

Convolution Neural Network For Breast Cancer Image Recognition

Author: Logan R. Cole



Introduction:

Breast cancer is a problem plaguing society, in 2023 there were 1.9 million new cases of cancer in the United States alone.(Citation) Of these 1.9 million cases breast cancer was one of the most prevalent with about 30% of cancer being breast cancer. Additionally it is expected that 43,700 women will die from breast cancer in 2023 alone. One way to help prevent serious harm by cancer is early detection. This requires a large number of examinations to check to ensure that the population is cancer free. For the case of breast cancer this can be done through a mammogram. According to the American Cancer Society 1 in 8 mammograms experience a false negative. So to help with the hopefully increased volume of mammograms and to increase the accuracy a AI model should be aimed to create to help aid in the diagnosis process. This is the aim of this project. The Radiological Society of North America (RSNA) has provided patient restricted patient information and the associated mammogram to aid in the development of said model.

Ethical Considerations:

When working with healthcare it is important to always put the patient first. This includes keeping their information confidential as well as their health first. That is why when collecting information it is important to ensure that there isn't any way to link the data to the patient and only required information is provided. Additionally, when developing a model it is important that there are no false negatives. If a diagnosis is missed then the patient could go untreated and the outcome could be detrimental when compared to if a clinician had viewed the data and it had been caught early on. Furthermore, additional testing can be carried out and so a false positive can be dealt with when compared to a false negative. That is why the model shouldn't tolerate false positives especially when compared with false positives.

Data:

54,706 mammograms are provided and each mammogram provided information about what site and machine was used, the view that was used, laterality of the image. The age of each patient was provided for some images information about the density of the breast and if implants were present. Each mammogram results were provided by identifying whether cancer was present, BIRADS, whether the cancer was invasive, if a biopsy was conducted and the difficulty of

diagnosis. As for the images themselves each mammogram was provided in a dcm format. However, due to limited computation capabilities a modified 256x256 png formatted images

```
In [266]: # Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import precision_score, recall_score, accuracy_score, roc_auc_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
import os
import cv2
import glob
import shutil

import keras.backend as K
import tensorflow as tf
from tensorflow.keras.applications.resnet_v2 import ResNet50V2
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import load_model
from imblearn.over_sampling import RandomOverSampler
from imblearn.tensorflow import balanced_batch_generator
from sklearn.utils import class_weight
from collections import Counter
```

```
In [68]: df = pd.read_csv('C:\\\\Users\\\\logan\\\\Capstone\\\\train\\\\train.csv', sep=',')
print(df.shape)
df.head()
```

(54706, 14)

```
Out[68]:
```

	site_id	patient_id	image_id	laterality	view	age	cancer	biopsy	invasive	BIRADS	implant
0	2	10006	462822612	L	CC	61.0	0	0	0	NaN	
1	2	10006	1459541791	L	MLO	61.0	0	0	0	NaN	
2	2	10006	1864590858	R	MLO	61.0	0	0	0	NaN	
3	2	10006	1874946579	R	CC	61.0	0	0	0	NaN	
4	2	10011	220375232	L	CC	55.0	0	0	0	0.0	

Site_ID: A categorical variable determining location for imaging.

patient_id: Unique identifier to the patient.

Image_id: Associated identifier for the image for each patient.

laterality: Determines orientation of the image L for left R for right breast.

Age: Age of patient.

Cancer: Whether or not the patient had cancer. 0 is negative, 1 is positive.

Biopsy: 0 indicates there was not a biopsy conducted. 1 indicates that a biopsy was conducted.

Implant: Whether or not there was a breast implant. 0 for no implant and 1 if implant was

present in image.

Density: Density of the breast tissue. Categorical, presented in 4 categories:A,B,C,D.

Machine_id: Unique identifier for each machine.

Difficul_negative_case: This says whether or not there was a difficult time identifying diagnosis.

0 if diagnosis was standard and 0 if diagnosis was difficult.

BIRADS: 0 a follow up is required 1 was for negative for cancer 2 is normal

Exploratory Data Analysis:

Based on the first 5 entries it can already be seen that there are missing values which could be problematic for model development. To start checking the quality of the data we will first see how many missing values are present.

```
In [67]: df.isnull().sum(axis=0)
```

```
Out[67]: site_id          0
patient_id         0
image_id           0
laterality         0
view              0
age              37
cancer            0
biopsy            0
invasive          0
BIRADS           28420
implant           0
density          25236
machine_id        0
difficult_negative_case  0
dtype: int64
```

The BIRADS category is missing 51.95% of entries, density is missing 46.13% of entries, and age is missing 0.07% of entries. This raises concerns for classifying breast cancer based upon their BIRADS. However, all cancer labels are present and that is the primary objective of the model. Upon diagnosis the severity and how the patient should proceed can be further discussed with their doctor.

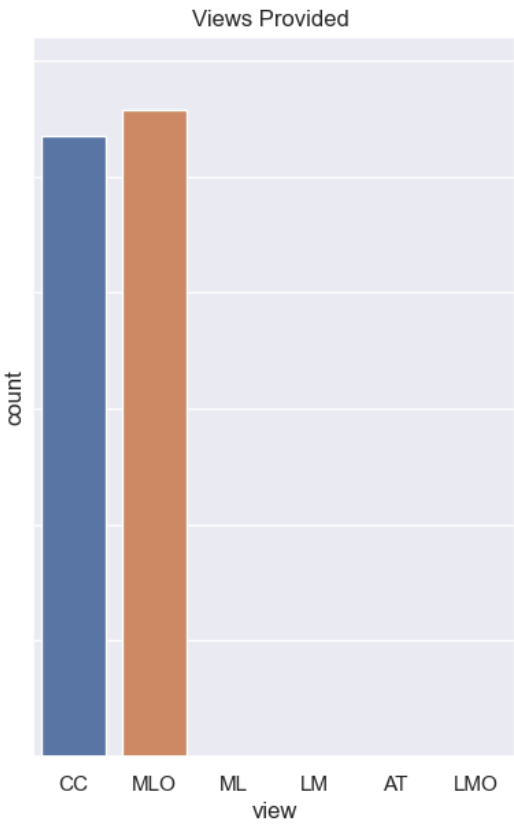
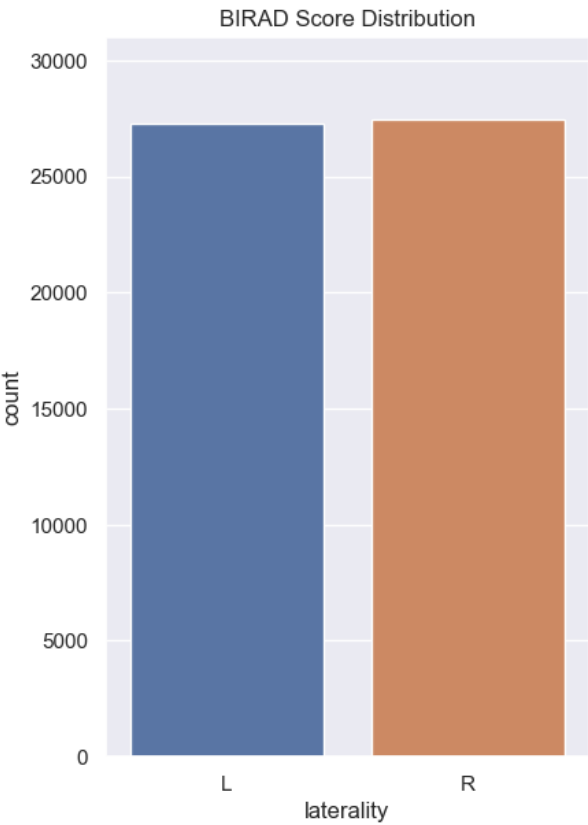
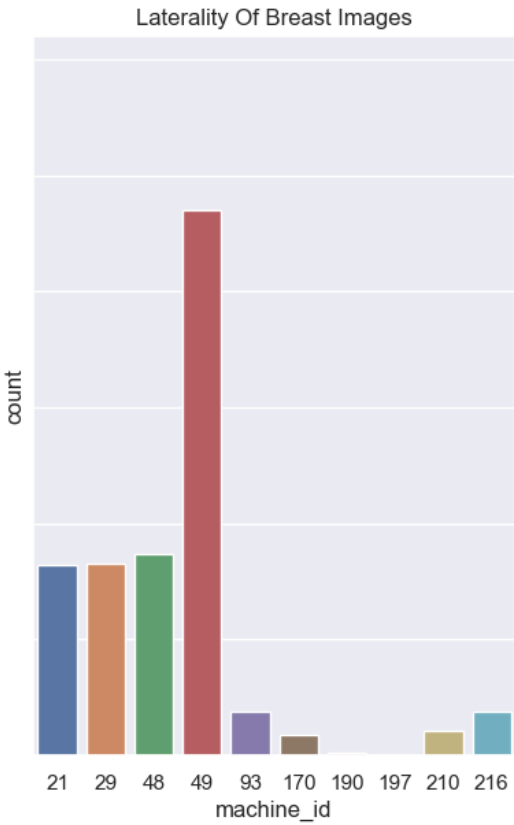
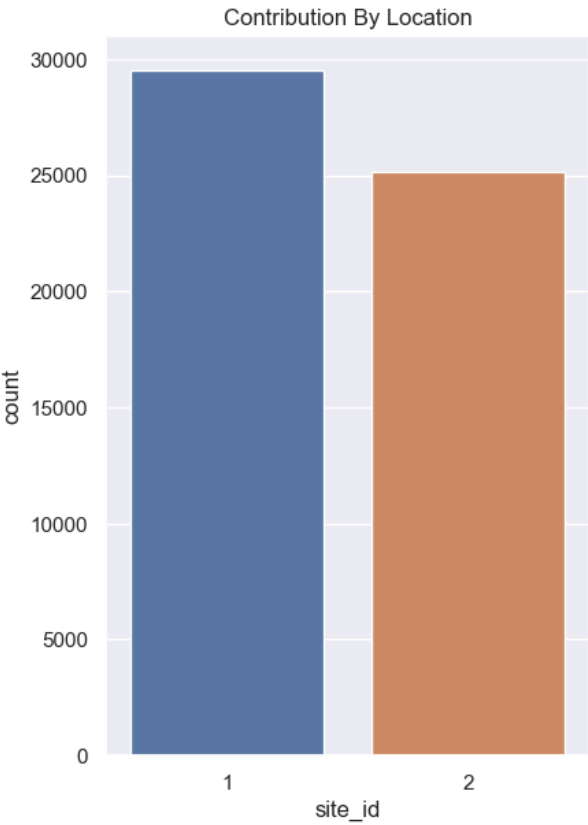
Next, the information surrounding the images themselves will be analyzed. This will consist of the number of sites providing images, number of machines used, laterality of images, and what view was used.

```
In [72]: fig,axes=plt.subplots(nrows=2, ncols=2, sharex=False, sharey=True,figsize=(10,10))
sns.set(style="darkgrid")
#Fig 1
sns.countplot(x="site_id", data=df,ax=axes[0,0]).set(title='Contribution By Location')

#Fig 2
sns.countplot(x="machine_id", data=df,ax=axes[0,1]).set(title='Laterality Of Breast')
#Fig 3
sns.countplot(x="laterality", data=df,ax=axes[1,0]).set(title='BIRAD Score Distribution')

#Fig 4
sns.countplot(x="view", data=df,ax=axes[1,1]).set(title='Views Provided ')
```

```
Out[72]: [Text(0.5, 1.0, 'Views Provided ')]
```



```
In [73]: print("Site ID:")
print(df['site_id'].value_counts(normalize=True))
print("Laterality:")
print(df['laterality'].value_counts(normalize=True))
print("View:")
print(df['view'].value_counts(normalize=True))
```

```
Site ID:
1    0.539593
2    0.460407
Name: site_id, dtype: float64
Laterality:
R    0.501572
L    0.498428
Name: laterality, dtype: float64
View:
MLO    0.510054
CC     0.489252
AT     0.000347
LM     0.000183
ML     0.000146
LMO    0.000018
Name: view, dtype: float64
```

