**一、File Structure**

├── data/

│ ├── \_\_init\_\_.py

│ ├── isic2016.py

│ ├── isic2017.py

│ └── isic2018.py

├── models/

│ ├── \_\_init\_\_.py

│ ├── dual\_head\_unet.py

│ ├── teacher\_student.py

│ └── uncertainty.py

├── losses/

│ ├── \_\_init\_\_.py

│ ├── supervised\_loss.py

│ ├── auxiliary\_loss.py

│ └── pseudo\_loss.py

├── utils/

│ ├── \_\_init\_\_.py

│ ├── augmentations.py

│ ├── metrics.py

│ └── visualization.py

├── configs/

│ ├── \_\_init\_\_.py

│ └── udamt.yaml

├── checkpoints/

│ └── README.md

├── results/

│ ├── logs/

│ └── visualization/

├── train.py

├── evaluate.py

├── requirements.txt

└── README.md

**二、Core code files**

**1. Data loading module (data/isic2018.py)**

import os

import torch

from torch.utils.data import Dataset

from PIL import Image

import numpy as np

from torchvision.transforms import Compose, Resize, ToTensor, Normalize

class ISIC2018Dataset(Dataset):

def \_\_init\_\_(self, root\_dir, split='train', labeled\_ratio=0.05):

self.root\_dir = root\_dir

self.split = split

self.labeled\_ratio = labeled\_ratio

self.image\_files = sorted([f for f in os.listdir(os.path.join(root\_dir, 'images')) if f.endswith('.jpg')])

self.mask\_files = sorted([f for f in os.listdir(os.path.join(root\_dir, 'masks')) if f.endswith('.png')])

# Split labeled and unlabeled data

num\_labeled = int(len(self.image\_files) \* labeled\_ratio)

if split == 'train':

self.image\_files = self.image\_files[:num\_labeled] # Labeled

self.mask\_files = self.mask\_files[:num\_labeled]

elif split == 'unlabeled':

self.image\_files = self.image\_files[num\_labeled:] # Unlabeled

self.mask\_files = self.mask\_files[num\_labeled:]

self.transform = Compose([

Resize((224, 224)),

ToTensor(),

Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

])

def \_\_len\_\_(self):

return len(self.image\_files)

def \_\_getitem\_\_(self, idx):

img\_path = os.path.join(self.root\_dir, 'images', self.image\_files[idx])

mask\_path = os.path.join(self.root\_dir, 'masks', self.mask\_files[idx])

image = Image.open(img\_path).convert('RGB')

mask = Image.open(mask\_path).convert('L')

image = self.transform(image)

mask = Resize((224, 224))(mask)

mask = ToTensor()(mask).squeeze(0)

return image, mask

**2. Dual-head segmentation network (models/dual\_head\_unet.py)**

import torch

import torch.nn as nn

from torchvision.models import resnet50

from torchvision.models.segmentation import deeplabv3\_resnet50

class DualHeadSeg(nn.Module):

def \_\_init\_\_(self, num\_classes=1):

super().\_\_init\_\_()

# Shared Encoder (ResNet50)

self.encoder = nn.Sequential(\*list(resnet50(pretrained=True).children())[:-2]

# Main Head (FCN)

self.main\_head = nn.Sequential(

nn.Conv2d(2048, 512, kernel\_size=3, padding=1),

nn.ReLU(),

nn.Conv2d(512, num\_classes, kernel\_size=1)

# Auxiliary Head (DeepLabv3+)

self.aux\_head = deeplabv3\_resnet50(pretrained=True).classifier

self.aux\_head[4] = nn.Conv2d(256, num\_classes, kernel\_size=1)

def forward(self, x):

features = self.encoder(x)

main\_out = self.main\_head(features)

main\_out = nn.functional.interpolate(main\_out, scale\_factor=16, mode='bilinear')

aux\_out = self.aux\_head(features)

aux\_out = nn.functional.interpolate(aux\_out, scale\_factor=16, mode='bilinear')

return main\_out, aux\_out

**3. Teacher-student framework and EMA update (models/teacher\_student.py)**

import copy

import torch.nn as nn

class MeanTeacher(nn.Module):

