Summer 7-14-2020

Deep Learning for Identifying Breast Cancer

Yihong Li

DEEP LEARNING FOR IDENTIFYING BREAST CANCER

A Thesis
Presented to
The Faculty of the
Computing and Software
Engineering Department

By

Yihong Li

In Partial Fulfillment
of Requirements for the Degree
Master of Computer Science in the
Kennesaw State University
Department of Computing and Software Engineering

Kennesaw State University

July 2020

DEEP LEARNING FOR IDENTIFYING BREAST CANCER

Thesis committee:

Dr. Mohammed Aledhari
Computing and Software Engineering
Kennesaw State University

Dr. Chih-Cheng Hung
Computing and Software Engineering
Kennesaw State University

Dr. Hisham Haddad
Computing and Software Engineering
Kennesaw State University

Date approved: July 14, 2020

ACKNOWLEDGMENTS

I would like to thank the members of the thesis committee for their help and suggestions in preparing for this work. Without them, I am doomed to be unable to complete it. Dr. Mohammed Aledhari helped me clarify many ideas, provide many new ideas and suggestions about the my project. It also gave me clear directions and guidance. In addition, it will meet with me every week to discuss the progress and content of the paper, and give great help in the process of the paper. Dr. Chih-Cheng Hung and Dr. Hisham Haddad also made very meaningful suggestions on the title and content of my thesis. I am very happy to invite these three doctors as members of my thesis committee.

Special thanks to friends and colleagues who made this work possible. Ms. Lin Wang is my friend, we have the same advisor. So, we often exchange ideas together on projects or preparations. This is very helpful for the completion of my thesis.

Here, I would like to express my heartfelt thanks to all the people who helped in the process from the submission to the completion of the paper.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Medical images are playing an increasingly important role in the prevention and diagnosis of diseases. Medical images often contain massive amounts of data. Professional interpretation usually requires a long time of professional study and experience accumulation by doctors. Therefore, the use of super storage and computing power in deep learning as a basis can effectively process a large amount of medical data. Breast cancer brings great harm to female patients, and early diagnosis is the most effective prevention and treatment method, so this project will create a new optimized breast cancer auxiliary diagnosis model based on ResNet. Analyze and process, realize medical aided diagnosis, and provide scientific diagnosis for breast cancer patients.

CHAPTER 1
INTRODUCTION AND BACKGROUND

1.1 Introduction

People's health is always a very important topic. Cancer is one of the most malignant diseases with the highest morbidity and mortality in the world, which seriously threatens people's health and life [1]. Today, the incidence of cancer continues to rise, and researchers around the world are making every effort to create and find more effective and advanced ways to fight and prevent cancer. Therefore, researchers around the world are participating in the fight against cancer. Cancer has become more and more clear and complicated in the public's vision, forming an inevitable huge medical component. Although, researchers and medical staff are doing their best to develop innovative drugs and medical methods to prevent, classify and treat cancer. However, it is still far from the goal [2]. In order to suppress the development of cancer to people; accelerate the prevention and treatment of cancer, all aspects of society make their contribution. For example, the 21st Century Cure Law provided 4.8 billion and the Precision Medicine Initiative [3] for Cancer Monthly Bulletin [4]. This method not only provides great help and encouragement to researchers, but also greatly encourages patients.

Among them, with the continuous maturity of computer technology, medical image recognition technology in computer technology has also been further developed, and has been widely used in benign or malignant tumors, brain function and mental disorders, cardiovascular and cerebrovascular diseases and other major diseases The clinical auxiliary screening, diagnosis, classification and treatment plan have made significant contributions [5]. Traditional machine learning mainly relies on manual feature extraction of image data, and uses the extracted features for model training. The process is time-consuming, labori-

ous, and unstable. But the deep learning assisted diagnosis technology can automatically extract features and greatly improve the stability, so this project is based on CNN to identify the benign and malignant breast tumors.

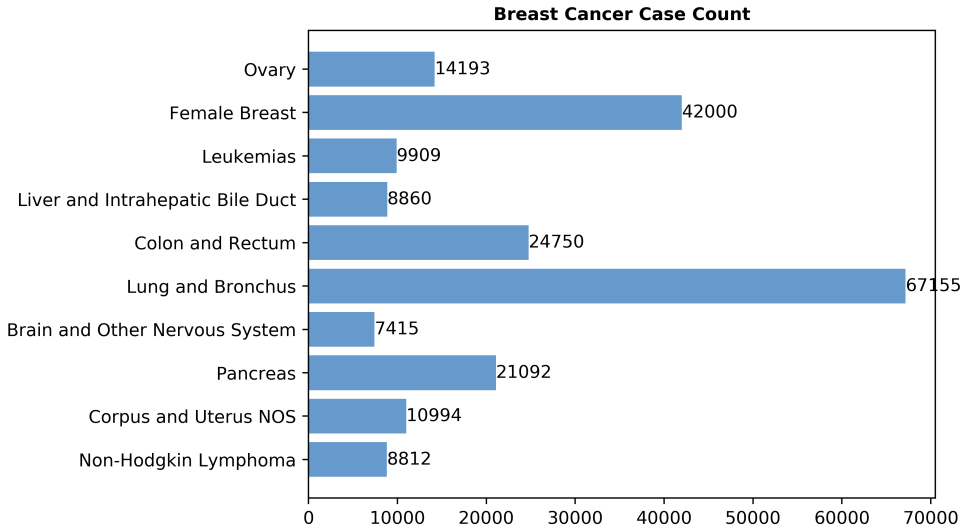


Figure 1.1: Top 10 Cancers by Rates of Cancer Deaths.

Breast cancer is the most common cause of cancer death among women. And can be ranked second [6], as shown in Figure 1.1. According to statistics from the American Cancer Society, approximately 25% of cancer patients are breast cancer patients. And in the past five years, the incidence of breast cancer has increased by a certain percentage every year [7]. There are many reasons for breast cancer, for example, oral contraceptives, short breastfeeding time, heavy drinking, family medical history. Lots of bad habits will increase the risk of breast cancer[8] [9]. But for most women, the current medical technology cannot be sure about the cause of breast cancer [10], and early breast cancer does not have the typical symptoms and characteristics, often be overlooked. However, this disease is usually fatal to women and therefore requires very effective rapid diagnosis and effective treatment. This has huge social benefits for the entire society and women around the world. At present, deep learning is a very effective detection and diagnosis method.

At present, we propose use BreakHis breast cancer dataset to train and test ResNet model, and will do a comprehensive comparison. Then, optimize the original ResNet model

and create a new model with better performance. For the proposed method, we chose to combine genetic algorithms. Compared with traditional algorithms, genetic algorithm can search multiple peaks in parallel, and has good operability and global optimization features. In addition, it only uses the objective function and the corresponding adaptive function, and does not need other auxiliary information. So, it can effectively solve the shortcomings of traditional algorithms. Applying this proposed method to clinical aspects can not only help experienced doctors to quickly screen for triage diseases, diagnose more patients in a limited time, but also help doctors who cannot yet accurately judge breast cancer to conduct breast cancer screening, saving The cost is conducive to a large-scale promotion and popularization in the society. Therefore, the research in this article is of great significance to the improvement of medical service capabilities and social development.

1.2 Significance of the Proposed Project

The beneficiaries of the project to identify malignant cancer are not only patients, but also have a very large contribution to medical aspect. For patients, if malignant tumors can be detected at an early stage of disease, the chance of treatment will be high. However, at present, due to the limitations of the collected medical equipment, manual diagnosis and reading medical films, there will still be cases of misdiagnosis, missed diagnosis and untimely diagnosis. Through this technology, patients can obtain more efficient diagnosis results and judgment efficiency, thereby avoiding the use of too much time in the hospital to wait in line and wait for the time for consultation; for the medical institutions, they have accumulated a lot of data but do not know how effective clinical use; for medical device manufacturers, they also want to use artificial intelligence to improve the product technology level to expand their market; for doctors, they hope to reduce the cost of reading time and reduce the probability of misdiagnosis.

Through training and experimenting with massive medical data, learning and research of medical images, computers can generate accurate prediction models, and provide early

warning and outcome evaluation for malignant diseases. This greatly solves the above-mentioned contradictions and problems and makes a huge contribution to cancer prevention and detection in the future.

Table 1.1: Different types of breast tumors.

Breast Tumors	**Diseases Name**
Benign Tumors	Adenoma (A) Fibroadenoma (F) Phyllodes Tumor (PT) Renal Tubular Adenocarcinoma (TA)
Malignant Tumors	Cancer (DC) Lobular Carcinoma (LC) Mucinous Carcinoma (MC) Papillary Carcinoma (PC)

1.3 Research Question

We proposed a method for breast cancer recognition based on ResNet model. And use BreakHis dataset. The BreakHis database contains microscopic biopsy images of benign and malignant breast tumors. They are collected through clinical research, and are images that have been distinguished by physicians, and are authoritative. There are also different types of breast tumors. First, breast tumors are divided into benign and malignant in Table 1.1. The types of benign tumors are adenosis (A), fibroadenoma (F), phyllodes tumor (PT), and tubular adenona (TA); and four malignant tumors (breast cancer): carcinoma (DC), lobular carcinoma (LC), mucinous carcinoma (MC) and papillary carcinoma (PC). Therefore, the purpose of this project is to distinguish and identify the characteristics of malignant tumors through medical data sets, help doctors quickly diagnose cancer, and reduce the chance of misdiagnosis and missed diagnosis. In our experiment, we made a comprehensive comparison between the original ResNet and the proposed method model, including activation function, learning rate, epoch, loss function, optimization function,

confusion matrix, accuracy, loss, F1 score, accuracy, Recall, specificity, and sensitivity. Finally, to achieve our experimental purpose.

The second part is that the BreakHis dataset includes images of breast tumors with different magnifications, including 40X, 100X, 200X, and 400X. We also will use the original ResNet and proposed method for comparison and analysis. Therefore, the impact of the magnification under the microscope on the diagnosis of the disease is analyzed, and the criteria for judging breast tumor diseases are also analyzed.

1.4 Purpose of Study

Medical image analysis and external performance have always been the focus of doctors. At present, an automated structure can effectively help doctors to improve the accuracy of recognition and judgment. It is very difficult for people to judge the type of breast cancer only by vision, so it is very important to use machines to perform accurate image processing and feature extraction and finally reach the level of a doctor. Each type of different breast cancer has its own unique characteristics, and the machine can learn them and accurately classify them is the ultimate goal of our project. Avoid the deterioration of people's breast cancer and lead to irreparable consequences, and help patients and doctors have a deeper understanding and understanding of breast cancers.

1.5 Audience

This research involves multiple fields, not only for individuals but also has a huge impact on medicine, so it has a wide and diverse target audience. Since it involves medicine, it is also aimed at experts of the Biomedical Imaging Society. For example, for individual patients, their early detection of diseased breast cancer will increase the chance of treatment. And can obtain more effective diagnosis results and judgment efficiency, thereby avoiding the disease from developing into cancer and waiting for consultation time to spend too much time; for medical device manufacturers, using artificial intelligence to improve the technical

level of products and expand the market; in addition, The doctor's aspect will reduce the time to read cases and X-rays, and reduce the possibility of misdiagnosis.

Through training and experiments on a large number of medical data, learning and research on medical images and medical records, the machine can generate accurate prediction models, and provide early warning and result evaluation of malignant diseases. This greatly solves the above contradictions and problems, and makes great contributions to the prevention and detection of breast malignant tumor in the future.

1.6 Motivation

Physical and mental health is necessary for normal study, work and life. Without a healthy body, you will not be able to maintain sufficient energy for a long time, and you may even suffer from certain diseases and cannot perform normal social activities. Malignant tumors of the breast are an incurable disease that has had a great impact on many people's bodies, leading to negative emotions in many patients. In addition to the psychological impact on the patient, it also has an impact on life, which cannot be successful in study, work and life. Therefore, health is very important [11]. This project can not only reduce the time to judge the health status, but also greatly improve the convenience of busy people who do not have time to go to the hospital for detailed examination, so that patients can get timely treatment as soon as possible. At the stage of the disease, not make the condition worse and cause more harm. People can get feedback in a short time and get more detailed treatment in the hospital to prevent missing the best time for diagnosis and treatment.

1.7 Evaluation Plan

The main purpose of this project is to compare original and proposed method architecture and identify the accuracy of breast cancer, we will use the following hyperparameters to evaluate the architecture, to show the architecture more clearly, we will use accuracy Equation 1.1, precision Equation 1.2, recall Equation 1.3, activation function, learning rate,

epoch, loss function, optimization function, confusion matrix, loss, F1 score, specificity Equation 1.5, sensitivity Equation 1.4 to analyze the original and proposed method architecture.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{1.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{1.3}$$

$$Sensitivity = \frac{\text{True Positive}}{\text{Positive}} \tag{1.4}$$

$$Specificity = \frac{\text{True Negative}}{\text{Negative}} \tag{1.5}$$

Among them, True positive (TP) is actually a positive sample and you are predicting a positive sample; False negative (FN) is actually a positive sample and you are predicting a negative sample; False positive (FP) is actually a negative sample and you are predicting a positive sample; and True negative (TN) is actually a negative sample and you are predicting a negative sample.

In our experiment, the accuracy and recall affect each other. In an ideal state, we hope to get a high accuracy and recall, but the actual situation is that these two cannot achieve the best at the same time, they will restrict each other. If you want a high accuracy in the experiment, the recall is low; on the contrary, the pursuit of a high recall usually affects the accuracy. Therefore, considering the actual situation, we are dealing with the medical disease detection, so we must consider increasing the recall rate if we want to ensure accuracy.

1.8 Paper Goals

The goal of the neural network is to build a model that can predict human health problems based on the input of picture features. On the basis of other aspects, we hope to find a better way to form a new architecture and have good performance in breast tumors. Finally, the architecture was optimized so that higher and better accuracy can be achieved, thereby creating a new architecture. At present, we are using the BreakHis breast disease data set, and the final accuracy of this data set is 98.48%. The association between the data set and the sentiment scale leads us to believe that better accuracy can be achieved through more tests.

CHAPTER 2

LITERATURE REVIEW

With the development of the computer industry, research combining the medical field and computer technology has a long history. The great success of deep learning in the field of computer vision has inspired scholars around the world to apply it in the field of medical analysis. Professor Wells of the Harvard Medical University pointed out in [12] that the application of deep learning to solve medical image analysis tasks is a development trend in this field. Deep learning technology is closely connected with medical image recognition and disease recognition. In order to research and find more advanced recognition assistance methods and innovative frameworks, researchers have conducted a lot of work and experiments. Since 2016, a number of medical experts have summarized, commented, and discussed the research status and problems of deep learning in medical image analysis [13] [14] [15] [16] [17]. In the medical field, the application of deep learning algorithms in the normal work of physicians has made rapid progress in medical academic and commercial aspects. Various publications, academic articles, magazines and journals have shown a rapid upward trend year by year. By 2019 alone, more than 10,000 academic articles will be published on PubMed (see Figure 2.1). In addition, the review published by Medical Image Analysis provides a more comprehensive summary and summary of the research on deep learning in medical image classification, detection and segmentation, registration and retrieval [18].

2.1 Convolutional Neural Network (CNN) Based Methods Compare

CNN is one of the most popular deep learning algorithms. Deep learning is a type of machine learning. The model of this algorithm directly learns and performs classification tasks from images, videos, texts or sounds. It provides an end-to-end deep learning model. The

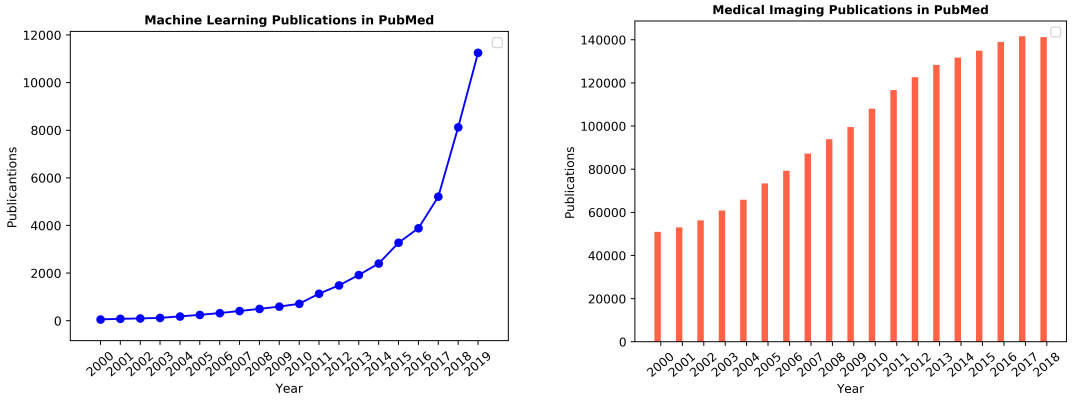


Figure 2.1: Machine learning and medical imaging publications in PubMed by year through 2019 showing the exponential growth of interest in the field, as reported by the US National Library of Medicine of the National Institutes of Health [19].

parameters in the model can be trained by traditional gradient descent methods. The trained CNN can extract the features in the image and complete the extraction and classification of the image features. And there is no need to preprocess the image and train the image features. Because it is particularly suitable for classifying images using features to identify objects, faces and scenes, and does not require manual extraction.

CNN is a special deep feedforward network, which usually includes input layer, convolutional layer, pooling layer, fully connected layer and output layer. However, in the network structure, in order to make the output more accurate and feature extraction more abundant, usually the network model uses a network model that combines multiple convolutional layers and multiple pooling layers [20].

As early as the early 21st century, AlexNet proposed by Krizhevsky [21] won the first place in the ImageNet image classification competition. After AlexNet, new convolutional neural network models have emerged, such as the VGG model of Oxford University. At present, its network depth has been increased to 19 layers. The model is characterized by widening and deepening the network structure. In 2014, Google also introduced the GoogLeNet [22] model, which initially consisted of three layers of ordinary convolutions, followed by three sets of subnetworks, three of which had 2, 5, and 2 initial modules. Finally, there are the average pool layer and the complete connection layer. Later, Microsoft's

ResNet model training fusion was faster, and successfully trained hundreds or nearly thousand layers of CNNs. As the depth of the convolutional neural network deepens, more attention needs to be paid to maintaining the shortening and accuracy of the training network time, so that it can better adapt to the data set required by the actual application and improve the classification effect. From the research point of view, the future of convolutional neural networks is full of infinite possibilities.

Convolutional neural networks (CNN) are based on LeNet [23], AlexNet [21], ZFNet [24], VGG [25], Inception [22] [26], ResNet [27], Inception-ResNet [28], Xception [29], DenseNet [20] and NASNet [30]. The following table comprehensively compares the more classic CNN model Table 2.1.

2.2 Computer-Aided Diagnosis (CAD)

Deep learning technology can solve many medical problems to a certain extent. Although the results also have many problems, such as poor data quality, patient privacy, and improper supervision, the development of deep learning in the medical field has become unstoppable. Through the long-term exploration and development of many researchers and companies, with the help of human medical experts, tumors, cardio-cerebrovascular, neurology, facial features and other research fields have made good progress, and scientific research results continue to emerge. In order to reduce the misdiagnosis and misdiagnosis in the diagnosis process, and at the same time improve the specificity of detecting cancer, relevant experts have proposed the method of second reading [31]. The specific method is that two doctors diagnose the same breast X-ray picture separately, and combine the diagnosis results of the two doctors to give the final result, but the disadvantage of this method is that it will consume higher costs. With the reform and prosperity of information technology, the integration of computers into medical treatment has appeared in people's lives, that is, computer-aided diagnosis technology.

Table 2.1: Comparative analysis of improved CNN model

Model	Technology	Structural Features
LeNet-5	Relu; Softmax regression	Structure is simple; Model depth is shallow; Image feature is bad; Overfitting
AlexNet	Relu; Dropout technology; Data enhancement; Multi-GPU parallel training	Avoid overfitting; Convergence speed stable; Speed is faster; Calculation increases; More parameters
ZF-Net	Relu; Dropout technology; Data enhancement; Multi-GPU parallel training; Smaller filter; Softmax regression	Adjusted parameters; Stronger than AlexNet
VGGNet	Relu; Dropout technology; Data enhancement; Multi-GPU parallel training; Softmax regression; 1*1, 3*3 small convolution kernels	More judgmental; Fewer parameters; Large amount of calculation
GoogLeNet	Relu; Dropout technology; Data enhancement; Multi-GPU parallel training; Inception structure; Softmax regression	Reduce calculations; Fewer parameters; Replace all fully connected layers with simple global average pooling;
ResNet	Relu; Multi-GPU parallel training; Residual blocks; Average pooling; Softmax regression	Directly pass the input information to the output to protect the integrity of the information; Entire network only needs to learn the input and output, simplifying the learning goals

Computer-aided diagnosis (CAD) [31] [32] [33] technology refers to the combination of medical imaging technology, image processing technology and computer technology to help doctors diagnose the diseased area, reduce the probability of misdiagnosis and missed diagnosis, and improve the sensitivity and accuracy of the diagnosis of the diseased area. In the CAD system, the function of the computer is mainly to acquire, process, display and understand the input medical data. Adding a CAD system to the clinical diagnosis of breast cancer can not only maintain false positives, but also improve the accuracy [33].

Giger et al. [34] combined breast cancer and CAD systems to increase the specificity of the diagnosis by 10%, while at the same time, the sensitivity also increased by 14%.

Kumar et al. [35] [36] proposed a CAD system for detecting liver cancer in 2013. Among them, the CAD system is an effective computer-aided tool. Its functions include preprocessing input medical data, fully automatic separation of diseased parts, then feature extraction, and finally a final judgment based on the processing results, whether it is benign or malignant. Finally, a very high accuracy is obtained, reaching 96.7%.

Vincey Jeba Malar et al. [37] proposed a CAD system for liver cancer detection in 2013. The main optimization part is feature extraction. In order to avoid the shortcomings of traditional algorithms, it is proposed to use the Hidden Markov Model (HMM) to complete the experimental project, and a test rate of 96.5% is obtained.

Priyanjana et al. [38] proposed in 2013 to classify four CAD systems related to liver diseases based on CT images. In the experiment, five different feature texture analysis methods were used: first-order statistics (FOS), legal texture energy measurement (TEM), spatial gray-scale dependent matrix (SGLDM), fractal dimension measurement (FDM) and gray scale Difference matrix. For comparison and experimental purposes.

Pereira et al.[39] used the convolutional neural network architecture in the field of disease exploration and segmentation, tried a small kernel, deep network architecture, and used data processing methods such as grayscale normalization and data enhancement to The enhanced part and core part of the imaged brain tumor were segmented, and won the

first place in the 2013 public challenge.

2.3 Breast Cancer Detect Method

With the development of computer technology, computers can also perceive things that humans can perceive, such as the identification of people, the identification of pictures, the judgment of diseases, etc. According to the development of artificial intelligence, artificial neural networks are the most important findings. However, with the development of science and technology, artificial neural networks will not stop at the most basic technology but will propose a deeper learning structure on this basis [40]. From the development of many years, the use of deep CNN to detect cancer diseases is very popular among the public, and there are many examples. Biomedical image analysis has always been an important research area in deep learning. In this way, diseases can be detected early and preventive measures can be taken. Disease detection is usually preferred by viewing computer tomography images. Computer-aided diagnosis (CAD) systems often rely on machine learning techniques to detect cancer. The different methods and researches that have been published on breast cancer recognition are:

Ismail et al. [41]. used the IRMA dataset to detect breast cancer, and evaluated the VGG16 and ResNet50 deep learning model networks. The results showed that VGG16 performed better, reaching 94%. But no improvement was made on this architecture, only make the comparison of these two architectures.

Bayramoglu et al. [42]. proposed a multi-task CNN architecture to use BreaKHis dataset for image classification of breast cancer, and the recognition rate is about 83%. This method is different from the method based on manual features, and can benefit from the additional label training data in a direct way. But this method still needs to improve the accuracy.

Teresa Araujo et al. [43]. proposed a method based on deep learning, using CNN algorithm and CNN + SVM algorithm to classify breast images, achieving an accuracy of

83.30%. Comparing the accuracy of the two methods, CNN + SVM performs better than the CNN. The accuracy of the results needs to be improved, and more advanced techniques can be combined to improve the accuracy.

Jongwon et al. [44]. proposed to use BreaKHis database to detect breast cancer based on Google Inception v3 model. The results show that the AUC of transfer learning is 0.89. The classification accuracy of the model for benign tumor is 0.83, and the classification accuracy for malignant tumor is 0.89. However, this work has not been optimized, just compare.

Abirami et al. [45] Classify breast using wavelet features and obtain a high accuracy of 93%.

Uppal and Naseem [46] used a combination of discrete cosine transform and discrete wavelet transform to classify mammograms into three categories.

Sahiner et al.[47] used convolutional neural networks in medical image processing in 1996. In this work, the researchers extracted texture features of target tissues from breast, and then applied volume the product neural network is used for classification. The network contains only one input layer, two hidden layers and one output layer, which realizes the detection of tumors and normal tissues.

In [48], the CNN model is used to segment breast images for more accurate analysis.

Jamieson et al. [49] The adaptive deconvolution network (ADN) is used to simplify the characteristics of breast cancer into two distinctions between malignant and benign.

Mert et al. [50] For the two classification methods, radial basis function neural network (RBFNN) with independent component analysis (ICA) was proposed to analyze breast cancer.

Mahboubeh et al. [51]. proposed to use Inception and ResNet for detection on the TMA dataset and BreaKHis dataset. The results show that the accuracy rates are both between 90% to 93%. The ResNet framework detects cancer up to 98.7%.

Ankit et al. [52]. proposed to use AlexNet to analyze breast cancer. Using the BreakHis

dataset, the accuracy range is between 93.8% to 95.7%. This scheme is a simple method that combines the transfer learning method and is only classified.

Wisconsin's breast cancer diagnosis data set is a numerical data set obtained by the fine needle aspiration (FNAC) method. Singh S. et al. [53] used the data in this dataset for analysis and used a convolutional neural network model for classification, with an accuracy rate of 96%.

