

Santa Clara University

**Scholar Commons**

---

Bioengineering Senior Theses

Engineering Senior Theses

---

Spring 2021

## **Classification of Breast Cancer Using Deep Learning and Mammogram Images**

Travis Kay

Derrick Dang Nguyen

Lashan Wijayawickrama

Follow this and additional works at: [https://scholarcommons.scu.edu/bioe\\_senior](https://scholarcommons.scu.edu/bioe_senior)



Part of the [Biomedical Engineering and Bioengineering Commons](#)

---

**SANTA CLARA UNIVERSITY**

Department of Bioengineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED  
UNDER MY SUPERVISION BY

Travis Kay, Derrick Dang Nguyen, Lashan Wijayawickrama

ENTITLED

**Classification of Breast Cancer Using Deep Learning and  
Mammogram Images**

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF

**BACHELOR OF SCIENCE**  
IN  
**BIOENGINEERING**

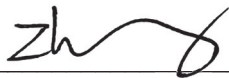


June 9, 2021

---

Thesis Advisor(s) (use separate line for each advisor)

date



June 9, 2021

---

Department Chair(s) (use separate line for each chair)

date

# Classification of Breast Cancer Using Deep Learning and Mammogram Images

By

Travis Kay, Derrick Dang Nguyen, Lashan Wijayawickrama

## **SENIOR DESIGN PROJECT REPORT**

Submitted to  
the Department of Bioengineering

of

**SANTA CLARA UNIVERSITY**

in Partial Fulfillment of the Requirements  
for the degree of  
Bachelor of Science in Bioengineering

Santa Clara, California

Spring 2021

# Abstract

Breast cancer is the second leading cause of cancer deaths among US women. Thus, it is important for doctors to detect and diagnose breast cancer as early as possible. Mammography has been used for about 30 years, but there have been rapid developments using digital mammography technology and computer aided systems to help improve breast imaging. Deep learning techniques are being developed to provide a more effective tool for the classification of breast cancer. We adopt a transfer learning approach and fine-tune a pre-trained convolutional neural network model for accurate classification of breast masses based on screening mammograms. The model is retrained and tested using the CBIS-DDSM (Curated Breast Imaging Subset of Digital Database for Screening Mammography) dataset. We are able to achieve a training accuracy of 71.1% and a test accuracy of 68.7%.

# Acknowledgments

We would like to thank our project advisor, Dr. Yan for guiding and supporting us throughout this entire process. Her knowledge and experience was invaluable and helped make our project's direction more clear.

We would also like to thank graduate students, Joshua Vincent and Pradnya Patel, for sharing their experiences with machine learning with us and also providing useful tips on how to optimize our model.

Lastly, we would like to thank Santa Clara University School of Engineering for providing us with this opportunity to learn and grow as engineers.

# Table of Contents

<b>Abstract</b>	<b>iii</b>
<b>Acknowledgments</b>	<b>iv</b>
<b>Table of Contents</b>	<b>v</b>
<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>viii</b>
<b>Chapter 1: Introduction</b>	<b>1</b>
1.1: Overview	1
1.2: Breast Cancer	1
1.3: CNN and Machine Learning for Medical Imaging	2
1.4: Problem Statement	2
1.5: Project Goal	2
<b>Chapter 2: Background and Significance</b>	<b>3</b>
2.1: Overview	3
2.2: ResNet CNN Models	4
2.3: Pretrained ResNet Architecture	5
2.4: Advancements in Current Field	5
<b>Chapter 3: Materials and Methods</b>	<b>7</b>
3.1: Overview	7
3.2: Materials	8
3.3: Preprocessing	9
3.3.1: CBIS-DDSM Dataset	9
3.3.2: Various methods for preprocessing	9
3.3.3: Our method for preprocessing	10
3.4: Training our Model	11
3.4.1: Training Parameters	11
3.4.2: Experimental Training Process	12
<b>Chapter 4: Results</b>	<b>15</b>
4.1 Introduction	15
4.2 Training Model with Cropped Images of 250x250	15
4.2.1 Results of Using Cropped Images of 250x250	16
4.2.2 Data interpretation of Using Cropped Images 250x250	17
4.3 Training Model with Cropped Images of 500x500	18

4.3.1 Results of Using Cropped Images 500x500	18
4.3.2 Data Interpretation of Using Cropped Images 500x500	19
4.4 Training Model with Resized Images of 250 x 250	19
4.4.1 Results of Resizing Images to 250x250	19
4.4.2 Data Interpretation of Resizing Images	20
4.5 Training a Model with Images less than 620x620	21
4.5.1 Results of Manually picking out Images larger than 620x620	21
4.5.2 Data Interpretation of Manually picking out images	22
4.6 Overall Results	22
4.6.1 Testing Results Using an Ants and Bees Dataset	22
4.6.2 Overall Analysis and Interpretation	23
4.7 Future Steps	24
<b>Chapter 5: Engineering Ethics with Breast Cancer Classification</b>	<b>25</b>
5.1: Ethical Justifications of the Project	25
5.2: Our Project's Engineering Standards	25
5.3: Breast Cancer Classification Models, Safety, Risk and Informed Consent	26
<b>Chapter 6: Conclusion</b>	<b>28</b>
<b>References</b>	<b>29</b>
<b>Appendix</b>	<b>A-1</b>
Appendix A: Example of Our Model Code	A-1
Appendix B: Bash Shell Script Code to Preprocess Images	B-1
Appendix C: Supplementary Images and Tables	C-1
Appendix D: Specific Ethical Concerns of Deep Learning Classification	D-1

# List of Figures

<b>Figure 2.1:</b> Diagram of a deep neural network	3
<b>Figure 2.2:</b> Representation of a CNN model	4
<b>Figure 3.1:</b> Example of cropping and zero-padding images	10
<b>Figure 3.2:</b> Comparison of cropped images	13
<b>Figure 3.3:</b> Comparison of resized images	13
<b>Figure 4.1:</b> Accuracy of using ResNet-18 with cropped 250x250 images	16
<b>Figure 4.2:</b> Accuracy of using ResNet-34 with cropped 250x250 images	16
<b>Figure 4.3:</b> Accuracy of using ResNet-50 with cropped 250x250 images	17
<b>Figure 4.4:</b> Accuracy of using ResNet-18 with cropped 500x500 images	18
<b>Figure 4.5:</b> Accuracy of using ResNet-34 with cropped 500x500 images	18
<b>Figure 4.6:</b> Accuracy of using ResNet-18 with resized 250x250 images	19
<b>Figure 4.7:</b> Accuracy of using ResNet-34 with resized 250x250 images	20
<b>Figure 4.8:</b> Accuracy of using ResNet-18 with images less than 620x620	21
<b>Figure 4.9:</b> Accuracy of using ResNet-34 with images less than 620x620	21
<b>Figure 4.10:</b> Comparing mammogram images with nonmedical images	23
<b>Figure C.1:</b> Accuracy of using ResNet-18 for classification between bees and ants	C-1
<b>Figure C.2:</b> Differences in learning rates with respect to its loss	C-2



# List of Tables

**Table C.1:** Results of Training ResNet-34 Models

C-1

# Chapter 1: Introduction

## 1.1: Overview

Breast cancer is the most common cancer globally; it accounts for 13% of annual new cases worldwide [1]. This widespread diagnosis raises concerns on whether mammogram images can be properly diagnosed. With the advancements in machine learning, specifically deep learning, deep learning could be used as a tool for radiologists to be able to more accurately diagnose breast cancer in mammogram images. This thesis aims to help tackle this problem through the use of deep learning techniques.

## 1.2: Breast Cancer

Breast cancer is the uncontrolled growth of breast cells. The most common kinds of breast cancer are invasive ductal carcinoma and invasive lobular carcinoma. In invasive ductal carcinoma, the cancer cells will grow outside of the ducts into other parts of breast tissue. In invasive lobular carcinoma, cancer cells will grow from the lobules and into other parts of the breast tissue. Common symptoms of breast cancer may include changes to the breast size or shape, changes to the skin such as redness or scaling, and a newly inverted nipple [2].

Screening mammography is a type of breast imaging that uses low doses of x-ray radiation in order to detect breast cancer early before the woman starts experiencing symptoms. Conventional mammograms are captured and read on photographic films. However, digital mammograms also use the same x-ray system but instead of using films to store the images, solid state detectors are used to convert the x-rays that are passed into electrical signals. These signals are then converted into images using a computer. Computer aided detection (CAD) systems, which primarily are made using machine learning techniques, will search the mammogram images for abnormalities of density, mass, or calcifications. The CAD systems will highlight these abnormalities so that the radiologist can assess these areas of possible breast cancer.

When a radiologist is looking at the mammograms, he or she will be looking for any breast changes such as the growth of calcifications or masses. Calcifications are tiny calcium deposits within the breast tissue and appear as small white spots on the mammogram. In most cases, these calcifications are benign. However, if the calcifications are irregular in shape or size, a biopsy will be needed to determine if it is benign or malignant. Masses are areas of dense breast tissue with distinct shapes and edges. Cysts are one type of mass that are fluid filled sacs and are non cancerous. However, solid masses are a concern and can be cancerous. An ultrasound is usually performed in order to determine if the mass is a cyst or a solid mass.

### 1.3: CNN and Machine Learning for Medical Imaging

Currently, radiologists use CAD systems as a tool in order to minimize inaccurate diagnoses. These CAD systems use machine learning to create models that classify breast cancer. A subcategory of machine learning and AI is deep learning. Deep learning algorithms have the ability to learn on their own by using computer models to extract information from images. Thus, deep learning is widely used for recognition and segmentation of objects, disease classification, and speech recognition [10-12]. One technique for using deep learning is to use convolutional neural networks (CNNs). CNNs are the most widely used deep neural networks to classify images. These networks are created using a hierarchical model in which each neuron in one layer is connected to all the neurons in the next layer.

### 1.4: Problem Statement

Mammograms can be difficult to interpret which can lead to inaccurate results. There is a variety of breast tissue density among women. Dense breasts can be harder to visualize in an image which hinders the mammogram's ability to accurately diagnose cancer. Screening mammograms do not find roughly 1 in 5 breast cancers [3].

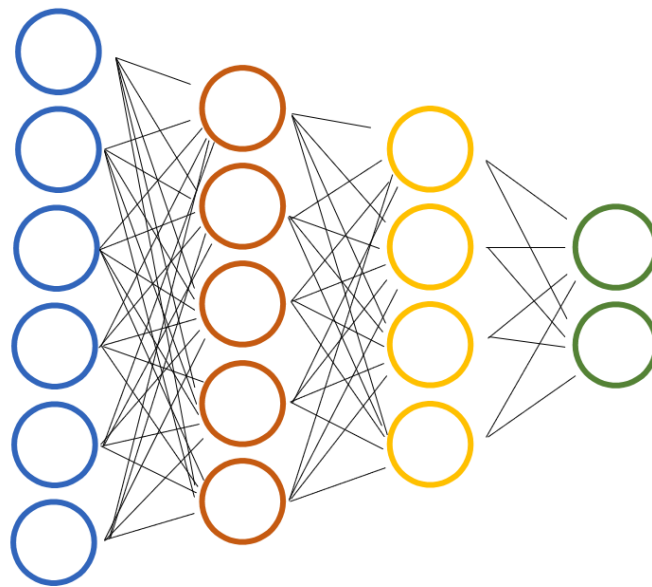
### 1.5: Project Goal

The aim of this project is to train convolutional neural network models to find the best model for classifying masses and cysts for potential breast cancer in screening mammogram images.

