# 6000B Project 3 Option 3 Domain Adaptation

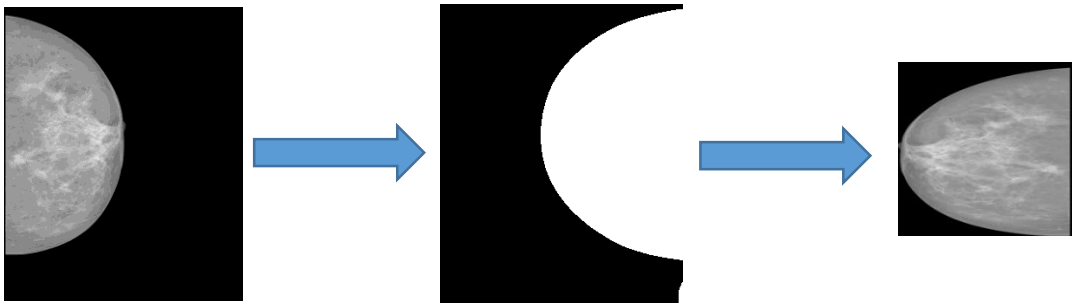
*Group Member: Meng LI, Lu YI, Ning YAN, Wenting LUO, Jingyi WANG*

## 1. Introduction

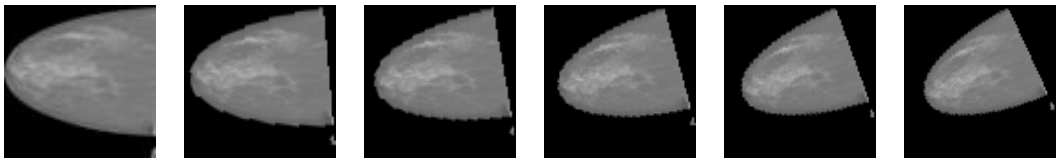
X-ray devices differs in qualities and resolutions, hence in machine learning, model trained on the dataset from one device may fail to predict on the dataset from another device. In this situation, domain adaptation helps address the problem of differences between the two datasets. In our project, we investigated different domain adaptation approaches including standard supervised domain adaptation and two approaches of unsupervised domain adaptation on breast cancer datasets.

## 2. Data Preprocessing

In this section, we transform the original images into a 64x64 size images. First we adopt Otsu's segmentation to remove the background. The Otsu is used to automatically perform the reduction of a gray level image to a binary image. And we reverse all the left breast in row. Then we adopt the opening morphology to the original images to remove noise. The original image and processed image are shown as below.



Because the dataset size is too small, adopt the original dataset cannot obtain a satisfied result, we rotate each image by 5°, 10°, 15°, 25°,etc.



## 3. Supervised Domain Adaptation

### 3.1 Base Model

The Support Vector Machine(SVM) is selected as the baseline in this project. Since the dataset is small that only contains around 400 pictures, using the convolution neural network(CNN) to classify the breast cancer dataset would

not perform well. SVM can be treated as a simple neural network while it handles the small dataset better than the deep neural network, including the CNN.

Unlike the deep learning methods, which can train a model based on the raw picture, SVM needs to extract the features of pictures and then the features are input to train the SVM model. However, based on this method, the input dimension would be very high that the curse of dimensionality would occur. Therefore, the dimension reduction is needed before training the model. In this project, the principle component analysis is applied to reduce the dimension of the features extracted from the pictures.

The main steps for building the base model is shown as below.

**Algorithm 1** Base Model Construction for Breast Cancer Image Classification

```
Img_list ← file path of all images
labels ← labels for all images
for i = 1,...,N do
    read image i in Img_list
    transform image data matrix to one dimension array, arr
    train_data.append(arr)
end for

train_data ← pca.transform(train_data)
train a SVM model with train_data
```

The train data used to train the SVM includes both training data from dataset A and dataset B. The trained model is used to predict the label of test data in dataset B. In order to test the performance of the SVM model, the cross validation is applied. The average training accuracy is 0.779 while the validation accuracy is around 0.65. The Bayes Naïve algorithm is applied as a metric to select the model.

