类脑智能大作业报告

李嘉睿	陶也	杜亦开
522030910119	522030910126	522030910121

1 摘要

近年来，随着对能效和生物启发型计算的关注，脉冲神经网络（Spiking Neural Networks, SNNs）逐渐成为一种极具潜力的研究方向。SNN通过事件驱动的计算模式，能够显著降低能耗，同时在时间序列建模和稀疏性处理上展现了独特优势，尤其适合资源受限环境中的智能应用。在这一背景下，我们聚焦于SNN在不同任务上的应用与探索。

本次研究的主要贡献包括以下三个方面：首先，我们成功实现了基于SNN的卷积网络在经典MNIST数据集上的分类任务，验证了SNN在简单分类任务中的基本性能；其次，我们进一步将Transformer架构引入SNN框架，探索其在FashionMNIST及相关变种数据集上的分类能力，展示了SNN在更复杂任务中的潜力；最后，我们将研究重点放在高级任务——利用SNN进行深度估计，并结合量化的方法提出了一种高效的实现方案，不仅在性能上达到了较好的结果，还显著节约了计算资源和能耗。

2 初级部分

2.1 SNN基础原理

SNN的主要特点是事件驱动的计算模式，相比于ANN能较大程度上节约能耗。SNN的核心机制基于Leaky Integrate-and-Fire (LIF) 神经元模型。每个神经元的膜电位随时间变化，当膜电位积累到一定阈值时，神经元发放脉冲，并将膜电位重置。其数学模型如下：

$$\tau \frac{dV(t)}{dt} = -V(t) + I(t)$$

其中，$V(t)$是神经元的膜电位，τ是时间常数，$I(t)$是输入电流。当$V(t)$达到设定阈值时，神经元发放脉冲，并将膜电位重置为零。

2.2 全连接脉冲神经网络

全连接脉冲神经网络（Fully Connected Spiking Neural Network, FC-SNN）是SNN中最基本的结构形式，其中每一层的神经元都与前一层的所有神经元连接。输入数据通过脉冲信号进入网络，神经元通过发放脉冲进行信息传递，最终通过输出层进行分类。

FC-SNN的训练过程较为复杂，传统的反向传播算法不适用于SNN，因为脉冲信号的离散性。为此，学习课件中的方法采用了替代梯度法（Surrogate Gradient），通过近似的连续激活函数来计算梯度，进而优化网络参数。这里不过多赘述STDP学习策略等内容。

2.2.1 简单的单层SNN

```
nn.Sequential(
    layer.Flatten(),
    layer.Linear(28 * 28, 10, bias=False
        ),
    neuron.LIFNode(tau=tau,
        surrogate_function=surrogate.
        ATan())
    )
```

Listing 1: 单层SNN网络实现

2.3 卷积脉冲神经网络

2.3.1 模型结构分析

CSNN的网络结构主要包括卷积层、脉冲神经元层（IF节点）、池化层和全连接层。代码实现这里略去。 CSNN的关键特性体现在以下几个方面：

- 卷积层与池化层： 卷积层用于提取输入图像的空间特征，池化层通过下采样减少计算量并保留主要特征。
- 脉冲神经元层： 使用IF（Integrate-and-Fire）节点对卷积输出进行时序处理，通过代理梯度方法（如$ATan$函数）解决脉冲非连续性问题。
- 多步时序处理： 在前向传播中，利用时间步长T对输入数据进行复制扩展，模拟脉冲神经网络的动态行为。

2.3.2 公式化描述

CSNN的整体计算过程可以分为两部分：

1. 卷积层提取空间特征：

$$F = \text{Conv}(X), \tag{1}$$

其中 F 表示卷积操作后的特征张量。

2. 脉冲神经元层处理时序数据：

$$Y = SN(F), \tag{2}$$

通过多个时间步的发放率均值，输出最终结果：

$$\text{Output} = \frac{1}{T}\sum_{t=1}^{T} Y_t. \tag{3}$$

2.4 实验

在初级部分的实验中，我们实现了基于SNN的简单分类网络，并在MNIST数据集上，采用adam优化器与交叉熵损失函数进行训练。我们通过调整网络的层数,脉冲发放的时间步长，观察了分类准确率的变化，结果如图 1所示。我们发现多层SNN的表现在时间步增加时反而下降，推测原因为规模过小。同时随着csnn的卷积层数的增加，模型精度有提升，但是训练时间变长。由于重点在高级部分，这里略去调参实验。

Model	τ	Accuracy
SNN-Single Layer	2	86.9%
SNN-Multi Layer	2	94.17%
SNN-Multi Layer	4	92.95%
CSNN	4	99.48%
CSNN-modified	4	99.52%

