Using Deep Learning for Breast Cancer Diagnosis

LYU1706
Submitted by
Li Qi(1155062147)
Li Wei(1155062148)
supervised by Prof. Michael Lyu

July 14, 2024

Abstract

We implemented an efficient, user-friendly and reliable computer-aid diagnosis system to help pathologist do breast cancer diagnosis faster, easier and more accurate. The system can be divided into two parts: one front-end web-page and the back-end program. Users, usually pathologist or community doctors, can upload suspicious patient's samples, which can be of either histopathological images or mammogram and then a patient report will be generated in a few seconds indicating whether the suspicious patient has breast cancer with corresponding probability and whether the cancer is benign or malignant with a probability. The back-end program is based on the state-of-the-art technology, deep learning, specifically, ResNet to do classification and Mask RCNN to do breast mass segmentation. Moreover, we fine-tune the models by combining features of medical images and deep learning models. For classification task, our proposed fine-tuned model outperforms all existing methods and achieves a 5% accuracy improvement from 84% to 89&, for mass segmentation task, our work achieves a 5× speedup than previous quickest work and outperforms it. Also, our front-end website shows a great convenience for users even without coding experience and computer science knowledge.

Acknowledgements

Our deeper gratitude goes from first and foremost to our final year project advisor professor Michael R. Lyu and his PhD student Zeng Jichuan. Without their help and guidance, this research wouldn't be in the right path, especially professor Michael, who listened to our weekly presentation carefully every time and gave us many suggestions such as trying to do kinds of testing by controlling variables. Also, we are also extremely grateful for all authors in reference papers. This is our first time to truly understand the sentence by Newton: *"If I have seen further, it is by standing on the shoulders of giants"*. We truly feel our lack of knowledge when standing on the shoulder of so many great engineering elites, which encourages us to be a great elite like them.

Contents

1 Introduction

1.1 Motivation

Reviewing patient's biological tissue samples by a pathologist is a conventional method for many diseases diagnosis, especially for cancer such as breast cancer. However, reviewing samples are laborious and time-intensive, which may delay decision-making. The reviewing of pathology slides is a very complex task. Sometimes agreement in diagnosis for some forms of breast cancer can be as low as 48% [1]. The difficulty in diseases diagnosis by pathologists is inevitable because the pathologists need to review all slides per patient while each of slide is 10+ gigapixels when digitized at 40X magnification.

On the other hand, current automatic medical diagnosis attempts are not targeted at pathologists with little artificial intelligence background. Pathologist may not understand terms describing an AI or statistics an AI produces. There exists possibility that pathologist cannot interpret a computer generated report very well. With such limitation, cooperating with AI may instead delay decision-making. Therefore, we will try to implement a complete automated breast cancer diagnosis system.

1.2 Background

Since AlphaGo showed the possibility that AI can beat human in real world tasks [2], more and more people in universities or industries are interested in AI for medical usage. The number of papers about AI diagnosis is growing exponentially.

1.2.1 Development of AI Classifier

The classification problem is an important component in the field of deep learning. It is targeted on judging a new sample belongs to which predefined sample category, according to a train set containing certain number of known samples. The classification problem is also called supervised classification, since all samples in train set are labeled, and all categories are predefined [3]. Classifier is one of the pattern recognition applications.

The most widely applied AI classifier is spam email filter, which classify each email into "regular" or "junk". Generally speaking, each instance in the classification problem will be transform into a computer analyzable vector, which is usually called "features". A feature can be an enumeration or a number.

Then the Naïve Bayes classifier was proposed in 1950s. It is a group of simple classifiers derived from the Bayes' Theorem, assuming that all features in the samples are strongly independent. Since its publish, it has been widely researched. Things turned out that it performed well for text classification, with number of occurrence of words as features. It can do the aforementioned email classification task at a

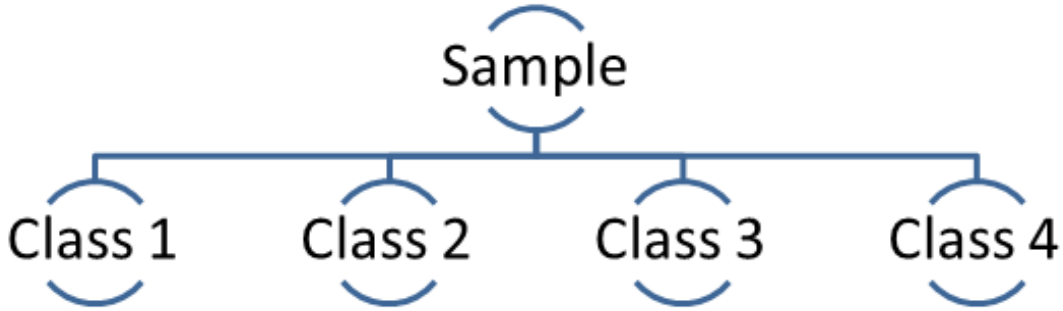


Figure 1: AI classification

relatively low computation amount compared to more recent algorithms while still achieve acceptable accuracy [4]. With appropriate preprocessing, it is still competitive.

$$\text{posterior} = \frac{\text{prior} \times \text{liklihood}}{\text{evidence}}$$

$$\Pr(C_k|F_1, \ldots, F_n) = \frac{1}{Z} \Pr(C_k) \sum_{i=1}^{n} \Pr(F_i|C_k)$$

$$\text{classify}(f_1, \ldots, f_n) = \arg\max \Pr(C = c) \prod_{i=1}^{n} \Pr(F_i = f_i|C = c)$$

The Naïve Bayes classifier can have different assumptions for the underlying distribution of features. For continuous variables, we can assume they are under the classic Gaussian distribution. For text data, the standard assumption is multinomial distribution, where the number of occurrence of a word is taken into account. A simplified version is Bernoulli distribution, which only consider whether a word appears or not.

The Naïve Bayes classifier is much more extensible than other algorithms. Number of parameters it needs to learn is linear to number of features, therefore the training time complexity is also linear. Moreover, the training process has a well close-formed expression. For email classification problem, the number of parameters is merely the number of unique words in all emails. This avoid the expensive linear approximation many other classifiers use.

Later Support Vector Machine (SVM) was introduced by Vladimir Naumovich Vapnik and Alexey Yakovlevich Chervonenkis [5]. Given a train set, each sample is represented by a point the hyperspace. For SVM, samples are treated as p-dimensional vectors; SVM assumes that we can separate these points with a (p-1)-dimensional hyper plain. There may be may such hyper plain, and SVM will separate different categories with a hyper plain with as large margin as possible. Thus, we will get the

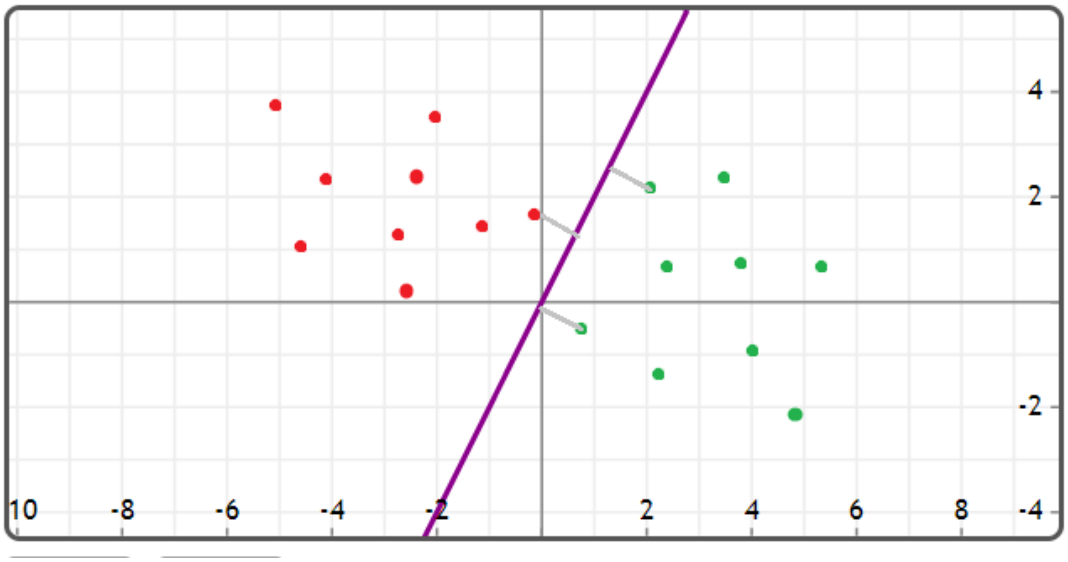


Figure 2: Support Vector Machine

hyper plain whose distance to nearest data points of two categories is maximized. This is also why it was named "Support Vector" machine.

SVM is usually a linear classifier. However, with some tricks called "Kernel trick", SVM can also do nonlinear classification. The main idea is, by mapping the original sample space to a higher dimensional space, the original non-linear separable set may become separable.

1.2.2 Development of AI Object Detection

As the demand is rising for autonomous vehicles, intelligent monitoring and various other applications, object detection is becoming a hot spot in the AI field. These system need to not only recognize and classify the whole images, but also locating each objects in images. We want to find faces or cars from a complex real-world image. This makes object detection a hard task compared to the traditional classification.

The very first object detection algorithm was based on extracting features from the images [6]. The core idea of this algorithm is simple. By finding correspondence between the given reference and the input target object, we can detect a specific object. It estimate the scale transformation and the rotation applied, and find the matching features. It can also handle some level of occlusion.

Feature-based object detection requires the object has a unique texture pattern, which reduces noise in feature matches. The performance is unlikely to be good for large objects with a uniform color, such as a car. Plus, it is designed to find a specific instance rather than a class of instances, i.e. to find the one car rather than any cars. Unfortunately, these limitations restrict its usage.

Figure 3: Object Detection by Point Feature Matching

Viola-Jones object detection was later proposed in 2001 by Paul Viola and Michael Jones [7]. It is the first algorithm that can process the image in real-time and still gives satisfiable detection rate. While it is capable for multiple class object detection, it was originally designed and mainly used for face detection. It uses features extracted from sums of rectangular areas in the image, which is called harr features. As for the face detection task, this algorithm find the similar features in all faces, and then represents the image with an integral image. The summing process can be done in constant time, so this algorithm is faster than its competitors. It then uses a cascade architecture to assemble a strong classifier from many weak classifiers. This architecture tolerates poor weak classifiers very well, which also reduces its running time.

Viola-Jones algorithm was the first one that enables object detection in the consumer technologies. Face detection functions on nowadays cameras are mainly based on this algorithm. It however is still a feature based solution, so still has the same drawback as other similar solutions. Its performance drops when detecting multiple classes.

Then SVM classification was also introduced to this area by Navneet Dalal and Bill Triggs to further improve performance [8]. With histograms of oriented gradients (HOG) features, we can achieve a higher accuracy. This method extract HOG descriptors from positive and negative samples, then train a linear SVM on these descriptors. It then apply hard-negative mining on inputs, i.e. slide a window through the image and compute the SVM at each window. Since SVM can give an accuracy boost in classification task, it is not surprising that it will also give one in object detection task.

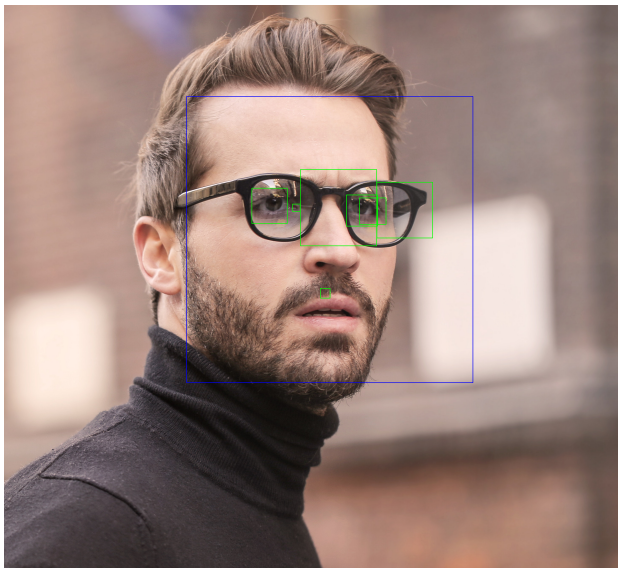


Figure 4: Viola-Jones Object Detection

1.2.3 Development of Deep Learning

Deep learning is a subset of machine learning. It is a family of feature learning algorithms in the area of machine learning. Observation values can be represented in various ways, such as a vector containing RGB values of each pixel, or more abstractly a series of edges and areas [9]. It attempts to do highly abstract data computation with multiple process layers which may contain a complicated structure or non-linear mapping. In general, it is a boarder machine learning method, as it is not specific to any task. There are multiple deep learning frameworks already widely used, such as deep neural network, convolutional neural network and recursive neural network. Deep learning has been widely used in applications, including computer vision, natural language processing and bioinformatics, and achieves supreme results.

In 1989, Yann LeCun [10] proposed the deep learning mode. Through it could run, the computation cost was so large that the training took about three days. The very first deep learning attempt therefore failed going into real application. The trend of AI then shifted into Support Vector Machine. However, in 1992, Schmidhuber [11] proposed an effective algorithm to train neural networks. This algorithm treats each layer in the network as an unsupervised, and then tune its parameters with supervised back propagation algorithm. In experiment, it was shown that this training method can indeed improve the train speed of supervised learning.

The advantage of deep learning is that it uses effective unsupervised or Semi-supervised feature learning and layered feature extraction instead of man-powered feature extraction. The aim of feature learning is to seek for better representation of data and to create better model to learn these representations from large-scale

unlabeled dataset. The representation is like development of real neural network, and is based on the understanding of how information is processed and transmitted in neural-like systems [12].

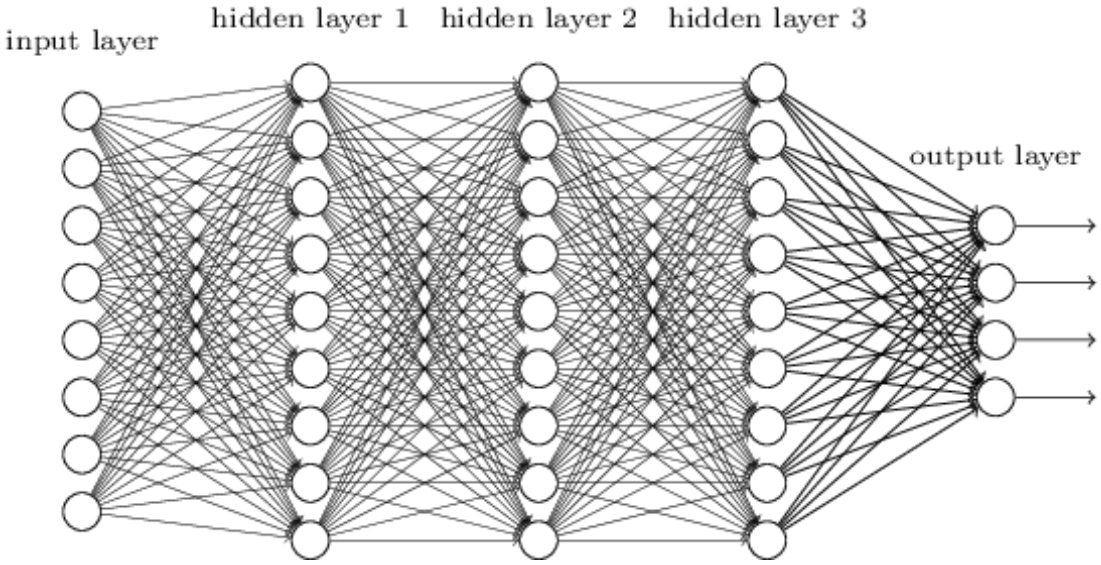


Figure 5: A deep neural network

The basis of deep learning is the distributed representation in machine learning. "Distributed" means the assumption that the observation is resulted from interaction between different factors. Furthermore, deep learning assumes that such interaction can be spliced into multiple layers, which means the multiple abstraction of the observed value. Different number of layers and different size of layers can be used to represent different degree of abstraction. This idea of layered abstraction indicates that higher-level concepts are learned from lower-level concepts. This structure is usually constructed with greedy algorithm, which helps the machine to learn more significant features. Many deep learning methods are unsupervised algorithms, which enables deep learning to be applied to unlabeled data. This is a great advantage over other algorithms. The amount of available unlabeled data is much larger than labeled ones; unlabeled data is also cheaper to acquire.

What even more encouraged researchers is General-Purpose computing on Graphics Processing Units (GPGPU). The development of more powerful hardware and increase in available data made deeper neural networks realizable. In 2009, Nvidia stepped into the area of deep learning and started promoting its GPU. It was confirmed that the involvement of GPU can increase the training speed by more than 100 times. Since GPU is quite suitable for matrix/vector computation in deep learning algorithm, a GPU can reduce the time required from weeks to days.

Since the emerge of deep learning, it has become one part of the most advanced systems in various areas, especially in computer vision and speed recognition. On

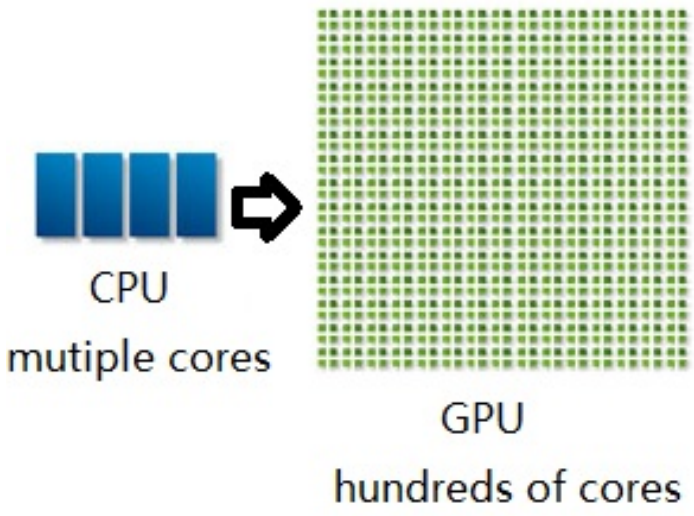


Figure 6: From CPU to GPU

standard verification datasets such as Cifar 10, experiments showed that deep learning can improve recognition accuracy. A deep learning method, convolution neural network, processed about 10% to 20% checks in US. Due to the development of deep learning, the year 2010 witnessed a bunch of the very first industrial speech recognition products.

1.2.4 Development of Deep Learning for Medical Images

In the area of medical image proceeding, deep learning is becoming more and more attractive. The recent development in deep learning has achieved a great leap. Generally speaking, research on deep learning for medical images is mainly focused on four aspects: structures detection, segmentation, labeling and captioning, and computer aided detection or diagnosis.

Structure detection is one of the most important steps in medical image process. Pathologists generally accomplish this task by recognizing some anatomical feature in the image. Though the success of deep learning in this area mainly depends on how many anatomical feature the algorithm can extract. The recent trend indicates deep learning is mature enough to solve real world problems. Shin et al. [13] proved deep learning in computer vision applicable for medical images. On top of this, they detected multiple organs in a series of MRI images. Meanwhile, Roth et al. [14] presented a method to detect organ at certain body part. They trained their deep neural network with 4298 images and achieved an error rate of 5.9

Segmentation is the process of dividing a digital image into many sub- images [15]. A segment is a set of pixels, and therefore is also called hyper pixel. The aim of image segmentation is to simplify or alter the representation of the image so that

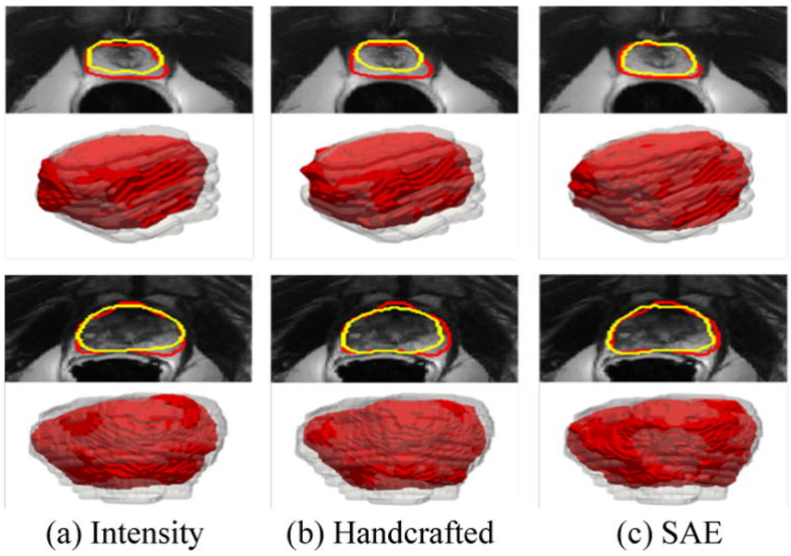


Figure 7: Typical Segmentation

it becomes more easy to understand or analyze. Segmentation is usually used to locate objects or edges in the image. More precisely, segmentation is a process to label each pixel in the image, which makes pixels with the same label have a similar visual feature, such as color, brightness or texture. Moeskops et al. [16] designed a multiple-scale CNN for accurate tissue segmentation, using multiple patch sizes and multiple convolution kernel sizes to gain multiple scale information of each pixel, and achieved accuracy from 82% to 91%. Zhang et al. [17] tested four CNN on the task of brain tissue segmentation. Their experiment uses three convolution layers and a fully connected layer, and proved CNN significantly better than traditional methods.

Labeling and captioning is the most widely used way to describe contents in an image. It is the classic classification problem in the area of medical images. Continuous effort is being put in to ensure disease-specific auto labeling. Inspired by neural networks for regular images, some research [18] [19] introduced RNN together with latest advance in computer vision to caption chest radiographs in certain contexts. The authors used image captions in public available dataset to train the CNN. To avoid large error, many normalization techniques were applied. Then the network was used to describe the situation of detected disease.

Computer aided detection or diagnosis involves finding or locating abnormalities and suspicious area, and then alert clinicians. The main aim of computer aided detection is to increase the detection rate of infected area and to decrease false negative due to observer's mistake. Though it is considered a mature area in medical images, deep learning further improved performance in many applications and enabled some design that was impossible in the past. Traditionally, computer detection requires

a preprocessed candidate region and manpower to extract features such as shape or statistics in the region; inly after then the features can be feed into the classifier. However, the advantage of feature learning is the core of the new developments. Deep learning can learn the hierarchical features from the dataset independently instead of depending handcrafted features specially targeted for certain area of knowledge. It soon proved to be the most advanced technology. Ciompi et al. [20] trained CNN with predefined OverFeat as feature extractor, and showed that CNN is feasible to provide useful feature description in lung images. Gao et al. [21] trained the model from the very beginning. They solved the overfitting problem by randomly cropping or jittering the original image, and then feed the sub images into the model. Finally, the model was able to classify patches into normal, fibrosis and other four abnormal classes.

Due to the prosperity in research, more and more commercial attempts is being conducted recently. Startups entering the medical AI area is increasing. From 2012 to 2016, investments in medical AI increases from 20 cases per year to 70 cases per year. More than 100 large companies are trying to apply deep learning in order to decrease time to provide aids to patient and to automatically diagnosis disease with medical images. IBM Watson Group is supporting a research to screen cancer patients with an affordable procedure. They are trying to make deep learning suitable for production. Other startups include SkinVision, Flatiron Health and Entopsis [22].

1.3 Objective

Deep learning has a natural advantage in features learning, which means that it has a potential to be applied to this problem mentioned above. Therefore, we will try to implement a complete automated breast cancer diagnosis system. In this system, we will train a deep learning program which can give advice to pathologists, even if s/he do not know anything about AI.

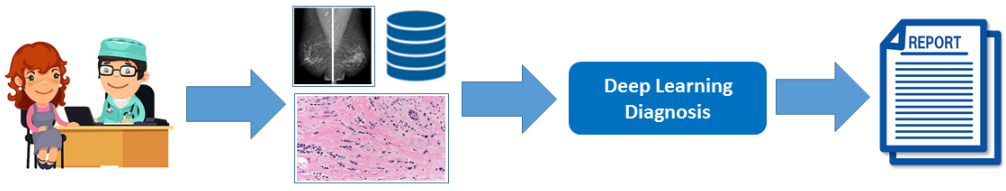


Figure 8: Our Diagnosis System

This project involves image classification, object detection and image caption together. It is designed to be able to perform mammogram analysis or pathology

analysis, and detect possible tumor location. A deliverable diagnosis and tumor positioning report will be generated at the end which can help them make a more accurate decision on diagnosis. The whole system will have the following functionalities:

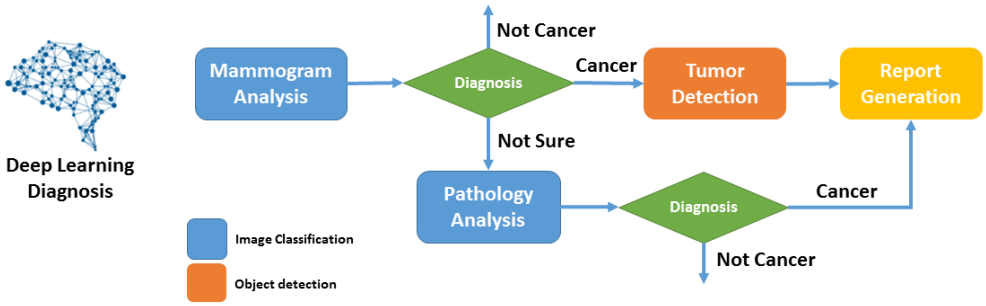


Figure 9: Workflow of Our Diagnosis System

1. Perform mammogram analysis first

   To determine if a tumor is benign or malignant, we will first require the patient's magnification mammogram image. The deep learning program will try to make a preliminary classification: cancer, not cancer, or not sure. More detailed diagnosis should follow.

2. Detect possible tumor location if classified positive If the program categorizes image as positive, it will further detect the exact existence of tumor. It will point out the most suspicious regions in the image for pathologists' reference.

3. Make a more confident judgment with pathology analysis If the program cannot achieve a pre-defined certainty threshold, it will suggest a pathology analysis. As the pathology analysis can give more information, very likely the program will approach the correct inference.

4. Generate human-readable report At the last, the program will describe its output in an understandable way. The report will indicate all its findings.

In term one, our primary objective was to build an accurate breast cancer histopathological image classification model, the first computer diagnosis procedure in the workflow. This is the entry point, and will be the most frequently used module in the system.

In term two, our primary objective is to finish up the rest parts of the project, i.e. the tumor detector and the report generator. The program will try to locate tumor cells and tell the pathologist its suggestions. Since it will give more accurate information about the sample, the accuracy may be lower in turn. For users' reference, we plan to explicitly show confidence level on the output reports.

2 Literature Review

2.1 Naïve Bayes for Breast Cancer Diagnosis

Many attempts have been made to predict medical image classes since the emerging and development of Artificial Intelligence. The work from [23] uses a traditional Naïve Bayes classifier for automated breast cancer diagnosis. This work is quite valuable because it turned out that Artificial Intelligence is feasible for medical diagnosis since simple classifier of AI, specially classifier in machine learning , can also do a good job.

2.1.1 Thesis

In their thesis, the first step was preprocessing, just same with other image classification tasks. Their original data was not of high quality and resolution, moreover, there were many noisy pixels in the image. They used Gaussian filter to blur the image and reduce the noise. Then they stretched the histogram to improve contrast.

The second step is segmentation of nuclei, since classification of tumor requires identifying nuclei in each cell. they implemented four clustering algorithm: competitive neural network, fuzzy C-means, K-means and Gaussian mixture model. Then 42 features were extracted from each segment. The features were selected by experienced human pathologists.

Then the features were feed into classifiers. They trained a Naïve Bayes classifier which was using estimated kernel densities. 500 real medical images from 50 patients formed the train dataset.

2.1.2 Results

The performance was measured with n-fold cross validation method.Their accuracy rate was about 96% to 100%, which can be found in table 1, which indicated AI in breast cancer diagnosis was quite promising for production. It showed that their preprocessing procedure and data collecting procedure could assure accurate and objective dataset.

