Capstone Project 2: Detecting Breast Cancer in Histopathological Images

Context

After lung cancer, breast cancer accounts for the second highest mortality rate in women, and invasive ductal carcinoma (IDC) is the most common form of breast cancer. IDC is Invasive Ductal Carcinoma; cancer that develops in a milk duct and invades the fibrous or fatty breast tissue outside the duct; it is the most common form of breast cancer forming 80% of all breast cancer diagnoses. In 2020, an estimated 276,480 new cases of invasive breast cancer are expected to be diagnosed in women in United States, along with 48,530 new cases of non-invasive (in situ) breast cancer. In women under 45, breast cancer is more common in African-American women than white women. Overall, African-American women are more likely to die of breast cancer. (American Cancer Society, 2020)[1].

To assign an aggressiveness grade to a whole mount sample, pathologists typically focus on the regions which contain the IDC. As a result, one of the common pre-processing steps for automatic aggressiveness grading is to delineate the exact regions of IDC inside of a whole mount slide. Accurately identifying and categorizing breast cancer subtypes is an important clinical task, and automated methods can be used to save time and reduce error. Early detection of cancer cases can give patients more treatment options.

Deep learning algorithms have the potential for being the unifying approach for the many tasks in digital pathology (digitized to produce high-resolution images), having previously been shown to produce state-of-the-art results across varied domains, including mitosis detection2 3 4, tissue classification5, and immunohistochemical staining. Deep learning techniques are already becoming transformational force in detecting breast cancer in histopathological images. Histology is the study of the microscopic structure of tissues.

Objective

In this project, we'll build a classifier to train on 80% of a breast cancer histology image dataset. Of this, we'll keep 10% of the data for validation. Using PyTorch, we'll define a CNN (Convolutional Neural Network), call it CancerNet, and train it on our images. We'll then derive a confusion matrix to analyze the performance of the model to detect IDC breast cancer in histopathological images. Two classes will be detected, 0 and 1: 0 denotes absence of IDC and 1 denotes presence of IDC.

Dataset

We'll use the breast cancer histology image dataset (the IDC_regular dataset) from Kaggle. This dataset holds 2,77,524 patches of size 50×50 extracted from 162 whole mount slide images of breast cancer specimens scanned at 40x. Of these, 1,98,738 test negative and 78,786 test positive with IDC. The dataset is available in public domain and you can download it here. You'll need a

[1] American Cancer Society. Cancer Facts & Figures 2020. Available at: https://www.cancer.org/content/dam/cancer-org/research/cancer-facts-and-statistics/annual-cancer-facts-and-figures/2020/cancer-facts-and-figures-2020.pdf.

[2] 6. Veta M, van Diest PJ, Willems SM, Wang H, Madabhushi A, Cruz-Roa A, et al. Assessment of algorithms for mitosis detection in breast cancer histopathology images. Med Image Anal. 2015;20:237– 48. [PubMed: 25547073]

[3] Roux L, Racoceanu D, Loménie N, Kulikova M, Irshad H, Klossa J, et al. Mitosis detection in breast cancer histological images An ICPR 2012 contest. J Pathol Inform. 2013;4:8. [PMCID: PMC3709417] [PubMed: 23858383]

[4] Ciresan DC, Giusti A, Gambardella LM, Schmidhuber J. Mitosis detection in breast cancer histology images with deep neural networks. Med Image Comput Comput Assist Interv. 2013;16(Pt 2):411–8. [PubMed: 24579167]

[5] Cruz-Roa A, Basavanhally A, González F, Gilmore H, Feldman M, Ganesan S, et al. Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks. SPIE Medical Imaging. 2014;9041:904103-904103-15.

minimum of 3.02GB of disk space for this. Filenames in this dataset look like this: 8863_idx5_x451_y1451_class0. Here, 8863_idx5 is the patient ID, 451 and 1451 are the x- and y-coordinates of the crop, and 0 is the class label (0 denotes absence of IDC).

Approach

First, we will declare the path to the input dataset, that for the new directory, and the paths for the training, validation, and testing directories using the base path. We will also declare that 80% of the entire dataset will be used for training, and of that, 10% will be used for validation.

We'll import from config, imutils, random, shutil, and os. We'll build a list of original paths to the images, then shuffle the list. Then, we calculate an index by multiplying the length of this list by 0.8 so we can slice this list to get sublists for the training and testing datasets. Next, we further calculate an index saving 10% of the list for the training dataset for validation and keeping the rest for training itself. Now, datasets is a list with tuples for information about the training, validation, and testing sets. These hold the paths and the base path for each.

For each setType, path, and base path in this list, we'll print, say, 'Building testing set'. If the base path does not exist, we'll create the directory. And for each path in originalPaths, we'll extract the filename and the class label. We'll build the path to the label directory (0 or 1)- if it doesn't exist yet, we'll explicitly create this directory. Now, we'll build the path to the resulting image and copy the image here- where it belongs.

We use the Sequential API to build CancerNet and SeparableConv2D to implement depthwise convolutions. The class CancerNet has a static method build that takes four parameters- width and height of the image, its depth (the number of color channels in each image), and the number of classes the network will predict between, which, for us, is 2 (0 and 1).

We will define three Depthwise Convolution (DEPTHWISE_CONV) => activation function (RELU) => POOL layers; each with a higher stacking and a greater number of filters. The softmax classifier outputs prediction percentages for each class. In the end, we return the model.

We set initial values for the number of epochs, the learning rate, and the batch size. We'll get the number of paths in the three directories for training, validation, and testing. Then, we'll get the class weight for the training data so we can deal with the imbalance.

We will initialize the training data augmentation object. This is a process of regularization that helps generalize the model. This is where we slightly modify the training examples to avoid the need for more training data. We'll initialize the validation and testing data augmentation objects.

We will initialize the training, validation, and testing generators so they can generate batches of images of size batch_size. Then, we'll initialize the model using the Adagrad optimizer and compile it with a binary_crossentropy loss function. To fit the model, we make a call to fit_generator().

After successfully trained our model, we will evaluate the model on our testing data. We'll reset the generator and make predictions on the data. Then, for images from the testing set (20% of the dataset), we get the indices of the labels with the corresponding largest predicted probability. And we'll display a classification report.

We'll compute the confusion matrix and get the raw accuracy, specificity, and sensitivity, and display all values. Finally, we'll plot the training loss and accuracy.

Deliverable

The deliverable from this project will be a final paper, a slide deck and codes. All will be posted on github.