All of the images were provided by two facilities, 54% provided by facility 1 and 46% provided by facility2. Among these 2 facilities all images were obtained from 10 machines. However, a majority came from machine 49 and most images can be contributed to 4 of the machines. As one might expect the laterality of the images provided were fairly even with approximately 50% from each direction. Lastly, there are 6 types of views provided. However, about 51% are MLO, about 49% are CC, and about .05% can be contributed to the other 4 image view's.

There are some patients who had multiple mammograms taken so we will look to see how many patients mammograms have been provided.

```
In [76]: df['patient_id'].unique().shape
```

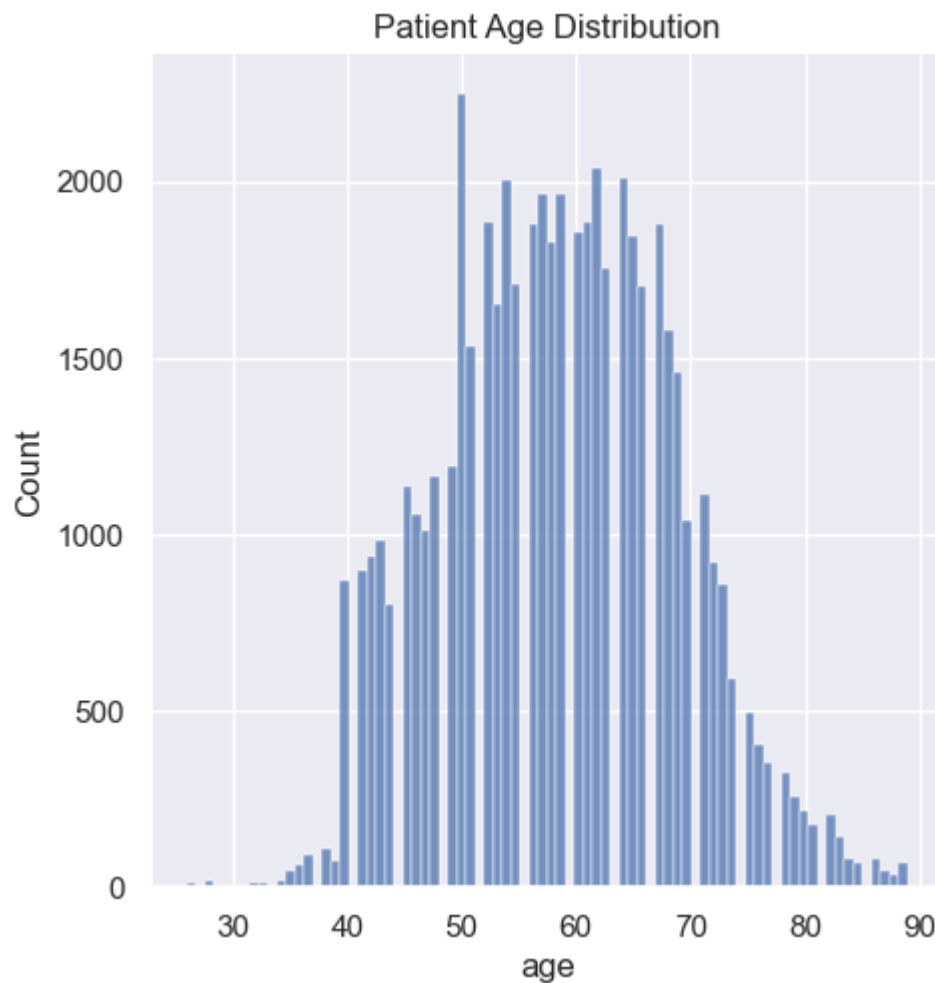
```
Out[76]: (11913,)
```

Of the 54,706 mammograms there are only 11,913. This means that on average each patient contributed about 4.6 mammograms. The multiple mammograms can occur for a couple of reasons. For some patients multiple views could be obtained to aid in the diagnosis of cancer. An image could be inconclusive but concern could be raised and additional images could be obtained to aid in diagnosis. Potentially further images could have been taken to view the state of the cancer after treatment.

Next, information surrounding the patient such as their age, breast implants, and breast density will be analyzed.

```
In [81]: sns.displot(data=df, x="age").set(title='Patient Age Distribution')
```

```
Out[81]: <seaborn.axisgrid.FacetGrid at 0x1637580fd90>
```



```
In [86]: df['age'].describe()
```

```
Out[86]: count    54669.000000
mean         58.543928
std          10.050884
min           26.000000
25%          51.000000
50%          59.000000
75%          66.000000
max           89.000000
Name: age, dtype: float64
```

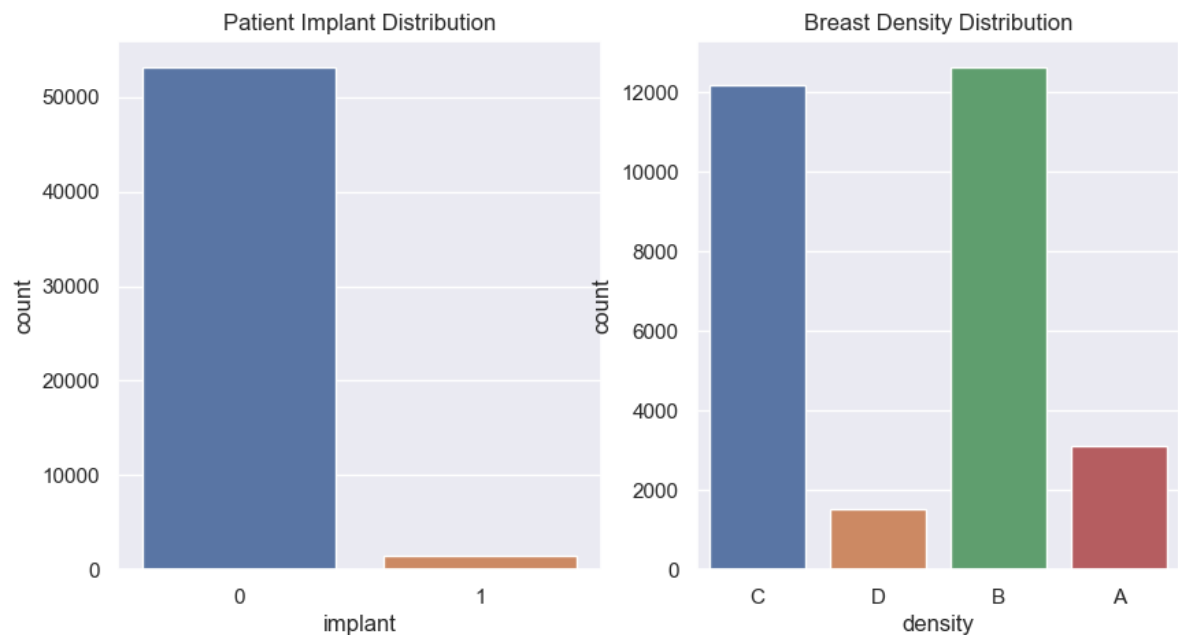
The age of the patients for the images appears to be normally distributed. The mean age was 58.5 while the median age was 59 and 50% of patients fell with the range of 66 to 51 years of age. This makes sense because as the age of a patient increases the risk of breast cancer increases. The concern and checking for health problems also increases upon the age of 50 with prostate exams typically being a frequent exam at the age of 50.

```
In [85]: fig,axes=plt.subplots(nrows=1, ncols=2,figsize=(10,5))
sns.set(style="darkgrid")

# Fig 1
sns.countplot(x="implant", data=df,ax=axes[0]).set(title='Patient Implant Distri

# Fig 2
sns.countplot(data=df, x="density",ax=axes[1]).set(title='Breast Density Distr
```

Out[85]: [Text(0.5, 1.0, 'Breast Density Distribution')]



```
In [87]: print("Implant:")
print(df['implant'].value_counts(normalize=True))
print("Density:")
print(df['density'].value_counts(normalize=True))
```

```
Implant:
0    0.973001
1    0.026999
Name: implant, dtype: float64
Density:
B    0.429284
C    0.413132
A    0.105361
D    0.052223
Name: density, dtype: float64
```

A small portion of images contained breast implants with only about 2.7% of images having implants. As for the densities of the breasts provided 43% fell into group B, 41% into group B, 10.5% into group A and only 5% into group D. This shows that a larger porportion of individuals being screened fall into the B and C category with only 15% fall into A and D.

Next, observations for what conditions are present for the images taken.

