脉冲神经网络的神经架构搜索

摘要

由于其固有的高稀疏性激活，脉冲神经网络 (SNN) 作为传统人工神经网络 (ANN) 的潜在节能替代品而受到了广泛关注。然而，大多数现有的 SNN 方法使用类似 ANN 的架构（例如 VGGNet 或 ResNet），这可能为 SNN 中二进制信息的时间序列处理提供次优性能。为了解决这个问题，在本文中，我们引入了一种新颖的神经架构搜索 (NAS) 方法来寻找更好的 SNN 架构。受到最近的 NAS 方法的启发，这些方法在初始化时从激活模式中找到最佳架构，我们选择了无需训练即可表示跨不同数据样本的不同脉冲激活模式的架构。此外，为了进一步利用脉冲之间的时间信息，我们搜索层之间的前馈连接以及后向连接（即时间反馈连接）。有趣的是，我们的搜索算法发现 SNASNet 通过后向连接实现了更高的性能，这证明了设计 SNN 架构以适当使用时间信息的重要性。我们对三个图像识别基准进行了广泛的实验，结果表明 SNASNet 以显着较低的时间步长（5 个时间步长）实现了最先进的性能。

关键词：脉冲神经网络；神经架构搜索；神经形态计算

1 引言

脉冲神经网络（SNN）作为低功耗智能的一种有前途的范例而受到越来越多的关注。受生物神经元功能的启发，SNN 通过多个时间步长的二进制脉冲处理视觉信息。到目前为止，SNN 的大部分工作都集中在图像分类问题，以开发人工神经网络 (ANN) 的节能替代方案。为此，最近的 SNN 工作利用了由人类专家设计的 ANN 架构（例如 VGG-Net 或 ResNet ）。虽然 SNN 在能源效率方面显示出令人印象深刻的优势，但它们在准确性方面仍然落后于 ANN。

在本文中，我们认为 ANN 和 SNN 之间固有的结构/功能差异会导致不可忽视的架构差距，从而导致当我们在 SNN 上简单地部署 ANN 架构时得到次优解决方案。具体来说，与具有 ReLU 神经元的 ANN 不同，SNN 由存储和传输时间信息的泄漏整流放电 (LIF) 神经元组成。然而，手动搜索 SNN 友好的架构非常费力。因此，我们使用神经架构搜索（NAS），它可以自动发现最佳的 SNN 架构。尽管 NAS 已成为各种 ANN 任务中的流行技术，但用于设计 SNN 的 NAS 尚未得到研究。

2 相关工作

2.1 脉冲神经网络

脉冲神经网络（SNNs）作为与标准人工神经网络（ANNs）相比的能效替代方案，已经引起了极大的关注。SNNs 通过权重连接和漏电整流-放电（LIF）神经元 [6] 处理时间信息，它作为 SNNs 中的非线性激活。LIF 神经元具有称为膜电位的内存，可以通过累积传入的脉冲信号存储时间脉冲动态。如果膜电位超过发放阈值，神经元将生成一个后突触脉冲。神经元的整流-放电行为引入了不可微的传递函数。因此，在训练阶段应用标准反向传播是困难的 [11]。为了解决这个问题，提出了各种方法来规避不可微的反向传播问题。其中，替代梯度学习方法变得流行，因为它们在性能和时间步数方面均优于其他训练技术。它们在计算反向梯度时为 LIF 神经元定义了一个替代函数。串联学习利用辅助的 ANN 为 SNN 训练提供稳定的错误反向传播。一系列工作训练 LIF 神经元中的膜衰减或发放阈值，提高了 SNN 的表征能力。此外，批量归一化（BN）[5] 已应用于加速 SNN 训练过程。尽管 SNN 训练技术最近有了发展，但所有以前的方法都利用了 ANN 架构，如 VGG 和 ResNet 系列。这些架构可能为 SNNs 提供次优的解决方案。

2.2 LIF（Leaky Integrate-and-Fire）神经元

LIF 神经元广泛用于构建 SNN [3]。神经元具有存储时间脉冲信息的膜电位。将上述连续微分方程转换为离散版本，如之前的工作 [13] 所示：

$$u_i^t = (1 - \frac{1}{\tau_m})u_i^{t-1} + \frac{1}{\tau_m}\sum_j w_{ij}o_j^t. \tag{1}$$

