基于自适应时间步脉冲神经网络的高效图像分类

李千鹏 [1, 2] 贾顺程 [1, 2] 张铁林 [1, 2, 3] 陈 亮 [1, 2]

摘 要 脉冲神经网络 (Spiking neural network, SNN) 由于具有相对人工神经网络 (Artifcial neural network, ANN) 更低的计算能耗而受到广泛关注. 然而, 现有 SNN 大多基于同步计算模式且往往采用多时间步的方式来模拟动态的信息整合过程, 因此带来了推理延迟增大和计算能耗增高等问题, 使其在边缘智能设备上的高效运行大打折扣. 针对这个问题, 本文提出一种自适应时间步脉冲神经网络 (Adaptive timestep improved spiking neural network, ATSNN) 算法. 该算法可以根据不同样本特征自适应选择合适的推理时间步, 并通过设计一个时间依赖的新型损失函数来约束不同计算时间步的重要性. 与此同时, 针对上述 ATSNN 特点设计一款低能耗脉冲神经网络加速器, 支持 ATSNN 算法在 VGG 和 ResNet 等成熟框架上的应用部署. 在 CIFAR10、CIFAR100、CIFAR10-DVS 等标准数据集上软硬件实验结果显示, 与当前固定时间步的 SNN 算法相比, ATSNN 算法的精度基本不下降, 并且推理延迟减少 36.7% ~ 58.7%, 计算复杂度减少 33.0% ~ 57.0%. 在硬件模拟器上的运行结果显示, ATSNN 的计算能耗仅为 GPU RTX 3090Ti 的 4.43% ~ 7.88%. 显示出脑启发神经形态软硬件的巨大优势.

关键词 脉冲神经网络, 低功耗推理, 高效训练, 低延迟

引用格式 李千鹏, 贾顺程, 张铁林, 陈亮. 基于自适应时间步脉冲神经网络的高效图像分类. 自动化学报, 2024, **50**(9): 1724−1735

DOI 10.16383/j.aas.c230656

Adaptive Timestep Improved Spiking Neural Network for Efficient Image Classification

LI Qian-Peng[1, 2] JIA Shun-Cheng[1, 2] ZHANG Tie-Lin[1, 2, 3] CHEN Liang[1, 2]

Abstract Spiking neural network (SNN) has received broad attention for its relatively lower computational energy consumption compared to artificial neural network (ANN). However, most conventional SNNs use a synchronous computation paradigm, whereby multiple timesteps are commonly used to simulate the dynamic process of information integration, resulting in some problems such as extended inference delay and increased computational energy consumption, which lead to a serious efficiency discount during the realistic application of edge intelligent devices. In this paper, we propose an adaptive timestep improved spiking neural network (ATSNN) algorithm, which can automatically choose a proper inference timestep based on different features of input samples, and regulate the importance of different timesteps by designing an innovative time-dependent loss function. Besides, a low energy consumption SNN accelerator is designed based on the characteristics of ATSNN mentioned above to support applications and deployments of ATSNN algorithm on some mature frameworks (such as VGG and ResNet). The results of software and hardware experiments on standard datasets such as CIFAR10, CIFAR100, and CIFAR10-DVS show that, compared to conventional SNN algorithms using static timesteps, the ATSNN algorithm can reach a comparable accuracy but with a decreased inference delay (around 36.7% ~ 58.7%) and reduced computational complexity (around 33.0% ~ 57.0%). Furthermore, the running results on the hardware simulator indicate that the computational energy consumption of ATSNN is only around 4.43% ~ 7.88% of GPU RTX 3090Ti. It shows great advantages of brain-inspired neuromorphic hardware and software.

Key words Spiking neural network (SNN), low power consumption inference, efficient training, low latency

Citation Li Qian-Peng, Jia Shun-Cheng, Zhang Tie-Lin, Chen Liang. Adaptive timestep improved spiking neural network for efficient image classification. *Acta Automatica Sinica*, 2024, **50**(9): 1724−1735

人工神经网络 (Artificial neural network, ANN) 已经被广泛应用于各个场景, 如计算机视觉、自然语言处理、机器人控制、医疗诊断等. 然而, 随着模型参数规模的不断扩大, ANN 需要更多的计

收稿日期 2023-10-25 录用日期 2024-03-15
Manuscript received October 25, 2023; accepted March 15, 2024
国家重点研发计划 (2021ZD0200300) 资助
Supported by National Key Research and Development Program of China (2021ZD0200300)
本文责任编委 杨健
Recommended by Associate Editor YANG Jian
1. 中国科学院自动化研究所 北京 100190 2. 中国科学院大学人工智能学院 北京 101408 3. 中国科学院脑科学与智能技术卓越创新中心 上海 200031

1. Institute of Automation, Chinese Academy of Sciences, Beijing 100190 2. School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 101408 3. Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai 200031

算能耗以支持实现更复杂的应用任务, 这无疑限制了边缘智能设备的应用推广[1]. 尽管研究人员已经开展了关于网络量化、结构剪枝以及知识蒸馏等模型压缩技术研究, 但功耗问题依然十分突出.

作为第三代人工神经网络的脉冲神经网络 (Spiking neural network, SNN), 脉冲式异步编码特征为高能效计算提供了一种生物合理的解决方案, 支持多尺度和多维度的时空信息高效处理. 与以浮点运算为主的 ANN 相比, SNN 通过单比特的激活值避免了高能耗的浮点乘法操作, 从而实现高能效计算; 同时 SNN 使用具有丰富时间维度信息的生物动力学神经元, 如采用泄漏整合发放 (Leaky-integrated and fire, LIF) 神经元模型, 可以帮助动态积累时序输入信息, 并且只有当膜电位超过阈值时才会编码异步事件脉冲. SNN 在保持生物合理性的同时展示出了计算高效性, 并在许多人工智能任务上展现出与 ANN 相近的精度表现. 这些关键特性使得面向 SNN 的类脑芯片成为研究热点, 如 TrueNorth[2]、Loihi[3] 和 TianjiC[4] 等类脑芯片, 这些神经形态芯片具有低延迟、低功耗和并行度高的特点[5].

然而, 在现有计算设备上模拟时序动态信息需要多时间步计算, 而为了保证 SNN 的运算精度, 这个多时间步的步长又往往设置的很高, 这就导致了推理延迟过大和推理能耗过高的问题. 如现有的 SNN 学习方法部分采用先 ANN 训练再 SNN 转换的方式, 在较为复杂的数据集任务或深层网络结构上, 通常需要几十到数百个时间步才能获得比较好的运算精度, 因此这类 SNN 模型的部署能耗很高. 此外, 部分学者提出时空依赖的反向传播方法 (Spatio-temporal backpropagation, STBP)[6], 可以采用反向传播的微调方法来直接训练 SNN, 能够降低计算时间步到十几个量级. 由此可见, 无论是转换方法还是直接训练方法, SNN 都需要运行多个时间步, 这不可避免地增加了计算延迟与能量消耗, 对边缘设备的低功耗推理带来了挑战; 另一方面, 现有计算设备如 GPU, 无法充分利用 SNN 稀疏二值脉冲特性, 仍然在使用高能耗的浮点乘法计算. 设计一款更适用于 SNN 量化计算本身的硬件加速器也迫在眉睫.

针对上述挑战, 本文从 SNN 的学习方法、计算能耗、硬件部署等实际应用需求出发, 提出一系列降低 SNN 软硬件部署能耗的方式方法, 并在实际的标准分类任务上测试通过, 本文的主要贡献如下:

1) 提出一种自适应时间步 (Adaptive timestep, AT) 选择算法. 通过关联输出层置信度和推理所需要的时间步, 支持更加精准地获得样本正确分类所需最小时间步, 可以有效降低平均推理延迟和能耗.

2) 提出一种表征不同时间步重要性的损失函数 L_{imp}. 通过设置时间步越低重要性越高的基本原则, 并使用不重置膜电位的神经元组成输出层, 保留输出层的动态时间信息. 与基于标准交叉熵损失函数的 SNN 相比, 本文在仅使用 1 个时间步时的推理准确率就可以提升 27% ~ 55%.

3) 设计一款低能耗脉冲神经网络加速器以进一步降低能耗. 在硬件模拟器上开展 CIFAR10、CIFAR100、CIFAR10-DVS 数据集能耗实验, 自适应时间步脉冲神经网络 (Adaptive timestep improved spiking neural network, ATSNN) 算法在不降低精度的情况下, 推理延迟减少 36.7% ~ 58.7%、计算复杂度减少 33.0% ~ 57.0%, 并且能耗仅为 GPU RTX 3090Ti 的 4.43% ~ 7.88%, 显示出边缘端智能应用的巨大优势.