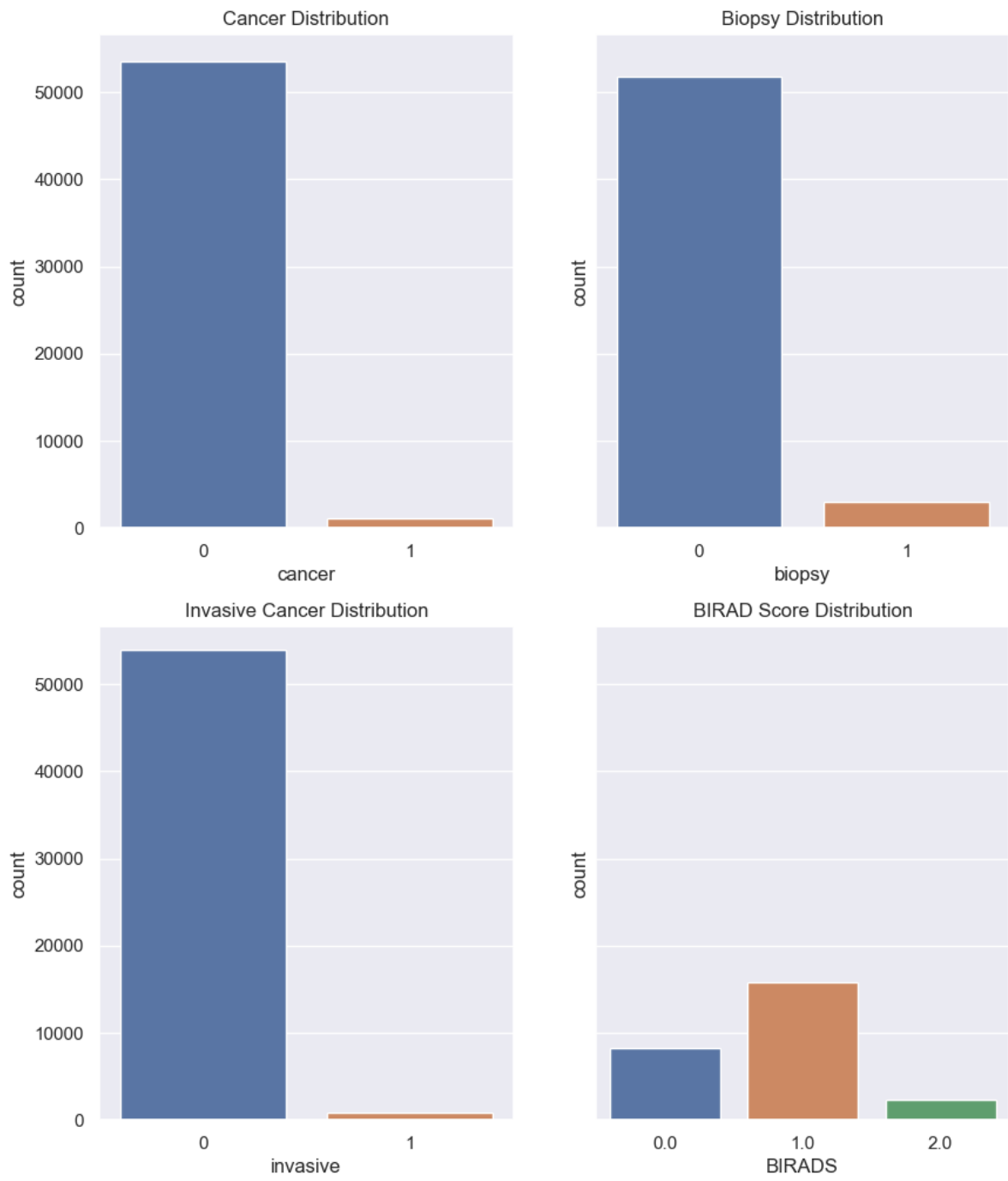

```
In [90]: fig,axes=plt.subplots(nrows=2, ncols=2, sharex=False, sharey=True,figsize=(10,10))
sns.set(style="darkgrid")
#Fig 1
sns.countplot(x="cancer", data=df,ax=axes[0,0]).set(title='Cancer Distribution')

#Fig 2
sns.countplot(x="biopsy", data=df,ax=axes[0,1]).set(title='Biopsy Distribution')

#Fig 3
sns.countplot(x="invasive", data=df,ax=axes[1,0]).set(title='Invasive Cancer Distribution')

#Fig 4
sns.countplot(x="BIRADS", data=df,ax=axes[1,1]).set(title='BIRAD Score Distribution')
```

```
Out[90]: [Text(0.5, 1.0, 'BIRAD Score Distribution')]
```



```
In [257]: print("Cancer:")
print(df['cancer'].value_counts(normalize=True))
print(df['cancer'].value_counts())
print("Biopsy:")
print(df['biopsy'].value_counts(normalize=True))
print("Invasive:")
print(df['invasive'].value_counts(normalize=True))
print("BIRADS:")
print(df['BIRADS'].value_counts(normalize=True))
```

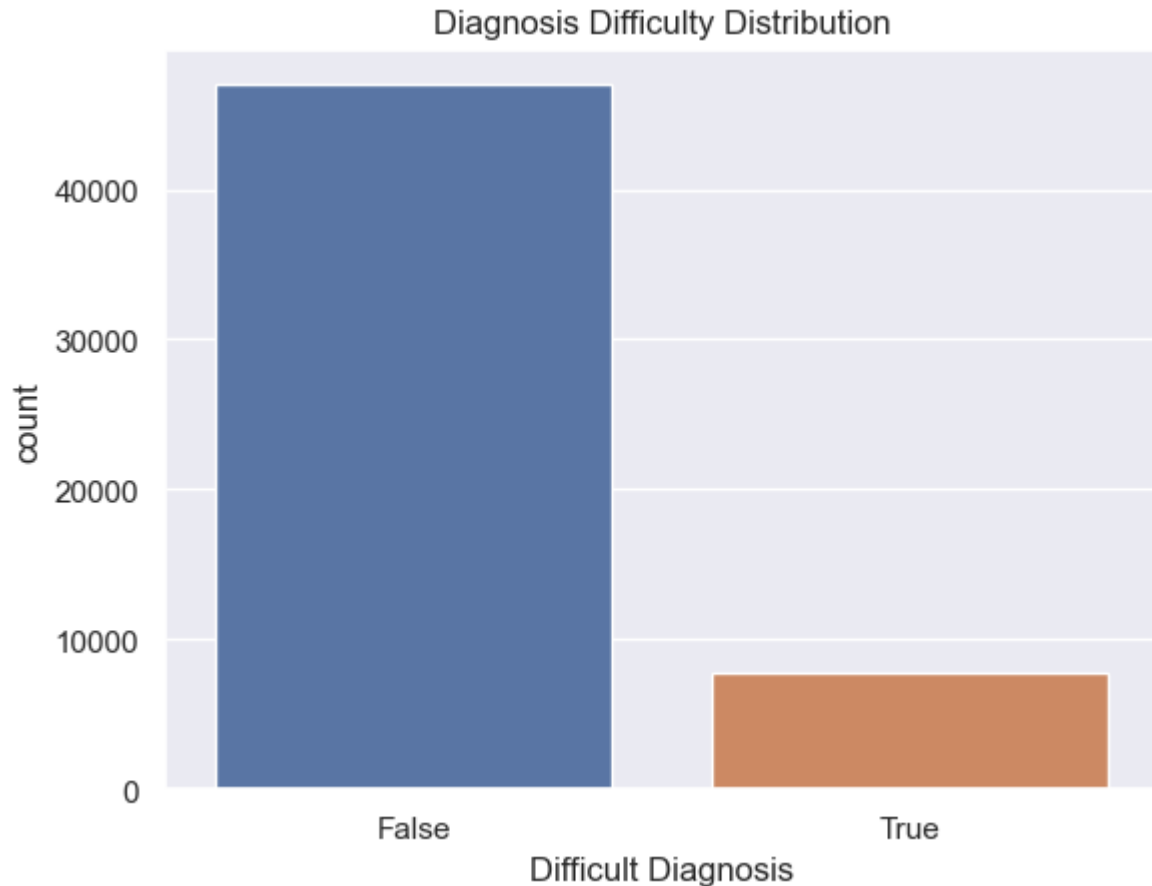
```
Cancer:
0    0.978832
1    0.021168
Name: cancer, dtype: float64
0    53548
1     1158
Name: cancer, dtype: int64
Biopsy:
0    0.945728
1    0.054272
Name: biopsy, dtype: float64
Invasive:
0    0.985047
1    0.014953
Name: invasive, dtype: float64
BIRADS:
1.0    0.600015
0.0    0.313817
2.0    0.086168
Name: BIRADS, dtype: float64
```

Of all the images taken only 2.1% of the images were recognized as cancer. This can be expected since this is a screening tool and a large portion of the population do not have breast cancer. Of all these instances 5.5% experienced a biopsy to confirm the diagnosis. Only 1.5% of patients were classified as having invasive cancer. As for the BIRADS: 31.4% were told follow ups were required, 60% were told that no cancer was present, and only 9% were told that the breast imaging looked normal and there are no abnormalities. This shows how important consistent check ups are. 91% of patients experienced some form of abnormality and something was not right. That is why it is important to consistently check and to help prevent issues from occur more frequent checkups are required which would result in a higher volume making it so that the process is in need of being streamlined.

The difficulty in diagnosis will be observed. In some instances further imaging is required and other tools must be used to obtain a diagnosis. Difficult diagnosis could pose a challenge for image modeling since at times other views, tools, or techniques must be used to obtain a affirmation on the diagnosis.

```
In [92]: sns.countplot(x="difficult_negative_case", data=df).set(xlabel='Difficult Diagn
```

```
Out[92]: [Text(0.5, 0, 'Difficult Diagnosis'),  
         Text(0.5, 1.0, 'Diagnosis Difficulty Distribution')]
```



```
In [95]: print('difficult_negative_case:')  
df['difficult_negative_case'].value_counts(normalize=True)
```

difficult_negative_case:

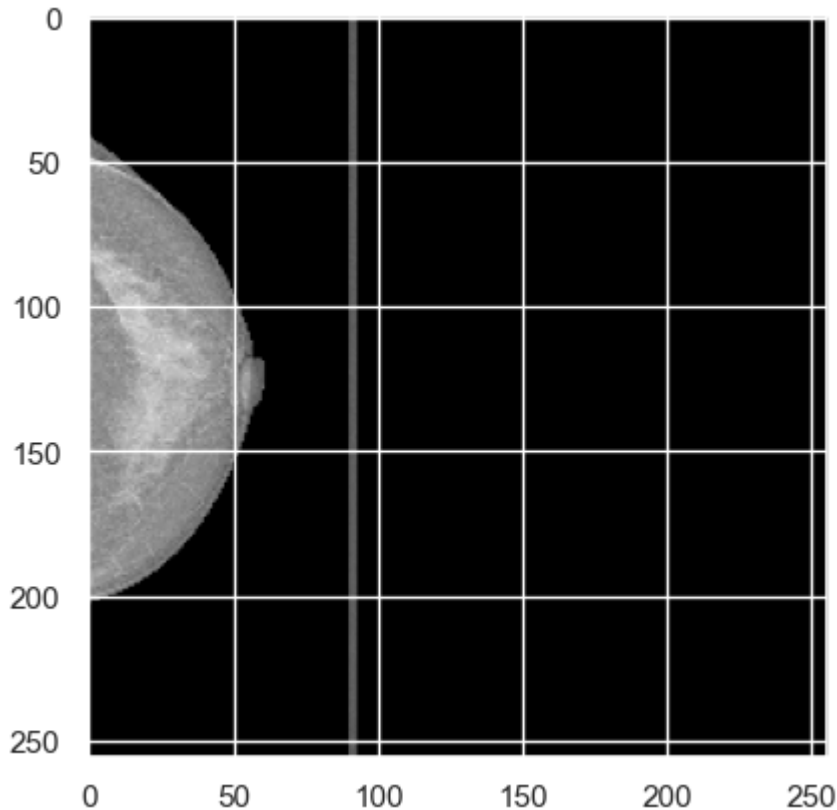
```
Out[95]: False    0.859156  
        True     0.140844  
        Name: difficult_negative_case, dtype: float64
```

14% of diagnosis were difficult to to diagnose. These 14 posed difficulties for the practitioner. However, hopefully a image recognition model trained on a large volume will be able to have an easier time identifying anomalies.

