**通信系统综合设计**

**技术报告**



项目名称： 肺炎检测助手

团队成员： 李鑫、李明晓、李欣竹、王籽予

指导老师： 郑来波

完成时间： 2023.03.28

目录

[1项目说明 3](#_Toc26874)

[2 项目原理与解决方案 4](#_Toc31749)

[2.1 数据集说明 4](#_Toc25291)

[2.2 图像识别 4](#_Toc1589)

[2.2.2 分类网络原理介绍 4](#_Toc11894)

[2.2.2.1 神经网络 4](#_Toc28963)

[2.2.2.2 VGGNet 5](#_Toc24061)

[2.3病灶分割 7](#_Toc831)

[2.2.3 BiSeNet V2原理介绍 7](#_Toc28855)

[2.3 小样本学习+孪生网络 9](#_Toc28292)

[3代码结构 10](#_Toc18609)

[3.1 Pyqt界面代码 10](#_Toc3293)

[3.2 VGG分类模型代码 10](#_Toc8179)

[3.2.1 环境配置 10](#_Toc212)

[3.2.2 VGGNet代码架构 11](#_Toc9463)

[3.2.3 VGGNet文件说明 11](#_Toc25774)

[3.3BiSeNet V2分割模型代码 11](#_Toc24536)

[3.3.1 环境配置 11](#_Toc17242)

[3.3.2 BiSeNet V2代码架构 12](#_Toc19974)

[3.3.3 BiSeNet V2文件说明 12](#_Toc12764)

[4 使用方法 13](#_Toc20605)

[总结反思： 22](#_Toc21792)

[附录 24](#_Toc10451)

[分类模型代码 24](#_Toc16217)

[分割模型代码 36](#_Toc25497)

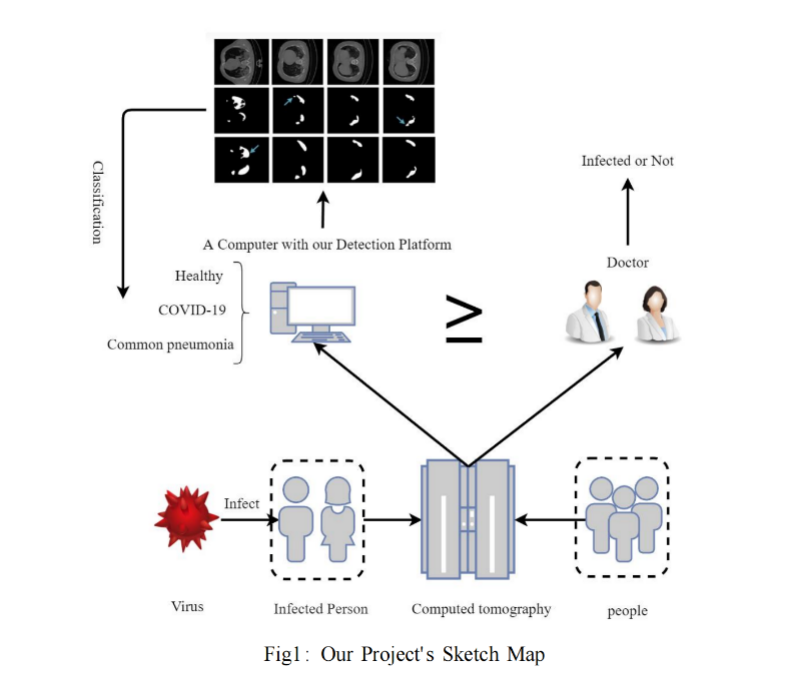
[PyQt代码 36](#_Toc10580)

# 1项目说明

新冠疫情爆发期间，核酸检测是最方便、最流行的检测手段，但是经常出现阳性被误诊为阴性的情况，这加大了阻断疫情传播工作的难度。

对此，我们团队认为，肺部X光和CT影像分析可以作为一种新的新冠检测手段，辅助核酸检测。考虑到传统的以医院为基础的医生分析检测方法，难以满足大量待检肺部CT图像的情况，我们团队致力于开发一个基于OpenCV和深度学习的肺炎检测平台，利用图像处理算法和Pyqt，对肺部影像进行分析处理。该平台使用Pyqt开发，数据集来源于网络上开源的肺炎X光数据集和肺部病灶分割数据集。

本项目通过图像识别和图像分割算法，代替传统的人工检测方式，最终实现了对正常肺部X光、病毒性肺炎X光、新冠肺炎X光进行分类和对肺部CT进行病灶分析，可以很好地作为医生诊断的辅助手段。



图一：项目流程图

# 项目原理与解决方案

## 2.1 数据集说明

在本项目中我们使用网络上开源的新冠数据集，其中分类数据集包括主流的数据集chest-x-rey数据集，其中包括2000张左右的正常图像和2000张左右的病毒性肺炎图像，还有200张左右的新冠肺炎图像，数据存在明显的长尾分布情况，我们采用了数据增强、预训练加微调的方式进行了基于该数据集的训练和测试，最终效果最好的mAP大约为98.47%。

分割数据集采用的是MosMedData包含了20多个病人的肺部CT文件，格式为nii，我们使用的BiSeNet效果并不是很好，主要是因为目前对于CT数据的分割也没有很好的方式，加之我们的算力平台算力不够，所以我们最终选择了精度和速度之间的中和—BiSeNetV2,dice大约为0.65左右。

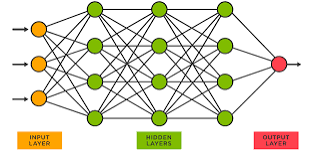
## 2.2 图像识别

本项目通过使用百度PaddlePaddle深度学习框架，基于网络上开源的肺炎X光影像数据集，搭建了计算机视觉中的经典模型VGGNet，并通过VGGNet对肺炎X光影像进行识别。通过训练VGGNet，神经网络在测试集上面最终得到了96%的准确率，之后通过导出ONNX模型，并将其部署到C#窗体程序中，最终实现了肺炎检测平台上的X光识别模块。

### **2.2.2 分类网络原理介绍**

#### **2.2.2.1 神经网络**

人工神经网络是一个能够学习，能够总结归纳的系统，也就是说它能够通过已知数据的实验运用来学习和归纳总结。人工神经网络通过对局部情况的对照比较（而这些比较是基于不同情况下的自动学习和要实际解决问题的复杂性所决定的），它能够推理产生一个可以自动识别的系统。[1]



图二：人工神经网络

#### 2.2.2.2 VGGNet

VGGNet是由牛津大学的计算机视觉小组在2014年ILSVRC中提出来的网络结构，在2014年的ILSVRC比赛中，VGG 在Top-5中取得了92.3%的正确率，其提出的许多网络设计理念对后续神经网络的结构变迁产生了极大的影像，并且仍然活跃于许多迁移任务中。

论文Very Deep Convolutional Networks For Large-Scale Image Recognition[2]主要提出了以下几点创新：

使用非常小的3×3卷积核来增加网络的深度进而对网络的性能进行研究，基于此方法可以将网络的深度推广到16-19层。[2]

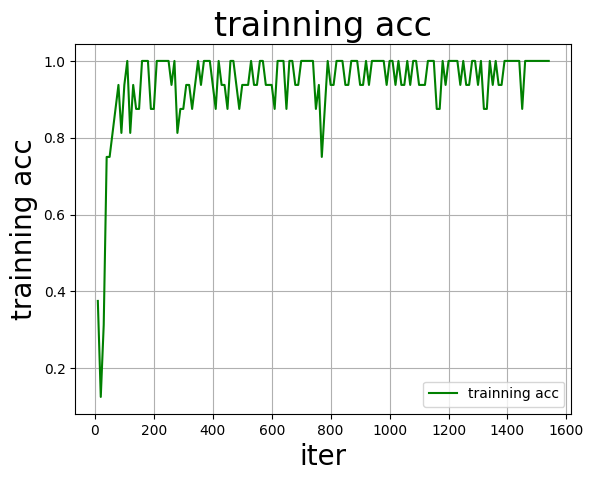
3×3的卷积核可以通过堆叠增加其感受野，两个3×3等同于一个5×5的感受野，三个等同于一个7×7卷积核的感受野。多个小卷积核的好处就是相对于一个大卷积核在其后面增加的多个非线性激活函数可以增强网络的非线性，同时也可以降低参数。[2]

1×1卷积核的使用：增加非线性，1×1卷积核本质上是对输入做了一个线性变换，之后经过了非线性的激活函数，因而增加了网络的非线性。[2]



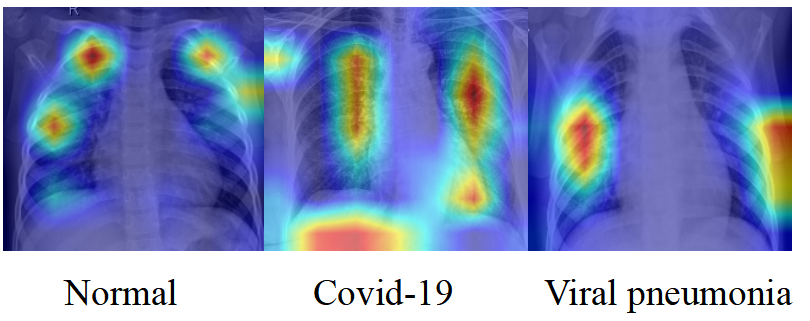
图三：VGGNet网络结构图

在本项目中，我们对比使用了VGGNet、ResNet18[3]、MobileNet[4]以及2020年爆火的ViT[5]模型和2022年CVPR的Swin Transformer，发现VGG在该数据集上表现最佳，ResNet18的效果略差于VGGNet，XceptionNet由于轻量化设计，减少了许多参数，但是对应地，精度也有所下降，ViT的效果只能达到92%左右，从实验结果上也证明了ViT对比卷积模型拥有更小的归纳偏置[5]，在小数据集上的表现不佳。

图四：VGGNet训练监视图

我们通过使用Grad-CAM方法，获得了VGGNet下的类别激活图可以看出网络对于不同的肺炎种类的关注区域差距较大。



图五：类别激活图

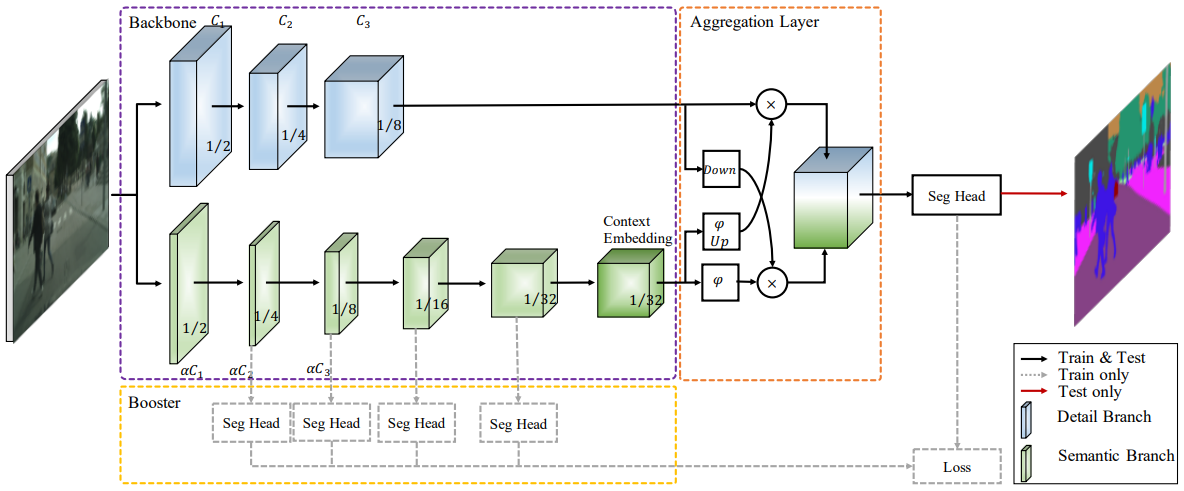
## 2.3病灶分割

CT影像技术在医学影像检测、分割、分类等诊断任务扮演着至关重要的角色, 同时对于医生的诊断提供了证明与依据。尤其是在新型冠状病毒肺炎（COVID-19）近三年的流行期间，针对CT影像的深度学习图像处理与分析，很大程度上对于医生的决策起着不可忽视的辅助作用。因此能实现CT影像中的病灶分割，对医生的精确诊断与病人的尽早治疗提供更加高效的途径。但是新冠肺炎的病灶分割存在着一定的挑战，因为其早期呈现为磨玻璃样，边界与周围的组织器官边界模糊，存在结节状、斑片状或片状磨玻璃样密度影，病灶可相对局限或弥漫。[6]

本项目主要针对新冠肺炎的肺部CT，使用PaddlePaddle深度学习框架，基于开源的新冠肺炎CT影像数据集，考虑项目的通常部署场景，搭建了计算机视觉中的轻量模型BiSeNetV2，并通过BiSeNetV2对新冠肺炎CT影像中的病灶部分进行精准分割，并且在数据预处理与后处理阶段使用OpenCV计算机视觉库。最后采用了PyQt框架，实现了检测平台的开发与部署，但是考虑到医学影像数据一般是Nifit等三维数据，而训练过程中使用的是2D数据。因此在对影像数据进行推理时，是采用分层推理的方式，再堆叠结果进行层层展示，并在原图在病灶位置处绘制轮廓，最终实现了新冠肺炎检测平台上的病灶分割模块。

### **2.2.3 BiSeNet V2原理介绍**

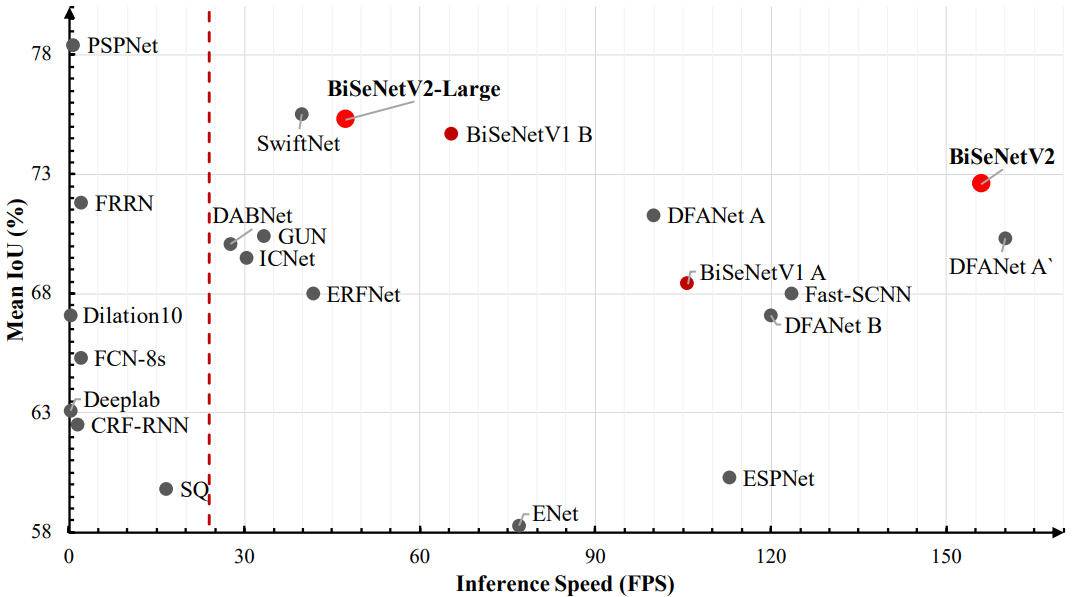
BiSeNetV2[7]是由华中科技大学计算机视觉小组提出来的一种在速度和准确性之间进行良好权衡的双边分割网络（BiSeNet V2）。其核心思想是，浅层细节（low-level details）和深层语义（high-level semantics）对语义分割都非常重要，可以考虑将空间细节和类别语义分开处理，以实现精度和推理速度之间的平衡。模型的架构图如下：



图六：BiSeNet V2网络结构图

其中Detail Branch的任务是保存低级特征的空间细节信息，所以该分支的设计理念就是通道多，层数少，下采样比率低。为了防止模型速度降低，没有采用residual connection[3]。Semantic Branch的目标是捕获高级的语义信息，所示他的设计理念是通道少，层数多，下采样比率大（增加感受野）。

