COMP9417 ASSIGNMENT

T2 2019

GANK

Cong CONG z3414050
Guanqun ZHOU z5174741
Xiao TAN z3413898
Rui ZHANG z5193703

Introduction

A pneumothorax is an abnormal collection of air in the pleural space between the lung and the chest wall [1] . Such disease, also known as collapsed lung, could be a consequence of a blunt chest injury from the outside or damage caused by lung disease from the inside. The considerable difficulty of diagnosis could threaten a potential patient's life and result in death. Although observing a chest x-ray is a common and effective mean to diagnose collapsed lung, the accuracy of diagnosis is overly reliant on the solid professional knowledge and rich personal experience of the radiologist. Fortunately, the widespread use of artificial intelligence in medical imaging is providing strong support for diagnosis, which is also the goal of our project. More specifically, we were applying three different algorithms, two machine learning algorithms and one deep learning algorithm, on a set of x-ray images, so that distinguish images of collapsed lungs from those of healthy lungs, with lesion sites marked.

In the machine learning algorithm, we processed the Local binary patterns of the images instead of the original images to force the algorithm to put more attention on the local textures features of the images. Then we put the patterns into a random forest classifier which combined all the optimal results from a group of decision tree classifiers. Each decision tree classifier only processed a portion of attributes of partial images set, where the images and their attributes were both selected randomly from the whole image set and the attribute set. The structure ensured each decision tree become a specialist of its own unique data set, and the final decisions were determined by the decisions made by the majority of the decision trees in the forest.

In deep learning algorithms, we firstly transformed the images from the original document format .dcm into .png to simplify the following work. Then we separated the images along with their labels into two sets, training set and validating set. To diversify the set, a process of data augmentation was applied to the images before they were put into the deep learning models. The lost function we took in our training and validating stages is a combination of binary cross entropy and dice loss. After training, the network outputs a mask image for each potential pneumothorax positive input image with the image size staying the same (256*256) and a number of -1 for a healthy lung input image. The masks highlighted each possible lesion sites in the images by sitting the pixels into one while the rest pixels into 0.

Related Work

Computer-aided diagnosis is playing an important part in modern medical analysis. This technology is usually being used in diagnosing mammary gland and nodular lung lesions. After image pre-processing and image feature extraction, decision tree, ANN, Bayes network or rule extraction would usually be used in the image processing step. In a study where a support vector machine-based pixel classifier was used to segment neuronal membrane structures from brain tissue samples (Saadia Iftikhar, Afzal Godil 2013), each pixel was represented by a set of distinct binary features so that unwanted regions could be removed easily. Due to the difficulty of accessing a generous enough number of medical images, in other studies where U-Net was used to segment neuronal structures (Olaf Ronneberger 2015), the data was efficiently used with a design of a contracting path and a symmetric expanding path.

Methodology

Machine Learning Based Approaches

LBP

Local Binary Pattern (LBP) describes the spatial structure of local image texture. It firstly divides the image into circular cells of a certain radius. Each pixel in a cell is then compared with its circular neighbours in clockwise/anticlockwise direction. If the neighbour pixels' value is greater than the centre value, write ‘0' otherwise ‘1'. This will produce an 8-digit binary number, which is then combined with all other binary numbers within the same cell. Next, a histogram can be generated to compute the frequency of each binary numbers occurred in this cell. Finally, the histograms from all cells are concatenated to obtain image-level descriptor.

The reason we chose LBP feature is that LBP can be insensitive to rotations and resolutions. From the image data provided, pneumothorax may have different sizes and shapes with various angles. In addition, the lesion areas are barely distinguishable from the rest of the lungs compared to bones or other objects (which we consider them as major noises). Hence global feature descriptors will emphasize less on the region we interest in. However, LBP, as a local descriptor, focuses on regional patterns and less likely to be affected by those major noises.

Procedure - Combine images & features

The LBP features were extracted using the built-in methods from Sk-learn package. We mainly focused on two parameters needed for the feature extraction: radius of circle and number of neighbour points. The radius was chosen from 6 with an increment of 2, and the number of neighbour points is set to the radius multiplied by 8.

The training data was originally set to be the LBP feature map, which is the same size as the original image. The data is then sent to the random forest classifier to train the model. However, the model generated this way produced incorrect images. Hence we combine the LBP feature map with the original image by stacking one on top of another and then feed the reshaped data (of size (1024*1024, 2)) and label (of size (1024*1024, 1)) into random forest classifier.

