Team:LaTIM
Technical Report for MICCAI2022 Challenge: GOALS

Yihao Li; Rachid Zeghlache; Ikram Brahim;
El Habib Daho Mostafa

* INSERM, LaTIM, UMR 1101, France.

Abstract

OCT (Optical Coherence Tomography) is an extremely powerful tool for diagnosing ocular diseases, since it provides images of the main retinal structures in real time using a contactless, non-invasive method. Baidu's Glaucoma Oct Analysis and Layer Segmentation (GOALS) Challenge designs two tasks around OCT images: a segmentation task for the retinal nerve fiber layer and an automatic diagnosis task of glaucoma. In this challenge, we achieved excellent performance on two tasks using VNET and ResNet backbones. The sections in red are our updates that can provide reference and help with challenge papers.

Keywords: Glaucoma diagnosis, Deep learning, Computer-aided diagnosis

1 Introduction

Baidu's Glaucoma Oct Analysis and Layer Segmentation (GOALS) Challenge is an international ophthalmology competition held at MICCAI2022. The MICCAI conference is a comprehensive academic conference on medical image computing and computer assisted intervention. In conjunction with MICCAI 2022, Baidu will organize the 9th Ophthalmic Medical Image Analysis Workshop (OMIA9).

The leading cause of blindness is glaucoma. The glaucomatous neurodegeneration causes a disconnection between the retina and the brain, resulting in irreversible blindness. By 2040, around 111.8 million people are expected to suffer from glaucoma [9]. Using the coherence tomography (OCT) can help diagnose glaucoma. In OCT, images of the main retinal structures are acquired through a contactless, noninvasive method, providing real-time images of the main retinal structures. As opposed to color fundus images, which only provide surface information about the retina, OCT images can provide a cross-sectional view of the retina, so it can provide a more accurate assessment of the retinal structure. In the diagnosis of retinal and optic nerve disorders such as glaucoma, macular degeneration or diabetic retinopathy, layer thickness can be segmented and quantified. OCT is more effective than fundus color images in detecting early glaucoma cases [2].

In this challenge, we face two main tasks: (1) A segmentation task to determine the retinal nerve fiber layer, ganglion cellinner plexiform layer, and choroidal layer, which are helpful for diagnosis and differentiation of glaucoma; (2) An automatic diagnosis task of glaucoma.

In this challenge, we completed the segmentation and classification tasks by means of deep learning methods. The main contributions of this report are as follows.
(1) We have compared the performance of different backbones on the segmentation task;
(2) After several training sessions, the V-Net network we have used performs well on the segmentation task;
(3) We have further improved the segmentation performance by using the ensemble method with multiple checkpoint averaging;
(4) The proposed post-processing method can automatically detect some errors in the segmentation results and make modifications to them;
(5) The ResNet network we have used achieves excellent performance on the glaucoma detection task.

2 Summary and highlights of the methodology

2.1 Segmentation task

After comparing six traditional segmentation networks, we selected the original V-Net [6] as our segmentation network. Compared to the other participating teams, our V-Net is not a novel network structure, but it has performed reasonably consistently in the segmentation task and can serve as a baseline for subsequent segmentation tasks. Our V-Net could achieve a score of 8.7799 in the preliminary round.

The ensemble strategy involved selecting the five checkpoints that performed best on different validation sets, determining their weight coefficients, and integrating their results into a weighted average. Our ensemble strategy improved the result by 0.0036 (8.7835-8.7799) in the preliminary round.

In our approach, post-processing is crucial. Three laws can be observed in ground truth images based on the training set data (background 0, labels 1,2,3).

(1) There are two non-zero concatenated domains in all data;
(2) For each column of the image, there are 3 labels and the background data. Each column contains four values (0,1,2,3).
(3) For each label, the values between the highest and lowest levels are the same. The RNFL and GCIPL layers are also close.

Automated detection of these three rules was implemented using Python scripts. Several images failed to meet the requirements after the predicted results were tested. These errors can be divided into two categories: those resulting from similar data being present in the training set but not correctly classified in the test set and those resulting from prediction errors in the model, such as extra small regions or voids in the layers.