***Packages used:*** *OpenCv, sklearn*

***Environment:*** *python 2*

```
0 Fold svm Train accuracy:0.777778, Test accuracy:0.676923
0 Fold bayes Train accuracy:0.758974, Test accuracy:0.569231
1 Fold svm Train accuracy:0.776068, Test accuracy:0.553846
1 Fold bayes Train accuracy:0.747009, Test accuracy:0.692308
2 Fold svm Train accuracy:0.774359, Test accuracy:0.692308
2 Fold bayes Train accuracy:0.733333, Test accuracy:0.692308
3 Fold svm Train accuracy:0.801709, Test accuracy:0.630769
3 Fold bayes Train accuracy:0.752137, Test accuracy:0.538462
4 Fold svm Train accuracy:0.758974, Test accuracy:0.707692
4 Fold bayes Train accuracy:0.736752, Test accuracy:0.723077
5 Fold svm Train accuracy:0.774359, Test accuracy:0.661538
5 Fold bayes Train accuracy:0.747009, Test accuracy:0.661538
6 Fold svm Train accuracy:0.782906, Test accuracy:0.707692
6 Fold bayes Train accuracy:0.747009, Test accuracy:0.630769
7 Fold svm Train accuracy:0.776068, Test accuracy:0.692308
7 Fold bayes Train accuracy:0.740171, Test accuracy:0.661538
8 Fold svm Train accuracy:0.781197, Test accuracy:0.584615
8 Fold bayes Train accuracy:0.758974, Test accuracy:0.600000
9 Fold svm Train accuracy:0.786325, Test accuracy:0.630769
9 Fold bayes Train accuracy:0.741880, Test accuracy:0.630769
svm train accuracy: 0.778974, Test accuracy: 0.653846
bayes train accuracy: 0.746325, Test accuracy: 0.640000
```

## 3.2 Supervised Domain Adaptation

The domain adaption aims at using a well performing model learnt from a source dataset to train a different but related target dataset. Since the dataset is small, totally based on the dataset A to train a CNN model with good performance is difficult, therefore, the CNN model for dataset A is trained by fine tuning the pre-trained model, VGG16. Firstly, using the VGG16 model to extract the features of the images from dataset A and then input them to fully connected layers to train a new model. Based on the parameters of the new model, fitting the model with the images from dataset B. However, for the images from dataset B, the VGG16 model is also used to extract the features. When training the dataset B, the accuracy decreases first, and then increases to a stable level. The training accuracy and validation accuracy for training dataset A is 0.9887 and 1.000 respectfully. The training accuracy and validation accuracy for training dataset B is 0.9484 and 0.6500. The probable reasons that the performance on dataset B are that the validation data from dataset B is much different from the training data in that dataset; the training data of dataset B is extremely small. In addition, the structure of the model is not designed well can also lead poor performance.

```
--
Training Step: 37  | total loss: 0.06513 | time: 0.143s
| Adam | epoch: 005 | loss: 0.06513 - acc: 0.9755 -- iter: 050/410
Training Step: 38  | total loss: 0.16847 | time: 0.277s
| Adam | epoch: 005 | loss: 0.16847 - acc: 0.9646 -- iter: 100/410
Training Step: 39  | total loss: 0.13708 | time: 0.378s
| Adam | epoch: 005 | loss: 0.13708 - acc: 0.9714 -- iter: 150/410
Training Step: 40  | total loss: 0.11164 | time: 0.480s
| Adam | epoch: 005 | loss: 0.11164 - acc: 0.9768 -- iter: 200/410
Training Step: 41  | total loss: 0.09147 | time: 0.624s
| Adam | epoch: 005 | loss: 0.09147 - acc: 0.9810 -- iter: 250/410
Training Step: 42  | total loss: 0.07854 | time: 0.757s
| Adam | epoch: 005 | loss: 0.07854 - acc: 0.9844 -- iter: 300/410
Training Step: 43  | total loss: 0.07060 | time: 0.894s
| Adam | epoch: 005 | loss: 0.07060 - acc: 0.9872 -- iter: 350/410
Training Step: 44  | total loss: 0.06093 | time: 1.032s
| Adam | epoch: 005 | loss: 0.06093 - acc: 0.9894 -- iter: 400/410
Training Step: 45  | total loss: 0.05600 | time: 2.182s
| Adam | epoch: 005 | loss: 0.05600 - acc: 0.9878 | val_loss: 0.01454 - val_acc:
1.0000 -- iter: 410/410
--
```

```
--
Training Step: 136  | total loss: 0.32853 | time: 0.316s
| Adam | epoch: 024 | loss: 0.32853 - acc: 0.9408 -- iter: 050/240
Training Step: 137  | total loss: 0.29816 | time: 0.806s
| Adam | epoch: 024 | loss: 0.29816 - acc: 0.9467 -- iter: 100/240
Training Step: 138  | total loss: 0.27127 | time: 1.245s
| Adam | epoch: 024 | loss: 0.27127 - acc: 0.9520 -- iter: 150/240
Training Step: 139  | total loss: 0.28632 | time: 1.649s
| Adam | epoch: 024 | loss: 0.28632 - acc: 0.9488 -- iter: 200/240
Training Step: 140  | total loss: 0.26135 | time: 2.922s
| Adam | epoch: 024 | loss: 0.26135 - acc: 0.9539 | val_loss: 1.00144
val_acc: 0.6500 -- iter: 240/240
--
```