# Chapter 2: Background and Significance

## 2.1: Overview

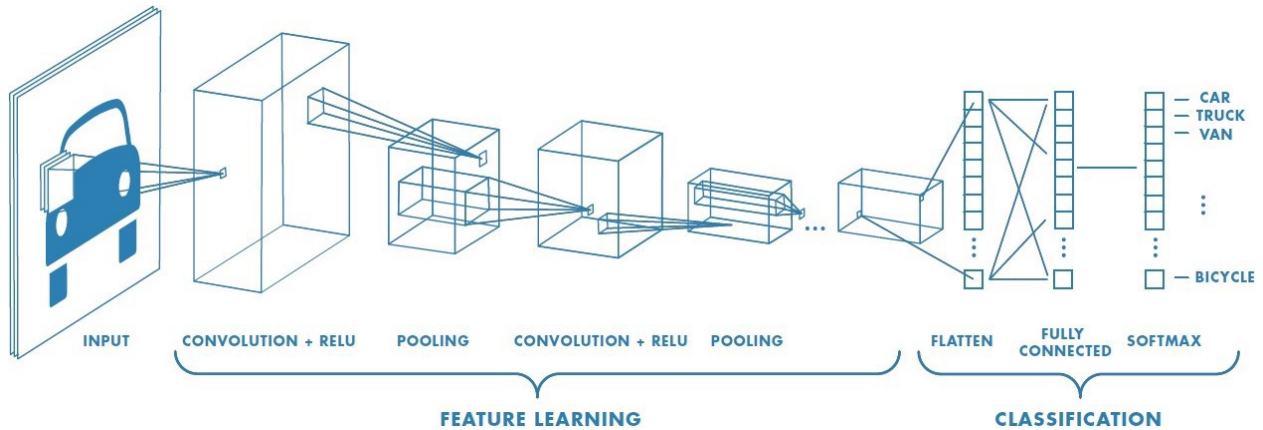
Our model will be using the CNN architecture as it's the foundation for deep learning. There are two major components for CNNs but for deep learning in general, there is a priority on neural networks where the neurons connect with each other to process information. We have the input layer, where the data, such as a mammogram image, is converted into a tensor and prepared to be processed by the hidden layers. The neurons each have their own weights that activate if a specific feature is found in the image tensor. It goes through all these activations as the image goes through each layer until it gets to the end where there is the output classification. Neural networks are trained by passing data through. The training data adjusts the weights when the image passes through the model. It then checks if the ending classification is correct and back propagates to modify the weights with the correct answer.



**Figure 2.1: Diagram of a deep neural network.** Each circle represents a neuron, which are each connected to its previous and next layers.

While the previous paragraph explains deep learning on a high level, CNNs have two parts with the second part being these fully connected neural network layers. Before the creation of CNNs,

if a person wanted to use a neural network, they would have to manually do the feature extractions to process an image before feeding it into the model. Each image needs to be manually modified into a tensor. A CNN differs in that the model does the feature extractions by itself through its convolutional layers. These layers process the features such as rounded edges in an image and directly input these features into a neural network [13].



**Figure 2.2: Representation of a CNN model [4].**

## 2.2: ResNet CNN Models

For this project, we focused specifically on the residual networks (Resnet) architecture for our modeling. ResNet is a complicated model that has skip connections in the convolutional layers where the model skips sections in itself. These skip connections typically decrease loss and errors as the model continues to train because the skip connections bypass over compounding biases that occur with each round of training [14]. Once the convolutional layers are set, the data goes through a neural network. Our project uses Resnet-18 and Resnet-34. The different numbers indicate the number of layers in the ResNet model with Resnet-18 having 18 layers and Resnet-34 having 34 layers.

## 2.3: Pretrained ResNet Architecture

The ResNet models would be pretrained and transferred to our mammogram model. The pretrained model we are using will be pretrained on the imagenet database which is a database of hundreds of thousands of images. These images are in many categories from planes to acorns and

balls to volcanoes. The idea of transfer learning is to use the initial features from a large dataset as a baseline for the model. We will attempt to transfer those feature identification layers to mammogram images in order to extract features for our model's neural network layers to process. We will then train our model's neural network layers to improve classifications.

The initial recognition layers from training on Imagenet are the feature extractor layers. From that, we transfer those layers to our model and focus on fine tuning the last three layers of the fully connected neural network layers to be more specialized for mammogram images for a final classification through passing mammogram images into our ResNet model. By fine tuning only up to the last three layers, we tried to see if transfer learning of the imagenet feature extraction layers is effective with mammogram images when only the neural network layers are fine tuned. From that, we finalize the weights when satisfied with the results and therefore have a finalized classification model.

## 2.4: Advancements in Current Field

There has been significant progress in using ResNet and transfer learning for breast cancer classification in recent years. In particular, Angeles et al. created a CNN model called Mammo [5] that used a pretrained ResNet-50. The model was trained twice using the CBIS-DDSM dataset which is what our model uses. After the second time, Mammo was able to increase the accuracy from .33 to .43 by just balancing the dataset more. One cause for such low accuracy was that no data augmentation was performed that allows for fewer instances of each case to be used for the training of the model. In addition, the model was not a binary classification model, but rather it classified the mammogram into one of four categories: benign calcification, malignant calcification, benign mass, and malignant mass. Since this model was a multiclass model, it requires the model to be able to differentiate more specific features of each classification which allows for more room for error.

Another group by Tsochatzidis et al. tested the accuracies of multiple CNN models using both a finetuning method and also by training a model from scratch [6]. For both approaches, the CBIS-DDSM dataset was used to train and validate the models. Using ResNet-50 and by training the weights from scratch, it achieved its highest accuracy of .627. By using a pretrained

ResNet-50 and finetuning some of its features, the model was able to achieve its highest accuracy of 0.749. This model used pretrained weights based on the ImageNet database. In addition, the fully connected layers of the pretrained model were replaced by randomly initialized layers. The fully connected layer was the last layer of the CNN. Both approaches also used data augmentation techniques of random rotation and flipping of images. The finetuning method's higher accuracy compared to training the model from scratch is largely a result of a pretrained model being optimized on larger datasets so it is more familiar with what features to look for in image classification.

Another example of using transfer learning for breast cancer classification is illustrated by the study done by Agarwal et al. [7]. This group also incorporated a pretrained ResNet-50 model that was initially trained on the ImageNet database. Data augmentation of horizontal flipping, rescaling, and rotating up to 30 degrees was conducted on the dataset. The model was able to achieve an accuracy of 0.84. A model that used randomly initialized weights rather than a pretrained model was tested and got an accuracy of 0.82. This study demonstrates how using transfer learning for the classification of mammograms is beneficial.

Lastly, Falconi et al. were able to achieve an accuracy of 0.86 using the transfer learning approach [8]. This was done by using a pretrained ResNet-50 model that was initially trained on ImageNet. In this approach, the last few layers were replaced with the original model that was trained on the ImageNet dataset. The layers added were a global average pooling 2d layer, a full connecting layer, dropout layer, and a classification layer. Like the previous studies, the model was trained on the CBIS-DDSM dataset. One important attribute to the high accuracy was the data augmentation done as the dataset was increased from 3464 images to a total of 60,000 images. The Augmentor library in Python was used and some operations that were performed were rotations, brightness changes, histogram equalization, and zooming.

# Chapter 3: Materials and Methods

## 3.1: Overview

For our deep learning model, there are no physical materials required. Instead, we used Python as the programming language and supporting libraries as mentioned in the materials list below. Moreover, we emphasized using multiple methods and programs to achieve our goals for preprocessing the CBIS-DDSM Region of Interest (ROI) dataset for training and testing our model.

Beginning the overview for creating the model, we focused on the methods for preprocessing the CBIS-DDSM database. We attempted multiple ways of processing the input size of the images from cropping the initial images, manually inspecting the images, and resizing the images. Once we prepared our input dataset, we experimented with our model hyperparameters such as learning rate and momentum to set the baseline parameters for our project. Then, we could focus on our project's main goal of optimizing and fine tuning the CNN's neural network layers. However, once the hyperparameters were set, we focused on Resnet18 with the layers frozen and layers trained for our models. We measured for accuracy and loss for every epoch trained and used the best results for each model. Once we went through the analysis and the overall process for one ResNet architecture, we repeated the process for other higher level ResNet models.



## 3.2: Materials

CBIS-DDSM database

- ROI images of mammograms

Lenovo Legion Y520

- CPU: Intel(R) Core(™) i7-7700HQ CPU@ 2.80GHz
- GPU: NVIDIA GeForce GTX 1060
- Secondary GPU: Intel(R) HD Graphics 630

Python 3.9

- Anaconda3 (2021.x)
  - Jupyter Notebook
- Imported Libraries:
  - Copy
  - Matplotlib
  - Numpy
  - Pytorch
    - Built (Stable 1.81)
    - Conda Package
    - Computer platform: CUDA 10.2
  - Time

Notepad++

Ubuntu on Windows 10

- DCMTK package

### 3.3: Preprocessing

In order to use the CBIS-DDSM dataset, the images must be preprocessed. Preprocessing is the process in which images are modified in such a way that allows it to be inputted into the model. Since the model does not accept images with varying sizes, preprocessing must be done to standardize all the images. The images must have the same size. With this, there are several ways to manipulate the images in such a way that allows the model to easily read the images to have better results.

#### 3.3.1: CBIS-DDSM Dataset

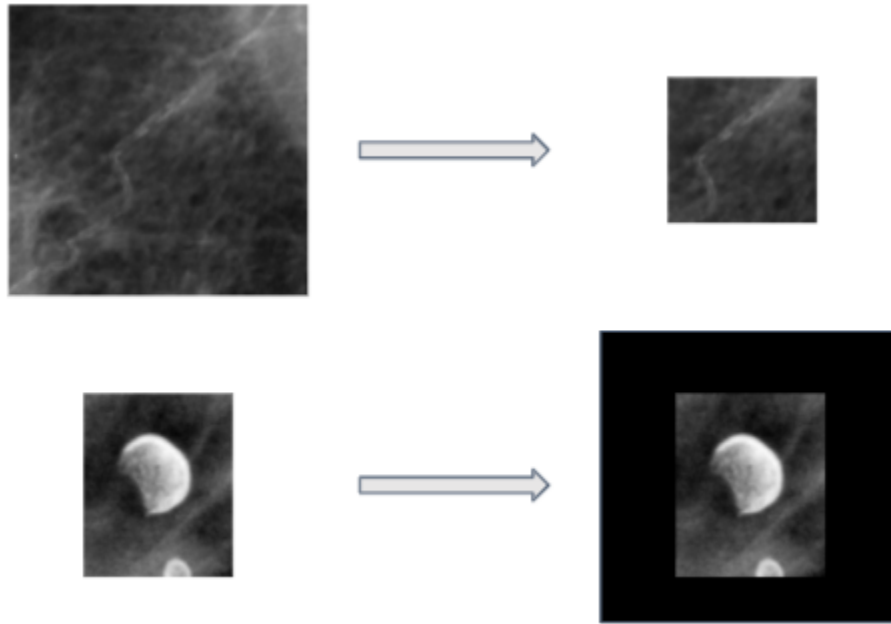
The CBIS-DDSM dataset is an open source dataset that consists of mammogram images. The dataset is initially split into a Training and Testing folder. These images are classified inside each folder into four different categories: calcification benign, calcification malignant, mass benign, and mass malignant. Each image varies in resolution, size, aspect ratios, and how each image was centered on. These images are stored as a dicom file format. Included in the dataset are each image's mask, which allows the model to only focus within the ROI where the breast cancer may appear. Also included in the dataset are csv files that define each image to its ID, its classification, and how each image was sorted into training and testing groups.

#### 3.3.2: Various methods for preprocessing

The most important step in preprocessing is manipulating the dataset in such a way that allows the entire dataset to be consistent in terms of image size, while keeping as much of the original content of each image as possible. This may include keeping the original aspect ratio through resizing, focusing on important parts of the image through cropping, or changing the color dimensions of each datapoint such as brightness, contrast, and more.

Cropping is the process of cutting an image down to a smaller dimension. This usually involves trimming away a part of a border in order to obtain a smaller size. The reverse of cropping is called zero-padding, where a border, usually black, is added to make the image have a bigger size. Both of these processes are the easiest way to make all images have the same size between each other. However, cropping may cause important details of the image to be deleted, which

may cause the model to provide an inaccurate answer, while zero-padding may cause the model to be thrown off course due to the unexpected black border around images.



**Figure 3.1: Example of cropping and zero-padding images.** The first image is about 800x800 pixels and was center-cropped to 250x250. The second image is about 100x150 and was zero-padded to 250x250. Note that these images shown are not to scale.

Resizing is another process that combats the issues raised by cropping and zero-padding. By shrinking or expanding an image to a particular size, all the details within the image are saved. However, most images have different aspect ratios between each other. Resizing them all into one specific aspect ratio may cause distortions for each image. In addition, pixelation may occur when resizing small images into bigger sizes, which may cause the model to be thrown off course.

### 3.3.3: Our method for preprocessing

Since the CBIS-DDSM is inconsistent across all images, preprocessing must be done to get all images into a standard size. Since we want our model to train as a binary classification model, we first need to rework the given csv files to a binary classification: benign and malignant. Changing the classification involves using a simple search and replace finder under Excel.

Once that was done, we wrote a bash shell script to convert each image from a dicom file format into a png file format using the DCMTK package (see Appendix B). These images are then renamed and sorted into their respective binary classification folders based on the reworked csv files. The png file format allows us to more easily insert into the model. Since there were thousands of images, some errors were bound to happen due to outliers. As a result, some of the images were swapped with their masks and placed in the wrong folders, so images had to be manually sorted into their respective folders and then renamed into the correct naming format through a simple script (see Appendix B). 289 images from our benign training folder, 297 images from our malignant training folder, and 2 images from the malignant test folder were swapped and replaced with the proper images.