	KM	FCM	GMM	CNN
Patients Accuracy	100.00%	96.00%	100.00%	98.00%
Image Accuracy	90.22%	85.78%	88.00%	89.56%

Table 1: Performance of Different Classifiers

2.2 SVM for Remote Breast Cancer Diagnosis

The work from George et al.[24] proposed a more advanced system than previous work for breast cancer diagnosis. They proposed a fully automatic nuclei detection and segmentation method. Then they developed the AI tumor classification system. They proposed 12 features for research on the most effective model. At last, they experimentally pushed their computer aided detection and diagnosis system to production, connecting it to a remote medical platform. This web based service was expected to provide an intelligent and convenient diagnosis for breast cancer patients.

2.2.1 Method

Their first step was preprocessing. Since preprocessing is the most critical and calculation-consuming factor in image processing, they shrank the image size from 2560x1920 to 640x480. Then contrast enhancement and edge sharpening was used to manipulate the image. They used contrast limited adaptive histogram equalization to enhance the quality of the image. CLAHE worked within each tile of the image instead of the whole image, so that contrast was enhanced in each tile. The next step is cell nuclei detection. They implemented a detector combining circle detection and local maximum finder. In the images, there may exist some blood cells which were unwanted noisy markers.

They used Fuzzy C-Means Clustering method to remove such cells. The noise free image was then separated into individual objects with marker-controlled watershed transform. They used some meaningful features to classify the image. They proposed two textural features and ten shape features that could yield a good discrimination ability. The features include boundary, smoothness, etc. Then the features were feed into SVM. The workflow of SVM system is shown in figure 10.

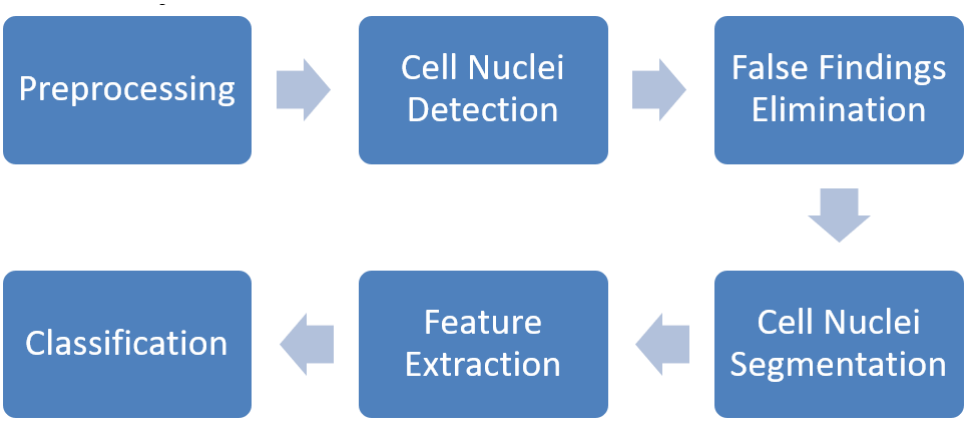


Figure 10: Workflow of SVM System

2.2.2 Results

The train set and test set was generated with ten-fold cross validation method. A total of 3260 images were used in the experiment. The experiment result showed that their method was still effective for bloody images or noisy images. However, due to the extreme lack of data, their accuracy was capped at 82.6%. Some data set still did not the training goal after 200 epochs. This paper illustrated some effective ways to preprocess images, and proved that the performance converge is greatly correlated to the size of train set.

2.3 Classification of Skin Cancer with DNN

With the development of deep learning and GPU calculation capacity, some most recent research on medical deep learning discussed deep neural networks for classification of medical images, which is absolutely the most direct application of DNN in medical field. The work [25] is one of the most classic and promising method to do image classification for cancer diagnosis, specially, skin cancer.

2.3.1 Model Description

Instead of highly standardized images generated from specialized instrument such as microscope, their classifier was mainly focused on classifying images from general purpose photography instruments like smartphone. The variety of zooming, angle and brightness brought new challenge to the task. They used data driven method to overcome this difficulty – they increased the size of dataset to 1.41 million which was impossible for standardized images. The number of images made classification more robust to the variety in images. Compared to previous work that required many preprocess, segmentation and feature extraction, they required no handmade functions in the classification. Their model directly read the original image and original pixels and perform an end to end training.

Their classification includes 2032 single diseases arranged in a tree structure. Three root nodes represented benign, malignant and non-tumor lesions. It was given in the bottom to top structure and therefore was very suitable for machine classifiers.

They utilized the GoogleNet Inception v3 CNN architecture [26], which was previously trained for 2014 ImageNet challenge, then transferred to the skin cancer dataset with transfer learning technology. This is a deep CNN architecture which achieved 93% accuracy in the challenge. They deleted the final classification layer, and retrained the network with the skin cancer dataset, the detail of the structure they used can be found in figure 11 and fine tune parameters of each layer. During the train process, they shrank size of each image to 299x299 pixels so that it could fit with the input sized of the original Inception v3 network structure, and used ImageNet to pre-train the image feature learning ability of the network. This

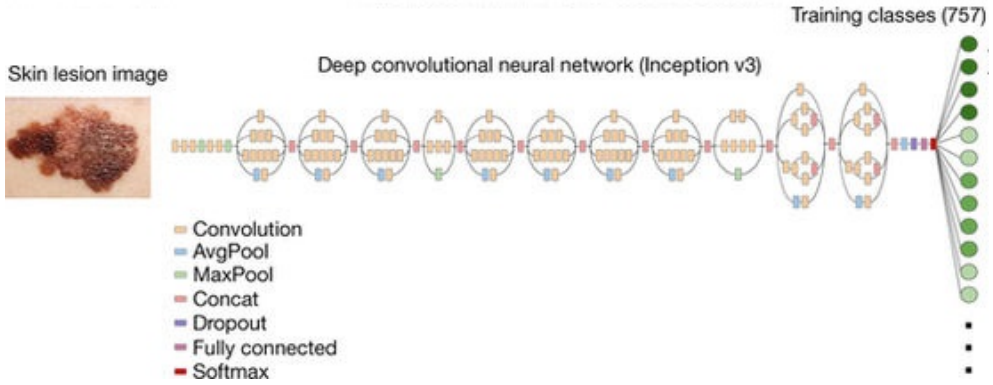


Figure 11: Structure of Inception v3

process was called transferred learning, which could result in the best result with given number of data.

2.3.2 Results

They trained the Convolution with back propagation algorithm. All layers in the network was assign the same global learning rate. They used Tensorflow, a deep learning framework by Google to train, validate and test their network. They tested their network with two methods, using nine-fold cross validation. First, they used three top-level nodes for classification, which classified each image into benign, malignant or non-tumor. In this task, CNN achieved 72.1±0.9% accuracy for each patient. Two human dermatologists achieved 65.56% and 66.0% on a subset of the test set. Second, they classified images into different medical care requirements. CNN achieved 55.4±1.7% while two dermatologists achieved 55.0% and 53.3%. This demonstrated the effectiveness of deep learning for cancer diagnosis. This method is mainly bounded by data; if given enough data, it can be suitable for many other image problems.

2.4 Deep Multi-instance Networks with Sparse Label Assignment for Whole Mammogram Classification

This work[27] focused on the classification of mammogram, which has been demonstrated to be an effective way for early detection and diagnosis. Traditional mammogram classification requires extra annotations such as bounding box for detection or mask ground truth for segmentation considering the high-resolution feature of mammogram, however, these methods require training data to be annotated with segmentation ground truths and bounding boxes which are hand-crafted features and require expert domain knowledge and costly effort to obtain.

Considering all the points above, this work focuses on perform classification based

on a raw whole mammogram, where each patch of a mammogram instance can be treated as a pixel of the whole mammogram.

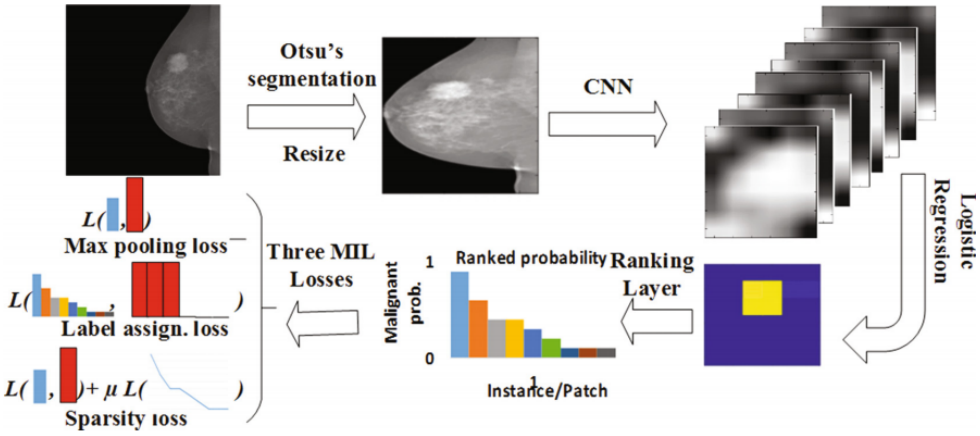


Figure 12: The framework of whole mammogram classification.

2.4.1 Multiple Instance Learning Framework

Through the study of previous work, this work investigates three different schemes, i.e., max pooling, label assignment, and sparsity, to perform deep MIL for the whole mammogram classification task.

The framework for proposed deep MIL of mammogram classification is shown in figure 12. Firstly, the Otsu's segmentation is used to remove the background and then the mammogram is resized to 227×227, secondly, the resized training data is fed into the DNN model, typically, the AlexNet model [28] of MIL and the output of the fine-tuned AlexNet will be the probability for the input mammogram as malignant pixel-wise. Thirdly, the malignant probability of each instance in one patch will be ranked. Finally, three kinds of losses will be calculated for the three different schemes they investigate.

Max Pooling-Based Multi-instance Learning Unlike the general DNN classification task, whose output is simply the prediction class of input image. The Multi-instance learning is this work regards input image as a patch and each pixel of the image is corresponding instance. Therefore, the generated result of deep MIL model will be $\boldsymbol{r} = (r_{(0,0)}, ..., r_{(row,col)})$, where row, col is the value of row number and column number of input image and $r_{i,j}$ represents corresponding probability to be malignant of pixel at position (i, j).

According to the general assumption of multi-instance learning, if there exists an instance that is positive in one patch, then the patch is positive. Therefore the

final cost function considering max-pooling should be the cross-entropy-based cost function between the result of $\boldsymbol{r}$'s max-pooling and the true label, and the cost function can be therefore defined as:

$$L_{max-pooling} = -\frac{1}{N}\sum_{n=1}^{N} log(p(y_n|\boldsymbol{I_n}, \boldsymbol{\theta})) + \frac{\lambda}{2}||\boldsymbol{\theta}||^2 \tag{1}$$

where the first sum part is for calculating corresponding cross entropy, N is the total number of mammograms, and Λ is the regularizer that controls the model complexity and avoid overfitting.

However, this kind of cost function considers only the influence of the max malignant pixel, which ignores other pixels' information. Therefore, one scheme is obviously not enough to gain a reliable result.

Label Assignment-Based Multi-instance Learning The biggest difference between this kind of MIL and above one is that the label assignment-based method assume that the first k largest malignant probabilities of model outputs should be assigned with the same class label as that of whole mammogram, and the rest pixels results should be labeled as negative in the label assignment-based MIL. Considering two assumptions and transfer them into cross entropy symbols, the cost function can be defined as:

$$L_{label-assign} = -\frac{1}{mN}(\sum_{n=1}^{N}(\sum_{j=1}^{k} log(p(y_n|\boldsymbol{I_{n,j}}, \boldsymbol{\theta})) + \sum_{j=k+1}^{m} log(p(y=0|\boldsymbol{I_{n,j}}, \boldsymbol{\theta})))) + \frac{\lambda}{2}||\boldsymbol{\theta}||^2 \tag{2}$$

One advantage of the label assignment-based MIL is that it makes use of all the information of pixels to train the model.the optimization problem of label assignmentbased MIL is exactly a k-sparse problem for the positive data points, where we expect the k largest corresponding label being 1 and others being 0. The disadvantage of label assignment-based MIL is that the estimation or tuning of the hyperparameter k is hard and time-consuming. Thus, a relaxed assumption for the MIL or an adaptive way to estimate the hyper-parameter k is preferred.

Sparse Multi-instance Learning The sparsity feature is found through the analysis of the raw data: the mass typically comprises about 2mammogram on average, which means that the mass region should be quite sparse in the whole mammogram. Therefore, the author applies this feature and assumes that almost all pixels label should be sparse, which indicates benign. Considering this feature and all study above, the loss function of sparse MIL can be defined as:

$$L_{sparse} = \frac{1}{N}\sum_{n=1}^{N}(-log(p(y_n|\boldsymbol{I_n}, \boldsymbol{\theta})) + \mu||\boldsymbol{r'_n}||_1 + \frac{\lambda}{2}||\boldsymbol{\theta}||^2 \tag{3}$$

We can see that the only difference between equation 3 and 1 is the L_1 constrains, which makes probabilities sparse.

The advantage of sparse MIL is that, firstly, this kind of learning convert the label assignment-based MIL to simple L_1 constrain, which is easier to do calculation than cross entropy, the second advantage is that this method can be regarded as a trade-off between max pooling-based MILL (slack assumption) and label assignment-based MIL (hard assumption).

2.4.2 Results

The proposed model is validated on the mammographic mass classification dataset, INbreast dataset [28] instead of another frequently used mammograms datasets, DDSM dataset [29]. And the validation is performed using 5-fold cross validation method. Also, common data augmentation skills such as rotation, flip and adding noise are used.

The result is compared with other related works, such as pretrained CNN, which uses three CNN models to do ROI detection, ROI segmentation and ROI classification respectively. From the results paper claimed, The max pooling-based deep MIL obtains better performance than the pretrained CNN. This shows the superiority of end-to-end trained deep MIL for whole mammogram classification, which is also available for our research. According to the accuracy metric, the sparse deep MIL is better than the label assignment-based MIL, which is better than the max pooling-based MIL.

The visualization result for this work is shown in figure 13, we can see that the model can not only learns the classification of the whole mammogram, but also the rough region segmentation by the different probabilities of different pixels. This, therefore shows the potential application which may be used to do rough annotation automatically.

2.5 Multi-scale mass segmentation for mammograms via cascaded random forests

This work [30] is highly related to our project, they propose a novel approach for detecting and segmenting breast masses in mammography based on multi-scale morphological filtering and a self-adaptive cascade of random forests, which are the knowledge of machine learning. In this section, we will briefly cover the methods they use and the results they get using the same dataset in our project.

2.5.1 Methods

A system for breast masses detection and segmentation can be explained in the Figure. 14.

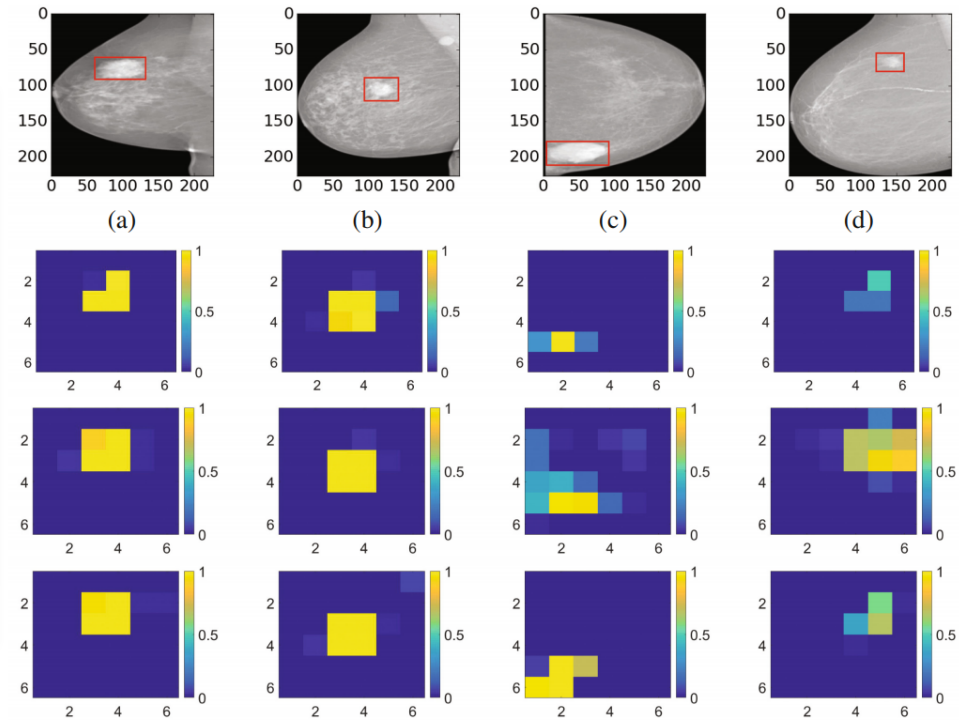


Figure 13: The visualization of predicted malignant probabilities for instances/patches in four preprocessed mammograms. The first row is the resized mammogram. The red rectangle boxes are mass regions from the annotations on the dataset. The color images from the second row to the last row are the predicted malignant probability from logistic regression layer for (a) to (d) respectively, which are the malignant probabilities of patches/instances. Max pooling-based, label assignment-based, sparse deep MIL are in the second row, third row, fourth row respectively.

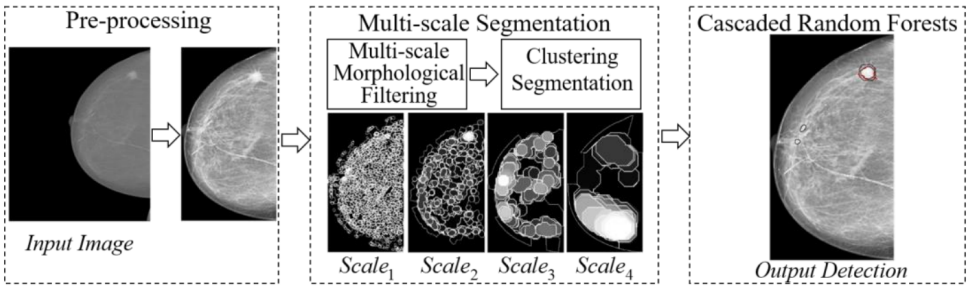


Figure 14: The diagram of the proposed mammographic CAD system

Firstly, a preprocess method, which is used in our project and elucidated in section 4.1.2, is performed. secondly, a multi-scale oversegmentation is applied to generate mass region candidates, which is corresponding to the Region Proposal Network in our project.In this stage, multi-scale grayscale morphological filtering (MGMF) and simple linear iterative clustering (SLIC) are used to extract and segment elements within the size range of breast masses at multiple scales.Finally, They use a cascaded ensemble learning approach as the base classifier to classify the region candidates generated by the previous stage.

2.5.2 Results

One example of the clustering segmentation result can be found in Figure 4.1.2. And before introducing the classification evaluation result, some terms they used must be specified in advance.

Definition of positive and negative masks Different with the definition in traditional object detection/segmentation tasks, The masks are labelled as positive or negative considering if their overlapping ratios (OR) with the ground truth pass a certain threshold (ORT). The ORT is 0.5 for our used dataset, DDSM considering the fact that it does not include precise annotations of masses compared with other mammogram dataset such as INbreast [31]. The Figure 15 shows some "positive" masks used in the experiment.

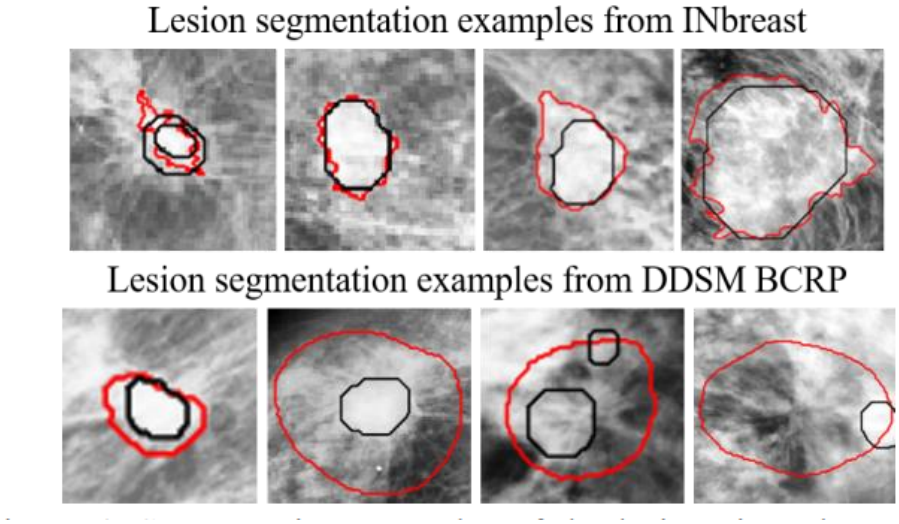


Figure 15: Segmentation examples of the lesions in INbreast and DDSM. The black lines represent the segmentation of our method, and the red lines are the ground truth.

False Positive Per Image(FPPI) The FPPI is defined as the number of masks labeled as false positive per image, the higher the value is, the user will be harder

to figure out the truth box. A performance comparison between this work and

Method	Mean Sensitive@FPPI
This work	0.77@3.93
[32]	0.75@4.8,0.7@4
[33]	0.70@8

Table 2: Comparison between this method and other works.

previous approaches is shown in Table 2.We can see that this method demonstrates competitive performance in both locating and segmenting the masses that compares favorably to the state-of-the-art.

3 Technical Support and Preliminary Study

3.1 Breast Cancer Diagnosis

In this section, we will discuss about the medical background we studied for this project. Topics covered include histopathological image, mammography image, pathophysiology and current diagnosis method of breast cancer.

3.1.1 Histopathological Image

Microscopic biopsy image is the standard tool for pathologists to diagnose breast cancer. Pathologists will inspect the size, shape, structure of cells and tissue and try to find some specific dangerous features in the image. Some signal used in this procedure include how each cell looks like, how each nuclei looks like and how the tissue looks like. Figure 16 [34] is a sample from the database.

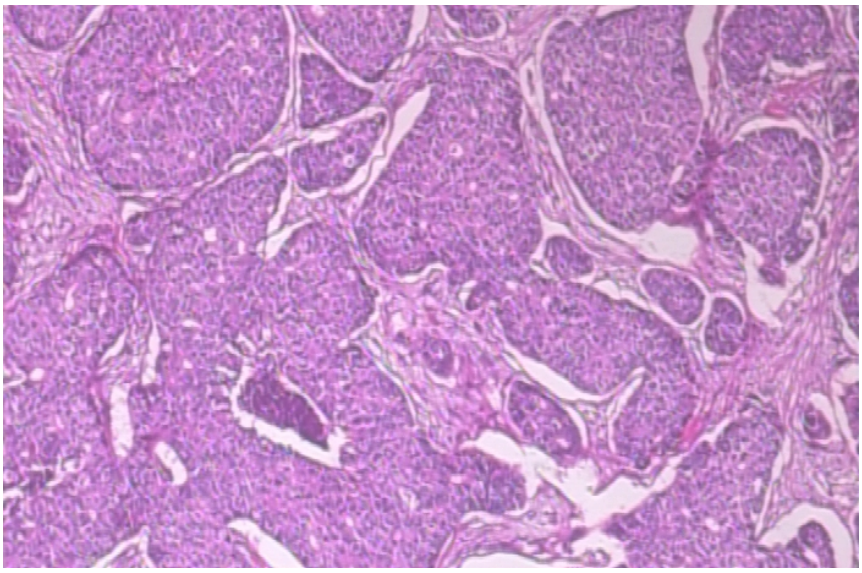


Figure 16: Sample of Histopathological Image

Shape and size of the cells Observations show that cells in a piece of tissue usually do not derivate too much from the average overall size and shape. However, a cancerous cell will lose its normal appearance, being either bigger or smaller than other cells. Well-functioning cells have even shapes and structures. On the other hand, cancer cells hardly function in a meaningful way, often with their shapes uneven.

Size and shape of the cell's nucleus Cancer cells often do not have a nucleus with normal size or shape. On the contrary to healthy nucleus, cancer nucleus is less likely to be located at the center of the cell. The cancer cell tends to have an

appearance like an omelet, where the nucleus is the yolk. The nuclei of it is also bigger and darker compared with that of a normal cell.

Distribution of the cells in tissue Besides things inside each cell, the functionality of tissue also depends on how cells are distributed and arranged. If the number of healthy cells is reduced, the overall texture and even color will also change accordingly, which leads to the shape and morphology features pathologists can directly observe from the tissue. This is more significant in diagnosis.

3.1.2 Mammography Image

Mammography image is another standard tool for pathologists to diagnose breast cancer. It uses low-energy x-ray system to image the inner side of the breasts. It enables pathologists to have a closer look to the structure of small breast cells, and especially to find abnormal tissue growths or micro calcifications. Since it is non-invasive, it is more convenient than a biopsy procedure which will obtain actual tissue from specious area. A sample mammography image is demonstrated in figure 17.

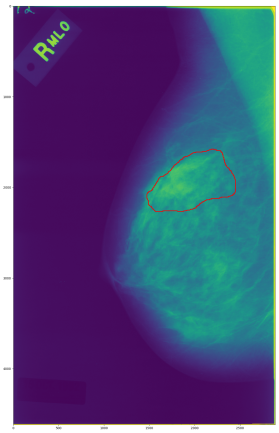


Figure 17: Sample of Histopathological Image

3.1.3 Pathophysiology

We investigated the pathophysiology explanation of breast cancer. This will help us understand features in the images, and help us develop a system more specific to our task.

Cancer is immune defense failure The immune system normally seeks out cancer cells and cells with damaged DNA and destroys them. Breast cancer may be a result of failure of such an effective immune defense and surveillance.

Cancer involved stromal cells and epithelial cells These are several signaling systems of growth factors and other mediators that interact between stromal cells and epithelial cells. Disrupting these may lead to breast cancer as well.

Risk factors of cancer vary

1. Age: The risk of developing breast cancer increases with age.
2. Personal history: A personal history of breast cancer is also a significant risk factor for the development of a second ipsilateral or contralateral breast cancer.
3. Breast pathology: Proliferative breast disease is associated with an increased risk of breast cancer.
4. Family history: A woman's risk of breast cancer is increased if she has a family history of the disease.

Lifestyle contributes to cancer

1. Alcohol consumption: Alcohol consumption has been associated with increased breast cancer risk that is statistically significant.
2. Physical activity: It has been observed that frequent physical activity can lower the risk of breast cancer.
3. Obesity: Obesity, specifically in postmenopausal women, has also been shown to increase a woman's risk of breast cancer.
4. Radiation: Radiation exposure from various sources including medical treatment and nuclear explosion will increase the risk of breast cancer by a slight amount.

3.1.4 Current Diagnosis Methods

We also studied the current standard diagnosis method of breast cancer. This will equip us with the knowledge about how to simplify the traditional diagnosis process.

Breast cancer screening Breast cancer screening is defined as the medical screening process among women appear to be healthy for early symptoms of breast cancer [35]. It is proposed in the will to diagnose It is widely believed that early detection will improve patients lone-term survival rate.

Microscopic analysis of a biopsy by pathologists If the screening result is inconclusive, the doctor may require a microscopic analysis. The doctor will sample the fluid in the lump to do a further diagnosis. This procedure involves needle aspiration. If the fluid is clear, it is highly likely that the patient is healthy; however, if there exist bloody fluid, a more detailed microscope inspect will be needed and it is possible that the lump is affected [36].

This method is the most widely employed procedure. However, it is also laborious and time consuming. The probability of misdiagnoses is high because there can be too many variations in the process. Considering the incredible amount of data involved, it is a huge work.

3.2 Image Processing

Preprocessing is an important step in the process. The phrase "garbage in, garbage out" is particularly applicable to our project. Though the image gathering methods are often strictly controlled for our dataset (i.e. same microscope), the original data still have different attributes such as brightness, contrast and saturation. Analyzing data that has not been carefully normalized can produce misleading results. Thus, the representation and quality of input data should be assured before training.

3.2.1 Feature Detection

Feature detection is a concept in the area of computer vision and image processing. It means use computer to extract information from image and to decide if each pixel of the image belongs to a feature or not. A sample of feature detection is illustrated in figure 18

Up till now there is no universal definition of "useful" or "accurate" features. The precise choice of features usually depends on the problem or specific application. It is a primary computation of many computer image analysis algorithms, in other words, the start point of them. It checks each pixel to determine if a feature can be extracted from that pixel. Therefore, whether an algorithm can succeed sometimes is determined by the features it defines and uses. There are many feature detection algorithms developed to meet different kinds of requirements. Features they extract vary; their computation complexity and repeatability also differs. Some most popular shape features include perimeter, area, compactness and smoothness. Textual features such as grey scale are also used. There are no general rules for choosing features – we can only choose by experience and experiment, which adds difficulty to image classification tasks.

