Parallel Spiking Neurons with High Efficiency and Ability to Learn Long-term Dependencies

具有高效率和长期依赖学习能力的并行脉冲神经元

方维 [1,2], 余肇飞 [4], 周昭坤 [3,2], 陈鼎 [5,2], 陈彦骐 [1,2],
马征宇 [2*], **Timothée Masquelier**[6], 田永鸿 [1,2,3*]

[1] 北京大学计算机学院，中国
[2] 鹏城实验室，中国
[3] 北京大学深圳研究生院信息工程学院，中国
[4] 北京大学人工智能学院，中国
[5] 上海交通大学计算机科学与工程系，中国
[6] 脑与认知研究中心（CERCO），法国国家科学研究中心 UMR5549 - 图卢兹第三大学，法国

Abstract

脉冲神经网络 (SNN) 中的普通脉冲神经元使用电荷-发射-重置神经元动力学，只能连续模拟，并且很难学习长时间依赖性。我们发现，当移除重置时，神经元动力学可以以非迭代形式重新表述并并行化。通过重写神经元动力学而不重置为通用公式，我们提出了并行脉冲神经元（PSN），它生成独立于其前身的隐状态，从而产生可并行的神经元动力学和极高的模拟速度。PSN 中输入的权重是完全连接的，这最大化了时间信息的利用。为了避免使用未来的输入进行逐步推理，可以屏蔽 PSN 的权重，从而得到屏蔽的 PSN。通过基于掩码 PSN 跨时间步共享权重，提出了滑动 PSN 来处理不同长度的序列。我们在仿真速度和时态/静态数据分类方面评估了 PSN 系列，结果表明 PSN 系列在效率和准确性方面具有压倒性优势。据我们所知，这是第一个关于并行化脉冲神经元的研究，可以成为脉冲深度学习研究的基石。我们的代码可在 `https://github.com/fangwei123456/Parallel-Spiking-Neuron` 处获取。

*通讯作者

1 简介

脉冲神经网络 (SNN) 是下一代 [1] 人工神经网络 (ANN)，使用生物神经系统的较低抽象。脉冲神经元是 SNN 的关键组件，它处理具有复杂神经元动力学的输入电流，并在膜电位达到阈值时将激发脉冲作为输出。SNN 使用离散脉冲在层之间进行通信，这使得事件驱动的计算范例成为可能，并且在 True North[2]、Loihi [3] 和 Tianjic [4] 等类脑芯片中具有极高的功效。由于其高度的生物学合理性，SNN 已被神经科学家视为分析、模拟和学习生物系统 [5, 6] 的有用工具。随着深度学习方法 [7, 8, 9, 10, 11, 12] 的引入，SNN 在实际任务上的性能得到了极大的提高，并且扩展了 SNN 的应用范围 [13, 14]。作为神经科学和计算科学之间的桥梁，SNN 近年来引起了越来越多的研究兴趣。

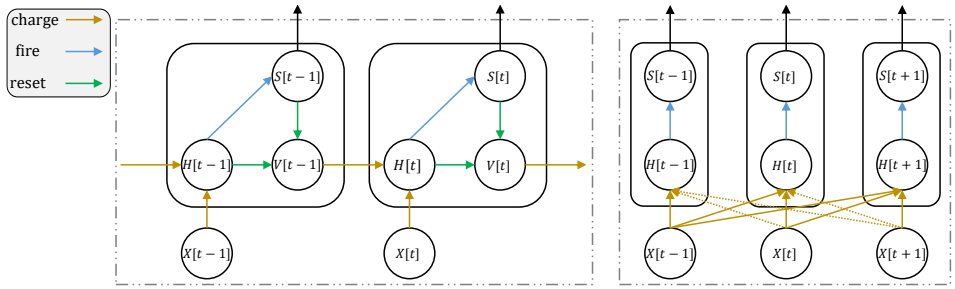
脉冲的二元特性导致大量信息丢失，这是导致 SNN 准确率低于 ANN 的主要因素。在弥合性能差距的努力中，脉冲神经元的改进是一种通用方法，并已被广泛探索 [15, 16, 17, 18, 19, 20, 21]。然而，之前的工作仅限于图1（a）所示的现有串行充电-点发放-复位计算范式，其仿真速度慢且难以学习长期依赖。

在本文中，我们提出了并行脉冲神经元 (PSN) 及其变体、掩蔽 PSN 以及滑动 PSN (SPSN)。图1(b) 显示了 PSN 的计算图，它使用非前驱依赖方法来生成隐状态，从而实现可并行的神经元动力学和极高的模拟速度。由于从输入到 $H[t]$ 的连接是直接的，因此 PSN 比其他普通脉冲神经元具有更好的长期依赖学习能力。为了在 $X[t]$ 到达时立即生成 $H[t]$，可以屏蔽来自 $X[i], i \geq t+1$ 的连接，如图1（b）中的虚线所示，然后神经元成为屏蔽的 PSN。masked PSN 的参数也可以设置为时不变的，从而导出滑动 PSN，并且对于可变长度的输入序列更加灵活。本文的主要贡献如下：

1） 我们分析了从广泛使用的充电-发射-复位神经元动力学中去除复位的影响，表明脉冲神经元可以在不复位的情况下并行化。

2） 通过重写神经元动力学而不重置为通用公式，我们获得了具有完全可并行化神经元动力学的 PSN。对于逐步串行前向和可变长度序列处理，还导出了掩蔽 PSN 和滑动 PSN。

3） 我们比较了 PSN 家族和 vanilla 脉冲神经元的模拟速度，显示出并行神经元动力学具有压倒性的性能优势。我们在顺序、静态和类脑数据分类任务上评估 PSN 系列，并获得比以前的脉冲神经元更高的准确性。



(a) The vanilla spiking neuron (b) The parallel spiking neuron

图 1: 普通脉冲神经元和并行脉冲神经元的计算图。图 (a) 引自 [15]。图（b）中的虚线是可以屏蔽以进行逐步计算的权重。$X[t], S[t]$ 是输入电流和输出脉冲，$H[t], V[t]$ 是时间步 t 的隐状态。

2 相关工作

2.1 脉冲神经网络的深度学习

ANN 到 SNN 转换 (ANN2SNN) [22, 23, 24, 25, 26] 和代理训练 [27, 28] 是 SNN 的两种主要深度学习方法。ANN2SNN 方法使用 SNN 中的激发率来近似 ANN 中的激活，方法是将权重从 ANN 转换为 SNN，并进行额外的归一化以减少误差。它在 ImageNet 数据集 [29] 中实现了与 ANN 接近的性能，但需要许多时间步来估计准确的发放率。代理训练方法通过平滑且可微的代理函数重新定义了脉冲生成过程中使用的 Heaviside 函数的梯度。利用代理梯度，可以通过时间反向传播 (BPTT) 和梯度下降来训练 SNN。替代训练方法不像 ANN2SNN 那样基于速率编码，因此时间步数更少。然而，在训练过程中使用 BPTT 比 ANN2SNN 需要更多的训练时间和内存使用。

