# MICCAI2D Technical Solution for the Fourth Place in t

## 1 Overall process

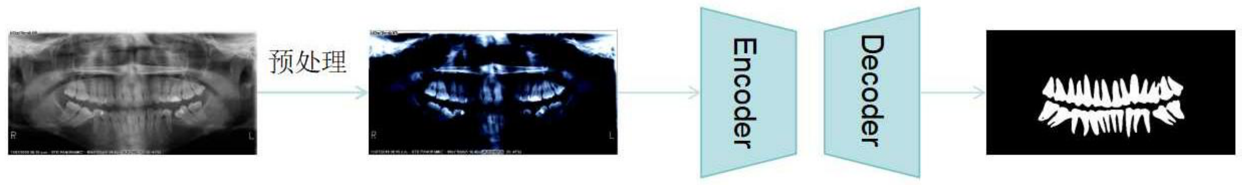
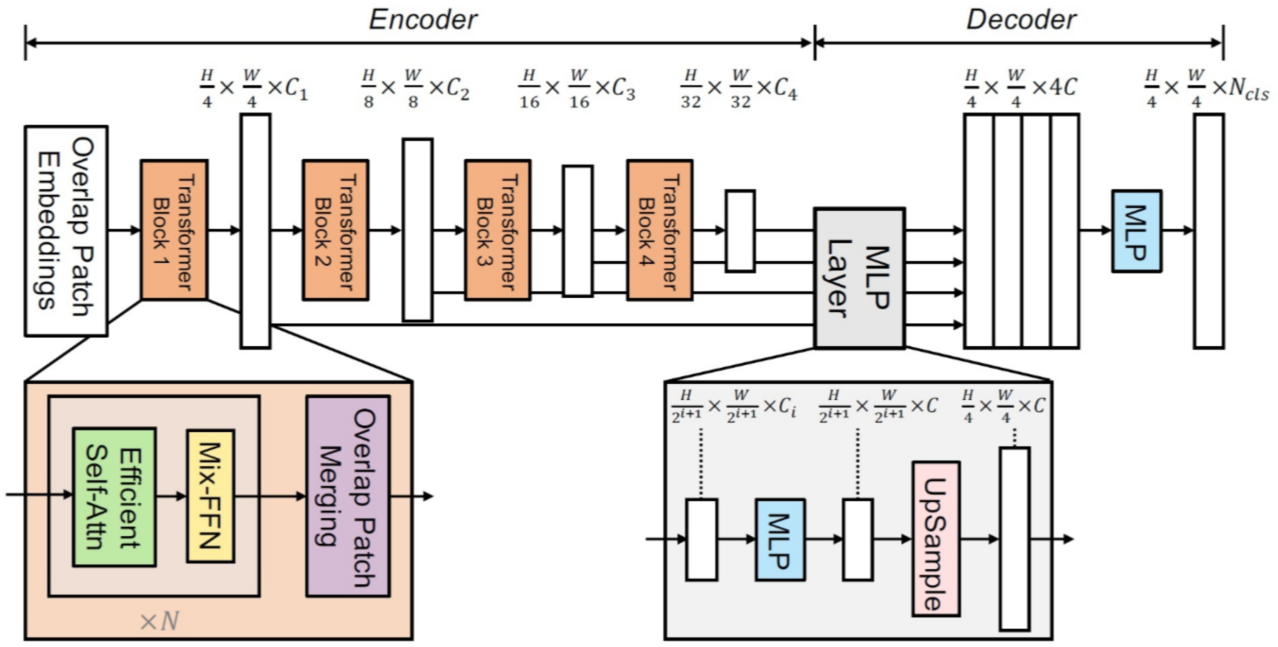


Figure 1 Overall process



Encoder
Decoder
$\frac{H}{4}\times\frac{W}{4}\times C_1$
$\frac{H}{8}\times\frac{W}{8}\times C_2$
$\frac{H}{16}\times\frac{W}{16}\times C_3$
$\frac{H}{32}\times\frac{W}{32}\times C_4$
$\frac{H}{4}\times\frac{W}{4}\times 4C$
$\frac{H}{4}\times\frac{W}{4}\times N_{cls}$
Overlap Patch Embeddings
Transformer Block 1
Transformer Block 2
Transformer Block 3
Transformer Block 4
MLP Layer
MLP
Efficient Self-Attn
Mix-FFN
Overlap Patch Merging
$\times N$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C_i$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C$
$\frac{H}{4}\times\frac{W}{4}\times C$
MLP
UpSample

Figure 2 Encoder internal architecture

The overall process is show$X \in$ $^{C\times H\times W}$n Figure 1. The original 2D tooth image is normalized by data augmentation an d preprocessing and is sent to the MixVisionTransformer Encoder (Figure 2) to extract multi-scale features of mul tiple stages. Assuming that the original image, H and W are the sizes of the picture, respectively, in this track, C is fixed to $_1$320 and 640, C is the number of image channels equal to 3. The feature map sizes of the four different sta ges of the features extracted by MixVisionTransformer Encoder are , , , , , , , , respectively, C1, C2, C3, and C4 ar e 64, 128, 256 and 512, respectively. 4 4 $H$ $\times$ $W$ $\times C$ 8 8 $H$ $\times$ $W$ $\times C$ $_2$ 16 16 $_3$
$H$ $\times$ $W$ $\times C$ $_4$ 32 32 $H$ $\times$ $W$ $\times C$

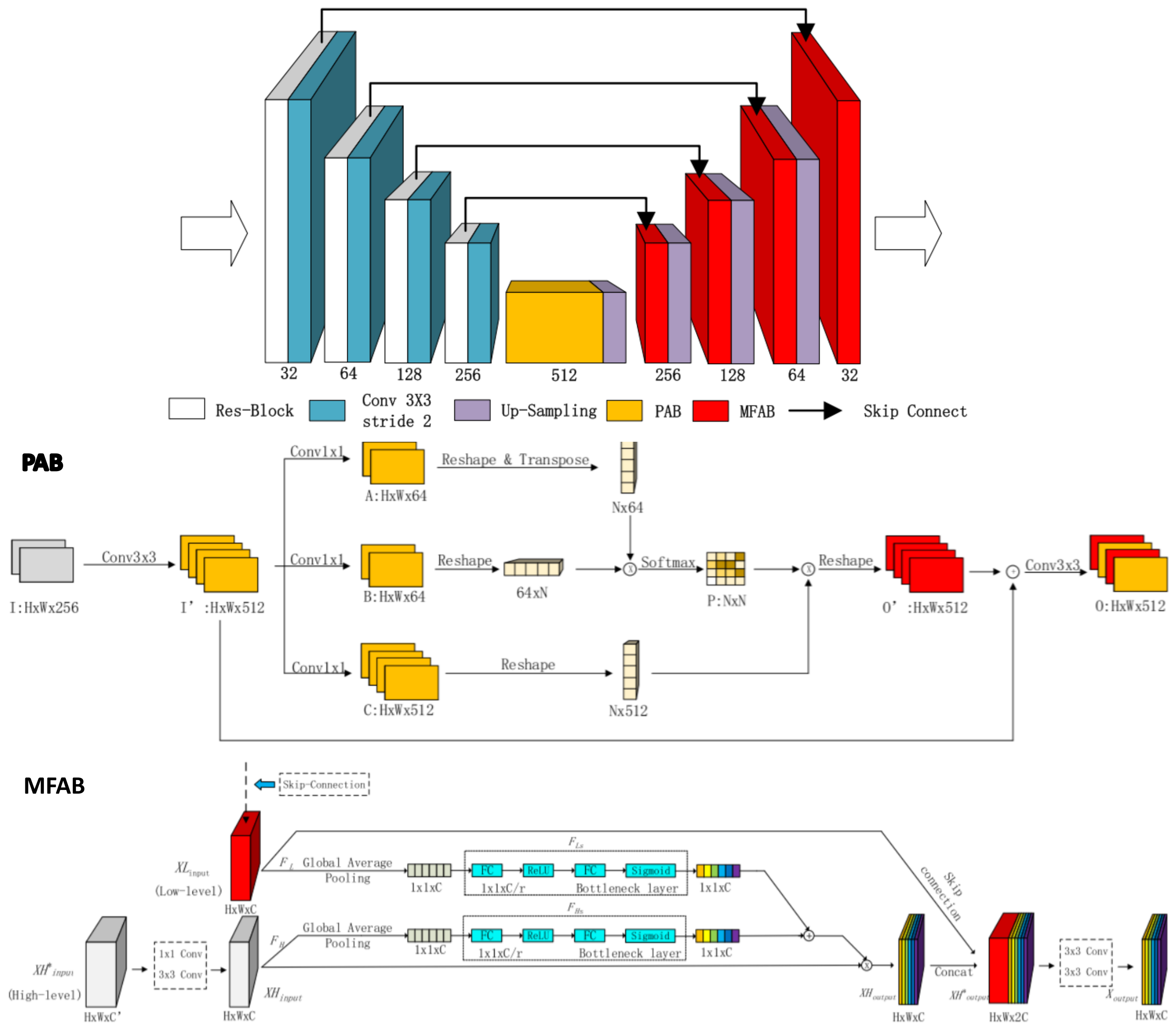


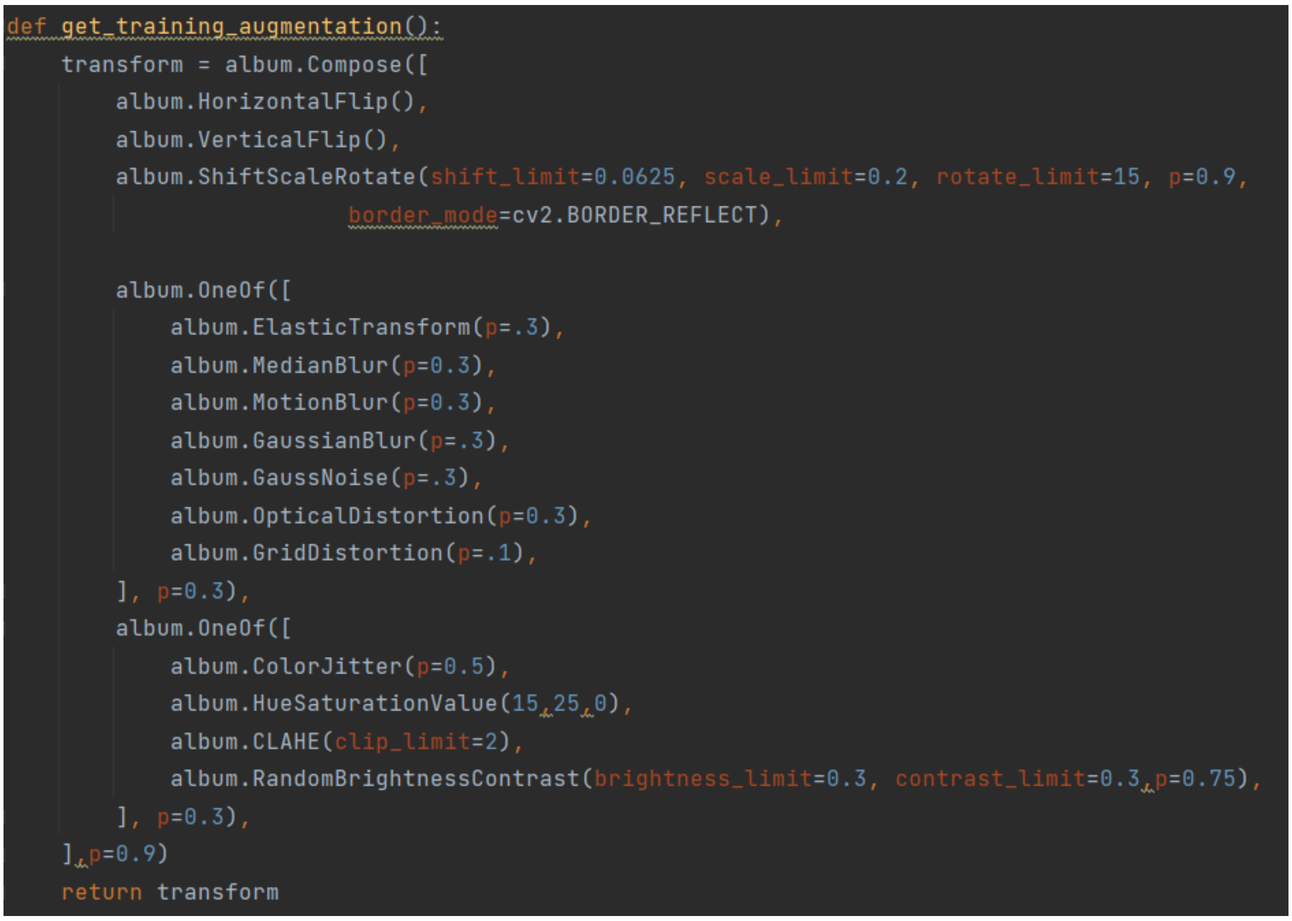
Figure 3 Decoder internal architecture

The Decoder section uses Manet, which contains two different attention mechanism modules called Position-wise Attention Block (PAB) and Multi-scale Fusion Attention Block (MFAB), which are used to capture space and channel-level attention feature maps.

## 2 Experiment details

### 2.1 Data Enhancement Section

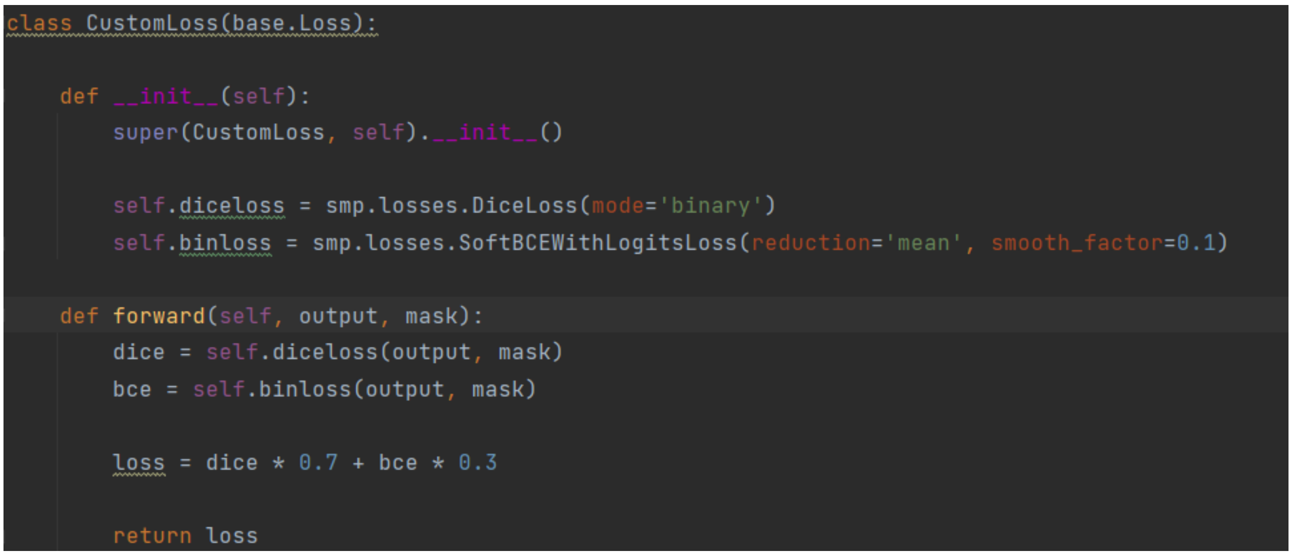
Using RandomFlip, ShiftScaleRotate, ElasticTransform, GaussianBlury and some color contrast changes such as ColorJitte and RandomBrightnessContrast. Since the image size is fixed to 320*640, no Crop and Resize operations are used.



```python
def get_training_augmentation():
    transform = album.Compose([
        album.HorizontalFlip(),
        album.VerticalFlip(),
        album.ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.2, rotate_limit=15, p=0.9,
                                 border_mode=cv2.BORDER_REFLECT),

        album.OneOf([
            album.ElasticTransform(p=.3),
            album.MedianBlur(p=0.3),
            album.MotionBlur(p=0.3),
            album.GaussianBlur(p=.3),
            album.GaussNoise(p=.3),
            album.OpticalDistortion(p=0.3),
            album.GridDistortion(p=.1),
        ], p=0.3),
        album.OneOf([
            album.ColorJitter(p=0.5),
            album.HueSaturationValue(15,25,0),
            album.CLAHE(clip_limit=2),
            album.RandomBrightnessContrast(brightness_limit=0.3, contrast_limit=0.3,p=0.75),
        ], p=0.3),
    ],p=0.9)
    return transform
```

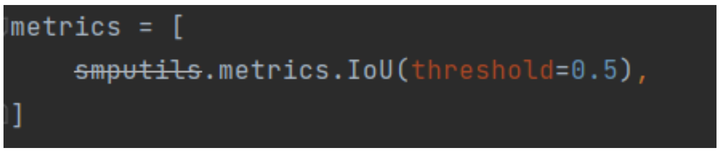
## 2.2 Loss function and evaluation index part

A combination of Dice and SoftBCEloss was used, with a proportional coefficient of 7:3.



```python
class CustomLoss(base.Loss):

    def __init__(self):
        super(CustomLoss, self).__init__()

        self.diceloss = smp.losses.DiceLoss(mode='binary')
        self.binloss = smp.losses.SoftBCEWithLogitsLoss(reduction='mean', smooth_factor=0.1)

    def forward(self, output, mask):
        dice = self.diceloss(output, mask)
        bce = self.binloss(output, mask)

        loss = dice * 0.7 + bce * 0.3

        return loss
```

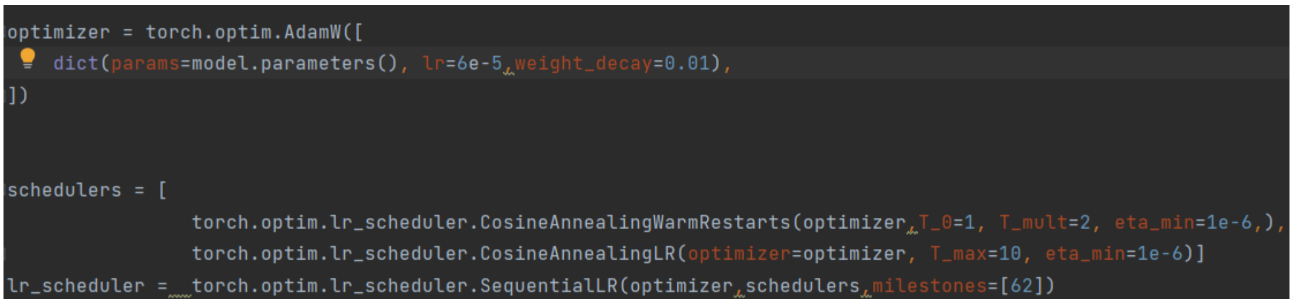
Evaluation indicators are set to 0.5 using the Iou indicator threshold.



```python
metrics = [
    ~~smputils~~.metrics.IoU(threshold=0.5),
]
```

## 2.3 Training

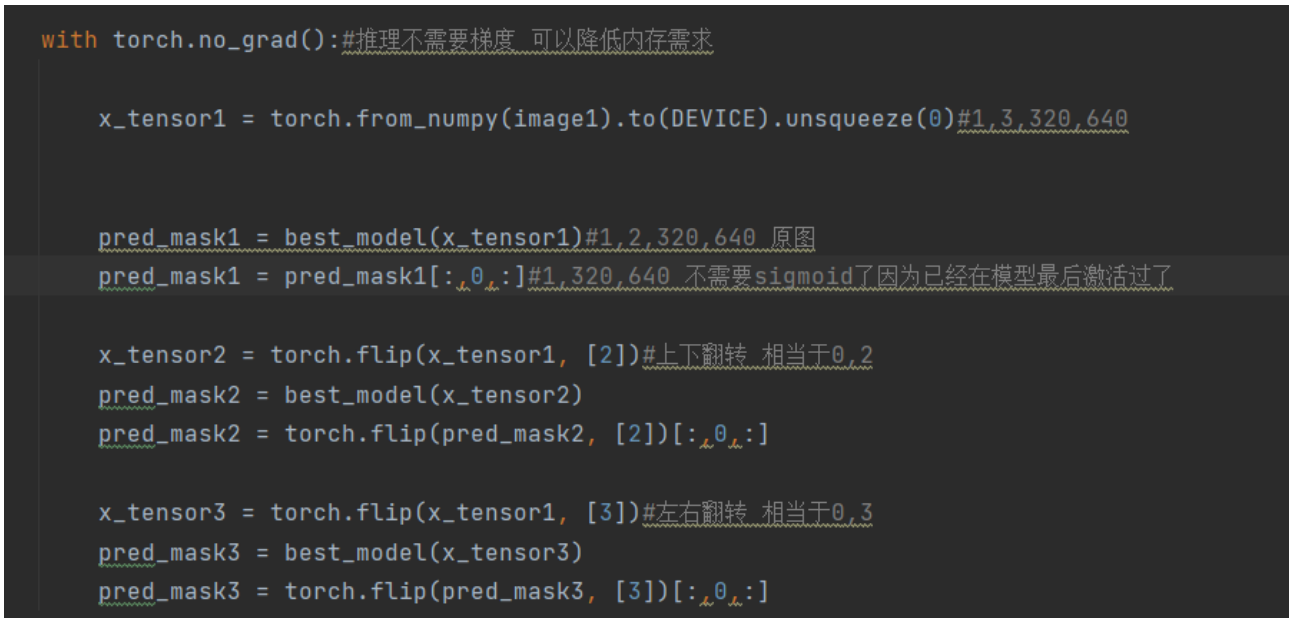
Use five-fold cross-training, Epoch is 200, Batchsize is 8, the optimizer uses AdamW, the initial learning rat e is 6e-5, and the cosine annealing combination learning rate strategy is as follows:



```python
optimizer = torch.optim.AdamW([
    dict(params=model.parameters(), lr=6e-5,weight_decay=0.01),
])


schedulers = [
                torch.optim.lr_scheduler.CosineAnnealingWarmRestarts(optimizer,T_0=1, T_mult=2, eta_min=1e-6,),
                torch.optim.lr_scheduler.CosineAnnealingLR(optimizer=optimizer, T_max=10, eta_min=1e-6)]
lr_scheduler =  torch.optim.lr_scheduler.SequentialLR(optimizer,schedulers,milestones=[62])
```

## 2.4 Reasoning

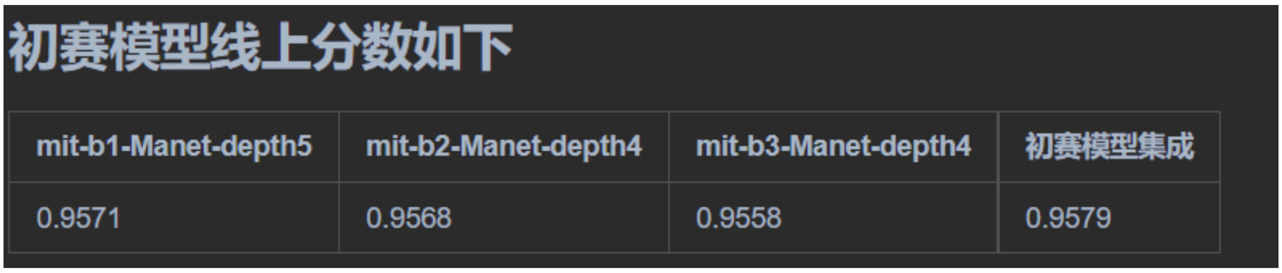
During the inference process, Test Time Augmentation (TTA) is used to predict the original image horizonta lly and vertically, then flip it back, and finally the average of the three prediction results is taken as the final result .



```
with torch.no_grad():#推理不需要梯度 可以降低内存需求

    x_tensor1 = torch.from_numpy(image1).to(DEVICE).unsqueeze(0)#1,3,320,640


    pred_mask1 = best_model(x_tensor1)#1,2,320,640 原图
    pred_mask1 = pred_mask1[:,0,:]#1,320,640 不需要sigmoid了因为已经在模型最后激活过了

    x_tensor2 = torch.flip(x_tensor1, [2])#上下翻转 相当于0,2
    pred_mask2 = best_model(x_tensor2)
    pred_mask2 = torch.flip(pred_mask2, [2])[:,0,:]

    x_tensor3 = torch.flip(x_tensor1, [3])#左右翻转 相当于0,3
    pred_mask3 = best_model(x_tensor3)
    pred_mask3 = torch.flip(pred_mask3, [3])[:,0,:]
```