CHAPTER 3

METHOD

3.1 ResNet-50

Breast cancer image classification and recognition research itself is a binary classification study. The model used in this experiment is a convolutional neural network. In theory, the more network layers, the better the extracted features. If you want to further improve the accuracy of the model, the most direct method is to design the deepest network. The effect of the model will get better and better. However, the actual situation does not allow this, such as the LeNet model has only 5 layers, the AlexNet model has 8 layers, and the VGG-16 model has 16 layers, all of which have achieved good results in the ImageNet competition. These network models do not have hundreds of layers but still get such good results. This is because the directly stacked network, when the network reaches a certain depth, the model's effect is getting worse and worse, and the model becomes difficult to train.

In our experiment, the network model we adopted is the ResNet-50 model, because the ResNet-50 model has excellent results in image classification. Other convolutional neural network structures have no way to determine the optimal number of network layers, and the ResNet-50 model can well avoid the process of finding the optimal network layer. Its main solution is to solve the problem of gradient disappearance or gradient dispersion with the increase of depth in the convolutional network. ResNet can simplify this problem and bring excellent results. The training of convolutional neural networks is based on the chain rule. As long as one of the multiplication factors is close to zero or infinite, the final result will be close to zero or infinite. This will form a problem that the network is difficult to train or improve, and the effect will be counterproductive.

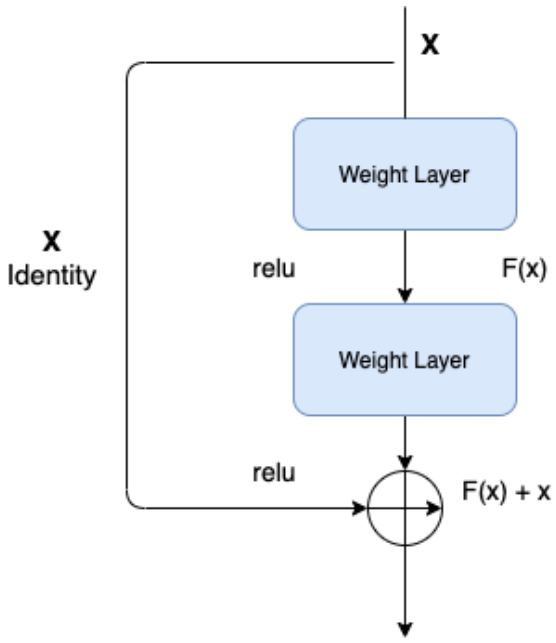


Figure 3.1: Residual block of deep residual network.

It can be clearly seen from the residual block shown in Figure 3.1 that the structure of the residual block can solve the problem of gradient disappearance or gradient dispersion caused by the chain rule from the source. For the residual structure, there is the following relationship as Equation 3.1:

$$F = W_2\sigma\left(W_1x\right) \tag{3.1}$$

Where σ represents the activation function ReLU, and the output is obtained through a residual unit structure and a second activation function as Equation 3.2:

$$y = F\left(x, \{W_i\}\right) + x \tag{3.2}$$

As can be seen from the formula, this greatly solves the contradiction of the traditional convolutional neural network, and ResNet uses batch normalization and ReLU activation unit, which reduces the training difficulty of ResNet, so we chose this network model for our experiment.

The characteristic of ResNet model:

1) Increase the network layer and improve the network segmentation accuracy.

2) More jump connections can be added in the middle of the network, which can better combine the background semantic information of the image to perform multi-scale segmentation.

3) ResNet has the advantages of rapid convergence and reducing the amount of model data.

4) ResNet makes the model easier to train, which prevents the model from degenerating, but also prevents the gradient from disappearing, and the loss does not converge.

3.2 Gentic Algorithm

Evolutionary algorithms are a large category, for example, includes genetic algorithms, genetic programming, evolution strategies, and evolution programming methods. What we want to use to improve our proposed method is the genetic algorithm. Evolutionary algorithm (EAs), it is not a specific algorithm, but a general name and basis for a class of algorithms based on Darwin's theory of evolution. It does this by simulating the evolution of biology in nature, reproduction, mutation, competition and selection. Correspondingly, the algorithm will also generate operations such as genetic coding, population initialization, cross-mutation operators, and management retention mechanisms [54]. Among the many calculation methods, evolutionary calculation is a special calculation method with high performance, so this is why we choose this optimization algorithm, it can be applied to a variety of architectures and models, and has real-time evolution And excellent robustness features. In addition, it can also self-adjust, so as to be more appropriately integrated into the new architecture or model. And, to complete more powerful improvements, you can also deal with ways that traditional methods cannot.

In the application process, it can give a coding scheme to the entire parameter space instead of processing specific parameters, that is to say, instead of searching from a single initial point, it automatically adjusts the algorithm control parameters and coding accuracy. Therefore, evolutionary algorithms are widely applicable, highly nonlinear, easy to modify, and parallelize.

Compared with ordinary search methods, evolutionary computing is an iterative algorithm. The difference is that in the search process, evolutionary computing starts from a set of solutions to a problem and improves to another set with better results. The group issued further improvements. Imitating biological inheritance methods, mainly adopting a series of three operations of replication, exchange and mutation to derive the next generation of individuals. Then, according to the size of the fitness, the survival of the fittest will be improved, and the quality of the new generation group will be improved. After repeated iterations, the optimal solution will be obtained. From a mathematical point of view, the evolutionary algorithm is essentially a method of searching for optimization, as shown in Figure 3.2.

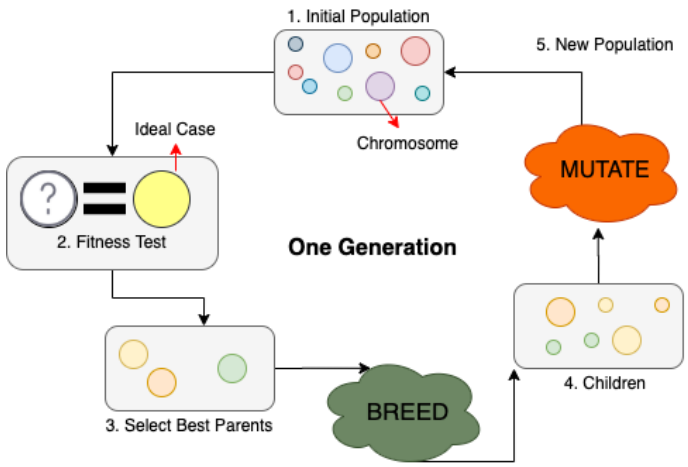


Figure 3.2: Simplify the basic operation process of the evolutionary algorithm.

Firstly, propose a complete plan; evaluate the advantages and disadvantages of the plan; select a part of the plan as the basis for the next steps; iterate and get results; after reaching the goal, you will get excellent results, otherwise repeat the iterative steps, looking again

the better plan. [55]. Evolutionary computing is a very promising method and is used in many professional applications, such as pattern recognition, image processing, artificial intelligence, economic management, mechanical engineering, electrical engineering, communications, and biology [56]. The development of evolutionary computing is very rapid and has been widely recognized by academia.

3.2.1 NNI (Neural Network Intelligence) Framework

In the actual operation of machine learning, a lot of manual intervention is usually required, for example, feature extraction, model selection, parameter adjustment, etc. When we want to create an optimized architecture, we need to make appropriate adjustments in the above aspects and achieve more excellent results and architecture. The emergence of the NNI framework helps researchers to no longer repeat parameter adjustments or try various combinations of hyperparameters, but can directly help researchers automatically obtain optimized architectures, which relate architectures to features, models, optimization, and evaluation. The important steps are to learn automatically, without human intervention, to obtain high-quality models to achieve the results we want.

NNI is Microsoft's open source AutoML framework for deep neural networks [57]. It is now an automated toolkit provided by Microsoft. By combining the NNI framework into different neural network architectures, it can automatically use a variety of tuning algorithms to distinguish the best performing neural network and hyperparameters, and support different operating environments such as stand-alone, local multi-machine, cloud. It is very conducive to the optimization of architecture under the condition of limited conditions. In addition, it also provides a developer environment to facilitate users to debug NNI.

Compared with other automated machine learning tools, the advantages of the NNI framework are [57]: 1. NNI supports most mainstream deep learning frameworks. 2. A large number of tuning algorithms, with very good flexibility. 3. Suitable for developers at all levels.

3.3 Datasets

The rapid development of medical imaging and recognition technology reflects the strong demand for medical information acquisition. Compared to the language or text on the case, the image can carry more information. Medical imaging can provide a wealth of information, and its role in medical diagnosis is increasingly prominent. However, it is difficult to judge cancer by machine:

1) Data source: Cancer images are an important branch of medical image data. They are built in the data collection of hospitals or medical research for many years. The acquisition channels are special, and the process is complicated. Therefore, a set that can fully cover the types of cancer cases and is marked by professionals. Tracking research data is a costly task.

2) Computer processing: In the data obtained, the normal samples are usually the majority, the pathological samples are few, and the number of positive and negative samples is unbalanced, resulting in most of the time spent in training on the normal samples, which is a waste of time and may lead to overfitting. In order to apply existing small-scale data sets more efficiently, a lot of parameter tuning work is required.

With the development of imaging technology, medical images are widely used in various fields of medical research and diagnosis. Common medical images include computed tomography (CT) [58], magnetic resonance imaging (MRI) [58], ultrasound images [59], endoscopic images [60] [61] [62], and microscope images [63]. From the perspective of imaging signals, endoscopic images and microscope images are visible light imaging, and usually have better resolution ability in details such as color and texture; CT images, nuclear magnetic resonance images, and ultrasound images are non-invasively obtained internal tissue images. Belong to non-visible light reconstruction imaging.

Microscopic image refers to cutting a certain size of diseased tissue, using hematoxylin and eosin (H&E) and other staining methods to make the sliced tissue into a pathologi-

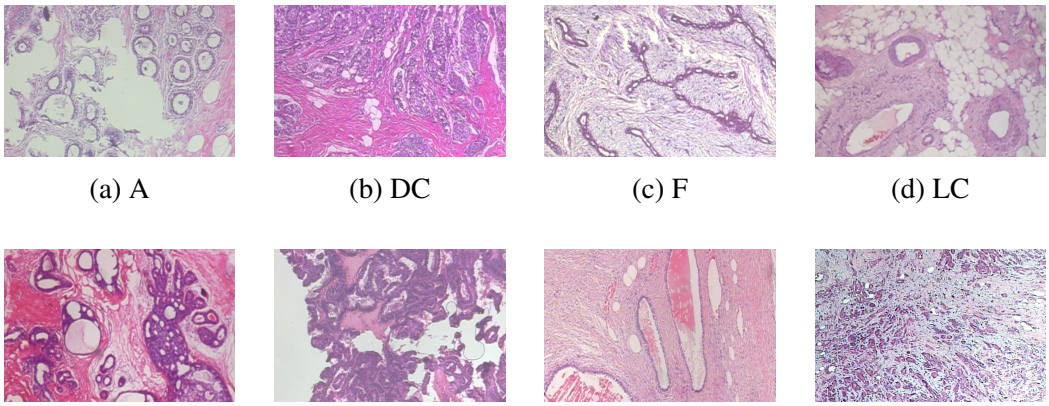
cal slide, and then using microscopic imaging technology to image microscopic cells and glands. By analyzing pathological pictures, the causes, pathogenesis, and pathogenesis of lesions can be explored to make pathological diagnosis. The use of convolutional neural networks to train medical images requires high data quality, and deep convolutional neural networks extract high-latitude features, which can easily lead to redundancy and affect the training results of the classifier. [18] So, we chose the BreakHis dataset as our experiment.

3.3.1 BreakHis Dataset

Breast Cancer Histopathology Image Classification (BreakHis) was established in cooperation with P & D Lab-Pathology Anatomy and Cytopathology in Paraná, Brazil. It consists of 7,909 microscopic images of breast tumor tissue collected from 82 patients using different magnifications, such as 40X, 100X, 200X, and 400X in Table 3.1. Currently, it contains 2480 benign and 5429 malignant samples. The format of the data is 700X460 pixels, 3 channels of RGB, and 8-bit depth per channel. It is a dataset with PNG format. The dataset BreaKHis is divided into two main categories: benign tumors and malignant tumors. Benign means that the lesion does not meet any malignant criteria, such as obvious cell atypia, mitosis, basement membrane destruction, metastasis, etc. Under normal circumstances, benign tumors are relatively safe. But a malignant tumor is the cancer that we will detect. Its lesions can invade and destroy adjacent structures and spread to a distance, eventually leading to the death of people.

Table 3.1: The detail structure of BreakHis dataset, include 7909 images of breast tumor tissue.

Magnification	**Benign**	**Malignant**	**Total**
40X	652	1370	1995
100X	644	1437	2081
200X	623	1390	2013
400X	588	1232	1820
Total of Images	2480	5429	7909



(a) A (b) DC (c) F (d) LC

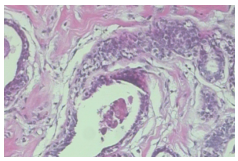
(e) MC (f) PC (g) PT (h) TA

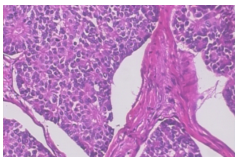
Figure 3.3: 40X images of eight different breast tumor tissue.

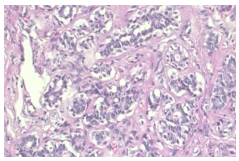
In the current version, the samples present in the dataset are collected by the SOB method, also known as partial mastectomy or resection biopsy. Compared to any needle biopsy method, this type of procedure can remove larger sized tissue samples. The benign and malignant breast tumors can be divided into different types according to the way the tumor cells are observed under the microscope. The data set currently contains four histologically different types of benign breast tumors: adenoma (A), fibroadenoma (F), phyllodes tumor (PT) and renal tubular adenocarcinoma (TA); breast cancer: carcinoma (DC), lobular carcinoma (LC), mucinous carcinoma (MC) and papillary carcinoma (PC) in Figure 3.3 Figure 3.4 Figure 3.5 Figure 3.6. To label each image, they have unique file information, as biopsy method, tumor classification, tumor type, patient identification, magnification.

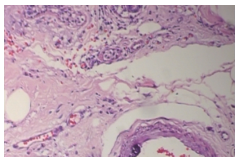
The optical technical parameters of the microscope include numerical aperture, resolution, magnification, depth of focus, field of view width, working distance, poor coverage, etc. Not all of these parameters are higher, the better. They are interrelated and restrictive. The actual use should be adjusted according to the actual situation.

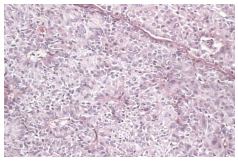
Regarding the magnification of the microscope, it is not the greater the better, we need to choose the most appropriate size for the diagnosis of breast tumors. The larger the magnification of the microscope, the smaller the field of view, the larger the cells you see,

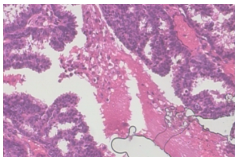

(a) A

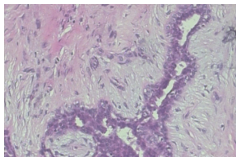

(b) DC


(c) F


(d) LC


(e) MC


(f) PC


(g) PT

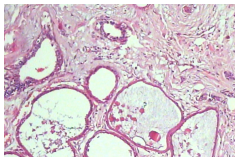
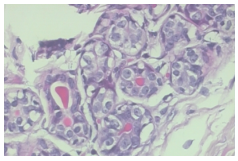
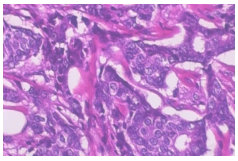
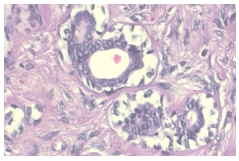
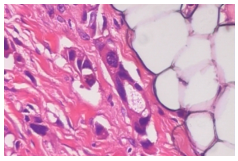

(h) TA

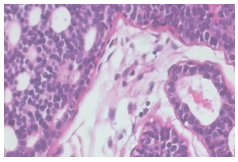
Figure 3.4: 100X images of eight different breast tumor tissue.

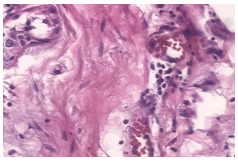

(a) A

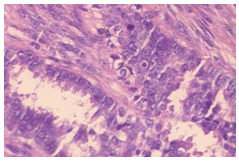

(b) DC


(c) F


(d) LC


(e) MC


(f) PC


(g) PT

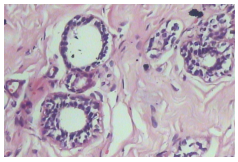
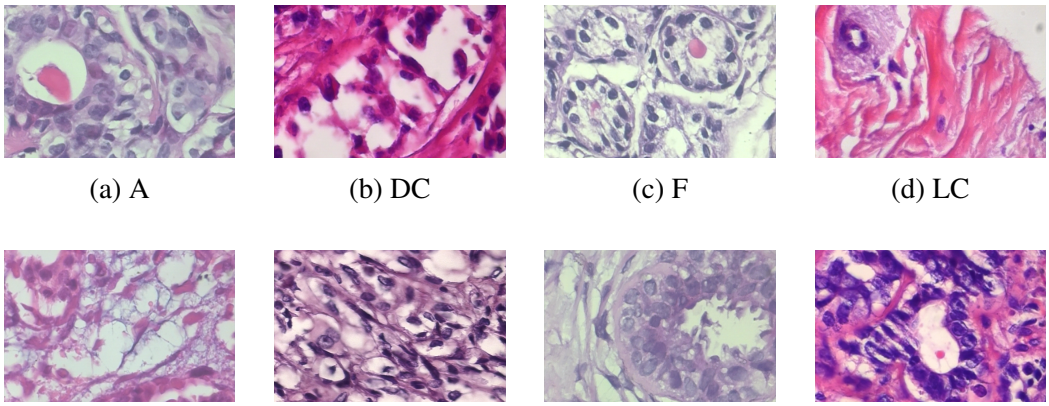

(h) TA

Figure 3.5: 200X images of eight different breast tumor tissue.

but the smallest number; the smaller the magnification of the microscope, the larger the field of view, the smaller the cells you see, but the largest number; in order to see Be clear about the object to be observed. If the microscope magnification is larger, the structure to be observed may not be in the field of view. Therefore, it is not better to enlarge the magnification. It is necessary to adjust it according to the needs and choose the appropriate magnification. Therefore, in our experiments, we will also choose the most appropriate method for diagnosing breast tumors.



(a) A (b) DC (c) F (d) LC

(e) MC (f) PC (g) PT (h) TA

Figure 3.6: 400X images of eight different breast tumor tissue.

CHAPTER 4
EXPERIMENT AND ANALYSIS

In our project, we will conduct model building and evaluation according to the following steps:

- Step 1: Data analysis and preprocessing.
- Step 2: Create the model.
- Step 3: Model training.
- Step 4: Model evaluation.
- Step 5: Architecture optimization.

In the experiment, the entire task was completed using the keras framework and tensorflow-gpu. In addition, we will introduce matplotlib, pandas, numpy in python libraries; sklearn libraries to help us complete experiments.

4.1 Data Pre-processing

Regarding the data processing part, the auxiliary diagnosis of medical diseases requires a large amount of data, and it also needs to be updated at any time. However, at present we have not been able to meet such standards and requirements. Therefore, I chose the dataset with guaranteed quality and quantity, the BreakHis dataset. But this dataset is not perfect, so we need to process it in a unified standard before using it. Because in the neural network model, in most cases, people think that different types of data are evenly distributed, and many algorithms are also based on this assumption. But under real circumstances, this is often not the case. For example, the situation that the machine sends a failure is what we

want to predict, but in fact the probability of failure is very low, so the sample size that causes the failure is very small. Even if you set all the prediction results to normal, the accuracy rate is still very high, but this model is a useless model, and similar examples are very common. Therefore, we consider pre-processing in the dataset. Our experiments are mainly from the data level.

The first is the environment used in our experiment, which requires Python3, tensorflow-gpu and keras. Based on the needs of our experiment, we use ImageDataGenerator to process pictures in batches. It is a picture sample intensifier. The image generated after each sample iteration is a modified picture to achieve the purpose of data enhancement. Batches of tensor image data are generated through real-time data enhancement, and the data will continue to cycle. In addition, it can also expand the size of the data set and enhance the generalization ability of the model. We will optimize the parameters Table 4.1 in the experiment.

Table 4.1: Data processing parameters in ImageDataGenerator.

datagen = ImageDataGenerator
rotation_range = rotation_range,
shear_range = shear_range,
zoom_range = zoom_range,
horizontal_flip = horizontal_flip,
vertical_flip = vertical_flip

4.2 Experimental Environment

In our experiments, for the dataset in Table 4.2, we divided the dataset into 70% training split of 5537 images, 10% validation split of 791 images, and 20% test split of 1581 images.

The following Table 4.3 provides our experimental environment:

Table 4.2: Dataset information.

Name	BreakHis Dataset
Size	4.27G
Training dataset	5537
Testing dataset	1581
Validation dataset	791

Table 4.3: Experimental environment.

Hardware & Software	
Processor	i-8700k @3.7GHz
Memory	64G
Graphics Memory	2080Ti 11G
Operating System	ubuntu
Language	python3.7.3
Library	Keras

4.3 Original ResNet-50

The residual network uses shortcut connections to solve the problem of degradation. Both the training set and the check set prove that the deeper the network, the smaller the error rate. Because for a neural network model, if the model is optimal, then training can easily optimize the residual mapping to 0. At this time, only identity mapping is left, so no matter how much depth is added, in theory the network will always be in an optimal state. Because it is equivalent to all the additional networks behind it will transmit information along the identity mapping, it can be understood that the layers behind the optimal network do not have the ability to extract features, and actually have no effect. In this way, the performance of the network will not decrease as the depth increases.

Therefore, by comparison, we chose the ResNet-50 as our experimental architecture. First, we set the input dataset to a uniform size mode with a height of 115 and a width of 175. Then set the parameters in the model uniformly, and set the channels to 3. In

the original ResNet, for each layer, after each block, the number of channels becomes 4 planes, so the number of input channels for the next block should be 4 planes, so we define self.inplanes = planes * block.expansion. The planes are unchanged, and are always the incoming parameters, such as 64, 128, 256, and 512. It becomes the number of output channels after each block, which means that the number of input channels of the next block changes. The stride parameter passed in when constructing each layer is different. This parameter represents the construction of the first block, but other blocks are not affected by stride. In addition, it may affect the size of the feature map at this time.

In the ResNet-50 architecture, like Figure 4.1, the first layer: a total of 3 blocks, each block has 3 layers, a total of 9 layers, are not affected by stride, that is stride=1, so the size of the feature map is not changed; the second layer: A total of 4 blocks, each block has 3 layers, a total of 12 layers. The second layer of the first block is affected by stride, which may change the size of the feature map, the remaining 11 layers do not change the size of the feature map; the third layer: a total of 6 blocks, each block has 3 layers, a total of 18 layers . The second layer of the first block is affected by stride, which may change the size of the feature map, and the remaining 17 layers do not change the size of the feature map; the fourth layer: a total of 3 blocks, each block has 3 layers, a total of 9 layers . In order to improve the accuracy and performance of the model, we have added additional layers for classification. We set the parameters of layers to 2048, 512, and 32.

To solve the problem of overfitting, that is to say, the model performs well on the training data, but performs poorly on the test data. The best way is to increase the training data, but in the case of a certain training data, in order to prevent the model from overfitting, we generally use the dropout method, by randomly discarding certain neurons in a layer to achieve the purpose of reducing overfitting. Therefore, we set the dropout layer to 0.3.

The experimental parameters of our project are also given in Table 4.4:

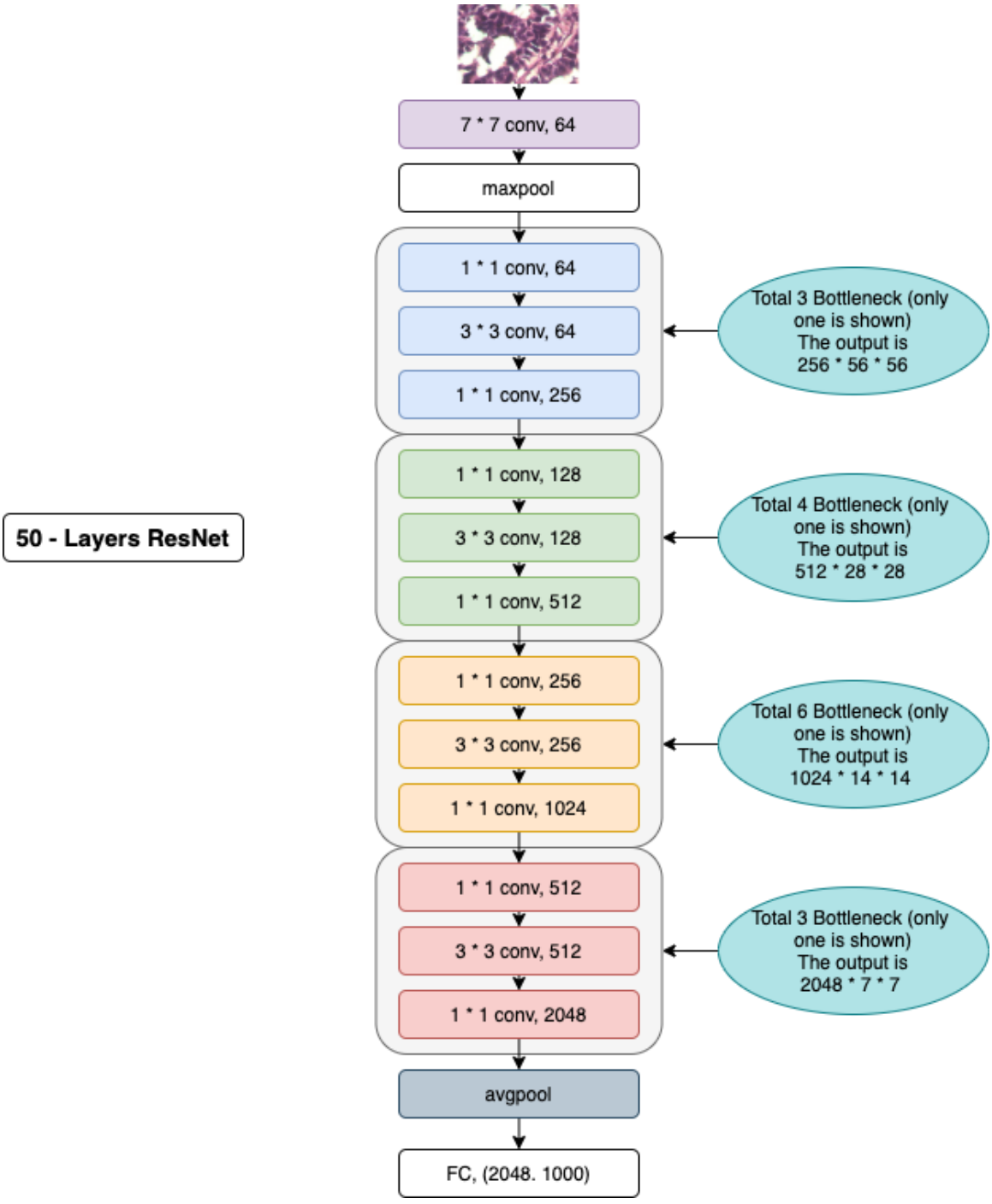


Figure 4.1: 50 layer ResNet architecture.

