```
import random
import torch
import torch.nn as nn
import numpy as np
import os
from PIL import Image #读取图片数据
from torch.utils.data import Dataset, DataLoader
from tqdm import tqdm
from torchvision import transforms
import time
import matplotlib.pyplot as plt
from model_utils.model import initialize_model
def seed_everything(seed):
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.benchmark = False
    torch.backends.cudnn.deterministic = True
    random.seed(seed)
    np.random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
#################################################################
seed_everything(0)
###############################################


HW = 224



train_transform = transforms.Compose(
    [
        transforms.ToPILImage(),   #224, 224, 3模型 ：3, 224, 224
        transforms.RandomResizedCrop(224),
        transforms.RandomRotation(50),
        transforms.ToTensor()
    ]
)

val_transform = transforms.Compose(
    [
        transforms.ToPILImage(),   #224, 224, 3模型 ：3, 224, 224
        transforms.ToTensor()
    ]
)

class food_Dataset(Dataset):
    def __init__(self, path, mode="train"):
        self.mode = mode
        if mode == "semi":
            self.X = self.read_file(path)
        else:
            self.X, self.Y = self.read_file(path)
            self.Y = torch.LongTensor(self.Y)  #标签转为长整形\

        if mode == "train":
            self.transform = train_transform
        else:
            self.transform = val_transform

    def read_file(self, path):
        if self.mode == "semi":
            file_list = os.listdir(path)
            xi = np.zeros((len(file_list), HW, HW, 3), dtype=np.uint8)
            # 列出文件夹下所有文件名字
            for j, img_name in enumerate(file_list):
                img_path = os.path.join(path, img_name)
                img = Image.open(img_path)
                img = img.resize((HW, HW))
                xi[j, ...] = img
            print("读到了%d个数据" % len(xi))
            return xi
        else:
            for i in tqdm(range(11)):
                file_dir = path + "/%02d" % i
                file_list = os.listdir(file_dir)

                xi = np.zeros((len(file_list), HW, HW, 3), dtype=np.uint8)
                yi = np.zeros(len(file_list), dtype=np.uint8)

                # 列出文件夹下所有文件名字
                for j, img_name in enumerate(file_list):
                    img_path = os.path.join(file_dir, img_name)
                    img = Image.open(img_path)
                    img = img.resize((HW, HW))
                    xi[j, ...] = img
                    yi[j] = i

                if i == 0:
                    X = xi
                    Y = yi
                else:
                    X = np.concatenate((X, xi), axis=0)
                    Y = np.concatenate((Y, yi), axis=0)
            print("读到了%d个数据" % len(Y))
            return X, Y

    def __getitem__(self, item):
        if self.mode == "semi":
            return self.transform(self.X[item]), self.X[item]
        else:
            return self.transform(self.X[item]), self.Y[item]
```

```python
    def __len__(self):
        return len(self.X)

class semiDataset(Dataset):
    def __init__(self, no_label_loder, model, device, thres=0.99):
        x, y = self.get_label(no_label_loder, model, device, thres)
        if x == []:
            self.flag = False

        else:
            self.flag = True
            self.X = np.array(x)
            self.Y = torch.LongTensor(y)
            self.transform = train_transform
    def get_label(self, no_label_loder, model, device, thres):
        model = model.to(device)
        pred_prob = []
        labels = []
        x = []
        y = []
        soft = nn.Softmax()
        with torch.no_grad():
            for bat_x, _ in no_label_loder:
                bat_x = bat_x.to(device)
                pred = model(bat_x)
                pred_soft = soft(pred)
                pred_max, pred_value = pred_soft.max(1)
                pred_prob.extend(pred_max.cpu().numpy().tolist())
                labels.extend(pred_value.cpu().numpy().tolist())

        for index, prob in enumerate(pred_prob):
            if prob > thres:
                x.append(no_label_loder.dataset[index][1])   #调用到原始的getitem
                y.append(labels[index])
        return x, y

    def __getitem__(self, item):
        return self.transform(self.X[item]), self.Y[item]
    def __len__(self):
        return len(self.X)

def get_semi_loader(no_label_loder, model, device, thres):
    semiset = semiDataset(no_label_loder, model, device, thres)
    if semiset.flag == False:
        return None
    else:
        semi_loader = DataLoader(semiset, batch_size=16, shuffle=False)
        return semi_loader

class myModel(nn.Module):
    def __init__(self, num_class):
        super(myModel, self).__init__()
        #3 *224 *224  -> 512*7*7 -> 拉直 -》全连接分类
        self.conv1 = nn.Conv2d(3, 64, 3, 1, 1)    # 64*224*224
        self.bn1 = nn.BatchNorm2d(64)
        self.relu = nn.ReLU()
        self.pool1 = nn.MaxPool2d(2)   #64*112*112


        self.layer1 = nn.Sequential(
            nn.Conv2d(64, 128, 3, 1, 1),    # 128*112*112
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.MaxPool2d(2)   #128*56*56
        )
        self.layer2 = nn.Sequential(