1　已有工作概述

Izhikevich[7] 指出, 神经元模型的生物可解释性越高, 其计算复杂度越大. LIF 神经元模型因为计算复杂度低以及具有生物合理性, 成为 SNN 最常使用的一种神经元模型. 原始的 LIF 模型使用微分方程表达, 通常使用的 LIF 模型欧拉展开迭代表达式如式 (1) ~ (2) 所示

$$\boldsymbol{u}_{t+1}^{l}=\tau \boldsymbol{u}_{t}^{l}\left(\boldsymbol{1}-\boldsymbol{s}_{t}^{l}\right)+\boldsymbol{W}^{l} \boldsymbol{s}_{t+1}^{l-1} \tag{1}$$

$$(\boldsymbol{s}_{t+1}^{l})_i=\begin{cases}1, & (\boldsymbol{u}_{t+1}^{l})_i>V_{th}\\ 0, & \text{otherwise}\end{cases} \tag{2}$$

式中, $\boldsymbol{u}_{t+1}^{l}$ 是在时间步 $t+1$ 时第 l 层的神经元膜电位, $(\boldsymbol{u}_{t+1}^{l})_i$ 是在时间步 $t+1$ 时第 l 层的第 i 个神经元的膜电位, τ 是膜电位衰减系数, $\boldsymbol{W}^{l}$ 是第 l 层的权重, $(\boldsymbol{s}_{t+1}^{l-1})_i$ 是在时间步 $t+1$ 时第 $l-1$ 层的第 i 个神经元发放的脉冲, V_{th} 是发放脉冲的膜电位阈值.

SNN 每个时间步的计算与 ANN 十分相似, 只需将 ANN 的激活函数换成具有时间信息的生物神经元模型. 处理数据集样本时, 需要通过速率编码、时间编码、相位编码等方法添加时间维度信息[8], 或者将数据样本重复输入多次. 假设时间步 t 时第 i 个样本的输入数据为 $\boldsymbol{x}_t^i$, SNN 的前向推理公式可以定义为

$$f_t(\boldsymbol{x}_t^i)=h\circ g^{L}\circ g^{L-1}\circ\cdots\circ g^{1}(\boldsymbol{x}_t^i) \tag{3}$$

式中, $f_t(\boldsymbol{x}_t^i)$ 是在时间步 t 时网络的输出, h 是网络的输出层, g^L 是第 L 个包含卷积池化层、全连接层

和 LIF 神经元的模块, $\circ$ 表示函数的复合.

由于 SNN 的输出是脉冲序列, 需要对脉冲序列解码来完成后续任务. 常见的解码方式有第一峰值解码、计数解码[9]. 对于分类任务, 输出层神经元数目和类别数相同, 第一峰值解码选择第一个出现脉冲的神经元所在类为预测类别; 计数解码统计多个时间步输出层神经元的脉冲数, 选择脉冲数最多的神经元所在类为预测类别. 此外, 也有些模型如 EfficientLIF-Net[10], 将神经元输出层替换成全连接层, 统计多个时间步全连接输出层的累计值, 选择累计值最大的神经元所在类为预测类别.

STBP 是训练 SNN 常用且有效的方法. 为解决脉冲发放函数不可导的问题, STBP 定义了一系列函数来逼近发放函数的导数, 如式 (4) 所示, 式中 α 是一个超参数. 通过梯度近似, SNN 的梯度能够在时间和空间两个尺度进行传播, 使网络易于快速收敛.

$$\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{u}} = \frac{1}{\sqrt{2\pi\alpha}} \mathrm{e}^{-\frac{(\boldsymbol{u}-\boldsymbol{V}_{th})^2}{2\alpha}} \tag{4}$$

在直接训练低延迟网络方面, Bu 等[11] 提出膜电位的最佳初始化策略, 不仅实现高精度 ANN 到 SNN 转换, 还能缩短推理延迟. Jiang 等[12] 提出 KLIF 神经元模型, 在训练过程中动态调整代理梯度函数的斜率和宽度. Fang 等[13] 提出基于 SEW (Spike element wise) 的残差块实现 SNN 中的残差学习, 克服梯度爆炸问题, 以较少时间步实现高准确率. Zheng 等[14] 提出阈值依赖的批归一化, 在时间和空间两个维度归一化输入特征. Rathi 和 Roy[15] 提出训练神经元膜电位衰减系数和发放阈值的算法. Guo 等[16] 提出信息最大化损失函数和动态调整代理梯度大小方案, 实现训练后期更为准确的反向梯度以及低延迟性能.

除了直接训练低延迟 SNN 外, 也有学者通过执行部分时间步获得较高准确率. 在 ANN 领域有早期退出 (Early exit) 策略, 如 BranchyNet[17] 通过在网络某些层加入分类器, 当分类器的置信度大于预期设定的阈值时, 就选择当前分类器的结果作为网络的输出, 否则网络继续向前传播直至后续分类器的置信度大于阈值或者传播至输出层. 早期退出策略存在早期退出分支多、网络参数多、分类器阈值难于调整等问题.

与早期退出策略相似的是时间步选择策略. Kim 和 Panda[18] 提出在时间维度归一化输入特征, 并使用神经元随时间步变化的发放阈值作为早期退出条件. Li 等[19] 提出的 DTSNN 在选择时间步时, 计算当前时间步样本的归一化信息熵, 选择熵小于预期设定阈值的最小时间步作为该样本的推理时间步. DTSNN 的动态选择时间步算法如式 (5) ~ (7) 所示

$$\pi_t^i(\boldsymbol{x}) = \frac{\exp\left(\frac{1}{t}\sum\limits_{k=1}^{t}(f_k(\boldsymbol{x}_k))_i\right)}{\sum\limits_{j=1}^{C}\exp\left(\frac{1}{t}\sum\limits_{k=1}^{t}(f_k(\boldsymbol{x}_k))_j\right)} \tag{5}$$

$$E_f^t(\boldsymbol{x}) = -\frac{1}{\lg C}\sum_{i=1}^{C}\pi_t^i(\boldsymbol{x})\lg(\pi_t^i(\boldsymbol{x})) \tag{6}$$

$$\hat{T} = \arg\min_{\hat{T}}\{E_f^{\hat{T}}(\boldsymbol{x}) < \theta | 1 \le \hat{T} < T\} \cup \{T\} \tag{7}$$

式中, C 是类别数, $\pi_t^i(\boldsymbol{x})$ 是样本 $\boldsymbol{x}$ 在时间步 t 时预测为第 i 类的概率, $E_f^t(\boldsymbol{x})$ 是样本 $\boldsymbol{x}$ 运行 t 个时间步时累积的熵, θ 是选择时间步的阈值, $\hat{T}$ 是熵小于 θ 的最小时间步, T 是最大时间步.

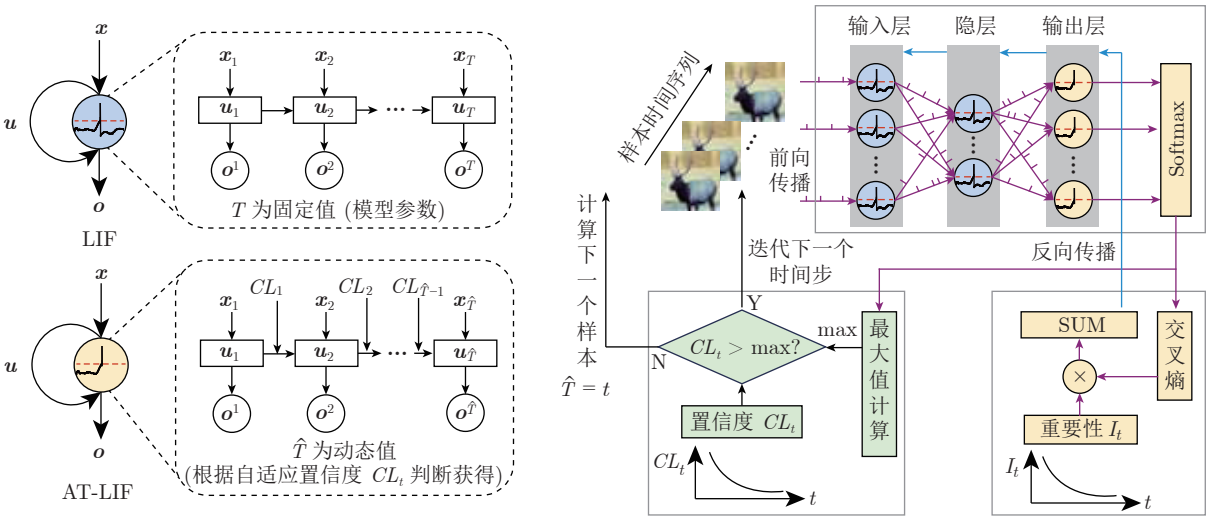
使用网络剪枝、减少脉冲发放次数与量化技术, 通常会有效降低 SNN 能耗. Deng 等[20] 通过在损失函数中引入脉冲发放率正则化项来降低脉冲发放率. Stöckl 和 Maass[21] 提出具有反馈通路的神经元来减少脉冲数目. Chen 等[22] 通过定义权重梯度来权衡连接关系剪枝与生长, 获得稀疏网络结构. Eshraghian 等[23]、Wang 等[24] 分别将网络权重量化为 4 比特和 2 比特.

