VRDL Final Project: Ultrasound Nerve Segmentation

Jia-Wei Liao, Kuok-Tong Ng, and Yi-Cheng Hung
Department of Applied Mathematics, NYCU

January 6, 2022

Abstract

In this project, we implement the deep learning-based techniques and proposed two novel methods Erosion Mask Smoothing (ELS) and adaptive Single Model Ensemble (ASME) to do ultrasound nerve segmentation. We use the UNet with the backbone of the ResNet family and EfficientNet. The UNet is the famous network for semantic segmentation. It is an encoder-decoder-based model. First, We try many backbones of the ResNet family but the performance is not good. And then, we use the EfficientNet which has been proposed in 2019. It's lightweight and accurate so that it is popular. We train the EfficientUNet and tune the hyper-parameter, then we get the test private score of 0.70111. After we use ASME, we beat the baseline and have the test private score of 0.72341. Our code is available at https://github.com/Jia-Wei-Liao/Ultrasound_Nerve_Segmentation.

1 Introduction

The introduction of this competition we selected is a semantic segmentation task. The background is under the post-surgery. Surgery inevitably brings discomfort, and oftentimes involves significant post-surgical pain. In the past, the way to decrease the pain is to inject an anesthetic but bring a bevy of unwanted side effects. This competition's sponsor committed to improving pain management through the use of indwelling catheters that block or mitigate pain at the source. Pain management catheters reduce dependence on narcotics and speed up patient recovery. It is a critical step in finding the exact location so can assist to install the device. The task in this competition is to segment a collection of nerves called the Brachial Plexus (BP) in ultrasound images. We are dealing with

ultrasound images data set which contained 5635 training images, masks and 5508 test images. The size of each gray-scale image is 580×420.

Our goal is to train a segmentation model to segment the BP in the image. We did the data prepossessing and proposed Erosion Label Smoothing (ELS), implement the UNet model based with different encoder. In pros-processing, we proposed a novel method named as Adaptive Single Model Ensemble (ASME) which be introduce in section 3. In competition, the score is evaluated on the mean Dice coefficient.

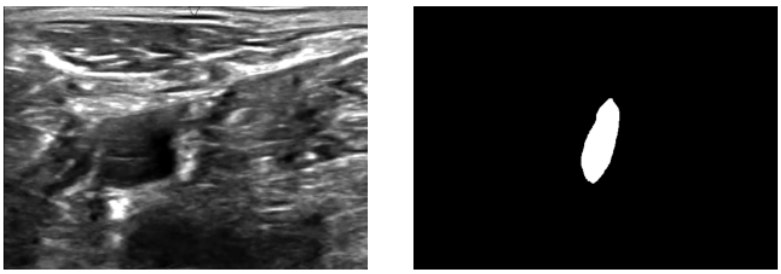


Figure 1. Ultrasonic image 2_77.tif image and mask 2_77_mask.tif

1.1 Data Exploration

We can observed in the mask distribution from the below bar chart, the sample of the mask without BP is more the contained BP image. That is to say, more than half of the ultrasonic data of the training set does not have a brachial plexus, which is worth noting. In addition, we have observed that there exists the same image, but the ground truth mask is completely different in the following Figure 4. In this pie chart present data imbalance.

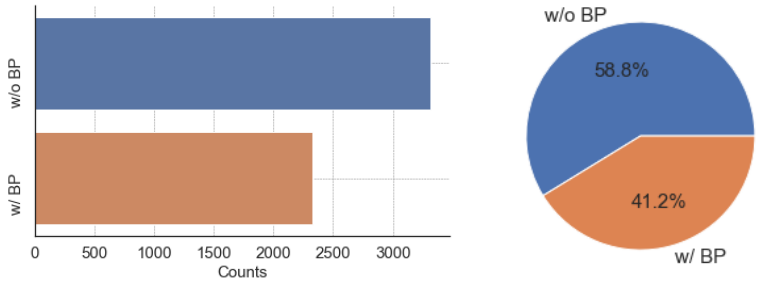


Figure 2. Bar chart and pie chart

We count down the number of non-zero pixels in the mask and display them as boxplots in Figure 3. The number of mask pixels is distributed in the range of 2,000 to 12,000, and each pixel is connected. The minimum is 2684.

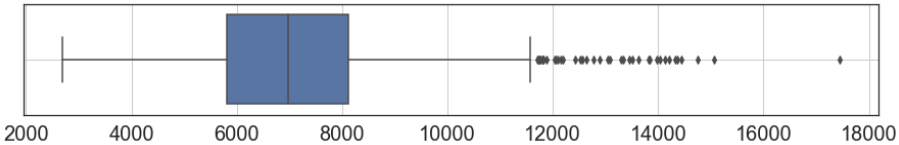


Figure 3. Box plot

1.2 The Difficulties

This task have the following difficulties:

1. Since the dataset contains images where BP is not present. Once we assign any pixel to BP class for those image, we will get zero dice score for those image.
2. Annotators were trained by experts. Most of us can't distinguish. The most difficult is there are many similar images in the dataset but some have BP, some don't. The images 36_49.tif and 36_50.tif on the Figure 4 are similar.
3. We discover there are 9 data which image is the same but the ground truth is different. The images 8_56.tif and 8_70.tif on the Figure 4 are the same (all the pixels are equal) but 8_56.tif have BP, 8_70 don't.

