则

$$\breve{w}_{ik} \leftarrow \breve{w}_{ik} + dF(\breve{w}_{ik}|\boldsymbol{\theta}, \text{sgn}(p_i))$$

结束条件判断

结束循环

{切换斩波器并按如下方式重置 μ_{ij} 和 μ_{ij}^{past}。}

如果 t 取模 $\left\lfloor \frac{1}{\rho} \right\rfloor = 0$ 则

设置 $c_k \leftarrow -c_k$

对于 $i = 1$ 到 m 执行

设置 $\mu_{ik}^{\text{past}} \leftarrow \mu_{ik}$

设置 $\mu_{ik} \leftarrow 0$

结束循环

结束条件判断

结束条件判断

结束循环

调用以完成随机梯度下降迭代

直到达到收敛。

足够大，才能使梯度矩阵 A发生显著变化，因此需要相应地缩放学习率 η。由于不同层的梯度幅度往往不同，并且也可能随时间变化，我们会动态地将 η 除以 μ_x 输入 $m_x = \max_j |x_j|$和输入梯度 $m_d = \max_i |d_i|$绝对值最大值的近期滑动平均值μ_d。

$$\eta = \frac{\eta_0 l_{\max} \delta_w}{\mu_x \mu_d} \tag{9}$$

请注意， m_x 和 m_d 梯度更新时已经需要这些量（参见算法 1），因此除泄漏平均计算外，无需进行任何额外计算。由于 $l_{\max}\delta_w$ 大致是器件材料在一次更新期间能够改变的最大值（$l_{\max}$ 是所用的脉冲数量，参见算法 1），公式 9 意味着在 $\eta_0 = 1$ 的情况下，梯度更新乘积 $x_i d_j > \mu_x \mu_d$将被裁剪。虽然 η_0 的默认值为 1，但在某些情况下，较大的值有助于改善学习效果。对于此处介绍的所有算法，我们都使用这两种缩放定律。

4 结果

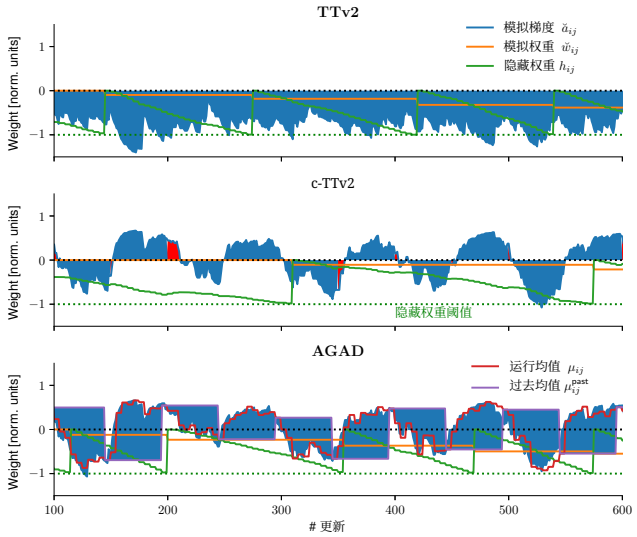
下面我们首先介绍若干数值测试用例，以说明并比较训练算法的机制。随后，我们将这些算法应用于不同器件材料和参考偏移设置下的随机梯度下降与深度神经网络训练，并进行比较。仿真使用基于 PyTorch 的开源工具包（Paszke 等人，2019年）[5] IBM 模拟硬件加速工具包（AIHWKit）（Rasch 等人，2021年）。

4.1 无偏置的恒定平均梯度

为说明不同学习算法的差异，我们首先考察一个简单情形，其中激活值为$x = -X$，梯度输入为 $d = \alpha X + (1-\alpha)Y$，其中 $X, Y \sim \mathcal{N}(0,1)$ 是高斯随机变量。因此，在这种情况下，激活值与梯度之间的相关性为 α，预期平均更新仅沿一个方向 $\Delta\breve{w} \propto -\alpha$。图 3 展示了三种算法 TTv2、c-TTv2 和 AGAD 中各变量随更新次数的变化。这里我们假设$\breve{}$参考矩阵 R 被精确设置为$\breve{}$ A 的对称点，因此不存在偏置。

请注意，对于 TTv2（图 3上方面板），所选 $\breve{a}$ 的轨迹明显偏向负值，因此能够正确累积净梯度。不过，由于底层器件模型的特性（见公式 1），它会在某一水平饱和。由于偶尔会进行读取，隐藏权重会持续累积，直到在 -1 达到阈值（绿色轨迹）；此时，权重 $\breve{w}$ 通过一个脉冲进行更新（橙色轨迹）。蓝色阴影区域表示 $\omega = c(\breve{a} - \breve{r})$ 的值（对于 TTv2，不使用斩波器 $c = 1$）。如果该值为正，区域将显示为红色，这表示隐藏权重更新方向有误。

[5]工具包中已实现 TTv2 算法；不过，我们通过采用融合 CUDA内核设计（而非现有的“分块”内核，见图 A.2），将其 GPU 实现的运行速度提升了最高 $40\times$，并实现了我们的新算法。我们还实现了一种定制的新器件模型，可显式减去对称点（该功能在工具包中不可用）。



