Deep Residual Learning in Spiking Neural Networks

脉冲神经网络中的深度残差学习

方维 [1,2], 余肇飞 [1,2*], 陈彦骐 [1,2],

黄铁军 [1,2], Timothée Masquelier[3], 田永鸿 [1,2*]

[1] 北京大学计算机科学与技术系

[2] 鹏城实验室，中国深圳 518055

[3]Centre de Recherche Cerveau et Cognition, UMR5549 CNRS - 图卢兹第三大学，法国图卢兹

Abstract

深度脉冲神经网络（SNN）由于离散的二值激活以及复杂的时空动力学，给基于梯度的方法带来了优化困难。考虑到 ResNet 在深度学习领域取得的巨大成功，使用残差学习来训练深度 SNN 是很自然的事情。以往的 Spiking ResNet 模仿 ANN 中的标准残差块，只是简单地将 ReLU 激活层替换为脉冲神经元，因此会遭遇退化问题，并且很难真正实现残差学习。在本文中，我们提出了脉冲元素级（SEW）ResNet 来实现深度 SNN 中的残差学习。我们证明 SEW ResNet 可以轻松实现恒等映射并克服 Spiking ResNet 的梯度消失/爆炸问题。我们在 ImageNet、DVS Gesture 和 CIFAR10-DVS 数据集上评估了 SEW ResNet，结果表明 SEW ResNet 在准确率和时间步数两方面都优于当前最先进的直接训练 SNN。此外，SEW ResNet 只需添加更多层即可实现更高的性能，为训练深度 SNN 提供了一种简单的方法。据我们所知，这是第一次直接训练超过 100 层的深度 SNN 成为可能。我们的代码可在 https://github.com/fangwei123456/Spike-Element-Wise-ResNet 处获取。

1 简介

人工神经网络（ANN）在许多任务中取得了巨大的成功，包括图像分类 [28, 52, 55]、物体检测 [9, 34, 44]、机器翻译 [2]，以及游戏 [37, 51]。人工神经网络成功的关键因素之一是深度学习 [29]，它使用多层来学习具有多个抽象级别的数据表示。已经证明，更

*通讯作者

深的网络在计算成本和泛化能力方面比浅层网络具有优势 [3]。深层网络表示的函数可能需要具有一个隐藏层的浅层网络的指数数量的隐藏单元 [38]。此外，网络的深度与网络在实际任务中的表现密切相关 [52, 55, 27, 52]。然而，最近的证据 [13, 53, 14] 表明，随着网络深度的增加，准确性会饱和，然后迅速下降。为了解决这个退化问题，提出了残差学习 [14, 15]，残差结构被广泛应用于“非常深”的网络中，实现了领先的性能 [22, 59, 18, 57]。

脉冲神经网络（SNN）因其高生物学合理性、事件驱动特性和低功耗而被视为 ANN 的潜在竞争对手 [45]。最近，深度学习方法被引入 SNN，深度 SNN 在一些简单的分类数据集 [56] 中取得了与 ANN 相近的性能，但在复杂任务中仍然比 ANN 差，例如对 ImageNet 数据集进行分类 [47]。为了获得更高性能的 SNN，探索像 ResNet 这样更深层次的网络结构是很自然的。Spiking ResNet [25, 60, 21, 17, 49, 12, 30, 64, 48, 42, 43] 作为 ResNet 的脉冲版本，是通过模仿 ANN 中的残差块并用脉冲神经元替换 ReLU 激活层而提出的。从 ANN 转换而来的 Spiking ResNet 在几乎所有数据集上都实现了最先进的精度，而直接训练的 Spiking ResNet 尚未经过验证可以解决退化问题。

在本文中，我们证明了 Spiking ResNet 不适用于所有神经元模型来实现恒等映射。即使满足恒等映射条件，Spiking ResNet 也会遇到梯度消失/爆炸的问题。因此，我们提出了 Spike-Element-Wise (SEW) ResNet 来实现 SNN 中的残差学习。我们证明 SEW ResNet 可以轻松实现恒等映射，同时克服梯度消失/爆炸问题。我们在静态 ImageNet 数据集和类脑 DVS 手势数据集 [1]、CIFAR10-DVS 数据集 [32] 上评估 Spiking ResNet 和 SEW ResNet。实验结果与我们的分析一致，表明较深的 Spiking ResNet 存在退化问题——较深的网络比较浅的网络具有更高的训练损失，而 SEW ResNet 可以通过简单地增加网络深度来实现更高的性能。此外，我们表明 SEW ResNet 在准确性和时间步长方面均优于最先进的直接训练 SNN。据我们所知，这是首次探索超过 100 层的直接训练深度 SNN。

2 相关工作

2.1 脉冲神经网络的学习方法

ANN 到 SNN 的转换 (ANN2SNN) [20, 4, 46, 49, 12, 11, 6, 54, 33] 和代理梯度反向传播 [40] 是获得深度 SNN 的两种主要方法。ANN2SNN 方法首先使用 ReLU 激活训练 ANN，然后通过用脉冲神经元替换 ReLU 并添加权重归一化和阈值平衡等缩放操作，将 ANN 转换为 SNN。最近的一些转换方法已经通过 VGG-16 和 ResNet [12, 11, 6, 33] 实现了近乎无损的精度。然而，由于转换基于速率编码 [46]，转换后的 SNN 需要更长的时间才能在精度上与原始 ANN 相媲美，这增加了 SNN 的延迟并限制了实际应用。反向传播方法可以分为两类 [26]。第一类方法通过在模拟时间步长 [31, 19, 58, 50, 30, 40] 上展开网络来计算梯度，这类似于随时间反向传播（BPTT）的思想。由于相对于阈值触发的发射的梯度是不可微的，因此经常使用替代梯度。通过代理方法训练的 SNN 不仅限于速率编码，还可以应用于时间任务，例如对类脑数据集 [58, 8, 16] 进行分类。第二种方法计算现有脉冲时间相对于脉冲时间 [5, 39, 24, 65, 63] 处膜电位的梯度。

2.2 脉冲残差结构

之前的 ANN2SNN 方法注意到普通前馈 ANN 和残差 ANN 之间的区别，并对转换进行了具体的归一化。Hu 等人 [17] 是第一个在 ANN2SNN 中应用残差结构并在 SNN 中使用缩放快捷方式来匹配原始 ANN 的激活的人。Sengupta 等人 [49] 提出了 Spike-Norm 来平衡 SNN 的阈值，并通过将 VGG 和 ResNet 转换为 SNN 来验证他们的方法。现有的基于反向传播的方法使用与 ResNet 几乎相同的结构。Lee 等人 [30] 在深度不超过 ResNet-11 的浅层 ResNet 上评估了他们的自定义代理方法。Cheng 等人 [64] 提出了阈值相关批归一化（td-BN）来替代朴素批归一化（BN）[23]，并通过在快捷方式中添加 td-BN，成功地直接用代理梯度训练 Spiking ResNet-34 和 Spiking ResNet-50。

3 方法

3.1 脉冲神经元模型

脉冲神经元是 SNN 的基本计算单元。与 Fang 等人 [8] 类似，我们使用统一的模型来描述所有的动态各种脉冲神经元，其中包括以下离散时间方程：

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

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

$$V[t] = H[t]\ \ (1 - S[t]) + V_{reset}\ S[t], \tag{3}$$

其中，$X[t]$ 是时间步长 t 的输入电流，$H[t]$ 和 $V[t]$ 分别表示神经元动力学后和时间步长 t 脉冲触发后的膜电位。V_{th} 是触发阈值，$\Theta(x)$ 是 Heaviside 阶跃函数，对于 $x \geq 0$ 由 $\Theta(x) = 1$ 定义，对于 $x < 0$ 由 $\Theta(x) = 0$ 定义。$S[t]$ 是时间步 t 处的输出脉冲，如果存在脉冲则等于 1，否则等于 0。V_{reset} 表示复位电位。方程 (1) 中的函数 $f(\cdot)$ 描述了神经元动力学，并针对不同的脉冲神经元模型采取不同的形式。例如，积分激发（IF）模型和泄漏积分激发（LIF）模型的函数 $f(\cdot)$ 可以分别由方程 (4)和方程 (5)来描述。

