# CCFBDCI自验报告

姓名： 陈远腾 赵昱杰 谷朝阳 彭睿思 游昆霖

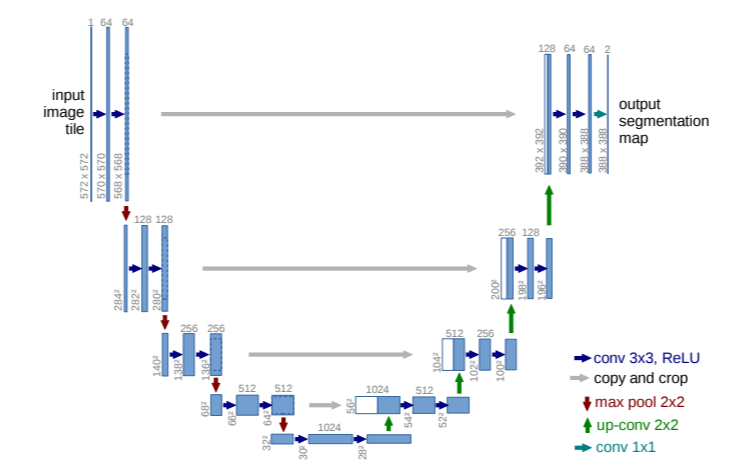
单位： 中国科学院大学

邮箱： chenyuanteng20@ucas.ac.cn

# 所使用的算法简介

UNet属于FCN的一种变体，它可以说是最常用、最简单的一种分割模型，它简单、高效、易懂、容易构建，且可以从小[数据集](https://so.csdn.net/so/search?q=%E6%95%B0%E6%8D%AE%E9%9B%86&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_41731861/article/details/_blank)中训练。于2015年在论文*[U-Net: Convolutional Networks for Biomedical Image Segmentation](https://arxiv.org/abs/1505.04597)*中首次被提出。UNet 的初衷是为了解决医学[图像分割](https://so.csdn.net/so/search?q=%E5%9B%BE%E5%83%8F%E5%88%86%E5%89%B2&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_41731861/article/details/_blank)的问题，在解决细胞层面的分割的任务方面获得了非常优秀的结果，之后，UNet 凭借其突出的分割效果而被广泛应用在语义分割的各个方向（如卫星图像分割，工业瑕疵检测等）。

## 网络模型结构简介：



UNet 网络结构如上图所示，其网络结构是对称的，形似英文字母 U，故而被称为 UNet 。就整体而言，UNet 是一个Encoder-Decoder的结构（与 FCN 相同），前半部分是特征提取，后半部分是上采样。

Encoder 由卷积操作和下采样操作组成，所用卷积结构统一为 3x3 的卷积核，padding=0 ，striding=1。没有 padding 所以每次卷积之后特征图的 H 和 W 变小了，在跳层连接（Skip connection）时需注意特征图的维度。

Decoder 用以恢复特征图的原始分辨率，除了卷积以外，该过程的关键步骤就是上采样与跳层连接。上采样常用转置卷积和插值两种方式实现。在插值实现方式中，双线性插值（bilinear）的综合表现较好也较为常见 。要想网络获得好的效果，跳层连接基本必不可少。UNet 中的跳层连接通过拼接将底层的位置信息与深层的语义信息相融合。需要注意的是，在 FCN 中，特征图是以相加的方式进行融合，而UNet通过通道数的拼接，以形成更厚的特征，保留了更多的维度/位置信息，这使得后面的网络层可在浅层特征与深层特征间自由选择，这对语义分割任务来说更具优势。

总的来说，UNet的优点主要有：

（1）深/浅层特征有着各自意义：网络越深，感受野越大，网络关注那些全局特征（更抽象、更本质）；浅层网络则更加关注纹理等局部特征特征。

（2）通过特征拼接来实现边缘特征的找回。通过上采样（转置卷积）固然能够得到更大尺寸的特征图，但特征图的边缘是缺少信息的。毕竟每一次下采样提取特征的同时，必然会损失一些边缘特征，而上采样并不能找回这些失去的特征。

## 数据集：

所使用的数据集是2010年至2018年期间在Minnesota大学医学中心接受部分或根治性肾切除术治疗的肾肿瘤患者的术前动脉期腹部CT扫描。图像和真实语义标签以匿名NIFTI格式提供，形状为（num\_slices, height, width）。其中num\_slices对应于轴向视图，并且随着切片索引的增加从上到下移动。在所有情况下，患者在图像采集期间都是仰卧的，因此height-width的起点位于患者的左前部。切片厚度范围为1毫米至5毫米。

CT图像和分割标签的数据格式分别为float32和uint8，在分割标签中0代表背景区域、1代表肾脏区域、2代表肿瘤区域。

## 代码提交地址：

https://gitee.com/mingzwhy/UCAS\_lumor\_segementation

## 其它：

本次提交的代码，支持在GPU环境下的训练，代码中包含nii医学数据的处理、Unet模型的搭建、训练过程、及对无标签图像的预测和结果的保存。

# 代码目录结构说明

├── Unet\_model

├── default13217197自验报告 //自验报告

├── README.md //说明文档

├── main.py //主函数

├── dataset.py //数据集处理函数

├── Unet\_mode.py //Unet模型搭建

├── train.py //训练脚本

├── predict.py //评估脚本

# 自验结果

## 自验环境：

华为云modelarts平台，Ascend910 mindspore1.0 和GPU V100 mindspore1.1，以及自己台式机的2080ti GPU（mindspore1.1）环境。

## 训练超参数：

文件目录可通过main.py中的参数进行修改，训练参数可通过train.py中的参数进行修改。

1. main.py中的参数如下：

Part1 路径参数

\

|-- dir\_path: 根目录。

|-- load\_nii\_dir\_path: 数据集所在文件夹路径。

|-- save\_image\_dir\_path: 处理后的原图像的保存目录。

|-- save\_segemen\_dir\_path: 处理后的标签图像的保存目录。

|-- save\_model\_dir\_path: 训练好的模型的参数保存目录。

|-- save\_evaluate\_image: 对无标签图像预测结果的保存路径。

Part2 其他参数

\

|-- if\_save\_weights: 是否保存训练好的模型参数。

|-- learn\_rate: 学习速率，默认为0.0001。

|-- epochs: 训练轮数，默认为20。

1. train.py中的参数如下：

Part1 训练参数

\

|-- height: 训练图像裁剪高度。

|-- width: 训练图像裁剪宽度。

|-- train\_ratio: 训练集所占比例，默认为0.8。

|-- batch\_size: 批训练长度，默认为8。

|-- shuffle\_size: 批打乱长度，默认为200。

|-- learn\_rate: 学习速率，默认为0.0001。

|-- epochs： 训练轮数，默认为20.

