Deep learning method for Breast Cancer Classification

Libo Xu and Md Rajibul Islam

Background

Breast cancer is one of the leading causes of women death all over the world. Which roots about 523 thousand death and addition of 2.4 million new patients in each year [1]. In USA, breast cancer is the most frequently diagnosis cancer among women and it has been remarked as the second major cause of cancer death after lung and bronchus cancer, mentioned by Alom et all in [2]. Early diagnosis of any disease has a very positive impact and reduces the death rate significantly. Breast cancer detection in earlier stages is quite curable. But the technique that are still using in clinical practices are time consuming and questionable concerning the accuracy. Computer-aided diagnosis (CAD) system is becoming quite popular in this filed. This might be very useful regarding the efficiency as well as the diagnosis accuracy. ResNet deep learning approach with proper parameter settings for cancer detection is more reliable and effective strategy compared to the other conventional approaches [1].

Dataset

Original datasets

The datasets are achieved from the website `pyimagesearch.com`. According to the website, dataset has been prepared from 279 patients. There are total 277523 patches of 50×50 pixels images. The image data files are on the following format: $8863\_idx5\_x1001\_y801\_class1.png$. Where each part of the file represents specific meaning and it has been remarked as follows:
Patient ID: 8883_idx5
x-coordinate of the crop: 1001
y-coordinate of the crop: 801
Class label: 1 (1 indicates positive result and 0 indicates no cancer)

Dataset preparation

The dataset are downloaded and prepared in our local computer. In this process, three folders are created and named as *train, test and validation* and each of this three folder contain two sub-folders 0 and 1. Then all the data files are copied and data files with negative results are placed into 0 and the positive results are copied into 1 under the train folder. Then 20% dataset are moved from train to the corresponding test folder. Finally, from the remaining dataset of the train, 10% are passed to the corresponding validation folders. For this work, the dataset statistic is presented on the following table.

container	negative example (0)	positive example (1)	total data files	total files in %
train	143091	56725	199816	72 %
test	39748	15757	55505	20 %
validation	15899	6303	22202	8 %
total	198738	78785	**277523**	
dataset in %	71.71 %	28.39 %		

From the the table, it can be seen that there are about 72 % data files are with negative examples. Whereas, only about 28 % data files are with positive results. Therefore, a significant data imbalanced is remarked.

Method

For this classification problem, we choose to use ResNet as our deep learning model. As you know, in the deep network, the size of the output feature map of each layer varies greatly with the depth of the network, which means the height and width of the inputs are smaller and the depth of the output feature map increases as the network goes deeper. On the other hand, the reduction in the size of input in each layer can reduce the amount of computation. Besides, the number of available features also increases when the depth of feature map increases. However, the network having too many layers may result in the network degradation. But the deep residual networks can fix this problem.

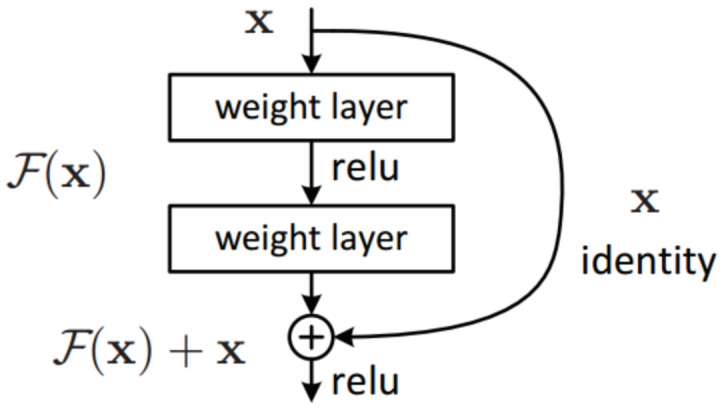

Figure 1: Shortcut connection of ResNet

There are five different architectures of ResNet, which is ResNet-18/34/50/101/152. By stacking different numbers of residual building blocks in the figures above, these networks could be realized. Among them ResNet-34/50 consists of basic residual blocks and Res-50/101/152 consists of bottleneck building blocks. We apply ResNet-18 for our project. The architecture of ResNet-18 is shown in the figures below.