In addition, it takes a tremendously long time to train with full-size images and features. Therefore, we implemented sub-sampling by randomly choosing a certain amount of features (pixels) across the images and the corresponding indices from the labels (masks). The number of sub-samples is tested from 10000 to 150000, and the training time was significantly reduced while not affecting much of the results.

Result

Image shown below is an example output of the random forest model. It is not too surprising to see that it provided incorrect result.

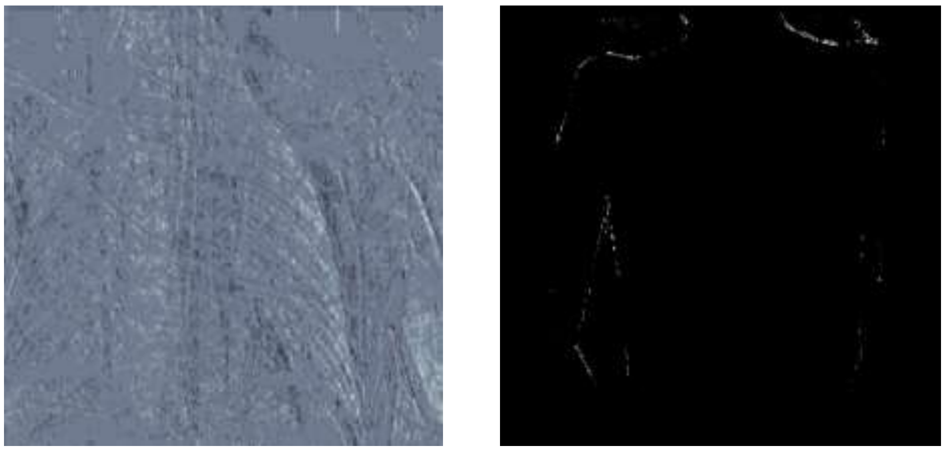


Figure 1: Left: An example of LBP feature map. Right: An example output of the random forest model.

Evaluation

Machine learning methods we implemented using LBP and random forest in chest X-ray images:

1. There is dramatic variance in expressions of the context of training data. The different posture of the human bodies resulted in various occlusions of the lung by the bones. In addition, images also have a different level of details and qualities as they were captured by different devices. Hence it is hard to obtain consistent features directly, and common feature extraction methods cannot handle these situations really well.
2. Furthermore, there are significant noises in the images such as random objects captured by the X-ray machine like clothes buttons or other clothing accessories, which are unlikely to be removed in pre-processing without potentially affecting the parts we interest in.
3. The objects to be segmented are usually very small and even not exist for most of the cases. Therefore, it is hard for machine learning algorithms to train the model effectively because of inadequate features that are related to the object of interests.

In general, machine learning methods are considered outperformed by deep learning methods in this task.

Deep Learning Based Approaches

In many computer vision tasks, machine learning-based approaches are overwhelmed by deep learning-based methods. Based on the problem description, the chosen problem can be treated as an image segmentation task in which the trained deep learning model should be able to detect if the given chest x-ray contains pneumothorax. In this report, two deep learning-based models are described, namely U-net and U-net++ with EfficientNet Encoder. Models are evaluated based on the dice coefficient. Moreover, the score of each model outputs are shown, and comparisons between both models are made.

Pipeline

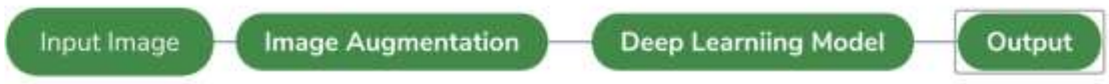


Figure 2: Pipeline

Image Augmentation

Wang & Perez [2](2017) mentioned that the performance of image classifier could be improved by implementing proper image augmentation. Besides, overfitting on the model can also be decreased by augmenting image data. For this project, we have implemented a combination of different image augmentation methods, which includes horizontal flip, random contrast, random brightness, elastic transformation and grid distortion. Below are some visualization results:

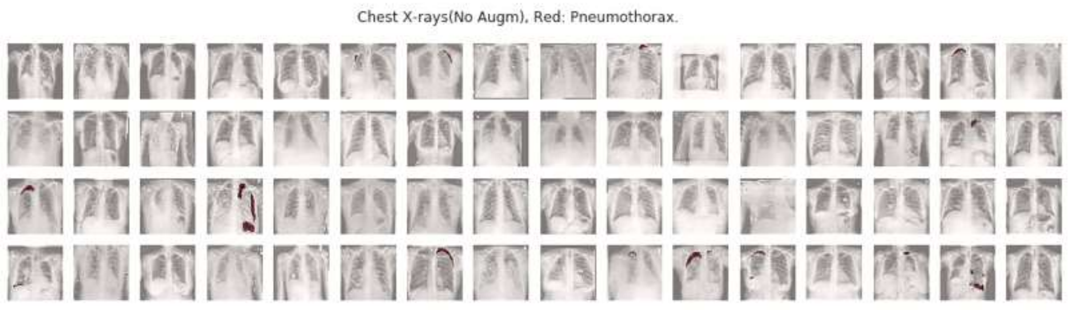


Figure 3: Original Images

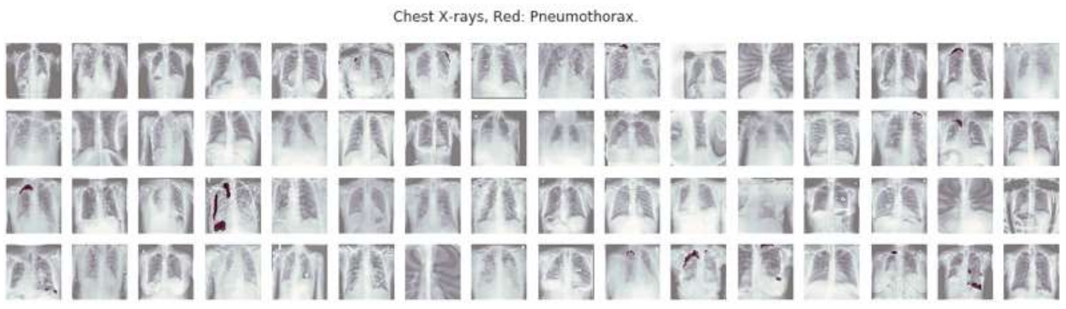


Figure 4: Augmented Images

Model with Image Augmentation	0.786
Model without Image Augmentation	0.165

Table 1: Dice Coefficient (U-net)

Loss Functions

Two forms of loss functions are implemented in this project, namely binary cross entropy and binary cross entropy combined with dice loss. Based on the given image masks, a binary thresholding algorithm is applied to them, the pixel values of mask image after thresholding contains only 0 or 1. Then it is reasonable to treat this problem as a binary classification task. The loss function which is normally used for binary classification problems is the binary classification loss function:

$$E = \sum_{i=1}^{n} y_i \cdot \log\big(p(y_i)\big) + (1 - y_i) \cdot \log(1 - p(y_i))$$

By setting the model loss function as binary cross entropy, the learning curve is shown below:

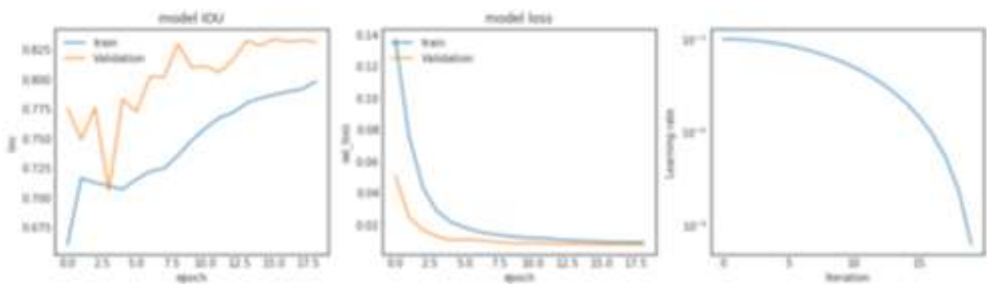


Figure 5: learning curve with setting the model loss function as binary cross entropy

Histogram:

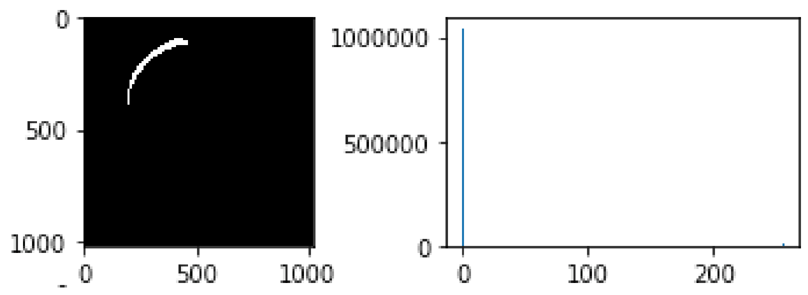


Figure 6: Histogram

From the plotted histogram of given image labels, it can be seen that the pixel classes are extremely imbalanced, in order to design the model to tackle the problem, dice loss function is introduced:

$$E = 1 - \frac{\sum_{i=1}^{n} p_n * r_n}{\sum_{i=1}^{n} p_n + r_n}$$

By combining binary cross entropy and dice loss, the learning curve can be shown to have faster convergence rate.