In order to solve the first type of error, we repeat similar images several times from the training set and add them to the training set so that the target-trained model is able to correctly identify the incorrect images. The results of the incorrect images are replaced with the results of the targeted training model. To correct the second type of error, we use common methods such as removing extra regions, filling in holes, smoothing, etc. By automatically detecting and correcting errors in post-processing, we have improved the results by 0.0251 (8.8086 - 8.7835).

Our training approach and implementations are shown in tables 1 and 2. Besides the default data augmentation (including random crop, random flip and random rotation.), RandomAffine and Rand2DElastic were also used.

Table 1: Summary of method for segmentation task.

Team	Architecture	Preprocessing	Ensemble	Post-processing
LaTIM	V-Net [6]	Resize to 800×1120, RandomAffine Rand2DElastic Default Data Augmentation	Pick 5 best checkpoints on 5 different validation set. Ensemble the results by taking the weighted average.	Replace results with target-trained models, Remove redundant areas, Fill holes, Smooth edges

RandomAffine: degrees=(-40, 40), translate=(0.4, 0.4), scale_ranges=(0.6, 1.4), shears=(-40, 40)
Rand2DElastic: spacing=(50, 50), magnitude_range=(5, 6), scale_range=(0.2, 0.2), translate_range=(100, 100)

2.2 Classification task

To improve model robustness, we train with Resnet50 and Resnet101, respectively. We used the pretrained model from the Timm library, and the model weights were based on the ImageNet dataset. Training is conducted using random cropping, and validation and test sets are conducted using center cropping. To ensemble the two models, we used the

Table 2: Configuration information for segmentation task.

Team	Optimizer	Loss	Batch_size	Epochs
LaTIM	Adam lr = 5e-4 weight_decay=1e-4 ExponentialLR (gamma = 0.99)	CrossEntropyLoss + DiceLoss + $0.1 \times$SurfaceDistanceLoss	4	800

averaging method based on the model that performs best on the different validation sets. We have used CrossEntropyLoss as the loss function, batch_size = 4, epochs = 300.

Table 3: Summary of classification task.

Team	Architecture	Preprocessing	Optimizer	Ensemble
LaTIM	pretrained Resnet50 pretrained Resnet101	Randomcrop to 648×648, Default Data Augmentation	Adam lr = 1e-4 weight_decay=1e-4 ExponentialLR (gamma = 0.99)	Pick 2 best models on 2 different validation folds. Ensemble the results by taking the average.

2.3 Conclusion and discussion

In this challenge, we performed segmentation and classification using deep learning algorithms. The V-Net network we used achieved a good performance on the segmentation task without using additional datasets. The performance of segmentation has been significantly improved through the use of model ensembles and targeted training post-processing methods. As evidenced by the finalists' results, many emerging models, especially the fusion of CNN and transformer, are achieving excellent results. We will use V-Net as the basis for further optimisation and testing of model architectures. The classification task might also be further improved with more data augmentation and model ensembles to prevent overfitting.

3 Methods

In the medical field, many deep learning algorithms have achieved excellent performance in various fields. The following networks have been selected for testing.

3.1 ResNet

Residual Networks, or ResNets, learn residual functions with reference to the layer inputs, instead of learning unreferenced functions. Residual nets let layers fit residual maps rather than hoping that they directly match an underlying mapping. It has been shown empirically that these types of networks are easier to optimize and can gain accuracy as depth increases [4].

3.2 U-Net

U-Net is a convolutional neural network that was developed for biomedical image segmentation at the Computer Science Department of the University of Freiburg. The network is based on the fully convolutional network and its architecture was modified and extended to work with fewer training images and to yield more precise segmentations [8].

3.3 BasicUNet

U-Net is a generic deep-learning solution for frequently occurring quantification tasks such as cell detection and shape measurements in biomedical image data. BasicUNet present an ImageJ plugin that enables non-machine-learning experts to analyze their data with U-Net on either a local computer or a remote server/cloud service. The plugin comes with pretrained models for single-cell segmentation and allows for U-Net to be adapted to new tasks on the basis of a few annotated samples [1].

3.4 SegResNet

SegResNet describes a semantic segmentation network for tumor subregion segmentation from 3D MRIs based on encoder-decoder architecture. Due to a limited training dataset size, a variational auto-encoder branch is added to reconstruct the input image itself in order to regularize the shared decoder and impose additional constraints on its layers. The SegResNet approach won 1st place in the BraTS 2018 challenge [7].

