Capstone Project Mahmoud Zaher
Machine Learning Engineer Nanodegree November 23, 2019

Definition

Project Overview

Breast cancer is one of the most common cancers that affect women's it's ranked the first in women and the second in both mens and womens [1]. In 1971 United States President Richard Nixon announces war against cancer from that time and until now billion of dollars were gone to find solution to it [2] , but with no solution but that helped us a lot to understand it .
My project aims to help doctors to find solution for cancer using some of images called microscopic or histology images. my goal is to predict from the images supplied to my model which is 400 images 100 per each category if image was benign , normal , in situ or invasive [3] . which will help doctors a lot and reduce their time and maybe with some improvements get higher accuracy rather than them by seeing details that they can't . This project is part of grand challenge [link].

Problem Statement

The problem is that microscopic images is difficult for doctors to understand rather than mammography which is very common type of x-ray in breast cancer , so what I tried to do is to build a model that can understand shape of cancer and try to see the details that we can't .the steps would be like the following :
1.load our data and split it to train and test
2. we will try to make some data preprocessing that include image resizing and choosing the right filter .
3. We will put our train data after preprocessing to the CNN to try to find the pattern of each type of the four categories .
4. Try to optimize our model and increase its accuracy .
Our goal is to supply our model of microscopic image that we don't know it's categorie and we hope our model classify it true .

---

[1] According to the International Agency for Research on Cancer (IARC) .
[2] According to wikipedia https://en.wikipedia.org/wiki/War_on_Cancer .
[3] link of challenge https://iciar2018-challenge.grand-challenge.org/ .
[4] part A in breast cancer histology images challenge .

Metrics

One of the most simple and good metrics is accuracy , but to use accuracy with image classification your data should be balanced that mean you have the same number of pictures per each category to get true value to accuracy .

$$Accuracy = \frac{true\ positive + true\ negative}{dataset\ size}$$

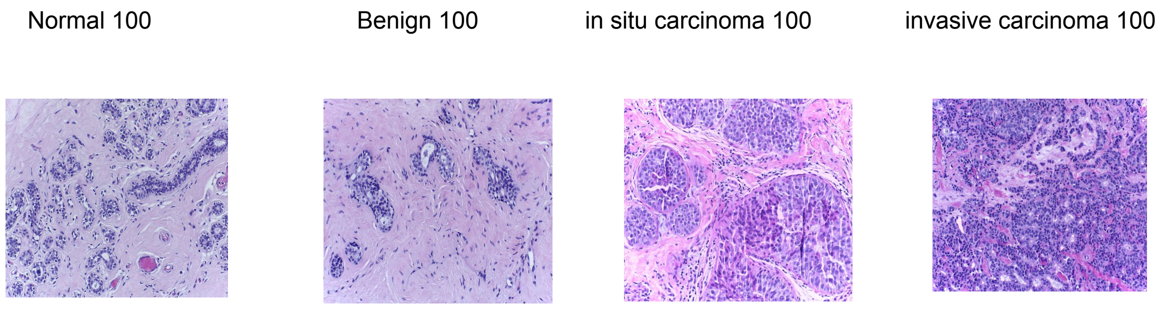
Fortunately our data is balanced according to data supplied by our challenge we have 400 image 100 normal , 100 benign , 100 in suit and 100 invasive so that condition of each category have same number is satisfied so that we can use accuracy as metric
Without any problems .