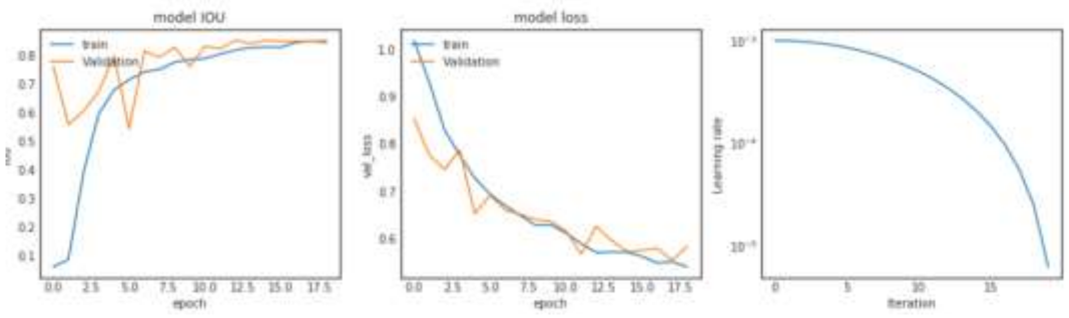


Figure 7: learning curve with combining binary cross entropy and dice loss

With 20 epochs, it can be seen that after setting loss function as a combination of binary cross entropy and dice loss, the dice coefficient of the model converges in a faster rate and ends up having a higher dice coefficient.

Loss Functions	Dice Coefficient (U-net++)
binary cross entropy	0.830
binary cross entropy and dice loss	0.833

Table 2: Dice Coefficient with Loss Functions (U-net++)

Stochastic Weight Averaging (SWA)

In this project we have found that using Stochastic Weight Averaging (SWA) improves the performance of our model.

SWA has some advantageous properties:

- SWA perform excellently on the test set.
- SWA can work very well on test set even it may not work well on train set

Stochastic Weight Average weight update equation:

$$\omega_{swa} \leftarrow \frac{\omega_{swa} \cdot n_{modles} + \omega}{n_{modles} + 1}$$

Where ω_{swa} is the average value of model weights and ω is used to traverse the weight space. By implementing SWA, we can finish prediction much faster. Comparison of model with SWA and without SWA. (Score is from Kaggle)

Number of epoch	Initial learning rate	Train/validation	With SWA	Score
20	1e-3	0.9/0.1	Yes	0.8326
20	1e-3	0.9/0.1	No	0.8382

Table 3: Comparison of model with SWA and without SWA

So we can find that model with SWA would perform better than the one without SWA.

Models:

U-net

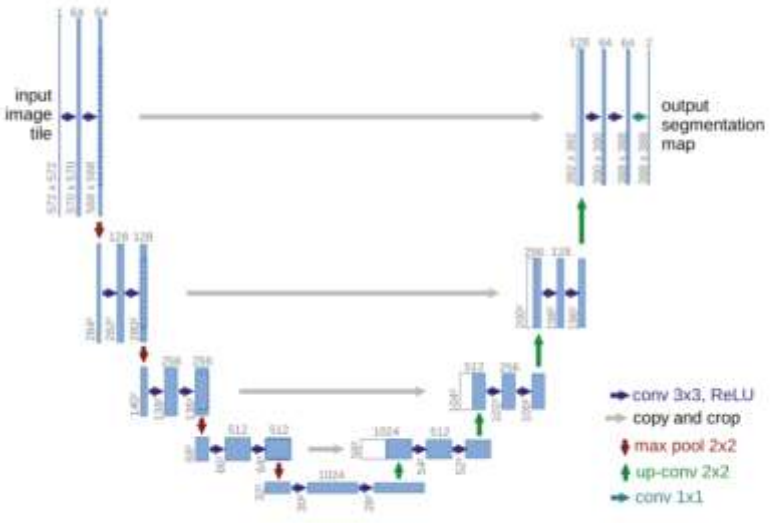


Figure 8: U-net

The U-net structure used for this task is constructed based on the original U-net structure by Ronnerberger, Ficher and Brox [3](2015). This network consists of two main parts, downsampling (lefthand side) and upsampling (righthand side). Downsampling layers consist of two convolution layers, and one maximum pooling layer, input images after down-sampled will have their image channel increased, and image size increased. The functionality of downsampling images is to extract image features. Whereas, upsampling layers consist of one upsampling layer and two convolution layers. In order to localize, the output from downsampling layers is concatenated with the output from lower upsampling layers as the layer input. Upsampling layers help to retain image detail.

