USER MANUAL

Project: Automatic Organs Localization in 3D Medical Images

Yujie XIE

This project is built on a modified version of Caffe which supports the 3D Region Proposal Network. The model code has been compiled and passed on **Windows 10 (64 bits)** using **Visual Studio 2013**. The python codes have been compiled and passed on **python 3.8**.

How to build

Model Building Requirements: **Visual Studio 2013**, **ITK-4.10 (C++)**, **CUDA 10.0** and **cuDNN 7.6.5** (Ensure version of cuDNN is compatible with your CUDA version)

Scripts Running Requirements: **Python 3.8** (Python Library Requirements are summarized below) and **MATLAB R2020b**

Python Library	Version
ITK	5.1.1
SimpleITK	2.0.0
dicom2nifti	2.2.12
numpy	1.18.5
nibabel	3.1.1

The original code of the 3D RPN model is available on: https://github.com/superxuang/caffe_3d_faster_rcnn. The "caffe_3d_RPN**" model code was built and extended from this original version.

Steps:

1. Decompressed the Zip File. All source codes, sample results, sample model are inside this Zip File.
2. Decompressed the Zip File called "**caffe_3d_RPN.zip**".
3. In "**. \caffe_3d_RPN\windows\Caffe.bat"**, set **ITK_PATH** to ITK install path, which contains ITK **include**, **lib** folders.
4. Run "**. \caffe_3d_RPN\windows\Caffe.bat"** and build the project **caffe** in **Visual Studio 2013**. To conduct experiments on other projects like **convert_cifar_data**. You can directly build the whole solution.

NOTE:

- The model was built in the **debug mode** for easy debugging purpose. If you try to build the caffe project in the **release mode**, you need to change some visual studio setting.
- You need to **build the ITK(C++)** using same visual studio configuration.
- In VS2013, you may need to update **NugetPackage** before successfully build the caffe project.
- You can also follow the instructions on https://github.com/Microsoft/caffe to build caffe framework in Windows.

How to use

Dataset Name:

Abdominal CT Dataset = LiTS Dataset

Independent Abdominal CT Dataset = BTCV Dataset

Abdominal MR Dataset = CHAOS Dataset

Download Image Data and Prepare Annotations:

For LiTS CT images, please download and unzip the **training and testing CT images** from LiTS challenge. The annotation files for this dataset can be downloaded from IEEE DataPort

For BTCV CT images, please download and unzip the **raw CT training folders (including images and labels)** from BTCV challenge. For easy computation, please ensure the name conversions of CT images in BTCV challenge are consistent with the one in LiTS challenge testing set (e.g.test-volume-0.nii). To extract the annotation files for BTCV dataset, you need to re-orient the label images using "**reorient_BTCV_ct_label.py**", followed by running "**read_BTCV_ct_label.py**".

For CHAOS MR images, please download and unzip the **MR training folders** (**including images and labels**) from CHAOS challenge. Next, extract the MR images from MR subfolders, and run "**MRI_preprocessing.py**" to convert MR images from ***.dcm** format to ***.nii** format, and convert MR labels from ***.png** format to ***.nii** format. To extract the annotation files for CHAOS dataset, you can directly run "**read_MRI_label.py**".

Process Image Data:

To generate the gradient magnitude of CT images in LiTS and BTCV datasets with the Sobel operator, run "**gradient_CT.py**" to preprocess the raw CT images. To generate the gradient magnitude of MR images in CHAOS datasets with the Sobel operator, run "**gradient_MRI.py**" to preprocess the raw MRI images.

To generate multi-resolution gradient magnitude of medical images, run "**image_mulGM.m**". Note that, you should modify method parameters and directory depends on which image types and dataset you input.

Prepare Data:

Note that, through the processes mentioned above, all original and processed CT and MR images are in ***.nii** format. The input images should be in ***.mhd** format. Therefore, all images in ***.nii** format should be converted to ***.mhd** format using the script "**nifti2mhd.py**" before fed into the network.

All data folder for three different challenges should be organized in the same way. Here, we take the training set in LiTS challenge for an example. Move the CT images and the bounding-box annotations to a data folder and create an entry list file **(_train_set_list.txt)** in the same folder. To this end, the data folder is organized in the following way:

```
└── data folder
    ├── _train_set_list.txt
    ├── volume-0.mhd
    ├── volume-0.raw
    ├── segmentation-0.mhd
    ├── segmentation-0.raw
    ├── segmentation-0.txt
    ├── volume-1.mhd
    ├── volume-1.raw
    ├── segmentation-1.mhd
    ├── segmentation-1.raw
    ├── segmentation-1.txt
    │   ......................
    ├── volume-130.mhd
    ├── volume-130.raw
    ├── segmentation-130.mhd
    ├── segmentation-130.raw
    └── segmentation-130.txt
```