Table 4.4: Original ResNet experimental parameters.

Parameters	
Activation Function	ReLU
Learning Rate	0.001
Optimizer	Adam
Loss Function	categorical crossentropy
Batch Size	64
Epoch	30
Dropout Layer	0.3

4.4 Proposed Method

Based on the original ResNet-50 architecture, we will optimize the original architecture. For the evolutionary algorithm accepted earlier, we will use naive evolution from the NNI framework to optimize the ResNet-50 architecture. The evolutionary algorithm comes from "large-scale evolution of image classifiers" [55]. It will randomly generate an overall spatial search based on the ResNet architecture. In each generation, better results will be selected, and some mutations will be made to its next generation (for example, changing a superparameter, adding or subtracting one layer). It can find the best combination of parameters and improve the performance of the architecture.

Tuner in NNI framework is an automatic machine learning algorithm that will generate a new configuration for the next trial. The new trial will run with this set of configurations. We chose one of Naïve Evolution (Naive Evolution Algorithm) to implement our experiment. The naive evolutionary algorithm requires many trials to be effective, but it is also very simple and it is easy to extend new functions. This is the method we actually applied to the experiment.

According to the experimental steps, we first install the NNI framework. About the realization of genetic algorithm in NNI framework, all training datasets in the search phase are set to epoch is 30, population size is 20, concurrency is 4 and max iteration number is

100. Therefore, the parameter settings are based on the reference values set by the NNI framework and the content of the literature. The parameters we will search are: batch size, learning rate, optimizer, three classification layer parameters, dropout, horizontal flip, vertical flip, rotation range, shear range, zoom range. Because doing so can get better results, but can control fewer calculations and shorten the search time.

Then perform a neural network search to obtain the best hyperparameters. In this process, in addition to the function of using genetic algorithms, it also has the function of supporting the weight migration model. After the hyperparameter search is completed, we change the hyperparameters in the original ResNet to create a proposed method for the last step. According to the hyperparameter search results Table 4.5, among them, we adjust the newly added three classification layer parameters to 64, 0, 64. There is also a dropout layer set to 0.108875.

Table 4.5: Proposed method experimental parameters.

Parameters	
Activation Function	ReLU
Activation Function of Output Layer	Softmax
Learning Rate	0.0001
Optimizer	Adam
Loss Function	categorical crossentropy
Batch Size	16
Epoch	30
Dropout Layer	0.108875

First, in order to get a better dataset and balanced data, we first adjust the parameters Table 4.6 in ImageDataGenerator to improve the quality of the dataset.

Table 4.6: Proposed method data processing parameters in ImageDataGenerator.

ImageDataGenerator Parameters	
rotation_range	20
shear_range	0.0667
zoom_range	0.4885
horizontal_flip	False
vertical_flip	True

4.4.1 Loss Function

The loss function is one of the most important concepts in machine learning. By calculating the size of the loss function, it is the main basis in the machine learning process and an important criterion to judge the merits of the algorithm after learning. Therefore, I chose cross entropy as the basis for proposed method architecture. Cross entropy is used as a loss function in neural networks. Among them Equation 4.1, y represents the distribution of real labels, a is the predicted label distribution of the trained model, and the cross-entropy loss function can measure the similarity of y and a. Another benefit of cross-entropy as a loss function is to avoid the problem of a decrease in the learning rate of the mean square error loss function, because the learning rate can be controlled by the output error.

$$C = -\frac{1}{n}\sum_{x}[y \ln a + (1-y)\ln(1-a)] \tag{4.1}$$

Among them, the selected categorical crossentropy Equation 4.2 is more suitable for multi-classification problems, and using softmax as the activation function of the output layer is our choice.

$$loss = -\sum_{i=1}^{n} \hat{y}_{i1} \log y_{i1} + \hat{y}_{i2} \log y_{i2} +, \ldots, +\hat{y}_{im} \log y_{im} \tag{4.2}$$

n is the number of samples and m is the number of classifications. Note that this is a

multi-output loss function, so its loss calculation is also multiple in Equation 4.3:

$$\begin{aligned}
\frac{\partial loss}{\partial y_{i1}} &= -\sum_{i=1}^{n} \frac{\hat{y}_{i1}}{y_{i1}} \\
\frac{\partial loss}{\partial y_{i2}} &= -\sum_{i=1}^{n} \frac{\hat{y}_{i2}}{y_{2}} \\
&\cdots\cdots\cdots \\
\frac{\partial loss}{\partial y_{im}} &= -\sum_{i=1}^{n} \frac{\hat{y}_{im}}{y_{im}}
\end{aligned} \tag{4.3}$$

When using the categorical crossentropy loss function, my labels are in multi-category mode. For example, if you have 10 categories, the label of each sample should be a 10-dimensional vector. The vector has a value of 1 in the corresponding index position and the rest is 0. This is how our experiments are conducted.

4.4.2 Activation Function

In order to better improve the performance of the model, we chose different activation functions. For the three classification layers we added, we used the ReLU activation function. For the input layer, we used the softmax activation function. Below I will talk about their advantages.

ReLU Activation Function

Rectified Linear Unit (ReLU) is a commonly used activation function in artificial neural networks [64]. It usually refers to a nonlinear function represented by a ramp function and its variants. Linear rectification is considered to have a certain biological principle, and because it usually has a better effect than other commonly used activation functions in practice, it is widely used in the field of computer vision artificial intelligence such as image recognition by today's deep neural networks [64].

In a general sense, the linear rectification function refers to the ramp function in mathematics, namely Equation 4.4:

$$f(x) = \max(0, x) \tag{4.4}$$

In the neural network, the linear rectification as the activation function of the neuron defines the linear transformation of the neuron $\mathbf{w}^T\mathbf{x} + b$ after the nonlinear output. In other words, for the input vector X from the previous neural network that enters the neuron, the neuron using the linear rectification activation function will output Equation 4.5:

$$\max\left(0, \mathbf{w}^T\mathbf{x} + b\right) \tag{4.5}$$

To the next layer of neurons or as the output of the entire neural network. Compared with traditional neural network activation functions, such as logistic sigmoid, tanh hyperbolic functions, ReLU has the following advantages:

1) More efficient gradient descent and back propagation, avoiding the problems of gradient explosion and gradient disappearance. Moreover, since the gradient of the non-negative interval is constant, ReLU can solve the sigmoid vanishing gradient problem, so that the convergence rate of the model is maintained at a stable state;

2) For linear functions, ReLU is more expressive, especially in deep networks;

3) For non-linear functions, the calculation gradient is super simple, which makes the overall calculation cost of the neural network decrease;

4) Principles of biomimicry: Related brain studies have shown that the coding of biological neurons is usually scattered and sparse. Linearity correction and regularization can be used to debug the activity of neurons in the machine neural network.

Softmax Activation Function

The sigmoid function and softmax function in the activation function are mainly used for the output of the neural network output layer. The softmax function can be regarded as a general case of the Sigmoid function, which is used for multi-classification problems. Because the multi-classification problem is to use the softmax activation function with the classification cross-entropy function to achieve better results, we choose the softmax activation function as our output layer function.

The Softmax function compresses the K-dimensional real number vector into another K-dimensional real number vector, where each element in the vector has a value between (0, 1). Commonly used in multi-classification problems. Suppose we have an array, S, representing the i element in S, then the softmax value of this element is Equation 4.6:

$$S_i = \frac{e^i}{\sum_j e^j} \tag{4.6}$$

4.5 Results

4.5.1 Breast Cancer - 40X Result

For different microscope magnification datasets, we have done separate analysis. Below I will show the results separately, and finally make a comprehensive comparison and summary. In the 40X dataset, using the original ResNet and the proposed method, we compared the loss, accuracy of the training dataset and the validation dataset. The following line chart Figure 4.2, Figure 4.3, Figure 4.4 and Figure 4.5 shows the experiment results. Compared with the original ResNet, the proposed method has greatly improved performance and is a very meaningful experimental project.

Table 4.7, Table 4.8, Table 4.9, Figure 4.6 shows the seven parameters in the original ResNet and proposed method architecture in detail. By comparing the training dataset, validation dataset and testing dataset, the proposed method experimental data has also been

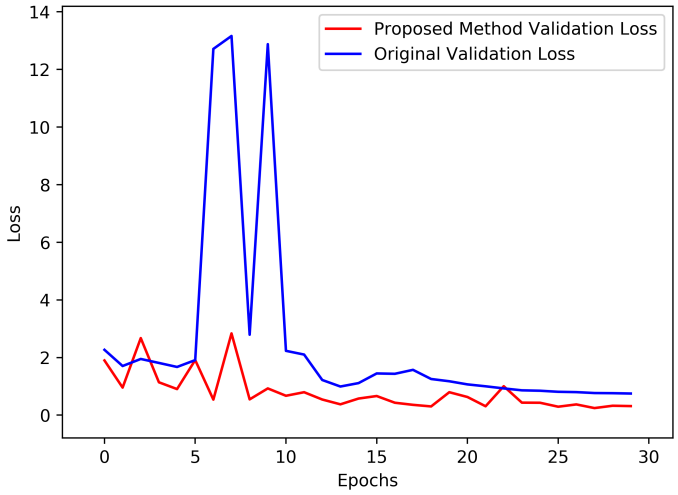


Figure 4.2: Original ResNet and proposed method 40X validation loss compare.

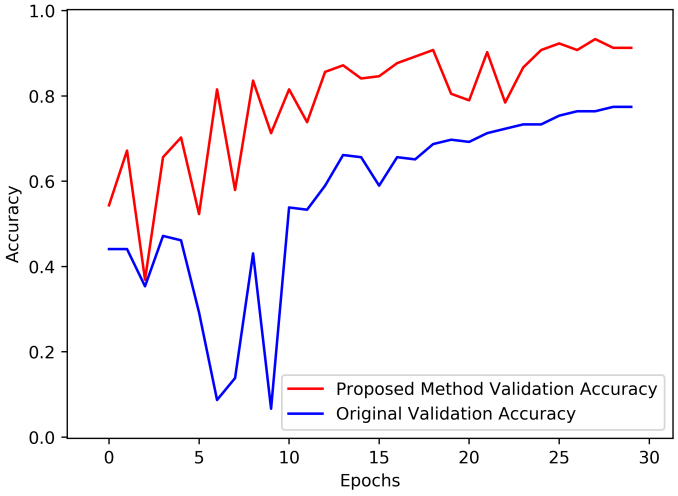


Figure 4.3: Original ResNet and proposed method 40X validation accuracy compare.

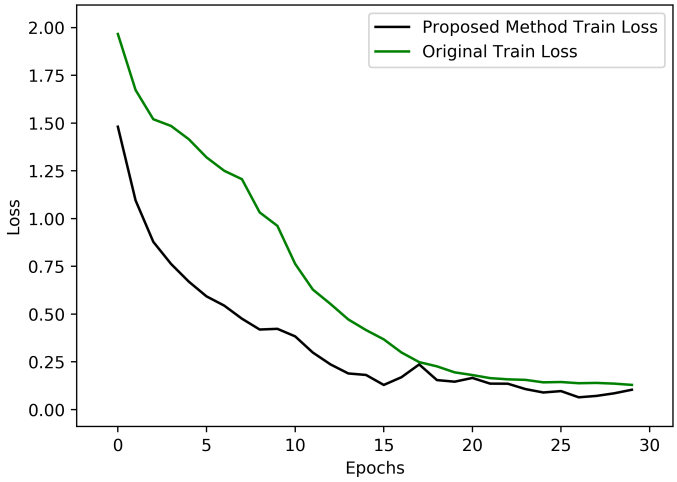


Figure 4.4: Original ResNet and proposed method 40X training loss compare.

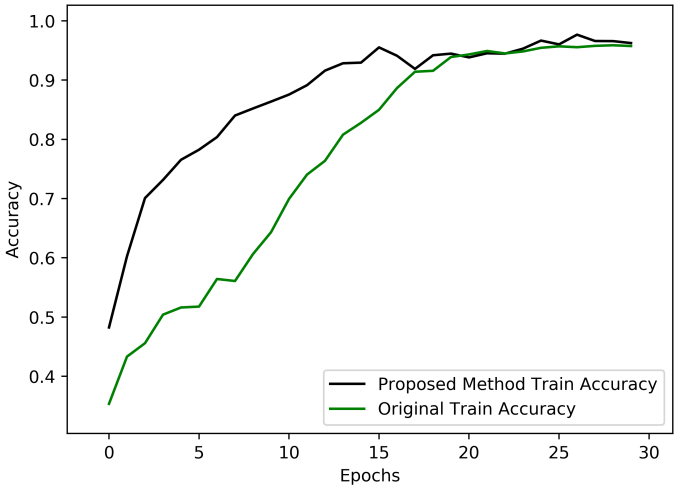


Figure 4.5: Original ResNet and proposed method 40X training accuracy compare.

improved, which proves that our experiment the significance can improve the auxiliary diagnosis of breast tumor diseases. The improvement of its parameters also indicates the success of the experiment. We can intuitively see the results of our test after the training and validation process. The accuracy of the original ResNet model can be achieved 94.17%, while the proposed method model can be achieved 98.48%. It is increased 4.31%.

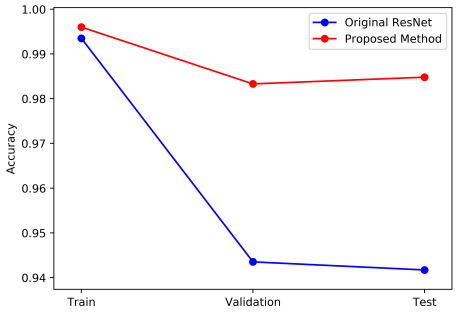
Table 4.7: Hyperparameters compare between original ResNet and proposed method for 40X training dataset.

Hyperparameters	**Original ResNet**	**Proposed method**
Accuracy	0.9935	0.9960
F1 Score	0.9711	0.9859
Precision	0.9765	0.9794
Recall	0.9667	0.9935
Sensitivity	0.9667	0.9935
Specificity	0.9954	0.9977
Correlation coefficient	0.9670	0.9835

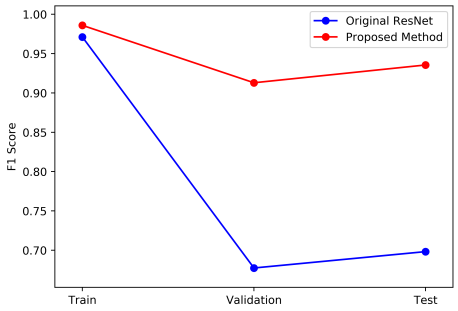
Table 4.8: Hyperparameters compare between original ResNet and proposed method for 40X validation dataset.

Hyperparameters	**Original ResNet**	**Proposed method**
Accuracy	0.9435	0.9833
F1 Score	0.6774	0.9129
Precision	0.7360	0.9061
Recall	0.6770	0.9255
Sensitivity	0.6770	0.9255
Specificity	0.9659	0.9900
Correlation coefficient	0.6592	0.9040

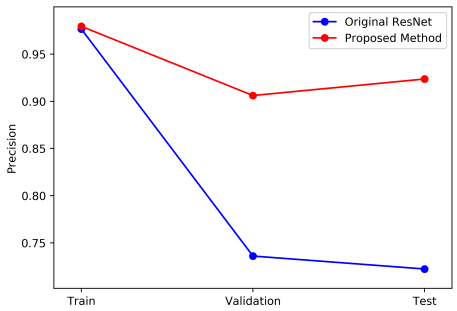
In order to make a more comprehensive comparison, we also conducted a horizontal comparison from different types of breast tumors. The results of training Figure 4.7,



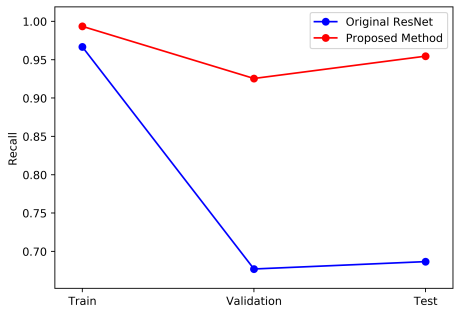
(a) Accuracy



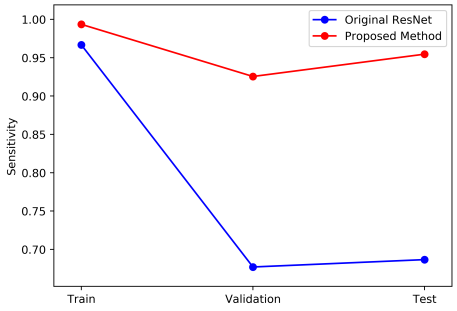
(b) F1 Score



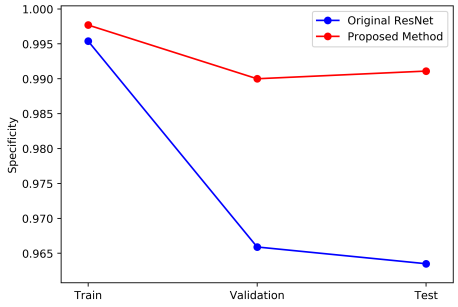
(c) Precision



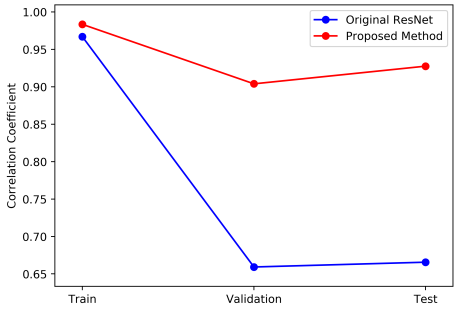
(d) Recall



(e) Sensitivity



(f) Specificity



(g) Correlation Coefficient

Figure 4.6: Hyperparameters compare between original ResNet and proposed method for 40X dataset.

Table 4.9: Hyperparameters compare between original ResNet and proposed method for 40X testing dataset.

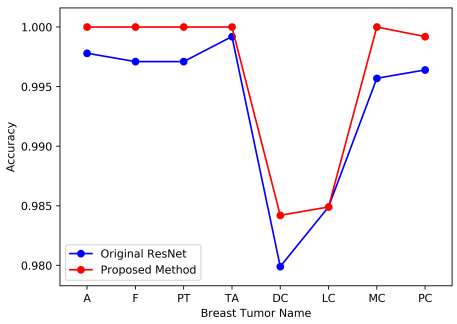
Hyperparameters	**Original ResNet**	**Proposed method**
Accuracy	0.9417	0.9848
F1 Score	0.6982	0.9355
Precision	0.7223	0.9236
Recall	0.6866	0.9546
Sensitivity	0.6866	0.9546
Specificity	0.9635	0.9911
Correlation coefficient	0.6656	0.9276

validation Figure 4.8 and testing Figure 4.9 can clearly show that for four benign breast tumors and four malignant breast tumors, the overall ResNet architecture has been optimized. Performance is better than the original architecture. This phenomenon shows that our proposed method is meaningful, it can effectively help doctors play a certain role in the field of auxiliary medicine.

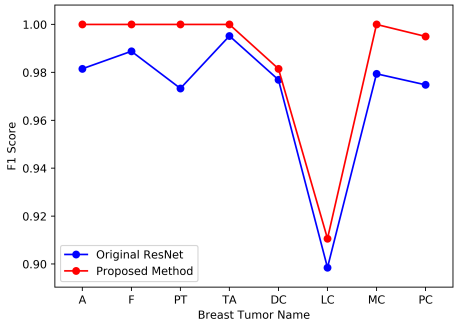
4.5.2 Breast Cancer - 100X Result

In the 100X dataset, using the original ResNet and the proposed method, we compared the loss, accuracy of the training dataset and the validation dataset. The following line chart Figure 4.10, Figure 4.11, Figure 4.12, and Figure 4.13 shows the experiment results. Compared with the original ResNet, the proposed method has greatly improved performance and is a very meaningful experimental project.

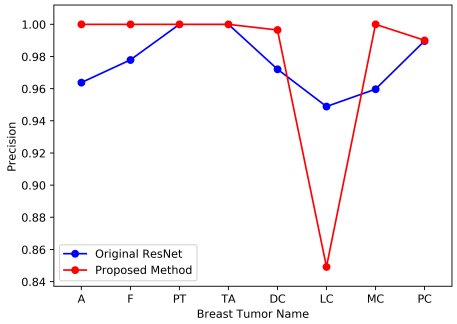
Table 4.10, Table 4.11, Table 4.12, Figure 4.14 shows the seven parameters in the original ResNet and proposed method architecture in detail. By comparing the training dataset, validation dataset and testing dataset, the proposed method experimental data has also been improved, which proves that our experiment the significance can improve the auxiliary diagnosis of breast tumor diseases. The improvement of its parameters also indicates the success of the experiment. We can intuitively see the results of our test after the training and



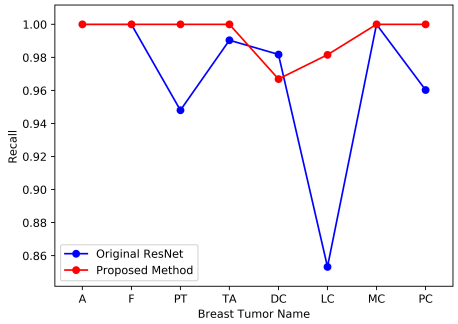
(a) Accuracy



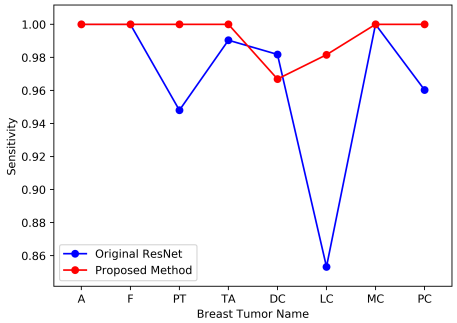
(b) F1 Score



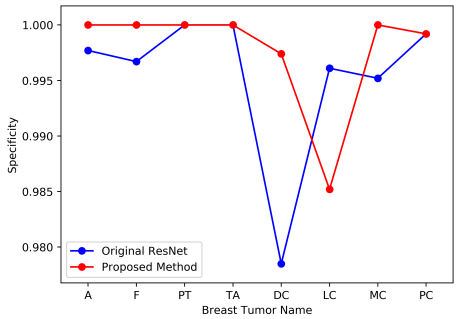
(c) Precision



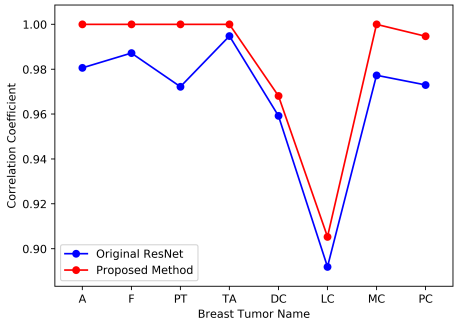
(d) Recall



(e) Sensitivity

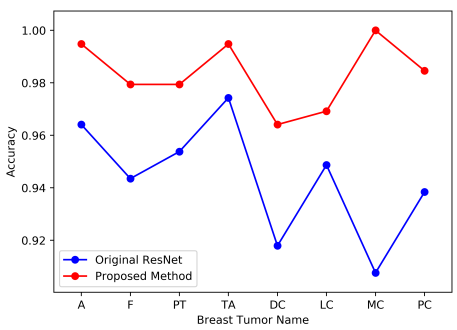


(f) Specificity

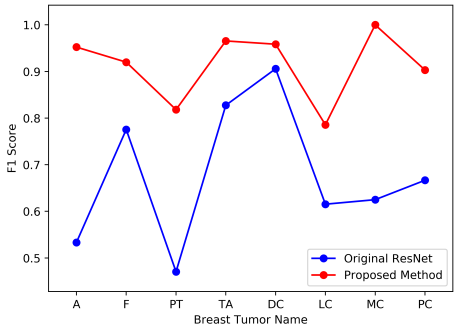


(g) Correlation Coefficient

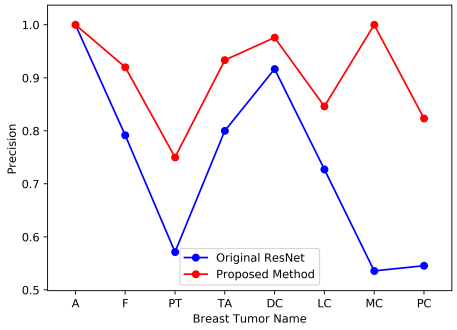
Figure 4.7: Original and proposed method hyperparameters compare between eight breast tumors from 40X training dataset.