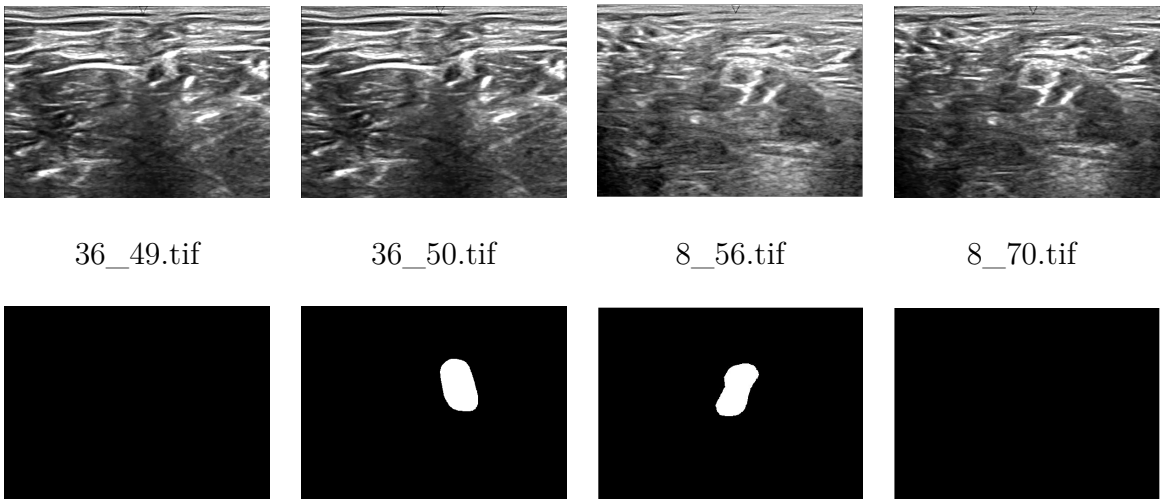


Figure 4. Training images and corresponding masks

2 Related Work

Semantic segmentation is an important topic in computer vision. In 2014, Fully Convolutional Networks (FCN) [5] has been proposed. FCN substitute the fully conneted layer (last layer) by transposed convolution layer in the classification model. Therefore, the model can accept variant size of input image and can be trained end-to-end. FCN has establish the foundation of semantic segmentation models. In 2015, UNet [6] has been proposed. UNet use a symmetric encoder-decoder structure, which means the decoder part is more robust than FCN. Moreover, skip connection between encoder and decoder

that can keeps the finer feature (from the encoder), mitigate the problem of information loss that FCN faced.

UNet has been widely used in medical image segmentation. There are many variant of UNet in this recent years. For example, inspired by ResNet and DenseNet, ResUNet [8] and DenseUNet [2] has been proposeed in 2018. nnUNet(no-new-Net) [4] proposed in 2019, which concentrate on data pre-processing, training scheme and inference-scheme and get a great improvement. This proves the importance of understanding of the data. TransUNet [1], which proposed in 2021, try to enhance the encoder part by combining CNN and transformer.

3 Proposed Approach

3.1 Data Pre-processing

- **Remove the same data:** There are 9 of the same image in the training set which has different annotation. We took off them to avoid the model happen ambiguity.
- **Splitting training and validation set:** The amount of data of training and validation set is 4:1. Let us denote the mask with/without BP as positive/negative mask, respectively. In order to encourage the model to distinguish positive and negative mask. We keep the ratio for positive and negative mask to 1:1 in training and validation dataset. However, the number of negative in the dataset is larger. We ignore the remaining negative mask.
- **Resize the image:** Since UNet model's encoder had repeated downsampling 5 times, the image size at least is $2^5 = 32$ or the feature will be vanish. In order to avoid unnecessary photo size issues and distortion, we adjusted the image size from (580, 420) to $(576, 448)$ so it can be divisible by 32 and maintained the similar photo ratio.

3.2 Data Augmentation

- **Random flip**
- **Random adjust brightness:** Set the bright value range from 0.9 to 1.1, and modified the grayscale, clipped the value into [0, 1] if out of range
- **Random add noise:** To increase the generalization of our model, We add the noise with probability by $I(i, j) + \sigma \cdot s$ where $s \sim \mathcal{N}(0, 1)$ and I is the grayscale.

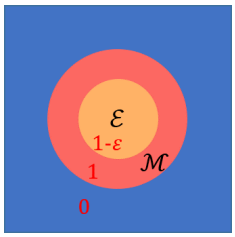
Comparing the random augmentation in the horizontal flip, vertical flip, rotation, brightness and add the Gaussian noise. The probability in $(0.5, 0.25, 0, 0.1, 0.1)$, respectively, would more suitable than other we had tried. We believe that this is related to the direction of data input. When scanning the data, there will be no rotation of 90 to read files, the image size isn't square, and the human body structure is symmetrical, so the probability of horizontal turning is higher.

3.3 Erosion Mask Smoothing (EMS)

In order to enhance the predict accuracy in boundary, we adjust the mask by using erosion morphological operation. It can highlight the boundary loss and relax the internal loss.