表格 1: Model performance comparison

这里的csnn-modified是增加csnn的层数，虽然有一些性能的提升，但是训练时间也相应增加。

2.5 可视化

实现了简单的分类后，我们选取第一张图片数字"7"进行可视化。在T为4的时间步长下，firing rate结果如图 1，可以看到除了正确类别对应的神经元外，其它神经元均未发放任何脉冲。

同时我们还可视化了中间卷积层的信息如图 2每个小图代表卷积层的一个通道（feature map）。小图中的像素点表示神经元的空间激活状态，黑色背景表示未释放脉冲，白色表示对应位置神经元在当前时间步释放了脉冲。整个图像由多个小图组成，每个小图对应卷积层输出的一个通道（共 C个通道）。

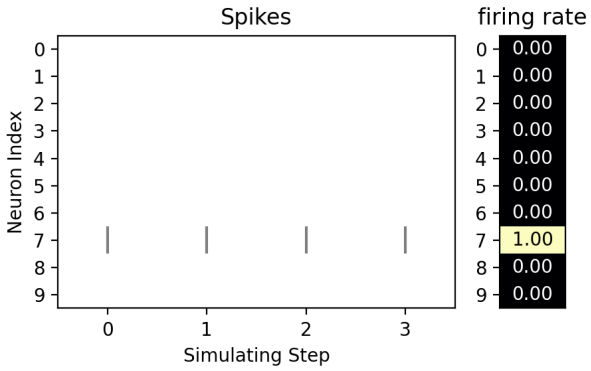

Figure 1: 输入为7的图片时，时间步长为4时的firing rate

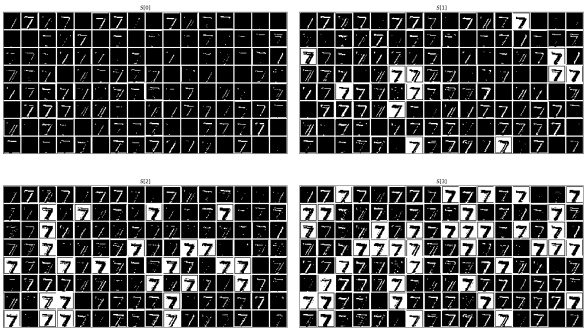

Figure 2: 输入为7的图片时，每个时间步对应的卷积可视化

3 中级部分

中级部分我们选取了将snn和transformer进行结合。在将 Transformer 应用于 SNN 的过程中，主要面临以下两大难点：

- **Attention** 中 **Q**、**K**、**V** 的计算与稀疏计算不匹配

  在传统的 Transformer 中，Attention 机制需要通过计算查询向量 Q 和键向量 K 的点积，生成一个注意力矩阵。随后，使用包含指数运算和归一化操作的 Softmax 函数对矩阵进行归一化，得到注意力分布，并利用该分布对值向量 V 进行加权。然而，SNN 的计算特点决定了其擅长稀疏事件驱动的操作，而点积计算与指数运算均属于密集计算，不符合 SNN 的特性。此外，SNN 通常避免乘法操作，依赖加法来实现计算，这进一步增加了实现传统 Attention 的难度。

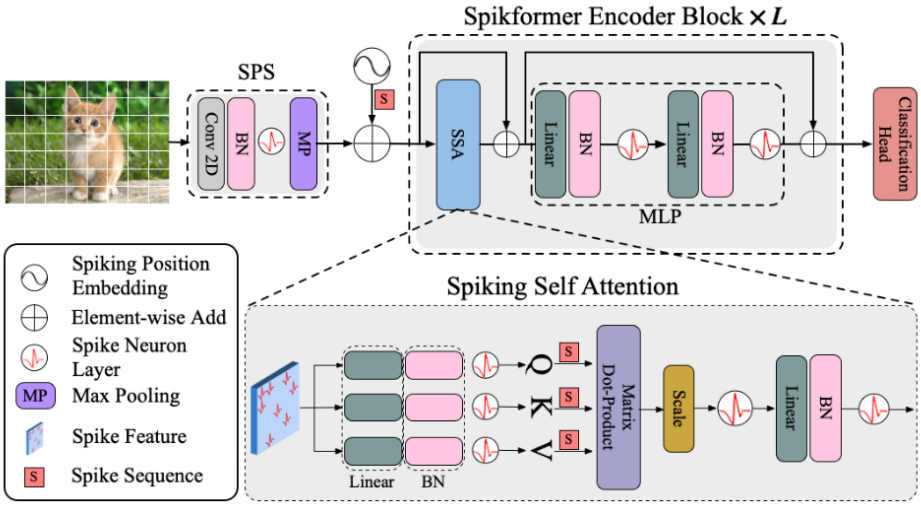

Figure 3: spikeformer 架构 ((Zhou et al., 2023)