架构设计概括为：细节分支（Detail Branch）被设计为宽通道、层数较浅，用于捕捉空间细节；语义分支（Semantic Branch）通道较窄、层数较深，用于提取类别语义。此外，作者设计了一个引导聚合层(Guided Aggregation Layer)，以增强双向连接并融合这两种类型的特征表示。还设计了一种增强训练策略来提高分割性能，而不需要任何额外的推理成本。



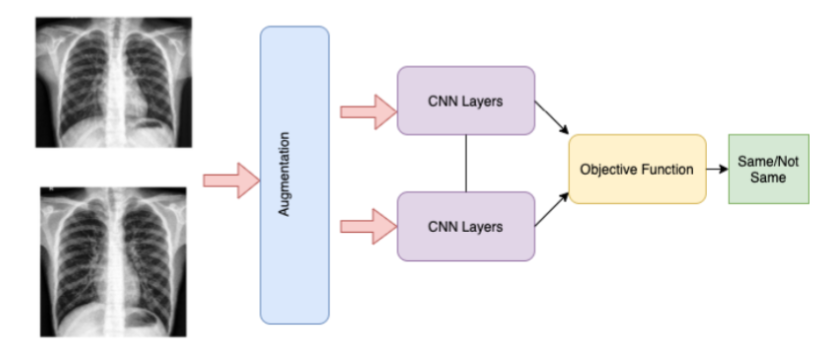
图七：Cityscapes测试集下性能测试

与几种最新的实时语义分割方法相比，BiSeNet V2具有良好的性能：模型的推理速度比现有方法快得多的同时，可以实现更好的分割精度。

我们在实现CT影像分割时，在UNet，BiSeNet V2等模型中，综合考虑分割的精度与部署的场景，选择BiSeNet V2作为最终的系统模型。

## 小样本学习+孪生网络

在本项目中我们也实现了2021年的文章COVID-19 detection from scarce chest x-ray image data using deep learning中使用DenseNet121作为孪生网络的Backbone 以及迁移学习的代码，同时也通过无监督学习等方式对该小样本数据集进行了处理，实验复现了文章中关于不同方式测得的准确率精度排序。



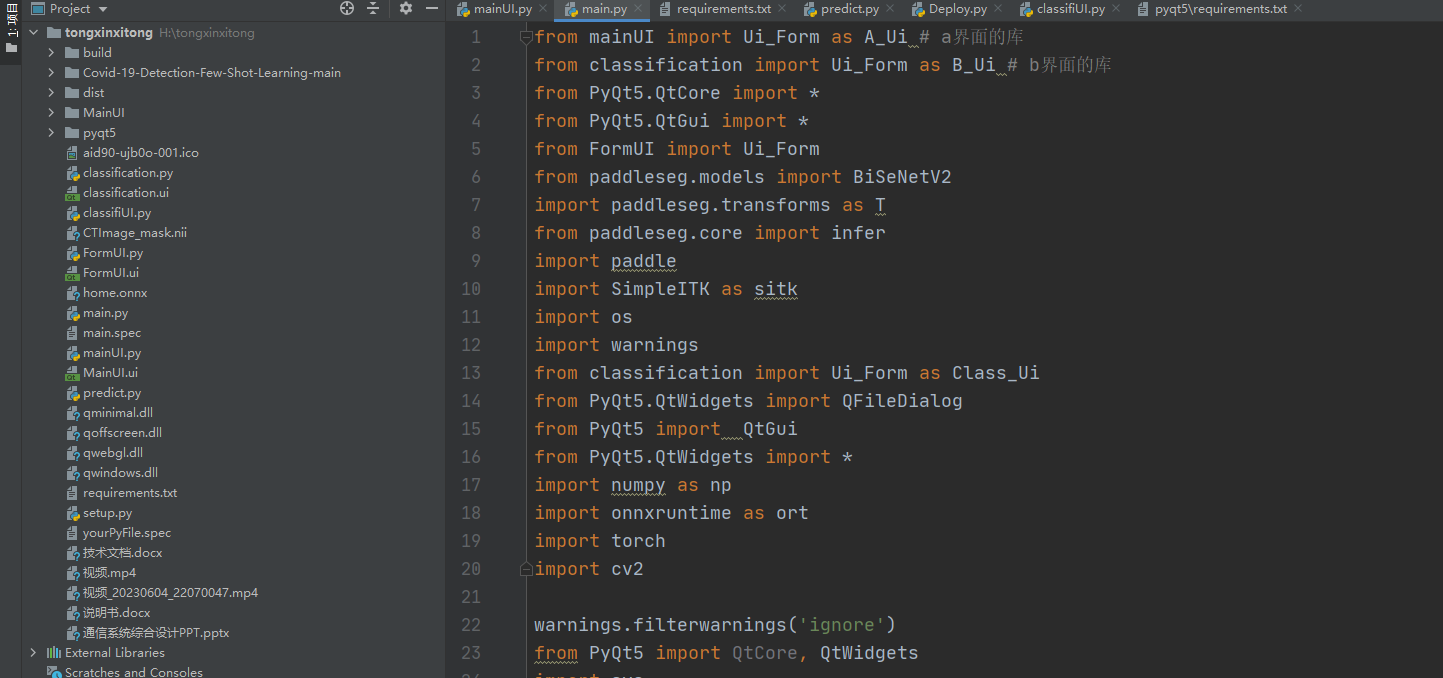
图八：孪生网络+迁移学习

# **3**代码结构

## 3.1 Pyqt界面代码

我们通过使用QT编写了三个任务的界面，之后放在Main.py文件中进行调用其设计得到的库，通过事件绑定函数的面向对象编程的思路完成了界面的编写。

其中我们通过创建新的线程来合理分配CPU等资源的调度，这样可以增大系统资源的利用率，同时也可以界面上有关图像或组件的使用更为流畅。



图九：PyQt主代码

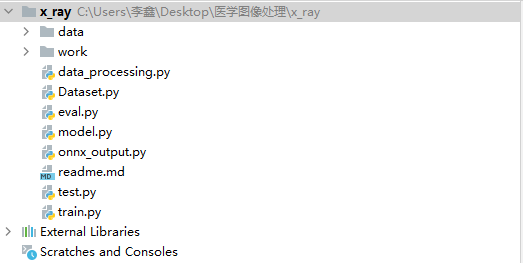
## 3.2 VGG分类模型代码

### **3.2.1 环境配置**

表一：分类模型环境要求

|  |  |
| --- | --- |
| 库名 | 版本 |
| python | 3.7 |
| paddle | 2.2.2 |
| json5 | 0.9.5 |
| matplotlib | 2.2.3 |
| numpy | 1.21.6 |
| Pillow | 8.2.0 |

### **3.2.2 VGGNet代码架构**



图十：VGGNet分类代码架构

### **3.2.3 VGGNet文件说明**

train.py:训练脚本

test.py:训练脚本

eval.py:评估脚本

model.py:定义网络结构

data\_processing.py:解压数据、划分数据集

Dataset.py:继承paddle的dataset,适配本数据集。

onnx\_output.py:输出onnx模型

work/checkpoints：存放训练后参数

data/dataset:存放数据集的目录

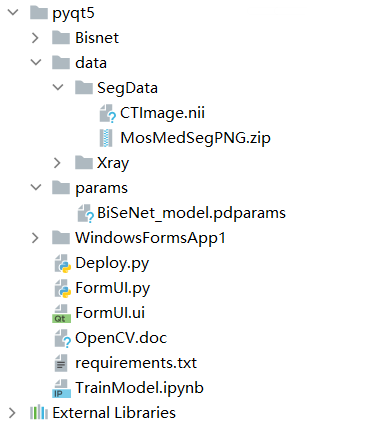
## **3.3**BiSeNet V2分割模型代码

### **3.3.1 环境配置**

表二：分割模型环境要求

|  |  |
| --- | --- |
| 库名 | 版本 |
| python | 3.7 |
| numpy | 1.21.2 |
| opencv-python | 4.5.3.56 |
| paddlepaddle | 2.1.3 |
| paddleseg | 2.3.0 |
| PyQt5 | 5.15.6 |
| SimpleITK | 2.1.1 |

### **3.3.2 BiSeNet V2代码架构**



图十一：BiSeNet V2分割代码架构

### **3.3.3 BiSeNet V2文件说明**

CTImage：CT影像文件

BiSeNet\_model.pdparams：BiSeNet V2模型的训练参数

Deploy.py：具体部署与系统的实现

FormUI.py：窗体的转换py文件

FormUI.ui：窗体的UI文件

requirements.txt：项目所需的环境依赖包以及对应版本

TrainModel.ipynb：模型训练文件，采用paddle平台训练

MosMedSegPNG.zip：训练采用的数据集

# 4 使用方法

进入程序主界面，中间为项目的主要功能区，分别是X光胸片分析和肺部CT分析，X光胸片分析模块通过C#部署VGG模型的ONNX文件实现，ONNX作为一个可跨平台的开放式神经网络模型格式，非常适合在各种嵌入式设备、云端以及各大主流语言中部署。肺部CT分析通过使用PyQt进行GUI界面设计，C#调用脚本，主要推理是通过BiSeNet v2 模型进行处理，BiSeNet作为一款轻量级的分割网络，在性能和推理速度上较于其他网络具有极大的优势。



图十二：项目主页面

点击Xray肺炎分类，进入Xray肺炎分类诊断页面，点击上传图片，之后任选取一个数据集中的图片，点击AI分析，软件将会使用VGG模型的ONNX文件，对于要进行处理的医学影像进行推理，得到其对应置信度和检测结果，并展示出来，分别选取新冠肺炎、病毒性肺炎、正常人的肺部X光图片，模型推理全部正确，并且在测试集上达到了97%的准确率，运行结果如下所示

结果说明：AI诊断该患者极有可能患有病毒性肺炎，置信度为99.92%



图十三：病毒性肺炎

结果说明：AI诊断该患者极有可能患有新冠肺炎，置信度为99.98%



图十四：新冠肺炎

结果说明：AI诊断该患者极有可能未患肺炎，置信度为97.67%

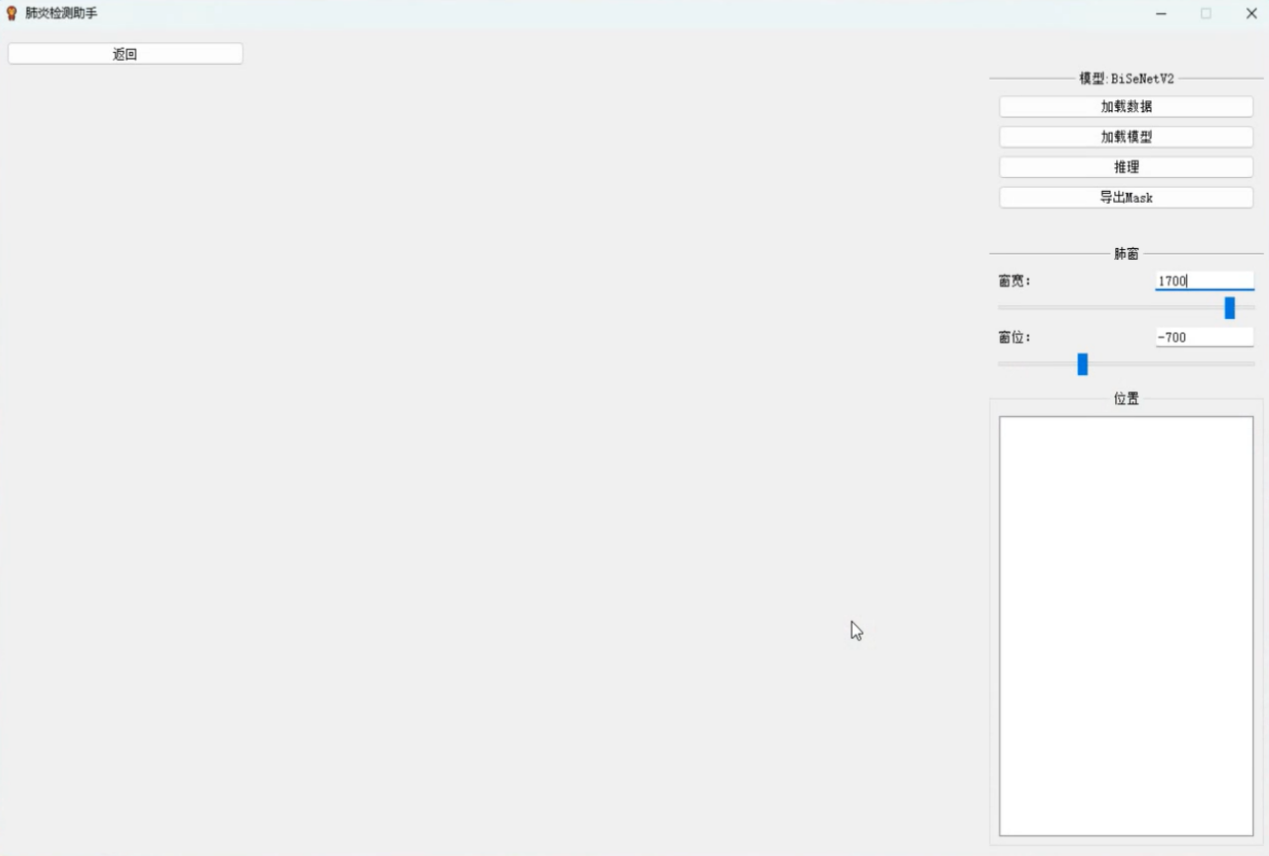


图十五：未患肺炎

点击返回，返回主页面

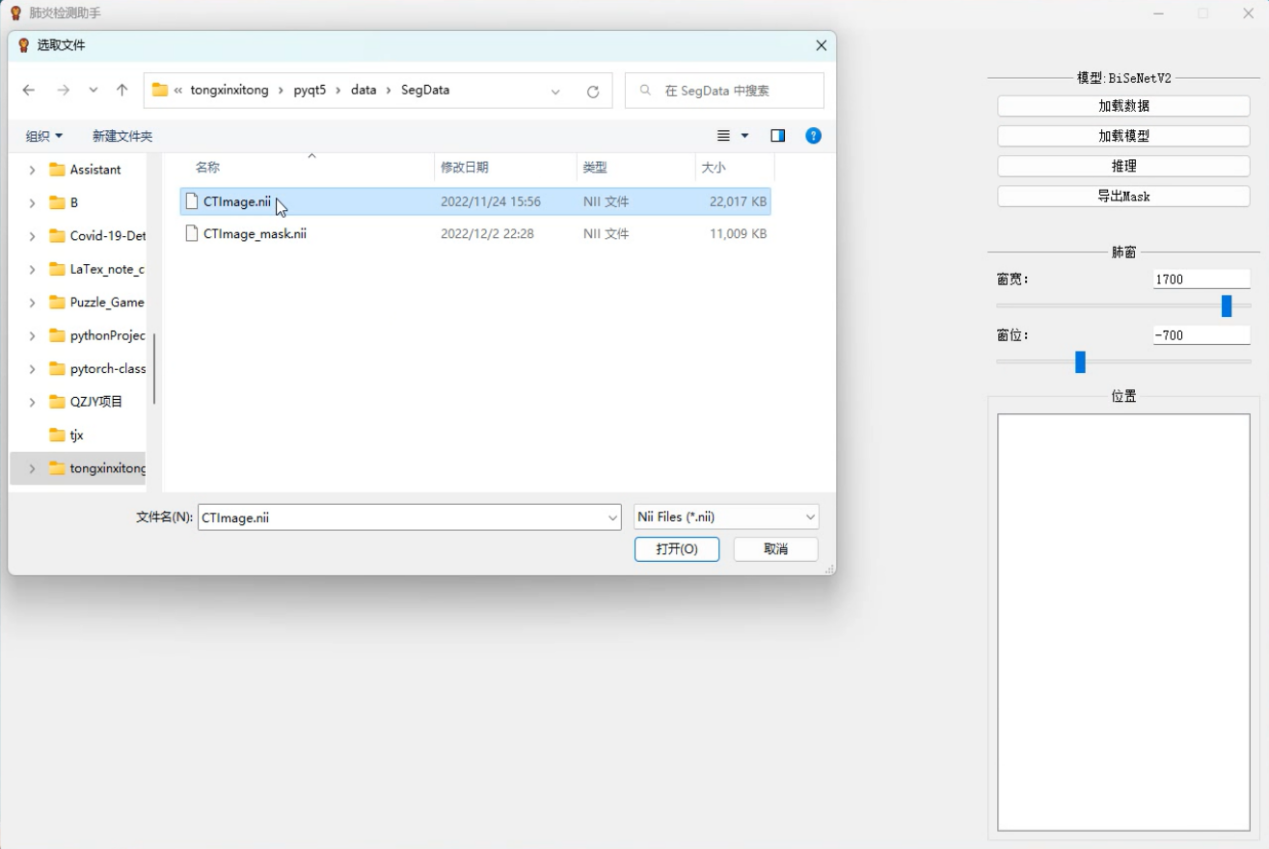
点击CT病灶分割，软件会跳转到CT病灶分割的界面，数据采用的是nii医学文件，是一种3D格式的数据，而BiSeNet V2是一种2D的分割神经网络，所以软件采用了对nii医学文件进行竖直方向上的切片处理，进而将所得到的2D切片送入部署好的分割网络，通过使用OpenCV进行后处理，在网络输出的掩膜的基础上，进行轮廓提取等操作，最终得到了较好的病灶分割结果，该结果对于传统医学的医生分析具有较好的辅助作用。