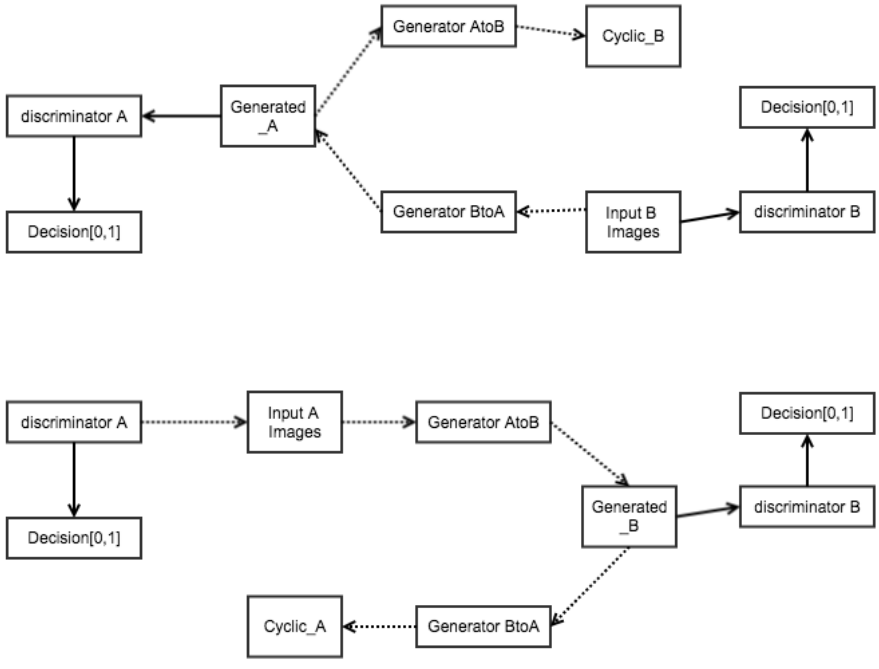
***Packages used:*** *tflearn, keras*
***Environment:*** *python3.5.2*

# 4. Unsupervised Domain Adaptation

## 4.1 Cycle GAN

### 4.1.1 Introduction

One of the unsupervised domain adaptation methods: Cycle GAN, which is unpaired image-to-image translation using Cycle-Consistent Adversarial Networks. As for the adversarial network, it contains a generator network and discriminator network playing against each other. The generator tries to generate a sample from the desired distribution, and the discriminator tries to predict whether the sample came from the actual distribution or from the generator. The generator and discriminator are trained jointly, so that the generator learns to approximate the underlying distribution completely and the discriminator is left guessing randomly. In this model, the network learns to translate an image from a source domain A to a target domain B without a paired instance. The aim of our model is to learn the mapping G: X→Y such that the image distribution in G (X) is indistinguishable from the distribution Y using adversarial loss. Since the mapping is highly under constrained, we couple it with an inverse mapping F: Y → X, introducing a cyclic consistency loss to force F (G (X)) ≈X (and vice versa).



