Medical Image Processing with Deep Learning: Mammogram Classification and Automatic Lesion Detection

Yijie Jin
City University of Hong Kong
yijiejin3-c@my.cityu.edu.hk

Yifan Zheng
The Chinese University of Hong Kong
ethanzhengyf@gmail.com

August 2, 2019

Abstract

Breast cancer is the most common cancer for women in the world [1]. Deep learning has been a tremendous success in computer vision and also has some applications in medical imaging. We trained and evaluated several convolutional neural networks for mammogram classification and tumor detection. We used the MIAS database which contains 322 mammograms. Before the training, we performed data augmentation including cropping, rotating and flipping, followed by discarding part of images without significant information according to several thresholds. We modified the VGG16 and SSD algorithms to fit our dataset, and obtained mAP of 0.842 in tumor detection. In addition, some trials of changing hyperparameters and model structure are made to accelerate the training process. Our implementation is available at: `https://bitbucket.org/EDKLW/medical-images-2019-a/`

1 Introduction

1.1 Background

Approximately one eighth of women in the U.S. will develop breast cancer during their lifetimes. The mortality of breast cancer largely depends on whether the lesion could

be detected at an early stage. Mammography is the process of using low-energy X-rays to examine human breast for diagnosis and screening. We need well-trained radiologists to examine the CT images traditionally, which is costly and time-consuming. Adopting computer aided detection system can accelerate the diagnosing process as well as enhancing the diagnosis accuracy.

Deep learning has been a tremendous success in image processing and has many applications such as image reconstruction, object detection etc. As the performance of deep neural network is reaching or even surpassing human performance, it provides possibilities to apply it to medical imaging area.

1.2 Objectives

1. Classify the mammograms into 3 categories, normal, benign and malignant.
2. Automatically detect the tumorous lesion without prior information of the presence of a cancerous lesion.

1.3 Related Work

Computer aid diagnosis has been a popular research area in recent years with the development of computing power and application of machine learning algorithms. We reviewed some work done by other researchers. P Xi. et.al. [2] adopted VGGNet for classification and ResNet for localizing abnormalities. N Wu. et. al. [3] applied ResNet-22 for lesion detection and they also build a model which combines the prediction of neural network and interpretations by radiologists.

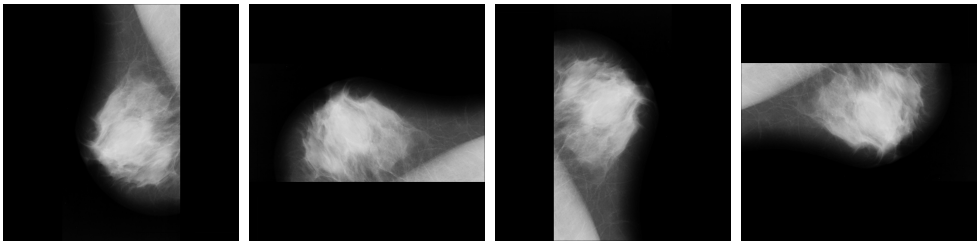
2 Data

Dataset In the study of mammogram classification, a lot of researchers use their own databases. However, due to some privacy reasons, most databases for mammograms are not public. In our study, we applied our algorithm on the benchmark database, Mammographic Image Analysis Society (MIAS) database. The database comprises 322 images with labels marked by radiologists. Each image has the information of its categories (normal, benign or malignant), the coordinates of the tumor center and the radius of the tumor.

Preprocessing Although all of the images include a feature-wise label, some of the cancerous cases have no information of the tumor center and tumor radius. Besides, several images has more than one tumor in them, which brings difficulty to label them.

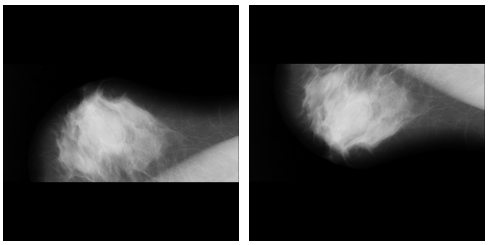
Hence, these images were discarded primarily. The remaining images include 209 normal, 56 benign and 46 malignant cases.

Data Augmentation As the database has only around 300 images, which is far from enough for training a neural network. It's essential to apply some data augmentation techniques to increase the number of images before training. We've rotated the images by 90, 180 and 270 degrees respectively and flipped them vertically afterwards. After this process, we could get 8 images out of 1.



(a) Original (b) Rotated by 90° (c) Rotated by 180° (d) Rotated by 270°

Figure 1: Data Augmentation: Rotation



(a) Original (b) Flipped

Figure 2: Data Augmentation: Flip

Image Cropping The original size of the images is 1024×1024, which is too large for the neural network to learn features. We cropped the images into small patches (128×128 and 256×256, respectively) and sampled several times from the original images. After the pre-processing procedure, each image could generate 225 patches if they are cropped into 128×128 pixels.

Since some tumors have radius larger than 128 pixels, the tumors could be separated into different patches and would be hard for the neural network to capture the features for tumor edges. Based on the average radius of the cancerous tissue, we cropped the images to 256×256 pixels with tumor center located at the image patch center.

Data Cleansing Most mammograms have black background, which could be misleading for the neural network. Hence, it's crucial to remove the black background. We set the numerical range of pixel and discarded the patches with more than 20

3 Methods

Deep neural network has made a tremendous success in image processing area, including image classification, object detection etc. The performance of neural network surpasses traditional mathematical approaches in image processing. In our study, we applied VGG16 for mammograms classification and SSD for automatic lesion detection.

