**SNN程序目录中的代码共给出了两个模型，一个单步一个多步，研究单步和多步的共同点和区别。多步相比单步的优势在哪里，分析模型实现什么功能。**

**单步模型（SNN\_model）与多步模型（Convolutional\_SNN）**

**共同点**

**两种模型都共享脉冲神经网络基础，**也即均使用 spikingjelly 框架的脉冲神经元LIFNode/IFNode和替代梯度；输出层均为10个神经元（对应MNIST的10分类）；均采用 ATan 作为反向传播的**替代梯度函数**，解决脉冲不可微问题。

**核心区别**

|  |  |  |
| --- | --- | --- |
| **特性** | **单步模型 (SNN\_model)** | **多步模型 (Convolutional\_SNN)** |
| **时间维度** | 无时间步（单步推理） | 显式时间步 T（多步动态） |
| **结构** | 单层全连接（Flatten + Linear） | 深度卷积结构（卷积层+池化+全连接） |
| **神经元类型** | LIFNode | IFNode（积分发放，无泄漏） |
| **输入处理** | 直接处理静态数据 | 时间复制 |
| **输出机制** | 直接输出脉冲 | 时间平均脉冲发放率 |
| **计算复杂度** | 极低（仅1层） | 较高（多层卷积+时序展开） |

**多步模型的核心优势**

多步模型通过时间步长T模拟生物神经元的时序动态特性，能够有效捕获输入数据的时间依赖性，在处理动态视觉信息时展现出显著优势——例如在分析视频帧序列时，可准确识别物体运动轨迹，而单步模型仅限于静态帧处理。该模型利用脉冲发放率编码机制，将多个时间步的离散脉冲信号转化为连续概率值输出，大幅提升了分类的鲁棒性，有效克服了单步模型瞬时脉冲输出易受噪声干扰的缺陷。同时，其独特的时空融合架构结合了卷积层的空间特征提取能力（如边缘和纹理识别）与脉冲神经元的时间信息整合机制，形成高效的时空联合表征。这种多步仿真机制更贴近生物神经元的工作特性，通过模拟膜电位累积、阈值触发发放和时间信息整合等生物过程，在保持高生物可解释性的同时，实现了对复杂时空模式的有效解析。

**模型功能实现**

**单步模型 (SNN\_model)**

**功能**：极简脉冲分类器

输入：展平的MNIST图像（[Batch, 784]）→ 全连接层 → LIF神经元 → 输出10维脉冲。

**局限**：仅适合静态数据分类，无法处理时序信息，特征提取能力弱。

**多步模型 (Convolutional\_SNN)**

**功能**：深度脉冲卷积网络

输入 [N,1,28,28] → 复制T次 → 卷积+池化（空间特征）→ IF脉冲神经元（时间整合）→ 全连接 → 输出脉冲发放率

**关键流程**：

* 1. **时间扩展**：输入数据复制 T 次，形成 [T, N, C, H, W]。
  2. **卷积特征提取**：两层卷积+池化（输出尺寸 7x7）。
  3. **脉冲时序整合**：IFNode 在多个时间步累积膜电位，触发脉冲发放。
  4. **脉冲率解码**：对 T 步输出取平均，将脉冲计数转化为分类概率,最高脉冲率对应预测类别。

**优势场景**：动态视觉识别、时序信号处理。

**问题一：提升模型性能的参数调整（看实验）**

**可调整的关键参数及其影响：**

值得注意的是，我做了特别多组对比实验，受到实验时间的限制，我选定了10个epoch，使用单卡A800进行训练与测试，实验证明10个epoch已经过了上升期，后续的训练只是微小的增进。

1. **时间步长 T**：

增加 T 能提高时间分辨率，使脉冲发放更精确；但会增加计算开销，存在收益递减点

在训练代码中调整方法为：parser.add\_argument('T', default=8, type=int)

实验如下：

我们控制tau, V\_threshold,surrogate\_func不变，T设置4,8,16,32四组实验，结果如下图1-1至1-4，结果大致证明随着T增加，分类效果越来越好。

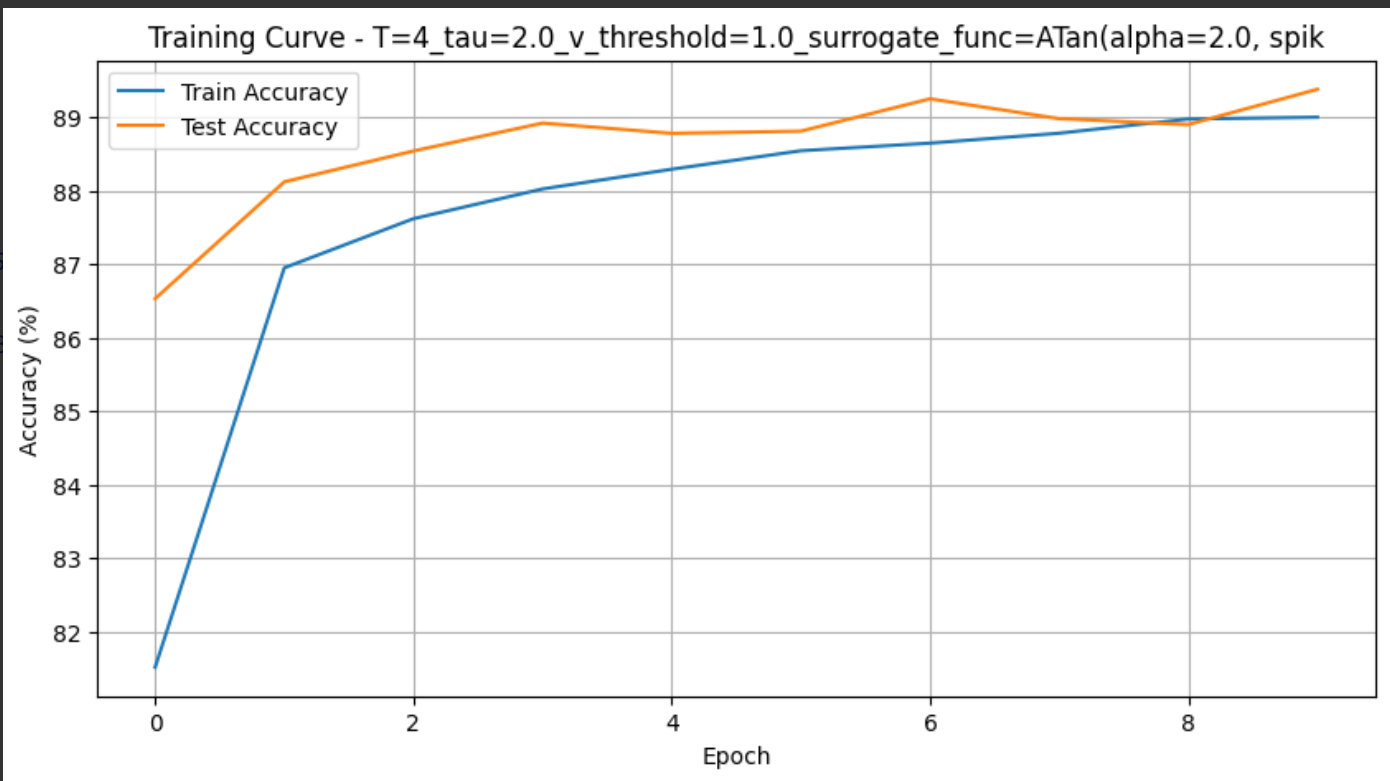


图1-1

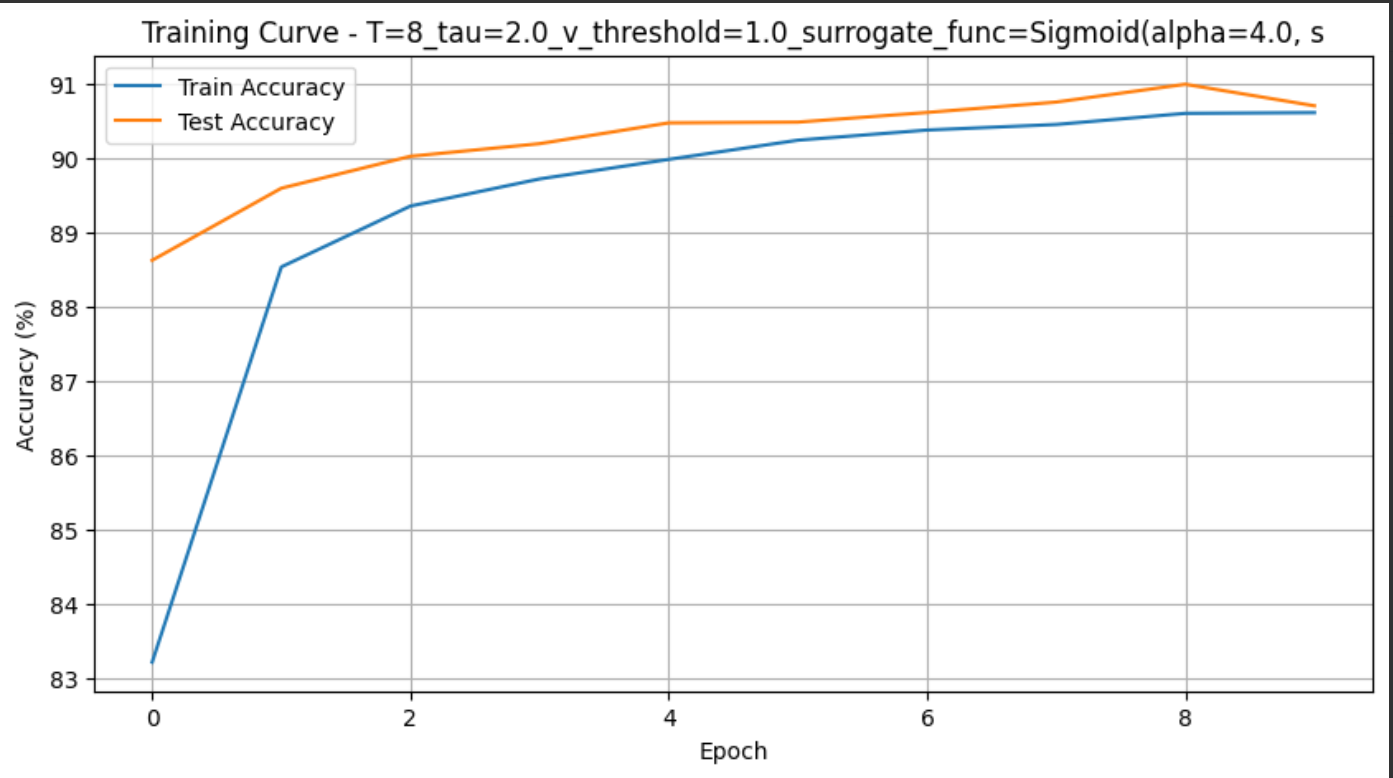


图1-2

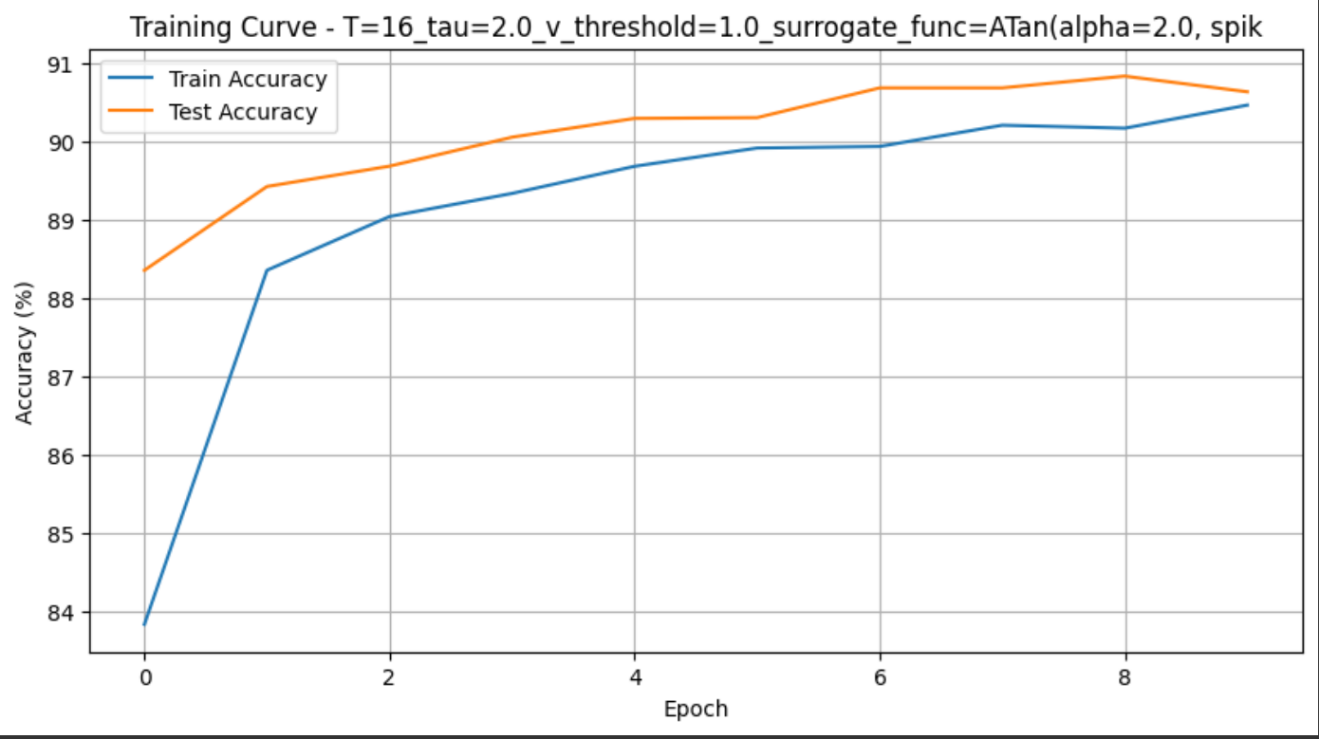


图1-3

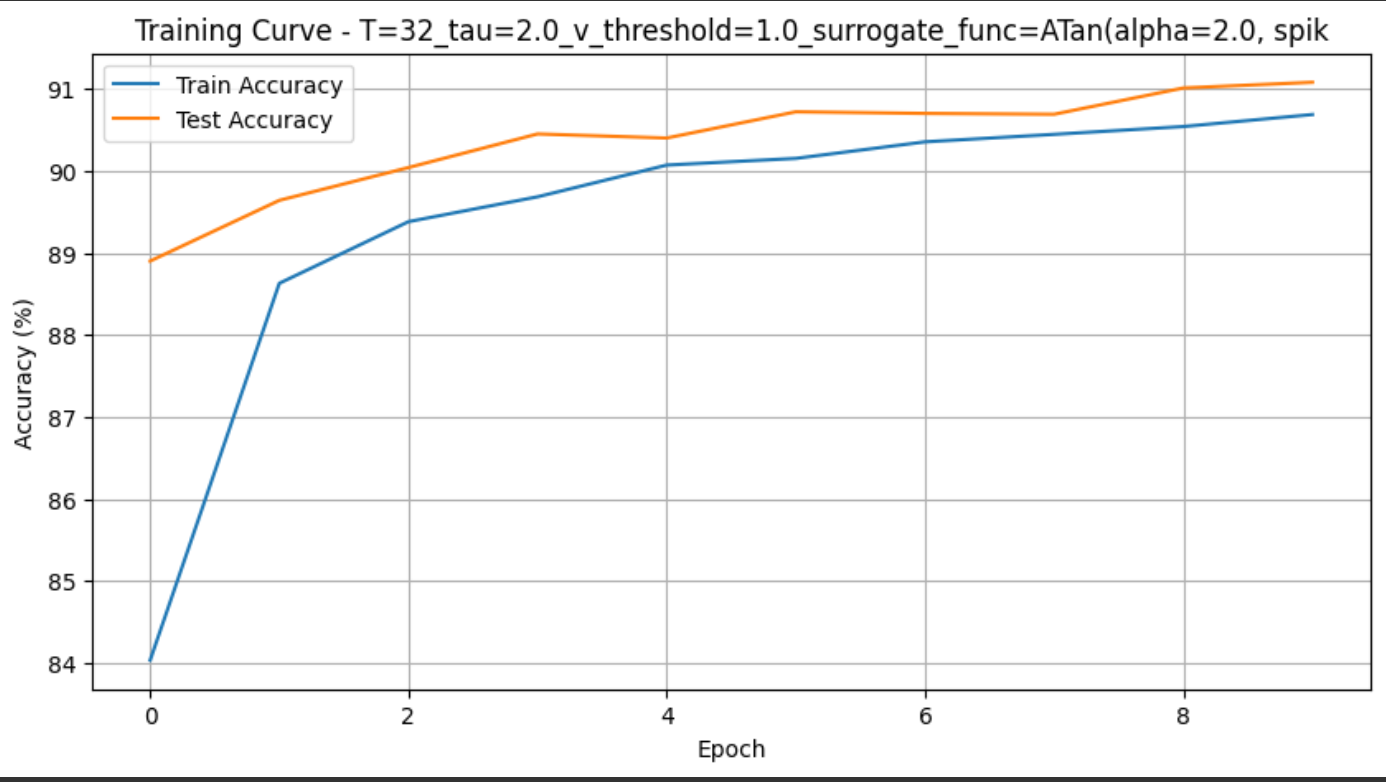


图1-4

1. **发放阈值 v\_threshold**：

降低阈值会使神经元更易激活，增加脉冲发放频率；提高阈值会降低脉冲发放频率，增加稀疏性。

在神经元定义中修改方法为：

neuron.LIFNode(tau=tau, v\_threshold=1.0, surrogate\_function=surrogate.ATan())