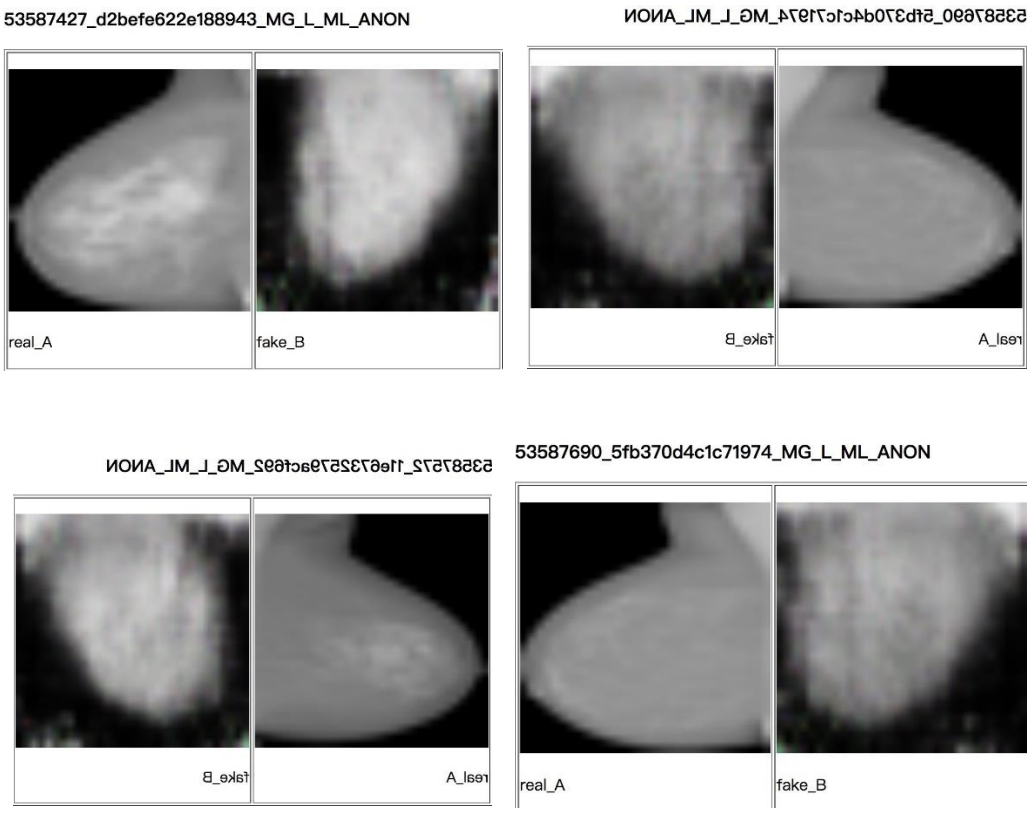

### 4.1.2 Experiment

In our experiment, the dataset mainly contains two parts, trainA and trainB, our goal is to translate the images from trainA to the target domain trainB, therefore, these two data sets are all put into the Cycle GAN without labels.

Environment Configuration
Linux or macOS; Python 2 or 3; CPU or NVIDIA GPU + CUDA CuDNN; packages: PyTorch and dependencies, Torch vision, visdom, dominate.

### 4.1.3 Details of Implementation

We mainly use the network from Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks; un-Yan Zhu*, Taesung Park*, Phillip Isola, Alexei A. Efros; In arxiv, 2017. And we make some minor adjustments to this network: both the numbers of generator filters and discriminator filters are 32, the learning rate is 0.001, the pool size is 20, the size of image is 32, the number of threads is 50. Actually, it takes a long time to train this model, so we change some parameters so that the program can run in a shorter time. As for the architecture, it has two convolutions with 2 strides, six blocks for 16*16 images, 9 blocks for 32*32 images, two fractionallystrided convolutions with stride 1/2. We keep the learning rate 0.001 for the first 100 epochs and linearly decay the rate to zero over the next 100 epochs. After getting new images by Cycle GANs, we convert the format of images to another format which can be processed in the standard classification model.




