Student Name: Ibraheem Ziad Alkhuzami
Student ID: 1535386
Email: i.5ozami@gmail.com

“Invasive Ductal Carcinoma Prediction with Convolutional Neural Networks”

ABSTRACT

Cancer tumors can be classified into two types: malignant and benign. A benign tumor is one which does not invade its surrounding tissues whereas a malignant tumor is one which may spread to its surrounding tissues or other parts of the body.

In this paper we applied Convolutional Neural Networks to classify the Breast Cancer Image Dataset. This report investigates the Breast Cancer Dataset provided by Kaggle through Deep Learning approach using CNN Model as classifier. The results show that this CNN Model scores an accuracy of 80% roughly.

INTRODUCTION

There is no doubt that Cancer is one of the most dangerous disease. According to statistics there are millions dying every year because of cancer despite the the trillions of dollars spent on the treatment. It has been always a trendy problem for every era. So we decided to apply Deep Learning Approaches like Convolutional Neural Networks to investigate how Deep Learning Algorithms/Approaches may contribute to the society and how accurate it could be about classifying one of the most dangerous and dreadful disease worldwide.

So since we are using Convolutional Neural Networks our Dataset is expected to be an image Dataset. We had to do some research to find a good Dataset for us to investigate and do our analysis on it. We found that Kaggle has a large Dataset that is 1.6 GB. It is 277524 images collected from 279 patients.

Our input is an image that is of 3 channels; RGB Images that we resized to images of height of 50 pixels, width of 50 pixels and we kept the depth the same as it is 3 channels instead of converting it to gray scale images. We just decreased the images size a bit to

Student Name: Ibraheem Ziad Alkhuzami
Student ID: 1535386
Email: i.5ozami@gmail.com

ensure that all images are of same size and another reason that is avoiding memory errors that might occur while training our CNN model due to the large size of the images regardless of the training runtime.

Now that we discussed and explained the input to our model, we need to explain what the expected output is, but first we will first explain the structure of our CNN model.

The model is a sequential which allows us to create the model layer-by-layer. The architecture consists of Convolutional layers, max pooling layers, dropout layers and fully connected layers.

The first layer is a Convolutional layer with 32 filters each of size 3 x 3. We are also required to specify the input shape in the first layer, which is 50 x 50 x 3 in our case. We will be using the Rectified linear unit (ReLU) activation function for all the layers except the final output layer. ReLU is the most common choice for activation function in the hidden layers and has shown to work pretty well.

The second layer is a pooling layer. The pooling layers are used to reduce dimension. Max Pooling with a 2x2 window only considers the maximum value in the 2x2 window.

The third layer is again a Convolutional layer of 64 filters each of size 3 x 3 followed by another max pooling layer of 2x2 window. Usually, the number of filters in the Convolutional layer grows after each layer. The first layers with lower number of filter learns simple features of the images whereas the deeper layers learn more complex features.

The next two layers are again Convolutional layers with the same filter size but increasing number of filters; 128 and 256.

We need to flatten the 3D feature map output from the Convolutional layer to 1D feature vectors before adding in the fully connected layers. This is where the flatten layer comes in.

The next layer is a dropout layer with a dropout rate of 0.5 . A dropout layer with dropout rate of 0.5 means 50% of the neurons will be turned off randomly. This helps prevent overfitting by making all the neurons learn something about the data and not rely on just a few neurons. Randomly dropping neurons during training means other

Student Name: Ibraheem Ziad Alkhuzami
Student ID: 1535386
Email: i.5ozami@gmail.com

neurons will have to do the work of the turned-off neurons, thus generalizing better and prevent overfitting.

The value 0.5 is taken from the original paper by Hinton (2012), which has proved to be very effective. These dropout layers are added after each of the fully connected layers before the output. Dropout also reduces the training time for each epoch.

The following dense layer (fully connected layer) has 128 neurons. This is followed by another dropout layer with a dropout rate of 0.5

The next layer is another dense layer with 128 neurons.

The final output layer is another dense layer which has number of neurons equal to the number of classes. The activation function in this layer is sigmoid because the problem in hand is a binary classification problem. For multi-class classification problem, the activation function should be set to softmax.

The model is compiled with binary cross entropy loss function and the Adam optimizer is used. The ‘accuracy’ metric is used to evaluate the model.

Adam is an optimization algorithm which updates the network weights in an iterative manner.

Although the initial learning rate for Adam can be set (we have set it to 0.00001 in our case), this is the initial learning rate and the learning rate for each parameter is adapted as training begins. This is how Adam (short for adaptive moment estimation) is different from stochastic gradient descent, which maintains a single learning rate for all weight updates.