            nn.Conv2d(128, 256, 3, 1, 1),
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.MaxPool2d(2)   #256*28*28
        )
        self.layer3 = nn.Sequential(
            nn.Conv2d(256, 512, 3, 1, 1),
            nn.BatchNorm2d(512),
            nn.ReLU(),
            nn.MaxPool2d(2)   #512*14*14
        )

        self.pool2 = nn.MaxPool2d(2)    #512*7*7
        self.fc1 = nn.Linear(25088, 1000)   #25088->1000
        self.relu2 = nn.ReLU()
        self.fc2 = nn.Linear(1000, num_class)  #1000-11

    def forward(self, x):
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu(x)
        x = self.pool1(x)
        x = self.layer1(x)
        x = self.layer2(x)
        x = self.layer3(x)
        x = self.pool2(x)
        x = x.view(x.size()[0], -1)
        x = self.fc1(x)
        x = self.relu2(x)
        x = self.fc2(x)
        return x

def train_val(model, train_loader, val_loader, no_label_loader, device, epochs,
optimizer, loss, thres, save_path):
    model = model.to(device)
    semi_loader = None
    plt_train_loss = []
    plt_val_loss = []

    plt_train_acc = []
    plt_val_acc = []

    max_acc = 0.0

    for epoch in range(epochs):
        train_loss = 0.0
        val_loss = 0.0
        train_acc = 0.0
        val_acc = 0.0
        semi_loss = 0.0
        semi_acc = 0.0


        start_time = time.time()

        model.train()
        for batch_x, batch_y in train_loader:
            x, target = batch_x.to(device), batch_y.to(device)
            pred = model(x)
            train_bat_loss = loss(pred, target)
            train_bat_loss.backward()
            optimizer.step()  # 更新参数 之后要梯度清零否则会累积梯度
            optimizer.zero_grad()
            train_loss += train_bat_loss.cpu().item()
            train_acc += np.sum(np.argmax(pred.detach().cpu().numpy(), axis=1) ==
target.cpu().numpy())
        plt_train_loss.append(train_loss / train_loader.__len__())
        plt_train_acc.append(train_acc/train_loader.dataset.__len__()) #记录准确率,

        if semi_loader!= None:
            for batch_x, batch_y in semi_loader:
                x, target = batch_x.to(device), batch_y.to(device)
                pred = model(x)
                semi_bat_loss = loss(pred, target)
                semi_bat_loss.backward()
                optimizer.step()  # 更新参数 之后要梯度清零否则会累积梯度
                optimizer.zero_grad()
                semi_loss += train_bat_loss.cpu().item()
                semi_acc += np.sum(np.argmax(pred.detach().cpu().numpy(), axis=1) ==
target.cpu().numpy())
            print("半监督数据集的训练准确率为", semi_acc/train_loader.dataset.__len__())


        model.eval()
        with torch.no_grad():
            for batch_x, batch_y in val_loader:
                x, target = batch_x.to(device), batch_y.to(device)
                pred = model(x)
                val_bat_loss = loss(pred, target)
                val_loss += val_bat_loss.cpu().item()
                val_acc += np.sum(np.argmax(pred.detach().cpu().numpy(), axis=1) ==
target.cpu().numpy())
        plt_val_loss.append(val_loss / val_loader.dataset.__len__())
        plt_val_acc.append(val_acc / val_loader.dataset.__len__())

        if epoch%3 == 0 and plt_val_acc[-1] > 0.6:
            semi_loader = get_semi_loader(no_label_loader, model, device, thres)

        if val_acc > max_acc:
            torch.save(model, save_path)
            max_acc = val_loss

        print('[%03d/%03d] %2.2f sec(s) TrainLoss : %.6f | valLoss: %.6f Trainacc : %.6f
| valacc: %.6f' % \
              (epoch, epochs, time.time() - start_time, plt_train_loss[-1],
plt_val_loss[-1], plt_train_acc[-1], plt_val_acc[-1])
              )  # 打印训练结果。 注意python语法， %2.2f 表示小数位为2的浮点数， 后面可以对应。

    plt.plot(plt_train_loss)
    plt.plot(plt_val_loss)
    plt.title("loss")
    plt.legend(["train", "val"])
    plt.show()


    plt.plot(plt_train_acc)
    plt.plot(plt_val_acc)
    plt.title("acc")
    plt.legend(["train", "val"])
    plt.show()

# path = r"F:\pycharm\beike\classification\food_classification\food-11\training\labeled"
# train_path = r"F:\pycharm\beike\classification\food_classification\food-
11\training\labeled"
# val_path = r"F:\pycharm\beike\classification\food_classification\food-11\validation"
train_path = r"F:\pycharm\beike\classification\food_classification\food-
11_sample\training\labeled"
val_path = r"F:\pycharm\beike\classification\food_classification\food-
11_sample\validation"
no_label_path = r"F:\pycharm\beike\classification\food_classification\food-
11_sample\training\unlabeled\00"

train_set = food_Dataset(train_path, "train")
val_set = food_Dataset(val_path, "val")
no_label_set = food_Dataset(no_label_path, "semi")

train_loader = DataLoader(train_set, batch_size=16, shuffle=True)
val_loader = DataLoader(val_set, batch_size=16, shuffle=True)
no_label_loader = DataLoader(no_label_set, batch_size=16, shuffle=False)

# model = myModel(11)
```

```python
model, _ = initialize_model("vgg", 11, use_pretrained=True)


lr = 0.001
loss = nn.CrossEntropyLoss()
optimizer = torch.optim.AdamW(model.parameters(), lr=lr, weight_decay=1e-4)
device = "cuda" if torch.cuda.is_available() else "cpu"
save_path = "model_save/best_model.pth"
epochs = 15
thres = 0.99



train_val(model, train_loader, val_loader, no_label_loader, device, epochs, optimizer,
loss, thres, save_path)
```