使用流程为：先加载数据，再加载模型，最后进行推理



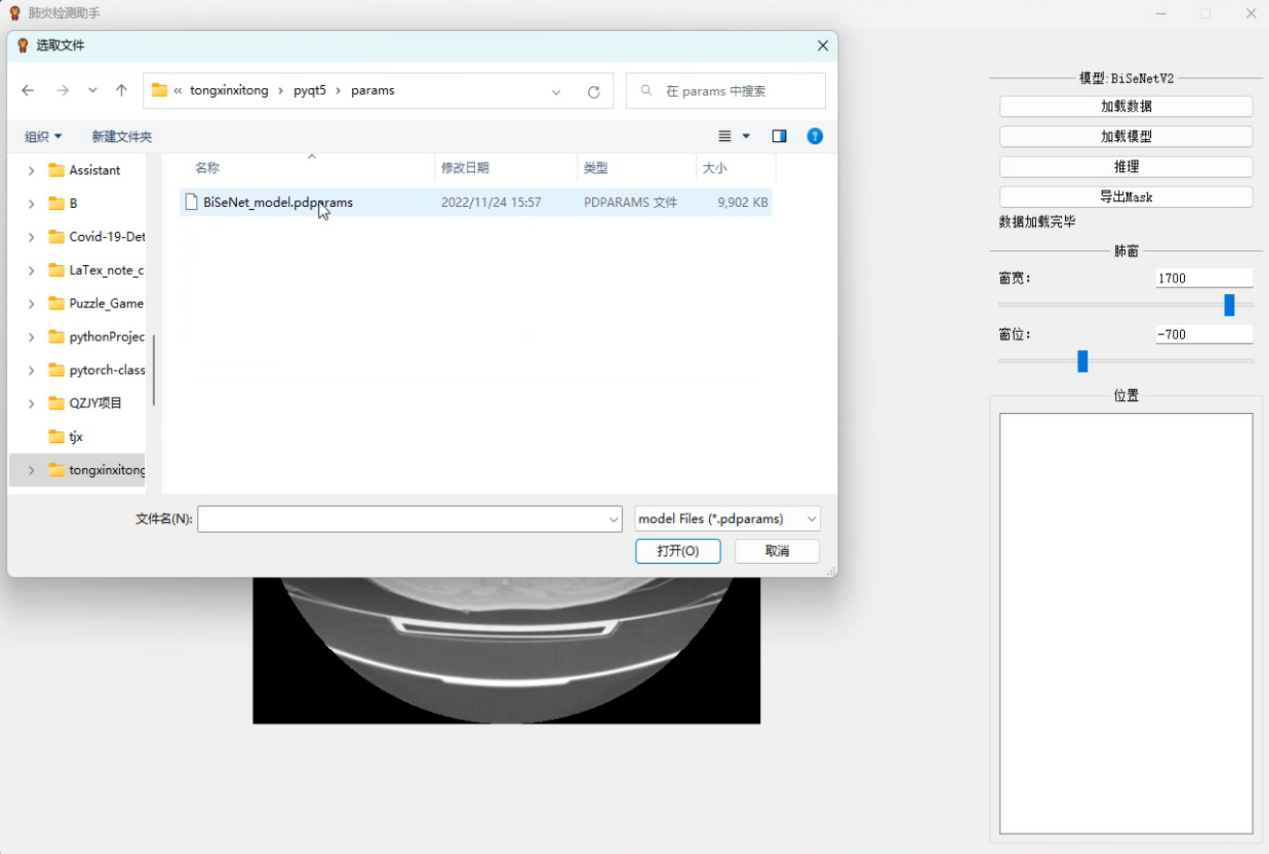
图十六：CT分割初始页面

点击加载数据，选择要进行推理的文件，本软件目前只支持nii文件的处理，将数据加载之后，数据会显示在面板之上。



图十七：加载数据

本软件通过使用BiSeNet v2在肺部病灶分割数据集上训练后的参数文件(即BiSeNet\_model.pdparams文件)对病理图片进行分割操作，BiSeNetV2是由华中科技大学计算机视觉小组提出来的一种在速度和准确性之间进行良好权衡的双边分割网络（BiSeNet V2）。其核心思想是，浅层细节（low-level details）和深层语义（high-level semantics）对语义分割都非常重要，可以考虑将空间细节和类别语义分开处理，以实现精度和推理速度之间的平衡。



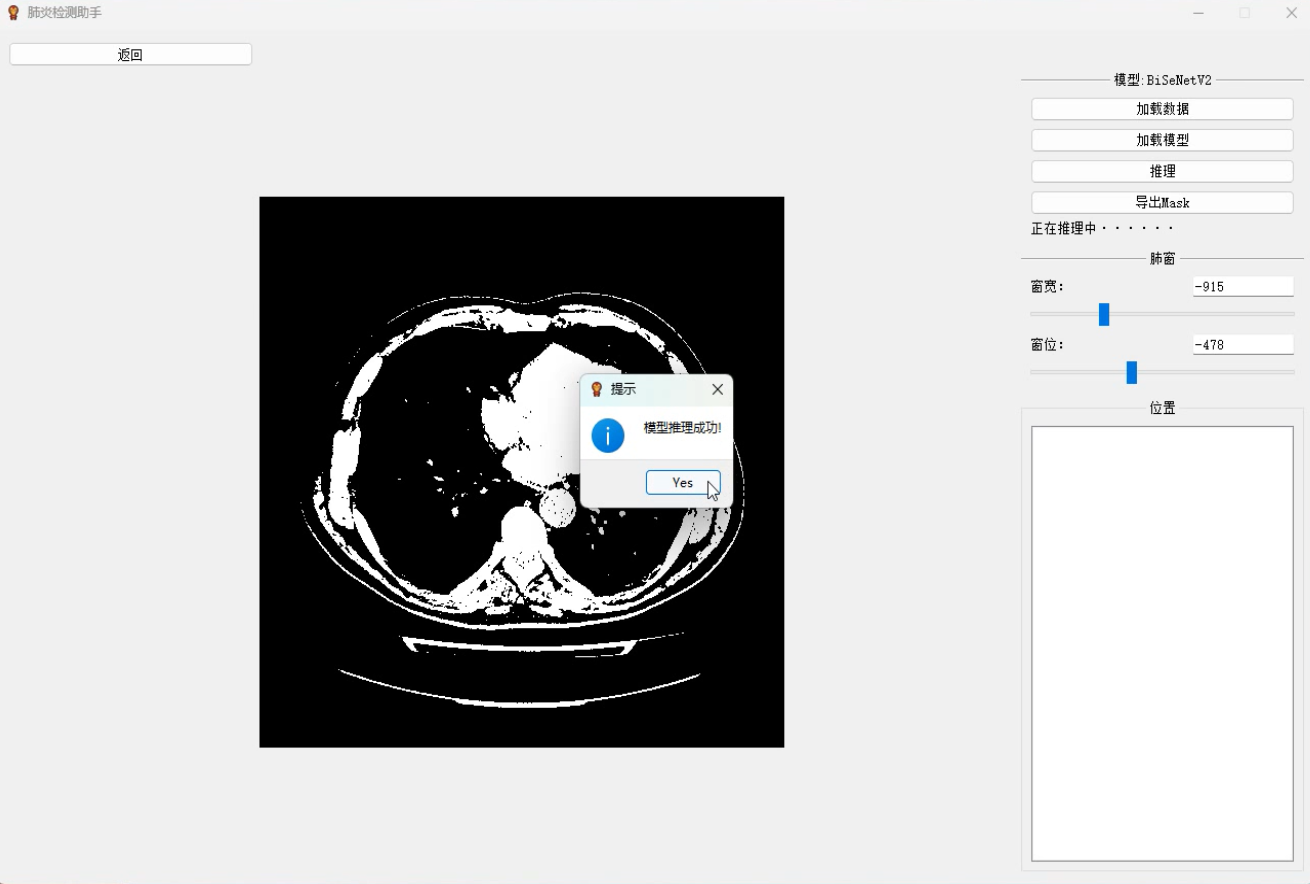
图十八：加载模型

加载完数据和模型之后，点击推理按钮，软件上会显示正在推理中，在一段时间之后会完成推理，推理的时间中，软件主要在执行将待处理图片通过神经网络输出掩膜，并通过OpenCV进行后处理操作，最终得到完整的病灶分割图。



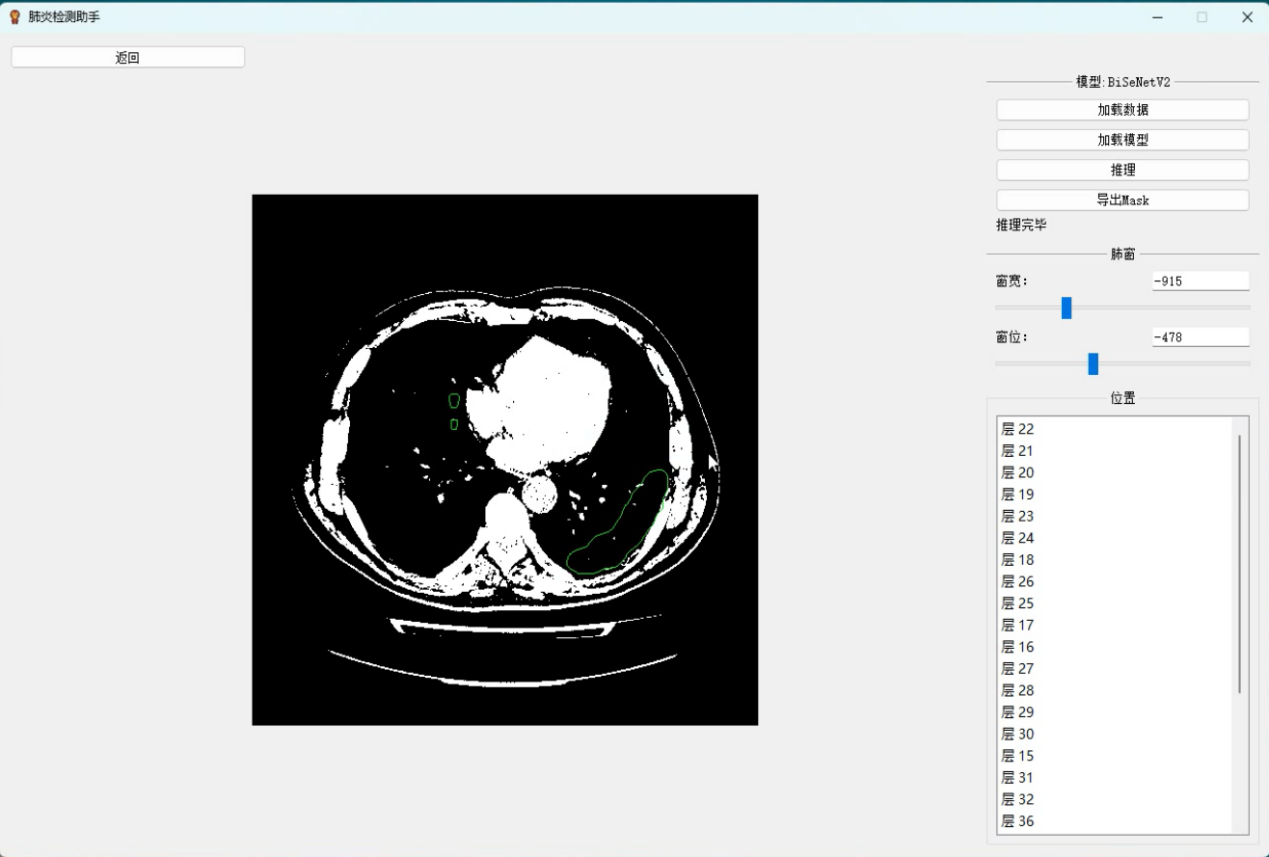
图十九：模型推理

推理成功后会有弹窗提醒，单击弹出窗口的确定按钮，得到nii文件的分割处理结果。



图二十：推理成功

待推理成功之后，软件界面左部面板将会输出最终的分割效果图，右部可以通过调节肺窗的窗宽和窗位进行展示效果的调整。同时选择不同的层，可以得到肺部不同位置的病灶分割结果，有助于医生进行更为全面的诊疗。

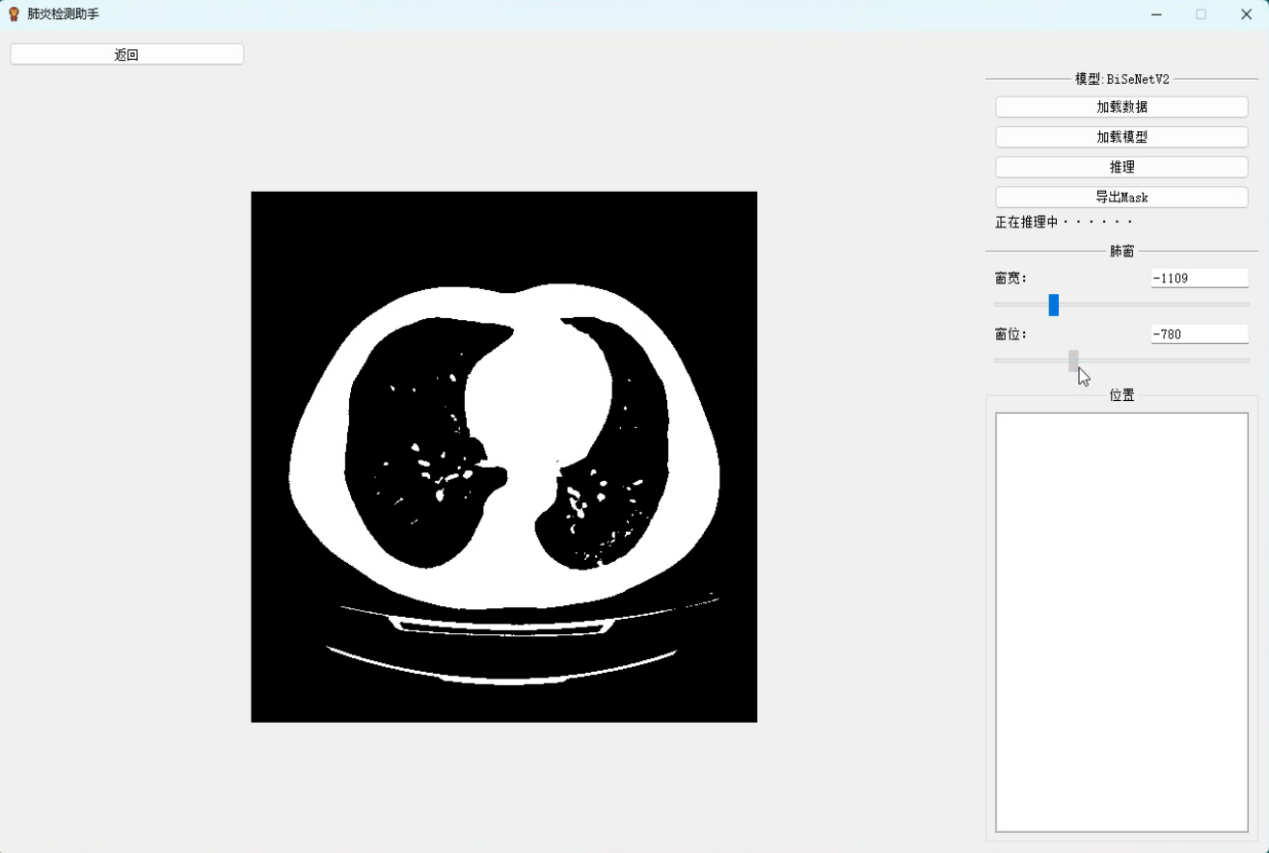


图二十一：推理结果展示

通过调整肺窗的窗宽和窗位，可以对展示的分割效果进行一些像素级的变化，进而一些较为微小的病灶也会显现出来，下面两张图即为分别调整肺窗的窗宽和窗位得到的推理结果展示。