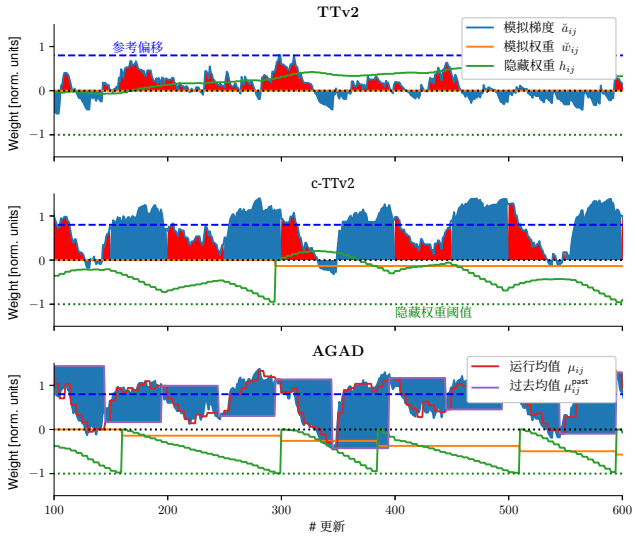


图3.在完美的 SP 估计下，具有恒定（负）梯度输入的算法机制。参数：$\delta_w = 0.05$, $\sigma_b = \sigma_\pm = \sigma_{\text{d-to-d}} = \sigma_{\text{c-to-c}} = 0.3$ $\gamma_0 = 200$ $\lambda = 0.1$ $n_s = 1$,,,, $\beta = 0.5$ $\rho = 0.1$ $l_{\max} = 5$ $\eta = 1$ $\mu_r = \sigma_r = 0$,,,, 和 。

图 3中间面板展示了 c-TTv2 算法在相同输入下的行为。为便于说明，这里选用了固定的斩波器周期。由于输入梯度为恒定负值，斩波器符号的调制会导致 $\breve{a}$中的梯度累积出现振荡。不过，由于符号经过校正，隐藏矩阵（大部分情况下）会朝正确方向更新（蓝色区域表示符号已校正）。下文将会看到，这种符号翻转可以抵消偏移量（此处假定偏移量为 0）。在本例中，由于存在瞬态，读取前 $\breve{a}$ 的轨迹并非总能回到对称点，这会导致隐藏权重在某些瞬态下朝错误方向更新（红色区域）。尽管如此，权重 $\breve{w}$ 仍能正确更新，因为隐藏权重成功地对瞬态进行了平均。不过， $\breve{w}$的变化率在一定程度上会受到瞬态平均的影响。

由于 AGAD（图 3 下方面板）使用近期累积梯度的（平均）值μ^{past} $\breve{a}$ 作为参考˘点（而非编程到 R上的静态 SP），因此不会受到同样瞬态的困扰。事实上，范围的增大使隐藏矩阵更新得更快，随后权重 $\breve{w}$也更快更新。

4.2 带偏置的恒定平均梯度

˘由于图 3 中的参考值 R 恰好设为对称点，因此定点没有偏置。在这种情况下，TTv2 确实表现良好，并且可能是首选算法，因为它所需的数字计算量最少。不过，现在我们来考察存在较大偏置的情况，˘ ˘使得 $A - R$的有效 SP 不为零。在图 4中，我们

恒定梯度输入且参考偏移显著时算法的机制。参数同图3，但 $\mu_r = 0.8$。

重复图 3 中的实验，加入偏置 $\mu_r = 0.8$（图 4中的虚线）。我们注意到，对于 TTv2，恒定梯度会将累积梯度 $\breve{a}$ 推离对称点（蓝色虚线）；然而，由于算法未将偏置考虑在内，因此对隐藏矩阵的更新是错误的。事实上，在本例中，隐藏权重 h （绿色曲线）的更新方向甚至也是错误的（注意，预期它会变得更负）。

另一方面，由于斩波器的作用，即使是如此大的偏置，也能通过 c-TTv2 算法成功消除（图 4 中间图）。请注意，隐藏值 h 以及权重 $\breve{w}$ 都在正确地减小。然而，由于偏置较大，明显的振荡（红色区域）会对 h上的累积造成干扰，从而降低梯度累积的速度和保真度。

由于 AGAD 算法采用动态参考点计算，在这种情况下参考偏移几乎无关紧要（图 4 下图），因此可以完美地校正任何偏置。

4.3 输入诱导衰减

TTv2 算法的一个显著特征是输入诱导衰减：随机输入或梯度波动会将˘的值 A 拉向对称点（见式4），从而使其在算法上衰减至˘零（由于参考 R，见式5）。如图A.3所示，在梯度突然关闭后，隐藏矩阵会迅速衰减。这是由于本例中输入的噪声性˘导致 A 迅速衰减至其对称点。