我们固定tau=2, T=8，surrogate\_function=surrogate.ATan()，设置v\_threshold为0.5,1.0,1.5,2.0，结果如下图2-1至2-4：

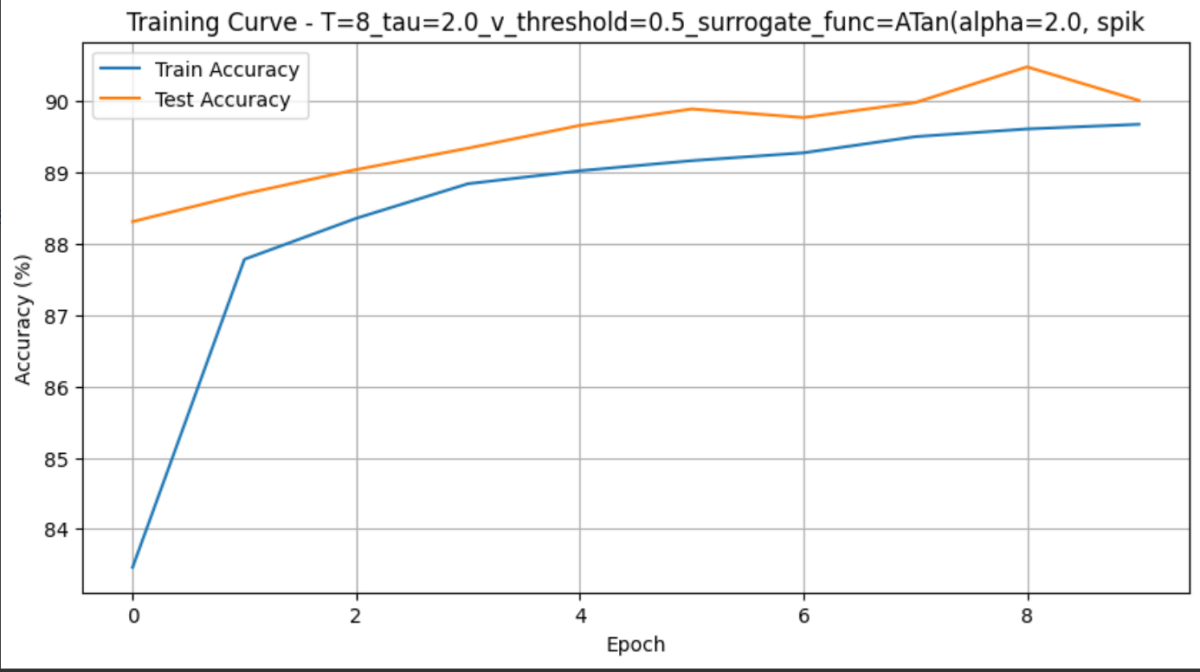


图2-1

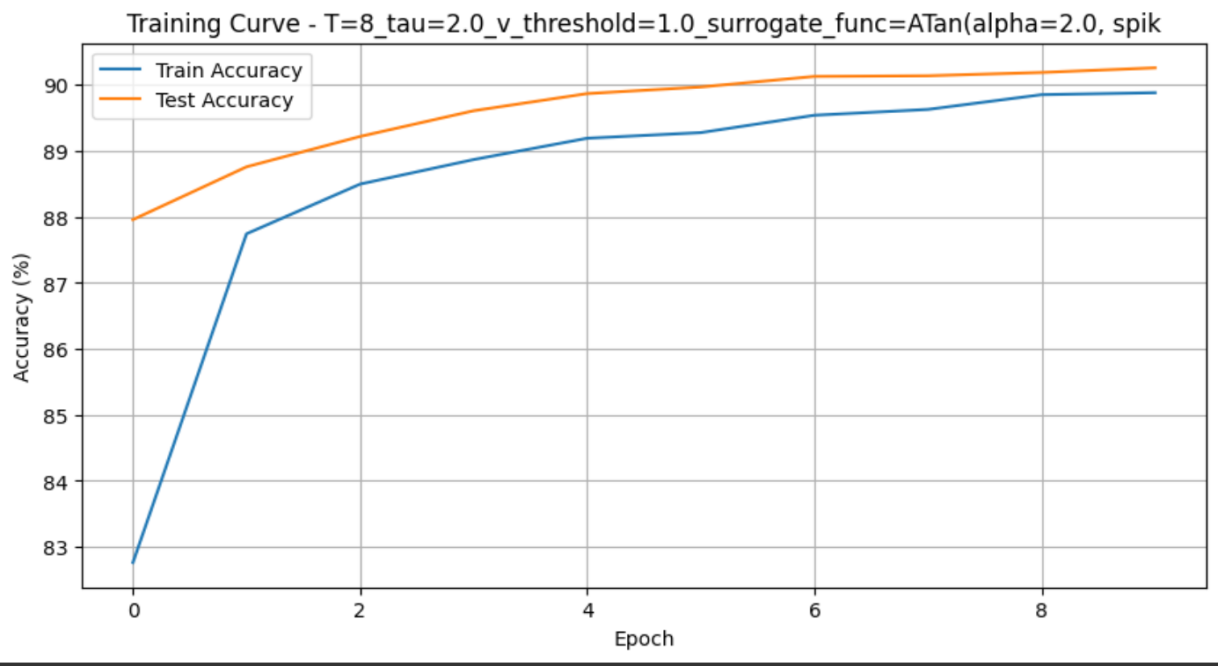


图2-2

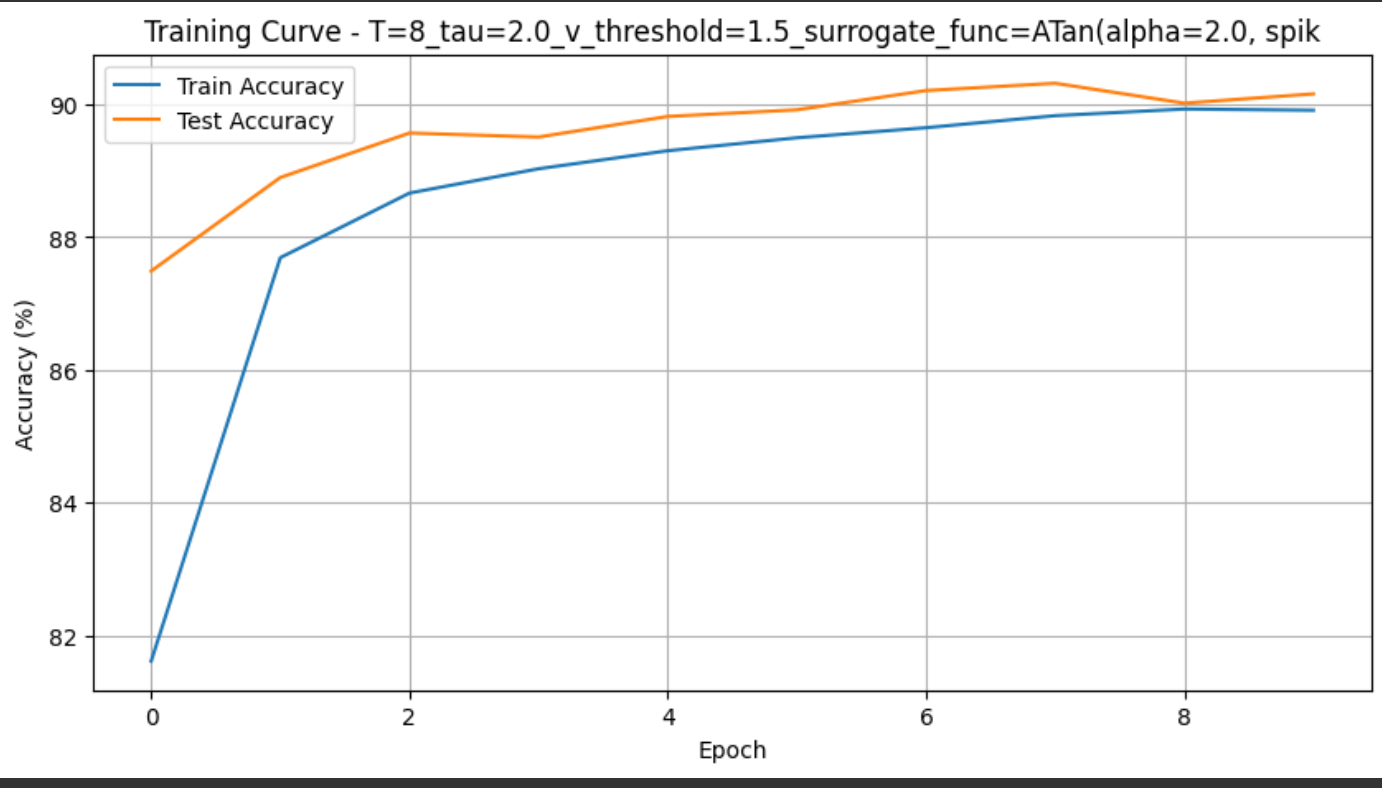


图2-3

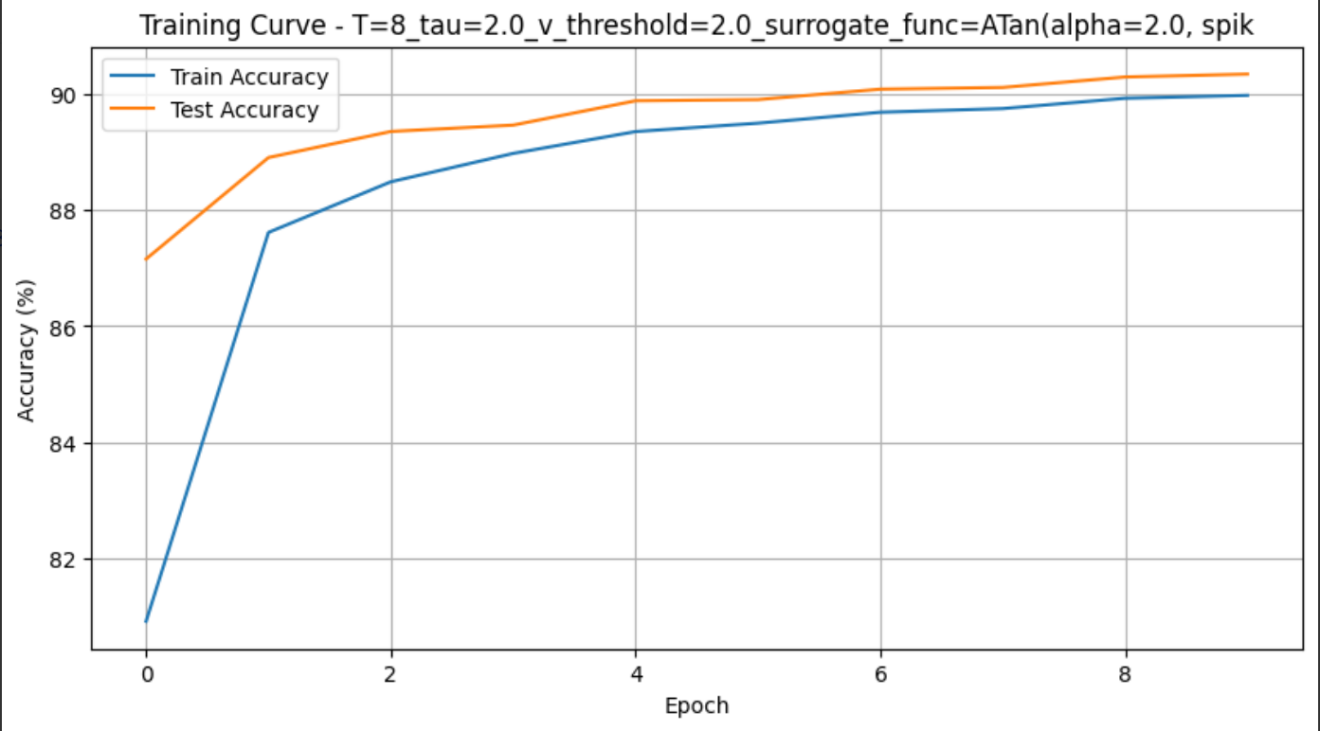


图2-4

通过实验分析，v\_threshold 的变化对结果影响并不是很明显。仔细讨论之后，我觉得：如果输入脉冲强度或初始权重很小，所有膜电位都集中在比较低的区间，即使把阈值从 0.5 提到 2.0，也都没超过阈值；反之，如果权重很大，膜电位一旦超过最小阈值，就会迅速远超所有阈值设置，从而每个时间步输出几乎相同的发放模式。

那如何解释这个膜电位集中在平稳区间？ 当 tau=2（意味着每步膜电位衰减到上一步的大约 e^{-1/2}=0.6），再加上 T=8 步的输入累积，膜电位最终分布可能已经形成一个平稳区间，连续的多次发放-重置循环让不同阈值的影响被“平滑”掉了。

1. **泄漏时间常数 tau**：

较小的 tau使神经元快速遗忘历史信息；较大的 tau使神经元保留更多历史信息

在模型初始化中调整：SNN\_model(tau=args.tau) tau范围1.1,2.0,4.0，具体结果如图3-1至3-3：

可以看到，结果tau较小，性能稍好。这是符合直觉的，MNIST 这类静态图像分类任务不依赖长时序记忆，MNIST 每张手写数字图像本身就是一个静态样本，我们人为将它“切分”到多个时间步上，只是为了给 SNN 提供离散的脉冲输入窗口。本质上它并不需要跨很长时间步去“记住”早先的脉冲模式，快速遗忘让网络更专注于每个时间步的当前输入脉冲，避免无关的历史信息干扰；长时记忆反而会把前几个时间步的脉冲残留过度保留到后面，可能把早期微弱或噪声信号都累积进来，降低判别清晰度。

当 tau 较小时，膜电位衰减快，只有当前强度最高的脉冲输入最可能触发放电，隐藏层／输出层的活动模式更“干净”、更稀疏；这种稀疏而专注的放电模式反过来提供了更集中、更稳定的梯度信号，有助于权重学习。反之，过大的 tau 会让膜电位在多个时间步内持续累积，导致放电频率变高且难以区分关键特征，梯度信号也更分散、噪声更多。