Analysis

Data Exploration

The dataset consists of microscopy images available in ``.tiff ” format. which are labelled as (normal, benign, in situ carcinoma or invasive carcinoma) according to the predominant cancer type in each image. The annotation was performed by two medical experts and images where there was disagreement were discarded,The dataset contains a total of 400 .
images are on tiff format and have the following specifications:

- Color model: R(ed)G(reen)B(lue)
- Size: 2048 x 1536 pixels
- Pixel scale: 0.42 µm x 0.42 µm
- Memory space: 10-20 MB (approx.)
- Type of label: image-wise

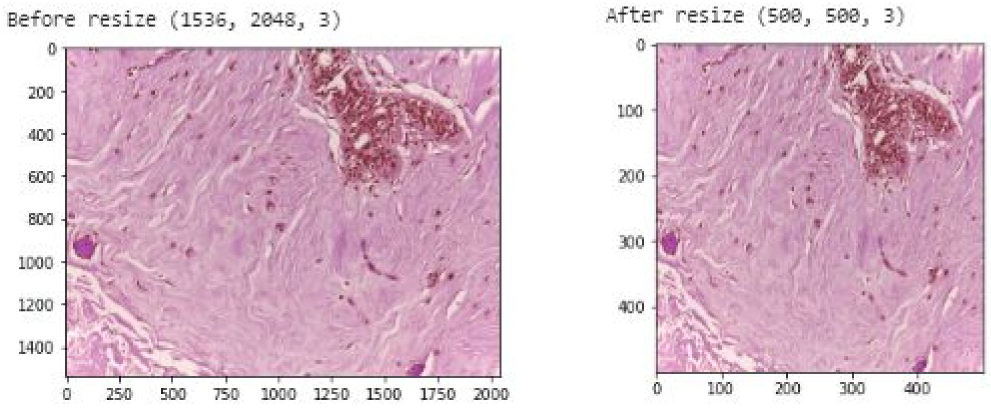


fig(1)

The images are represented each type has it's label respectively,This is sample images of each type,The data size is nearly "20 GB" so it's very big to upload so I put samples of it in folder called "dataset sample" supplied in the zip.
The input of our CNN is the images and the output is each category label .

Exploratory Visualization

Actually The annotation was performed by two medical experts and images where there was disagreement were discarded. So to me to find relevant characteristic or feature about the data is too hard , but after opening images and plotting it i found some features that may lead to thigh . the first is that the image had a part with specified shape which differ from others .
Secondly that part can be seen in the same place so using augmentation would help us a lot with this part to look it in different ways also to have additional dataset
And finally from the visualization below that reduce the size would affect us badly although it reduces time to train ,but it think it would reduce the accuracy as the shape of the tumour had changed .
fig(2)



Algorithms and Techniques

I will use Convolutional Neural Network as a classifier , for most image processing CNN would be used as a state-of-the-art algorithm. The problem here with me is that cnn needs a large amount of training data compared to other approaches, but I think we could fix that using image augmentation . The algorithm outputs an assigned probability for each class of the 4 classes that we had . our goal is to increase the accuracy of the total number of true classified over total number of classes . so as a sequence our goal is to optimize the classifier so The following parameters should be tuned:
Preprocessing parameters like we mentioned before we will try to use image augmentation to increase the total number of images .

For training parameters
1.Training length which is number of the total epochs
2.Batch size number of images to look at once during a single training step
3.Learning rate this can be dynamic and alos static it mean how fast to learn

For neural network architecture
1.Number of total layers
2.Layer types (convolutional, fully-connected, dropout ,or pooling)

During both validation and train the data would be loaded to ram then random batches are selected from the data in ram to be moved to the GPU .
I used rmsprop as optimizer and loss as categorical_crossentropy .

Benchmark

My benchmark model would be naive model that always predicts the most common class , but as our classes is balanced so each class has the same number of data we can say that our benchmark would always choose any class from the 4 classes that would lead to accuracy = (¼) which is 25 % . so with this little number of data my goal is to beat naive model by getting accuracy greater than 25 % .