2.2 脉冲神经元的改进

受生物机制或人工神经网络技术启发，改进脉冲神经元是一种实用的方法。参数泄漏集成激发 (PLIF) 脉冲神经元 [15] 结合了可学习的膜时间常数，能够将突触权重和神经元动力学的学习结合起来。门控 LIF (GLIF) 脉冲神经元 [16] 添加了可学习门来融合不同的积分、衰减和重置方法，从而扩大了表示空间并增加了脉冲神经元的异质性和适应性。基于 k 的 Leaky Integrate-and-Fire (KLIF) 神经元 [18] 添加了一个可学习的缩放因子来调整替代梯度，并应用 ReLU 激活来避免负膜电位。多级激发（MLF）单元 [19] 包含具有不同级别阈值的 LIF 神经元，缓解了梯度消失问题并提高了表达能力。蓬吉兰等人。[20] 通过添加输入门、遗忘门、循环连接和多位输出来改进脉冲神经元，从而减少实际梯度和代理梯度之间的不匹配。

2.3 训练方法和网络结构的改进

除了脉冲神经元之外，修改训练技术和网络结构也是提升 SNN 性能的有效方法。

归一化　阈值相关批归一化（TDBN）[30] 对时间和空间维度上的特征进行归一化，并消除阈值引入的方差，这是比批归一化 [31] 更适合 SNN 的归一化方法。Batch Normal-

ization Through Time (BNTT) [32] 沿着时间步解耦 BN 并使用时间参数，从而更精确地估计时间变化输入的分布。时间有效批归一化 (TEBN) [33] 在每个时间步上添加可学习的仿射变换，并且具有更好的捕获时间分布的能力。

训练技术 深度连续局部学习（DECOLLE）[34] 利用局部损失来避免 BPTT，实现 SNN 的在线学习。通过时间进行在线训练（OTTT）[35] 通过随时间步更新资格轨迹来近似具有时间依赖性的迭代梯度，这也实现了在线训练，并且只需要与时间步无关的持续训练内存。稀疏脉冲梯度下降 [36] 仅使用膜电位在给定范围内的神经元进行反向传播，从而实现了反向传播的加速和记忆减少。膜电位 [37] 的注意力机制消除了冗余信息，降低了发放率，并减少了过度拟合。Dspike [38] 使用有限差分来估计梯度，并且比普通代理梯度方法具有更高的精度。时间高效训练（TET）[39] 计算每个时间步的损失并对它们进行平均，而不是使用平均输出来计算损失，这提高了 SNN 的时间可扩展性。

网络结构 Spike-Element-Wise ResNet [40] 解决了 sigmoid-like 代理函数引起的 plain Spiking ResNet 的梯度消失/爆炸问题，成功训练了第一个超过 150 层的深度 SNN。Spikformer [41] 将 Transformer [42] 中基于 softmax 的自注意力修改为基于脉冲的公式，这适用于 SNN，并在 ImageNet 数据集上实现了最先进的精度。

2.4 序列处理加速

使用图形处理单元 (GPU) [43] 进行大规模并行计算是推动深度学习研究的关键因素之一。然而，SNN 和循环神经网络（RNN）等循环结构的串行计算特性无法充分发挥 GPU 的并行计算能力。为了加速序列处理，基于卷积的方法 [44, 45] 放弃了循环结构并采用卷积层，这些层在 GPU 上训练期间完全并行化。门控脉冲线性循环网络 [46] 是另一种使用并行前缀和（扫描）算法 [47] 来并行化线性循环的解决方案。Transformer [42] 用自注意力和位置编码代替循环，比循环和卷积编码器-解码器结构实现更快的训练速度。

3 方法

在本节中，我们介绍去除神经元重置的动机、将没有重置的脉冲神经元推广为 PSN 的想法，以及掩蔽 PSN 和滑动 PSN 的推导。请注意，我们使用 X 等常规字母表示标量，使用 $\boldsymbol{X}$ 等粗体字母表示张量。

3.1 普通脉冲神经元 W/WO 重置

SNN 中的脉冲神经元具有丰富的神经元动力学，这赋予了 SNN 时间信息处理能力。在大多数情况下，脉冲神经元的行为可以通过三个离散时间方程 [15] 来描述：

$$H[t] = f(V[t-1], X[t]), \tag{1}$$

$$S[t] = \Theta(H[t] - V_{th}), \tag{2}$$

$$V[t] = \begin{cases} H[t] \cdot (1 - S[t]) + V_{reset} \cdot S[t], & \text{hard reset} \\ H[t] - V_{th} \cdot S[t], & \text{soft reset} \end{cases}, \tag{3}$$

其中 $X[t]$ 是输入电流，$H[t]$ 是充电后但发射前的膜电位，$V[t]$ 是发射后的膜电位，$S[t]$ 是时间步长 t 的输出脉冲。式 (2) 中的 V_{th} 为阈值电位，式 (3) 中的 V_{reset} 为复位电位。$\Theta(x)$ 是 Heaviside 阶跃函数，$\Theta(x) = 1$ 是所有 $x \geq 0$ 的函数，否则是 $\Theta(x) = 0$。Eq.(1) 是神经元充电方程，f 特定于不同的脉冲神经元。充电后，$H[t]$ 将与 V_{th} 进行比较并确定是否发射脉冲，这由方程（2）描述。发射后，膜电位将重置，如方程（3）所示。请注意，重置有两种，即硬重置和软重置。如果神经元触发，硬重置会将 $V[t]$ 设置为 V_{reset}，而软重置会将 $V[t]$ 减少 V_{th}。硬重置广泛用于代理训练，以在实验 [48] 中观察到更好的性能，而软重置在 ANN2SNN 中是首选，以降低转换误差。在模拟 SNN 时，采用了以下式（1）-（3）在时间步长上的迭代过程，其时间复杂度为 $\mathcal{O}(T)$，其中 T 是时间步数。