Fortunately, the idea of Neural Networks saves us from the work. They are designed to require little preprocessing – All the works is done automatically be the program. This ability of learning the features is the first reason why people invented Neural

Figure 18: Points Detected in Sample

Networks. However, we still need some slight amends to ensure things will not go wrong.

3.2.2 Data Augmentation

There is another thing to note: data augmentation. In deep learning, to avoid the well-known overfitting problem, we usually need to feed enough data into the model. Therefore, the amount of available data sometimes is the most critical issue for deep learning. The problem is high quality data is expensive and limited. One method to overcome the shortage of data is data augmentation. We need to perform geometric transformation on the original dataset, change pixel positions of the image while keep the original features.

Figure 19: Demonstration of Data Augmentation

Data augmentation is very likely to improve accuracy since the model can see more samples. The exact amount depends, though. There are many ways to augment the dataset. Adding noise is an intuitive approach. More generally we have simple

transformations. For sparse holes in the dataset, we can perform dimensional reduction. Several more complicated ways include combinations of rotation, translation, rescaling, flipping, shearing, and stretching.

3.2.3 OpenCV

Open Source Computer Vision Library (OpenCV) is an open source library dedicated to the field of machine learning and computer vision. It was built with the idea to provide a reusable common infrastructure for computer vision applications, and to encourage the use of machine learning in real products. The library was originally proposed by the CPU company Intel, and was later maintained by other organizations.

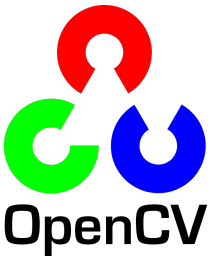


Figure 20: Logo of OpenCV

There are more than 2500 optimized algorithm included in this library. This includes both traditional and most advanced machine learning algorithms. This brings us convenience in developing our deep Neural Network.

OpenCV support programming languages from C, C++ to Java and Python. The main focus of it is to improve computational efficiency and therefore to enable interactive applications that can respond quickly to changing inputs. It has a backend optimized with C/C++, and can take the full advantage of multicore processors. It can also utilize hardware acceleration provided by different platform.

3.3 Convolution Neural Network

Most importantly, we searched for the latest technology and tools in the field of deep learning. With these knowledge, we will try to build a more advanced deep learning program.

In machine learning, convolutional neural network is a type of feed-forward neural network. It is inspired by biological processes in animal vision system [37]. Various projects have applied convolutional neural network in analyzing visual imagery. In recent years, Convolutional Neural Network has become the state-of-art in image recognition problems, beating different competitors. It has been observed from existing papers [38] [39] that CNN is feasible to do microscopic and macroscopic

image classification tasks, and is possible to surpass other classifiers. It is now believed to be the first choice for image classification type tasks.

Just like other Neural Networks, CNN consists of an input layer, multiple hidden layers and an output layer. A notable feature of CNN is that it assumes inputs are pictures. In this way, it can do some more specialized optimization. In convolution layers, the neurons will only connect to a limited region of the previous layer. This reduced computation complexity, and enables CNN to make full use of the 2D structure of the input data. Therefore, compared to other deep learning architecture, CNN can often lead to better result in image or speed recognition.

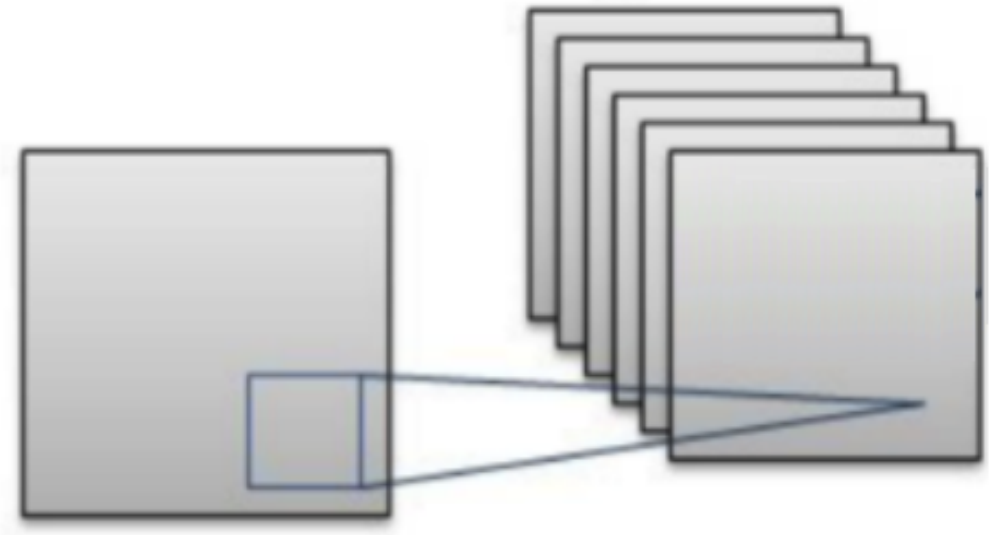


Figure 21: Illustration of Convolution

CNNs use less preprocessing than other image classifiers. This feature learning property reduces requirement of prior knowledge and hence human effort, making CNN an attractive architecture.

3.3.1 Input Layer

This is the first layer of the network. It received non-linear input data and prepare data to be fed into convolutional layers after it. Some simple transformation such as normalization can be applied in this layer. It produces the initial feature maps. In our experiment, the input is an image, and the network is parameterized according to the image width, height and depth.

3.3.2 Convolutional Layer

The convolution layer takes data from previous layers and a group of trainable filters as input. A filter is just a neuron connected to a limited area of the previous layer.

Each filter will produce a feature map in the output. In the convolution layer, filter will do convolutional computation on local input data. The data window will keep sliding after filter finishes the local computation, until it finishes all data from the previous layer. A sample is illustrated in figure 22.

While the input data may have a large size, the filter will only compute the convolution on a partial data window, which is called local perception mechanism in CNN. It is a simulation of animal focusing on a specific object. Meanwhile, as the data window slides and the input data changes, the filter weight is fixed during this iteration; in other words, focusing on different area will not change the way an animal see the world. This is the weight sharing concept in CNN.

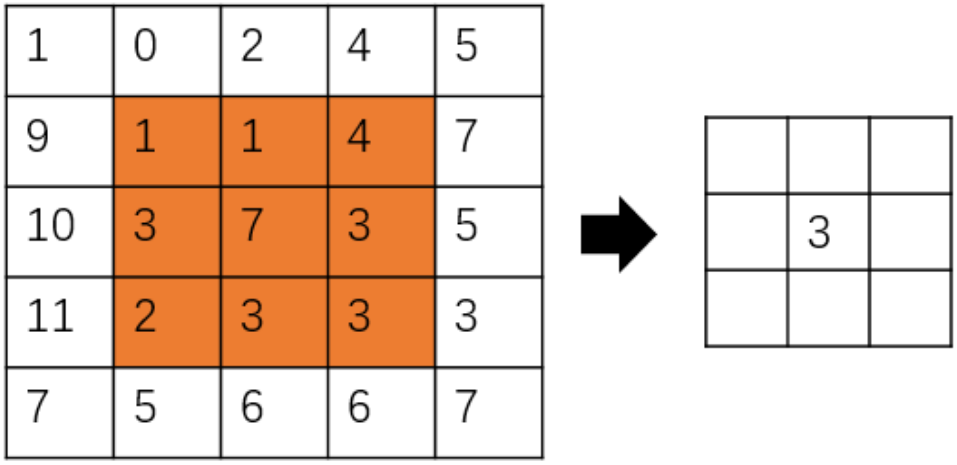


Figure 22: Convolution Layer

3.3.3 Residual Layer

The idea of stacking up more layers is not new, but it became attracting only recently, as a result of the rapid development of Graphic Processing Units. GPUs can perform high computational intensive tasks at pretty low cost, thanks to their parallel architecture. However, as the depth of the network increase, the accuracy may not proportionally increase.

Moreover, deeper networks will face the vanishing gradient problem. The problem becomes more serious when the network is going deeper. The hidden layer near the output layer will update its weight normally, but the layers in the front of the network can only update their weights very slowly, which makes the weights almost unchanged after training. It makes the first several hidden layers merely a forward layer that do a same mapping for all inputs. The deep network is now just equivalent to a shallow network with the last several layers.

He et al. [40] presented a residual learning framework to ease the training of networks that are substantially deeper than those used previously in 2015. They explicitly reformulated the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. There is empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset, He evaluated residual nets and achieved 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task.

If multiple non-linear layers can be approximated by a function, we can also represent the residual of this hidden layer as a function. Suppose a hidden layer is $H(x) - x \rightarrow F(x)$, we can intuitively have

$$H(x) = F(x) + x$$

Then we can have the residual block. The output of the residual block is the sum of the output of multiple cascade convolutional layers and the input element itself, activated by an activation function where we choose ReLU.

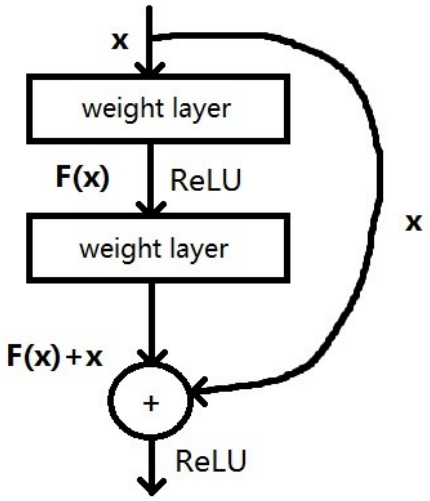


Figure 23: A Simplified Residual Block

The residual network has some nice features. It is thin, having the number of parameters under control. There is layered structure which can ensure the feature expression ability of the network. It can perform subsampling without pooling layers, and therefore improved the efficiency of back propagation.

For actual usage, the number of convolutional layers wrapped by a residual block may depends on scenario.

3.3.4 Dropout

From experience, overfitting is a common problem in deep neural network. Due to that large amount of trainable parameters in CNN, the model may simply memorize all train data without figuring out the internal regulation in the dataset, and cannot be generalized to new data, which leads to high accuracy on train set but low accuracy on test set.

Dropout refers to the method that temporarily disable some neural network unit at some certain probability during the train process of CNN. The discard is temporary, and its weight is preserved. For random gradient decline, since units are randomly disabled, the training is actually on different networks for each mini-batch. It forces one neuron to work with other randomly selected neurons, forces "free riders" to be trained equally, and hence decreased the correlation among neurons. In this way, we are actually training 2^n models for a neural network with n nodes, while keeping the number of parameters unchanged. In other words, we are training more models with the same computation complexity. This results in a visible improvement in the generalization ability of the network.

3.3.5 Pooling Layer

The pooling layers are used to perform subsampling. The size of its output will be reduces, but the depth will keep unchanged. It will reduce the amount of data and the number of parameters in the model. During the training process, it can therefore lower the computation complexity and avoid overfitting. The polling layer uses the same sliding window mechanism as convolution layers, and is defined as

$$y = \max_{local\ window}(x)$$

In our model, there will be a pooling layer after each convolution layer, so their activity is strictly determined by convolution layers.

3.3.6 Activation Layer

Activation layers are introduced for adding non-liner classification ability to neural networks. Though it is logically just a function, usually we regard it as a layer. In our model, we use Rectified Linear Unit (ReLU) as the activation function. It is a commonly used one for CNN. The ReLU function is defined as

$$f(x) = \max(0, x)$$

As shown in graph 24, the activation function ReLU that we used is just a threshold at zero. It is proved to be a better simulation of animal brains [41]. For particle usage, it simplifies the computation required.

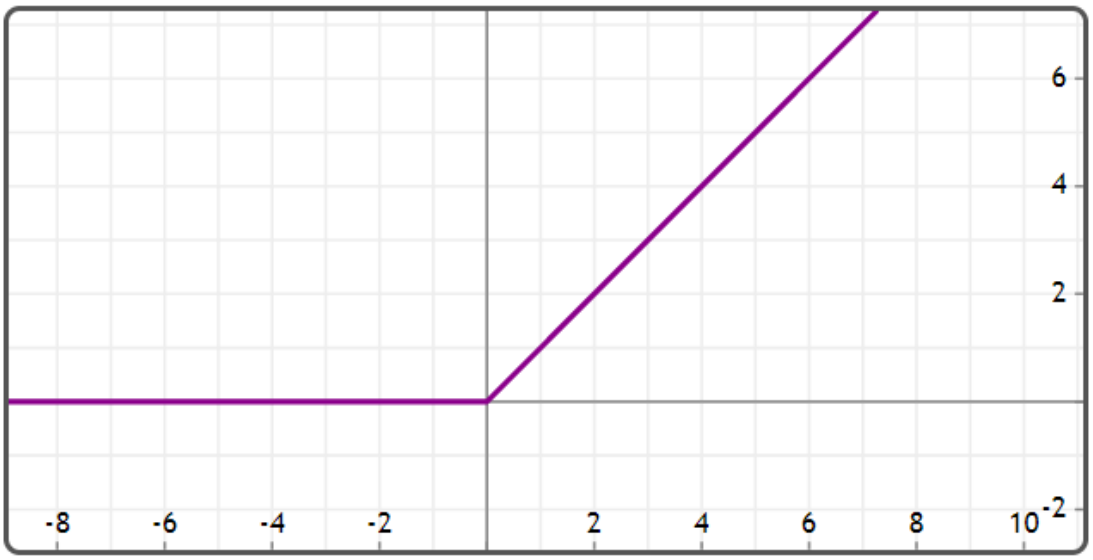


Figure 24: ReLU

3.3.7 Fully Connected Layer

The fully connected layer has connection to all neutrons of the previous layer. We have only one fully connected layer. It is used at the end of the network to produce final prediction results.

3.3.8 Tensorflow

In our project, we use Tensorflow. It is also an open source software library. By using data flow graphs, it is capable for large scale numerical computation, one of which is machine learning. Besides fast speed, it also supports various high-level APIs for machine learning programs [42].

Figure 25: Logo of TensorFlow

Tensorflow supports platforms with or without GPU, from mobile, desktop to clusters. With limited overhead, Tensorflow + Python environment provides a much clearer program: we describe the data flow diagram with Python, benefiting from the conciseness of this language; then Tensorflow will execute the diagram with a C++ or CUDA backend, making full use of the computer hardware.

Tensorflow introduces two new concepts: Tensor, and data flow graph. Data flow graph is a graph whose nodes are Tensors. Tensors are actually matrixes; however, they can be connected to from a data flow graph. The matrix together with the relations defines a Tensor. The word "flow" means that data will flow from one node to another, and the computation occurs in the transition. This gives us a very good simulation of CNNs: they both are graphs, and both incur computation during transitions.

We are using Tensorflow 1.3.0, which was the latest version available at the time we start to develop our project.

3.3.9 Comparing Tensorflow with Other Tools

Generally speaking, Tensorflow is more friendly to beginners than other tools like Caffe. This partly results from Google, the author of Tensorflow. Most other tools are supported by university academics, while Tensorflow is supported by a commercial company. This results in difference in available documents, tutorials and communities. Developing with Tensorflow is generally more comfortable.

Though tools built for academics can provide a more detailed control over the model, this feature is mostly not required for implementing a model that has already been tested for many times. On the other hand, Tensorflow is more high-level, providing conciseness in development.

Developer can use Tensorboard, the bundled debug tool along with Tensorflow, to monitor real time statistics of the diagram. Considering Googles' experience in user interfaces, debugging Tensorflow models is much more convenience than debugging Caffe models.

Moreover, as Google is a commercial company, Tensorflow is designed for production usage at the very beginning. We can easily export the model trained by Tensorflow and set up a RESTful query server in a couple of lines. As our project is a medical project, we should expect users may not have much Machine Learning background. The ease in pushing experiment results to production is an advantage.

3.4 Regional Convolution Neural Network

Regional Convolution Neural Network is one of the most popular application of CNN, which is to solve object detection and segmentation problem,in a word, once an image is given, the model should be able to detect instances of semantic objects of a certain class (such as benign mass and malignant mass in our project). The key of the Reginal Convolution Neural Network is that **feature matters**, which means that the feature of image calculated by CNN can be applied to many tasks with a high performance. Usually, the procedure of objection detection in region-based CNN [43] usually includes several parts, Region proposals generation, feature

extraction, proposal classification and bounding box regression. Each part achieved a great breakthrough in recent years and the specification is illustrated in Table 3,the details of each part will be covered one by one.

Part	Functionality	Input	Output
Region Proposals Generation	Generate possible area including objects we need to detect	The whole image (1024,1024,3)	A list indicating possible areas (Number of proposals,4)
Feature Extraction	Generate a feature map of image	the whole image (1024,1024,3)	Corresponding feature maps (H,W,N), the size depends on the architecture of base model
Classification and Regression	Detect the class of proposal and suggested regression value	Proposals generated from part Region Proposals Generation	Predicted label and suggested regression value.
Mask Generation	Generate the predicted mask for each ROI	Proposals generated from part Region Proposals Generation	Corresponding masks with binary type

Table 3: A summary of the functionality of each part in Regional Convolution Neural Network

3.4.1 Region Proposals Generation

Region proposals generation is the first step for object detection and is for generating multiple proposals for following classification, where proposals mean "possible" area of target object. Two commonly used methods will be introduced.

Selective Search In an object recognition program, it is impossible to distinguish different classes with a single strategy. We need to take into consideration the diversity of different objects. For example, a car consists of a body and tires, but these two part may have very different texture or color. Therefore, the hierarchical nature of object definitions will be important in the design of a good algorithm.

Selective search was proposed in 2011 [44] to find region proposals in an image. The key idea in it is called "multiscale". Firstly, it can capture all scales. exhaustive selective changes the window size to do so, and selective search also cannot avoid this issue. However, with image segmentation and a hierarchical algorithm, selective search effectively relieve this problem. Secondly, it can handle diversification in the images. It uses color, texture, size and various other strategy to merge regions found in the first step. And more preferably, it is fast to compute.

For the region search step, this algorithm does not resize the image or the window. Instead, it utilizes an image segmentation method by Felzenszwalb, Pedro F and

Huttenlocher, Daniel P [45] to compute the regions in the image, compute the similarity between each pair of regions, merge the most similar two regions, and iterates through this process until the whole image is merged into a large region. Then it assigns a score to each region, and extract the top-k ranking subset.

To deal with the diversification problem, this method introduces two strategies. The first one is color space diversification, and the second one is similarity computation diversification. Color space diversification applies eight different ways to generate the raw region in the region search step. Similarity diversification is achieved by four different ways to merge similar regions.

At the publish time, this algorithm was used together with the traditional feature based SVM object detection model. It proved to have good accuracy.

Region Proposal Network(RPN) The idea of RPN is proposed in [46] because traditional method, selective search is too time-consuming and has become the bottleneck that limits the development of real-time object detection.

A Region Proposal Network (RPN) takes an image as input and outputs a set of rectangular object proposals, each with an objectness score. Firstly, the image will be the input of one base model, which shares weights with other part's networks, the base model will output a corresponding feature maps.Then, one **sliding window** of size 3×3 will scan the whole feature maps, at each sliding window location, corresponding region proposals in the feature maps will be fed into RPN, which is actually two sibling 1×1 convolutional layers followed by two sibling fully-connected layer, a box-regression layer(**reg**) and a box-classification layer(**cls**). The proposals here are also called **Anchors**. The mini-network is illustrated at a single sliding window position in Figure 26.

3.4.2 Feature extraction

Feature extraction is the key reason for a successful RCNN because the features extracted in this part will be used and weight-shared in most procedures. Also, the difference of this part is only the difference of base model, in our project, we use ResNet illustrated in Section 3.3.3 as our base model. The feature maps are usually obtained using the original base model excluding its last fully connected layers so that keeping previous convolution layers only. Usually, the feature map is the representation of whole image while in other parts, such as Classification and Regression, we only consider the features inside any valid region of interest, and also the CNN requires a fixed input size, which means that we need some tricks to enable converting the features inside any valid region of interest into a small feature map with a fixed spatial extent of $H \times W$.

ROI pooling The region of interest (ROI) pooling layer is for solving the problem we mentioned above, RoI max pooling works by dividing the $h \times w$ RoI window

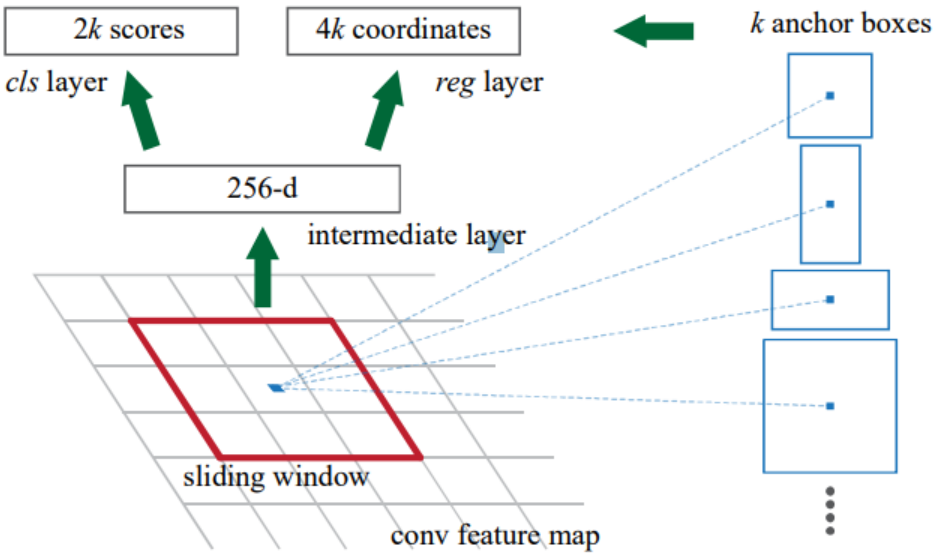


Figure 26: Region Proposal Network (RPN)

(ROI's corresponding area in feature maps) into an $H \times W$ grid of sub-windows of approximate size $h/H \times w/W$ and then max-pooling the values in each sub-window into the corresponding output grid cell. Pooling is applied independently to each feature map channel, as in standard max pooling.

RoIAlign ROI pooling firstly quantizes a floating-number RoI to the discrete granularity of the feature map while quantization is robust to small translations, it has a large negative effect on predicting pixel-accurate masks.

To overcome the deficiency of ROI pooling,RoIAlign layer is proposed in [47] that removes the harsh quantization of RoIPool, properly aligning the extracted features with the input. The proposed change is simple: avoid any quantization of the RoI boundaries or bins (i.e., use x/16 instead of [x/16]). Use bilinear interpolation to compute the exact values of the input features at four regularly sampled locations in each RoI bin, and aggregate the result (using max or average), see Figure 27 for details.

3.4.3 Classification and Regression

Classification and Regression is for the classification of proposals generated in Region Proposals Generation part and the suggested regression of the proposals including shift and scale.

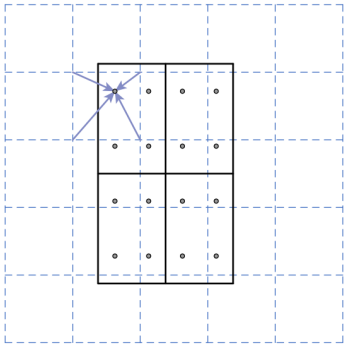


Figure 27: The dashed grid represents a feature map, the solid lines an RoI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points.

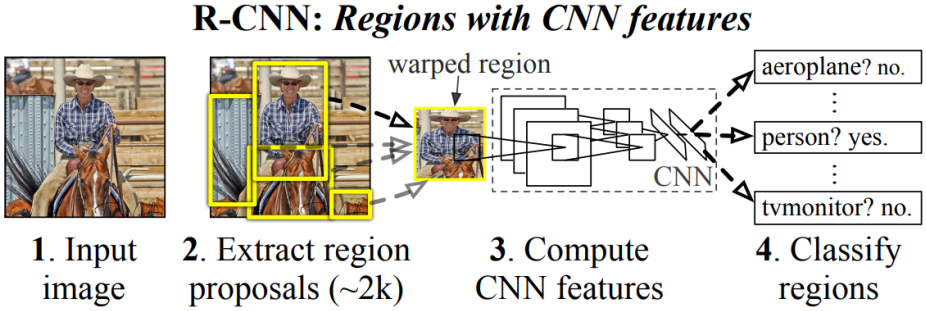


Figure 28: **RCNN overview.** the system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs

SVM Calssifier SVM is the classic method for classification. This method is applied in [43] for the classification of proposals behind the Feature Extraction part, the overview of the system can be found in Figure 28, where the fourth step is the SVM classifiers using features of proposals as input.

Fully Conected Layer classifies In [46], SVM classifier is substituted by a fully connected layer connecting the feature maps directly, the fully connected layer regard the feature maps as input and output a class-specific array indicating the confidence of each class using softmax.

Bouding Box Regression The proposals generated in Region Proposals Generation Part cannot be the final object location result and a bounding box regression stage can be applied to improve localization performance. The regression method also adopts fully connected layer method, which is similar to its classifier and actually the two layers use same feature maps as input. The architecture overview is illustrated in Figure 29

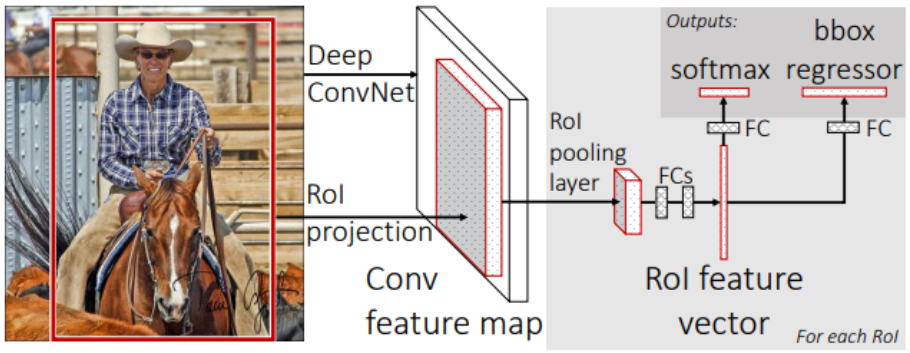


Figure 29: Fast R-CNN architecture.The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets

3.4.4 Mask Generation

A mask has a KHW dimensional output for each RoI, which encodes K binary masks of resolution $H \times W$, one for each of the K classes. The architecture for mask generation is a series of convolution layers connecting the feature maps directly, and can be regarded as another branch of classification layer.

3.5 User Interface

One primary goal of our project is to provide a convenient interface for doctors without deep learning background. A web portal will be more friendly to end users.

In this section, we will discuss the technologies to build a modern web application that requires no prior training to use.

3.5.1 Web Application

Web application(Webapp) is a kind of application that uses browser as the client program that handles the user interface and all client-side logic. This concept is used widely in nowadays services, such as webmail, online store and online instant messaging.

This kind of application requires no installation of an extra client software, and can be easily ported to different platforms. Once written, the program requires only slight capability changes to work on multiple browsers. For users that do not know how to or simply do not want to install softwares, webapp provides a simple solution.

The other advantage of webapp is that it can update the content of the program dynamically. The webapp can asynchronously load resources from server and react to users' input responsively. This provide us the opportunity to use the browser as merely an input collector, and process the actual data at the server. For computation intensive tasks that denends on user input, this provides good load balance between client and server.

3.5.2 Node.js

Node.js is a cross-platform javascript runtime for servers based on Chrome V8 engine. It utilizes an event-driven non-blocking model, making it light and efficient. It can optimize the transport of programs, and thus is usually used for data intensive program.

Figure 30: Logo of Node.js

Node.js has most of its core module written in javascript. Before node.js, javascript is mainly used as a client side language for programs run inside browsers, and is not considered as a language for complex tasks. The creation of node.js enables javascript on server. It has various built-in moudles that can be used as a standalone server without Apache or IIS. Therefore, node.js dramatically reduces the learning cost to switch between languages for developers.

Node.js runtime provides a simple and concise application interface to communicate with. It is easy to develop a javascript object notation(JSON) interface with it, since it supports JSON natively as a language feature.

4 Method

4.1 Histopathological Image Classification

This is what we have done in term one. We used the BreakHis dataset to train our ResNet convolutional neural network. The following sections will introduce the design of this model.