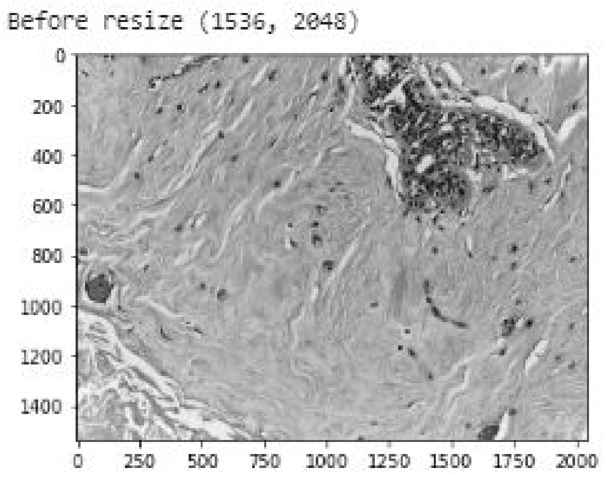
Methodology

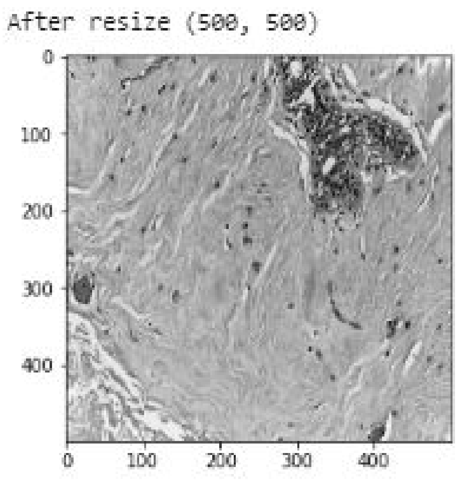
Data Preprocessing

As i said before in DataSet section that our data is filtered by a specialist doctors and is balanced so there isn't a lot of data preprocessing to do , but I think we could mention the following steps :

1. As i used colab so the first step was to load data which is in 4 folders each folder or class contains 100 images .
2. The images are converted to grayscale .
3. Then the images would be resized .
4. Then we split data to x,y array x contain features and y contain labels .
5. Then splitting our data to train , test and validation .
6. Then we will do categorical to images .
7. Rescale our images by dividing it by 255 as our models works well with values between 0 and 1 .

fig(3)





As we can see that now our image is converted to grayscale and also resized to 500 by 500 pixels .

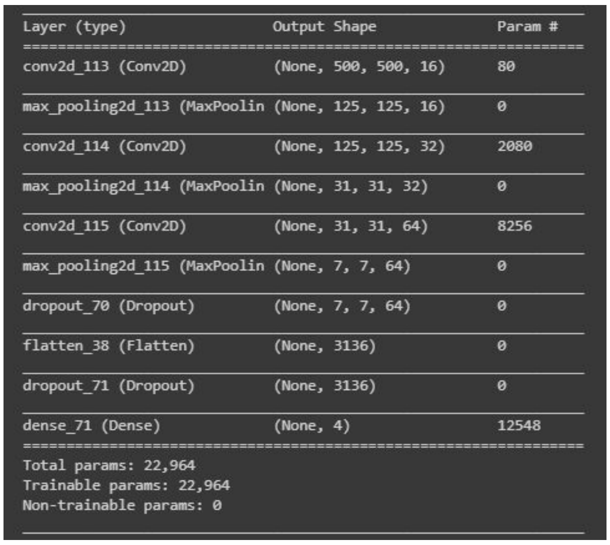
Implementation

The implementation was done in colab which i will download it and put it in the zip file it's called "Nanodegree Breast Cancer(histology).ipynb" it contain all implementation with comment in each section . all libraries we needed was supplied by colab .

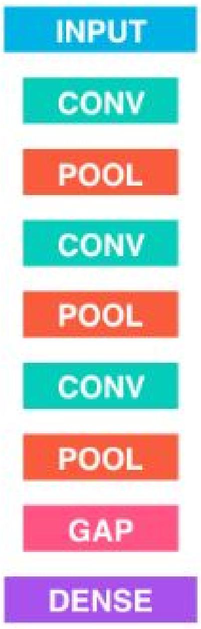
Our implementation stages is :

1. We upload the data to our google drive to make it easy to access it from colab
2. Mount our drive with workspace we work in .
3. Load the data from 4 folders for each class "Benign","InSitu","Invasive","Normal" .
4. Preprocessing the data to firstly read it in grayscale then to resize it and finally to split it .
5. The next stage is to define our CNN architecture and we fill put summary of our model
6. Finally to compile and finally to the model and check the accuracy .