### 4.1.4 Future works

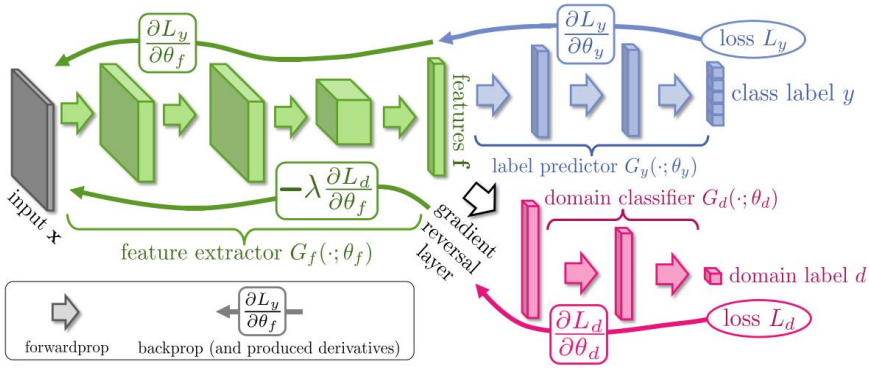
Our result is actually not good, for the image size is too small to be put in the model for learning. If we change the size from 32*32 to the larger size, it will cost more time to finish learning, even though the program runs on

GPUs. In the future, we can put images of 256*256 into the Cycle GAN model, and change the learning rate to a smaller value.

## 4.2 Gradient Reversal Layer (GRL)

### 4.2.1 Introduction

In this project, we use Gradient Reverse as a second approach to perform unsupervised domain adaptation. The main idea is to embed domain adaptation into the process of learning representation, so that the final classifications are based on features that are both discriminative and invariant to the change of domains. We add a gradient reversal layer(GRL) into the normal training which can extract gradient and multiply it with a negative constant during the process of backpropagation, in order to keep domain invariant features.




### 4.2.2 Implementation

In this unsupervised domain adaptation, our model consists of three parts, feature extractor, label predictor and domain classifier. We use only source domain data to train the label predictor, while use both domains to train the domain classifier. However, as we add an gradient reversal layer during the backpropagation for domain classifier, it would struggle in the training process and finally should fail to distinguish target domain data from source domain. In other words, the accuracy of domain classifier converges to 0.5 theoretically.

For both breast cancer datasets, we use pre-trained VGG16 model from Keras package as feature extractor. For label predictor on source domain, we construct two fully-connected layers after feature embedded layer. 2-layers domain classifier (x $\rightarrow$ 512 $\rightarrow$ 2) is constructed after the gradient reversal layer.

During the training procedure, we use stochastic gradient descent with 0.9 momentum and the learning rate annealing described by the following formula: $\mu_p = \frac{\mu_0}{(1+\alpha \cdot p)^\beta}$, where p is linearly changing from 0 to 1 in the

training process, and $\mu_0 = 0.01, \alpha = 10, \beta = 0.75$ as suggested in (Ganin, Y., & Lempitsky, V. (2014)) to promote convergence and reduce error in source domain.

For GRL, in order to suppress noisy signal from the domain classifier at the early stages of the training procedure, instead of fixing the adaptation factor λ, we gradually change it from 0 to 1 according to the following fomula:

$\lambda = \frac{2}{1+\exp(-\gamma \cdot p)} - 1$, where $\gamma = 10$ in experiment.