3.1 Classification

Network Architecture We built a binary classifier with VGG16 and Convolutional Neural Network Improvement for Breast Cancer Classification (CNNI-BCC), respectively.

CNNI-BCC CNNI-BCC is a 30-layer convolutional neural network specialized at learning features in mammograms proposed by F. F. Ting et.al [4]. It is modified based on VGG16. It has 28 convolutional layers, 1 pooling layer and 1 fully connected layer. Instead of using normal 2-dimensional convolutional layer, depthwise separable convolution layer is adopted to reduce the number of parameters needed to train. Average pooling operation is used in the pooling layer and a dense layer is appended at the end.

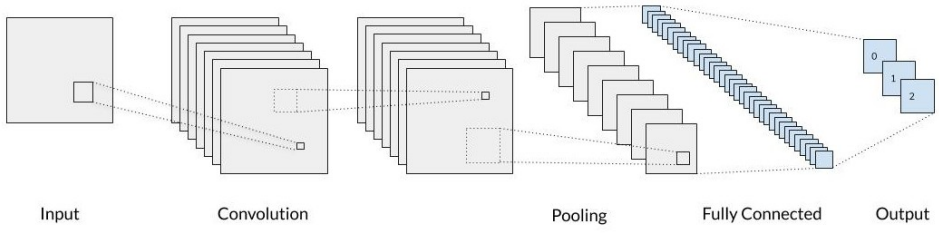


Figure 3: CNNI-BCC Architecture

VGG16 VGG16 is a 16-layer convolutional neural network proposed by K. Simonyan and A. Zisserman which has shown remarkable performance in image classification and been widely used. [9] It contains 5 blocks. The first two blocks have two convolutional layers followed by a max-pooling layer and the last three blocks consist of three convolutional layers followed by a max-pooling layer. After each convolutional layer, a rectified linear unit (ReLU) is appended as activation function. Two fully-connected layer are concatenated with the convolutional blocks for classification.

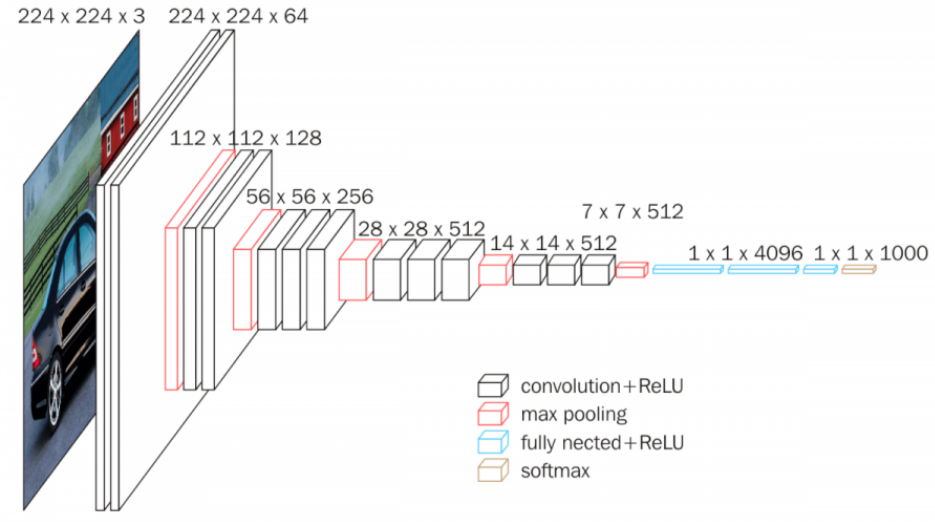


Figure 4: VGG16 Architecture

The original input images for VGG16 have spatial dimension of $224 \times 224 \times 3$ (depth $\times$ width $\times$ channel). In our case, the input image dimension is changed to $128 \times 128 \times 1$ as we have the cropped mammograms in grayscale. Other hyperparameters such as number of filters remain the same as the standard set utilised in the original paper.

3.2 Object Detection

Single Shot Multibox Detector (SSD) SSD is one of the most popular object detection algorithms due to its ease of implementation and good accuracy. We applied SSD on our preprocessed data to see how it works on tumor detection. Below is the general structure of SSD, followed by the table illustrating the modified layers we used in our tests. In the front part of SSD model, several convolutional layers and max pooling layers are used to generate feature maps of different scales to detect objects of various sizes. In the source code, there are SSD models suitable for 300×300 and 512×512 pictures. When we performed the test, a portion of layers were changed to fit our 256×256 images. The latter section of SSD model are layers to compute multiple classes confidences and learn the localization detection, which remain unchanged.

Implementation We implemented our models using Python with Keras based on Tensorflow backend. Our programs were tested on PSC bridges with a 8G memory GPU.