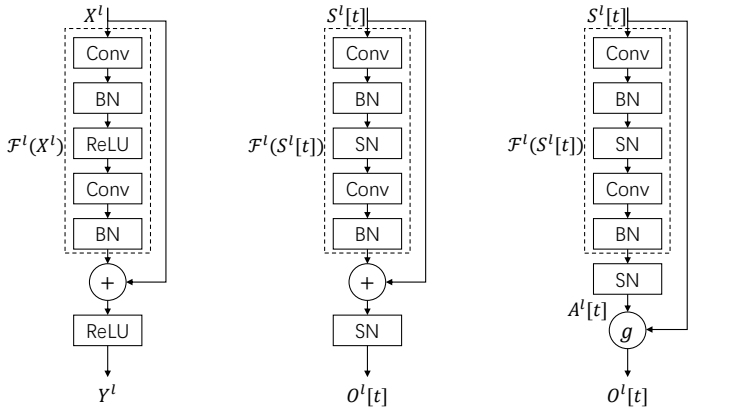
$$H[t] = V[t-1] + X[t], \tag{4}$$

$$H[t] = V[t-1] + \frac{1}{\tau}(X[t] - (V[t-1] - V_{reset})), \tag{5}$$

其中 τ 表示膜时间常数。方程 (2)和方程 (3)描述了脉冲的产生和重置过程，这对于各种脉冲神经元模型都是相同的。本文采用代理梯度法定义误差反向传播时的 $\Theta'(x) \triangleq \sigma'(x)$，其中 $\sigma(x)$ 表示代理函数。

3.2 Spiking ResNet 的缺点

残差块是 ResNet 的关键组成部分。图 1(a) 显示了 ResNet [14] 中的基本块，其中 X^l, Y^l 是 ResNet 中第 l 块的输入和输出，Conv 是卷积层，BN 表示批归一化，ReLU 表示修正线性单元激活层。[64, 17, 30] 中使用的 Spiking ResNet 的基本块只是通过用脉冲神经元 (SN) 替换 ReLU 激活层来模仿 ANN 中的块，如图 1(b) 所示。这里 $S^l[t], O^l[t]$



(a) Basic block in ResNet (b) Basic block in Spiking ResNet (c) Spike-Element-Wise block

图 1: ResNet、Spiking ResNet 和 SEW ResNet 中的残差块。

是 Spiking ResNet 中第 l 块在时间步长 t 的输入和输出。基于上述定义，下面我们将分析 Spiking ResNet 的缺点。

Spiking ResNet 不适用于所有神经元模型来实现恒等映射。 ResNet 中的关键概念之一是恒等映射。他等人 [14] 指出，如果添加的层实现恒等映射，则更深的模型的训练误差不应大于其较浅的对应模型。然而，它无法在可行的时间内训练添加的层来实现恒等映射，导致更深的模型比更浅的模型表现更差（退化问题）。为了解决这个问题，残差学习被提出，通过添加快捷连接（如图 1（a）所示）。如果我们使用 $\mathcal{F}^l$ 来表示 ResNet 和 Spiking ResNet 中第 l 个残差块的残差映射，例如两个卷积层的堆栈，则图 1(a) 和图 1(b) 中的残差块可以表示为

$$Y^l = \mathrm{ReLU}(\mathcal{F}^l(X^l) + X^l), \tag{6}$$

$$O^l[t] = \mathrm{SN}(\mathcal{F}^l(S^l[t]) + S^l[t]). \tag{7}$$

方程 (6) 的残差块使得在 ANN 中实现恒等映射变得容易。看到这一点，当 $\mathcal{F}^l(X^l) \equiv 0$，$Y^l = \mathrm{ReLU(X^l)}$。大多数情况下，$X^l$ 是前面的 ReLU 层和 $X^l \geq 0$ 的激活。因此，$Y^l = \mathrm{ReLU}(X^l) = X^l$，即恒等映射。

与 ResNet 不同的是，Spiking ResNet 中的残差块（Eq. (7)）限制脉冲神经元的模型来实现恒等映射。当 $\mathcal{F}^l(S^l[t]) \equiv 0$ 时，$O^l[t] = \mathrm{SN}(S^l[t]) \neq S^l[t]$。为了传输 $S^l[t]$ 并生成 $\mathrm{SN}(S^l[t]) = S^l[t]$，第 l 残差块中的最后一个脉冲神经元（SN）需要在接收到脉冲后发射脉冲，并在时间步 t 没有接收到脉冲后保持沉默。它适用于方程 (4) 描述的 IF 神经元。具体来说，我们可以设置 $0 < V_{th} \leq 1$ 和 $V[t-1] = 0$，保证 $X[t] = 1$ 通向 $H[t] \geq V_{th}$，$X[t] = 0$ 通向 $H[t] < V_{th}$。然而，当考虑一些具有复杂神经元动力学的脉冲神经元模型时，很难实现 $\mathrm{SN}(S^l[t]) = S^l[t]$。例如，[66, 8, 61] 中使用的 LIF 神经元考虑可学习的膜时间常数 τ，其神经元动力学可以用方程 (5)来描述。当 $X[t] = 1$ 和 $V[t-1] = 0$、$H[t] = \frac{1}{\tau}$ 时。很难找到确保 $H[t] > V_{th}$ 的触发阈值，因为优化器在训练中会更改 τ。

Spiking ResNet 存在梯度消失/爆炸问题。 考虑一个具有 k 顺序块的脉冲 ResNet 来传输 $S^l[t]$，并且满足恒等映射条件，例如，脉冲神经元是具有 $0 < V_{th} \leq 1$ 的 IF 神经元，那么我们有 $S^l[t] = S^{l+1}[t] = ... = S^{l+k-1}[t] = O^{l+k-1}[t]$。将 $S^l[t]$ 和 $O^l[t]$ 中的第 j 元

素分别表示为 $S_j^l[t]$ 和 $O_j^l[t]$，则可以逐层计算第 $(l+k-1)$ 残差块的输出相对于第 l 残差块的输入的梯度：

$$\frac{\partial O_j^{l+k-1}[t]}{\partial S_j^l[t]} = \prod_{i=0}^{k-1} \frac{\partial O_j^{l+i}[t]}{\partial S_j^{l+i}[t]} = \prod_{i=0}^{k-1} \Theta'(S_j^{l+i}[t] - V_{th}) \to \begin{cases} 0, & \text{if } 0 < \Theta'(S_j^l[t] - V_{th}) < 1 \\ 1, & \text{if } \Theta'(S_j^l[t] - V_{th}) = 1 \\ +\infty, & \text{if } \Theta'(S_j^l[t] - V_{th}) > 1 \end{cases}, \tag{8}$$

其中 $\Theta(x)$ 是 Heaviside 阶跃函数，$\Theta'(x)$ 由代理梯度定义。第二个等式为 $O_j^{l+i}[t] = \mathrm{SN}(S_j^{l+i}[t])$。鉴于 $S_j^l[t]$ 只能取 0 或 1，$\Theta'(S_j^l[t] - V_{th}) = 1$ 对于 [40] 中提到的常用代理函数并不满足。因此，梯度消失/爆炸问题在更深的 Spiking ResNet 中很容易发生。

基于以上分析，我们认为之前的 Spiking ResNet 忽略了脉冲神经元带来的高度非线性，很难实现残差学习。尽管如此，图 1(b) 中的基本块对于具有额外归一化 [17, 49] 的 ANN2SNN 来说仍然不错，因为从 ANN 转换而来的 SNN 旨在使用激发率来匹配原始 ANN 的激活。

3.3 脉冲元素级 ResNet

在这里，我们提出了 Spike-Element-Wise (SEW) 残差块来实现 SNN 中的残差学习，它可以轻松实现恒等映射，同时克服梯度消失/爆炸问题。如图 1(c) 所示，SEW 残差块可以表示为：

$$O^l[t] = g(\mathrm{SN}(\mathcal{F}^l(S^l[t])), S^l[t]) = g(A^l[t], S^l[t]), \tag{9}$$

其中 g 表示以两个脉冲张量作为输入的逐元素函数。这里我们用 $A^l[t]$ 来表示要学习的残差映射为 $A^l[t] = \mathrm{SN}(\mathcal{F}^l(S^l[t]))$。