Let $\mathcal{M}$ be the mask, $\mathcal{E}$ be the region after eroding the mask with 9×9 kernel, and p be the pixel in the mask.

$$\tilde{\mathcal{M}}(p) = \begin{cases} 1-\varepsilon, & \text{if } p \in \mathcal{E} \\ 1, & \text{if } p \in \mathcal{M} \setminus \mathcal{E} \\ 0, & \text{if } p \in \mathcal{M}^c \end{cases}$$



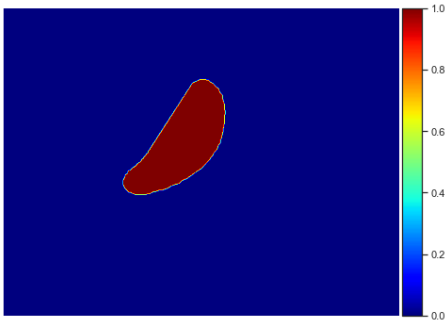


Figure 5. Origin mask

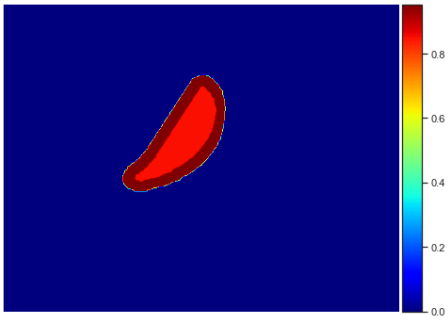


Figure 6. EMS

3.4 Model Architecture

In this task, we adopt a UNet based model. We will introduce the model architecture in this section. First, we may want to generalize the architecture of UNet to a abstract topological structure. As shown in the figure below, we may divide UNet architecture into three parts: Encoder, Bridge and Decoder. Therefore, in the next subsections, we will introduce them one by one.

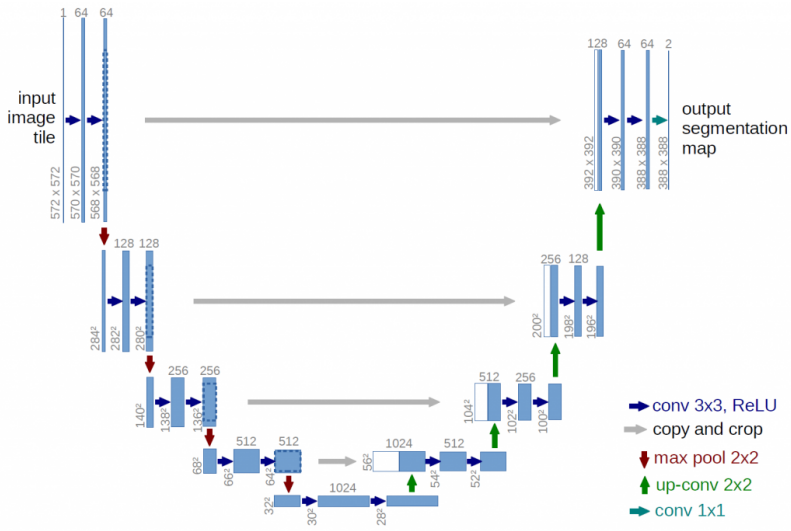


Figure 7. UNet architecture

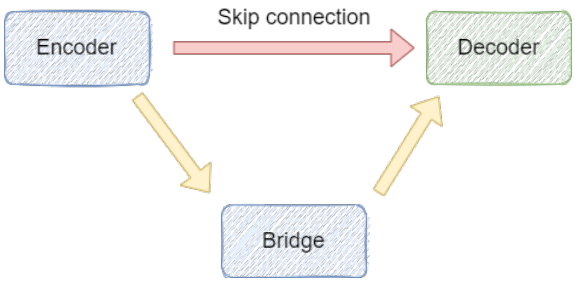


Figure 8. Abstract topological structure of UNet

3.4.1 Encoder and Bridge

The goal of the encoder is to extract useful features. However, it is not necessary to follow the encoder in the original UNet. Any CNN based networks (encoder) with down-sample steps can be used as an encoder. Therefore, we may choose pre-trained model as our encoder in order to get a better result. In this task, we choose ResNet [3] and EfficientNet [7] as our encoder.

ResNet [3] has been widely used as a pre-trained backbone. It adopted residual learning to get a faster convergence. As shown in Figure 7 and Figure 8. ResNet is composed by residual blocks. Moreover, the deeper network will get a better performance. However, the deeper network will be time consuming, we only try ResNet34 and ResNet50 in this task.

