3 Steps Are All You Need to Achieve SOTA in MICCAI 2020 Thyroid Nodule Segmentation Challenge

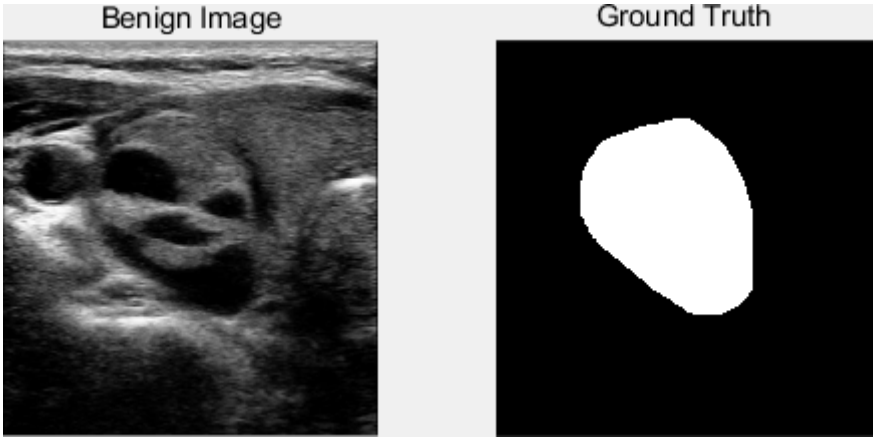
Segmentation is the most popular tasks in MICCAI 2020 challenges, including 15 out of 24 challenges. In this tutorial, we focus on the segmentation task in thyroid nodule segmentation and classification challenge (TN-SCUI 2020). In particular, we show how to use U-Net with 3 steps to achieve IoU 0.8199 on the official leaderboard, which is very close to the Top 1 score (0.8333, 500+ participants with 800+ submissions).

Task and Dataset

The target is to segment thyroid nodules from ultrasound (US) images.

- The training set consists of 3644 images with `png` format (1641 benign cases and 2003 malignant cases). The annotations are binary images with value `{0,255}`.
- The testing set consists of 910 images where 400 images are randomly selected as validation set, and 510 images are used for final ranking.

Let's show two examples in the training set.



Step 1. Preparing environment and training data

Our solution is based on U-Net with its great extension nnU-Net. Thanks to the out-of-the-box and flexible nature of nnU-Net, we can easily adapt the training set to the required dataset format of nnU-Net.

1.1 Installation

- Ubuntu 16.04 or 18.04
- Install PyTorch (1.3+)
- Install Nvidia Apex.

```
git clone https://github.com/NVIDIA/apex
cd apex
pip install -v --no-cache-dir ./
```

- Install nnU-Net

```
git clone https://github.com/MIC-DKFZ/nnUNet.git
cd nnUNet
pip install -e .
```

- Set path in `nnUNet/nnunet/paths.py`

```
# line 29:
base = 'your path to store training data, e.g.,
./MICCAI2020/nnUNetData'
# line 30
preprocessing_output_dir = 'your path to store
preprocessing data, e.g.,
./MICCAI2020/nnUNetData/pre_data' # SSD is highly
recommanded
# line 31
network_training_output_dir_base = 'your path to
save trained models, e.g., ./MICCAI2020/Models'
```

1.2 Preparing dataset

Create following folders

```
    MICCAI2020/
    ├── nnUNetData
    |   └── nnUNet_raw_data
    |       └── Task600_Thyroid2D
    |           └── imagesTr
    |           └── imagesTs
    |           └── labelsTr
    |
    ├── Models
    ├── OriData
    |   └── TNSCUI2020
    |       └── TNSCUI2020_train # directly decompressing
TNSCUI2020_train.rar
    |       └── tnscui2020_testset # directly
decompressing tnscui2020_testset.rar
```

nnU-Net is designed for 3D images with `nifti` format, while the data format in thyroid nodule task is the 2D image with `png` format. We can expand all the 2D images with an additional dimension and convert them to `nifti` format with this code. Now, the files in `MICCAI2020/nnUNetData/nnUNet_raw_data/Task600_Thyroid2D` are

```
    MICCAI2020/
    ├── nnUNetData
    |   └── nnUNet_raw_data
    |       └── Task600_Thyroid2D
    |           └── imagesTr
    |               ├── 2_0000.nii.gz
    |               ├── 4_0000.nii.gz
    |               ├── ..._0000.nii.gz
    |           └── imagesTs
    |               ├── test_1_0000.nii.gz
    |               ├── test_2_0000.nii.gz
    |               ├── test_..._0000.nii.gz
    |           └── labelsTr
    |               ├── 2.nii.gz
    |               ├── 4.nii.gz
    |               ├── ....nii.gz
    |           └── dataset.json # download from
https://github.com/JunMa11/TNSCUI2020/blob/master/Task600_
Thyroid2D/dataset.json
```

