A nnUNet-based method for 3-dimensional intracranial hemorrhage segmentation

Team name : superembrace User name: Willinphila Real name :Liangwei
Department of Biomedical Engineering, Shantou University, Shantou, China

Abstract: U-Net, a method that came up in 2015 with an easy to understanding building principles, still shows its value in today's phase. After a period of training, the processing of the data is the aspect that I think is more important than the network construction. Therefore, I have chosen a variation of it, that's nnUNet which has achieved huge success in various medical images. In preprocessing phase, I focus my work on queue interception of grayscale values, downsampling and normalization, outward expansion of 3D images, and windowing. after my tuning, this model which only used 30 patients in training can now reach 16th place among 400+ teams with the surface dice in evaluation data of 0.5019, dice coefficient in evaluation data of 0.7112, and Relative Volume Difference in evaluation data of 0.274.

Keywords: Medical Image Segmentation, nnUNet , preprocessing

Introduction

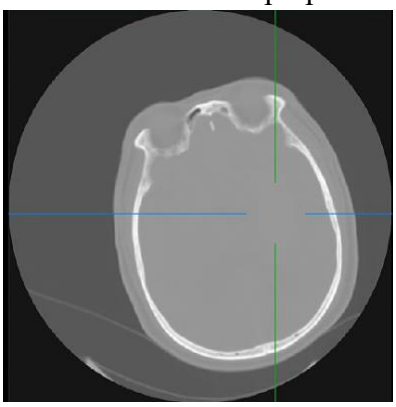
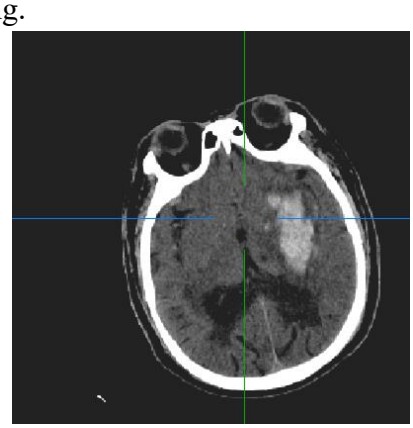
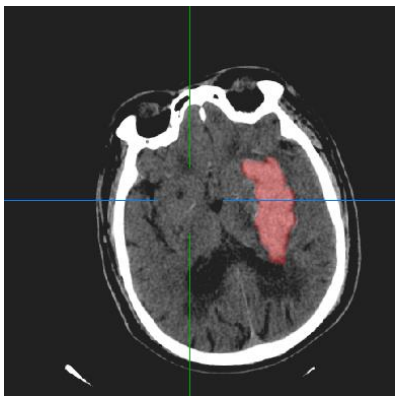
In the beginning, I first had a brief experience of the dataset through 2D Unet. Under the 2D model, I found that the connection between each layer was not strong enough, and often the location of the hemorrhage in the first layer was in the basal ganglia, while the location of the hemorrhage in the second layer appeared in the brainstem, so I decided not to adopt the 2D approach and pursue the 3D approach instead. However, during the time I struggled in 2D Unet, I found some valuable methods that I can still work on preprocessing part in 3D Unet. Since it was the first time to try 3D, I referred to the model on GitHub for research and found that the effect was much better than 2D, I firmly decided on my choice. After practicing in serval 3D models and master some trick, my tutor suggested I to try nnUNet.

nnU-Net is the first segmentation method that is designed to deal with the dataset diversity found in the domain. It condenses and automates the keys decisions for designing a successful segmentation pipeline for any given dataset which is quite convenient and suitable for me to do more study in processing data.

Method

Preprocessing:

In the training process, I first deal with the data in method windowing. Windowing, also known as grey-level mapping, contrast stretching, histogram modification, or contrast enhancement is the process in which the CT image greyscale component of an image is manipulated via the CT numbers; doing this will change the appearance of the picture to highlight particular structures. The brightness of the image is, adjusted via the window level. The contrast is adjusted via the window width. After thousands of observations in MITK, I decide to choose a width of 59 and a center of 96 for the img windowing. Because in most cases, this combo takes the best performance. It clearly shows both the area of bleeding at the edge of the brain and the area of bleeding in the brain stem. After windowing, in order to arrange the information of img, I used a threshold to ensure the gray value of the image in a certain standard interval, unified data input. Going through this part, the next part is downsampling. Downsampling the X and y axes, normalize the spacing of the slice axis to Slice_down_scale. The sampling layer is realized by pooling related techniques, aiming to reduce the dimension of features and retain effective information to avoid overfitting to a certain extent. But pooling is more than that. It aims to maintain rotation, translation, expansion, and expansion without deformation, etc. A sampling includes maximum sampling, average sampling, summation area sampling, and random area sampling. In the next phase, we extend the imported network silce one level out to make the data of the imported model orderly in depth. Here shows the whole process of preprocessing phase. nnUNet does the rest of the preprocessing.