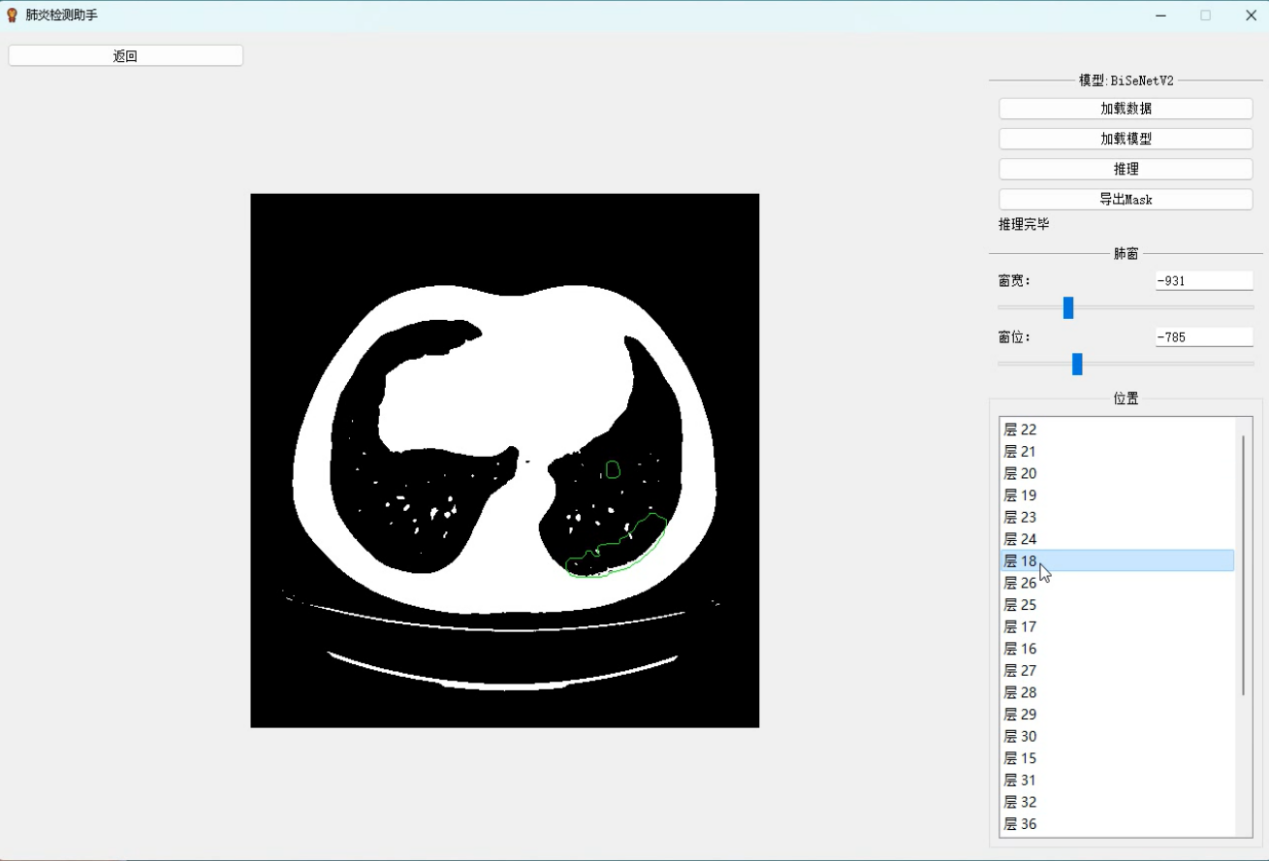


图二十二：改变窗宽

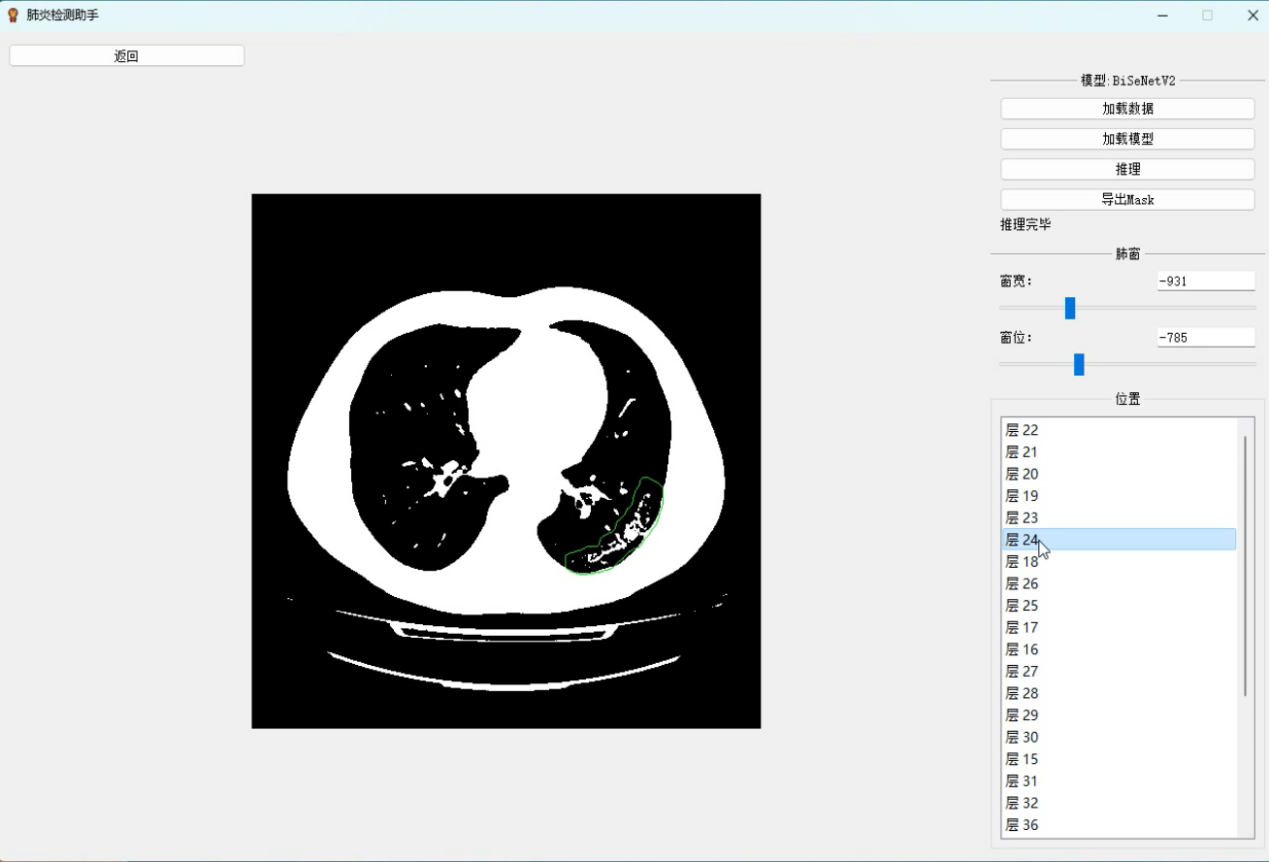


图二十三：改变窗位

除了调整窗宽、窗位，还可以通过位置层的选择，精准地还原三维图像中的病灶分布，有助于医生对于病人肺部整体受感染情况有较为完全的了解。下面的两张图分别为位置层为18和位置层为24的病灶分割结果的展示图。



图二十四：位置层为18的病灶分割结果展示



图二十五：位置层为24的病灶分割结果展示

# 总结反思：

通过本学期的通信系统综合设计，我们小组做的十基于深度学习的新冠肺炎识别平台，在本次设计中，我们使用到的工具有paddlepaddle、pytorch、pyqt以及onnx，完成了训练、测试、部署的整个流程，感受颇深，尤其是通过传统图像处理向深度学习为主的计算机视觉的技术转变，在我们看来，端到端的技术发展是必然需求，未来科技的变化方向的开发门槛会变得越来越高，而使用门槛会变得越来越低，一些专业的专家知识，将会变成神经元中的权重与偏置，从而更好的将一些社会资源流动，人们的生活因此也发生更大的改变。

本次项目对我们的启发主要是在科学研究上面，我们通过该项目接触到了医学图像处理的一些基本知识，如窗函数、识别、病灶分割等，这些是未来减轻医疗压力的主要推动力，并且充满着极大的市场背景。通过使用孪生网络加迁移学习，我们得到了小样本数据集下的良好准确率，这对我们也是一种科研上的启示—能否从更少的样本通过一定的策略得到更好的效果，能不能将这种做法迁移到目前其他如自然图像上去？这都是我们的收获。

本次开发我们使用的国产的框架Paddlepaddle，对比之前测试国产化处理器性能，我们认为目前国产化难在框架不统一，不同厂家采用不同的框架、语言实现；难在国外产品成熟度高、受众广，国内产品难以撼动其统治地位；学习成本高，学习资料鱼龙混杂，厂家甚至自己也拿不出成熟的产品方案，没有成熟的既定方案和完整的部署策略，导致用户学习成本高、学习效率低。单线程（或单进程）情况下专业推理卡的速度不如英伟达的消费级显卡，迎头赶上还需假以时日。

目前行业成熟的方案大多数为数据驱动的全监督学习，比如智慧监狱的安防系统就是通过YOLO进行目标检测，YOLO+Deepsort进行目标跟踪。全监督学习需要大量的标记数据才能训练有效的模型，这意味着需要耗费大量的人力和时间来进行数据标注。此外，全监督学习可能会受到标记数据质量的限制，标记数据的缺陷或错误可能会导致训练出的模型的准确性降低。未来的人工智能一定是朝着弱监督甚至无监督的方向发展。

该项目作为我们接触人工智能的第一步，让我们认识到了深度学习算法的强悍与未来极大的发展潜力，该项目还有很多可以拓展的服务，比如训练一个更大的数据集进行一个多病症检测平台，当人们做全身体检的时候可以直接输出不同的病症，并且可以加入一些多模态和大语言模型，能够达到对病症进行自主诊断的效果，以及如果检测初传染疾病，及时通知防控科等，相信如果有机会和时间，我们一定会将其变得更完善，更有效。

参 考 文 献

1.<https://zh.m.wikipedia.org/zhhans/%E4%BA%BA%E5%B7%A5%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C>

2. K. Simonyan and A. Zisserman. (2014). “Very deep convolutional networks for large-scale image recognition.” [Online]. Available: [https://arxiv.org/abs/1409.1556](https://link.zhihu.com/?target=https%3A//arxiv.org/abs/1409.1556" \t "_blank).

3. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

4. Chollet F. Xception: Deep learning with depthwise separable convolutions[J]. arXiv preprint, 2017: 1610.02357.

5. Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. arXiv preprint arXiv:2010.11929, 2020.

6. Lei J,Li J,Li X,et al.CT Imaging of the 2019 Novel Coronavirus(2019-nCoV) Pneumonia[J].Radiology,2020,295(1):18

7. Yu, C., Gao, C., Wang, J. et al. BiSeNet V2: Bilateral Network with Guided Aggregation for Real-Time Semantic Segmentation. Int J Comput Vis 129, 3051–3068 (2021).

# 项目分工：

李鑫：分类、分割模型的搭建与训练、pyqt界面设计、论文阅读、数据集处理

李欣竹：论文阅读、分割模型搭建、界面设计

王籽予：论文阅读、分类模型搭建、数据集处理

李明晓：论文阅读、分类模型搭建、数据集处理

# 附录

## 分类模型代码

data\_processing.py

1. import os
2. import zipfile
3. import random
4. import json
5. '''
6. 参数配置
7. '''
8. train\_parameters = {
9. "input\_size": [3, 224, 224],  # 输入图片的shape
10. "class\_dim": 3,  # 分类数
11. "src\_path": "data/dataset.zip",  # 原始数据集路径
12. "target\_path": "data/dataset",  # 要解压的路径
13. "train\_list\_path": "data/train.txt",  # train.txt路径
14. "eval\_list\_path": "data/eval.txt",  # eval.txt路径
15. "readme\_path": "data/readme.json",  # readme.json路径
16. "label\_dict": {},  # 标签字典
17. "num\_epochs": 20,  # 训练轮数
18. "train\_batch\_size": 64,  # 训练时每个批次的大小
19. "skip\_steps": 30,
20. "save\_steps": 300,
21. "learning\_strategy": {  # 优化函数相关的配置
22. "lr": 0.0001  # 超参数学习率
23. },
24. "checkpoints": "work/checkpoints"  # 保存的路径
25. }
26. def unzip\_data(src\_path, target\_path):
27. '''
28. 解压原始数据集，将src\_path路径下的zip包解压至target\_path目录下
29. '''
30. if (not os.path.isdir(target\_path + "Chinese Medicine")):
31. z = zipfile.ZipFile(src\_path, 'r')
32. z.extractall(path=target\_path)
33. z.close()
34. def get\_data\_list(target\_path, train\_list\_path, eval\_list\_path):
35. '''
36. 生成数据列表
37. '''
38. # 存放所有类别的信息
39. class\_detail = []
40. # 获取所有类别保存的文件夹名称
41. data\_list\_path = target\_path + "/train/"
42. class\_dirs = os.listdir(data\_list\_path)
43. # 总的图像数量
44. all\_class\_images = 0
45. # 存放类别标签
46. class\_label = 0
47. # 存放类别数目
48. class\_dim = 0
49. # 存储要写进eval.txt和train.txt中的内容
50. trainer\_list = []
51. eval\_list = []
52. # 读取每个类别
53. for class\_dir in class\_dirs:
54. if class\_dir != ".DS\_Store":
55. class\_dim += 1
56. # 每个类别的信息
57. class\_detail\_list = {}
58. eval\_sum = 0
59. trainer\_sum = 0
60. # 统计每个类别有多少张图片
61. class\_sum = 0
62. # 获取类别路径
63. path = data\_list\_path + class\_dir
64. # 获取所有图片
65. img\_paths = os.listdir(path)
66. for img\_path in img\_paths:  # 遍历文件夹下的每个图片
67. if img\_path.split(".")[-1] == "png":
68. name\_path = path + '/' + img\_path  # 每张图片的路径
69. if class\_sum % 8 == 0:  # 每8张图片取一个做验证数据
70. eval\_sum += 1  # test\_sum为测试数据的数目
71. eval\_list.append(name\_path + "\t%d" % class\_label + "\n")
72. else:
73. trainer\_sum += 1
74. trainer\_list.append(name\_path + "\t%d" % class\_label + "\n")  # trainer\_sum测试数据的数目
75. class\_sum += 1  # 每类图片的数目
76. all\_class\_images += 1  # 所有类图片的数目
77. else:
78. continue
79. # 说明的json文件的class\_detail数据
80. class\_detail\_list['class\_name'] = class\_dir  # 类别名称
81. class\_detail\_list['class\_label'] = class\_label  # 类别标签
82. class\_detail\_list['class\_eval\_images'] = eval\_sum  # 该类数据的测试集数目
83. class\_detail\_list['class\_trainer\_images'] = trainer\_sum  # 该类数据的训练集数目
84. class\_detail.append(class\_detail\_list)
85. # 初始化标签列表
86. train\_parameters['label\_dict'][str(class\_label)] = class\_dir
87. class\_label += 1
88. # 初始化分类数
89. train\_parameters['class\_dim'] = class\_dim
90. # 乱序
91. random.shuffle(eval\_list)
92. with open(eval\_list\_path, 'a') as f:
93. for eval\_image in eval\_list:
94. f.write(eval\_image)
95. random.shuffle(trainer\_list)
96. with open(train\_list\_path, 'a') as f2:
97. for train\_image in trainer\_list:
98. f2.write(train\_image)
99. # 说明的json文件信息
100. readjson = {}
101. readjson['all\_class\_name'] = data\_list\_path  # 文件父目录
102. readjson['all\_class\_images'] = all\_class\_images
103. readjson['class\_detail'] = class\_detail
104. jsons = json.dumps(readjson, sort\_keys=True, indent=4, separators=(',', ': '))
105. with open(train\_parameters['readme\_path'], 'w') as f:
106. f.write(jsons)
107. print('生成数据列表完成！')