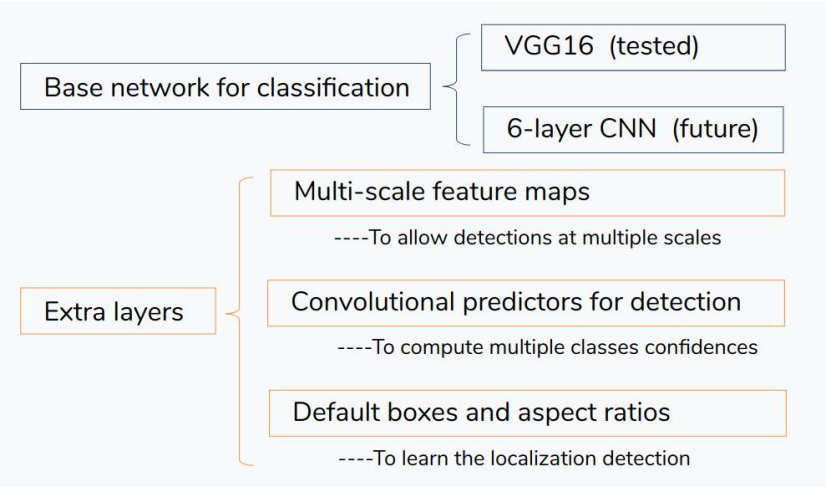


Figure 5: SSD Architecture

4 Experiments

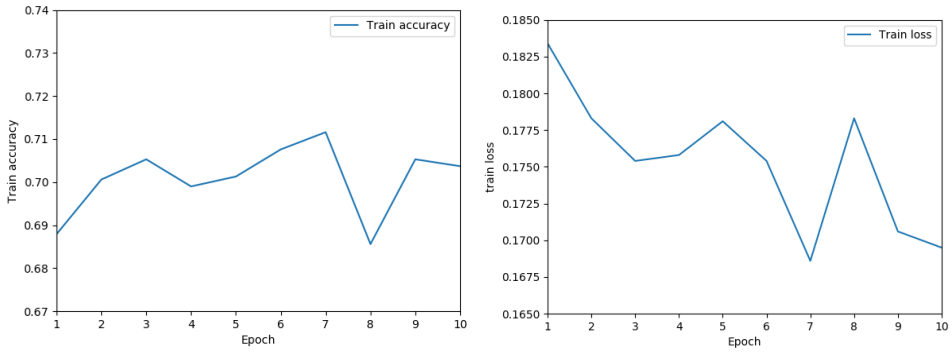
4.1 Classification

We have conducted several experiments for mammogram classification with VGG16 and CNNI-BCC, respectively. In all the experiments, the validation set is split from the last 10% of the training set and the ratio between the training set (including the validation set) and the test set is 9:1.

CNNI-BCC Although CNNI-BCC is a 30-layer convolutional neural network, we only tested part of it due to the restriction of computational resources. Only the first 4 convolutional layers and the fully connected layer at the end were employed. The model was trained with 1269 image patches for 10 epochs with batch size of 20 and tested with 157 image patches. We set a constant learning rate of 0.002 and used mean squared error as loss function and stochastic gradient descent as optimizer.

Layer Type	Filters	Size	Strides	Output
Conv2D	64	(3, 3)	(1, 1)	
Conv2D	64	(3, 3)	(1, 1)	
MaxPooling2D		(2, 2)	(2, 2)	(128, 128)
Conv2D	128	(3, 3)	(1, 1)	
Conv2D	128	(3, 3)	(1, 1)	
MaxPooling2D		(2, 2)	(2, 2)	(64, 64)
Conv2D	256	(3, 3)	(1, 1)	
Conv2D	256	(3, 3)	(1, 1)	
Conv2D	256	(3, 3)	(1, 1)	
MaxPooling2D		(2, 2)	(2, 2)	(32, 32)
Conv2D	512	(3, 3)	(1, 1)	
Conv2D	512	(3, 3)	(1, 1)	
Conv2D	512	(3, 3)	(1, 1)	
MaxPooling2D		(2, 2)	(2, 2)	(16, 16)
Conv2D	512	(3, 3)	(1, 1)	
Conv2D	512	(3, 3)	(1, 1)	
Conv2D	512	(3, 3)	(1, 1)	
MaxPooling2D		(3, 3)	(1, 1)	
Conv2D	1024	(3, 3)	(1, 1)	
Conv2D	1024	(1, 1)	(1, 1)	
Conv2D	256	(1, 1)	(1, 1)	
Conv2D	512	(3, 3)	(2, 2)	(8, 8)
Conv2D	128	(1, 1)	(1, 1)	
Conv2D	256	(3, 3)	(2, 2)	(4, 4)
Conv2D	128	(1, 1)	(1, 1)	
Conv2D	256	(3, 3)	(1, 1)	(2, 2)
Conv2D	128	(1, 1)	(1, 1)	
Conv2D	256	(2, 2)	(1, 1)	(1, 1)

Figure 6: Structure of SSD256 model used (only the part extracting feature maps)



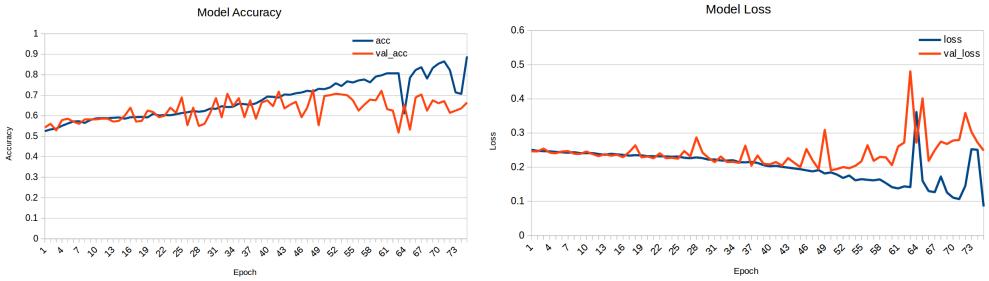
(a) Accuracy (b) Loss

Figure 7: Results of training 6-layer CNNI-BCC with 1269 image patches for 10 epochs

The training loss and accuracy are as above. The loss decreases a little during the training process and the accuracy is around 70% at the end. The accuracy on the test set is 73.24%.

However, as normal cases comprise approximately of the total training set, the model is prone to learn more features of normal cases and thus make more predictions of normal, which inflates the accuracy. Consequently, we balanced the number of images for three classes in later experiments.

