Breast Cancer Classification using Transfer Learning

Murali Mallikarjuna Rao Perumalla
perumallamurali857@gmail.com

UDACITY

School of Artificial Intelligence
Machine learning engineer nanodegree

I. INTRODUCTION

Breast cancer is most commonly occurring cancer in the women and it's the second most common cancer overall. According to the 2018 statistics, there were over 2million cases all over the world. Belgium and Luxembourg have the highest rate of cancer. Early detection of breast cancer has reduced many deaths. Earlier CAD systems used to be second opinion for radiologists and clinicians. Machine learning and deep learning has brought tremendous changes in medical diagnosis and imagining. Early detection and classification of cancer helps specialists to diagnose accurately and reduces the patient re-visit rate.

Breast cancer is a serious issue for women. According to the recent statistics in 2018, it is estimated that 268,600 women in US are diagnosed with IBC, and 62,930 women are diagnosed as positive for situ breast cancer. Around 2,670 men in the US are diagnosed with breast cancer.

Detecting the breast cancer early has reduced the death rate. There is a scope for error in manually detecting the breast cancer and patients need to be callback which results in the wastage of the resources, expenditure etc. With the help of the machine learning and deep learning researchers have developed different approaches. These approaches help in detecting the breast cancer early and helps the radiologists, doctors for efficient detection of the cancer and reduces the patient callbacks.

In project I am using a transfer learning approach to perform 8-class classification on breast cancer. I have used various pretrained models and VGG16 showed good results. Snice the dataset I used is small I decided to fine tune the vgg16 by removing last convolution block i.e. layers containing the high-level features.

II. PROBLEM STATEMENT

It is known that Deep learning needs very huge amount of data and high computing resources. In case of breast cancer almost all datasets are small. And dataset is imbalanced. Small datasets are hard to converge. Imbalance in datasets leads to the overfitting, less test accuracy.

III. DATASET AND INPUTS

In this project I am using BreakHis breast cancer dataset. The Breast Cancer Histopathological Image Classification (BreakHis) is composed of 9,109 microscopic images of breast tumor tissue collected from 82 patients using different magnifying factors (40X, 100X, 200X, and 400X). To date, it contains 2,480 benign and 5,429 malignant samples (700X460 pixels, 3-channel RGB, 8-bit depth in each channel, PNG format). This database has been built in collaboration with the P&D Laboratory – Pathological Anatomy and Cytopathology, Parana, Brazil (http://www.prevencaoediagnose.com.br). We believe that researchers will find this database a useful tool since it makes future benchmarking and evaluation possible.

Magnification	Benign	Malignant	Total
40X	652	1,370	1,995
100X	644	1,437	2,081
200X	623	1,390	2,013
400X	588	1,232	1,820
Total of Images	2,480	5,429	7,909

The dataset currently contains four histological distinct types of benign breast tumors:

- adenosis (A),
- fibroadenoma (F),
- phyllodes tumor (PT),
- tubular adenona (TA);

four malignant tumors (breast cancer):

- carcinoma (DC),
- lobular carcinoma (LC),
- mucinous carcinoma (MC)
- papillary carcinoma (PC).

IV. PROJECT WORKFLOW

1. Data Preparation:

The dataset is taken from BreakHis site. They divided the dataset into benign and malignant parts. the structure of the dataset given is :

```
Breakhis/
    benign/
        class0/images..
        class1/images..
        class2/images..
        class3/images..
    malignant/
        class4/images..
        class5/images..
        class6/images..
        class7/images..
```

In order to train a model we need to prepare the data i.e. the divided into train, test and valid data. The structure of dataset after data preparation is:

```
Breakhis/ ,
    train/
        class0/images..
        class1/images..
        class2/images..
        class3/images..
        class4/images..
        class5/images..
        class6/images..
        class7/images..
    test/
        class0/images..
        class1/images..
        class2/images..
        class3/images..
        class4/images..
        class5/images..
        class6/images..
        class7/images..
    valid/
        class0/images..
        class1/images..
```

class2/images..
class3/images..
class4/images..
class5/images..
class6/images..
class7/images..

2. Data Pre-Processing:

Data Pre-processing is an important step in a machine learning project. Since, dataset is small and imbalanced, I decide to perform

- Re-Size to 256*256
- Augmentation: flipping(vertical and Horizontal) ,rotation(15°), crop(224)
- Normalize data
- WeightRandomSampler

In order to deal with imbalanced data, I have used PyTorch inbuilt sampler called WeightedRandomSampler(). It assigns weights to every sample based on the amount of imbalance in data.

