PraNet: Parallel Reverse Attention Network for Polyp Segmentation

Wujiatai

摘要

结肠镜检查是发现与结直肠癌高度相关的结直肠息肉的有效技术。在临床实践中，从结肠镜图像中分割息肉具有重要意义，因为它为诊断和手术提供了有价值的信息。然而，准确的息肉分割是一项具有挑战性的任务，原因有两个：(1) 相同类型的息肉具有大小、颜色和纹理的多样性; 和 (2) 息肉和其周围粘膜之间的边界不清晰。为了解决这些问题，通过阅读文献，我决定复现一种并行反向注意网络（PraNet），用于结肠镜图像中准确的息肉分割。并且，在阅读文献过程中我发现皮肤镜医学图像也具有与肠息肉类似的特征。因此，我决定尝试将 PraNet 运用到皮肤镜医学图像中，结果在我的数据集中，得到了不错的分割效果。

关键词：肠息肉检测；并行反向注意网络

1 引言

肠癌是目前全球第三大常见癌症类型。而大部分肠癌来源于肠息肉癌变，结肠镜检查是一种有效的筛查结直肠癌和预防的技术，它可以提供结直肠息肉的位置和外观信息，使医生在发展为结直肠癌之前切除。许多研究表明，早期的结肠镜检查降低了 30% 的肠癌发生率。因此，在早期的肠镜检查中细分肠息肉是相当重要的。但是，分割肠息肉有两大难题:

1. 即使是同种类型的肠息肉，通常在外观上也会有很大不同，比如: 大小，形状，颜色等。这导致了使用特征组合的方法来分割它的漏检率比较高，比如通过大小，形状和颜色这些基本特征组合分割。

2. 在肠镜检查的图像中，息肉与周围粘膜的边界比较模糊。这些问题导致息肉的分割不准确，有时甚至导致息肉的检测缺失。因此，一种自动准确的早期检测所有可能息肉的息肉分割方法对于预防结直肠癌具有重要意义。

因此，我在老师推荐和自己查阅文献时选择了在肠息肉研究方面比较有名的一篇文章——发表在 2020 年 MICCAI 上的 PraNet。这篇文章比较好的解决了以上两大难题，首先使用并行部分解码器（PPD）聚合高级层中的特征。基于组合的特征，随后生成全局图作为以下组件的初始引导区域。此外，使用反向注意（RA）模块来挖掘边界线索，该模块能够建立区域和边界线索之间的关系。并且该文献给定了比较清晰的源代码与数据集，源代码与数据集地址在 https://github.com/DengPingFan/PraNet

对于这篇论文，我的目标是首先熟练掌握论文本身和论文代码，然后通过 PraNet 网络自身的特点，将其应用到皮肤镜医学图像上，通过对结果的评估，得出 PraNet 是否适配在皮肤镜医学图像上的结论。

2 相关工作

首先，要多次认真阅读论文及实现代码，通过对代码的整理初步了解如何建立模型的框架。然后再通过以往读过的相关论文来优化部分作者的代码，将作者通过 matlab 实现的评估代码统一改为 python 实现，最后将 PraNet 模型适配到皮肤镜医学图像上，通过与常用方法对比和评估程序得出 PraNet 适合应用在皮肤镜医学图像上的结论。

3 本文方法

本文提出了一种新的深度神经网络，称为并行反向注意网络（PraNet），用于息肉分割任务。作者的动机源于这样一个事实，即在息肉注释期间，临床医生首先粗略地定位息肉，然后根据局部特征精确地提取其轮廓掩模。因此，作者认为面积和边界是区分正常组织和息肉的两个关键特征。与其他常见分割模型不同的是，首先预测粗糙区域，然后通过反向注意隐式地对边界进行建模。该策略具有学习能力强、泛化能力强、训练效率高等优点。