3.5 RegUNet

RegUNet presents a network and training strategy that relies on the strong use of data augmentation to use the available annotated samples more efficiently. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. It show that such a network can be trained end-to-end from very few images and outperforms the prior best method (a sliding-window convolutional network) on the ISBI challenge for segmentation of neuronal structures in electron microscopic stacks [8].

3.6 nnU-Net

The U-Net was presented in 2015. With its straight-forward and successful architecture it quickly evolved to a commonly used benchmark in medical image segmentation. The adaptation of the U-Net to novel problems, however, comprises several degrees of freedom regarding the exact architecture, preprocessing, training and inference. These choices are not independent of each other and substantially impact the overall performance. The nnU-Net ('no-new-Net'), which refers to a robust and self-adapting framework on the basis of 2D and 3D vanilla U-Nets. nnUnet argue the strong case for taking away superfluous bells and whistles of many proposed network designs and instead focus on the remaining aspects that make out the performance and generalizability of a method. 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 [5].

3.7 VNET

VNet is a UNet-based network that incorporates a residual block into the network. Residual linking helps the VNet training process to converge. In addition, they introduce a novel objective function, based on the Dice coefficient [6].

3.8 UNETR

Inspired by the recent success of transformers for Natural Language Processing (NLP) in long-range sequence learning, UNETR reformulate the task of volumetric (3D) medical image segmentation as a sequence-to-sequence prediction problem. They introduce a novel architecture, dubbed as UNEt TRansformers (UNETR), that utilizes a transformer as the encoder to learn sequence representations of the input volume and effectively capture the global multi-scale information, while also following the successful "U-shaped" network design for the encoder and decoder. The transformer encoder is directly connected to a decoder via skip connections at different resolutions to compute the final semantic segmentation output [3].

4 Material and experiments: Task1

4.1 Data

The dataset released by GOALS was provided by Sun Yat-sen Ophthalmic Center, Sun Yat-sen University, Guangzhou, China. The dataset contains 300 Circumpapillary Optical Coherence Tomography (Circumpapillary OCT). Data was divided into three equal groups, and 100 data were prepared for each step of the training process, the preliminary competition stage, and the final stage [2].

4.2 Backbone selection

Initially, we tested different backbones and then selected the one that performed the best for further training. In the preliminary round, 85 patients' data were selected as the training set and 15 as the validation set (random state = 42). To validate the performance of each backbone, we selected the best checkpoint based on the results of the validation set, and then used the checkpoint to make predictions on the pre-test set. Finally, we submitted our predictions to the AI Studio platform [1] to learn how well the backbone performed on the pre-test set.

Our training details for the backbone selection phase are as follows: (1) All backbones are provided by PyTorch's monai library. MONAI is a PyTorch-based, open-source framework for deep learning in healthcare imaging, part of PyTorch Ecosystem [2].
(2) The network input size is 800x800 pixels. Resize was used during training to resize the images without cropping. The ground truth image was resized using INTER_NEAREST.
(3) For data augmentation, we have used the affine transformation from the torchvision library.
(4) We have used DiceLoss + CrossEntropyLoss as the loss function and an Adam optimiser with an initial learning rate of 1e-3, accompanied by an ExponentialLR learning rate decay strategy.
(5) We have used the SurfaceDistanceMetric from the monai library as our criterion for selecting the best checkpoint. When comparing multiple image segmentations, performance metrics that assess how closely the surfaces align can be a useful difference measure. This group of surface distance based measures computes the closest distances from all surface points on one segmentation to the points on another surface, and returns performance metrics between the two.
(6) The batch size is 4 and the epoch is 1000.
(7) Due to nnU-Net's highly integrated system, it does not need to comply with the above configuration. It's our baseline. There were two versions tested. In nnU_Netv1, the 2d image was converted into a 3d volume with only one layer on the Z-axis and trained with

[1] `https://aistudio.baidu.com/aistudio/competition/detail/230/0/submit-result`
[2] `https://monai.io/`

nnU-Net, while in nnU-Net_v2, the original image was filtered with a Gaussian filter before being normalised.

The performance of each backbone is shown in Table I. This score was obtained with a perfect score for Task 2. Despite its excellent performance, nnU-Net is difficult to train reproducibly due to its highly integrated system and long training period. Finally, we chose VNET as the backbone of our segmentation. Training codes for the different backbones are available in the Backbone selection file.

