ECE 435, Medical Image Processing
Assignment 4: Mammogram Classification for Breast Cancer Detection using Traditional Machine Learning and Deep Learning

Spring 2020

This assignment focuses on the different steps involved in a medical imaging classification pipeline: data preparation, preprocessing, feature extraction and classification. The data extraction and classification phases will be done using two distinct approaches: traditional patter recognition and deep learning.

The dataset used in this assignment is the **Mammographic Image Analysis Society (MIAS) MiniMammographic Database** [1], which is composed by 322 mammograms of 1024×1024 pixels. These annotated mammograms can be used to train and evaluate a system for the autonomous detection of breast cancer. Each sample is annotated based on multiple sub-classes of three characteristics, 1) character of background tissue; 2) class of abnormality present and 3) severity of abnormality. For simplicity purposes, in this assignment we will classify each mammogram based only on their **character of the background tissue**, which can be one of the following:

1. **F**: Fatty
2. **G**: Fatty-glandular
3. **D**: Dense-glandular

In order to complete this assignment you need to download and install MATLAB's Deep Learning Toolbox [2] and ResNet-50 Toolbox [3]. Although not mandatory, the use of a GPU [4] will considerably accelerate the training processes. Note that the completion of problems problems 6 and 7 **will be awarded with up to two extra points** (for a total possible mark of 10 in this assignment).

[1] `http://peipa.essex.ac.uk/info/mias.html`

[2] `https://www.mathworks.com/products/deep-learning.html`

[3] `https://www.mathworks.com/help/deeplearning/ref/resnet50.html`

[4] `https://www.mathworks.com/help/parallel-computing/gpu-computing-in-matlab.html`

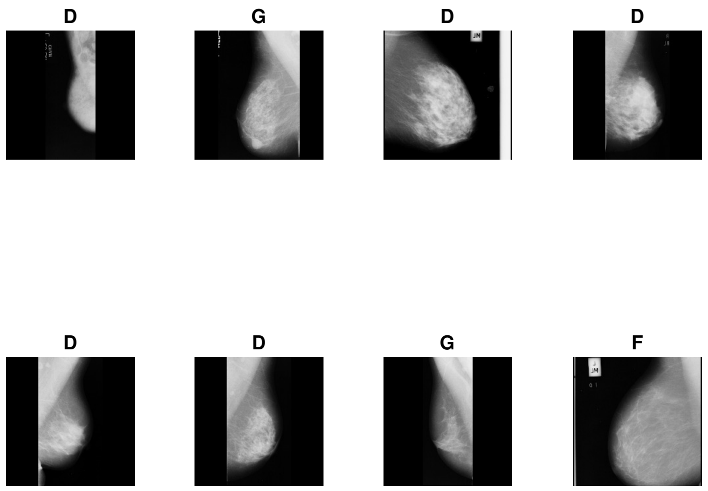


Figure 1: Sample mammograms from the MiniMammographic Database. Character of tissue background classes are highlighted on top of the samples.

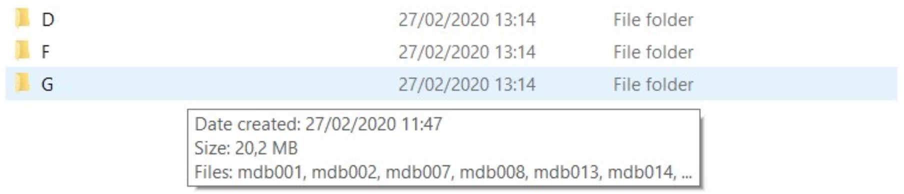
1 Data preparation and preprocessing

The data provided in medical imaging datasets often presents different file formats (e.g., DICOM, pgm, png) and custom annotation styles. In order to illustrate that, the first step of the assignment is to read the "labels.txt" file containing information about the dataset samples. Each of the 7 columns in this text file have a specific meaning, however we are interested only on the first and second ones, which refer, respectively, to the name of the sample and the classification of its background tissue (i.e., F, G or D).