Upon sorting the images into their respective folders, we then wrote a Python script to center crop and zero-pad the images such that each image has a size of 250x250. This script is included in the model's code (see Appendix A). In some of our models, we changed the size of the images to 500x500 instead.

In the models to which images were resized instead of cropped or zero-padded, preprocessing these images follow a similar procedure. Included within the DCMTK package was a resizing option upon converting the images from a dicom to a png file format. This resizing option did not allow changing each image's aspect ratio, so each image was resized to 250xN, where N corresponded to each image's aspect ratio. Then the same procedures were applied to change each image to 250x250.

## 3.4: Training our Model

### 3.4.1: Training Parameters

For deep learning, there must be training parameters set for consistent training. In experimenting, we set specific numbers for each finetuning parameter and focused on the dataset input size into the convolutional network layers and on varying the amount of neural network layers modified in the training process.

While we did experiment with some parameters, throughout our research we settled on the following parameters. Our CNN model used Stochastic Gradient Descent (SGD) as its

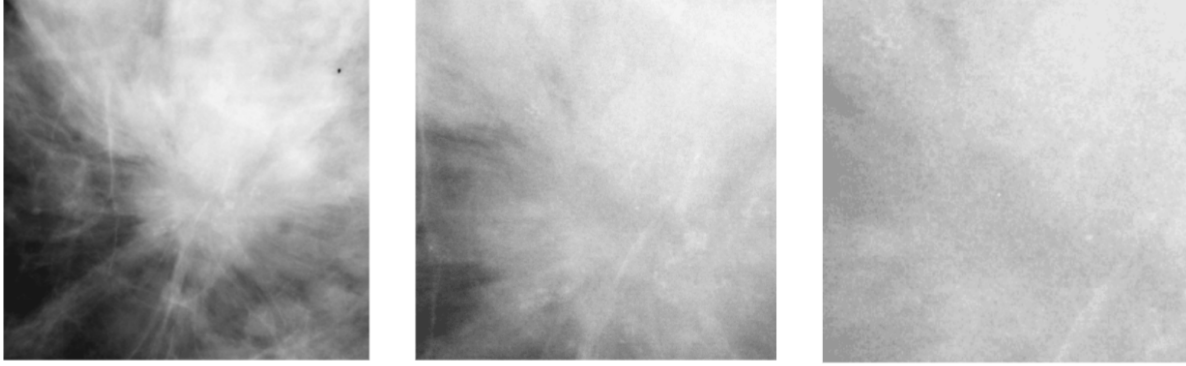
optimization function with an initial learning rate of 0.001. The learning rate has a step factor of 0.1 every 7 epochs to lessen the rate over time. In addition, the optimization function also had a momentum of 0.9. For each experiment, we had a batch number of 16 while training each model 30 to 40 epoch.

### 3.4.2: Experimental Training Process

Multiple iterations of finetuning and training our model occurred. These iterations varied primarily based on which preprocessing technique was used on the CBIS-DDSM image dataset. We had a base CBIS-DDSM image dataset grouped into a “Train” folder and a “Test” folder. Each folder has subcategories for the benign and malignant images. We used this base format and preprocessed the images specifically for the different iterations of training. We categorized our iterations into four different attempts: Cropping images to 250x250, cropping images to 500x500, resizing images to 250x250, and manually removing images larger than 620x620.

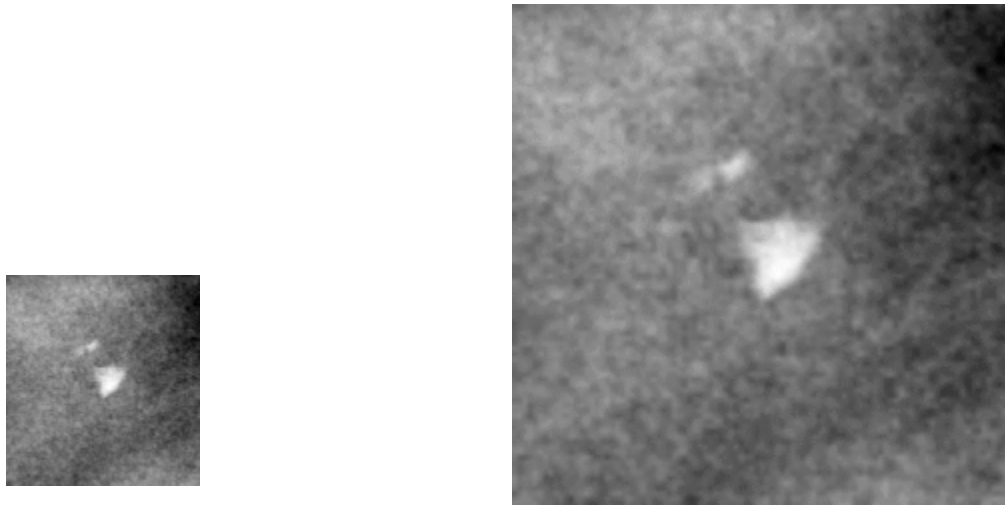
For our first attempt, we cropped the CBIS-DDSM images to 250x250. These dimensions were picked as the pretrained models were originally trained on images of 224x224 so we needed to ensure that the images were around this same size in order to improve the performance of the model. The kernels of the pretrained model are trained to extract features from a 224x224 size so it is important to have the dimensions of the image around that size. We first trained our model by incorporating a pretrained Resnet-18 model. During our training attempts for Resnet-18 using the 250x250 images, we froze the first 15, 16, and 17 layers (leaving the last 3, 2, and 1 layers for finetuning) to see how it changes the accuracy of our model. After we ran our model by incorporating the ResNet-18 model, we tried using a deeper CNN model, a pretrained ResNet-34 model. This was done in order to see if our model was underfit in terms of not being complex enough to differentiate between the features of a benign and malignant image.

For our second attempt, we shifted to training our model with larger cropped images of 500x500. We picked these dimensions since a lot of the images were much larger than 250x250 so we selected a size that allowed for more of the images to keep most of their features. We repeated the same procedure as the previous attempt of freezing up to the second to last layer. For this attempt, we also used a pretrained ResNet-18 and ResNet-34 model.



**Figure 3.2: Comparison of cropped images.** The first image is an original malignant image(1170x1160) from the CBIS-DDSM dataset. The second image is the same image but center cropped to 500x500 which is what we used for our model in attempt 2, and the third image is the same image but center cropped to 250x250 which is what we used for our model in attempt 1.

For our third attempt, we resized the images to 250x250. Unlike cropping, resizing would keep all the features of the image intact. However, one drawback we knew that would happen is that the resolution of the images would decrease. Like the previous attempt, we tried freezing up to the second to last layer. We also used both the ResNet-18 and ResNet-34 pretrained models.



**Figure 3.3: Comparison of resized images.** The left image is an original image(97x101) from the dataset. The image on the right is the resized version of the image to 250x250.

For our final attempt, we manually removed all the images from the dataset that were larger than 620x620. These dimensions were arbitrarily picked since most of the images fell under these

dimensions. 17% of the training images were removed. The images used in this attempt were first cropped to 500x500 and then resized to 250x250. We believed that removing the very large images would improve the accuracy since the images that were left had more of the unique features and edges still in place after the cropping.

Throughout all four attempts, some constant parameters that were set were the learning rate of 0.001, the number of epochs at 40, and a batch size of 16 images. A batch size of 16 images is commonly used and the number of epochs was arbitrarily chosen. The learning rate was chosen of 0.001 as a smaller learning rate allows for more training spent on the weights of each layer. See (Appendix C, Figure C.2) to see how the loss of the model increased when increasing learning rate. For all of the iterations, our model gave us its training accuracy, test accuracy, test loss, and training loss.

# Chapter 4: Results

## 4.1 Introduction

For the results, we focused on the training and testing accuracy given by our code. While we collected data for loss, the loss results were relatively similar and deemed negligible as our fine tuning parameters were set similar to each other. The goal of analyzing accuracy is to gather the initial results by modifying the datasets and determining if the data is sufficient for training to optimize our classification model. From collecting and analyzing accuracy, we have the baseline idea of whether or not our model was underfitting or overfitting itself to the data. Moreover, the best resulting dataset will be used for future work in more complicated finetuning of our classification models.

## 4.2 Training Model with Cropped Images of 250x250

For our first attempt, we used center cropped images of 250x250 when training our models. We used pretrained ResNet-18, 34, and 50 models for this iteration.



#### 4.2.1 Results of Using Cropped Images of 250x250

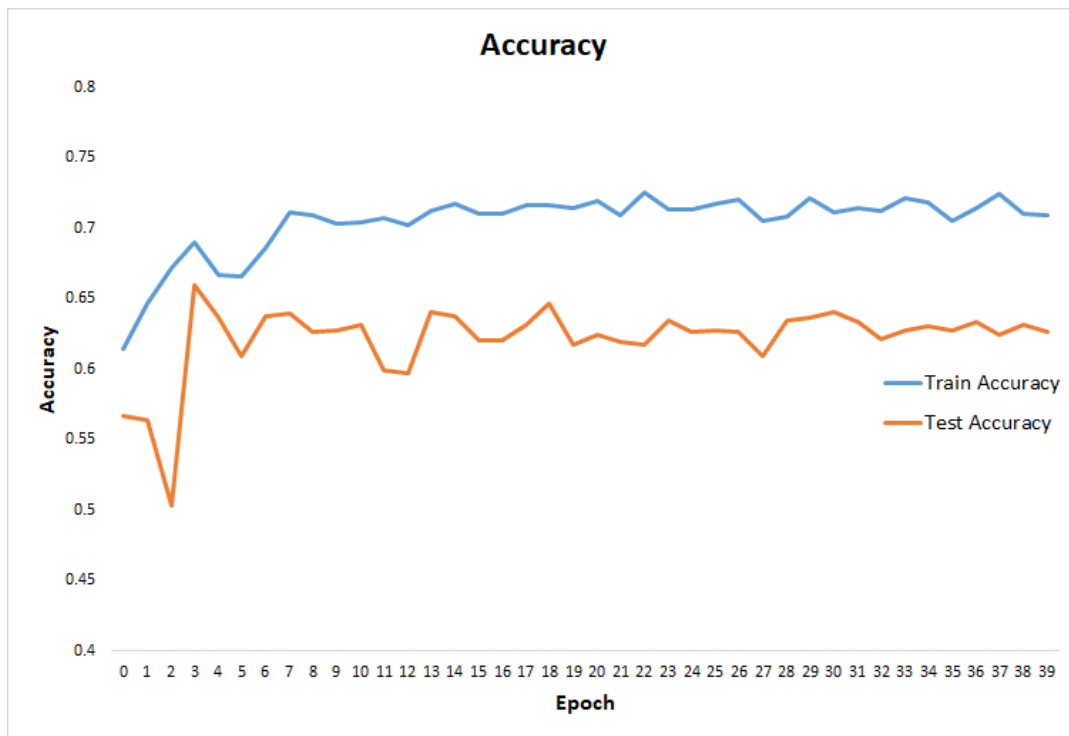


Figure 4.1: Accuracy of using pretrained ResNet-18 with cropped 250x250 images.