- **Softmax** 与 **SNN** 的逻辑不兼容

  Softmax 函数的计算过程同样包含指数运算与除法，这些操作与 SNN 中的稀疏、事件驱动特性不相容。

3.1 实现方法和架构

为了解决上述问题，我们主要参考了spike-former的方法 (Zhou et al., 2023)，主要架构如图 3所示。主体结构为encoder-only，SPS模块负责划分特征块，提取脉冲特征；Spikeformer编码器包含SSA注意力机制+MLP；最后多分类头完成分类任务。

在稀疏自注意力（SSA）机制中，查询（Q）、键（K）和值（V）均为稀疏脉冲序列，仅由0和1构成。这种构成方式使得注意力计算可以通过逻辑与（AND）和加法完成。具体公式如下：

$$\begin{aligned} &SSA(Q,K,V) \\ &= SN(BN(Linear(Q \cdot K^T \cdot V \cdot s))) \end{aligned} \tag{4}$$

其中，SN表示稀疏归一化，BN表示批归一化，$Linear$表示线性变换，s是缩放因子。

在SSA中，脉冲形式的Q和K生成的注意力矩阵天然非负，因此无需Softmax操作。Softmax通常用于将注意力分数转换为概率分布，但在SSA中，由于注意力矩阵已经是非负的，直接使用这些分数即可。

3.2 实验

实验环节我们在fashionmnist数据集上进行实验。时间步长设置为4，同时对比了CSNN以及ANN和SNN版transformer的性能。虽然SNN的主要优势不在正确率上有较明显的提升，但是仍然取得了较好的效果。

模型	正确率 (%)
CSNN	93.0
ANN + Transformer	94.0
SNN + Transformer	93.5

表格 2: 不同模型的准确率

4 高级部分

4.1 简介

在这一部分，我们使用 SNN 架构去做深度估计的任务。我们主要是将 monodepth2 (Godard et al., 2019)的方法中的 Resnet 解码器重新实现为了 SNN 架构的 SnnResnet，为适应 SNN 架构也对 Resnet 结构做了一些微调(Hu et al., 2024)。接着，由于 SNN 时间步的引入，训练时长变得不可忍受，于是我们采用多比特量化的方案，在保留 SNN 无乘法、低能耗特征的同时，降低了训练时长，取得了不错的效果(Luo et al., 2025)。

4.2 方法

传统的 SnnResnet 网络，只是把 Resnet 的结构中的激活函数由 ReLU 函数改为 具有尖峰神经网络特性的 LIF 节点，加载的在 ImageNet 数据集上的训练参数也是原 Resnet 中的参数，而不是训练后的更适配 Snn 结构的参数，这样的修改无疑是照猫画虎的。

实验表明，这样修改后得到的网络在增加深度时会产生严重的退化现象(Hu et al., 2024)。

在经过改进后，我们先采用单步进模式，也就是不采用 TimeWindow 的模式进行训练，但结果不尽如人意，Loss 曲线和指标在训练

架构	T	D	abs_rel	sq_rel	rmse	rmse_log	a1	a2	a3	训练时长	推理能耗
Resnet w/o pretrained	无	无	0.134	1.044	5.242	0.210	0.842	0.947	0.977	9h	259.5mJ
SnnResnet pretrained	20	无	0.134	1.038	5.187	0.211	0.837	**0.948**	0.977	73h	45.1mJ
SnnResnet w/o pretrained	2	4	0.142	1.051	5.348	0.212	0.839	**0.948**	0.977	13h	51.3mJ
SnnResnet pretrained	2	4	0.135	1.040	**5.183**	0.210	0.838	**0.948**	0.977	13h	50.5mJ
SnnResnet pretrained	2	8	**0.133**	**1.036**	5.187	**0.209**	**0.844**	**0.948**	**0.978**	13h	51.2mJ

表格 3: 模型性能比较，T表示时间步长，D表示量化值，后面四个指标分别代表平均相对误差，平方相对误差，均方根误差，对数空间的均方根误差，a_i分别代表预测深度值与真实值的比例在 $[1.25^{-i}, 1.25^{i}]$ 范围内的像素百分比，误差越小越好而百分比越大越好。