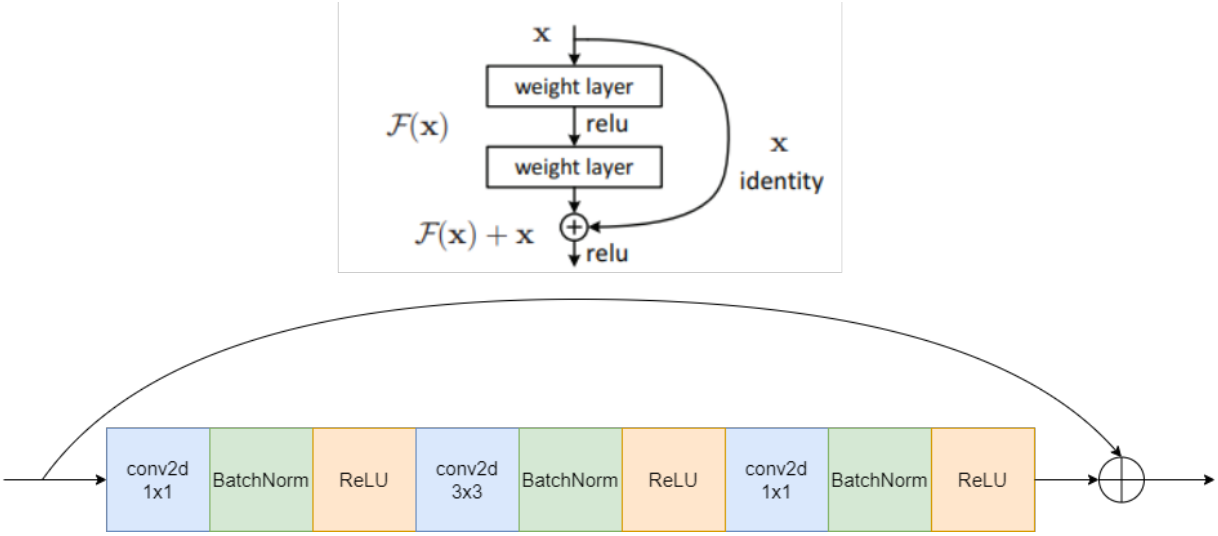


Figure 9. Residual learning

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 10. ResNet architecture

EfficientNet [7] is proposed in 2019. Scaling up convolution network is widely used to achieve better accuracy. For example, ResNet [3] scaling up the network's depth (ResNet18 to ResNet152). Wide Residual Network (WRN) [10] scaling up the network's width (increasing the number of output channels). Moreover, scaling up models by image resolution also gives a better accuracy too. The goal of EfficientNet is to scaling up depth/width/resolution simultaneously, as shown in Figure 11. Intuitively, scaling up all of them makes sense. Once input image become larger, the model needs more layers to ensure the receptive fields is large enough and more channels to capture more features. Finally, EfficientNet's accuracy gives a great accuracy, as shown in Figure 12. As we mentioned in section 3.1, our input image size (576, 448) is large. Therefore, it is reasonable to choose EfficientNet as an encoder.