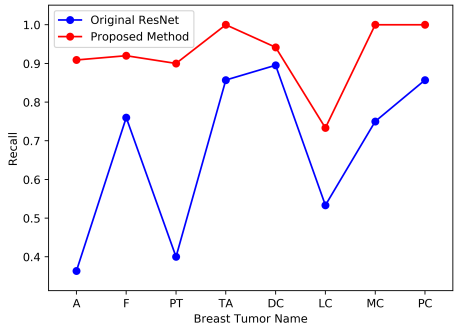
(a) Accuracy



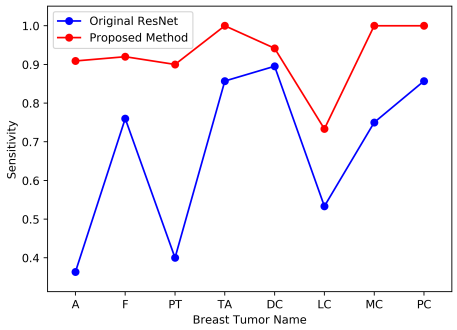
(b) F1 Score



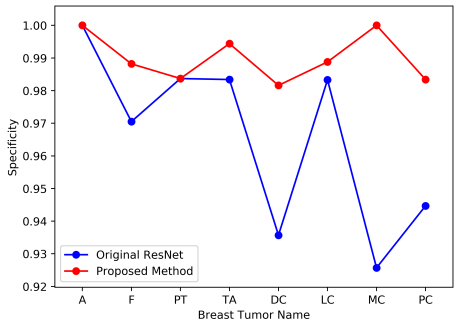
(c) Precision



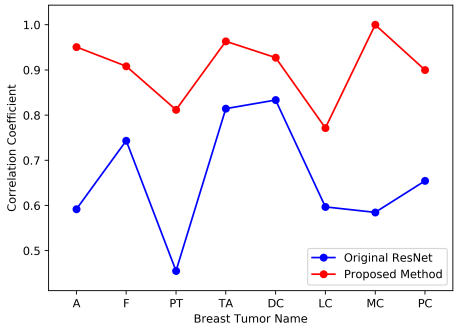
(d) Recall



(e) Sensitivity

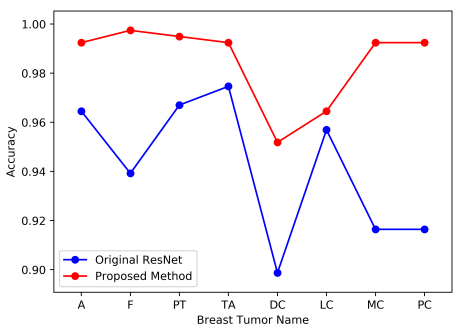


(f) Specificity

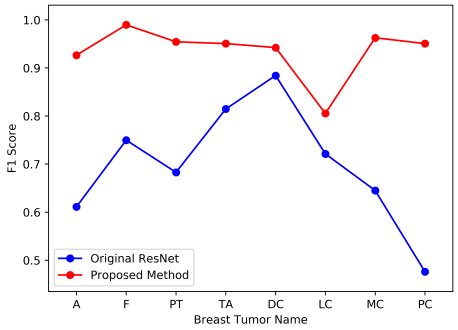


(g) Correlation Coefficient

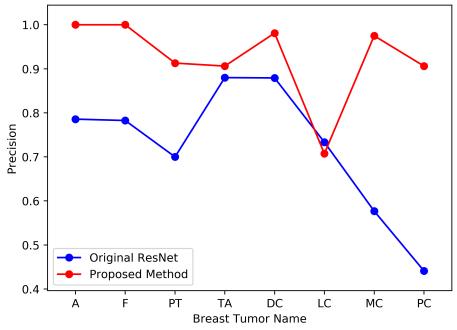
Figure 4.8: Original and proposed method hyperparameters compare between eight breast tumors from 40X validation dataset.



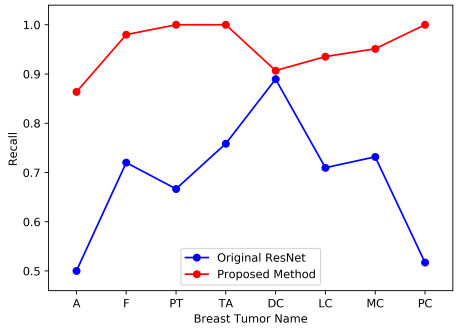
(a) Accuracy



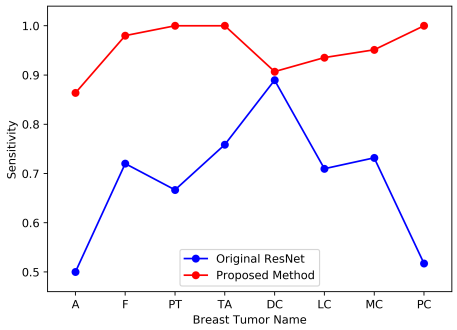
(b) F1 Score



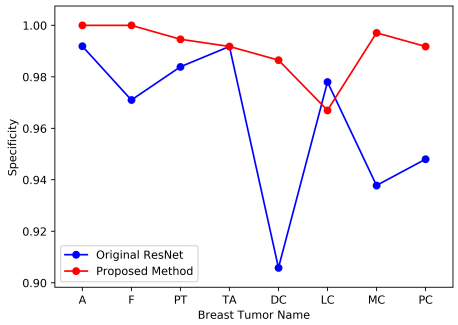
(c) Precision



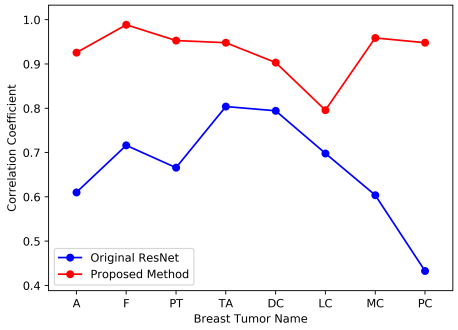
(d) Recall



(e) Sensitivity



(f) Specificity



(g) Correlation Coefficient

Figure 4.9: Original and proposed method hyperparameters compare between eight breast tumors from 40X testing dataset.

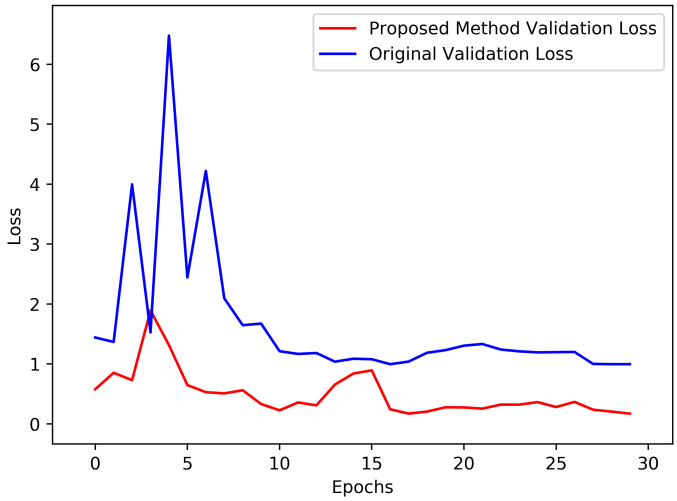


Figure 4.10: Original ResNet and proposed method 100X validation loss compare.

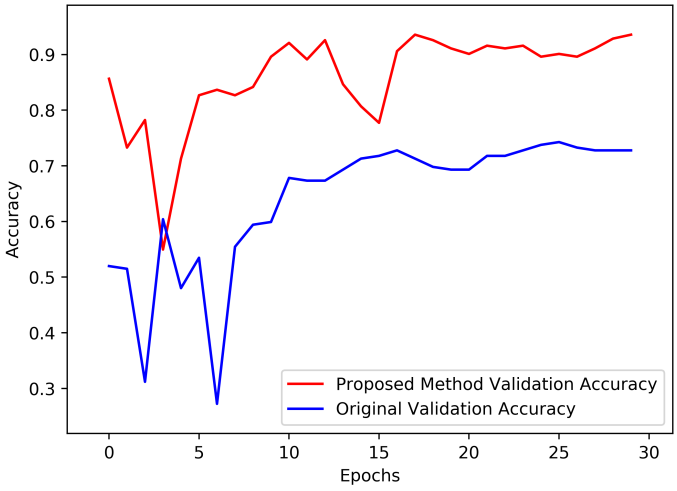


Figure 4.11: Original ResNet and proposed method 100X validation accuracy compare.

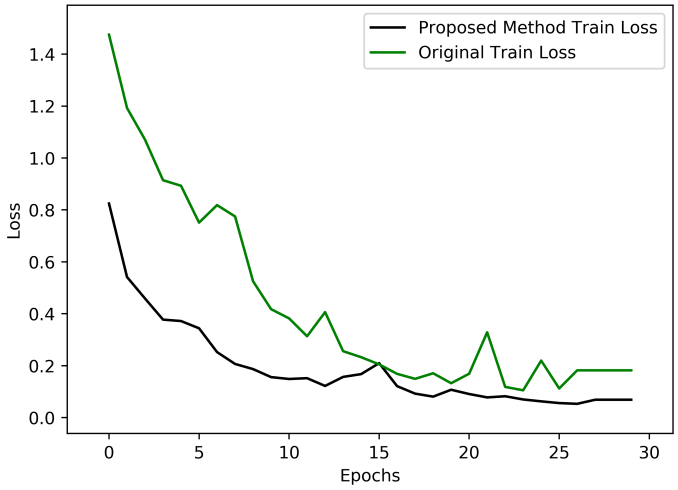


Figure 4.12: Original ResNet and proposed method 100X training loss compare.

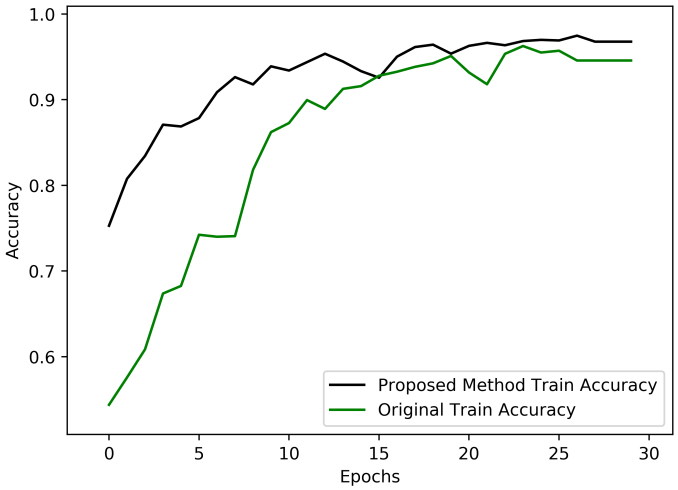


Figure 4.13: Original ResNet and proposed method 100X training accuracy compare.

validation process. The accuracy of the original ResNet model can be achieved 94.36%, while the proposed method model can be achieved 97.46%. It is increased 3.1%.

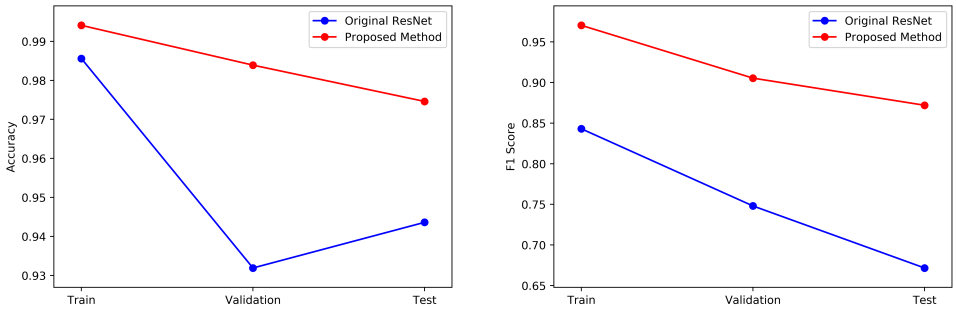
Table 4.10: Hyperparameters compare between original ResNet and proposed method for 100X training dataset.

Hyperparameters	**Original ResNet**	**Proposed method**
Accuracy	0.9856	0.9941
F1 Score	0.8430	0.9704
Precision	0.8540	0.9693
Recall	0.8529	0.9718
Sensitivity	0.8529	0.9718
Specificity	0.9914	0.9963
Correlation coefficient	0.7860	0.9666

Table 4.11: Hyperparameters compare between original ResNet and proposed method for 100X validation dataset.

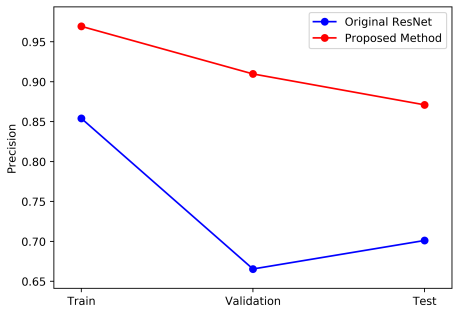
Hyperparameters	**Original ResNet**	**Proposed method**
Accuracy	0.9319	0.9839
F1 Score	0.7480	0.9054
Precision	0.6654	0.9097
Recall	0.5955	0.9051
Sensitivity	0.5955	0.9051
Specificity	0.9560	0.9899
Correlation coefficient	0.5430	0.8962

In order to make a more comprehensive comparison, we also conducted a horizontal comparison from different types of breast tumors. The results of training Figure 4.15, validation Figure 4.16, and testing Figure 4.17 can clearly show that for four benign breast tumors and four malignant breast tumors, the overall ResNet architecture has been optimized. Performance is better than the original architecture. This phenomenon shows that

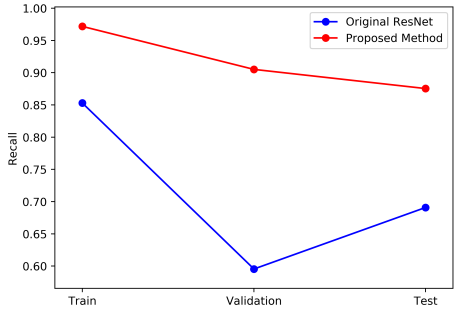


(a) Accuracy

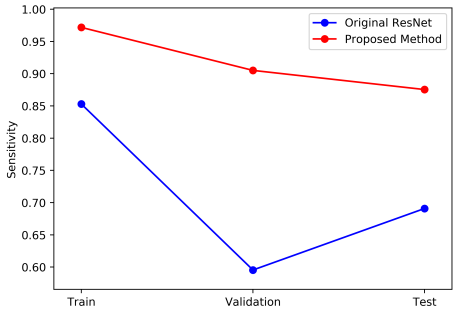
(b) F1 Score



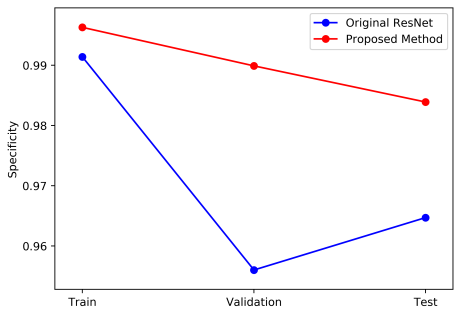
(c) Precision



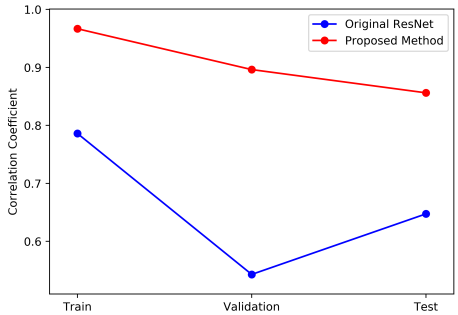
(d) Recall



(e) Sensitivity



(f) Specificity



(g) Correlation Coefficient

Figure 4.14: Hyperparameters compare between original ResNet and proposed method for 100X dataset.

Table 4.12: Hyperparameters compare between original ResNet and proposed method for 100X testing dataset.

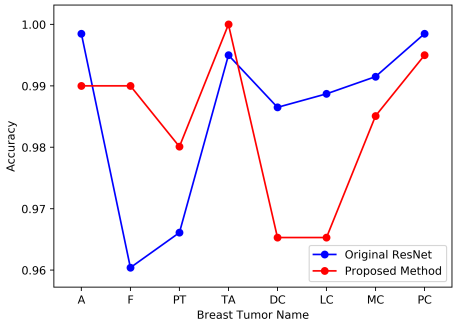
Hyperparameters	**Original ResNet**	**Proposed method**
Accuracy	0.9436	0.9746
F1 Score	0.6716	0.8720
Precision	0.7011	0.8710
Recall	0.6909	0.8754
Sensitivity	0.6909	0.8754
Specificity	0.9647	0.9839
Correlation coefficient	0.6474	0.8561

our proposed method is meaningful, it can effectively help doctors play a certain role in the field of auxiliary medicine.

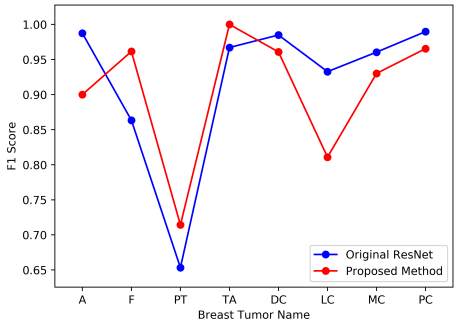
4.5.3 Breast Cancer - 200X Result

In the 200X dataset, using the original ResNet and the proposed method, we compared the loss, accuracy of the training dataset and the validation dataset. The following line chart Figure 4.18, Figure 4.19, Figure 4.20 and Figure 4.21 shows the experiment results. Compared with the original ResNet, the proposed method has greatly improved performance and is a very meaningful experimental project.

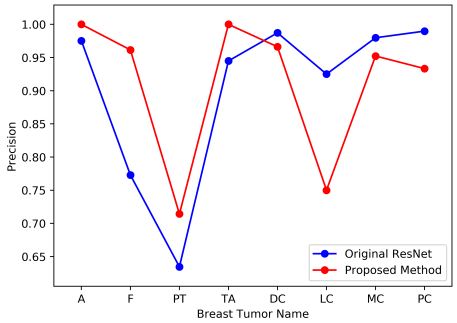
Table 4.13, Table 4.14, Table 4.15, Figure 4.22 shows the seven parameters in the original ResNet and proposed method architecture in detail. By comparing the training dataset, validation dataset and testing dataset, the proposed method experimental data has also been improved, which proves that our experiment the significance can improve the auxiliary diagnosis of breast tumor diseases. The improvement of its parameters also indicates the success of the experiment. We can intuitively see the results of our test after the training and validation process. The accuracy of the original ResNet model can be achieved 93.00%, while the proposed method model can be achieved 98.31%. It is increased 5.31%.



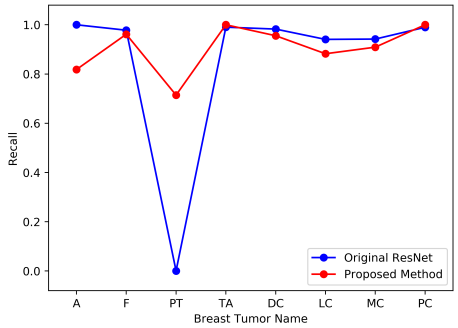
(a) Accuracy



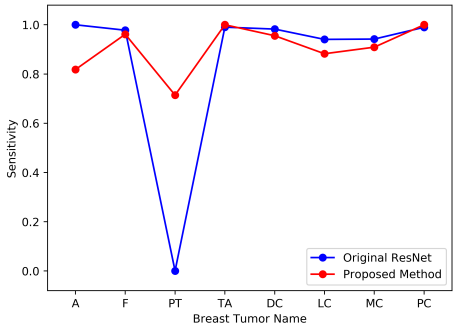
(b) F1 Score



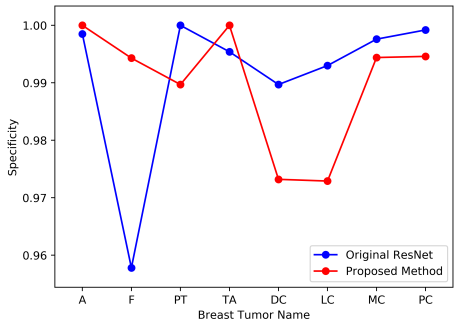
(c) Precision



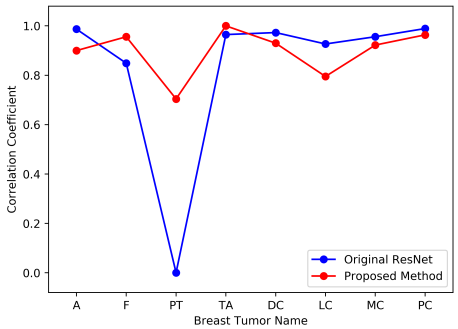
(d) Recall



(e) Sensitivity

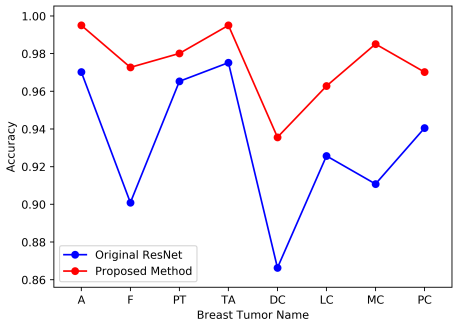


(f) Specificity

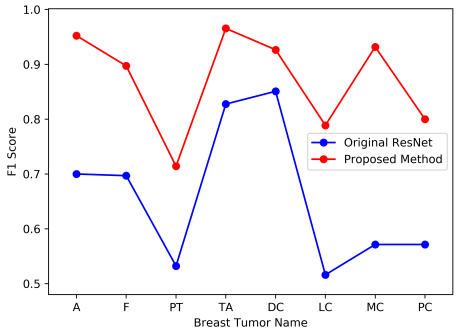


(g) Correlation Coefficient

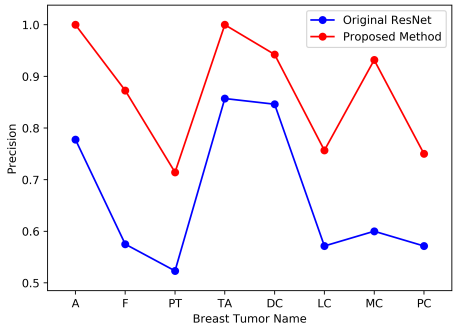
Figure 4.15: Original and proposed method hyperparameters compare between eight breast tumors from 100X training dataset.



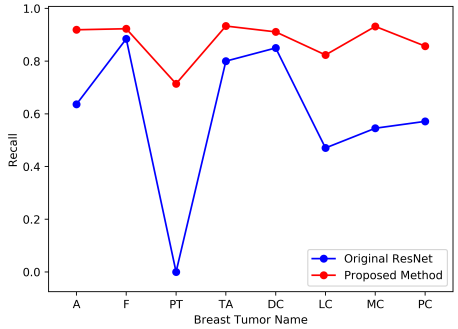
(a) Accuracy



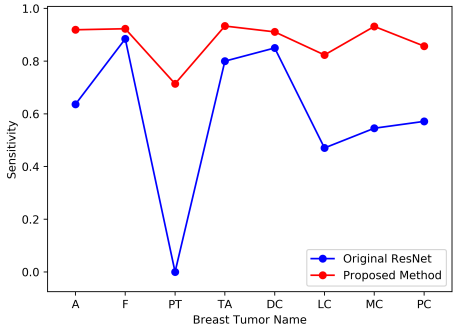
(b) F1 Score



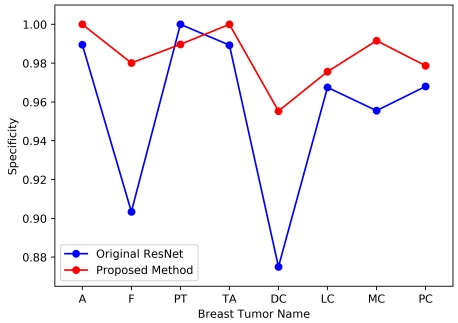
(c) Precision



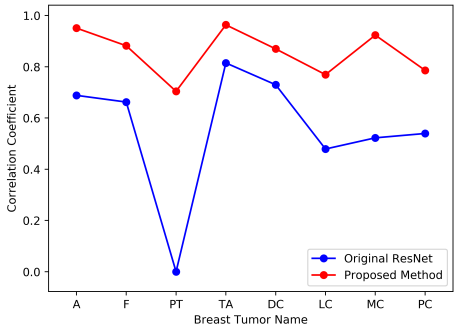
(d) Recall



(e) Sensitivity

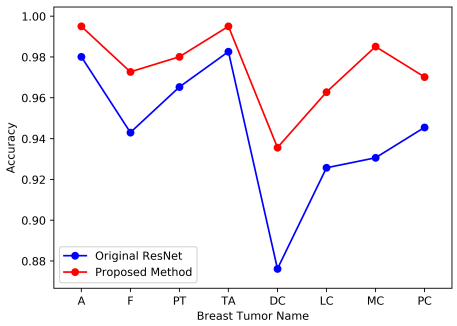


(f) Specificity

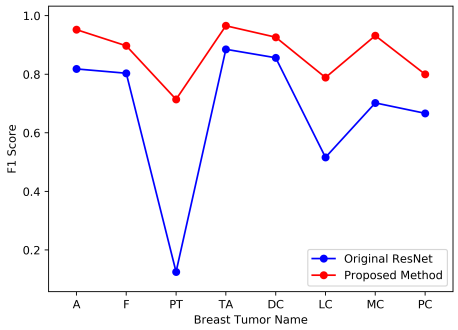


(g) Correlation Coefficient

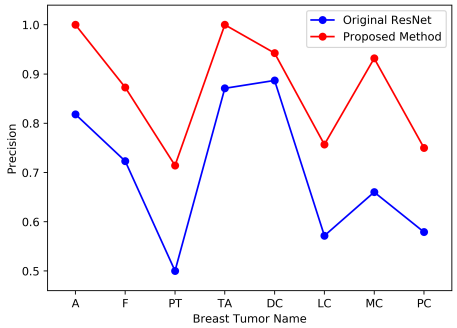
Figure 4.16: Original and proposed method hyperparameters compare between eight breast tumors from 100X validation dataset.