The learning rate determines how fast we are adjusting the weights of our network towards the local minima. Too high of a learning rate can result in such high weight changes that it might result in overshooting the local minima. This causes the training or validation error to fluctuate drastically between consecutive epochs. Too low of a learning rate can result in taking longer time to train our network. Thus, the learning rate is one of the most important hyperparameters that needs to be tuned when building the model.

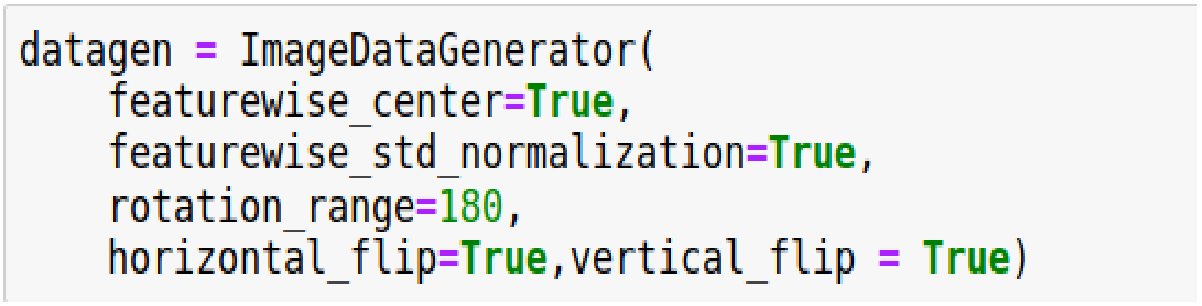
Student Name: Ibraheem Ziad Alkhuzami
Student ID: 1535386
Email: i.5ozami@gmail.com

DATA SET AND FEATURES

Our Dataset is provided from Kaggle which is a large Dataset that is 1.6 GB. It is 277524 images collected from 279 patients.
We resized our image Dataset to be of shape (50,50,3) as we explained why in the introduction section.

Generally, the more data we have, the better deep learning tends to work. Keras ImageDataGenerator generates real time images during training using data augmentation. Transformations are performed on the mini-batches on-the-fly. Data augmentation helps generalize the model by reducing the network's capacity to overfit the training data. Rotation, vertical and horizontal flipping are some of the common data augmentation techniques used.

Keras ImageDataGenerator provides a variety of data augmentation techniques. However, we only used few of them.

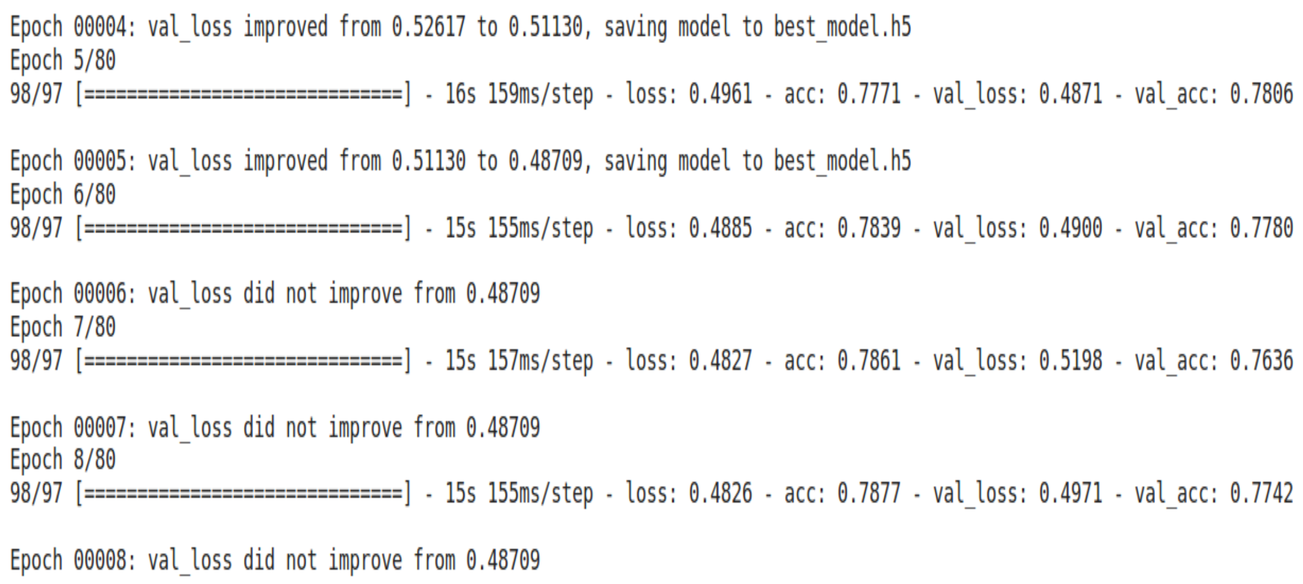


```
datagen = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True,
    rotation_range=180,
    horizontal_flip=True,vertical_flip = True)
```