过程中一直表现糟糕。实验结果表明，单步进模式下的 Snn 网络无法完成该任务。究其原因，是 Snn 层输出只有 0 和 1 的特性，在不使用多步进的情况下缺少多次重复，丢失了太多信息，在 TimeWindow 较小的时候也是如此。经过查阅资料，Resnet50 可能需要 TimeWindow=512 才能不损失性能。于是我们又采用了 TimeWindow=20，这次训练结果确实没什么损失。但另一个问题又随之出现，取 TimeWindow=20 相当于把之前的网络重复了 20 遍放在显存中，这导致了我们的 batchSize 必须调为1，否则就会导致显存爆炸，这进一步使得训练时长对我们来说变得不可接受（3～4天）。于是在叶老师的建议下，我们去阅读了 SpikeYolo 这篇论文，这里面提出了一些减少显存使用的方法。

图4是具体方法的一个例子，I-LIF 层即是改进后的激活函数架构。由于正常的 Snn 的激活层只能输出 0 或 1，这大大减少了能耗（因为不用计算乘法），但是也导致信息丢失过多。由图4的上半图可知，当输出为 2.1 时，它直接被 Snn 量化为 1，造成了很大的量化损失。于是就可以想到，我们可以让激活函数不止输出 0 和 1，而是 0 到 2，0 到 4 等等，比如这里 D=2（表示取值量化到0 1 2），当输出为 2.1 时，它会被量化为 2，这样就减少了很多量化损失。具体来说，是把正常 Snn 的激活层公式变成了

$$
\begin{aligned}
U[t] &= H[t-1] + X[t] \\
S[t] &= \text{Clip}(\text{round}(U[t]), 0, D) \\
\text{上面一行代替了} S[t] &= \theta(U[t] - V_{th}) \\
H[t] &= \beta(U[t] - S[t])
\end{aligned} \tag{5}
$$

随着量化位数D的增加，SNN网络的表征能力得到显著提升，这使得我们可以适当减小时间步长T的取值。值得注意的是，量化位数D的增大并不会增加显存占用，因此该方法能够有效降低模型训练过程中的显存需求。在推理阶段，我们将较大的量化值序列重新转换为0-1序列（如图4下半部分所示），从而保持了SNN网络在推理时不进行乘法运算的特性，确保了其推理能耗仍显著低于传统ANN网络。

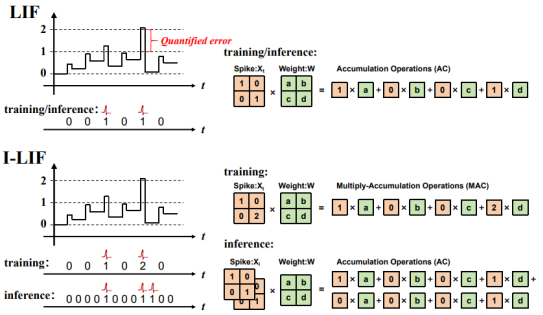

Figure 4: spikingyolo 减少显存占用的方法

4.3 实验

推理时的能耗计算公式与(Luo et al., 2025)中计算方式一致，具体公式为

$$E_{SNN} = \sum fr \times flops \times K_{flops}$$

在计算时，先计算出每一层的发放率 fr（统计输出中 1 占的百分比），然后手动与每一层的参数量和 K_{flops}（对 Snn 是0.9）乘起来即为能耗。

本文在 KITTI 数据集上对所提出的方法进行了实验验证，并将实验结果汇总于表3。实验结果表明，相较于未使用量化方法时73小时的训练时长及相对较差的模型性能，采用量化方法不仅显著缩短了训练时间，同时也提升了模型的整体表现。此外，通过在 ImageNet 数据集上对 SnnResnet 进行预训练，深度估计模型的性能得到了进一步提升。通过对比量化位数 D 分别取 4 和 8 时的实验结果可以发现，在特定范围内增加量化位数能够有效增强模型的表征能力，从而获得更优的性能表现。

5 分工

杜亦开主要完成了初级部分的任务和初级部分报告的撰写，陶也主要完成了中级部分的任务以及摘要、初级部分、中级部分报告的撰写，

李嘉睿主要完成了高级部分和高级部分报告的撰写。

References

Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. 2019. Digging into self-supervised monocular depth prediction.

Yifan Hu, Lei Deng, Yujie Wu, Man Yao, and Guoqi Li. 2024. Advancing spiking neural networks toward deep residual learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Xinhao Luo, Man Yao, Yuhong Chou, Bo Xu, and Guoqi Li. 2025. Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection. In *European Conference on Computer Vision*, pages 253–272. Springer.

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