Next, the image's themselves will be analyzed.

```
In [243]: img = cv2.imread('C:\\Users\\loga\\Capstone\\BreastCancer\\10006_462822612.png')
plt.imshow(img)
print(img.shape)
```

(256, 256, 3)



Each image is in a 256,256,3 format and gray scaled. This is less detailed than the original image provided and likely will create difficulty in identifying the presence of cancer. However, as mentioned due to computation restrictions a smaller format will be used for this initial model development.

Feature Engineering:

The goal of this model is to improve the process of analyzing mammogram's. This means that the goal is to analyze the images to determine if cancer is present. This is why a convolution neural network will be used for image recognition. First, for the data frame the path to each image needs to be added. The file is named with the format "patient_id"_"image_id".png. So a for loop will be added to add the path to each image.

```
In [99]: for i in range(len(df)):
          df.loc[i, 'path'] = 'C:\\Users\\logan\\Capstone\\BreastCancer\\'+ str(df.loc[i, 'patient_id']) + str(df.loc[i, 'image_id']) + '.png'
          df.head()
```

```
Out[99]:
```

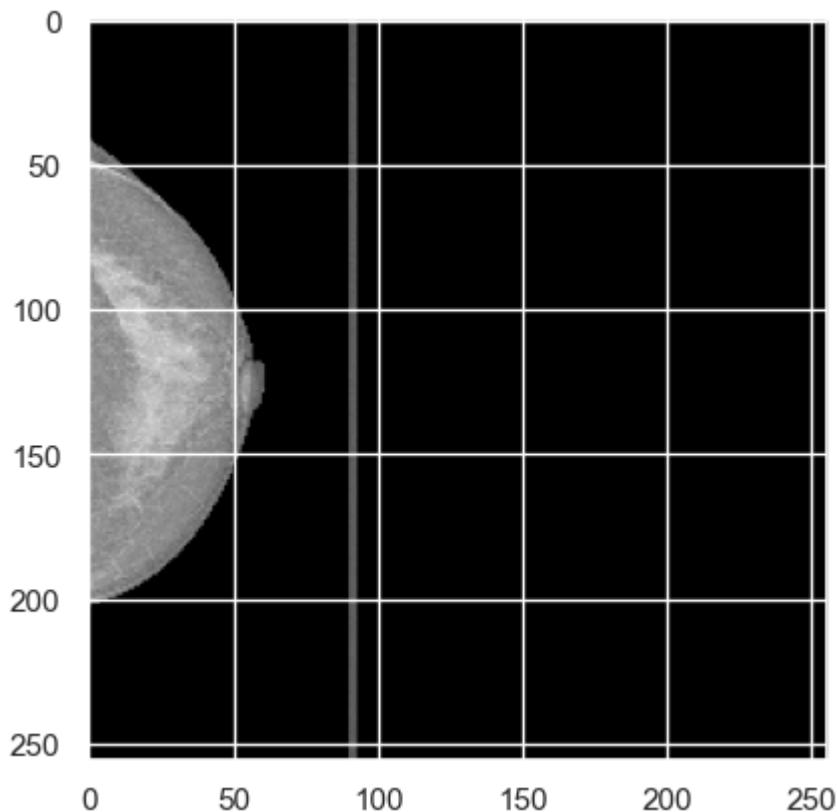
	site_id	patient_id	image_id	laterality	view	age	cancer	biopsy	invasive	BIRADS	implant
0	2	10006	462822612	L	CC	61.0	0	0	0	NaN	
1	2	10006	1459541791	L	MLO	61.0	0	0	0	NaN	
2	2	10006	1864590858	R	MLO	61.0	0	0	0	NaN	
3	2	10006	1874946579	R	CC	61.0	0	0	0	NaN	
4	2	10011	220375232	L	CC	55.0	0	0	0	0.0	

```
In [100]: path=df.loc[1, 'path']
           path
```

```
Out[100]: 'C:\\Users\\logan\\Capstone\\BreastCancer\\10006_1459541791.png'
```

```
In [101]: im = cv2.imread(path)
           plt.imshow(img, cmap = 'gray')
```

```
Out[101]: <matplotlib.image.AxesImage at 0x16310f4fbb0>
```



This illustrates that the path was correctly added for each image. Next the train and validation sets need to be created for the training of the model. Due to the imbalance of the data set it is important to stratify the two groups so that there is an equal amount of cancer in both groups.

The train test split feature from sklearn will be used to split the model into 80% training and 20%

```
In [102]: train, val = train_test_split(df, test_size = 0.2, random_state = 1, stratify = d
print("train:")
print(train.shape)
print('val:')
print(val.shape)
```

```
train:
(43764, 15)
val:
(10942, 15)
```

The CNN will be trained on the 43,764 images and the model will be validated on each iteration on 10,942 images. Next the number of cancer and normal images will be analyzed into each group.

```
In [103]: train_normal = train[train['cancer'] == 0].reset_index(drop = True)
train_cancer = train[train['cancer'] == 1].reset_index(drop = True)
val_normal = val[val['cancer'] == 0].reset_index(drop = True)
val_cancer = val[val['cancer'] == 1].reset_index(drop = True)
print(train_normal.shape)
print(train_cancer.shape)
print(val_normal.shape)
print(val_cancer.shape)
```

```
(42838, 15)
(926, 15)
(10710, 15)
(232, 15)
```

This shines a light on one of the primary problems with the current selections of images. Since the images are only provided from two locations and only 1,159 cancer images are present the large class imbalance will make it difficult to train a model that can accurately distinguish between the two groups.

Now to prepare the images for the tensorflow CNN the images in each group need to have their own folder so flow from directory can work and the model can access and recognize each group. To accomplish this shutil copy will copy the images into their respective folders.

```
In [ ]: for path in train_normal['path']:
        shutil.copy2(path, 'C:\\Users\\logan\\Capstone\\CNN\\train\\normal')
for path in train_cancer['path']:
        shutil.copy2(path, 'C:\\Users\\logan\\Capstone\\CNN\\train\\cancer')
for path in val_normal['path']:
        shutil.copy2(path, 'C:\\Users\\logan\\Capstone\\CNN\\val\\normal')
for path in val_cancer['path']:
        shutil.copy2(path, 'C:\\Users\\logan\\Capstone\\CNN\\val\\cancer')
```

Now that the images have been sorted all the data has been prepared for the model to be developed.

Model Training:

The ResNet50V2 base will be used for the CNN which will provide the base layers for the CNN. Other notable features is that a dropout layer with 0.4 will be used to help prevent overfitting of the data and smoothe out the learning. Once all the steps have been carried out the final layer will be a sigmoid activation to create a probability of it falling into one of each groups. This will allow for the boundry of cutoff to be modified which could additionally help with the imbalance in the data and the fact that there is a more sever punishment for false negative than a false positive. While training the model accuracy and AUC will be measured. Accuracy is important since the goal is to streamline the process by filtering through mammogram images. However, due to the class imbalance and the reduced tolerance for false negatives the AUC must also be analyzed.

```
In [123]: model_b = ResNet50V2(weights = 'imagenet', input_shape = (256, 256, 3), include_top=False)
for layer in model_b.layers:
    layer.trainable = False

model = Sequential()
model.add(model_b)
model.add(GlobalAveragePooling2D())
model.add(Dense(128, activation = 'relu'))
model.add(Dropout(0.4))
model.add(Dense(1, activation = 'sigmoid'))

model.compile(optimizer = "adam", loss = 'binary_crossentropy', metrics = ["accuracy"])
```

```
In [124]: model.summary()
```

Model: "sequential_4"

Layer (type)	Output Shape	Param #
=====		
resnet50v2 (Functional)	(None, 8, 8, 2048)	23564800
global_average_pooling2d_4 (GlobalAveragePooling2D)	(None, 2048)	0
dense_8 (Dense)	(None, 128)	262272
dropout_4 (Dropout)	(None, 128)	0
dense_9 (Dense)	(None, 1)	129
=====		
Total params: 23,827,201		
Trainable params: 262,401		
Non-trainable params: 23,564,800		


```
In [104]: train_datagen = ImageDataGenerator(rescale = 1./255.)
val_datagen = ImageDataGenerator(rescale = 1./255.)
train_path = 'C:\\Users\\logan\\Capstone\\CNN\\train'
val_path = 'C:\\Users\\logan\\Capstone\\CNN\\val'
train_generator = train_datagen.flow_from_directory(
    train_path,
    target_size = (256, 256),
    batch_size = 20,
    class_mode = 'binary'
)

validation_generator = val_datagen.flow_from_directory(
    val_path,
    target_size = (256, 256),
    batch_size = 20,
    class_mode = 'binary'
)
```

Found 43764 images belonging to 2 classes.
 Found 10942 images belonging to 2 classes.