Data ready!

Next, we can train 2D U-Net models.

Step 2. Training five-fold cross validation models

We train five models for cross validation. Open terminal and run

```
nnUNet_train 2d nnUNetTrainerV2 Task600_Thyroid2D 0
nnUNet_train 2d nnUNetTrainerV2 Task600_Thyroid2D 1
nnUNet_train 2d nnUNetTrainerV2 Task600_Thyroid2D 2
nnUNet_train 2d nnUNetTrainerV2 Task600_Thyroid2D 3
nnUNet_train 2d nnUNetTrainerV2 Task600_Thyroid2D 4
```

The five trained models will be automatically saved in `MICCAI2020/Models/nnUNet/2d/Task600_Thyroid2D/nnUNetTrainerV2__nnUNetPlansv2.1/fold_0,1,2,3,4`

```
MICCAI2020/
├── Models
|   └── nnUNet
|       └── 2d
|           └── Task600_Thyroid2D
|               └── nnUNetTrainerV2__nnUNetPlansv2.1
|                   └── fold_0
|                       ├── model_final_checkpoint.model
|                       ├──
model_final_checkpoint.model.pkl
|                   └── fold_1
|                       ├── model_final_checkpoint.model
|                       ├──
model_final_checkpoint.model.pkl
|                   └── fold_2
|                       ├── model_final_checkpoint.model
|                       ├──
model_final_checkpoint.model.pkl
|                   └── fold_3
|                       ├── model_final_checkpoint.model
|                       ├──
model_final_checkpoint.model.pkl
|                   └── fold_4
|                       ├── model_final_checkpoint.model
|                       ├──
model_final_checkpoint.model.pkl
|                   └── plans.pkl #
```

Trained models will be publicly available in Github when the challenge submission is closed (31/7/2020).

Step 3. Inferring testing set and making a submission

Run

```
nnUNet_predict -i path to
MICCAI2020/nnUNetData/nnUNet_raw_data/Task600_Thyroid2D/im
agesTs -o path to
MICCAI2020/nnUNetData/nnUNet_raw_data/Task600_Thyroid2D/In
fer_imagesTs/ -t Task600_Thyroid2D -m 2d
```

The inference results are in

```
MICCAI2020/
├── nnUNetData
|   └── nnUNet_raw_data
|       └── Task600_Thyroid2D
|           └── Infer_imagesTs
|               ├── test_1.nii.gz
|               ├── test_2.nii.gz
|               ├── test_....nii.gz
```

Then, we convert the `nifti` files to `PNG` format.

```
import nibabel as nib
from skimage import io
seg_path = 'path to
MICCAI2020/nnUNetData/nnUNet_raw_data/Task600_Thyroid2D/In
fer_imagesTs/'
save_path = 'path to
MICCAI2020/OriData/TNSCUI2020/UNet_submission/'

for i in range(1, 911):
    seg = nib.load(join(seg_path,
'test_'+str(i)+'.nii')).get_fdata()
    seg_2d = seg_data[:,:,1]
    io.imsave(join(save_path, 'test_'+str(i)+'.PNG'),
seg_2d)
```

The most exciting moment comes!

Zip the folder `UNet_submission` and submit it to the official portal.

The results obtain IoU 0.819, which is very close to the Top 1 IoU 0.8333.

Remark:

> *All the code are available in Github and can be used out-of-the-box by simply setting the data path.*
>
> *It should be noted that when I write this tutorial, the challenge is still ongoing. Thus, it is not appreciate to make this tutorial and the corresponding Github repository publicly available now.*
>
> *I will make this tutorial publicly available when the challenge submission is closed (31/07/2020)!*