尽管激活值和梯度通常可以很好地近似为噪声过程

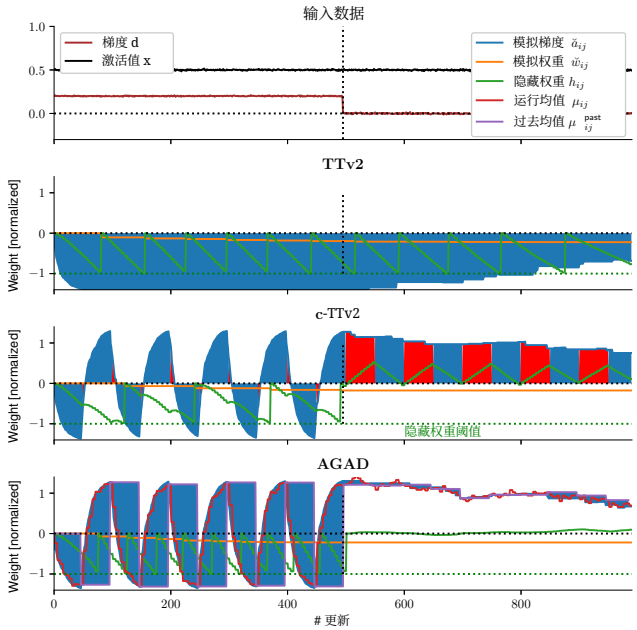


图 5. 梯度输入的突变。当输入波动较小时，˘TTv2 的梯度矩阵 A 会出现长时间衰减，这可能会在梯度输入稀疏的情况下导致权重更新产生偏差（注意，在 500 处关闭净梯度后，TTv2 仍然达到了隐藏权重阈值；绿色曲线）。我们的新算法纠正了这一行为。参数：同图3。

（参见Onen 等人，2022年的讨论）），但在某些情况下，深度神经网络训练期间的梯度可能非常稀疏，例如使用 ReLU 的卷积神经网络。如果经过一段时间有输入后，突然不再向 TTv2 提供输入，例如元素(i, j) 属于 dW 且为零，那么$\breve{a}_{ij}$ 不会发生变化，但仍会持续被读入 h_{ij}，进而错误地更新 $\breve{w}_{ij}$。图5展示了这样一个（人为构造的）例子：与之前一样，在 500 次更新后将梯度设为零（图A.3），但降低了输入噪声。请注意，由于输入波动依赖型衰减，TTv2 算法的有效动量项此时会变得非常大（关闭梯度后出现的蓝色区域）。权重更新最终会反馈至输入梯度并对其进行校正，但梯度更新的质量仍可能受到影响。

在我们提出的新算法（c-TTv2 和 AGAD）中，不存在这种伪影，因为这两种算法中，施加到隐藏矩阵上的梯度更新都会受到斩波器符号翻转的调制（图 A.3 和图 5 下方的面板）。具体而言，稀疏梯度不会导致任何残余权重更新，因为如果 $\breve{a}_{ij}$ 在多个斩波器周期内保持恒定，就会被视为偏置并予以扣除。对于 c-TTv2，这种扣除发生在隐藏矩阵上（见图 5中的红色和蓝色区域）。对于 AGAD，恒定的 $\breve{a}_{ij}$ 甚至不会被添加到隐藏矩阵中，因为切换斩波器相位时，动态参考点调节会即时消除任何残余偏置（图 5 下方）。

4.4 单个线性层上的随机梯度下降

接下来，我们测试使用随机梯度下降时算法的表现。我们考虑训练一个线性层，使其输出$f_i(\mathbf{x}) = \sum_{j=1}^{n} w_{ij}x_j$，以得到给定的目标权重矩阵 ˆ$W$。我们将误差定义为（使用目标权重 W 得到的期望输出与实际输出之间的均方偏差）

$$L(\mathbf{x}|W, \hat{W}) = \frac{1}{2m}\sum_{i}^{m}\left(f_i(\mathbf{x}) - \sum_{j=1}^{n}\hat{w}_{ij}x_j\right)^2 \qquad (10)$$

自然，使用随机梯度下降最小化偏差并ˆ更新 W时，误差在 $W = W$ 处达到最小（假设ˆ权重矩阵满秩）。我们将 W 设为随机值$\mathcal{N}(0, 0。3)$，并使用 $x_j \sim \mathcal{N}(0, 1)$ 作为输入。收敛后，我们根据达到的权重误差 $\epsilon_w^2 = \left\langle (w_{ij} - \hat{w}_{ij})^2 \right\rangle$评估不同算法；该误差是学习得到的权重与目标权重之间的标准差。图6展示了一个 20×20 权重矩阵的结果。

我们比较了三种算法（TTv2、c-TTv2 和 AGAD），以及直接在权重上进行梯度更新的普通内存中随机梯度下降˘（梯度更新直接在权重上完成） W （算法 1）。我们探究了两种参数变化下的鲁棒性：（1）参考偏置的标准差 σ_r，以及器件状态数 n_{states}。与预期一致，在不存在偏置的情况下 $\sigma_r = 0$，且与原始研究（Gokmen，2021年）相符，TTv2 算法表现出色，远胜于内存中随机梯度下降，尤其是在状态数较少时（例如，对于 20 个状态和完全相同的目标权重矩阵，分别为 $\epsilon_w \approx 8\%$ 对比 $> 25.0\%$；见图 6AB）。然而，参考偏移变化$\sigma_r > 0$ 会严重影响 TTv2 的性能。一旦 $\sigma_r = 0.1$，权重误差便会显著增大，给当前的器件材料和参考权重编程技术带来挑战。

