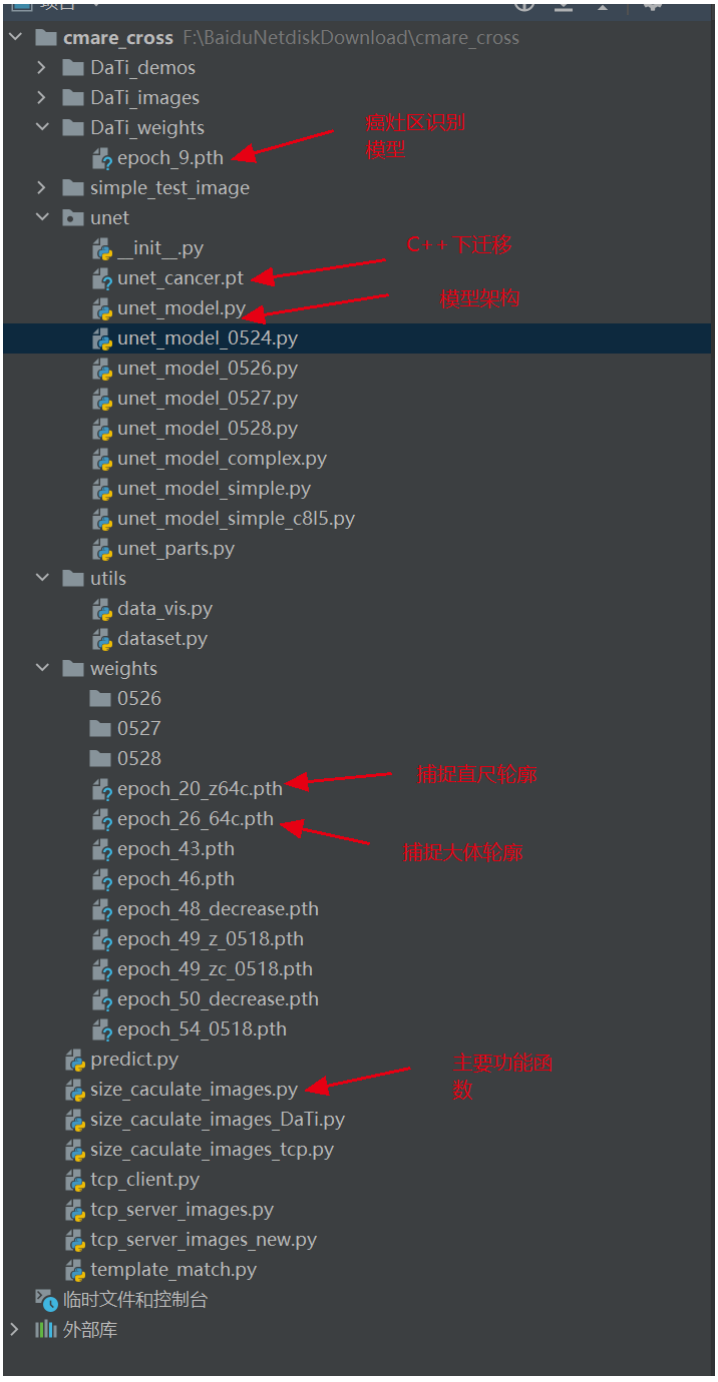
交接:

cmare_cross（大体轮廓识别）



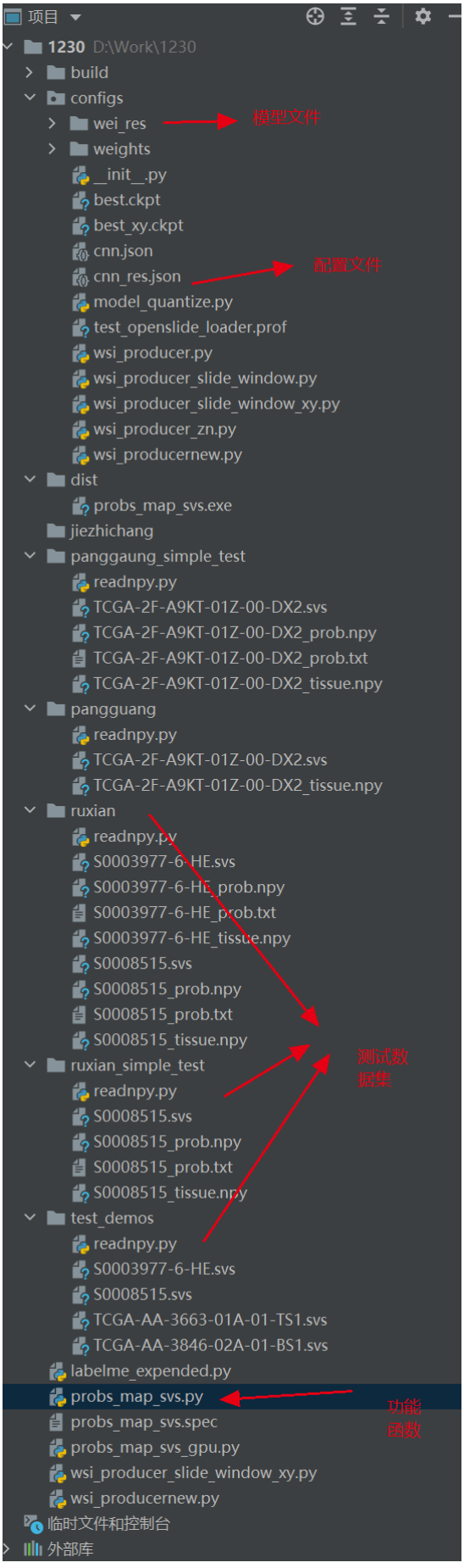
API:

- predict.py
  - ```
    """
    net: 为大体识别模型
    net_zc: 为直尺识别模型
    full_img: 检测图像
    device: 对应设备
    scale_factor: 缩放因子
    out_threshold: 预测阈值
    """
    ```
    ```

```
def predict_img(net,
                net_zc,
                net_cancer,
                full_img,
                device,
                scale_factor=1,
                out_threshold=0.5):
```

- size_caculate_images.py:
  - ```
    # 计算直尺的宽度比例,主要用于比例尺的计算工作
    """
    edges: 边缘信息
    cnts_zc: 直尺轮廓
    pixelPerMetric: 比例尺参数
    image: 原始图像
    """
    def process_contours_zc(edges, cnts_zc, pixelsPerMetric, image):
    ```
  - ```
    # 对癌灶的轮廓进行处理，并进行七点取样
    """
    cnts: 大体轮廓
    image： 原始图像
    pxielsPerMetric: 图像比例尺
    mask: 掩码
    whole_mean_color: 大体的整体颜色均值
    """
    def process_contours(cnts, image, pixelsPerMetric, mask,
    whole_mean_color, whole_cnt):
    ```
  - ```
    """
    找到轮廓的四个最值点
    """
    def find_largest_rectangle(points, arg_max_x, arg_min_x, arg_max_y,
    arg_min_y):
    ```
  - ```
    # 找到list中各个点之间的距离
    def find_clearest(points):
    ```
  - ```
    # 对相应的颜色进行捕捉
    def color_capture(image):
    ```
  - ```
    # 在图像中显示中文
    def cv2ImgAddText(img, text, left, top, textColor=(0, 255, 0),
    textSize=20):
    ```
  - ```
    # ***** 点到直线的距离:P到AB的距离*****
    # P为线外一点，AB为线段两个端点
    def getDist_P2L(PointP, Pointa, Pointb):
    ```