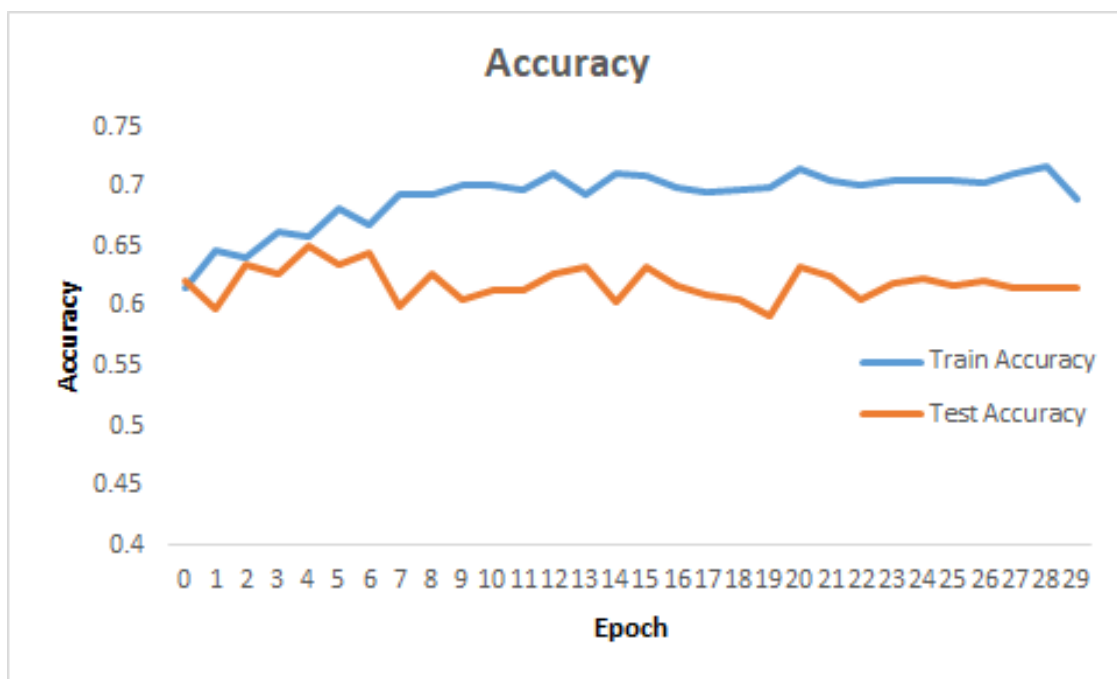
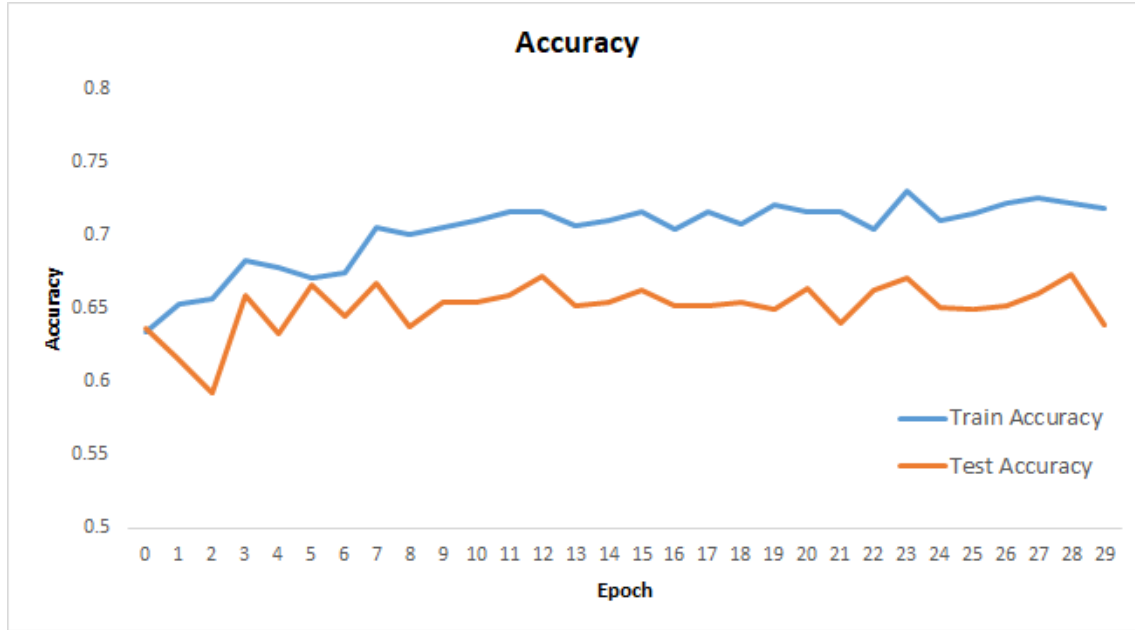


Figure 4.2: Accuracy of using pretrained ResNet-34 with cropped 250x250 images.



**Figure 4.3: Accuracy of using pretrained ResNet-50 with cropped 250x250 images.**

As seen in **Figures 4.1-4.3**, the highest accuracy achieved using cropped images of 250 x 250 was with the pretrained ResNet-50 model. This model's highest test accuracy was 0.67 at epoch 28 and the training accuracy was .72 at the same epoch. When using the ResNet-34 model, the highest accuracy was slightly lower. The highest test accuracy was .65 and its respective training accuracy was .66 at epoch 4. Using the pretrained ResNet-18 model, the highest test accuracy was .66 and its training accuracy was .69.

#### 4.2.2 Data interpretation of Using Cropped Images 250x250

The accuracy throughout the epochs remained relatively constant. The training accuracy and test accuracy for each model were similar which indicates a low variance and a high bias. The high bias shows that the model has a tendency to miss the relations between specific features and its correct output. However, since ResNet-50 had a slightly higher accuracy than the other two models, it indicates that shifting to a more complex model such as ResNet-101 or ResNet-152 could increase the accuracy more. In addition, another possible reason for the low accuracy could be that cropping the images to 250x250 removes too many unique features and edges which makes it harder for the model to classify the image as many of the images had dimensions over 500 pixels.

## 4.3 Training Model with Cropped Images of 500x500

We decided to shift to center cropping to larger dimensions of 500x500 in order to keep more of the image features intact. For this attempt, we tried using both a pretrained ResNet-18 and ResNet-34 model.

### 4.3.1 Results of Using Cropped Images 500x500

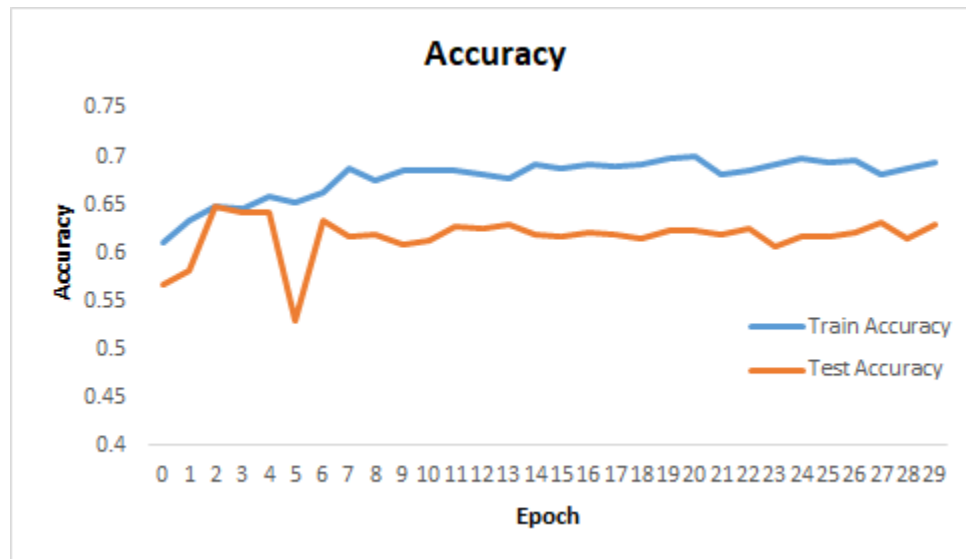


Figure 4.4: Accuracy of using ResNet-18 model with cropped 500x500 images.

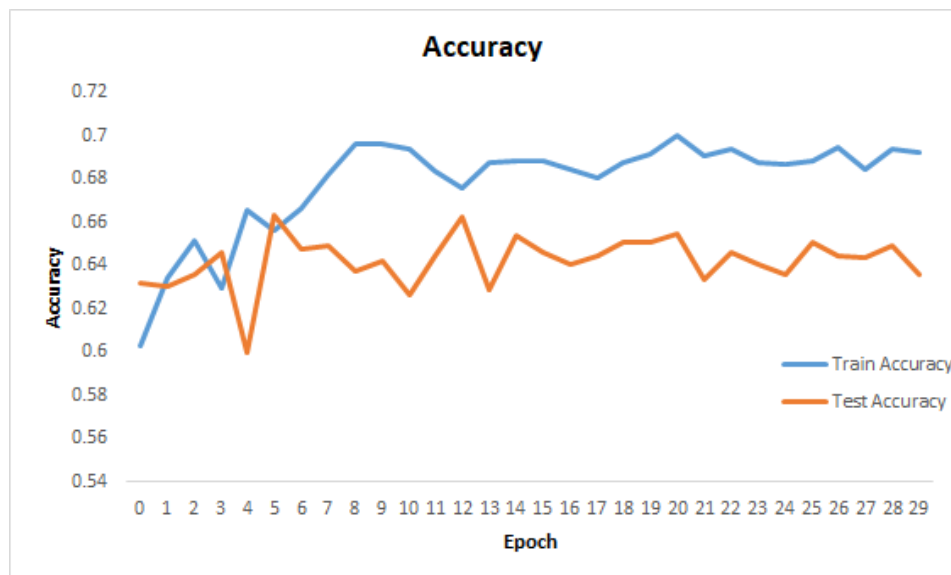


Figure 4.5: Accuracy of using ResNet-34 model with cropped 500x500 images.

As seen in **Figures 4.4-4.5**, the highest accuracy achieved using cropped images of 500x500 was achieved using the pretrained ResNet-34 model. This model's highest test accuracy was .664 and the training accuracy was 0.656. Using ResNet-18, the highest test accuracy was .648 and the training accuracy was .674.

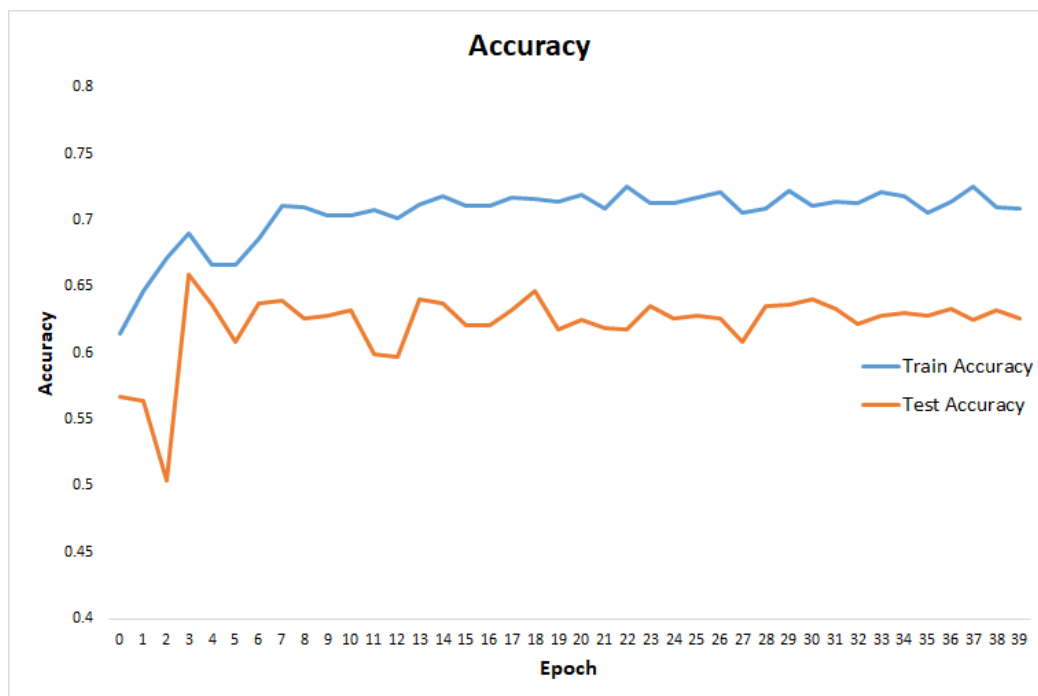
#### 4.3.2 Data Interpretation of Using Cropped Images 500x500

The accuracies compared to using cropped images of 250x250 were relatively the same. Since the ResNet-34 model had a slightly higher accuracy, it could indicate that the model is slightly underfit in terms of not being complex enough to differentiate between specific features of the images. Cropping to a larger size had a minimal impact on the accuracy.