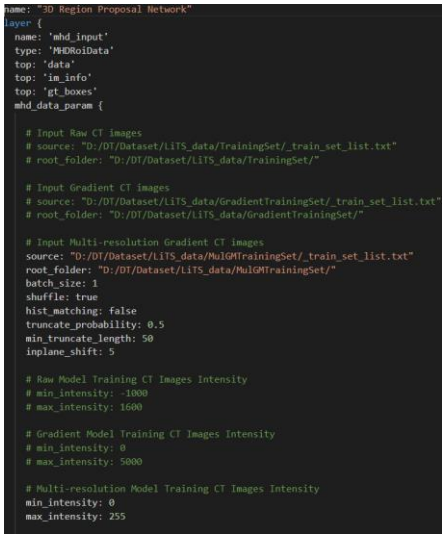
The entry list file **_train_set_list.txt** stores the filenames that are used for training. Each line of the entry list file corresponds a data sample. Here is an example of the entry list file **_train_set_list.txt** corresponding to above data folder. **Note that, the segmentation mask file (segmentation-N.mhd and segmentation-N.raw) is not necessary for neither training nor testing. You could just make the segmentation mask files absent and fill the second column of the entry list file with a non-existent filename.**

```
volume-0.mhd segmentation-0.mhd segmentation-0.txt
volume-1.mhd segmentation-1.mhd segmentation-1.txt
......................
volume-130.mhd segmentation-130.mhd segmentation-130.txt
```

Start the training

If you want to use sample model instead of training by yourself, you can move the model from Sample Model folder to ". **\caffe_3d_RPN\models\ours\snapshot". Or you can download more sample models from this link: "https://uniofnottm-my.sharepoint.com/:f:/g/personal/scyyx1_nottingham_ac_uk/EpnSnLq5mA5IlU8wNOzZ160BNgheQ59uzrv-dJGMQOM6wA?e=XbjEgh".

Modify the **source, root_folder** and **intensities** parameter of datalayer in "**.\caffe_3d_RPN\models\ours\net.prototxt**", according to your image directory and image type.



Then run "**.\caffe_3d_RPN \models\ours\train.bat**" to start training the model. (Note: If you run caffe project in release mode, you need to change the project location in 'train.bat')

Start the Testing

Prepare the testing file to ensure the data folder is organized as the example shown above.

Modify the test image **source**, **root_folder** and **intensity** parameters in **mhd_input** layer in "**. \caffe_3d_RPN \models\ours\net.prototxt**", according to your image directory and image type.

```
batch_size: 1
shuffle: true
hist_matching: false
truncate_probability: 0.5
min_truncate_length: 50
inplane_shift: 5

# Raw Model LiTS and BTCV CT testing images intensity
min_intensity: -1000
max_intensity: 1600

# Raw Model CHAOS MR testing images intensity
# min_intensity: 0
# max_intensity: 400

# Gradient Model (CT and MR) testing images intensity
# min_intensity: 0
# max_intensity: 5000
```

Modify the **source** and **roi_root_folder** parameters in **rpn_output** layer in "**.\caffe_3d_RPN \models\ours\net.prototxt**", according to your image directory and image type.

```
layer {
  name: 'rpn_output'
  type: 'RPNOutput'
  bottom: 'rpn_cls_prob_reshape'
  bottom: 'rpn_bbox_pred'
  bottom: 'im_info'
  rpn_output_param {
    anchor_size {
      width: 15
      width: 30
      width: 60
      width: 120
      height: 15
      height: 30
      height: 60
      height: 120
      length: 15
      length: 30
      length: 60
      length: 120
   }
    feat_stride_xy: 8
    feat_stride_z: 8
    rpn_min_size: 2

    # LiTS raw model result
    source: "D:/DT/Dataset/LiTS_data/Testing/img/_test_set_list.txt"
    roi_root_folder: "D:/DT/Dataset/LiTS_data/Testing/img/results/"

    # LiTS gradient model result
    # source: "D:/DT/Dataset/LiTS_data/GradientTesting/img/_test_set_list.txt"
    # roi_root_folder: "D:/DT/Dataset/LiTS_data/GradientTesting/img/results/"
```

Run"**.\caffe_3d_RPN\models\ours\test.bat**" to start testing the model. **Note that -iterations=num where num means number of testing images.** (Note: If you run caffe project in release mode, you need to change the project location in test.bat)

Evaluation

You can run either "**LiTS_ct_test_result.py**", "**BTCV_ct_test_result.py**" or "**CHAOS_mr_test_result.py**" depending on which dataset you use. "**crop_img.py**" helps you to visualize the final output to easily detect the accuracy.

** The original test results are included in zip file, containing both visualization results and quantitative results.

Following table summarize the functionality of different scripts. **Please modify the image path inside each script to your personal image storing directory.**

File Name	Purpose
BTCV_ct_test_result.py	Summarize the test result of BTCV challenge dataset
crop_img.py	Crop CT and MR images according to bounding box location.
gradient_CT.py	Generate the gradient magnitude of CT images using the Sobel operator
gradient_MRI.py	Generate the gradient magnitude of MR images using the Sobel operator
LiTS_ct_test_result.py	Summarize the test results of LiTS challenge dataset
image_mulGM.m	Generate the multi-resolution gradient magnitude of CT and MR images
MRI_preprocessing.py	Preprocessing MR images and labelled images to convert them from *. dcm and *. png to *. nii, respectively
CHAOS_mr_test_result.py	Summarize the test results of CHAOS challenge dataset
mulGM_normalization.py	Normalize CT and MR multi-resolution gradient magnitude
nifti2mhd.py	Convert images from NIFTI format to MHD format
read_BTCV_ct_label.py	Extract annotations for BTCV CT dataset
read_MRI_label.py	Extract annotations for CHAOS MR dataset
reorient_BTCV_ct_label.py	Reorient the label images in BTCV CT dataset