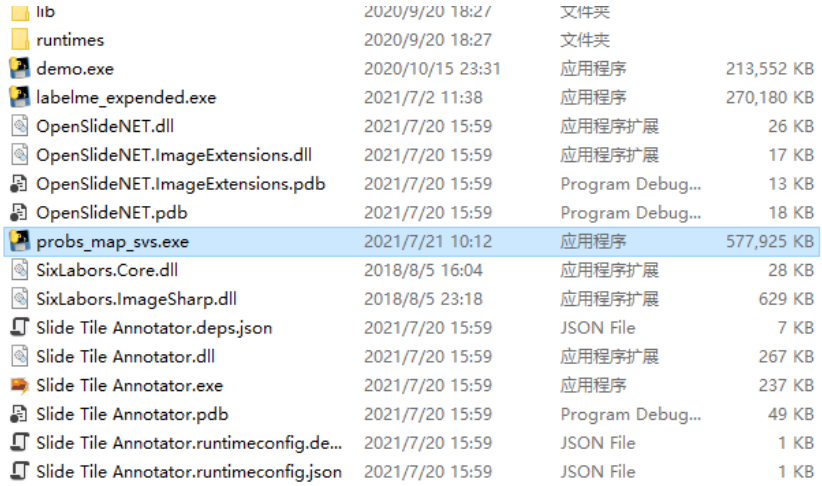
自动勾勒（1230）
    ```



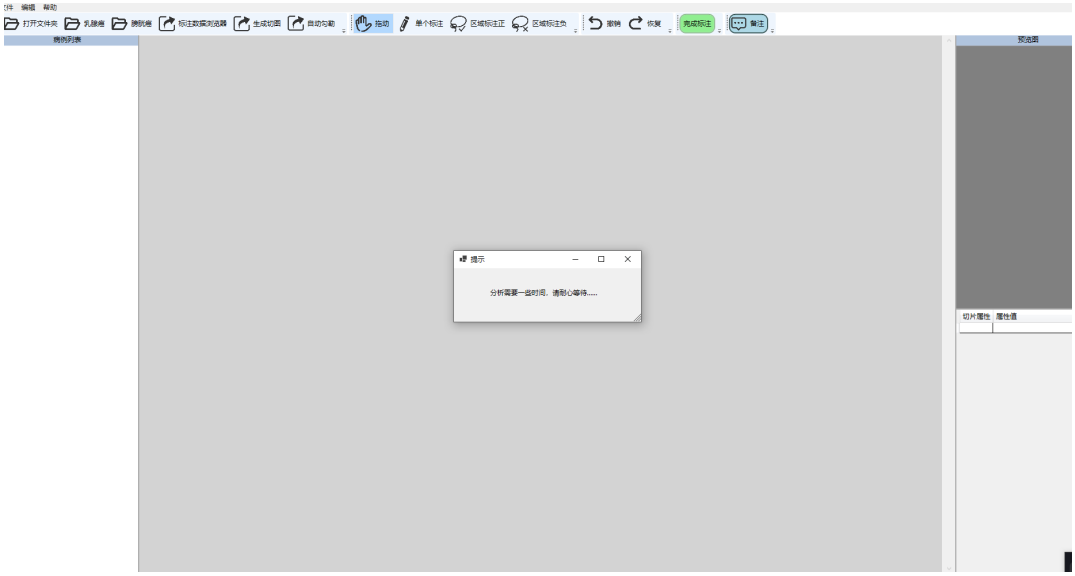
API

- probs_map_svs.py

    - ```
      """
      根据不同的情况选择不同的模型
      """
      def chose_model(mod):
      ```
    - ```
      # 转换颜色空间，ostu阈值分割，开闭形态学运算
      def get_mask_svs(args):
      ```
    - ```
      """
      获得概率图
      """
      def get_probs_map(model, dataloader):
      ```
    - ```
      '''
      线程池类
      '''
      class ThreadPool(object):
      ```

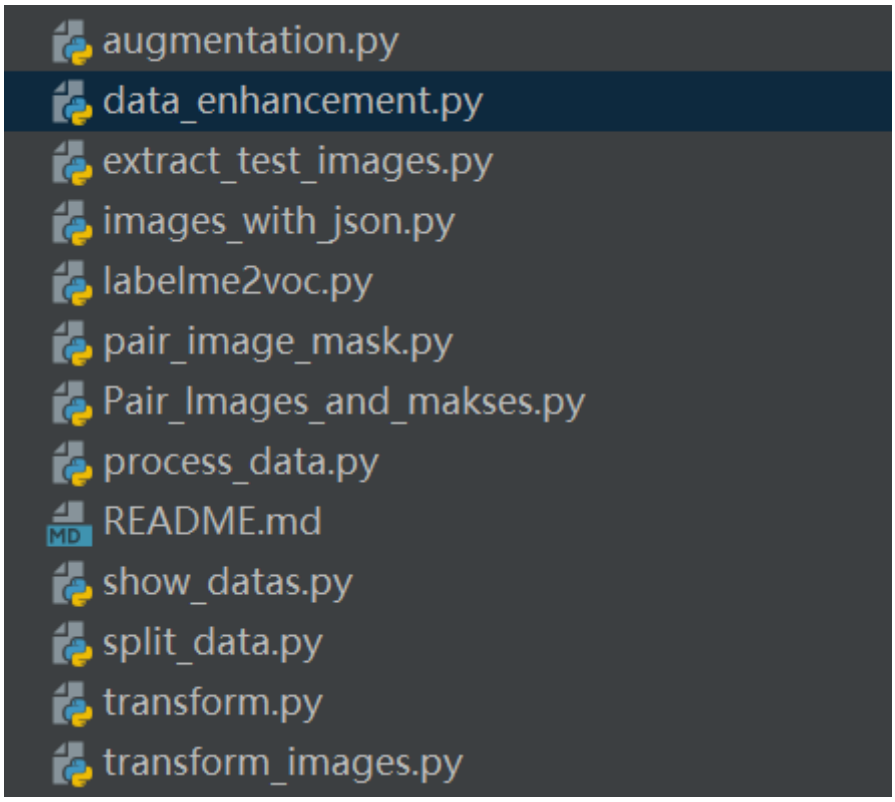
- wsi_producer_slide_window_xy.py
    - 该类是雪媛姐采用的数据加载及处理类
- wsi_producernew.py
    - 该类是赵娜姐采用的数据加载及处理类
- 根据不同的癌种来选取不同的模型以及不同的数据加载方式
- ```
  with open(cnn_path) as f:  # 加载对应的json文件
      cnn = json.load(f)
  if args.model == 'resnet50':
      ckpt = torch.load(r"D:\Work\1230\configs\wei_res\best_resnet50.ckpt", map_location
  elif args.model == 'resnet18':
      ckpt = torch.load(r"D:\Work\1230\configs\wei_res\best_resnet18.ckpt", map_location
  elif args.model == 'mobile_quantile':
      ckpt = torch.load(r"D:\Work\1230\configs\wei_res\mobile_quantile.ckpt")
  model = chose_model(args.model)  # 选择对应的模型
  # 非量化模型才能进行该处理
  ```
    - 由于路径参数都是绝对路径需要配置好
- 使用PyInstaller 进行python程序的exe打包，之后供c#调用
- 

    - slide Tile Annotator为C#打包好的主程序
    - prob_map_svs.exe为打包好供C#调用的py程序
  ```

- 

  - 主界面

label and expand data




```
* 第一步：  使用transform.py文件对原始的json文件进行转换
* 第二步：  使用Pair_Images_and_masks.py从转化好的json文件中取出对应的images和masks
* 第三步：  使用data_enhancement来对数据进行增强
```

Libtorch

API

- main.cpp

```
int main() 主程序入口

// 处理癌灶区轮廓
void  process_contours_cancer(vector<Point> contour, cv::Mat image, int i,
double pixelDistance, cv::Mat threshold, vector< vector<Point>> whole_contour,
String selected_model) {

// 处理大体轮廓
void  process_contours(vector<Point> contour, cv::Mat image, int i, double
pixelDistance, cv::Mat threshold) {

// 计算两个点之间的距离(int)
double twoPointDistance_int(Point point1, Point point2) {

// 计算两个点之间的距离（float）
double twoPointsDistance(cv::Point2f point1, cv::Point2f point2)

// 计算点到直线的距离
double pointToLinesDistance(cv::Vec4i lines1, cv::Vec4i lines2)

// 对直线段进行排序
bool compareLineIndex(cv::Vec4i lines1, cv::Vec4i lines2) {
    auto i = lines1[1];
    auto j = lines2[1];
    return (i < j);
}

// 对轮廓按照x的大小进行降序排序
bool compareContourIndex_x(Point point1, Point point2)

// 对轮廓按照x的大小进行降序排列
bool compareContourIndex_y(Point point1, Point point2)

// 比较候选点与轮廓之间的距离
Point2f compare_distance(vector<Point2f> candidate_points, vector<Point>
contour, string selecte_model)
```

- predict.h

```
// 根据输入图像和权重文件来取得对应的mask输出
Mat preprocess(Mat pil_img, float scale, String path, float out_threshold = 0.5)
```