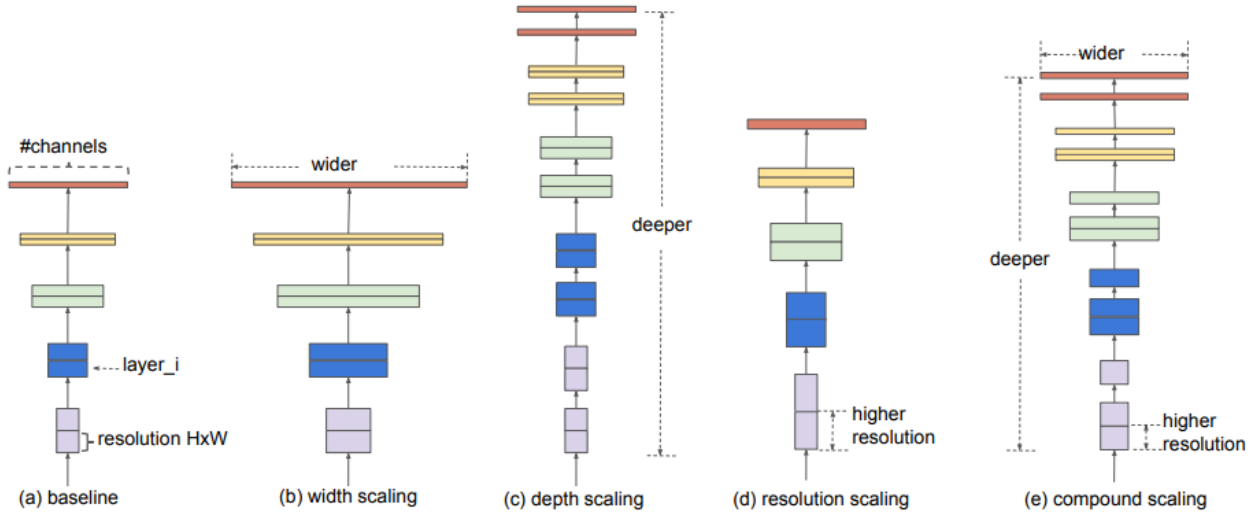


Figure 11. Model scaling

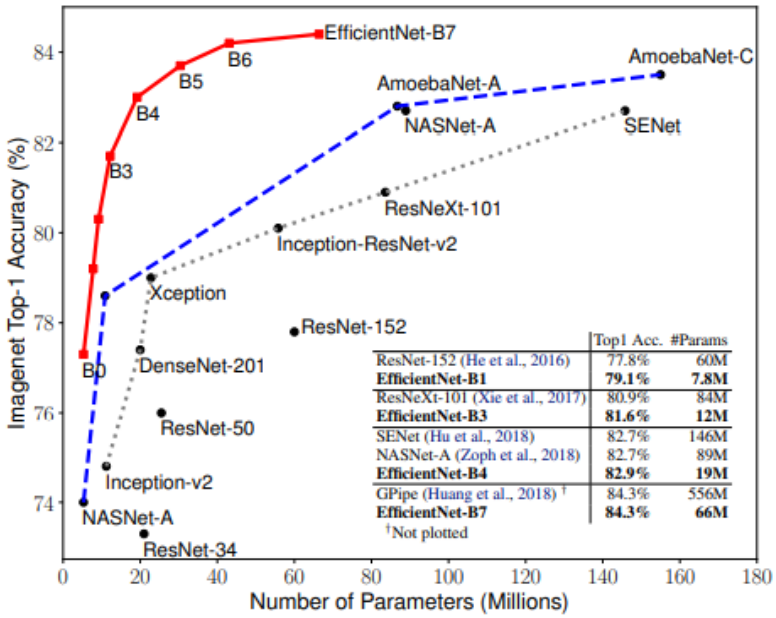


Figure 12. EfficientNet accuracy

Both ResNet and Efficient has down-sampled five times, with the identity mapping, we got six stages here, see Figure 13. The last stage of ResNet and EfficientNet will be treated as the Bridge in Figure 8.

3.4.2 Decoder

For the decoder part, we follow the original UNet architecture. The only difference is that the transpose convolution is replaced by nearest interpolation.

Combining the Encoder, Bridge and Encoder that we mentioned above, the model

architecture looks like the following Figure.

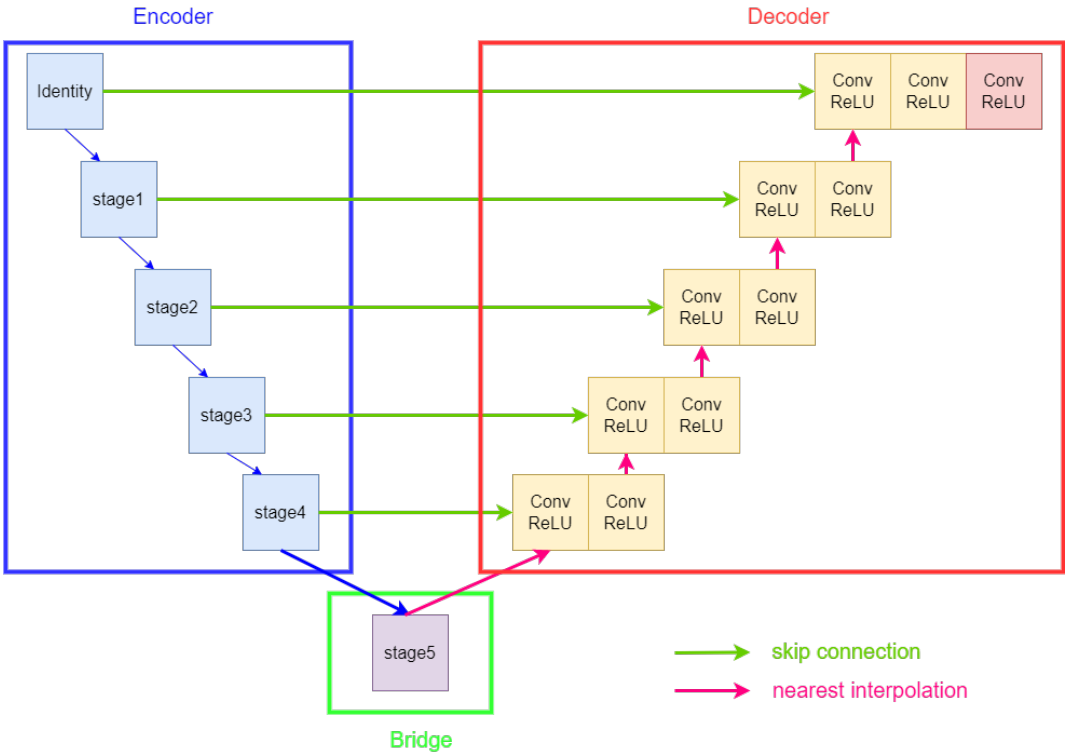


Figure 13. Model architecture

3.5 Loss Function

Let $Y, \hat{Y} \in \mathbb{R}^{448\times576\times2}$ be the ground truth with one-hot label and prediction. The last dimension of Y and $\hat{Y}$ indicate the category. We use the linear combination with dice loss and focal loss as the segmentation loss.

$$\mathcal{L}_{Seg}(\hat{Y}, Y) = \lambda_1 \mathcal{L}_{DSC}(\hat{Y}, Y) + \lambda_2 \mathcal{L}_{FL}(\hat{Y}, Y), \text{ with } \lambda_1, \lambda_2 > 0$$

We will introduce in detail as the following.

3.5.1 Dice Loss

The dice loss is defined as

$$\mathcal{L}_{DSC}(\hat{Y}, Y) = 1 - \frac{2 \cdot \sum_{i,j,c} Y_{i,j,c} \cdot \hat{Y}_{i,j,c}}{\sum_{i,j,c} Y_{i,j,c} + \sum_{i,j,c} \hat{Y}_{i,j,c}}$$

3.5.2 Focal Loss

The candidate object in a picture has most of the proportions of the background rather than the foreground, so there will be an imbalance in calculating the loss. The focal loss is down-weighted for the easy example. The hard example can be trained as much as

possible during the training process and ignored those easy examples. So it can solve the category imbalanced problem. The focal loss [9] is defined as

$$\mathcal{L}_{FL}(\hat{Y}, Y) = -\sum_{i,j} \alpha \left(1 - \hat{Y}_{i,j,C}\right)^{\gamma} \log \hat{Y}_{i,j,C}$$

where C is category of $Y_{i,j}$ and $\alpha, \gamma \geq 0$ are hyper-parameter.