3.1 本文方法概述

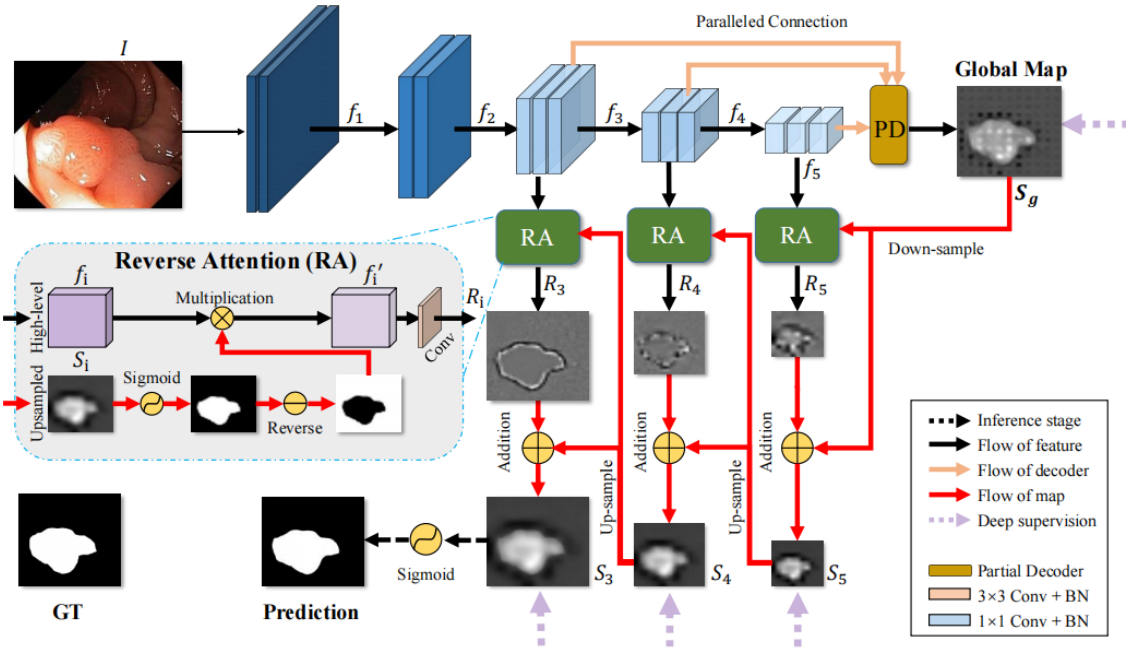

图 1: PraNet 网络

图 1显示了我们的 PraNet，它利用并行部分解码器生成高级语义全局图和一组反向注意模块，用于从结肠镜检查图像中准确分割息肉。接下来将对其各个模块进行详解。

3.2 基于并行部分解码器的特征聚合

当前流行的医学图像分割网络通常依赖于 UNet[1]或 U-Net 类网络 (例如，U-Net++[2]、ResUNet[3]等)。这些模型本质上是编码器-解码器框架，其通常聚集从 CNN 提取的所有多级特征。如 Wu 等人[4]所示，与高级特征相比，低级特征需要更多的计算资源，因为其空间分辨率更大，但对性能的贡献较小。受此启发，作者提出用并行部分解码器组件来聚合高级特征。更具体地，对于大小为 h*w 的输入息肉图像，可以从基于 Res2Net 的骨干网络中提取五个水平的特征分辨率为 [h/2k−1，w/2k−1] 的网络。然后，将这五个特征中的较低两层分为低级特征，并将较高的三层分为高级特征。之后，引入了部分解码器 pd(.)，这是一个新的 SOTA 解码器组件，通过并行连接聚合高级功能来计算部分解码器特征，最终我

们可以获得全局映射 Sg。

3.3 反向注意模块

在临床环境中，医生首先粗略地定位息肉区域，然后仔细地检查局部组织以准确地标记息肉。3.2 获得的的全局图 Sg 来自最深的 CNN 层，它只能捕获息肉组织的相对粗略位置，而没有结构细节（见图 1）。为了解决这一问题，我们提出了一种通过擦除前景对象的方式逐步挖掘区分性息肉区域的原则策略。与常规检测方法中聚合所有层次的特征不同，本文提出在三个并行的高层特征中自适应地学习反向注意. 换句话说，我们的架构可以通过从高级侧输出特征擦除现有估计息肉区域来顺序地挖掘互补区域和细节，其中现有估计从更深层上采样。具体地，我们通过将高级输出特征乘以 (逐元素 $\odot$) 反向关注权重 A_i 来获得输出反向关注特征 R_i，如下：

$$R_i = f_i \bigodot A_i \tag{1}$$

反向注意力权重 A_i 实际上用于计算机视觉团体中的显著对象检测，并且可以公式化为：

$$A_i = \ominus(\sigma(\rho(S_{i+1}))) \tag{2}$$

其中 ρ(.) 表示上采样操作，$\sigma(\bullet)$ 是 Sigmoid 函数，而 (.) 是从矩阵 E 中减去输入的逆操作，其中所有元素都是 1。图 1(RA) 显示了该过程的细节。值得注意的是，反向注意驱动的擦除策略最终可以将不精确、粗糙的估计细化为准确、完整的预测图。

3.4 损失函数

损失函数被定义为 $\iota = \iota^w_{IoU} + \iota^w_{BCE}$，其中 ι^w_{IoU} 和 ι^w_{BCE} 表示全局限制和局部（像素级）限制的加权 IoU 损失和二进制交叉熵（BCE）损失。与分割任务中广泛采用的标准 IoU 损失不同，加权 IoU 损失增加了硬像素的权重以突出其重要性。此外，与标准 BCE 损失相比，ι^w_{BCE} 更关注硬像素，而不是为所有像素分配相等的权重。这些损失的有效性已经在显著对象检测领域中得到验证。因此，PraNet 的总损失可表示为：$\iota_{total} = \iota(G, S^{up}_g) + \sum_{i=3}^{i=5} \iota(G, S^{up}_i)$。

4 复现细节

使用 pytorch 进行 PraNet 模型，将所有的输入都统一调整为 352×352, 采用多尺度的训练策略 [0.75、1、1.25] 而不是数据增大。采用亚当优化算法优化总体参数的学习速率 1e−4。整个网络是一个端到端的方式训练, 经过 32 分钟在 20 次迭代后卷积核大小收敛在 16 左右。

4.1 与已有开源代码对比

源代码中评估方法使用 matlab 编写，现改为用 python 编写，由于两者语法差异比较大，故把源代码和更改后的代码列出。

源代码：

```
clear all;
close all;
clc;
% —- 1. ResultMap Path Setting —-
ResultMapPath = '../results/';
Models = 'PraNet';%'UNet','UNet++','PraNet','SFA';
```

```
modelNum = length(Models);
% —- 2. Ground-truth Datasets Setting —-
DataPath = '../data/TestDataset/';
Datasets = 'CVC-300','CVC-ClinicDB'; %'CVC-ClinicDB', 'CVC-ColonDB','ETIS-LaribPolypDB',
'Kvasir','CVC-300';
% —- 3. Evaluation Results Save Path Setting —-
ResDir = './EvaluateResults/';
ResName='_result.txt'; % You can change the result name.
Thresholds = 1:-1/255:0;
datasetNum = length(Datasets);
for d = 1:datasetNum
tic;
dataset = Datasetsd % print cur dataset name
fprintf('Processing %d/%d: %s Dataset',d,datasetNum,dataset);
ResPath = [ResDir dataset '-mat/']; % The result will be saved in *.mat file so that you can used it for the
next time.
if exist(ResPath,'dir')
mkdir(ResPath);
end
resTxt = [ResDir dataset ResName]; % The evaluation result will be saved in '../Resluts/Result-XXXX'
folder.
fileID = fopen(resTxt,'w');
for m = 1:modelNum
model = Modelsm % print cur model name
gtPath = [DataPath dataset '/masks/'];
resMapPath = [ResultMapPath '/' model '/' dataset '/'];
imgFiles = dir([resMapPath '*.png']);
imgNUM = length(imgFiles);
[threshold_Fmeasure, threshold_Emeasure, threshold_IoU] = deal(zeros(imgNUM,length(Thresholds)));
[threshold_Precion, threshold_Recall] = deal(zeros(imgNUM,length(Thresholds)));
[threshold_Sensitivity, threshold_Specificity, threshold_Dice] = deal(zeros(imgNUM,length(Thresholds)));
[Smeasure, wFmeasure, MAE] =deal(zeros(1,imgNUM));
for i = 1:imgNUM
name = imgFiles(i).name;
fprintf('Evaluating(%s Dataset,%s Model, %s Image): %d/%d',dataset, model, name, i,imgNUM);
%load gt
```

```
gt = imread([gtPath name]);
if (ndims(gt)>2)
gt = rgb2gray(gt);
end
if  islogical(gt)
gt = gt(:,:,1) > 128;
end
%load resMap
resmap = imread([resMapPath name]);
%check size
if size(resmap, 1)  = size(gt, 1) || size(resmap, 2)  = size(gt, 2)
resmap = imresize(resmap,size(gt));
imwrite(resmap,[resMapPath name]);
fprintf('Resizing have been operated!! The resmap size is not math with gt in the path: %s!!!', [resMap-
Path name]);
end
resmap = im2double(resmap(:,:,1));
%normalize resmap to [0, 1]
resmap = reshape(mapminmax(resmap(:)',0,1),size(resmap));
% S-meaure metric published in ICCV'17 (Structure measure: A New Way to Evaluate the Foreground
Map.)
Smeasure(i) = StructureMeasure(resmap,logical(gt));
% Weighted F-measure metric published in CVPR'14 (How to evaluate the foreground maps?)
wFmeasure(i) = original_WFb(resmap,logical(gt));
MAE(i) = mean2(abs(double(logical(gt)) - resmap));
[threshold_E, threshold_F, threshold_Pr, threshold_Rec, threshold_Iou] = deal(zeros(1,length(Thresholds)));
[threshold_Spe, threshold_Dic] = deal(zeros(1,length(Thresholds)));
for t = 1:length(Thresholds)
threshold = Thresholds(t);
[threshold_Pr(t), threshold_Rec(t), threshold_Spe(t), threshold_Dic(t), threshold_F(t), threshold_Iou(t)]
= Fmeasure_calu(resmap,double(gt),size(gt),threshold);
Bi_resmap = zeros(size(resmap));
Bi_resmap(resmap>=threshold)=1;
threshold_E(t) = Enhancedmeasure(Bi_resmap, gt);
end
threshold_Emeasure(i,:) = threshold_E;
```

```
threshold_Fmeasure(i,:) = threshold_F;
threshold_Sensitivity(i,:) = threshold_Rec;
threshold_Specificity(i,:) = threshold_Spe;
threshold_Dice(i,:) = threshold_Dic;
threshold_IoU(i,:) = threshold_Iou;
end
%MAE
mae = mean2(MAE);
%Sm
Sm = mean2(Smeasure);
%wFm
wFm = mean2(wFmeasure);
%E-m
column_E = mean(threshold_Emeasure,1);
meanEm = mean(column_E);
maxEm = max(column_E);
%Sensitivity
column_Sen = mean(threshold_Sensitivity,1);
meanSen = mean(column_Sen);
maxSen = max(column_Sen);
%,Specificity
column_Spe = mean(threshold_Specificity,1);
meanSpe = mean(column_Spe);
maxSpe = max(column_Spe);
%Dice
column_Dic = mean(threshold_Dice,1);
meanDic = mean(column_Dic);
maxDic = max(column_Dic);
%IoU
column_IoU = mean(threshold_IoU,1);
meanIoU = mean(column_IoU);
maxIoU = max(column_IoU);
save([ResPath model],'Sm', 'mae', 'column_Dic', 'column_Sen', 'column_Spe', 'column_E','column_IoU','m
fprintf(fileID, '(Dataset:%s; Model:%s) meanDic:%.3f;meanIoU:%.3f;wFm:%.3f;Sm:%.3f;meanEm:%.3f;MA
fprintf('(Dataset:%s; Model:%s) meanDic:%.3f;meanIoU:%.3f;wFm:%.3f;Sm:%.3f;meanEm:%.3f;MAE:%.3f
end
```

toc;

end

改为 python 后的代码：

```
import os
import argparse
import tqdm
import sys
import numpy as np
from PIL import Image
from tabulate import tabulate
filepath = os.path.split(os.path.abspath(__file__))[0]
repopath = os.path.split(filepath)[0]
sys.path.append(repopath)
from utils.eval_functions import *
from utils.utils import *
def evaluate(opt, args):
if os.path.isdir(opt.Eval.result_path) is False:
os.makedirs(opt.Eval.result_path)
method = os.path.split(opt.Eval.pred_root)[-1]
Thresholds = np.linspace(1, 0, 256)
headers = opt.Eval.metrics #['meanDic', 'meanIoU', 'wFm', 'Sm', 'meanEm', 'mae', 'maxEm', 'maxDic',
'maxIoU', 'meanSen', 'maxSen', 'meanSpe', 'maxSpe']
results = []
if args.verbose is True:
print('' * 20, 'Start Evaluation', '' * 20)
datasets = tqdm.tqdm(opt.Eval.datasets, desc='Expr - ' + method, total=len(
opt.Eval.datasets), position=0, bar_format='desc:<30percentage:3.0f%|bar:50r_bar')else : datasets =
opt.Eval.datasets
for dataset in datasets:
pred_root = os.path.join(opt.Eval.pred_root, dataset)
gt_root = os.path.join(opt.Eval.gt_root, dataset, 'masks')
preds = os.listdir(pred_root)
gts = os.listdir(gt_root)
preds.sort()
gts.sort()
threshold_Fmeasure = np.zeros((len(preds), len(Thresholds)))
```

```
threshold_Emeasure = np.zeros((len(preds), len(Thresholds)))
threshold_IoU = np.zeros((len(preds), len(Thresholds)))
# threshold_Precision = np.zeros((len(preds), len(Thresholds)))
# threshold_Recall = np.zeros((len(preds), len(Thresholds)))
threshold_Sensitivity = np.zeros((len(preds), len(Thresholds)))
threshold_Specificity = np.zeros((len(preds), len(Thresholds)))
threshold_Dice = np.zeros((len(preds), len(Thresholds)))
Smeasure = np.zeros(len(preds))
wFmeasure = np.zeros(len(preds))
MAE = np.zeros(len(preds))
if args.verbose is True:
samples = tqdm.tqdm(enumerate(zip(preds, gts)), desc=dataset + ' - Evaluation', total=len( preds), position=1, leave=False, bar_format='desc:<30percentage:3.0f|bar:50r_bar')
else:
samples = enumerate(zip(preds, gts))
for i, sample in samples:
pred, gt = sample
assert os.path.splitext(pred)[0] == os.path.splitext(gt)[0]
pred_mask = np.array(Image.open(os.path.join(pred_root, pred)))
gt_mask = np.array(Image.open(os.path.join(gt_root, gt)))
if len(pred_mask.shape) != 2:
pred_mask = pred_mask[:, :, 0]
if len(gt_mask.shape) != 2:
gt_mask = gt_mask[:, :, 0]
assert pred_mask.shape == gt_mask.shape
gt_mask = gt_mask.astype(np.float64) / 255
gt_mask = (gt_mask > 0.5).astype(np.float64)
pred_mask = pred_mask.astype(np.float64) / 255
Smeasure[i] = StructureMeasure(pred_mask, gt_mask)
wFmeasure[i] = original_WFb(pred_mask, gt_mask)
MAE[i] = np.mean(np.abs(gt_mask - pred_mask))
threshold_E = np.zeros(len(Thresholds))
threshold_F = np.zeros(len(Thresholds))
threshold_Pr = np.zeros(len(Thresholds))
threshold_Rec = np.zeros(len(Thresholds))
threshold_Iou = np.zeros(len(Thresholds))
```

```
        threshold_Spe = np.zeros(len(Thresholds))
        threshold_Dic = np.zeros(len(Thresholds))
        for j, threshold in enumerate(Thresholds):
        threshold_Pr[j], threshold_Rec[j], threshold_Spe[j], threshold_Dic[j], threshold_F[j],
        threshold_Iou[j] = Fmeasure_calu(pred_mask, gt_mask, threshold)
        Bi_pred = np.zeros_like(pred_mask)
        Bi_pred[pred_mask >= threshold] = 1
        threshold_E[j] = EnhancedMeasure(Bi_pred, gt_mask)
        threshold_Emeasure[i, :] = threshold_E
        threshold_Fmeasure[i, :] = threshold_F
        threshold_Sensitivity[i, :] = threshold_Rec
        threshold_Specificity[i, :] = threshold_Spe
        threshold_Dice[i, :] = threshold_Dic
        threshold_IoU[i, :] = threshold_Iou
        result = []
        mae = np.mean(MAE)
        Sm = np.mean(Smeasure)
        wFm = np.mean(wFmeasure)
        column_E = np.mean(threshold_Emeasure, axis=0)
        meanEm = np.mean(column_E)
        maxEm = np.max(column_E)
        column_Sen = np.mean(threshold_Sensitivity, axis=0)
        meanSen = np.mean(column_Sen)
        maxSen = np.max(column_Sen)
        column_Spe = np.mean(threshold_Specificity, axis=0)
        meanSpe = np.mean(column_Spe)
        maxSpe = np.max(column_Spe)
        column_Dic = np.mean(threshold_Dice, axis=0)
        meanDic = np.mean(column_Dic)
        maxDic = np.max(column_Dic)
        column_IoU = np.mean(threshold_IoU, axis=0)
        meanIoU = np.mean(column_IoU)
        maxIoU = np.max(column_IoU)
        result.extend([meanDic, meanIoU, wFm, Sm, meanEm, mae, maxEm, maxDic, maxIoU, meanSen, maxSen,
meanSpe, maxSpe])
        results.append([dataset, *result])
```

```
out = []
for metric in opt.Eval.metrics:
out.append(eval(metric))
result.extend(out)
results.append([dataset, *result])
csv = os.path.join(opt.Eval.result_path, 'result_' + dataset + '.csv')
if os.path.isfile(csv) is True:
csv = open(csv, 'a')
else:
csv = open(csv, 'w')
csv.write(', '.join(['method', *headers]) + '')
out_str = method + ','
for metric in result:
out_str += ':.4f'.format(metric) + ','
out_str += ''
csv.write(out_str)
csv.close()
tab = tabulate(results, headers=['dataset', *headers], floatfmt='.3f')
if args.verbose is True:
print(tab)
print('#'*20, 'End Evaluation', '#'*20)
return tab
if __name__ == '__main__':
args = parse_args()
opt = load_config(a.config)
evaluate(opt, args)
```