4.1.1 Dataset

For our project, we are using the Breast Cancer Histopathological Image Classification (BreakHis) dataset. It is composed of 9,109 breast tumor tissue microscopic images. The researchers collected samples from 82 patients, and used different magnifying factors (40x, 100x, 200x, and 400x) to process them [40]. The statistics of samples are illustrated in table 4

Class	40x	100x	200x	400x
Benign	625	644	623	588
Malignant	1370	1437	1390	1232
Total	1995	2081	2013	1820

Table 4: Distribution of Images

The samples are stained with hematoxylin and eosin. The author of the dataset uses breast tissue biopsy slides to generate these samples. Pathologists from the P&D lab labeled them. The breast tumor specimens were asses by Immune histochemistry. The biopsy procedure was Surgical Open Biopsy.

An Olympus BX-50 system microscope was used to capture the images. As aforementioned, they captured image under four magnification factor, 40x, 10x 200x and 400x. The raw image was stored into the dataset without any normalization of color standardization to avoid loss of information and complexity in analysis. The images were in Portable Network Graphics (PNG) format, in 3-channel RGB, 8-bit depth. A sample was demonstrated in figure 31.

4.1.2 Preprocess

In this section, we will discuss how we manipulate the image before feeding it into the model. We proposed different methods, and would compare them in experiments.

Data Augmentation Since we are training a deep learning neural network, the amount of train data is a critical problem. The size of the original dataset, 9109, is relatively small for our model, and is therefore very likely to cause overfitting. Summarizing the methods used in past works [41], we can propose multiple ways to extend the dataset systematically.

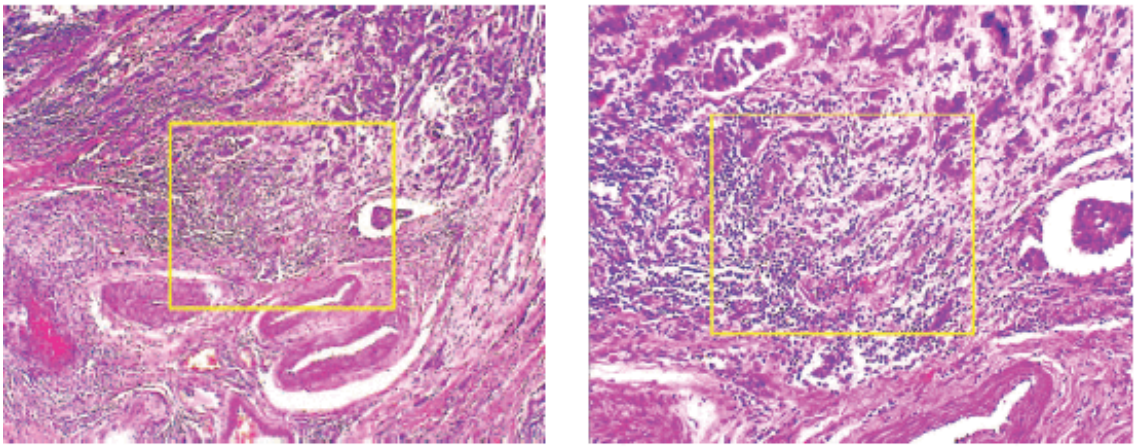


Figure 31: Same Tumor under Different Magnification

We do not propose any color standardization since all images have the same color pattern, i.e. pink or purple. This is due to the stain method applied to tissue samples. The data augmentation methods we propose include only geometric transformation. They include:

1. rotations: random with angle
2. translations: random with shift
3. flipping: true or false
4. shearing: random with angle
5. stretching: random with stretch factor between 1/1.3 and 1.3

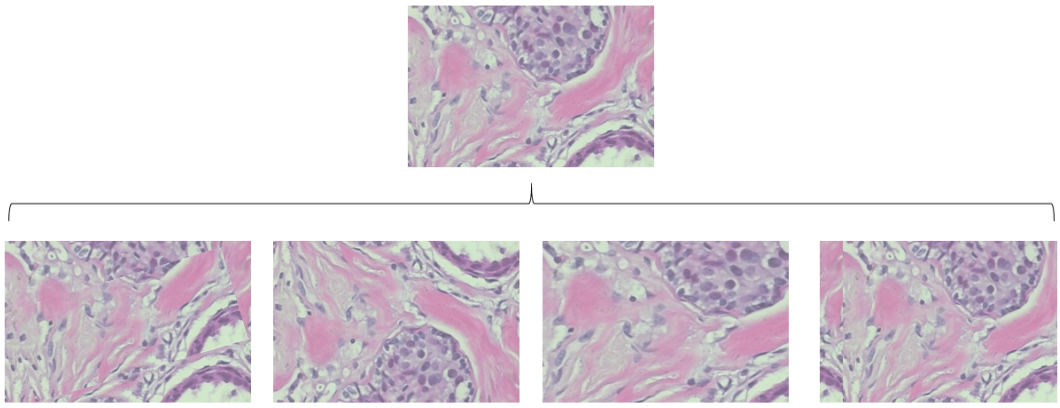


Figure 32: Examples of Data Augmentation

Sliding Window Crop It is hard to process the high-resolution images since applying deep learning algorithms on larger image sizes will tend to make the model

architecture more complicated. The model will usually have more layers, more parameters which increase the complexity dramatically. Training and tuning the model may be very costly in such case.

One way to solve this problem is sliding window crop. Set a window of size n×n, slide through the image at $step = 0.5n$, and then crop [42].

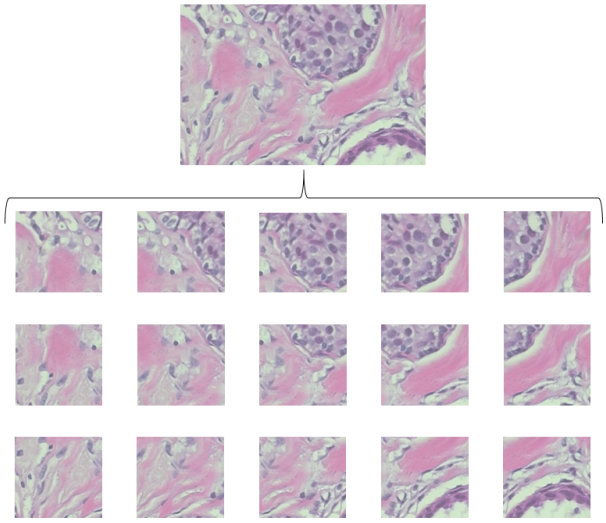


Figure 33: Examples of Sliding Window Crop

Overlaps between crops are deliberately designed to avoid damaging the structure information too much. The number of total crops is given by the following formula:

$$\#(crop) = 2 \times \frac{IMGWIDTH}{n} \times 2 \times \frac{IMGHEIGHT}{n}$$

Random Crop Another way to solve the aforementioned oversized problem is random crop. Set a window of size n×n, do random crop instead of sliding. This is similar to the previous method.

The number of total crops is not fixed. However, a higher number of crops will give more information. There will be no limit on how the random selector crop: it may or may not capture the most important features.

For benign samples, there will be no problem. However, for malignant samples, we cannot make sure tumor exist in every crop. Crops extracted from malignant images may actually contains no tumor and should be classifies as benign. This introduces noise in train data.

The gain, on the other hand, is we keep the size of network small. This benefits in various ways: less computation complexity, less logic complication, and most

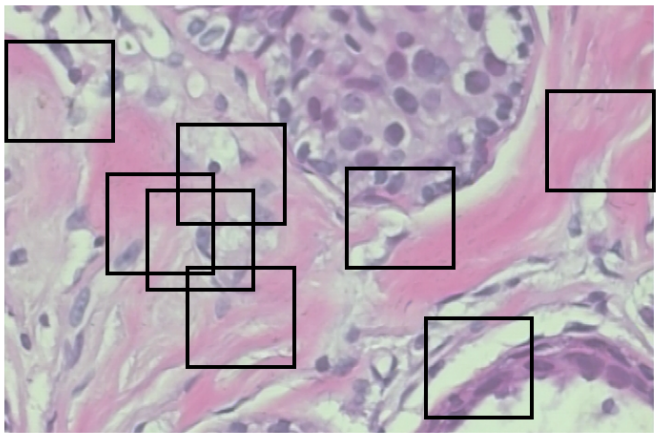


Figure 34: Examples of Random Crop

importantly, it reduces chance of overfitting by limiting the parameters of the model to a reasonable amount.

Resizing There always exists the method of simply shrinking the image. To avoid moiré after resizing, we will resample the image using pixel area relation. This is the best image interpolation method for decimation since it tends to give a clearer image. This makes the high-resolution image generation pointless, however.

Whitening Whitening is the one of the standard preprocess methods for machine learning. The main idea is to remove extra information dimensions in the image. First, we represent the input dataset as

$$\{x_1, \ldots, x_m\}$$

Then we computes the covariance matrix of x

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} x_i x_i^T$$

Therefore, we can have

$$x_{rot} = U^T x$$

where U is the eigenvector of Σ.

This process maps x into a new space that eliminates the correlation between features. Then we can have

$$x_{PCAwhite} = \frac{x_{rot}}{\sqrt{\lambda_i}}$$

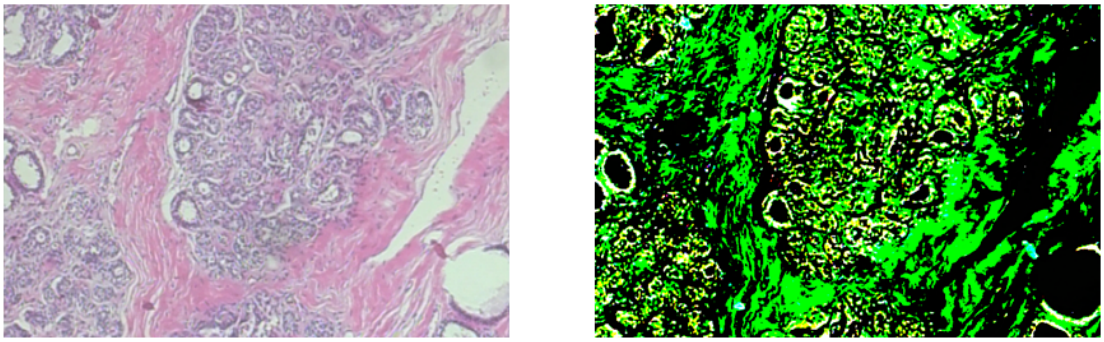


Figure 35: Before and After Whitening

which normalizes the dataset [48].

After whitening, the new image satisfies two properties: features are less correlated, and features have the same variance. This will significantly accelerate the training process.

Contrast Limited AHE Contrast-Limited Adaptive Histogram Equalization (CLAHE) can improve local contrast without damaging the image too much. Consider an image whose pixel values are limited to a specific range, it would be better to have the values distributed in all regions of the channel. This will usually improve the contrast of the image. Therefore, we need further scatter pixels clustered in the "brighter" regions.

Adaptive Histogram Equalization (AHE) will do this work. However, it sometimes will cause loss of information due to over exposing some region that is already bright. This is because the image is not perfectly limited in a small region of the channel. To solve this problem, we can use CLAHE [49]. The image is divided into tiles, and each tile can perform AHE on its own. For a tile, the brightness across this small area is more likely to be confined. In this way, the image will be clearer.

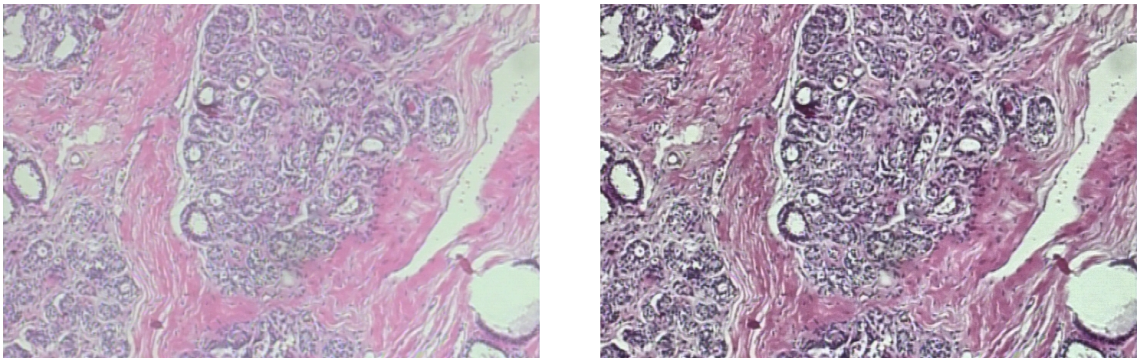


Figure 36: Before and After CLAHE

Generally speaking, CLAHE is more important than whitening for deep neural networks since the network can learn how to whiten images itself without manually specify it should do it.

4.1.3 Model Construction

In this section, a detailed description of our normal model architecture will be given. To understand the construction easier, an visualization version is shown in Figure 41, left, the format of the text in the box represents kernel size and output channel number of the box's corresponding layer.

Firstly, we considered the input tensor $\boldsymbol{I}$ with size $(N, H, W, 3)$, where N, H, W are number of batch, image height, image width respectively, which are specified as a hyper-parameters and 3 indicates that the image has 3-channel(RGB). When we feed the input tensor into our model, two conventional convolution layers and one maximum pooling will firstly be passed through. The output tensor size will become $(N, H/2, W/2, 16)$.

The output tensor will be transformed into Residual Network, one box of the Residual Network part in Figure 41 is actually one residual block which is introduced in section 3.3.3. When the dimensions increase, the layer is performed with a stride of 2 so that the feature maps size is half. Therefore, the tensor size will undergo a transformation: $(N, H/2, W/2, 16) \rightarrow (N, H/4, W/4, 32) \rightarrow (N, H/8, W/8, 64) \rightarrow (N, H/16, W/16, 128) \rightarrow (N, H/32, W/32, 256)$

After passing through all residual blocks, a average pooling layer of will be inserted, making the tensor size changed from $(N, H/32, W/32, 256)$ to $(N, 1, 1, 256)$ Then, a fully-connected layer which indicates the prediction labels (binary classification tasks: malignant or benign) will be deployed so that the final tensor shape is $(N, 1, 1, 2)$.

4.1.4 Model Evaluation

In the previous part, we introduced kinds of methods to do image preprocess, image segmentation and model construction. In this part, we will compare different methods and parameters together and compete with the past paper result using the same dataset to see whether our model is optimized enough.

Following the experimental protocol proposed in [50], we used cross-validation method [51] to do evaluation, the dataset was split so that patients used to build the training set (75% patients) are not used for the testing set (25% patients) to guarantee that our model can generalize to those patients not in the dataset, the results presented in this work are the average of four trials with the selected results after converging and a suitable early stop.

Training protocol used here is the purely supervised type, the Stochastic Gradient

Descent (SGD) method [52], with backpropagation to compute gradients was used to update the network's parameters. All fixed hyper-parameters of training are given in the Implementation section.

The ResNet model were trained on a NVIDIA Tesla K40m GPU [53] using the Tensorflow framework. Training took about 5 hours for the 256×256 input size and 10 hours for the 512×512, which is corresponding to a much more complex training set.

When we discuss the results of medical images, there are three ways to report the results in our report: batch level, image level and patient level.

Batch level can be understood by batch-wise, the unit is simply each input we fit into the neuron network. The recognition error at the image level can be calculated by:

$$\text{Image Recognition Accuracy} = \frac{N_{\text{correct}}}{N_{\text{all}}}$$

Where N_{correct} is the number of cancer images which is correctly classified, and N_{all} is the number of cancer images in the test dataset.

Patient level is a little different, each patient score is defined as:

$$\text{Patient Score} = \frac{N_{\text{correct-in-p}}}{N_{\text{p}}}$$

Where $N_{\text{correct-in-p}}$ is the number of cancer images of Patient P which is correctly classified, N_p is the number of cancer images of Patient P. Then the global patient error is calculated by:

$$\text{Patient Error} = 1 - \frac{\sum \text{Patient Score}}{\text{Total Number of Patients}}$$

Besides basic error results, we also calculated confusion matrix, precision, recall and F1 score [54] on either/both image level or/and patient level. Precision, recall and F1 score are defined as:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Negative} + \text{False Negative}}$$

$$\text{F1 score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Also, we use Area under the curve (AUC) [55] to measure the performance of different models, the AUC of a classifier is equal to the probability that the classifier will

rank a randomly chosen positive example higher than a randomly chosen negative example, i.e.

$$AUC = P(score(x+) > score(x-))$$

Because there are millions of parameters and hundreds of hyper-parameters, therefore which parameters need to be tuned among the test should be considered carefully. Through our study on both medical and deep learning field, we selected three major hyper-parameters (directions) to do our test: Preprocess method, model architecture and image segmentation method.

For each hyper-parameter, we tested kinds of values or situations based on our guess and motivation, so each block below will contain several sub-blocks to explain each guess and its corresponding results in detail.

4.2 Mammogram Tumor Detection

We continue to develop the rest parts of our project in term two. We used the Digital Database for Screening Mammography(DDSM) to train our Mask-RCNN convolutional neural network. The following sections will introduce the design of this model.

4.2.1 Dataset

For our project, we are using the Digital Database for Screening Mammography (DDSM) [29] as our train dataset. It is specially designed for usage by image analysis researchers. It was a collaborative project by the Massachusetts General Hospital, the University of South Florida, and Sandia National Laboratories. There are approximately 2500 cases in this database, each of which includes a MLO view and a CC view of two breasts, together with some meta information. The statistics of samples are illustrated in table 5

Class	Count
Benign	870
Malignant	914
Normal	695
Total	2479

Table 5: Distribution of Images

The meta information associated with a image includes age at time of study, keyword description of abnormalities, scanner resolution, etc. Images with suspicious area are linked with an extra file representing the boundary and types of the region. This provides data for training a more sophisticated object detection algorithm.

This dataset consists of images from different kind of scanners including DBA, HOWTEK and LUMYSIS. The resolution varies from 42 microns to 50 microns. This adds extra complexity to data normalization. The properties of the images are slightly different from each other. Fortunately, the content of images are of the same pattern and is capable for analysis. A example is show below, where it can been seen that each image carries a different size.

4.2.2 Preprocess

Unfortunately, the DDSM dataset was made 20 years ago. It is using a data format that modern softwares do not recognize. The preprocessing step requires extra care.

LJPEG The dataset compresses images with Lossless JPEG(LJPEG) format which is developed by Stanford University. It perform best for gray scale images such as mammogram slides. Since the scanner used to generate these images were set a

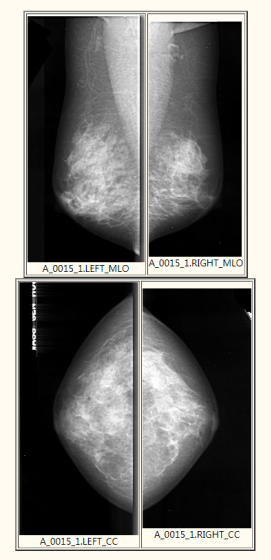


Figure 37: Sample from DDSM dataset

resolution between 42 to 100 microns, the files are very large. After decompressing, the expected output file usually has a four times larger size. However, this format not a widely supported.

Chain Code Boundaries of suspicious regions are represented by chain codes. The chain code uses one digit to specify the relative position of the next pixel in the boundary. Given a starting point, simply move the cursor up, down, left or right to draw a circle on the image, and the circle will be the boundary. The numbers to the direction correlation can be illustrated as in figure 6

	X→		
Y ↓	7	0	1
	6	·	2
	5	4	3

Table 6: Chain Code Direction

Contrast Limited AHE Similar to term 1, we also use Contrast-Limited Adaptive Histogram Equalization (CLAHE) to improve the quality of images. The detail of the algorithm is illustrated in section 4.1.2.

Image Augmentation Commonly used image augmentation methods are adopted in our project, including left-flip and right-flip. Rotation is not used considering the property of mammogram: all mammograms we used to train and test are left-view

or right-view, we do not hope to break this property. But rotation is still a good point to have a try.

4.2.3 Model Construction

Similar with section 4.1.3, this section is a detailed description on how our model works. The input tensor, firstly, is a preprocessed image including resizing, padding and cropping and the shape of the input tensor is $(2048, 2048, 3)$ to maximize the use of image information.

Firstly, the input tensor is fed into a **base model**, ResNet101 in our case, and the returned tensor are actually the feature maps of image, then the feature maps wil used as input in other models including RPN, Classification and Mask Generation. All of the functionality of these models are introduced in Section 3.4.

RPN RPN (Section 3.4.1) is the first network using feature maps as input because all other networks need proposals as input, which is the output of RPN. RPN generates a set of ROIs (still in the format of tensor) with shape $(N_{ROI}, 4)$, where N_{ROI} represents the number of predicted region proposals. Typically, the RPN outputs some "valid" anchors calculated from:

- **anchor location** $(A_l = (x, y))$ indicates the middle point location of anchor.
- **anchor scale**(A_s) indicates the scale of the anchor, there are 5 options eg.$(32\times 32, 64 \times 64, 128 \times 128, 256 \times 256, 512 \times 512)$.
- **anchor ratio**$(A_r = (r_x, r_y))$ indicates the length:width ratio of the anchor, there are 3 options eg.$(1:1, 1:2, 2:1)$
- **anchor class**(A_c) indicates whether the anchor is background or not.
- **anchor regression**$(A_{reg} = (reg_x, reg_y, reg_w, reg_h))$ indicates the regression value calculated, where reg_x, reg_y indicate the shift value of anchor location A_l and reg_w, reg_h indicate the shift value of anchor scale.

Finally, anchors whose class are not background after regression are the final predicted proposals and are regarded as input of following networks.

ROIAlign ROIAlign is the middle network to process the proposals generated by RPN and the effect of ROIAlign is for getting the corresponding areas in feature maps (from backbone model) of proposals (from RPN), the method details are explained in Section 3.4.2. Colloquially speaking, ROIAlign provides a bridge connecting RPN which outputs proposals in *image* and following fully connected layers which require *feature maps* of images.

Fully Connected Layers and Mask Generation Layer both fully connected layers and mask generation layer require feature maps generated by ROIAlign as input. Fully connected layer outputs two informations, proposal's classification and regression strategy. The mask generation layer is actually a CNN whose output is class-specific binary masks of same resolution with proposals. The relationship between ROIAlign and following layers is illustrated in Figure 38. Therefore, the final results will contain two elements: the one is the class label for the mask and another one is its corresponding masks in raw image.

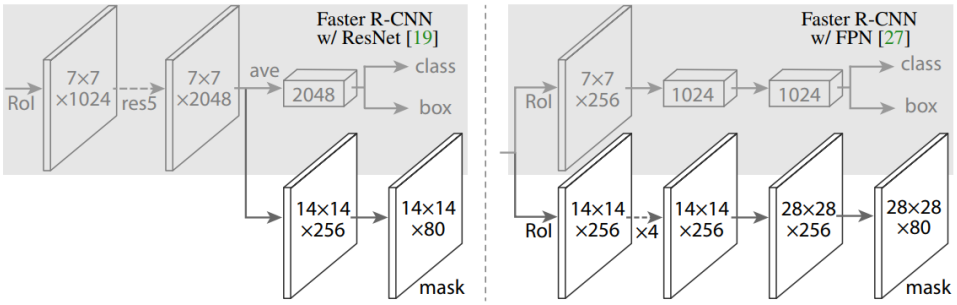


Figure 38: The overview architecture of fully connected layers and mask generation layer and the relationship with ROIAlign

4.2.4 Loss Function

This work is a multi-stage work which contains multiple sub-stacks including RPN,ROIAlign and so on. Therefore the loss function should be treated carefully because accumulated loss has a critical influence on this kind of work. Typically, **we proposed a new loss function for bounding box regression**, which has an good influence on the result. The loss of different part will also be covered one by one.

RPN loss includes two parts, classification loss (whether the predicted box is positive or not) and regression loss. The loss function for an anchor is defined as:

$$L(p,t) = L_{class}(p,p') + \Lambda p' L_{reg}(t,t'). \tag{4}$$

Here, p is the predicted probability of this anchor being an object. p' is the ground truth label and 1 if the anchor is positive and 0 if negative. t is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t' is the corresponding parameterized coordinates of one positive anchor. The classification loss L_{class} is a log loss over two classes. For regression loss L_{reg}, smooth L_1 is used so that $L_{reg}(t,t') = R(t-t')$ where R is the robust loss function.

In [47], the parametrizations of t is adopted by:

$$t_x = (x-x')/w', t_y = (y-y')/h', t_w = log(w/w'), t_h = log(h,h') \tag{5}$$

where x,y,w,h denote the box's center coordinates and its width and height. Variables x' is for the ground truth box respectively (likewise for y,w,h). Our new parametrizations is similar with former one:

$$t_x = \begin{cases} 0, & \text{if} int(x'_0 > x_0) + int(x'_3 > x_3) = 1 \\ \frac{max(x'_0 - x_0, x'_3 - x_3)}{w}, & \text{otherwise} \end{cases}$$

where x_0, x_3 is the first and last x value from left-up to right-bottom. The loss function is changed due to fit with the postive dicision method(Overlapping Ratio).

4.2.5 Model Evaluation

This task is actually a object detection task, a commonly used evaluation method specified in [55] is used.

Overlapping Ratios and Intersection of Union Firstly, the concept of *Overlapping Ratios*(**OR**) is proposed. It should be noticed that we **do not** adopt a commonly index , *Intersection of Union(IOU)* to define whether one proposal is positive or negative when do evaluation. IOU is only used to define positive or negative samples when training. IOU takes the set A of proposed object pixels and the set of true object pixels B and calculates:

$$IoU(A, B) = \frac{A \cap B}{A \cup B} \tag{6}$$

Commonly, $IoU > 0.5$ means that it was a hit, otherwise it was a fail. Also, considering the property of our cases, which are medical rumor detection. Therefore,$A \cap B$ is much important, another evaluation value is proposed by us: *Overlapping Ratios*(**OR**),which is calculated by:

$$OR(A, B) = \frac{A \cap B}{A} \tag{7}$$

Usually, higher IOU and OR stands for a more accurate prediction of proposal.

Mean Average Precision and False Positive Per Image For each class, one can calculate the

- **True Positive** TP(c): a proposal was made for class c and there actually was an object of class c
- **False Positive** FP(c): a proposal was made for class c, but there is no object of class c.
- **Average Precision** for class c: $\frac{TP(c)}{TP(c)+FP(c)}$
- **False Positive Per Image(FPPI)** is the average number of false positive samples per image.

Then, the **mAP** (mean average precision),which is used to do the final evaluation,is calculated by:

$$mAP = \frac{1}{|classes|} \sum_{c \in classes} \frac{\#TP(c)}{\#TP(c) + \#FP(c)} \tag{8}$$

Usually, higher mAP and lower FPPI indicates better performance of detection model.

Mean Sensitive *Sensitive* is defined as:

$$Sensitive = \frac{Number\ of\ succefully\ predicted\ truth\ box}{Number\ of\ truth\ box} \tag{9}$$

MeanSensitive is the average sensitive among all images' sensitive. According to the definition, it is clearly that mean sensitive, especially in medical field, is highly important: when the value of mean sensitive is approximately 1.0, it means that almost all breast mass are found by the model, which is pretty valuable.

4.3 User Interface

At last, we designed a user interface for doctors. It has a web portal, and can generate human-readable reports for users with little prior knowledge about the inner design of the system. The following sections will discuss the design of the user interface.

4.3.1 Report Generator

The report generator will get diagnosis result from our model, and then make a human readable report. It should display the confidence level on the report for users to further decide if more check should be performed. The report will consist of an image showing suspected tumor location and a short text suggestion.

4.3.2 Web Front end

The web front end will accept image file input from users and display the detection result. The user should be able to select an image from the hard drive and upload the file with ease. The web page will perform basic file validation before really uploading it to server. Since the image tends to be very large, and it takes sometime for server to process an image, a progress bar indicating which step the image is in will be helpful for users. A simple authentication is possible to be added, but may not be required if the server is in local network.

4.4 Workflow

To develop a more accurate model, we have the following development workflow cycle in figure 39. The cycle includes five elements: design, implement, train, validate and test.

After we implement a model, we will train it. Validation will be performed occasionally. If the validation result is not satisfying, we will cut off the training and attempt to find out the reason. After the train accuracy converges, we will do a thorough test of the model and compute some quantitative measurement to determine if our design works well. We will try to analyze the factors that affect the performance, and perform incremental modifications accordingly.