2 ATSNN 算法

传统的 SNN 使用固定时间步长推理样本, 这带来较大的推理延迟和能量消耗. 我们发现部分样本使用较小时间步推理依然可以正确预测类别信息, 而对所有样本使用相同的低时间步推理性能不佳, 同时网络在极低时间步的推理性能较差.

为此, 本文提出自适应时间步脉冲神经网络, 提出不重置神经元膜电位的输出层、低时间步更重要的损失函数 L_{imp} 和自适应时间步选择算法. 图 1(a) 是 LIF 神经元模型与 ATSNN 输出层等价的神经元模型, 相较于传统的使用固定时间步长的神经元计算过程, ATSNN 的输出层神经元膜电位不重置, 同时根据置信度 CL_t 自适应地选择合适的推理时间步, 无需运行所有的时间步; 图 1(b) 是 ATSNN 训练和动态时间步推理过程, 黄色部分为本文的神经元及损失函数, 绿色部分为自适应时间步选择模块.

本文模型一方面在固定的低时间步推理的准确率得到了大幅提升, 另一方面能够在推理过程中自适应选择合适的时间步而不损失网络性能. 此外, 本文也针对 ATSNN 设计一款低功耗脉冲神经网络加速器, 并在硬件模拟器上验证 ATSNN.



(a) 神经元结构
(a) Structure of neuron

(b) 训练推理流程
(b) Training and inference process

图 1　ATSNN 结构及训练和动态时间推理流程图

Fig.1　ATSNN structure and training and dynamic time inference flow chart

2.1　不重置膜电位的输出层

低延迟模型如 SEWResNet[13] 以及 DTSNN[19] 均使用全连接层并使用式 (5) 表达网络的输出. STBP[6] 使用 LIF 神经元全连接层并将该层脉冲在多个时间步的累加值作为网络的输出. 第一种方案的输出层将带有时间信息的脉冲序列直接转化为实数空间的值, 损失了时间信息; 第二种方案将实数空间的神经元膜电位量化为脉冲序列并统计脉冲数, 信息表达能力大幅下降. 考虑到重置膜电位会丢失部分时间信息, 为了在输出层保留更多的信息, 并且不降低网络输出信息表达能力, 本文提出使用不重置膜电位的简化 LIF 模型来保留时间信息和降低量化误差, 并统计膜电位的 Softmax 累积值作为网络的输出, 如式 (8) ~ (10) 所示

$$\boldsymbol{u}_{t+1}^{l}=\tau\boldsymbol{u}_{t}^{l}+\boldsymbol{W}^{l}\boldsymbol{s}_{t+1}^{l-1} \tag{8}$$

$$\boldsymbol{o}_{i}^{t}=\frac{\exp((f_t(\boldsymbol{x}_t))_i)}{\sum_{j=1}^{C}\exp((f_t(\boldsymbol{x}_t))_j)} \tag{9}$$

$$\boldsymbol{y}_t=\frac{1}{t}\sum_{k=1}^{t}\boldsymbol{o}^{k} \tag{10}$$

式中, $(f_t(\boldsymbol{x}_t))_i$ 是在时间步 t 时最后一层的第 i 个神经元的膜电位, $\boldsymbol{x}_t$ 代表当前时刻输入网络的数据, $\boldsymbol{o}_i^t$ 是在时间步 t 时网络预测为第 i 类的概率; $\boldsymbol{y}_t$ 是 t 个时间步的累积输出向量, 在训练时根据 $\boldsymbol{y}_T$ 最大值所在位置作为当前样本的类别. 相比于式 (1) ~ (2) 的基础 LIF 模型, 本文的神经元模型没有重置膜电位和发放脉冲操作.

通过使用本文提出的输出层, 一方面可以保持 SNN 丰富的时间信息, 另一方面使用了膜电位而不是神经元脉冲序列, 这将保持网络输出表达空间、消除膜电位转换脉冲带来的量化误差并提升网络分类置信度.

2.2　低时间步更重要的损失函数 L_{imp}

SNN 输出层的信息随着时间进行累积, 时间步越大输出层的信息量越大. 现有的 SNN 大多采用交叉熵损失函数, 这将平等地对待每个时间步网络的输出, 造成使用固定的低时间步推理网络准确率低的问题. 本文希望增加低时间步网络的信息量, 以此提升直接固定低时间步网络的推理性能. 为此本文设计表征低时间步更重要的函数, 如式 (11) 所示, 时间步越小重要性越高, 并将重要性函数与交叉熵函数结合, 提出低时间步更重要的损失函数 L_{imp}, 如式 (13) 所示

$$I_t=1+\exp\left(-1-\frac{t+1}{T}\right) \tag{11}$$

$$D=\sum_{t=1}^{T}I_t \tag{12}$$

$$L_{\text{imp}}=-\frac{1}{B}\sum_{t=1}^{T}\frac{I_t}{D}\sum_{i=1}^{C}\boldsymbol{z}_i\lg(\boldsymbol{o}_i^t) \tag{13}$$

式中, I_t 为第 t 个时间步的重要性, D 是归一化系数, $\boldsymbol{o}_i^t$ 是在时间步 t 时网络预测为第 i 类的概率, C

是类别数, $\boldsymbol{z}_i$ 是输入样本属于第 i 类的概率, B 是批次大小.

基于本文的输出层以及损失函数, 使用 STBP 进行模型训练, 并使用式 (14) 所示的代理梯度函数

$$\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{u}} = \max\left(\boldsymbol{0}, \boldsymbol{1} - \frac{|\boldsymbol{u} - \boldsymbol{V}_{th}|}{\boldsymbol{V}_{th}}\right) \tag{14}$$

2.3 自适应时间步选择算法

随着时间步的增加, 网络的性能也会随之增加. 对于最大时间步为 T 的 SNN, 对每个样本使用不同的时间步依然能获得相同的准确率. 经过 Softmax 函数归一化的时间步 t 时的输出 $\boldsymbol{o}^t$, 如果 $\boldsymbol{o}_c^t$ 越接近于 1, 这个样本属于第 c 类的可能性就越大. 考虑到本文提出的损失函数 L_{imp} 会使长时间步的输出重要性减弱, 为此提出基于可变置信度的自适应时间步选择算法, 如式 (15)、式 (16) 所示

$$CL_t = CL_{\text{init}} \times \tau_{CL}^{t-1} \tag{15}$$

$$\hat{T} = \min(\{t|\max(\boldsymbol{o}^t) > CL_t, 1 \le t < T\} \cup \{T\}) \tag{16}$$

式中, 超参数 CL_{init} 是初始化置信度, 标量 τ_{CL} 是置信度 CL_t 的衰减系数, $\hat{T}$ 是选择的时间步, $\boldsymbol{o}^t$ 是时间步 t 时网络的输出. 若当前时间步网络输出的最大值小于置信度 CL_t, 认为此时网络的输出并不可信, 需要进行下一个时间步的计算, 直至获得可信的结果. 由于式 (11) 的时间步重要性随时间增大而减小, 置信度 CL_t 变化趋势应与时间步重要性保持一致, 如式 (15) 所示.