SEW ResNet 可以轻松实现恒等映射。利用脉冲的二元性质，我们可以找到满足恒等映射的不同元素函数 g（如表 1所示）。具体来说，当选择 ADD 和 IAND 作为逐元素函数 g 时，通过设置 $A^l[t] \equiv 0$ 来实现恒等映射，只需将 $\mathcal{F}^l$ 中最后一个批归一化层 (BN) 的权重和偏置设置为零即可实现。那么我们可以得到 $O^l[t] = g(A^l[t], S^l[t]) = g(\mathrm{SN}(0), S^l[t]) = g(0, S^l[t]) = S^l[t]$。这适用于所有神经元模型。当使用 AND 作为逐元素函数 g 时，我们设置 $A^l[t] \equiv 1$ 来获得恒等映射。它可以通过将最后一个 BN 的权重设置为零并将偏差设置为足够大的常数以引起脉冲来实现，例如，当最后一个 SN 是 IF 神经元时将偏差设置为 V_{th}。然后我们有 $O^l[t] = 1 \land S^l[t] = S^l[t]$。请注意，使用 AND 可能会遇到与 Spiking ResNet 相同的问题。控制一些具有复杂神经元动力学的脉冲神经元模型以在指定时间步产生脉冲是很困难的。

名称	$g(A^l[t], S^l[t])$ 的表达式
ADD	$A^l[t] + S^l[t]$
AND	$A^l[t] \land S^l[t] = A^l[t] \cdot S^l[t]$
IAND	$(\neg A^l[t]) \land S^l[t] = (1 - A^l[t]) \cdot S^l[t]$

表 1: 逐元素函数列表 g.

下采样块的公式。值得注意的是，当一个块的输入和输出具有不同维度时，快捷方式被设置为步长为 > 1 的卷积层，而不是恒等连接来执行下采样。ResNet 和 Spiking ResNet

(a) Downsample basic block (b) Downsample SEW block

图 2: Spiking ResNet 和 SEW ResNet 中的下采样块。

使用 {Conv-BN}，而没有使用 ReLU 的快捷方式（图 2(a)）。相反，我们在快捷方式中添加一个 SN（图 2（b））。

SEW ResNet 可以克服梯度消失/爆炸。 SEW 块类似于 ANN 中的 ReLU before Add (RBA) 块 [15]，可以表示为

$$Y^l = \mathrm{ReLU}(\mathcal{F}^l(X^l)) + X^l. \tag{10}$$

RBA 块被 He 等人 [15] 批评为 $X^{l+1} = Y^l \geq X^l$，这会导致深层输出无限。[15] 中的实验结果也表明 RBA 块的性能比基本块差（图1（a））。从某种程度上来说，SEW 区块是 RBA 区块的延伸。请注意，使用 AND 和 IAND 作为 g 将输出脉冲（即二进制张量），这意味着 ANN 中的无限输出问题永远不会出现在具有 SEW 块的 SNN 中，因为所有脉冲都小于或等于 1。当选择 ADD 作为 g 时，可以缓解无限输出问题，因为 k 顺序 SEW 块的输出不会大于 $k+1$。此外，当 g 为 ADD 时，下采样 SEW 模块会将输出调节为不大于 2。

当实现恒等映射时，可以逐层计算第 $(l+k-1)$ 个 SEW 块的输出相对于第 l 个 SEW 块的输入的梯度：

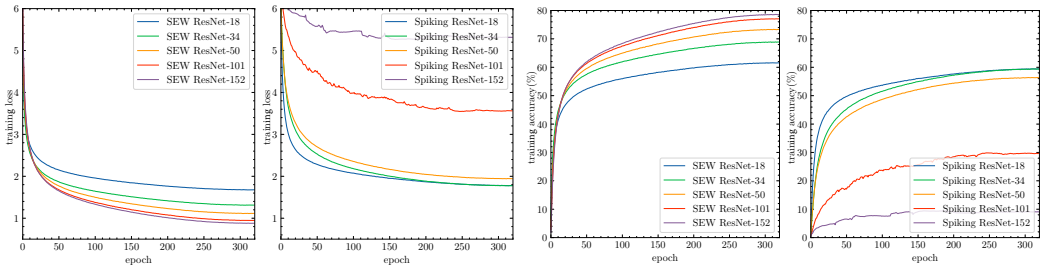
$$\frac{\partial O_j^{l+k-1}[t]}{\partial S_j^l[t]} = \prod_{i=0}^{k-1} \frac{\partial g(A_j^{l+i}[t], S_j^{l+i}[t])}{\partial S_j^{l+i}[t]} = \begin{cases} \prod_{i=0}^{k-1} \frac{\partial (0 + S_j^{l+i}[t])}{\partial S_j^{l+i}[t]}, \text{if } \ g = ADD \\ \prod_{i=0}^{k-1} \frac{\partial (1 \cdot S_j^{l+i}[t])}{\partial S_j^{l+i}[t]}, \text{if } \ g = AND \\ \prod_{i=0}^{k-1} \frac{\partial ((1-0) \cdot S_j^{l+i}[t])}{\partial S_j^{l+i}[t]}, \text{if } \ g = IAND \end{cases} = 1. \tag{11}$$

第二个等式成立，因为恒等映射是通过为 g = AND 设置 $A^{l+i}[t] \equiv 1$ 和为 g = ADD/IAND 设置 $A^{l+i}[t] \equiv 0$ 来实现的。由于方程 (11)中的梯度是一个常数，SEW ResNet 可以克服梯度消失/爆炸问题。

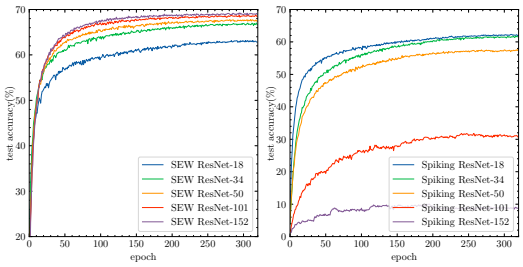
4 实验

4.1 ImageNet 分类

由于 ImageNet 2012 的测试服务器不再可用，我们无法报告实际的测试精度。相反，我们使用 validation 集上的准确率作为测试准确率，这与 [17, 64] 相同。他等人 [14] 在



(a) Training loss (b) Training accuracy



(c) Test accuracy

图 3: ImageNet 上训练损失、训练精度和测试精度的比较。

网络	SEW ResNet (ADD)		脉冲 ResNet	
	Acc@1(%)	Acc@5(%)	Acc@1(%)	Acc@5(%)
ResNet-18	63.18	84.53	62.32	84.05
ResNet-34	67.04	87.25	61.86	83.69
ResNet-50	67.78	87.52	57.66	80.43
ResNet-101	68.76	88.25	31.79	54.91
ResNet-152	69.26	88.57	10.03	23.57

表 2: 在 ImageNet 上测试准确性。

ImageNet 数据集上评估了 18/34/50/101/152 层 ResNet。为了进行比较，我们考虑具有相同网络架构的 SNN，除了基本残差块（图1（a））分别被脉冲基本块（图1（b））和 SEW 块（图1（c））替换，其中 g 为 ADD。我们将具有基本块的 SNN 表示为 Spiking ResNet，将具有 SEW 块的 SNN 表示为 SEW ResNet。静态 ImageNet 数据集采用 IF 神经元模型。在 ImageNet 上的训练过程中，我们发现 Spiking ResNet-50/101/152 无法收敛，除非我们使用零初始化 [10]，这在训练开始时将所有块设置为恒等映射。因此，本文报道的 Spiking ResNet-18/34/50/101/152 的结果是零初始化的。

脉冲 ResNet vs. SEW ResNet。 我们首先评估 Spiking ResNet 和 SEW ResNet 的性能。Tab. 2 报告 ImageNet 验证的测试准确性。结果表明，较深的 34 层 Spiking ResNet 的测试精度低于较浅的 18 层 Spiking ResNet。随着层数的增加，Spiking ResNet 的测试精度下降。为了揭示原因，我们比较了 Spiking ResNet 在训练过程中的训练损失、训练精度和测试精度，如图 3 所示。我们可以发现 Spiking ResNet 的退化问题——更深的网络比更浅的网络有更高的训练损失。相比之下，较深的 34 层 SEW ResNet 比较浅的 18 层 SEW ResNet 具有更高的测试精度（如 Tab. 2 所示）。更重要的是，从图 3可以发现，随着深度的增加，我们的 SEW ResNet 的训练损失减少，训练/测试精