This is the model summary : fig(4)



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_113 (Conv2D)          (None, 500, 500, 16)      80
_________________________________________________________________
max_pooling2d_113 (MaxPoolin (None, 125, 125, 16)      0
_________________________________________________________________
conv2d_114 (Conv2D)          (None, 125, 125, 32)      2080
_________________________________________________________________
max_pooling2d_114 (MaxPoolin (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_115 (Conv2D)          (None, 31, 31, 64)        8256
_________________________________________________________________
max_pooling2d_115 (MaxPoolin (None, 7, 7, 64)          0
_________________________________________________________________
dropout_70 (Dropout)         (None, 7, 7, 64)          0
_________________________________________________________________
flatten_38 (Flatten)         (None, 3136)              0
_________________________________________________________________
dropout_71 (Dropout)         (None, 3136)              0
_________________________________________________________________
dense_71 (Dense)             (None, 4)                 12548
=================================================================
Total params: 22,964
Trainable params: 22,964
Non-trainable params: 0
_________________________________________________________________
```

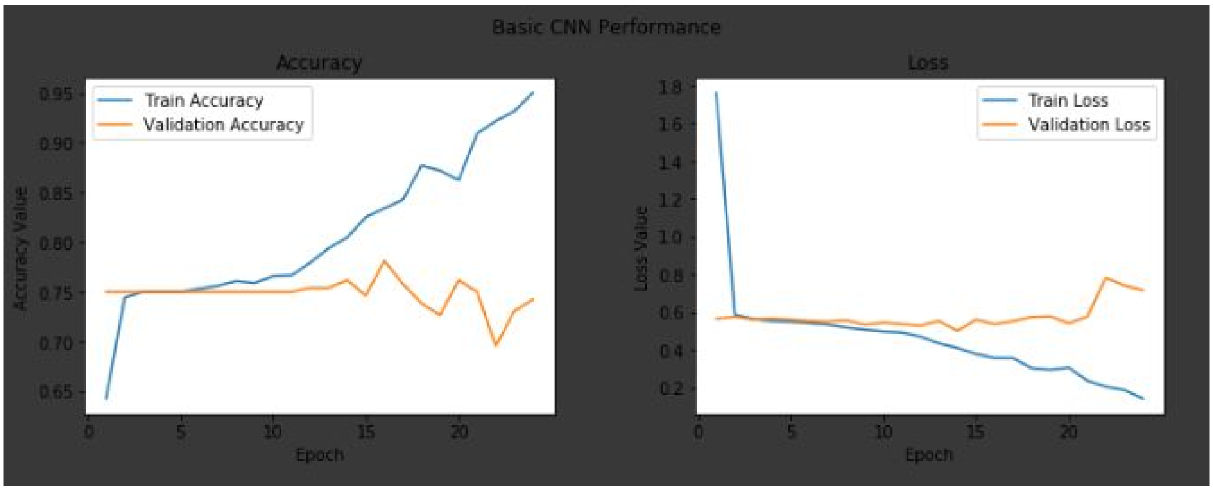


This architecture is inspired from udacity “Dog breeds project ” ,but our train params is so big as we used 500 by 500 pixel we will also try with different size to reduce the total params . To calculate accuracy we would use np.argmax(model.predict()) as our output is a list contain probability of each category so we would choose the highest probability and get it’s index which represents the predicted categories . Then we would calculate the true predication and divide it by the total length of the predication .

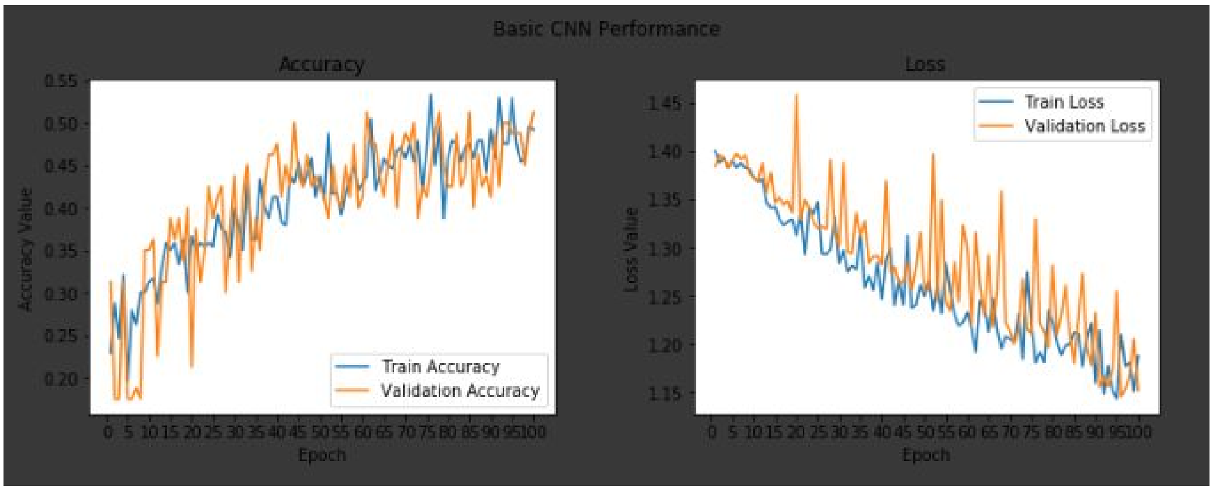
The problem I had was the training time also the accuracy wasn’t good as tran data wasn’t enough so we tried image augmentation to overcome this problem .

Refinement

Our train process was very hard due to the leak in the number of pictures that we have which is 100 per each category .
At first , I used image size of 500 by 500 using cnn architecture that we described before . without augmentation i am using early stop callback . I set number of epochs to 100 , but model stopped at 26 with 48 % for test accuracy and 84 % for train accuracy . that's obvious that our model is overfitting . fig(5)



We can see that from accuracy vs epochs graph that train accuracy leave validation accuracy .
To try to overcome that the solution was to add more photos , but we haven't any more data so we would move to the second solution which is to use augmentation to increase our accuracy .
By using augmentation we could to overcome overfit and get accuracy of 41.25 %