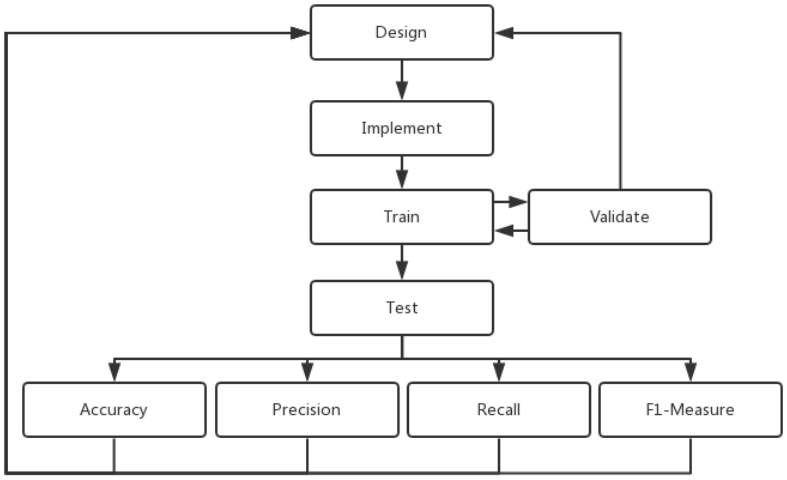


Figure 39: Development Workflow

5 Implementation

In this section, we will briefly demonstrate how we implement the aforementioned ideas about our diagnosis system. The main language we used to build the model is python, but we also uses bash scripts or javascript to provide utility functions.

5.1 Histopathological Image Classification

Applying and combining methodologies we mentioned above, we have successfully implemented a ResNet model, and trained it with preprocessed BreakHis dataset, which has also been introduced before. In this part, we will divide our implementation to several parts according to code logic and introduce details of each part one by one. Also, we will explain and show our assigned parameters in this section.

5.1.1 Data Loader and Preprocess

This part is about the implementation of loading data and preprocessing images to fit them into ResNet model.

Data Loader There are kinds of diseases such as mucinous carcinoma and adenosis in original dataset. Each disease is divided into two classes, benign and malignant, and has a file to record paths and image numbers of it.

```
SOB/mucinous_carcinoma/SOB_M_MC_14-18842/200X : 16
SOB/mucinous_carcinoma/SOB_M_MC_14-18842/400X : 9
SOB/mucinous_carcinoma/SOB_M_MC_14-18842/100X : 22
```

```
SOB/mucinous_carcinoma/SOB_M_MC_14-18842/40X : 15
SOB/mucinous_carcinoma/SOB_M_MC_14-13418DE/200X : 14
SOB/mucinous_carcinoma/SOB_M_MC_14-13418DE/400X : 11
SOB/mucinous_carcinoma/SOB_M_MC_14-13418DE/100X : 15
SOB/mucinous_carcinoma/SOB_M_MC_14-13418DE/40X : 15
SOB/mucinous_carcinoma/SOB_M_MC_14-18842D/200X : 16
```

To read all files recording the path of each disease, we use regex in python and store all image paths in a list *shuffled_walk*.

```
shuffled_walk = []
regex = re.compile(pattern)
for dirname, _, filenames in os.walk(data_set_dir):
 if regex.match(dirname):
  metainfo = regex.match(dirname).group(...)
  shuffled_walk.append(metainfo)
 print (shuffled_walk)
```

To divide data into train dataset and test dataset, and keep them unchanged on accuracy test of different model, we simply use remainder of *shuffled_walk* 's index to divide the data.

```
i1 = [ i for i in range(len(shuffled_walk)) if i%4 == 0]
i2 = [ i for i in range(len(shuffled_walk)) if i%4 != 0]
index = i1 + i2
tmp = shuffled_walk
shuffled_walk = []
for i in range(len(tmp)):
 shuffled_walk.append(tmp[index[i]])
```

Image Slicing After storing image paths, we need to do image segmentation to fit image with proper size into our DNN model. As mentioned above, we use three kinds of strategies to do image segmentation: sliding window crop, random crop and resizing, which have been introduced before.

```
def resizing(image,sub_slides):
 image_shape = np.shape(image)
 indexes = np.random.choice(image_shape[1] - image_shape[0], 50)
 for i in indexes:
  sub_slides.append(image[:,i:i+IMG_HEIGHT])
 return sub_slides

def sliding_window_crop(image, sub_slides):
 image_shape = np.shape(image)
 col_step = int(compute_slide_step)
```

```
 row_step = int(compute_slide_step)
 for col in range(0, image_shape[0]-IMG_WIDTH + 1,col_step):
  for row in range(0, image_shape[1]-IMG_HEIGHT + 1,row_step):
   sub_slides.append(np.array(crop))
 return sub_slides

def random_crop(image,sub_slides):
 image_shape = np.shape(image)
 x = np.random.choice(image_shape[0]-IMG_HEIGHT,100)
 y = np.random.choice(image_shape[1]-IMG_WIDTH,100)
 for i in range(100):
  sub_slides.append(np.array(crop))
 return sub_slides
```

Preprocess After slicing images into patches, we implemented different preprocess methods to test whether is suitable for histopathological image, which has been introduced before.

```
def whitening_image(image_np):
 for i in range(np.shape(image_np)[0]):
  mean = np.mean(image_np[i])
  std = np.max([np.std(image_np[i]), 1.0 /
                np.sqrt(IMG_HEIGHT*IMG_WIDTH*IMG_DEPTH)])
  image_np[i] = (image_np[i] - mean) / std
 return image_np

def subtract_gaussian_smooth_image_and_CLAHE(image_np):
 for i in range(np.shape(image_np)[0]):
  blur = cv2.GaussianBlur(...)
  clahe_input = cv2.cvtColor(image_np[i] - blur, cv2.COLOR_BGR2YUV)
  clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
  clahe_input[:, :, 0] = clahe.apply(clahe_input[:, :, 0])
  image_np[i] = cv2.cvtColor(clahe_input, cv2.COLOR_YUV2BGR)
 return image_np

def CLAHE_image(image_np):
 for i in range(np.shape(image_np)[0]):
  clahe_input = cv2.cvtColor(image_np[i], cv2.COLOR_BGR2YUV)
  clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
  clahe_input[:, :, 0] = clahe.apply(clahe_input[:, :, 0])
  image_np[i] = cv2.cvtColor(clahe_input, cv2.COLOR_YUV2BGR)
 return image_np

def past_pre(image_np):
 mean = np.mean(image_np,axis=0)
```

5.1.2 Model

Our model is not implemented in a single inference function, but we implement functions for different usage. The parameters and outputs of each function are explained and shown in Appendix.

5.1.3 Train and Validation

As usually used in DNN, a model is trained by firstly feeding it input and generates the output (prediction) for comparison with the label of input. This kind of comparison is done by calculating the loss. We used cross entropy to represent loss function [56].

```
def loss(self, logits, labels):
 labels = tf.cast(labels, tf.int64)
 cross_entropy = tf.nn.sparse_softmax_cross_entropy_with_logits(...)
 cross_entropy_mean = tf.reduce_mean(cross_entropy)
 return cross_entropy_mean
```

5.1.4 Hyper Parameters

In this section, we will briefly explain the parameters related to research results we used and its assigned value.

- learning rate: 0.001, initial leaning rate.
- learning rate decay factor: 0.5, how much to decay the learning rate each time.
- decay_step_0: 500, the first step to decay the learning rate.
- decay_step_1: 2000, the second step to decay the learning rate.
- weight decay: 0.0002, weight decay for L2 regularization.
- train batch size: 64
- dropout proportion: 0.5
- train steps: 3000
- regularizer: L2 regularizer, a process of introducing additional information to reduce overfitting.

5.2 Mammogram Tumor Detection

As stated above, we implemented a Mask-RCNN model with tensorflow, and trained it with the preprocessed DDSM dataset. Again, we will introduce our program architecture from different aspects.

5.2.1 Data Loader and Preprocess

Data Loader Each image in the DDSM dataset is about 10 MB large and the whole dataset is about 80GB large. Handling such amount of data at once is very likely to cause data corruption. Therefore this dataset is divided into 2GB "volumes". Each volume can be labeled by normal, benign or cancer.

```
normal_01 111 5.8 GB DBA 16bit 42 microns
normal_02 117 6.6 GB DBA 16bit 42 microns
normal_03 38 4.1 GB DBA 16bit 42 microns
normal_04 57 5.1 GB DBA 16bit 42 microns
normal_05 47 4.3 GB DBA 16bit 42 microns
normal_06 60 5.5 GB DBA 16bit 42 microns
normal_07 78 6.2 GB HOWTEK 12bit 43.5 microns
normal_08 27 2.8 GB HOWTEK 12bit 43.5 microns
```

File Format Conversion The original image file is in LJPEG format which is not common to modern tools. Before we feed these images into the model, we need to convert them. Each converted PNG file is about 30 MB, so we must perform the conversion batch by batch to avoid running out of temporary storage. This is generally done in a bash script. The procedure is generally

1. create a large enough temp storage
2. wget ftp://figment.csee.usf.edu/pub/DDSM/cases$1 and save it to temp storage
3. convert the downloaded LJPEG to PNG
4. save all metadata and the converted PNG
5. clear temp storage

Metadata Parser Then we need to parse the metadata file and convert the boundary chain code to a machine readable format such as a Python array. Firstly, we build up regular expressions to match certain metadata entries. Then we iterate through the metadata file and extract matchings, and store these matchings into a dictionary for later use.

```
def read_case(case_dir_path):
 with open(glob.glob(case_dir_path + '*.ics')[0]) as f:
  f.readline().strip()
  for line in f:
   extract information with regex_dict[key].match(line)
 for key in ['left_cc', 'left_mlo', 'right_cc', 'right_mlo']:
  ret[key] = cv2.imread(png file name)
  if ret['metainfo'][key.upper()][2] == 'OVERLAY':
```

```
    ret[key+'_overlay'] = read_overlay(overlay file name)
  return ret
```

Boundary Parser As for the boundary, we simply simulate the movement of the cursor as in the specification. We use a variable to represent the current position of the cursor.

```
def parse_boundary(str):
  chain_code = str.split('-')
  chain_code = list(map(int, chain_code[:-1]))
  x = [chain_code[0]]
  y = [chain_code[1]]
  chain_code = chain_code[2:]

  for movement in chain_code:
    if movement == 0:
      x.append(x[-1])
      y.append(y[-1] - 1)
    elif movement == 1:
      ...
    ...
    else:
      raise 'unrecognized chain code'

  return x, y
```

Preprocess and Cache After all these parsing steps, we apply the same preprocess methods as in section 5.1.1 to durther improve the quality of the image. To further shorten the train time, we also cached preprocessed images.

```
image = self.path_load(case['left_cc'])
image = self.CLAHE_image(image)
np.save(case_dir_path+'/'+'left_image.npy',image)
```

5.2.2 Model Construction

Overall Structure As stated before, our model is inspired by Mask RCNN introduced in 2017 [47]. The framework is quite similar to the Faster RCNN one, in the way that it replaced the ROI Pooling layer with the ROI Align layer, and adds a new parallel Fully Convolution Network to retrieve the mask information. The number of tasks has changed from two (classification + regression) to three(classification + regression + segmentation). The overall structure of our model is illustrated as follows:

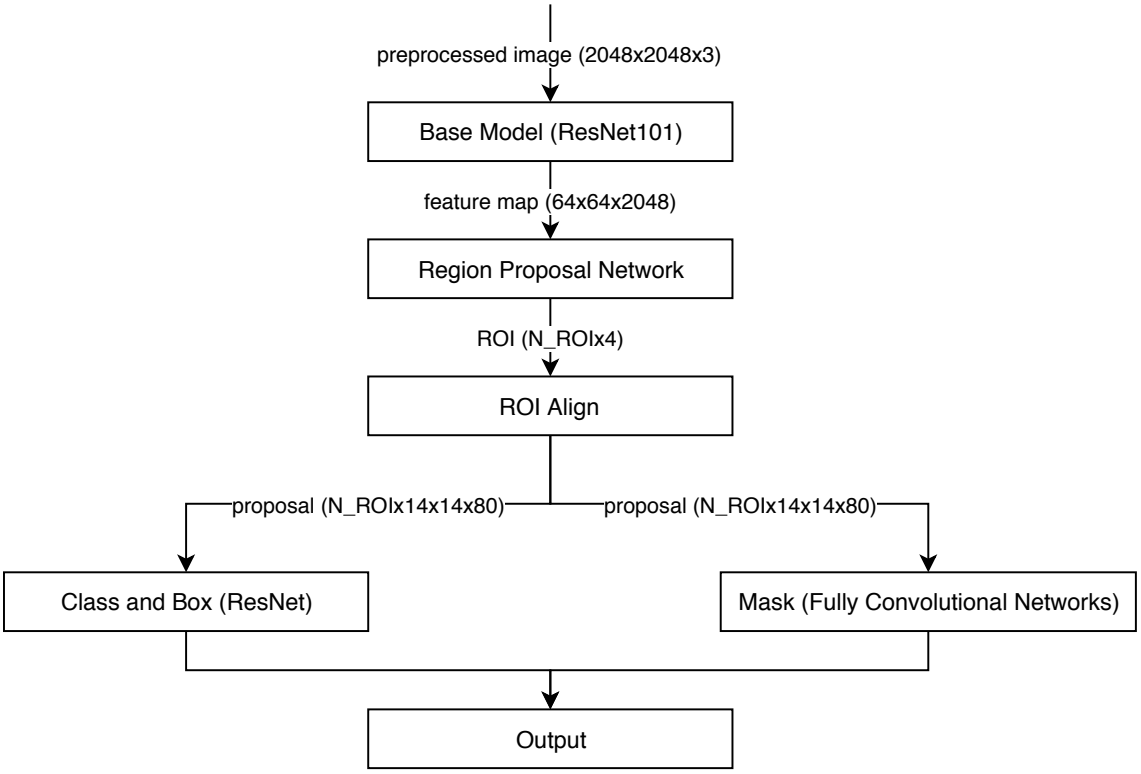


Figure 40: Structure Mammogram Tumor Detection Model

The Faster RCNN base structure is easy to train, and the added FCN is a small-scale network, so Mask RCNN will not be difficult to use. As shown in the figure, Mask RCNN adopts the two step method introduced by Faster RCNN, it find the regional proposals first, and secondly find the classification, box offset and binary mask.

Base Model In our model, we firstly feed preprocessed image into the base model. These images are normalized to size $2048 \times 2048 \times 3$ for consistency. We use is ResNet101 in our project as the base model. The detailed implementation logic is shown as follows.

```
def resnet(input_image):
 # 1
 KL.ZeroPadding2D((3, 3))
 KL.Conv2D(64, (7, 7), strides=(2, 2))
 BatchNorm()
 KL.Activation('relu')
 KL.MaxPooling2D((3, 3), strides=(2, 2), padding="same")
 # 2
 conv_block(3, [64, 64, 256], strides=(1, 1))
```

```
 identity_block(3, [64, 64, 256])
 identity_block(3, [64, 64, 256])
 # 3
 conv_block(3, [128, 128, 512])
 identity_block(3, [128, 128, 512])
 identity_block(3, [128, 128, 512])
 identity_block(3, [128, 128, 512])
 ...
 return ...
```

where the layer constructors have parameter signature of

```
def identity_block(kernel_size, filters, use_bias=True):
 ...

def conv_block(kernel_size, filters, strides=(2, 2),
               use_bias=True):
 ...
```

Regional Proposal Network We will omit the final output layers of ResNet, and export the feature map. A regional proposal network will take the feature map as input and find the proposals. The detailed implementation logic is shown as follows.

```
def region_proposal_network(inputs):
 # trim to top anchors by score
 pre_nms_limit = min(6000, anchors.shape[0])
 ix = tf.nn.top_k(scores, pre_nms_limit, sorted=True).indices
 scores = utils.batch_slice([scores, ix], ...)
 deltas = utils.batch_slice([scores, ix], ...)
 anchors = utils.batch_slice([scores, ix], ...)

 boxes = utils.batch_slice([anchors, deltas],
                           apply_box_deltas_graph, ...)
 boxes = utils.batch_slice(boxes, clip_boxes_graph, ...)
 normalized_boxes = boxes / np.array([[h, w, h, w]])
 proposals = utils.batch_slice([normalized_boxes, scores],
                               non_max_suppression, ...)

 return proposals
```

where the helper functions have the parameter signature of

```
def apply_box_deltas_graph(boxes, deltas):
 ...
```

```
def clip_boxes_graph(boxes, window):
 ...

def non_max_suppression(normalized_boxes, scores):
 ...
```

ROI Align Layer The proposals will then be feed into the ROI Align layer. It takes the proposal as input, and extract features from each regions. It is a more advanced version of the ROI Pooling layer. The detailed implementation logic is shown as follows.

```
def roi_align_layer(inputs):
 roi_level = log2(sqrt(h * w) / (224.0 / sqrt(image_area)))
 roi_level = min(5, max(2, 4 + round(roi_level) as int32))
 roi_level = squeeze(roi_level, 2)

 for i, level in [2...6]:
  box_indices = arg(roi_level == level)
  box_to_level.append(box_indices)
  box_indices = tf.stop_gradient(box_indices)
  pooled.append(tf.image.crop_and_resize(apply ROI pooling))

 sorting_tensor = box_to_level[:, 0] * 100000 + box_to_level[:, 1]
 pooled = tf.gather(pooled, top_k of sorting_tensor)
 return pooled
```

where the utility `log2` is simply

```
def log2(x):
 return tf.log(x) / tf.log(2.0)
```

Classification and Bounding Box The output of ROI Align layer is expected to have a shape of $N_{ROI} \times 14 \times 14 \times 80$. A final Resnet will be then used to find bounding box regression and classification. Since we have already implemented Resnet, we can simply use the same logic structure.

Mask The output of ROI Align layer is also used by another fully convolutional network to further find the binary mask of the object. In out project, we use Feature Pyramid Network to find the mask. It is specially optimized for small object detection, and hence is quite suitable for our project since there usually will not be many tumor cells. The detailed implementation logic is shown as follows.

```
def feature_pyramid_network(inputs):
 KL.TimeDistributed(KL.Conv2D(256, (3, 3), ...))
 KL.TimeDistributed(BatchNorm(axis=3))
 KL.Activation('relu')

 ...(repeat)

 KL.TimeDistributed(KL.Conv2DTranspose(256, (2, 2), strides=2, activation="re
 KL.TimeDistributed(KL.Conv2D(3, (1, 1), strides=1, activation="sigmoid"))
```

5.2.3 Train and Validation

During train, the model defines a multi-task loss function

$$L = L_{classification} + L_{box} + L_{mask}$$

to fully express loss in the whole process. $L_{classification}$ and L_{box} are the same as the ones in Faster RCNN. L_{mask} utilizes the new ROI mask branch output, and has changed from single pixel softmax to single pixel sigmoid.

The detailed implementation logic of each loss function is shown as follows.

```
def classification_loss(inputs):
 loss = tf.nn.sparse_softmax_cross_entropy_with_logits(
  labels=target_labels,
  logits=logits
 )
 loss = loss * active
 loss = tf.reduce_sum(loss) / tf.reduce_sum(active)
 return loss

def box_loss(inputs):
 loss = K.switch(size of target_bbox > 0, smooth_l1_loss(...))
 loss = K.mean(loss)
 return loss

def mask_loss(inputs):
 loss = K.switch(size of 0y_true > 0, K.binary_crossentropy(...))
 loss = K.mean(loss)
```

5.2.4 Hyper Parameter

The hyper parameter set in Mask RCNN has an identical definition of that of Faster RCNN. In our project, we tweaked some parameters to get a better result. The following list explains the definition of some critical parameters and the value we chose.

- `STEPS_PER_EPOCH` = 1000: Number of training steps per epoch
- `VALIDATION_STEPS` = 50: Number of validation steps to run at each epoch
- `BACKBONE_STRIDES` = [4, 8, 16, 32, 64]: The strides of each layer of the FPN Pyramid
- `RPN_ANCHOR_SCALES` = (32, 64, 128, 256, 512): Length of square anchor
- `RPN_ANCHOR_RATIOS` = [0.5, 1, 2]: Ratios of anchors at each cell
- `RPN_ANCHOR_STRIDE` = 1: Anchors are created for each cell in the feature map
- `RPN_NMS_THRESHOLD` = 0.7: Non-max suppression threshold
- `RPN_TRAIN_ANCHORS_PER_IMAGE` = 256: Number of anchors per image
- `POST_NMS_ROIS_TRAINING` = 2000: ROIs kept after non-maximum supression
- `ROI_POSITIVE_RATIO` = 0.33: Percent of positive ROIs used for training
- `POOL_SIZE` = 7: Pooled ROIs
- `MASK_POOL_SIZE` = 14
- `MASK_SHAPE` = [28, 28]
- `LEARNING_RATE` = 0.001: Learning rate
- `WEIGHT_DECAY` = 0.0001: Weight decay regularization

5.3 User Interface

In our design, the system will have a user interface that enables general users to fully understand the output of the model. Therefore, we implemented a report generator together with a web front end.

5.3.1 Report Generator

The report generator will take in the original file, the masks and confidence levels. Then it will apply the given mask to the image, display bounding boxes and add comments to each of the masks. Different kind of regions will be marked with different colors. The detailed implementation logic is shown as follows.

```
def generate_report(inputs):
 for each region:
  bounding_box = rectangle(...)
  figure.add_patch(bounding_box)

  label = class_names[...]
  comment = "{} - {:.3f}".format(label, score)
```

```
    figure.text(comment)

    image[:, :] = np.where(
      mask == 1,
      image[:, :] * color for this kind of region,
      image[:, :]
    )
    mask = polygon(...)
    figure.add_patch(bounding_box)
  return figure
```

5.3.2 Application Interface

We have a web portal for end users, so an application interface was developed for communication between browsers and the server. User input includes username and password for accessing our GPU cluster and the image file. The application interface should be robust to incorrect inputs, such as invalid credentials or invalid image file. Since the process will take some time, this application interface will not response immediately. Fortunately, a typical diagnosis will be done in two minutes, so a fully synchronized interface is not required. Hence, we implemented a simple HTTP POST interface.

The detailed implementation logic is shown as follows:

```
...
Connection: keep-alive
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary...

------WebKitFormBoundary...
Content-Disposition: form-data; name="username"

username
------WebKitFormBoundary...
Content-Disposition: form-data; name="password"

password
------WebKitFormBoundary...
Content-Disposition: form-data; name="file"; filename="mammography-sample.png"
Content-Type: image/png


------WebKitFormBoundary...--
```

5.3.3 Web Front end

To work with the aforementioned application interface, we have a simple web page for general users to input the required form fields. The portal will provide a preview of the image file to be uploaded before initiating a request. The detailed implementation logic is shown as follows:

```
const pngInput = <input
  type="file"
  name="file"
  accept="image/x-png"
  required="true"
  onchange={() => preview.src = URL.createObjectURL(...))}
/>;
```

Since the diagnosis will take sometime, there should be a notice to let the user know the time s/he needs to wait. We also added progress bars showing the current diagnosis step. The detailed implementation logic is shown as follows:

```
const reportProgress = <progress></progress>;
central.addEventListener('report-progress', ({ detail: { loaded } }) => {
 reportProgress.value = loaded;
});
```

Once the process is finished, the user can get the generated report of his/her submission. The webpage should be automatically scrolled if the report cannot be displayed in the current view. The detailed implementation logic is shown as follows:

```
const outcome = <img style="max-width: 100%;"></img>;
central.addEventListener('report-load', ({ detail }) => {
 outcome.src = URL.createObjectURL(detail);
});
outcome.onload = () => outcome.scrollIntoView();
```

6 Results and Analysis

In the previous part, we introduced our overall implementations in different aspects, data IO, model construction and evaluation. In this part, we will divide and discuss our final year project's results into three parts, histopathological image classification task, mammogram tumor detection task and breast cancer diagnosis-aided system.

6.1 Histopathological Image Classification Results

In this part, we will compare different methods and parameters together and compete with past paper results using the same dataset to see whether our model is optimized enough.

Following the experimental protocol proposed in [50], we used cross-validation method [51] to do evaluation, the dataset was split so that patients used to build the training set (75% patients) are not used for the testing set (25% patients) to guarantee that our model can generalize to those patients not in the dataset, the results presented in this work are the average of four trials with the selected results after converging and a suitable early stop.

Training protocol used here is the purely supervised type, the Stochastic Gradient Descent (SGD) method [52], with back-propagation to compute gradients was used to update the network's parameters. All fixed hyper-parameters of training are given in the Implementation section.

The ResNet model were trained on a NVIDIA Tesla K40m GPU [53] using the Tensorflow framework [42]. Training took about 5 hours for the 256×256 input size and 10 hours for the 512×512 with the hyper parameters specified in **??**, which is corresponding to a much more complex training set.

Because there are millions of parameters and hundreds of hyper-parameters, therefore which parameters need to be tuned among the test should be considered carefully. Through our study on both medical and deep learning field, we selected three major hyper-parameters (directions) to do our test: Preprocess method, model architecture and image segmentation method.

For each hyper-parameter, we tested kinds of values or situations based on our guess and motivation, so each block below will contain several sub-blocks to explain each guess and its corresponding results in detail.

6.1.1 Results of Different Image Preprocess Methods

Preprocess is one of the most important part in image classification, especially in histopathological image classification, which can be concluded in previous literature review of section 2. According to the previous section, we have introduced different

kinds of preprocess method and showed the code, in this part, we will test different preprocess method by keeping other parameters same.

Typically, the model architecture of all cases in this part is normal model architecture (left,figure 41) and all inputs are segmented by different functions with size 256×256. Figure 5.4 and 5.5 show the preprocess results of one given image to offer reader an intuitive feeling. Tables in the section 8, appendix, report the results of different preprocess methods in both batch level and image level in detail respectively, while table 7 is a rough comparison among different preprocess methods.

From the table 7, we can find that different preprocess method has a huge influence on the results, typically, CLAHE shows a best performance on the higher magnification, where it shows that it is able to achieve an accuracy of about 5% better than the results of raw input. However, CLAHE won't work when the magnification factor is $40\times$ while whiten operation can help model to overcome this problem.

6.1.2 Results of Different Model Architecture

Model architecture is also one of the features we selected to test the result and it is usually the most critical part in DNN. Previous section has introduced the basic structure of residual block, in this part, we will evaluate the results of slightly different model architectures, which are all based on residual blocks.

Tested architectures follow the form in Figure 41, the detail can be found in Table 8. These networks' inputs are segmented by Resizing and preprocessed by method CLAHE with size 256×256.

Normal Model Architecture Our normal model architecture is shown on Figure 41. The first layer is a convolution layer with 7×7 kernel size , followed by a 3×3 convolution and a 2×2 max pooling layer with stride 2, to reduce the input size of following residual network. Then we use a stack of NumOfSize$\times$NumOfBlocks residual blocks with 3×3 convolution on the feature maps of sizes 128,64,32,16,8 respectively. NumOfSize represents the number of sizes, and the value is 5 in this model. NumOfBlocks is one of the hyper-parameters we can set, the value is also 5 in this model. Table in the Appendix section 8 shows the results of normal model and the analysis of the result will be discussed later while we will focus on the comparison with this base model in this section.

First Convolution with Kernel Size 3×3 In our medical research, our current goal is to classify the tumor. However, there are four kinds of magnification factors in our dataset, which means that tumor in different images may have different sizes, for example, tumor in $40\times$ image is much smaller than $400\times$ image.