VGG16 Though CNNI-BCC has fewer parameters to train due to the replacement of Conv2D layer with depthwise convolutional layer, it sacrificed the accuracy. Hence, we employed a classical CNN (VGG16) to resolve our problem. Instead of categorizing the images into three classes, we tried with two classes (normal and abnormal) first. The image patches are shuffled before being fed into the neural network. We used 2520 image patches to train for 80 epochs.



(a) Training and validation accuracy (b) Training and validation Loss

Figure 8: Results of training VGG16 with 2520 image patches for 80 epochs

As demonstrated above, the training accuracy reaches 89.40% at 76th epoch. However,

the validation accuracy fluctuates at around 65% which indicates that the model is overfitting and has a poor performance of predicting images it has not encountered.

4.2 Automatic Lesion Detection

Metrics mAP (mean Average Precision) is a popular metric in measuring the accuracy of object detectors like Faster R-CNN, SSD, etc. [5] The computation is complicated and I will illustrate it below. But before that, we will do a brief recap on precision, recall, and IoU first.

Precision and Recall Precision measures how accurate the predictions are. Recall measures how good you find all the positives.
Here are their mathematical definitions:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

where TP = True Positive, FP = False Positive, FN = False Negative.

In our case of testing for cancer:

$$\text{Precision} = \frac{\text{TP}}{\text{Total Positive Results Shown by Model}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{Total Cancer Cases in Ground Truth}}$$

IoU (Intersection over Union) IoU measures the overlap between 2 boundaries: our predicted boundary and the ground truth (the real object boundary). In real application, we predefine an IoU threshold (0.5 in our code) in classifying whether the prediction is a true positive or a false positive.

mAP is the average of AP (Average Precision) among all classes. To compute AP, Precision-recall curve needs to be drawn first. We collect all the predictions made for tumors in all the images and rank it in descending order according to the predicted confidence level. Then we determine the correctness of the predictions by IoU. As we go down the prediction ranking, recall values increase. Meanwhile, precision has a zigzag pattern — it goes down with false positives and goes up again with true positives. Then we can plot the precision against the recall value to obtain the Precision-recall curve.

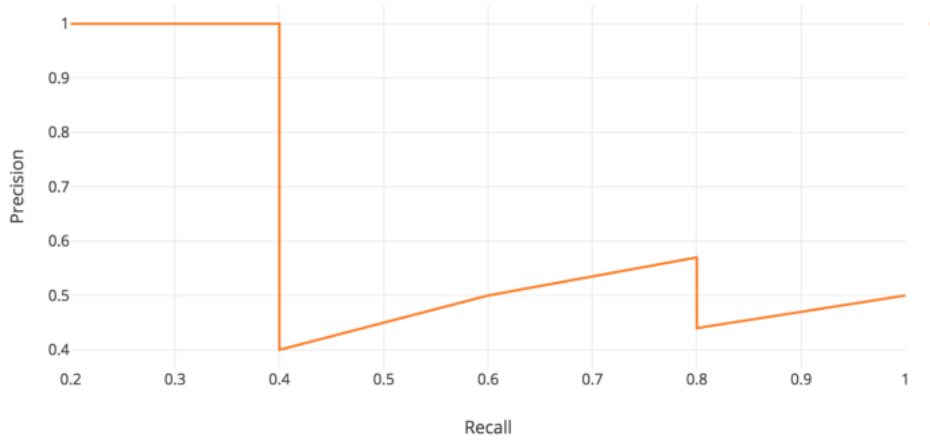


Figure 9: Precision-Recall Curve

The general definition for the Average Precision (AP) is finding the area under the precision-recall curve above.

$$\text{AP} = \int_0^1 p(r)dr$$

But when calculating AP in the objection detection, we often smooth out the zigzag pattern first. Graphically, at each recall level, we replace each precision value with the maximum precision value to the right of that recall level.

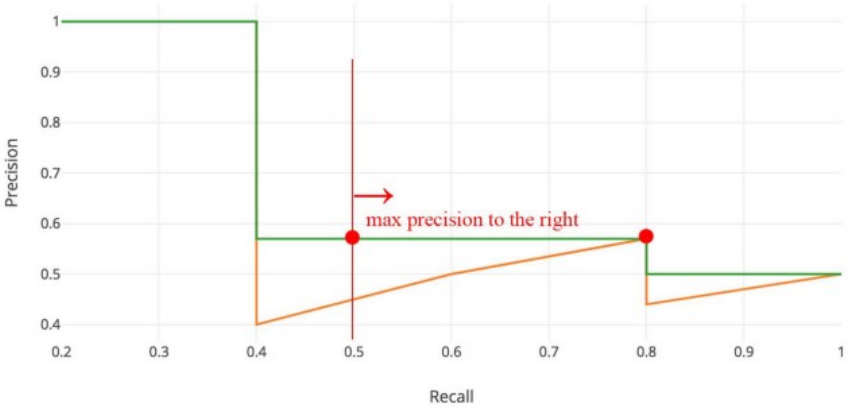


Figure 10: Smoothed Precision-Recall Curve

To compute AP, first, we divide the recall value from 0 to 1.0 uniformly —- into 20 points when we evaluated our model. Next, we compute the average of maximum precision value for these 20 recall values. Here are the more precise mathematical definitions.

$$\text{AP} = \frac{1}{20} \sum_{r \in \{0.0,\ldots,1.0\}} p_{interp}(r)$$

where

$$p_{interp}(r) = \underset{\tilde{r} \geqslant r}{max}\, p(\tilde{r})$$

All in all, mAP is a number between 0 and 1. The closer mAP evaluated from the model is to 1, the better performance the model has.