对于常用的脉冲神经元，Eq.(1) 是线性的，如果我们在某些情况下可以忽略 Eq.(3)，则可以将 Eq.(1) 重新表述为非迭代方程，例如，在一段时间内所有 t 的亚阈值状态 $H[t] < V_{th}$ 中。那么 $V[t] = H[t]$ 就是 t，在这种情况下我们只会使用 $H[t]$。更具体地说，我们以积分激发（IF）神经元和泄漏积分激发（LIF）神经元为例。为简单起见，假设两种类型的神经元均为 $H[-1] = 0$，LIF 神经元的静息电位为 0。IF 神经元的神经元电荷方程或 Eq.(1) 为

$$H[t] = H[t-1] + X[t]. \tag{4}$$

方程（4）可以很容易地重新表述为

$$H[t] = \sum_{i=0}^{t} X[i]. \tag{5}$$

LIF 神经元的神经元电荷方程为

$$H[t] = (1 - \frac{1}{\tau_m}) \cdot H[t-1] + \frac{1}{\tau_m} \cdot X[t], \tag{6}$$

其中 τ_m 是膜时间常数。方程（6）也可以重新表述为

$$H[t] = \frac{1}{\tau_m} \cdot \sum_{i=0}^{t} (1 - \frac{1}{\tau_m})^{t-i} \cdot X[i]. \tag{7}$$

使用非迭代方程，可以并行计算所有时间步长的 $H[t]$ 或 $\boldsymbol{H} = \{H[0], H[1], ..., H[T-1]\}$。对于累加和运算得到的每个 $H[t]$，使用 Parallel Prefix Sum (Scan) 算法时，时间复杂度可以低至 $\mathcal{O}(\log(t)) \leq \mathcal{O}(\log(T))$。当给定 $\boldsymbol{H}$ 时，逐元素运算 Eq.(2) 也可以应用于整个

$\boldsymbol{H}$，并且在支持并行计算的设备 (例如 CUDA 设备) 中时间复杂度为 $\mathcal{O}(1)$。因此，当我们可以忽略神经元重置时，神经元充电和放电的整个时间复杂度降低到 $\mathcal{O}(\log(T))$。

那么，我们如何才能忽略所有时间步长的神经元重置呢？一种可能的方法是设置 $V_{th} = +\infty$，使 Eq.(3) 成为恒等函数。然而，这种情况下神经元只能输出 0，这是没有意义的。另一种方法是直接从神经元动力学中删除神经元重置，这意味着仅使用方程（1）和方程2。这种粗略而简单的方法可能会引起进一步的担忧，即缺乏神经元重置将导致神经元不间断地放电。请注意，发放率由输入、充电、阈值和重置决定。因此，仅移除重置对发放率的影响是不确定的。另外，下一节的实验结果将表明不会出现不间断发射的问题。

3.2 并行脉冲神经元

去除神经元重置后，脉冲神经元的充电方程可以表示为非迭代方程，并且 $H[t]$ 的解变成累积和运算。可以发现，式 (5) 和式 (7) 是 $X[i]$ 线性组合的两种具体情况。更具体地说，我们将一般线性组合表述为

$$H[t] = \sum_{i=0}^{T-1} W_{t,i} \cdot X[i], \tag{8}$$

其中 $W_{t,i}$ 是 $X[i]$ 和 $H[t]$ 之间的权重。那么，$W_{t,i} = \Theta(t-i)$ 为没有复位的 IF 神经元，$W_{t,i} = \frac{1}{\tau_m}(1-\frac{1}{\tau_m})^{t-i} \cdot \Theta(t-i)$ 为没有复位的 LIF 神经元。基于上述分析，我们提出了并行脉冲神经元（PSN），其神经元动力学如下：

$$\boldsymbol{H} = \boldsymbol{W}\boldsymbol{X}, \qquad \boldsymbol{W} \in \mathbb{R}^{T\times T}, \boldsymbol{X} \in \mathbb{R}^{T\times N} \tag{9}$$

$$\boldsymbol{S} = \Theta(\boldsymbol{H} - \boldsymbol{B}), \qquad \boldsymbol{B} \in \mathbb{R}^{T}, \boldsymbol{S} \in \{0,1\}^{T\times N} \tag{10}$$

其中 $\boldsymbol{X}$ 是输入序列，$\boldsymbol{W}$ 是可学习权重矩阵，$\boldsymbol{H}$ 是隐状态序列，$\boldsymbol{B}$ 是可学习阈值，$\boldsymbol{S}$ 是二进制输出脉冲序列。N 是批量大小的数量，T 是时间步的数量。请注意，$\boldsymbol{W}$ 是一个 $T \times T$ 矩阵，这表明 $H[t]$ 可以直接整合所有时间步的信息，而不是最后一两个时间步的信息。因此，PSN 是一个 T 阶神经元。不使用迭代方程，PSN 的计算是完全并行的。PSN 的时间复杂度由方程 (9) 决定，它是矩阵-矩阵乘法。PSN 的模拟比普通脉冲神经元的模拟要快得多，因为矩阵-矩阵乘法在 Intel MKL 和 cuBLAS 等线性代数库中进行了高度优化。BPTT 在训练期间仍然使用，但也是并行的，而不是串行的。

3.3 k-阶屏蔽并行脉冲神经元

PSN 的高阶特性也是一把双刃剑。只有当所有 $X[t]$ 都到达时才能生成输出序列 $\boldsymbol{S}$，这表明每个脉冲神经元层都会增加整个 SNN 的延迟 T。在一些首次峰值时间 SNN [49, 50, 51] 上也报告了此类延迟问题。请注意，虽然延迟是不可避免的，但吞吐量仍然可以接近具有多级管道 [49] 的普通 SNN。为了解决 PSN 的延迟问题，我们在生成 $\boldsymbol{H}$ 之前添加一个掩码 $\boldsymbol{M}_k$ 来逐个乘以 $\boldsymbol{W}$，并提出 k 阶掩码 PSN，其神经元充电方程为

$$\boldsymbol{H} = (\boldsymbol{W} \cdot \boldsymbol{M}_k)\boldsymbol{X}, \qquad \boldsymbol{W} \in \mathbb{R}^{T\times T}, \boldsymbol{M}_k \in \mathbb{R}^{T\times T}, \boldsymbol{X} \in \mathbb{R}^{T\times N} \tag{11}$$

其中 $\boldsymbol{M}_k$ 定义为

$$\boldsymbol{M}_k[i][j] = \begin{cases} 1, & j \leq i \leq j+k-1 \\ 0, \text{otherwise} \end{cases}. \tag{12}$$

使用掩码 $\boldsymbol{M}_k$，$H[t]$ 仅取决于最新的 k 输入 $\{X[t-k+1], ..., X[t-1], X[t]\}$（我们假设所有 $i<0$ 都为 $X[i]=0$），一旦收到 $X[t]$，就可以计算 $S[t]$ 并将其发送到下一层。

在训练 k 阶掩码 PSN 时，采用渐进掩码方法，首先使用全 1 矩阵 $\mathbf{1}$ 作为掩码，并逐渐用 $\boldsymbol{M}_k$ 代替 $\mathbf{1}$。更具体地说，我们使用

$$\boldsymbol{M}_k(\lambda) = \lambda \cdot \boldsymbol{M}_k + (1-\lambda) \cdot \mathbf{1} \tag{13}$$

