FS-STBP 融合结构测试总结

一、说明

本阶段的工作基于 Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes 和 Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks 两篇文献，将第一篇文章的 FS 神经元模型和第二篇文章的 STBP 梯度替代算法相融合，实现了基于 FS 神经元的直接监督学习。

（1）K 值和 alpha 值的改进情况

原本的 FS 神经元模型每一层的 K 值和 alpha 值为定值，但是实际上，各层神经元平均发射脉冲数(平均单个神经元发的脉冲数)的分布是不均匀的。某些层数发射的脉冲数较多，某些层数发射的脉冲数较少，一般呈现出前层多后层少的特点。在 FS 网络中 K 值如果过大，则平均脉冲数一般会增大，从而使 SNN 芯片能耗增大。同时，由于 FS 神经元编码的误差为 alpha/2^k，K 值过小也会导致误差的增大。

```
K = [4, 3, 3, 3, 2]
alpha = [1, 3, 4, 2, 1]

d1 = H1 = T1 = alpha[0] * 2 ** (-K[0]) * np.array([float(2 ** (K[0] -
i)) for i in range(1, K[0] + 1)]).astype(np.float32)
d2 = H2 = T2 = alpha[1] * 2 ** (-K[1]) * np.array([float(2 ** (K[1] -
i)) for i in range(1, K[1] + 1)]).astype(np.float32)
d3 = H3 = T3 = alpha[2] * 2 ** (-K[2]) * np.array([float(2 ** (K[2] -
i)) for i in range(1, K[2] + 1)]).astype(np.float32)
d4 = H4 = T4 = alpha[3] * 2 ** (-K[3]) * np.array([float(2 ** (K[3] -
i)) for i in range(1, K[3] + 1)]).astype(np.float32)
d5 = H5 = T5 = alpha[4] * 2 ** (-K[4]) * np.array([float(2 ** (K[4] -
i)) for i in range(1, K[4] + 1)]).astype(np.float32)
```

因此我对融合后的代码进行了修改，使得每一层的 K 值和 alpha 值都可以灵活调控，从而寻找更优的超参数设置方案。

（2）脉冲发射函数的修改情况

原来的 FS 神经元代码中，每一个 step 权重都要乘 0.5（相当于 d(t)的作用）

```
self.fc1.weight.data = self.fc1.weight.data * 0.5
```

但是如果加入 STBP 算法会使反传失败，无法训练 SNN。

所以此次代码进行了修改：

```
    z = act_fun(mem - th) * 0.5
```

将每个 step 的 0.5 系数补偿乘在发射函数后，反传成功，SNN 网络可以得到训练。

（3）训练方式的改进情况

与常用的 SNN 训练方法 STDP、ANN 转化法不同的是，FS-STBP 融合结构采用的是 STBP 的反向传播方法，使用脉冲发射函数导数的近似处理函数进行梯度近似，解决了脉冲发射函数发射处不可导的问题。

```
    class ActFun(torch.autograd.Function):
    '''
    Approaximation function of spike firing rate function
    '''

    @staticmethod
    def forward(ctx, input):
        ctx.save_for_backward(input)  # ctx：类似于 self，这里保存下来 input 供反传用
        return input.gt(0.).float()  # input：膜电位，gt 比较是否超过阈值 thresh，return 脉冲 0/1
```

```
    @staticmethod
    def backward(ctx, grad_output):
        input, = ctx.saved_tensors
        grad_input = grad_output.clone()
        temp = abs(input) < lens# 膜电压在门限一定范围内，置 1，发射脉冲，
temp 置 1，该时间步的反传梯度为 1
        return grad_input * temp.float()


act_fun = ActFun.apply
```

可以在不使用 ANN 进行预训练的前提下，实现 SNN 网络的直接监督学习。同时，与大多数采用 LIF-BP 算法的 SNN 相比，FS-STBP 融合结构采用的 FS 神经元具有更加优良的脉冲稀疏性和准确性，有望进一步改进 SNN 芯片性能。