Dataset.py

1. import os
2. import zipfile
3. import random
4. import json
5. import paddle
6. import sys
7. import numpy as np
8. from PIL import Image
9. import matplotlib.pyplot as plt
10. from paddle.io import Dataset
11. from data\_processing import  unzip\_data,train\_parameters
12. class dataset(Dataset):
13. def \_\_init\_\_(self, data\_path, mode='train'):
14. """
15. 数据读取器
16. :param data\_path: 数据集所在路径
17. :param mode: train or eval
18. """
19. super().\_\_init\_\_()
20. self.data\_path = data\_path
21. self.img\_paths = []
22. self.labels = []
23. if mode == 'train':
24. with open(os.path.join(self.data\_path, "train.txt"), "r", encoding="utf-8") as f:
25. self.info = f.readlines()
26. for img\_info in self.info:
27. img\_path, label = img\_info.strip().split('\t')
28. self.img\_paths.append(img\_path)
29. self.labels.append(int(label))
30. else:
31. with open(os.path.join(self.data\_path, "eval.txt"), "r", encoding="utf-8") as f:
32. self.info = f.readlines()
33. for img\_info in self.info:
34. img\_path, label = img\_info.strip().split('\t')
35. self.img\_paths.append(img\_path)
36. self.labels.append(int(label))
37. def \_\_getitem\_\_(self, index):
38. """
39. 获取一组数据
40. :param index: 文件索引号
41. :return:
42. """
43. # 第一步打开图像文件并获取label值
44. img\_path = self.img\_paths[index]
45. img = Image.open(img\_path)
46. if img.mode != 'RGB':
47. img = img.convert('RGB')
48. img = img.resize((224, 224), Image.BILINEAR)
49. img = np.array(img).astype('float32')
50. img = img.transpose((2, 0, 1)) / 255
51. label = self.labels[index]
52. label = np.array([label], dtype="int64")
53. return img, label
54. def print\_sample(self, index: int = 0):
55. print("文件名", self.img\_paths[index], "\t标签值", self.labels[index])
56. def \_\_len\_\_(self):
57. return len(self.img\_paths)

eval.py

1. import paddle
2. import numpy as np
3. from Dataset import dataset
4. from model import VGGNet
5. '''
6. 模型评估
7. '''
8. # 测试数据加载
9. eval\_dataset = dataset('/home/aistudio/data', mode='eval')
10. eval\_loader = paddle.io.DataLoader(eval\_dataset, batch\_size=8, shuffle=False)
11. eval\_dataset.print\_sample(0)
12. print(eval\_dataset.\_\_len\_\_())
13. print(eval\_dataset.\_\_getitem\_\_(10)[0].shape)
14. print(eval\_dataset.\_\_getitem\_\_(10)[1].shape)
15. model\_\_state\_dict = paddle.load('work/checkpoints/save\_dir\_1500.pdparams')
16. model\_eval = VGGNet()
17. model\_eval.set\_state\_dict(model\_\_state\_dict)
18. model\_eval.eval()
19. accs = []
20. for \_, data in enumerate(eval\_loader()):
21. x\_data = data[0]
22. y\_data = data[1]
23. predicts = model\_eval(x\_data)
24. acc = paddle.metric.accuracy(predicts, y\_data)
25. accs.append(acc.numpy()[0])
26. print('模型在验证集上的准确率为：',np.mean(accs))

paddle.py

1. import paddle
2. class ConvPool(paddle.nn.Layer):
3. '''卷积+池化'''
4. def \_\_init\_\_(self,
5. num\_channels,
6. num\_filters,
7. filter\_size,
8. pool\_size,
9. pool\_stride,
10. groups,
11. conv\_stride=1,
12. conv\_padding=1,
13. ):
14. super(ConvPool, self).\_\_init\_\_()
15. for i in range(groups):
16. conv2d = self.add\_sublayer(  # 添加子层实例
17. 'bb\_%d' % i,
18. paddle.nn.Conv2D(  # layer
19. in\_channels=num\_channels,  # 通道数
20. out\_channels=num\_filters,  # 卷积核个数
21. kernel\_size=filter\_size,  # 卷积核大小
22. stride=conv\_stride,  # 步长
23. padding=conv\_padding,  # padding
24. )
25. )
26. self.add\_sublayer(
27. 'relu%d' % i,
28. paddle.nn.ReLU()
29. )
30. num\_channels = num\_filters
31. self.add\_sublayer(
32. 'Maxpool',
33. paddle.nn.MaxPool2D(
34. kernel\_size=pool\_size,  # 池化核大小
35. stride=pool\_stride  # 池化步长
36. )
37. )
38. def forward(self, inputs):
39. x = inputs
40. for prefix, sub\_layer in self.named\_children():
41. # print(prefix,sub\_layer)
42. x = sub\_layer(x)
43. return x
44. # 使用上面的ConvPool模块定义VGGNet
45. class VGGNet(paddle.nn.Layer):
46. def \_\_init\_\_(self):
47. super(VGGNet, self).\_\_init\_\_()
48. # #3:通道数，64：卷积核个数，3:卷积核大小，2:池化核大小，2:池化步长，2:连续卷积个数(每两组之间)
49. self.convpool1 = ConvPool(3, 64, 3, 2, 2, 2)
50. self.convpool2 = ConvPool(64, 128, 3, 2, 2, 2)
51. self.convpool3 = ConvPool(128, 256, 3, 2, 2, 3)
52. self.convpool4 = ConvPool(256, 512, 3, 2, 2, 3)
53. self.convpool5 = ConvPool(512, 512, 3, 2, 2, 3)
54. self.convpool5\_shape = 512 \* 7 \* 7
55. self.fc1 = paddle.nn.Linear(self.convpool5\_shape, 4096)
56. self.fc2 = paddle.nn.Linear(4096, 4096)
57. self.fc3 = paddle.nn.Linear(4096, 3)
58. def forward(self, inputs, label=None):
59. x = self.convpool1(inputs)
60. x = self.convpool2(x)
61. x = self.convpool3(x)
62. x = self.convpool4(x)
63. x = self.convpool5(x)
64. x = paddle.reshape(x, [-1, 512 \* 7 \* 7])
65. x = self.fc1(x)
66. x = self.fc2(x)
67. out = self.fc3(x)
68. if label is not None:
69. acc = paddle.metric.accuracy(input=out, label=label)
70. return out, acc
71. else:
72. return out

onnx\_output.py

1. import paddle
2. class ConvPool(paddle.nn.Layer):
3. '''卷积+池化'''
4. def \_\_init\_\_(self,
5. num\_channels,
6. num\_filters,
7. filter\_size,
8. pool\_size,
9. pool\_stride,
10. groups,
11. conv\_stride=1,
12. conv\_padding=1,
13. ):
14. super(ConvPool, self).\_\_init\_\_()
15. for i in range(groups):
16. conv2d = self.add\_sublayer(  # 添加子层实例
17. 'bb\_%d' % i,
18. paddle.nn.Conv2D(  # layer
19. in\_channels=num\_channels,  # 通道数
20. out\_channels=num\_filters,  # 卷积核个数
21. kernel\_size=filter\_size,  # 卷积核大小
22. stride=conv\_stride,  # 步长
23. padding=conv\_padding,  # padding
24. )
25. )
26. self.add\_sublayer(
27. 'relu%d' % i,
28. paddle.nn.ReLU()
29. )
30. num\_channels = num\_filters
31. self.add\_sublayer(
32. 'Maxpool',
33. paddle.nn.MaxPool2D(
34. kernel\_size=pool\_size,  # 池化核大小
35. stride=pool\_stride  # 池化步长
36. )
37. )
38. def forward(self, inputs):
39. x = inputs
40. for prefix, sub\_layer in self.named\_children():
41. x = sub\_layer(x)
42. return x
43. # 使用上面的ConvPool模块定义VGGNet
44. class VGGNet(paddle.nn.Layer):
45. def \_\_init\_\_(self):
46. super(VGGNet, self).\_\_init\_\_()
47. # #3:通道数，64：卷积核个数，3:卷积核大小，2:池化核大小，2:池化步长，2:连续卷积个数(每两组之间)
48. self.convpool1 = ConvPool(3, 64, 3, 2, 2, 2)
49. self.convpool2 = ConvPool(64, 128, 3, 2, 2, 2)
50. self.convpool3 = ConvPool(128, 256, 3, 2, 2, 3)
51. self.convpool4 = ConvPool(256, 512, 3, 2, 2, 3)
52. self.convpool5 = ConvPool(512, 512, 3, 2, 2, 3)
53. self.convpool5\_shape = 512 \* 7 \* 7
54. self.fc1 = paddle.nn.Linear(self.convpool5\_shape, 4096)
55. self.fc2 = paddle.nn.Linear(4096, 4096)
56. self.fc3 = paddle.nn.Linear(4096, 3)
57. def forward(self, inputs, label=None):
58. x = self.convpool1(inputs)
59. x = self.convpool2(x)
60. x = self.convpool3(x)
61. x = self.convpool4(x)
62. x = self.convpool5(x)
63. x = paddle.reshape(x, [-1, 512 \* 7 \* 7])
64. x = self.fc1(x)
65. x = self.fc2(x)
66. out = self.fc3(x)
67. if label is not None:
68. acc = paddle.metric.accuracy(input=out, label=label)
69. return out, acc
70. else:
71. return out
72. # 实例化模型
73. model = VGGNet()
74. model.eval()
75. # 加载预训练模型参数
76. model.set\_dict(paddle.load("work/checkpoints/save\_dir\_1500.pdparams"))
77. # 定义输入数据
78. input\_spec = paddle.static.InputSpec(shape=[None, 3, 224, 224], dtype='float32', name='image')
79. # ONNX模型导出
80. paddle.onnx.export(model, 'home', input\_spec=[input\_spec])

Train.py

1. import paddle
2. import matplotlib.pyplot as plt
3. from Dataset import dataset
4. from model import VGGNet
5. from data\_processing import unzip\_data, train\_parameters, get\_data\_list
6. '''
7. 参数初始化
8. '''
9. src\_path = train\_parameters['src\_path']
10. target\_path = train\_parameters['target\_path']
11. train\_list\_path = train\_parameters['train\_list\_path']
12. eval\_list\_path = train\_parameters['eval\_list\_path']
13. '''
14. 解压原始数据到指定路径
15. '''
16. unzip\_data(src\_path, target\_path)
17. '''
18. 划分训练集与验证集，乱序，生成数据列表
19. '''
20. # 每次生成数据列表前，首先清空train.txt和eval.txt
21. with open(train\_list\_path, 'w') as f:
22. f.seek(0)
23. f.truncate()
24. with open(eval\_list\_path, 'w') as f:
25. f.seek(0)
26. f.truncate()
27. # 生成数据列表
28. get\_data\_list(target\_path, train\_list\_path, eval\_list\_path)
29. train\_dataset = dataset('data', mode='train')
30. train\_loader = paddle.io.DataLoader(train\_dataset, batch\_size=16, shuffle=True)
31. train\_dataset.print\_sample(200)
32. #print(train\_dataset.\_\_len\_\_())
33. def draw\_process(title, color, iters, data, label):
34. plt.title(title, fontsize=24)
35. plt.xlabel("iter", fontsize=20)
36. plt.ylabel(label, fontsize=20)
37. plt.plot(iters, data, color=color, label=label)
38. plt.legend()
39. plt.grid()
40. plt.show()
41. model = VGGNet()
42. model.train()
43. cross\_entropy = paddle.nn.CrossEntropyLoss()
44. optimizer = paddle.optimizer.Adam(learning\_rate=train\_parameters['learning\_strategy']['lr'],
45. parameters=model.parameters())
46. steps = 0
47. Iters, total\_loss, total\_acc = [], [], []
48. for epo in range(train\_parameters['num\_epochs']):
49. for \_, data in enumerate(train\_loader()):
50. steps += 1
51. x\_data = data[0]
52. y\_data = data[1]
53. predicts, acc = model(x\_data, y\_data)
54. loss = cross\_entropy(predicts, y\_data)
55. loss.backward()
56. optimizer.step()
57. optimizer.clear\_grad()
58. if steps % train\_parameters["skip\_steps"] == 0:
59. Iters.append(steps)
60. total\_loss.append(loss.numpy()[0])
61. total\_acc.append(acc.numpy()[0])
62. # 打印中间过程
63. print('epo: {}, step: {}, loss is: {}, acc is: {}' \
64. .format(epo, steps, loss.numpy(), acc.numpy()))
65. # 保存模型参数
66. if steps % train\_parameters["save\_steps"] == 0:
67. save\_path = train\_parameters["checkpoints"] + "/" + "save\_dir\_" + str(steps) + '.pdparams'
68. print('save model to: ' + save\_path)
69. paddle.save(model.state\_dict(), save\_path)
70. paddle.save(model.state\_dict(), train\_parameters["checkpoints"] + "/" + "model\_final.pdparams")
71. draw\_process("trainning loss", "red", Iters, total\_loss, "trainning loss")
72. draw\_process("trainning acc", "green", Iters, total\_acc, "trainning acc")

## 分割模型代码

TrainModel.ipynb 文件：

1. # 解压数据,数据来源：https://mosmed.ai/
2. #原本数据是NiFit格式数据，然后转换成png格式，输入PaddleSeg进行训练
3. !unzip data/data114821/MosMedSegPNG.zip -d /home/aistudio/work
4. #支持配置化训练和API方式训练,这里采用API方式进行训练
5. !pip install paddleseg SimpleITK
6. ### 划分数据集
7. random.seed(1000)
8. path\_origin = '/home/aistudio/work/MosMedSegPNG/origin'
9. files = list(filter(lambda x: x.endswith('.png'), os.listdir(path\_origin)))
10. random.shuffle(files)
11. rate = int(len(files) \* 0.8)#训练集和测试集8：2
12. train\_txt = open('/home/aistudio/work/MosMedSegPNG/train\_list.txt','w')
13. val\_txt = open('/home/aistudio/work/MosMedSegPNG/val\_list.txt','w')
14. for i,f in enumerate(files):
15. image\_path = os.path.join(path\_origin, f)
16. label\_path = image\_path.replace("origin", "mask")
17. if i < rate:
18. train\_txt.write(image\_path + ' ' + label\_path+ '\n')
19. else:
20. val\_txt.write(image\_path + ' ' + label\_path+ '\n')
21. train\_txt.close()
22. val\_txt.close()
23. print('完成')