Student Name: Ibraheem Ziad Alkhuzami
Student ID: 1535386
Email: i.5ozami@gmail.com

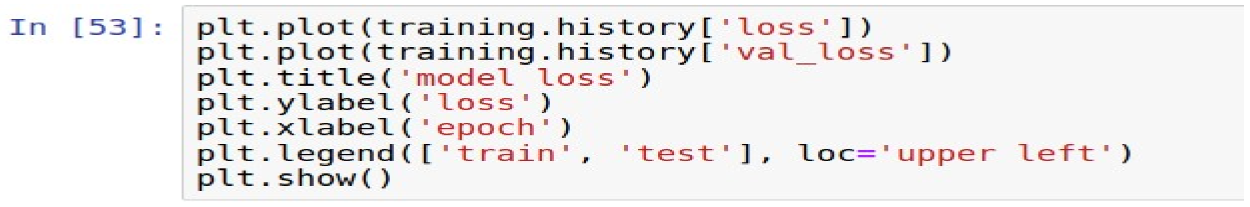
EXPERIMENTS

Training stops after 8 epochs due to Early Stopping. Hence, the best model saved is the one during epoch 5, with a validation accuracy of 78.06%

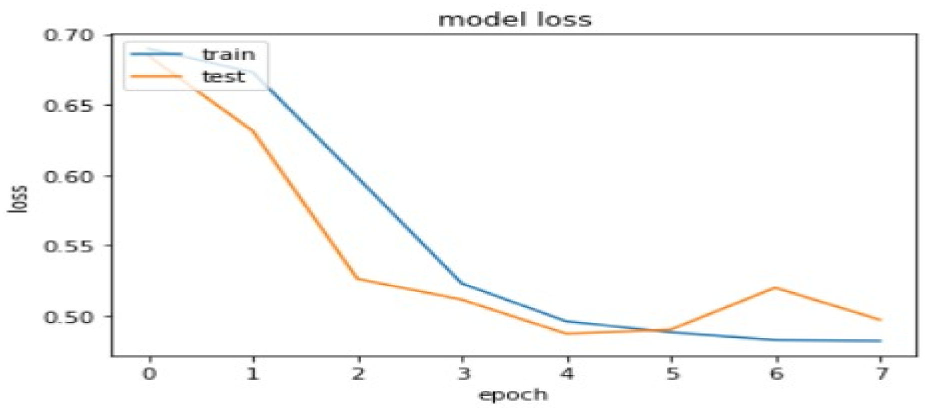


```
Epoch 00004: val_loss improved from 0.52617 to 0.51130, saving model to best_model.h5
Epoch 5/80
98/97 [==============================] - 16s 159ms/step - loss: 0.4961 - acc: 0.7771 - val_loss: 0.4871 - val_acc: 0.7806

Epoch 00005: val_loss improved from 0.51130 to 0.48709, saving model to best_model.h5
Epoch 6/80
98/97 [==============================] - 15s 155ms/step - loss: 0.4885 - acc: 0.7839 - val_loss: 0.4900 - val_acc: 0.7780

Epoch 00006: val_loss did not improve from 0.48709
Epoch 7/80
98/97 [==============================] - 15s 157ms/step - loss: 0.4827 - acc: 0.7861 - val_loss: 0.5198 - val_acc: 0.7636

Epoch 00007: val_loss did not improve from 0.48709
Epoch 8/80
98/97 [==============================] - 15s 155ms/step - loss: 0.4826 - acc: 0.7877 - val_loss: 0.4971 - val_acc: 0.7742

Epoch 00008: val_loss did not improve from 0.48709
```

Plotting the training set and validation set loss, we find that the variance is very low. This plot ensures that our model is not overfitting.



```
In [53]: plt.plot(training.history['loss'])
         plt.plot(training.history['val_loss'])
         plt.title('model loss')
         plt.ylabel('loss')
         plt.xlabel('epoch')
         plt.legend(['train', 'test'], loc='upper left')
         plt.show()
```



Student Name: Ibraheem Ziad Alkhuzami
Student ID: 1535386
Email: i.5ozami@gmail.com

CONCLUSION / FUTURE WORK

In conclusion to our experiments with the Convolutional Neural Networks we achieved an validation accuracy of nearly 80%.
In future work we will try to improve this accuracy and make it reach +90% and deploy our model to a web application for doctors to help them in the diagnosis process.