The U-net model that is implemented in this project takes input image size of 256 × 256. With combining binary cross entropy and dice loss as loss function and proper image augmentation, the results can be shown:

Number of Epoch	Training Dice Coefficient	Testing Dice Coefficient
20	0.776	0.783
40	0.786	0.778
60	0.789	0.778

Table 4: output of U-net models with different numbers of epoch

From the table, it can be noticed that the model would suffer the effect of overfitting with a higher number of epochs. Thus, for this U-net model, 20 epochs are used to train the model. Besides, the number of images needed for one epoch is calculated using:

$$Step\ Per\ Epoch = \frac{Number\ of\ Training\ Images}{Batch\ Size}$$

10674 images are provided, 90% of them are selected as training images, and the rest are used as validation images. Batch size is chosen as 16. Thus, for every epoch, 600 images with augmentation are used to train our model. With the above setting, some visualization results of the model outputs are shown below:

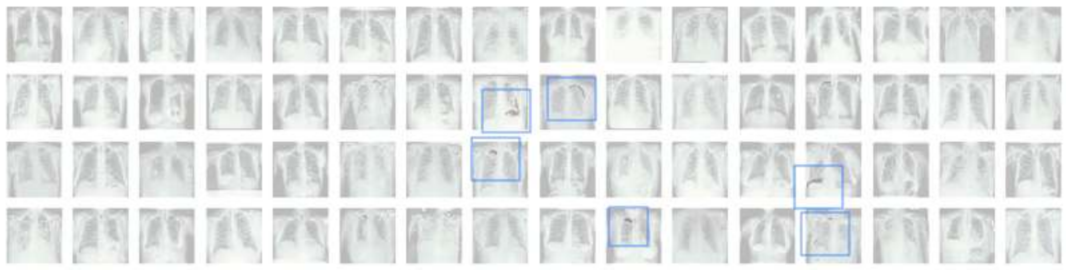
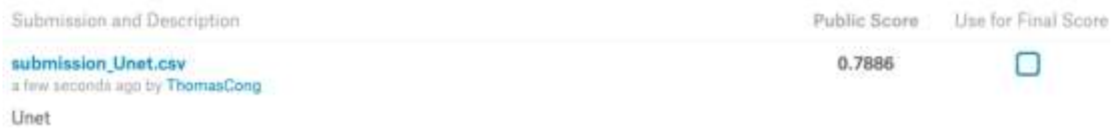


Figure 9: visualization results

Trained U-net model produces an image of the same size as the input image; if the input image contains pneumothorax, these areas will be marked as dark shades (circled areas on the above image). Whereas, if the input image does not contain pneumothorax, the output will be empty. The above image shows some of the result images concatenate with the input image, which shows more meaningful results.

Solving the task using U-net model and submitting the output csv file will give us a final score of 0.7886.



Submission and Description	Public Score	Use for Final Score
submission_Unet.csv a few seconds ago by ThomasCong Unet	0.7886	☐

Figure 10: final score

U-Net++

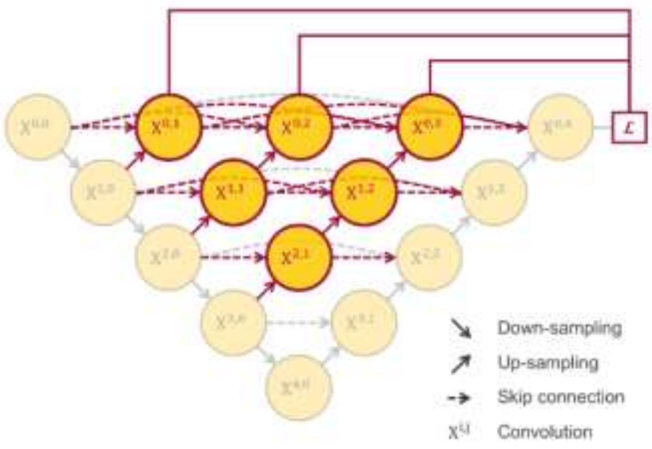


Figure 11: U-Net++

In this project, U-net++ shows a more powerful performance than U-net. The main idea behind U-Net++ is to fill in the gaps between encoders and decoders by stacking outputs from the early stages. The stack formula used in U-net++ is shown below:

$$x^{i,j} = \begin{cases} H(x^{i-1,j}), & j = 0 \\ H\left(\left[[x^{i,k}]_{k=0}^{j-1}, u(x^{i+1,j-1})\right]\right), & j > 0 \end{cases}$$

Where $x^{i,j}$denote the output of node $X^{i,j}$, $H()$ is a convolution operation $u()$ indicates an upsampling layer, and $[\cdot]$ indicated the concatenation operation.