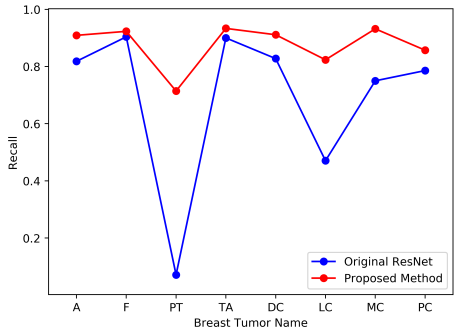
(a) Accuracy



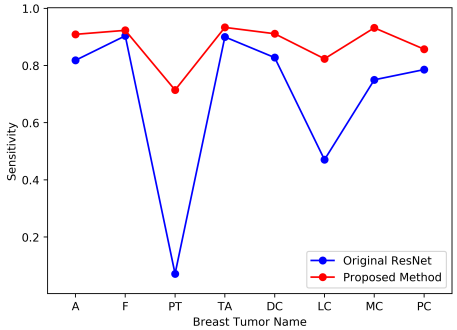
(b) F1 Score



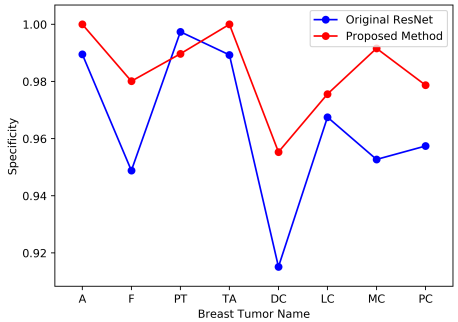
(c) Precision



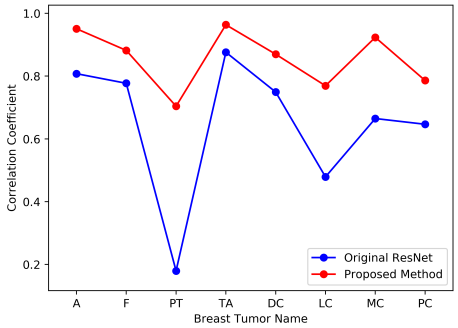
(d) Recall



(e) Sensitivity



(f) Specificity



(g) Correlation Coefficient

Figure 4.17: Original and proposed method hyperparameters compare between eight breast tumors from 100X testing dataset.

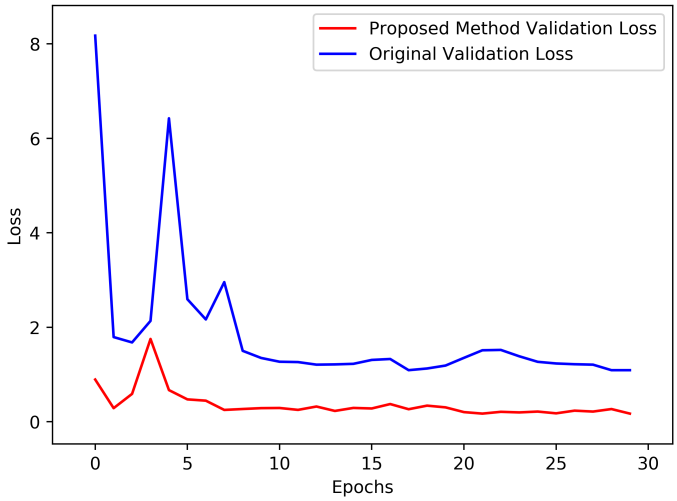


Figure 4.18: Original ResNet and proposed method 200X validation loss compare.

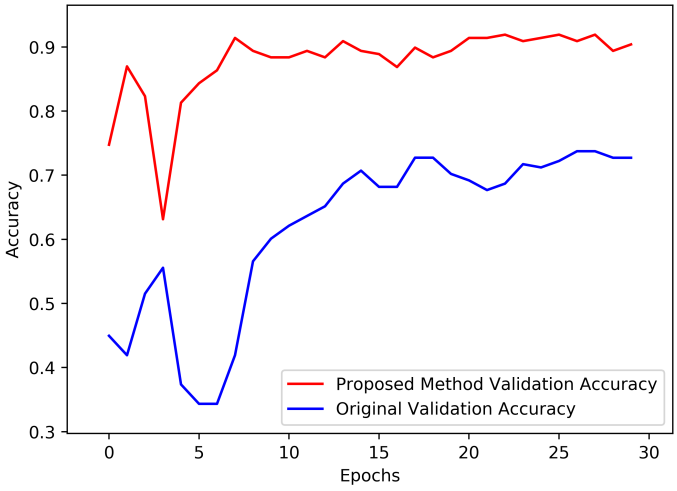


Figure 4.19: Original ResNet and proposed method 200X validation accuracy compare.

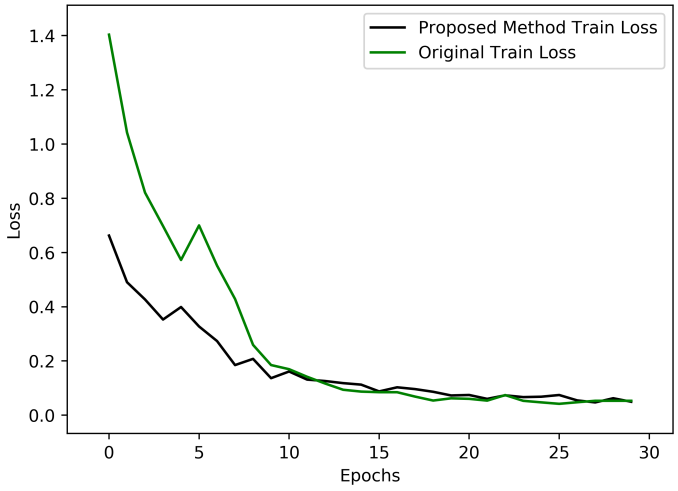


Figure 4.20: Original ResNet and proposed method 200X training loss compare.

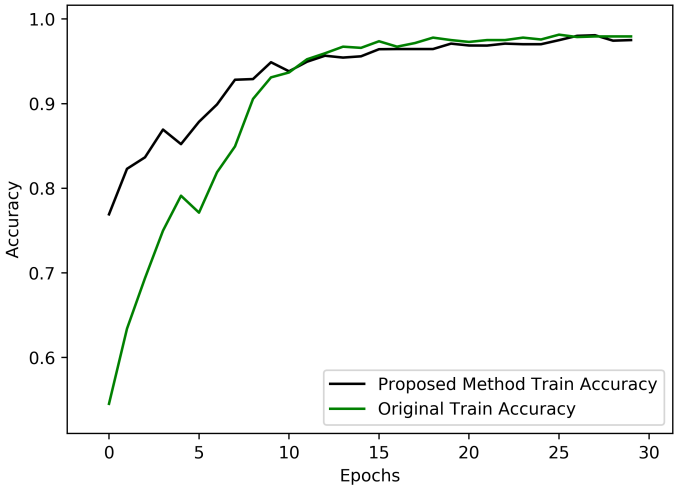


Figure 4.21: Original ResNet and proposed method 200X training accuracy compare.

Table 4.13: Hyperparameters compare between original ResNet and proposed method for 200X training dataset.

Hyperparameters	**Original ResNet**	**Proposed method**
Accuracy	0.9944	0.9966
F1 Score	0.9750	0.9875
Precision	0.9743	0.9853
Recall	0.9762	0.9899
Sensitivity	0.9762	0.9899
Specificity	0.9965	0.9977
Correlation coefficient	0.9714	0.9851

Table 4.14: Hyperparameters compare between original ResNet and proposed method for 200X validation dataset.

Hyperparameters	**Original ResNet**	**Proposed method**
Accuracy	0.9318	0.9785
F1 Score	0.6408	0.8995
Precision	0.6695	0.9140
Recall	0.6355	0.8877
Sensitivity	0.6355	0.8877
Specificity	0.9584	0.9845
Correlation coefficient	0.6056	0.8858

Table 4.15: Hyperparameters compare between original ResNet and proposed method for 200X testing dataset.

Hyperparameters	**Original ResNet**	**Proposed method**
Accuracy	0.9300	0.9831
F1 Score	0.6288	0.9238
Precision	0.6499	0.9251
Recall	0.6196	0.9272
Sensitivity	0.6196	0.9272
Specificity	0.9568	0.9888
Correlation coefficient	0.5890	0.9140

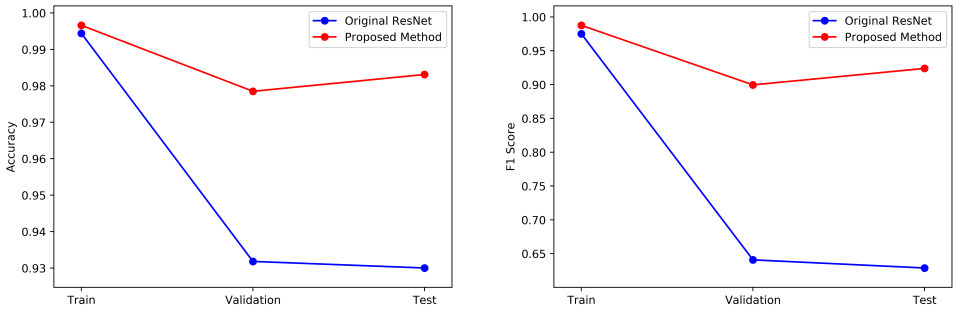
In order to make a more comprehensive comparison, we also conducted a horizontal comparison from different types of breast tumors. The results of training Figure 4.24, validationFigure 4.23, and testing Figure 4.25 can clearly show that for four benign breast tumors and four malignant breast tumors, the overall ResNet architecture has been optimized. Performance is better than the original architecture. This phenomenon shows that our proposed method is meaningful, it can effectively help doctors play a certain role in the field of auxiliary medicine.

4.5.4 Breast Cancer - 400X Result

For the 400X dataset, using the original ResNet and the proposed method, we compared the loss, accuracy of the training dataset and the validation dataset. The following line chart Figure 4.26, Figure 4.27, Figure 4.28 and Figure 4.29 shows the experiment results. Compared with the original ResNet, the proposed method has greatly improved performance and is a very meaningful experimental project.

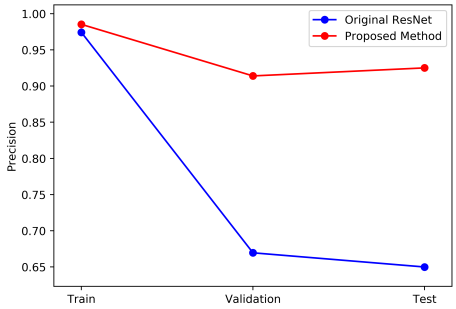
Table 4.16, Table 4.17, Table 4.18, Figure 4.30 shows the seven parameters in the original ResNet and proposed method architecture in detail. By comparing the training dataset, validation dataset and testing dataset, the proposed method experimental data has also been improved, which proves that our experiment the significance can improve the auxiliary diagnosis of breast tumor diseases. The improvement of its parameters also indicates the success of the experiment. We can intuitively see the results of our test after the training and validation process. The accuracy of the original ResNet model can be achieved 90.78%, while the proposed method model can be achieved 97.22%. It is increased 6.44%.

In order to make a more comprehensive comparison, we also conducted a horizontal comparison from different types of breast tumors. The results of trainingFigure 4.32, validationFigure 4.31, and testing Figure 4.33 can clearly show that for four benign breast tumors and four malignant breast tumors, the overall ResNet architecture has been opti-

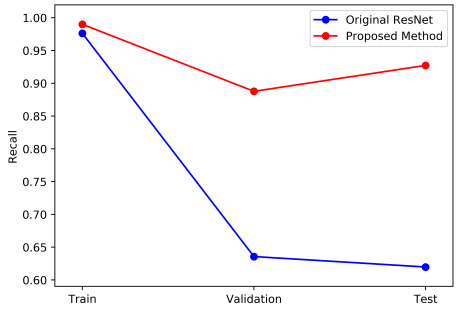


(a) Accuracy

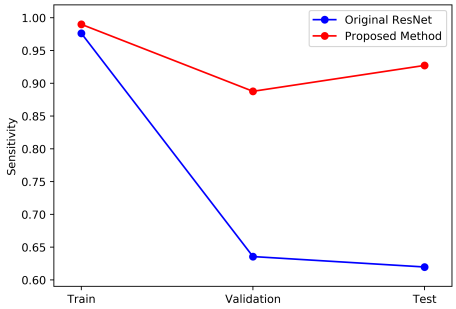
(b) F1 Score



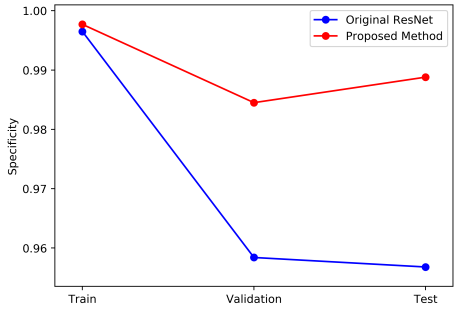
(c) Precision



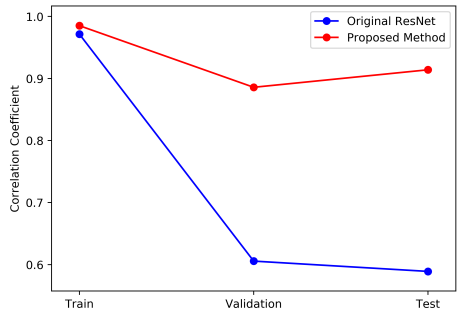
(d) Recall



(e) Sensitivity

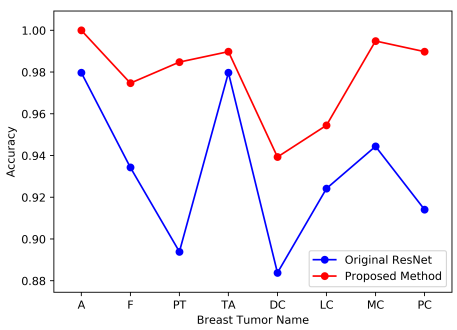


(f) Specificity

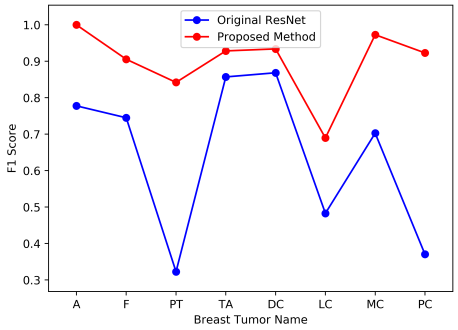


(g) Correlation Coefficient

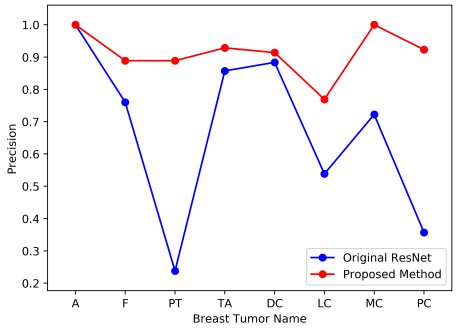
Figure 4.22: Hyperparameters compare between original ResNet and proposed method for 200X dataset.



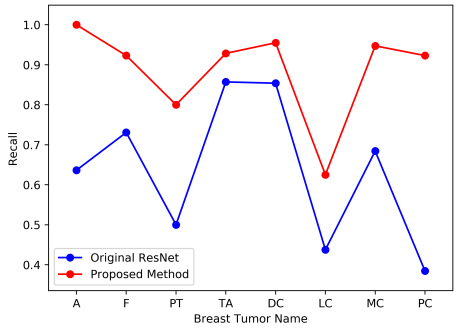
(a) Accuracy



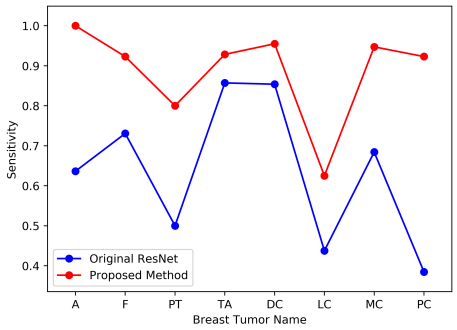
(b) F1 Score



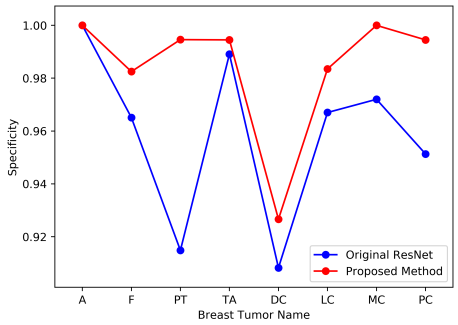
(c) Precision



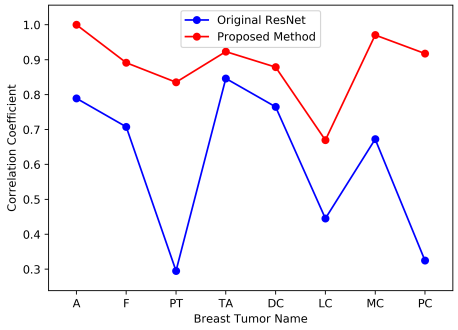
(d) Recall



(e) Sensitivity

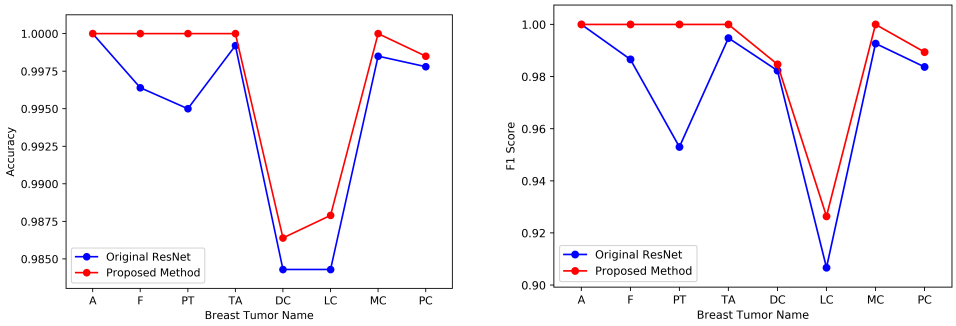


(f) Specificity



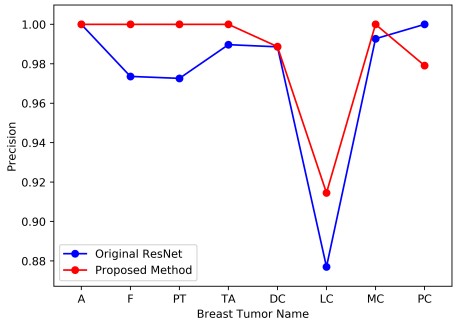
(g) Correlation Coefficient

Figure 4.23: Original and proposed method hyperparameters compare between eight breast tumors from 200X validation dataset.

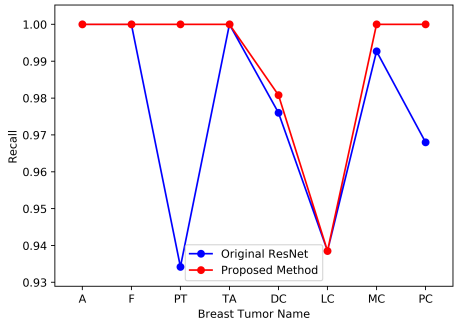


(a) Accuracy

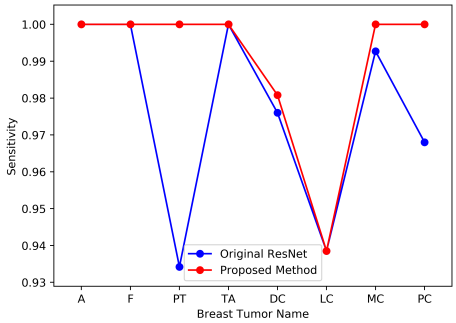
(b) F1 Score



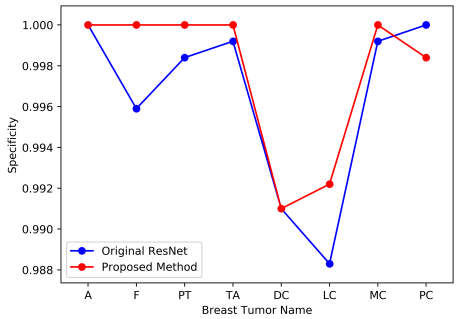
(c) Precision



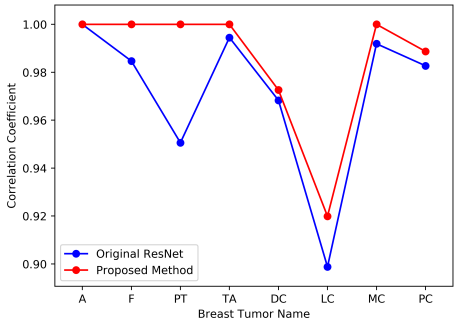
(d) Recall



(e) Sensitivity

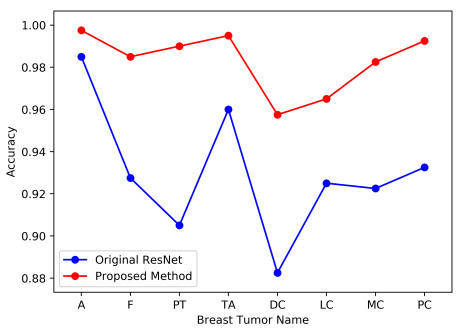


(f) Specificity

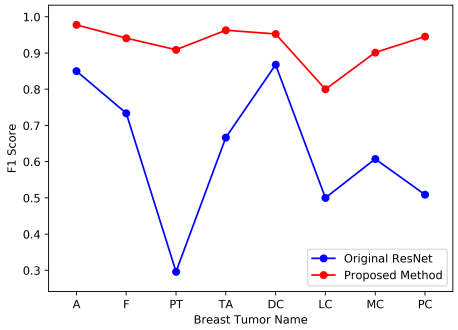


(g) Correlation Coefficient

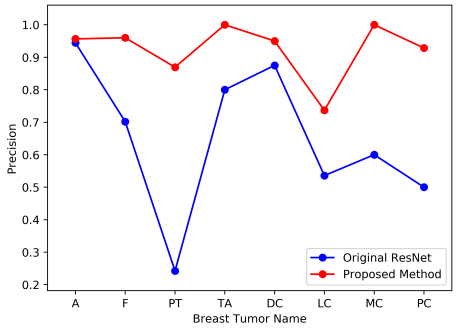
Figure 4.24: Original and proposed method hyperparameters compare between eight breast tumors from 200X training dataset.



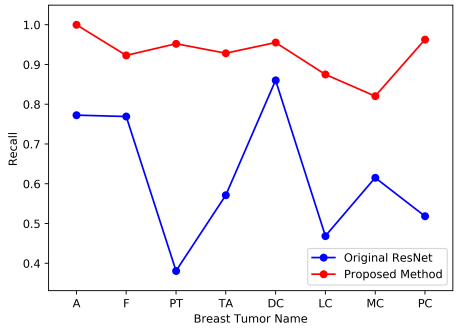
(a) Accuracy



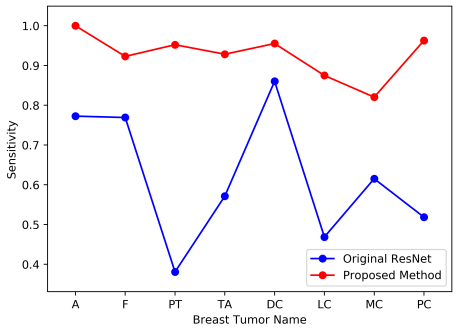
(b) F1 Score



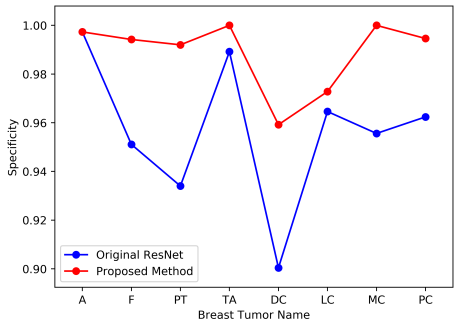
(c) Precision



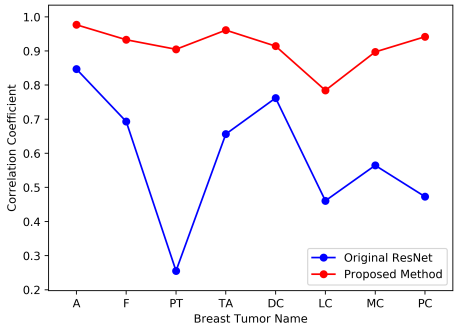
(d) Recall



(e) Sensitivity



(f) Specificity



(g) Correlation Coefficient

Figure 4.25: Original and proposed method hyperparameters compare between eight breast tumors from 200X testing dataset.

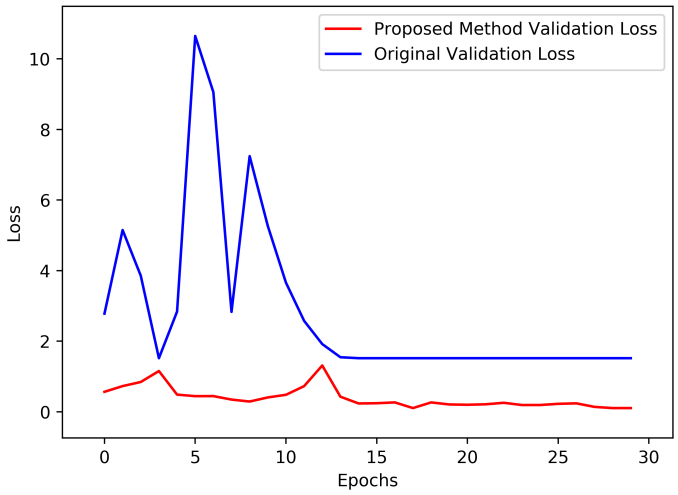


Figure 4.26: Original ResNet and proposed method 400X validation loss compare.

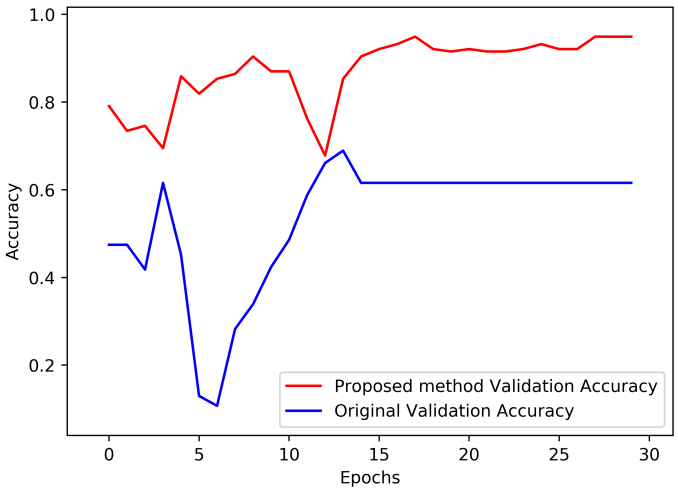


Figure 4.27: Original ResNet and Proposed method 400X validation accuracy compare.