ExperimentA - Best Results We ran SSD algorithm on the tumorcenter dataset, which contains 600 pictures in training set, 100 in validation set, and 116 in test set. We set the batch size to be 100, steps per epoch to be 60. So in each epoch, all pictures were gone through for 10 times, which can save time compared with running 10 epochs and going through all pictures for once in each epoch, since in that case lot of time will be used to comparing the performance from last epoch and saving model. We ran the training process for 110 epochs and each epoch cost around 760s. Below is the chart illustrating the trend of loss decline. It can be seen that the model converges quit slow, after 110 epochs the validation loss achieves 3.14813, while after 50 epochs the number is 5.06561.

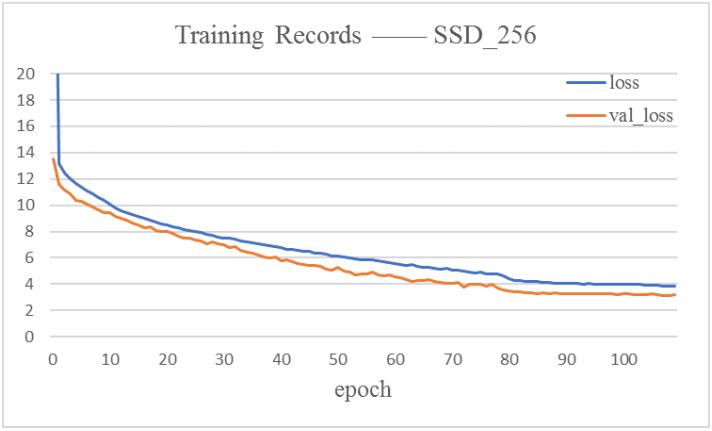


Figure 11: Training Records of SSD-256 Model

The mAP of the model trained for 110 epochs is 0.842 (0.839 AP for benign cases and 0.846 AP for malignant cases), which is similar to the results from the original SSD paper. In the paper, SSD 512×512 and 300×300 models were trained and tested on PASCAL VOC2007 and VOC2012 datasets, which contain around 20 classes of objects to be detected and each class is consist of approximately 1000 images. Therefore, our result shows that SSD algorithm also suits tumor detection. And it is expected that with more mammogram data, better results will be achieved.

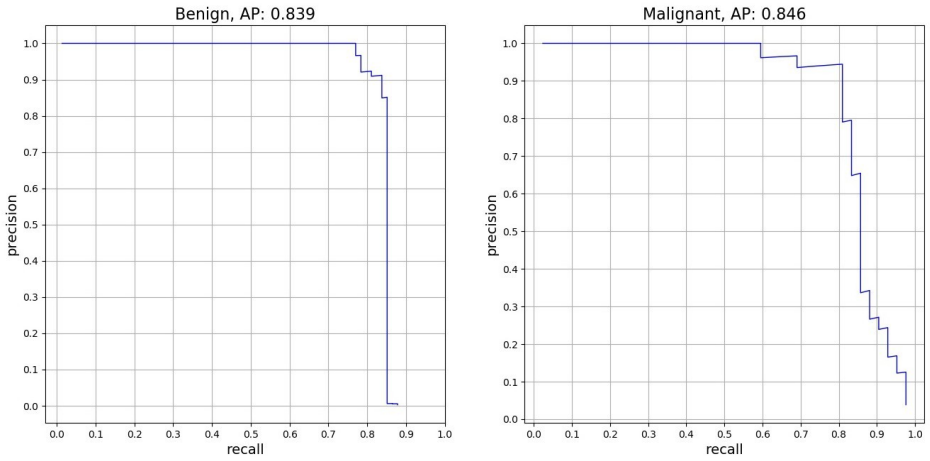
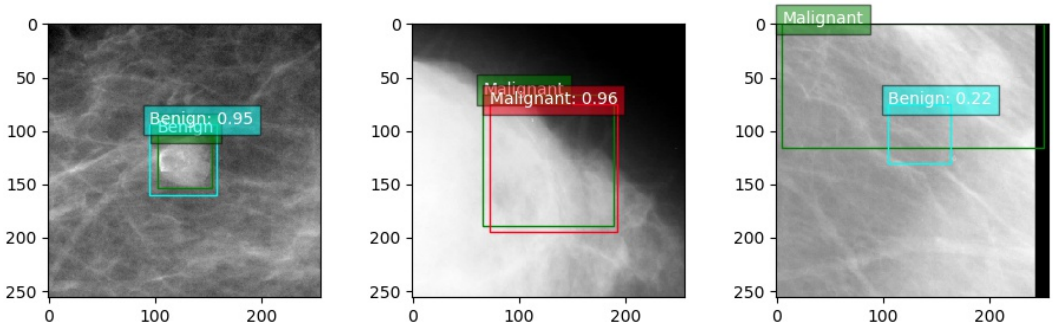


Figure 12: Precision-Recall Curve of SSD-256 Model on Tumor-centered Test Set

Below are some prediction examples of the model. The first two pictures are from the model after 110 epochs of training, while the last picture is from the model after 50 epochs. As is shown in the graphs, the model trained for 110 epochs shows much higher accuracy of both localization and classification than the one trained for 50 epochs, even though the loss just decreases for less than 2. What's more, the model trained for 50 epochs performs so poorly that most tumors in the images cannot be detected, and hence mAP cannot be calculated.