2.4 脉冲神经网络加速器

脉冲神经网络加速器可以对 SNN 低功耗推理. 图 2(a) 为 SNN 常用的加速器, 由于神经元膜电位更新需要衰减操作, 因此使用了带有高能耗乘法的处理元件 (Process elements, PE) 用于膜电位计算, 同时 SNN 使用固定时间步进行推理, 没有时间步选择模块.

图 2(b) 是 DTSNN 的硬件加速器, 在实现方式上, 采用数模混合的方式: PE 更新膜电位时使用模拟信号, DT 选择动态时间步时使用数字信号. 由于加速器采用了大量的模数转换器 (Analog to digital converter, ADC), 引入了额外的功耗开销, 芯片面积也会明显增大. 动态时间步选择模块 DT 为全数字实现, 通过使用在 6 KB 存储器查表的方式避免复杂的指数、对数运算.

本文设计脉冲神经网络加速器, 并实现所提出的自适应时间步选择算法. 图 2(c) 是本文加速器的

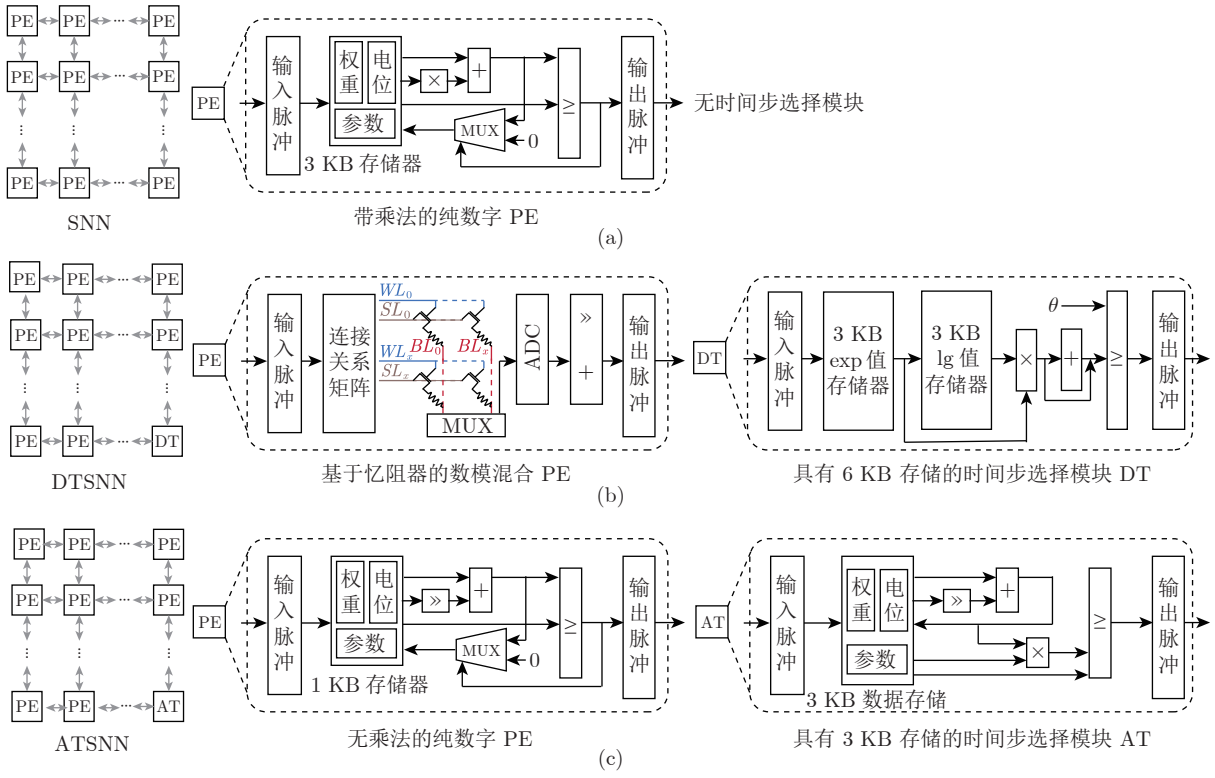


图 2 加速器架构 ((a) 常规加速器; (b) DTSNN 的加速器; (c) 本文加速器)

Fig. 2 Accelerator architecture ((a) Conventional accelerator; (b) Accelerator of DTSNN; (c) Our accelerator)

架构, 阵列大小为 16 × 16, 包含 255 个 PE 单元、1 个 AT 单元. PE 和 AT 均有 128 B 的输入脉冲缓冲器和 128 B 的输出脉冲缓冲器; PE 和 AT 数据存储器的大小分别为 1 KB、3 KB.

PE 是完成神经元计算、发放脉冲的处理单元, 完成输入电流累积、膜电位更新、发放脉冲三个操作. 针对式 (1) 中膜电位衰减需要高能耗的乘法计算, 本文使用移位和加法计算代替乘法计算, 从而降低加速器的功耗.

AT 单元用于实现 ATSNN 的输出层以及 AT 算法, 由于不重置膜电位, 相较于 PE 少了一个多路选择器 (Multiplexer, MUX), 同时加入了一个乘法器用于实现 AT 算法. 为降低硬件功耗, 将 AT 算法修改为硬件友好型, 如式 (17)、式 (18) 所示

$$\boldsymbol{o}_i^t = \exp((f_t(\boldsymbol{x}_t))_i) \tag{17}$$

$$CL_t = CL_{\text{init}} \times \tau_{CL}^{t-1} \times \sum_{j=1}^{C} \exp((f_t(\boldsymbol{x}_t))_j) \tag{18}$$

相较于式 (9) 和式 (15), 硬件上的 AT 算法在计算输出时, 将高能耗与高延迟的除法计算转换为乘法计算; 同时在硬件设计上, 通过对在数据存储参数区预存的指数值线性插值, 避免式 (17) 中高能耗的指数计算. 相较于 DTSNN 的加速器, 本文的加速器避免大量 ADC 带来的额外功耗开销, 同时具有更小的存储器开销.

3　实验结果与分析

3.1　数据集与实验设置

本文使用四个静态图像数据集和两个动态数据集训练并测试网络. 静态数据集 CIFAR10 和 CIFAR100[25], 每个数据集均有尺寸为 3 × 32 × 32 的 50 000 张训练图像和 10 000 张测试图像, 类别数分别为 10 和 100. TinyImageNet[26] 总共有 200 类尺寸为 3 × 64 × 64 的 100 000 张训练图像和 10 000 张测试图像. ImageNet-100 数据集是 ImageNet-1K[27] 的子集, 具有 100 类尺寸为 3 × 224 × 224 的 126 689 张训练图像和 5 000 张测试图像. 动态数据集 CIFAR10-DVS[28] 通过使用 DVS 相机将 10 000 个 CIFAR10 图像转换成数据流, 相机的空间分辨率为 128 × 128, 数据集的类别数为 10. 动态数据集 N-Caltech-101[29] 有 101 类共 8 709 个数据流, 相机空间分辨率为 240 × 180.

基于 NVIDIA GeForce RTX 3090Ti 训练并推理本文提出的算法, 网络结构使用 VGG16 和 ResNet19. 对于静态数据集, 使用均值和方差归一化图像. 对于数据集 CIFAR10-DVS, 由于原数据集没有划分训练集和测试集, 本文参照 DTSNN 将其划分为 9 000 个训练样本和 1 000 个测试样本; 对于数据集 N-Caltech-101, 参照 NDA[30] 划分数据集, 并使用 Spikingjelly[31] 将动态数据集转换成与相机画幅相同的双通道图像帧.

对于所有的数据集和网络, 均使用随机梯度下降 (Stochastic gradient descent, SGD) 优化器, 权值衰减设置为 0.000 5, 动量设置为 0.9. 初始学习率为 0.1, 使用余弦退火方法调整学习率; 批大小设置为 256, 最大迭代次数设置为 300. 本文首先使用固定时间步对网络进行训练, 然后使用提出的时间步选择算法对训练后的网络选择合适的时间步. 网络的性能有三个评价指标: 准确率、吞吐率和复杂度. 复杂度 E 计算如式 (19) 所示

$$E = \sum_{l=1}^{L} fr^{l-1} \times T \times conn^l \tag{19}$$

式中, fr^{l-1} 为第 $l-1$ 层的脉冲发放率, $conn^l$ 为第 l 层每个输入神经元实际连接的权重数目.