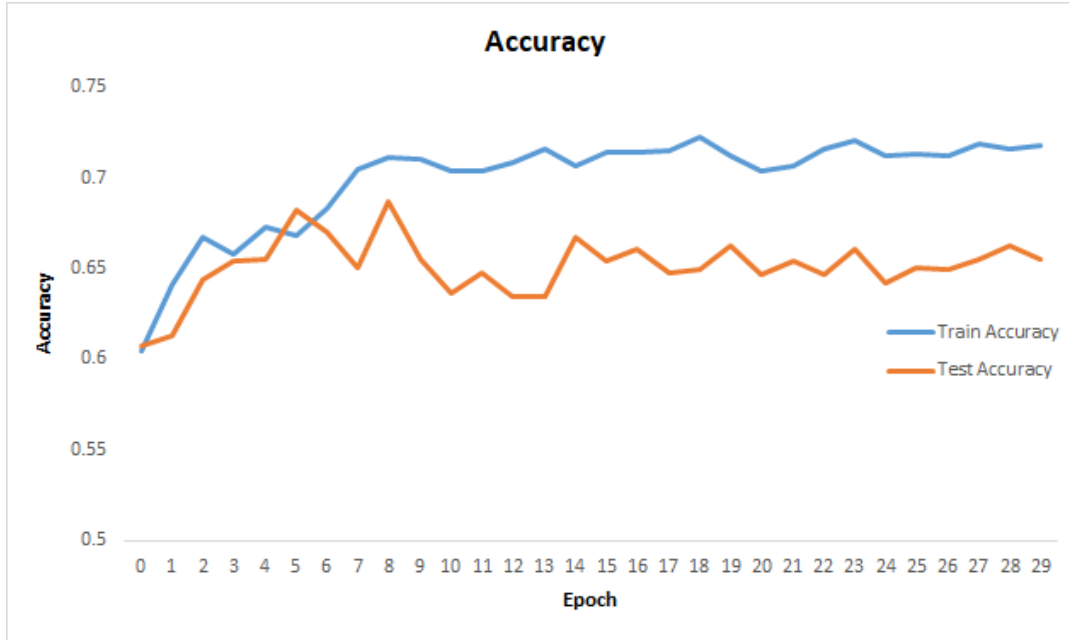
### 4.4 Training Model with Resized Images of 250 x 250

Since the accuracy using cropped images was relatively low, we shifted to training the model with resized images. We used the pretrained ResNet-18 and 34 models for this attempt.

#### 4.4.1 Results of Resizing Images to 250x250



**Figure 4.6: Accuracy of using ResNet-18 with resized 250x250 images.**



**Figure 4.7: Accuracy of using ResNet-34 with Resized 250x250 images.**

As seen in **Figures 4.6-4.7**, the highest accuracy using resized images to 250x250 was achieved using a pretrained ResNet-34 model. This model's test accuracy was .687 and the training accuracy was .711. Using the ResNet-18 model, the accuracy was slightly lower at .66 with a training accuracy of .69.

#### 4.4.2 Data Interpretation of Resizing Images

This proved to be the best method in terms of accuracy. Resizing ensures that the entire image is kept intact which makes it easier for the model to distinguish between a malignant and benign tumor. However, the accuracy using this preprocessing technique still did not significantly improve. One detriment with resizing is that resizing a really large image or really small image would impact the image quality as the resolution would decrease. This could make it harder for the model to distinguish between the specific features of a benign and malignant tumor.

#### 4.5 Training a Model with Images less than 620x620

In order to overcome the poor quality images due to resizing, we decided to take out images that were larger than 620x620. The images used in this iteration were first cropped to 500x500 and then resized to 250x250. We used pretrained ResNet-18 and ResNet-34 models.

#### 4.5.1 Results of Manually picking out Images larger than 620x620

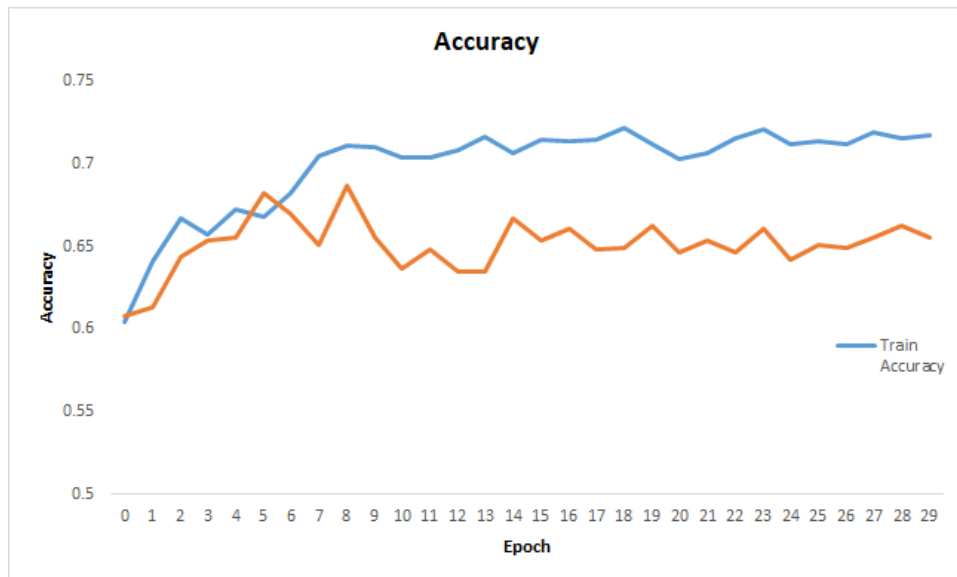


Figure 4.8: Accuracy of using ResNet-18 with images less than 620x620.

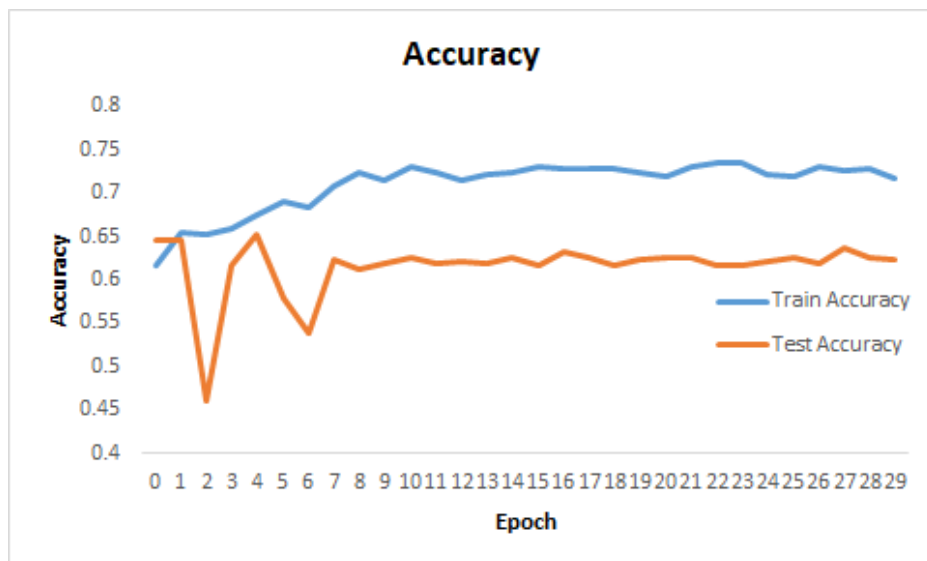


Figure 4.9: Accuracy of using ResNet-34 with images less than 620x620.

When looking at **Figures 4.8-4.9**, the results were very similar to the previous attempts. Both models had the same test accuracy of .653, while the ResNet-18 model had a training accuracy of 0.679 and the ResNet-34 model had a training accuracy of 0.674.

### 4.5.2 Data Interpretation of Manually picking out images

Only using images less than 620x620 had no improvement in accuracy. This conveys that our model may not be finetuned enough to classify mammogram images.

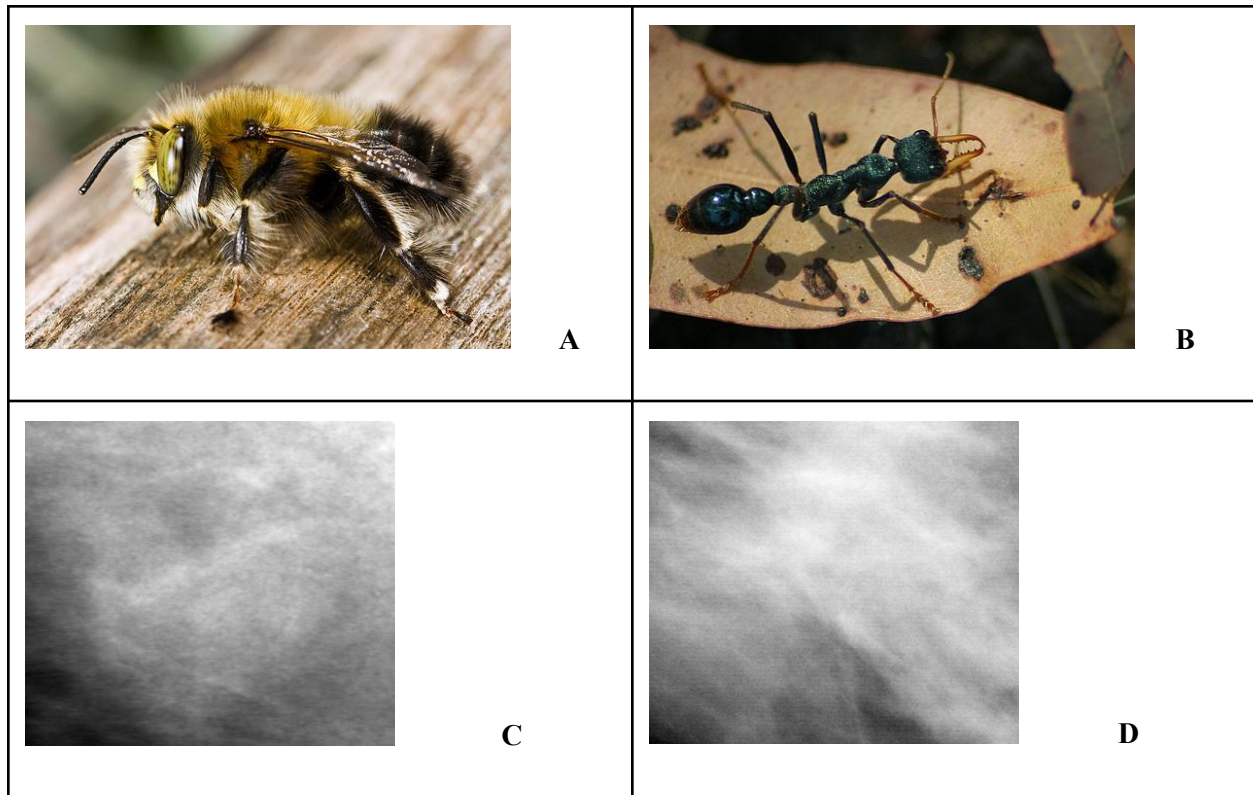
## 4.6 Overall Results

Our results, with a highest testing accuracy of 68.7% and a training accuracy of 71.1% using resnet-34 model and resized images, were not as high as we hoped our model would achieve. We decided to see if our code for our model was correct by using a commonly used dataset for image classification training.

### 4.6.1 Testing Results Using an Ants and Bees Dataset

To test whether the low accuracy we obtained may have been due to a coding problem, we tested one of our models on an ants and bees dataset. We tested this on a pretrained Resnet-18 model, which was run on a dataset provided by pytorch of bees and ants. Our model gave a testing accuracy of around 93% (See Appendix C, Figure C.1). This high accuracy eliminated the possibility of a coding error as it was able to differentiate between ants and bees relatively well.

ResNet was trained on a database used for multiple classes with all sorts of shapes and features. From a visual look, the bees and ants have sharp edges and features can be easily extracted and distinguished that make them bees and ants (See Figure 4.10). However, from the benign tumor and the malignant tumor images, that distinction isn't as clear. Because of this, some of the benign and malignant tumor images may be too similar for our model to pick up. The Imagenet feature extraction layers were not differentiating benign and malignant tumor features as well as it did on the ants and bee features. The features imagenet extracts may be irrelevant without fine tuning the convolutional layers since the tumors are much more vaguely defined. Another possible explanation may be that our models were still underfitted even with Resnet-34, so our model was too simple to find the differences between malignant and benign tumors in more vague images.



**Figure 4.10: Comparing mammogram images with nonmedical images.** A and B are images taken from the Pytorch tutorial dataset classifying bees and ants [9]. C and D is from the CBIS-DDSM dataset. C is a benign tumor and D is a malignant tumor.

#### 4.6.2 Overall Analysis and Interpretation

From the four different preprocessing attempts, resizing the images to 250x250 with ResNet-34 achieved the highest accuracy. However, due to the relatively low accuracy, our model could have been underfit as it might need a deeper neural network with more CNN layers to differentiate the features between a malignant and benign tumor. Modifying the parameters such as epochs, batch size, and learning rate did not improve the accuracy. Another important parameter was the number of layers that were frozen. Frozen layers are layers whose weights are not trained with the training data set and can't be changed as a result. We tried freezing up to the 3rd to last layer but there was a minimal difference in the accuracies (See Appendix C, Table C.1). After achieving a high accuracy with the ants and bees dataset using our model, we can conclude that our model works well for image classification for most datasets that have clear and sharp features. However, since the mammograms when comparing benign and malignant have



less distinguishing differences, the convolutional layers of our model would need to be finetuned rather than just optimizing the parameters and preprocessing technique used.

