类脑智能

520030910393 马逸川

3.0 目录结构

`main.py` 定义训练中的各项参数，是训练的主函数。

`Train_funcs.py` 分别定义SNN和CNN的训练函数。

`models.py` 定义SNN,CNN模型。

3.1 模型描述

使用`Spikingjelly`工具包提供的函数搭建模型。下面将具体从神经元、拓扑结构、输入输出和所使用的学习方法四个方面介绍模型的结构。

3.1.1 神经元

使用`spikingjelly.activation_based`中提供的`neuron.IFNode()`作为脉冲神经元。其中，定义神经元的参数为: `v_threshold=1.0, v_reset=0.0`。

参考文档，脉冲神经元的充放电、重置两类方程分别如下:

$$H[t] = f(V[t-1], X[t])$$
$$S[t] = \Theta(H[t] - V_{threshold})$$

Hard方式重置方程为:

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

Soft方式重置方程为:

$$V[t] = H[t] - V_{threshold} \cdot S[t]$$

由于定义了`v_reset=0`，故实际中使用的是`hard`重置方式。

在文档中，神经元重置方式的实现如下:

```
def neuronal_reset(self):
    if self.v_reset is None:
        self.v = self.v - self.spike * self.v_threshold
    else:
        self.v = (1. - self.spike) * self.v + self.spike * self.v_reset
```

在实际的网络结构中，在每一层卷积层后加上`neuron.IFNode()`层。

3.1.2 拓扑结构

参考网络中主要特征的`self.conv`层说明，代码如下：

```
        self.conv = nn.Sequential(
            nn.Conv2d(2, 8, kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(8),
            neuron.IFNode(v_threshold=v_threshold, v_reset=v_reset,
surrogate_function=surrogate.ATan()),
            nn.MaxPool2d(2, 2)  # 7 * 7
        )
```

模型的拓扑结构:

首先图像输入`torch.nn`的卷积层，经过卷积操作后，以此步输出的张量作为输入，经过`neuron.IFNode`的脉冲神经元。神经元的方程见3.1.1，脉冲神经元按照以上的方程处理输入张量的数据，输出相同大小的张量，再经过flatten操作得到向量，经过线性前馈网络输出分类结果。

3.1.3 输入输出

输入:使用的数据集是`Cifar10-DVS`数据集，处理为`frames_number=8`的数据，`batch_size=32`，通过`Dataloader`批量处理。

每一批的输入张量维度是: (32, 8, 2, 32, 32)

为了使用frame维度的数据，将输入作transpose操作处理为(8,32,2,32,32)维度的操作。

输出：通过网络处理后得到10分类数据，维度是:(32,10)

3.1.4 使用的学习方法

使用的optimizer是`torch.optim.SGD`， 也即最基础的随机梯度下降， 每次用输入的一个batch的数据对模型进行梯度更新。它的更新函数如下所示:

$$\theta = \theta - \alpha * \frac{dJ(\theta)}{d\theta}$$

这种优化方法的更新流程如下:

1.初始化参数（θ，学习率α）

2.计算每个θ处的梯度

3.更新参数

4.重复步骤2 和3，直到代价值稳定

这种方法能够在小批量数据上计算损失函数的梯度，从而迭代地更新权重。

3.2 实验结果

训练参数如下定义:

```
    device = 'cuda:0'
    batch_size = 32
    learning_rate = 5e-2
    train_epoch = int(20)
    # 下面的参数是为SNN定义的
    T = int(8)
    tau = float(2.0)
```

精度对比：

SNN：训练20 epochs，最佳分类精度为66.68%

CNN：训练20 epochs，最佳分类精度为68.20%

二者训练精度类似，SNN的分类网络精度上略低于CNN的效果。

效率对比:

在个人的笔记本上,`(RTX 3070ti)`运行速度如下:

SNN:

```
time cost of 100 batch 23.75813102722168 s
```

CNN:

```
time cost of 100 batch 3.7237465381622314 s
```

考虑到SNN每个batch输入数据量是CNN的8倍，乘以8倍后的相同数据量下，CNN耗时为29.79s。

综合对比来看，SNN处理同样规模数据的速度比CNN快了25%，SNN的运行效率高于CNN。