3.2　ATSNN 获得优于 DTSNN 精度

为验证本文提出的 ATSNN 的有效性, 将本文的算法与未使用 ATSNN 模块的 SNN 和 Li 等[19] 提出的 DTSNN 进行对比. 本文使用相同的参数训练 SNN、DTSNN 和 ATSNN, 对比推理所用的平均时间步和复杂度, 并将 SNN 的复杂度归一化以作为基准.

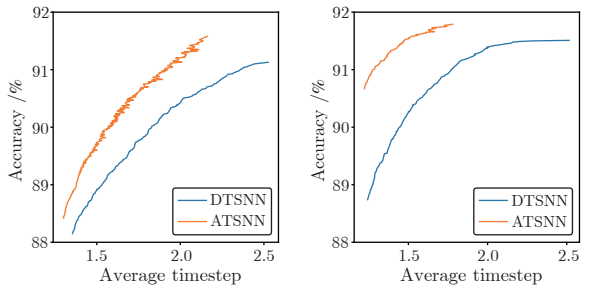
表 1 给出各个模型在三个数据集上的性能, 表中时间步一列如 2.53 (4) 表示推理平均时间步为 2.53, 最大时间步为 4. 从表 1 可以看出, 在 CIFAR10 数据集上, ATSNN 只需要 2.19 和 2.00 个时间步就能达到另外两个模型相当的准确率, 复杂度仅为 SNN 的 67%; 在 CIFAR100 数据集上, ATSNN 只需要 2.49 和 2.53 个时间步就能获得最好的准确率, 复杂度仅为 SNN 的 64%; 而在 CIFAR10-DVS 数据集上, ATSNN 的准确率接近其他两个算法, 但平均时间步和复杂度最小.

通过调整时间步选择模块的参数, 获得 DTSNN 和 ATSNN 的准确率随平均推理时间步变化曲线. 如图 3 所示, 可以看到在相同准确率下, ATSNN 具有更低的平均推理时间步, 因此推理延迟更低; 在相同的平均推理时间步下, ATSNN 具有更高的准确率. 如图 3(d) 所示, 在平均时间步均约为 2.4 的情况下, ATSNN 准确率高于 DTSNN. 通过以上

表 1 SNN、DTSNN 和 ATSNN 在时间步、准确率和复杂度的对比

Table 1 Comparison of SNN, DTSNN and ATSNN in timestep, accuracy and complexity

网络结构	算法	CIFAR10			CIFAR100			CIFAR10-DVS		
		时间步	准确率 (%)	复杂度	时间步	准确率 (%)	复杂度	时间步	准确率 (%)	复杂度
VGG16	SNN	4.00	91.35	1.00	4.00	66.99	1.00	10.00	72.60	1.00
	DTSNN	2.53 (4)	91.13	0.79	3.58 (4)	69.68	0.96	5.70 (10)	73.30	0.56
	ATSNN	2.19 (4)	91.61	0.67	2.49 (4)	70.05	0.63	4.13 (10)	73.00	0.43
ResNet19	SNN	4.00	91.86	1.00	4.00	67.22	1.00	10.00	70.00	1.00
	DTSNN	2.52 (4)	91.51	0.88	3.54 (4)	67.58	1.02	6.95 (10)	70.50	0.94
	ATSNN	2.00 (4)	91.84	0.67	2.53 (4)	70.64	0.64	5.57 (10)	69.50	0.63

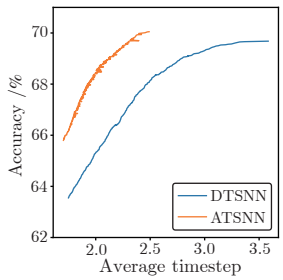


(a) 在 CIFAR10 上使用 VGG16 的结果

(a) The results on CIFAR10 by using VGG16

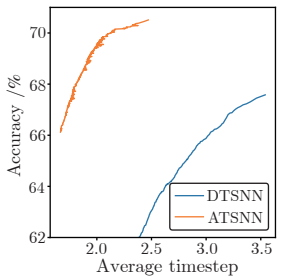
(b) 在 CIFAR10 上使用 ResNet19 的结果

(b) The results on CIFAR10 by using ResNet19



(c) 在 CIFAR100 上使用 VGG16 的结果

(c) The results on CIFAR100 by using VGG16



(d) 在 CIFAR100 上使用 ResNet19 的结果

(d) The results on CIFAR100 by using ResNet19

图 3 不同平均推理时间步时的准确率

Fig. 3 Accuracy at different average inference timesteps

结果与分析, ATSNN 在准确率和推理延迟上优于 DTSNN.

此外, 为进一步探究 ATSNN 和 DTSNN 的计算耗能情况, 在 ImageNet-100 和 N-Caltech-101 两个数据集上对比每个时间步 ATSNN 相对 DTSNN 的计算复杂度和网络性能, 对比结果如表 2 所示. 表中 T1 表示第一个时间步的相对计算复杂度, $\tilde{T}$ 是使用时间步选择算法后 ATSNN 相对于 DTSNN 的平均推理时间步, ΔACC 是 ATSNN 相对于 DTSNN 提升的准确率. 从表中可以看出在两个数据集上 ATSNN 能够取得更好的准确率, 同时 ATSNN 的计算复杂度和 DTSNN 相差不大, 即两个算法每个时间步计算耗能相似. 由于 ATSNN 使用更少的时间步进行推理, 故计算耗能更低, 计算效率更高.

3.3 有效的损失函数 L_{imp}

ATSNN 的损失函数 L_{imp} 期望低时间步网络输出的重要性更高. 为测试损失函数的效果, 在 SNN 的基础上, 仅将损失函数替换为交叉熵 (Cross entropy) 函数、DTSNN 模型的损失函数 (DTSNN-loss) 和本文的损失函数 (L_{imp}) 并训练网络. 然后所有网络均不使用时间步选择算法, 仅使用固定的时间步 (1、2、3、4) 测试网络性能, 结果如图 4 所示.

从图 4 中可以看出, 使用本文的损失函数 L_{imp} 在 1 个时间步的推理性能优于另外两个损失函数, 这为后续选择时间步算法实现低推理延迟且无准确率损失奠定了基础. 由此可见, 本文的损失函数 L_{imp} 提升了网络在低时间步的表达能力.

3.4 低能耗 SNN 硬件加速器

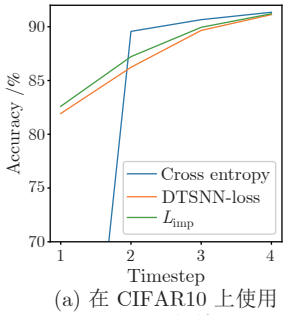
本文使用 SNN 硬件模拟器 SATA-Sim[32] 实现脉冲神经网络加速器, 使用 4 个工作在 28 nm 工艺、500 MHz 时钟频率条件的并行加速器运行本文模型, 同时也与 GPU 对比总耗能、帧率 (Frames per second, FPS).

实验结果如表 3 所示, 表 3 最后一列的数值如 606.1 (5.46%) 表示本文加速器总耗能 606.1 J, 仅为 GPU 耗能的 5.46%. 可以明显看到, 本文的加速器能够达到 GPU 相同的推理速度, 但仅消耗 4.43% ~ 7.88% GPU 的能量. 同时本文也分析 AT 单元的功耗, 由于最后一层神经元数目少, AT 单元的操作数远少于 PE 单元, 所以尽管有乘法器的存在, AT 单元的总功耗不及 PE 的千分之一. 因此本文的硬件平台具有显著的能耗优势, 并且 AT 算法不会引入较大的资源消耗, 非常适合部署在终端智能设备.