Backbone	Score
nnU-Net_v1	8.7186
nnU-Net_v2	8.7297
BasicUNet	8.5938
SegResNet	8.5980
RegUNet	7.5690
VNET	8.6430
UNETR	8.3179

Table 4: Scores of different backbones.

4.3 VNET architecture

V-Net is shown as above [6]. The left part of the network consists of a compression path, while the right part decompresses the signal until its original size is reached.

The left side of the network is divided in different stages that operate at different resolutions. Each stage comprises one to three convolutional layers. At each stage, a residual function is learnt. The input of each stage is used in the convolutional layers and processed through the non-linearities and added to the output of the last convolutional layer of that stage in order to enable learning a residual function. This architecture ensures convergence compared with non-residual learning network such as U-Net. Replacing pooling operations with convolutional ones helps to have a smaller memory footprint during training, due to the fact that no switches mapping the output of pooling layers back to their inputs are needed for back-propagation.

For the right side of the network, the network extracts features and expands the spatial support of the lower resolution feature maps in order to gather and assemble the necessary information to output a two channel volumetric segmentation. At each stage, a deconvolution operation is employed in order increase the size of the inputs followed by one to three

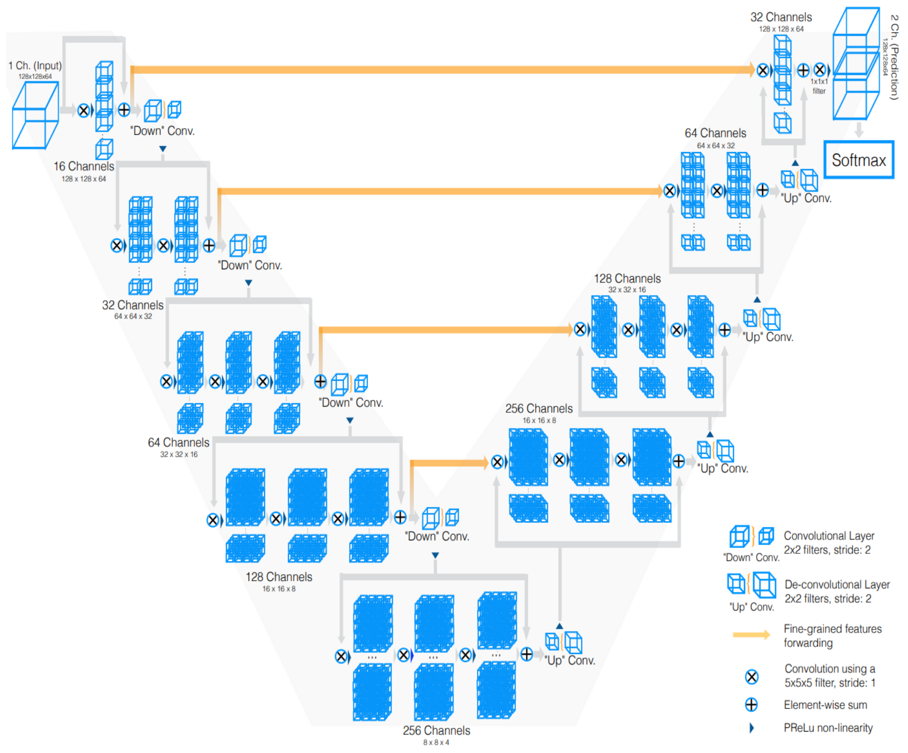

Figure 1: AUC curves of the different tests.

convolutional layers, involving half the number of 5×5×5 kernels employed in the previous layer.

Similar to U-Net, location information is lost in the compression path. Thus, the features extracted from early stages of the left part of the CNN are forwarded to the right part through horizontal connections. This can help to provide location information to the right part, and improve the quality of the final contour prediction.

As part of our task, we transformed it into a 2D architecture and changed the input size.

4.4 VNET Optimisation

We have optimized the VNET based on previous tests. The following areas have been optimized mainly. Following each optimization, the best performing checkpoint was selected to continue the training. Trainings did not fix the random_state used to divide the dataset.