其中，u_i^t 表示神经元 i 在时间步 t 的膜电位，τ_m 是衰减膜电位的时间常数。另外，w_{ij} 代表神经元 j 和神经元 i 之间的权重连接。神经元 i 积累膜电位，并在膜电位超过阈值时生成脉冲输出 o_i^t 。发射后，膜电位重置为零。

2.3 神经架构搜索

神经架构搜索（NAS）已被提出用于发现高性能的网络。NAS 算法的早期阶段使用强化学习或进化算法。然而，这些方法需要为每个搜索步骤从头开始训练所搜索的架构，这是非常计算密集的。为了解决这个问题，提出了权重共享方法。它们一次性训练包含所有架构候选的超网络。例如，Darts [8] 联合优化网络参数和每个架构候选的重要性。此外，SPOS [4] 使用均匀正向路径抽样训练权重参数，并通过进化策略找到最优架构。与先前的 NAS 算法相比，权重共享方法不需要在每个搜索步骤中从头开始训练架构，从而具有更高的效率。在最近的研究中，NAS 技术的关注重点一直是其效率，这要归功于数据集和架构规模的不断增长。有趣的是，一系列工作提出了在搜索阶段无需训练的 NAS 概念，即网络在搜索阶段无需训练。这可以显著降低搜索最优架构的计算成本。在 NAS 在图像分类领域取得成功后，NAS 已被部署到各种任务，如目标检测、分割、GAN 、Transformer 和人体姿态估计。尽管 NAS 算法在 ANN 领域取得了巨大的进展，但至今尚未开发用于 SNN 的 NAS。

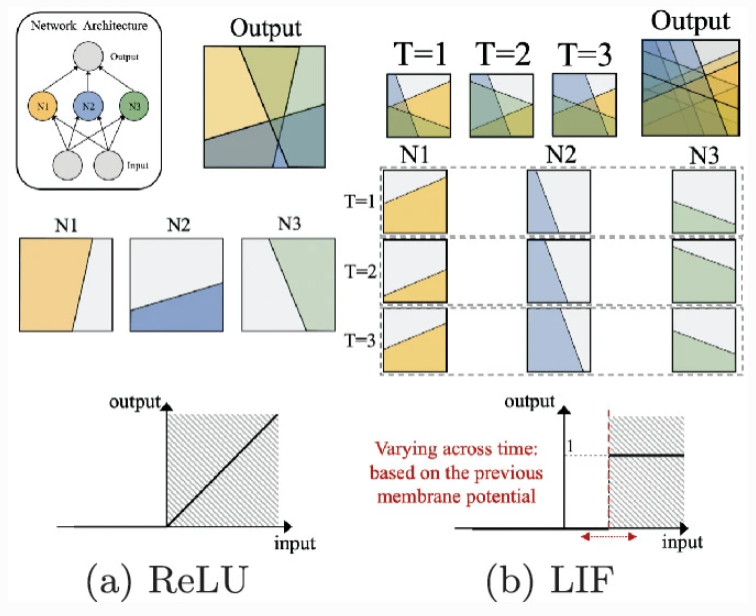

图 1. ReLU 和 LIF 神经元线性区域概念的图示。每个 ReLU（或 LIF 神经元）将二维输入空间划分为活动区域和非活动区域。

3 本文方法

在本节中，我们首先基于神经网络中线性区域的概念介绍 LIF 神经元的时间二进制指示器。之后，我们提出了稀疏感知汉明距离，它解释了 LIF 神经元的稀疏变化。最后，我们为 NAS 算法提供搜索空间，在其中找到前向和后向连接。

3.1 LIF 神经元的线性区域

ANN 领域中无训练的 NAS 方法基于神经网络中线性区域的理论概念。也就是说，每个分段线性函数（例如 ReLU）将输入空间划分为多个线性区域。多个分段线性函数的组合在输入空间上带来了多个线性区域。这种线性区域的模式用于通过比较不同样本之间的模式来测量初始化网络的表示能力。在这里，基于之前的工作，我们引入了分段线性函数中神经元转移（即线性区域的边界）的定义。

定义 1.（Raghu 等人 [12]）对于固定 W，如果其激活函数在 x 和 x + δ 之间切换线性区域，我们称神经元在输入 x、x+δ 之间具有分段线性区域转换。

例如，ReLU 和 Hard Tanh 的神经元转换分别在 0 和 $-1, 1$ 处。图 1(a) 还显示了具有三个 ReLU 神经元的简单示例。输入空间被单个 ReLU 神经元划分为两个区域。通过组合 ReLU 神经元，输入空间被划分为多个区域，其中每个区域代表不同的线性函数。

根据定义 1，LIF 神经元可以被视为分段线性函数。对于每个时间步长，如果膜电位低于激发阈值，LIF 神经元会传输 0，否则会生成 1（即脉冲）。因此，当给定输入产生输出脉冲时，就会发生神经元转换。我们在图 1(b) 中说明了 LIF 神经元的传递函数。与 ReLU 神经元不同，LIF 神经元的输出不仅仅依赖于输入。LIF 神经元的输出基于当前输入以及先前的膜电位。因此，神经元转变点可以随时间变化。例如，假设激发阈值为 1，前一时间步的膜电位为 0.3。在这种情况下，神经元转换发生在输入 = 0.7 时。神经元放电后，膜电位重置为 0，其中神经元转变点变为 1。通过这种时变传递函数，SNN 的线性区域变得更加多样化。

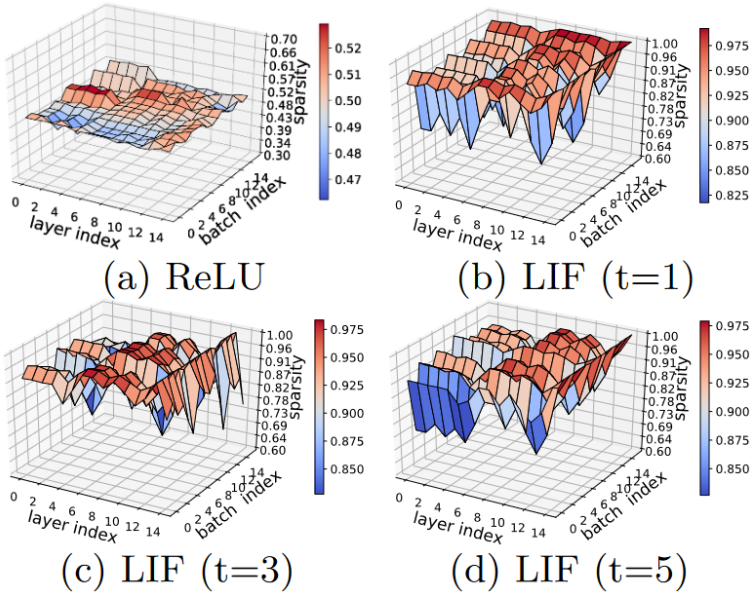

图 2. 小批量中不同层和不同样本的激活模式的稀疏方差。与 ReLU 神经元相比，LIF 神经元的激活模式方差更高。

3.2 稀疏感知汉明距离

在 NASWOT [9] 中，汉明距离 (HD) 是比较两个不同小批量样本 i、j 之间的二元激活模式 c_i、c_j 的关键指标。然而，由于 LIF 的二元激活模式 c 的稀疏性方差较大，标准 HD 给出的 SNN 距离测量不准确。这里，术语“稀疏度”表示给定时间步 t 的一层的二进制激活模式 c 中 0 的百分比。请注意，这里“稀疏性”的定义与之前的工作略有不同，之前的工作从所有时间步的激活中定义“稀疏性”。

观察激活模式的稀疏性。 ReLU 神经元提供了一种二元激活模式，其稀疏性来自高斯/均匀权重初始化，这在所有数据样本中都是相似的。另一方面，LIF 神经元在不同的数据样本中显示出很大的稀疏性变化，因为激活模式基于每个样本中不同的先前膜电位。在图 2 中，我们可视化了具有 16 个小批量样本的二元激活模式的稀疏性。结果表明，LIF 神经元导致不同样本之间存在较大的稀疏性变化。**由于稀疏性变化较大而导致的问题**。这种巨大的稀疏性变化导致了不同尺度的 HD。为了解释这一点，对于数据样本 i，我们将 LIF 神经元输出（在每个时间步）的分布建模为独立同分布。伯努利分布，其中观察到 1 的概率为 $1-r_i^l$：

$$o_i^l \sim \text{Bern}(1-r_i^l). \tag{2}$$

这里，r_i^l 是 l 层二元激活模式的稀疏度。那么，两个数据样本 i、j 之间的激活差异（在同一神经元位置）的概率可以表示为：

$$Pr(|o_i^l - o_j^l = 1) = \text{Bern}(r_i^l(1-r_j^l) + (1-r_i^l)r_j^l). \tag{3}$$

考虑到二元激活模式 $c^l \in R^{N_A^l}$ 的每个元素都是从伯努利分布（方程 2）中采样的，其中 N_A^l 表示第 l 层神经元的数量。那么，HD 的期望（即 $d_H(c_i^l, c_j^l)$）可以表示为：

$$E[d_H(c_i^l, c_j^l)] = N_A^l E[Pr(|o_i^l - o_j^l = 1)] = N_A^l\{r_i^l(1-r_j^l) + (1-r_i^l)r_j^l\}, \tag{4}$$

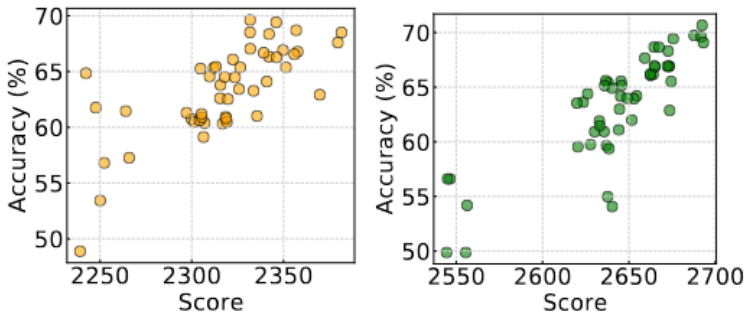

图 3. 架构分数的准确性。我们从搜索空间中随机选择 50 个架构。我们展示了 Kendall's τ 相关性以进行定量比较（HD：$\tau = 0.519$，SAHD：$\tau = 0.646$）。

注意，方程 2, 3, 4 中的所有数量在每个时间步长评估，我们对它们跨时间步长进行平均。

正如我们可以在方程式 4 中观察到的那样。HD 的期望是稀疏度 r_i^l 和 r_j^l 的函数。因此，HD 将为 SNN 提供不准确的距离测量，其中稀疏性 r^l 在数据样本之间有很大的变化（图 2）。例如，如果两个激活处于极端情况下，即高度稀疏 ($r \to 1$) 或高度密集 ($r \to 0$)，则 HD 可能很小。另一方面，如果两次激活处于中等范围内 ($r \approx 0.5$)，则 HD 可能会很高。因此，基于两次激活的稀疏性，HD 对最终分数 s 有不同的贡献（等式 3）；理想的情况是所有 HD 具有相同的贡献。

提出的解决方案。为了解决这个问题，我们提出了稀疏感知汉明距离（SAHD），其中汉明距离根据两个二元激活模式的稀疏性进行归一化。这可以通过将 HD 值的期望标准化为常数 α 来简单地完成：

$$d_{SAH}(c_i^l, c_j^l) = \frac{\alpha}{N_A^l\{r_i^l(1-r_j^l) + (1-r_i^l)r_j^l} d_H(c_i^l, c_j^l). \tag{5}$$

我们通过累积所有层的逐层 SAHD 来计算全局 SAHD 分数，即 $d_{SAH}(c_i, c_j) = \sum_l d_{SAH}(c_i^l, c_j^l)$。我们使用 SAHD 来计算每个时间步的内核矩阵，而不是 HD。之后，我们将所有内核矩阵相加计算最终分数。在图 3 中，我们比较了 HD 和 SAHD 的架构得分和训练后准确率之间的相关性。结果表明，所提出的 SAHD 具有更高的 Kendall's τ 值，这意味着它是一个更准确的架构选择指标。

3.3 搜索前向和后向连接

基于细胞的方法广泛应用于 NAS 研究。这些方法通常搜索连接拓扑以及每个连接的相应操作。然后，多个生成的单元架构构建整个网络。在我们的搜索算法中，我们还研究基于单元的架构。图 4 显示了我们的 SNN 架构的宏观骨架。第一个块是脉冲编码层，它像以前的工作一样直接将浮点值图像转换为脉冲 [13]。骨骼主体由两个搜索神经元细胞和一个还原细胞组成。缩减单元包括一个卷积层和步长为 2 的 2×2 平均池。最后，使用线性分类器进行预测。

细胞搜索策略。我们的单元搜索空间与 NAS-Bench-201 [2] 相同（除了后向连接），其中每个单元包括 $V = 4$ 个节点从具有多个连接的操作集 $O = \{$归零、跳过连接、1×1 卷积、3×3 卷积，3×3 平均池化$\}$ 中采样（见图 4 ）。每个节点包含来自边操作的所有传入特征图的总和。

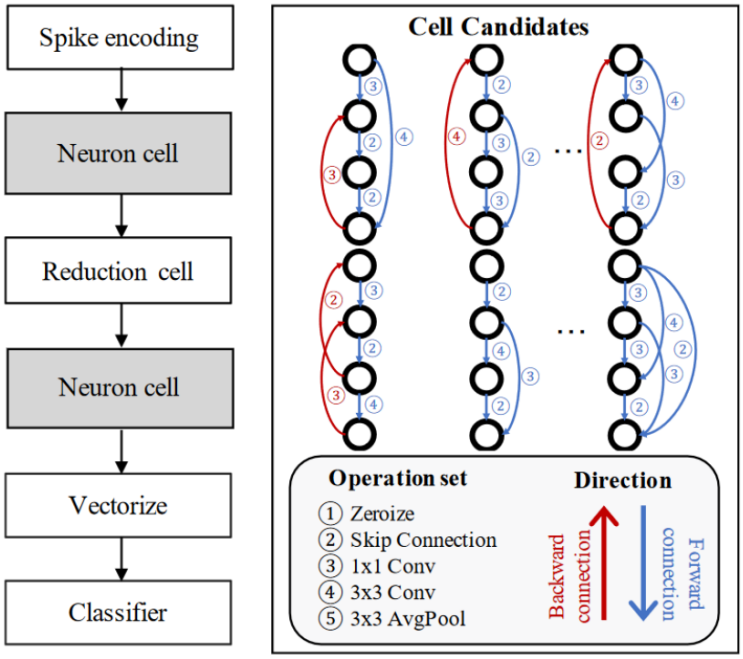

图 4. 基于细胞的神经架构搜索的图示。

然而，与 [2] 不同的是，我们除了前向连接之外还搜索后向连接。在后向操作中，我们将时间步 $t-1$ 处第 l 层的变换后节点特征添加到时间步 t 处第 l' $(l' < l)$ 层的节点。后向连接也具有与前向连接相同的操作集搜索空间 O。在图 4 中，我们展示了候选单元的示例。在预定义的搜索空间中，我们选择最佳的脉冲神经元细胞。缩减单元对特征图的空间大小进行下采样。为了简单起见，我们没有说明清零操作。前向连接和后向连接可以无缝组合。令人惊讶的是，添加后向连接提高了 SNN 的准确性，尤其是在 **CIFAR100** 和 **Tiny-ImageNet** 等复杂数据集上。为了训练搜索到的 SNN，我们使用代理梯度训练 [13]。

4 复现细节

4.1 与已有开源代码对比

本次复现，主要引用了开源在Github的代码。主要的区别在于，在搜索空间中，可选的五个操作集，将 1×1 卷积和 3×3 卷积更换为 3×3 卷积和 5×5 卷积，两个下采样层中的平均池化替换为最大池化。

4.2 实施细节

我使用单个 NVIDIA TITAN X 12GB 完成所有实验。并在 CIFAR10 [7]、CIFAR100 [7]、TinyImageNet [1] 数据集上评估。我的实现基于 PyTorch 。我使用标准 SGD 来训练网络，动量为 0.9，权重衰减为 0.0005，并对输入图像应用随机裁剪和水平翻转。我将训练的批量大小设置为 64。CIFAR10、CIFAR100、TinyImageNet 的基本学习率分别设置为 0.2、0.1、0.1。我使用余弦学习率调度。在这里，我将 CIFAR10、CIFAR100、TinyImageNet 的总 epoch 数分别设置为 300、300、200。我在方程 1 中设置 τ_m 为 $\frac{4}{3}$。我在方程 5 中设置 α 为 $0.5N_A^l$ 以在 LIF 神经元中获得与 ReLU 神经元类似的稀疏度。此外，我从搜索空间中搜索 5000 个候选架

构。我使用 SpikingJelly 包来实现 LIF 神经元。

4.3 创新点

在 [3] 中，重新评估了 SNN 中最大池化和平均池化的性能，发现之前的工作低估了最大池化的性能。并且建议在 SNN 中使用最大池化，因为它具有更低的计算成本、更高的时间拟合能力，以及接收脉冲和输出脉冲的特性，而不是像平均池化那样使用浮点值。因此，我将本次复现工作中的下采样层都替换成最大池化操作。在 [10] 中，搜索空间中使用的是大小为 3×3 和 5×5 的卷积内核，并发现，所搜索到的最优架构，偏向使用 5×5 的卷积内核，或许是因为更大的内核大小，获取信息的能力更强，使得模型的容量增加，因此，我将本次复现工作中的搜索空间操作集的 1×1 卷积和 3×3 卷积更换为 3×3 卷积和 5×5 卷积。

5 实验结果分析

根据原文的方法，进行复现，如表 1 所示。复现结果与原文给出的结果仍有一点差距，可能是因为 NAS 方法本身搜索的随机性，可能不能一次就找到最优架构，加上硬件架构之间的差异也可能会导致这方面的问题。然后我在 CIFAR100 上重复搜索五次，得到平均精度为 70.3±2.12，证实我的猜测。因为硬件资源的限制，原文在 CIFAR10 数据集上，设置的通道数为 128，而我的单个 GPU 无法运行，因此进行了通道数减半的操作进行复现。

表 1. 根据原文的复现结果，其中，SNASNet 是原文所提出的网络，FW 表示前向连接，BW 表示后向连接。

	Dataset	Timesteps	Accuracy(%)
SNASNet-FW	CIFAR10	5	93.12±0.42
SNASNet-FW	CIFAR10	8	93.64±0.35
SNASNet-BW	CIFAR10	5	93.73±0.32
SNASNet-BW	CIFAR10	8	94.12±0.25
SNASNet-FW	CIFAR100	5	70.06±0.45
SNASNet-BW	CIFAR100	5	73.04±0.36
SNASNet-FW	TinyImageNet	5	52.81±0.56
SNASNet-BW	TinyImageNet	5	54.60±0.48
my-FW	CIFAR10	5	92.44
my-BW	CIFAR10	5	92.75
my-FW	CIFAR100	5	69.73
my-BW	CIFAR100	5	70.30±2.12
my-FW	TinyImageNet	5	51.96
my-BW	TinyImageNet	5	53.89

原文是首个在 SNN 网络实现 NAS 方法，因此对比的工作都是手动设计的 SNN 网络，原

文的精度均达到了 SOTA。接着，根据我的改进进行了实验，实现结果如表 2 所示。由于进行一次实验的开销较大，从搜索架构到找到最佳的网络并训练到收敛，需要大约 3 天的时间，因此，改进实验主要针对 CIFAR100 数据集进行，在该数据集上的后向连接设置上，进行了五次实验取其平均。可以看到，在相同的硬件环境下，改进结果还是不错的，且搜索到的网络的精度波动也是相对小一些，并且，最高的精度也已经超过原文提供的结果。

表 2. 基于 CIFAR100 数据集进行的改进实验。improvement 为改进的结果。

	Dataset	Timesteps	Accuracy(%)
SNASNet-FW	CIFAR100	5	70.06±0.45
SNASNet-BW	CIFAR100	5	73.04±0.36
my-FW	CIFAR100	5	69.73
my-BW	CIFAR100	5	70.30±2.12
improvement-BW	CIFAR100	5	72.81±0.79

6 总结与展望

在本文中，使用初始化网络的时间激活模式来搜索更好的 SNN 架构。搜索空间除了前向连接之外还考虑后向搜索连接，这带来了使用时间信息的好处。通过实现比以前的工作更好的性能，证明了一种新型架构更适合 SNN，其中脉冲通过多个时间步传递信息。在未来的工作中，尝试将此方法迁移到不同的任务中，目前该方法仅适用于基于卷积的图像分类任务。

参考文献

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[2] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

[3] 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*, pages 2661–2671, 2021.

[4] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *Computer Vision – ECCV 2020*, pages 544–560, Cham, 2020. Springer International Publishing.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456. PMLR, 2015.

[6] E.M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572, 2003.

[7] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[8] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2019.

[9] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, pages 7588–7598. PMLR, 2021.

[10] Byunggook Na, Jisoo Mok, Seongsik Park, Dongjin Lee, Hyeokjun Choe, and Sungroh Yoon. Autosnn: Towards energy-efficient spiking neural networks. In *International Conference on Machine Learning*, pages 16253–16269. PMLR, 2022.

[11] 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.

[12] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2847–2854. PMLR, 06–11 Aug 2017.

[13] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1311–1318, 2019.