(a) (b) (c)

Figure 13: Prediction Examples of the Model in Experiment A

ExperimentB - Increasing Learning Rate Considering that the SSD algorithm in Experiment A converges quite slow and each epoch takes too much time beyond expectation, we tried to increase learning rate to accelerate the training process. The original learning rate schedule was: 10^{-3} for the first 90 epochs, 10^{-4} for the 91-100 epochs, and 10^{-5} for the last 10 epochs. Note that the learning rate was set to be

stepped down descent in order to improve convergence. We doubled the learning rates setting the schedule to be 2×10^{-3}, 2×10^{-4}, 2×10^{-5} correspondingly.
However, this change led to explosion of loss in several epochs. For example, at the 130th epoch, the training loss rockets from 23 to 1.3×10^{10}, and the validation loss from 20 to 1.15×10^{8}.

ExperimentC - Enlarging Batch Size In general, enlarging batch size will accelerate processing speed with same amount of data and determine the direction of descent more accurately when training. Therefore, we tried training setting with larger batch size.

Here are the training setting and results:

Batch Size	Steps Per Epoch	Epoch	Validation Loss	Speed
300	2	300	9.24585	87s/epoch
200	3	300	8.64310	80s/epoch

Table 1: Comparison between Different Batch Sizes

In each epoch, both of these two processes went through all images in training set once. This is different from the setting in Experiment A, in which all images were undergone for 10 times in each epoch. So after 300 epoch, the performance of these two model should be similar to that of the model in Experiment A after 30 epochs, whose validation loss is 7.024. However, training process with larger batch size not only shows worse performance on validation loss, but also on training speed. This phenomenon is kind of irrational and unexpected. As a result, it seems to be better to make the batch size remain unchanged.

ExperimentD - Using Separable Convolution To accelerate training process, we attempted to use separable convolution layers to take place of convolution layers in our model. The basic idea is that the number of parameters to be trained of separable convolution layers is fewer than that of the normal convolution layers, which is due to the difference of convolution procedure.

For instance, for the first layer in our model shown before, input channel is 3, output channel is 64, kernel size is 3×3. If we use normal convolution layer, the parameter number is $(3 \times 3 \times 3 + 1) \times 64 = 1792$, that is, (input channel $\times$ kernel size + bias) $\times$ output channel. If we use separable convolution layer, the parameter number will be $3 \times 3 \times 3 + (3 \times 1 \times 1 + 1) \times 64 = 283$, that is, (input channel$\times$kernel size)+(input channel$\times$ 1×1 + bias) $\times$ output channel. The concrete procedure of normal convolution is that the input channel data is traversed by 64 different convolution kernels of size 3×3.

And the concrete procedure of separable convolution is kind of complicated: the input channel data is traversed by 3 different kernels of size $3 \times 3 \times 3$ to obtain 3 middle channels, then the middle channels is traversed by 64 different kernels of size $3 \times 1 \times 1$ to combining the information learnt from different input channels.

Here is the parameter calculated by the model.summary() function of keras. The left two columns are for normal convolution, while the right two are for separable convolution. As we can see, the computation above is exactly correct, and the number of total parameters is only 20% of that before replacement.

conv1_1(Conv2D)	1,792	conv1_1(SeparableConv2D)	283
Total Parameters	23,715,730	Total Parameters	4,820,525

Table 2: Comparison between Using Normal Conv2D and SeparableConv2D

We conducted several experiments under different settings on batch size and steps per epoch, traversing all data once in each epoch. It can be observed that SeparableConv2D-model takes more time to train per epoch, and this is a bit irrational since this model has much fewer parameters than Conv2D-model. We think this phenomenon may be because of some differences of underlying logic and backend algorithm between separable convolution and normal convolution. Positive sign is that SeparableConv2D-model converges faster than Conv2D-model in terms of epochs (but slower in terms of real time). So if we can find some way to modify the backend algorithm, our training procedure may speed up.

Batch Size	Steps Per Epoch	Epoch	Validation Loss	Speed
100	6	150	4.36554	245 s/epoch
200	3	150	4.80492	245 s/epoch
300	2	150	5.03458	245 s/epoch

Table 3: Comparison between Using Normal Conv2D and SeparableConv2D

ExperimentE - Testing and Retraining on More General Cases The results in Experiment A seems to be excellent, but as we mentioned, all images were preprocessed to be centered at tumor, which may mislead the training and teach the model wrong character that all tumors are in the center of the picture. This may lead to low accuracy when detecting the cancer in which tumor is near the edge of the mammogram. To test and verify, we fed some uniformly cut pictures into the trained model to see the performance. The mAP is 0.284 (0.31 AP for benign case and 0.257 AP for malignant case), which shows that there are some overfitting problems. But as is shown in prediction examples, when detecting non-center tumor, the model also shows high accuracy on both localization and classification. The problem is that the confidence

value is much lower than that when detecting center tumor, which leads to the drop of mAP.