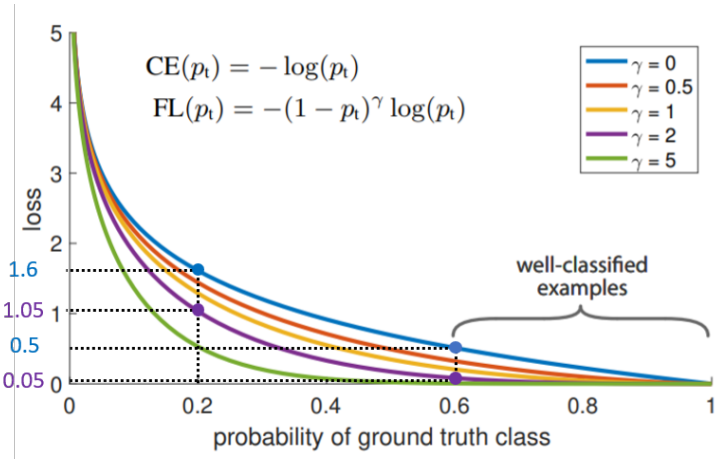


Figure 14. Focal loss

Why the focal loss can address data imbalance?

We believe that the data which rarely appear would get the lower probability of ground class. On the contrary, the data which frequent appear would get the high probability of ground class. We will explain the reason why the focus loss can improve the data imbalance.

First, we focus on the blue line and purple link in the Figure 14. The blue line is cross-entropy loss which is the special case of the focal loss. the purple line is focal loss with $\gamma = 2$. Compared with two lines, if we set 0.2, 0.6 to the probability of ground truth class, the blue line would get 1.6, 0.5 of the loss and the purple line would get 1.05, 0.05 of the loss. We discover the differences of the purple line are more significant by the following equation

$$\frac{1.6}{0.5} = 3.2 < 21 = \frac{1.05}{0.05}$$

This shows the focal loss is able to highlight the data which perform worst, and it doesn't have to work hard on good performance data.

3.6 Optimization

We use AdamW as the optimizer. It is a stochastic optimization method that modifies the typical implementation of weight decay in Adam, by decoupling weight decay from the gradient update. We give the algorithm as the following:

Algorithm AdamW

Input: $f(\theta)$ (objective), γ (lr), $\beta_1, \beta_2, \theta_0, \varepsilon, \lambda$ (weight decay)
Initial: $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ (second moment),
for $t = 1$ to ... **do**
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  $\theta_t \leftarrow \theta_{t-1} - \gamma\lambda\theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
  $\hat{m}_t \leftarrow \dfrac{m_t}{1-\beta_1^t}$
  $\hat{v}_t \leftarrow \dfrac{v_t}{1-\beta_2^t}$
  $\theta_t \leftarrow \theta_{t-1} - \gamma \dfrac{\hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}}$
end for
return θ_t

3.7 Learning Rate Scheduler

We tried step and cosine learning scheduler, their rate of change was concentrated in different parts. The former with a high rate of change in the first 20 epochs and tends to be stable in the remaining periods. The latter is smooth in the front and tail, decreasing in the middle part.

- **Step decay:** Decay in each step by 0.95 factor.
- **Cosine decay:** Use half of the cosine period to change the learning rate.

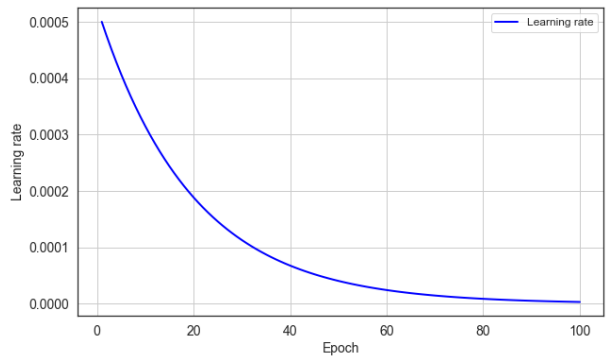


Figure 15. Step decay

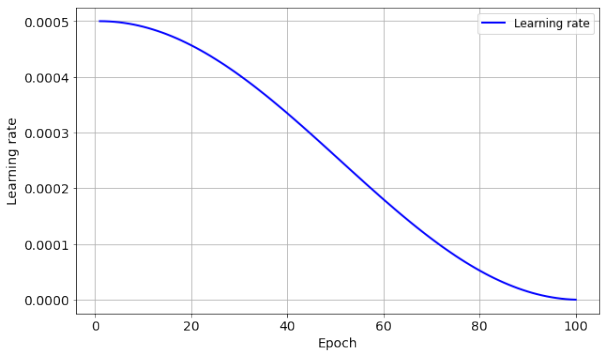


Figure 16. Cosine decay

3.8 Metrics

In this task, the measurement indicator is evaluated on the mean dice coefficient. The Dice coefficient is equivalence to the f1-score. When applied to Boolean data, using the definition of true positive (TP), false positive (FP), and false negative (FN), it can be written as