Now that the CNN framework has been set and the train generator and validation generator have been set up the model is ready for training. However, because of the class imbalance one more feature must be added to the CNN. To help with the class imbalance weights will be added to penalize the miss labeling a cancer image than a normal image. This also fall's into line with the fact that greater tolerance is present for false positives than false negatives. To compute the class weights module will be used from sklearn.

```
In [105]: class_weights=class_weight.compute_class_weight('balanced',classes=np.unique(t
weights=dict(zip(np.unique(train_generator.classes),class_weights))
weights
```

```
Out[105]: {0: 23.630669546436284, 1: 0.5108081609785704}
```

These class weights were too heavy and resulted in the model accuracy greatly reducing and too many images were being labeled as cancer. To hopefully help smoothe the sever correction the weight for cancer will be dropped to 15 from 23.6. Now, that the weights and the CNN framework has been setup it is now time for the model to be trained.

```
In [122]: weight={0:15,1:0.5}
```

```
In [125]: callback = tf.keras.callbacks.EarlyStopping(monitor = "val_auc", mode = "max",  
history = model.fit(train_generator, validation_data = validation_generator, s
```

Epoch 1/25

30/30 [=====] - 394s 13s/step - loss: 1.5341 - accuracy: 0.6183 - auc: 0.4106 - val_loss: 0.4764 - val_accuracy: 0.7612 - val_auc: 0.5599

Epoch 2/25

30/30 [=====] - 392s 13s/step - loss: 0.9820 - accuracy: 0.6800 - auc: 0.5229 - val_loss: 0.1824 - val_accuracy: 0.9560 - val_auc: 0.5189

Epoch 3/25

30/30 [=====] - 397s 14s/step - loss: 0.6424 - accuracy: 0.6800 - auc: 0.4658 - val_loss: 0.2281 - val_accuracy: 0.9461 - val_auc: 0.5465

Epoch 4/25

30/30 [=====] - 400s 14s/step - loss: 1.2509 - accuracy: 0.5233 - auc: 0.4560 - val_loss: 0.5543 - val_accuracy: 0.7148 - val_auc: 0.5138

Epoch 5/25

30/30 [=====] - 401s 14s/step - loss: 0.6415 - accuracy: 0.8267 - auc: 0.4717 - val_loss: 0.3298 - val_accuracy: 0.9273 - val_auc: 0.5441

Epoch 6/25

30/30 [=====] - 398s 14s/step - loss: 0.5534 - accuracy: 0.6767 - auc: 0.6594 - val_loss: 0.2714 - val_accuracy: 0.9418 - val_auc: 0.5419

Epoch 7/25

30/30 [=====] - 400s 14s/step - loss: 0.9347 - accuracy: 0.5800 - auc: 0.5414 - val_loss: 0.3944 - val_accuracy: 0.8986 - val_auc: 0.5182

Epoch 8/25

30/30 [=====] - 399s 14s/step - loss: 0.7475 - accuracy: 0.7350 - auc: 0.5379 - val_loss: 0.6360 - val_accuracy: 0.5998 - val_auc: 0.5477

Epoch 9/25

30/30 [=====] - 398s 14s/step - loss: 0.7961 - accuracy: 0.7317 - auc: 0.4313 - val_loss: 0.5086 - val_accuracy: 0.8144 - val_auc: 0.5547

Epoch 10/25

30/30 [=====] - 399s 14s/step - loss: 0.4700 - accuracy: 0.7033 - auc: 0.5754 - val_loss: 0.4063 - val_accuracy: 0.9693 - val_auc: 0.5642

Epoch 11/25

30/30 [=====] - 400s 14s/step - loss: 0.5734 - accuracy: 0.7683 - auc: 0.5172 - val_loss: 0.5978 - val_accuracy: 0.7444 - val_auc: 0.5350

Epoch 12/25

30/30 [=====] - 400s 14s/step - loss: 0.5675 - accuracy: 0.6900 - auc: 0.4281 - val_loss: 0.6343 - val_accuracy: 0.6816 - val_auc: 0.5440

Epoch 13/25

30/30 [=====] - 398s 14s/step - loss: 0.6434 - accuracy: 0.6783 - auc: 0.6399 - val_loss: 0.5816 - val_accuracy: 0.6920 - val_auc: 0.5715

Epoch 14/25

30/30 [=====] - 399s 14s/step - loss: 0.6039 - accuracy: 0.6900 - auc: 0.4368 - val_loss: 0.4253 - val_accuracy: 0.8562 - val_auc: 0.5161

Epoch 15/25

```
30/30 [=====] - 399s 14s/step - loss: 0.4493 - accur
acy: 0.8583 - auc: 0.5951 - val_loss: 0.2216 - val_accuracy: 0.9765 - val_au
c: 0.5157
Epoch 16/25
30/30 [=====] - 400s 14s/step - loss: 0.6488 - accur
acy: 0.7233 - auc: 0.5291 - val_loss: 0.5185 - val_accuracy: 0.7620 - val_au
c: 0.5334
Epoch 17/25
30/30 [=====] - 400s 14s/step - loss: 0.3647 - accur
acy: 0.6933 - auc: 0.6841 - val_loss: 0.2572 - val_accuracy: 0.9772 - val_au
c: 0.5465
Epoch 18/25
30/30 [=====] - 402s 14s/step - loss: 0.6378 - accur
acy: 0.8750 - auc: 0.4779 - val_loss: 0.5724 - val_accuracy: 0.7294 - val_au
c: 0.5265
Epoch 19/25
30/30 [=====] - 402s 14s/step - loss: 0.6106 - accur
acy: 0.5967 - auc: 0.6288 - val_loss: 0.5108 - val_accuracy: 0.8519 - val_au
c: 0.5542
Epoch 20/25
30/30 [=====] - 407s 14s/step - loss: 0.6132 - accur
acy: 0.8283 - auc: 0.5098 - val_loss: 0.4165 - val_accuracy: 0.9466 - val_au
c: 0.5467
Epoch 21/25
30/30 [=====] - 401s 14s/step - loss: 0.6567 - accur
acy: 0.4533 - auc: 0.5795 - val_loss: 0.7544 - val_accuracy: 0.2445 - val_au
c: 0.5453
Epoch 22/25
30/30 [=====] - 403s 14s/step - loss: 0.5700 - accur
acy: 0.3733 - auc: 0.4432 - val_loss: 0.6562 - val_accuracy: 0.5622 - val_au
c: 0.5558
Epoch 23/25
30/30 [=====] - 404s 14s/step - loss: 0.5810 - accur
acy: 0.4283 - auc: 0.5318 - val_loss: 0.6758 - val_accuracy: 0.3985 - val_au
c: 0.5672
Epoch 24/25
30/30 [=====] - 401s 14s/step - loss: 0.5925 - accur
acy: 0.5083 - auc: 0.6122 - val_loss: 0.6405 - val_accuracy: 0.6312 - val_au
c: 0.5795
Epoch 25/25
30/30 [=====] - 402s 14s/step - loss: 0.5471 - accur
acy: 0.7833 - auc: 0.4515 - val_loss: 0.5483 - val_accuracy: 0.9751 - val_au
c: 0.5784
```

```
In [126]: model.save('model1.h5')
```

```
In [127]: accuracy = history.history['accuracy']  
          val_accuracy = history.history['val_accuracy']  
  
          loss = history.history['loss']  
          val_loss = history.history['val_loss']  
  
          auc = history.history['auc']  
          val_auc = history.history['val_auc']
```

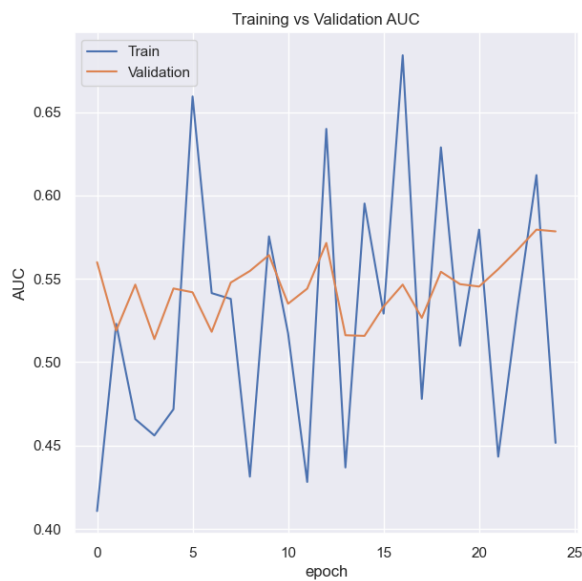
```
In [128]: plt.figure(figsize = (15,15))

plt.subplot(2, 2, 1)
plt.plot(accuracy, label = "Training Accuracy")
plt.plot(val_accuracy, label = "Validation Accuracy")
plt.ylim(0.4, 1)
plt.legend(['Train', 'Validation'], loc = 'upper left')
plt.title("Training vs Validation Accuracy")
plt.xlabel('epoch')
plt.ylabel('accuracy')

plt.subplot(2, 2, 2)
plt.plot(loss, label = "Training Loss")
plt.plot(val_loss, label = "Validation Loss")
plt.legend(['Train', 'Validation'], loc = 'upper left')
plt.title("Training vs Validation Loss")
plt.xlabel('epoch')
plt.ylabel('loss')

plt.subplot(2, 2, 3)
plt.plot(auc, label = "Training AUC")
plt.plot(val_auc, label = "Validation AUC")
plt.legend(['Train', 'Validation'], loc = 'upper left')
plt.title("Training vs Validation AUC")
plt.xlabel('epoch')
plt.ylabel('AUC')
```

```
Out[128]: Text(0, 0.5, 'AUC')
```



It appears that the accuracy and AUC were very volatile for training of the model having spikes between each epoch. Since without the class weights the model would classify every image as normal it appears that the weights did help penalize and encourage the adam optimizer to prioritize appropriate cancer diagnosis. However, ultimately it appeared that the model struggled to learn how to classify between the two groups. The volatility in the performance shows the difficulty in training and learning from the images to differentiate between the two classes.