EfficientNet

We combine Efficientnet-B4 and U-net ++ in this project.

Advantages

Unlike conventional approaches, EfficientNet is able to scales each dimension with a fixed set of scaling coefficients and EfficientNet can complete the job with up to 10x better efficiency.

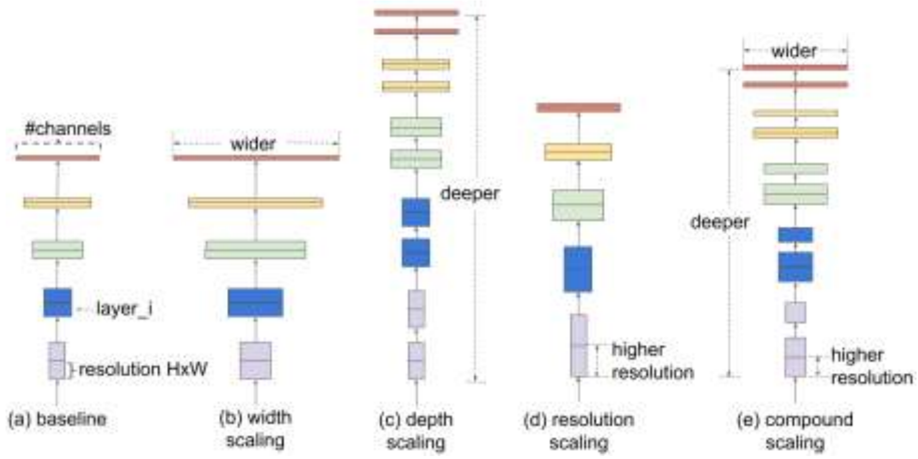


Figure 12: Comparison of model with different scaling

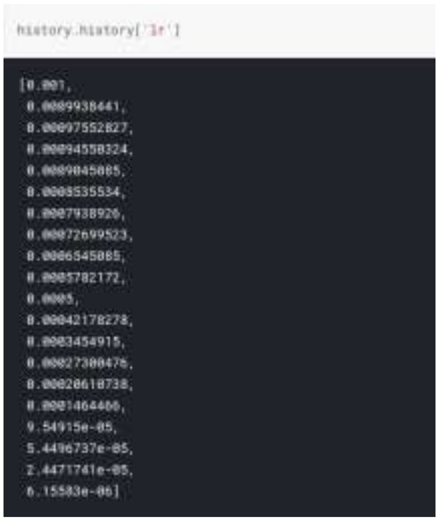
Performance
We can see from the picture above that in general, the EfficientNet models achieve both higher accuracy and better efficiency over existing CNNs.

Parameter
Under U-Net++ with EfficientNet Encoder, we tried a different combination of the number of epochs, initial learning rate and loss function, aiming to find the most optimal combination for this model.

i. Initial learning rate

Number of Epoch	Initial learning rate	Train/validation	With SWA	Score
20	1e-1	0.9/0.1	No	0.7685
20	1e-3	0.9/0.1	No	0.8326

Table 5: Scores with initial learning rate



```
history.history['lr']
```

```
[0.001,
 0.0009938441,
 0.00097552827,
 0.00094558324,
 0.0009045085,
 0.0008535534,
 0.0007938926,
 0.00072699523,
 0.0006545085,
 0.0005782172,
 0.0005,
 0.00042178278,
 0.0003454915,
 0.00027300476,
 0.00020610738,
 0.0001464466,
 9.54915e-05,
 5.4496737e-05,
 2.4471741e-05,
 6.15583e-06]
```

Figure 13: Changes of learning rate

From table and figure above, we can find that the learning rate kept decreasing in training duration, we can say that when initial learning sate is set as 1e-3, the model performs better.

ii. Number of epochs

Number of Epoch	Initial learning rate	Train/validation	With SWA	Score
20	1e-3	0.9/0.1	Yes	0.8382
40	1e-3	0.9/0.1	Yes	0.8375