We can see that there is a difference between test and train , but that difference is acceptable due to the small number of pictures that we have in our data . fig(6)



1. We tried difference values of image_size and we believe that would help us , but we got error from colab "OOM" which means that the memory can't do that .
2. We tried full coloured images but that didn't change accuracy a lot .

3. We changed the value of steps_per_epoch in model.fit_generator but that give bad result as it follow this relationship steps_per_epoch =len(X_train_notscalled)//batch_size
4. We changed the percentage of data used for validation but the result was bad the results was about 27 % , The best result was for .2 for test , .2 for validation .6 for training . I founded it’s simple to use this method for splitting rather than k-fold .
5. we used Batch_size of 60 but the result wasn’t good was 37.5 % .

Results

Model Evaluation and Validation

The final model is CNN model with layers each layer perform a job .choosing the final architecture and hyperparameters were chosen by the performance and accuracy. As we see in fig(4) which contain the layers to describe it more :

1. The input layer which is convolutional layer with relu activation function that receive image with 500 * 500 *1 that mean both width and height of 500 and gray scale .
2. The first convolutional layer learns 16filters, the second learns 32filters and third 64 filter .
3. Then we use maximum pool that chose the maximum value which reduce the time to train a lot the pool size is equal to 4
4. We added a dropout layer to try to overcome overfit .
5. Then we put flatten layer .
6. Then we added a dense layer with softmax as an activation a 4 output categories that give us probability of each type .

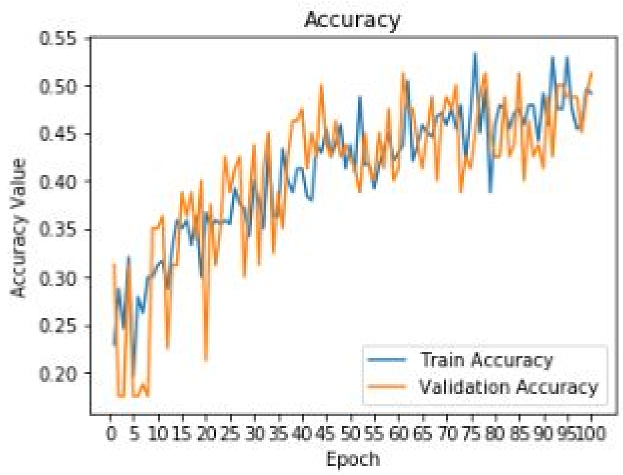
The final model seems to be not bad to make sure of that we tested our model to test data we have which is 80 pictures from different categories and we get accuracy of 41.25 % which seems to be not bad , but with improvement i think we could more good results .
The problem here is this data belong to challenge and i only have 400 images and can’t get any more , so I tested the model on these 80 image . also the images size is very big as each picture from the data could be 100 MB which is very big and a lot of problems faced me when i tried to improve the model due to the limitation on colab memory which couldn’t work with more complex architecture and even with more image size .

Justification

Our benchmark accuracy was naive assumption in which we chose the same class every time so our accuracy would be 25% as we have only 4 classes .
Our model accuracy is greater than that so i think our accuracy is better than the benchmark , but we still need to improve our model by improving it more data .
That means if we have 100 image our model would classify 41 image true from them , but the benchmark would classify only 25 from them true .