作为掩码，训练期间 λ 从 0 增加到 1。在训练的早期阶段，$\boldsymbol{M_k}(\lambda) \approx \mathbf{1}$，表明可以利用未来的信息来提供合适的主要参数并帮助网络收敛。

3.4 k-阶滑动并行脉冲神经元

PSN 和掩码 PSN 的参数是时间相关的，这需要额外的操作来处理可变长度的序列。为了解决这个问题，我们修改了跨时间步共享的参数，并提出了 k 阶滑动 PSN，其神经元动力学为

$$H[t] = \sum_{i=0}^{k-1} W_i \cdot X[t-k+1+i], \tag{14}$$

$$S[t] = \Theta(H[t] - V_{th}), \tag{15}$$

其中 $\boldsymbol{W} = [W_0, W_1, ..., W_{k-1}] \in \mathbb{R}^k$ 是可学习权重，$X[j]=0$ 代表所有 $j<0$，V_{th} 是可学习阈值。与屏蔽 PSN 类似，滑动 PSN 的 $H[t]$ 也仅取决于最新的 k 输入。更具体地说，权重在输入上滑动并生成隐状态，这是标准的一维卷积运算。另外，还可以通过矩阵-矩阵乘法来实现。当输入序列 $\boldsymbol{X} \in \mathbb{R}^{T \times N}$ 到达并且其长度 T 已知时，矩阵 $\boldsymbol{A} \in \mathbb{R}^{T \times T}$ 可以生成为

$$\boldsymbol{A}[i][j] = \begin{cases} W_{k-1-i+j}, & i+1-k \leq j \leq i \\ 0, \text{otherwise} \end{cases}, \tag{16}$$

那么 $\boldsymbol{H} = \boldsymbol{A}\boldsymbol{X}$。根据我们的实验结果，使用矩阵-矩阵乘法比使用卷积更快。

3.5 PSN 家族总结

作为总结，图 2 显示了 PSN 系列的参数数量 n_{param} 和隐状态生成的比较。值得注意的是，在深度 SNN 中使用 PSN 会添加可忽略不计的参数数量，因为 T 在直接训练的 SNN 中很小。例如，使用带有 $T=4$ 的 PSN 将在脉冲 ResNet-18 和 VGG-11 中添加 340 和 200 个参数，这只会增加原始 SNN 的 0.00291% 和 0.00015% 参数。此外，PSN 系列只有一个隐状态 $\boldsymbol{H}$，这使得它们在训练过程中比具有至少两个隐状态 $\boldsymbol{H}, \boldsymbol{V}$ 的普通脉冲神经元消耗更少的内存，这一点通过补充材料中的实验进行了分析和验证。

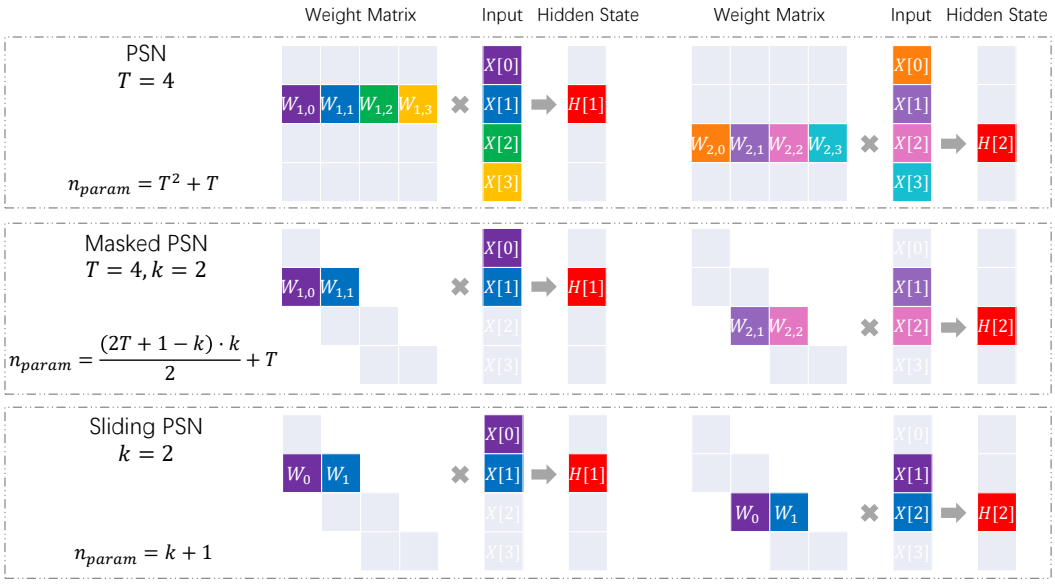


图 2: PSN 家族在参数数量 n_{param} 和隐状态生成上的比较。

4 实验

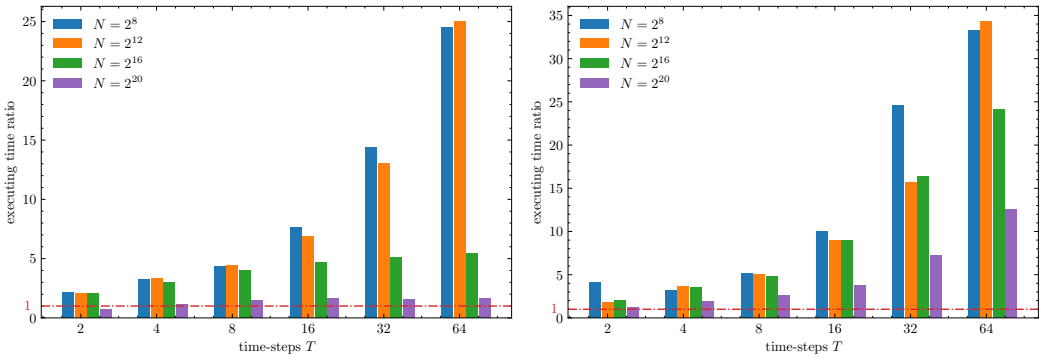
在本节中，我们报告在仿真性能和时态/静态数据分类精度方面评估 PSN 系列的实验结果。所有实验均基于 SpikingJelly [52] 框架。培训的详细信息在补充材料中提供。

