HW1-2 分类问题

520030910393 马逸川

2.1 在原始数据集上训练Resnet

首先，通过jittor框架提供的函数，下载并加载Cifar10数据集，同时实现了Resnet模型，包含8个`Residual_Block`。

在原数据集上训练得到的Resnet模型，在未修改的数据集上训练并测试，训练参数如下:

```
    jt.flags.use_cuda = 0
    batch_size = 64
    learning_rate = 0.005
    momentum = 0.91
    weight_decay = 1e-4
    epochs = 30
```

损失函数为`cross_entorpy_loss`，优化器为SGD。

训练30epochs后，在测试集上测试，Test acc = 79.16%

2.2 在修改后的数据集上调整模型

修改数据集后，数据的疏密发生了变化。首先，第一个解决思路是，降低仅后五个样本的数据在loss下降时的权重。简单修改代码。在`Train.py`中，此方案训练函数为`Train_edit_A`:

```
# loss re-weighted
# flag 表示是否切分
if flag:
    loss = nn.cross_entropy_loss(outputs, targets)
else:
    loss = nn.cross_entropy_loss(outputs, targets) * 0.001
```

按照同样的参数训练，在测试集上的表现: Test acc = 66.95%

然而，上述的方案虽然能够训练模型，但并没有充分利用数据，有投机取巧之嫌。第二种方案实现如下:

step1: 当前batch的数据被切分时，首先用被切分的数据乘以0.001的权重，进行模型预热。这一部分的实现如下:

```
if not flag:
    # 不充分data训练
    outputs = model(inputs)
    weight = jt.array([0.001, 0.001, 0.001, 0.001, 0.001, 1., 1., 1., 1., 1.])
    loss = nn.cross_entropy_loss(outputs, targets, weight) * 0.001

    optimizer.step(loss)

    if batch_idx % 100 == 0:
        print('Epoch: {} batch:[{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
            epoch + 1, batch_idx, int(len(train_loader) / 64),
            100. * batch_idx / (len(train_loader) / 64.), loss.data[0] * 1000 ))
```

step2: 在batch迭代过程中，记录数据直至出现第一个未被切分的样本，将记录的数据整合进行过采样，(使用的软件包为`imblearn`)得到均匀的数据样本，再通过模型正常下降，这一部分的实现如下:

```
else:
    # Resample
    # images为当前记录的所有数据
    nsamples, nx, ny, nz = images.shape
    reshaped_images = images.reshape((nsamples, nx * ny * nz))
    # undersampling
    Resample_type = RandomOverSampler(random_state=0)
    reshaped_images, labels = Resample_type.fit_resample(reshaped_images,
labels)

    # reshaped again
    nsamples, nsum = reshaped_images.shape
    images = reshaped_images.reshape((nsamples, nx, ny, nz))

    images = jt.array(images)
    labels = jt.array(labels)
    # print(images.shape)
    # print(labels.shape)

    outputs = model(inputs)
    # loss re-weighted
    weight = jt.array([1.1, 1.1, 1.1, 1.1, 1.1, 1., 1., 1., 1., 1.])
    loss = nn.cross_entropy_loss(outputs, targets, weight)

    optimizer.step(loss)

    if batch_idx % 100 == 0:
        print('Epoch: {} batch:[{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
            epoch + 1, batch_idx, int(len(train_loader) / 64),
            100. * batch_idx / (len(train_loader) / 64.), loss.data[0]))
```

按照同样的参数训练，在测试集上的表现: Test acc = 66.19%