## 4.7 Future Steps

In order to better evaluate the performance of our model, a receiver operating characteristic (ROC) analysis should be performed. The ROC curve will show how well the model is at predicting a benign versus malignant tumor. This evaluation will allow us to visualize the pitfalls of the model and make modifications to improve its accuracy. Once an ROC curve is generated, we will need to focus on how to improve the model. The first approach would be to apply data augmentation through methods such as rotating, scaling, and flipping. Increasing the dataset will give our model more cases of each classification so the model can become adept to mammogram images. The second approach would be to try deeper CNN models such as ResNet-101 and ResNet-152. Another approach would be to use a pretrained model that was originally trained on a medical dataset.

# Chapter 5: Engineering Ethics with Breast Cancer Classification

## 5.1: Ethical Justifications of the Project

For healthcare, AI can be used for detecting diseases and allows an objective method to detect diseases like cancer better than the medical professionals. Our senior project strived to create a deep learning classification model that radiologists can use as a tool for diagnosing breast cancer in mammogram images. If done successfully, the model would allow for more patients to be seen with more accuracy and speed. Moreover, a successful deep learning model increases the likelihood that patients keep their universal right for healthcare.

If a deep learning-based diagnosis tool detects diseases at the same effectiveness or better than a radiologist, it should be used since the model would provide a precise method of detecting diseases so more lives may be treated. Radiologists would also save the time working through a diagnosis to work with other tasks like seeing more patients. Society betters itself by providing a more efficient healthcare system, reducing time for better diagnosis for patients, and allowing medical professionals more time to do their duty with more people. Ostensibly, the models better both society and the stakeholders in healthcare.

## 5.2: Our Project's Engineering Standards

The previous section alludes to our duty as bioengineers to enhance the safety, health, and welfare of the public. This duty is fundamental for specific virtues learned throughout the year working on our project. More specifically, we will be mentioning documentation and overall rigor for good engineering.

Throughout the design process, documentation became a prominent part of our project. This allows us to enhance safety by negating potential issues. For our deep learning training, images of benign and malignant tumors go into the model and adjust the model for cancer detection. Training with faulty data amplifies risks through using the images for multiple cycles. Bad data

makes a failed model. So, to minimize human error and this devastating risk, we established proper documentation and labeling through thoroughly reviewing and labeling.

Our project showed us the precision and detail-orientation needed to make an experiment happen. This is exemplified in our project with the largest issue we had. For our project we used an open-source dataset for training our deep learning model due to the limitations in gathering massive amounts of medical images. In the dataset, we have ~3,400 (depending on the experiment) mammogram images split into malignant and benign folders for training and testing. We documented and communicated to each other about all the augmentations we did to the dataset, from combining different classes of malignant tumors together and different classes of benign tumors together, as well as marking the exact number of images used per model training session. Yet, we initially made a fundamental error in assuming the dataset was labeled correctly. Through meticulous actions, we visually confirmed the dataset and found that the dataset labels had mislabeled and swapped the mammogram images with their corresponding masks. Without thoroughness, our dataset would have been flawed from the beginning. We picked out 289 images from our benign training folder, 297 images from our malignant training folder, and 2 images from the malignant test folder and replaced them with the proper images. If we continued with our original assumption that the dataset is correct, these incorrect images would have skewed our classification of cancer. This experience showed us in using public data, while beneficial to have the large public datasets, the database needs to be checked for errors. Blind trust without verification may have severe consequences to creating a proper product.

### 5.3: Breast Cancer Classification Models, Safety, Risk and Informed Consent

While we learned the foundations to create an ethical model through our senior design project, and if the model is successful, there are still ethical concerns with implementation. If we imagine the general steps to training a model, first there is the input images, then the model itself, and the output classification. The training images may be different from the image of the disease and cause issues. This is amplified from the model being a black box where we do not know precisely what the model is detecting to adjust itself, causing epistemic issues in classification

and justification. When considering the output classification, significant issues for the patient's safety as well as ethical questions for the radiologist to use the model for diagnosis.

Overall, deep learning algorithms have the potential to change the future of healthcare. After all, even now there are models with high accuracy in detecting specific diseases. However, there needs to be more information to clarify the unknowns. For example, medical images are hard to come by due to patient privacy. And even if we get the images, there is a lack of certainty in the diversity of the dataset and even a lack of certainty if the images are classified correctly by the radiologists for training. In addition to images, there is significant concern with not knowing the justifications for a model's diagnosis even if the model is accurate. With time, if there is clarity for both the images and how a model adjusts itself, a combination of the accuracy and reasoning for diagnosis paves the way for significant change in healthcare.

More information in Appendix D for ethical concerns for deep learning classification models in general.

## Chapter 6: Conclusion

We were able to train pretrained CNN models of ResNet-18, 34, and 50 on an open source mammogram dataset to predict whether an image contains a benign or malignant tumor. Across all of our attempts, resizing the images to 250x250 on ResNet-34 had the highest testing accuracy of 68.7% and a training accuracy of 71.1%. Although this accuracy may be a little bit low, we suspect that this low accuracy may be due to how image features are not easily distinguishable between benign and malignant tumors. To improve upon our accuracy, future work of creating an ROC curve should be made to realize the pitfalls of the model and make modifications based on the curve. In addition, using a deeper pretrained model such as ResNet-101 and 152 or a pretrained medical model could improve upon its accuracy further.

# References

1. *U.S. Breast Cancer Statistics*. (2021, February 4). Breastcancer.org. Retrieved June 6, 2021, from [https://www.breastcancer.org/symptoms/understand\\_bc/statistics](https://www.breastcancer.org/symptoms/understand_bc/statistics)
2. Mayo Clinic Staff. (n.d.) *Breast Cancer*. Mayo Clinic. <https://www.mayoclinic.org/diseases-conditions/breast-cancer/symptoms-causes/syc-20352470>
3. *Limitations of Mammograms: How Often are Mammograms Wrong?* American Cancer Society. (2019, October 3). <https://www.cancer.org/cancer/breast-cancer/screening-tests-and-early-detection/mammograms/limitations-of-mammograms.html#:~:text=Overall%2C%20screening%20mammograms%20do%20not,when%20in%20fact%20they%20do>
4. Saha, Sumit. (2018, December 15). *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. towards data science. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
5. Angeles, S. (2018). Mammo: An Application of Convolutional Neural Networks on Breast Cancer Screening. *University of the Philippines Manila College of Arts and Sciences*.
6. Tsochatzidis, L., Costaridou, L., & Pratikakis, I. (2019). Deep learning for breast cancer diagnosis from mammograms—a comparative study. *Journal of Imaging*, 5(3), 37. doi:10.3390/jimaging5030037
7. Agarwal, R., Diaz, O., Lladó, X., Yap, M. H., & Martí, R. (2019). Automatic mass detection in mammograms using deep convolutional neural networks. *Journal of Medical Imaging*, 6(03), 1. doi:10.1117/1.jmi.6.3.031409
8. Falconi, L. G., Perez, M., Aguila, W. G., & Conci, A. (2020). Transfer learning and fine tuning in breast mammogram abnormalities classification on cbis-ddsm database. *Advances in Science, Technology and Engineering Systems Journal*, 5(2), 154-165. doi:10.25046/aj050220
9. Python Engineer. (2020, February 12). PyTorch Tutorial 15 - Transfer Learning. Retrieved from <https://www.youtube.com/watch?v=K0lWSB2QoIQ&list=PLqnsIRFeH2UrcDBWF5mfPGpqQDSta6VK4&index=15>
10. Houssein, E. H., Emam, M. M., Ali, A. A., & Suganthan, P. N. (2021). Deep and machine learning techniques for medical imaging-based breast cancer: A comprehensive review. *Expert Systems with Applications*, 167, 114161. <https://doi.org/10.1016/j.eswa.2020.114161>

11. Ibrahim, A., Gamble, P., Jaroensri, R., Abdelsamea, M. M., Mermel, C. H., Chen, P.-H. C., & Rakha, E. A. (2020). Artificial intelligence in digital breast pathology: Techniques and applications. *The Breast*, 49, 267–273. <https://doi.org/10.1016/j.breast.2019.12.007>
12. Wu, N., Phang, J., & Shen, Y. (2020). Deep Neural Networks Improve Radiologists' Performance in Breast Cancer Screening. *IEEE Transactions on Medical Imaging*.
13. Guo, M., & Du, Y. (2019). Classification of Thyroid Ultrasound Standard Plane Images using ResNet-18 Networks. *2019 IEEE 13th International Conference on Anti-Counterfeiting, Security, and Identification (ASID)*. <https://doi.org/10.1109/icasid.2019.8925267>
14. Yari, Y., Nguyen, T. V., & Nguyen, H. T. (2020). Deep Learning Applied for Histological Diagnosis of Breast Cancer. *IEEE Access*, 8, 162432–162448. <https://doi.org/10.1109/access.2020.3021557>

# Appendix

## Appendix A: Example of Our Model Code

Example Code of our general modelling format. Example from our ResNet-18 with custom “picked” CBIS-DDSM dataset training.

```
import os
import numpy as np
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
import torch.optim as optim
import time
import copy
import torchvision
from torch.optim import lr_scheduler
from torchvision import datasets, models, transforms
from torchvision.transforms import Compose
from torchvision.transforms import ToTensor
import torch.nn.functional as F

mean = np.array([0.5, 0.5, 0.5])
std = np.array([0.25, 0.25, 0.25])

#### Load Data ####
# Additional modifications to image size done here #
data_transforms = {
    'train': transforms.Compose([
        transforms.CenterCrop(500),
        transforms.Resize(250),
        transforms.ToTensor(),
        transforms.Normalize(mean, std)
    ]),
    'test': transforms.Compose([
        transforms.CenterCrop(500),
        transforms.Resize(250),
        transforms.ToTensor(),
        transforms.Normalize(mean, std)
    ]),
}
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
data_dir = 'CBIS_png'
image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x),
                                                    data_transforms[x])
                  for x in ['train', 'test']}
dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size=16,
```



```

        shuffle=True, num_workers=0)
    for x in ['train', 'test']}
dataset_sizes = {x: len(image_datasets[x]) for x in ['train', 'test']}
class_names = image_datasets['train'].classes

##### Testing to see if dataset loaded and results of our python-based modifications #####

#Check if data is loaded
#class_names = image_datasets['train'].classes
image=image_datasets['train']
print(image)

#class_names = image_datasets['test'].classes
image_test=image_datasets['test']
print(image_test)

print(class_names)
#Check for correct preprocessing
def imshow(inp, title):
    """Imshow for Tensor."""
    inp = inp.numpy().transpose((1, 2, 0))
    inp = std * inp + mean
    inp = np.clip(inp, 0, 1)
    plt.imshow(inp)
    plt.title(title)
    plt.show()

# Get a batch of training data
inputs, classes = next(iter(dataloaders['train']))

# Make a grid from batch
out = torchvision.utils.make_grid(inputs)

imshow(out, title=[class_names[x] for x in classes])

##### CNN Model parameters #####
def train_model(model, criterion, optimizer, scheduler, num_epochs=30):
    since = time.time()

    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0

    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))
        print('-' * 10)

        # Each epoch has a training and validation phase
        for phase in ['train', 'test']:

```

```

if phase == 'train':
    model.train() # Set model to training mode
else:
    model.eval() # Set model to evaluate mode

running_loss = 0.0
running_corrects = 0

# Iterate over data.
for inputs, labels in dataloaders[phase]:
    inputs = inputs.to(device)
    labels = labels.to(device)

    # forward
    # track history if only in train
    with torch.set_grad_enabled(phase == 'train'):
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        loss = criterion(outputs, labels)

    # backward + optimize only if in training phase
    if phase == 'train':
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    # statistics
    running_loss += loss.item() * inputs.size(0)
    running_corrects += torch.sum(preds == labels.data)

if phase == 'train':
    scheduler.step()

epoch_loss = running_loss / dataset_sizes[phase]
epoch_acc = running_corrects.double() / dataset_sizes[phase]

print('{} Loss: {:.4f} Acc: {:.4f}'.format(
    phase, epoch_loss, epoch_acc))

# deep copy the model
if phase == 'test' and epoch_acc > best_acc:
    best_acc = epoch_acc
    best_model_wts = copy.deepcopy(model.state_dict())

print()

time_elapsed = time.time() - since
print("Training complete in {:.0f}m {:.0f}s".format(
    time_elapsed // 60, time_elapsed % 60))