另一方面，使用斩波器可显著提高对偏移量的鲁棒性（图 6C、D）。c-TTv2 算法在偏移量较大时仍能保持相同的权重误差，尤其是在状态数较少的情况下。状态数较多时，偏移量的校正效果较差，这与状态数增加会导致瞬态问题加剧的现象一致（见公式 4）。对于 AGAD 而言，参考偏移并不重要，因为参考值会动态地即时计算。此外，与 c-TTv2 不同，AGAD 在状态数较多时同样表现良好，表明瞬态也不会造成问题。不过，与 c-TTv2 相比，在状态数较少且偏移量较小时，权重误差会略有增加，这可能是由于即时估计参考值引入了一些不准确性。

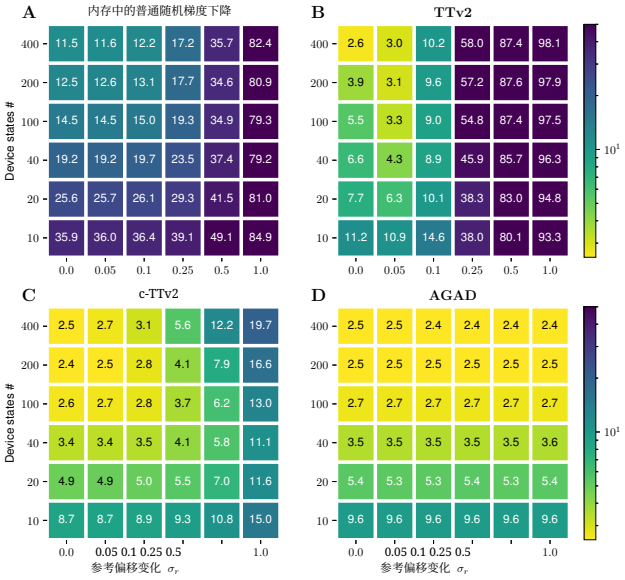


图6. 使用不同学习算法进行权重编程时的编程误差。参数：与图 3 相同，唯独 σ_r 和 δ_w 有所变化。˘此外，我们仅对 $\sigma_b = 0$ 设置 W（以免将结果与无法用 $\breve{W}$表示目标权重的情况混淆）。每种情况下，目标矩阵、器件间差异和输入均保持不变，以便更好地进行比较。结果取 3 次运行的平均值。

4.5 深度神经网络训练实验

最后，我们比较了不同学习算法在实际深度神经网络训练中的表现。为便于比较，我们基本沿用了此前用于评估早期算法的相同深度神经网络。这些网络包括 MNIST 上的全连接深度神经网络（Gokmen& Vlasov，2016年）、MNIST 上的 LeNet（Gokmen 等人，2017年），以及双层长短期记忆网络（Gokmen 等人，2018年；Gokmen，2021年）。我们再次使用不同的参考偏移变化对这些深度神经网络进行了训练（见图 7；详见第 A.1 节中的仿真细节），并采用相同的具有挑战性的器件模型（见 $n_{\text{states}} = 20$ 图 A.1中的器件响应轨迹示例）。不同深度神经网络的结果高度一致，并证实了单层权重编程实验中发现的趋势（对比图 6）：如果偏移量能够得到完美校正，所有算法的表现都非常相近。然而，正如预期的那样，参考偏移对 TTv2 的影响相当显著，而 c-TTv2 在偏移量变得过大之前能够对其进行较大程度的校正。另一方面，AGAD 完全不受偏移量影响，通常表现最佳（图 7B-D）。

我们发现，即使没有偏移量，这两种新算法的表现也都优于最先进水平的 TTv2。此外，既然现在即便存在偏移量以及 A 上的˘瞬态，梯度也能计算得如此准确，那么未校正˘的 W 上的 W 对称点所产生的二阶效应便开始变得显著。事实上，对于这两种算法，测试误差都进一步降低，甚至优于浮点（FP）测试误差

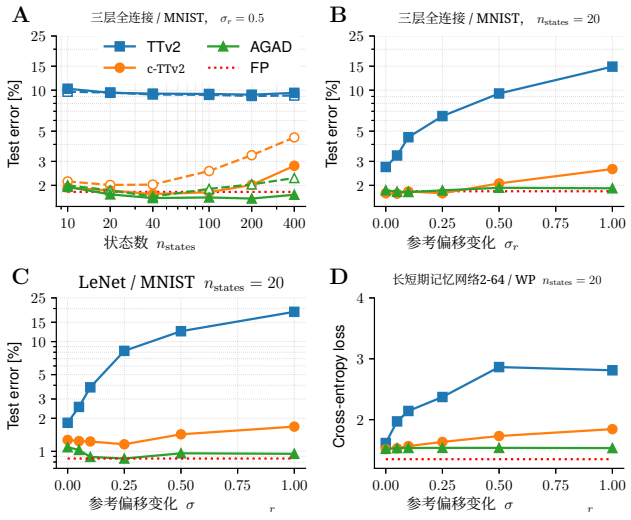


图7.采用不同模拟学习算法进行深度神经网络训练。˘W的对称点（SP）可以进行校正（实心符号），也可以不进行校正（空心符号和虚线）。(A) 在较大的参考偏移变化 $\sigma_r = 0.5$ 下，状态数与测试误差的关系。(B、C、D) 在 $n_{\text{states}} = 20$ 下，参考偏移变化与测试误差的关系。深度神经网络：(A、B) MNIST 上的三层全连接深度神经网络；(C) MNIST 上的 LeNet；(D) 《战争与和平》数据集上的两层长短期记忆网络。