### 4.2.1 Experiment

***Environment**: MATLABr2017b, python3.5*

***Package Used**: numpy, tensorflow, tflearn, keras, VGG16*

We first perform source only training on source domain data and then directly fit the model to target domain data without training. Training results of our GRL approach is shown as follows.

```
Source only traning:
Source (A) accuracy: 0.75
Target (B) accuracy: 0.45
loss: 5.036018  d_acc: 0.437500  p_acc: 0.500000  p: 0.000000  l: 0.000000  lr: 0.010000
loss: 3.191512  d_acc: 0.531250  p_acc: 0.875000  p: 0.066667  l: 0.321513  lr: 0.006817
loss: 3.630186  d_acc: 0.500000  p_acc: 0.812500  p: 0.133333  l: 0.582783  lr: 0.005297
loss: 4.188465  d_acc: 0.406250  p_acc: 0.687500  p: 0.200000  l: 0.761594  lr: 0.004387
loss: 3.511569  d_acc: 0.375000  p_acc: 0.812500  p: 0.266667  l: 0.870062  lr: 0.003774
loss: 3.810500  d_acc: 0.468750  p_acc: 0.625000  p: 0.333333  l: 0.931110  lr: 0.003330
loss: 3.571302  d_acc: 0.531250  p_acc: 0.750000  p: 0.400000  l: 0.964028  lr: 0.002991
loss: 4.395787  d_acc: 0.437500  p_acc: 0.812500  p: 0.466667  l: 0.981368  lr: 0.002723
loss: 4.485137  d_acc: 0.500000  p_acc: 0.687500  p: 0.533333  l: 0.990390  lr: 0.002505
loss: 3.691796  d_acc: 0.468750  p_acc: 0.875000  p: 0.600000  l: 0.995055  lr: 0.002324
loss: 3.868949  d_acc: 0.468750  p_acc: 0.812500  p: 0.666667  l: 0.997458  lr: 0.002170
loss: 3.758812  d_acc: 0.500000  p_acc: 0.937500  p: 0.733333  l: 0.998694  lr: 0.002039
loss: 3.711430  d_acc: 0.468750  p_acc: 0.625000  p: 0.800000  l: 0.999329  lr: 0.001925
loss: 3.893004  d_acc: 0.437500  p_acc: 0.625000  p: 0.866667  l: 0.999656  lr: 0.001824
loss: 4.101520  d_acc: 0.406250  p_acc: 0.687500  p: 0.933333  l: 0.999823  lr: 0.001735

Domain Adaptation training:
Source (A) accuracy: 0.725
Target (B) accuracy: 0.625
Domain accuracy: 0.475
```

## 5. Results

| Supervised Domain Adaptation | | | | | |
|---|---|---|---|---|---|
| SVM | | CNN+VGG16 | | | |
| Training A | Validation A | Training A | Validation A | Training B | Validation B |
| 0.7789 | 0.6538 | 0.9887 | 1 | 0.9484 | 0.6500 |

| Unsupervised Domain Adaptaion | | | | | | |
|---|---|---|---|---|---|---|
| Circle Gan | | GRL | | | Without adaptation | |
| Training A | Validation B | Training A | A validation | B validation | A validation | B validation |
| 0.8560 | 0.6500 | 0.8976 | 0.7250 | 0.6250 | 0.7500 | 0.45 |

For GRL, it's clearly that the result of without adaptation are worse than GRL, because the reversal gradient layer is used to confuse domains which maps the two domains data to a common space. In this way, the model build on the dataset A can be used on dataset B. If we don't adopt the reversal layer, than the model will be trained to distinguish domains which can easy recognize the which domain the input data belong, then the training model isn't a common model. It explain why the result below 0.5.
From the GRL result, we can see that in the A validation set, the accuracy rate is far below than the training accuracy. We guess it's because that the number of validation is small and the examples are unbalance. The test rate is lower than the rate on A validation set, may be because our training model still not good enough to extract feature.
As for the result of the model with Cycle GAN, it is better than the baseline model's. While the result is still not good enough. Because the result of supervised domain adaptation is better. And the predictions for test data set are all zero, which means that all the patients being tested are diagnosed with no breast cancer. It seems a little bit unreasonable, but we can only follow the result trained by the model. We can improve the result in the future, based on the lack of available time.

## *Reference*


1. Geras, K., Wolfson, S., Shen, Y., Kim, S., Moy, L., & Cho, K. (2017). High-Resolution Breast Cancer Screening with Multi-View Deep Convolutional Neural Networks.
2. Ganin, Y., & Lempitsky, V. (2014). Unsupervised Domain Adaptation by Backpropagation.
3. Zhu, J., Park, T., Isola, P., & Efros, A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks.