1. Pre-Processing

a. Data set:

- The original dataset consisted of 162 whole-mount slide images of Breast Cancer (BCa) specimens scanned at 40x.
- From that, **277,524 patches** of size 50 x 50 were extracted (**198,738 IDC negative** and **78,786 IDC positive**).
- Each patch's file name is of the format: u_xX_yY_classC.png — > example 10253_idx5_x1351_y1101_class0.png.
    - Where u is the patient ID (10253_idx5), X is the x-coordinate of where this patch was cropped from, Y is the y-coordinate of where this patch was cropped from, and C indicates the class where 0 is **non-IDC** and 1 is **IDC**.

```
breast_img = glob.glob('data_set/IDC_regular_ps50_idx5/**/*.png', recursive = True)

for imgname in breast_img[:3]:
    print(imgname)
```

```
data_set/IDC_regular_ps50_idx5\10253\0\10253_idx5_x1001_y1001_class0.png
data_set/IDC_regular_ps50_idx5\10253\0\10253_idx5_x1001_y1051_class0.png
data_set/IDC_regular_ps50_idx5\10253\0\10253_idx5_x1001_y1101_class0.png
```

b. Labeling data:

- We plot the image using the Keras utils library to visualize the data for positive and negative images of Breast cancer.
- I initialized two arrays, one for non-cancerous images and another for cancerous images. The code then reads and resizes each image to a fixed size of 50x50 pixels, assigning a label of 0 for non-cancerous and 1 for cancerous. This process prepares the data for training or evaluating a machine learning model designed to classify medical images into these two categories.

c. Data Splitting – we used the in-build function of the sklearn library (train_test_split) to separate features(X) and the target variable (y) into training/testing in the ratio of 80/20 with a random state of 42.

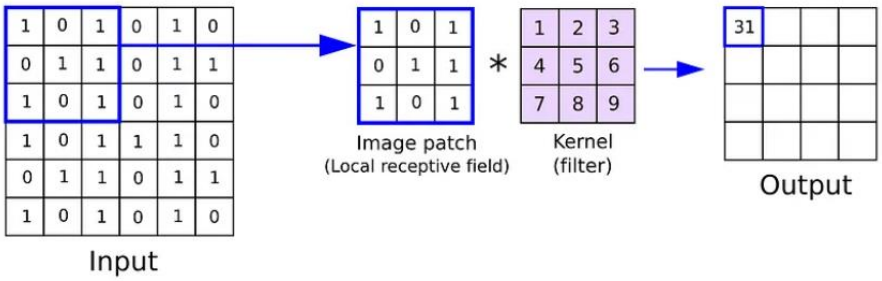
2. Machine Learning Methods: Convolutional Neural Network(CNN):