Table 6: Comparison of number of epochs

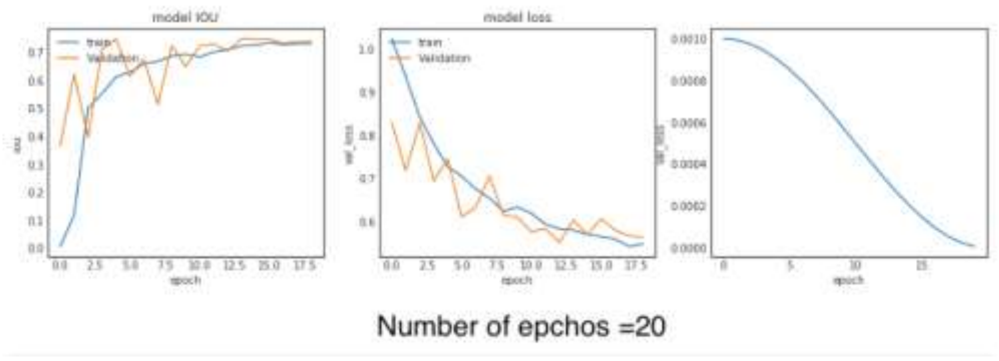


Figure 14: Trend of corresponding parameters when number of epochs =20

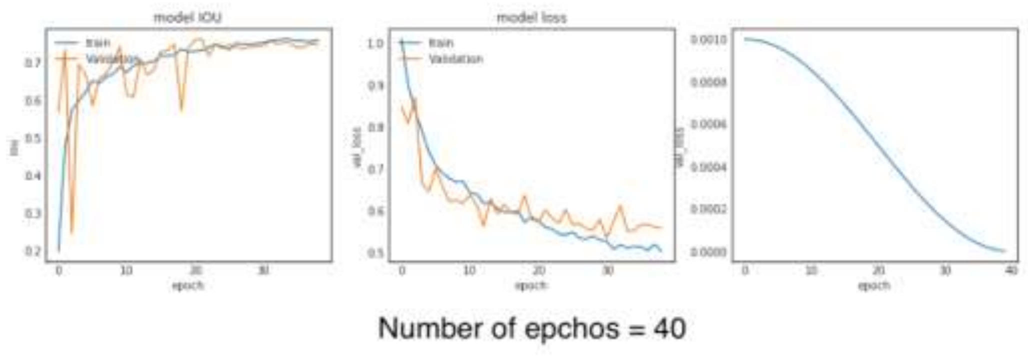


Figure 15: Trend of corresponding parameters when number of epochs =40

From the table and line charts above, we can find that when the number of epochs = 20 the performance of our model is better. Moreover, when the number of epochs is set as 40, the validation decreases lightly after about 38th epoch. And according to learning rate recording of different number of epoch, we found that when the number of epochs = 40, the learning rate kept decreasing in training and stop at around of 6.15583e-6 which is exactly the final learning rate when number of epochs =20, so we can make sure that it is reasonable to set initial learning rate to1e-3.

iii. Loss function

We have two options for loss function:

1. Binary cross entropy
2. Binary cross entropy + dice loss

Number of Epoch	Initial learning rate	Train/validation	With SWA	Score	Loss function
20	1e-3	0.9/0.1	Yes	0.830	Binary cross entropy
20	1e-3	0.9/0.1	Yes	0.838	Binary cross entropy + dice loss

Table 7: comparison of model with different los function

However, it can be found that with binary cross entropy, 20 epochs may not be enough as it is still underfitting. Thus, more epochs are tried:

Number of Epoch	Initial learning rate	Train/validation	With SWA	Score	Loss function
20	1e-3	0.9/0.1	Yes	0.830	Binary cross entropy
40	1e-3	0.9/0.1	Yes	0.834	Binary cross entropy
60	1e-3	0.9/0.1	Yes	0.836	Binary cross entropy

Table 8: comparison of number of epochs under Binary cross entropy

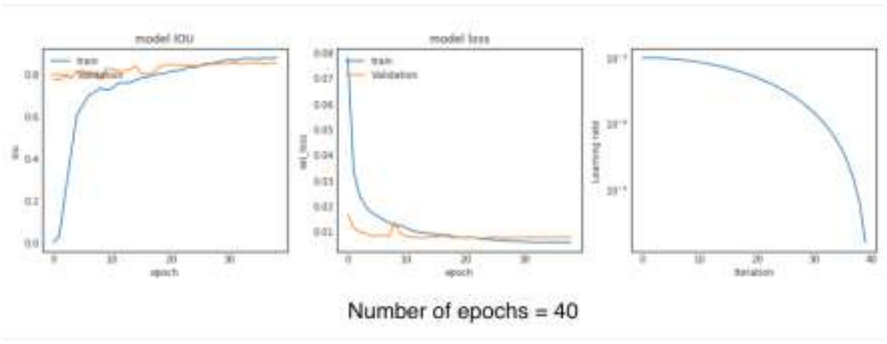


Figure 16: Trend of corresponding parameters when number of epochs =40 (under Binary cross entropy)