According to table in the Appendix section 8 we have mentioned last part, the model gained a nice accuracy on both $400\times$ and $200\times$ images, but is not so exciting

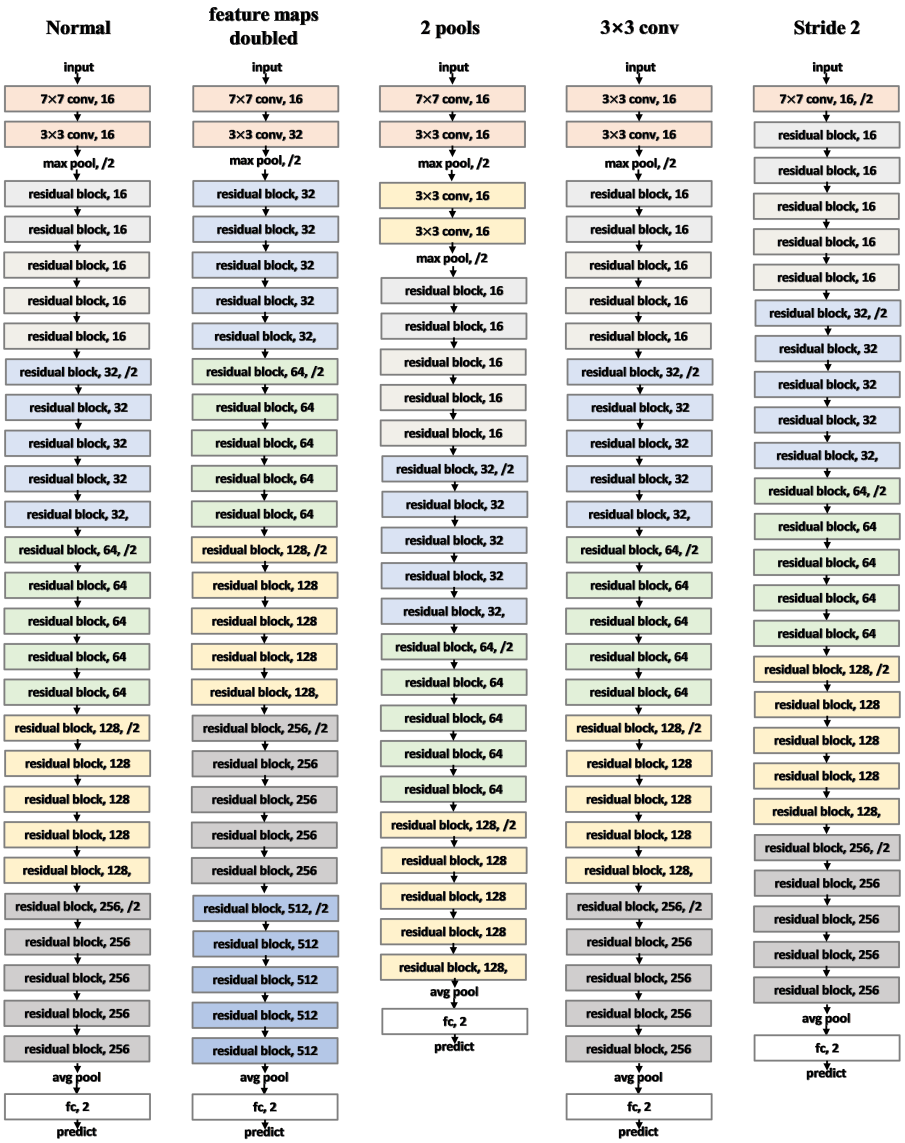


Figure 41: Five network architectures adopted in experiments. Each box indicates one layer and different Residual blocks/layers and different feature map sizes are labeled with different background colors. The '/2' symbol increases dimensions and implements a down-sampling operation. **Left**: The standard normal architecture we employed in most cases as a baseline model. **Middle-Left**: The number of feature maps is doubled in all layers. **Middle**:The head of the model increases one pooling layer to do extra down-sampling. **Middle-Right**: The kernel size of convolution layers in the head is changed to 3×3 instead of 7×7. **Right**: The max pooling layer after convolution layers is substituted with changing the stride of convolution layer to 2 to do down-sampling.

magnification	preprocess method	image level			batch level	
		best aggregation method	accuracy (%)	F1 score (%)	accuracy (%)	AUC (%)
40×	raw	vote	81.95	86.85	80.03	80.68
	Gaussian, CLAHE	exist	68.42	80.37	65.09	68.89
	CLAHE, whiten	vote	**87.03**	**91.05**	**86.17**	**82.80**
	CLAHE	exist	81.20	89.59	82.93	82.19
	whiten, CLAHE	average	86.28	90.48	85.84	80.97
	whiten	vote	86.64	90.96	85.82	78.65
	demean	vote/average	79.51	85.64	79.42	82.41
100×	raw	exist3	78.64	85.58	79.09	79.42
	Gaussian, CLAHE	vote	69.12	80.41	69.28	70.39
	CLAHE, whiten	exist3	81.69	87.50	81.44	79.42
	CLAHE	exist3	**84.74**	**89.39**	**83.37**	76.98
	whiten, CLAHE	vote	82.23	87.76	81.42	**82.23**
	whiten	vote	83.12	88.25	82.32	82.19
	demean	exist3	79.89	86.10	79.01	81.54
200×	raw	vote/average	**88.87**	92.13	87.74	**88.36**
	Gaussian, CLAHE	average	77.19	83.83	75.90	81.52
	CLAHE, whiten	vote	85.77	90.15	84.96	85.41
	CLAHE	vote	**88.87**	**92.87**	**88.33**	85.02
	whiten, CLAHE	vote/average	85.22	89.79	84.65	87.63
	whiten	average	85.22	89.73	84.31	86.22
	demean	vote/average	84.67	89.45	83.91	82.62
400×	raw	exist	82.99	88.22	81.09	**85.73**
	Gaussian, CLAHE	vote	80.37	85.64	78.26	82.15
	CLAHE, whiten	exist3	80.56	86.73	80.15	82.05
	CLAHE	exist3	**86.73**	**90.62**	**86.15**	82.61
	whiten, CLAHE	vote	82.80	87.99	81.75	83.38
	whiten	vote/average	81.31	87.01	80.49	83.11
	demean	exist	84.67	89.24	82.96	82.97

Table 7: Overall results using different preprocess methods

Layer name	Output size	Normal	3×3 conv	Stride conv	Block number changed	Feature map doubled	2 pools
conv00_x	256×256	7×7, 16	3×3, 16	7×7, 16, stride 2	7×7, 16		
	256×256	3×3, 16		NA	3×3, 16	3×3, 32	3×3, 16
	128×128	2×2 max pool, stride 2			2×2 max pool, stride 2		
conv01_x	128×128	NA					3×3, 16
	128×128						3×3, 16
	64×64						2×2 max pool, stride 2
conv1_x	128×128	$\begin{pmatrix} 3\times3 & 16 \\ 3\times3 & 16 \end{pmatrix}\times5$			$\begin{pmatrix} 3\times3 & 16 \\ 3\times3 & 16 \end{pmatrix}\times4$	$\begin{pmatrix} 3\times3 & 32 \\ 3\times3 & 32 \end{pmatrix}\times5$	NA
conv2_x	64×64	$\begin{pmatrix} 3\times3 & 32 \\ 3\times3 & 32 \end{pmatrix}\times5$			$\begin{pmatrix} 3\times3 & 32 \\ 3\times3 & 32 \end{pmatrix}\times4$	$\begin{pmatrix} 3\times3 & 64 \\ 3\times3 & 64 \end{pmatrix}\times5$	$\begin{pmatrix} 3\times3 & 16 \\ 3\times3 & 16 \end{pmatrix}\times5$
conv3_x	32×32	$\begin{pmatrix} 3\times3 & 64 \\ 3\times3 & 64 \end{pmatrix}\times5$			$\begin{pmatrix} 3\times3 & 64 \\ 3\times3 & 64 \end{pmatrix}\times5$	$\begin{pmatrix} 3\times3 & 128 \\ 3\times3 & 128 \end{pmatrix}\times5$	$\begin{pmatrix} 3\times3 & 32 \\ 3\times3 & 32 \end{pmatrix}\times5$
conv4_x	16×16	$\begin{pmatrix} 3\times3 & 128 \\ 3\times3 & 128 \end{pmatrix}\times5$			$\begin{pmatrix} 3\times3 & 128 \\ 3\times3 & 128 \end{pmatrix}\times7$	$\begin{pmatrix} 3\times3 & 256 \\ 3\times3 & 256 \end{pmatrix}\times5$	$\begin{pmatrix} 3\times3 & 64 \\ 3\times3 & 64 \end{pmatrix}\times5$
conv5_x	8×8	$\begin{pmatrix} 3\times3 & 256 \\ 3\times3 & 256 \end{pmatrix}\times5$			$\begin{pmatrix} 3\times3 & 256 \\ 3\times3 & 256 \end{pmatrix}\times5$	$\begin{pmatrix} 3\times3 & 512 \\ 3\times3 & 512 \end{pmatrix}\times5$	$\begin{pmatrix} 3\times3 & 128 \\ 3\times3 & 128 \end{pmatrix}\times5$
	2 × 1	average pool, fc, softmax					

Table 8: Detailed architectures of evaluated models, building blocks are shown in brackets with the numbers of blocks stacked. Down-sampling is performed by conv1_1, conv2_1, conv3_1, conv4_1, and conv5_1 with a stride of 2.

on smaller magnification factors such as 40×, we doubted that it was due to the small tumor so that 7×7 first convolution is too big to catch the local feature of small tumor. Therefore, we tried to reduce the 7×7 to 3×3 of the first convolution's kernel, and detected its performance on small magnification factor.

Figure 41, middle-right shows the basic structure of this model and Table 16 indicates the performance of this model, surprisingly, we found that 3×3 convolution model gains a better batch level accuracy on bigger magnification factor and a better AUC on all factors. We think the reason may be that local feature of tumor is always smaller than 7×7, no matter what the magnification factor is. Therefore, smaller first convolution layer's kernel size can gain a better result in tumor classification tasks, which is different from the ResNet used in normal image classification problems [40].

First Convolution with Stride 2 We perform down-sampling by pool layers in normal model, in this model, we changed the stride of the first convolution from 1 to 2 and discarded the pool layer before residual blocks, which is similar with the model architecture in [40], this work is motivated to evaluate the influence of convolution layer and pooling layer we added before residual blocks.

The model structure is briefly shown in Fig 41, right, and the detailed results is in Table 17. The main discovery from Table 17 is that no matter what the magnification is, the best aggregation method is always vote or average, which are the most valid methods. At the same time, stride 2 shows a almost wonderful results comparing with normal model, which means that stride is usually better than pool layer when doing down-sampling.

Model with Feature Map Doubled This model is easy to understand, which is simply double the feature maps comparing with the normal model structure, the idea is inspired by [57], which claims that wider ResNet is helpful for image classification. Therefore, we want to know whether it works on histopathological images or not.

The model structure diagram is shown in Figure 41, middle-lfet, and the detailed result can be found in Table 18, we can see that doubled feature maps can in deed increase the study capacity of model because almost all magnification's AUC increase, which is like 3×3 convolution model. Typically, sum becomes a pretty good aggregation method in this model.

Model with Two Pooling Layers Before Resnet Our model faces a serious over-fitting problem, which will be introduced in detail in the following section 6.1.4. This model, with 2 pool layers before ResNet, is a try to solve the overfitting problem because we doubt that the study capacity of ResNet is so large that the net remembers all special features of train dataset, which results in over-fitting. Therefore, we want to apply more naïve convolution layers, which has a smaller study

magnification	Image level					Batch level			
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix		
40×	sum	81.58	87.85	**84.97**	86.39	82.94	AUC(%)	81.81	
	vote	**84.02**	93.79	84.05	**88.65**				predict
	average	**84.02**	93.79	84.05	**88.65**				malignant / benign
	exist	81.77	**99.15**	78.88	87.86		actual	malignant	16469 / 1231
	exist3	82.71	98.02	80.32	88.30			benign	3306 / 5594
100×	sum	81.33	89.56	**83.16**	86.24	83.99	AUC(%)	82.38	
	vote	**84.92**	97.80	82.41	**89.45**				predict
	average	**84.92**	97.80	82.41	**89.45**				malignant / benign
	exist	83.84	**99.45**	80.44	88.94		actual	malignant	17626 / 603
	exist3	84.20	**99.45**	80.80	89.16			benign	3863 / 5808
200×	sum	87.59	96.71	86.31	91.21	88.08	AUC(%)	87.02	
	vote	**88.87**	98.90	**86.36**	**92.21**				predict
	average	**88.87**	98.90	**86.36**	**92.21**				malignant / benign
	exist	85.40	**100.0**	82.02	90.12		actual	malignant	17975 / 275
	exist3	86.86	99.73	83.68	91.00			benign	2992 / 6158
400×	sum	86.92	97.67	**84.42**	90.57	87.14	AUC(%)	84.77	
	vote	87.66	99.42	84.24	91.20				predict
	average	**87.85**	99.71	84.28	**91.34**				malignant / benign
	exist	85.61	**100.0**	81.71	89.93		actual	malignant	17064 / 163
	exist3	86.36	**100.0**	82.49	90.41			benign	3284 / 6289

Table 9: The results of model with dropout

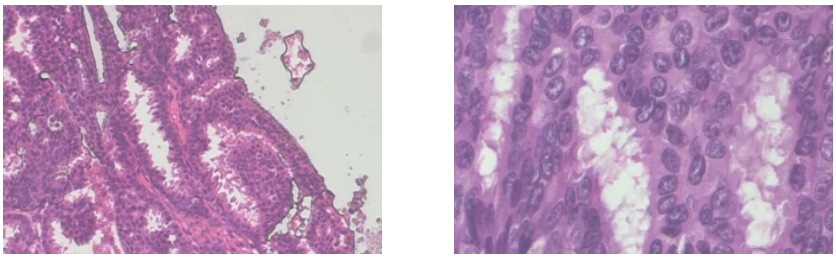
capacity than ResNet, and less Residual blocks.

The model diagram is shown in Figure 41, middle, and the results can be found in Table 19. We can see that almost all results have no difference from normal architecture, which means that Residual blocks are not the reason for overfitting.

Normal Model with Dropout Dropout is well known to be an effective way to solve over-fitting [58], therefore we also tried to apply dropout in our network. The model architecture is same as the normal model (Fig 42, left) , and we set up an additional dropout before the final fc layer with dropout rate 0.5. The result of normal model with dropout is in Table 9. The accuracy has a little improve comparing with the result of normal model, but we can still regard it as an effective method since almost all results are changing with a nice direction.

Overall Comparison among Different Model Structures This part compares the results of different model architectures and the comparison is shown in Table 10.

From the results shown in table 10, we can find that there are no huge differences comparing with the input pre-process because we adopted ResNet as our fundamen-



(a) 100×,the tumor is small (b) 400×,the tumor is obvious

Figure 42: One example of 100× and 400× image

tal, which has a huge learning ability to instances features. no matter what little changes we implemented in the model architecture while preprocess is changing the sources: the features of the instances.

However, there are still some rules that can be found in the results, among all models, we can conclude that dropout and feature maps doubled are helpful for classification no matter what the magnification is, and stride 2 has a huge improvement on dataset of magnification 40× and 100×. However, model with 2 pools, the contrast of feature maps doubled model, which reduces the complexity of model, do not get a performance boost. In comparison, we conclude that more complex structure can still make learn the features better.

6.1.3 Results of Different Segmentation Methods

Different segmentation methods will produce inputs of different size, which will absolutely be fed into different model architectures. Last section introduces the results of different model architectures, and the difference between these two parts is that the former one focused on the model architecture difference and kept input size same, while this section will mainly discussion the influence of different image segmentation methods.

When we study on the dataset, we found that tumor in low magnification images, such as 100×, was too small to be obviously found (Figure 42). We guess that the input size should be smaller when magnification is smaller to catch the local feature of tumor. To verify our guess, we implement and test our methods. Figure 43 shows the structure we used for different segmentation size and Table 11 is the overall comparison among different segmentation methods.

According to Table 11 , 64×64 and 128×128 ranks top 2 on both 40× and 100× test dataset while 256×256 and 512×512 dominates the results of 200× and 400× dataset, which is keeping with our guess. Also, we found that random segmentation method, which increases the variance of train dataset, is a little better than sliding window method.

magnification	Image level					Batch level			
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix		
40×	sum	81.58	87.85	**84.97**	86.39	82.94	AUC(%)	81.81	
	vote	**84.02**	93.79	84.05	**88.65**				predict
	average	**84.02**	93.79	84.05	**88.65**				malignant benign
	exist	81.77	**99.15**	78.88	87.86		actual	malignant	16469 1231
	exist3	82.71	98.02	80.32	88.30			benign	3306 5594
100×	sum	81.33	89.56	**83.16**	86.24	83.99	AUC(%)	82.38	
	vote	**84.92**	97.80	82.41	**89.45**				predict
	average	**84.92**	97.80	82.41	**89.45**				malignant benign
	exist	83.84	**99.45**	80.44	88.94		actual	malignant	17626 603
	exist3	84.20	**99.45**	80.80	89.16			benign	3863 5808
200×	sum	87.59	96.71	86.31	91.21	88.08	AUC(%)	87.02	
	vote	**88.87**	98.90	**86.36**	**92.21**				predict
	average	**88.87**	98.90	**86.36**	**92.21**				malignant benign
	exist	85.40	**100.0**	82.02	90.12		actual	malignant	17975 275
	exist3	86.86	99.73	83.68	91.00			benign	2992 6158
400×	sum	86.92	97.67	**84.42**	90.57	87.14	AUC(%)	84.77	
	vote	87.66	99.42	84.24	91.20				predict
	average	**87.85**	99.71	84.28	**91.34**				malignant benign
	exist	85.61	**100.0**	81.71	89.93		actual	malignant	17064 163
	exist3	86.36	**100.0**	82.49	90.41			benign	3284 6289

Table 10: Overall results using slightly different model methods, the visualization version of model structure is in Figure 41

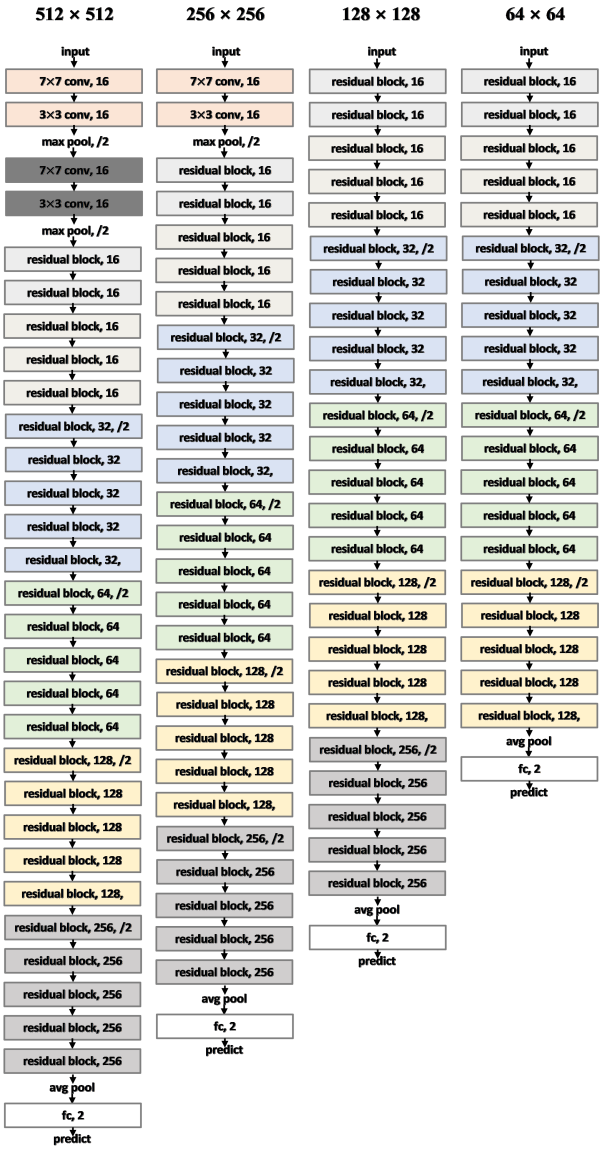


Figure 43: Structure for Different Segmentation Methods. **From left to right**: The input size is $512 \times 512, 256 \times 256, 128 \times 128, 64 \times 64$ respectively

magnification	Segmentation method	Input size	image level			batch level	
			best aggregation method	accuracy (%)	F1 score (%)	accuracy (%)	AUC (%)
40×	random	512×512	NA	NA	NA	NA	NA
	Random	256×256	exist	81.20	89.59	82.93	82.19
	random	64×64	vote	85.71	90.13	83.20	78.90
	sliding window	128×128	average	**87.41**	**91.15**	**84.56**	**82.69**
	sliding window	64×64	sum	85.34	89.54	83.88	82.12
100×	random	512×512	NA	NA	NA	NA	NA
	Random	256×256	exist3	84.74	89.39	83.37	76.98
	random	64×64	vote/average	**87.61**	**91.34**	**84.80**	81.61
	sliding window	128×128	vote/average	86.36	90.45	83.66	**86.65**
	sliding window	64×64	vote	86.89	90.86	84.40	83.86
200×	random	512×512	NA	NA	NA	NA	NA
	Random	256×256	vote	88.87	**92.87**	**88.33**	85.02
	random	64×64	sum	88.14	91.68	86.27	86.05
	sliding window	128×128	vote/average	**89.05**	92.41	87.10	86.42
	sliding window	64×64	average	88.50	92.06	86.84	**89.38**
400×	random	512×512	vote/average	**87.10**	90.71	**86.56**	85.26
	Random	256×256	exist3	86.73	90.62	86.15	82.61
	random	64×64	average	**87.10**	**90.76**	84.22	85.00
	sliding window	128×128	vote	86.91	90.72	84.37	85.78
	sliding window	64×64	vote/average	86.73	90.55	82.89	**86.81**

Table 11: Overall results using different image segmentation methods, segmentation method has been introduced in section TODO: and 512×512 input size is too large to run correctly in $40\times, 100\times$ and $200\times$ magnification factors.

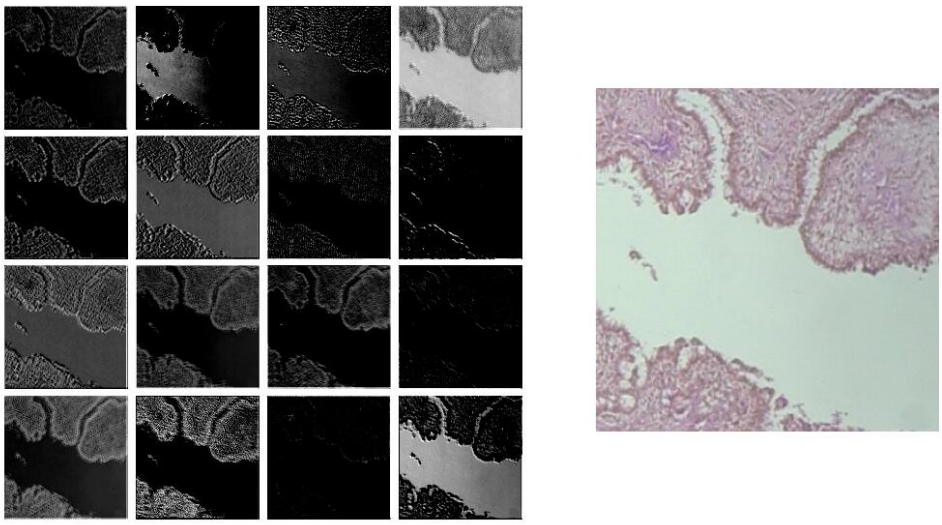


Figure 44: feature maps learned by first convolution layer. **Right**: the raw data,**left**: 16 feature maps the model learned

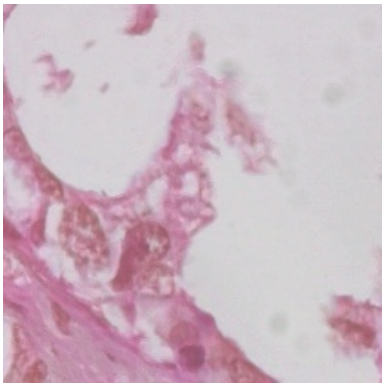
6.1.4 Analysis

In this section, unlike previous sections, which focus on results, specially, accuracy of our work, we will discuss some more advanced analysis on our model such as feature map analysis.

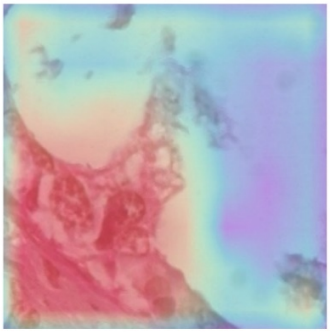
Feature maps One of the advantages using DNN are that we needn't design a feature extractor by a medical expert, but instead the model will learn it by itself. Figure 44 displays the 16 feature maps learned on the first convolution layer of our model. We can see that first convolution actually learned a edge detection rule by itself.

Location prediction From the idea of [59], we are able to visualize the location prediction of our model. We use the filter of last layer (shape 256×2) and the output of penultimate layer (shape 8×8×256) and implement a tensor-multiplication, after getting two feature maps with size 8×8, we resize the 8×8 image to input size, which is 256×256. Finally, we can use the resized image to visualize the local prediction to input of our model. Figure 45 shows an example of this kind of analysis, we can see that our work have a potential to implement a rough model for localization.

Advanced Results Analysis According to former experiments we have done, we can get a solid conclusion that datasets with different magnification factors need different hyper-parameters considering features of tumor. Typically, in this part, we implemented an "best" model combing former conclusions we got. We adapt



(a) raw input



(b) resized 256 × 256 prediction, red means more likely, blue means less likely.

Figure 45: One example of localization prediction.

the model architecture of Stride 2 (Fig 41, right), and add a dropout layer before the final fc layer. And CLAHE (section TODO:) is used to preprocess the data when magnification factor is not 40×, otherwise the preprocess method is CLAHE + whiten (TODO:). Table 12 shows the detailed results of "best" model and Fig 46 indicates one example of its ROC curve.

We can obtain some general information about general result, aggregation methods and AUC value from these results above (comparison of different methods has been discussed above):

- Our model achieves really high precision on image level, which is very practical because almost all malignant patients can be predicted as malignant.
- Five aggregation methods we apply above have slightly different influence on results of image level, in summary, vote/average shows a better performance.
- Lower magnification results have a lower AUC value, which means that more batches are labeled with not solid predictions. (Prediction of probabilities are closer to [0.5,0.5]). Therefore, we can conclude that lower magnification images have less information for learning.

6.1.5 Comparison with Previous Works

Table 13 shows the overall comparison between our results and past paper's using same dataset. Compared with accuracy and F1 score, which we defined earlier, our methods out-performs pervious work in [50], [60], [61], [62] at both patient and image level generally. Our work is better than other research using same dataset in almost all of cases, only in the 40× zoom level our results are a little worse than previous best work. In the remaining cases, the accuracy and F1 score achieved at

magnification	Image level					Batch level			
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix		
40×	sum	**88.72**	97.74	**86.93**	**92.02**	86.80	**AUC(%)**	82.82	
	vote	87.41	99.44	84.41	91.30				predict
	average	87.78	99.44	84.82	91.55				malignant benign
	exist	81.58	**100.0**	78.32	87.84		actual	malignant	17522 178
	exist3	83.08	**100.0**	79.73	88.72			benign	3334 5566
100×	sum	84.92	94.78	**84.15**	89.15	85.22	**AUC(%)**	82.35	
	vote	**85.46**	97.90	82.38	**89.89**				predict
	average	**85.46**	97.90	82.38	**89.89**				malignant benign
	exist	82.94	**100.0**	79.30	88.46		actual	malignant	17959 273
	exist3	84.38	**100.0**	80.71	89.33			benign	3850 5818
200×	sum	88.50	98.08	**86.47**	91.91	88.50	**AUC(%)**	89.85	
	vote	**89.05**	99.73	86.05	**92.39**				predict
	average	88.87	99.45	86.02	92.25				malignant benign
	exist	86.31	**100.0**	82.95	90.68		actual	malignant	18043 207
	exist3	87.77	**100.0**	84.49	91.59			benign	2945 6205
400×	sum	**86.73**	93.60	**86.79**	90.07	90.43	**AUC(%)**	89.94	
	vote	86.17	96.51	84.26	89.97				predict
	average	86.35	96.80	84.30	**90.12**				malignant benign
	exist	85.61	**99.13**	82.17	89.86		actual	malignant	16494 739
	exist3	86.17	98.55	83.09	90.16			benign	2937 6630

Table 12: The results of "best" model whose parameters are selected manually to get good results

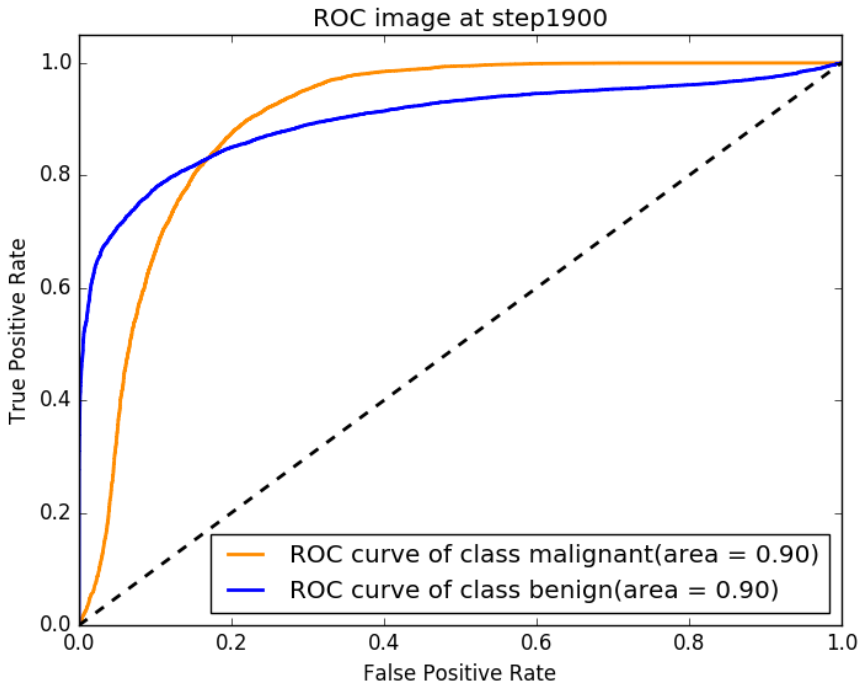


Figure 46: one example of ROC in our model results

least 0.5% better, and the difference can be as large as 5% in most cases. Which means that our method is much better than pervious methods.