Table 4.16: Hyperparameters compare between original ResNet and proposed method for 400X training dataset.

Hyperparameters	**Original ResNet**	**Proposed method**
Accuracy	0.9187	0.9962
F1 Score	0.5294	0.9857
Precision	0.6851	0.9860
Recall	0.5150	0.9854
Sensitivity	0.5150	0.9854
Specificity	0.9445	0.9973
Correlation coefficient	0.5203	0.9830

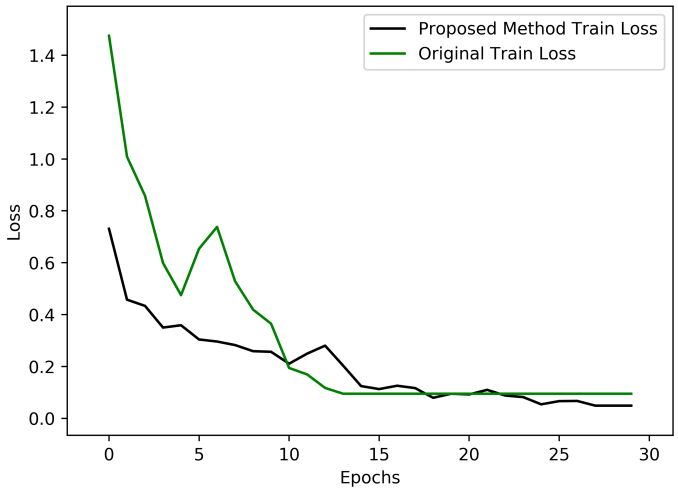


Figure 4.28: Original ResNet and proposed method 400X training loss compare.

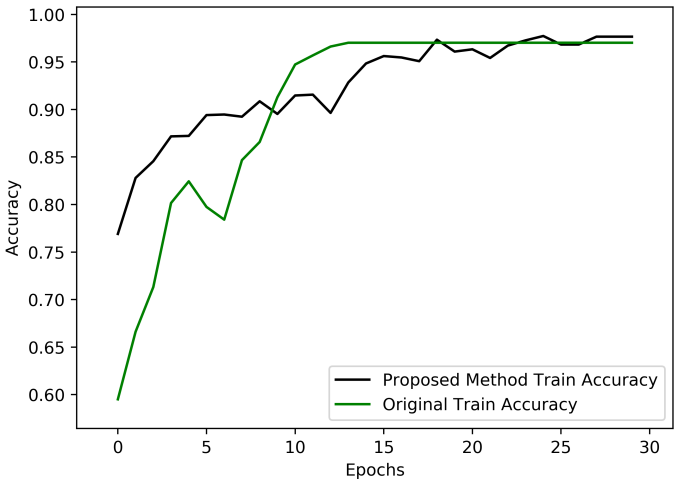
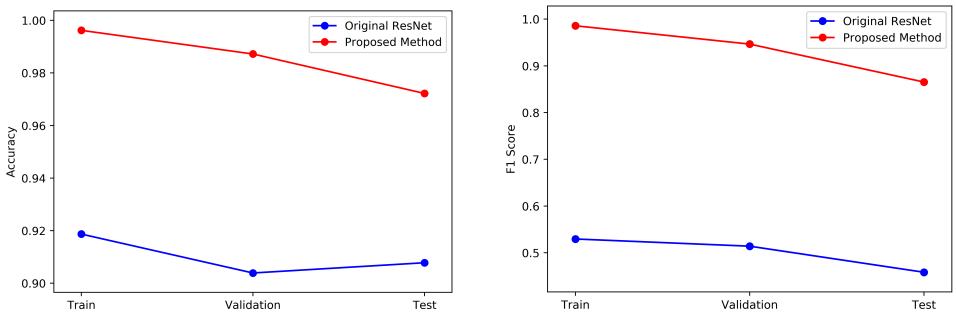


Figure 4.29: Original ResNet and proposed method 400X training accuracy compare.

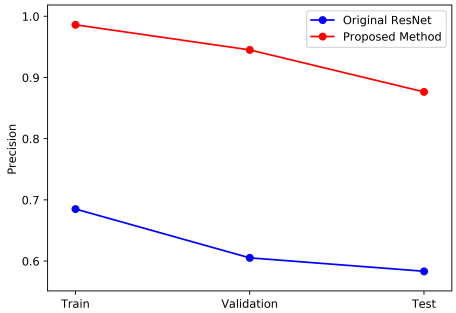
Table 4.17: Hyperparameters compare between original ResNet and proposed method for 400X validation dataset.

Hyperparameters	**Original ResNet**	**Proposed method**
Accuracy	0.9039	0.9872
F1 Score	0.5140	0.9465
Precision	0.6053	0.9450
Recall	0.4309	0.9497
Sensitivity	0.4309	0.9497
Specificity	0.9345	0.9915
Correlation coefficient	0.5105	0.9383

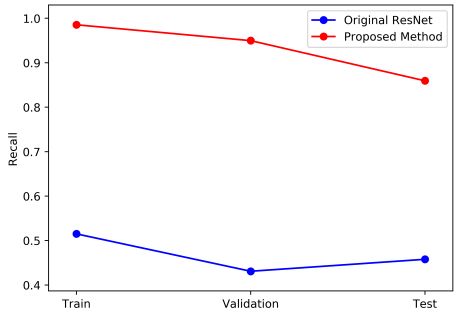


(a) Accuracy

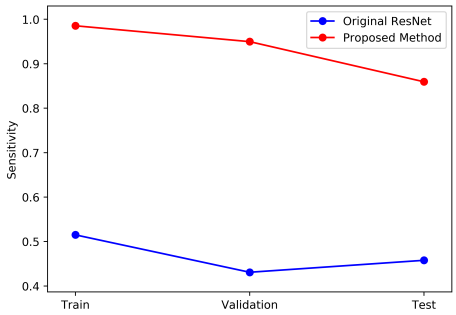
(b) F1 Score



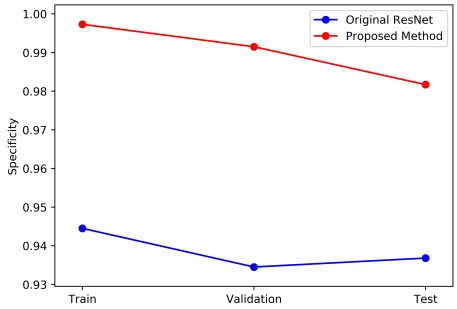
(c) Precision



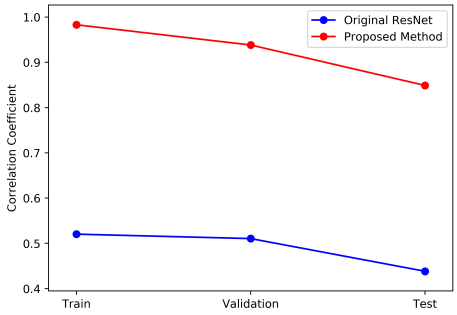
(d) Recall



(e) Sensitivity



(f) Specificity



(g) Correlation Coefficient

Figure 4.30: Hyperparameters compare between original ResNet and proposed method for 400X dataset.

Table 4.18: Hyperparameters compare between original ResNet and proposed method for 400X testing dataset.

Hyperparameters	**Original ResNet**	**Proposed method**
Accuracy	0.9078	0.9722
F1 Score	0.4582	0.8653
Precision	0.5834	0.8764
Recall	0.4578	0.8594
Sensitivity	0.4578	0.8594
Specificity	0.9368	0.9817
Correlation coefficient	0.4383	0.8490

mized. Performance is better than the original architecture. This phenomenon shows that our proposed method is meaningful, it can effectively help doctors play a certain role in the field of auxiliary medicine.

4.6 Evaluation Metrics

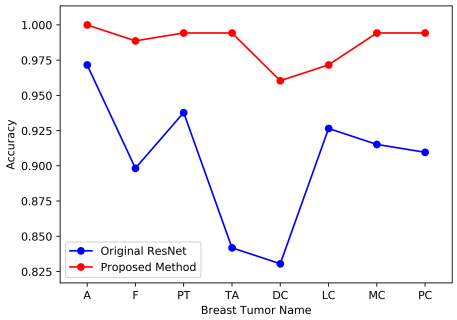
Eight different breast tumors have different characteristics. These characteristics are what the machine wants to learn and are the main basis for classification.

Adenosis is characterized by lobular acinar, peripheral ducts and connective tissue hyperplasia, and the lobular structure is basically preserved. According to different histological changes at different development stages, it can be divided into 2 types:

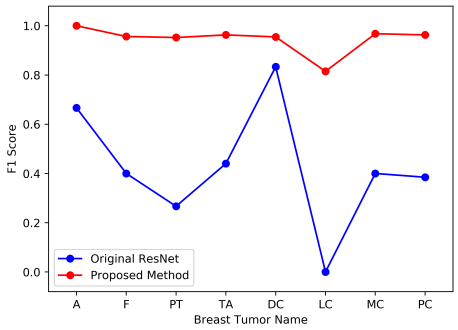
1) The lobular hyperplasia is mainly manifested by the number of lobules and the number of acinar in the lobule. The epithelial cells are not significantly changed or may be double-layered or multi-layered, and the intralobular duct may be slightly expanded.

2) Leaflet fibrosis type, the main features are interstitial fibrosis and acinar atrophy in the leaflet. The outline of the leaflet sometimes exists, but it can also disappear, leaving only some atrophied ducts.

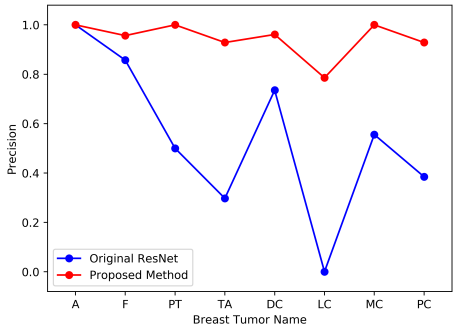
Fibroadenoma Figure 4.34, the main feature is that in addition to the peripheral ducts



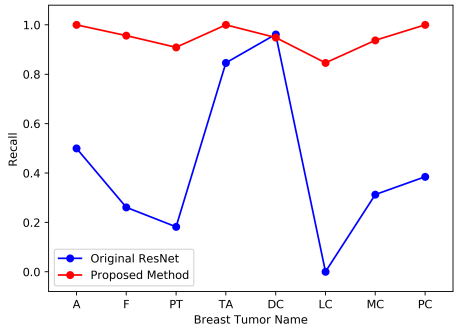
(a) Accuracy



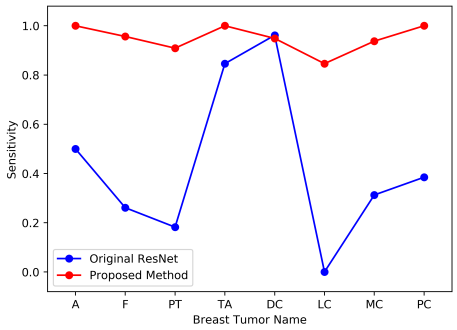
(b) F1 Score



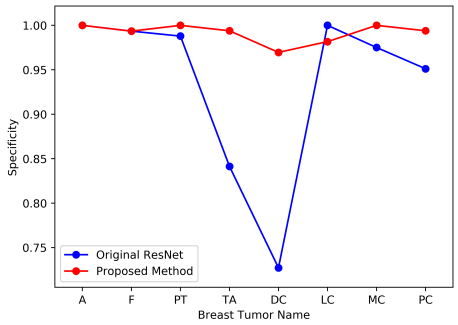
(c) Precision



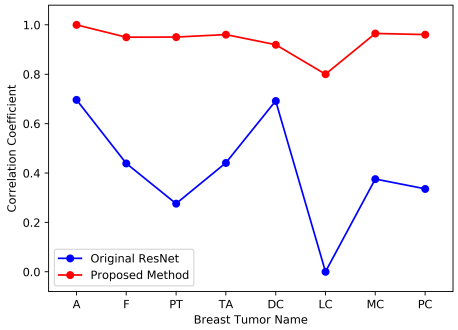
(d) Recall



(e) Sensitivity

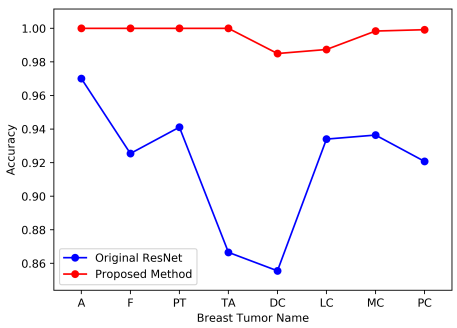


(f) Specificity

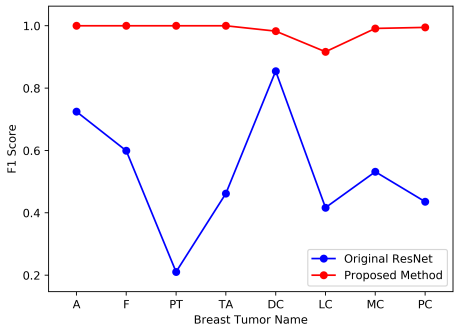


(g) Correlation Coefficient

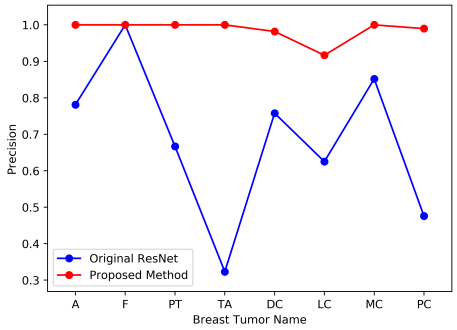
Figure 4.31: Original and proposed method hyperparameters compare between eight breast tumors from 400X validation dataset.



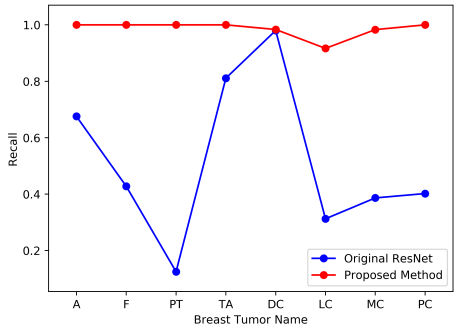
(a) Accuracy



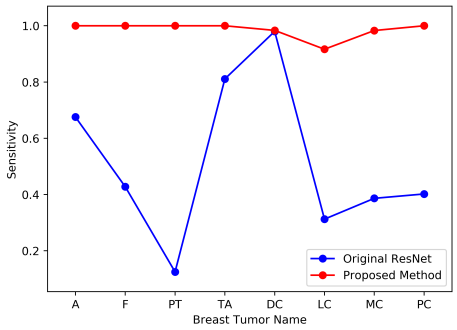
(b) F1 Score



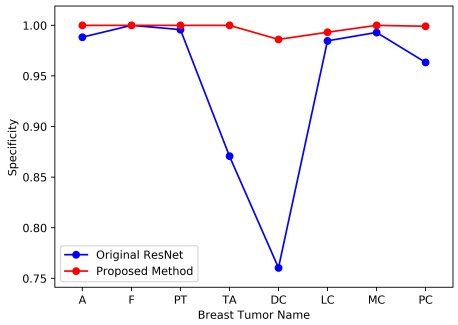
(c) Precision



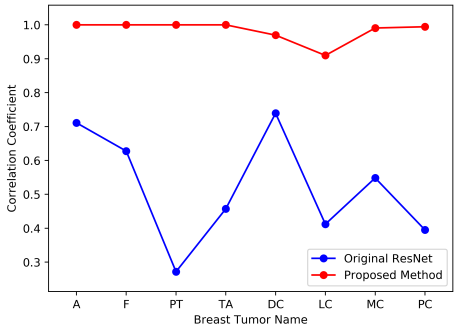
(d) Recall



(e) Sensitivity

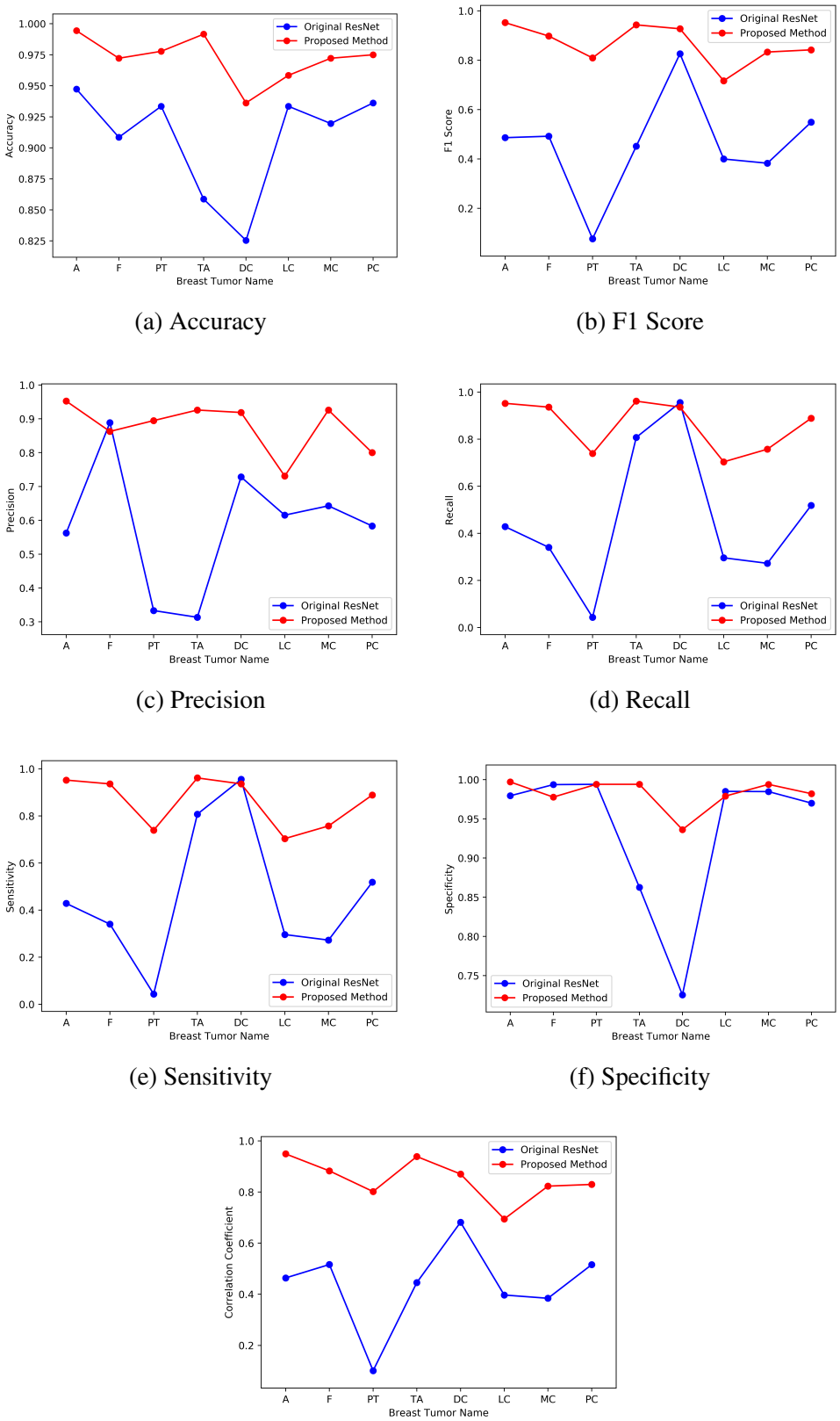


(f) Specificity



(g) Correlation Coefficient

Figure 4.32: Original and proposed method hyperparameters compare between eight breast tumors from 400X training dataset.



(a) Accuracy

(b) F1 Score

(c) Precision

(d) Recall

(e) Sensitivity

(f) Specificity

(g) Correlation Coefficient

Figure 4.33: Original and proposed method hyperparameters compare between eight breast tumors from 400X testing dataset.

and acinar hyperplasia, the interstitial connective tissue also has obvious hyperplasia in the leaflets. In the early stage, the lobule was enlarged due to the continued proliferation of acinus; in the later stage, the connective tissue in the lobule was significantly increased, causing the acinar to disperse and deform.

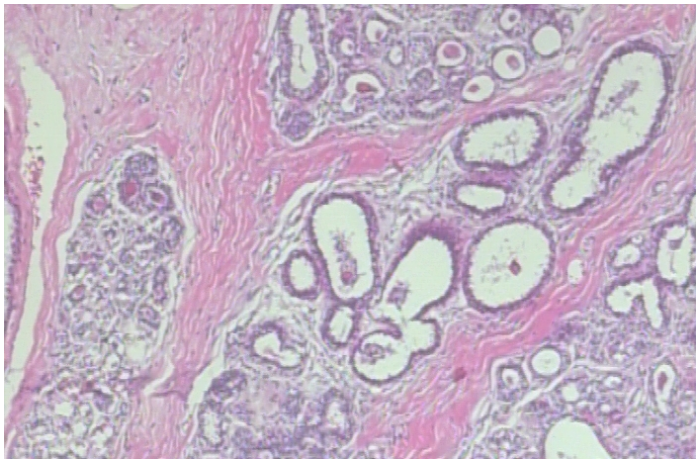


Figure 4.34: Peripheral ducts, acinars and interstitium all showed obvious hyperplasia, some acinars and ducts expanded, and some lymphocytes infiltrated in the interstitium.

Phyllodes tumor Figure 4.35 are a group of tumors that are basically similar to fibroadenoma, with clear boundaries and bidirectional differentiation. Its histological feature is that the bilayer epithelial cells with fissure-like distribution are surrounded by overgrown cell-rich mesenchymal components, forming a typical leaf-like structure. PT typically manifests as a way of growing into the lumen, accompanied by leaf-like protrusions that protrude into the expansion cavity. In the area in close contact with the epithelial components, the interstitial cell components are more abundant and distributed in a band shape; in the loose interstitial area, the cell density is low.

Tubular adenona of breast is usually less than 4cm in diameter. Under the microscope, the tumor is composed of round or oval glands with a uniform size. The gland contains two layers of epithelial cells, with few interstitial components, and may contain a small number of lymphocytes.

Histological characteristics of lobular carcinomaFigure 4.36: The lobular structure still exists, but it becomes larger, and the acinar cells are piled up and arranged irregularly. The

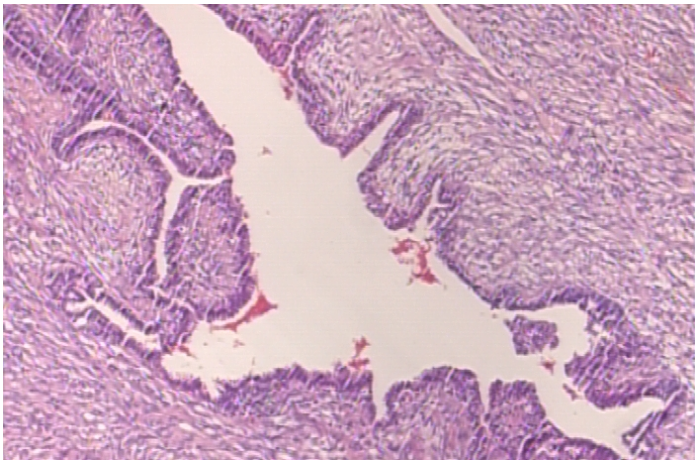


Figure 4.35: Leaf-like protrusions protruding into the expansion cavity are typical features.

size and shape of the cells are relatively uniform, round, with intense nuclear staining, and may have mitotic figures.

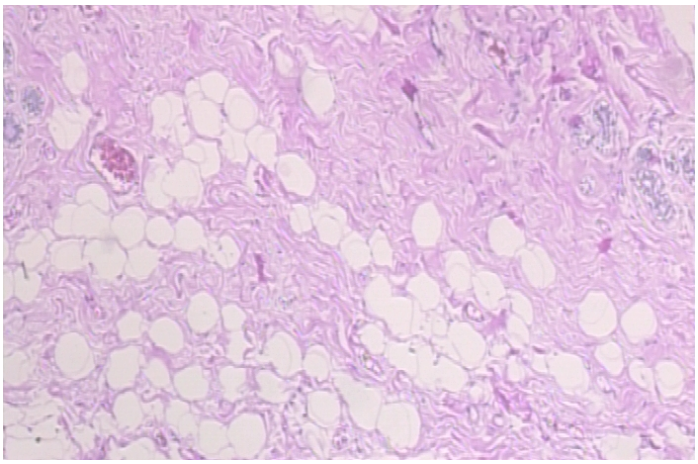


Figure 4.36: Acinar cells are stacked together and appear round.

The pathological manifestation of mucinous adenocarcinoma is that a large number of extracellular mucus floats with solid tumors, ropes, glandular tubes, and sieve-like structural cancer tissue foci.

4.7 Discussion

The question about the magnification of the microscope is not as large as possible, we analyzed it through the table of test results in the experimental part Table 4.9,Table 4.12,Table 4.15,

and Table 4.18, the accuracy of the test result of the dataset with a magnification of 40X is 98.48%, the accuracy of the test result of the data set of 100X is 97.46%, the accuracy of the test result of the data set of 200X is 98.31%, The accuracy of the test results with the 400X data set is 97.22%, which clearly shows that the 40X size dataset is very suitable for the auxiliary detection of breast tumors, because the magnification of the microscope is related to the number of cells in the field of view, and the 40X size can be very good, and shows the special characteristics of the disease. Choosing the appropriate microscope magnification to conduct experiments can effectively improve the accuracy and effectiveness of our experiments.

Figure 4.6, Figure 4.14, Figure 4.22, and Figure 4.30, the red lines in the figure represent the proposed method, and the blue lines represent original ResNet can intuitively indicate that the proposed method is superior to the original one. The original ResNet training dataset hyperparameters performed very well, but after the validation dataset and the test dataset, the results were significantly reduced. Our analysis may have the following problems: model problems, dataset problems, or overfitting problems. We generally divide the dataset into training dataset, validation dataset and test dataset. The training dataset is used to train the parameters of the model; the validation dataset is used to verify the performance of different models; the test dataset is used to test the performance of the trained model. Our dataset is collected by the official and has a detail label, which is a very powerful source, so we believe that the quality of the dataset is not a problem. Secondly, regarding overfitting, overfitting means that the parameters are adjusted too in line with the sample, and the method to be solved is the same as the above problem. Adjust the dataset samples and parameters. Regularization, dropout, redesign of the model, and early termination of training can all be procedures to avoid overfitting. So we want to prevent overfitting problems, we adjusted and optimized the dropout layer in proposed method.