二、测试结果

MLP结构：121-64-124-64-124-10					
			Epoch=80 SNN_FS-STBP结果		
alpha/K	[1, 1, 1, 1, 1]/[8, 7, 6, 5, 4]	[1, 1, 1, 1, 1]/[4, 4, 3, 2, 1]	1, 1, 0.75, 0.5, 0.25]/[4, 4, 3, 2, 1	1, 1, 0.5, 0.5, 0.25]/[4, 4, 3, 2, 1	[1.2, 1, 0.5, 0.5, 0.25]/[4, 4, 3, 2, 1]
acc	96.35	96.81	96.95	96.88	96.86
ave_spikes_1st/121-64	1.42083	0.71889	0.68621	0.69189	0.60811
ave_spikes_2ed/64-124	0.66967	0.39054	0.39547	0.45264	0.47685
ave_spikes_3rd/124-64	0.69528	0.34917	0.34454	0.35987	0.33601
ave_spikes_4th/64-124	0.11507	0.07364	0.08644	0.08654	0.09057
ave_spikes_5th/124-10	0	0	0	0	0
alpha/K	, 0.75, 0.5, 0.5, 0.25]/[4, 3, 3, 2,	[1, 2, 2, 1, 1]/[4, 5, 5, 4, 4]	[1, 2, 3, 1, 1]/[4, 5, 6, 4, 4]	[1, 1, 2, 1, 1]/[4, 4, 5, 4, 4]	[1, 1, 2, 1, 2]/[4, 4, 5, 4, 5]
acc	96.74	97.21	97.1	96.96	96.98
ave_spikes_1st/121-64	0.50559	0.82659	0.84291	0.70354	0.68876
ave_spikes_2ed/64-124	0.41834	0.54722	0.57198	0.51098	0.53501
ave_spikes_3rd/124-64	0.34448	0.42147	0.41611	0.37637	0.50321
ave_spikes_4th/64-124	0.08858	0.0655	0.07137	0.06272	0.06222
ave_spikes_5th/124-10	0	0	0	0	0
alpha/K	[1, 1, 2, 1, 1]/[3, 4, 5, 4, 3]	[1, 1, 1, 1, 1]/[4, 4, 5, 4, 4]	[1, 1, 1, 1, 1]/[4, 5, 6, 5, 4]	[1, 2, 3, 2, 1]/[4, 5, 6, 5, 4]	[1, 1.25, 1.5, 1.25, 1]/[4, 5, 6, 5, 4]
acc	96.83	96.94	96.81	97.34	96.98
ave_spikes_1st/121-64	0.66011	0.65485	0.86493	0.74747	0.88943
ave_spikes_2ed/64-124	0.51814	0.53684	0.69544	0.58867	0.72808
ave_spikes_3rd/124-64	0.47871	0.48003	0.54291	0.47204	0.52753
ave_spikes_4th/64-124	0.0663	0.06109	0.11393	0.06605	0.13155
ave_spikes_5th/124-10	0	0	0	0	0
alpha/K	[1, 2, 3, 2, 1]/[3, 4, 5, 4, 3]	.75, 1, 1.25, 1, 0.75]/[3, 4, 5, 4,	[1, 2, 2, 2, 1]/[3, 4, 5, 4, 3]	[1, 3, 3, 3, 1]/[3, 4, 5, 4, 3]	[1, 3, 4, 3, 1]/[3, 4, 5, 4, 3]
acc	96.99	96.8	96.91	96.98	97.06
ave_spikes_1st/121-64	0.64621	0.60871	0.6198	0.54058	0.52484
ave_spikes_2ed/64-124	0.47926	0.54905	0.57825	0.49151	0.4877
ave_spikes_3rd/124-64	0.33317	0.4897	0.35096	0.31728	0.32821
ave_spikes_4th/64-124	0.06262	0.06589	0.0705	0.05516	0.0658
ave_spikes_5th/124-10	0	0	0	0	0

alpha/K	[1, 3, 6, 2, 1]/[4, 4, 4, 3, 1]	[1, 2, 3, 2, 1]/[4, 3, 3, 3, 2]	[1, 2, 2, 2, 1]/[4, 3, 3, 3, 2]	[1, 3, 3, 2, 1]/[4, 3, 3, 3, 2]	[1,3,2,2,1]/[4,3,3,3,2]
acc	97.24	97.38	97.04	97.37	97.36
ave_spikes_1st/121-64	0.57567	0.50722	0.52828	0.44699	0.43627
ave_spikes_2ed/64-124	0.28961	0.26196	0.29765	0.27613	0.30752
ave_spikes_3rd/124-64	0.32581	0.30885	0.29728	0.29826	0.3122
ave_spikes_4th/64-124	0.08829	0.0668	0.05502	0.06339	0.06329
ave_spikes_5th/124-10	0	0	0	0	0
alpha/K	[1, 2, 2, 2, 1]/[4, 3, 3, 3, 2]	[1, 3, 4, 2, 1]/[4, 3, 3, 3, 2]	[1, 3, 4, 2, 1]/[4, 3, 3, 4, 1]	[1, 3, 4, 2, 1]/[4, 4, 3, 3, 1]	[1, 3, 4, 2, 1]/[4, 4, 4, 3, 1]
acc	97	97.28	97.17	97.37	97.38
ave_spikes_1st/121-64	0.50469	0.47745	0.45777	0.63857	0.56089
ave_spikes_2ed/64-124	0.32554	0.24289	0.22454	0.24125	0.37373
ave_spikes_3rd/124-64	0.31888	0.30288	0.4039	0.32652	0.33521
ave_spikes_4th/64-124	0.05299	0.06687	0.10406	0.09616	0.10257
ave_spikes_5th/124-10	0	0	0	0	0

每一层的 K 值和 alpha 值设置测试数据如上图。

经过测试，可以看到，第一层 K 值设为 4，后层 K 值递减；alpha 值按照“小大小”排列的训练效果较好，可以达到 97.3%左右。在 K 不大于 4 的情况下，通过调参可以使每层每个神经元的平均脉冲数降至 0.5 以下，同时还能有 97.37%左右的准确率。