˘当 W 的对称点被减去并因此得到校正时（实心符号），c-TTv2 和 AGAD 的表现如上；但若未校正（空心符号），误差则会有所增加。尤其是在状态数较多时，AGAD 的表现优于 c-TTv2（图7 A）。

5 讨论

我们提出了两种用于交叉阵列并行内存训练的新学习算法。这些算法的一个关键特点是，计算完整随机梯度下降（SGD）所需的所有操作——即前向传播、反向传播和更新——所需时间为 [6] $\mathcal{O}(n)$。尤其是，权重矩阵更新过程中计算外积通常需要的 $\mathcal{O}(n^2)$，由于采用固定数量脉冲的重合，在 $\mathcal{O}(1)$ 时间内即可完成。读取一列同样完全并行，耗时为 $\mathcal{O}(1)$；而额外的数字计算一次处理一列，因此耗时为 $\mathcal{O}(n)$。

由于 c-TTv2 仅引入符号翻转，因此为 TTv2 增加的计算复杂度可以忽略不计。对于 AGAD，参考点是在运行时估算的，这需要额外的两个数字权重矩阵，其加载和存储也会增加运行时间。但由于额外操作仍然一次只在一列上执行，AGAD 保持了$\mathcal{O}(n)$复杂度。降低 AGAD 数字计算量的一种方法是设置 $\beta = 1$，从而不对参考点计算近期平均值，

[6]为简化起见，我们在此假设整个矩阵可以放入一个交叉开关中。

这会将额外存储需求降至一个矩阵（仅 μ^{past}），但可能会引入更多噪声。

AGAD 的优势在于，它能提升材料状态较多时的性能，并使额外的参考交叉开关 R 变得过时，从而有望降低单元˘的复杂度。或者，也可以不移除 R，而是将这个交叉开关˘用于存储 W 的 SP，从而在单元设计不变的情况下进一步提高精度。

我们发现，c-TTv2 和 AGAD 都提升了内存训练的性能上限，同时放宽了对器件材料状态数量的要求。事实上，我们表明，即使存在 SP 波动或长期瞬态，对于本文示例中的深度神经网络，20 种状态现在已足以实现近乎浮点精度的训练。

参考文献

Agarwal, S., Plimpton, S. J., Hughart, D. R., Hsia, A. H., Richter, I., Cox, J. A., James, C. D., 和 Marinella, M. J. 神经网络算法加速器对电阻式存储器器件的要求。载于《2016 年国际联合神经网络会议（IJCNN）》，第 929–938 页。IEEE，2016 年。

Ambrogio, S., Narayanan, P., Tsai, H., Shelby, R. M., Boy-bat, I., Nolfo, C., Sidler, S., Giordano, M., Bodini, M., Farinha, N. C. 等。使用模拟存储器进行等精度加速神经网络训练。《自然》，558(7708):60，2018年。

Burr, G. W., Shelby, R. M., Sebastian, A., Kim, S., Kim, S., Sidler, S., Virwani, K., Ishii, M., Narayanan, P., Fu-marola, A. 等。使用非易失性存储器进行神经形态计算。《物理学进展：X》，2(1):89–124，2017年。

Enz, C. C. 和 Temes, G. C. 减少运算放大器缺陷影响的电路技术：自动归零、相关双采样和斩波稳定。《IEEE会刊》，84(11):1584–1614，1996年。

Fick, L.、Skrzyniarz, S.、Parikh, M.、Henry, M. B. 和 Fick, D. 用于边缘人工智能实时视频分析的模拟矩阵处理器。载于 2022年 IEEE 国际固态电路会议（ISSCC），第65卷，第260–262页。IEEE，2022年。

Frascaroli, J.、Brivio, S.、Covi, E. 和 Spiga, S. 用于神经形态计算的模拟忆阻器件中软边界行为的证据。《科学报告》，8(1):1–12，2018年。

Fusi, S. 和 Abbott, L. 有界突触的记忆存储容量限制。《自然-神经科学》，10(4): 485–493，2007年。

Gokmen, T. 在噪声硬件上实现神经网络训练。《人工智能前沿》，4:1–14，2021年。Gokmen, T. 和 Haensch, W. 电阻式器件阵列上神经网络训练的算法。《神经科学前沿》，14，2020年。Gokmen, T. 和 Vlasov, Y. 利用电阻式交叉点器件加速深度神经网络训练：设计考量。《神经科学前沿》，10:333，2016年。