Problem 1: Data organization
(a) (**1 point**): Create one folder for each of the three classes and write a script that reads the "labels.txt" file and places a **.png** version of each sample into its appropriate folder, as illustrated in Figure 2.
Hint. Use MATLAB's *readtable*, *imwrite* and *split* functions.

Problem 2: Data preprocessing
Specific data preprocessing steps will vary depending on the characteristics of the images composing each dataset. For the mammograms of the MiniMammographic Database, we will apply a simple routine of transformations for each sample:

1. Resize the image to 224×224 pixels (allowing for the training of a specific deep learning-based network).


D 27/02/2020 13:14 File folder

F 27/02/2020 13:14 File folder

G 27/02/2020 13:14 File folder

Date created: 27/02/2020 11:47

Size: 20,2 MB

Files: mdb001, mdb002, mdb007, mdb008, mdb013, mdb014, ...

Figure 2: Folder structure and sample names from folder G (i.e., fatty-glandular mammograms).

2. Turn the gray-scale image into a 3-channel image by duplicating its values into three channels.
3. Apply a histogram equalization process.

(a) (**1.5 points**): Create the *preProcess* function to apply the aforementioned transformations (see 'solutionTemp.m' for a template of it). All the images in the dataset should be preprocessed and saved in their adequate folders (i.e., F, G or D) as .png images. Figure 3 presents an original mammogram and its preprocessed version.
Note: at the end of this phase, only the preprocessed samples should be kept in the folder (you can use the original sample's name as the name of the preprocessed ones).

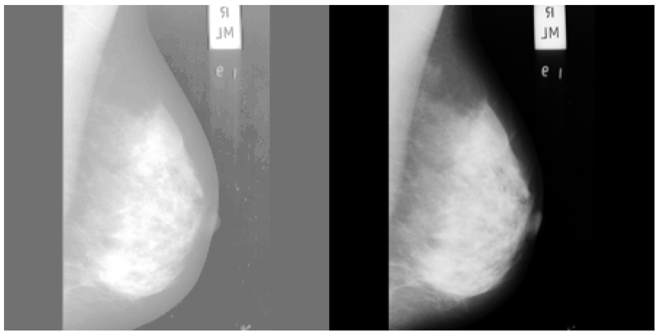


Figure 3: **Left:** Preprocessed mammogram. **Right:** Original mammogram (resized to 224 × 224 pixels for visualization purposes).

Problem 3: Data division

In order to train a machine-learning-based classifier, one must divide the data samples into *training* and *testing* sets. The samples from the training set are only used to train the classifier, while those from the test set are only considered when evaluating it.

(a) (**0.5 point**): Load the data into an *imageDatastore* element using MATLAB's *imageDatastore* function with 'foldernames' as 'LabelSource'. That will facilitate the eventual shuffling, training and evaluation processes. Report the number of samples in each class of the imageDatastore using MATLAB's *countEachLabel* function.

(b) (**0.5 point**): Use MATLAB's *splitEachLabel* function to **randomly** divide the data into two sets: *imdsTrain* and *imdsTest*. The training set must contain 70% of the data, while the testing set must hold the remaining 30%.

2 Classification using Pattern Recognition

In traditional pattern recognition, hand-crafted features of the data are first extracted and then used as input for some classification method (e.g., decision trees, random forests, nearest neighbors, linear classifiers, support vector machines).

We are going to extract the image's features using the popular Histogram of Oriented Gradients (HoG) method [5], which determine feature descriptors based on the number of occurrences of gradient orientations in different positions of an image. These features are going to be used to train and evaluate a Support Vector Machine [6] (SVM) classifier.

Problem 4: Histogram of Oriented Gradients

(a) (**1 point**): Create the *extractHOG* function to calculate the HoG features from all the images of an imageDatastore element using an specified cell size (e.g., 2×2, 4×4). Refer to the 'solutionTemp.m' file for a template of this function. You should use MATLAB's *extractHOGFeatures* to extract the HoG features form each sample. The feature array holding the HoG features of each mammogram should be stored in an individual row of a matrix.

Use your *extractHOG* function to extract the HoG features from the mammograms in the training and testing imageDatastores created on Problem 3 with a 4×4 window.