$$DSC = \frac{2TP}{2TP + FP + FN}$$

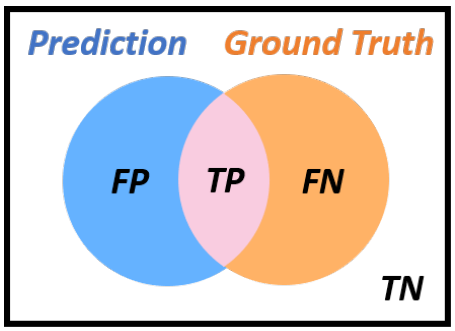


Figure 17. Venn diagram of prediction and ground truth

3.9 Adaptive Single Model Ensemble (ASME)

We proposal a novel ensemble method by single model. It can adaptive the weight of aggregation by solving the eigenvector of dice similarity matrix.

Let $\{T\}_{k=1}^{N-1}$ be the $N-1$ transformations, I be the image, f is segmentation model and $\hat{Y}$ be the prediction after ensemble. Our goal is aggregating $\{I, T_1(I), ..., T_{N-1}(I)\}$. Most of the time, people like to use the uniform blending, but it's susceptible to outliers. Hence we propose a method that can automatically detecting outliers and giving the higher weight of the important prediction.

From now, we called the DSC between two predictions dice similarity. First, we compute the dice similarity of $\{I, T_1(I), ..., T_{N-1}(I)\}$ pairwise. It can compose the adjacency matrix A which is symmetric semi-definite. Next, we calculate the eigenvector of A with respect to the largest eigenvalue $\rho(A)$, denote $v = [v_0, ..., v_{N-1}]^\top$. The we take v to the exponential and do the normalize. This process is similar to the softmax. Finally, the component of vector is between 0, 1 and sum of the all component is 1. We represent the equation as the following:

$$c_i = \frac{a^{v_i}}{\sum_{k=1}^{N-1} a^{v_k}}$$

$$\hat{Y} = c_0 f(I) + \sum_{k=1}^{N-1} c_k (T_k^{-1} \circ f \circ T_k)(I)$$

where $a > 0$ is a hyper-parameter.

We will display the example as the following. First, we compute the dice similarity matrix as Table ?.

Dice similarity	Origin	Fliplr	Flipud	Fliplr+ud
Origin	1	0	0.52	0.79
Fliplr	0	1	0	0
Flipud	0.52	0	1	0.65
Fliplr+ud	0.79	0	0.65	1

Table 1. Dice similarity matrix of 9.tif in test set

Next, we solve the eigenvector of A with respect to the largest eigenvalue which is

$$[0.5810 \quad 0 \quad 0.5341 \quad 0.6141]^\top$$

Then we take the exponential $a = 100$ and do the normalize

$$[0.3291 \quad 0.0227 \quad 0.2651 \quad 0.3831]^\top =: [c_0, c_1, c_2, c_3]^\top$$

Finally, we take $c_0, ..., c_3$ as the coefficient of weighted sum for the ensemble result.

4 Experiments

After trying many encoders, we concluded that using the ResNet family's loss value will oscillate in the later stage. EfficientNet can improve these problems, and the loss value will also be decline steadily. The introduction of the activation function mish although made the loss surface smoother, but the oscillation of the erosion mask smoothing was smaller.

In Figure 18, 19, 20, 21, the first two present best model's (EfficientNet-b1) result training in 100 epochs and the last two apply the mask smoothing to smooth the loss curve.