表 2　每个时间步的相对计算复杂度及网络性能

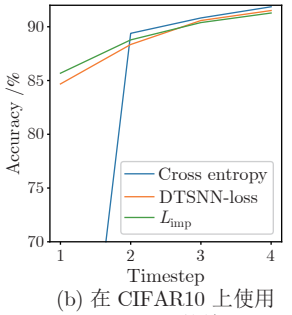
Table 2　Relative computational complexity per timestep and network performance

数据集	T1	T2	T3	T4	T5	$\tilde{T}$ (%)	ΔACC (%)
ImageNet-100	0.998	1.009	1.009	1.010	1.003	68.49	0.90
N-Caltech-101	0.959	0.977	0.978	0.959	0.989	68.32	0.45



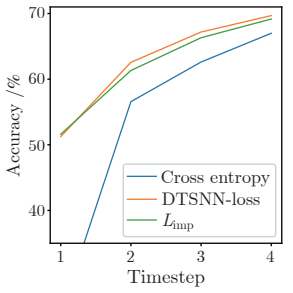
(a) 在 CIFAR10 上使用 VGG16 的结果

(a) The results on CIFAR10 by using VGG16



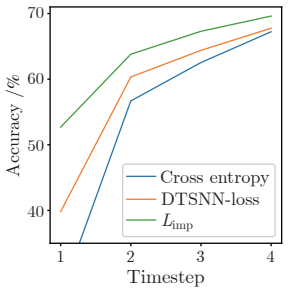
(b) 在 CIFAR10 上使用 ResNet19 的结果

(b) The results on CIFAR10 by using ResNet19



(c) 在 CIFAR100 上使用 VGG16 的结果

(c) The results on CIFAR100 by using VGG16



(d) 在 CIFAR100 上使用 ResNet19 的结果

(d) The results on CIFAR100 by using ResNet19

图 4　固定时间步时使用三种损失函数的准确率

Fig. 4　Accuracy by using three loss functions at fixed timestep

表 3　加速器与 GPU 关于 FPS、耗能对比

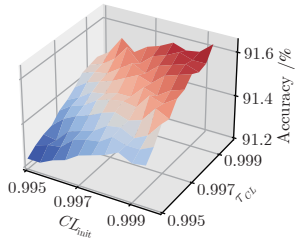
Table 3　Comparison of FPS and energy consumption between accelerator and GPU

网络结构	数据集	GPU		本文加速器	
		FPS	耗能 (J)	FPS	耗能 (J)
VGG16	CIFAR10	215	6 814	248	537.6 (7.88%)
	CIFAR100	200	7 499	228	543.6 (7.24%)
ResNet19	CIFAR10	215	11 088	212	606.1 (5.46%)
	CIFAR100	172	13 914	180	617.2 (4.43%)

3.5　超参数的影响分析

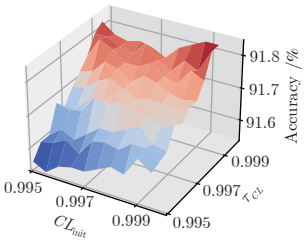
本文的自适应时间步选择算法有两个超参数 CL_{init} 和 τ_{CL}, 在 CIFAR10 和 CIFAR100 数据集上遍历两个超参数分别从 0.995 0 到 0.999 8, 步长为 0.000 2, 统计不同参数对算法准确率以及平均时间步的影响.

图 5、图 6 分别为不同超参数条件下的准确率和平均时间步. 从图中可以看到, 随着两个超参数的不断增大, 算法的准确率呈增大趋势, 同时所有算法的平均时间步也在提升. 这是因为当 CL_{init} 变大、τ_{CL} 变大时, ATSNN 输出层需要更高的置信度才能退出选择时间步模块, 在提升推理准确率的同时也带来了时间步的增加. 超参数的选择会对网络的性能带来一定的影响, 为此设置 CL_{init} 和 τ_{CL} 均为 0.999, 此时 ATSNN 的性能如表 4 所示. 与表 1 的结果相比, 本文的算法仍然具有较低的平均时间步和较高的准确率.



(a) 在 CIFAR10 上使用 VGG16 的结果

(a) The results on CIFAR10 by using VGG16

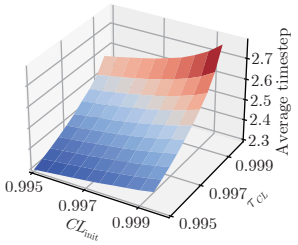


(b) 在 CIFAR10 上使用 ResNet19 的结果

(b) The results on CIFAR10 by using ResNet19

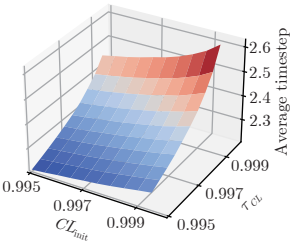
图 5　超参数对准确率的影响

Fig. 5　The effect of hyperparameters on accuracy



(a) 在 CIFAR10 上使用 VGG16 的结果

(a) The results on CIFAR10 by using VGG16



(b) 在 CIFAR10 上使用 ResNet19 的结果

(b) The results on CIFAR10 by using ResNet19

图 6　超参数对时间步的影响

Fig. 6　The effect of hyperparameters on timestep

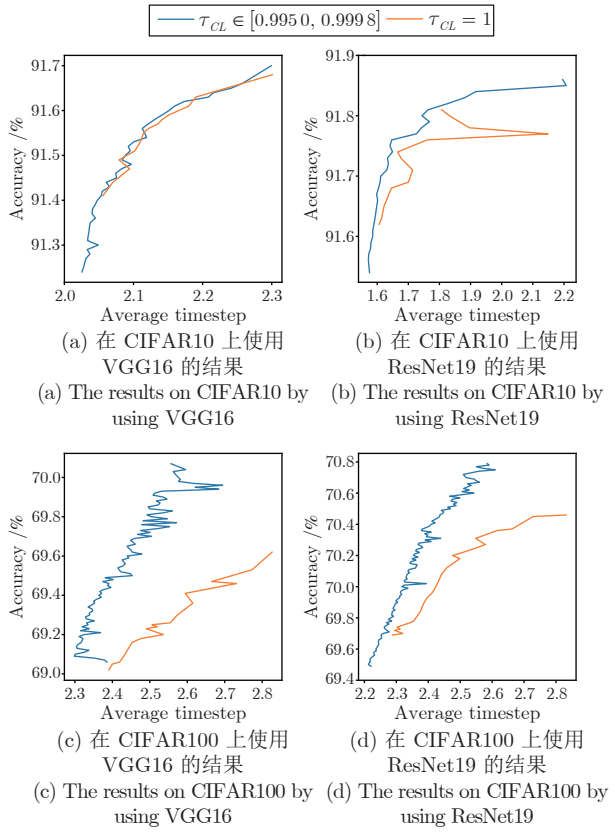
当 τ_{CL} 等于 1 时, ATSNN 的自适应时间步选择算法退化为单一阈值比较, 这在 CNN 的早期退出经常使用. 对比本文提出的动态时间步选择方案 ($\tau_{CL} \neq 1$) 以及 $\tau_{CL} = 1$ 时的退化方案, 结果如图 7 所示. 实验结果表明, 本文动态时间步选择方案比

退化的方案具有更好的性能. 虽然本文方案具有两个超参数, 但从前文的分析可以得知, 使用默认的超参数也可以获得较好的性能, 并且增大超参数能够提升准确率.

表 4 默认超参数的性能

Table 4 Performance with default hyperparameters

数据集	网络结构	时间步	准确率 (%)
CIFAR10	ResNet19	1.897 (4)	91.77
CIFAR10	VGG16	2.183 (4)	91.57
CIFAR100	ResNet19	2.501 (4)	70.46
CIFAR100	VGG16	2.642 (4)	69.58



(a) 在 CIFAR10 上使用 VGG16 的结果

(a) The results on CIFAR10 by using VGG16

(b) 在 CIFAR10 上使用 ResNet19 的结果

(b) The results on CIFAR10 by using ResNet19

(c) 在 CIFAR100 上使用 VGG16 的结果

(c) The results on CIFAR100 by using VGG16

(d) 在 CIFAR100 上使用 ResNet19 的结果

(d) The results on CIFAR100 by using ResNet19

图 7 $\tau_{CL} \neq 1$ 和 $\tau_{CL} = 1$ 时的性能对比

Fig. 7 Performance comparison between $\tau_{CL} \neq 1$ and $\tau_{CL} = 1$

3.6 消融实验分析

为验证本文提出的各个模块对低时间步推理的合理性和有效性, 针对网络构建过程使用到的三个模块进行消融实验. ATSNN 使用的网络模型相较于 SNN 模型做了三处改动: 替换输出层神经元、改变输出表征以及更换损失函数. 本节在基准 SNN 的基础上逐步替换模块来进行消融实验, 实验使用 ResNet19 结构并在 CIFAR10 和 CIFAR100 两个数据集上进行, 训练的时间步均为 4. 如表 5 所示, 第 2 ~ 4 列分别表示是否替换相应模块, 如第 3 行表示仅将输出层替换为本文的膜电位不重置 LIF 神经元; 第 5 ~ 8 列分别表示推理时间步为 1 ~ 4 的网络准确率.