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 50, 50, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 25, 25, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 25, 25, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 25, 25, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 128)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 6, 6, 128)         147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 3, 3, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense (Dense)                (None, 128)               147584
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 258
=================================================================
Total params: 388,674
Trainable params: 388,674
Non-trainable params: 0
```

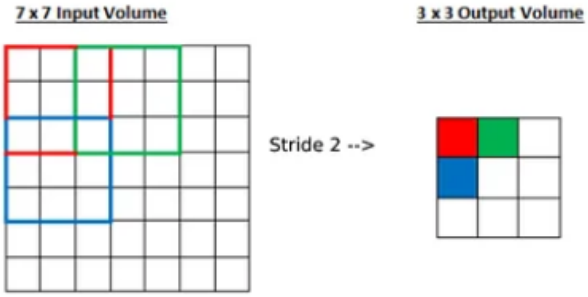
CNN image classifications take an input image, process it, and classify it under certain categories. Computers see an input image as an array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d(h = Height, w = Width, d = Dimension). E.g., An image of a 6 x 6 x 3 array of the matrix of RGB (3 refers to RGB values) and an image of a 4 x 4 x 1 array of matrix of grayscale image.

a. Convolution Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as an image matrix (50,50,3) and a filter or kernel(3,3).



b. **Strides:**

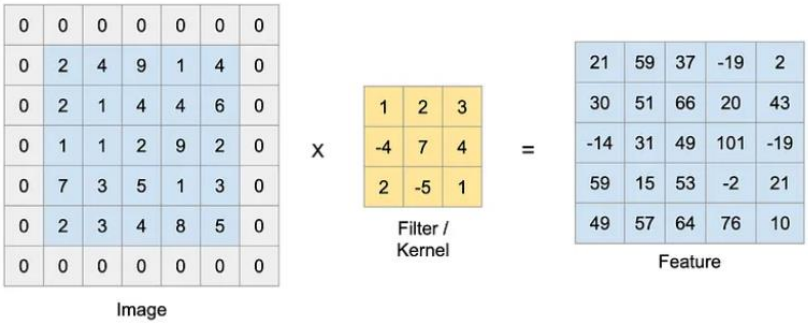


Stride is the number of pixels that shift over the input matrix. When the stride is 1, then we move the filters to 1 pixel at a time.

When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

c. **Padding** :

we are using the "**same**" hyperparameter in the CNN using Karas Library.



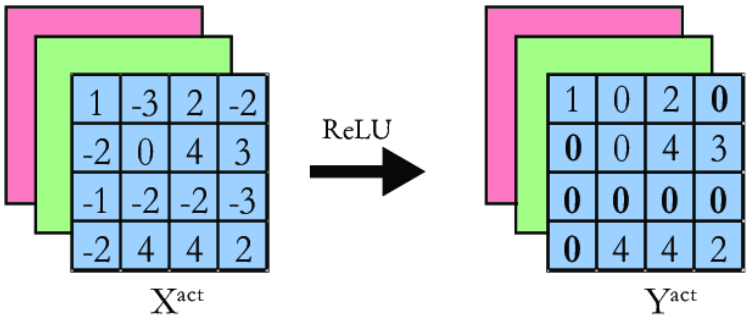
.

When we use the filter or Kernel to scan the image, the size of the image will go smaller.

We have to avoid that because we want to preserve the original size of the image to extract some low-level features.

Therefore, we will add some extra pixels outside the image.

d. Non-Linearity (ReLU): activation function



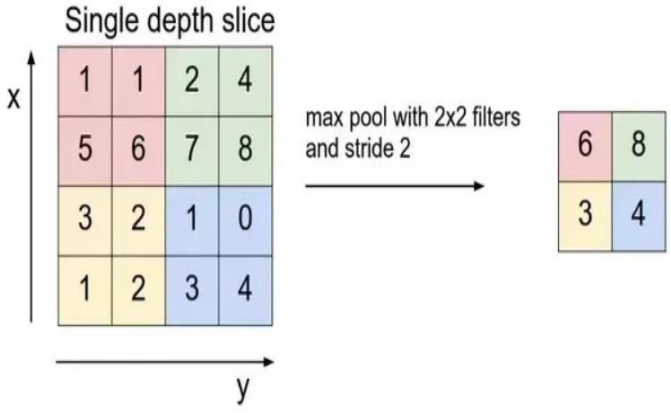
ReLU stands for **Rectified Linear Unit** for a non-linear operation. The output is.