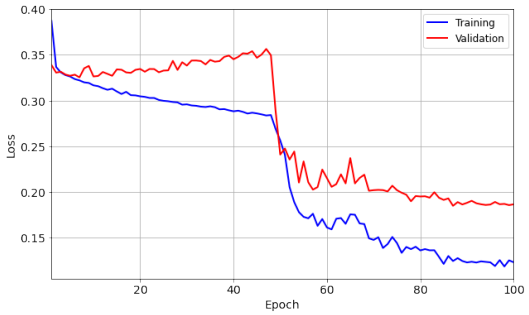


Figure 18. Loss w/o EMS

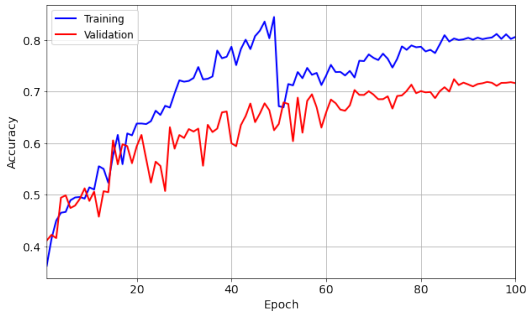


Figure 19. Accuracy w/o EMS

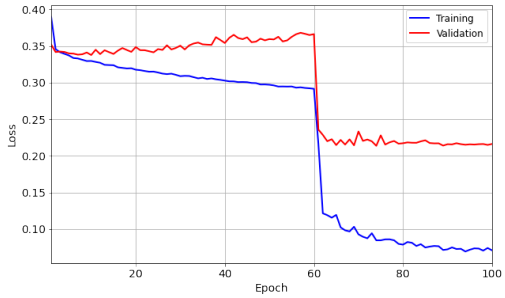


Figure 20. Loss w/ EMS

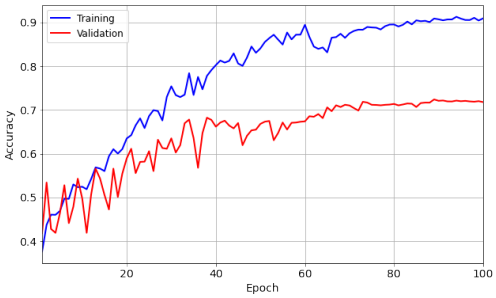


Figure 21. Accuracy w/ EMS

The goal of backbone is to extract the features from the image. In this task, our model uses UNet-based model and made a trial in a variety of backbone like ResNet family and EffcientNet. We use the data augmentation, learning scheduler for training and implement the different backbones. According to the information of Figure 3 in previous, we set the threshold to eliminate the prediction mask which predict less than 2,500 pixels. Our experiments result display as Table 2. In our experiment, the ResNet family's loss are not stable, we found that have dramatic oscillation in the last 50 epochs. In activate function, we had tried the various function like LReLU, mish and swish but not change much.

Model	Backbone	Validation	Test public	Test private
UNet	**ResNet34**	0.70921	0.69759	0.68806
UNet+LReLU	**ResNet50**	0.66040	0.67340	0.67828
UNet	**ResNeXt50**	0.67162	0.66699	0.66704
UNet	**ResNeSt26d**	0.71936	0.67015	0.69009
UNet	**RegNet32**	0.71839	0.67906	0.68959
UNet	**EfficientNet-b0**	0.69933	0.67448	0.68851
UNet+mish	**EfficientNet-b0**	0.69932	0.65637	0.66954
UNet	**EfficientNet-b1**	0.72408	0.70332	**0.70111**
DeepLabv3+	**EfficientNet-b0**	0.71949	0.68078	0.69328

Table 2. Validation and test DSC

In Table 3 , we compared the private score between the single model and applied the ASME method. Implemented this method in pos-processing increase our dice score by 1%∼3%.

Model	Backbone	Origin	ASME
UNet	**ResNet50**	0.67828	0.70857 (0.03029)
UNet	**EfficientNet-b0**	0.68959	0.70233 (+0.01274)
UNet	**EfficientNet-b1**	0.70111	**0.72341** (+0.02300)
DeepLabv3+	**EfficientNet-b0**	0.69328	0.70041 (+0.00713)

Table 3. Improve private DSC by ASME

5 Conclusion

In this project, we implemented a UNet based model with different backbone. For the imbalance of foreground and background, we try to mitigate this problem by combining DiceLoss and Focal Loss. Moreover, we proposed Erosion Mask Smoothing, which makes the loss curve more stable. For the post-processing, we proposed Adaptive Single Model Ensemble, which can improve the dice score around 2 %.

By trying different combinations, we can achieve our highest dice score 0.72341 by using UNet with EfficientNet-b1 as encoder, train on the mask that applied Erosion Mask

Smoothing, applying augmentations such as random flipping, random brightness, post-processing with Adaptive Single Model Ensemble.

Figure 22. Score on Kaggle

Team	Rank	Score (DSC)
AbdulWahab	1	0.73226
OPML (our)	**7**	**0.72341**
benuix	20	0.70753 (baseline)

Table 4. Rank

6 Contribution

Tasks	**contributors (%)**
Literature survey	309652008 (33%), 309652030 (33%), 310653005 (33%)
Approach design	309652008 (33%), 309652030 (33%), 310653005 (33%)
Experiment	309652008 (33%), 309652030 (33%), 310653005 (33%)
Slide making and presentation	309652008 (33%), 309652030 (33%), 310653005 (33%)

References

[1] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L. Yuille, and Yuyin Zhou, TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation *CVPR*, 2021.

[2] Steven Guan, Amir A. Khan, Siddhartha Sikdar, and Parag V. Chitnis, Fully Dense UNet for 2-D Sparse Photoacoustic Tomography Artifact Removal, *IEEE*, 2018.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, *CVPR*, 2015.

[4] Fabian Isensee, Paul F. Jäger, Simon A. A. Kohl, Jens Petersen, Klaus H, and Maier-Hein, Automated Design of Deep Learning Methods for Biomedical Image Segmentation, *CVPR*, 2020.

[5] Jonathan Long, Evan Shelhamer and Trevor Darrell, Fully Convolutional Networks for Semantic Segmentation *CVPR*, 2015.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox U-Net: Convolutional Networks for Biomedical Image Segmentation, *CVPR*, 2015.

[7] Mingxing Tan, and Quoc V. Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, *ICML*, 2019.

[8] Xiao Xiao, Shen Lian, Zhiming Luo, and Shaozi Li, Weighted Res-UNet for High-Quality Retina Vessel Segmentation, *IEEE*, 2018.

[9] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár, Focal Loss for Dense Object Detection, *CVPR*, 2017.

[10] Sergey Zagoruyko and Nikos Komodakis, Wide Residual Networks, *BMVC*, 2016

[11] Kaggle website:
https://www.kaggle.com/c/ultrasound-nerve-segmentation/overview