网络	方法	准确度 (%)	T
SEW ResNet-34	基于脉冲的 BP	67.04	4
带 td-BN 的脉冲 ResNet-34(大)† [64]	基于脉冲的 BP	67.05	6
带 td-BN 的脉冲 ResNet-34 [64]	基于脉冲的 BP	63.72	6
脉冲 ResNet-34 [12]	ANN2SNN	69.89	4096
脉冲 ResNet-34 [49]	ANN2SNN	65.47	2000
脉冲 ResNet-34 [33]	ANN2SNN	74.61	256
脉冲 ResNet-34 [43]	ANN2SNN 和基于脉冲的 BP	61.48	250
SEW ResNet-50	基于脉冲的 BP	67.78	4
带 td-BN 的脉冲 ResNet-50 [64]	基于脉冲的 BP	64.88	6
脉冲 ResNet-50 [17]	ANN2SNN	72.75	350
SEW ResNet-101	基于脉冲的 BP	68.76	4
SEW ResNet-152	基于脉冲的 BP	69.26	4

表 3: 与之前 ImageNet 上的 Spiking ResNet 进行比较。† 与标准 Spiking ResNet-34 具有相同的网络结构，但使用的卷积核数量是标准 Spiking ResNet-34 的四倍。

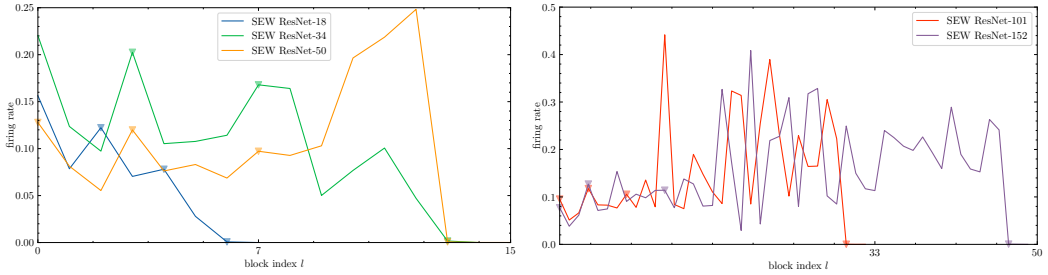


图 4: ImageNet 上 SEW 块中 A^l 的发射率。

度增加，这表明我们可以通过简单地增加网络深度来获得更高的性能。所有这些结果都表明 SEW ResNet 很好地解决了退化问题。

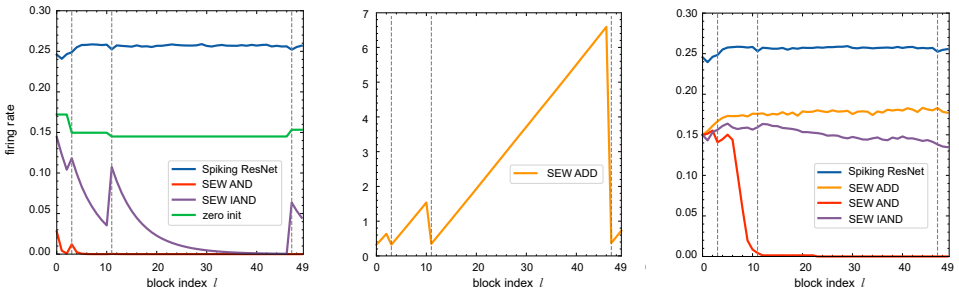
与最先进方法的比较。在 Tab. 3 中，我们将 SEW ResNet 与之前在 ImageNet 上取得最佳结果的 Spiking ResNet 进行了比较。据我们所知，SEW ResNet-101 和 SEW ResNet-152 是迄今为止唯一超过 100 层的 SNN，并且没有其他具有相同结构的网络可供比较。当网络结构相同时，即使时间步长 T 更少，我们的 SEW ResNet 也优于直接训练的 Spiking ResNet 的最先进精度。SEW ResNet-34 的精度略低于带有 td-BN 的 Spiking ResNet-34（大）（67.04% 与 67.05%），后者使用 1.5 倍的模拟时间步 T（6 与 4）和 4 倍的参数数量（85.5M 与 67.05%）。21.8M），与我们的 SEW ResNet 相比。最先进的 ANN2SNN 方法 [33, 17] 比我们的 SEW ResNet 具有更好的精度，但它们分别使用我们的 64 和 87.5 倍的时间步长。

SEW 块的脉冲响应分析。图 4显示了 ImageNet 上 SEW ResNet-18/34/50/101/152 中 A^l 的发射率。SEW ResNet-18 有 7 个块，SEW ResNet-34 和 SEW ResNet-50 有 15 个块，SEW ResNet-101 有 33 个块，SEW ResNet-152 有 50 个块。下采样 SEW 块由三角形下符号 $\triangledown$ 标记。当我们选择 ADD 作为逐元素函数 g 时，较低的触发率意味着 SEW 块更接近于实现恒等映射（下采样块除外）。请注意，下采样块的快捷方式不是恒等映射，如图 2(b) 所示。图 4显示 SEW 块中的所有脉冲神经元的放电率都很低，最后两个块中的脉冲神经元的放电率甚至几乎为零。由于时间步长 T 为 4 并且放电率不大

于 0.25，SEW ResNet-18/34/50 中的所有神经元在整个模拟过程中平均放电不超过 1 个脉冲。此外，SEW ResNet-101/152 中的所有放电率均不大于 0.5，表明所有神经元平均放电不超过两个脉冲。总的来说，SEW 块中 A^l 的触发率处于较低水平，验证了大多数 SEW 块充当恒等映射。

ResNet-152 结构的梯度检查。 方程 (8) 和方程 (11) 利用恒等映射分析多个块的梯度。为了验证 SEW ResNet 可以克服梯度消失/爆炸问题，我们检查了 Spiking ResNet-152 和 SEW ResNet-152 的梯度，它们是最深的标准 ResNet 结构。我们考虑相同的初始化参数以及有/没有零初始化。

由于 SNN 的梯度受到发射率的显着影响（参见第 A.4 节），我们首先分析发射率。图 5(a) 显示了第 l 块的输出 O^l 的初始发射率。下采样块的索引由垂直虚线标记。两条相邻虚线之间的块代表恒等映射区域，并且具有相同形状的输入和输出。当使用零初始化时，Spiking ResNet、SEW AND ResNet、SEW IAND ResNet 和 SEW ADD ResNet 具有相同的发射率（绿色曲线），即 zero init 曲线。在没有零初始化的情况下，静音问题发生在 SEW AND 网络（红色曲线）中，并通过 SEW IAND 网络（紫色曲线）得到缓解。图 5(b) 显示了 A^l 的发射率，它代表第 l 块中最后一个 SN 的输出。可以发现，虽然 SEW ADD ResNet 中 O^l 的发射率在恒等映射区域线性增加，但每个块中的最后一个 SN 仍然保持稳定的发射率。请注意，当 g 为 ADD 时，SEW 块的输出不是二进制的，并且发射率实际上是平均值。SEW IAND ResNet 的 SN 保持足够的发射率，并随深度略有衰减（紫色曲线），而 SEW AND ResNet 深层的 SN 保持沉默（橙色曲线）。沉默问题可以解释如下。使用 AND 时，$O^l[t] = \mathrm{SN}(\mathcal{F}^l(O^{l-1}[t])) \wedge O^{l-1}[t] \leq O^{l-1}[t]$。由于很难在每个时间步长 t 处保持 $\mathrm{SN}(\mathcal{F}^l(O^{l-1}[t])) \equiv 1$，因此在 AND 为 g 的 SEW ResNet 中可能会经常出现静默问题。使用 IAND 代替 AND 可以缓解这个问题，因为在每个时间步 t 很容易保持 $\mathrm{SN}(\mathcal{F}^l(O^{l-1}[t])) \equiv 0$。



(a) 输出 O^l 的发放率　　(b) A^l 的发放率

图 5: 152 层网络上第 l 块中输出 O^l 和 A^l 的初始发射率。

我们在所有实验中使用的代理梯度函数是 $\sigma(x) = \frac{1}{\pi}\arctan(\frac{\pi}{2}\alpha x) + \frac{1}{2}$，因此是 $\sigma'(x) = \frac{\alpha}{2(1+(\frac{\pi}{2}\alpha x)^2)}$。当 $V_{th} = 1, \alpha = 2$ 时，各块的梯度幅值 $\left\|\frac{\partial L}{\partial S^l}\right\|$ 如图 6所示。请注意 $\alpha = 2$、$\sigma'(x) \leq \sigma'(0) = \sigma'(1 - V_{th}) = 1$ 和 $\sigma'(0 - V_{th}) = 0.092 < 1$。可以发现，在没有零初始化的情况下，Spiking ResNet-152 的梯度从较深层到较浅层的恒等映射区域衰减，这是由 $\sigma'(x) \leq 1$ 引起的。值得注意的是，衰减也发生在零初始化的 Spiking ResNet-152