4.1 模拟速度基准

我们评估了 PSN 和 LIF 神经元的模拟速度基准，它广泛用于深度 SNN，并充当普通脉冲神经元的基线。请注意，屏蔽 PSN 和滑动 PSN 的实现和模拟速度几乎与 PSN 相同。因此，我们以 PSN 作为 PSN 家族的基准。该基准测试以两种模式运行，即推理和训练。在推理中，我们使用 PyTorch 即时 (JIT) 编译将所有时间步长的 LIF 神经元的 Eq.(1)-(3) 融合到单个函数中，从而避免过多微小 CUDA 内核的调用开销来加速。然而，PyTorch JIT 不支持修改融合前向函数的后向，因此 Eq.(2) 后向中使用的代理方法破坏了训练中所有三个方程的融合。因此，我们必须在训练中的 LIF 神经元的每个时间步使用三个较小的 JIT 函数。对于 PSN，推理和训练的实现没有区别，都涉及矩阵-矩阵乘法和逐元素运算。我们测试了神经元编号 $N = 2^8, 2^{12}, 2^{16}, 2^{20}$ 和时间步编号 $T = 2, 4, 8, 16, 32, 64$，它们是深度 SNN 的典型选项。将两个神经元的执行时间分别表示为 t_{PSN} 和 t_{LIF}，则比率 $\frac{t_{LIF}}{t_{PSN}}$ 如图3所示。可以发现，在大多数情况下，PSN 的模拟比 LIF 神经元的模拟要快得多。

4.2 学习长期依赖

为了验证不同脉冲神经元的长期依赖的学习能力，我们评估了它们在分类顺序 CIFAR10 和 CIFAR100 上的性能。在这些任务中，图像将被逐列发送到 SNN，这类似于人类从左到右阅读的方式。顺序图像分类任务广泛用于评估网络的长期依赖学习能力。网络



(a) Forward (inference)　　(b) Forward and backward (training)

图 3: LIF 神经元和 PSN 在推理中的一个前向和在训练中的一个前向和后向的执行时间比率。

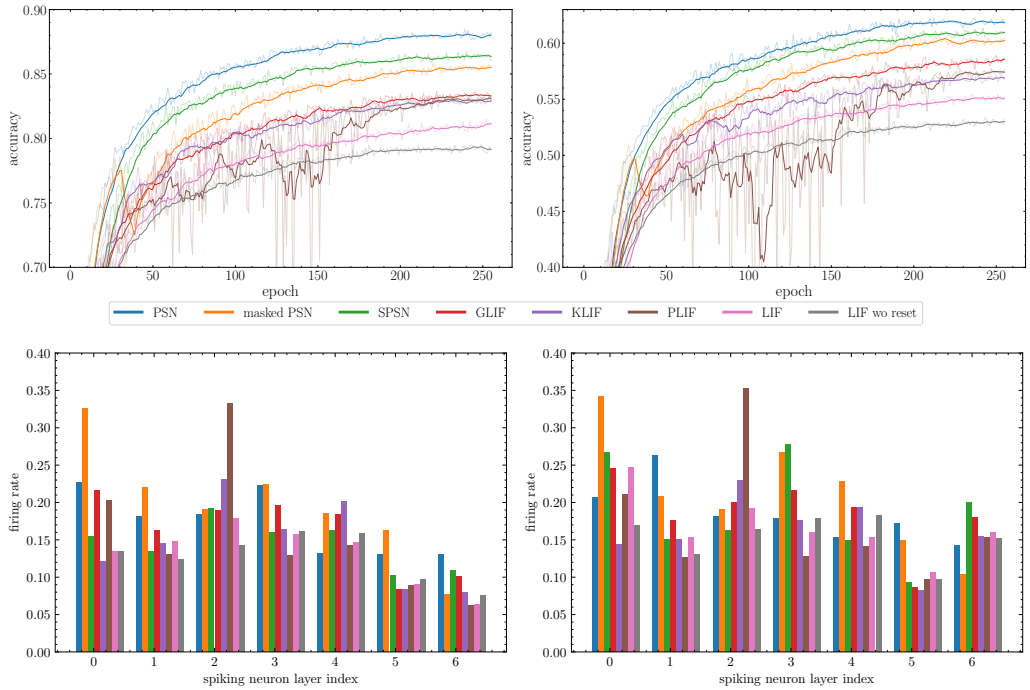
Dataset \ Neuron	PSN	masked PSN	sliding PSN	GLIF[16]	KLIF[18]	PLIF[15]	LIF	LIF wo reset
Sequential CIFAR10	88.45	85.81	86.70	83.66	83.26	83.49	81.50	79.50
Sequential CIFAR100	62.21	60.69	62.11	58.92	57.37	57.55	55.45	53.33

表 1: 不同脉冲神经元在顺序 CIFAR 上的测试精度（%）。

结构在 [15] 的基础上进行了修改，用 1D 卷积层和平均池化层替换 2D 卷积/最大池化层，并删除了投票层。由于是逐列输入，T 就是 32，也就是图像的宽度。比较的神经元包括 PSN、32 阶屏蔽 PSN、32 阶滑动 PSN、具有通道参数的 GLIF、KLIF、PLIF 和带/不重置的 LIF 神经元。我们对不同的神经元使用通过消融实验确定的最佳重置选项，即对 PLIF 和 LIF 神经元使用软重置和分离重置 [53]，对 KLIF 神经元使用软重置。GLIF 神经元使用可学习的门来控制所有选项，不需要手动设置。对于屏蔽的 PSN，$\lambda = \min(1, 8 \cdot epoch/(epochs - 1))$，其中 $epoch$ 表示当前训练时期，$epochs$ 表示时期总数。

Tab.1 显示了顺序 CIFAR 上所有神经元的准确性。可以发现，准确率的排序为 PSN > slider PSN > masked PSN > GLIF > PLIF > KLIF > LIF > LIF wo Reset，符合预期。PSN 利用所有时间步的信息，具有最高的学习能力，但它可能会因比较不公平而受到批评，因为它可以一次读取所有列。因此，屏蔽 PSN 和滑动 PSN 的准确性更有说服力，因为它们仍然可以以逐步模式运行。由于引入了高阶信息，它们比任何其他传统的脉冲神经元都工作得更好。同时，GLIF 神经元是传统脉冲神经元中最好的，这表明融合不同类型的充电、衰减和重置可以提高学习能力。没有重置的 LIF 神经元的准确度低于 LIF 神经元，这表明直接去除重置会降低普通脉冲神经元的学习能力。因此，有必要使用高阶信息作为 PSN 系列来解决这种下降问题。

为了进行清晰的比较，我们绘制了具有不同脉冲神经元的经过训练的 SNN 的每个脉冲神经元层的训练精度曲线和发放率，如图 4 所示。这些精度曲线是 8 个 epoch 的移动平均值，浅色曲线显示了原始精度。PSN 家族的曲线几乎总是高于其他曲线，这清楚地表明了它们在收敛速度和最终精度上的优越性能。请注意，在 epoch 32 左右，掩码