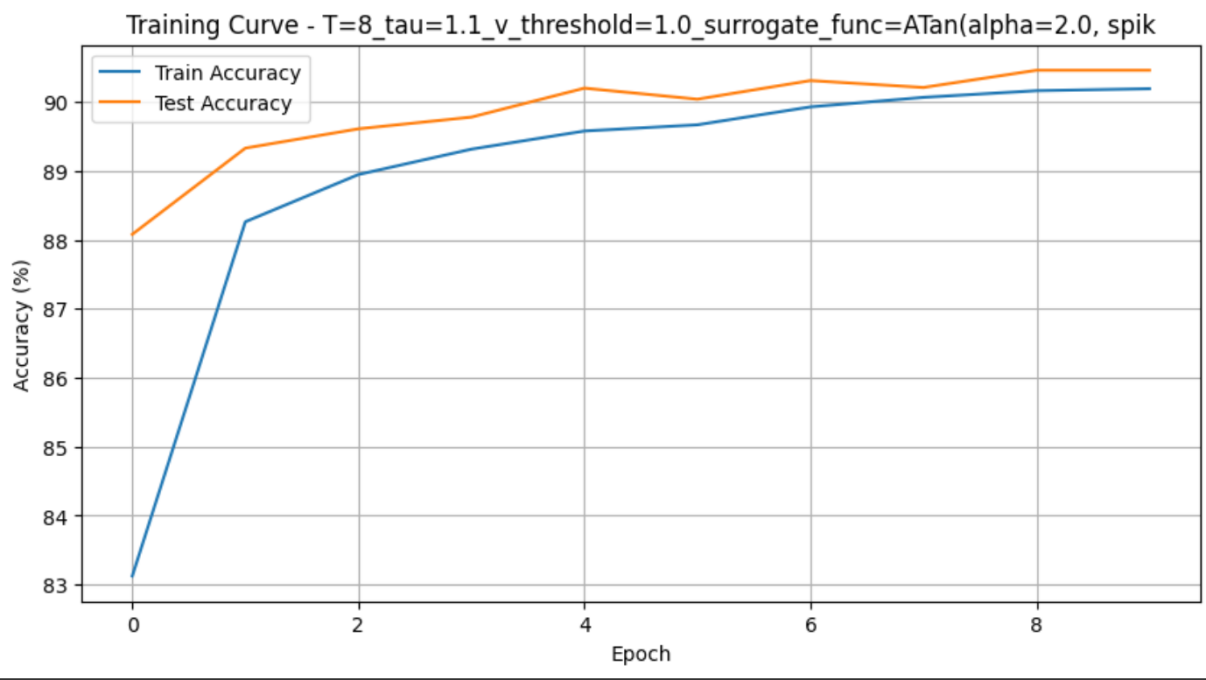


图3-1

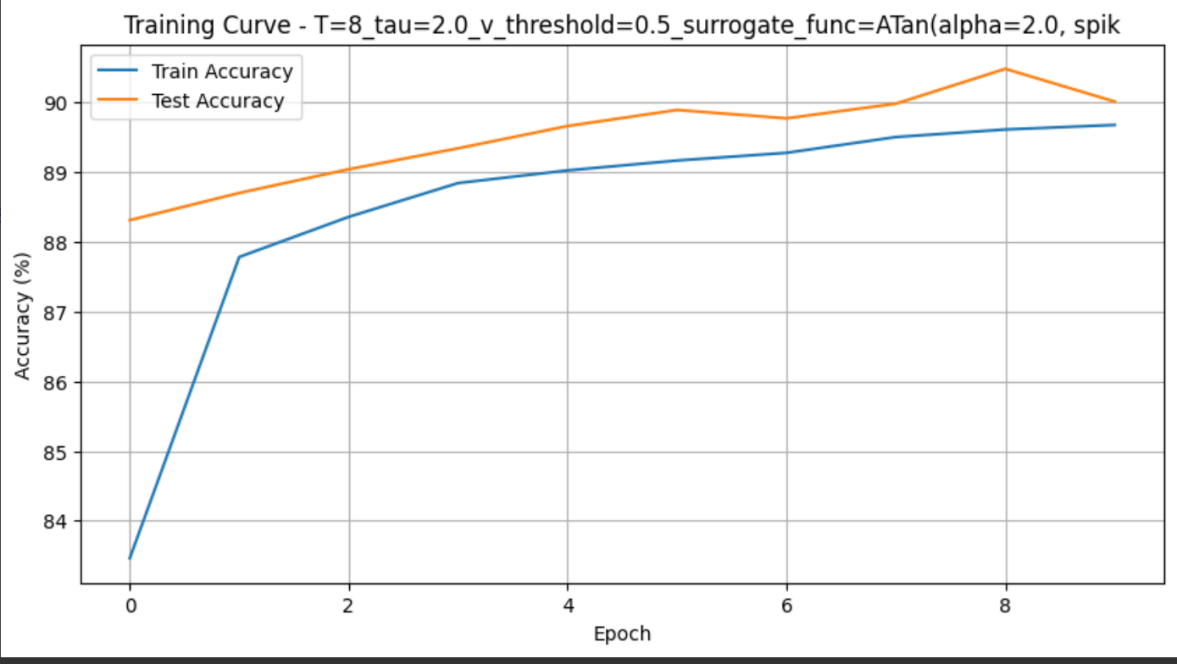


图3-2

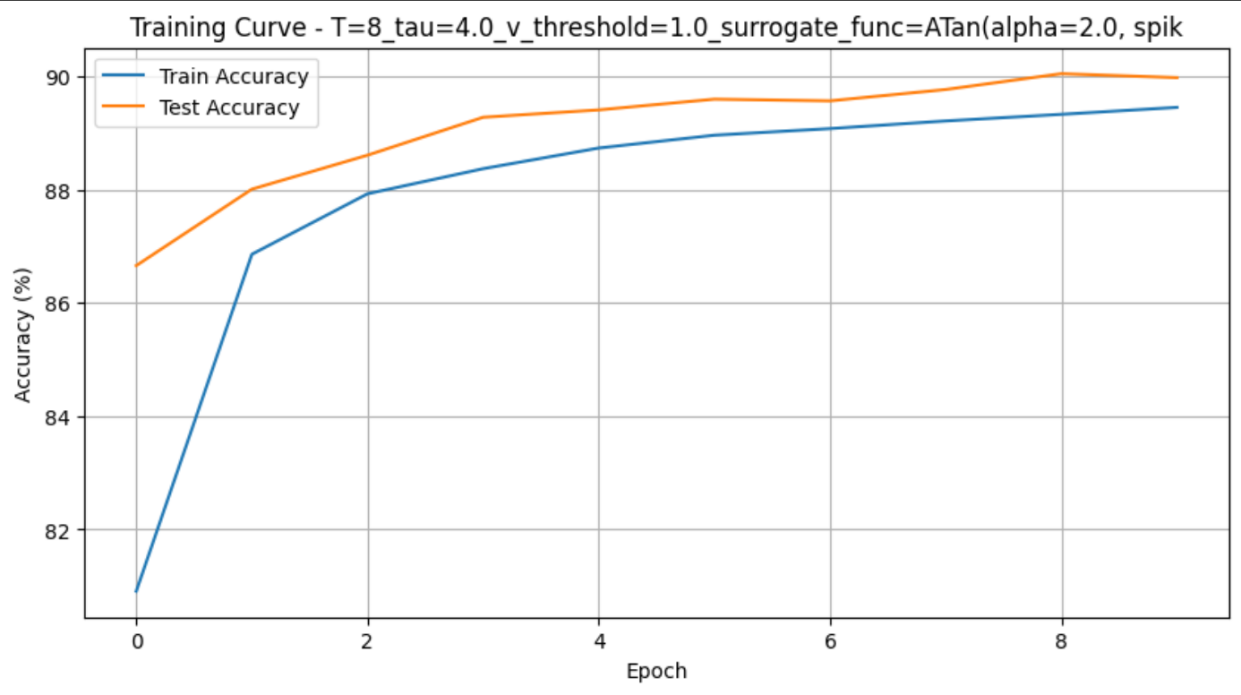


图3-3

1. **替代函数类型，具体结果如图4-1至4-2：**
   * 不同替代函数影响梯度传播特性：

# 可选替代函数  
surrogate.ATan() # 计算量适中，性能稳定  
surrogate.Sigmoid() # 梯度平滑，训练稳定

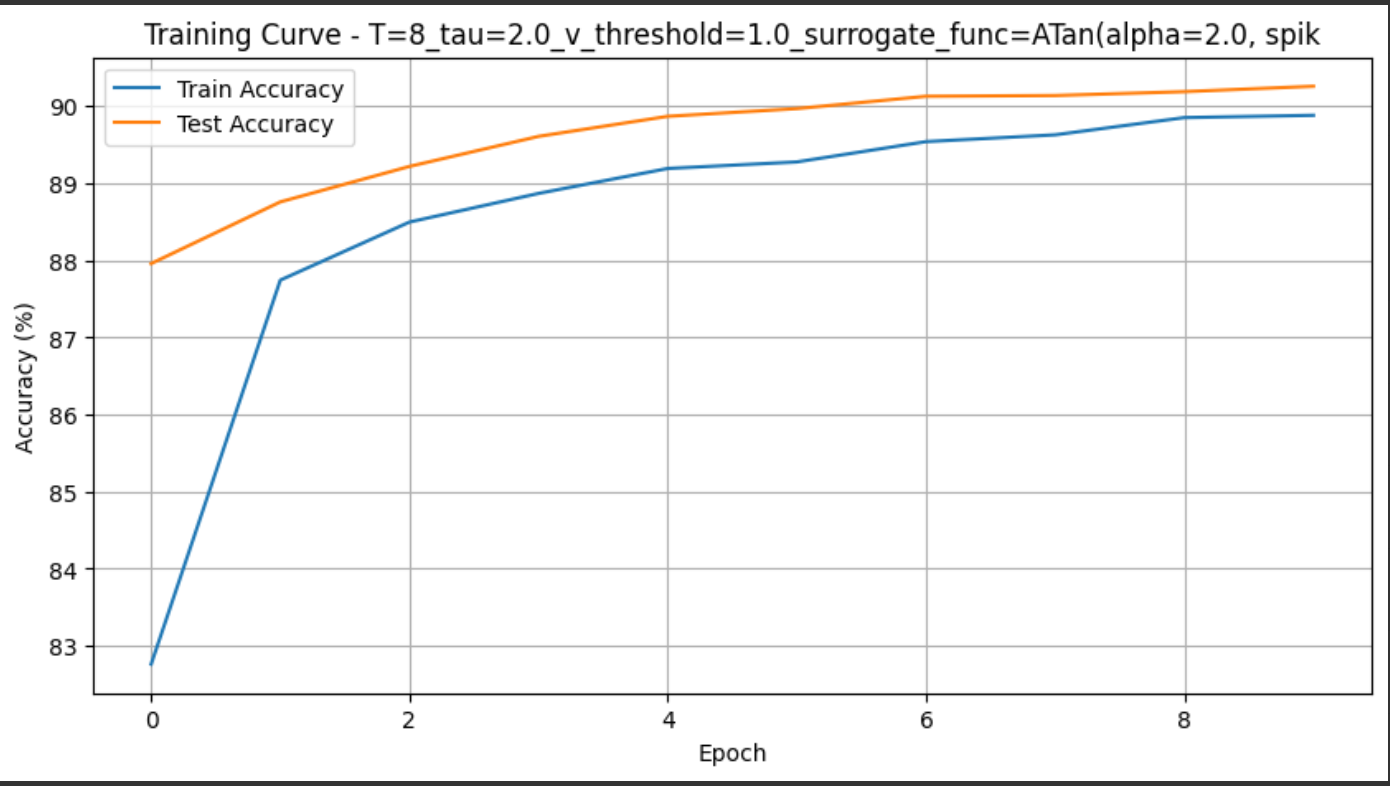


图4-1

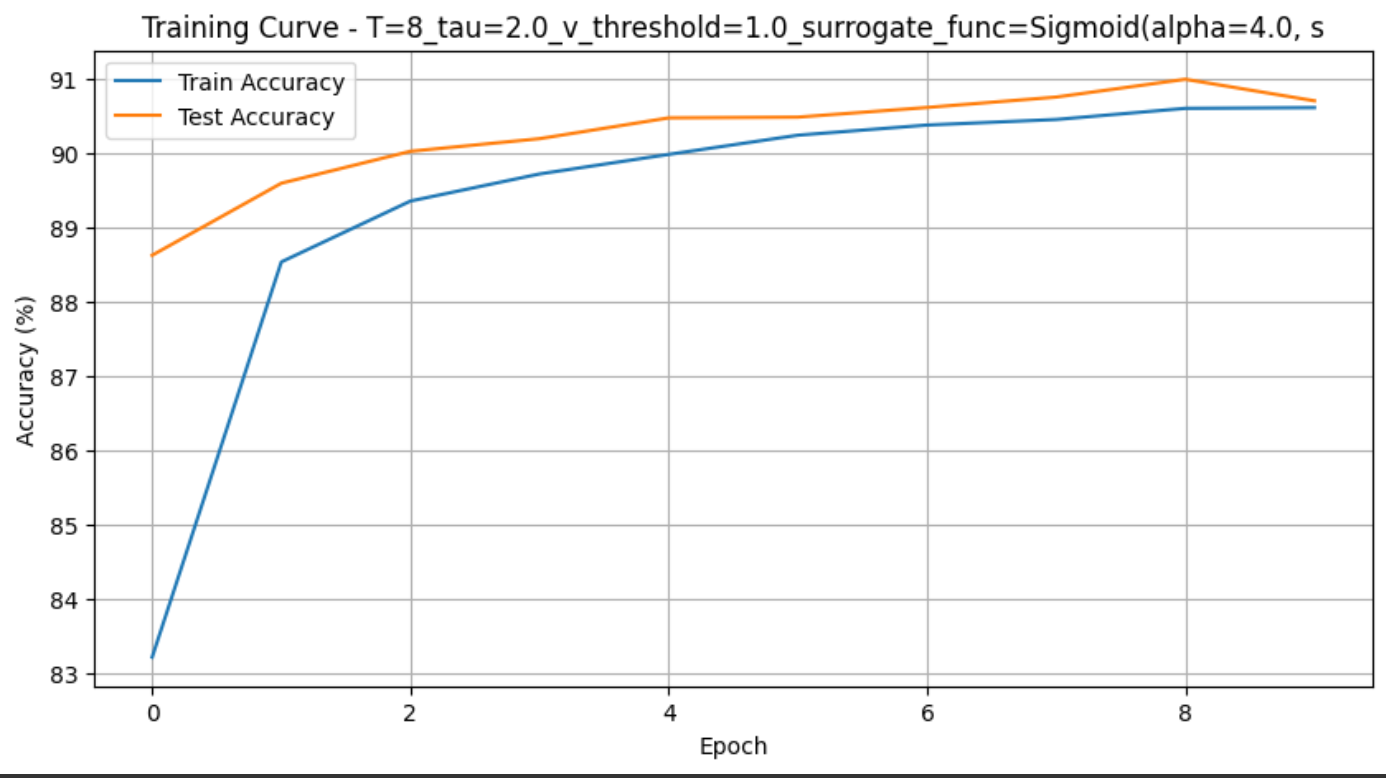


图4-2

假设我们对 LIF 神经元的脉冲 做“硬”阶跃：

为了解决死区不可导问题，训练时用代理梯度 来近似阶跃函数的导数。两者常见形式约简如下（以输入 为自变量）：

**ArcTan surrogate**

在 附近导数最大：（其实最陡处不在 0 点，而是两侧小偏移处）；

**衰减较快**： 当 时梯度趋近 0。

也就是说：在微小偏移处梯度最陡，但过了这个区间，就迅速衰减到接近 0，导致两种风险：如果神经元膜电位始终偏离阈值超过小范围，梯度几乎为零；当残差漂移到饱和区（也即远离阈值）时，学习停滞。

**Sigmoid surrogate**

在 处导数最大：；对中等大小的 （如 ）仍保留较显著梯度。

**Sigmoid** 对阈值上下更大的范围（大约 ）都能保留可用梯度，意味着：即便膜电位偏差稍大，也能收到梯度信号；更新步更平缓、不易“跳过”最佳区间，训练更加稳定。

因此，MNIST 通常分多帧喂脉冲，但核心是静态特征分类，不需要非常精准的“阈值时刻”控制；Sigmoid 的宽阈值带来的平滑梯度更契合这种弱时序、弱依赖的场景。 ArcTan 导致更窄的放电窗口，容易过度稀疏，关键时刻可能放电不足；Sigmoid 平滑残留，让隐藏/输出层在多个时刻都有合适发放，信息更充分。从收敛的角度，Sigmoid 在多数初始权重和膜电位分布下，都能保证“落到”有梯度的区间进行有效更新；ArcTan 则对初始分布更敏感，可能需要更精细地调参才能稳定训练。

1. **膜电位复位机制**：

**硬复位** vs 软复位（保留超过阈值的残余电位）：

neuron.LIFNode(..., reset\_mechanism='subtract') # 软复位（默认）  
neuron.LIFNode(..., reset\_mechanism='zero') # 硬复位

该任务更适合硬复位，这很显然。

1. **神经元类型选择**：
   * LIFNode ：更接近生物神经元特性
   * IFNode ：计算更简单，无泄漏项

在 SNN 中，膜电位复位机制可以选择“硬复位”或“软复位”：硬复位在神经元发放脉冲后会将膜电位直接清零，这种方式使放电更稀疏、每次脉冲互不干扰，却也彻底丢弃了阈值以上的残余信息；而软复位仅减去阈值并保留超阈残余，能够让连续高强度输入更快触发多次放电，但若残余叠加过多，又容易导致膜电位过度饱和与频繁发放；与此同时，神经元模型的选择也对网络行为影响显著，LIFNode 带有指数泄漏项，会随时间自然衰减旧信息，从而平衡新旧输入并维持脉冲稀疏，适合需要跨时序记忆和生物拟真度较高的任务；而 IFNode 则完全累积输入、不做衰减，结构更简单、计算量更小，却容易在持续输入下出现膜电位过载或数值发散、梯度不稳定的问题。根据具体任务对时序依赖、稀疏性和计算资源的要求，应在复位机制和神经元类型之间进行权衡，选择最合适的组合。

**总结：** 在使用 LIFNode 时，我们可以通过调整或新增的参数来精细控制模型的时序行为、稀疏性和训练稳定性。

tau决定了膜电位随时间的衰减速度，较小的 tau 会让神经元快速忘却历史输入，适合静态或短时依赖任务；而较大的 tau 则保留更多历史信息，但容易导致过饱和和高频放电。

v\_threshold越低神经元越易触发脉冲、放电率提升，反之则更稀疏；

reset\_mechanism 则可选“硬复位”或“软复位”，前者在放电后将膜电位清零以避免残余累积，后者保留超过阈值部分以加速连续高强度输入的放电反应。

代理梯度函数 surrogate\_function（如 ATan、Sigmoid）影响反向传播时的梯度分布——平滑且衰减慢的 Sigmoid 往往能在静态任务上获得更稳定的训练，而曲线更陡峭的 ATan 则适合对阈值周围变化更敏感的应用。