通过表 5 可以看到, 当只替换输出层时, 推理时只使用一个时间步的网络性能得到了提升; 如果只更换损失函数, 推理时只使用一个时间步的网络准确率得到了大幅度提升, 基本上与基准 SNN 使用两个时间步的推理准确率相似; 当替换输出层和损失函数时, 网络在低时间步的推理性能进一步得到提升, 说明两个模块的结合有助于提升网络的性能; 最后一行为三个模块全部替换的结果, 此时低时间步推理性能和网络最终的准确率 (时间步为 4 的推理性能) 达到最高. 不过在 CIFAR10 数据集上, 尽管替换部分模块导致网络最终准确率下降, 但低时间步推理性能得到了提升, 这对后续的动态时间步选择十分重要. 此外在替换全部模块的情况下, 网络最终性能能够达到基准 SNN 的性能, 甚至超越. 因此本文提出的模块在大幅提升低时间步推理准确率的同时, 能够保持甚至提高网络最终的推理准确率.

3.7 低延迟特性对比

本文也和目前具有低延迟的 SNN 算法进行了比较, 结果如表 6 所示. 表 6 中时间步一列如 2.51 (4) 表示训练时间步为 4, 推理时选择的平均时间步为 2.51. 网络结构一列 VGG16 的参数量为 138 M、ResNet19 的参数量为 11.2 M、SEW-ResNet34 的参数量为 21.5 M.

通过表 6 的结果可以看出, ATSNN 获得了最低的推理延迟. 在 CIFAR10 数据集上, 能以 2.71 个时间步获得 92.38% 的准确率; 在 CIFAR100 数据集上获得了最高分类准确率; 在 CIFAR10-DVS 数据集上, 本文的算法仍能以 5.57 个时间步获得较高准确率; 在规模更大的 ImageNet-100、TinyImageNet 和 N-Caltech-101 数据集上, ATSNN 实现了最低的推理延迟和最高的准确率. 通过以上分析可以得到, 本文的算法不论是与固定低时间步的算法还是与动态选择时间步的算法相比, 均有较低的时间步以及较高的准确率, 具有较强的低延迟推理能力.

3.8 讨论

通过实验可以看出, 本文算法具有推理延迟低、复杂度低的特点, 在所有的实验中均取得了较低

表 5　ATSNN 消融实验

Table 5　Ablation experiment of ATSNN

数据集	输出层神经元	损失函数	输出表征	T1	T2	T3	T4
CIFAR10				34.43	89.39	90.82	91.86
	√			47.26	89.85	90.52	91.38
		√		85.68	88.78	90.39	91.28
	√	√		86.93	89.96	91.63	91.79
	√	√	√	86.93	90.14	91.39	91.86
CIFAR100				27.13	56.68	62.53	67.22
	√			30.38	57.19	63.37	67.70
		√		52.69	63.80	67.29	69.61
	√	√		53.47	64.35	67.53	69.90
	√	√	√	53.47	65.23	68.17	70.25

表 6　ATSNN 与低延迟算法的对比

Table 6　Comparison between ATSNN and low-latency algorithms

数据集	算法	算法类型	网络结构	时间步	准确率 (%)
CIFAR10	Conversion[11]	ANN 转 SNN	VGG16	8	90.96
	STDB[33]	转换+训练	VGG16	5	91.41
	EfficientLIF-Net[10]	直接训练	VGG16	5	90.30
	DTSNN[19]	直接训练	VGG16	**2.53 (4)**	91.13
	本文	直接训练	VGG16	2.56 (10)	**92.09**
	STBP-tdBN[14]	直接训练	ResNet19	6	**93.16**
	DTSNN[19]	直接训练	ResNet19	**2.51 (4)**	91.51
	本文	直接训练	ResNet19	2.71 (10)	92.38
CIFAR100	STDB[33]	转换+训练	VGG16	5	66.46
	Diet-SNN[15]	直接训练	VGG16	5	69.67
	Real Spike[34]	直接训练	VGG16	5	70.62
	RecDis-SNN[35]	直接训练	VGG16	5	69.88
	本文	直接训练	VGG16	**3.86 (10)**	**71.37**
	DTSNN[19]	直接训练	ResNet19	3.54 (4)	67.58
	本文	直接训练	ResNet19	**2.53 (10)**	**70.64**
CIFAR10-DVS	STBP-tdBN[14]	直接训练	ResNet19	6	67.80
	DTSNN[19]	直接训练	ResNet19	6.95 (10)	**70.50**
	本文	直接训练	ResNet19	**5.57 (10)**	69.50
ImageNet-100	EfficientLIF-Net[10]	直接训练	ResNet19	5	79.44
	LocalZO[36]	直接训练	SEW-ResNet34	4	81.56
	本文	直接训练	ResNet19	**1.76 (5)**	**81.96**
TinyImageNet	EfficientLIF-Net[10]	直接训练	ResNet19	5	55.44
	DTSNN[19]	直接训练	ResNet19	3.71 (5)	57.18
	本文	直接训练	ResNet19	**2.47 (5)**	**57.61**
N-Caltech-101	MC-SNN[37]	直接训练	VGG16	20	81.24
	DTSNN[19]	直接训练	VGG16	3.21 (10)	82.26
	本文	直接训练	VGG16	**2.19 (10)**	**82.63**
	NDA[30]	直接训练	ResNet19	10	78.60
	本文	直接训练	ResNet19	**2.56 (10)**	**80.56**

的推理延迟. 但在处理部分数据集时, 如表 6 的 CIFAR10, 本文算法的准确率不及 STBP-tdBN 方法; 在 CIFAR10-DVS 数据集上, 本文算法准确率略低于 DTSNN.

出现这种情况的原因主要是训练出的网络精度较低. 通过在训练时增大时间步, 发现本文算法依然比上述算法具有更低的推理延迟, 准确率相对于低时间步训练的网络得到了提升. 因此可以通过增加训练时间步解决这个问题, 同时也考虑对网络改进以提升性能.

此外, 在第 3.3 节中, 直接使用本文提出的损失函数 L_{imp} 会出现准确率下降的现象 (图 4 时间步为 4). 这是因为这项对比实验使用到的 SNN 模型, 其输出层为无时间信息的全连接层. 由于 L_{imp} 会使长时间步的输出重要性减弱, 如果网络输出携带较少的时间信息, 权重无法得到有效更新, 从而导致准确率下降. 但是在本文的算法中, 输出层能够携带时间信息, 因此使用 L_{imp} 的 ATSNN 能够相较其他算法提升网络的准确率.

从时间步选择算法的计算过程来看, 根据式 (5) ~ (7), DTSNN 每个时间步分别需要进行 C 次除法、指数、对数运算; 从式 (17)、式 (18) 可以看出, 本文算法每个时间步需要 C 次指数运算. 由于除法和对数运算的计算过程较为复杂, 计算延迟高、能耗大, 难以在硬件上直接部署. 尽管 DTSNN 的加速器设计了专用的时间步模块实现该算法, 但所需的存储器资源是本文加速器的两倍. 综上, 本文方案相比 DTSNN 对硬件更加友好, 同时具有较高的推理性能、较低的推理延迟以及较低的能耗, 适合在低功耗智能终端设备上部署.

4 结论

本文介绍了一种自适应时间步脉冲神经网络, 能够在推理过程中自动选择适当的时间步. 本文改进了 SNN 输出层与表征方式, 构建了具有时间重要性的损失函数 L_{imp}, 提出了可自适应选择时间步的方法, 并分析了所提自适应时间步选择算法的有效性. 此外本文也设计了一款适用于 ATSNN 的低功耗脉冲神经网络加速器. 实验结果表明, ATSNN 的延迟减少 36.7% ~ 58.7%, 计算复杂度减少 33.0% ~ 57.0%. 与目前最先进的动态时间步脉冲神经网络 DTSNN 相比, ATSNN 能有效缩短时间步, 提升网络准确率, 具有更高的能效. 同时, 本文加速器具有与 GPU 相当的推理速度, 能耗仅为 GPU RTX 3090Ti 的 4.43% ~ 7.88%. 综上, ATSNN 能在推理过程中自适应地选择时间步, 降低复杂度、推理延迟和能耗, 有效解决了多时间步带来的问题, 非常适合在低功耗智能终端设备上部署.