(1) VNET_v1: Resize the input image to 800x1024 pixels to minimise the loss of image information.
(2) VNET_v2: Improved robustness of the model by adding data augmentation operations (Rand2DElastic from the monai library).
(3) VNET_v3: Changed the ratio of the training and validation sets (0.2).
(4) VNET_v4: Changed the ratio of the training and validation sets (0.3).
(5) VNET_v5: Resize the input image to 800x1088 pixels to minimise the loss of image information.

Optimisation	Score
VNET_v1	8.7105
VNET_v2	8.7130
VNET_v3	8.7270
VNET_v4	8.7418
VNET_v5	8.7626

Table 5: Scores of different backbones.

In these iterations, the goal is to maximize performance on the preliminary test set by maximizing data from the training set. It was not very meaningful to divide the validation set since the model had already learned all the data in the training set. The validation set was therefore discarded in the next test. According to previous tests, checkpoints with a smaller SurfaceDistanceMetric tend to perform better. As a result, we determined the best checkpoint by finding the minimum value of SurfaceDistanceMetric on the training set.
(6) VNET_v6: Resize the input image to 800x1120 pixels to minimise the loss of image information.
(7) VNET_v7: We used optuna to test different hyperparameter combinations and set three variables: the strength of the data augmentation, the optimisers, and the learning rate.

Optimisation	SurfaceDistanceMetric	Score
VNET_v6	8.7418	8.7737
VNET_v7	8.7626	8.7067

Table 6: Scores of different versions.

In the table above, the parameters found by optuna perform well on the training set, but overfit on the test set. It is important to find the checkpoint that minimises SurfaceDistanceMetric before overfitting. We added the SurfaceDistanceMetric to the loss function in subsequent tests and found our optimal point by continuously adjusting the weights.

(8) VNET_v8: SurfaceDistanceMetric has a weight of 0.1 in the loss function.
(9) VNET_v9: SurfaceDistanceMetric has a weight of 0.2 in the loss function.
(10) VNET_v10: SurfaceDistanceMetric has a weight of 0.3 in the loss function.
(11) VNET_v11: SurfaceDistanceMetric has a weight of 0.4 in the loss function.
(12) VNET_v12: SurfaceDistanceMetric has a weight of 0.5 in the loss function.
(13) VNET_v13: SurfaceDistanceMetric has a weight of 0.8 in the loss function.

Optimisation	SurfaceDistanceMetric	Score
VNET_v8	1.066	8.7799
VNET_v9	1.061	8.7730
VNET_v10	1.059	8.7751
VNET_v11	1.061	8.7750
VNET_v12	1.060	8.7764
VNET_v13	1.055	8.7656

Table 7: Scores of different versions.

The versions 8-12 all perform well, while the version 13 displays overfitting issues.

4.5 Ensemble

Five best checkpoints were selected: v8, v12, v10, v11, v6. We tested the results with different weights, but time constraints prevented us from exploring more possibilities. On the basis of the weights of each checkpoint, we calculate the mean of their output and then make predictions based on the results. Version 5 achieved the best performance.

Ensemble	Weights	Score
Ensemble_v1	0.5*v8 + 0.5*v12	8.7815
Ensemble_v2	0.33*v8 + 0.33*v12 +0.33*v10	8.7819
Ensemble_v3	0.25*v8 + 0.25*v12 +0.25*v10 + 0.25*v11	8.7822
Ensemble_v4	0.4*v8 + 0.3*v12 +0.2*v10 + 0.1*v11	8.7824
Ensemble_v5	0.3*v8 + 0.2*v6 + 0.2*v12 +0.15*v10 +0.15*v11	8.7835
Ensemble_v6	0.4*v8 + 0.15*v6 + 0.52*v12 +0.1*v10 +0.1*v11	8.7829

Table 8: Scores of different versions.

Checkpoints for the best five versions are included in the Ensemble file. Please contact the author if you need a checkpoint for a previous version.

4.6 Postprocessing

In the training set, all ground truth data satisfy the following three rules (background 0, labels 1,2,3).
(1) There are two non-zero concatenated domains in all data;
(2) For each column of the image, there are 3 labels and the background data. Each column contains four values (0,1,2,3).
(3) For each label, the values between the highest and lowest levels are the same. The RNFL and GCIPL layers are also close.