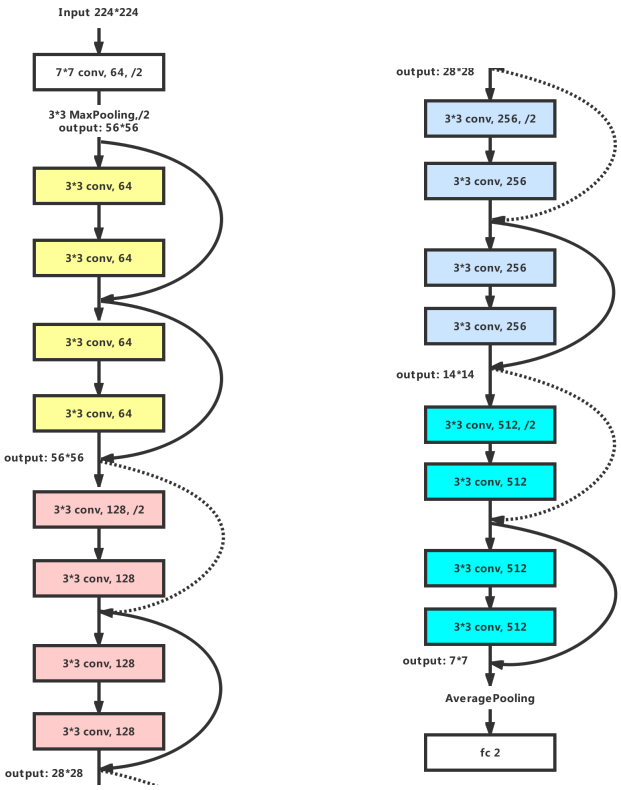

Figure 2: ResNet18

ResNet-18 consists of 17 convolutional layers and 1 fully-connected layer. It also contains 1 root block and 4 stacks including 2 basic building blocks. The dotted curve means that the size of identity and the size of output feature map of next block are not matched so the size of identity need to be reduced. The default size of input image is 224×224, so the size of output after going through all convolutional layers would be 7×7, to which the average pooling would be applied and then it will be input to the fully-connected layers to output the vector having the size of number of classes, which is 2 in this case.

Optimizer

Application of ImageDataGenerator

First of all, the size of our dataset is around 2GB. In order not to store the train and validation data to different parameters in memory every time we run the code for training, we use ImageDataGenerator from Keras, in which way the images can be called from the directories by using flow_from_directory. Besides, because the labels of the images are marked in the name of each file, label files need to be created manually. However, flow_from_directory method could assign the label for each file as long as the images are stored in the corresponding subfolder in the directory and the subfolders should be named as the name of classes, which is 0 and 1 in this case. What's more, there are almost 200000 images in the training dataset. At first we ran the code on the whole dataset and it took 2 hours to finish 1 epoch. To save time to test different settings of parameters, we decided to take 15000 images randomly from the training set and 2250 images from validation set. Among these images the distribution of two classes is negative example(0) : positive example(1) = 2:1. Since the size of the dataset decreases, ImageDataGenerator could also implement image augmentation to increase the number of training images to improve the performance of the model to some extent.

Parameters setting

batch size	lr	epochs	loss	optimizer	metrics
32	1e-5	150	categorical_crossentropy	Adam	accuracy

For this task, we choose a relatively large batch size because the larger batch size could increase the speed of data processing and the direction of the gradient decline would be more accurate which would minimize the training oscillations. However, if the training process want to achieve the same accuracy as those with smaller batch size, more epochs are needed. So during the testing for the number of epochs, we use 40, 50 and 80 but the learning curve does not converge so the epochs is set to 150. As for the learning rate, at first we set it as 1e-2, 1e-3 and 1e-4 but they all result in large oscillations in the learning curve. Hence, the 1e-5 is applied for the learning rate because it allows the training converge to better performance despite that more epochs are needed. whats more, categorical cross entropy loss function is applied because softmax activation function is used in the dense layer of ResNet.

Imbalance data issue

Among the whole dataset, 71.71% of them are negative(0) and 28.39% of them are positive. Therefore, the data is not balanced in this case, which would affect the performance of the model. To reduce the impact of imbalance on model performance, class_weight is calculated. Class_weight is an argument in fit_generator method, which is able to tell the model to pay more attention to samples from an under-represented class during the training. After we extract a smaller dataset for training, the class_weight is [1., 2.] in this case.