```

```

print('Best test Acc: {:.4f}'.format(best_acc))

# load best model weights
model.load_state_dict(best_model_wts)
return model

#### Resnet ####
#Parameters specifically for loaded in Pretrained Resnet model
model_conv = torchvision.models.resnet18(pretrained=True)

#Freezing the first few layers done here. Here, freezing the first 17 layers #
ct = 0
for name, child in model_conv.named_children():
    ct += 1
    if ct < 17:
        for name2, params in child.named_parameters():
            params.requires_grad = False

num_fts = model_conv.fc.in_features
model_conv.fc = nn.Linear(num_fts, 2)

model_conv = model_conv.to(device)

criterion = nn.CrossEntropyLoss()

# Only parameters of final layer are being optimized
optimizer_conv = optim.SGD(model_conv.fc.parameters(), lr=0.001, momentum=0.9)

# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=7, gamma=0.1)

model_conv = train_model(model_conv, criterion, optimizer_conv,
                          exp_lr_scheduler, num_epochs=30)

```

## Appendix B: Bash Shell Script Code to Preprocess Images

```
#!/bin/bash

# made by Derrick D. Nguyen
# convert cbis-ddsm dicom files into png and then put in folders
# skipping full dicom image

# !!!run the script right outside of the CBIS-DDSM folder!!!
# !!!make sure to change BENIGN_WITHOUT_CALLBACK to BENIGN in the csv files!!!

# make folders
mkdir CBIS_png
mkdir CBIS_png/train
mkdir CBIS_png/train/benign CBIS_png/train/malignant CBIS_png/train/benign_mask
CBIS_png/train/malignant_mask
mkdir CBIS_png/test
mkdir CBIS_png/test/benign CBIS_png/test/malignant CBIS_png/test/benign_mask CBIS_png/test/malignant_mask
mkdir CBIS_png/error

# convert test calcification images
{
    # read calc_case_description_test_set.csv
    read          # skips first row

    # patient_id, breast_density, left or right breast, image view, abnormality id, abnormality type, calc type,
    # calc distribution, assessment, pathology, the rest of the fields
    # only using patient_id, left_right, view, id, pathology
    while IFS=, read -r patient_id density left_right view id type calc_type distribution assessment pathology
    junk; do
        echo "$patient_id" "$left_right" "$view" "$id" "$pathology"

        # convert cropped
        dcmj2pnm --write-png --min-max-window
        CBIS-DDSM/Calc-Test_"$patient_id"_"$left_right"_"$view"_"$id"/*/1-1.dcm
        Calc-Test_"$patient_id"_"$left_right"_"$view"_"$id"_"$pathology".png
        dcmj2pnm --write-png --min-max-window
        CBIS-DDSM/Calc-Test_"$patient_id"_"$left_right"_"$view"_"$id"/*/1-2.dcm
        Calc-Test_"$patient_id"_"$left_right"_"$view"_"$id"_"$pathology"_mask.png

        # if conversion failed, output in error file
        # check to see if file missing is cropped or mask or both
        if [ ! -e *BENIGN.png ] && [ ! -e *MALIGNANT.png ]; then
            echo Calc-Test_"$patient_id"_"$left_right"_"$view"_"$id"_"$pathology" failed >>
            CBIS_png/error/error.txt
        elif [ ! -e *mask.png ]; then
```

```

        echo Calc-Test_"$patient_id"_"$left_right"_"$view"_"$id"_"$pathology" no mask >>
CBIS_png/error/error.txt
    else
        echo files exist pog
    fi

    # move png based on malignant or not
    # cropped
    if [ -e *MALIGNANT.png ]; then
        mv *MALIGNANT.png CBIS_png/test/malignant
    else
        mv *BENIGN.png CBIS_png/test/benign
    fi

    # mask
    if [ -e *MALIGNANT_mask.png ]; then
        mv *MALIGNANT_mask.png CBIS_png/test/malignant_mask
    else
        mv *BENIGN_mask.png CBIS_png/test/benign_mask
    fi

    # if for some reason pngs no move, put in error folder
    if [ -e *.png ]; then
        for bad in *.png; do
            echo bad
            mv "$bad" CBIS_png/error
        done
    fi
    echo
done
} < calc_case_description_test_set.csv

# convert train calcification images
{
    # read calc_case_description_train_set.csv
    read          # skips first row

    # patient_id, breast_density, left or right breast, image view, abnormality id, abnormality type, calc type,
    # calc distribution, assessment, pathology, the rest of the fields
    # only using patient_id, left_right, view, id, pathology
    while IFS=, read -r patient_id density left_right view id type calc_type distribution assessment pathology
    junk; do
        echo "$patient_id" "$left_right" "$view" "$id" "$pathology"

        # convert cropped
        dcmj2pnm --write-png --min-max-window
        CBIS-DDSM/Calc-Training_"$patient_id"_"$left_right"_"$view"_"$id"/*/1-1.dcm
        Calc-Train_"$patient_id"_"$left_right"_"$view"_"$id"_"$pathology".png
    done
}

```

```

        dcmj2pnm --write-png --min-max-window
CBIS-DDSM/Calc-Training_"$patient_id"_"$left_right"_"$view"_"$id"/*/1-2.dcm
Calc-Train_"$patient_id"_"$left_right"_"$view"_"$id"_"$pathology"_mask.png

        # if conversion failed, output in error file
        # check to see if file missing is cropped or mask or both
        if [ ! -e *BENIGN.png ] && [ ! -e *MALIGNANT.png ]; then
            echo Calc-Train_"$patient_id"_"$left_right"_"$view"_"$id"_"$pathology" failed >>
CBIS_png/error/error.txt
        elif [ ! -e *mask.png ]; then
            echo Calc-Train_"$patient_id"_"$left_right"_"$view"_"$id"_"$pathology" no mask >>
CBIS_png/error/error.txt
        else
            echo files exist pog
        fi

        # move png based on malignant or not
        # cropped
        if [ -e *MALIGNANT.png ]; then
            mv *MALIGNANT.png CBIS_png/train/malignant
        else
            mv *BENIGN.png CBIS_png/train/benign
        fi

        # mask
        if [ -e *MALIGNANT_mask.png ]; then
            mv *MALIGNANT_mask.png CBIS_png/train/malignant_mask
        else
            mv *BENIGN_mask.png CBIS_png/train/benign_mask
        fi

        # if for some reason png no move, put in error folder
        if [ -e *.png ]; then
            for bad in *.png; do
                mv "$bad" CBIS_png/error
            done
        fi
        echo
    done
} < calc_case_description_train_set.csv

# convert test mass images
{
    # read mass_case_description_test_set.csv
    read          # skips first row

    # patient_id, breast_density, left or right breast, image view, abnormality id, abnormality type, mass shape,
    mass margins, assessment, pathology, the rest of the fields

```

```

# only using patient_id, left_right, view, id, pathology
while IFS=, read -r patient_id density left_right view id type mass_type margin assessment pathology junk;
do
    echo "$patient_id" "$left_right" "$view" "$id" "$pathology"

    # convert cropped
    dcmj2pnm --write-png --min-max-window
    CBIS-DDSM/Mass-Test_"$patient_id"_"$left_right"_"$view"_"$id"/*/1-1.dcm
    Mass-Test_"$patient_id"_"$left_right"_"$view"_"$id"_"$pathology".png
    dcmj2pnm --write-png --min-max-window
    CBIS-DDSM/Mass-Test_"$patient_id"_"$left_right"_"$view"_"$id"/*/1-2.dcm
    Mass-Test_"$patient_id"_"$left_right"_"$view"_"$id"_"$pathology"_mask.png

    # if conversion failed, output in error file
    # check to see if file missing is cropped or mask or both
    if [ ! -e *BENIGN.png ] && [ ! -e *MALIGNANT.png ]; then
        echo Mass-Test_"$patient_id"_"$left_right"_"$view"_"$id"_"$pathology" failed >>
        CBIS_png/error/error.txt
    elif [ ! -e *mask.png ]; then
        echo Mass-Test_"$patient_id"_"$left_right"_"$view"_"$id"_"$pathology" no mask >>
        CBIS_png/error/error.txt
    else
        echo files exist pog
    fi

    # move png based on malignant or not
    # cropped
    if [ -e *MALIGNANT.png ]; then
        mv *MALIGNANT.png CBIS_png/test/malignant
    else
        mv *BENIGN.png CBIS_png/test/benign
    fi

    # mask
    if [ -e *MALIGNANT_mask.png ]; then
        mv *MALIGNANT_mask.png CBIS_png/test/malignant_mask
    else
        mv *BENIGN_mask.png CBIS_png/test/benign_mask
    fi

    # if for some reason png no move, put in error folder
    if [ -e *.png ]; then
        for bad in *.png; do
            mv "$bad" CBIS_png/error
        done
    fi
    echo
done
} < mass_case_description_test_set.csv

```

```

# convert train mass images
{
    # read mass_case_description_train_set.csv
    read          # skips first row

    # patient_id, breast_density, left or right breast, image view, abnormality id, abnormality type, mass shape,
    # mass margins, assessment, pathology, the rest of the fields
    # only using patient_id, left_right, view, id, pathology
    while IFS=, read -r patient_id density left_right view id type mass_type margin assessment pathology junk;
do
    echo "$patient_id" "$left_right" "$view" "$id" "$pathology"

    # convert cropped
    dcmj2pnm --write-png --min-max-window
    CBIS-DDSM/Mass-Training "$patient_id" "$left_right" "$view" "$id" /*/*1-1.dcm
    Mass-Train "$patient_id" "$left_right" "$view" "$id" "$pathology".png
    dcmj2pnm --write-png --min-max-window
    CBIS-DDSM/Mass-Training "$patient_id" "$left_right" "$view" "$id" /*/*1-2.dcm
    Mass-Train "$patient_id" "$left_right" "$view" "$id" "$pathology"_mask.png

    # if conversion failed, output in error file
    # check to see if file missing is cropped or mask or both
    if [ ! -e *BENIGN.png ] && [ ! -e *MALIGNANT.png ]; then
        echo Mass-Train "$patient_id" "$left_right" "$view" "$id" "$pathology" failed >>
    CBIS_png/error/error.txt
    elif [ ! -e *mask.png ]; then
        echo Mass-Train "$patient_id" "$left_right" "$view" "$id" "$pathology" no mask >>
    CBIS_png/error/error.txt
    else
        echo files exist pog
    fi

    # move png based on malignant or not
    # cropped
    if [ -e *MALIGNANT.png ]; then
        mv *MALIGNANT.png CBIS_png/train/malignant
    else
        mv *BENIGN.png CBIS_png/train/benign
    fi

    # mask
    if [ -e *MALIGNANT_mask.png ]; then
        mv *MALIGNANT_mask.png CBIS_png/train/malignant_mask
    else
        mv *BENIGN_mask.png CBIS_png/train/benign_mask
    fi
fi

```



```

        # if for some reason png no move, put in error folder
        if [ -e *.png ]; then
            for bad in *.png; do
                mv "$bad" CBIS_png/error
            done
        fi
        echo
    done
} < mass_case_description_train_set.csv

#####

# commands used to fix naming of misplaced images
# run this on bash command window to rename misplaced masks (insert “_mask” name)
for f in *.png; do mv "%f" "${f%.png}_mask.png"; done

# run this on powershell to rename misplaced images (remove “_mask” name)
get-childitem *.png | foreach { rename-item $_ $_.Name.Replace("_mask", "") }