The automatic detection of these three rules was implemented using Python scripts and tested on the ground truth set of images. All images met the requirements. Postprocessing scripts can be found in check_train.ipynb in the Postprocessing file.

The predicted results were then tested, and many images failed to meet the requirements. In fact, we believe that most of these errors are caused by a small sample size for the training set. The curvature of the retina, for example, makes it difficult for the model to distinguish between the RNFL and GCIPL layers in 0114.png (Figure 2). The repetitive spacing on the right side of 0115.png (Figure 3) makes the model's assessment of the choroidal layer inaccurate. 0121.png's shading affects the model's judgement on the left (Figure 4).

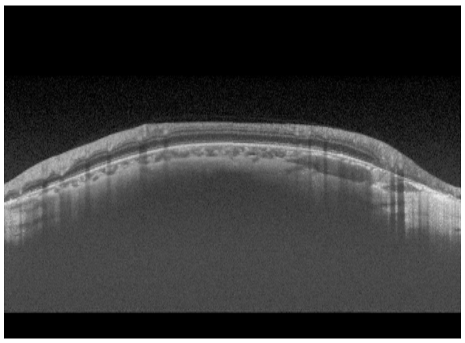

Figure 2: 0114.png

We artificially select some image data in the training set that have these problems and repeat them several times. For example, 0059.png and 0114.png are similar so we repeated 0059.png 20 times in the training set, 0047.png, 0061.png, and 0024.png also have repetitive intervals, and we repeated them 10, 5, and 5 times in the training set, respectively. We used the new dataset (100 patients, 140 images) for training, and we enhanced the data augmentation during training so that the model could enhance the judgement of the

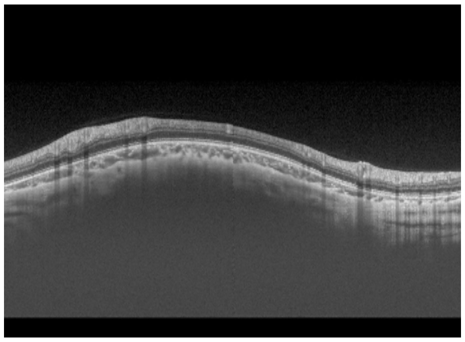

Figure 3: 0115.png

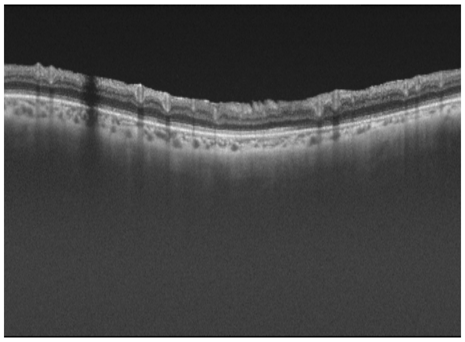

Figure 4: 0121.png

shaded parts.

In the Postprocessing file, re-train.py contains the training code. Unfortunately, the resulting checkpoint performs poorly, probably due to too many repetitions resulting in mediocre generalization of the model. Maybe we should reduce the number of repetitions or integrate it with other checkpoints. However, time constraints prevented us from conducting any other tests. Nevertheless, the new checkpoint can effectively solve the above problem. For those images that did not match the rules, we replaced them with the results generated by the new checkpoint.

There are still some images that do not meet the requirements after processing, so we detect them using automatic detection functions and then correct them using automated image processing functions.

We first detect those images that violate rule one. For example, if there are outliers in the 0158.png image (Figure 5), we remove these tiny outliers with the delet_small function. Function delet_small calculates the area of each concatenated domains and removes the concatenated domains whose area is below a threshold. The concatenated domains of outliers is often only a few pixels, so it is easy to detect.

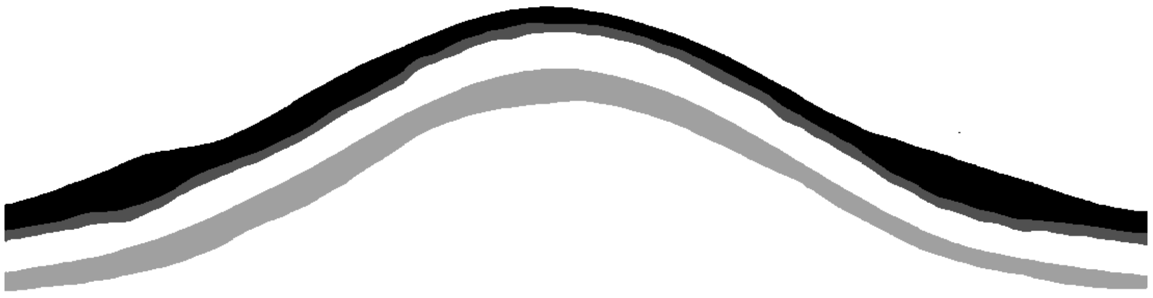

Figure 5: 0158.png violate rule one, there are outliers.