Evaluation

Our result has been evaluated by using the confusion matrix. A confusion matrix is a specific table that visualize the performance of the algorithm. It contains two rows and two columns and reports the false positive, false negative, true positive and true negative statistic. Which can be represented as follows.

True positives (TP)	False negative (FN)
False positive (FP)	True negatives (TN)

If the number of real positive cases in the data is represented by P and the number of real negative cases in the data is N then we can compute the following parameters as below:
accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
sensitivity/recall (True positive rate) = $\frac{TP}{P} = \frac{TP}{TP+FN}$
specificity (True negative rate) = $\frac{TN}{N} = \frac{TN}{TN+FP}$
precision (Positive predictive value) = $\frac{TP}{TP+FP}$
F_1 score = $\frac{2TP}{2TP+FP+FN}$

Results

From our experiment the following results are achieved.

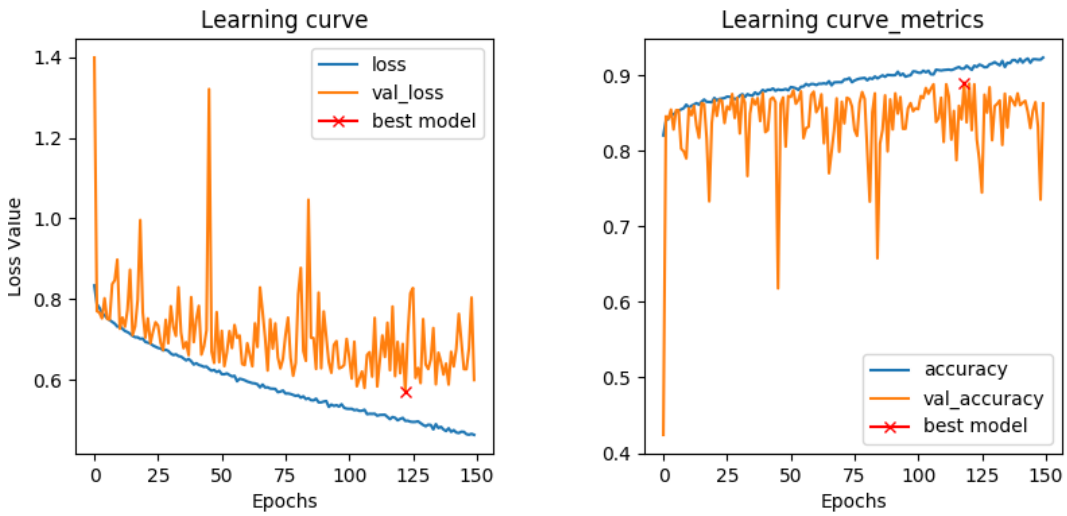

Figure 3: Learning curves of loss & accuracy

Validation loss : 0.65696
Validation accuracy : 0.8901

Confusion matrix:

TP: 35251	FN: 4468
FP: 3140	TN: 12646

Results comparison

Parameters	our results	pyimagesearch.com
Accuracy	0.8629	0.8483
Recall	0.8875	0.8503
Precision	0.9182	N/A
Specificity	0.8011	0.8470
F_1	0.9026	N/A

From the Figure [3], we can see that the learning curves of the validation set are fluctuating in some range, but they tend to converge. And from the learning curves of training, the model is still learning. Hence, the model is a little bit overfitting and we still need to fix the parameter settings.

Compared to the results from the blog where our topic of the project derives, we have higher accuracy and recall values than he does, but the value of specificity is lower, which means our model does not perform well enough on predicting the images with cancer because this is a under-represented class. Therefore the imbalanced data issue still affects the performance of our model.

Future work

In spite having quite good achievement in this work, we are still away from some issues concerning this project. First of all, we need to figure out the application of the decay of learning rate in the Adam optimizer. Secondly, deeper ResNet architectures like ResNet-34 or even ResNet-152 might be tested to train the dataset. Last but not least, the whole dataset which is more than 2,70,000 images need to be used for training .

Bibliography

[1] Mehadi HM; Mahboobeh J; et al. *Breast Cancer Histopathological Image Classification: A Deep Learning Approach*, Jan 4' 2018.

[2] MZ Alom, C Yakopcic et al. *Breast Cancer Classification from Histopathological Image with Inception Recurrent Residual Convolution Neural Network*, Nov 10' 2018.