## PyQt代码

Deploy.py 部署文件

1. # -\*- coding: utf-8 -\*-
2. # pyuic5 -o SegGroundClassUI.py SegGroundClassUI.ui
3. import sys
4. from PyQt5.QtWidgets import \*
5. from PyQt5.QtCore import \*
6. from PyQt5.QtGui import \*
7. from FormUI import Ui\_Form
8. from paddleseg.models import BiSeNetV2
9. import paddleseg.transforms as T
10. from paddleseg.core import infer
11. import paddle
12. import numpy as np
13. import SimpleITK as sitk
14. import os
15. import cv2
16. def windowwc(sitkImage, ww=1500, wc=-550):
17. """
18. 主要用于设置窗宽窗位
19. @param sitkImage:SimpleITK图像数据
20. @param ww:窗宽窗位
21. @param wc:窗宽窗位
22. @return:sitkImage
23. """
24. min = int(wc - ww / 2.0)
25. max = int(wc + ww / 2.0)
26. intensityWindow = sitk.IntensityWindowingImageFilter()
27. intensityWindow.SetWindowMaximum(max)
28. intensityWindow.SetWindowMinimum(min)
29. sitkImage = intensityWindow.Execute(sitkImage)
30. return sitkImage
31. def readNii(path, ww, wc, isflipud=True, ):
32. """
33. 读取和加载数据。如果图像是上下翻转的，就将其翻转过来
34. @param path: 文件路径
35. @param ww:窗宽窗位
36. @param wc:窗宽窗位
37. @param isflipud: 是否需要翻转
38. @return: data
39. """
40. if type(path) == str:
41. img = windowwc(sitk.ReadImage(path), ww, wc)
42. else:
43. img = windowwc(path, ww, wc)
44. data = sitk.GetArrayFromImage(img)
45. if isflipud:
46. data = np.flip(data, 1)
47. return data
48. class InferThread(QThread):
49. """
50. 调用PyQt5.QtCore，建立一个任务线程类, 进行推理任务
51. """
52. # 收集推理失败的信号
53. signal\_infer\_fail = pyqtSignal()
54. # 传递推理结果
55. signal\_infer\_result = pyqtSignal(np.ndarray)
56. def \_\_init\_\_(self, sitkImage, model):
57. super(InferThread, self).\_\_init\_\_()
58. self.sitkImage = sitkImage
59. self.model = model
60. self.transforms = T.Compose([
61. T.Resize(target\_size=(512, 512)),
62. T.Normalize()
63. ])
64. def run(self):
65. """
66. 在启动线程后任务开始执行
67. """
68. try:
69. data = readNii(self.sitkImage, 1500, -500)
70. inferData = np.zeros\_like(data)
71. d, h, w = data.shape
72. for i in range(d):
73. img = data[i].copy()
74. img = img.astype(np.float32)
75. pre = self.nn\_infer(self.model, img, self.transforms)
76. inferData[i] = pre
77. self.signal\_infer\_result.emit(inferData)
78. except Exception as e:
79. print(e)
80. self.signal\_infer\_fail.emit()
81. def nn\_infer(self, model, im, transforms):
82. """
83. 预测结果
84. @param model: 模型参数
85. @param im: 图像数据
86. @param transforms:传入transforms方法
87. @return: 预测结果pred
88. """
89. img, \_ = transforms(im)
90. img = paddle.to\_tensor(img[np.newaxis, :])
91. pre = infer.inference(model, img)
92. pred = paddle.argmax(pre, axis=1).numpy().reshape((512, 512))
93. return pred.astype('uint8')
94. class MainWindow(QWidget, Ui\_Form):
95. def \_\_init\_\_(self, \*args, \*\*kwargs):
96. super().\_\_init\_\_(\*args, \*\*kwargs)
97. self.setupUi(self)
98. self.initUI()
99. self.setWindowTitle('肺炎检测助手')
100. # 打开nii文件选择器
101. self.bn\_open.clicked.connect(self.openFile)
102. # 打开模型文件选择器
103. self.bn\_loadModel.clicked.connect(self.openModleFile)
104. # 推理按钮
105. self.bn\_infer.clicked.connect(self.infer)
106. self.bn\_output.clicked.connect(self.outputFile)
107. self.sitkImage = object()
108. self.npImage = object()
109. # 记录当前第几层
110. self.currIndex = 0
111. # 记录数据的最大层
112. self.maxCurrIndex = 0
113. # 记录数据的最小层，其实就是0
114. self.minCurrIndex = 0
115. self.baseFileName = ''
116. self.isInferSucceed = False
117. # 判断是否按下鼠标右键
118. self.isRightPressed = bool(False)
119. self.model = object()
120. # 判断模型是否加载成功
121. self.isModelReady = False
122. # 宽宽窗位滑动条
123. self.slider\_ww.valueChanged.connect(self.resetWWWcAndShow)
124. self.slider\_wc.valueChanged.connect(self.resetWWWcAndShow)
125. # 设置窗宽窗位文本框只能输入一定范围的整数
126. intValidator = QIntValidator(self)
127. intValidator.setRange(-2000, 2000)
128. self.line\_ww.setValidator(intValidator)
129. self.line\_ww.editingFinished.connect(self.resetWWWcAndShow)
130. self.line\_wc.setValidator(intValidator)
131. self.line\_wc.editingFinished.connect(self.resetWWWcAndShow)
132. self.listWidget.itemDoubleClicked.connect(self.changeLayer)
133. def initUI(self):
134. try:
135. # 定义展示的窗体及其初始的参数
136. self.wwwcList = {'肺窗': [1700, -700]}
137. self.line\_ww.setText(str(1700))
138. self.line\_wc.setText(str(-700))
139. self.slider\_ww.setValue(1700)
140. self.slider\_wc.setValue(-700)
141. self.ww = 1700
142. self.wc = -700
143. self.currWw = self.ww
144. self.currWc = self.wc
145. except Exception as e:
146. print(e)
147. def openFile(self):
148. """
149. 打开医学影像文件选择器
150. """
151. try:
152. filename, \_ = QFileDialog.getOpenFileName(self,
153. "选取文件",
154. "./",
155. "Nii Files (\*.nii);;Nii Files (\*.nii.gz);;All Files (\*)")
156. if filename:
157. # 清空列表
158. self.listWidget.clear()
159. self.isInferSucceed = False
160. self.text\_loadModel.setText("数据加载完毕")
161. self.baseFileName = os.path.basename(filename).split('.')[0]
162. self.sitkImage = sitk.ReadImage(filename)
163. self.npImage = readNii(self.sitkImage, self.ww, self.wc)
164. self.maxCurrIndex = self.npImage.shape[0]
165. self.currIndex = int(self.maxCurrIndex / 2)
166. self.showImg(self.npImage[self.currIndex])
167. except Exception as e:
168. print(e)
169. def openModleFile(self):
170. """
171. 打开模型文件选择器
172. """
173. filename, \_ = QFileDialog.getOpenFileName(self, "选取文件", "./", "model Files (\*.pdparams)")
174. if filename:
175. try:
176. self.text\_loadModel.setText(" ")
177. num\_class = int(2)
178. self.model = BiSeNetV2(num\_classes=num\_class)
179. para\_state\_dict = paddle.load(filename)
180. self.model.set\_dict(para\_state\_dict)
181. self.text\_loadModel.setText("模型加载完毕")
182. self.isModelReady = True
183. except Exception as e:
184. self.text\_loadModel.setText("模型加载失败")
185. print(e)
186. def wheelEvent(self, event):
187. """
188. 定义鼠标滑轮事件
189. """
190. try:
191. if self.maxCurrIndex != self.minCurrIndex:
192. self.angle = event.angleDelta() / 8
193. self.angleY = self.angle.y()
194. if self.angleY > 0:
195. if self.currIndex < self.maxCurrIndex - 1:
196. self.currIndex += 1
197. if self.isInferSucceed:
198. self.showImg(self.drawContours(self.npImage, self.inferData, self.currIndex))
199. else:
200. self.showImg(self.npImage[self.currIndex])
201. elif self.angleY < 0:
202. if self.currIndex != self.minCurrIndex:
203. self.currIndex -= 1
204. if self.isInferSucceed:
205. # self.npImage = self.drawContours(self.npImage, self.inferData)
206. self.showImg(self.drawContours(self.npImage, self.inferData, self.currIndex))
207. else:
208. self.showImg(self.npImage[self.currIndex])
209. except Exception as e:
210. print(e)
211. def mousePressEvent(self, event):
212. """
213. 重载鼠标单机事件
214. """
215. # 左键按下
216. if event.buttons() == Qt.RightButton:
217. # 左键按下(图片被点住),置Ture
218. self.isRightPressed = True
219. self.preMousePosition = event.pos()
220. elif event.buttons() == Qt.MidButton | Qt.RightButton:
221. self.isRightPressed = False
222. def mouseReleaseEvent(self, event):
223. if event.button() == Qt.RightButton:
224. self.isRightPressed = False
225. def mouseMoveEvent(self, event):
226. """
227. 重载一下鼠标移动事件
228. """
229. try:
230. if self.maxCurrIndex != self.minCurrIndex:
231. # 右键按下
232. if self.isRightPressed:
233. # 鼠标当前位置-先前位置=单次偏移量
234. self.endMousePosition = event.pos() - self.preMousePosition
235. self.preMousePosition = event.pos()
236. ww = self.endMousePosition.x() + self.currWw
237. wc = self.endMousePosition.y() + self.currWc
238. if ww < -2000:
239. ww = -2000
240. elif ww > 2000:
241. ww = 2000
242. if wc < -2000:
243. wc = -2000
244. elif wc > 2000:
245. wc = 2000
246. self.currWw = ww
247. self.currWc = wc
248. self.slider\_ww.setValue(int(self.currWw))
249. self.slider\_wc.setValue(int(self.currWc))
250. self.line\_ww.setText(str(self.currWw))
251. self.line\_wc.setText(str(self.currWc))
252. self.resetWWWcAndShow()
253. except Exception as e:
254. print(e)
255. def showImg(self, img):
256. """
257. 显示图片
258. @param img: 待显示的图片
259. """
260. try:
261. if img.ndim == 2:
262. img = np.expand\_dims(img, axis=2)
263. img = np.concatenate((img, img, img), axis=-1).astype(np.uint8)
264. elif img.ndim == 3:
265. img = img.astype(np.uint8)
266. qimage = QImage(img, img.shape[0], img.shape[1], img.shape[1] \* 3, QImage.Format\_RGB888)
267. pixmap\_imgSrc = QPixmap.fromImage(qimage)
268. self.canvas.setPixmap(pixmap\_imgSrc)
269. except Exception as e:
270. print(e)
271. def resetWWWcAndShow(self):
272. """
273. 通过四个方式可以修改医学图像的窗宽窗位，每次修改后都会在界面呈现出来
274. """
275. if hasattr(self.sender(), "objectName"):
276. objectName = self.sender().objectName()
277. else:
278. objectName = None
279. try:
280. if objectName == '':
281. self.line\_ww.setText(str(1700))
282. self.line\_wc.setText(str(-700))
283. self.slider\_ww.setValue(1700)
284. self.slider\_wc.setValue(-700)
285. self.ww = 1700
286. self.wc = -700
287. self.currWw = self.ww
288. self.currWc = self.wc
289. if objectName == 'slider\_ww' or objectName == 'slider\_wc':
290. self.currWw = self.slider\_ww.value()
291. self.currWc = self.slider\_wc.value()
292. self.line\_ww.setText(str(self.currWw))
293. self.line\_wc.setText(str(self.currWc))
294. elif objectName == 'line\_ww' or objectName == 'line\_wc':
295. self.currWw = int(self.line\_ww.text())
296. self.currWc = int(self.line\_wc.text())
297. self.slider\_ww.setValue(self.currWw)
298. self.slider\_wc.setValue(self.currWc)
299. if self.maxCurrIndex != self.minCurrIndex:
300. self.npImage = readNii(self.sitkImage, self.currWw, self.currWc)
301. if self.isInferSucceed:
302. self.showImg(self.drawContours(self.npImage, self.inferData, self.currIndex))
303. else:
304. self.showImg(self.npImage[self.currIndex])
305. except Exception as e:
306. print(e)
307. def infer(self):
308. """
309. 模型分割预测
310. """
311. if self.maxCurrIndex != self.minCurrIndex and self.isModelReady:
312. self.bn\_infer.setEnabled(True)
313. # 创建推理线程
314. self.infer\_thread = InferThread(self.sitkImage, self.model)
315. # 绑定推理失败的槽函数
316. self.infer\_thread.signal\_infer\_fail.connect(self.infer\_fail)
317. # 绑定推理成功的槽函数
318. self.infer\_thread.signal\_infer\_result.connect(self.infer\_result)
319. self.infer\_thread.start()
320. self.text\_loadModel.setText("正在推理中······")
321. else:
322. QMessageBox.warning(self, "提示", "推理失败，推理前请先确保:\n1.加载模型\n2.加载数据", QMessageBox.Yes, QMessageBox.Yes)
323. def infer\_result(self, inferData):
324. """
325. 分割模型预测成功后，结果保存在self.inferData
326. @param inferData: 推理数据
327. """
328. # 推理成功，并显示结果
329. try:
330. self.inferData = inferData.astype(np.uint8)
331. QMessageBox.information(self, "提示", "模型推理成功!", QMessageBox.Yes, QMessageBox.Yes)
332. self.text\_loadModel.setText("推理完毕")
333. self.isInferSucceed = True
334. self.infer\_thread.quit()
335. self.addListInfo(self.inferData)
336. self.showImg(self.drawContours(self.npImage, self.inferData, self.currIndex))
337. except Exception as e:
338. print(e)
339. def infer\_fail(self):
340. """
341. 如果推理失败，则报错
342. """
343. QMessageBox.warning(self, "警告", "模型推理失败!", QMessageBox.Yes, QMessageBox.Yes)
344. def outputFile(self):
345. """
346. 将保存模型预测结果为nii格式文件
347. """
348. try:
349. if self.isInferSucceed:
350. filedir = QFileDialog.getExistingDirectory(None, "文件保存", os.getcwd())
351. if filedir:
352. # 读取nii文件时转换np文件时对数据进行上下翻转，再输入模型推理，保存回nii文件时要翻转回来。
353. self.inferData = np.flip(self.inferData, 1)
354. pre\_sitkImage = sitk.GetImageFromArray(self.inferData)
355. pre\_sitkImage.CopyInformation(self.sitkImage)
356. pre\_sitkImage = sitk.Cast(pre\_sitkImage, sitk.sitkUInt8)
357. save\_path = os.path.join(filedir, self.baseFileName + '\_mask.nii')
358. sitk.WriteImage(pre\_sitkImage, save\_path)
359. else:
360. QMessageBox.warning(self, "警告", "无进行过推理，无法保存！", QMessageBox.Yes, QMessageBox.Yes)
361. except Exception as e:
362. print(e)
363. def drawContours(self, npImage, inferData, currIndex):
364. """
365. 通过OpenCV将mask转换成轮廓绘制在原图上
366. @param npImage: 图像数据
367. @param inferData: 模型推理的结果
368. @param currIndex: 层数序号
369. @return: 绘制轮廓后的图片
370. """
371. img = npImage[currIndex]
372. img = np.expand\_dims(img, axis=2)
373. img = np.concatenate((img, img, img), axis=-1).astype(np.uint8)
374. ret, thresh = cv2.threshold(inferData[currIndex], 0, 255, cv2.THRESH\_BINARY)
375. thresh = cv2.dilate(thresh, kernel=np.ones((5, 5), np.uint8), iterations=1)
376. contours, hierarchy = cv2.findContours(thresh, 1, 2)
377. # 绘制轮廓过程
378. img = cv2.drawContours(img, contours, -1, (0, 255, 0), 1)
379. return img
380. def addListInfo(self, inferData):
381. """
382. 增加列表信息
383. @param inferData:模型推理的结果
384. """
385. self.listWidget.clear()
386. d, h, w = inferData.shape
387. result = {}
388. for i in range(d):
389. img = inferData[i]
390. if np.sum(img > 0) != 0:
391. result[str(i)] = np.sum(img > 0)
392. result = sorted(result.items(), key=lambda x: x[1], reverse=True)
393. for key, value in result:
394. self.listWidget.addItem("层 " + str(int(key) + 1))
395. def changeLayer(self, item):
396. """
397. 点击列表自动展示该层
398. @param item: 控制层数的对象
399. """
400. self.currIndex = int(item.text().split(' ')[1]) - 1
401. if self.isInferSucceed:
402. self.showImg(self.drawContours(self.npImage, self.inferData, self.currIndex))
403. else:
404. self.showImg(self.npImage[self.currIndex])
405. if \_\_name\_\_ == "\_\_main\_\_":
406. app = QApplication(sys.argv)
407. main = MainWindow()
408. main.show()
409. sys.exit(app.exec\_())