Gokmen, T.、Onen, M. 和 Haensch, W. 使用电阻式交叉点器件训练深度卷积神经网络。《神经科学前沿》，11:538，2017年。

Gokmen, T.、Rasch, M. J. 和 Haensch, W. 使用电阻式交叉点器件训练长短期记忆网络。《神经科学前沿》，12:745，2018年。

Gong, N., Rasch, M., Seo, S.-C., Gasasira, A., Solomon, P., Bragaglia, V., Consiglio, S., Higuchi, H., Park, C., Brew, K., 等。《14纳米互补金属氧化物半导体兼容忆阻式随机存取存储器阵列中的深度学习加速：器件、材料与算法协同优化》。发表于 IEEE国际电子器件会议，2022年。

Haensch, W., Gokmen, T. 和 Puri, R.《下一代深度学习硬件：模拟计算》。《IEEE会刊》，107(1):108–122，2019年。

Jain, S. 等。《采用阻性交叉开关的神经网络加速器设计：机遇与挑战》。《IBM研究与开发期刊》，63(6):10–1，2019年。

Khaddam-Aljameh, R., Stanisavljevic, M., Mas, J. F., Karunaratne,G., Braendli, M., Liu, F., Singh, A., M¨uller, S. M., Egger, U., Petropoulos, A., 等。《Hermes核心——一种基于14纳米互补金属氧化物半导体和相变存储器的存内计算核心，采用由300ps/lsb线性化压控振荡器型模数转换器阵列及本地数字处理构成的架构》。发表于2021年超大规模集成电路会议，第1–2页。IEEE，2021年。

Kim, H., Rasch, M. J., Gokmen, T., Ando, T., Miyazoe, H., Kim, J.-J., Rozen, J., and Kim, S. 用于在电阻式交叉点阵列上进行深度神经网络训练的零点偏移技术。arXiv 预印本 arXiv:1907.10228，2019。

Kingma, D. P. and Ba, J. Adam：一种随机优化方法。arXiv 预印本 arXiv:1412.6980，2014。

Krizhevsky, A., Hinton, G., 等。从小型图像中学习多层特征。2009。

LeCun, Y.、Bottou, L.、Bengio, Y. 和 Haffner, P. 应用于文档识别的基于梯度的学习。《IEEE会刊》，86(11):2278–2324，1998。

Nandakumar, S. R.、Le Gallo, M.、Boybat, I.、Rajendran, B.、Sebastian, A. 和 Eleftheriou, E. 基于计算存储器的用于训练深度神经网络的混合精度架构。载于《2018年IEEE国际电路与系统研讨会》（ISCAS），第1–5页，2018年。Nandakumar, S. R.、Le Gallo, M.、Piveteau, C.、Joshi, V.、Mariani, G.、Boybat, I.、Karunaratne, G.、Khaddam-Aljameh, R.、Egger, U.、Petropoulos, A.、Antonakopoulos, T.、Rajendran, B.、Sebastian, A. 和 Eleftheriou, E. 基于计算存储器的混合精度深度学习。《神经科学前沿》，14，2020年。Narayanan, P.、Ambrogio, S.、Okazaki, A.、Hosokawa, K.、Tsai, H.、Nomura, A.、Yasuda, T.、Mackin, C.、Lewis, S.、Friz, A. 等。通过对基于相变存储器的权重进行精确的逐行编程以及采用持续时间格式的并行向量传输，实现14纳米全片上乘累加。载于《2021年超大规模集成电路技术研讨会》，第1–2页。IEEE，2021年。Onen, M.、Gokmen, T.、Todorov, T. K.、Nowicki, T.、Del Alamo, J. A.、Rozen, J.、Haensch, W. 和 Kim, S. 使用非对称交叉点器件进行神经网络训练。《人工智能前沿》，5，2022年。Paszke, A.、Gross, S.、Massa, F.、Lerer, A.、Bradbury, J.、Chanan, G.、Killeen, T.、Lin, Z.、Gimelshein, N.、Antiga, L. 等。PyTorch：一种命令式风格的高性能深度学习库。《神经信息处理系统进展》，32，2019年。Rasch, M.、Moreda, D.、Gokmen, T.、Gallo, M. L.、Carta, F.、Goldberg, C.、Maghraoui, K. E.、Sebastian, A. 和 Narayanan, V. 用于模拟模拟交叉开关阵列上的训练和推理的灵活快速PyTorch工具包。IEEE人工智能电路与系统国际会议（AICAS），第1–4页，2021年。Sebastian, A.、Le Gallo, M.、Khadda m-Aljameh, R. 和 Eleftheriou, E. 存储器件及其在存内计算中的应用。Nature Nanotechnology, 15:529–544，2020年。Steinbuch, K.《学习矩阵》。《控制论》，1(1):36–45，1961年。Sze, V.、Chen, Y.-H.、Yang, T.-J. 和 Emer, J. S. 深度神经网络的高效处理：教程与综述。《IEEE会刊》，105(12):2295–2329，2017年。Wan, W.、Kubendran, R.、Schaefer, C.、Eryilmaz, S. B.、Zhang, W.、Wu, D.、Deiss, S.、Raina, P.、Qian, H.、Gao, B. 等。基于电阻式随机存取存储器的存内计算芯片。《自然》，608(7923):504–512，2022年。

Xue, C.-X., Chiu, Y.-C., Liu, T.-W., Huang, T.-Y., Liu, J.- S., Chang, T.-W., Kao, H.-Y., Wang, J.-H., Wei, S.-Y., Lee, C.-Y., 等. 一种基于电阻式随机存取存储器、集成于 CMOS 的人工智能边缘设备存算一体宏单元。《Nature Electronics》，4(1):81–90，2021年。

Yang, J. J., Strukov, D. B. 和 Stewart, D. R. 用于计算的忆阻器件。《自然纳米技术》，8(1):13，2013年。Yao, P., Wu, H., Gao, B., Tang, J., Zhang, Q., Zhang, W., Yang, J. J. 和 Qian, H. 完全硬件实现的忆阻器卷积神经网络。《自然》，577 (7792):641–646，2020年。Zahoor, F., Azni Zulkifli, T. Z. 和 Khanday, F. A. 阻变随机存取存储器（RRAM）：材料、开关机制、性能、多级单元（MLC）存储、建模及应用概述。《纳米尺度研究快报》，15(1):1–26，2020年。

A 附录

A.1 深度神经网络训练仿真细节

此处介绍图中所做的深度神经网络训练模拟的详细信息。7。

A.1.1 三层全连接/ MNIST

这里我们使用 MNIST 数据集上的三层全连接深度神经网络，其采用 Sigmoid 激活函数，隐藏层大小分别为 255 和 127，如（Gokmen & Vlasov，2017年）中所述。对于模拟矩阵-向量乘法（前向传播和反向传播），我们基本采用 AIHWKIT 的标准设置，其中包括输出噪声（量化 bin 宽度的 0.5%）、量化和截断（输出范围设为 20，输出噪声设为 0.1，输入和输出量化为 8 位）。其中使用了（Gokmen 等人，2017年）中所述的噪声和边界管理技术。为了将输出范围调整到深度神经网络的范围，我们在每个交叉开关阵列输出之后添加一个可学习的（标量）浮点因子，并使用标准动量随机梯度下降（动量项为 0.9）对其进行更新。

我们训练 80 个训练轮次，并将学习率分为 3 个阶段进行调整（分别为 35、35 和 10 个训练轮次），衰减系数为 0.1。初始学习率设为 0.05（浮点运算时为 0.1），批量大小设为 10。此外，我们设定 $n_s = 1$ 和$\gamma_0 = 10000$，并通过一次测试运行确认这是合理的设置。随后，我们运行模拟网格，并报告最后 3 个训练轮次的平均测试准确率。每种条件下使用相同的网格，即模拟以下设置：$\eta = 0.05$、0。1、0。2、0。5、1。0、2。0。图中报告了每种条件下的最佳测试误差（最后 3 个训练轮次的平均值）。状态数按图中所示进行变化（见图 7；参数 δ_w）。其他参数的设置如图 3所示。斩波器概率设为 $\rho = 0.1$；按照算法中的说明，c-TTv2 使用随机切换，AGAD 使用常规切换（分别见算法 2 和算法 3）。

对于图 7 A，我们还针对每种条件调整了学习率，并从两个 λ =0.05,0.025中选取效果较好的一个。因此，仅此子图就进行了 432 次训练仿真。

A.1.2 LeNet / MNIST

本文采用了 LeNet5（LeCun 等人，1998）模型架构的一种变体，该模型包含 2 个卷积层、2 个最大池化层和 2 个全连接层，并在 MNIST 数据集上训练。原始 LeNet5 模型的架构包含 3 个卷积层。模型训练采用与上述相同的 AIHWKIT设置，用于模拟矩阵-向量乘法（前向传播和反向传播），并使用上文（第 A.1.1节）所述的相同器件模型。

模型使用批量大小 8 和 $\gamma_0 = 10000$ 训练 60 个训练轮次。针对 $\sigma_r = 0.0$，测试了不同的学习率 λ，即 0.01、0.02、0.03、0.04 和 0.05；对所考虑的每种方法，均选取测试准确率最佳时对应的学习率，最终 TTv2、c-TTv2 和 AGAD 的学习率分别为 0.03、0.04 和 0.04。学习率分为两个阶段进行调度（45 和 15 个训练轮次），衰减因子为 0.10。斩波器概率（针对 c-TTv2 和 AGAD）设为 $\rho = 0.1$；c-TTv2 使用随机切换，AGAD 使用常规切换。

η 和 n_s 的初始值分别设为 1 和 1。首先，调整 η 的值，直到获得最佳结果。之后，再调整 n_s 的值。建议采用这一调整顺序，因为与 n_s相比，η 对模型测试准确率的影响更大。对于所有方法，均选用能取得最佳结果的这组超参数值。我们发现，所有方法中 $n_s = 1$ 效果最佳，并将 TTv2、c-TTv2 和 AGAD 的 η 分别设为 0.05、0.1 和 0.075。所报告的测试准确率是最后三个训练轮次后所得测试误差的平均值。

上述得到的 η 和 n_s 超参数随后用于获取 σ_r =0。05、0。1、0。25、0。5、1。0 对应的测试准确率，并用得到的测试准确率生成图7 C。

A.1.3 长短期记忆网络 /《战争与和平》

我们使用一个长短期记忆网络，该网络由 2 个堆叠的长短期记忆网络模块组成，隐藏向量维度为 64，后接一个全连接层，如（Gokmen 等人，2018年）中所述。我们在《战争与和平》（WP）小说上训练了该网络。数据集分为训练集和测试集，分别包含 2,933,246 个字符和 325,000 个字符，词汇表总计包含 87 个字符。