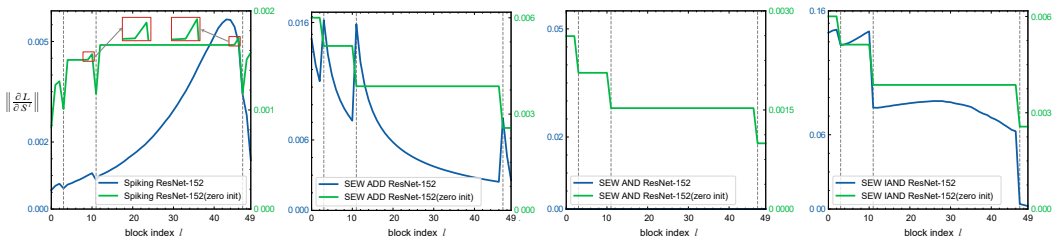


图 6: $V_{th}=1,\alpha=2$ 时第 l 块的梯度幅值 $\left\|\frac{\partial L}{\partial S^l}\right\|$。

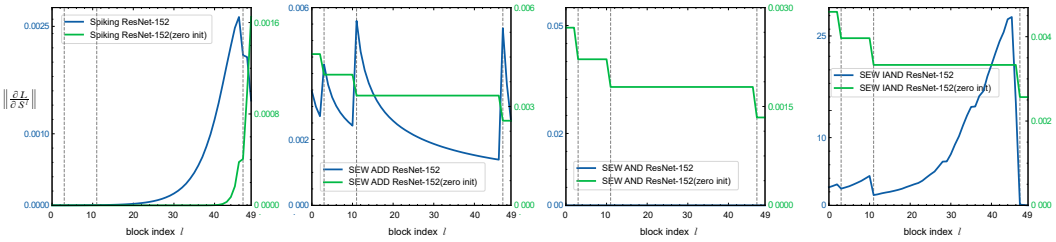


图 7: $V_{th}=0.5,\alpha=2$ 时第 l 块的梯度幅值 $\left\|\frac{\partial L}{\partial S^l}\right\|$。

中。虚线附近的小凸 $\bigwedge$ 是由那些 $S_j^l[t]=0$ 的梯度消失引起的。当这些梯度完全衰减到 0 后，$\left\|\frac{\partial L}{\partial S^l}\right\|$ 将是一个常数，因为其余梯度是由 $S_j^l[t]=1$ 和 $\sigma'(1-V_{th})=1$ 计算的，这也可以解释为什么梯度指数曲线在某些区域是水平的。当参考零初始化的 SEW ResNet-152 时，可以发现无论我们选择什么 g，所有梯度指数曲线都是相似的。这是因为在恒等映射区域中，S^l 对于所有索引 l 都是恒定的，并且梯度也成为恒定的，因为它不会流过 SN。如果没有零初始化，SEW AND ResNet-152 中会出现梯度消失，这是由静默问题引起的。由于图 5 所示的足够的发射率，SEW ADD、IAND 网络的梯度从较深层传播到较浅层时缓慢增加。

当 $V_{th}=0.5,\alpha=2$、$\sigma'(0-V_{th})=\sigma'(1-V_{th})=0.288<1$ 时，表明向 SN 传输脉冲容易引起梯度消失，如图 7所示。在零初始化的情况下，Spiking ResNet-152 中的衰减更加严重，因为来自 $\mathcal{F}^l$ 的梯度无法做出贡献。无论我们选择什么 g，SEW ResNet-152 都不会受到影响。当 $V_{th}=1,\alpha=3$、$\sigma'(1-V_{th})=1.5>1$ 时，表明向 SN 传输脉冲容易引起梯度爆炸。图 8 显示了这种情况下的梯度。与图 6中的原因相同，代理函数的改变会增加所有没有零初始化的网络的梯度，但不会影响零初始化的 SEW ResNet-152。Spiking ResNet-152 遇到梯度爆炸，而 SEW ADD、IAND ResNet-152 中这个问题并不严重。

4.2 DVS 手势分类

原始 ResNet 是为对复杂的 ImageNet 数据集进行分类而设计的，对于 DVS 手势数据集来说太大了。因此，我们设计了一个名为 7B-Net 的微型网络，其结构为 c32k3s1-BN-PLIF-{SEW Block-MPk2s2}*7-FC11。这里 c32k3s1 表示通道 32、内核大小 3、步长 1 的卷积层。MPk2s2 是内核大小 2、步长 2 的最大池化。符号 {} * 7 表示七个重复结构，PLIF 表示具有可学习膜时间常数的参数泄漏积分和火脉冲神经元，其在 [8] 可由方程 (5) 描述。有关 AER 数据预处理的详细信息，请参阅 Sec.A.1。

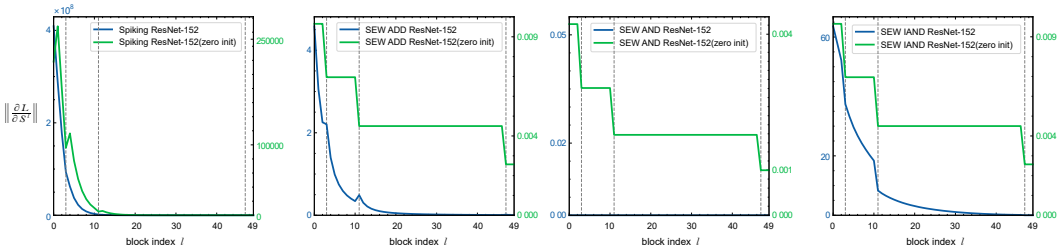
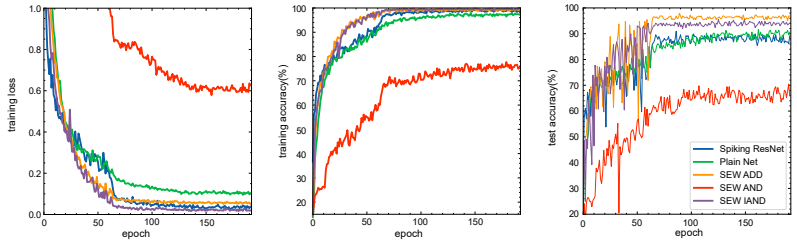


图 8: $V_{th} = 1, \alpha = 3$ 时第 l 块的梯度幅值 $\left\| \frac{\partial L}{\partial S^l} \right\|$。



(a) Training loss (b) Training accuracy (c) Test accuracy

图 9: DVS Gesture 数据集上的训练损失、训练精度和测试精度的比较。

脉冲 ResNet vs. SEW ResNet。我们首先比较 SEW ResNet 与 ADD 逐元素函数 (SEW ADD ResNet) 和用基本块替换 SEW 块的 Spiking ResNet 的性能。如图 9和 Tab. 4所示，虽然 Spiking ResNet（蓝色曲线）的训练损失低于 SEW ADD ResNet（橙色曲线），但测试精度低于 SEW ADD ResNet（90.97% vs 97.92%），这意味着 Spiking ResNet 比 SEW ADD ResNet 更容易过拟合。

不同逐元素函数和普通块的评估。由于 SNN 在 DVS 手势数据集上的训练成本远低于 ImageNet，因此我们在 DVS 手势数据集上进行了更多的消融实验。我们用普通块（无快捷方式连接）替换 SEW 块并测试性能。我们还在 Tab. 1 中评估了各种逐元素函数 g。图 9 显示了 DVS 手势的训练损失和训练/测试精度。早期时期的剧烈波动是由较大的学习率引起的（参见第 A.1 节）。我们可以发现训练损失是 SEW IAND<Spiking ResNet<SEW ADD<Plain Net<SEW AND。由于过拟合问题，较低的损失并不能保证较高的测试精度。表 4显示了所有网络的测试精度。SEW ADD ResNet 比其他网络获得最高的准确度。

网络	逐元素函数 g	精度 (%)
SEW ResNet	ADD	97.92
SEW ResNet	IAND	95.49
平网	–	91.67
脉冲 ResNet	–	90.97
SEW ResNet	AND	70.49

