案例7：使用ResNet5对Mnist做分类

2023 年 4 月 26 日

[1]:
```
import sys
sys.path.append(r"D:\Rhitta_GPU")
import numpy as np
import cupy as cp
import rhitta.nn as nn
from sklearn.preprocessing import OneHotEncoder
onehot_encoder = OneHotEncoder(sparse_output=False)
```

第一步：载入数据集

[2]:
```
loader=nn.MnistLoader()
train_x,number_labels=loader.load(r"D:\Rhitta_GPU\data\dataset")
labels = cp.array(onehot_encoder.fit_transform(cp.asnumpy(number_labels).
 ↪reshape(-1, 1)))

train_x.shape,number_labels.shape,labels.shape
```

[2]:
```
((60000, 784), (60000,), (60000, 10))
```

第二步：构造模型

[3]:
```
model=nn.ResNet_simple(in_channels=1)
print(model)
```

```
Model:
Layer 1:  Conv2D(in_channels=1,out_channels=4,kernel_size=5,stride=1,padding=0)
kernel : [4,5,5]  num_params: 104
Layer 2: LayerNorm()
Layer 3:  ReLU()
Layer 4:  Pooling(in_channels=4,window_size=2,stride=2,mode=MaxPooling)
Layer 5:  Conv2D(in_channels=4,out_channels=4,kernel_size=3,stride=1,padding=1)
kernel : [4,3,3]  num_params: 40
Layer 6: LayerNorm()
Layer 7:  ReLU()
Layer 8:  Conv2D(in_channels=4,out_channels=4,kernel_size=3,stride=1,padding=1)
kernel : [4,3,3]  num_params: 40
Layer 9: LayerNorm()
Layer 10: ResAdd()
Layer 11:  Conv2D(in_channels=4,out_channels=4,kernel_size=5,stride=1,padding=0)
kernel : [4,5,5]  num_params: 104
Layer 12:  Pooling(in_channels=4,window_size=4,stride=4,mode=AveragePooling)
Layer 13:  Linear(16,10)   num_params: 170
Total params: 458
```

构造完整计算图：输入输出节点，模型，损失

[4]:
```
x=nn.to_tensor(size=(28,28))
label=nn.to_tensor(size=(1,10))
out=model(x)
predict=nn.Softmax(out)
loss=nn.CrossEntropyLoss(out,label)
```

第三步：选择并初始化优化器

[5]:
```
learning_rate = 0.01
optimizer = nn.Adam(nn.default_graph, loss, learning_rate=learning_rate)
```

第四步：开始训练

本框架慢的离谱，就拿16个样本跑通试一试

[6]:
```
epochs = 20
batch_size = 4


for epoch in range(epochs):
```

```
N = 16
count = 0
# 遍历样本训练
for i in range(N):
    x.set_value(train_x[i].reshape(28,28))
    label.set_value(labels[i])
    optimizer.one_step()
    count+=1
    if count >= batch_size:
        optimizer.update()
        count=0
# 遍历样本求准确率
pred=[]
for i in range(N):
    x.set_value(train_x[i].reshape(28,28))
    label.set_value(labels[i])
    predict.forward()
    pred.append(predict.value.flatten())
temp=(cp.array(pred).argmax(axis=1) == number_labels[:N])
accuracy=temp.sum()/N
print("epoch:{} accuracy:{}".format(epoch+1,accuracy))
```

```
epoch:1 accuracy:0.125
epoch:2 accuracy:0.125
epoch:3 accuracy:0.25
epoch:4 accuracy:0.1875
epoch:5 accuracy:0.1875
epoch:6 accuracy:0.125
epoch:7 accuracy:0.0625
epoch:8 accuracy:0.125
epoch:9 accuracy:0.125
epoch:10 accuracy:0.25
epoch:11 accuracy:0.25
epoch:12 accuracy:0.25
epoch:13 accuracy:0.25
epoch:14 accuracy:0.25
epoch:15 accuracy:0.375
```

```
epoch:16 accuracy:0.4375
epoch:17 accuracy:0.5
epoch:18 accuracy:0.5
epoch:19 accuracy:0.5625
epoch:20 accuracy:0.5625
```