In summary, we think that the most likely problem is the ResNet model. The performance of a model is not due to its error on the training dataset, but whether its error on the

test dataset is close to the error on the training dataset. That is to say, when the accuracy of the validation dataset and the test dataset is low, it may not be a problem of the data itself, but the model has not found the most suitable parameters and settings, and does not have good generalization ability. Therefore, we have made a lot of parameter adjustments in the proposed method to achieve the best model. Our proposed method model has a very good average performance. It has shown good performance in the training dataset, validation dataset and testing dataset, avoiding the problems mentioned above.

The linear charts of Figure 4.2, Figure 4.3, Figure 4.10, Figure 4.11, Figure 4.18, Figure 4.19, Figure 4.27 and Figure 4.26 can show that the validation loss of the original ResNet is very unstable because the parameter is the loss value calculated by our preset loss function, if it is unstable, it means that the model does not have good prediction ability. Loss is used in the model to optimize parameters and achieve gradient descent. In general, the smaller the loss, the higher the network optimization. When training through the model, most of them indicate that the overall trend is that loss decreases and accuracy increases. In order to maintain a stable loss, we make improvements in the proposed method architecture. It can be seen that after the improvement, the trend of loss is very stable and maintains a relatively low value. Therefore, the proposed method model is effective.

CHAPTER 5
CONCLUSION

Through a comprehensive comparison of the experimental part, the proposed method has made significant progress relative to the original ResNet, indicating that the selected optimal hyperparameters have brought significant progress to the ResNet and improved the accuracy of training and testing. The proposed method can help researchers to analyze breast tumor diseases more comprehensively and accurately, which is a very useful improvement. Because in the neural network, in addition to finding the best weight and deviation parameters, it is also important to set appropriate hyperparameters. For example, the number of neurons in each layer, the value of the batch size, the learning rate when the parameter is updated, the weight decay coefficient, or the learning epoch. The process of finding hyperparameters is usually accompanied by many repeated experiments and errors, so it is very important to find hyperparameters as efficiently as possible. This is what our project did.

By adjusting the hyperparameters in the architecture, these hyperparameters will affect the learning speed of the neural network and the final classification results, because deep learning usually takes a lot of time, therefore, in the process of finding hyperparameters, it is necessary to abandon those illogical problems and use appropriate hyperparameters as soon as possible. The table in the experimental part effectively shows that the proposed method is superior to the original ResNet in all aspects. Although the performance of the original ResNet is also very good, after selecting more suitable hyperparameters, you will get even better results.

In this project, we used genetic algorithms to optimize the ResNet model and created an proposed method model based on the BreakHis breast tumor dataset. Compared with the original ResNet model, it has been significantly improved. For the 40X dataset, the accuracy rate of the original ResNet model is 94.17%, and our result is improved to 98.48%;

for the 100X dataset, the accuracy rate of the original ResNet model is 94.36%, and our result is improved to 97.46%; For the 200X dataset, the accuracy rate of the original ResNet model is 93.00%, and our result is improved to 98.31%; for the 400X dataset, the accuracy rate of the original ResNet model is 90.78%, and our result is improved to 97.22%. This is a very big improvement. This project can help doctors or patients to judge benign or malignant breast tumors in time, which is of great significance to society.

In summary, compared to the BreakHis breast tumor dataset, the proposed method is very useful compared to the original ResNet, and can provide very good medical aided diagnostic methods. But medical data needs to be updated at any time, and a large amount is required. Although the BreakHis dataset has detailed labels and high-quality images, it still has some defects in quantity. In the training of neural network architecture, the imbalance in the number and type of datasets has a very large impact on the results produced. It is hoped that in the future, higher quality and quantitative breast tumor datasets can be used for further experiments to create more meaningful models.

Appendices

APPENDIX A

MAIN SOURCE CODE

```
import os
    os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
    os.environ["CUDA_VISIBLE_DEVICES"] = "1"
    import numpy as np
    import pandas as pd
    import random
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    from skimage.transform import resize
    from models import *
    from data_processing import data_split
    from training_fn import *
    import keras
    from keras.layers import *
    from keras.models import *
    from keras import layers
    from keras.utils.data_utils import get_file
    from keras import backend as K
    from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
    from keras.optimizers import Adam
    from keras.preprocessing.image import ImageDataGenerator
    from sklearn.linear_model import LogisticRegression
    from sklearn.svm import SVC
```

```
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import cross_validate
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score
import time
import logging
import argparse
import tensorflow as tf
import keras.backend.tensorflow_backend as KTF
seed_value = 42
os.environ['PYTHONHASHSEED']=str(seed_value)
np.random.seed(seed_value)
random.seed(seed_value)
tf.random.set_random_seed(seed_value)
config = tf.ConfigProto()
config.gpu_options.allow_growth=True
session = tf.Session(config=config)
KTF.set_session(session)
# Name list and magnification list.
magnification_list = ['40X', '100X', '200X', '400X']
benign_list = ['adenosis', 'fibroadenoma', 'phyllodes_tumor', 'tubular_adenoma']
malignant_list = ['ductal_carcinoma', 'lobular_carcinoma', 'mucinous_carcinoma', 'pap-
illary_carcinoma']
cancer_list = benign_list + malignant_list
models = [vgg16_model, vgg19_model, xception_model, resnet_model, inception_model,
inception_resnet_model]
model_num = 3 # Select resnet as the backbone.
model_name = models[model_num].__name__
```

```
# Set image size.
image_height=115
image_width=175
n_channels=3
# Get the timestamp and set it as weight name.
timestampEND = time.strftime("%H%M%S") + '' + time.strftime("%d%m%Y")
weight_name = './models/weights' + timestampEND +'.h5'
# Hyperparameters.
# epochs = 2
# batch_size = 32
# learning_rate = 0.0001
# optimizer = "ADAM"
# lr_decay = 0.1
# dropout = 0.3
# layer1 = 512
# layer2 = 128
# layer3 = 32
# horizontal_flip = True
# vertical_flip = True
# rotation_range = 10.0
# shear_range = 0.2
# zoom_range = 0.2
epochs = 2
batch_size = 32
learning_rate = 0.0001
optimizer = "ADAM"
lr_decay = 0.1
```

dropout = 0.3

layer1 = 0

layer2 = 0

layer3 = 0

horizontal_flip = True

vertical_flip = True

rotation_range = 10.0

shear_range = 0.2

zoom_range = 0.2

if optimizer == ”ADAM”:

optimizer_type = Adam

if optimizer == ”SGD”:

optimizer_type = SGD

iteration = 0

average_accuracy = 0.0

for types in magnification_list:

if iteration == 0:

load_wt = ”Yes”

else:

load_wt = ”No”

Load data.

training_images, training_labels, validation_images, validation_labels, testing_images, testing_labels = data_split(magnification = types, validation_percent = 0.1, testing_percent = 0.2)

Image augmentation.

datagen = ImageDataGenerator(

rotation_range=rotation_range,

```
shear_range=shear_range,
zoom_range=zoom_range,
horizontal_flip=horizontal_flip,
vertical_flip=vertical_flip,
)
datagen.fit(training_images)
# Build the model.
for i in range(len(models)):
if models[i].__name__ == model_name:
base_model = models[i]
base_model = base_model(image_height=image_height,image_width=
image_width,n_channels=n_channels,load_wt=load_wt)
# Add additional layers for classification.
x = base_model.output
x = Dense(2048, activation = 'relu')(x)
x = Dropout(dropout)(x)
if layer1 >0:
x = Dense(layer1, activation = 'relu')(x)
x = Dropout(dropout)(x)
if layer2 >0:
x = Dense(layer2, activation = 'relu')(x)
x = Dropout(dropout)(x)
if layer3 >0:
x = Dense(layer3, activation = 'relu')(x)
out = Dense(8, activation = 'softmax')(x)
inp = base_model.input
model = Model(inp, out)
```

```
    # Load model weight.
    try:
    model.load_weights(weight_name)
    print('Weights loaded!')
    except:
    print('No weights defined!')
    pass
    # Get the timestamp and set it as model name.
    model_timestamp = time.strftime("%H%M%S") + '-' + time.strftime("%d%m%Y")
    saved_model_name = './models/-.hdf5'.format(types, model_timestamp)
    model.compile(
    loss="categorical_crossentropy",
    optimizer=optimizer_type(lr=learning_rate),
    metrics=[f1,'accuracy'])
    early_stopping = EarlyStopping(patience=10, verbose=2)
    model_checkpoint = ModelCheckpoint(saved_model_name, save_best_only=True, ver-
bose=2)
    reduce_lr = ReduceLROnPlateau(factor=lr_decay, patience=5, verbose=2)
    history = model.fit_generator(
    datagen.flow(training_images, training_labels, batch_size=batch_size),
    steps_per_epoch=len(training_images) /batch_size, validation_data=[validation_images,
validation_labels],
    callbacks=[early_stopping, model_checkpoint, reduce_lr], epochs=epochs)
    # history = model.fit(training_images, training_labels, #
    validation_data=[validation_images, validation_labels], # epochs=epochs,
    # verbose = 0,
    # batch_size=batch_size,
```

```
    # callbacks=[early_stopping, model_checkpoint, reduce_lr])
    # Load the best model.
    model = keras.models.load_model(saved_model_name, custom_objects='f1': f1)
    # Get the test metrics at last step.
    test_loss, test_acc, test_f1 = model.evaluate(testing_images, testing_labels)
    model.save_weights(weight_name)
    print("The test metrics at last step: ")
    print("The test accuracy for " + model_name + " with magnification "+ types +" is ",
test_acc, " with F1 score of ", test_f1, "
n")
    print()
    # Get the average test accuracy.
    average_accuracy += test_acc / 4.0
    # Print the metrics.
    if True:
    # Print the training metrics.
    train_logits = model.predict(training_images)
    train_pred = np.argmax(train_logits, axis=1)
    train_true = np.argmax(training_labels, axis=1)
    print("="*15 + "  Training metrics ".format(types) + "="*15)
    print_metrics(train_true, train_pred)
    # Print the validation metrics.
    val_logits = model.predict(validation_images)
    val_pred = np.argmax(val_logits, axis=1)
    val_true = np.argmax(validation_labels, axis=1)
    print("="*15 + "  Valiation metrics ".format(types) + "="*15)
    print_metrics(val_true, val_pred)
```

```
# Print the test metrics.
test_logits = model.predict(testing_images)
test_pred = np.argmax(test_logits, axis=1)
test_true = np.argmax(testing_labels, axis=1)
print("="*15 + "  Test metrics ".format(types) + "="*15)
print_metrics(test_true, test_pred)
# Destory the useless model.
iteration += 1
del model
keras.backend.clear_session()
```

APPENDIX B

ORIGINAL RESNET SOURCE CODE

```
import os
    os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
    os.environ["CUDA_VISIBLE_DEVICES"] = "1"
    import nni
    import numpy as np
    import pandas as pd
    import random
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    from skimage.transform import resize
    from models import *
    from data_processing import data_split
    from training_fn import *
    import keras
    from keras.layers import *
    from keras.models import *
    from keras import layers
    from keras.utils.data_utils import get_file
    from keras import backend as K
    from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
    from keras.optimizers import Adam, SGD
    from keras.preprocessing.image import ImageDataGenerator
    from sklearn.linear_model import LogisticRegression
```

```
from sklearn.svm import SVC
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import cross_validate
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score
import time
import logging
import argparse
import shutil
import tensorflow as tf
import keras.backend.tensorflow_backend as KTF
logger = logging.getLogger('BreakHist')
class SendMetrics(keras.callbacks.Callback):
def on_epoch_end(self, epoch, logs=):
nni.report_intermediate_result(logs["f1"])
seed_value = 42
os.environ['PYTHONHASHSEED']=str(seed_value)
np.random.seed(seed_value)
random.seed(seed_value)
tf.random.set_random_seed(seed_value)
config = tf.ConfigProto()
config.gpu_options.allow_growth=True
session = tf.Session(config=config)
KTF.set_session(session)
# Name list and magnification list.
magnification_list = ['40X', '100X', '200X', '400X']
benign_list = ['adenosis', 'fibroadenoma', 'phyllodes_tumor', 'tubular_adenoma']
```

malignant_list = ['ductal_carcinoma', 'lobular_carcinoma', 'mucinous_carcinoma', 'papillary_carcinoma']

cancer_list = benign_list + malignant_list

Model list.

models = [vgg16_model, vgg19_model, xception_model, resnet_model, inception_model, inception_resnet_model]

model_num = 3 # Select resnet as the backbone.

model_name = models[model_num].__name__

Set image size.

image_height=115

image_width=175

n_channels=3

def main(args):

Creater model dir.

if not os.path.exists("./models/"):

os.makedirs("./models/")

Get the timestamp and set it as weight name.

timestampEND = time.strftime("%H%M%S") + '-' + time.strftime("%d%m%Y")

weight_name = './models/weights-' + timestampEND +'.h5'

Hyper-parameters.

epochs = args['epochs']

batch_size = args['batch_size']

learning_rate = args['learning_rate']

optimizer = args['optimizer']

lr_decay = args['lr_decay']

dropout = args['dropout']

layer1 = args['layer1']

```
layer2 = args['layer2']
layer3 = args['layer3']
horizontal_flip = args['horizontal_flip']
vertical_flip = args['vertical_flip']
rotation_range = args['rotation_range']
shear_range = args['shear_range']
zoom_range = args['zoom_range']
is_search = args['search']
if optimizer == "ADAM":
optimizer_type = Adam
if optimizer == "SGD":
optimizer_type = SGD
iteration = 0
average_f1 = 0.0
for types in magnification_list:
if iteration == 0:
load_wt = "Yes"
else:
load_wt = "No"
# Load data.
training_images, training_labels,
validation_images, validation_labels,
testing_images, testing_labels =
data_split(magnification = types, validation_percent = 0.1, testing_percent = 0.2)
# Image augmentation.
datagen = ImageDataGenerator(
rotation_range=rotation_range,
```

```
shear_range=shear_range,
zoom_range=zoom_range,
horizontal_flip=horizontal_flip,
vertical_flip=vertical_flip,
)
datagen.fit(training_images)
# Build the model.
for i in range(len(models)):
if models[i].__name__ == model_name:
base_model = models[i]
base_model = base_model(image_height=image_height,image_width
=image_width,n_channels=n_channels,load_wt=load_wt)
# Add additional layers for classification.
x = base_model.output
x = Dense(2048, activation = 'relu')(x)
x = Dropout(dropout)(x)
if layer1 >0:
x = Dense(layer1, activation = 'relu')(x)
x = Dropout(dropout)(x)
if layer2 >0:
x = Dense(layer2, activation = 'relu')(x)
x = Dropout(dropout)(x)
if layer3 >0:
x = Dense(layer3, activation = 'relu')(x)
out = Dense(8, activation = 'softmax')(x)
inp = base_model.input
model = Model(inp, out)
```

```
    # Load model weight.
    try:
    model.load_weights(weight_name)
    print('Weights loaded!')
    except:
    print('No weights defined!')
    pass
    # Get the timestamp and set it as model name.
    model_timestamp = time.strftime("%H%M%S") + '-' + time.strftime("%d%m%Y")
    saved_model_name = './models/-.hdf5'.format(types, model_timestamp)
    model.compile(
    loss="categorical_crossentropy",
    optimizer=optimizer_type(lr=learning_rate),
    metrics=[f1,'accuracy'])
    early_stopping = EarlyStopping(patience=10, verbose=2)
    model_checkpoint = ModelCheckpoint(saved_model_name, save_best_only=True, ver-
bose=2)
    reduce_lr = ReduceLROnPlateau(factor=lr_decay, patience=5, verbose=2)
    history = model.fit_generator(
    datagen.flow(training_images, training_labels, batch_size=batch_size),
    steps_per_epoch=len(training_images)/ batch_size, validation_data=[validation_images,
validation_labels],
    callbacks=[early_stopping, model_checkpoint, reduce_lr, SendMetrics()], epochs=epochs)
    # Load the best model.
    model = keras.models.load_model(saved_model_name, custom_objects='f1': f1)
    # Get the final validation metrics at last step.
    val_loss, val_f1, val_acc = model.evaluate(validation_images, validation_labels)
```

```
    model.save_weights(weight_name)
    print("The validation metrics at last step: ")
    print("The validation accuracy for " + model_name + " with magnification "+ types +"
is ", val_acc, " with F1 score of ", val_f1, "
n")
    print()
    # Get the average val f1_score.
    average_f1 += val_f1 / 4.0
    # Print the metrics.
    if not is_search:
    # Print the training metrics.
    train_logits = model.predict(training_images)
    train_pred = np.argmax(train_logits, axis=1)
    train_true = np.argmax(training_labels, axis=1)
    print("="*15 + "  Training metrics ".format(types) + "="*15)
    print_metrics(train_true, train_pred)
    # Print the validation metrics.
    val_logits = model.predict(validation_images)
    val_pred = np.argmax(val_logits, axis=1)
    val_true = np.argmax(validation_labels, axis=1)
    print("="*15 + "  Valiation metrics ".format(types) + "="*15)
    print_metrics(val_true, val_pred)
    # Print the test metrics.
    test_logits = model.predict(testing_images)
    test_pred = np.argmax(test_logits, axis=1)
    test_true = np.argmax(testing_labels, axis=1)
    print("="*15 + "  Test metrics ".format(types) + "="*15)
```

```
print_metrics(test_true, test_pred)
# Destory the useless model.
iteration += 1
del model
keras.backend.clear_session()
# report final result
nni.report_final_result(average_f1)
if is_search:
# Delete all temp models.
shutil.rmtree('./models/')
def get_params():
# Training settings
parser = argparse.ArgumentParser(description='BreakHist')
# For model architecture.
parser.add_argument("–layer1", type=int, default=2048)
parser.add_argument("–layer2", type=int, default=512)
parser.add_argument("–layer3", type=int, default=32)
parser.add_argument("–dropout", type=float, default=0.3)
# For training hyper-parameters.
parser.add_argument('–batch_size', type=int, default=64)
parser.add_argument('–learning_rate', type=float, default=0.001)
parser.add_argument('–lr_decay', type=float, default=0.1)
parser.add_argument('–optimizer', type=str, default="ADAM")
parser.add_argument('–epochs', type=int, default=30)
# For data augmentation.
parser.add_argument('–horizontal_flip', type=bool, default=False)
parser.add_argument('–vertical_flip', type=bool, default=False)
```

```
parser.add_argument('–rotation_range', type=float, default=0.0)
parser.add_argument('–shear_range', type=float, default=0)
parser.add_argument('–zoom_range', type=float, default=0)
# Identify the phase: train or search.
parser.add_argument('–search', type=bool, default=False)
args, _ = parser.parse_known_args()
return args
if __name__ == '__main__':
try:
# get parameters form tuner
tuner_params = nni.get_next_parameter()
params = vars(get_params())
params.update(tuner_params)
main(params)
except Exception as exception:
logger.exception(exception)
raise
```

APPENDIX C

PROPOSED METHOD SOURCE CODE

```
import os
    os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
    os.environ["CUDA_VISIBLE_DEVICES"] = "1"
    import nni
    import numpy as np
    import pandas as pd
    import random
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    from skimage.transform import resize
    from models import *
    from data_processing import data_split
    from training_fn import *
    import keras
    from keras.layers import *
    from keras.models import *
    from keras import layers
    from keras.utils.data_utils import get_file
    from keras import backend as K
    from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
    from keras.optimizers import Adam, SGD
    from keras.preprocessing.image import ImageDataGenerator
    from sklearn.linear_model import LogisticRegression
```

```
from sklearn.svm import SVC
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import cross_validate
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score
import time
import logging
import argparse
import shutil
import tensorflow as tf
import keras.backend.tensorflow_backend as KTF
logger = logging.getLogger('BreakHist')
class SendMetrics(keras.callbacks.Callback):
def on_epoch_end(self, epoch, logs=):
nni.report_intermediate_result(logs["f1"])
seed_value = 42
os.environ['PYTHONHASHSEED']=str(seed_value)
np.random.seed(seed_value)
random.seed(seed_value)
tf.random.set_random_seed(seed_value)
config = tf.ConfigProto()
config.gpu_options.allow_growth=True
session = tf.Session(config=config)
KTF.set_session(session)
# Name list and magnification list.
magnification_list = ['40X', '100X', '200X', '400X']
benign_list = ['adenosis', 'fibroadenoma', 'phyllodes_tumor', 'tubular_adenoma']
```

malignant_list = ['ductal_carcinoma', 'lobular_carcinoma', 'mucinous_carcinoma', 'papillary_carcinoma']

cancer_list = benign_list + malignant_list

Model list.

models = [vgg16_model, vgg19_model, xception_model, resnet_model, inception_model, inception_resnet_model]

model_num = 3 # Select resnet as the backbone.

model_name = models[model_num].__name__

Set image size.

image_height=115

image_width=175

n_channels=3

def main(args):

Creater model dir.

if not os.path.exists("./models/"):

os.makedirs("./models/")

Get the timestamp and set it as weight name.

timestampEND = time.strftime("%H%M%S") + '-' + time.strftime("%d%m%Y")

weight_name = './models/weights-' + timestampEND +'.h5'

Hyper-parameters.

epochs = args['epochs']

batch_size = args['batch_size']

learning_rate = args['learning_rate']

optimizer = args['optimizer']

lr_decay = args['lr_decay']

dropout = args['dropout']

layer1 = args['layer1']

```
layer2 = args['layer2']
layer3 = args['layer3']
horizontal_flip = args['horizontal_flip']
vertical_flip = args['vertical_flip']
rotation_range = args['rotation_range']
shear_range = args['shear_range']
zoom_range = args['zoom_range']
is_search = args['search']
if optimizer == "ADAM":
optimizer_type = Adam
if optimizer == "SGD":
optimizer_type = SGD
iteration = 0
average_f1 = 0.0
for types in magnification_list:
if iteration == 0:
load_wt = "Yes"
else:
load_wt = "No"
# Load data.
training_images, training_labels,
validation_images, validation_labels,
testing_images, testing_labels =
data_split(magnification = types, validation_percent = 0.1, testing_percent = 0.2)
# Image augmentation.
datagen = ImageDataGenerator(
rotation_range=rotation_range,
```

```
shear_range=shear_range,
zoom_range=zoom_range,
horizontal_flip=horizontal_flip,
vertical_flip=vertical_flip,
)
datagen.fit(training_images)
# Build the model.
for i in range(len(models)):
if models[i].__name__ == model_name:
base_model = models[i]
base_model = base_model(image_height=image_height,image_width
=image_width,n_channels=n_channels,load_wt=load_wt)
# Add additional layers for classification.
x = base_model.output
x = Dense(2048, activation = 'relu')(x)
x = Dropout(dropout)(x)
if layer1 >0:
x = Dense(layer1, activation = 'relu')(x)
x = Dropout(dropout)(x)
if layer2 >0:
x = Dense(layer2, activation = 'relu')(x)
x = Dropout(dropout)(x)
if layer3 >0:
x = Dense(layer3, activation = 'relu')(x)
out = Dense(8, activation = 'softmax')(x)
inp = base_model.input
model = Model(inp, out)
```

```
# Load model weight.
try:
model.load_weights(weight_name)
print('Weights loaded!')
except:
print('No weights defined!')
pass
# Get the timestamp and set it as model name.
model_timestamp = time.strftime("%H%M%S") + '-' + time.strftime("%d%m%Y")
saved_model_name = './models/-.hdf5'.format(types, model_timestamp)
model.compile(
loss="categorical_crossentropy",
optimizer=optimizer_type(lr=learning_rate),
metrics=[f1,'accuracy'])
early_stopping = EarlyStopping(patience=10, verbose=2)
model_checkpoint = ModelCheckpoint(saved_model_name, save_best_only=True, ver-
bose=2)
reduce_lr = ReduceLROnPlateau(factor=lr_decay, patience=5, verbose=2)
history = model.fit_generator(
datagen.flow(training_images, training_labels, batch_size=batch_size),
steps_per_epoch=len(training_images) / batch_size,
validation_data=[validation_images, validation_labels],
callbacks=[early_stopping, model_checkpoint, reduce_lr, SendMetrics()],
epochs=epochs)
# Load the best model.
model = keras.models.load_model(saved_model_name, custom_objects='f1': f1)
# Get the final validation metrics at last step.
```

```
    val_loss, val_f1, val_acc = model.evaluate(validation_images, validation_labels)
    model.save_weights(weight_name)
    print("The validation metrics at last step: ")
    print("The validation accuracy for " + model_name + " with magnification "+ types +"
is ", val_acc, " with F1 score of ", val_f1, "
n")
    print()
    # Get the average val f1_score.
    average_f1 += val_f1 / 4.0
    # Print the metrics.
    if not is_search:
    # Print the training metrics.
    train_logits = model.predict(training_images)
    train_pred = np.argmax(train_logits, axis=1)
    train_true = np.argmax(training_labels, axis=1)
    print("="*15 + "  Training metrics ".format(types) + "="*15)
    print_metrics(train_true, train_pred)
    # Print the validation metrics.
    val_logits = model.predict(validation_images)
    val_pred = np.argmax(val_logits, axis=1)
    val_true = np.argmax(validation_labels, axis=1)
    print("="*15 + "  Valiation metrics ".format(types) + "="*15)
    print_metrics(val_true, val_pred)
    # Print the test metrics.
    test_logits = model.predict(testing_images)
    test_pred = np.argmax(test_logits, axis=1)
    test_true = np.argmax(testing_labels, axis=1)
```

```
print("="*15 + "  Test metrics ".format(types) + "="*15)
print_metrics(test_true, test_pred)
# Destory the useless model.
iteration += 1
del model
keras.backend.clear_session()
# report final result
nni.report_final_result(average_f1)
if is_search:
# Delete all temp models.
shutil.rmtree('./models/')
def get_params():
# Training settings
parser = argparse.ArgumentParser(description='BreakHist')
# For model architecture.
parser.add_argument("–layer1", type=int, default=64)
parser.add_argument("–layer2", type=int, default=0)
parser.add_argument("–layer3", type=int, default=64)
parser.add_argument("–dropout", type=float, default=0.10887510008284884)
# For training hyper-parameters.
parser.add_argument('–batch_size', type=int, default=16)
parser.add_argument('–learning_rate', type=float, default=0.0001)
parser.add_argument('–lr_decay', type=float, default=0.5)
parser.add_argument('–optimizer', type=str, default="ADAM")
parser.add_argument('–epochs', type=int, default=30)
# For data augmentation.
parser.add_argument('–horizontal_flip', type=bool, default=False)
```

```
parser.add_argument('–vertical_flip', type=bool, default=True)
parser.add_argument('–rotation_range', type=float, default=20)
parser.add_argument('–shear_range', type=float, default=0.06663526112980617)
parser.add_argument('–zoom_range', type=float, default=0.48854077282376673)
# Identify the phase: train or search.
parser.add_argument('–search', type=bool, default=False)
args, _ = parser.parse_known_args()
return args
if __name__ == '__main__':
try:
# get parameters form tuner
tuner_params = nni.get_next_parameter()
params = vars(get_params())
params.update(tuner_params)
main(params)
except Exception as exception:
logger.exception(exception)
raise
```