(a) Sequential CIFAR10　　(b) Sequential CIFAR100

图 4: 序列 CIFAR 分类的不同神经元的准确率曲线和发放率。

PSN 的曲线急剧下降，这是由于掩码完全变成二元造成的。发放率表明，随着神经元重置的去除，不会发生不间断的放电。一般来说，PSN 和屏蔽 PSN 比其他 PSN 更容易被激活，并且表现出更高的发放率。同时，未重置的滑动 PSN 和 LIF 神经元的发放率略高于普通脉冲神经元。因此，我们得出结论，取消重置确实会增加发放率，但增加程度取决于神经元的结构。另一方面，考虑到模拟效率和长期依赖学习能力的巨大提升，稍微高一些的发放率是可以接受的。

图 5 显示了蒙版 PSN 和滑动 PSN 在连续 CIFAR100 数据集上的准确度-阶次曲线，其中最高准确度用红色 ★ 标记。可以看出，当阶数从 1 增加到 4 时，精度提升很快。当我们继续增加阶数时，改进是微乎其微的，准确性甚至会下降。最高的准确度是通过较大但不是最大的阶数获得的，对于掩码 PSN 和滑动 PSN 分别为 20 和 31。但大多数情况下，用 $k = T$ 就可以保证高精度。值得注意的是，滑动 PSN 的曲线几乎总是高于掩模 PSN 的曲线，并且当 $k = 21$ 和 $k = 31$ 时优于 PSN，表明时不变参数具有更好的处理时间信息的泛化能力。

4.3 静态和类脑数据分类

静态和类脑数据分类任务也是 SNN 的常见基准。我们在静态 CIFAR10、ImageNet 数据集和类脑 CIFAR10-DVS [54] 数据集上评估了 PSN 系列。静态数据集的时间步数很少，延迟不是主要问题。因此，我们在 CIFAR10 和 ImageNet 中使用 PSN。根据顺序 CIFAR 的结果，滑动 PSN 使用较少的参数但获得较高的精度，我们将其用于 CIFAR10-DVS。结果列于 Tab.2中。

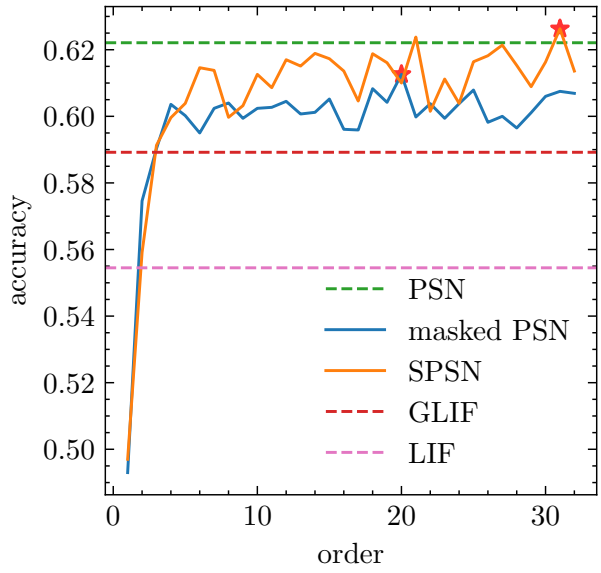


图 5: 顺序 CIFAR100 上的精度阶次曲线。

CIFAR10 我们通过使用平均池化和删除投票层来修改 [15] 或 PLIF 网络的网络结构。我们的 PSN 使用最短的时间步长 $T = 4$ 并达到 95.32% 的最高精度。

ImageNet 我们使用 SEW ResNet [40] 对 ImageNet 数据集进行分类。我们在 SEW ResNet-18/34 上验证了 SPN 的性能，与带有 IF 神经元的原始 SEW ResNet 相比，获得了稳定的 3+% 精度提升。我们的方法达到了 70.54% 的精度，仅次于多一个时间步长的 Dspike 方法和参数数量是我们的 6.3 倍的 VGG-16。

CIFAR10-DVS 我们使用了 [39] 中的 VGG 结构。考虑到 DVS 数据包含时间特征并且 T 比静态数据集大，我们使用 2 阶滑动 PSN。我们使用 $T = 4$ 实现了 82.3 % 精度，这是第一个在如此少的时间步长内获得 80+% 精度的工作。当时间步增加到 $T = 8$ 时，我们实现了 85.30% 的精度，比之前的 SOTA 方法 [33] 高了 2+%，并且我们减少了两个时间步。有了 $T = 10$，精度进一步提高到 85.90%。

Dataset	Method	Spiking Network	Time-steps	Accuracy(%)
CIFAR10	Dspike[38]	Modified ResNet-18	6	94.25
	TET[39]	ResNet-19	6	94.50
	TDBN[30]	ResNet-19	6	93.16
	TEBN[33]	ResNet-19	6	94.71
	PLIF[15]	PLIF Net	8	93.50
	KLIF[18]	Modified PLIF Net	10	92.52
	GLIF[16]	ResNet-19	6	95.03
	PSN	Modified PLIF Net	4	95.32
ImageNet	Dspike[38]	ResNet-34	6	68.19
		VGG-16	5	71.24
	TET[39]	SEW ResNet-34	4	68.00
	TDBN[30]	ResNet-34 with double channels	6	67.05
	TEBN[33]	SEW ResNet-34	4	68.28
	GLIF[16]	ResNet-34	4	67.52
	SEW ResNet[40]	SEW ResNet-18	4	63.18
		SEW ResNet-34	4	67.04
	PSN	SEW ResNet-18	4	67.63
		SEW ResNet-34	4	70.54
CIFAR10-DVS	Dspike[38]	ResNet-18	10	75.40
	TET[39]	VGG	10	83.17
	TDBN[30]	ResNet-19	10	67.80
	TEBN[33]	VGG	10	84.90
	PLIF[15]	PLIF Net	20	74.80
	KLIF[18]	Modified PLIF Net	15	70.90
	GLIF[16]	Wide 7B Net	16	78.10
	SEW ResNet[40]	Wide 7B Net	16	74.40
	sliding PSN ($k = 2$)	VGG	4, 8, 10	82.30, 85.30, 85.90

表 2: PSN 家族与其他方法的比较。

5 结论

在本文中，我们从普通脉冲神经元的神经元动力学中去除重置，并提出 PSN 家族，包括 PSN、屏蔽 PSN 和滑动 PSN。PSN 家族可以并行模拟，这大大加速了深度 SNN 的训练。PSN 系列中输入的权重可以完全连接，也可以通过自定义订单进行屏蔽/共享。与之前的脉冲神经元相比，模拟速度和时间/静态数据分类的实验结果验证了 PSN 系列的效率和准确性。我们的工作可能是一个里程碑，也是现代脉冲神经元的基石。

Acknowledgments and Disclosure of Funding

这项工作得到了国家自然科学基金 (62027804、61825101、62088102 和 62176003)、鹏城实验室重大重点项目 (PCL2021A13) 和国家科学研究所 ANR-20-CE23-0004-04 DeepSee 的资助。

References

[1] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[2] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[3] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[4] Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

[5] Chris Eliasmith, Terrence C. Stewart, Xuan Choo, Trevor Bekolay, Travis DeWolf, Yichuan Tang, and Daniel Rasmussen. A large-scale model of the functioning brain. *Science*, 338(6111):1202–1205, 2012.