5 实验结果分析

实验结果分为对原肠息肉图像的实验结果和对皮肤镜医学图像的实验结果两部分。

5.1 肠息肉图像实验结果

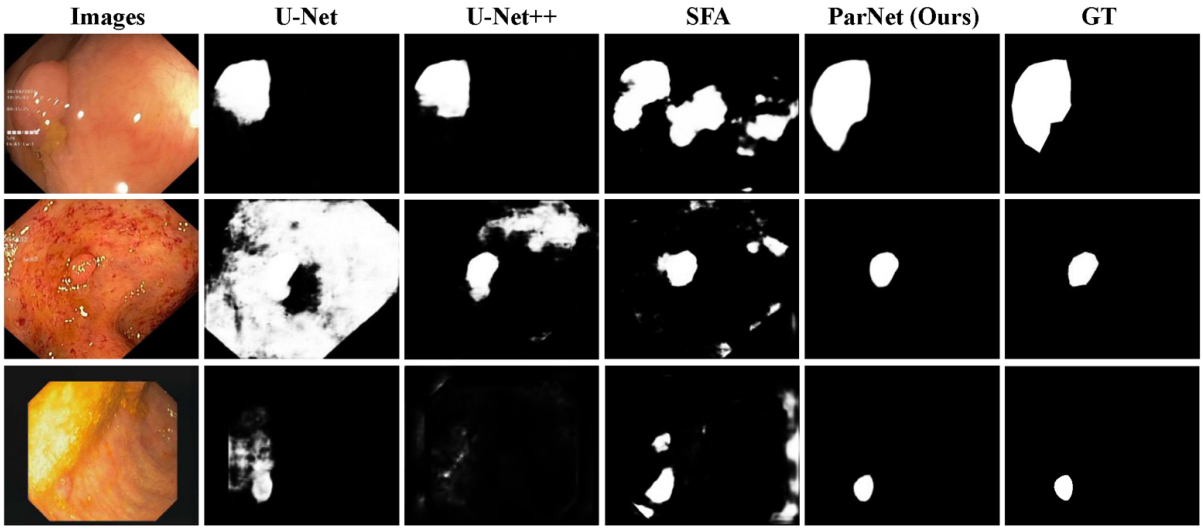

图 2: 原文结果示意

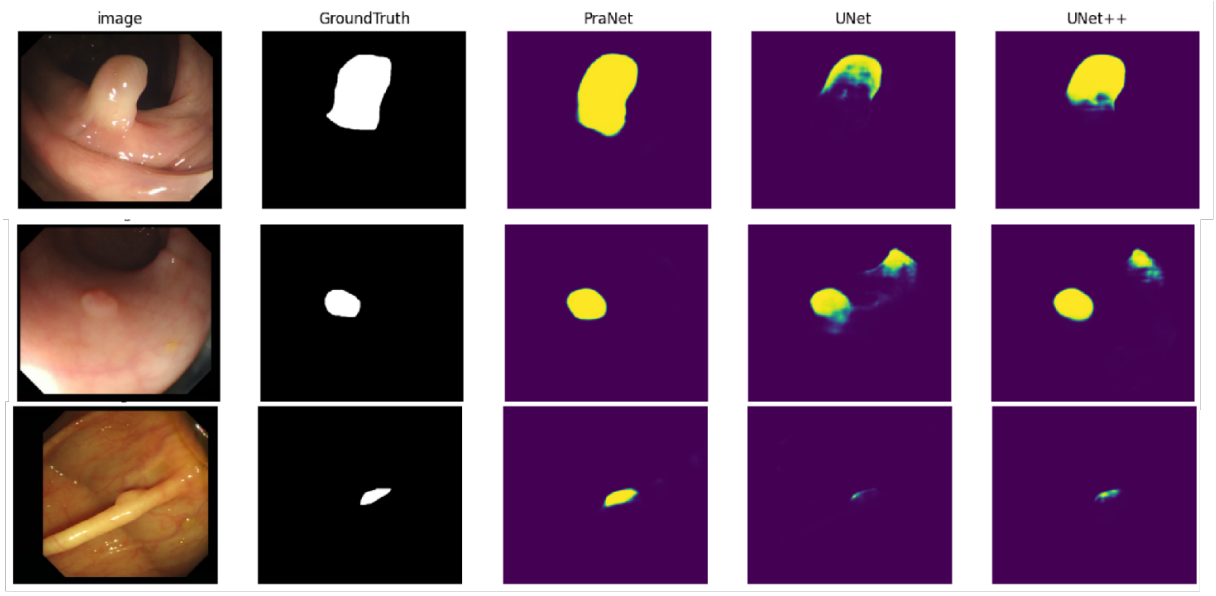

图 3: 我的结果示意

dataset	meanDic	meanIoU	wFm	Sm	meanEm	mae	maxEm	maxDic	maxIoU	meanSen
CVC-300	0.910	0.849	0.901	0.937	0.977	0.005	0.980	0.913	0.853	0.940
CVC-ClinicDB	0.926	0.880	0.928	0.943	0.974	0.006	0.976	0.929	0.883	0.943
Kvasir	0.912	0.859	0.902	0.917	0.955	0.025	0.958	0.915	0.862	0.923
CVC-ColonDB	0.751	0.678	0.746	0.835	0.875	0.039	0.878	0.753	0.680	0.754
ETIS-LaribPolypDB	0.766	0.689	0.740	0.859	0.903	0.012	0.905	0.769	0.691	0.813

图 4: 定量结果

从图 2，图 3 可以比较明显的看出不管是我训练的结果还是原文，PraNet 相较于 UNet 和 UNet++ 都具有更为优秀的结果，在三组类型不同的肠息肉图片中，PraNet 模型都能较为清晰地识别出肠息肉的位置。而从图 4 的定量结果中，meanDic 一栏中，可以看出在五组不同的数据集中，PraNet 都有着不错的性能，说明其泛用性也挺不错的。