def \_\_init\_\_(self, student\_model, alpha=0.99):

super().\_\_init\_\_()

self.student = student\_model

self.teacher = copy.deepcopy(student\_model)

self.alpha = alpha

self.\_freeze\_teacher()

def \_freeze\_teacher(self):

for param in self.teacher.parameters():

param.requires\_grad = False

def update\_teacher(self):

for t\_param, s\_param in zip(self.teacher.parameters(), self.student.parameters()):

t\_param.data = self.alpha \* t\_param.data + (1 - self.alpha) \* s\_param.data

def forward(self, x, is\_teacher=True):

return self.teacher(x) if is\_teacher else self.student(x)

**4. Loss function (losses/pseudo\_loss.py)**

import torch

import torch.nn as nn

class PseudoLabelLoss(nn.Module):

def \_\_init\_\_(self, threshold=0.2):

super().\_\_init\_\_()

self.threshold = threshold

self.bce\_loss = nn.BCEWithLogitsLoss(reduction='none')

def forward(self, student\_pred, teacher\_pred, uncertainty\_map):

mask = (uncertainty\_map < self.threshold).float()

loss = self.bce\_loss(student\_pred, teacher\_pred.sigmoid().detach())

masked\_loss = (loss \* mask).mean()

return masked\_loss

**5. Training script (train.py)**

import torch

from torch.utils.data import DataLoader

from data.isic2018 import ISIC2018Dataset

from models.dual\_head\_unet import DualHeadSeg

from models.teacher\_student import MeanTeacher

from losses.supervised\_loss import SupervisedLoss

from losses.pseudo\_loss import PseudoLabelLoss

from utils.augmentations import WeakStrongAugmentation

import yaml

# Load config

with open("configs/udamt.yaml", "r") as f:

config = yaml.safe\_load(f)

# Initialize datasets

labeled\_dataset = ISIC2018Dataset(root\_dir="data/ISIC2018", split='train', labeled\_ratio=config['labeled\_ratio'])

unlabeled\_dataset = ISIC2018Dataset(root\_dir="data/ISIC2018", split='unlabeled')

labeled\_loader = DataLoader(labeled\_dataset, batch\_size=config['batch\_size'], shuffle=True)

unlabeled\_loader = DataLoader(unlabeled\_dataset, batch\_size=config['batch\_size'], shuffle=True)

# Initialize models

student = DualHeadSeg()

teacher = MeanTeacher(student, alpha=config['ema\_alpha'])

optimizer = torch.optim.Adam(student.parameters(), lr=config['lr'])

# Loss functions

supervised\_loss = SupervisedLoss()

pseudo\_loss = PseudoLabelLoss(threshold=config['uncertainty\_threshold'])

# Training loop

for epoch in range(config['total\_epochs']):

# Supervised phase (first 20 epochs)

if epoch < 20:

for images, masks in labeled\_loader:

main\_pred, aux\_pred = student(images)

loss = supervised\_loss(main\_pred, masks) + supervised\_loss(aux\_pred, masks)

optimizer.zero\_grad()

loss.backward()

optimizer.step()

# Semi-supervised phase

else:

for (labeled\_imgs, labeled\_masks), unlabeled\_imgs in zip(labeled\_loader, unlabeled\_loader):

# Labeled loss

main\_pred, aux\_pred = student(labeled\_imgs)

sup\_loss = supervised\_loss(main\_pred, labeled\_masks) + supervised\_loss(aux\_pred, labeled\_masks)

# Generate pseudo-labels

with torch.no\_grad():

teacher\_pred, \_ = teacher(unlabeled\_imgs, is\_teacher=True)

uncertainty = teacher.calculate\_uncertainty(unlabeled\_imgs)

# Pseudo-label loss

student\_pred, \_ = student(unlabeled\_imgs)

unsup\_loss = pseudo\_loss(student\_pred, teacher\_pred, uncertainty)

# Total loss

total\_loss = sup\_loss + config['lambda\_pseudo'] \* unsup\_loss

optimizer.zero\_grad()

total\_loss.backward()

optimizer.step()

teacher.update\_teacher()