References

1 Dampfhoffer M, Mesquida T, Valentian A, Anghel L. Are SNNs really more energy-efficient than ANNs? An in-depth hardware-aware study. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2023, **7**(3): 731−741

2 Merolla P A, Arthur J V, Alvarez I R, Cassidy A S, Sawada J, Akopyan F, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 2014, **345**(6197): 668−673

3 Davies M, Srinivasa N, Lin T H, Chinya G, Cao Y Q, Choday S H, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 2018, **38**(1): 82−99

4 Pei J, Deng L, Song S, Zhao M G, Zhang Y H, Wu S, et al. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature*, 2019, **572**(7767): 106−111

5 Bouvier M, Valentian A, Mesquida T, Rummens F, Reyboz M, Vianello E, et al. Spiking neural networks hardware implementations and challenges: A survey. *ACM Journal on Emerging Technologies in Computing Systems*, 2019, **15**(2): Article No. 22

6 Wu Y J, Deng L, Li G Q, Zhu J, Shi L P. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, DOI: 10.3389/fnins.2018.00331

7 Izhikevich E M. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 2004, **15**(5): 1063−1070

8 Taherkhani A, Belatreche A, Li Y H, Cosma G, Maguire L P, McGinnity T M. A review of learning in biologically plausible spiking neural networks. *Neural Networks*, 2020, **122**: 253−272

9 Liang Z Z, Schwartz D, Ditzler G, Koyluoglu O O. The impact of encoding-decoding schemes and weight normalization in spiking neural networks. *Neural Networks*, 2018, **108**: 365−378

10 Kim Y, Li Y H, Moitra A, Yin R K, Panda P. Sharing leaky-integrate-and-fire neurons for memory-efficient spiking neural networks. *Frontiers in Neuroscience*, DOI: 10.3389/fnins.2023.1230002

11 Bu T, Ding J H, Yu Z F, Huang T J. Optimized potential initialization for low-latency spiking neural networks. arXiv preprint arXiv: 2202.01440, 2022.

12 Jiang C M, Zhang Y L. KLIF: An optimized spiking neuron unit for tuning surrogate gradient slope and membrane potential. arXiv preprint arXiv: 2302.09238, 2023.

13 Fang W, Yu Z F, Chen Y Q, Huang T J, Masquelier T, Tian Y H. Deep residual learning in spiking neural networks. arXiv preprint arXiv: 2102.04159, 2021.

14 Zheng H L, Wu Y J, Deng L, Hu Y F, Li G Q. Going deeper with directly-trained larger spiking neural networks. arXiv preprint arXiv: 2011.05280, 2020.

15 Rathi N, Roy K. 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*, 2023, **34**(6): 3174−3182

16 Guo Y F, Chen Y P, Zhang L W, Liu X D, Wang Y L, Huang X H, et al. IM-loss: Information maximization loss for spiking neural networks. In: Proceedings of the 36th Conference on Neural Information Processing Systems. New Orleans, USA: Curran Associates Inc., 2022. 156−166

17 Teerapittayanon S, McDanel B, Kung H T. BranchyNet: Fast inference via early exiting from deep neural networks. In: Proceedings of the 23rd International Conference on Pattern Recognition (ICPR). Cancun, Mexico: IEEE, 2016. 2464−2469

18 Kim Y, Panda P. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Fronti-*

ers in Neuroscience, DOI: 10.3389/fnins.2021.773954

19 Li Y H, Moitra A, Geller T, Panda P. Input-aware dynamic timestep spiking neural networks for efficient in-memory computing. In: Proceedings of the 60th ACM/IEEE Design Automation Conference (DAC). San Francisco, USA: IEEE, 2023. 1–6

20 Deng L, Wu Y J, Hu Y F, Liang L, Li G Q, Hu X, et al. Comprehensive SNN compression using ADMM optimization and activity regularization. *IEEE Transactions on Neural Networks and Learning Systems*, 2023, **34**(6): 2791–2805

21 Stöckl C, Maass W. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *Nature Machine Intelligence*, 2021, **3**(3): 230–238

22 Chen Y Q, Yu Z F, Fang W, Huang T J, Tian Y H. Pruning of deep spiking neural networks through gradient rewiring. arXiv preprint arXiv: 2105.04916, 2021.

23 Eshraghian J K, Lammie C, Azghadi M R, Lu W D. Navigating local minima in quantized spiking neural networks. In: Proceedings of the IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS). Incheon, South Korea: IEEE, 2022. 352–355

24 Wang Y X, Xu Y, Yan R, Tang H J. Deep spiking neural networks with binary weights for object recognition. *IEEE Transactions on Cognitive and Developmental Systems*, 2021, **13**(3): 514–523

25 Krizhevsky A. Learning Multiple Layers of Features From Tiny Images [Master thesis], University of Toronto, Canada, 2009.

26 Le Y, Yang X. Tiny imagenet visual recognition challenge [Online], available: http://cs231n.stanford.edu/tiny-imagenet-200.zip, June 11, 2023

27 Deng J, Dong W, Socher R, Li L J, Li K, LI F F. ImageNet: A large-scale hierarchical image database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Miami, USA: IEEE, 2009. 248–255

28 Li H M, Liu H C, Ji X Y, Li G Q, Shi L P. CIFAR10-DVS: An event-stream dataset for object classification. *Frontiers in Neuroscience*, DOI: 10.3389/fnins.2017.00309

29 Orchard G, Jayawant A, Cohen G K, Thakor N. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, DOI: 10.3389/fnins.2015.00437

30 Li Y H, Kim Y, Park H, Geller T, Panda P. Neuromorphic data augmentation for training spiking neural networks. In: Proceedings of the 17th European Conference on Computer Vision. Tel Aviv, Israel: Springer, 2022. 631–649

31 Fang W, Chen Y Q, Ding J H, Yu Z F, Masquelier T, Chen D, et al. SpikingJelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 2023, **9**(40): Article No. eadi1480

32 Yin R K, Moitra A, Bhattacharjee A, Kim Y, Panda P. SATA: Sparsity-aware training accelerator for spiking neural networks. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, 2023, **42**(6): 1926–1938

33 Datta G, Kundu S, Beerel P A. Training energy-efficient deep spiking neural networks with single-spike hybrid input encoding. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN). Shenzhen, China: IEEE, 2021. 1–8

34 Guo Y F, Zhang L W, Chen Y P, Tong X Y, Liu X D, Wang Y L, et al. Real spike: Learning real-valued spikes for spiking neural networks. In: Proceedings of the 17th European Conference on Computer Vision. Tel Aviv, Israel: Springer, 2022. 52–68

35 Guo Y F, Tong X Y, Chen Y P, Zhang L W, Liu X D, Ma Z, et al. RecDis-SNNS: Rectifying membrane potential distribution for directly training spiking neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). New Orleans, USA: IEEE, 2022. 326–335

36 Mukhoty B, Bojkovic V, Vazelhes W, Zhao X H, Masi G, Xiong H, et al. Direct training of SNN using local zeroth order method. In: Proceedings of the Thirty-seventh Conference on Neural Information Processing Systems. New Orleans, USA: Curran Associates Inc., 2023. 18994–19014

37 Li X J, Tang J X, Lai J H. Learning high-performance spiking neural networks with multi-compartment spiking neurons. In: Proceedings of the 12th International Conference on Image and Graphics. Nanjing, China: Springer, 2023. 91–102

李千鹏 中国科学院自动化研究所硕士研究生. 主要研究方向为类脑智能, 类脑处理器.
E-mail: liqianpeng2021@ia.ac.cn
(**LI Qian-Peng** Master student at the Institute of Automation, Chinese Academy of Sciences. His research interest covers brain-inspired intelligence and brain-inspired processor.)

贾顺程 中国科学院自动化研究所博士研究生. 主要研究方向为类脑脉冲神经网络模型与学习算法.
E-mail: jiashuncheng2020@ia.ac.cn
(**JIA Shun-Cheng** Ph.D. candidate at the Institute of Automation, Chinese Academy of Sciences. His research interest covers brain-inspired spiking neural network model and learning algorithm.)

张铁林 中国科学院脑科学与智能技术卓越创新中心研究员. 主要研究方向为类脑计算.
E-mail: zhangtielin@ion.ac.cn
(**ZHANG Tie-Lin** Researcher at the Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences. His main research interest is brain-inspired computing.)

陈 亮 中国科学院自动化研究所副研究员. 主要研究方向为计算机架构与集成电路设计, 类脑计算. 本文通信作者. E-mail: liang.chen@ia.ac.cn
(**CHEN Liang** Associate researcher at the Institute of Automation, Chinese Academy of Sciences. His research interest covers computer architecture and integrated circuit design and brain-inspired computing. Corresponding author of this paper.)