Report what would be the size of the feature array of each sample for 2×2, 4×4 and 8×8 windows.

Problem 5: Classification using a Support Vector Machine

(a) (**1.5 points**): Create an SVM classifier element using MATLAB's *fitcecoc* function. The inputs of this function should be your training HoG features (extracted on Problem 4) and their equivalent labels (i.e., F, G or D).

(b) (**1 point**): Use MATLAB's *predict* function to test your classifier by using it to predict the classes of the test HoG features (extracted on Problem 4). The inputs of this function should be your trained SVM and the testing HoG features.

(c) (**1 point**): Use MATLAB's *plotconfusion* function to display the results of your SVM-based classifier's predictions. The test labels and classifier predictions are the inputs of this function. Report your confusion matrix. Discuss

[5] `https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf`
[6] `https://scikit-learn.org/stable/modules/svm.html`

the results across different classes. Figure 4 displays a sample confusion matrix from the results of an SVM classification using HoG features.

3 Classification using Deep Learning

Deep learning-based methods are data-driven: they rely on a large number of samples from which patterns from the data are extracted and eventually recognized. Among the various sub-fields of deep learning, the *image classification* one uses Convolutional Neural Netowrks [7] (CNNs) to extract features from images and classify them into classes using Neural Networks (NNs). We are going to explore the use of a popular CNN architecture called ResNet [8].

The main advantage of deep learning-based systems with respect to pattern recognition methods is the fact that the convolutional kernels used in CNNs can capture complex and comprehensive characteristics of the data in different image compositions, offering a generic and efficient approach for feature extraction and classification.

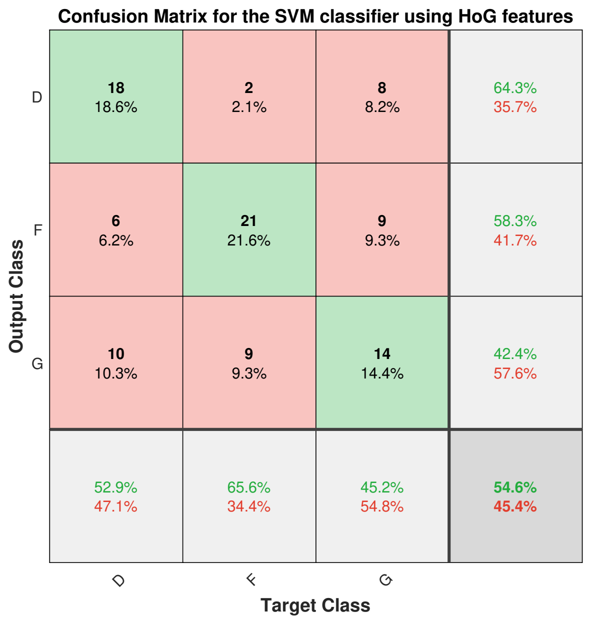


Figure 4: Confusion matrix presenting the results of an SVM-based classifier using HoG features.

Problem 6: Validation Set and Augmentation

[7]`https://cs231n.github.io/convolutional-networks/`
[8]`https://arxiv.org/abs/1512.03385`

In order to analyze the progression in performance while the classifier is trained, a common practice in deep learning is to create a *validation* set. During set intervals in the training, this set is going to be evaluated by the in-training network. However, the loss it generates will not be used to update the parameters of the network.

Another typical data preparation step is to do *data augmentation* on the training set. This generates samples that are slightly different from those on the training set, but still valid to its labels (e.g., cars that are different from those in the original images, but should still be classified as cars). This process ensures that the network can generalize better to previously unseen samples.

(a) (Optional) (**Bonus 0.17 points**): Use MATLAB's *splitEachLabel* function to **randomly** divide the training data into two new sets: *imdsTrain* and *imdsValid*. The new training set must contain 70% of the old, while the validation set must hold the remaining 30%. Report the new number of training and validation samples.