Then for images that violate rule three, we fill in the blank pixels. We detect the top and bottom layers of each label and fill in the middle with the corresponding labels. For example 0120.png (Figure 6).

As a final note, we note that the 0133.png (Figure 7) and 0143.png images have strange protrusions on the RNFL layer. The training data does not reflect a similar situation. In the training set, we determined that the gradient of the uppermost layer of the RNFL layer did not exceed five per pixel, so we modified it accordingly. We do not know if this is in accordance with the rules of the challenge, but its detection function and correction function are performed automatically. We have done a similar operation for the other layers.

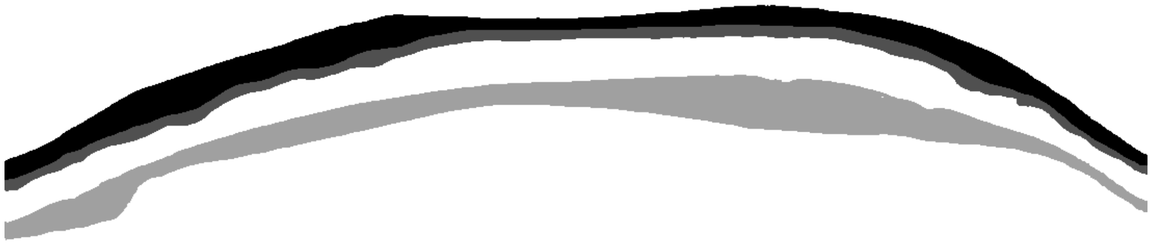

Figure 6: 0120.png violate rule three, there are blank pixels.

We first check the gradient of the top curve of the predicted data with the check_grad function, which returns points with anomalous gradients. Then for these anomalous points we change their gradients. We replace their gradients using random.uniform(0,0.35) and then smooth them using scipy.signal.savgol_filter and use the new curve as the upper bound of the RNFL layer. Figure 8 shows how the code works on 0133.png.

The postprocessing code can be found in check_test.ipynb in the Postprocessing folder. Exception images are detected and modified automatically.

4.7 Implementation details

In the training, we used four NVIDIA Tesla V100 32G graphics cards and one NVIDIA RTX A6000 48G graphics card. It takes less than 20 seconds for our VNET model to

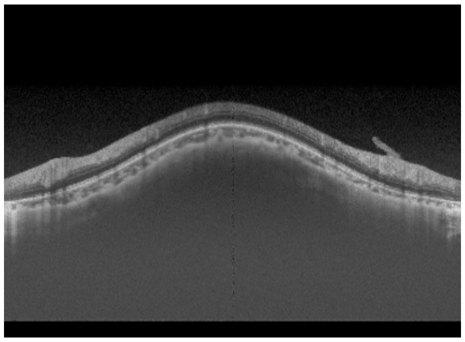

Figure 7: 0133.png have strange protrusions on the RNFL layer.

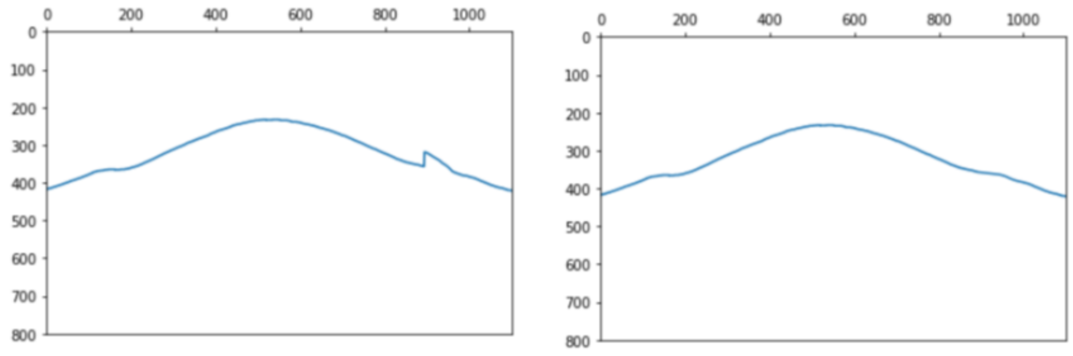

Figure 8: Automatic check/change of gradients, left: before, right: after.

predict 100 images.