FormUI.py 文件

1. # -\*- coding: utf-8 -\*-
2. # Form implementation generated from reading ui file 'SegGroundClassUI1.ui'
3. #
4. # Created by: PyQt5 UI code generator 5.15.4
5. #
6. # WARNING: Any manual changes made to this file will be lost when pyuic5 is
7. # run again.  Do not edit this file unless you know what you are doing.
8. from PyQt5 import QtCore, QtGui, QtWidgets
9. class Ui\_Form(object):
10. def setupUi(self, Form):
11. """
12. 定义窗体对象，并且设置基本参数
13. @param Form: 窗体对象
14. """
15. Form.setObjectName("Form")
16. Form.resize(1286, 794)
17. Form.setMaximumSize(QtCore.QSize(1286, 794))
18. self.gridLayout\_4 = QtWidgets.QGridLayout(Form)
19. self.gridLayout\_4.setObjectName("gridLayout\_4")
20. self.gridLayout\_5 = QtWidgets.QGridLayout()
21. self.gridLayout\_5.setObjectName("gridLayout\_5")
22. self.canvas = QtWidgets.QLabel(Form)
23. self.canvas.setMinimumSize(QtCore.QSize(512, 512))
24. self.canvas.setMaximumSize(QtCore.QSize(16777215, 16777215))
25. font = QtGui.QFont()
26. font.setPointSize(10)
27. self.canvas.setFont(font)
28. self.canvas.setMouseTracking(False)
29. self.canvas.setAutoFillBackground(False)
30. self.canvas.setFrameShadow(QtWidgets.QFrame.Plain)
31. self.canvas.setLineWidth(1)
32. self.canvas.setMidLineWidth(0)
33. self.canvas.setText("")
34. self.canvas.setObjectName("canvas")
35. self.gridLayout\_5.addWidget(self.canvas, 0, 1, 1, 1)
36. self.gridLayout\_8 = QtWidgets.QGridLayout()
37. self.gridLayout\_8.setObjectName("gridLayout\_8")
38. self.gridLayout\_6 = QtWidgets.QGridLayout()
39. self.gridLayout\_6.setObjectName("gridLayout\_6")
40. self.groupBox\_2 = QtWidgets.QGroupBox(Form)
41. self.groupBox\_2.setMinimumSize(QtCore.QSize(0, 0))
42. self.groupBox\_2.setMaximumSize(QtCore.QSize(16777215, 16777215))
43. font = QtGui.QFont()
44. font.setPointSize(10)
45. self.groupBox\_2.setFont(font)
46. self.groupBox\_2.setAlignment(QtCore.Qt.AlignCenter)
47. self.groupBox\_2.setFlat(True)
48. self.groupBox\_2.setObjectName("groupBox\_2")
49. self.gridLayout\_9 = QtWidgets.QGridLayout(self.groupBox\_2)
50. self.gridLayout\_9.setObjectName("gridLayout\_9")
51. self.verticalLayout\_2 = QtWidgets.QVBoxLayout()
52. self.verticalLayout\_2.setObjectName("verticalLayout\_2")
53. self.horizontalLayout = QtWidgets.QHBoxLayout()
54. self.horizontalLayout.setObjectName("horizontalLayout")
55. self.label = QtWidgets.QLabel(self.groupBox\_2)
56. self.label.setObjectName("label")
57. self.horizontalLayout.addWidget(self.label)
58. self.line\_ww = QtWidgets.QLineEdit(self.groupBox\_2)
59. self.line\_ww.setMaximumSize(QtCore.QSize(100, 16777215))
60. self.line\_ww.setObjectName("line\_ww")
61. self.horizontalLayout.addWidget(self.line\_ww)
62. self.verticalLayout\_2.addLayout(self.horizontalLayout)
63. self.slider\_ww = QtWidgets.QSlider(self.groupBox\_2)
64. self.slider\_ww.setMinimum(-2000)
65. self.slider\_ww.setMaximum(2000)
66. self.slider\_ww.setOrientation(QtCore.Qt.Horizontal)
67. self.slider\_ww.setObjectName("slider\_ww")
68. self.verticalLayout\_2.addWidget(self.slider\_ww)
69. self.horizontalLayout\_2 = QtWidgets.QHBoxLayout()
70. self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")
71. self.label\_5 = QtWidgets.QLabel(self.groupBox\_2)
72. self.label\_5.setObjectName("label\_5")
73. self.horizontalLayout\_2.addWidget(self.label\_5)
74. self.line\_wc = QtWidgets.QLineEdit(self.groupBox\_2)
75. self.line\_wc.setMaximumSize(QtCore.QSize(100, 16777215))
76. self.line\_wc.setObjectName("line\_wc")
77. self.horizontalLayout\_2.addWidget(self.line\_wc)
78. self.verticalLayout\_2.addLayout(self.horizontalLayout\_2)
79. self.slider\_wc = QtWidgets.QSlider(self.groupBox\_2)
80. self.slider\_wc.setMinimum(-2000)
81. self.slider\_wc.setMaximum(2000)
82. self.slider\_wc.setOrientation(QtCore.Qt.Horizontal)
83. self.slider\_wc.setObjectName("slider\_wc")
84. self.verticalLayout\_2.addWidget(self.slider\_wc)
85. self.gridLayout\_9.addLayout(self.verticalLayout\_2, 0, 0, 1, 1)
86. self.gridLayout\_6.addWidget(self.groupBox\_2, 2, 0, 1, 1)
87. self.groupBox = QtWidgets.QGroupBox(Form)
88. self.groupBox.setEnabled(True)
89. self.groupBox.setMinimumSize(QtCore.QSize(250, 0))
90. self.groupBox.setMaximumSize(QtCore.QSize(16777215, 16777215))
91. font = QtGui.QFont()
92. font.setPointSize(10)
93. self.groupBox.setFont(font)
94. self.groupBox.setLayoutDirection(QtCore.Qt.LeftToRight)
95. self.groupBox.setInputMethodHints(QtCore.Qt.ImhNone)
96. self.groupBox.setAlignment(QtCore.Qt.AlignCenter)
97. self.groupBox.setFlat(True)
98. self.groupBox.setCheckable(False)
99. self.groupBox.setObjectName("groupBox")
100. self.gridLayout = QtWidgets.QGridLayout(self.groupBox)
101. self.gridLayout.setObjectName("gridLayout")
102. self.bn\_open = QtWidgets.QPushButton(self.groupBox)
103. self.bn\_open.setObjectName("bn\_open")
104. self.gridLayout.addWidget(self.bn\_open, 0, 0, 1, 1)
105. self.bn\_output = QtWidgets.QPushButton(self.groupBox)
106. self.bn\_output.setMaximumSize(QtCore.QSize(16777215, 16777215))
107. self.bn\_output.setObjectName("bn\_output")
108. self.gridLayout.addWidget(self.bn\_output, 4, 0, 1, 1)
109. self.bn\_infer = QtWidgets.QPushButton(self.groupBox)
110. self.bn\_infer.setObjectName("bn\_infer")
111. self.gridLayout.addWidget(self.bn\_infer, 2, 0, 1, 1)
112. self.bn\_loadModel = QtWidgets.QPushButton(self.groupBox)
113. self.bn\_loadModel.setMaximumSize(QtCore.QSize(16777215, 16777215))
114. self.bn\_loadModel.setObjectName("bn\_loadModel")
115. self.gridLayout.addWidget(self.bn\_loadModel, 1, 0, 1, 1)
116. self.text\_loadModel = QtWidgets.QLabel(self.groupBox)
117. self.text\_loadModel.setText("")
118. self.text\_loadModel.setObjectName("text\_loadModel")
119. self.gridLayout.addWidget(self.text\_loadModel, 5, 0, 1, 1)
120. self.gridLayout\_6.addWidget(self.groupBox, 0, 0, 1, 1)
121. self.groupBox\_3 = QtWidgets.QGroupBox(Form)
122. self.groupBox\_3.setMinimumSize(QtCore.QSize(0, 0))
123. self.groupBox\_3.setMaximumSize(QtCore.QSize(16777215, 16777215))
124. font = QtGui.QFont()
125. font.setPointSize(10)
126. self.groupBox\_3.setFont(font)
127. self.groupBox\_3.setAlignment(QtCore.Qt.AlignCenter)
128. self.groupBox\_3.setObjectName("groupBox\_3")
129. self.gridLayout\_2 = QtWidgets.QGridLayout(self.groupBox\_3)
130. self.gridLayout\_2.setObjectName("gridLayout\_2")
131. self.listWidget = QtWidgets.QListWidget(self.groupBox\_3)
132. self.listWidget.setObjectName("listWidget")
133. self.gridLayout\_2.addWidget(self.listWidget, 0, 0, 1, 1)
134. self.gridLayout\_6.addWidget(self.groupBox\_3, 3, 0, 1, 1)
135. self.gridLayout\_8.addLayout(self.gridLayout\_6, 0, 2, 1, 1)
136. self.gridLayout\_5.addLayout(self.gridLayout\_8, 0, 3, 1, 1)
137. spacerItem = QtWidgets.QSpacerItem(225, 20, QtWidgets.QSizePolicy.Fixed, QtWidgets.QSizePolicy.Minimum)
138. self.gridLayout\_5.addItem(spacerItem, 0, 2, 1, 1)
139. self.gridLayout\_4.addLayout(self.gridLayout\_5, 0, 1, 1, 1)
140. spacerItem1 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
141. self.gridLayout\_4.addItem(spacerItem1, 0, 0, 1, 1)
142. self.retranslateUi(Form)
143. QtCore.QMetaObject.connectSlotsByName(Form)
144. def retranslateUi(self, Form):
145. """
146. 定义显示的文字
147. @param Form: 窗体对象
148. """
149. \_translate = QtCore.QCoreApplication.translate
150. Form.setWindowTitle(\_translate("Form", "Form"))
151. self.groupBox\_2.setTitle(\_translate("Form", "肺窗"))
152. self.label.setText(\_translate("Form", "窗宽："))
153. self.label\_5.setText(\_translate("Form", "窗位："))
154. self.groupBox.setTitle(\_translate("Form", "模型:BiSeNetV2"))
155. self.bn\_open.setText(\_translate("Form", "加载数据"))
156. self.bn\_output.setText(\_translate("Form", "导出Mask"))
157. self.bn\_infer.setText(\_translate("Form", "推理"))
158. self.bn\_loadModel.setText(\_translate("Form", "加载模型"))
159. self.groupBox\_3.setTitle(\_translate("Form", "位置"))
160. if \_\_name\_\_ == "\_\_main\_\_":
161. import sys
162. app = QtWidgets.QApplication(sys.argv)
163. Form = QtWidgets.QWidget()
164. ui = Ui\_Form()
165. ui.setupUi(Form)
166. Form.show()
167. sys.exit(app.exec\_())
168. from mainUI import Ui\_Form as A\_Ui *# a界面的库*
169. from classification import Ui\_Form as B\_Ui *# b界面的库*
170. from PyQt5.QtCore import \*
171. from PyQt5.QtGui import \*
172. from FormUI import Ui\_Form
173. from paddleseg.models import BiSeNetV2
174. import paddleseg.transforms as T
175. from paddleseg.core import infer
176. import paddle
177. import SimpleITK as sitk
178. import os
179. import warnings
180. from classification import Ui\_Form as Class\_Ui
181. from PyQt5.QtWidgets import QFileDialog
182. from PyQt5 import  QtGui
183. from PyQt5.QtWidgets import \*
184. import numpy as np
185. import onnxruntime as ort
186. import torch
187. import cv2
189. warnings.filterwarnings('ignore')
190. from PyQt5 import QtCore, QtWidgets
191. import sys
193. class AUi(QtWidgets.QMainWindow, A\_Ui):
194. def \_\_init\_\_(self):
195. super(AUi, self).\_\_init\_\_()
196. self.setupUi(self)
198. class BUi(QtWidgets.QMainWindow, B\_Ui):
199. def \_\_init\_\_(self):
200. super(BUi, self).\_\_init\_\_()
201. self.setupUi(self)
203. def windowwc(sitkImage, ww=1500, wc=-550):
204. """
205. 主要用于设置窗宽窗位
206. @param sitkImage:SimpleITK图像数据
207. @param ww:窗宽窗位
208. @param wc:窗宽窗位
209. @return:sitkImage
210. """
211. min = int(wc - ww / 2.0)
212. max = int(wc + ww / 2.0)
213. intensityWindow = sitk.IntensityWindowingImageFilter()
214. intensityWindow.SetWindowMaximum(max)
215. intensityWindow.SetWindowMinimum(min)
216. sitkImage = intensityWindow.Execute(sitkImage)
217. return sitkImage

220. def readNii(path, ww, wc, isflipud=True, ):
221. """
222. 读取和加载数据。如果图像是上下翻转的，就将其翻转过来
223. @param path: 文件路径
224. @param ww:窗宽窗位
225. @param wc:窗宽窗位
226. @param isflipud: 是否需要翻转
227. @return: data
228. """
229. if type(path) == str:
230. img = windowwc(sitk.ReadImage(path), ww, wc)
231. else:
232. img = windowwc(path, ww, wc)
233. data = sitk.GetArrayFromImage(img)
234. if isflipud:
235. data = np.flip(data, 1)
236. return data