[6] Marcel Stimberg, Romain Brette, and Dan FM Goodman. Brian 2, an intuitive and efficient neural simulator. *eLife*, 8:e47314, 2019.

[7] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[8] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural Networks*, 111:47–63, 2019.

[9] Yufei Guo, Xuhui Huang, and Zhe Ma. Direct learning-based deep spiking neural networks: a review. *Frontiers in Neuroscience*, 17:1209795, 2023.

[10] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015.

[11] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11:682, 2017.

[12] Saeed Reza Kheradpisheh and Timothée Masquelier. Temporal backpropagation for spiking neural networks with one spike per neuron. *International Journal of Neural Systems*, 30(06):2050027, 2020.

[13] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[14] Rui Yuan, Qingxi Duan, Pek Jun Tiw, Ge Li, Zhuojian Xiao, Zhaokun Jing, Ke Yang, Chang Liu, Chen Ge, Ru Huang, et al. A calibratable sensory neuron based on epitaxial vo2 for spike-based neuromorphic multisensory system. *Nature Communications*, 13(1):1–12, 2022.

[15] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2661–2671, 2021.

[16] Xingting Yao, Fanrong Li, Zitao Mo, and Jian Cheng. GLIF: A unified gated leaky integrate-and-fire neuron for spiking neural networks. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[17] Ilyass Hammouamri, Timothée Masquelier, and Dennis George Wilson. Mitigating catastrophic forgetting in spiking neural networks through threshold modulation. *Transactions on Machine Learning Research*, 2022.

[18] Chunming Jiang and Yilei Zhang. Klif: An optimized spiking neuron unit for tuning surrogate gradient slope and membrane potential, 2023.

[19] Lang Feng, Qianhui Liu, Huajin Tang, De Ma, and Gang Pan. Multi-level firing with spiking ds-resnet: Enabling better and deeper directly-trained spiking neural networks. In Lud De Raedt, editor, *International Joint Conferences on Artificial Intelligence Organization (IJCAI)*, pages 2471–2477, 7 2022. Main Track.

[20] Wachirawit Ponghiran and Kaushik Roy. Spiking neural networks with improved inherent recurrence dynamics for sequential learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 36, pages 8001–8008, 2022.

[21] Yufei Guo, Xinyi Tong, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Zhe Ma, and Xuhui Huang. Recdis-snn: Rectifying membrane potential distribution for directly

training spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 326–335, June 2022.

[22] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13558–13567, 2020.

[23] Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2328–2336. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.

[24] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[25] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[26] Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(1):11–21, Jun. 2023.

[27] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12, 2018.

[28] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1419–1428, 2018.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[30] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 11062–11070, 2021.

[31] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456. PMLR, 2015.

[32] Youngeun Kim and Priyadarshini Panda. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in Neuroscience*, 15, 2021.

[33] Chaoteng Duan, Jianhao Ding, Shiyan Chen, Zhaofei Yu, and Tiejun Huang. Temporal effective batch normalization in spiking neural networks. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[34] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 14, 2020.

[35] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[36] Nicolas Perez-Nieves and Dan F. M. Goodman. Sparse spiking gradient descent. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[37] Man Yao, Guangshe Zhao, Hengyu Zhang, Yifan Hu, Lei Deng, Yonghong Tian, Bo Xu, and Guoqi Li. Attention spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–18, 2023.

[38] Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[39] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. In *International Conference on Learning Representations (ICLR)*, 2022.

[40] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

[41] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng YAN, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *International Conference on Learning Representations (ICLR)*, 2023.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[43] Rajat Raina, Anand Madhavan, and Andrew Y Ng. Large-scale deep unsupervised learning using graphics processors. In *International Conference on Machine Learning (ICML)*, pages 873–880, 2009.

[44] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

[45] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning (ICML)*, pages 1243–1252. PMLR, 2017.

[46] Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations (ICLR)*, 2018.

[47] Mark Harris, Shubhabrata Sengupta, and John D Owens. Parallel prefix sum (scan) with cuda. *GPU gems*, 3(39):851–876, 2007.

[48] Eimantas Ledinauskas, Julius Ruseckas, Alfonsas Juršėnas, and Giedrius Buračas. Training Deep Spiking Neural Networks. *arXiv preprint arXiv:2006.04436*, 2020.

[49] Seongsik Park, Seijoon Kim, Byunggook Na, and Sungroh Yoon. T2fsnn: deep spiking neural networks with time-to-first-spike coding. In *2020 57th ACM/IEEE Design Automation Conference*, pages 1–6. IEEE, 2020.

[50] Dongwoo Lew, Kyungchul Lee, and Jongsun Park. A time-to-first-spike coding and conversion aware training for energy-efficient deep spiking neural network processor design. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 265–270, 2022.

[51] Lina Bonilla, Jacques Gautrais, Simon Thorpe, and Timothée Masquelier. Analyzing time-to-first-spike coding schemes. *Frontiers in Neuroscience*, 16, 2022.

[52] Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023.

[53] Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *BioRxiv*, 2020.

[54] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, 11, 2017.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[56] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[57] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[58] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[59] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6023–6032, 2019.

[60] Samuel G Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 774–782, 2021.

[61] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, volume 34, pages 13001–13008, 2020.

[62] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[63] Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Training high-performance low-latency spiking neural networks by differentiation on spike representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12444–12453, 2022.

[64] Nitin Rathi and Kaushik Roy. Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[65] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing

dependent backpropagation. In *International Conference on Learning Representations (ICLR)*, 2020.