Model Evaluation:

Now to start analysis of the model on the validation group will be used to determine how it performs when being used on the non training set.

```
In [129]: model = load_model('C:\\Users\\loga\\Capstone\\model1.h5')
```

```
In [130]: performance= model.predict(validation_generator)

548/548 [=====] - 358s 652ms/step
```

```
In [131]: performance
```

```
Out[131]: array([[0.5743191 ],
                 [0.6072306 ],
                 [0.64492106],
                 ...,
                 [0.64897203],
                 [0.5730711 ],
                 [0.5567812 ]], dtype=float32)
```

Now that the probabilities of being normal have been calculated the cutoff needs to be set to label each group. The standard for binary classification is 0.5 which will be the baseline. However, this can be changed to help accomodate for the class imbalance.

```
In [229]: y_pred = []
          for prob in performance:
              if prob >= 0.5:
                  y_pred.append(1)
              else:
                  y_pred.append(0)
```

```
In [230]: pd.Series(y_pred).value_counts()
```

```
Out[230]: 1    10804
          0      138
          dtype: int64
```

```
In [231]: y_true = validation_generator.classes
```

```
In [232]: cm = confusion_matrix(y_true, y_pred)
```

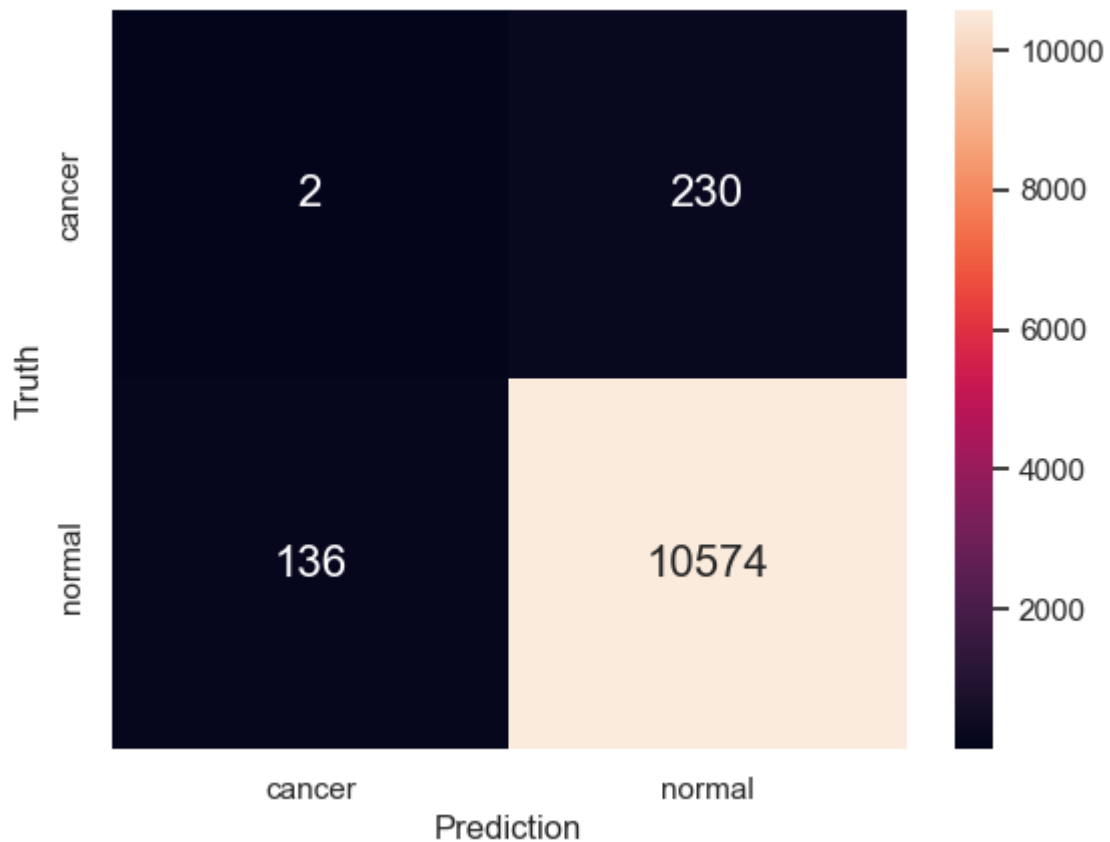


```
In [233]: class_names = ['cancer', 'normal']

# Create the heatmap with class names as tick labels.
ax = sns.heatmap(cm, annot = True, fmt = '.0f', annot_kws = {"size": 16},\
                 xticklabels = class_names, yticklabels = class_names)

# Set the axis labels.
ax.set_xlabel("Prediction")
ax.set_ylabel("Truth")
```

Out[233]: Text(47.25, 0.5, 'Truth')



```
In [239]: print(classification_report(y_true, y_pred))
```

	precision	recall	f1-score	support
0	0.01	0.01	0.01	232
1	0.98	0.99	0.98	10710
accuracy			0.97	10942
macro avg	0.50	0.50	0.50	10942
weighted avg	0.96	0.97	0.96	10942

As illustrated by the confusion matrix the model did a poor job in accurately predicting cancer. Of the 232 instances of cancer only 2 were identified which is less than 1% of cancer patients. Meanwhile, 136 false positives were present which is more than the amount of true positives by

116 fold. Changing the boundary could help improve the performance of the model since if it can

```
In [258]: y_pred = []
          for prob in performance:
              if prob >= 0.55:
                  y_pred.append(1)
              else:
                  y_pred.append(0)
```

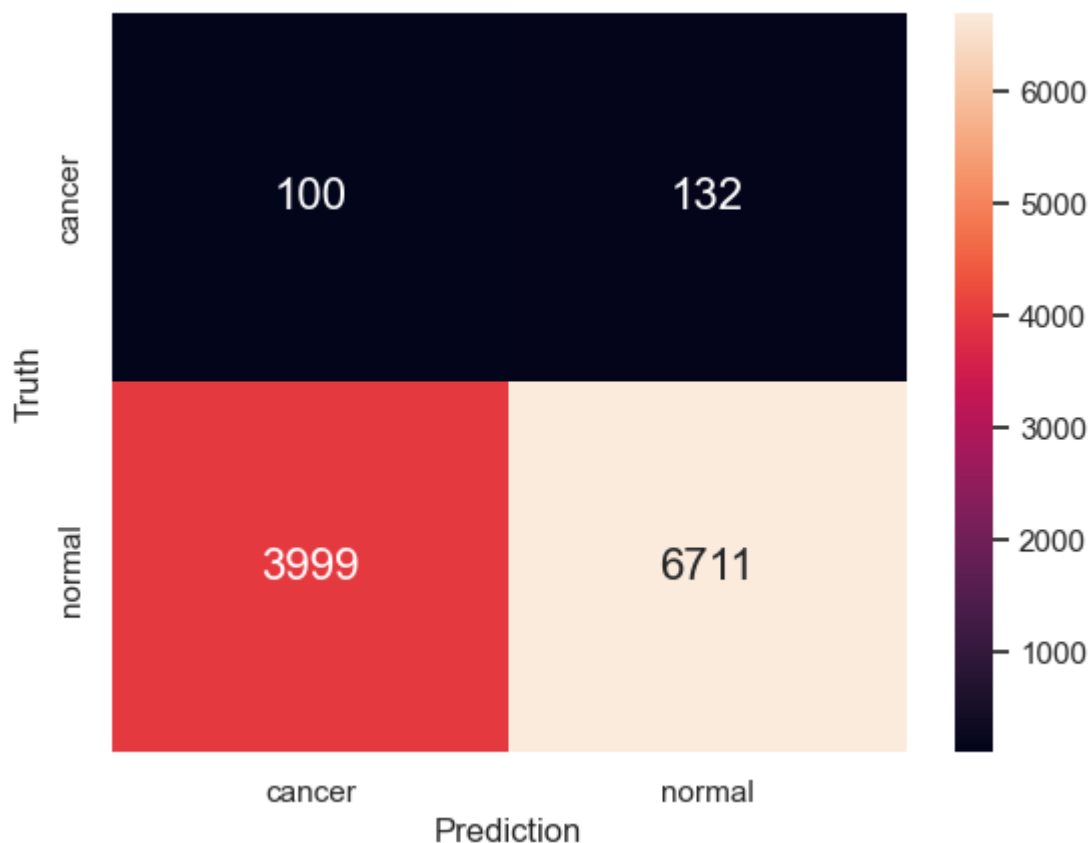
```
In [261]: cm = confusion_matrix(y_true, y_pred)
```

```
In [262]: class_names = ['cancer', 'normal']

          # Create the heatmap with class names as tick labels.
          ax = sns.heatmap(cm, annot = True, fmt = '.0f', annot_kws = {"size": 16},\
                           xticklabels = class_names, yticklabels = class_names)

          # Set the axis labels.
          ax.set_xlabel("Prediction")
          ax.set_ylabel("Truth")
```

```
Out[262]: Text(47.25, 0.5, 'Truth')
```



While the number of true positives went from 2 to 100 showing a 42% improvement and now 43% of cancer instances were correctly determined the number of false positives went up 29 fold. Potentially modifying the weights one more time slightly penalizing the miss cancer labels could help with crossing the distance.

Model 2:


The same CNN will be used. However, the weight will be increased from 15 to 17 which should hopefully allow for a slightly better performance without the large spike in false positives.

```
In [139]: model_2 = ResNet50V2(weights = 'imagenet', input_shape = (256, 256, 3), include_top=True)
for layer in model_2.layers:
    layer.trainable = False

model2 = Sequential()
model2.add(model_2)
model2.add(GlobalAveragePooling2D())
model2.add(Dense(128, activation = 'relu'))
model2.add(Dropout(0.4))
model2.add(Dense(1, activation = 'sigmoid'))

model2.compile(optimizer = "adam", loss = 'binary_crossentropy', metrics = ['accuracy'])
weight2={0:17,1:0.5}
```

```
In [140]: callback = tf.keras.callbacks.EarlyStopping(monitor = "val_auc", mode = "max",  
history = model2.fit(train_generator, validation_data = validation_generator, ...
```



```
Epoch 1/25
30/30 [=====] - 382s 13s/step - loss: 2.1119 - accuracy: 0.9833 - auc: 0.5291 - val_loss: 0.1215 - val_accuracy: 0.9788 - val_auc: 0.5256
Epoch 2/25
30/30 [=====] - 384s 13s/step - loss: 0.5960 - accuracy: 0.9617 - auc: 0.5875 - val_loss: 0.3075 - val_accuracy: 0.9788 - val_auc: 0.5522
Epoch 3/25
30/30 [=====] - 396s 14s/step - loss: 0.7364 - accuracy: 0.8883 - auc: 0.5083 - val_loss: 0.5601 - val_accuracy: 0.8938 - val_auc: 0.5654
Epoch 4/25
30/30 [=====] - 392s 13s/step - loss: 0.5872 - accuracy: 0.7550 - auc: 0.5780 - val_loss: 0.5666 - val_accuracy: 0.9455 - val_auc: 0.5590
Epoch 5/25
30/30 [=====] - 392s 13s/step - loss: 0.5717 - accuracy: 0.8383 - auc: 0.5717 - val_loss: 0.4767 - val_accuracy: 0.9788 - val_auc: 0.5512
Epoch 6/25
30/30 [=====] - 391s 13s/step - loss: 0.8515 - accuracy: 0.7850 - auc: 0.5517 - val_loss: 0.6436 - val_accuracy: 0.7374 - val_auc: 0.5828
Epoch 7/25
30/30 [=====] - 398s 14s/step - loss: 0.5084 - accuracy: 0.5683 - auc: 0.6342 - val_loss: 0.6048 - val_accuracy: 0.9442 - val_auc: 0.5776
Epoch 8/25
30/30 [=====] - 399s 14s/step - loss: 0.6785 - accuracy: 0.7550 - auc: 0.5026 - val_loss: 0.5963 - val_accuracy: 0.8929 - val_auc: 0.5580
Epoch 9/25
30/30 [=====] - 404s 14s/step - loss: 0.4899 - accuracy: 0.7367 - auc: 0.3614 - val_loss: 0.4492 - val_accuracy: 0.9769 - val_auc: 0.5140
Epoch 10/25
30/30 [=====] - 409s 14s/step - loss: 0.6123 - accuracy: 0.7617 - auc: 0.4915 - val_loss: 0.6184 - val_accuracy: 0.7935 - val_auc: 0.5028
Epoch 11/25
30/30 [=====] - 402s 14s/step - loss: 0.5437 - accuracy: 0.7072 - auc: 0.5205 - val_loss: 0.5614 - val_accuracy: 0.9642 - val_auc: 0.4953
Epoch 12/25
30/30 [=====] - 396s 14s/step - loss: 0.5801 - accuracy: 0.7850 - auc: 0.6032 - val_loss: 0.5580 - val_accuracy: 0.9642 - val_auc: 0.5328
Epoch 13/25
30/30 [=====] - 408s 14s/step - loss: 0.5090 - accuracy: 0.8400 - auc: 0.6227 - val_loss: 0.4516 - val_accuracy: 0.9733 - val_auc: 0.5017
Epoch 14/25
30/30 [=====] - 391s 13s/step - loss: 0.6249 - accuracy: 0.8417 - auc: 0.5588 - val_loss: 0.4917 - val_accuracy: 0.9788 - val_auc: 0.5600
Epoch 15/25
```

```
30/30 [=====] - 406s 14s/step - loss: 0.4755 - accur
acy: 0.9033 - auc: 0.6272 - val_loss: 0.3139 - val_accuracy: 0.9788 - val_au
c: 0.5564
Epoch 16/25
30/30 [=====] - 401s 14s/step - loss: 0.6696 - accur
acy: 0.7900 - auc: 0.6528 - val_loss: 0.5018 - val_accuracy: 0.9787 - val_au
c: 0.5630
Epoch 17/25
30/30 [=====] - 402s 14s/step - loss: 0.7645 - accur
acy: 0.6600 - auc: 0.4163 - val_loss: 0.6713 - val_accuracy: 0.5080 - val_au
c: 0.5441
Epoch 18/25
30/30 [=====] - 403s 14s/step - loss: 0.5662 - accur
acy: 0.5750 - auc: 0.5062 - val_loss: 0.5604 - val_accuracy: 0.9408 - val_au
c: 0.5440
Epoch 19/25
30/30 [=====] - 403s 14s/step - loss: 0.5835 - accur
acy: 0.6183 - auc: 0.4769 - val_loss: 0.5759 - val_accuracy: 0.9641 - val_au
c: 0.5397
Epoch 20/25
30/30 [=====] - 406s 14s/step - loss: 0.5935 - accur
acy: 0.6967 - auc: 0.4645 - val_loss: 0.5544 - val_accuracy: 0.9758 - val_au
c: 0.5145
Epoch 21/25
30/30 [=====] - 401s 14s/step - loss: 0.6698 - accur
acy: 0.6283 - auc: 0.4635 - val_loss: 0.6776 - val_accuracy: 0.3877 - val_au
c: 0.5691
Epoch 22/25
30/30 [=====] - 405s 14s/step - loss: 0.5739 - accur
acy: 0.5050 - auc: 0.4338 - val_loss: 0.6219 - val_accuracy: 0.6763 - val_au
c: 0.5694
Epoch 23/25
30/30 [=====] - 401s 14s/step - loss: 0.5050 - accur
acy: 0.6517 - auc: 0.4995 - val_loss: 0.5154 - val_accuracy: 0.9378 - val_au
c: 0.5778
Epoch 24/25
30/30 [=====] - 406s 14s/step - loss: 0.6020 - accur
acy: 0.6117 - auc: 0.6142 - val_loss: 0.5427 - val_accuracy: 0.8687 - val_au
c: 0.5796
Epoch 25/25
30/30 [=====] - 404s 14s/step - loss: 0.5114 - accur
acy: 0.7200 - auc: 0.6111 - val_loss: 0.4650 - val_accuracy: 0.9584 - val_au
c: 0.5841
```

```
In [145]: accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

auc = history.history['auc']
val_auc = history.history['val_auc']
```

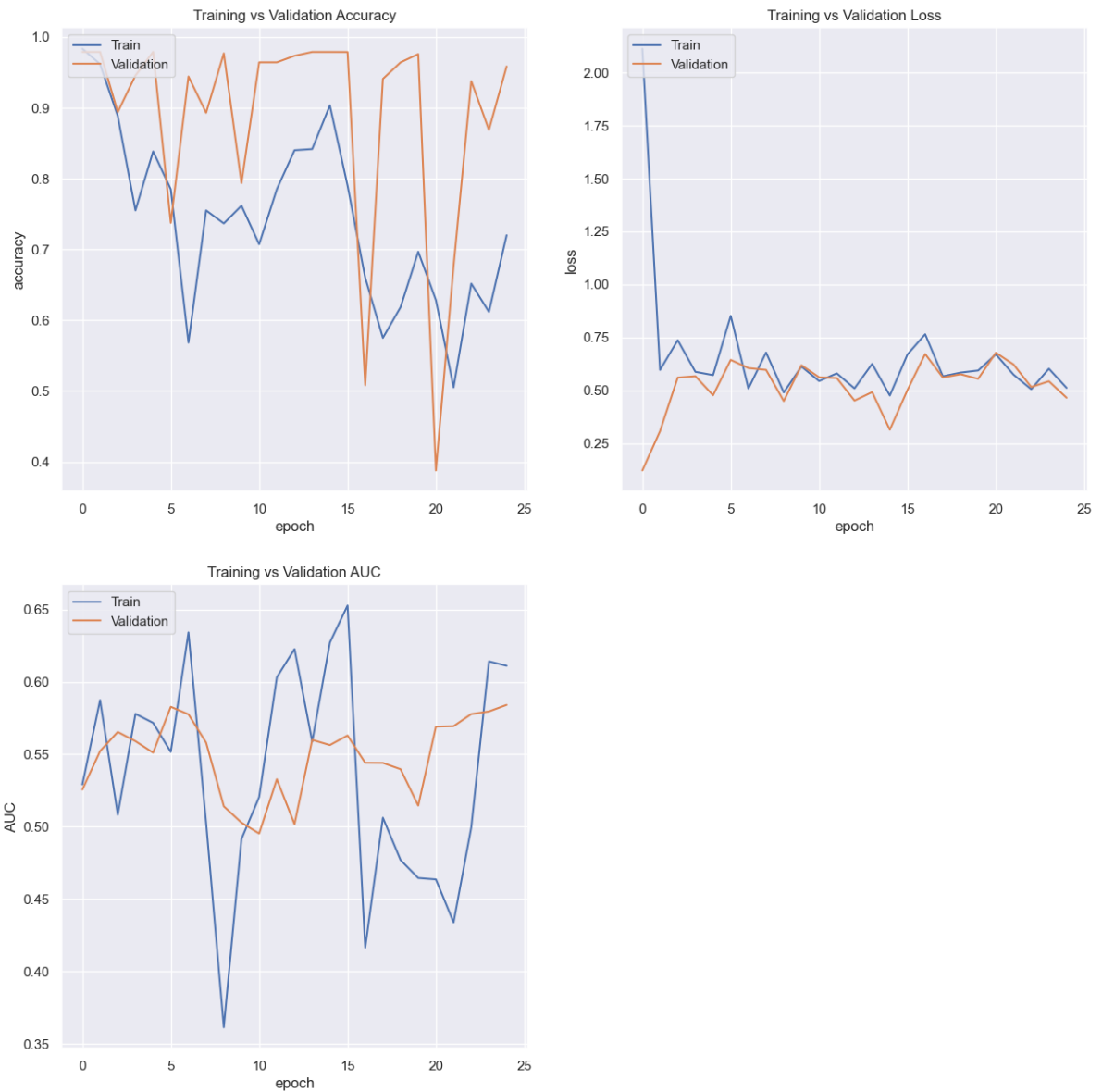
```
In [146]: plt.figure(figsize = (15,15))

plt.subplot(2, 2, 1)
plt.plot(accuracy, label = "Training Accuracy")
plt.plot(val_accuracy, label = "Validation Accuracy")
plt.legend(['Train', 'Validation'], loc = 'upper left')
plt.title("Training vs Validation Accuracy")
plt.xlabel('epoch')
plt.ylabel('accuracy')

plt.subplot(2, 2, 2)
plt.plot(loss, label = "Training Loss")
plt.plot(val_loss, label = "Validation Loss")
plt.legend(['Train', 'Validation'], loc = 'upper left')
plt.title("Training vs Validation Loss")
plt.xlabel('epoch')
plt.ylabel('loss')

plt.subplot(2, 2, 3)
plt.plot(auc, label = "Training AUC")
plt.plot(val_auc, label = "Validation AUC")
plt.legend(['Train', 'Validation'], loc = 'upper left')
plt.title("Training vs Validation AUC")
plt.xlabel('epoch')
plt.ylabel('AUC')
```

```
Out[146]: Text(0, 0.5, 'AUC')
```



This training experienced slightly less validity. However, the changes and dip in accuracy is a concern. After the first couple of iterations the loss was near its minimum and it appears that overfitting could have occurred. However, the validation AUC did improve overtime while the accuracy maintained high exeriencing less fluctuations showing an improvment in the model training. So to see if it ultimately did improve the performance will be measured on the validation set.

```
In [141]: model.save('model2.h5')
```

```
In [142]: model2 = load_model('C:\\Users\\logan\\Capstone\\model1.h5')
```

```
In [143]: performance2= model.predict(validation_generator)
```

```
548/548 [=====] - 376s 685ms/step
```

Similary the 0.5 boundary will be used to see how well it can destinguish between the two without changing the boundary to favor the cancer diagnosis.


```
In [234]: y_pred2= []
          for prob in performance2:
              if prob >= 0.5:
                  y_pred2.append(1)
              else:
                  y_pred2.append(0)
```

```
In [235]: pd.Series(y_pred2).value_counts()
```

```
Out[235]: 1    10804
          0      138
          dtype: int64
```