## 训练：

模型训练

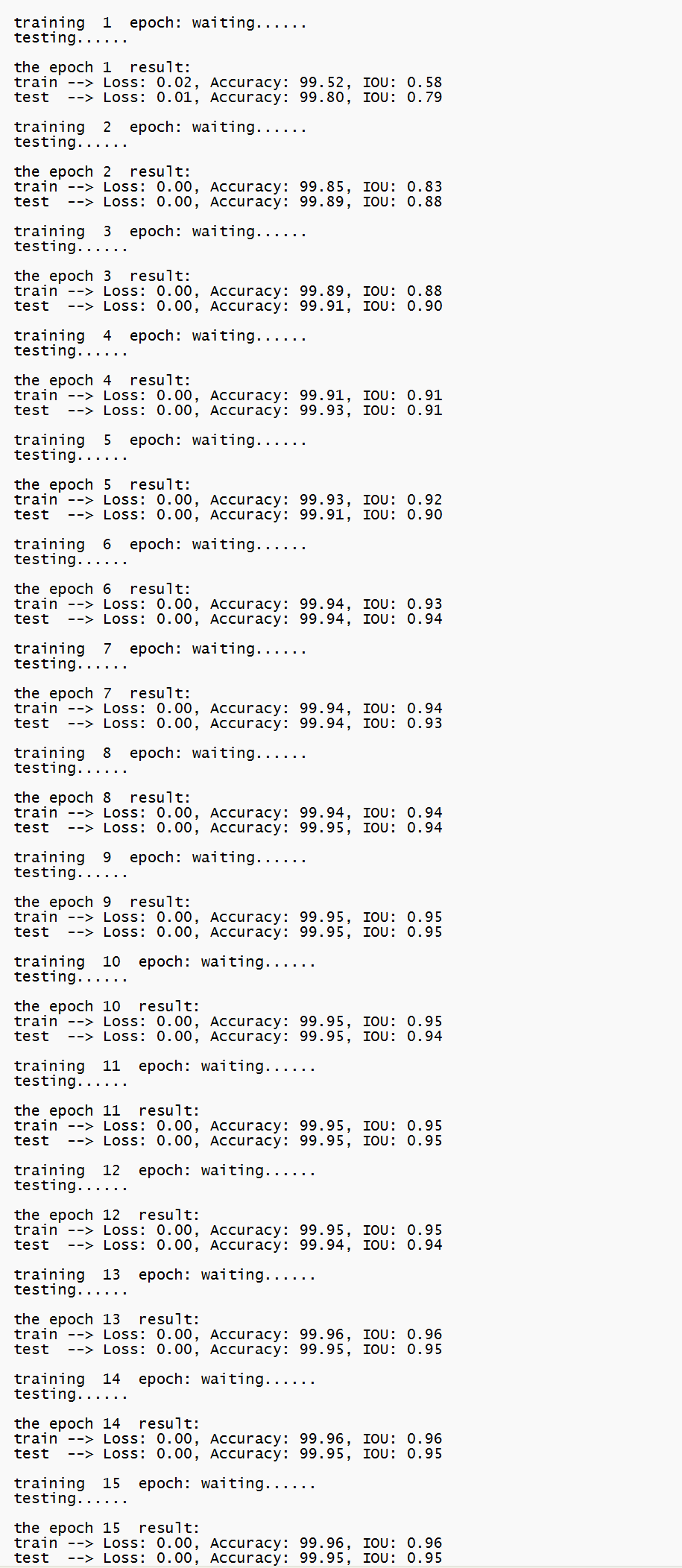
### 如何启动训练脚本：

GPU训练：

python main.py --nii\_data\_dir\_path your\_path --if\_save\_weights True --learn\_rate 0.0001 --train\_epochs 20

### 训练精度结果：

以下结果为在我自己的笔记本上用RTX 3060显卡以 [256,256] 的裁剪尺寸，0.0001的学习速率，BATCH\_SIZE=8, SHUFFLE\_SIZE=200，训练15个epochs得到的结果：



# 参考资料

## 参考论文：

|  |  |
| --- | --- |
|  | **Net: Convolutional Networks** **for Biomedical Image Segmentation** [Olaf Ronneberger](https://arxiv.org/search/cs?searchtype=author&query=Ronneberger,+O), [Philipp Fischer](https://arxiv.org/search/cs?searchtype=author&query=Fischer,+P), [Thomas Brox](https://arxiv.org/search/cs?searchtype=author&query=Brox,+T)  https://arxiv.org/abs/1505.04597 |

## 参考资料：

UNet浅析 ：

https://blog.csdn.net/qq\_41731861/article/details/120528269?spm=1001.2101.3001.6650.2&utm\_medium=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-2-120528269-blog-114901600.pc\_relevant\_vip\_default&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-2-120528269-blog-114901600.pc\_relevant\_vip\_default&utm\_relevant\_index=3