239. class InferThread(QThread):
240. """
241. 调用PyQt5.QtCore，建立一个任务线程类, 进行推理任务
242. """
243. *# 收集推理失败的信号*
244. signal\_infer\_fail = pyqtSignal()
245. *# 传递推理结果*
246. signal\_infer\_result = pyqtSignal(np.ndarray)
248. def \_\_init\_\_(self, sitkImage, model):
249. super(InferThread, self).\_\_init\_\_()
250. self.sitkImage = sitkImage
251. self.model = model
252. self.transforms = T.Compose([
253. T.Resize(target\_size=(512, 512)),
254. T.Normalize()
255. ])
257. def run(self):
258. """
259. 在启动线程后任务开始执行
260. """
261. try:
262. data = readNii(self.sitkImage, 1500, -500)
263. inferData = np.zeros\_like(data)
264. d, h, w = data.shape
266. for i in range(d):
267. img = data[i].copy()
268. img = img.astype(np.float32)
269. pre = self.nn\_infer(self.model, img, self.transforms)
270. inferData[i] = pre
272. self.signal\_infer\_result.emit(inferData)
273. except Exception as e:
274. print(e)
275. self.signal\_infer\_fail.emit()
277. def nn\_infer(self, model, im, transforms):
278. """
279. 预测结果
280. @param model: 模型参数
281. @param im: 图像数据
282. @param transforms:传入transforms方法
283. @return: 预测结果pred
284. """
285. img, \_ = transforms(im)
286. img = paddle.to\_tensor(img[np.newaxis, :])
287. pre = infer.inference(model, img)
288. pred = paddle.argmax(pre, axis=1).numpy().reshape((512, 512))
289. return pred.astype('uint8')
291. class MainWindow(QWidget, Ui\_Form):
292. def \_\_init\_\_(self, \*args, \*\*kwargs):
293. super().\_\_init\_\_(\*args, \*\*kwargs)
294. self.setupUi(self)
296. self.initUI()
297. self.setWindowTitle('肺炎检测助手')
298. *# 打开nii文件选择器*
299. self.bn\_open.clicked.connect(self.openFile)
300. *# 打开模型文件选择器*
301. self.bn\_loadModel.clicked.connect(self.openModleFile)
302. *# 推理按钮*
303. self.bn\_infer.clicked.connect(self.infer)
304. self.bn\_output.clicked.connect(self.outputFile)
306. self.sitkImage = object()
307. self.npImage = object()
308. *# 记录当前第几层*
309. self.currIndex = 0
310. *# 记录数据的最大层*
311. self.maxCurrIndex = 0
312. *# 记录数据的最小层，其实就是0*
313. self.minCurrIndex = 0
314. self.baseFileName = ''
315. self.isInferSucceed = False
317. *# 判断是否按下鼠标右键*
318. self.isRightPressed = bool(False)
320. self.model = object()
322. *# 判断模型是否加载成功*
323. self.isModelReady = False
325. *# 宽宽窗位滑动条*
326. self.slider\_ww.valueChanged.connect(self.resetWWWcAndShow)
327. self.slider\_wc.valueChanged.connect(self.resetWWWcAndShow)
329. *# 设置窗宽窗位文本框只能输入一定范围的整数*
330. intValidator = QIntValidator(self)
331. intValidator.setRange(-2000, 2000)
332. self.line\_ww.setValidator(intValidator)
333. self.line\_ww.editingFinished.connect(self.resetWWWcAndShow)
334. self.line\_wc.setValidator(intValidator)
335. self.line\_wc.editingFinished.connect(self.resetWWWcAndShow)
337. self.listWidget.itemDoubleClicked.connect(self.changeLayer)
339. def initUI(self):
340. try:
341. *# 定义展示的窗体及其初始的参数*
342. self.wwwcList = {'肺窗': [1700, -700]}
344. self.line\_ww.setText(str(1700))
345. self.line\_wc.setText(str(-700))
347. self.slider\_ww.setValue(1700)
348. self.slider\_wc.setValue(-700)
349. self.ww = 1700
350. self.wc = -700
352. self.currWw = self.ww
353. self.currWc = self.wc
354. except Exception as e:
355. print(e)
357. def openFile(self):
358. """
359. 打开医学影像文件选择器
360. """
361. try:
362. filename, \_ = QFileDialog.getOpenFileName(self,
363. "选取文件",
364. "./",
365. "Nii Files (\*.nii);;Nii Files (\*.nii.gz);;All Files (\*)")
366. if filename:
367. *# 清空列表*
368. self.listWidget.clear()
369. self.isInferSucceed = False
370. self.text\_loadModel.setText("数据加载完毕")
371. self.baseFileName = os.path.basename(filename).split('.')[0]
372. self.sitkImage = sitk.ReadImage(filename)
373. self.npImage = readNii(self.sitkImage, self.ww, self.wc)
374. self.maxCurrIndex = self.npImage.shape[0]
375. self.currIndex = int(self.maxCurrIndex / 2)
376. self.showImg(self.npImage[self.currIndex])
377. except Exception as e:
378. print(e)
380. def openModleFile(self):
381. """
382. 打开模型文件选择器
383. """
384. filename, \_ = QFileDialog.getOpenFileName(self, "选取文件", "./", "model Files (\*.pdparams)")
386. if filename:
387. try:
388. self.text\_loadModel.setText(" ")
389. num\_class = int(2)
390. self.model = BiSeNetV2(num\_classes=num\_class)
391. para\_state\_dict = paddle.load(filename)
392. self.model.set\_dict(para\_state\_dict)
393. self.text\_loadModel.setText("模型加载完毕")
394. self.isModelReady = True
395. except Exception as e:
396. self.text\_loadModel.setText("模型加载失败")
397. print(e)
399. def wheelEvent(self, event):
400. """
401. 定义鼠标滑轮事件
402. """
403. try:
404. if self.maxCurrIndex != self.minCurrIndex:
405. self.angle = event.angleDelta() / 8
406. self.angleY = self.angle.y()
407. if self.angleY > 0:
408. if self.currIndex < self.maxCurrIndex - 1:
409. self.currIndex += 1
410. if self.isInferSucceed:
411. self.showImg(self.drawContours(self.npImage, self.inferData, self.currIndex))
412. else:
413. self.showImg(self.npImage[self.currIndex])
414. elif self.angleY < 0:
415. if self.currIndex != self.minCurrIndex:
416. self.currIndex -= 1
417. if self.isInferSucceed:
418. *# self.npImage = self.drawContours(self.npImage, self.inferData)*
419. self.showImg(self.drawContours(self.npImage, self.inferData, self.currIndex))
420. else:
421. self.showImg(self.npImage[self.currIndex])
422. except Exception as e:
423. print(e)
425. def mousePressEvent(self, event):
426. """
427. 重载鼠标单机事件
428. """
429. *# 左键按下*
430. if event.buttons() == Qt.RightButton:
431. *# 左键按下(图片被点住),置Ture*
432. self.isRightPressed = True
433. self.preMousePosition = event.pos()
434. elif event.buttons() == Qt.MidButton | Qt.RightButton:
435. self.isRightPressed = False
437. def mouseReleaseEvent(self, event):
438. if event.button() == Qt.RightButton:
439. self.isRightPressed = False
441. def mouseMoveEvent(self, event):
442. """
443. 重载一下鼠标移动事件
444. """
445. try:
446. if self.maxCurrIndex != self.minCurrIndex:
447. *# 右键按下*
448. if self.isRightPressed:
449. *# 鼠标当前位置-先前位置=单次偏移量*
450. self.endMousePosition = event.pos() - self.preMousePosition
451. self.preMousePosition = event.pos()
452. ww = self.endMousePosition.x() + self.currWw
453. wc = self.endMousePosition.y() + self.currWc
454. if ww < -2000:
455. ww = -2000
456. elif ww > 2000:
457. ww = 2000
458. if wc < -2000:
459. wc = -2000
460. elif wc > 2000:
461. wc = 2000
462. self.currWw = ww
463. self.currWc = wc
464. self.slider\_ww.setValue(int(self.currWw))
465. self.slider\_wc.setValue(int(self.currWc))
466. self.line\_ww.setText(str(self.currWw))
467. self.line\_wc.setText(str(self.currWc))
468. self.resetWWWcAndShow()
469. except Exception as e:
470. print(e)
472. def showImg(self, img):
473. """
474. 显示图片
475. @param img: 待显示的图片
476. """
477. try:
478. if img.ndim == 2:
479. img = np.expand\_dims(img, axis=2)
480. img = np.concatenate((img, img, img), axis=-1).astype(np.uint8)
481. elif img.ndim == 3:
482. img = img.astype(np.uint8)
483. qimage = QImage(img, img.shape[0], img.shape[1], img.shape[1] \* 3, QImage.Format\_RGB888)
484. pixmap\_imgSrc = QPixmap.fromImage(qimage)
486. self.canvas.setPixmap(pixmap\_imgSrc)
487. except Exception as e:
488. print(e)
490. def resetWWWcAndShow(self):
491. """
492. 通过四个方式可以修改医学图像的窗宽窗位，每次修改后都会在界面呈现出来
493. """
494. if hasattr(self.sender(), "objectName"):
495. objectName = self.sender().objectName()
496. else:
497. objectName = None
498. try:
499. if objectName == '':
500. self.line\_ww.setText(str(1700))
501. self.line\_wc.setText(str(-700))
502. self.slider\_ww.setValue(1700)
503. self.slider\_wc.setValue(-700)
504. self.ww = 1700
505. self.wc = -700
506. self.currWw = self.ww
507. self.currWc = self.wc
508. if objectName == 'slider\_ww' or objectName == 'slider\_wc':
509. self.currWw = self.slider\_ww.value()
510. self.currWc = self.slider\_wc.value()
511. self.line\_ww.setText(str(self.currWw))
512. self.line\_wc.setText(str(self.currWc))
513. elif objectName == 'line\_ww' or objectName == 'line\_wc':
514. self.currWw = int(self.line\_ww.text())
515. self.currWc = int(self.line\_wc.text())
516. self.slider\_ww.setValue(self.currWw)
517. self.slider\_wc.setValue(self.currWc)
518. if self.maxCurrIndex != self.minCurrIndex:
519. self.npImage = readNii(self.sitkImage, self.currWw, self.currWc)
520. if self.isInferSucceed:
521. self.showImg(self.drawContours(self.npImage, self.inferData, self.currIndex))
522. else:
523. self.showImg(self.npImage[self.currIndex])
524. except Exception as e:
525. print(e)
527. def infer(self):
528. """
529. 模型分割预测
530. """
531. if self.maxCurrIndex != self.minCurrIndex and self.isModelReady:
532. self.bn\_infer.setEnabled(True)
533. *# 创建推理线程*
534. self.infer\_thread = InferThread(self.sitkImage, self.model)
535. *# 绑定推理失败的槽函数*
536. self.infer\_thread.signal\_infer\_fail.connect(self.infer\_fail)
537. *# 绑定推理成功的槽函数*
538. self.infer\_thread.signal\_infer\_result.connect(self.infer\_result)
539. self.infer\_thread.start()
540. self.text\_loadModel.setText("正在推理中······")
542. else:
543. QMessageBox.warning(self, "提示", "推理失败，推理前请先确保:\n1.加载模型\n2.加载数据", QMessageBox.Yes, QMessageBox.Yes)
545. def infer\_result(self, inferData):
546. """
547. 分割模型预测成功后，结果保存在self.inferData
548. @param inferData: 推理数据
549. """
550. *# 推理成功，并显示结果*
551. try:
552. self.inferData = inferData.astype(np.uint8)
553. QMessageBox.information(self, "提示", "模型推理成功!", QMessageBox.Yes, QMessageBox.Yes)
554. self.text\_loadModel.setText("推理完毕")
555. self.isInferSucceed = True
556. self.infer\_thread.quit()
557. self.addListInfo(self.inferData)
558. self.showImg(self.drawContours(self.npImage, self.inferData, self.currIndex))
559. except Exception as e:
560. print(e)
562. def infer\_fail(self):
563. """
564. 如果推理失败，则报错
565. """
566. QMessageBox.warning(self, "警告", "模型推理失败!", QMessageBox.Yes, QMessageBox.Yes)
568. def outputFile(self):
569. """
570. 将保存模型预测结果为nii格式文件
571. """
572. try:
573. if self.isInferSucceed:
574. filedir = QFileDialog.getExistingDirectory(None, "文件保存", os.getcwd())
575. if filedir:
576. *# 读取nii文件时转换np文件时对数据进行上下翻转，再输入模型推理，保存回nii文件时要翻转回来。*
577. self.inferData = np.flip(self.inferData, 1)
578. pre\_sitkImage = sitk.GetImageFromArray(self.inferData)
579. pre\_sitkImage.CopyInformation(self.sitkImage)
580. pre\_sitkImage = sitk.Cast(pre\_sitkImage, sitk.sitkUInt8)
581. save\_path = os.path.join(filedir, self.baseFileName + '\_mask.nii')
582. sitk.WriteImage(pre\_sitkImage, save\_path)
583. else:
584. QMessageBox.warning(self, "警告", "无进行过推理，无法保存！", QMessageBox.Yes, QMessageBox.Yes)
585. except Exception as e:
586. print(e)
588. def drawContours(self, npImage, inferData, currIndex):
589. """
590. 通过OpenCV将mask转换成轮廓绘制在原图上
591. @param npImage: 图像数据
592. @param inferData: 模型推理的结果
593. @param currIndex: 层数序号
594. @return: 绘制轮廓后的图片
595. """
596. img = npImage[currIndex]
597. img = np.expand\_dims(img, axis=2)
598. img = np.concatenate((img, img, img), axis=-1).astype(np.uint8)
599. ret, thresh = cv2.threshold(inferData[currIndex], 0, 255, cv2.THRESH\_BINARY)
600. thresh = cv2.dilate(thresh, kernel=np.ones((5, 5), np.uint8), iterations=1)
601. contours, hierarchy = cv2.findContours(thresh, 1, 2)
602. *# 绘制轮廓过程*
603. img = cv2.drawContours(img, contours, -1, (0, 255, 0), 1)
605. return img
607. def addListInfo(self, inferData):
608. """
609. 增加列表信息
610. @param inferData:模型推理的结果
611. """
612. self.listWidget.clear()
613. d, h, w = inferData.shape
614. result = {}
615. for i in range(d):
616. img = inferData[i]
617. if np.sum(img > 0) != 0:
618. result[str(i)] = np.sum(img > 0)
620. result = sorted(result.items(), key=lambda x: x[1], reverse=True)
621. for key, value in result:
622. self.listWidget.addItem("层 " + str(int(key) + 1))
624. def changeLayer(self, item):
625. """
626. 点击列表自动展示该层
627. @param item: 控制层数的对象
628. """
629. self.currIndex = int(item.text().split(' ')[1]) - 1
630. if self.isInferSucceed:
631. self.showImg(self.drawContours(self.npImage, self.inferData, self.currIndex))
632. else:
633. self.showImg(self.npImage[self.currIndex])

636. def predict(img\_path):
637. img = cv2.imread(img\_path)  *# 读取图片*
638. img = cv2.resize(img, (224, 224))  *# 调整图片尺寸*
640. img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  *# 把图片BGR变成RGB*
642. img = np.transpose(img, (2, 0, 1))  *# 调整维度将HWC - CHW*
643. img = np.expand\_dims(img, 0)  *# 添加一个维度 就是batch维度*
644. img = img.astype(np.float32)  *# 格式转成float32*
645. img /= 255
646. ort\_session = ort.InferenceSession("H:\\tongxinxitong\\home.onnx", providers=torch.device("cpu"))
647. *# 调用onnxruntime run函数进行模型推理*
648. outputs = ort\_session.run(
649. None,
650. {"image": img},
651. )
652. *# outputs的输出类型为list类型，所以要先将list转换成numpy再转换成torch*
653. outputs1 = torch.from\_numpy(np.array(outputs))
654. *# 通过softmax进行最后分数的计算*
655. value = float(torch.max(torch.softmax(outputs1[0], dim=1)))
656. *#outputs\_softmax = torch.softmax(outputs1[0], dim=1).numpy()[:, 0].tolist()[0]*
657. index = np.argmax(np.array(outputs))+1
659. return value,index
661. class CamShow(QMainWindow, Class\_Ui):
662. def \_\_init\_\_(self, parent=None):
663. super(CamShow, self).\_\_init\_\_(parent)
664. self.setupUi(self)
665. self.upload.clicked.connect(self.loadImage)
666. self.interference.clicked.connect(self.predict\_label)
667. *# 打开文件功能*
668. def loadImage(self):
669. self.fname, \_ = QFileDialog.getOpenFileName(self, '请选择图片', '.', '图像文件(\*.jpg \*.jpeg \*.png)')
670. self.result.setText(None)
671. self.score.setText(None)
672. if self.fname:
673. *#print(self.fname)*
674. *#self.Infolabel.setText("文件打开成功\n" + self.fname)*
675. *# self.Imglabel.set*
676. *#self.result.setText(self.fname)*
677. jpg = QtGui.QPixmap(self.fname).scaled(self.imglabel.width(), self.imglabel.height())
679. print(jpg)
680. self.imglabel.setPixmap(jpg)
681. else:
682. print("打开文件失败")
683. *#self.Infolabel.setText("打开文件失败")*
685. def predict\_label(self):
686. *# 开启线程*
687. *#self.result.setText(self.fname)*
688. if not self.fname:
689. self.result.setText("为空退出")
690. else:
691. value,index = predict(self.fname)
692. if index==1:
693. self.result.setText("新冠")
694. else :
695. if index==2:
696. self.result.setText("病毒性肺炎")
697. else:
698. if index==3:
699. self.result.setText("正常")
700. self.score.setText(str(value))
702. if \_\_name\_\_ == '\_\_main\_\_':
703. app = QtWidgets.QApplication(sys.argv)
704. a = AUi()
705. a.show()
706. b = CamShow()
707. c = MainWindow()
708. *# button是你定义的按钮*
709. a.Xray.clicked.connect(
710. lambda:{a.close(), b.show()}
711. )
712. a.CT.clicked.connect(
713. lambda: {a.close(), c.show()}
714. )
715. b.xrayback.clicked.connect(
716. lambda: {b.close(), a.show()}
717. )
718. c.CTback.clicked.connect(
719. lambda: {c.close(), a.show()}
720. )
721. a.setWindowIcon(QIcon("H:/tongxinxitong/aid90-ujb0o-001.ico"))
722. b.setWindowIcon(QIcon("H:/tongxinxitong/aid90-ujb0o-001.ico"))
723. c.setWindowIcon(QIcon("H:/tongxinxitong/aid90-ujb0o-001.ico"))
724. sys.exit(app.exec\_())