通过系统地在不同参数配置下对比脉冲率、梯度范数分布和最终分类精度，我感受到了这些参数对 SNN 模型性能的影响。

**问题二：输入数据编码方式及影响**

**代码中实现的编码方式：**

encoder = encoding.PoissonEncoder() # 泊松编码

**1. 加权相位编码器（WeightedPhaseEncoder）**

* **原理**：对输入 按二进制位展开，高位先发、低位后发，每位携带不同权重。
* **步骤**：
  1. 令残余 inputs = x，初始权重 。
  2. 对每个时刻 ：
     + 发放脉冲：。
     + 更新残余：。
     + 权重衰减：。
* **时序输出**：一共 步，每步输出与输入同形状的二值脉冲张量，携带对应二进制位信息。

class WeightedPhaseEncoder(StatefulEncoder):  
def \_\_init\_\_(self, K: int, step\_mode='s'):  
 super().\_\_init\_\_(K, step\_mode)  
  
def single\_step\_encode(self, x: torch.Tensor):  
 # 输入范围 [0, 1-2^{-K}]  
 assert (x >= 0).all() and (x <= 1 - 2 \*\* (-self.T)).all()  
 inputs = x.clone()  
 # 预分配 [T, \*x.shape] 的输出张量  
 self.spike = torch.empty((self.T,) + x.shape, device=x.device)  
 w = 0.5  
 # 从高位到低位逐步比较、减权重  
 for t in range(self.T):  
 self.spike[t] = inputs >= w  
 inputs -= w \* self.spike[t]  
 w \*= 0.5

**2. 泊松编码器（PoissonEncoder）**

* **原理**：根据输入强度 作为脉冲发放概率，逐元素生成随机二值脉冲。
* **公式**：

其中 为与 同形状的随机矩阵。

* **实现要点**：
  + 无状态，调用 forward(x) 即可；
  + 每次调用生成新的随机脉冲，适合需要增强时序多样性的场景。

class PoissonEncoder(StatelessEncoder):  
def \_\_init\_\_(self, step\_mode='s'):  
 super().\_\_init\_\_(step\_mode)  
  
def forward(self, x: torch.Tensor):  
 # 每个元素以 x 作为概率独立采样  
 return torch.rand\_like(x).le(x).to(x)

**3. 延迟编码器（LatencyEncoder）**

* **原理**：将连续输入 映射到一个**首发时刻** ，发放一次脉冲，其它时刻保持零。
* **映射函数**：
  + 线性版本：
  + 对数版本：
* **实现要点**：
  + 计算 后，用 one-hot 编码生成形状 的脉冲序列；
  + 在第 次调用时输出 spike[t]；
  + 有状态编码器，首次传入原始 后会缓存完整序列。

class LatencyEncoder(StatefulEncoder):  
def \_\_init\_\_(self, T: int, enc\_function='linear', step\_mode='s'):  
 super().\_\_init\_\_(T, step\_mode)  
 if enc\_function == 'log':  
 # 保证 t\_f(1) = T-1  
 self.alpha = math.exp(T - 1.) - 1.  
 elif enc\_function != 'linear':  
 raise NotImplementedError  
 self.enc\_function = enc\_function  
  
def single\_step\_encode(self, x: torch.Tensor):  
 # 计算首发时刻 t\_f  
 if self.enc\_function == 'log':  
 t\_f = (self.T - 1. - torch.log(self.alpha \* x + 1.)).round().long()  
 else:  
 t\_f = ((self.T - 1.) \* (1. - x)).round().long()  
  
 # one-hot -> [T, \*x.shape]  
 self.spike = F.one\_hot(t\_f, num\_classes=self.T).to(x)  
 # 轴变换：从 [\*, T] -> [T, \*]  
 dims = list(range(self.spike.ndim - 1))  
 dims.insert(0, self.spike.ndim - 1)  
 self.spike = self.spike.permute(dims)

在 MNIST 任务中，加权相位编码器 通过将每个像素值拆分为二进制位，并在每个时间步用相应位权（如 2⁻¹、2⁻²…）控制脉冲发放，既能在最早的几个步骤传递最高位信息，又能随着时间推移逐步补齐低位贡献，因而在保持稀疏性的同时极大地丰富了单次脉冲所携带的数值精度；这种方法在 MNIST 上往往能取得较高的准确率，但由于需要对每个输入像素展开多位二进制并在多步中进行比较，计算和存储开销都显著高于其它编码方式。相比之下，泊松编码器则直接依据像素强度生成服从该强度概率分布的随机脉冲序列，结构极其简单、无状态且易于并行实现，能够天然地为后续网络提供多样化的输入样本，但由于内在的随机性，它在 MNIST 任务上收敛速度较慢且需要多次滴定随机过程才能稳定表现，训练过程也更容易受到噪声干扰。具体结果如图5-1至5-3：

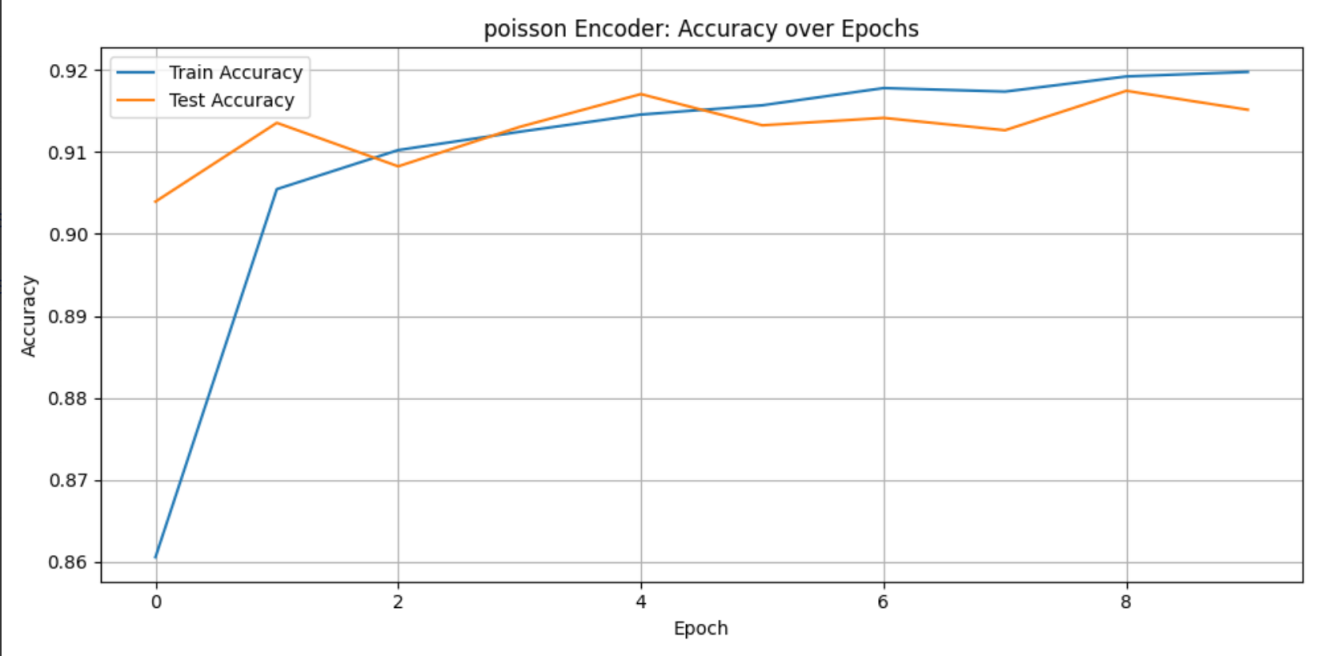


图5-1

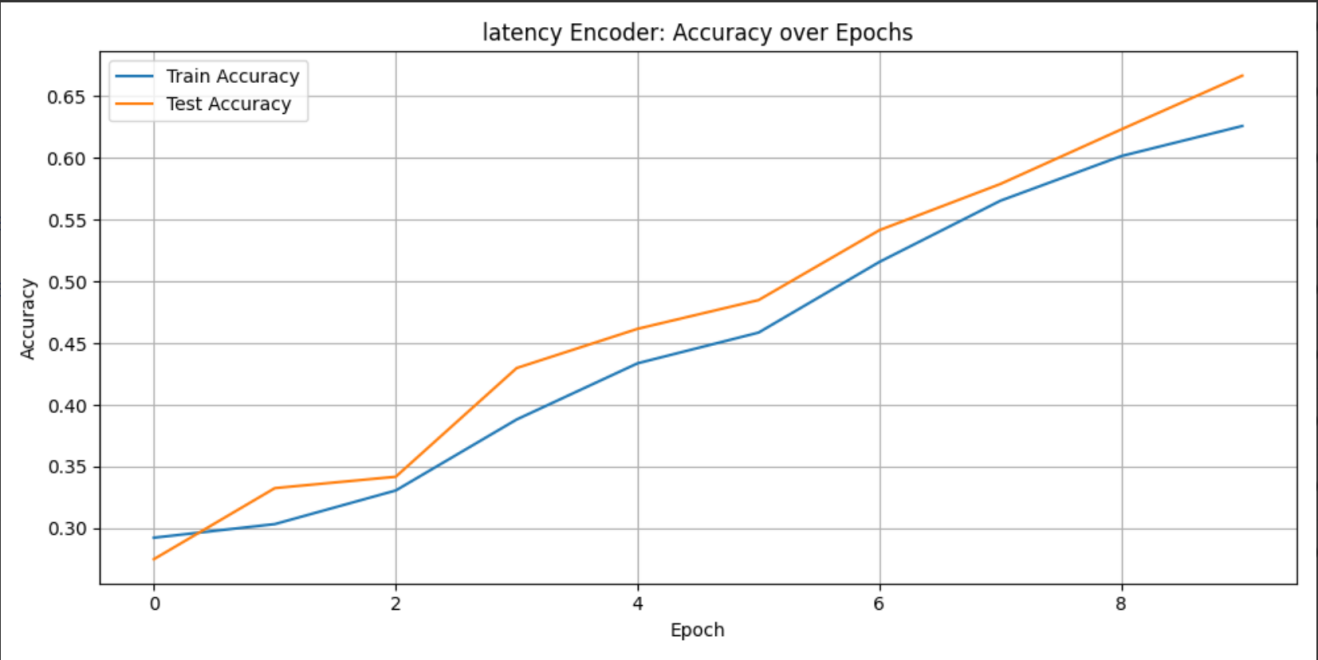


图5-2

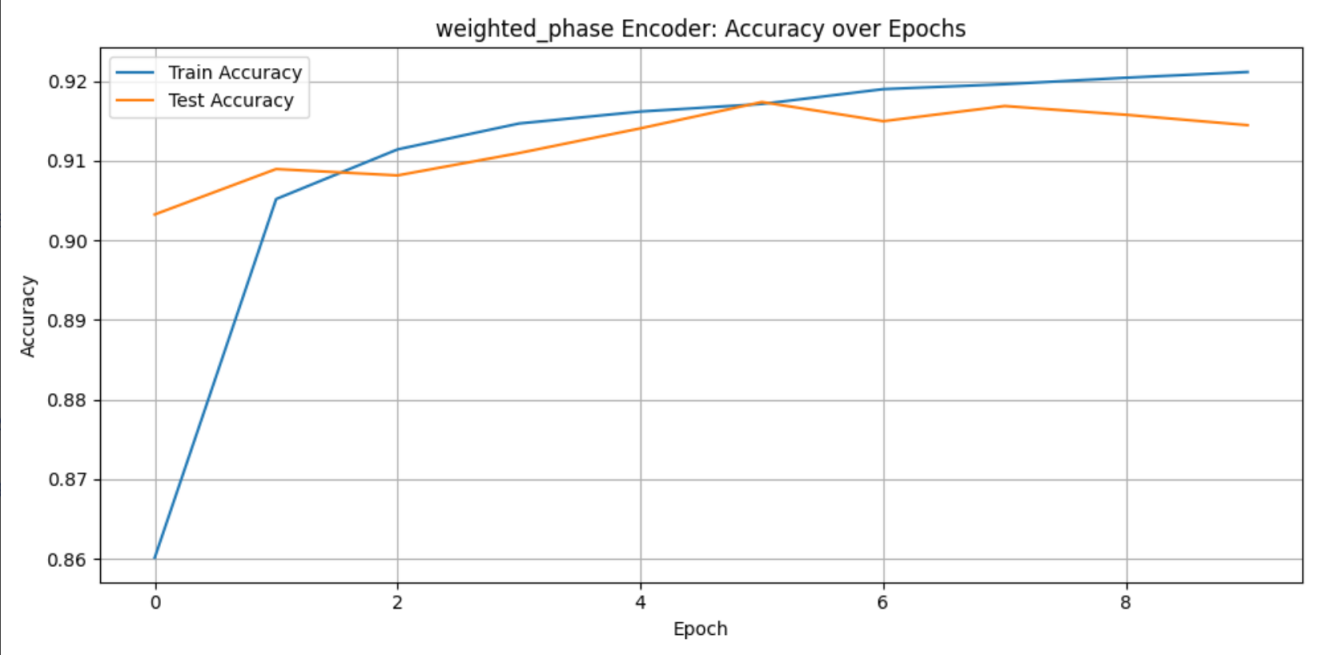


图5-3

延迟编码器之所以在 MNIST 上往往落后于加权相位编码和泊松编码，根本原因在于它每个神经元在整个仿真周期内仅发放一次脉冲，等于是用一个时刻的“单一事件”来承载像素强度，导致时间维度上的信息冗余和梯度信号极度匮乏；与此同时，所有神经元的脉冲高度集中在某几个映射时刻，使得隐藏层和输出层在大多数时间步几乎无活动，网络难以捕捉细微的强度差异，也无法通过连续的脉冲来平滑学习；此外，延迟编码对映射函数的量化精度和时间窗口长度极度敏感（实验不好设置，需要细细选取），哪怕是微小的量化误差或映射抖动，都可能造成显著性能波动。相比之下，加权相位编码通过二进制位在多个时间步分摊信息，而泊松编码凭借随机过程提供了更丰富的时序样本，因此两者在 MNIST 这类静态分类任务中均能保持更稳定、更充分的特征表达，从而取得更优的表现。

**问题三：与其他模型结合的改进方案**

**与VGG集成的代码实现：**

class VGG\_SNN(nn.Module):  
 def \_\_init\_\_(self, T: int, channels: int, use\_cupy=False):  
 super().\_\_init\_\_()  
 self.T = T  
 # VGG风格的卷积块  
 self.features = nn.Sequential(  
 layer.Conv2d(1, channels, kernel\_size=3, padding=1),  
 layer.BatchNorm2d(channels),  
 neuron.IFNode(surrogate\_function=surrogate.ATan()),  
   
 layer.Conv2d(channels, channels, kernel\_size=3, padding=1),  
 layer.BatchNorm2d(channels),  
 neuron.IFNode(surrogate\_function=surrogate.ATan()),  
 layer.MaxPool2d(2, 2),  
   
 layer.Conv2d(channels, channels\*2, kernel\_size=3, padding=1),  
 layer.BatchNorm2d(channels\*2),  
 neuron.IFNode(surrogate\_function=surrogate.ATan()),  
   
 layer.Conv2d(channels\*2, channels\*2, kernel\_size=3, padding=1),  
 layer.BatchNorm2d(channels\*2),  
 neuron.IFNode(surrogate\_function=surrogate.ATan()),  
 layer.MaxPool2d(2, 2),  
 )  
 self.classifier = nn.Sequential(  
 layer.Flatten(),  
 layer.Linear(channels\*2\*7\*7, 1024),  
 neuron.IFNode(surrogate\_function=surrogate.ATan()),  
   
 layer.Linear(1024, 10),  
 neuron.IFNode(surrogate\_function=surrogate.ATan()),  
 )  
   
 functional.set\_step\_mode(self, step\_mode='m')  
 if use\_cupy:  
 functional.set\_backend(self, backend='cupy')  
  
 def forward(self, x: torch.Tensor):  
 x\_seq = x.unsqueeze(0).repeat(self.T, 1, 1, 1, 1)  
 features = self.features(x\_seq)  
 output = self.classifier(features)  
 return output.mean(0)