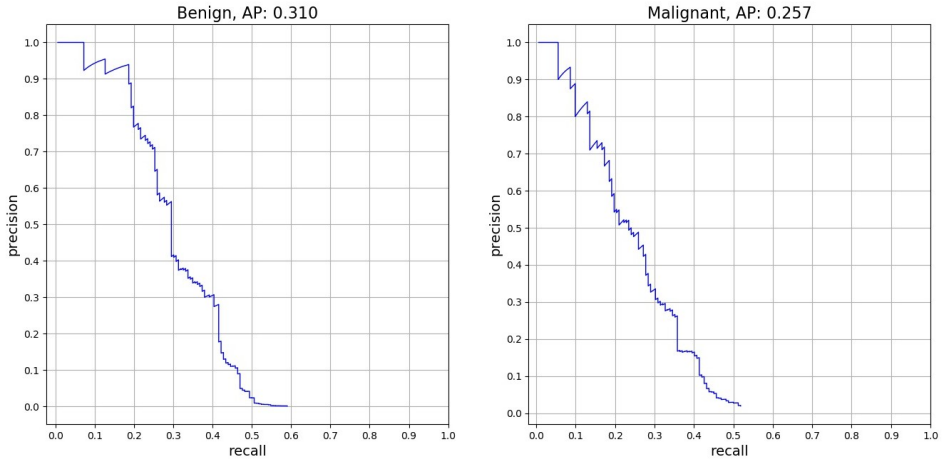


Figure 14: Precision-Recall Curve of SSD-256 Model in Experiment A

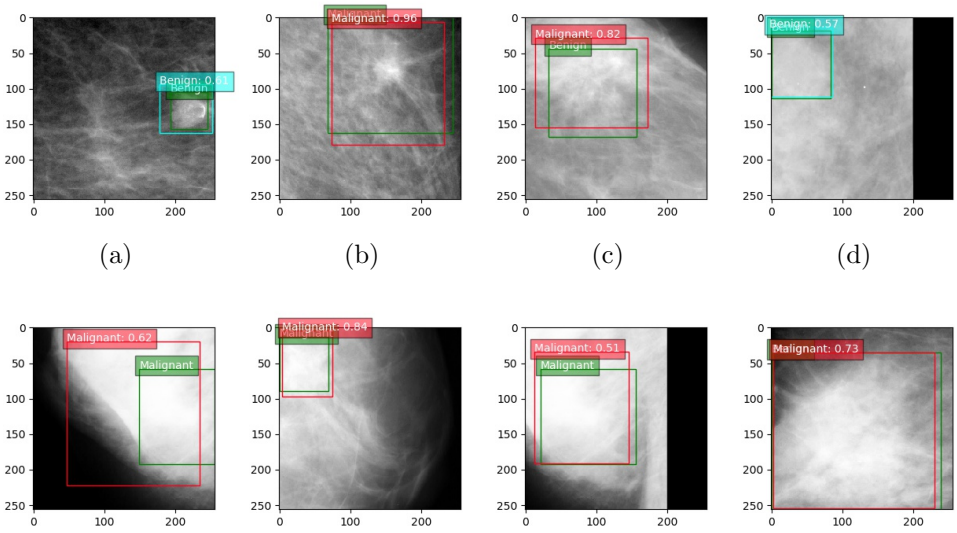


Figure 15: Prediction examples of the model in Experiment A on uniformly cut picture dataset

We trained the model on uniformly cut data set (consist of 3500 images for training set again, 500 for validation set, and 329 for test set) for 20 epochs with setting 100 for batch size and 35 for steps per epoch. Below is the evaluation result and training

records of whole 130 epochs (110 epochs in Experiment A, 20 epochs in Experiment E). The loss is steadily declining and mAP is steadily rising, showing the model is able to abstract more features once being fed corresponding data. However, the mAP tested on tumor-centered test set falls, encouraging that in futural training, the training set should involve tumor in different location.

Tumor-centered Test Set		Uniformly Cut Test set	
Benign AP	0.742	Benign AP	0.394
Malignant AP	0.54	Malignant AP	0.332
mAP	0.641	mAP	0.363

Table 4: Comparison between Using Normal Conv2D and SeparableConv2D

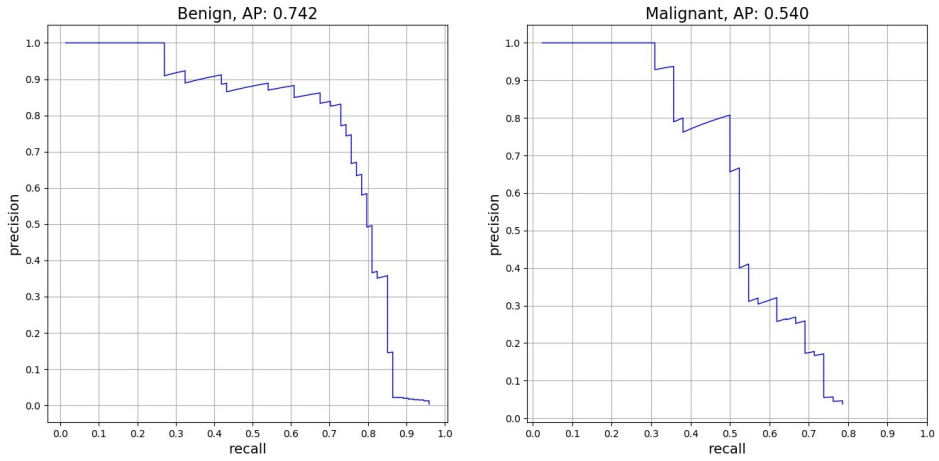


Figure 16: Precision-Recall Curve of Retrained SSD-256 Model on Tumor-centered Test Set