$f(x) = \max(0, x)$.

Why ReLU is important: ReLU's purpose is to introduce non-linearity in our **ConvNet.**

e. Pooling Layer:

The pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling is also called subsampling or downsampling which reduces the dimensionality of each map but retains important information. Here, we use Max pooling with 2*2 Matrix.

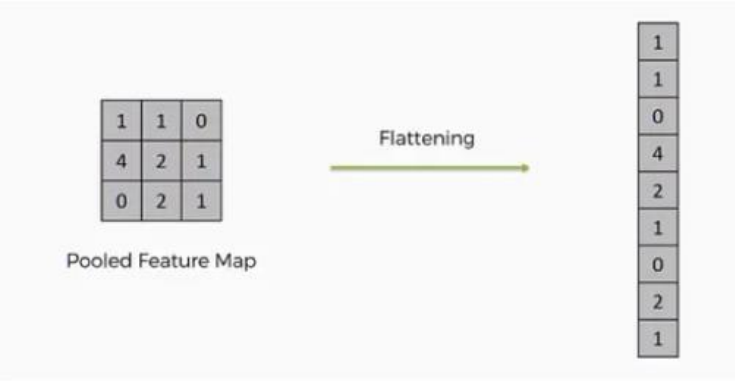


Spatial pooling can be of different types: Max Pooling, Average Pooling, Sum Pooling.

Max pooling takes the largest element from the rectified feature map.

Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling

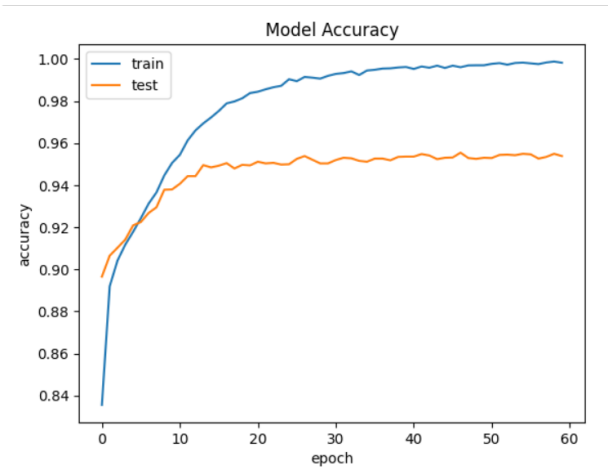
f. Flatten

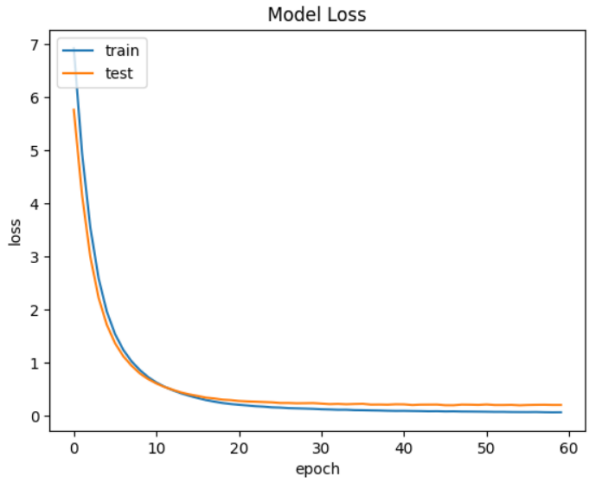


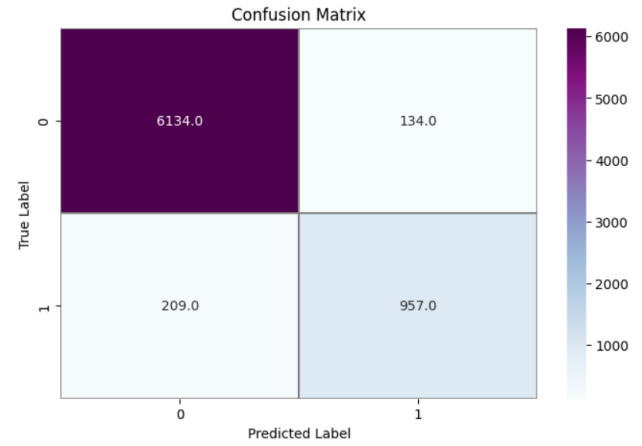
Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector. The flattened matrix is fed as input to the fully connected layer to classify the image.

3. Results

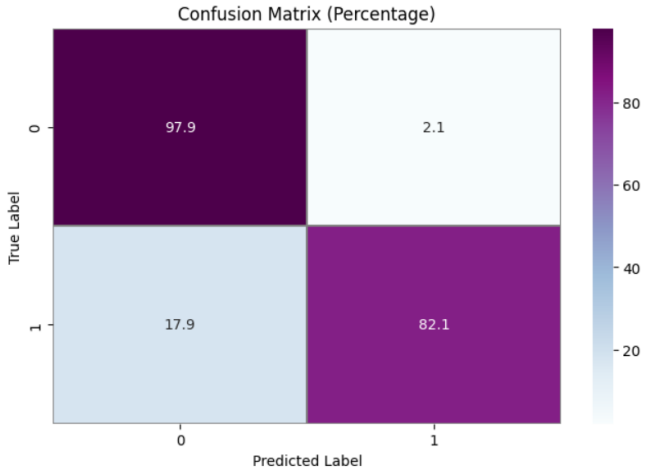
From the graph depicted below, it is evident that our training accuracy reaches 98%, while our testing accuracy attains a commendable 94%. Furthermore, the error rate remains minimal at 0.2.







Upon examining the confusion matrix, it becomes apparent that during the training phase, we encounter a mere 315 instances of misclassifications—comprising both false negatives and false positives—within a dataset of approximately 8000 breast cancer cases.



Similarly, in the testing phase, only 4 misclassifications are identified among nearly 200 breast cancer datasets.

4. Thoughts for future work about how the model can be improved.

- Experiment with different CNN architectures {"Adjust the depth (number of layers) and width (number of filters) of the network"}
- Fine-tuning model hyperparameters will help to improve the accuracy.
- Transfer learning can be a powerful technique, especially when working with limited labeled data.
- Optimize learning rates, batch sizes, and the number of epochs. Additionally, consider adjusting the dropout rates in fully connected layers to prevent overfitting.