5.2 皮肤镜医学图像实验结果

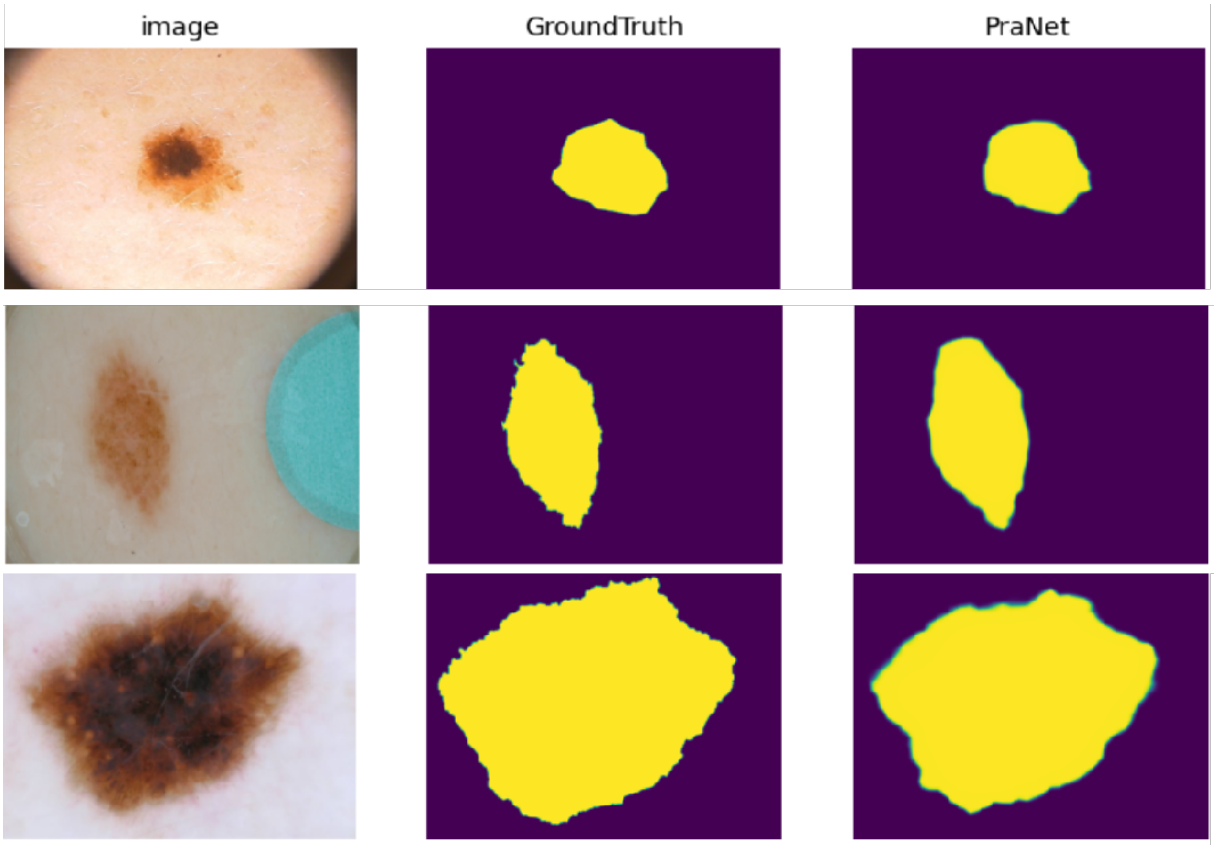

图 5: 将模型用于皮肤镜医学图像结果

图 5 是我将模型用于皮肤镜医学图像后，模型得出的结果，可以看出在针对不同特征的皮肤镜片时，PraNet 也能得出相对不错的结果，但由于我的数据集数量还不够多，定量结果和真实有比较大偏差，故没有给出，我下一步将收集更多皮肤镜片数据，再来测试 PraNet 是否适用于皮肤镜医学图像。

6 总结与展望

本文首先通过仔细阅读复现论文了解该论文的基本思想和特色的方法，再依据这些思想和方法对该论文进行复现。主要的代码量在于将原来由 matlab 编写的评估程序改为用 pytorch 实现。并且由于 python 这几年的更新，在一些过时代码处也通过询问学长学姐、上网查找解决方案等方式进行了更改，使代码在服务器上能够成功运行。本文不仅成功实现在肠息肉医学图像上，还通过作者创作网络的思想，将其运用在同样特征的皮肤镜医学图像上。目前，在我的数据集中表现不错。不足之处在于还未在其他数据集中进行尝试，这也将成为我未来的研究方向。

参考文献

[1] RONNEBERGER O, FISCHER P, BROX T. U-Net: Convolutional Networks for Biomedical Image Segmentation[J]. CoRR, 2015, abs/1505.04597.

[2] ZHOU Z, SIDDIQUEE M M R, TAJBAKHSH N, et al. UNet++: A Nested U-Net Architecture for Medical Image Segmentation[J]. CoRR, 2018, abs/1807.10165.

[3] ZHANG Z, LIU Q, WANG Y. Road Extraction by Deep Residual U-Net[J]. CoRR, 2017, abs/1711.10684.

[4] WU Z, SU L, HUANG Q. Cascaded Partial Decoder for Fast and Accurate Salient Object Detection[J].

CoRR, 2019, abs/1904.08739.