**6. Configuration file (configs/udamt.yaml)**

# Training parameters

total\_epochs: 100

batch\_size: 8

lr: 0.001

ema\_alpha: 0.99

lambda\_pseudo: 0.7

# Dataset settings

labeled\_ratio: 0.05 # 5% labeled data

input\_size: [224, 224]

# Uncertainty threshold

uncertainty\_threshold: 0.2

**7. Running instructions (README.md)**

# UDAMT: Uncertainty-Driven Auxiliary Mean Teacher for Skin Lesion Segmentation

## Requirements

- Python 3.8

- PyTorch 1.8.1

- CUDA 11.1

- Install dependencies: `pip install -r requirements.txt`

## Dataset Preparation

1. Download ISIC 2016/2017/2018 datasets from [ISIC Archive](https://challenge.isic-archive.com/data).

2. Place images in `data/ISIC2018/images` and masks in `data/ISIC2018/masks`.

## Training

```bash

python train.py --config configs/udamt.yaml --labeled\_ratio 0.05

**Evaluation**

python evaluate.py --checkpoint checkpoints/best\_model.pth --dataset ISIC2018

**Results**

* **ISIC 2018 (5% labeled)**:
  + Dice Coefficient: 87.84%
  + Inference Speed: 25.7 ms/image (NVIDIA V100)

**Necessity of unified description of datasets**

1. Experimental reproducibility: clarify the source, division method and preprocessing details of the dataset to ensure that others can reproduce the experiment.
2. Result comparability: explain the differences between different datasets (such as lesion type, image quality, annotation standards), and explain why these datasets are selected for comparison.
3. Model generalization verification: verify the model's adaptability to out-of-distribution data through multi-dataset experiments.

**1. Basic information of the dataset**

| **Dataset** | **ISIC 2016** | **ISIC 2017** | **ISIC 2018** |
| --- | --- | --- | --- |
| **Number of images** | 900 | 2000 | 2594 |
| **Resolution** | 768×512 | 1024×1024 | 1024×1024 |
| **Lesion type** | Mainly Melanoma | Melanoma, Nevus | Melanoma, Keratosis |
| **Labeling standard** | Pixel-level binary labeling | Pixel-level binary labeling | Pixel-level binary labeling |
| **Data division** | Training: 720 Validation: 90 Testing: 180 | Training: 1600 Validation: 200 Testing: 400 | Training: 2075 Validation: 259 Testing: 520 |

**2. Unified data preprocessing process**

Image size: uniformly scaled to 224×224 pixels (balance computational efficiency and detail preservation).

Normalization: pixel values are normalized to [0, 1], using ImageNet mean ([0.485, 0.456, 0.406]) and standard deviation ([0.229, 0.224, 0.225]).

Data augmentation:

Weak augmentation (for teacher model): random rotation (±15°), horizontal/vertical flip.

Strong augmentation (for student model): color jitter (brightness ±0.2, contrast ±0.3), Gaussian noise (σ=0.1).

**3. Annotation strategy**

Semi-supervised setting: only 5% and 10% of the labeled data are used (e.g. 5% of ISIC 2018 corresponds to about 104 labeled images).

Pseudo-label generation: The teacher model generates pseudo-labels for unlabeled data and excludes high-noise areas through uncertainty filtering (threshold 0.2).

**4. Challenges of the dataset**

ISIC 2016: Small data volume, suitable for verifying the robustness of the model in low-label scenarios.

ISIC 2017: Contains more lesion types (such as moles) to test the adaptability of the model to category diversity.

ISIC 2018: The image resolution is high and the lesion boundaries are blurred, verifying the model's ability to capture complex boundaries.

**Discussion points**

1. Generalization across datasets: The performance improvement on ISIC 2016 (small sample) and ISIC 2018 (high resolution) proves the robustness of the model to data distribution.

2. Noise robustness: The mixed mole lesions in ISIC 2017 verify the effectiveness of the uncertainty mechanism.

3. Computational efficiency: The unified input resolution (224×224) ensures the fairness of the comparative experiment.

**Notes**

**Data source: Clearly mark the dataset from the ISIC official website (https://challenge.isic-archive.com) to avoid copyright disputes.**