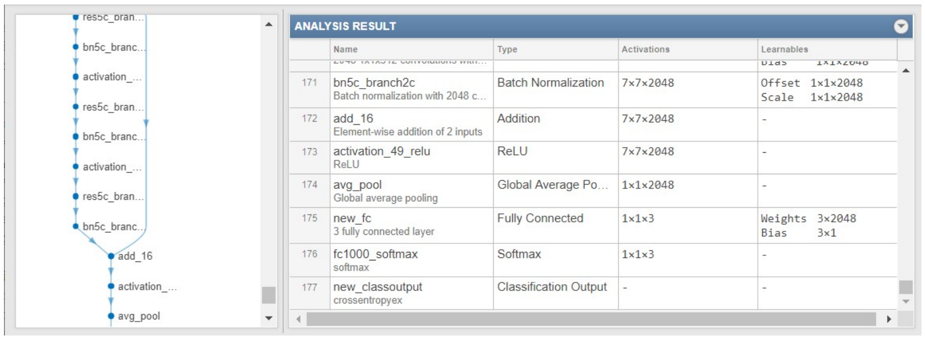
(b) (Optional) (**Bonus 0.17 points**): In order to augment the new training data, create an *imageDataAugmenter* element in MATLAB that adds a random reflection along the X-axis of the mammograms (so that mammograms from both breast orientations can be correctly classified). Then, use MATLAB's *augmentedImageDatastore* function to create a new ImageDatastore that uses the *imageDataAugmenter* element to create training samples with random X-axis reflections. Refer to the 'solutionTemp.m' file for a template of this step.

Problem 7: Creating, Modifying and Training a ResNet-50

ResNet is a CNN originally trained to classify an image in one of a thousand classes (e.g., goldfish, tree frog, desk). Since we only have three classes, the ResNet architecture has to be changed. In particular, the last two layers, 'fully-connected' and 'classification', need to adjusted to the reduced number of classes.

(a) (Optional) (**Bonus 0.5 point**): Create a resnet50 network. Then, create two new layers called 'new_fc' and 'new_classoutput', and substitute the last two fully-connected and classification layers from the ResNet-50 by them. You should use MATLAB's *layerGraph*, *fullyConnectedLayer*, *classificationLayer* and *replaceLayer* functions to do that. You network should have the last three layers as those illustrated in Figure 5.

(b) (Optional) (**Bonus 0.33 points**): Use the 'trainingoptions' object provided in the 'solutionTemp.m' file to train the network. You will use MATLAB's *trainNetwork* function to train it, while providing the augmented imageDatastore, modified Resnet-50 network and training options as inputs. Report the training curves (accuracy and loss) provided by the *trainNetwork* function. Figure 6 shows the accuracy and loss curves for a training process. Note that the accuracy of the training set is higher, indicating that the network is overfitting (i.e., "learning" how to identify the training set very well, while becoming less general).



ANALYSIS RESULT

	Name	Type	Activations	Learnables
171	bn5c_branch2c Batch normalization with 2048 c...	Batch Normalization	7×7×2048	Offset 1×1×2048 Scale 1×1×2048
172	add_16 Element-wise addition of 2 inputs	Addition	7×7×2048	-
173	activation_49_relu ReLU	ReLU	7×7×2048	-
174	avg_pool Global average pooling	Global Average Po...	1×1×2048	-
175	new_fc 3 fully connected layer	Fully Connected	1×1×3	Weights 3×2048 Bias 3×1
176	fc1000_softmax softmax	Softmax	1×1×3	-
177	new_classoutput crossentropyex	Classification Output	-	-

Figure 5: Last layers of the modified ResNet-50 to perform classification between three classes.