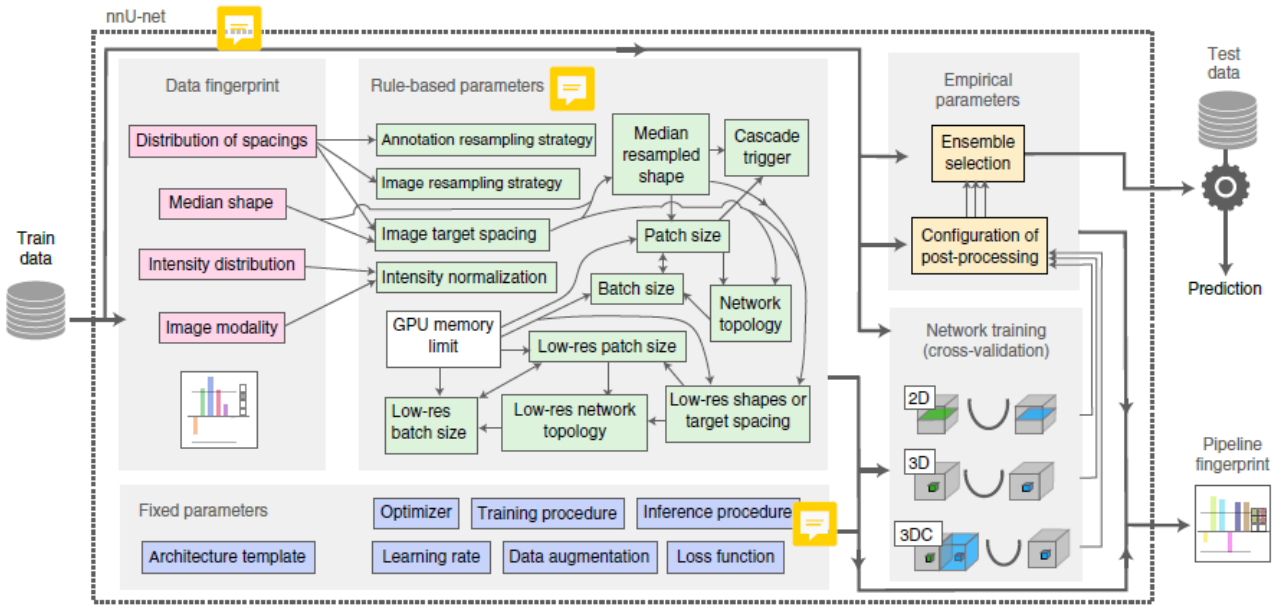
nnUNet also shows it ‘s value in preprocessing.
First, it deals with img with cropping. It works in several parts :1, Crop the data and seG according to nonzero_mask and remove the black edges. 2, Save the attributes as PKL files to speed up the later data loading. Continue to the next stage, nnUNet use data analyzer to 1, Count the size and spacing of all data in the dataset. 2, Calculate the intensity distribution of the dataset between different modalities 3, Calculate the percentage of the crop area. Then is the basic training framework of nnUNet, this module is common to most vision tasks, whether segmentation, detection, or classification, so you can modify this module to your own needs and add it to your own

tasks. The module includes train_val loops, tracking train, Val loss, and metrics, and uses early stopping to terminate training early and moving average to obtain smoother experimental results. nnUNet's subsequent training is based on this extension.

nnUNet architecture

U-Net is a classical encoder-decoder segmentation network and has drawn a lot of attention in recent years. The encoder pathway is similar to the typical classification network to extract more high-level semantic features layer by layer. Then the decoder pathway recovers the localization for every voxel and utilizes the feature information to classify it. To employ the position information embedded in an encoder, a direct connection is constructed between the layers in the same stage. In my project, I use a variant of UNet , that's nnUNet

a self-configuring method for deep learning-based biomedical image segmentation. Here is the flowchart explanation of nnUNet