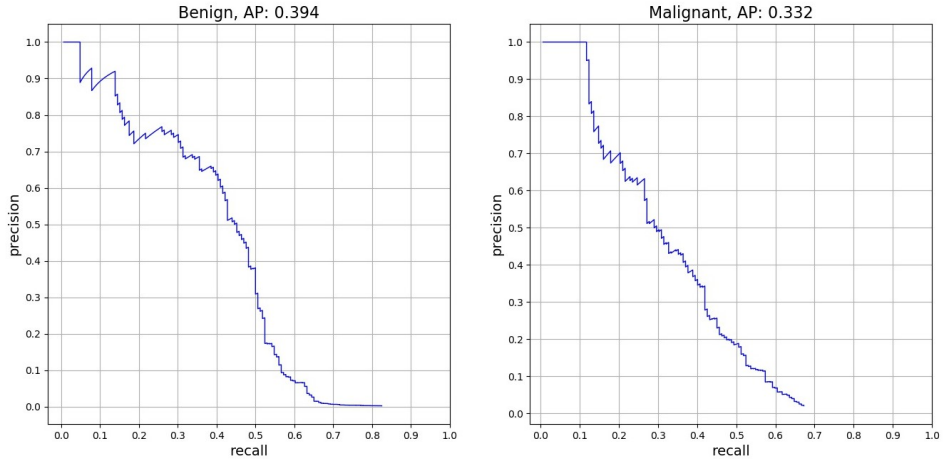


Figure 17: Precision-Recall Curve of Retrained SSD-256 Model on Uniformly Cut Test Set

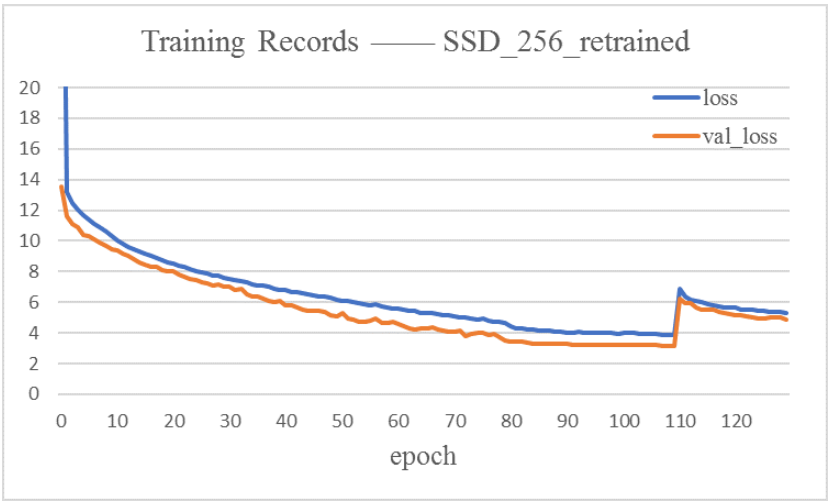


Figure 18: Training Records for 110 epochs in Experiment A Plus 20 Retraining Epochs

5 Future Works

Unlike some typical classification problem, such as training the neural network to distinguish whether the image contains a dog or cat, classifying mammograms is complicated in its nature as the tumor is usually located in a small region of the whole image. Hence, the ternary label of mammograms may not be an obvious feature for the neural network to learn.

Furthermore, the mammograms usually have low contrast and sometimes it's even hard for radiologists to interpret. Applying appropriate techniques for enhancing the image contrast before training could be a plausible approach to improve the performance of neural network.

In addition, VGG16 in its nature is designed for classifying images into 1000 categories, which leads to the great number of parameters in the last two dense layers. However, in our study, the mammograms only need to be classify into 3 categories, which implies a possibility of applying some hyperparameter tuning algorithm to find a better hypermarater set that could reduce the computing time without sacrificing the performance. Besides, it is also possible to employ pre-trained neural network with large dataset such as ImageNet to extract rudimentary features of the mammograms.

In the evaluation of mammogram classification, we only considered the accuracy but not the sensitivity and specificity. It is possible to take more metrics into consideration. We implemented 10-fold cross validation method and could test it in the future. Besides, we found a new database which we could use to increase the training set.

6 Acknowledgements

This work was an internship project at The University of Tennessee, sponsored by the National Science Foundation through Research Experience for Undergraduates (REU) award, with additional support from the Joint Institute of Computational Sciences. We would like to express our gratitude to Dr. Raymond Chan and Dr. Kwai Wong for mentoring the project and for helpful discussions and comments. We also thank the Extreme Science and Engineering Discovery Environment (XSEDE) for providing allocations.

References

[1] Ferlay, J., Héry, C., Autier, P. Sankaranarayanan, R. (2010). Global burden of breast cancer. *Breast cancer epidemiology*, 1–19, Springer.

[2] Xi, P. et.al. (2018). Abnormality Detection in Mammography using Deep Convolutional Neural Networks. *IEEE International Symposium on Medical Measurements and Applications (MeMeA).* doi:10.1109/MeMeA.2018.8438639

[3] Wu, N. et.al. (2019). Deep Neural Networks Improve Radiologists' Performance in Breast Cancer Screening. arXiv:1903.08297.

[4] Ting, F. F., Tan, Y. J., Sim, K. S. (2019). Convolutional neural network improvement for breast cancer classification. *Expert Systems with Applications*, 120, 103-115.

[5] Hui, J. (2018). mAP (mean Average Precision) for Object Detection. Retrieved from: A Medium Corporation(US): https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

[6] Simonyan,K., Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition . arXiv:1409.1556

[7] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., Berg, A. C. (2016, October). SSD: Single shot multibox detector. *European conference on computer vision (pp. 21-37).* Springer, Cham.

[8] Ferrari, P. (2018). SSD: Single-Shot MultiBox Detector implementation in Keras. *GitHub repository.* https://github.com/pierluigiferrari/ssd_keras

[9] Hassan, M. (2018). VGG16 – Convolutional Network for Classification and Detection. Retrieved from: https://neurohive.io/en/popular-networks/vgg16/

[10] Chollet, F. (2015). Keras: Deep learning library for theano and tensorflow. Retrieved from: https://keras.io/k,7(8).