```

## Appendix C: Supplementary Images and Tables

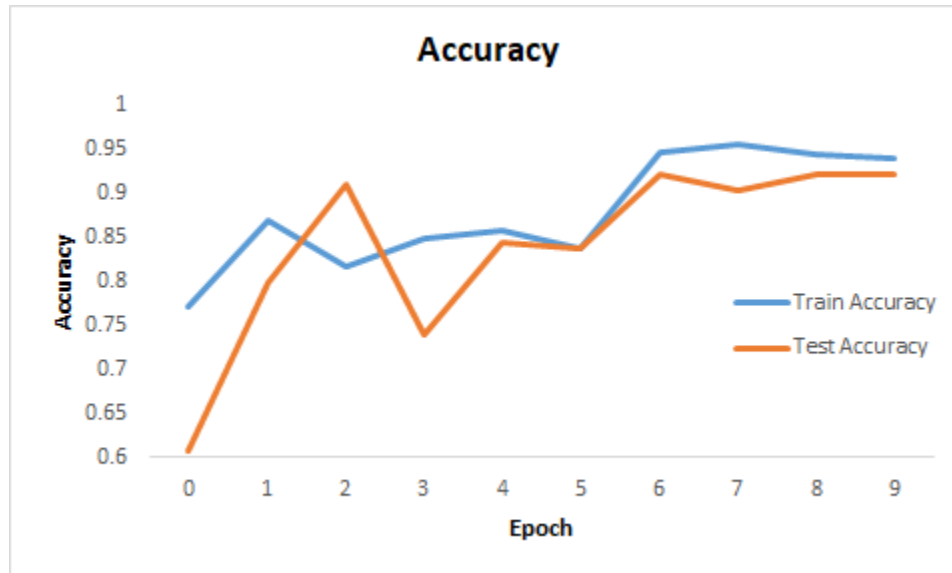
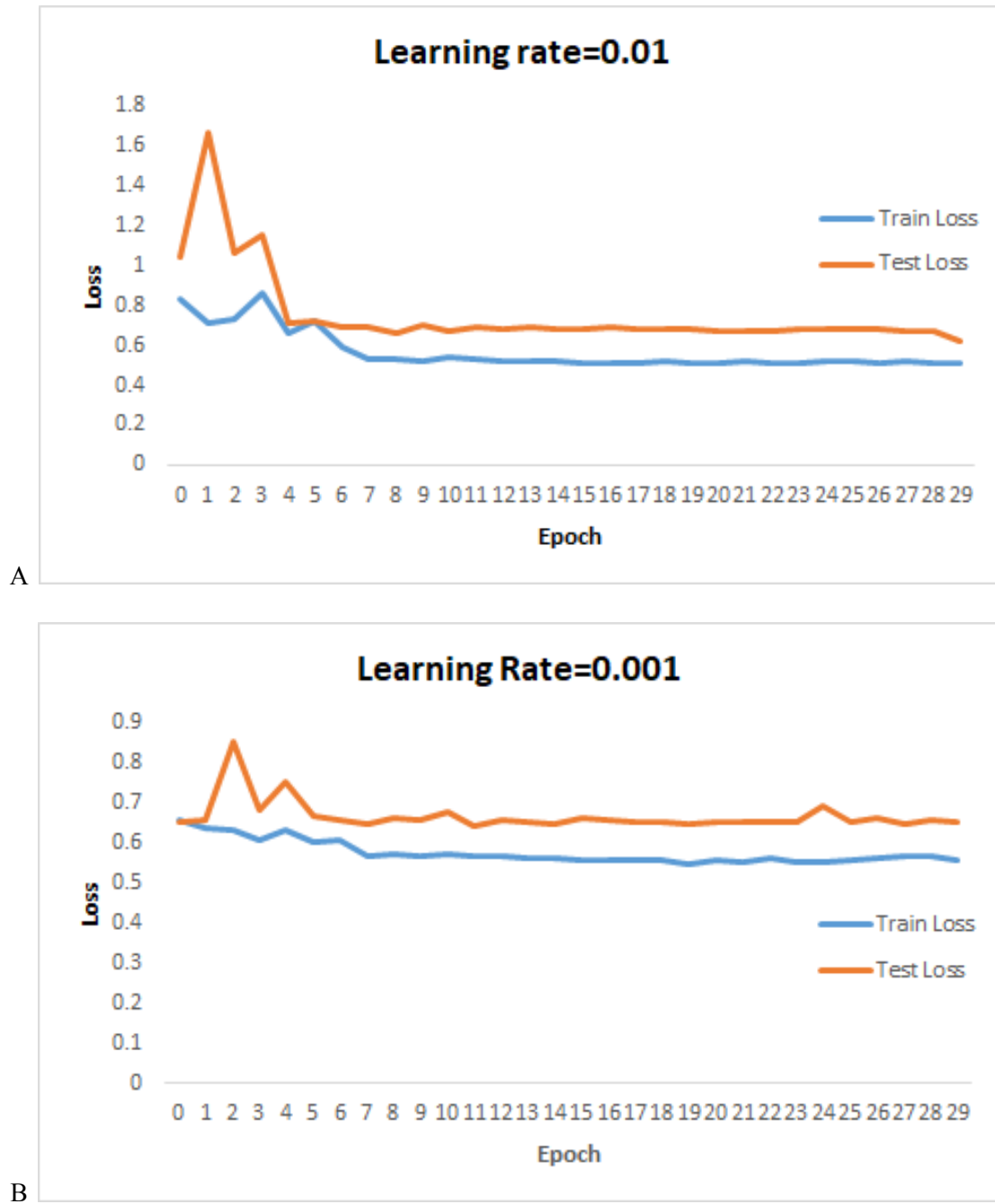


Figure C.1: Accuracy of using ResNet-18 for classification between bees and ants.

### Results of Training Resnet 34 Model

	Cropping to 250x250	Cropping to 500x500	Resizing to 250x250	Manually taking out large images
Best Test Accuracy	0.651	0.664	<b>0.687</b>	0.653
Train Accuracy	0.658	0.656	<b>0.711</b>	0.674
Layers unfrozen	3	2	2	2

**Table C.1: Results of Training ResNet-34 Models.** Resizing images to 250x250 gave the highest testing and training accuracy of 0.687 and 0.711, respectively.



**Figure C.2: Differences in learning rates with respect to its loss.** A and B demonstrate the loss based on changing the learning rate. This was done on a pretrained ResNet-18 model with 250x250 cropped images.

## Appendix D: Specific Ethical Concerns of Deep Learning Classification

There are ethical issues regarding safety, risk, and informed consent whenever any deep learning classification would be used. For our breast cancer classification, we created a binary classification model classifying tumors as either benign or malignant. While there are many potential ethical issues, we will be focusing on a few issues for each of the stated topics. First, the training process has ethical implications for the safety of the patients. Second, the results of the model create potential risks for both the patients and the medical professionals. Finally, there is a general issue of providing informed consent for both the training and usage of the model.

In evaluating the ethical concerns of creating our deep learning classification model, there are issues regarding safety based on the model's training. The goal of using a deep learning diagnosis model is to identify diseases better than radiologists. Elucidating on "better", we want a higher accuracy of identification as well as having an objective method of diagnosis. For instance, with radiologists, because they are human, there is a subjective nature to diagnosing a disease in images. This is emphasized if the images are unclear. Subjectivity stems from variations between radiologists and even within the same radiologist at different times. The model is supposed to have a higher accuracy in diagnosing a disease like breast cancer while keeping the same diagnosis every time. To get this accuracy, the model goes through supervised learning where it uses labeled images to finetune itself to perfect classification.

With the goal of the model's objectivity in mind, there are implications for creating the model. In our project, we mentioned finding a large amount of mammogram images mislabeled in the open-sourced database. For these mislabels, we were fortunate to be able to visually identify errors since images of tumors were replaced by images of black and white masks for the tumors. However, issues arise when we cannot visually identify mislabeled images.

There are significant risks to the safety of patients if the classification model uses mislabeled images. Because supervised learning is reliant on accurate labeling, in this case malignant tumors and benign tumors, the model will assume it is correct based on whatever data it is fed. In the open-source dataset, we are given the labels for each of the images. However, some of the images look extremely similar to each other and we would have no way of knowing if the labels

were correct or not. And we cannot omit those images because they are in a sense, the most important images to train the model. If the goal of the model is to be better at detecting hard to diagnose tumors in mammograms and if one cannot visually tell the difference between images, then those images are optimal for the model to be more accurate than a radiologist. In addition to training datasets, testing datasets may also be mislabeled and falsely confirm a high accuracy for a model. Because there is this unknown, a patient may not know the risks since nobody can confirm the datasets, as datasets are protected by privacy laws and radiologists would assume they made the correct labeling. There is still a need for radiologists to label the dataset, and with mislabeled data, the classification model would potentially misdiagnose patients and jeopardize their safety.

While mislabeled datasets pose a significant issue for training a model, the quality of the image of the patient's disease also poses potential issues with the results of using the model. Imagine having a breast cancer classification model with above 95% accuracy. This accuracy is based on a testing dataset that evaluates the performance of a model trained on a training dataset. If the training and testing dataset is like each other, then it would most likely give a high accuracy. If a radiologist takes a mammogram that has different aspects or any other differences from the model, then the accuracy may be falsely high since the model may take in the differences in images and give the wrong diagnosis.

In a more severe case, because each body is different, there may be physiological differences from the model's training and testing dataset as well as an epistemic concern of distinguishing how the model is detecting features of a disease for diagnosis. For instance, for detecting breast cancer in women, breast density may vary between people from different age groups and ethnicities. Because datasets usually do not label for ethnicity, there may be less diversity in the training and testing datasets, which gives a falsely high accuracy. A radiologist can factor in these variables. For instance, radiologists can account for breast density and they have features to look for, such as benign tumors are typically more round and malignant tumors are more misshaped. Yet for a model, people currently do not know the specifics an algorithm looks for since the model extracts numbers to feed through each layer of the neural network; a deep learning model is considered a black box where there is an input, the model, and an output without knowing specifically what the model is detecting. There are significant issues

implications. For example, a medical professional may unintentionally explain to a patient that the algorithm has a falsely high accuracy rate for diagnosis. If the accuracy rate is still high, then a reasonable person would usually still take the risk and try to get diagnosed. The issue lies in a lack of information from either the data or what the model is detecting: causing even us, the engineers, to not fully know how the model is classifying the diseases and therefore not knowing the true risks.

While we focused on the safety and the risks for patients, there are also risks to the radiologist in using a classification model due to a concern of liability. If a radiologist reviews an image and diagnoses a disease incorrectly, the radiologist would be liable for the misdiagnosis. However, the radiologist can justify how they got to the incorrect result. For a deep learning model, the justification for a diagnosis is missing regardless of the results. The black box method is a mixed blessing. People want to use the model for a high accuracy and for its ability to detect features that radiologists may miss. However, the same logic occurs for a misdiagnosis from detecting something outside of our classical definitions of the disease. There is no way of knowing if an algorithm is accidentally over factoring non-significant portions of the image like the border of an image outside of a tumor. If a model does misdiagnose a patient, without knowing the precise features and why the model concluded a classification, there is an epistemically concerning gap that radiologists need to evaluate.

From the epistemic gap between a classical definition of a disease and the model's diagnosis justification, radiologists must make ethical decisions in using the model. Ostensibly, if a model truly has a high accuracy ( $>95\%$ ), the model should be used to benefit the greater good. After all, if the model diagnoses more correctly than a radiologist could, then more people may get the correct treatment earlier and quicker. However, as mentioned in earlier paragraphs, there are more complicated factors for evaluating the usefulness of a model. Even if a model has a high accuracy, the radiologist must think about the possible patients misdiagnosed by the model. If a medical profession only had a duty for results in medical care, then there are no issues. However, medical professions also have the duty to be compassionate to their patients and respect human rights. With difficult to diagnose images, competent medical care and compassion becomes subjective. If one thinks about compassionate medical care, the justifications needed for a diagnosis overrides the model with a high accuracy without justification. For any case, and more

emphasized in misdiagnosis, the subjectivity of human diagnosis correlates more with the medical professional's duties to respect human dignity and rights. There is a conflict between a utilitarian ideal of results (high percentage of accuracy) and the justifications for diagnosis.

While we focused on the risks and safety of the stakeholders, there is also an important consideration for using medical images to train the deep learning models and the role a medical profession has in giving informed consent to a patient. Our group had used an open-source database for our images. In doing so, we avoided the procedures needed to protect patient privacy as we were not collecting mammogram images ourselves. However, there is a need to collect mammogram images to provide images with more information on labelling and diversity. In collecting these images, the patients need to be informed by the medical professionals on the use of their private images. However, as medical professionals, they may not be experts to successfully explain in detail how this process works. Thus, they may be unintentionally omitting legitimate informed consent if the patient does not properly understand how his/her data is used.

While there is informed consent for the use of medical images in training the model, this does not directly pose a risk to the patient. The patient's life is impacted by the diagnosis, and with the epistemic discontinuity between radiologist and deep learning classification. Informed consent is needed for such a decision since regardless of a radiologist's decision to use the information provided by a model, the model gives a second option. This second option may influence a radiologist's decision for better or for worse. Once again, the patient needs to gather information on how the deep learning algorithm works, and the medical professional may insufficiently explain the process due to their lack of expertise. Or, if there is a process of nudging, where either the use of a deep learning algorithm as a tool for the radiologist or the use of only the radiologist becomes a default option the patient opts out of, there is a risk of unintentionally influencing a patient's decision. By doing so, informed consent may again be omitted.