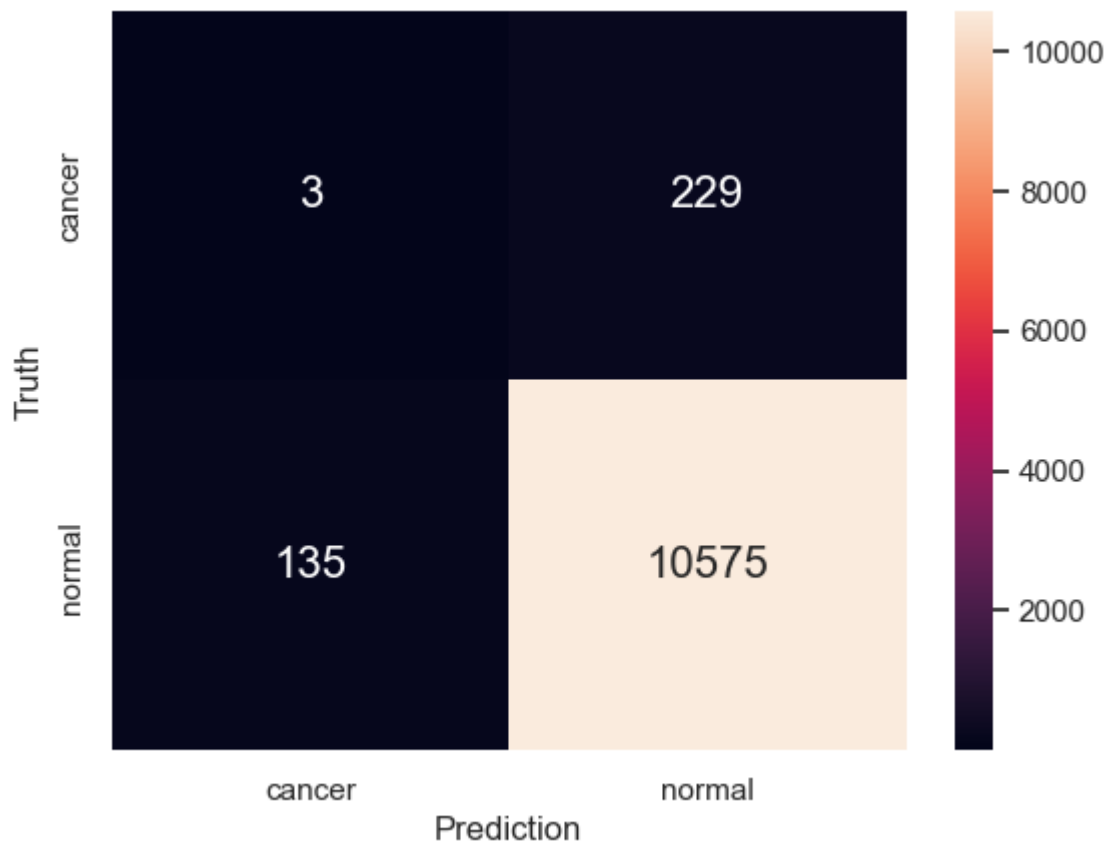
```
In [236]: cm2 = confusion_matrix(y_true, y_pred2)
```

```
In [237]: class_names = ['cancer', 'normal']

          # Create the heatmap with class names as tick labels.
          ax = sns.heatmap(cm2, annot = True, fmt = '.0f', annot_kws = {"size": 16},\
                           xticklabels = class_names, yticklabels = class_names)

          # Set the axis labels.
          ax.set_xlabel("Prediction")
          ax.set_ylabel("Truth")
```

```
Out[237]: Text(47.25, 0.5, 'Truth')
```



```
In [240]: print(classification_report(y_true, y_pred2))
```

	precision	recall	f1-score	support
0	0.02	0.01	0.02	232
1	0.98	0.99	0.98	10710
accuracy			0.97	10942
macro avg	0.50	0.50	0.50	10942
weighted avg	0.96	0.97	0.96	10942

Once again the model performed almost identically to the initial model with the only difference being one more correct cancer diagnosis. This shows potentially that the main issue arises do to the images and the training in itself. However, once again the performance while changing the boundary to 0.55 will be analyzed.

```
In [263]: y_pred2= []
for prob in performance2:
    if prob >= 0.55:
        y_pred2.append(1)
    else:
        y_pred2.append(0)
```

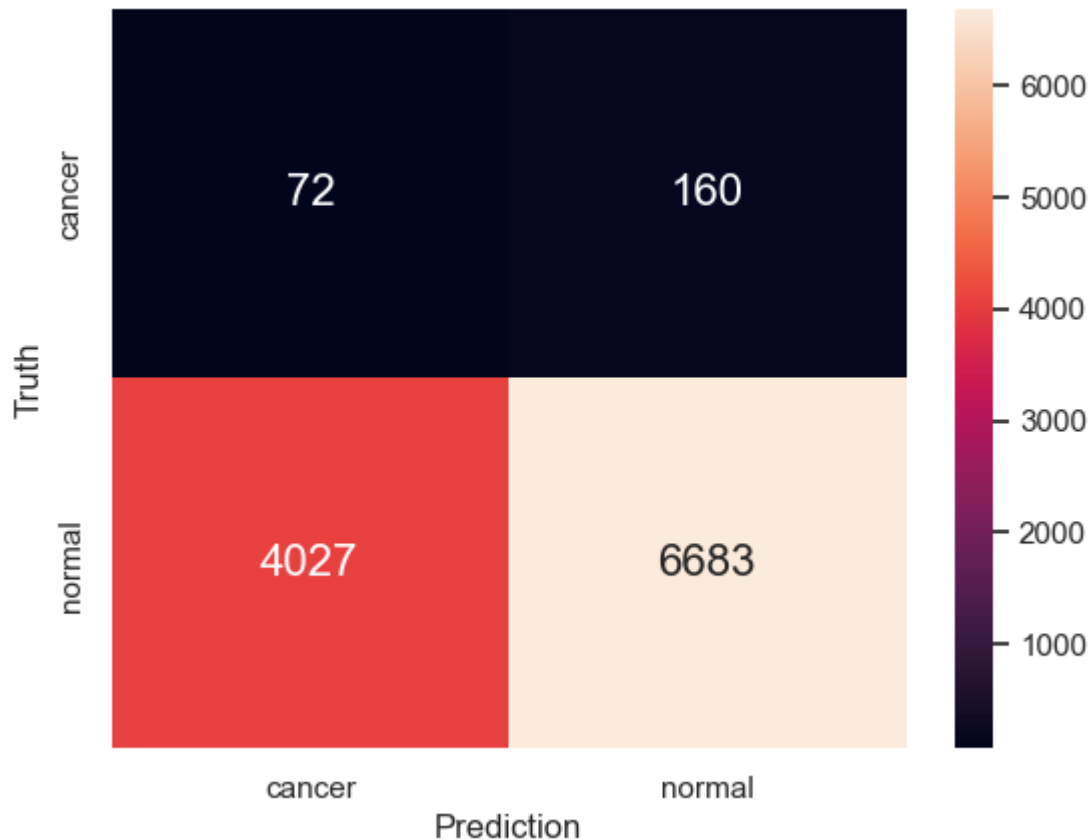
```
In [264]: cm2 = confusion_matrix(y_true, y_pred2)
```

```
In [265]: class_names = ['cancer', 'normal']

ax = sns.heatmap(cm2, annot = True, fmt = '.0f', annot_kws = {"size": 16},\
                xticklabels = class_names, yticklabels = class_names)

ax.set_xlabel("Prediction")
ax.set_ylabel("Truth")
```

```
Out[265]: Text(47.25, 0.5, 'Truth')
```



With this new boundary the model actually performed worse with more false positives and less true positives. This shows that the model does not have significant prediction power and the loss of information gain from the dcm to png file lost significant predicting power. Additionally the lack of cancer images and the class imbalance presented was too much a challenge for this model to overcome.

Conclusion:

While streamlining the mammogram process is feasible this model was unable to accomplish said task. One problem lies in the class imbalance cause by the lack of cancer images. The images came from two facilities and with the contribution from more facilities in the U.S. or internationally would help speed up the process of obtaining a sufficient number of images for the task. However, more notably this helps to illustrate the computational complexity of the problem at hand. The image quality is important for the process of image recognition and with the larger dcm file and neural net with more training capabilities a significant jump in computing power is required. Going forward in the near future the model should be trained with the initial

dcm files to hopefully see a noticeable improvement in the predicting power. However, I would expect that this would not be enough for reaching the point of streamlining the process and it is important for these sites to release the images and similar information for the training of said model. If this is done I would expect that a much more accurate model could be developed to not only streamline the process but perform better than medical professionals who have been noted to have a 85% accuracy.

References:

"Breast Cancer Statistics: How Common Is Breast Cancer?" Breast Cancer Statistics | How Common Is Breast Cancer?, [https://www.cancer.org/cancer/breast-cancer/about/how-common-is-breast-cancer.html#:~:text=It%20is%20about%2030%25%20\(or,\(DCIS\)%20will%20be%20diagnosed](https://www.cancer.org/cancer/breast-cancer/about/how-common-is-breast-cancer.html#:~:text=It%20is%20about%2030%25%20(or,(DCIS)%20will%20be%20diagnosed) ([https://www.cancer.org/cancer/breast-cancer/about/how-common-is-breast-cancer.html#:~:text=It%20is%20about%2030%25%20\(or,\(DCIS\)%20will%20be%20diagnosed](https://www.cancer.org/cancer/breast-cancer/about/how-common-is-breast-cancer.html#:~:text=It%20is%20about%2030%25%20(or,(DCIS)%20will%20be%20diagnosed)).

Chollet, François. Deep Learning with Python. Manning Publications Co. LLC, 2021.

"Limitations of Mammograms." How Accurate Are Mammograms?, <https://www.cancer.org/cancer/breast-cancer/screening-tests-and-early-detection/mammograms/limitations-of-mammograms.html#:~:text=About%20half%20of%20the%20women,t%20available%20for%20co> (<https://www.cancer.org/cancer/breast-cancer/screening-tests-and-early-detection/mammograms/limitations-of-mammograms.html#:~:text=About%20half%20of%20the%20women,t%20available%20for%20co>

McDowell, Sandy. "2023 Cancer Facts & Figures Cancer: Incidence Drops for Cervical Cancer but Rises for Prostate Cancer." 2023 Cancer Facts & Figures Cancer | Incidence Drops for Cervical Cancer But Rises for Prostate Cancer, 12 Jan. 2023, <https://www.cancer.org/latest-news/facts-and-figures-2023.html#:~:text=In%20the%20US%20in%202023,about%201%2C670%20deaths%20a%20day> (<https://www.cancer.org/latest-news/facts-and-figures-2023.html#:~:text=In%20the%20US%20in%202023,about%201%2C670%20deaths%20a%20day>

"RSNA Screening Mammography Breast Cancer Detection." Kaggle, <https://www.kaggle.com/competitions/rsna-breast-cancer-detection/overview> (<https://www.kaggle.com/competitions/rsna-breast-cancer-detection/overview>).

Viel, Theo. "RSNA Breast Cancer Detection - 256X256 PNGS." Kaggle, 29 Nov. 2022, <https://www.kaggle.com/datasets/theoviel/rsna-breast-cancer-256-pngs> (<https://www.kaggle.com/datasets/theoviel/rsna-breast-cancer-256-pngs>).