6 具有高效率和学习长期依赖能力的并行脉冲神经元的补充材料

6.1 仿真速度基准测试环境

模拟速度基准测试在配备 Intel Xeon(R) Gold 6226R CPU、192G 内存和 NVIDIA Quadro RTX 6000 (24G) GPU 的 Ubuntu 20.04.6 LTS 服务器上进行。PyTorch 愿景是 1.12.1，CUDA 版本是 11.3.1，GPU 驱动版本是 525.105.17。

6.2 参数初始化

对于 PSN 和掩码 PSN，权重由 kaiminguniform [55] 初始化。更具体地说，这些值是从 $\mathcal{U}(-\sqrt{5},\sqrt{5})$ 初始化的。对于滑动 PSN，权重通过指数衰减方法初始化

$$W_i = 2^{i-k+1}, \quad i = 0, 1, ..., k-1 \tag{17}$$

这与具有 $\tau_m = 2$ 并移除重置的 LIF 神经元的权重类似。PSN 系列的阈值均初始化为 1。

6.3 滑动 **PSN** 的实现

正如正文中提到的，滑动 PSN 可以通过矩阵-矩阵乘法和一维卷积来实现。然而，一维卷积比矩阵-矩阵乘法慢得多，这是由于：

1）卷积的并行度低于矩阵-矩阵乘法，我们可以发现，现代深度神经网络中 1×1 卷积通常由全连接层实现，例如 ConvNeXt [56]。

2）滑动 PSN 的一维卷积要求输入形状为 $(..., 1, T)$，而 SpikingJelly 中默认的数据格式为 $(T, N, ...)$。因此，必须使用转置操作来移动时间步维度，这会复制数据并降低速度。

6.4 训练超参数

不同数据集的主要超参数如表3所示。除非另有说明，权重衰减（wd）为 0，SGD 优化器的动量为 0.9，学习率调度器为余弦退发放调度器 [57]，使用自动混合精度训练，代理函数为反正切代理函数 $\sigma(x) = \frac{\alpha}{2(1+(\frac{\pi}{2}\alpha x)^2)}$ 和来自 [15] 的 $\alpha = 4$。其他培训选项列出如下。

静态/顺序 **CIFAR** 这些变换包括随机混合 [58] 与 $p = 1, \alpha = 0.2$、随机剪切混合 [59] 与 $p = 1, \alpha = 1$、随机选择两种混合方法与 $p = 0.5$、随机水平翻转与 $p = 0.5$、琐碎增强 [60]、归一化、随机擦除 [61] 与 $p = 0.1$ 以及标签平滑 [62] 金额 0.1。静态 CIFAR 的通道数为 256 个，顺序 CIFAR 的通道数为 128 个。

Dataset	Optimizer	Batch Size	Epoch	Learning Rate	GPUs	Loss
Sequential CIFAR10/100	SGD AdamW(sliding PSN)	128	256	0.1 0.001(sliding PSN)	1	CE
CIFAR10	SGD	128	1024	0.1	1	CE
ImageNet	SGD	64 (SEW-18) 32 (SEW-34)	320	0.1	8	TET
CIFAR10-DVS	SGD, wd=5e-4	32	200	0.1	2	TET

表 3: 针对不同数据集训练超参数。

T	N	$\mathcal{M}_{NO}$	$\mathcal{M}_{IF}$	$\mathcal{M}_{PSN}$	Δ_{IF-NO}	Δ_{PSN-NO}	$\frac{\Delta_{IF-NO}}{\Delta_{PSN-NO}}$	$\frac{\Delta_{IF-NO}-\Delta_{PSN-NO}}{T\cdot N}$
16	16	35129.9	70563.9	53675.9	35434	18546	1.9	66.0
8	16	18935.9	36863.9	28225.9	17928	9290	1.9	67.5
16	8	18933.9	36099.9	28223.9	17166	9290	1.8	61.5

表 4: 使用 IF 神经元和 PSN 训练 SNN 的内存消耗比较。

ImageNet 变换与 [40] 相同。我们加载了 ANN 社区提供的标准 ResNet 中的预训练权重作为更好的初始化参数。类似的技术已广泛应用于代理训练 [63, 64, 65]。

CIFAR10-DVS 变换与 [33] 相同。我们使用与 [33] 几乎相同的网络结构、训练超参数和选项，只是我们使用普通 BN 而不是 TEBN。

6.5 PSN 家族的内存消耗

SNN 在训练过程中的内存复杂度可以近似为 $\mathcal{O}(\mathcal{W}+T\cdot(\mathcal{X}+\mathcal{H}))$，其中 $\mathcal{W}$ 是 PSN 中神经元（例如 $\boldsymbol{W}$）内部神经元（例如卷积/线性层）和权重等突触的数量，T 是时间步数，$\mathcal{X}$ 是单个时间步上所有层的输入/输出，$\mathcal{H}$ 是所有层的隐状态在单个时间步。当比较使用普通脉冲神经元或 PSN 系列的 SNN 时，内存消耗的差异主要取决于 $\mathcal{H}$。

对于普通脉冲神经元，可以发现隐状态的数量至少为 2，即 $\boldsymbol{H}$ 和 $\boldsymbol{V}$。如果脉冲神经元集成了更复杂的神经元动力学（例如自适应阈值），隐状态的数量就会增加。而对于 PSN，仅使用 $\boldsymbol{H}$，隐状态的数量为 1。因此，可以发现 PSN 的内存消耗比 vanilla 脉冲神经元少 $\mathcal{O}(T\cdot N)$，其中 N 是 SNN 中的神经元数量。

我们设计了一个有趣的实验来验证我们的分析。我们测量 $\mathcal{M}_{NO}$，这是训练仅使用突触但没有神经元的网络来估计 $\mathcal{O}(\mathcal{W}+T\cdot\mathcal{X}$ 的内存消耗。然后我们测量 $\mathcal{M}_{IF}$ 和 $\mathcal{M}_{PSN}$，它们是具有最简单的普通脉冲神经元、IF 神经元和 PSN 的 SNN，以估计 $\mathcal{O}(\mathcal{W}+T\cdot(\mathcal{X}+\mathcal{H}))$。因此，我们可以通过 $\Delta_{IF-NO}=\mathcal{M}_{IF}-\mathcal{M}_{NO}$ 和 $\Delta_{PSN-NO}=\mathcal{M}_{PSN}-\mathcal{M}_{NO}$ 来估计两个 SNN 中隐状态的内存消耗。我们对 VGG-11 结构进行测试，结果如表4所示。可以发现，所有情况下都是 $\Delta_{PSN-NO}<\Delta_{IF-NO}$ 和 $\frac{\Delta_{IF-NO}}{\Delta_{PSN-NO}}\approx 2$，这与我们的分析相符。更具体地说，IF 神经元使用 2 隐状态，PSN 使用 1 隐状态，那么它们的额外内存消耗的比率应该是 2。还可以发现，$\frac{\Delta_{IF-NO}-\Delta_{PSN-NO}}{T\cdot N}$ 近似为一个常数，这也符合我们分析 PSN 的内存消耗比 IF 神经元少 $\mathcal{O}(T\cdot N)$。