APPENDIX D

SEARCH SOURCE CODE

```
import os
    os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
    os.environ["CUDA_VISIBLE_DEVICES"] = "1"
    import nni
    import numpy as np
    import pandas as pd
    import random
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    from skimage.transform import resize
    from models import *
    from data_processing import data_split
    from training_fn import *
    import keras
    from keras.layers import *
    from keras.models import *
    from keras import layers
    from keras.utils.data_utils import get_file
    from keras import backend as K
    from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
    from keras.optimizers import Adam, SGD
    from keras.preprocessing.image import ImageDataGenerator
    from sklearn.linear_model import LogisticRegression
```

```
from sklearn.svm import SVC
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import cross_validate
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score
import time
import logging
import argparse
import shutil
import tensorflow as tf
import keras.backend.tensorflow_backend as KTF
logger = logging.getLogger('BreakHist')
class SendMetrics(keras.callbacks.Callback):
def on_epoch_end(self, epoch, logs=):
nni.report_intermediate_result(logs["f1"])
seed_value = 42
os.environ['PYTHONHASHSEED']=str(seed_value)
np.random.seed(seed_value)
random.seed(seed_value)
tf.random.set_random_seed(seed_value)
config = tf.ConfigProto()
config.gpu_options.allow_growth=True
session = tf.Session(config=config)
KTF.set_session(session)
# Name list and magnification list.
magnification_list = ['40X', '100X', '200X', '400X']
benign_list = ['adenosis', 'fibroadenoma', 'phyllodes_tumor', 'tubular_adenoma']
```

malignant_list = ['ductal_carcinoma', 'lobular_carcinoma', 'mucinous_carcinoma', 'papillary_carcinoma']

cancer_list = benign_list + malignant_list

Model list.

models = [vgg16_model, vgg19_model, xception_model, resnet_model, inception_model, inception_resnet_model]

model_num = 3 # Select resnet as the backbone.

model_name = models[model_num].__name__

Set image size.

image_height=115

image_width=175

n_channels=3

def main(args):

Creater model dir.

if not os.path.exists("./models/"):

os.makedirs("./models/")

Get the timestamp and set it as weight name.

timestampEND = time.strftime("%H%M%S") + '-' + time.strftime("%d%m%Y")

weight_name = './models/weights-' + timestampEND +'.h5'

Hyper-parameters.

epochs = args['epochs']

batch_size = args['batch_size']

learning_rate = args['learning_rate']

optimizer = args['optimizer']

lr_decay = args['lr_decay']

dropout = args['dropout']

layer1 = args['layer1']

```
    layer2 = args['layer2']
    layer3 = args['layer3']
    horizontal_flip = args['horizontal_flip']
    vertical_flip = args['vertical_flip']
    rotation_range = args['rotation_range']
    shear_range = args['shear_range']
    zoom_range = args['zoom_range']
    is_search = args['search']
    if optimizer == "ADAM":
    optimizer_type = Adam
    if optimizer == "SGD":
    optimizer_type = SGD
    iteration = 0
    average_f1 = 0.0
    for types in magnification_list:
    if iteration == 0:
    load_wt = "Yes"
    else:
    load_wt = "No"
    # Load data.
    training_images, training_labels, validation_images, validation_labels, testing_images,
testing_labels =
    data_split(magnification = types, validation_percent = 0.1, testing_percent = 0.2)
    # I
    mage augmentation.
    datagen = ImageDataGenerator(
    rotation_range=rotation_range,
```

```
shear_range=shear_range,
zoom_range=zoom_range,
horizontal_flip=horizontal_flip,
vertical_flip=vertical_flip,
)
datagen.fit(training_images)
# Build the model.
for i in range(len(models)):
if models[i].__name__ == model_name:
base_model = models[i]
base_model = base_model(image_height=image_height,image_width
=image_width,n_channels=n_channels,load_wt=load_wt)
# Add additional layers for classification.
x = base_model.output
x = Dense(2048, activation = 'relu')(x)
x = Dropout(dropout)(x)
if layer1 >0:
x = Dense(layer1, activation = 'relu')(x)
x = Dropout(dropout)(x)
if layer2 >0:
x = Dense(layer2, activation = 'relu')(x)
x = Dropout(dropout)(x)
if layer3 >0:
x = Dense(layer3, activation = 'relu')(x)
out = Dense(8, activation = 'softmax')(x)
inp = base_model.input
model = Model(inp, out)
```

```
    # Load model weight.
    try:
    model.load_weights(weight_name)
    print('Weights loaded!')
    except:
    print('No weights defined!')
    pass
    # Get the timestamp and set it as model name.
    model_timestamp = time.strftime("%H%M%S") + '-' + time.strftime("%d%m%Y")
    saved_model_name = './models/-.hdf5'.format(types, model_timestamp)
    model.compile(
    loss="categorical_crossentropy",
    optimizer=optimizer_type(lr=learning_rate),
    metrics=[f1,'accuracy'])
    early_stopping = EarlyStopping(patience=10, verbose=2)
    model_checkpoint = ModelCheckpoint(saved_model_name, save_best_only=True, ver-
bose=2)
    reduce_lr = ReduceLROnPlateau(factor=lr_decay, patience=5, verbose=2)
    history = model.fit_generator(
    datagen.flow(training_images, training_labels, batch_size=batch_size),
    steps_per_epoch=len(training_images) / batch_size,
    validation_data=[validation_images, validation_labels],
    callbacks=[early_stopping, model_checkpoint, reduce_lr, SendMetrics()],
    epochs=epochs)
    # Load the best model.
    model = keras.models.load_model(saved_model_name, custom_objects='f1': f1)
    # Get the final validation metrics at last step.
```

```
    val_loss, val_f1, val_acc = model.evaluate(validation_images,
    validation_labels) model.save_weights(weight_name)
    print("The validation metrics at last step: ")
    print("The validation accuracy for " + model_name + " with magnification "+ types +"
is ", val_acc, " with F1 score of ", val_f1, "
n")
    print()
    # Get the average val f1_score.
    average_f1 += val_f1 / 4.0
    # Print the metrics.
    if not is_search:
    # Print the training metrics.
    train_logits = model.predict(training_images)
    train_pred = np.argmax(train_logits, axis=1)
    train_true = np.argmax(training_labels, axis=1)
    print("="*15 + "  Training metrics ".format(types) + "="*15)
    print_metrics(train_true, train_pred)
    # Print the validation metrics.
    val_logits = model.predict(validation_images)
    val_pred = np.argmax(val_logits, axis=1)
    val_true = np.argmax(validation_labels, axis=1)
    print("="*15 + "  Valiation metrics ".format(types) + "="*15)
    print_metrics(val_true, val_pred)
    # Print the test metrics.
    test_logits = model.predict(testing_images)
    test_pred = np.argmax(test_logits, axis=1)
    test_true = np.argmax(testing_labels, axis=1)
```

```
print("="*15 + "  Test metrics ".format(types) + "="*15)
print_metrics(test_true, test_pred)
# Destory the useless model.
iteration += 1
del model
keras.backend.clear_session()
# report final result
nni.report_final_result(average_f1)
if is_search:
# Delete all temp models.
shutil.rmtree('./models/')
def
get_params():
# Training settings
parser = argparse.ArgumentParser(description='BreakHist')
# For model architecture.
parser.add_argument("–layer1", type=int, default=0)
parser.add_argument("–layer2", type=int, default=0)
parser.add_argument("–layer3", type=int, default=0)
parser.add_argument("–dropout", type=float, default=0.3)
# For training hyper-parameters.
parser.add_argument('–batch_size', type=int, default=32)
parser.add_argument('–learning_rate', type=float, default=0.0001)
parser.add_argument('–lr_decay', type=float, default=0.1)
parser.add_argument('–optimizer', type=str, default="ADAM")
parser.add_argument('–epochs', type=int, default=30)
# For data augmentation.
```

```
parser.add_argument('–horizontal_flip', type=bool,
default=False)
parser.add_argument('–vertical_flip', type=bool, default=False)
parser.add_argument('–rotation_range', type=float, default=0.0)
parser.add_argument('–shear_range', type=float, default=0)
parser.add_argument('–zoom_range', type=float, default=0)
# Identify the phase: train or search.
parser.add_argument('–search', type=bool, default=True)
args, _ = parser.parse_known_args()
return args
if __name__ == '__main__':
try:
# get parameters form tuner
tuner_params = nni.get_next_parameter()
params = vars(get_params())
params.update(tuner_params)
main(params)
except Exception as exception:
logger.exception(exception)
raise
```

APPENDIX E

EVLATION CODE

```
import numpy as np
    import pandas as pd
    import random
    import os
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    from skimage.transform import resize
    from keras.layers import *
    from keras.models import *
    from keras import layers
    from keras.utils.data_utils import get_file
    from keras import backend as K
    from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
    from keras.optimizers import Adam
    from sklearn.linear_model import LogisticRegression
    from sklearn.svm import SVC
    from sklearn.feature_selection import SelectFromModel
    from sklearn.model_selection import cross_validate
    from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, confusion_matrix
    from data_processing import data_split
    from models import *
    import tensorflow as tf
    seed_value = 42
```

```
os.environ['PYTHONHASHSEED']=str(seed_value)
np.random.seed(seed_value)
random.seed(seed_value)
tf.random.set_random_seed(seed_value)
models = [vgg16_model, vgg19_model, xception_model, resnet_model, inception_model,
inception_resnet_model]
def counts_from_confusion(confusion):
"""
Obtain TP, FN FP, and TN for each class in the confusion matrix
"""
counts_list = []
# Iterate through classes and store the counts
for i in range(confusion.shape[0]):
tp = confusion[i, i]
fn_mask = np.zeros(confusion.shape)
fn_mask[i, :] = 1
fn_mask[i, i] = 0
fn = np.sum(np.multiply(confusion, fn_mask))
fp_mask = np.zeros(confusion.shape)
fp_mask[:, i] = 1
fp_mask[i, i] = 0
fp = np.sum(np.multiply(confusion, fp_mask))
tn_mask = 1 - (fn_mask + fp_mask)
tn_mask[i, i] = 0
tn = np.sum(np.multiply(confusion, tn_mask))
counts_list.append('Class': i,
'TP': tp,
```

```
'FN': fn,
'FP': fp,
'TN': tn)
return counts_list
def
print_metrics(y_ture, y_pred):
cm = confusion_matrix(y_ture, y_pred).astype('float')
results = counts_from_confusion(cm)
Get the metrics.
met = []
acc_avg, precision_avg, recall_avg, f1_avg, sen_avg, spe_avg, cc_avg = 0, 0, 0, 0, 0, 0, 0
for stat in results:
acc = (stat['TP'] + stat['TN']) / (stat['TP'] + stat['TN'] + stat['FP'] + stat['FN'])
precision = stat['TP'] / (stat['TP'] + stat['FP'])
recall = stat['TP'] / (stat['TP'] + stat['FN'])
f1_score = 2 * precision * recall / (precision + recall)
sensitivity = stat['TP'] / (stat['TP']+stat['FN'])
specificity = stat['TN'] / (stat['TN']+stat['FP'])
cc = ((stat['TP']*stat['TN']) - (stat['FN']*stat['FP']))/
((stat['TP']+stat['FN'])*(stat['TN']+stat['FP'])*
(stat['TP']+stat['FP'])*(stat['TN']+stat['FN']))**0.5
acc_avg += acc*1.0/len(results)
precision_avg += precision*1.0/len(results)
recall_avg += recall*1.0/len(results)
f1_avg += f1_score*1.0/len(results)
sen_avg += sensitivity*1.0/len(results)
spe_avg += specificity*1.0/len(results)
```

```
cc_avg += cc*1.0/len(results)
met.append(
'Class': stat['Class'],
'acc': acc,
'precision': precision,
'recall': recall,
'f1_score': f1_score,
'sensitivity': sensitivity,
'specificity': specificity,
'cc': cc,
)
# Print the metrics.
print()
print('0. Confusion matrix: ')
print(cm)
print()
print ('1. Accuracy mean ', acc_avg)
for i in range (0, len(met)):
print (met[i]['Class'], ' ', met[i]['acc'])
print()
print ('2. F1 score mean ', f1_avg)
for i in range (0, len(met)):
print (met[i]['Class'], ' ', met[i]['f1_score'])
print()
print ('3. precision score mean ', precision_avg)
for i in range (0, len(met)):
print (met[i]['Class'], ' ', met[i]['precision'])
```

```
print()
print ('4. recall score mean ', recall_avg)
for i in range (0, len(met)):
print (met[i]['Class'], ' ', met[i]['recall'])
print()
print ('5. sensitivities score mean ', sen_avg)
for i in range (0, len(met)):
print (met[i]['Class'], ' ', met[i]['sensitivity'])
print()
print ('6. specificity score mean ', spe_avg)
for i in range (0, len(met)):
print (met[i]['Class'], ' ', met[i]['specificity'])
print()
print ('7. correlation coefficient mean ', cc_avg)
for i in range (0, len(met)):
print (met[i]['Class'], ' ', met[i]['cc'])
print()
def f1(y_true, y_pred):
def recall(y_true, y_pred):
true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
recall = true_positives / (possible_positives + K.epsilon())
return recall
def precision(y_true, y_pred):
true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
precision = true_positives / (predicted_positives + K.epsilon())
```

```
return precision
precision = precision(y_true, y_pred)
recall = recall(y_true, y_pred)
return 2*((precision*recall)/(precision+recall+K.epsilon()))
def compile_n_fit(validation_percent, testing_percent,
image_height, image_width, n_channels, load_wt,dropout =
0.3, model_name = 'vgg16_model', magnification = '40X'):
Load data.
training_images, training_labels, validation_images,
validation_labels, testing_images, testing_labels =
data_split(magnification = magnification, validation_percent = validation_percent,
testing_percent = testing_percent)
for i in range(len(models)):
if models[i].__name__ == model_name:
base_model = models[i]
base_model = base_model(image_height=image_height,image_width
=image_width,n_channels=n_channels,load_wt=load_wt)
x = base_model.output
x = Dense(2048, activation = 'relu')(x)
x = Dropout(dropout)(x)
x = Dense(512, activation = 'relu')(x)
x = Dropout(dropout)(x)
x = Dense(128, activation = 'relu')(x)
x = Dropout(dropout)(x)
x = Dense(32, activation = 'relu')(x)
out = Dense(8, activation = 'softmax')(x)
inp = base_model.input
```

```
    model = Model(inp,out)
    try:
    model.load_weights(model_name + '_weight_1.h5')
    print('Weights loaded!')
    except:
    print('No weights defined!')
    pass
    model.compile(loss="categorical_crossentropy",
    optimizer=Adam(lr=0.0001), metrics=[f1,'accuracy'])
    early_stopping = EarlyStopping(patience=10, verbose=2)
    model_checkpoint = ModelCheckpoint(model_name + "_combine" +".model", save_best_only=True,
verbose=2)
    reduce_lr = ReduceLROnPlateau(factor=0.1, patience=5, verbose=2) #min_lr=0.00001,
    epochs = 100
    batch_size = 64
    history = model.fit(training_images, training_labels, validation_data=[validation_images,
validation_labels],
    epochs=epochs,
    verbose = 0,
    batch_size=batch_size,
    callbacks=[early_stopping,
    model_checkpoint, reduce_lr])
    test_loss, test_acc, test_f1 = model.evaluate(testing_images, testing_labels)
    model.save_weights(model_name + '_weight_1.h5')
    print("
nThe test accuracy for " + model_name + " with magnification "+ magnification +" is ",
test_acc, " with F1 score of ", test_f1, "
```

n")

REFERENCES

[1] R. Guan, X. Wang, M. Q. Yang, Y. Zhang, F. Zhou, C. Yang, and Y. Liang, "Multi-label deep learning for gene function annotation in cancer pathways," *Scientific reports*, vol. 8, no. 1, pp. 1–9, 2018.

[2] D. Hanahan, "Rethinking the war on cancer," *The Lancet*, vol. 383, no. 9916, pp. 558–563, 2014.

[3] M. DeBonis, "Congress passes 21st century cures act, boosting research and easing drug approvals," *The Washington Post*, 2016.

[4] J. Biden, *Inspiring a new generation to defy the bounds of innovation: A moonshot to cure cancer*, 2016.

[5] D. Howe, M. Costanzo, P. Fey, T. Gojobori, L. Hannick, W. Hide, D. P. Hill, R. Kania, M. Schaeffer, S. St Pierre, *et al.*, "The future of biocuration," *Nature*, vol. 455, no. 7209, pp. 47–50, 2008.

[6] C. for Disease Control, Prevention, *et al.*, "United states cancer statistics: Data visualizations," *Changes Over Time: Colon and Rectum. Available online: https://gis. cdc. gov/Cancer/USCS/DataViz. html (accessed on 21 November 2019)*, 2018.

[7] C. E. DeSantis, J. Ma, M. M. Gaudet, L. A. Newman, K. D. Miller, A. Goding Sauer, A. Jemal, and R. L. Siegel, "Breast cancer statistics, 2019," *CA: a cancer journal for clinicians*, vol. 69, no. 6, pp. 438–451, 2019.

[8] P. Boyle, B. Levin, *et al.*, *World cancer report 2008*. IARC Press, International Agency for Research on Cancer, 2008.

[9] J. V. Lacey, A. R. Kreimer, S. S. Buys, P. M. Marcus, S.-C. Chang, M. F. Leitzmann, R. N. Hoover, P. C. Prorok, C. D. Berg, P. Hartge, *et al.*, "Breast cancer epidemiology according to recognized breast cancer risk factors in the prostate, lung, colorectal and ovarian (plco) cancer screening trial cohort," *BMC cancer*, vol. 9, no. 1, p. 84, 2009.

[10] B. O. Anderson, C.-H. Yip, R. A. Smith, R. Shyyan, S. F. Sener, A. Eniu, R. W. Carlson, E. Azavedo, and J. Harford, "Guideline implementation for breast healthcare in low-income and middle-income countries: Overview of the breast health global initiative global summit 2007," *Cancer*, vol. 113, no. S8, pp. 2221–2243, 2008.

[11] *Health and its significance*.

[12] W. M. Wells III, *Medical image analysis–past, present, and future*, 2016.

[13] A. Madabhushi and G. Lee, *Image analysis and machine learning in digital pathology: Challenges and opportunities*, 2016.

[14] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," *Annual review of biomedical engineering*, vol. 19, pp. 221–248, 2017.

[15] H. Greenspan, B. Van Ginneken, and R. M. Summers, "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1153–1159, 2016.

[16] J. Weese and C. Lorenz, *Four challenges in medical image analysis from an industrial perspective*, 2016.

[17] D. Rueckert, B. Glocker, and B. Kainz, *Learning clinically useful information from images: Past, present and future*, 2016.

[18] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical image analysis*, vol. 42, pp. 60–88, 2017.

[19] PubMed, *Us national library of medicine national institutes of health*, 2019.

[20] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[21] 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.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, Springer, 2014, pp. 818–833.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[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] 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.

[28] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[29] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[30] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[31] M. L. GIGER, "Computer-aided diagnosis in mammography," *Handbook of medical imaging*, 2000.

[32] S. Astley and F. J. Gilbert, "Computer-aided detection in mammography," *Clinical radiology*, vol. 59, no. 5, pp. 390–399, 2004.

[33] T. W. Freer and M. J. Ulissey, "Screening mammography with computer-aided detection: Prospective study of 12,860 patients in a community breast center," *Radiology*, vol. 220, no. 3, pp. 781–786, 2001.

[34] M. Giger, "Overview of computer-aided diagnosis in breast imaging," *Comput.-Aided Diagnosis in Medical Imaging*, pp. 167–176, 1998.

[35] S. Kumar, R. Moni, and J. Rajeesh, "Automatic liver and lesion segmentation: A primary step in diagnosis of liver diseases," *Signal, Image and Video Processing*, vol. 7, no. 1, pp. 163–172, 2013.

[36] ——, "An automatic computer-aided diagnosis system for liver tumours on computed tomography images," *Computers & Electrical Engineering*, vol. 39, no. 5, pp. 1516–1526, 2013.

[37] J. Vincey and M. Jeba, "Computer aided diagnosis for liver cancer feature extraction," *Int J Eng Sci*, vol. 11, pp. 27–30, 2013.

[38] J. Stoitsis, I. Valavanis, S. G. Mougiakakou, S. Golemati, A. Nikita, and K. S. Nikita, "Computer aided diagnosis based on medical image processing and artificial intelligence methods," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 569, no. 2, pp. 591–595, 2006.

[39] S. Pereira, A. Pinto, V. Alves, and C. A. Silva, "Brain tumor segmentation using convolutional neural networks in mri images," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1240–1251, 2016.

[40] M. F. Keskenler and E. F. Keskenler, "Geçmişten günümüze yapay sinir ağları ve tarihçesi," *Takvim-i Vekayi*, vol. 5, no. 2, pp. 8–18, 2017.

[41] N. S. Ismail and C. Sovuthy, "Breast cancer detection based on deep learning,"

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

[43] T. Araújo, G. Aresta, E. Castro, J. Rouco, P. Aguiar, C. Eloy, A. Polónia, and A. Campilho, "Classification of breast cancer histology images using convolutional neural networks," *PloS one*, vol. 12, no. 6, e0177544, 2017.

[44] J. Chang, J. Yu, T. Han, H.-j. Chang, and E. Park, "A method for classifying medical images using transfer learning: A pilot study on histopathology of breast cancer," in *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*, IEEE, 2017, pp. 1–4.

[45] M. Elter and E. Haßlmeyer, "A knowledge-based approach to cadx of mammographic masses," in *Medical Imaging 2008: Computer-Aided Diagnosis*, International Society for Optics and Photonics, vol. 6915, 2008, p. 69150L.

[46] M. T. N. Uppal, "Classification of mammograms for breast cancer detection using fusion of discrete cosine transform and discrete wavelet transform features.," 2016.

[47] B. Sahiner, H.-P. Chan, N. Petrick, D. Wei, M. A. Helvie, D. D. Adler, and M. M. Goodsitt, "Classification of mass and normal breast tissue: A convolution neural network classifier with spatial domain and texture images," *IEEE transactions on Medical Imaging*, vol. 15, no. 5, pp. 598–610, 1996.

[48] K. Petersen, M. Nielsen, P. Diao, N. Karssemeijer, and M. Lillholm, "Breast tissue segmentation and mammographic risk scoring using deep learning," in *International workshop on digital mammography*, Springer, 2014, pp. 88–94.

[49] A. R. Jamieson, K. Drukker, and M. L. Giger, "Breast image feature learning with adaptive deconvolutional networks," in *Medical Imaging 2012: Computer-Aided Diagnosis*, International Society for Optics and Photonics, vol. 8315, 2012, p. 831 506.

[50] A. Mert, N. Kılıç, E. Bilgili, and A. Akan, "Breast cancer detection with reduced feature set," *Computational and mathematical methods in medicine*, vol. 2015, 2015.

[51] M. H. Motlagh, M. Jannesari, H. Aboulkheyr, P. Khosravi, O. Elemento, M. Totonchi, and I. Hajirasouliha, "Breast cancer histopathological image classification: A deep learning approach," *BioRxiv*, p. 242 818, 2018.

[52] A. Titoriya and S. Sachdeva, "Breast cancer histopathology image classification using alexnet," in *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, IEEE, 2019, pp. 708–712.

[53] S. Singh, J. Harini, and B. Surabhi, "A novel neural network based automated system for diagnosis of breast cancer from real time biopsy slides," in *International Conference on Circuits, Communication, Control and Computing*, IEEE, 2014, pp. 50–53.

[54] *Evolutionary algorithm*.

[55] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 2902–2911.

[56] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE transactions on Neural Networks*, vol. 5, no. 1, pp. 54–65, 1994.

[57] *Nni framework*.

[58] F. Ritter, T. Boskamp, A. Homeyer, H. Laue, M. Schwier, F. Link, and H.-O. Peitgen, "Medical image analysis," *IEEE pulse*, vol. 2, no. 6, pp. 60–70, 2011.

[59] Y. H. Yoon and J. C. Ye, "Deep learning for accelerated ultrasound imaging," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 6673–6676.

[60] P. Sasmal, Y. Iwahori, M. Bhuyan, and K. Kasugai, "Classification of polyps in capsule endoscopic images using cnn," in *2018 IEEE Applied Signal Processing Conference (ASPCON)*, IEEE, 2018, pp. 253–256.

[61] P. Cui, Z. Guo, J. Xu, T. Li, Y. Shi, W. Chen, T. Shu, and J. Lei, "Tissue recognition in spinal endoscopic surgery using deep learning," in *2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST)*, IEEE, 2019, pp. 1–5.

[62] A. KahsayGebreslassie, M. T. Hagos, *et al.*, "Automated gastrointestinal disease recognition for endoscopic images," in *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, IEEE, 2019, pp. 312–316.

[63] T. Treebupachatsakul and S. Poomrittigul, "Microorganism image recognition based on deep learning application," in *2020 International Conference on Electronics, Information, and Communication (ICEIC)*, IEEE, 2020, pp. 1–5.

[64] *Rectifier (neural networks)*.