## 3 Experimental results

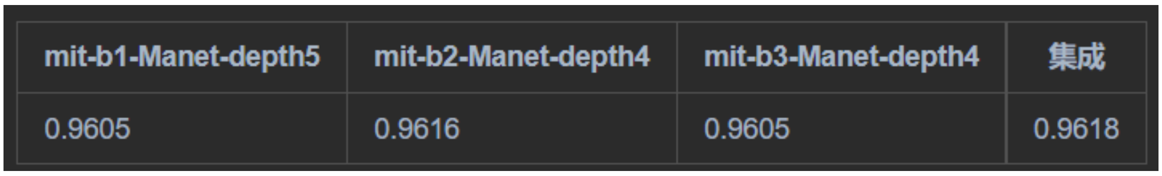
The first place in the preliminary round was won, and the preliminary round data contained 2,000 labeled dat a and 500 test data. Backbone used MixVisionTransformer(mit-b1, mit-b2, mit-b3)



**初赛模型线上分数如下**

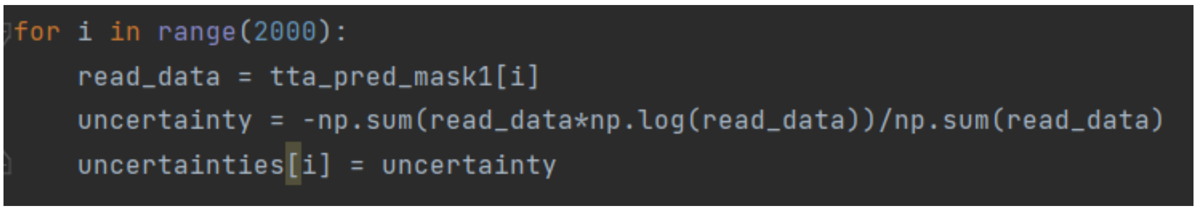
| mit-b1-Manet-depth5 | mit-b2-Manet-depth4 | mit-b3-Manet-depth4 | 初赛模型集成 |
|---|---|---|---|
| 0.9571 | 0.9568 | 0.9558 | 0.9579 |

The rematch is different from the preliminary round. It includes 900 labeled data, 2,000 unlabeled data and 5 00 test data. It can directly use the preliminary round integrated model to reach 0.9599 on the rematch ranking list , which is already a good score. At the beginning, I did not use unlabeled data, only used 900 labeled data training scores as follows:



| mit-b1-Manet-depth5 | mit-b2-Manet-depth4 | mit-b3-Manet-depth4 | 集成 |
|---|---|---|---|
| 0.9605 | 0.9616 | 0.9605 | 0.9618 |

Then I used the mit-b2-Manet-depth4 model to reason about the labelless data. At the same time, in order to reduce the low-quality pseudo-labels, I chose the top 1000 pseudo-labels to be added to the training set. The pseu do-code is as follows:



```
for i in range(2000):
    read_data = tta_pred_mask1[i]
    uncertainty = -np.sum(read_data*np.log(read_data))/np.sum(read_data)
    uncertainties[i] = uncertainty
```

Due to the limitation of submission times and time, I only chose the mit-b2-Manet-depth4 model. Finally, after adding the pseudo-label data, the results of the 5 mit-b2-Manet-depth4 model fusion of 5 fold cross-trained cross-training reached 0.9621 on the rematch test set (recent match fourth).

## 4. Ideas for improvement

Adjust the binary classification threshold. In this competition, I used the default threshold of 0.5, and maybe the optimal thresho

Consider multi-scale reasoning TTA and sliding window reasoning TTA. Try to use post-processing, such as removing fine holes, etc. Due to time constraints, the pseudo-label is only iterated once, and iterations can improve the quality of the pseudo-label.