Design choice	Required input	Automated (fixed, rule-based or empirical) configuration derived by distilling expert knowledge (more details in online methods)
Learning rate	–	Poly learning rate schedule (initial, 0.01)
Loss function	–	Dice and cross-entropy
Architecture template	–	Encoder–decoder with skip-connection ('U-Net-like') and instance normalization, leaky ReLU, deep supervision (topology-adapted in inferred parameters)
Optimizer	–	SGD with Nesterov momentum (μ = 0.99)
Data augmentation	–	Rotations, scaling, Gaussian noise, Gaussian blur, brightness, contrast, simulation of low resolution, gamma correction and mirroring
Training procedure	–	1,000 epochs × 250 minibatches, foreground oversampling
Inference procedure	–	Sliding window with half-patch size overlap, Gaussian patch center weighting
Intensity normalization	Modality, intensity distribution	If CT, global dataset percentile clipping & z score with global foreground mean and s.d. Otherwise, z score with per image mean and s.d.
Image resampling strategy	Distribution of spacings	If anisotropic, in-plane with third-order spline, out-of-plane with nearest neighbor Otherwise, third-order spline

Image target spacing	Distribution of spacings	If anisotropic, lowest resolution axis tenth percentile, other axes median. Otherwise, median spacing for each axis. (computed based on spacings found in training cases)
Network topology, patch size, batch size	Median resampled shape, target spacing, GPU memory limit	Initialize the patch size to median image shape and iteratively reduce it while adapting the network topology accordingly until the network can be trained with a batch size of at least 2 given GPU memory constraints. for details see online methods.
Trigger of 3D U-Net cascade	Median resampled image size, patch size	Yes, if patch size of the 3D full resolution U-Net covers less than 12.5% of the median resampled image shape
Configuration of low-resolution 3D U-Net	Low-res target spacing or image shapes, GPU memory limit	Iteratively increase target spacing while reconfiguring patch size, network topology and batch size (as described above) until the configured patch size covers 25% of the median image shape. For details, see online methods.
Configuration of post-processing	Full set of training data and annotations	Treating all foreground classes as one; does all-but-largest-component-suppression increase cross-validation performance? Yes, apply; reiterate for individual classes No, do not apply; reiterate for individual foreground classes

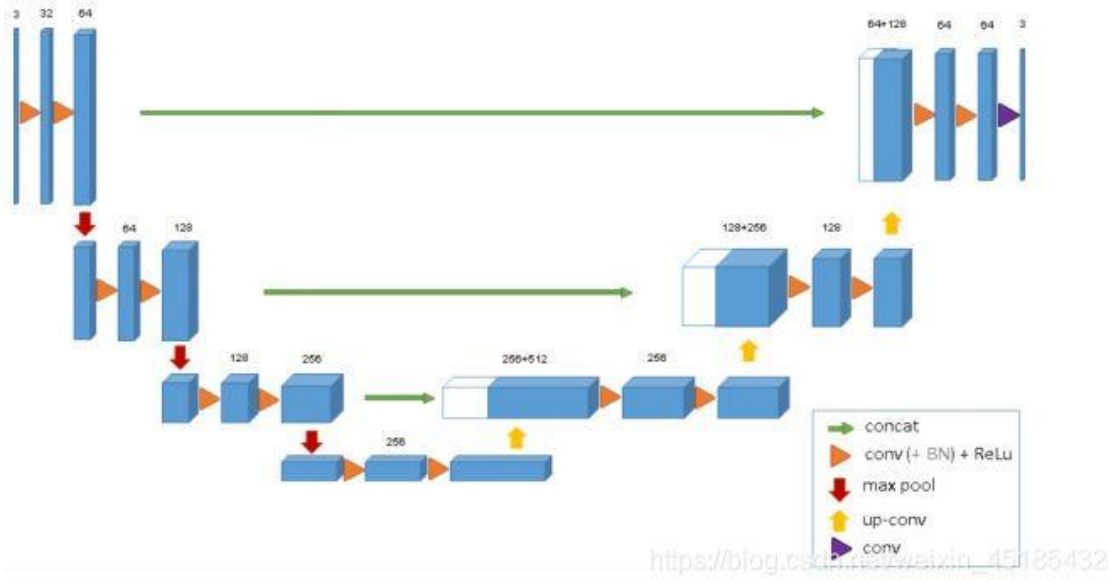
we mainly focus on two things in nnUNet, one is training strategies and the other is network structure.