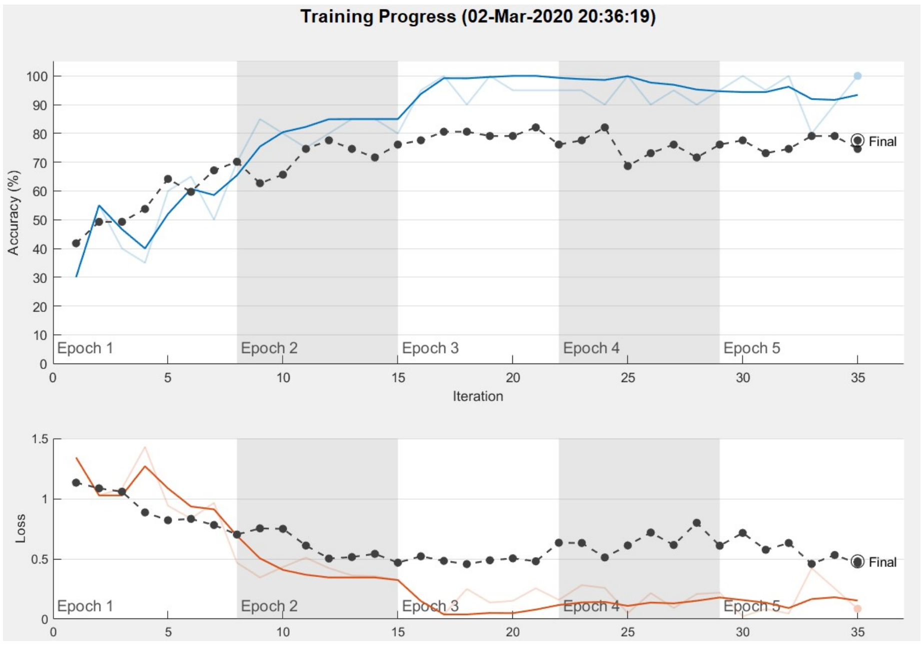


Figure 6: **Top:** Training (continuous) and validation (dotted) accuracy curves. **Bottom:** Training (continuous) and validation (dotted) loss curves.

(c) (Optional) (**Bonus 0.33 points**): Use MATLAB's *classify* and *plotconfusion* functions to evaluate and present the results of your deep learning-based classifier on the **test** set. Report this confusion matrix. Is the ResNet-50 result better or worst than the SVM-based one? Discuss. Figure 7 shows a confusion matrix generated with the results of a ResNet-50 classification of mammograms.

(d) (Optional) (**Bonus 0.5 points**): Change the hyper-parameters of the 'trainingoptions' elements, as well as the preprocesing steps, to achieve an accuracy on the testing set higher than 80%.

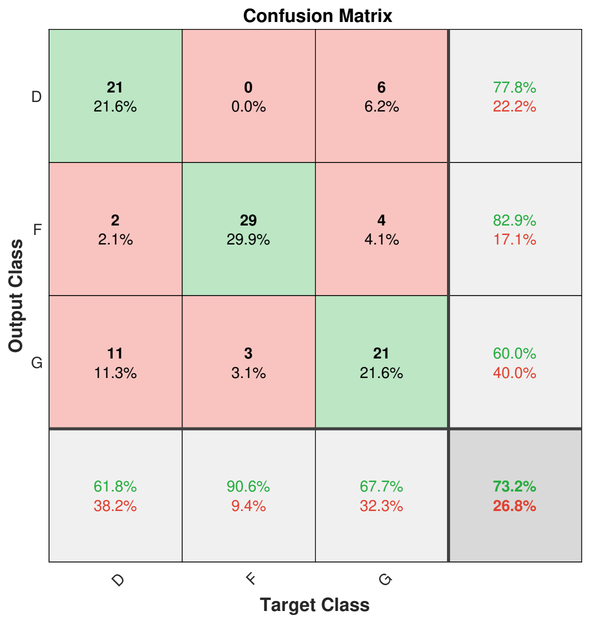


Figure 7: Confusion matrix of the deep learning-based (i.e., ResNet-50) classification of mammograms from the MiniMammographic Database.

Deliverables: When submitting your assignment, use the following format for the name of the folder possessing all the required files:
'FirstName_VNumber_Assignment4.zip'. For example,
'Claire_V00888888_Assignment4.zip'.
The folder should include:

- A '.pdf' file with your written report, which includes the results and discussion for the following questions: Problem 3 (a), Problem 4 (a), Problem 5 (c), and, optionally, Problem 6 (a), Problem 7 (b), (c) and (d).
- solution.m (your modified version of 'solutionTemp.m', which includes your code for all the questions)