表 4: 测试 DVS 手势的准确性。网络的顺序按准确性排名。

与最先进方法的比较。 Tab. 5 将我们的网络与 SOTA 方法进行了比较。可以发现，我们的 SEW ResNet 在准确性、参数数量和模拟时间步长方面优于 SOTA 工作。

网络	精度 (%)	参数	T
c32k3s1-BN-PLIF-{SEW 块 (c32)-MPk2s2}*7-FC11 (7B-Net)	97.92	0.13M	16
{c128k3s1-BN-PLIF-MPk2s2}*5-DP-FC512-PLIF-DP-FC110-PLIF-APk10s10 [8]	97.57	1.70M	20
带 td-BN 的脉冲 ResNet-17 [64]	96.87	11.18M	40
MPk4-c64k3-LIF-c128k3-LIF-APk2-c128k3-LIF-APk2-FC256-LIF-FC11[16]	93.40	23.23M	60

表 5: 与 DVS 手势数据集上最先进的 (SOTA) 方法进行比较。

4.3 CIFAR10-DVS 分类

我们还报告了 CIFAR10-DVS 数据集上的 SEW ResNet，该数据集是通过 DVS 摄像机在 LCD 监视器上记录 CIFAR-10 数据集的运动图像而获得的。由于 CIFAR10-DVS 比 DVS Gesture 更复杂，我们使用名为 Wide-7B-Net 的网络结构，它与 7B-Net 类似，但通道数更多。Wide-7B-Net 的结构为 c64k3s1-BN-PLIF-{SEW Block(c64)-MPk2s2}*4-c128k3s1-BN-PLIF-{SEW Block(c128)-MPk2s2}*3-FC10。

网络	精度 (%)	参数	T
c64k3s1-BN-PLIF-{SEW 块 (c64)-MPk2s2}*4-c128k3s1-BN-PLIF-{SEW 块 (c128)-MPk2s2}*3-FC10（Wide-7B-Net）	64.8、70.2、74.4	1.19M	4、8、16
{c128k3s1-BN-PLIF-MPk2s2}*4-DP-FC512-PLIF-DP-FC100-PLIF-APk10s10[8]	74.8	17.4M	20
带 td-BN 的脉冲 ResNet-19 [64]	67.8	11.18M	10

表 6: 与 CIFAR10-DVS 数据集上最先进的 (SOTA) 方法进行比较。

在 Tab.6 中，我们将 SEW ResNet 与之前的 Spiking ResNet 进行了比较。可以发现，与 Spiking ResNet [64] 相比，我们的方法实现了更好的性能（70.2% 与 67.8%）和更少的时间步（8 比 10）。我们还将我们的方法与 CIFAR10-DVS 上最先进的 (SOTA) 监督学习方法进行了比较。我们的 Wide-7B-Net 的精度略低于当前的 SOTA 方法 [8]（74.4% 与 74.8%），后者使用 1.25 倍的模拟时间步长 T（20 与 16）和 14.6 倍的参数数量（17.4M 与 74.8%）。1.19M）。此外，当将 T 形状简化为 $T = 4$ 时，我们的 Wide-7B-Net 仍然可以获得 64.8% 的精度。

5 结论

在本文中，我们分析了之前的 Spiking ResNet，其残差块模仿了 ResNet 的标准块，发现它很难实现恒等映射，并且存在梯度消失/爆炸的问题。为了解决这些问题，我们提出了 SEW 残差块并证明它可以实现残差学习。在 ImageNet、DVS Gesture 和 CIFAR10-DVS 数据集上的实验结果表明，我们的 SEW 残差块解决了退化问题，并且 SEW ResNet 可以通过简单地增加网络深度来实现更高的精度。我们的工作可能会揭示“非常深”的 SNN 的学习。

6 致谢

这项工作得到了国家自然科学基金委的资助，合同号为 62027804、61825101 和 62088102。

References

[1] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, and Dharmendra Modha. A low power, fully event-based gesture recognition system. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7243–7252, 2017.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations (ICLR), 2015.

[3] Yoshua Bengio, Yann LeCun, et al. Scaling learning algorithms towards ai. Large-scale Kernel Machines, 34(5):1–41, 2007.

[4] 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.

[5] Iulia M Comsa, Krzysztof Potempa, Luca Versari, Thomas Fischbacher, Andrea Gesmundo, and Jyrki Alakuijala. Temporal coding in spiking neural networks with alpha synaptic function. In International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 8529–8533. IEEE, 2020.

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

[7] Wei Fang, Yanqi Chen, Jianhao Ding, Ding Chen, Zhaofei Yu, Huihui Zhou, Yonghong Tian, and other contributors. Spikingjelly. https://github.com/fangwei123456/spikingjelly.

[8] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothee 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.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings

of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 580–587, 2014.

[10] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677, 2017.

[11] Bing Han and Kaushik Roy. Deep spiking neural network: Energy efficiency through time based coding. In European Conference on Computer Vision (ECCV), pages 388–404, 2020.

[12] 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.

[13] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5353–5360, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In European Conference on Computer Vision (ECCV), pages 630–645. Springer, 2016.

[16] Weihua He, YuJie Wu, Lei Deng, Guoqi Li, Haoyu Wang, Yang Tian, Wei Ding, Wenhui Wang, and Yuan Xie. Comparing snns and rnns on neuromorphic vision datasets: Similarities and differences. Neural Networks, 132:108–120, 2020.

[17] Yangfan Hu, Huajin Tang, Yueming Wang, and Gang Pan. Spiking deep residual network. arXiv preprint arXiv:1805.01352, 2018.

[18] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4700–4708, 2017.

[19] Dongsung Huh and Terrence J Sejnowski. Gradient descent for spiking neural networks. In Advances in Neural Information Processing Systems (NeurIPS), pages 1440–1450, 2018.

[20] Eric Hunsberger and Chris Eliasmith. Spiking deep networks with lif neurons. arXiv preprint arXiv:1510.08829, 2015.

[21] Sungmin Hwang, Jeesoo Chang, Min-Hye Oh, Kyung Kyu Min, Taejin Jang, Kyungchul Park, Junsu Yu, Jong-Ho Lee, and Byung-Gook Park. Low-latency spiking neural networks using pre-charged membrane potential and delayed evaluation. Frontiers in Neuroscience, 15:135, 2021.

[22] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. arXiv preprint arXiv:1602.07360, 2016.

[23] 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, 2015.

[24] 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.

[25] Jaehyun Kim, Heesu Kim, Subin Huh, Jinho Lee, and Kiyoung Choi. Deep neural networks with weighted spikes. Neurocomputing, 311:373–386, 2018.

[26] Jinseok Kim, Kyungsu Kim, and Jae-Joon Kim. Unifying activation- and timing-based learning rules for spiking neural networks. In Advances in Neural Information Processing Systems (NeurIPS), pages 19534–19544, 2020.

[27] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. arXiv preprint arXiv:1404.5997, 2014.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (NeurIPS), pages 1097–1105, 2012.

[29] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553):436–444, 2015.

[30] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. Frontiers in Neuroscience, 14, 2020.

[31] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. Frontiers in Neuroscience, 10:508, 2016.

[32] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: An event-stream dataset for object classification. Frontiers in Neuroscience, 11:309, 2017.

[33] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In International Conference on Machine Learning (ICML), volume 139, pages 6316–6325, 2021.

[34] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In European Conference on Computer Vision (ECCV), pages 21–37. Springer, 2016.

[35] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In International Conference on Learning Representations (ICLR), 2017.

[36] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. In International Conference on Learning Representations (ICLR), 2018.

[37] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 2015.

[38] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Advances in Neural Information Processing Systems (NeurIPS), pages 2924–2932, 2014.

[39] Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. IEEE Transactions on Neural Networks and Learning Systems, 29(7):3227–3235, 2017.

[40] 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.

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems (NeurIPS), pages 8026–8037, 2019.

[42] Nitin Rathi and Kaushik Roy. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. arXiv preprint arXiv:2008.03658, 2020.

[43] 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.

[44] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 779–788, 2016.

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

[46] 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.

[47] 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.

[48] Ali Samadzadeh, Fatemeh Sadat Tabatabaei Far, Ali Javadi, Ahmad Nickabadi, and Morteza Haghir Chehreghani. Convolutional spiking neural networks for spatio-temporal feature extraction. arXiv preprint arXiv:2003.12346, 2020.