One guess for the reason of low accuracy at 40× zoom level, may be that images in low magnification factors, such as 40× and 100×, has a fewer information and features for model to catch and learn, this is what we conclude in last section. However, the advantage of our applied model, learn capacity, cannot make contribution to the result, which makes the results similar at 40× and 100× zoom level.

6.1.6 Limitation and Difficulties

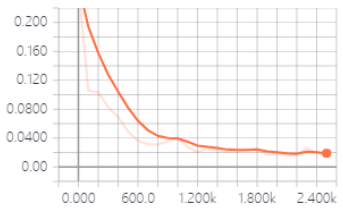
Despite the result we get as mentioned, we are also facing some limitation and difficulties. The following section will describe the problems and our proposed solutions.

Overfitting We faced serious overfitting problem since we adopted ResNet. As we can see in Figure 47, the train accuracy can be easily up to 99% but the test accuracy is not as good as we expected.

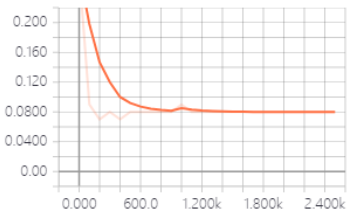
We have tried different technical to solve the problem, early stop, L2 regularization and dropout, all of them did not make a huge improvement but early stop can get an obvious increase, which can increase 2 to 3 percentage. We thought the reason may be the poor dataset, the dataset we used contain only 82 patients although

magnification	Approach	Patient level	Image level	
		accuracy (%)	accuracy (%)	F1 score (%)
40×	[58]	83.00	NA	
	[40]	83.80	82.80	87.80
	[47]	**88.60**	**89.60**	**92.90**
	[56]	84.00	84.60	88.00
	This work	88.26	88.72	92.02
100×	[58]	83.10	NA	
	[40]	82.10	80.7	86.10
	[47]	84.50	85.00	88.90
	[56]	83.90	84.80	88.80
	This work	**88.17**	**85.46**	**89.89**
200×	[58]	84.60	NA	
	[40]	85.10	84.20	88.50
	[47]	85.30	84.00	88.70
	[56]	86.30	84.20	88.70
	This work	**92.27**	**89.05**	**92.39**
400×	[58]	82.10	NA	
	[40]	82.30	81.20	**86.30**
	[47]	81.70	80.80	85.90
	[56]	82.10	81.60	86.70
	This work	**90.34**	**86.73**	**90.12**

Table 13: Accuracy and F1 score compared with those presented in [62], [50], [60] and [61]



(a) train error, which is close to 0 gradually.



(b) validation error, which maintains at 0.08 level.

Figure 47: Train and validation accuracy comparison.

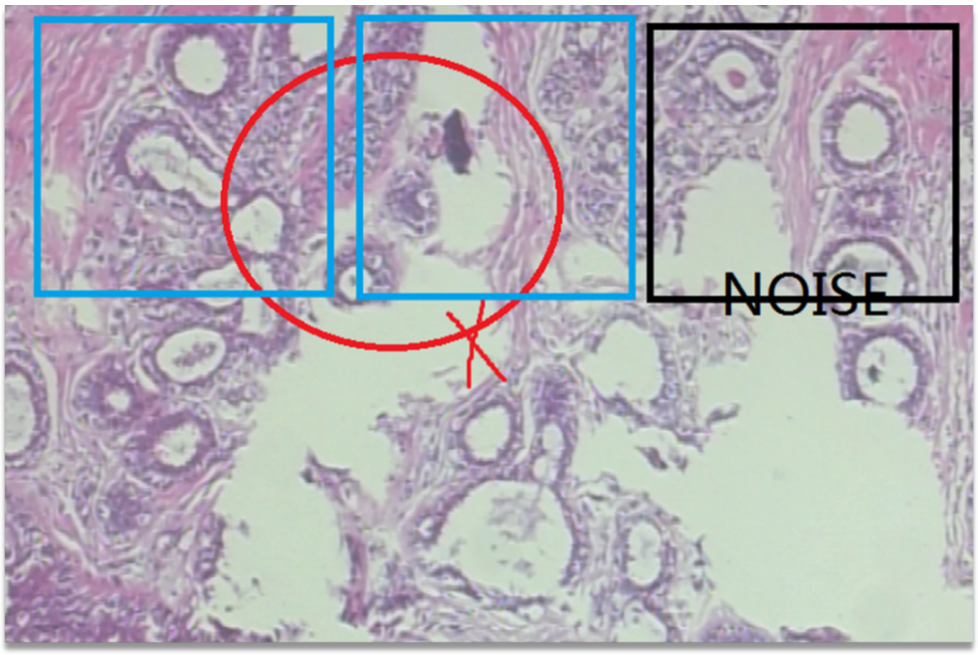


Figure 48: If red circle indicates a malignant tumor, then blue rectangle can be labeled as malignant correctly while black rectangle will become noise because there is no malignant tumor in it.

there are thousands of images. We thought overfitting may also be the reason why past paper did not get a good-looking accuracy as well.

Out of Memory Another difficulty we are facing now is the well-known problem, OOM. ResNet consumed plenty of GPU memory due to the deep layers. Bigger input size will consume bigger memory and previous work of ResNet only fit an input with size 64×64 or 32×32.

But current input size our model adopts is 256×256, because malignant images can contain normal cells. If the image is divided into small patches such as 32×32, it is not guaranteed that tumor appears in all patches. Malignant patches without tumor become noise during training, and confused the network (Figure 48). For higher magnification and bigger crop size, this problem is less severe, as tumor cells will be larger and hence less likely to be missed.

This is, therefore, the reason we build a traditional CNN above ResNet, we need a pool layer to implement down-sampling, which reduces the input size of ResNet to reduce memory allocation.

6.2 Mammogram Mass Detection Results

We evaluate our method using DDSM [29] dataset, the introduction of evaluation method and dataset we use to train and evaluate is in Section 4.2. We train a ResNet-101 model that shares features among RPN, fully connected layers and mask generation layers. The model runs at 3.35s per image on an Nvidia Geforce Titan X GPU (including two parts: preprocess and inference). Training with ResNet-101 on DDSM dataset takes 8 hours to achieve 12k steps using one Nvidia Geforce Titan X GPU. In this section, we firstly discuss the fundamental experimental results including the comparison of results of previous papers [32] [33] [30]. Secondly, some further discussions are proposed based on the results. Finally, we conclude our results of mammogram mass detection task.

6.2.1 Experimental Results

In this section, we simply show the experimental results using three kinds of technics and compare our results with previous works. Firstly kinds of experiments we perform are explained:

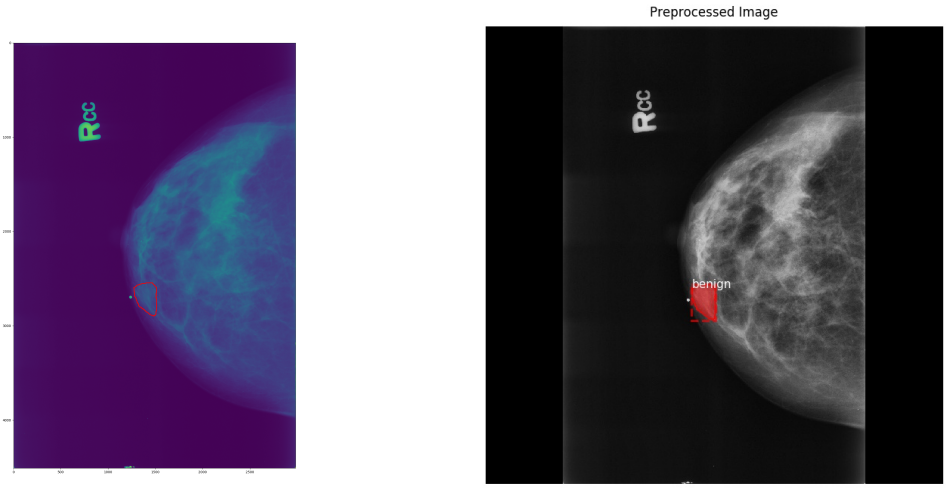
Original Results Original baseline result is obtained using same hyper parameters with [47] such as IOU threshold and anchor ratio, which is applied in common object detection such as COCO dataset [63].

New Preprocess The only difference between this kind of experiment setting and the original setting above is that we use a preprocess method, CLAHE, which we also adopt when do histopathological image classification task and introduced in Section 4.1.2 while original results do not adopt any image preprocessing method. Image 49 shows an example of mammogram after such preprocess method.

New Positive Decision Method In MaskRCNN [47], they treated an proposals as positive samples if its **Intersection of Union** is bigger than one threshold, in our experiment, we try to define proposals as positive if its **Overlapping Ratio** is bigger than one threshold(Both terms are introduced in Section 4.2.5). The reason for this change is the new evaluation method in mammogram, which is based on OR, instead of IOU.

New Loss Function We replaced the loss function of bounding box regression, the detail of the new loss function is introduced in Section 4.2.5.

We compare our method to previous works in Table 14. Our model benefits a lot from preprocess procedure, which makes our model highly competitive compared with other works. After considering preprocess, our model outperforms other works using RCNN [32] and Genetic System [33] both in mean sensitive and FPPI introduced in Section 4.2.5. Although our mean sensitive is lower than [30], we achieved a best



(a) Original mammogram (b) Mammogram proprocessed by CLAHE

Figure 49: Comparison between original mammogram and preprocessed mammogram.

FPPI among all related works, which means that the users (mostly doctors) won't get confused by the result of our system easily. The most impressive achievement of our work, compared with other works, is that we achieve 5× speedup compared with previous quickest work and outperform it, at the same time, we achieve a 150× speedup compared with most accurate work. When new positive decision method is performed, the mean sensitive is decreased while both mean AP and FPPI are moving in the right direction. More impressive result is obtained by considering both new positive decision method and loss function method, which achieves a nearly 90% mean AP, which is highly practical, which means 90% masks our model reported are right.

6.2.2 Results Analysis and Further Discussion

Besides performance results we discussed earlier, we also propose some trials and analysis. In this section, we will discuss and analysis our results in different views and topics.

Confidence Threshold in Test A confidence threshold T means that all predicted masks with confidence less than T are dropped, it is obvious that higher T causes less masks, which means lower mean sensitive and FPPI. Figure 50 illustrates the effect of threshold. When the threshold T is below 0.4, all values are almost not changed, this indicates that almost all predicted masks (actually generated by RPN) have a confidence larger than 0.3. At the same time, FPPI has a upper bound-

Method	Mean AP(%)	Mean Sensitive(%)	FPPI	Inference Time(s)
Our Work-O	45.1	57.9	2.37	1.32
Our Work-O+P	60.3	72.6	3.14	3.31
Our Work-O+P+E	63.0	65.3	0.72	3.26
Our Work-O+P+E+L	89.2	45.6	1.11	3.14
Our Work-O+P+L	67.6	62.5	1.4	3.35
Random Forest [30]	-	77	3.93	472
RCNN [32]	-	70	4	20
Genetic System [33]	-	70	8	108

Table 14: Comparison among kinds of technics we adopt and previous works. O: original results. P: preprocessed results.

ing with close to 1.2, which means that the RPN is too strict to generate enough proposals when we only consider preprocessing.

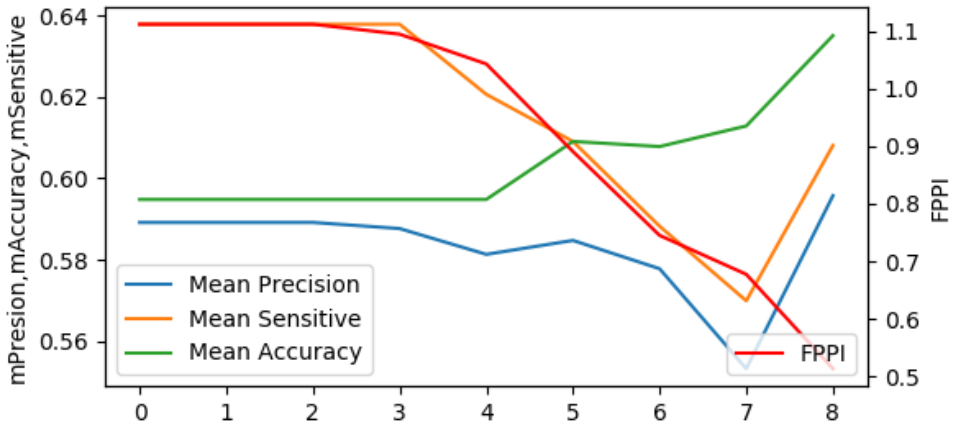


Figure 50: The effect of confidence threshold. x axis: $10 \times T$. The experiment is implemented only considering preprocessing and only train heads for 30 epochs.

Training Strategy Train strategy can also influence the performance seriously. We use the ImageNet [64] pretrained model for our initialized model, but mammogram has its special property so that the feature maps for common image is not appropriate, which means that we should figure out which part/stage in our model need to be re-trained. Figure 51 is our preliminary trial on training strategy, which is commonly used in other object detection tasks. We can get some general ideas from the figure:

- For each stage, 15 epochs $\times$ 500 steps/epoch is the most suitable options to

avoid overfitting.

- Mean Sensitive can benefit a lot from fine-tuning of the **backbone**.
- With the development of training, the FPPI decreases quickly, indicating that our model does work because the number of mark per image is quite small.

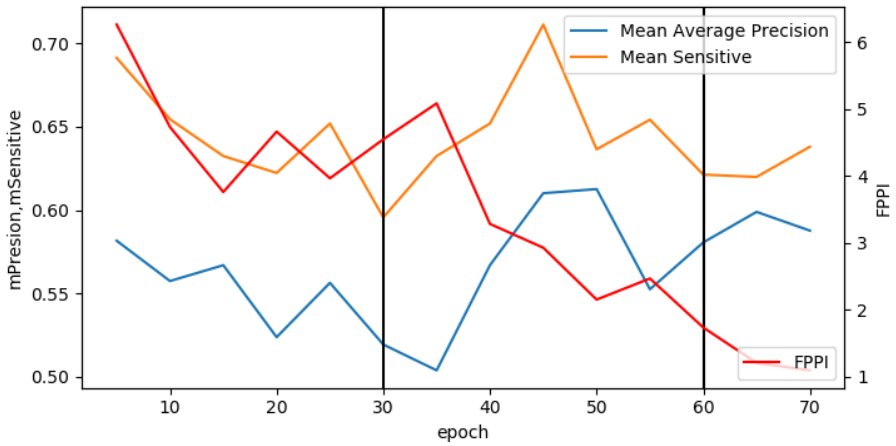


Figure 51: The effect of train strategy. **0-30 epochs**:train the heads(RPN,fc layers and mask generation layers).**30-60 epochs**:train the heads and part of backbone.**60-70 epochs**:train all layers

Besides the commonly used training strategy, we also adopt different training strategies from two perspectives. (1) **Training sequence**: the order of different training stages. (2) **Multi-task training**: the comparison of multi-stage-train and single-train-stage. Multi-stage-train means that train multiple stages simultaneously and the loss is accumulated, commonly multi-stage-training is beneficial for the collaboration of different stages [46]. However, things changed when the data becomes mammogram, when noticed that training multi-stages simultaneously is harmful for some stages' regression including FC layer for classification and bounding box regression. The overall comparison of different training strategy is shown in 15. We can see that the training sequence has no obvious influence on the final results while Single-stage-training is better than multi-stage-train as we expect. Also, four of six experiments are trained without long time training(15000k per stage), which may has a huge influence on our final decision, therefore more sufficient training must be implemented in future works.

6.2.3 Limitations

During our research on this topic, we gradually realize that our work still exists many limitations, some are unavoidable while some can be solved in future works. Following is the details of these limitations one by one:

Train sequence	Min Anchor Scale	Long Train	Mean AP(%)	Mean Sensitive(%)	FPPI
Head,4+,all	32	✓	60.3	72.6	3.14
(RPN,Mask,Backbone)×3,4+,all	16	✓	61.6	76.5	4.88
(RPN,Mask)×3,4+,all	16	✓	61.6	76.5	4.88
(Mask,RPN),4+×2,all×2	16		56.8	68.1	4.09
(RPN,Mask),4+×2,all×2	16		54.7	62.4	3.84
(Mask,RPN),4+×2,all×2	32		57.5	68.9	3.91
(RPN,Mask),4+×2,all×2	32		58.9	72.1	3.59

Table 15: Overall comparison of different training strategy and minimal anchor size. **RPN**: Train RPN individually.**Mask** Train FC layers and mask generation layers. **Backbone**: Train the backbone above 4th ResNet blocks.**head**: Train RPN and Mask simultaneously. **4+** Train the backbone above 4th ResNet block and the head simultaneously. **all** Train all layers simultaneously.

Low representation power of small feature maps Though [47] claims that they achieve a good results on small object detection using Feature Pyramid Networks [65], which translate small proposals in image to proposals in feature maps using multi-scale feature maps. But we noticed that the networks using features maps extracted by FPN did not get a good performance, typically, almost all experiments we have done based on FPN gained a high validation loss on final classification and bounding box regression, on the other hand, the validation bounding box regression loss of RPN is even better than the final bounding box regression. We guess the reason is that **the feature maps of small proposals have no enough representation information**. Therefore one direct idea is that using another network, which adopts proposals in image as input, instead of the feature maps, to get the final classification and regression score.

Few training data size With the study and understanding of the application of deep learning in CAD (computer aided diagnosis) field, we gradually realized that the key point of a successful work in CAD is not the **power of model**, but the **power of dataset**, which includes the quantity and quality of dataset. We achieved a satisfied results compared with other works using same dataset, but the results are still not impressive compared with other works using private dataset stored in hospital or university. Although we did many exploration on the development of model, we are confident that we are able to get more impressive results if we have more data.

6.2.4 Conclusion

A mass detection and segmentation system based on Mask RCNN was proposed and evaluated on public datasets. By the powerful representation ability of feature maps and development of GPU speed, we achieve 5× speedup compared with previous quickest work and outperform it, at the same time, we achieve a 150× speedup compared with most accurate work. Also, we proposed a new loss function and got a result with high precision. But our work still exists many potentials, the future work is discussed in Section 7

6.3 User Interface

In this section, we will demonstrate the appearance of our webapp front end. The user interface is designed for general users, so we have it as simple and clear as possible. The initial look of our project main page is shown in figure 52.

Figure 52: Overview of User Interface

6.3.1 Authentication

A valid GPU cluster account is required to use our program. Therefore, the user is expected to input credential on the webpage. Password text is obscured so that it cannot be read. In production, we will use HTTPS to encrypt the password. User agent will determine whether the credential should be saved or not. The appearance of this module is illustrated in figure 53.

Figure 53: Input Credential

6.3.2 Image Submission

Once a valid credential is provided, the user can pick PNG image files from local disk. This input accepts only PNG image files, and the file picker will display only files with an extension name of `png`. The appearance of this module is illustrated in figure 54.

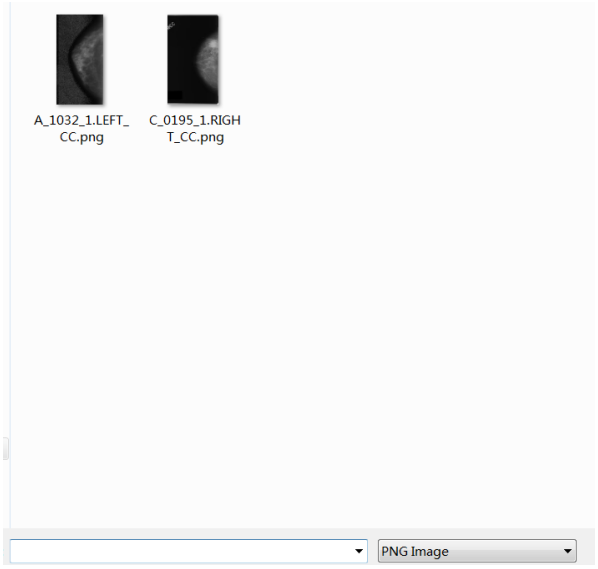


Figure 54: Pick a PNG File from the prompt

After selection, the png image file will be displayed on the page so that the user can preview the image to be uploaded. The appearance of this functionality is illustrated in figure 55.

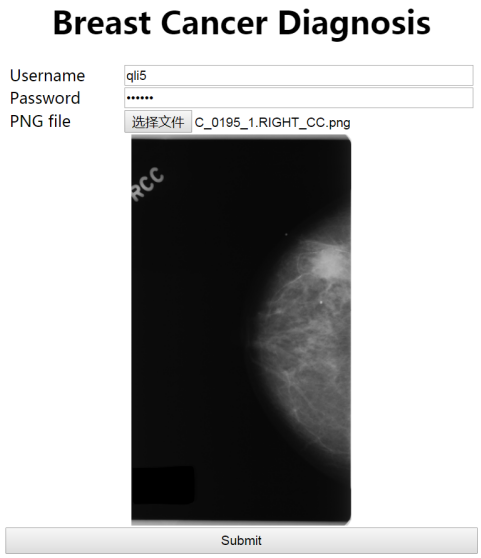


Figure 55: Preview the Selected Image

6.3.3 Progress Indicator

There are three steps in the program: uploading image, processing and generating report. The user can click the "submit" button to initiate a diagnosis request, which will upload the image and invoke the deep learning model on server. The user interface will display the estimated process time after submission. Since the diagnosis typically will take about two minutes, visual hints of the current status will be helpful. Three progress bars will be used to monitor of the program. The appearance of this module is illustrated in figure 56.

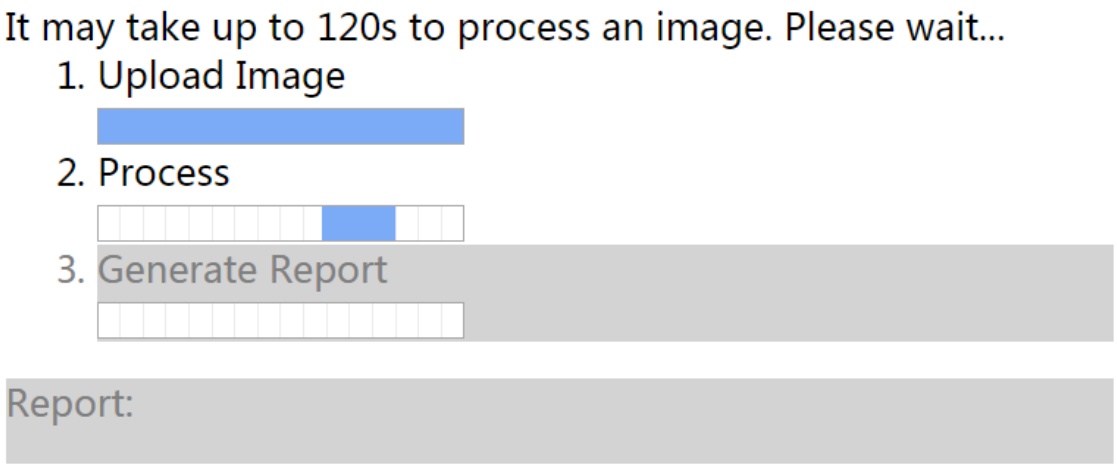


Figure 56: Indicate Progress of Each Step

The progress bar provides another convenience. If the program fails, we can easily know where the error occurs. This provide more details for debugging. The appearance of this functionality is illustrated in figure 57.

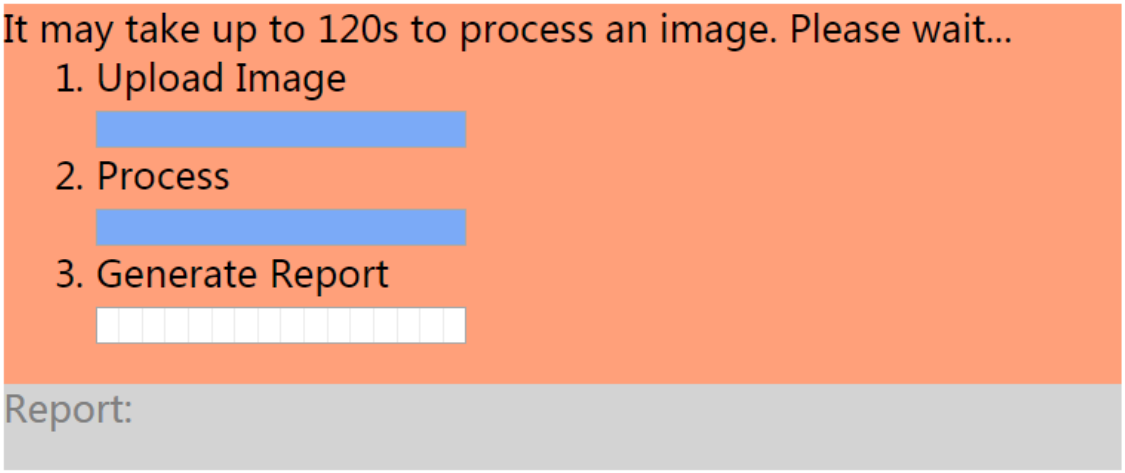


Figure 57: Indicate Error at Specific Step

6.3.4 Report Generator

If everything in the diagnosis system goes fine, a human readable report will be generated and displayed. The report will show the bounding box, the detailed range and the confidence level of the suspicious region. If an image contains multiple kinds of regions, they will be marked with different color. The appearance of a sample report is illustrated in figure 58.

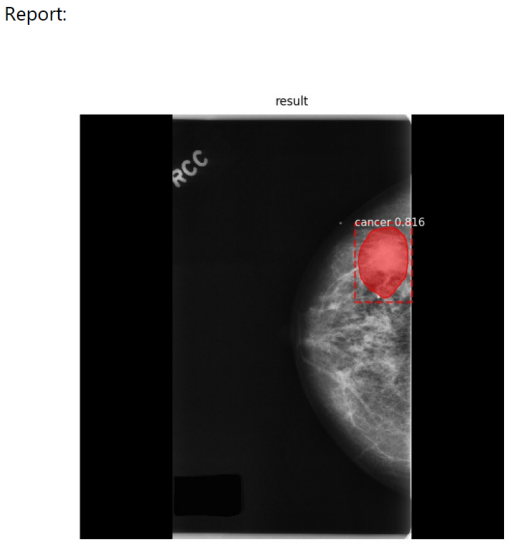


Figure 58: Generate a Human Readable Report

7 Conclusion

7.1 Project Review

When we started our final year project, we knew little about Tensorflow and image preprocess method or even some practical techniques in machine learning. We therefore regarded this project as a good chance for us to enhance our knowledge about deep learning and machine learning.