**与ResNet集成的代码实现：**

class ResidualBlock(nn.Module):  
 def \_\_init\_\_(self, in\_channels, out\_channels, stride=1):  
 super().\_\_init\_\_()  
 self.conv1 = layer.Conv2d(in\_channels, out\_channels, kernel\_size=3, stride=stride, padding=1, bias=False)  
 self.bn1 = layer.BatchNorm2d(out\_channels)  
 self.sn1 = neuron.IFNode(surrogate\_function=surrogate.ATan())  
   
 self.conv2 = layer.Conv2d(out\_channels, out\_channels, kernel\_size=3, padding=1, bias=False)  
 self.bn2 = layer.BatchNorm2d(out\_channels)  
 self.sn2 = neuron.IFNode(surrogate\_function=surrogate.ATan())  
   
 self.shortcut = nn.Sequential()  
 if stride != 1 or in\_channels != out\_channels:  
 self.shortcut = nn.Sequential(  
 layer.Conv2d(in\_channels, out\_channels, kernel\_size=1, stride=stride, bias=False),  
 layer.BatchNorm2d(out\_channels)  
 )  
  
 def forward(self, x):  
 residual = self.shortcut(x)  
 out = self.conv1(x)  
 out = self.bn1(out)  
 out = self.sn1(out)  
   
 out = self.conv2(out)  
 out = self.bn2(out)  
 out += residual  
 out = self.sn2(out)  
 return out  
  
class ResNet\_SNN(nn.Module):  
 def \_\_init\_\_(self, T: int, channels: int, num\_blocks=2, use\_cupy=False):  
 super().\_\_init\_\_()  
 self.T = T  
 self.in\_channels = channels  
   
 self.conv1 = nn.Sequential(  
 layer.Conv2d(1, channels, kernel\_size=3, padding=1, bias=False),  
 layer.BatchNorm2d(channels),  
 neuron.IFNode(surrogate\_function=surrogate.ATan())  
 )  
   
 self.layer1 = self.\_make\_layer(channels, num\_blocks, stride=1)  
 self.layer2 = self.\_make\_layer(channels\*2, num\_blocks, stride=2)  
 self.layer3 = self.\_make\_layer(channels\*4, num\_blocks, stride=2)  
   
 self.avgpool = layer.AdaptiveAvgPool2d((1, 1))  
 self.flatten = layer.Flatten()  
 self.fc = layer.Linear(channels\*4, 10)  
 self.snout = neuron.IFNode(surrogate\_function=surrogate.ATan())  
   
 functional.set\_step\_mode(self, step\_mode='m')  
 if use\_cupy:  
 functional.set\_backend(self, backend='cupy')  
   
 def \_make\_layer(self, out\_channels, num\_blocks, stride):  
 layers = []  
 layers.append(ResidualBlock(self.in\_channels, out\_channels, stride))  
 self.in\_channels = out\_channels  
 for \_ in range(1, num\_blocks):  
 layers.append(ResidualBlock(out\_channels, out\_channels, stride=1))  
 return nn.Sequential(\*layers)  
   
 def forward(self, x: torch.Tensor):  
 x\_seq = x.unsqueeze(0).repeat(self.T, 1, 1, 1, 1)  
 out = self.conv1(x\_seq)  
 out = self.layer1(out)  
 out = self.layer2(out)  
 out = self.layer3(out)  
 out = self.avgpool(out)  
 out = self.flatten(out)  
 out = self.fc(out)  
 out = self.snout(out)  
 return out.mean(0)

**性能影响分析：**

将经典的 VGG 或 ResNet 架构与脉冲神经网络结合，可以同时保留深度卷积网络在图像特征提取上的高表达能力和 SNN 在时序信息处理与能耗效率方面的天然优势：首先，VGG 以其层次化、统一的卷积块设计使得 ANN → SNN 的权重映射和阈值校准相对简单，通过逐层转换即可获得可靠的脉冲响应，进而利用脉冲稀疏性大幅降低能量开销；其次，ResNet 所特有的残差连接在 SNN 中同样能够有效缓解跨时间步传播时的梯度消失问题，保证深层网络即使在 Spike‐By‐Spike 的时序传递下也能维持稳定的特征信息流，从而提升收敛速度与最终精度；再者，这两种架构都能直接复用在大规模数据集上预训练得到的强泛化卷积核，在转换为脉冲形式后仍然保持高性能，同时利用 SNN 在专用神经形态硬件上的事件驱动执行特性，可实现低延迟、超低功耗的实时推理，尤其适合边缘端和嵌入式场景。将 VGG 的结构简洁性和 ResNet 的深度可扩展性与 SNN 的时序稀疏计算相结合，既能获得接近或超越传统 ANN 的分类精度，又能显著节省计算资源和电力消耗。具体结果如图6-1至6-3：

**VGG集成优势**：

（1）深层特征提取能力增强

（2）对复杂纹理的识别能力提升

**ResNet集成优势**：

（1）残差连接缓解梯度消失问题

（2）支持更深网络结构（10+层）

（3）训练稳定性提高，收敛速度加快

**训练代码调整**：

# 在训练代码中替换模型  
# net = SNN\_model(tau=args.tau) # 原始模型  
net = VGG\_SNN(T=args.T, channels=32) # VGG集成  
# net = ResNet\_SNN(T=args.T, channels=16) # ResNet集成

**性能对比预期：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模型类型** | **MNIST准确率** | **参数量** | **训练速度** | **适合任务** |
| 原始模型 | 90-92% | 最小 | 最快 | 简单分类 |
| VGG集成SNN | 98-99% | 很大 | 较慢 | 复杂图像 |
| ResNet集成SNN | 98-100% | 很大 | 最慢 | 高精度任务 |

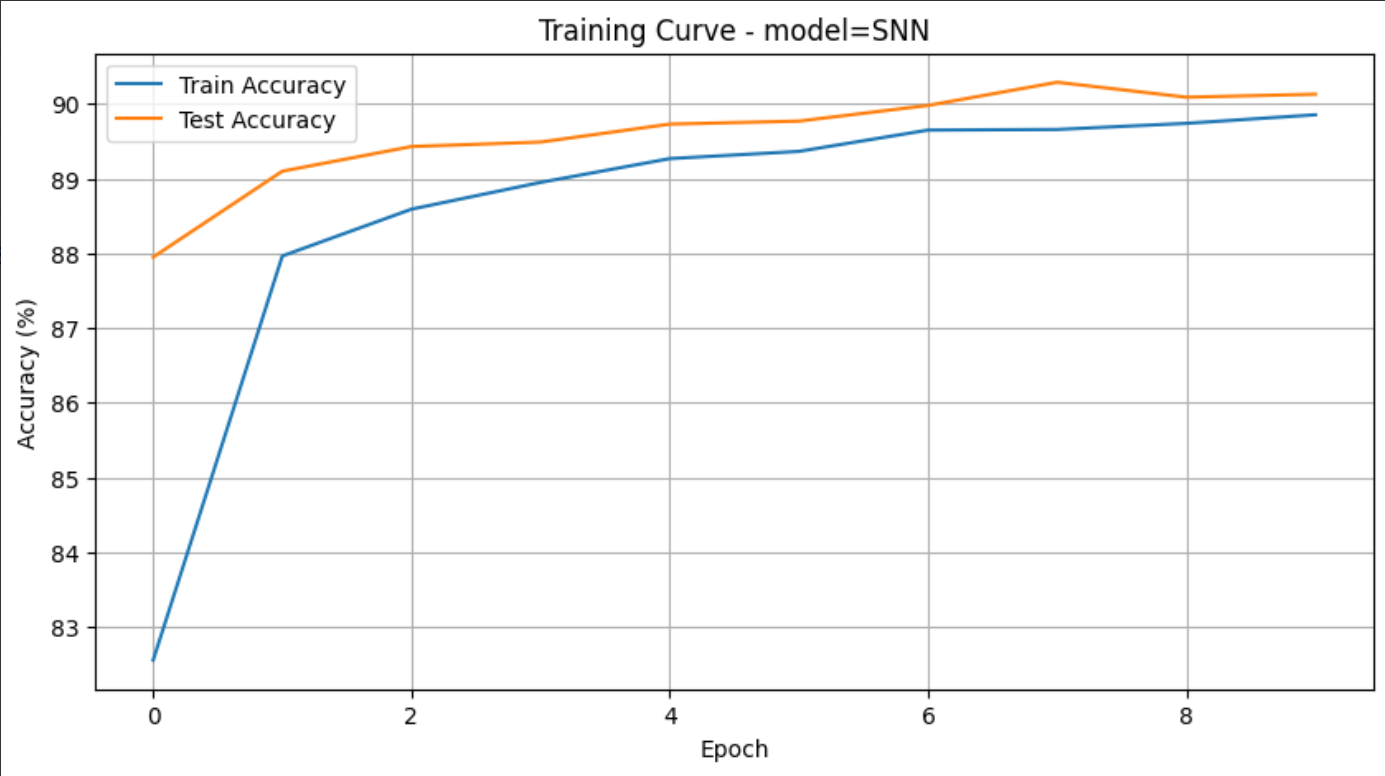


图6-1

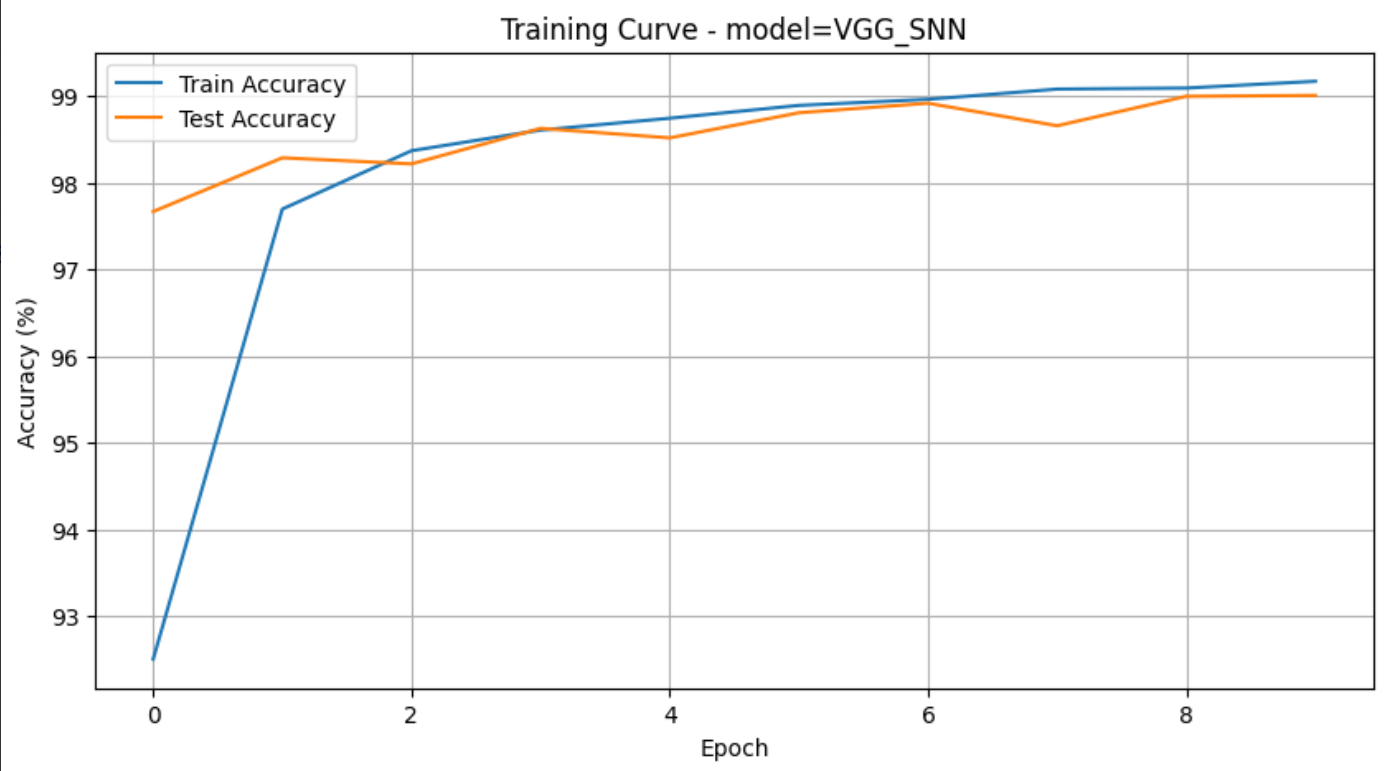


图6-2

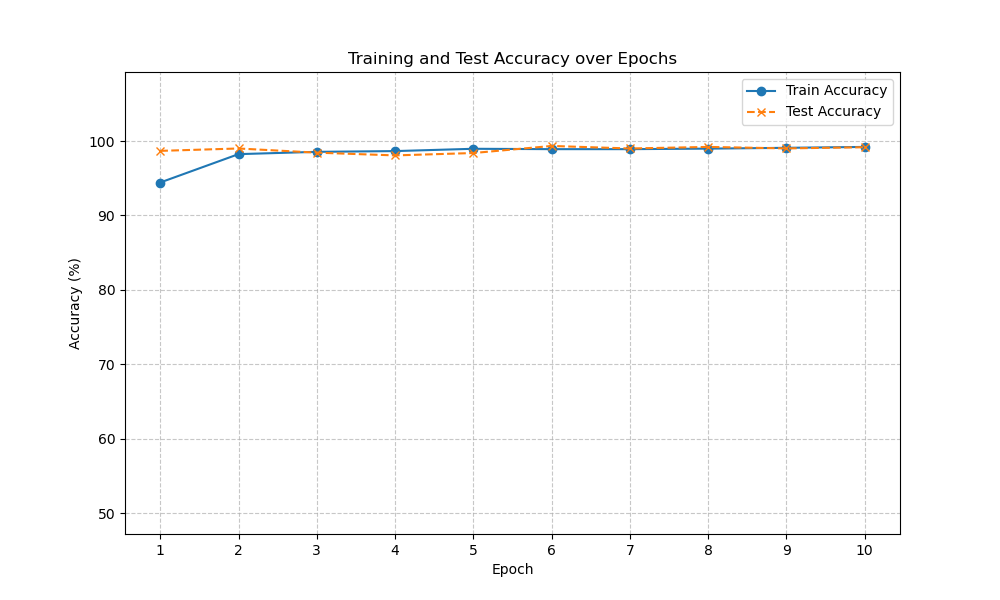


图6-3(突然终止了，手画了一张图，可见下方原训练结果）

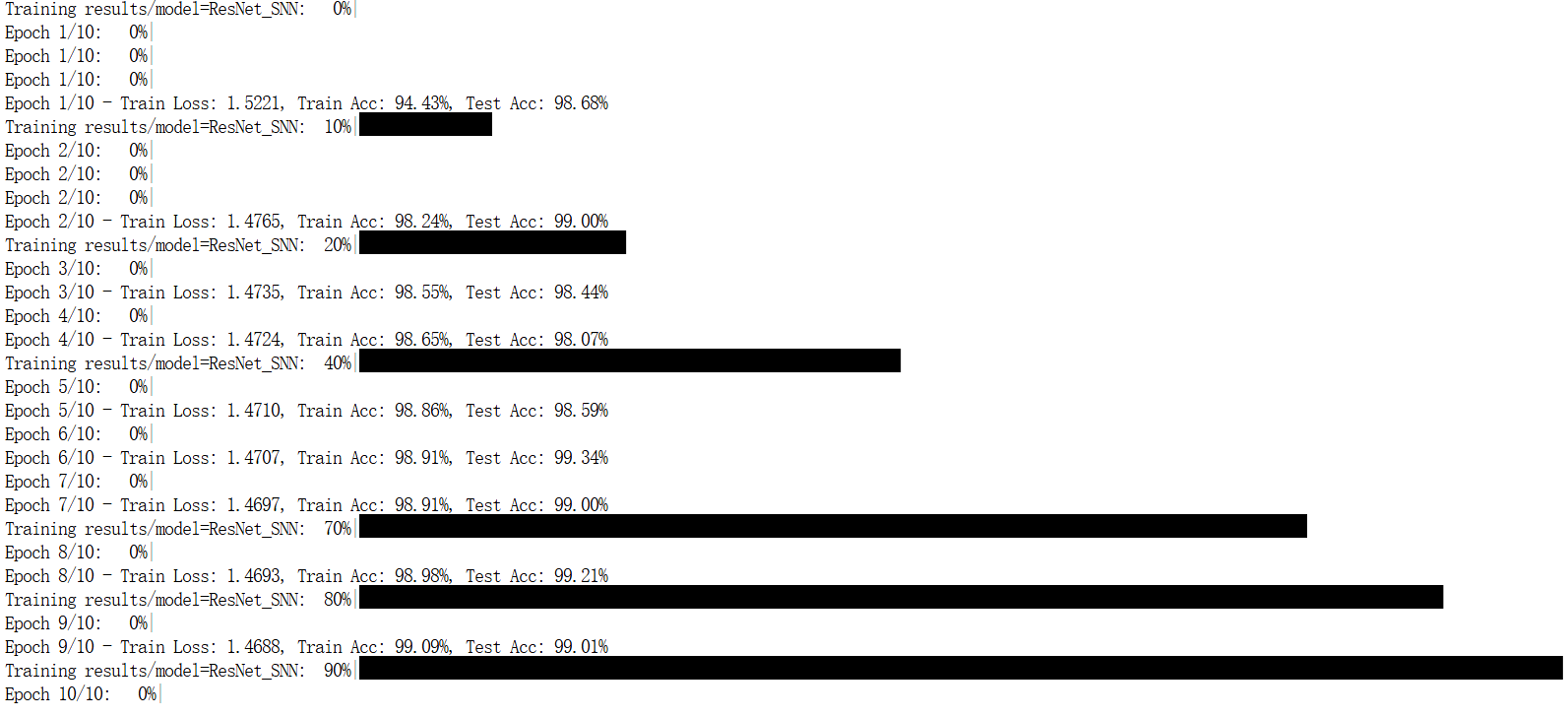


图6-4(原训练结果）

**训练相关日志：**

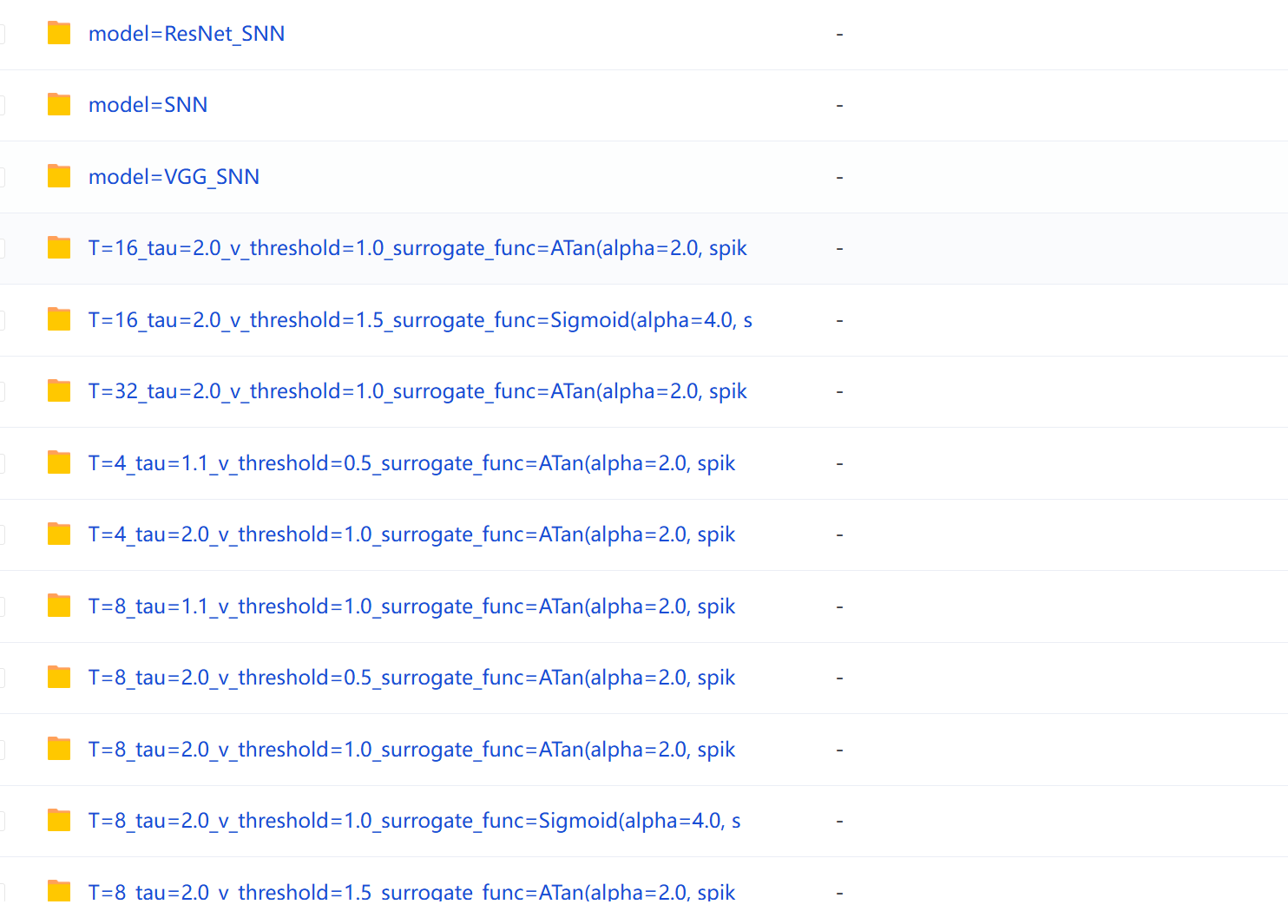


图7 对比实验结果图



图8 对比encoder结果图（因为不同encoder对数据输入要求不太一样，需要单独处理）

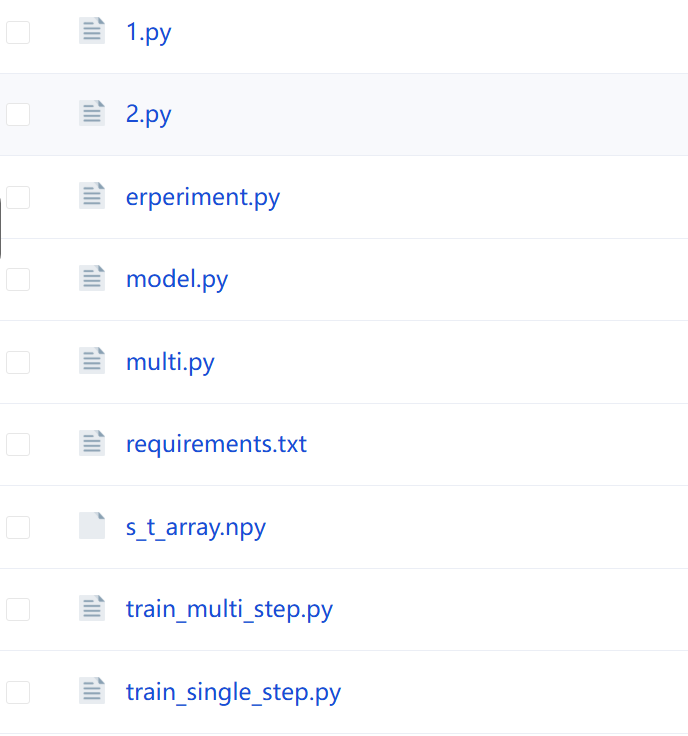


图 9 :相关脚本

以下为训练日志：

CUDA available: True

Current device: NVIDIA A800-SXM4-80GB

==================================================

Starting All Experiments

==================================================

Running SNN Parameter Experiments...

Starting Experiment: T=4\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

==============================

Starting Experiment: results/T=4\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

Model: SNN\_model

Parameters: T=4, tau=2.0, lr=0.001

==============================

Epoch 1/10: 0%| Epoch 1/10 - Train Loss: 1.6319, Train Acc: 81.52%, Test Acc: 86.53% Epoch 2/10: 0%| Epoch 2/10 - Train Loss: 1.5737, Train Acc: 86.95%, Test Acc: 88.12% Epoch 3/10: 0%| Epoch 3/10 - Train Loss: 1.5668, Train Acc: 87.62%, Test Acc: 88.54% Epoch 4/10: 0%| Epoch 5/10: 0%| Epoch 5/10 - Train Loss: 1.5605, Train Acc: 88.29%, Test Acc: 88.78% Epoch 6/10: 0%| Epoch 6/10 - Train Loss: 1.5585, Train Acc: 88.54%, Test Acc: 88.81% Epoch 7/10: 0%| Epoch 7/10 - Train Loss: 1.5577, Train Acc: 88.65%, Test Acc: 89.25% Epoch 8/10: 0%| Epoch 8/10 - Train Loss: 1.5566, Train Acc: 88.78%, Test Acc: 88.98% Epoch 9/10: 0%| Epoch 9/10 - Train Loss: 1.5556, Train Acc: 88.97%, Test Acc: 88.90% | 0/938 [00:07<?, ?batch/s, acc=89, loss=1.59]

Epoch 10/10: 0%| Epoch 10/10 - Train Loss: 1.5547, Train Acc: 89.00%, Test Acc: 89.38% Training results/T=4\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik: 100%|██████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [01:18<00:00, 7.80s/it]

Starting Experiment: T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

==============================

Starting Experiment: results/T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

Model: SNN\_model

Parameters: T=8, tau=2.0, lr=0.001

==============================

Epoch 1/10: 0%| Epoch 1/10 - Train Loss: 1.6332, Train Acc: 82.58%, Test Acc: 87.94% Epoch 2/10: 0%| Epoch 2/10 - Train Loss: 1.5737, Train Acc: 87.78%, Test Acc: 88.94% Epoch 3/10: 0%| Epoch 3/10 - Train Loss: 1.5662, Train Acc: 88.58%, Test Acc: 89.04% Epoch 4/10: 0%| Epoch 4/10 - Train Loss: 1.5624, Train Acc: 88.98%, Test Acc: 89.50% Epoch 5/10: 0%| Epoch 5/10 - Train Loss: 1.5598, Train Acc: 89.15%, Test Acc: 89.65% Epoch 6/10: 0%| Epoch 6/10 - Train Loss: 1.5580, Train Acc: 89.38%, Test Acc: 89.85% Epoch 7/10: 0%| Epoch 7/10 - Train Loss: 1.5569, Train Acc: 89.51%, Test Acc: 90.11% Epoch 8/10: 0%| Epoch 8/10 - Train Loss: 1.5557, Train Acc: 89.68%, Test Acc: 90.02% Epoch 9/10: 0%| Epoch 9/10 - Train Loss: 1.5544, Train Acc: 89.72%, Test Acc: 90.32% Epoch 10/10: 0%| Epoch 10/10 - Train Loss: 1.5537, Train Acc: 89.91%, Test Acc: 90.35% Training results/T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik: 100%|██████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [02:08<00:00, 12.87s/it]

Starting Experiment: T=16\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

==============================

Starting Experiment: results/T=16\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

Model: SNN\_model

Parameters: T=16, tau=2.0, lr=0.001

==============================

Epoch 1/10: 0%| Epoch 1/10 - Train Loss: 1.6308, Train Acc: 83.83%, Test Acc: 88.35% Epoch 2/10: 0%| Epoch 2/10 - Train Loss: 1.5736, Train Acc: 88.35%, Test Acc: 89.42% Epoch 3/10: 0%| Epoch 3/10 - Train Loss: 1.5657, Train Acc: 89.04%, Test Acc: 89.68% Epoch 4/10: 0%| Epoch 4/10 - Train Loss: 1.5620, Train Acc: 89.33%, Test Acc: 90.05% Epoch 5/10: 0%| Epoch 5/10 - Train Loss: 1.5595, Train Acc: 89.68%, Test Acc: 90.29% Epoch 6/10: 0%| Epoch 6/10 - Train Loss: 1.5577, Train Acc: 89.91%, Test Acc: 90.30% Epoch 7/10: 0%| Epoch 7/10 - Train Loss: 1.5563, Train Acc: 89.93%, Test Acc: 90.68% Epoch 8/10: 0%| Epoch 8/10 - Train Loss: 1.5551, Train Acc: 90.20%, Test Acc: 90.68% Epoch 9/10: 0%| Epoch 9/10 - Train Loss: 1.5540, Train Acc: 90.17%, Test Acc: 90.83% Epoch 10/10: 0%| Epoch 10/10 - Train Loss: 1.5531, Train Acc: 90.46%, Test Acc: 90.63% Training results/T=16\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik: 100%|█████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [03:17<00:00, 19.76s/it]

Starting Experiment: T=32\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

==============================

Starting Experiment: results/T=32\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

Model: SNN\_model

Parameters: T=32, tau=2.0, lr=0.001

==============================

Epoch 1/10: 0%| Epoch 1/10 - Train Loss: 1.6332, Train Acc: 84.03%, Test Acc: 88.90% Epoch 2/10: 0%| Epoch 2/10 - Train Loss: 1.5738, Train Acc: 88.63%, Test Acc: 89.64% Epoch 3/10: 0%| Epoch 3/10 - Train Loss: 1.5659, Train Acc: 89.38%, Test Acc: 90.04% Epoch 4/10: 0%| Epoch 4/10 - Train Loss: 1.5619, Train Acc: 89.68%, Test Acc: 90.45% Epoch 5/10: 0%| Epoch 5/10 - Train Loss: 1.5594, Train Acc: 90.07%, Test Acc: 90.40% Epoch 6/10: 0%| Epoch 6/10 - Train Loss: 1.5575, Train Acc: 90.15%, Test Acc: 90.72% Epoch 7/10: 0%| Epoch 7/10 - Train Loss: 1.5559, Train Acc: 90.35%, Test Acc: 90.70% Epoch 8/10: 0%| Epoch 8/10 - Train Loss: 1.5546, Train Acc: 90.44%, Test Acc: 90.69% Epoch 9/10: 0%| Epoch 9/10 - Train Loss: 1.5539, Train Acc: 90.54%, Test Acc: 91.01% Epoch 10/10: 0%| Epoch 10/10 - Train Loss: 1.5529, Train Acc: 90.69%, Test Acc: 91.08% Training results/T=32\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik: 100%|█████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [04:06<00:00, 24.67s/it]

Starting Experiment: T=8\_tau=1.1\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

==============================

Starting Experiment: results/T=8\_tau=1.1\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

Model: SNN\_model

Parameters: T=8, tau=1.1, lr=0.001

Epoch 1/10: 0%| Epoch 1/10 - Train Loss: 1.6073, Train Acc: 83.12%, Test Acc: 88.08% Epoch 2/10: 0%| Epoch 2/10 - Train Loss: 1.5649, Train Acc: 88.26%, Test Acc: 89.33% Epoch 3/10: 0%| Epoch 3/10 - Train Loss: 1.5599, Train Acc: 88.95%, Test Acc: 89.61% Epoch 4/10: 0%| Epoch 4/10 - Train Loss: 1.5575, Train Acc: 89.31%, Test Acc: 89.78% Epoch 5/10: 0%| Epoch 5/10 - Train Loss: 1.5558, Train Acc: 89.58%, Test Acc: 90.20% Epoch 6/10: 0%| Epoch 6/10 - Train Loss: 1.5545, Train Acc: 89.67%, Test Acc: 90.04% Epoch 7/10: 0%| Epoch 7/10 - Train Loss: 1.5535, Train Acc: 89.93%, Test Acc: 90.31% Epoch 8/10: 0%| Epoch 8/10 - Train Loss: 1.5527, Train Acc: 90.07%, Test Acc: 90.21% Epoch 9/10: 0%| Epoch 9/10 - Train Loss: 1.5520, Train Acc: 90.16%, Test Acc: 90.46% Epoch 10/10: 0%| Epoch 10/10 - Train Loss: 1.5518, Train Acc: 90.19%, Test Acc: 90.46% Training results/T=8\_tau=1.1\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik: 100%|██████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [02:32<00:00, 15.26s/it]

Starting Experiment: T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

==============================

Starting Experiment: results/T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

Model: SNN\_model

Parameters: T=8, tau=2.0, lr=0.001

==============================

Epoch 1/10: 0%| Epoch 1/10 - Train Loss: 1.6321, Train Acc: 82.70%, Test Acc: 87.88% Epoch 2/10: 0%| Epoch 2/10 - Train Loss: 1.5732, Train Acc: 87.92%, Test Acc: 89.04% Epoch 3/10: 0%| Epoch 3/10 - Train Loss: 1.5661, Train Acc: 88.56%, Test Acc: 89.53% Epoch 4/10: 0%| Epoch 4/10 - Train Loss: 1.5624, Train Acc: 88.98%, Test Acc: 89.66% Epoch 5/10 - Train Loss: 1.5600, Train Acc: 89.18%, Test Acc: 89.78% Epoch 6/10 - Train Loss: 1.5581, Train Acc: 89.41%, Test Acc: 90.00% Epoch 7/10: 0%| Epoch 7/10 - Train Loss: 1.5564, Train Acc: 89.53%, Test Acc: 89.83% Epoch 8/10: 0%| Epoch 8/10 - Train Loss: 1.5554, Train Acc: 89.71%, Test Acc: 90.17% Epoch 9/10: 0%| Epoch 9/10 - Train Loss: 1.5543, Train Acc: 89.82%, Test Acc: 90.04% Epoch 10/10 - Train Loss: 1.5535, Train Acc: 89.92%, Test Acc: 90.27% Training results/T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik: 100%|██████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [02:46<00:00, 16.66s/it]

Starting Experiment: T=8\_tau=4.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

==============================

Starting Experiment: results/T=8\_tau=4.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

Model: SNN\_model

Parameters: T=8, tau=4.0, lr=0.001

==============================

Epoch 1/10: 0%| | 0/938 [00:08<?, ?batch/s, acc=80.9, loss=1.71]

Epoch 1/10 - Train Loss: 1.6889, Train Acc: 80.91%, Test Acc: 86.66% | 0/938 [00:08<?, ?batch/s, acc=80.9, loss=1.71]

Epoch 2/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=86.9, loss=1.56]

Epoch 2/10 - Train Loss: 1.5917, Train Acc: 86.86%, Test Acc: 88.01% | 0/938 [00:05<?, ?batch/s, acc=86.9, loss=1.56]

Epoch 3/10: 0%| | 0/938 [00:04<?, ?batch/s, acc=87.9, loss=1.6]

Epoch 3/10 - Train Loss: 1.5784, Train Acc: 87.93%, Test Acc: 88.61% | 0/938 [00:04<?, ?batch/s, acc=87.9, loss=1.6]

Epoch 4/10: 0%| | 0/938 [00:06<?, ?batch/s, acc=88.4, loss=1.68]

Epoch 4/10 - Train Loss: 1.5718, Train Acc: 88.37%, Test Acc: 89.28% | 0/938 [00:06<?, ?batch/s, acc=88.4, loss=1.68]

Epoch 5/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=88.7, loss=1.55]

Epoch 5/10 - Train Loss: 1.5678, Train Acc: 88.74%, Test Acc: 89.41% | 0/938 [00:05<?, ?batch/s, acc=88.7, loss=1.55]

Epoch 6/10: 0%| | 0/938 [00:06<?, ?batch/s, acc=89, loss=1.59]

Epoch 6/10 - Train Loss: 1.5651, Train Acc: 88.96%, Test Acc: 89.60% | 0/938 [00:06<?, ?batch/s, acc=89, loss=1.59]

Epoch 7/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=89.1, loss=1.52]

Epoch 7/10 - Train Loss: 1.5629, Train Acc: 89.08%, Test Acc: 89.57% | 0/938 [00:05<?, ?batch/s, acc=89.1, loss=1.52]

Epoch 8/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=89.2, loss=1.6]

Epoch 8/10 - Train Loss: 1.5612, Train Acc: 89.22%, Test Acc: 89.77% | 0/938 [00:05<?, ?batch/s, acc=89.2, loss=1.6]

Epoch 9/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=89.3, loss=1.58]

Epoch 9/10 - Train Loss: 1.5598, Train Acc: 89.33%, Test Acc: 90.05% | 0/938 [00:05<?, ?batch/s, acc=89.3, loss=1.58]

Epoch 10/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=89.5, loss=1.54]

Epoch 10/10 - Train Loss: 1.5585, Train Acc: 89.45%, Test Acc: 89.98% | 0/938 [00:05<?, ?batch/s, acc=89.5, loss=1.54]

Training results/T=8\_tau=4.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik: 100%|██████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [01:06<00:00, 6.66s/it]

Starting Experiment: T=8\_tau=2.0\_v\_threshold=0.5\_surrogate\_func=ATan(alpha=2.0, spik

==============================

Starting Experiment: results/T=8\_tau=2.0\_v\_threshold=0.5\_surrogate\_func=ATan(alpha=2.0, spik

Model: SNN\_model

Parameters: T=8, tau=2.0, lr=0.001

==============================

Epoch 1/10: 0%| | 0/938 [00:14<?, ?batch/s, acc=83.5, loss=1.53]

Epoch 1/10 - Train Loss: 1.6125, Train Acc: 83.46%, Test Acc: 88.31% | 0/938 [00:14<?, ?batch/s, acc=83.5, loss=1.53]

Epoch 2/10: 0%| | 0/938 [00:13<?, ?batch/s, acc=87.8, loss=1.53]

Epoch 2/10 - Train Loss: 1.5698, Train Acc: 87.78%, Test Acc: 88.70% | 0/938 [00:13<?, ?batch/s, acc=87.8, loss=1.53]

Epoch 3/10: 0%| | 0/938 [00:09<?, ?batch/s, acc=88.4, loss=1.55]

Epoch 3/10 - Train Loss: 1.5641, Train Acc: 88.36%, Test Acc: 89.04% | 0/938 [00:09<?, ?batch/s, acc=88.4, loss=1.55]

Epoch 4/10: 0%| | 0/938 [00:19<?, ?batch/s, acc=88.8, loss=1.55]

Epoch 4/10 - Train Loss: 1.5609, Train Acc: 88.84%, Test Acc: 89.34% | 0/938 [00:19<?, ?batch/s, acc=88.8, loss=1.55]

Epoch 5/10: 0%| | 0/938 [00:11<?, ?batch/s, acc=89, loss=1.5]

Epoch 5/10 - Train Loss: 1.5585, Train Acc: 89.02%, Test Acc: 89.66% | 0/938 [00:11<?, ?batch/s, acc=89, loss=1.5]

Epoch 6/10: 0%| | 0/938 [00:12<?, ?batch/s, acc=89.2, loss=1.5]

Epoch 6/10 - Train Loss: 1.5572, Train Acc: 89.17%, Test Acc: 89.89% | 0/938 [00:12<?, ?batch/s, acc=89.2, loss=1.5]

Epoch 7/10: 0%| | 0/938 [00:17<?, ?batch/s, acc=89.3, loss=1.55]

Epoch 7/10 - Train Loss: 1.5558, Train Acc: 89.28%, Test Acc: 89.77% | 0/938 [00:16<?, ?batch/s, acc=89.3, loss=1.55]

Epoch 8/10: 0%| | 0/938 [00:15<?, ?batch/s, acc=89.5, loss=1.58]

Epoch 8/10 - Train Loss: 1.5547, Train Acc: 89.50%, Test Acc: 89.98% | 0/938 [00:15<?, ?batch/s, acc=89.5, loss=1.58]

Epoch 9/10: 0%| | 0/938 [00:08<?, ?batch/s, acc=89.6, loss=1.54]

Epoch 9/10 - Train Loss: 1.5534, Train Acc: 89.61%, Test Acc: 90.48% | 0/938 [00:08<?, ?batch/s, acc=89.6, loss=1.54]

Epoch 10/10: 0%| | 0/938 [00:13<?, ?batch/s, acc=89.7, loss=1.51]

Epoch 10/10 - Train Loss: 1.5531, Train Acc: 89.67%, Test Acc: 90.01% | 0/938 [00:13<?, ?batch/s, acc=89.7, loss=1.51]

Training results/T=8\_tau=2.0\_v\_threshold=0.5\_surrogate\_func=ATan(alpha=2.0, spik: 100%|██████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [02:27<00:00, 14.78s/it]

Starting Experiment: T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

==============================

Starting Experiment: results/T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

Model: SNN\_model

Parameters: T=8, tau=2.0, lr=0.001

==============================

Epoch 1/10: 0%| | 0/938 [00:13<?, ?batch/s, acc=82.6, loss=1.56]

Epoch 1/10 - Train Loss: 1.6327, Train Acc: 82.64%, Test Acc: 88.04% | 0/938 [00:13<?, ?batch/s, acc=82.6, loss=1.56]

Epoch 2/10: 0%| | 0/938 [00:15<?, ?batch/s, acc=88, loss=1.55]

Epoch 2/10 - Train Loss: 1.5733, Train Acc: 88.03%, Test Acc: 88.66% | 0/938 [00:15<?, ?batch/s, acc=88, loss=1.55]

Epoch 3/10: 0%| | 0/938 [00:16<?, ?batch/s, acc=88.6, loss=1.53]

Epoch 3/10 - Train Loss: 1.5660, Train Acc: 88.56%, Test Acc: 89.28% | 0/938 [00:16<?, ?batch/s, acc=88.6, loss=1.53]

Epoch 4/10: 0%| | 0/938 [00:14<?, ?batch/s, acc=89, loss=1.51]

Epoch 4/10 - Train Loss: 1.5624, Train Acc: 89.00%, Test Acc: 89.51% | 0/938 [00:14<?, ?batch/s, acc=89, loss=1.51]

Epoch 5/10: 0%| | 0/938 [00:13<?, ?batch/s, acc=89.2, loss=1.59]

Epoch 5/10 - Train Loss: 1.5599, Train Acc: 89.18%, Test Acc: 89.68% | 0/938 [00:13<?, ?batch/s, acc=89.2, loss=1.59]

Epoch 6/10: 0%| | 0/938 [00:12<?, ?batch/s, acc=89.4, loss=1.54]

Epoch 6/10 - Train Loss: 1.5580, Train Acc: 89.41%, Test Acc: 89.86% | 0/938 [00:12<?, ?batch/s, acc=89.4, loss=1.54]

Epoch 7/10: 0%| | 0/938 [00:16<?, ?batch/s, acc=89.6, loss=1.53]

Epoch 7/10 - Train Loss: 1.5565, Train Acc: 89.62%, Test Acc: 89.86% | 0/938 [00:16<?, ?batch/s, acc=89.6, loss=1.53]

Epoch 8/10: 0%| | 0/938 [00:12<?, ?batch/s, acc=89.7, loss=1.57]

Epoch 8/10 - Train Loss: 1.5558, Train Acc: 89.65%, Test Acc: 90.06% | 0/938 [00:12<?, ?batch/s, acc=89.7, loss=1.57]

Epoch 9/10: 0%| | 0/938 [00:14<?, ?batch/s, acc=89.8, loss=1.53]

Epoch 9/10 - Train Loss: 1.5545, Train Acc: 89.76%, Test Acc: 90.01% | 0/938 [00:14<?, ?batch/s, acc=89.8, loss=1.53]

Epoch 10/10: 0%| | 0/938 [00:15<?, ?batch/s, acc=89.9, loss=1.65]

Epoch 10/10 - Train Loss: 1.5539, Train Acc: 89.88%, Test Acc: 90.09% | 0/938 [00:15<?, ?batch/s, acc=89.9, loss=1.65]

Training results/T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik: 100%|██████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [02:34<00:00, 15.41s/it]

Starting Experiment: T=8\_tau=2.0\_v\_threshold=1.5\_surrogate\_func=ATan(alpha=2.0, spik

==============================

Starting Experiment: results/T=8\_tau=2.0\_v\_threshold=1.5\_surrogate\_func=ATan(alpha=2.0, spik

Model: SNN\_model

Parameters: T=8, tau=2.0, lr=0.001

==============================

Epoch 1/10: 0%| | 0/938 [00:15<?, ?batch/s, acc=81.6, loss=1.55]

Epoch 1/10 - Train Loss: 1.6522, Train Acc: 81.61%, Test Acc: 87.49% | 0/938 [00:15<?, ?batch/s, acc=81.6, loss=1.55]

Training results/T=8\_tau=2.0\_v\_threshold=1.5\_surrogate\_func=ATan(alpha=2.0, spik: 10%|██████████▋ | 1/10 [00:16<02:32, 16.97s/it]

Epoch 2/10: 0%| | 0/938 [00:17<?, ?batch/s, acc=87.7, loss=1.58]

Epoch 2/10 - Train Loss: 1.5786, Train Acc: 87.69%, Test Acc: 88.90% | 0/938 [00:17<?, ?batch/s, acc=87.7, loss=1.58]

Epoch 3/10: 0%| | 0/938 [00:22<?, ?batch/s, acc=88.7, loss=1.56]

Epoch 3/10 - Train Loss: 1.5692, Train Acc: 88.67%, Test Acc: 89.57% | 0/938 [00:22<?, ?batch/s, acc=88.7, loss=1.56]

Epoch 4/10: 0%| | 0/938 [00:06<?, ?batch/s, acc=89, loss=1.56]

Epoch 4/10 - Train Loss: 1.5648, Train Acc: 89.03%, Test Acc: 89.51% | 0/938 [00:06<?, ?batch/s, acc=89, loss=1.56]

Epoch 5/10: 0%| | 0/938 [00:06<?, ?batch/s, acc=89.3, loss=1.55]

Epoch 5/10 - Train Loss: 1.5617, Train Acc: 89.30%, Test Acc: 89.82% | 0/938 [00:06<?, ?batch/s, acc=89.3, loss=1.55]

Epoch 6/10: 0%| | 0/938 [00:00<?, ?batch/s, acc=89.5, loss=1.57]

Epoch 6/10 - Train Loss: 1.5595, Train Acc: 89.50%, Test Acc: 89.92% | 0/938 [00:00<?, ?batch/s, acc=89.5, loss=1.57]

Epoch 7/10: 0%| | 0/938 [00:08<?, ?batch/s, acc=89.7, loss=1.52]

Epoch 7/10 - Train Loss: 1.5581, Train Acc: 89.65%, Test Acc: 90.21% | 0/938 [00:08<?, ?batch/s, acc=89.7, loss=1.52]

Epoch 8/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=89.8, loss=1.53]

Epoch 8/10 - Train Loss: 1.5565, Train Acc: 89.83%, Test Acc: 90.32% | 0/938 [00:05<?, ?batch/s, acc=89.8, loss=1.53]

Epoch 9/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=89.9, loss=1.57]

Epoch 9/10 - Train Loss: 1.5556, Train Acc: 89.93%, Test Acc: 90.02% | 0/938 [00:05<?, ?batch/s, acc=89.9, loss=1.57]

Epoch 10/10: 0%| | 0/938 [00:06<?, ?batch/s, acc=89.9, loss=1.55]

Epoch 10/10 - Train Loss: 1.5549, Train Acc: 89.91%, Test Acc: 90.16% | 0/938 [00:06<?, ?batch/s, acc=89.9, loss=1.55]

Training results/T=8\_tau=2.0\_v\_threshold=1.5\_surrogate\_func=ATan(alpha=2.0, spik: 100%|██████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [01:44<00:00, 10.42s/it]

Starting Experiment: T=8\_tau=2.0\_v\_threshold=2.0\_surrogate\_func=ATan(alpha=2.0, spik

==============================

Starting Experiment: results/T=8\_tau=2.0\_v\_threshold=2.0\_surrogate\_func=ATan(alpha=2.0, spik

Model: SNN\_model

Parameters: T=8, tau=2.0, lr=0.001

==============================

Epoch 1/10: 0%| | 0/938 [00:11<?, ?batch/s, acc=80.9, loss=1.58]

Epoch 1/10 - Train Loss: 1.6688, Train Acc: 80.91%, Test Acc: 87.16% | 0/938 [00:11<?, ?batch/s, acc=80.9, loss=1.58]

Epoch 2/10: 0%| | 0/938 [00:09<?, ?batch/s, acc=87.6, loss=1.57]

Epoch 2/10 - Train Loss: 1.5842, Train Acc: 87.62%, Test Acc: 88.91% | 0/938 [00:09<?, ?batch/s, acc=87.6, loss=1.57]

Epoch 3/10: 0%| | 0/938 [00:13<?, ?batch/s, acc=88.5, loss=1.57]

Epoch 3/10 - Train Loss: 1.5730, Train Acc: 88.49%, Test Acc: 89.36% | 0/938 [00:13<?, ?batch/s, acc=88.5, loss=1.57]

Epoch 4/10: 0%| | 0/938 [00:17<?, ?batch/s, acc=89, loss=1.58]

Epoch 4/10 - Train Loss: 1.5677, Train Acc: 88.98%, Test Acc: 89.47% | 0/938 [00:17<?, ?batch/s, acc=89, loss=1.58]

Epoch 5/10: 0%| | 0/938 [00:13<?, ?batch/s, acc=89.4, loss=1.58]

Epoch 5/10 - Train Loss: 1.5642, Train Acc: 89.36%, Test Acc: 89.89% | 0/938 [00:13<?, ?batch/s, acc=89.4, loss=1.58]

Epoch 6/10: 0%| | 0/938 [00:15<?, ?batch/s, acc=89.5, loss=1.56]

Epoch 6/10 - Train Loss: 1.5615, Train Acc: 89.50%, Test Acc: 89.91% | 0/938 [00:15<?, ?batch/s, acc=89.5, loss=1.56]

Epoch 7/10: 0%| | 0/938 [00:09<?, ?batch/s, acc=89.7, loss=1.51]

Epoch 7/10 - Train Loss: 1.5597, Train Acc: 89.69%, Test Acc: 90.09% | 0/938 [00:09<?, ?batch/s, acc=89.7, loss=1.51]

Epoch 8/10: 0%| | 0/938 [00:14<?, ?batch/s, acc=89.8, loss=1.58]

Epoch 8/10 - Train Loss: 1.5582, Train Acc: 89.75%, Test Acc: 90.12% | 0/938 [00:14<?, ?batch/s, acc=89.8, loss=1.58]

Epoch 9/10: 0%| | 0/938 [00:15<?, ?batch/s, acc=89.9, loss=1.63]

Epoch 9/10 - Train Loss: 1.5571, Train Acc: 89.93%, Test Acc: 90.30% | 0/938 [00:15<?, ?batch/s, acc=89.9, loss=1.63]

Epoch 10/10: 0%| | 0/938 [00:14<?, ?batch/s, acc=90, loss=1.54]

Epoch 10/10 - Train Loss: 1.5562, Train Acc: 89.98%, Test Acc: 90.35% | 0/938 [00:14<?, ?batch/s, acc=90, loss=1.54]

Training results/T=8\_tau=2.0\_v\_threshold=2.0\_surrogate\_func=ATan(alpha=2.0, spik: 100%|██████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [02:30<00:00, 15.03s/it]

Starting Experiment: T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

==============================

Starting Experiment: results/T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik

Model: SNN\_model

Parameters: T=8, tau=2.0, lr=0.001

==============================

Epoch 1/10: 0%| | 0/938 [00:10<?, ?batch/s, acc=82.8, loss=1.52]

Epoch 1/10 - Train Loss: 1.6333, Train Acc: 82.76%, Test Acc: 87.96% | 0/938 [00:10<?, ?batch/s, acc=82.8, loss=1.52]

Epoch 2/10: 0%| | 0/938 [00:14<?, ?batch/s, acc=87.7, loss=1.59]

Epoch 2/10 - Train Loss: 1.5742, Train Acc: 87.75%, Test Acc: 88.76% | 0/938 [00:14<?, ?batch/s, acc=87.7, loss=1.59]

Epoch 3/10: 0%| | 0/938 [00:14<?, ?batch/s, acc=88.5, loss=1.6]

Epoch 3/10 - Train Loss: 1.5665, Train Acc: 88.50%, Test Acc: 89.22% | 0/938 [00:14<?, ?batch/s, acc=88.5, loss=1.6]

