In [ ]:

```
import torch as t
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torchvision
from torchvision import datasets, transforms
# Author: 坚定的唯物主义鼠鼠
```

In [ ]:

```
EMNIST=datasets.EMNIST('./data',split='bymerge', train=True
                       , download=True, transform=transforms.ToTensor())

EMNIST_val=datasets.EMNIST('./data', split='bymerge',train=False
                           , download=True, transform=transforms.ToTensor())
data=t.utils.data.DataLoader(EMNIST, batch_size=64, shuffle=True)
data_val=t.utils.data.DataLoader(EMNIST_val, batch_size=64, shuffle=True)
device=t.device('cuda' if t.cuda.is_available() else 'cpu')
```

In [ ]:

```
class Net(nn.Module):
    def __init__(self, *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)
        self.model=nn.Sequential(
        nn.Conv2d(1,64,3,1,padding=1),
        nn.BatchNorm2d(64),
        nn.Conv2d(64,64,3,1,padding=1),
        nn.BatchNorm2d(64),
        nn.ReLU(inplace=True),
        nn.MaxPool2d(2),

        nn.Conv2d(64,128,3,padding=1),
        nn.BatchNorm2d(128),
        nn.Conv2d(128,128,3,padding=1),
        nn.BatchNorm2d(128),
        nn.ReLU(inplace=True),
        nn.MaxPool2d(2,ceil_mode=False),

        nn.Conv2d(128,256,3,padding=1),
        nn.BatchNorm2d(256),
        nn.Conv2d(256,256,3,padding=1),
        nn.BatchNorm2d(256),
        nn.ReLU(inplace=True),
        nn.MaxPool2d(2,ceil_mode=False),

        nn.Conv2d(256,512,3,padding=1),
        nn.BatchNorm2d(512),
        nn.Conv2d(512,512,3,padding=1),
        nn.BatchNorm2d(512),
        nn.ReLU(inplace=True),
        nn.MaxPool2d(2,ceil_mode=False),

        nn.Flatten(),
        nn.Linear(512, 256),
        nn.ReLU(),
        nn.Linear(256,256),nn.ReLU(),
        nn.Linear(256, 47)
        )
    def forward(self, x):
        return self.model(x)
def train(epoches,model,opt,lossfn,data):
    for epoch in range(epoches):
        for img, label in data:
            img=img.to(device)
            label=label.to(device)
            opt.zero_grad()
            output=model(img)
            loss=lossfn(output, label)
            loss.backward()
            opt.step()
        print(f'epoch:{epoch}, loss:{loss.item()}')
```

In [ ]:

```
model=Net().to(device)
model=t.load('./cnn-CY.pth',map_location=device)

opt=optim.SGD(model.parameters(), lr=0.001)
```

In [ ]:

```
train(30,model,opt,nn.CrossEntropyLoss(),data)
t.save(model, './cnn-CY.pth')
```

In [ ]:

```
correct=0
total=0
with t.no_grad():
    for img,lable in data_val:
        output=model(img)
        _,predicted=t.max(output.data,1)
        total+=lable.size(0)
        correct+=(predicted==lable).sum().item()
print('Accuracy: %f %%' % (float(correct)/total*100))
print('总数: %f ' % (total))
print('正确数: %f ' % (correct))
```