Conclusion

Free-Form Visualization



From this graph for the accuracy i think that we want more images to train our model on it to have smooth curves and have high accuracy also we need to increase the value of image resize our model were trained on 500 by 500 pixels so we can increase the accuracy Also we can make accuracy high but we need more ram and

gpu to do that . Our goal is to reach higher accuracy to beat doctors accuracy in this field to save millions of lives .
Our model input would be the images of microscopic and the output would be the class that this image belong to all of that is done using CNN .

Reflection

To summarise what we do :

1. We make data preprocessing in which we resize the images and read it in gray color .
2. We then trained our model in cnn to try to learn model the pattern of each class and train him how to classify each type .
3. Then we calculated the accuracy of the model which was bad and led us to step 4 .
4. We recognized that our train data isn't enough so we used augmentation that increased the number of train data .
5. After calculating the accuracy and improving it we tried to improve it by changing parameter and changing the model architecture
6. Finally we reached our final accuracy with all what we can do with this data .

I faced a lot of problems in the project such as :

1. I didn't know anything about breast cancer or know what actually each class represent so i searched until I understood about it.
2. Model training was taking a lot of time so i reduced image size to reduce the load on the train and also the time .
3. Overfit appeared with me a lot will training due to the low number of images so i tried to add more dropout and image augmentation .
4. And finally one of the most difficult problems that I faced was how to upload the data to my workspace in colab so i tried a lot of solutions as the data size was a bout 12 GB and that would require a lot of time to upload because of my bad internet speed , so after a lot of attemption the solution was to use remote desktop which enabled me to download the data on my virtual machine then upload it to the colab workspace .

So I think that solution is good but not the best and we can improve it and also can be used in similar problems .

Improvement

I think that yes there better than what i did , but the problem with me is that the number of images isn’t suitable to get a good accuracy using CNN .
We can improve our model using :

1. More number of images in train .
2. Increase the image width and high we choose 500 , but we can increase till 1500 .
3. Look to the image in full color .
4. Increase the CNN depth and width .
5. Use transfer learning (That i tried to use , but faced problems that I couldn't solve).
6. Increase number of epochs .

And finally yes there is better than my model , I am satisfied from my model performance due to the leak in images .