Epoch 4/10: 0%| | 0/938 [00:14<?, ?batch/s, acc=88.9, loss=1.61]

Epoch 4/10 - Train Loss: 1.5625, Train Acc: 88.87%, Test Acc: 89.61% | 0/938 [00:14<?, ?batch/s, acc=88.9, loss=1.61]

Epoch 5/10: 0%| | 0/938 [00:06<?, ?batch/s, acc=89.2, loss=1.51]

Epoch 5/10 - Train Loss: 1.5601, Train Acc: 89.19%, Test Acc: 89.87% | 0/938 [00:06<?, ?batch/s, acc=89.2, loss=1.51]

Epoch 6/10: 0%| | 0/938 [00:15<?, ?batch/s, acc=89.3, loss=1.54]

Epoch 6/10 - Train Loss: 1.5581, Train Acc: 89.28%, Test Acc: 89.97% | 0/938 [00:15<?, ?batch/s, acc=89.3, loss=1.54]

Epoch 7/10: 0%| | 0/938 [00:15<?, ?batch/s, acc=89.5, loss=1.51]

Epoch 7/10 - Train Loss: 1.5565, Train Acc: 89.54%, Test Acc: 90.13% | 0/938 [00:15<?, ?batch/s, acc=89.5, loss=1.51]

Epoch 8/10: 0%| | 0/938 [00:14<?, ?batch/s, acc=89.6, loss=1.64]

Epoch 8/10 - Train Loss: 1.5558, Train Acc: 89.63%, Test Acc: 90.14% | 0/938 [00:14<?, ?batch/s, acc=89.6, loss=1.64]

Epoch 9/10: 0%| | 0/938 [00:14<?, ?batch/s, acc=89.9, loss=1.61]

Epoch 9/10 - Train Loss: 1.5546, Train Acc: 89.85%, Test Acc: 90.19% | 0/938 [00:14<?, ?batch/s, acc=89.9, loss=1.61]

Epoch 10/10: 0%| | 0/938 [00:08<?, ?batch/s, acc=89.9, loss=1.6]

Epoch 10/10 - Train Loss: 1.5536, Train Acc: 89.88%, Test Acc: 90.26% | 0/938 [00:08<?, ?batch/s, acc=89.9, loss=1.6]

Training results/T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=ATan(alpha=2.0, spik: 100%|██████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [02:18<00:00, 13.89s/it]

Starting Experiment: T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=Sigmoid(alpha=4.0, s

==============================

Starting Experiment: results/T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=Sigmoid(alpha=4.0, s

Model: SNN\_model

Parameters: T=8, tau=2.0, lr=0.001

==============================

Epoch 1/10: 0%| | 0/938 [00:23<?, ?batch/s, acc=83.2, loss=1.52]

Epoch 1/10 - Train Loss: 1.6318, Train Acc: 83.22%, Test Acc: 88.63% | 0/938 [00:23<?, ?batch/s, acc=83.2, loss=1.52]

Epoch 2/10: 0%| | 0/938 [00:19<?, ?batch/s, acc=88.5, loss=1.56]

Epoch 2/10 - Train Loss: 1.5720, Train Acc: 88.54%, Test Acc: 89.60% | 0/938 [00:19<?, ?batch/s, acc=88.5, loss=1.56]

Epoch 3/10: 0%| | 0/938 [00:07<?, ?batch/s, acc=89.4, loss=1.56]

Epoch 3/10 - Train Loss: 1.5643, Train Acc: 89.36%, Test Acc: 90.03% | 0/938 [00:07<?, ?batch/s, acc=89.4, loss=1.56]

Epoch 4/10: 0%| | 0/938 [00:13<?, ?batch/s, acc=89.7, loss=1.57]

Epoch 4/10 - Train Loss: 1.5605, Train Acc: 89.72%, Test Acc: 90.20% | 0/938 [00:13<?, ?batch/s, acc=89.7, loss=1.57]

Epoch 5/10: 0%| | 0/938 [00:16<?, ?batch/s, acc=90, loss=1.57]

Epoch 5/10 - Train Loss: 1.5576, Train Acc: 89.99%, Test Acc: 90.48% | 0/938 [00:16<?, ?batch/s, acc=90, loss=1.57]

Training results/T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=Sigmoid(alpha=4.0, s: 50%|█████████████████████████████████████████████████████▌ | 5/10 [01:28<01:24, 17.00s/it]

Epoch 6/10: 0%| | 0/938 [00:21<?, ?batch/s, acc=90.2, loss=1.52]

Epoch 6/10 - Train Loss: 1.5558, Train Acc: 90.25%, Test Acc: 90.49% | 0/938 [00:21<?, ?batch/s, acc=90.2, loss=1.52]

Epoch 7/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=90.4, loss=1.54]

Epoch 7/10 - Train Loss: 1.5545, Train Acc: 90.38%, Test Acc: 90.62% | 0/938 [00:05<?, ?batch/s, acc=90.4, loss=1.54]

Epoch 8/10: 0%| | 0/938 [00:21<?, ?batch/s, acc=90.5, loss=1.56]

Epoch 8/10 - Train Loss: 1.5534, Train Acc: 90.46%, Test Acc: 90.76% | 0/938 [00:21<?, ?batch/s, acc=90.5, loss=1.56]

Epoch 9/10: 0%| | 0/938 [00:13<?, ?batch/s, acc=90.6, loss=1.52]

Epoch 9/10 - Train Loss: 1.5525, Train Acc: 90.61%, Test Acc: 91.00% | 0/938 [00:13<?, ?batch/s, acc=90.6, loss=1.52]

Epoch 10/10: 0%| | 0/938 [00:15<?, ?batch/s, acc=90.6, loss=1.56]

Epoch 10/10 - Train Loss: 1.5516, Train Acc: 90.62%, Test Acc: 90.71% | 0/938 [00:15<?, ?batch/s, acc=90.6, loss=1.56]

Training results/T=8\_tau=2.0\_v\_threshold=1.0\_surrogate\_func=Sigmoid(alpha=4.0, s: 100%|██████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [02:51<00:00, 17.11s/it]

Running Architecture Experiments...

Starting Experiment: model=SNN

==============================

Starting Experiment: results/model=SNN

Model: SNN\_model

Parameters: T=8, tau=2.0, lr=0.001

==============================

Epoch 1/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=82.6, loss=1.6]

Epoch 1/10 - Train Loss: 1.6324, Train Acc: 82.56%, Test Acc: 87.95% | 0/938 [00:05<?, ?batch/s, acc=82.6, loss=1.6]

Epoch 2/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=88, loss=1.55]

Epoch 2/10 - Train Loss: 1.5732, Train Acc: 87.97%, Test Acc: 89.10% | 0/938 [00:05<?, ?batch/s, acc=88, loss=1.55]

Epoch 3/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=88.6, loss=1.57]

Epoch 3/10 - Train Loss: 1.5663, Train Acc: 88.59%, Test Acc: 89.43% | 0/938 [00:05<?, ?batch/s, acc=88.6, loss=1.57]

Epoch 4/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=89, loss=1.53]

Epoch 4/10 - Train Loss: 1.5623, Train Acc: 88.95%, Test Acc: 89.49% | 0/938 [00:05<?, ?batch/s, acc=89, loss=1.53]

Epoch 5/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=89.3, loss=1.54]

Epoch 5/10 - Train Loss: 1.5596, Train Acc: 89.27%, Test Acc: 89.73% | 0/938 [00:05<?, ?batch/s, acc=89.3, loss=1.54]

Epoch 6/10: 0%| | 0/938 [00:06<?, ?batch/s, acc=89.4, loss=1.54]

Epoch 6/10 - Train Loss: 1.5580, Train Acc: 89.36%, Test Acc: 89.77% | 0/938 [00:06<?, ?batch/s, acc=89.4, loss=1.54]

Epoch 7/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=89.7, loss=1.51]

Epoch 7/10 - Train Loss: 1.5565, Train Acc: 89.65%, Test Acc: 89.98% | 0/938 [00:05<?, ?batch/s, acc=89.7, loss=1.51]

Epoch 8/10: 0%| | 0/938 [00:05<?, ?batch/s, acc=89.7, loss=1.54]

Epoch 8/10 - Train Loss: 1.5557, Train Acc: 89.66%, Test Acc: 90.29% | 0/938 [00:05<?, ?batch/s, acc=89.7, loss=1.54]

Epoch 9/10: 0%| | 0/938 [00:10<?, ?batch/s, acc=89.7, loss=1.64]

Epoch 9/10 - Train Loss: 1.5545, Train Acc: 89.74%, Test Acc: 90.09% | 0/938 [00:10<?, ?batch/s, acc=89.7, loss=1.64]

Epoch 10/10: 0%| | 0/938 [00:01<?, ?batch/s, acc=89.9, loss=1.56]

Epoch 10/10 - Train Loss: 1.5537, Train Acc: 89.86%, Test Acc: 90.13% | 0/938 [00:01<?, ?batch/s, acc=89.9, loss=1.56]

Training results/model=SNN: 100%|████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [01:05<00:00, 6.51s/it]

Starting Experiment: model=VGG\_SNN

==============================

Starting Experiment: results/model=VGG\_SNN

Model: VGG\_SNN

Parameters: T=8, tau=2.0, lr=0.001

==============================

Epoch 1/10: 0%| | 0/938 [02:25<?, ?batch/s, acc=92.5, loss=1.49]

Epoch 1/10 - Train Loss: 1.5343, Train Acc: 92.50%, Test Acc: 97.67% | 0/938 [02:25<?, ?batch/s, acc=92.5, loss=1.49]

Training results/model=VGG\_SNN: 10%|███████████████▌ | 1/10 [02:34<23:08, 154.27s/it]

Epoch 2/10: 0%| | 0/938 [01:54<?, ?batch/s, acc=97.7, loss=1.47]

Epoch 2/10 - Train Loss: 1.4823, Train Acc: 97.70%, Test Acc: 98.29% | 0/938 [01:54<?, ?batch/s, acc=97.7, loss=1.47]

Training results/model=VGG\_SNN: 20%|███████████████████████████████▏ | 2/10 [04:32<17:46, 133.25s/it]

Epoch 3/10: 0%| | 0/938 [01:57<?, ?batch/s, acc=98.4, loss=1.47]

Epoch 3/10 - Train Loss: 1.4767, Train Acc: 98.37%, Test Acc: 98.22% | 0/938 [01:57<?, ?batch/s, acc=98.4, loss=1.47]

Epoch 4/10: 0%| | 0/938 [02:02<?, ?batch/s, acc=98.6, loss=1.48]

Epoch 4/10 - Train Loss: 1.4743, Train Acc: 98.61%, Test Acc: 98.63% | 0/938 [02:02<?, ?batch/s, acc=98.6, loss=1.48]

Epoch 5/10: 0%| | 0/938 [01:57<?, ?batch/s, acc=98.7, loss=1.47]

Epoch 5/10 - Train Loss: 1.4728, Train Acc: 98.75%, Test Acc: 98.52% | 0/938 [01:57<?, ?batch/s, acc=98.7, loss=1.47]

Epoch 6/10: 0%| | 0/938 [01:41<?, ?batch/s, acc=98.9, loss=1.46]

Epoch 6/10 - Train Loss: 1.4713, Train Acc: 98.89%, Test Acc: 98.81% | 0/938 [01:41<?, ?batch/s, acc=98.9, loss=1.46]

Epoch 7/10: 0%| | 0/938 [01:34<?, ?batch/s, acc=99, loss=1.47]

Epoch 7/10 - Train Loss: 1.4709, Train Acc: 98.97%, Test Acc: 98.92% | 0/938 [01:34<?, ?batch/s, acc=99, loss=1.47]

Epoch 8/10: 0%| | 0/938 [01:39<?, ?batch/s, acc=99.1, loss=1.48]

Epoch 8/10 - Train Loss: 1.4695, Train Acc: 99.08%, Test Acc: 98.66% | 0/938 [01:38<?, ?batch/s, acc=99.1, loss=1.48]

Training results/model=VGG\_SNN: 80%|████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████▊ | 8/10 [15:49<03:36, 108.37s/it]

Epoch 9/10: 0%| | 0/938 [01:42<?, ?batch/s, acc=99.1, loss=1.46]

Epoch 9/10 - Train Loss: 1.4694, Train Acc: 99.09%, Test Acc: 99.00% | 0/938 [01:42<?, ?batch/s, acc=99.1, loss=1.46]

Epoch 10/10: 0%| | 0/938 [01:40<?, ?batch/s, acc=99.2, loss=1.46]

Epoch 10/10 - Train Loss: 1.4687, Train Acc: 99.17%, Test Acc: 99.01% | 0/938 [01:40<?, ?batch/s, acc=99.2, loss=1.46]

Training results/model=VGG\_SNN: 100%|███████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████| 10/10 [19:20<00:00, 116.08s/it]

Starting Experiment: model=ResNet\_SNN

==============================

Starting Experiment: results/model=ResNet\_SNN

Model: ResNet\_SNN

Parameters: T=8, tau=2.0, lr=0.001

==============================

Training results/model=ResNet\_SNN: 0%| | 0/10 [00:00<?, ?it/s]

Epoch 1/10: 0%| | 0/938 [04:00<?, ?batch/s, acc=92, loss=1.48]

Epoch 1/10: 0%| | 0/938 [08:24<?, ?batch/s, acc=94.4, loss=1.46]

Epoch 1/10: 0%| | 0/938 [08:24<?, ?batch/s, acc=94.4, loss=1.46]

Epoch 1/10 - Train Loss: 1.5221, Train Acc: 94.43%, Test Acc: 98.68%

Training results/model=ResNet\_SNN: 10%|███████████████ | 1/10 [08:47<1:19:10, 527.86s/it]

Epoch 2/10: 0%| | 0/938 [00:31<?, ?batch/s, acc=98.3, loss=1.47]

Epoch 2/10: 0%| | 0/938 [07:35<?, ?batch/s, acc=98.2, loss=1.47]

Epoch 2/10: 0%| | 0/938 [07:35<?, ?batch/s, acc=98.2, loss=1.47]

Epoch 2/10 - Train Loss: 1.4765, Train Acc: 98.24%, Test Acc: 99.00%

Training results/model=ResNet\_SNN: 20%|██████████████████████████████▏ | 2/10 [16:35<1:05:39, 492.40s/it]

Epoch 3/10: 0%| | 0/938 [08:28<?, ?batch/s, acc=98.5, loss=1.47]

Epoch 3/10 - Train Loss: 1.4735, Train Acc: 98.55%, Test Acc: 98.44% | 0/938 [08:28<?, ?batch/s, acc=98.5, loss=1.47]

Epoch 4/10: 0%| | 0/938 [06:06<?, ?batch/s, acc=98.7, loss=1.48]

Epoch 4/10 - Train Loss: 1.4724, Train Acc: 98.65%, Test Acc: 98.07% | 0/938 [06:06<?, ?batch/s, acc=98.7, loss=1.48]

Training results/model=ResNet\_SNN: 40%|█████████████████████████████████████████████████████████████▏ | 4/10 [31:32<45:27, 454.58s/it]

Epoch 5/10: 0%| | 0/938 [06:06<?, ?batch/s, acc=98.9, loss=1.46]

Epoch 5/10 - Train Loss: 1.4710, Train Acc: 98.86%, Test Acc: 98.59% | 0/938 [06:06<?, ?batch/s, acc=98.9, loss=1.46]

Epoch 6/10: 0%| | 0/938 [05:56<?, ?batch/s, acc=98.9, loss=1.47]

Epoch 6/10 - Train Loss: 1.4707, Train Acc: 98.91%, Test Acc: 99.34% | 0/938 [05:56<?, ?batch/s, acc=98.9, loss=1.47]

Epoch 7/10: 0%| | 0/938 [06:31<?, ?batch/s, acc=98.9, loss=1.47]

Epoch 7/10 - Train Loss: 1.4697, Train Acc: 98.91%, Test Acc: 99.00% | 0/938 [06:31<?, ?batch/s, acc=98.9, loss=1.47]

Training results/model=ResNet\_SNN: 70%|███████████████████████████████████████████████████████████████████████████████████████████████████████████ | 7/10 [50:38<20:14, 404.90s/it]

Epoch 8/10: 0%| | 0/938 [06:16<?, ?batch/s, acc=99, loss=1.5]

Epoch 8/10 - Train Loss: 1.4693, Train Acc: 98.98%, Test Acc: 99.21% | 0/938 [06:16<?, ?batch/s, acc=99, loss=1.5]

Training results/model=ResNet\_SNN: 80%|██████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████▍ | 8/10 [57:06<13:19, 399.64s/it]

Epoch 9/10: 0%| | 0/938 [06:04<?, ?batch/s, acc=99.1, loss=1.47]

Epoch 9/10 - Train Loss: 1.4688, Train Acc: 99.09%, Test Acc: 99.01% | 0/938 [06:04<?, ?batch/s, acc=99.1, loss=1.47]

Training results/model=ResNet\_SNN: 90%|███████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████▉ | 9/10 [1:03:15<06:30, 390.14s/it]

Epoch 10/10: 0%| | 0/938 [00:44<?, ?batch/s, acc=99.3, loss=1.46]