Training strategies:

Train 1000 epochs, each epoch includes 250 mini batches; SGD + momentum (0.99) as optimizer; use 'poly' learning rate policy32; use CE loss + DICE loss as loss function ; GT was downsampled for deep supervision, weighting the losses for different resolutions, with exponential decay for lower resolutions and smaller weights; to deal with category imbalance, oversampling was used, with 66.7% of the samples coming from random locations in the selected training sample's, while 33.3% of the patches were guaranteed to contain one of the foreground classes present in the selected training sample (randomly selected). The number of foreground patches was rounded to force a minimum value of 1 (resulting in one random patch and one foreground patch with a batch size of 2). The data enhancements at the time of training are presented in the table.

Network structure:

Down-sampling until further down-sampling reduces the size of the feature map to less than 4 voxels, or the spacing of the feature map becomes anisotropic. The downsampling strategy is determined by the target spacing; high-resolution axes are downsampled separately until their resolution is within two factors of the low-resolution axes. Subsequently, all axes are downsampled simultaneously. Each axis is terminated separately until the respective feature map constraint is triggered. For 3D U-Net and 2D U-Net, the default kernel size for the convolution is 3 × 3 × 3 and 3 × 3, respectively. If there is an initial resolution difference between axes (defined as a spacing ratio greater than 2), the kernel size for the out-of-plane axes is set to 1 until the resolution is within a factor of 2. Note that for all axes, the convolution kernel size is kept at 3.



Blueprint parameters:

Structure Design

The different network architectures configured by nnUNet are all derived from the same network template, which is very similar to the original UNet and its 3D version. The theory is that a well-configured nnUNet would still be hard to beat, and proposed network architecture does not use any of the more recent and popular network structures, but only makes a few minor changes compared to the drastic changes made by the former. To accommodate the larger patch_size, nnUNet has designed the network's patch_size to be smaller. In fact, most 3D-Unet have a batch_size of only 2. What is Batch_normalization for? BN is used to speed up training while keeping it stable, and a small batch_size often does not reflect the value of BN. In this case, all nnUNet architectures use instance_normalization. furthermore, we use Leaky_Relu instead of Relu (with a slope of 0.01 on the negative axis)

Deep supervised training

Some changes in the network structure: they added additional losses to each layer of the decoder except for the bottom two layers in the decoding stage, which allows the gradient information to be injected deeper into the network while facilitating the training of all layers in the network. All Unet networks use the same operation at the same pixel level, both in the encoding area and in the decoding area (convolution →instance_normalization →Leaky_Relu). Downsampling is a convolution with step length and upsampling is a transposition operation of the convolution. To balance the training effect and memory consumption, the size of the initial feature_map is set to 32, which is halved if downsampling is done once, and doubled if upsampling is done. In order to limit the size of the final generated model, the number of feature_map is also limited, for example, 3D_Unet is limited to 320.

Conclusion

All in all, this project come up with a method 3D nnUNet model for intracranial hemorrhage, mainly putting strength in preprocessing. Proposed some image processing sessions for 3D images of intracranial hemorrhage. Creating a model that predicts areas of intracranial hemorrhage using a well-performing neural network This is my first time devoting myself to such a huge, creative, and entertaining competition, and also is my first year connected to coding, AI, deep learning, etc. I feel great honor to take part in this challenge, although it has puzzled me a lot. During the preparation time, I am always eager to get in touch with the skill and methods that I have never met before. Whether it's FCN or UNet or a more advanced way of structuring the network is constantly telling me that I have to keep trying to get to that unreachable faraway place. All in all, the time spent has almost reached the end and I‘m proud of myself to start my deep learning experience here. Finally, I would like to say a lot of thanks, first

of all to my family and my girlfriend, for giving me immense support and encouragement, and secondly, I would like to thank my mentor Xiang Yuan Ma who lead me in this field that I more than interested. Third, I would like to thank my seniors and we communicate with each other. He has helped me a lot.

Appendix

1, Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. (2020). nnU-Net: a self-configuring method
for deep learning-based biomedical image segmentation. Nature Methods, 1-9.