5 Material and experiments: Task2

This is a binary classification for glaucoma detection. We have chosen the following configuration. In the preliminary round, 80 patients' data were selected as the training set and 20 as the validation set (random_state = 42).
(1) We used the pre-trained ResNet backbone from the timm library. PyTorch Image Models (timm) is a collection of image models, layers, utilities, optimizers, schedulers, data-loaders / augmentations, and reference training / validation scripts that aim to pull together a wide variety of SOTA models with ability to reproduce ImageNet training results.
(2) The network input size is 648x648 pixels. We use randomcrop for training and center-crop on the validation and test sets.

(3) For data augmentation, We have used random rotation and flip from the torchvision library.
(4) We have used CrossEntropyLoss as the loss function and an Adam optimiser with an initial learning rate of 1e-4, accompanied by an ExponentialLR learning rate decay strategy.
(5) We have used the accuracy as our criterion for selecting the best checkpoint.
(6) The batch size is 4 and the epoch is 300.

The Task file contains the ResNet50 and ResNet101 codes used for training. Because the task is relatively simple, the models converge quickly, and we choose a checkpoint with an accuracy of 1 for both the training and validation sets. Our final prediction is based on the mean value of 2 checkpoints output.

6 Conclusion

In this challenge, we performed segmentation and classification using deep learning algorithms. The VNET network we used achieved a good performance on the segmentation task without using additional datasets. And the ResNet network we used achieved a perfect score on the classification task. However, due to time constraints, we did not do more exploration, such as the optimisation of the ensemble algorithm. If the submission window opens up after the challenge, we will do deeper research on the segmentation task.

References

[1] Falk, T., D. Mai, R. Bensch, Ö. Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald, A. Dovzhenko, O. Tietz, C. Dal Bosco, S. Walsh, D. Saltukoglu, T. L. Tay, M. Prinz, K. Palme, M. Simons, I. Diester, T. Brox, and O. Ronneberger (2019, Jan). U-net: deep learning for cell counting, detection, and morphometry. *Nature Methods 16*(1), 67–70.

[2] Fang, H., F. Li, H. Fu, J. Wu, X. Zhang, and Y. Xu (2022). Dataset and evaluation algorithm design for goals challenge.

[3] Hatamizadeh, A., Y. Tang, V. Nath, D. Yang, A. Myronenko, B. Landman, H. Roth, and D. Xu (2021). Unetr: Transformers for 3d medical image segmentation.

[4] He, K., X. Zhang, S. Ren, and J. Sun (2015). Deep residual learning for image recognition.

[5] Isensee, F., J. Petersen, A. Klein, D. Zimmerer, P. F. Jaeger, S. Kohl, J. Wasserthal, G. Koehler, T. Norajitra, S. Wirkert, and K. H. Maier-Hein (2018). nnu-net: Self-adapting framework for u-net-based medical image segmentation.

[6] Milletari, F., N. Navab, and S.-A. Ahmadi (2016). V-net: Fully convolutional neural networks for volumetric medical image segmentation.

[7] Myronenko, A. (2019). 3d mri brain tumor segmentation using autoencoder regularization. In A. Crimi, S. Bakas, H. Kuijf, F. Keyvan, M. Reyes, and T. van Walsum (Eds.), *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, Cham, pp. 311–320. Springer International Publishing.

[8] Ronneberger, O., P. Fischer, and T. Brox (2015). U-net: Convolutional networks for biomedical image segmentation.

[9] Tham, Y.-C., X. Li, T. Y. Wong, H. A. Quigley, T. Aung, and C.-Y. Cheng (2014). Global prevalence of glaucoma and projections of glaucoma burden through 2040: A systematic review and meta-analysis. *Ophthalmology 121*(11), 2081–2090.