[49] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. Frontiers in Neuroscience, 13:95, 2019.

[50] 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.

[51] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587):484–489, 2016.

[52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations (ICLR), 2015.

[53] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. arXiv preprint arXiv:1505.00387, 2015.

[54] Christoph Stöckl and Wolfgang Maass. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. Nature Machine Intelligence, 3(3):230–238, 2021.

[55] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–9, 2015.

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

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems (NeurIPS), pages 5998–6008, 2017.

[58] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. Frontiers in Neuroscience, 12:331, 2018.

[59] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5987–5995, 2017.

[60] Fu Xing, Ye Yuan, Hong Huo, and Tao Fang. Homeostasis-based cnn-to-snn conversion of inception and residual architectures. In International Conference on Neural Information Processing, pages 173–184. Springer, 2019.

[61] Bojian Yin, Federico Corradi, and Sander M Bohté. Effective and efficient computation with multiple-timescale spiking recurrent neural networks. In International Conference on Neuromorphic Systems, pages 1–8, 2020.

[62] Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. Neural Computation, 33(4):899–925, 2021.

[63] Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. In Advances in Neural Information Processing Systems (NeurIPS), pages 12022–12033, 2020.

[64] 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, volume 35, pages 11062–11070, 2021.

[65] Shibo Zhou, Xiaohua Li, Ying Chen, Sanjeev T. Chandrasekaran, and Arindam Sanyal. Temporal-coded deep spiking neural network with easy training and robust performance. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 11143–11151, 2021.

[66] Romain Zimmer, Thomas Pellegrini, Srisht Fateh Singh, and Timothée Masquelier. Technical report: supervised training of convolutional spiking neural networks with pytorch. arXiv preprint arXiv:1911.10124, 2019.

A 附录

A.1 超参数

对于所有数据集，代理梯度函数为 $\sigma(x) = \frac{1}{\pi}\arctan(\frac{\pi}{2}\alpha x) + \frac{1}{2}$，即 $\sigma'(x) = \frac{\alpha}{2(1+(\frac{\pi}{2}\alpha x)^2)}$，其中 α 是斜率参数。我们为所有神经元设置 $\alpha = 2$、$V_{reset} = 0$ 和 $V_{th} = 1$。优化器是 SGD，动量为 0.9。根据 [62] 的建议，我们在后向计算图中的神经元重置 Eq. (3) 中分离 $S[t]$ 以提高性能。我们使用混合精度训练 [36]，这将加速训练并减少内存消耗，但可能会导致比使用全精度训练稍低的精度。不同数据集的 SNN 的超参数如表 7所示。Tab. 8 显示了 DVS 手势的具有不同元素功能的 SNN 的学习率。三个数据集的数据预处理方法如下：

ImageNet [14] 中使用的数据增强方法也应用于我们的实验中。从图像或其水平翻转中随机采样 224×224 裁剪，并对训练样本进行数据归一化。对测试样本应用 224×224 调整大小和具有数据归一化的中心裁剪。

DVS128 手势 我们使用与 [8] 相同的 AER 数据预处理方法，并利用随机时间删除来缓解过拟合，如 A.2中所示。

CIFAR10-DVS 我们使用与 DVS128 Gesture 相同的 AER 数据预处理方法。我们不使用随机时间删除，因为 CIFAR10-DVS 是通过静态图像获得的。

数据集	学习率调度器	历元	$\boldsymbol{lr}$	批量大小	$\boldsymbol{T}$	$\boldsymbol{n_{gpu}}$
ImageNet	余弦退火 [35]、$T_{max} = 320$	320	0.1	32	4	8
DVS 手势	步进，$T_{step} = 64.\gamma = 0.1$	192	0.1	16	16	1
CIFAR10-DVS	余弦退火，$T_{max} = 64$	64	0.01	16	4, 8, 16	1

表 7: 三个数据集的 SNN 的超参数。

A.2 随机时间删除

为了减少过度拟合，我们针对顺序数据提出了一种简单的数据增强方法，称为 随机时间删除。将序列长度表示为 T，我们随机删除原始序列中的 $T - T_{train}$ 切片并在训练

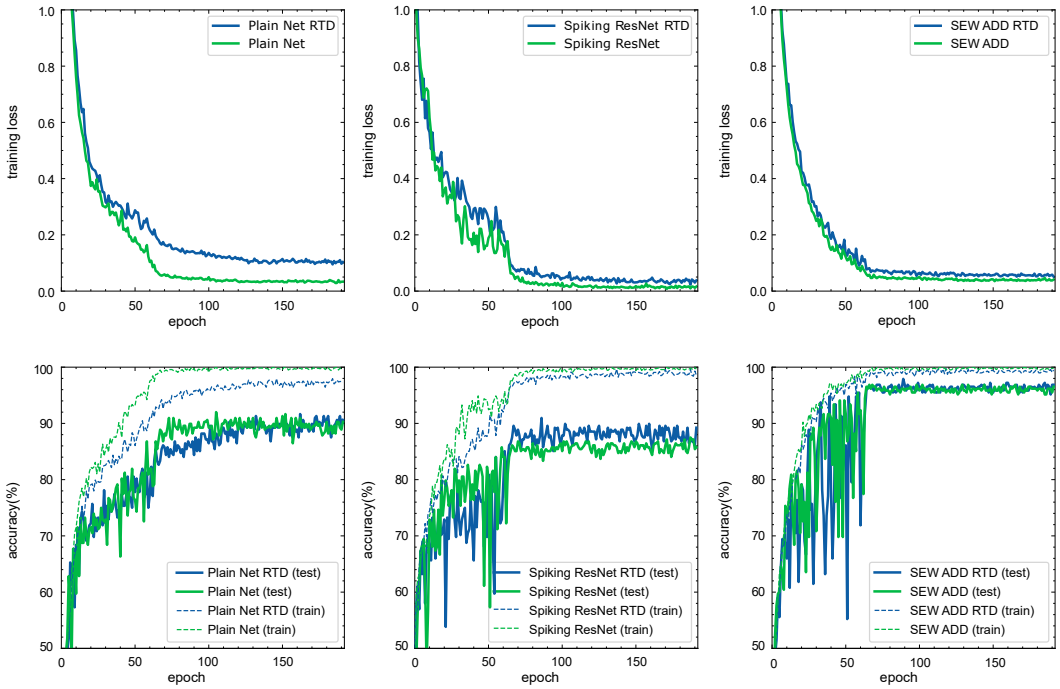


图 10: 有/没有随机时间删除（RTD）的训练损失和训练/测试准确性的比较。

网络	逐元素函数 g	学习率
SEW ResNet	ADD	0.001
SEW ResNet	AND	0.03
SEW ResNet	IAND	0.063
脉冲 ResNet	-	0.1
普通网	-	0.005

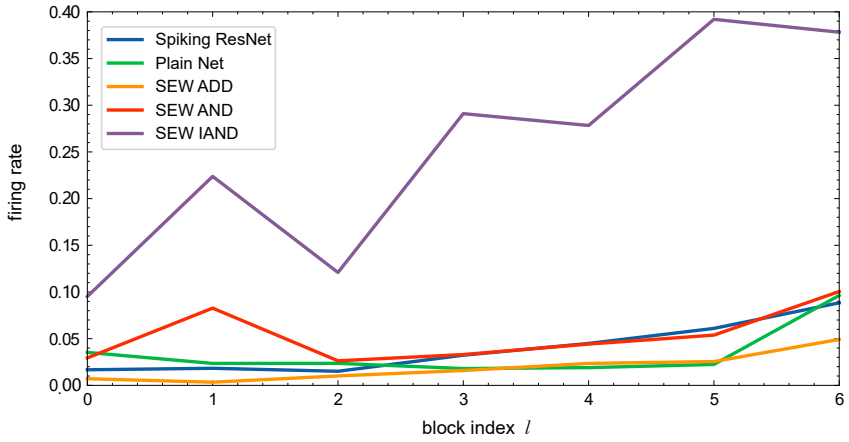
表 8: DVS 手势的 SNN 学习率。

期间使用 T_{train} 切片。在推理过程中，我们使用整个序列，即 $T_{test} = T$。我们在 DVS Gesture 的所有实验中都设置了 $T_{train} = 12, T = 16$。