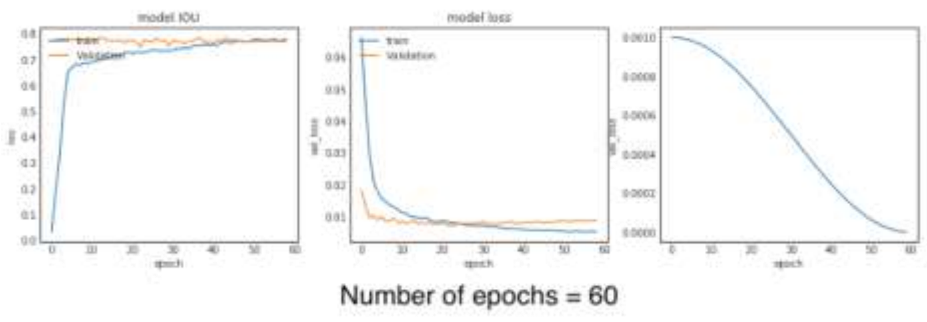
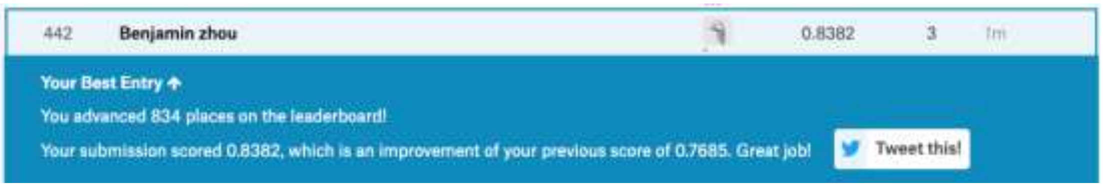


Figure 17: Trend of corresponding parameters when number of epochs =60 (under Binary cross entropy)

From table and figures above, it can be concluded that score would not increase significantly with increasing number of epochs. Thus, the optimal set of parameters for our model is

$$\text{number of epochs} = 20;$$
$$\text{initial learning rate} = 1\text{e} - 3;$$
$$\text{loss function} \ = \ \text{binary\_crosserntropy} + \text{dice\_loss}$$

We train model with this combination and submitted our output submission.csv file in Kaggle. As a result, our model scores 0.8382 and took the 442th place out of 1343 in leaderboard.



442 Benjamin zhou 0.8382 3 1m

Your Best Entry ↑

You advanced 834 places on the leaderboard!

Your submission scored 0.8382, which is an improvement of your previous score of 0.7685. Great job!

Figure 18: model score in Kaggle

Conclusion

For this assignment, an image segmentation task is tackled. Both machine learning-based approaches and deep learning-based approaches are implemented. It can be noted that on the one hand, machine learning-based approaches are simpler in theory and usually are less time-consuming. However, they normally perform poorly on image segmentation task. On the other hand, deep learning-based approaches outperform machine learning-based approaches in the area of computer vision. In this assignment, deep learning models output reasonably better result comparing with machine learning models. However, training deep learning models usually requires higher computational power and more time-consuming. Besides, tuning parameters of deep learning models is also a complicated job.

Future Work

For machine learning methods, it is difficult to extract valuable features for this task directly. Additional efforts can be made, such as image pre-processing, and develop a feature descriptor that is suitable for this scenario.

For deep learning models, one possible improvement can be made by patching image samples For deep learning models; one possible improvement can be made by patching image samples and merging model outputs to produce a final result of the original size. By decreasing the input image size, the training time of model might be decreased. Moreover, more detailed features of images are more likely to be extracted.

References

[1]. Bintcliffe, Oliver; Maskell, Nick (8 May 2014). "Spontaneous pneumothorax". BMJ (Clinical Research Ed.). 348: g2928. doi:10.1136/bmj.g2928. PMID 24812003.

[2]. Wang, J., Pererz, L. 2017, 'The Effectiveness of Data Augmentation in Image Classification using Deep Learning', accessed 8 August 2019.< http://cs231n.stanford.edu/reports/2017/pdfs/300.pdf>

[3]. Ronneberger, O., Fischer, P., Brox, T. 2015, 'U-net: convolutional networks for biomedical image segmentation'.

[4]. Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, Andrew Gordon Wilson 'Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs' arXiv:1802.10026 [stat.ML]

[5]. Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, Jianming Liang 'UNet++: A Nested U-Net Architecture for Medical Image Segmentation'arXiv:1807.10165 [cs.CV]

[6]. Mingxing Tan, Quoc V. Le 'EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks'arXiv:1905.11946 [cs.LG]