我们使用 AIHWKIT 训练该网络 100 个训练轮次，并将学习率安排为 3 个阶段（50、40 和 20 个训练轮次），衰减因子为 0.1。初始学习率设为 0.1，批量大小设为 16，序列长度设为 100，$\gamma_0 = 10000$ 斩波器概率（针对 c-TTv2 和 AGAD）设为 $\rho = 0.1$；c-TTv2 使用随机切换，AGAD 使用常规切换。

我们从参考偏移变化的条件开始进行多次仿真 $\sigma_r = 0$。我们探索不同的学习率 $\lambda = 0.05$、0。1、0。75，以及 $n_s = 1$的不同设置，3 以及 $\eta = 0.05$的不同设置：0。1、0。5、1。0，并为每种算法选取参数的最佳组合。使用找到的参数后，我们针对各种参考偏移变化进行仿真，并取最后一个训练轮次的损失，生成图 7D。

A.2 补充图

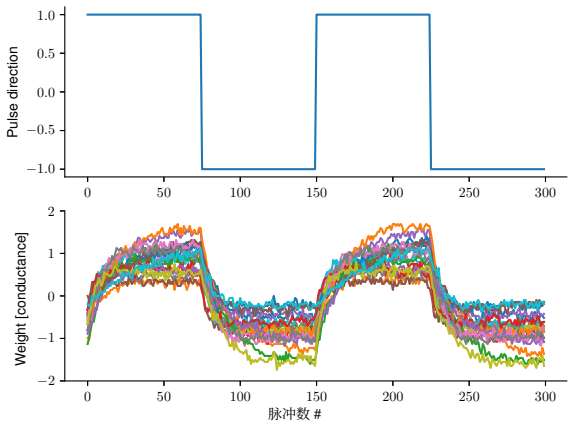


图 A.1. 所用模拟材料的响应轨迹（$n_{\text{states}} = 20$）。上图显示脉冲模式。下图显示 20 个器件的响应（不同颜色）。请注意，存在显著的不对称性、器件间差异和周期间差异。

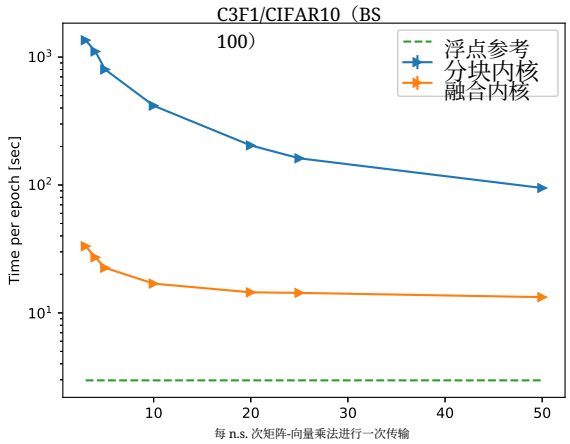


图 A.2. 我们为 TTv2 设计的改进版 CUDA 内核（“融合内核”）的运行时间，与开源工具包 AIHWKit中现有的“分块内核”进行比较。融合内核也用于我们提出的新算法 c-TTv2 和 AGAD。此处展示了标准小型卷积神经网络在 CIFAR10 数据集（Krizhevsky 等人，2009）上完成一个训练轮次所需的时间。针对不同的 n_s 参数评估运行时间（参见图2）。请注意，我们的融合设计大幅提升了速度，运行时间从 > 1000 秒缩短至最佳情况下仅需30 秒。

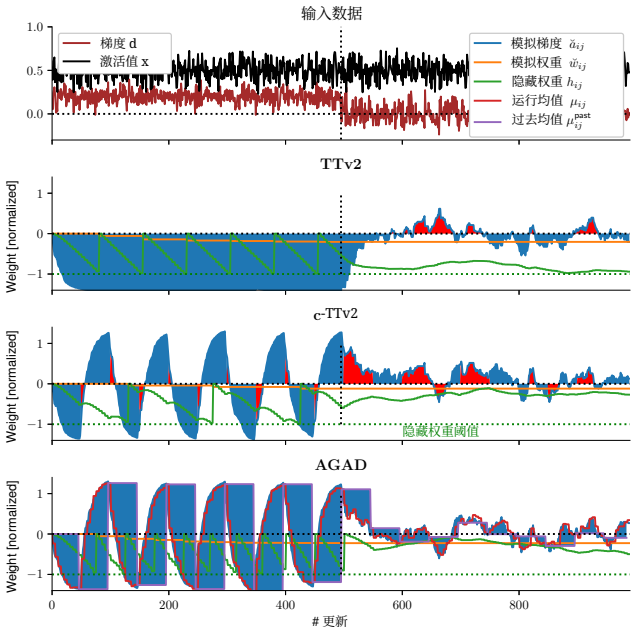


图 A.3. 关闭梯度输入后，如果输入波动较大，TTv2 的梯度矩阵 $\check{A}$ 会发生衰减。我们的新算法也能在这种情况下正确停止权重更新。参数：同图3。