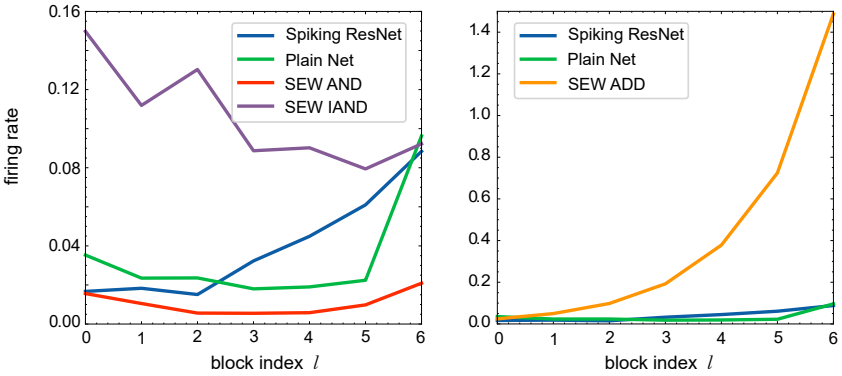
图 10 比较了有或没有 随机时间删除 (RTD) 的 Plain Net、Spiking ResNet 和 SEW ResNet 的训练损失和训练/测试精度。这里逐元素函数 g 是 ADD。可以发现，有 RTD 的网络比没有 RTD 的网络训练损失更高，训练精度更低，因为 RTD 会增加训练的难度。带 RTD 的网络测试精度高于不带 RTD 的网络，表明 RTD 会减少过拟合。三个网络上的结果是一致的，表明 RTD 是一种通用的顺序数据增强方法。

A.3 DVS 手势的发射率

图 11(a) 显示了来自 7B-Net 的 DVS 手势的每个块中 A^l 的触发率。请注意，如果 g 是 AND，则当发射率接近 1 时，SEW 块更接近恒等映射，而对于其他 g，当发射率



(a) DVS Gesture 上每个块中 A^l 的发放率



(b) DVS Gesture 上每个块输出 O^l 的发放率

图 11: DVS 上 7B-Net 各块中最后一个 SN 的发射率和输出 O^l 手势。

接近 0 时，SEW 块变为恒等映射。当所有 SEW 块都变为恒等映射时，7B-Net 将变为 c32k3s1-BN-PLIF-{MPk2s2}*7-FC11，这是一个过于简单的网络，不会导致欠拟合。因此，7B-Net 中的 SEW 块不必是恒等映射。图 11(b) 显示了每个块的输出 O^l 的发射率。由于块是通过最大池连接的，所以发射率不会随着块索引的增加而严格降低，这会挤压稀疏峰值并提高发射率。可以发现 SEW AND 网络中的块具有最低的发射率。SEW IAND 网络中的块比 SEW AND 网络具有更高的触发率，并且 SEW IAND 网络比 SEW AND 网络具有更高的精度（95.49% vs 70.49%），表明使用 IAND 代替 AND 可以缓解第4.1中讨论的沉默问题。

A.4 带有发射率的脉冲 ResNet 中的梯度

SNN 的梯度受到发射率的影响，这就是我们在 Sec.4.1 中分析梯度之前的发射率的原因。考虑一个具有 k 顺序块的脉冲 ResNet 来传输 $S^l[t]$，并且满足恒等映射条件，例如，脉冲神经元是具有 $0 < V_{th} \leq 1$ 的 IF 神经元，那么我们有 $S^l[t] = S^{l+1}[t] = ... =$

$S^{l+k-1}[t] = O^{l+k-1}[t]$。我们得到

$$\frac{\partial O_j^l[t]}{\partial S_j^l[t]} = \frac{\partial \mathrm{SN}(S_j^l[t])}{\partial S_j^l[t]} = \Theta'(S_j^l[t] - V_{th}) \tag{12}$$

$$\frac{\partial L}{\partial S_j^l[t]} = \frac{\partial L}{\partial O_j^l[t]}\Theta'(S_j^l[t] - V_{th}). \tag{13}$$

那么两个相邻块之间的梯度为

$$\frac{\partial L}{\partial O^{l+i}} = \frac{\partial L}{\partial O^{l+i+1}}\Theta'(S^{l+i+1} - V_{th}). \tag{14}$$

将神经元数量表示为 N，将 S^l 的放电率表示为 $\Phi = \frac{\sum_{j=0}^{N-1}\sum_{t=0}^{T-1} S_j^l[t]}{NT}$，则

$$\left\|\frac{\partial L}{\partial S^l}\right\| = \left\|\frac{\partial L}{\partial O^{l+k-1}}\right\| \cdot \left\|\prod_{i=0}^{k-1}\Theta'(S^{l+i} - V_{th})\right\|, \tag{15}$$

在哪里

$$\left\|\prod_{i=0}^{k-1}\Theta'(S^{l+i} - V_{th})\right\| = \sqrt{NT\Phi(\Theta'(1-V_{th}))^{2k} + NT(1-\Phi)(\Theta'(0-V_{th}))^{2k}}$$

$$\to \begin{cases} \sqrt{NT}, & \Theta'(1-V_{th}) = 1, \Theta'(0-V_{th}) = 1 \\ \sqrt{NT\Phi}, & \Theta'(1-V_{th}) = 1, \Theta'(0-V_{th}) < 1 \\ \sqrt{NT(1-\Phi)}, & \Theta'(1-V_{th}) < 1, \Theta'(0-V_{th}) = 1 \\ 0, & \Theta'(1-V_{th}) < 1,\ \Theta'(0-V_{th}) < 1 \\ +\infty, & \Theta'(1-V_{th}) > 1 \text{ or } \Theta'(0-V_{th}) > 1. \end{cases}$$

A.5 0/1 渐变实验

正如3.2中的分析所示，由于累加乘法，Spiking ResNet 中很容易出现梯度消失/爆炸问题。一个可能的解决方案是设置 $\Theta'(0-V_{th}) = \Theta'(1-V_{th}) = 1$。具体来说，我们通过在每个块的最后一个 SN 中设置 $V_{th} = 0.5$ 和 $\sigma'(x) = \frac{1+\frac{\pi^2}{4}}{1+(\pi x)^2}$ 来训练 ImageNet 上的 Spiking ResNet，以确保 $\Theta'(0-V_{th}) = \Theta'(1-V_{th}) = 1$。然而，该网络不会收敛，这可能是由于 SNN 对代理函数敏感所致。

[64] 使用矩形代理函数 $\sigma'(x) = \frac{1}{a}\mathrm{sign}(|x| < \frac{a}{2})$。如果我们设置 $a = 1$，那么 $\sigma'(x) \in \{0, 1\}$。根据方程 (8)，使用该代理函数可以避免 Spiking ResNet 中的梯度爆炸/消失问题。我们在 CIFAR-10 上的 SNN 中比较了不同的代理函数，包括矩形 ($\sigma'(x) = \mathrm{sign}(|x| < \frac{1}{2})$)、ArcTan ($\sigma'(x) = \frac{1}{1+(\pi x)^2}$) 和 Constant 1 ($\sigma'(x) \equiv 1$)。请注意，我们的目标是评估 0/1 梯度，而不是实现 SOTA 精度。因此，我们使用轻量级网络，其结构为 c32k3s1-BN-IF-{{SEW Block (c32)}*2-MPk2s2}*5-FC10。我们还通过替换 SEW 来使用 ADD 作为 g。表 9 中显示了每个代理函数的学习率。

Tab.9表明代理函数的选择对 SNN 的性能有相当大的影响。虽然矩形和常数 1 可以避免方程（8）中的梯度爆炸/消失问题，但它们仍然导致精度较低，甚至使优化不收敛。Tab.9 还表明，SEW ResNet 对代理梯度更加鲁棒，因为它始终比具有相同代理函数的 Spiking ResNet 具有更高的精度。

代理函数	SEW ResNet (ADD)	Spiking ResNet
ArcTan	0.8263	0.7733
矩形	0.8256	0.6601
常数 1	0.1256	0.1

表 9: 使用不同代理函数在 CIFAR-10 上测试 SEW ADD ResNet 和 Spiking ResNet 的准确性。

A.6 再现性

所有实验均使用 SpikingJelly [7] 实现，这是一个基于 PyTorch [41] 的开源 SNN 深度学习框架。源代码可在 https://github.com/fangwei123456/Spike-Element-Wise-ResNet 获取。为了最大限度地提高再现性，我们在所有代码中使用相同的种子。