3. Fine Tuning:

I have chosen vgg16 pretrained model. I also tried using vgg19,resnet18,densenet but vgg16 gave me good results. As dataset is small, the approach used is

- slice off all but some of the pre-trained layers near the beginning of the network
- add to the remaining pre-trained layers a new fully connected layer that matches the number of classes in the new data set
- randomize the weights of the new fully connected layer; freeze all the weights from the pre-trained network
- train the network to update the weights of the new fully connected layer

Because the data set is small, overfitting is still a concern. To combat overfitting, the weights of the original neural network will be held constant. But the original training set and the new data set do not share higher level features. In this case, the new network will only use the layers containing lower level features.

I have removed the last convolution block of vgg16 i.e. last 3 convolutions layers followed by activation functions and one max pooling layer.

4. Hyperparameter Tuning:

Hyperparameter Tuning is an important part in all AI projects. This helps model to converge and learn better. I have tried tuning different models and their parameters and finally came with the best parameters:

- Optimizer: Stochastic Gradient Descent
  - lr=0.007
  - momentum=0.9
  - weight_decay=0.1
  - nesterov=True
- Criterion: nn.CrossEntropyLoss()
- Lr_scheduler : ReduceLROnPlateau
  - optimizer
  - mode='max'
  - factor=0.7
  - patience=1,
  - verbose=True
- Epochs:50

5. Train and Test the model

- Train accuracy:92%
- Train loss:0.356620
- Validation loss:0.762146
- Test accuracy:70%

6. Evaluation metrics:

In this project I have sensitivity, specificity and confusion matrix as evaluation metric. Since sensitivity and specificity is for binary calssification and calculated the sensitivity and specificity for each class.

Sensitivity: Of all the people with cancer, how many were correctly diagnosed?
Specificity: Of all the people without cancer, how many were correctly diagnosed?

- Confusion matrix:

```
(([[ 82.,   0.,   4.,   1.,   3.,   1.,   1.,   0.],
   [  7., 483.,  10., 131.,  33.,  39.,   2.,   7.],
   [  9.,   9., 119.,   8.,   2.,   1.,  20.,  18.],
   [  3.,  19.,   0.,  83.,   4.,   2.,   0.,   0.],
   [  5.,  28.,   3.,  12., 111.,   4.,   0.,   9.],
   [  0.,  15.,   7.,   1.,   1.,  78.,   1.,   5.],
   [  6.,   1.,  11.,   0.,   1.,   4.,  64.,   6.],
   [  4.,   3.,   6.,   0.,   1.,   0.,   1.,  93.]])
```

- Class 0:
    - TP :82.0, TN :1456.0, FP :34.0, FN: 10.0
    - Sensitivity = 0.8913043737411499
    - Specificity = 0.9771811962127686
- Class 1:
    - TP :483.0, TN: 795.0, FP: 75.0, FN :229.0
    - Sensitivity = 0.6783707737922668
    - Specificity = 0.9137930870056152
- Class 2
    - TP 119.0, TN 1355.0, FP 41.0, FN 67.0
    - Sensitivity = 0.6397849321365356
    - Specificity = 0.9706303477287292
- Class 3
    - TP 83.0, TN 1318.0, FP 153.0, FN 28.0
    - Sensitivity = 0.7477477192878723
    - Specificity = 0.8959891200065613
- Class 4
    - TP 111.0, TN 1365.0, FP 45.0, FN 61.0
    - Sensitivity = 0.645348846912384
    - Specificity = 0.9680851101875305
- Class 5
    - TP 78.0, TN 1423.0, FP 51.0, FN 30.0
    - Sensitivity = 0.7222222089767456
    - Specificity = 0.9654002785682678
- Class 6
    - TP 64.0, TN 1464.0, FP 25.0, FN 29.0
    - Sensitivity = 0.6881720423698425
    - Specificity = 0.9832102060317993
- Class 7
    - TP 93.0, TN 1429.0, FP 45.0, FN 15.0
    - Sensitivity = 0.8611111044883728
    - Specificity = 0.9694707989692688

V. RESULTS AND DISCCUSIONS

In this project I have achieved a train accuracy of 92% and test accuracy of 70%. since accuracy is not the best metric for multi class classification I have used confusion matrix, sensitivity and specificity and we can see that specificity of each class is greater than 90% and sensitivity is greater than 63%. by looking at the results we can say that there is overfitting in model and imbalance data need to be taken care. As future work, we can remove imbalance by replacing or increasing the dataset size and try to reduce overfitting.