After continuous research and study from the related paper and trials of model constructions, we think we have achieved our basic goal, learning and understanding deep learning. The key effort and contributions we make in our final year project is listed below:

- We successfully reimplemented ResNet in histopathological image classification tasks and tuned the model structure using the knowledge of histopathological image features. Finally, we achieved pretty high accuracy which was up to 90% average comparing with 86% average in [50], [60], [61], [62] using same dataset.
- We used Mask RCNN as the base model and also modified the model structure to do a rough rumor detection, we also modified the loss function and positive sample decision method, which makes improves the precision significantly from 60% level to 90%.
- We built an efficient, user-friendly and reliable computer-aid diagnosis system using a web-site as the UI to help pathologist do breast cancer diagnosis faster, easier and more accurate.

7.2 Future Work

Despite the achievements we gain, there are still some directions which deserve a try:

- **Meta-information**(age, height and so on). One simple idea is to feed these normalized data into the fully connected layers in our model directly.
- **Pre-trained base model using mammogram as train data** instead of imageNet [64], which is misleading for model regression when training.
- **New network** using proposals in image as input stead of proposals in feature maps to solve the small feature maps problem we mentioned earlier.
- We have successfully demonstrated the feasibility of MaskRCNN in mammogram mass detection, therefore we are able to look for cooperation opportunity with other universities and hospitals to get more data, making our model more solid and compatible.

References

[1] J. G. Elmore, G. M. Longton, P. A. Carney, B. M. Geller, T. Onega, A. N. Tosteson, H. D. Nelson, M. S. Pepe, K. H. Allison, S. J. Schnitt *et al.*, "Diagnostic concordance among pathologists interpreting breast biopsy specimens," *Jama*, vol. 313, no. 11, pp. 1122–1132, 2015.

[2] F.-Y. Wang, J. J. Zhang, X. Zheng, X. Wang, Y. Yuan, X. Dai, J. Zhang, and L. Yang, "Where does alphago go: From church-turing thesis to alphago thesis and beyond," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 2, pp. 113–120, 2016.

[3] E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.

[4] J. Fulcher, "Computational intelligence: an introduction," in *Computational intelligence: a compendium*. Springer, 2008, pp. 3–78.

[5] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.

[6] "Documentation." [Online]. Available: https://www.mathworks.com/help/vision/examples/object-detection-in-a-cluttered-scene-using-point-feature-matching.html

[7] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–I.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[10] P. Werbos, "New tools for prediction and analysis in the behavioral science," *Ph. D. Dissertation, Harvard University*, 1974.

[11] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," *Neural Computation*, vol. 4, no. 2, pp. 234–242, 1992.

[12] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, p. 607, 1996.

[13] H.-C. Shin, M. R. Orton, D. J. Collins, S. J. Doran, and M. O. Leach, "Stacked autoencoders for unsupervised feature learning and multiple organ detection

in a pilot study using 4d patient data," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1930–1943, 2013.

[14] H. R. Roth, C. T. Lee, H.-C. Shin, A. Seff, L. Kim, J. Yao, L. Lu, and R. M. Summers, "Anatomy-specific classification of medical images using deep convolutional nets," in *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*. IEEE, 2015, pp. 101–104.

[15] L. Shapiro and G. Stockman, "Computer vision, march 2000."

[16] P. Moeskops, M. A. Viergever, A. M. Mendrik, L. S. de Vries, M. J. Benders, and I. Išgum, "Automatic segmentation of mr brain images with a convolutional neural network," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1252–1261, 2016.

[17] W. Zhang, R. Li, H. Deng, L. Wang, W. Lin, S. Ji, and D. Shen, "Deep convolutional neural networks for multi-modality isointense infant brain image segmentation," *NeuroImage*, vol. 108, pp. 214–224, 2015.

[18] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille, "Explain images with multimodal recurrent neural networks," *arXiv preprint arXiv:1410.1090*, 2014.

[19] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng, "Grounded compositional semantics for finding and describing images with sentences," *Transactions of the Association of Computational Linguistics*, vol. 2, no. 1, pp. 207–218, 2014.

[20] F. Ciompi, B. de Hoop, S. J. van Riel, K. Chung, E. T. Scholten, M. Oudkerk, P. A. de Jong, M. Prokop, and B. van Ginneken, "Automatic classification of pulmonary peri-fissural nodules in computed tomography using an ensemble of 2d views and a convolutional neural network out-of-the-box," *Medical image analysis*, vol. 26, no. 1, pp. 195–202, 2015.

[21] M. Gao, U. Bagci, L. Lu, A. Wu, M. Buty, H.-C. Shin, H. Roth, G. Z. Papadakis, A. Depeursinge, R. M. Summers *et al.*, "Holistic classification of ct attenuation patterns for interstitial lung diseases via deep convolutional neural networks," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 6, no. 1, pp. 1–6, 2018.

[22] "From virtual nurses to drug discovery: 106 artificial intelligence startups in healthcare," Feb 2018. [Online]. Available: https://www.cbinsights.com/research/artificial-intelligence-startups-healthcare/

[23] M. Kowal, P. Filipczuk, A. Obuchowicz, J. Korbicz, and R. Monczak, "Computer-aided diagnosis of breast cancer based on fine needle biopsy microscopic images," *Computers in biology and medicine*, vol. 43, no. 10, pp. 1563–1572, 2013.

[24] Y. M. George, H. H. Zayed, M. I. Roushdy, and B. M. Elbagoury, "Remote computer-aided breast cancer detection and diagnosis system based on cytological images," *IEEE Systems Journal*, vol. 8, no. 3, pp. 949–964, 2014.

[25] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, p. 115, 2017.

[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[27] W. Zhu, Q. Lou, Y. S. Vang, and X. Xie, "Deep multi-instance networks with sparse label assignment for whole mammogram classification," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2017, pp. 603–611.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[29] M. Heath, K. Bowyer, D. Kopans, R. Moore, and P. Kegelmeyer, "The digital database for screening mammography," *Digital mammography*, pp. 431–434, 2000.

[30] H. Min, S. S. Chandra, N. Dhungel, S. Crozier, and A. P. Bradley, "Multi-scale mass segmentation for mammograms via cascaded random forests," in *Biomedical Imaging (ISBI 2017), 2017 IEEE 14th International Symposium on*. IEEE, 2017, pp. 113–117.

[31] I. C. Moreira, I. Amaral, I. Domingues, A. Cardoso, M. J. Cardoso, and J. S. Cardoso, "Inbreast: Toward a full-field digital mammographic database," *Academic Radiology*, vol. 19, no. 2, pp. 236 – 248, 2012.

[32] N. Dhungel, G. Carneiro, and A. P. Bradley, "Automated mass detection in mammograms using cascaded deep learning and random forests," in *Digital Image Computing: Techniques and Applications (DICTA), 2015 International Conference on*. IEEE, 2015, pp. 1–8.

[33] M. Beller, R. Stotzka, T. O. Müller, and H. Gemmeke, "An example-based system to support the segmentation of stellate lesions," in *Bildverarbeitung für die Medizin 2005*. Springer, 2005, pp. 475–479.

[34] V. Kumar, A. K. Abbas, and J. C. Aster, *Robbins Basic Pathology E-Book*. Elsevier Health Sciences, 2017.

[35] J. P. Kösters and P. C. Gøtzsche, "Regular self-examination or clinical examination for early detection of breast cancer," *The Cochrane Library*, 2003.

[36] G. Majno and I. Joris, *Cells, tissues, and disease: principles of general pathology*. Oxford University Press, 2004.

[37] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[38] G. van Tulder and M. de Bruijne, "Combining generative and discriminative representation learning for lung ct analysis with convolutional restricted boltzmann machines," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1262–1272, 2016.

[39] Q. Dou, H. Chen, L. Yu, L. Zhao, J. Qin, D. Wang, V. C. Mok, L. Shi, and P.-A. Heng, "Automatic detection of cerebral microbleeds from mr images via 3d convolutional neural networks," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1182–1195, 2016.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[41] D. Attwell and S. B. Laughlin, "An energy budget for signaling in the grey matter of the brain," *Journal of Cerebral Blood Flow & Metabolism*, vol. 21, no. 10, pp. 1133–1145, 2001.

[42] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[43] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[44] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders, "Segmentation as selective search for object recognition," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1879–1886.

[45] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.

[46] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[47] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2980–2988.

[48] A. Kessy, A. Lewin, and K. Strimmer, "Optimal whitening and decorrelation," *The American Statistician*, pp. 1–6, 2018.

[49] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J. B. Zimmerman, and K. Zuiderveld, "Adaptive histogram equalization and its variations," *Computer vision, graphics, and image processing*, vol. 39, no. 3, pp. 355–368, 1987.

[50] F. A. Spanhol, L. S. Oliveira, C. Petitjean, and L. Heutte, "A dataset for breast cancer histopathological image classification," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 7, pp. 1455–1462, 2016.

[51] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2. Montreal, Canada, 1995, pp. 1137–1145.

[52] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[53] C. Nvidia, "Zone," 2016.

[54] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[55] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[56] L.-Y. Deng, "The cross-entropy method: a unified approach to combinatorial optimization, monte-carlo simulation, and machine learning," 2006.

[57] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[58] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[59] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free?-weakly-supervised learning with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 685–694.

[60] F. A. Spanhol, L. S. Oliveira, C. Petitjean, and L. Heutte, "Breast cancer histopathological image classification using convolutional neural networks," in

Neural Networks (IJCNN), 2016 International Joint Conference on. IEEE, 2016, pp. 2560–2567.

[61] F. A. Spanhol, L. S. Oliveira, P. R. Cavalin, C. Petitjean, and L. Heutte, "Deep features for breast cancer histopathological image classification," in *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on.* IEEE, 2017, pp. 1868–1873.

[62] N. Bayramoglu, J. Kannala, and J. Heikkilä, "Deep learning for magnification independent breast cancer histopathology image classification," in *Pattern Recognition (ICPR), 2016 23rd International Conference on.* IEEE, 2016, pp. 2440–2445.

[63] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision.* Springer, 2014, pp. 740–755.

[64] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[65] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," *ArXiv e-prints*, Dec. 2016.

magnification	Image level					Batch level			
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix		
40×	sum	81.77	89.83	**83.91**	86.76	81.91	AUC(%)	83.03	
	vote	**82.33**	96.61	80.66	87.92				predict
	average	**82.33**	96.61	80.66	87.92				malignant benign
	exist	81.95	**99.44**	78.92	88.00		actual	malignant	17013 687
	exist3	82.14	98.87	79.37	**88.05**			benign	4126 4774
100×	sum	81.15	88.74	**83.46**	86.02	83.57	AUC(%)	84.91	
	vote	83.84	94.51	83.09	88.43				predict
	average	83.84	94.51	83.09	88.43				malignant benign
	exist	**84.02**	**98.90**	80.90	**89.00**		actual	malignant	17147 1087
	exist3	**84.02**	97.52	81.61	88.86			benign	3496 6170
200×	sum	86.86	95.89	86.00	90.67	88.31	AUC(%)	89.78	
	vote	88.87	99.18	86.19	92.23				predict
	average	**89.05**	99.18	**86.40**	**92.34**				malignant benign
	exist	86.86	**100.0**	83.52	91.02		actual	malignant	17996 254
	exist3	87.96	**100.0**	84.69	91.71			benign	2949 6201
400×	sum	86.54	96.22	**84.87**	90.19	86.82	AUC(%)	89.84	
	vote	87.10	98.84	83.95	90.79				predict
	average	**87.29**	99.13	83.99	**90.93**				malignant benign
	exist	86.36	**100.0**	82.49	90.41		actual	malignant	16989 244
	exist3	86.36	99.71	82.65	90.38			benign	3287 6280

Table 16: The results of model with first convolution layer's kernel size as 3×3, the visualization version of model structure is in (Middle-Right,Figure 41)

8 Appendix

magnification	Image level					Batch level			
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix		
40×	sum	85.15	91.53	**86.86**	89.13	85.16	AUC(%)	86.97	
	vote	**86.65**	96.61	85.29	**90.60**				predict
	average	86.28	96.33	85.04	90.33				malignant benign
	exist	81.01	**99.15**	78.17	87.42		actual	malignant	16932 768
	exist3	83.08	**99.15**	80.14	88.63			benign	3179 5721
100×	sum	82.22	90.38	83.72	86.92	84.66	AUC(%)	79.13	
	vote	85.82	97.25	83.69	89.96				predict
	average	**86.00**	97.53	**83.73**	**90.10**				malignant benign
	exist	84.02	**99.45**	80.62	89.05		actual	malignant	17494 746
	exist3	84.92	**99.45**	81.53	89.60			benign	3534 6126
200×	sum	87.41	95.89	**86.63**	91.03	87.19	AUC(%)	85.07	
	vote	87.59	98.36	85.27	91.35				predict
	average	**87.77**	98.36	85.48	**91.46**				malignant benign
	exist	85.40	**100.0**	82.02	90.12		actual	malignant	17857 393
	exist3	86.50	**100.0**	83.14	90.80			benign	3116 6034
400×	sum	85.05	92.73	**85.29**	88.86	85.68	AUC(%)	87.09	
	vote	85.98	96.80	83.88	89.88				predict
	average	**86.36**	97.09	84.13	**90.15**				malignant benign
	exist	84.67	**98.84**	81.34	89.24		actual	malignant	16563 672
	exist3	85.23	**98.84**	81.93	89.59			benign	3165 6400

Table 17: The results of model with first convolution's stride as 2, the visualization version of model structure is in (Right,Figure 41)

magnification	Image level					Batch level			
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix		
40×	sum	82.89	90.40	**84.88**	87.55	84.91	AUC(%)	84.15	
	vote	**86.28**	97.46	84.35	**90.43**				predict
	average	86.09	97.46	84.15	90.31				malignant benign
	exist	83.83	**99.44**	80.73	89.11		actual	malignant	16962 738
	exist3	84.59	99.15	81.62	89.54			benign	3275 5625
100×	sum	81.87	90.11	**83.46**	86.66	81.93	AUC(%)	82.91	
	vote	**83.30**	95.05	82.19	**88.15**				predict
	average	83.12	95.05	81.99	88.04				malignant benign
	exist	79.17	**99.18**	76.16	86.16		actual	malignant	17080 1156
	exist3	80.43	98.63	77.54	86.16			benign	3885 5779
200×	sum	**88.87**	99.18	**86.19**	92.03	88.14	AUC(%)	91.21	
	vote	88.69	**100.0**	85.48	**92.17**				predict
	average	88.50	**100.0**	85.28	92.06				malignant benign
	exist	85.58	**100.0**	82.21	90.23		actual	malignant	18218 32
	exist3	86.13	**100.0**	82.77	90.57			benign	3217 5933
400×	sum	**87.85**	95.64	**86.81**	**91.01**	86.56	AUC(%)	89.53	
	vote	86.73	98.55	83.70	90.52				predict
	average	86.54	98.55	83.50	90.40				malignant benign
	exist	84.67	**99.71**	80.90	89.32		actual	malignant	16833 396
	exist3	85.61	**99.71**	81.86	89.91			benign	3207 6364

Table 18: The results of model with feature maps doubled, the visualization version of model structure is in (Middle-Left,Figure 41)

magnification	Image level					Batch level			
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix		
40×	sum	81.20	86.72	**85.28**	85.99	83.03	AUC(%)	79.76	
	vote	83.65	93.22	83.97	88.35				predict
	average	83.65	93.22	83.97	88.35				malignant benign
	exist	83.46	**99.44**	80.37	88.89		actual	malignant	16313 1387
	exist3	**84.21**	98.87	81.40	**89.29**			benign	3128 5772
100×	sum	80.79	89.29	82.70	85.87	83.40	AUC(%)	79.99	
	vote	84.38	96.15	**82.74**	88.95				predict
	average	84.02	95.60	82.66	88.66				malignant benign
	exist	**84.56**	**99.73**	79.08	88.21		actual	malignant	17386 843
	exist3	**84.56**	**99.73**	81.03	**89.41**			benign	3788 5883
200×	sum	**88.87**	98.36	**86.71**	92.17	88.15	AUC(%)	88.06	
	vote	88.69	**100.0**	85.48	92.17				predict
	average	**88.87**	**100.0**	85.68	**92.29**				malignant benign
	exist	84.67	**100.0**	81.29	89.68		actual	malignant	18181 69
	exist3	85.77	**100.0**	82.39	90.35			benign	3178 5972
400×	sum	**88.04**	97.38	**85.90**	**91.28**	86.10	AUC(%)	86.47	
	vote	86.17	99.13	82.77	90.21				predict
	average	86.36	99.42	82.81	90.36				malignant benign
	exist	83.36	**100.0**	79.45	88.55		actual	malignant	16971 256
	exist3	84.30	**100.0**	80.37	89.12			benign	3470 6103

Table 19: The results of model with 2 pool layers before Residual blocks, the visualization version of model structure is in (Middle,Figure 41)

magnification	Image level					Batch level			
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix		
40×	sum	79.51	84.46	**84.70**	84.58	80.03	**AUC(%)**	80.68	
	vote	**81.95**	89.55	84.31	**86.85**				predict
	average	81.77	89.27	84.27	86.69				malignant benign
	exist	77.63	**93.79**	77.34	84.80		actual	malignant	15594 2106
	exist3	78.95	92.37	79.37	85.38			benign	3205 5695
100×	sum	77.56	82.42	**83.10**	82.76	79.09	**AUC(%)**	79.42	
	vote	77.56	87.91	79.80	83.66				predict
	average	77.56	87.91	79.80	83.66				malignant benign
	exist	78.28	**98.35**	75.69	85.54		actual	malignant	16097 2131
	exist3	**78.64**	96.98	76.57	**85.58**			benign	3981 5691
200×	sum	88.32	95.89	**87.72**	91.62	87.84	**AUC(%)**	88.36	
	vote	**88.87**	97.81	87.07	**92.13**				predict
	average	**88.87**	97.81	87.07	**92.13**				malignant benign
	exist	84.67	**99.45**	81.57	89.63		actual	malignant	17699 551
	exist3	86.13	99.18	83.22	90.50			benign	2782 6368
400×	sum	77.94	86.92	80.38	83.52	81.09	**AUC(%)**	85.73	
	vote	82.24	95.06	80.74	87.32				predict
	average	82.42	95.35	**80.79**	87.47				malignant benign
	exist	**82.99**	**99.13**	79.49	**88.22**		actual	malignant	16016 1219
	exist3	**82.99**	97.97	80.05	88.10			benign	3850 5715

Table 20: The results using RAW image as input (no preprocess method) in both batch level and image level

magnification	Image level					Batch level			
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix		
40×	sum	62.40	65.54	**74.83**	69.87	65.09	**AUC(%)**	68.89	
	vote	65.22	84.75	69.61	76.43				predict
	average	65.79	85.59	69.82	76.90				malignant benign
	exist	**68.42**	**97.18**	68.52	**80.37**		actual	malignant	14944 2756
	exist3	67.48	95.20	68.37	79.57			benign	6529 2371
100×	sum	59.42	49.73	**80.80**	61.56	69.28	**AUC(%)**	70.39	
	vote	**69.12**	96.98	68.68	**80.41**				predict
	average	68.76	96.98	68.41	80.23				malignant benign
	exist	67.50	**98.90**	67.04	79.91		actual	malignant	17400 833
	exist3	67.68	98.63	67.23	79.95			benign	7739 1928
200×	sum	75.18	74.79	**86.12**	80.06	75.90	**AUC(%)**	81.52	
	vote	76.64	87.67	79.40	83.33				predict
	average	**77.19**	88.77	79.41	**83.83**				malignant benign
	exist	73.91	**96.99**	72.84	83.20		actual	malignant	15872 2378
	exist3	74.45	96.44	73.49	83.41			benign	4225 4925
400×	sum	77.76	81.40	**83.58**	82.47	78.26	**AUC(%)**	82.15	
	vote	**80.37**	90.99	80.88	**85.64**				predict
	average	80.18	91.28	80.51	85.56				malignant benign
	exist	71.78	**98.26**	69.98	81.74		actual	malignant	15433 1796
	exist3	74.21	**97.38**	72.20	82.92			benign	4031 5540

Table 21: The results whose images are preprocessed by subtracting Gaussian image and applying CLAHE

magnification	Image level					Batch level			
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix		
40×	sum	85.90	95.48	**85.14**	90.01	86.17	**AUC(%)**	82.80	
	vote	**87.03**	99.15	84.17	**91.05**				predict
	average	86.84	99.15	83.97	90.93				malignant benign
	exist	82.89	**100.0**	79.55	88.61		actual	malignant	17374 326
	exist3	84.21	**100.0**	80.82	89.40			benign	3352 5548
100×	sum	78.64	86.54	**81.82**	84.11	81.44	**AUC(%)**	79.42	
	vote	81.87	93.68	81.38	87.10				predict
	average	**82.05**	93.96	81.43	87.24				malignant benign
	exist	81.15	**98.90**	78.09	87.27		actual	malignant	17085 1142
	exist3	81.69	98.08	78.98	**87.50**			benign	4035 5638
200×	sum	81.39	88.49	**84.33**	86.36	84.96	**AUC(%)**	85.41	
	vote	**85.77**	97.81	83.61	**90.15**				predict
	average	85.40	97.26	83.53	89.87				malignant benign
	exist	82.48	**99.73**	79.30	88.35		actual	malignant	17691 559
	exist3	83.58	99.45	80.49	88.97			benign	3562 5588
400×	sum	77.94	86.63	**80.54**	83.47	80.15	**AUC(%)**	82.05	
	vote	**80.75**	95.06	79.18	86.39				predict
	average	80.56	94.77	79.13	86.24				malignant benign
	exist	78.69	**99.13**	75.44	85.48		actual	malignant	16182 1047
	exist3	80.56	98.84	77.27	**86.73**			benign	4273 5295

Table 22: The results using both CLAHE and whiten methods (keep the function order) in both batch and image level

magnification	Image level					Batch level			
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix		
40×	sum	81.58	86.72	**85.75**	86.23	82.93	**AUC(%)**	82.19	
	vote	82.89	93.50	82.96	87.92				predict
	average	**83.08**	93.50	83.17	88.03				malignant benign
	exist	81.20	**99.72**	78.10	**89.59**		actual	malignant	16542 1158
	exist3	82.89	99.15	79.95	88.52			benign	3382 5518
100×	sum	80.07	87.09	83.20	85.10	83.37	**AUC(%)**	76.98	
	vote	84.20	94.51	**83.50**	88.66				predict
	average	84.20	94.51	**83.50**	88.66				malignant benign
	exist	83.30	**98.90**	80.18	88.56		actual	malignant	17051 1185
	exist3	**84.74**	92.35	81.92	**89.39**			benign	3456 6208
200×	sum	87.77	96.99	**86.34**	91.35	88.33	**AUC(%)**	85.02	
	vote	**88.87**	99.73	85.85	**92.27**				predict
	average	88.69	99.73	85.65	92.15				malignant benign
	exist	86.31	**100.0**	82.95	90.68		actual	malignant	18128 122
	exist3	87.22	**100.0**	83.91	91.25			benign	3075 6075
400×	sum	83.18	91.86	83.60	87.53	86.15	**AUC(%)**	82.61	
	vote	**86.73**	97.67	**84.21**	90.44				predict
	average	**86.73**	97.67	**84.21**	90.44				malignant benign
	exist	86.17	99.71	82.45	90.26		actual	malignant	16743 487
	exist3	**86.73**	99.71	83.05	**90.62**			benign	3225 6345

Table 23: The results using CLAHE in both batch level and image level

magnification	Image level					Batch level				
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix			
40×	sum	84.40	91.24	**86.13**	88.61	85.84	**AUC(%)**	80.97		
	vote	86.09	98.02	83.82	90.36				predict	
	average	**86.28**	98.02	84.02	**90.48**				malignant	benign
	exist	84.40	**100.0**	81.01	89.51		actual	malignant	17278	422
	exist3	84.02	99.44	80.92	89.23			benign	3345	5555
100×	sum	79.17	88.46	**81.31**	84.74	81.42	**AUC(%)**	82.23		
	vote	**82.23**	97.53	79.77	**87.76**				predict	
	average	81.69	96.70	79.64	87.34				malignant	benign
	exist	80.43	**100.0**	79.64	86.98		actual	malignant	17614	617
	exist3	81.51	99.73	78.06	87.58			benign	4566	5103
200×	sum	82.66	89.32	**85.34**	87.28	84.65	**AUC(%)**	87.63		
	vote	**85.22**	97.53	83.18	**89.79**				predict	
	average	**85.22**	97.53	83.18	**89.79**				malignant	benign
	exist	81.39	**99.73**	78.28	87.71		actual	malignant	17731	519
	exist3	82.85	99.18	79.91	88.51			benign	3686	5464
400×	sum	79.81	90.41	**80.57**	85.21	81.75	**AUC(%)**	83.38		
	vote	**82.80**	97.97	79.86	**87.99**				predict	
	average	82.62	97.97	79.67	87.87				malignant	benign
	exist	79.07	**99.42**	75.66	85.93		actual	malignant	16668	565
	exist3	80.19	**99.42**	76.68	86.58			benign	4325	5242

Table 24: The results using both whiten and CLAHE methods (keep the function order) in both batch and image level

magnification	Image level					Batch level				
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix			
40×	sum	85.15	94.07	**85.17**	89.40	85.82	**AUC(%)**	78.65		
	vote	**86.84**	99.44	83.81	**90.96**				predict	
	average	86.65	99.44	83.61	90.84				malignant	benign
	exist	82.14	**100.0**	78.84	88.17		actual	malignant	17444	256
	exist3	83.46	**100.0**	80.09	88.94			benign	3516	5384
100×	sum	81.69	92.31	**81.95**	86.82	82.32	**AUC(%)**	82.19		
	vote	**83.12**	96.98	80.96	**88.25**				predict	
	average	82.94	96.70	80.92	88.11				malignant	benign
	exist	81.33	**99.73**	77.90	87.47		actual	malignant	17590	644
	exist3	82.05	98.90	78.95	87.80			benign	4288	5378
200×	sum	80.66	87.67	**83.99**	85.79	84.31	**AUC(%)**	86.22		
	vote	85.04	96.71	83.45	89.59				predict	
	average	**85.22**	96.99	83.49	**89.73**				malignant	benign
	exist	82.30	**99.73**	79.13	88.24		actual	malignant	17516	734
	exist3	83.58	99.45	80.49	88.97			benign	3564	5586
400×	sum	79.25	89.24	**80.58**	84.69	80.49	**AUC(%)**	83.11		
	vote	**81.31**	97.38	78.64	**87.01**				predict	
	average	**81.31**	97.38	78.64	**87.01**				malignant	benign
	exist	77.57	**99.13**	74.45	85.04		actual	malignant	16616	616
	exist3	78.69	98.84	75.56	85.64			benign	4613	4955

Table 25: The results using whiten method in both batch level and image level

magnification	Image level					Batch level			
	aggregation method	accuracy (%)	precision (%)	recall (%)	F1 score (%)	accuracy (%)	confusion matrix		
40×	sum	79.32	84.46	**84.46**	84.46	79.42	**AUC(%)**	82.41	
	vote	**79.51**	91.81	80.25	**85.64**				predict
	average	**79.51**	91.81	80.25	**85.64**				malignant benign
	exist	76.88	**96.33**	75.61	84.72		actual	malignant	16256 1444
	exist3	77.07	94.63	76.48	84.60			benign	4031 4869
100×	sum	72.35	75.27	81.07	78.06	78.01	**AUC(%)**	81.54	
	vote	78.10	87.64	80.56	83.95				predict
	average	78.54	87.64	**81.17**	84.28				malignant benign
	exist	78.99	**95.70**	77.02	85.75		actual	malignant	15927 2311
	exist3	**79.89**	95.33	78.51	**86.10**			benign	3823 5839
200×	sum	80.47	88.22	**83.42**	85.75	83.91	**AUC(%)**	82.62	
	vote	**84.67**	97.53	82.60	**89.45**				predict
	average	**84.67**	97.53	82.60	**89.45**				malignant benign
	exist	83.94	**98.90**	81.12	89.14		actual	malignant	17520 730
	exist3	84.12	98.63	81.45	89.22			benign	3680 5470
400×	sum	80.75	88.08	83.01	85.47	82.96	**AUC(%)**	82.97	
	vote	83.55	93.60	82.99	87.98				predict
	average	83.74	93.90	**83.03**	88.13				malignant benign
	exist	**84.67**	**98.84**	81.34	**89.24**		actual	malignant	16002 1237
	exist3	84.49	**97.38**	81.91	88.98			benign	3329 6232

Table 26: The results using preprocess method in past papers, simply demean the images