Project Development phase

No. Of Functional Features Included In The Solution

DATE	06 May 2023
Team ID	NM2023TMID18418
Project Name	CancerVision: Advanced Breast Cancer Prediction with Deep Learning

Converting a patch classifier to an end-to-end trainable whole image classifier using an all convolutional design. The function *f* was first trained on patches and then refined on whole images. We evaluated whether removing the heatmap improved information flow from the bottom layers of the patch classifier to the top convolutional layers in the whole image classifier. The magnifying glass shows an enlarged version of the heatmap. This figure is best viewed in color.

The function *h* accepts whole images as input and produces labels at the whole image level. Therefore, it is end-to-end trainable, providing two advantages over the two-step approach. First, the entire network can be jointly trained, avoiding sub-optimal solutions from each step; Second, the trained network can be transferred to another dataset without explicit reliance on ROI annotations

Large mammography databases with ROI annotations are rare and expensive to obtain. The largest public database with ROI annotations for digitized film mammograms – DDSM[37] – contains several thousand images with pixel-level annotations, which can be exploited to train a patch classifier *f*. Once the patch classifier is converted into a whole image classifier *h*, it can be fine-tuned on other databases using only image-level labels. This approach allows us to significantly reduce the requirement for ROI annotations, and has many applications in medical imaging in addition to breast cancer detection on screening mammograms.

Network design :

A modern CNN is typically constructed by stacking convolutional layers on top of the input, followed by one or more fully connected (FC) layers to join with the classification output. Max pooling layers are often used amid convolutional layers to improve translational invariance and to reduce feature map size. In this study, two popular CNN structures are compared: the VGG network[38] and the residual network (Resnet)[39]. Consecutive network layers can be naturally grouped into "blocks" so that the feature map size is reduced (typically by a factor of 2) either at the beginning or at the end of a block but stays the same elsewhere in the block. For example, a "VGG block" is a stack of several 3 × 3 convolutional layers with the same depth followed by a 2 × 2 max pooling layer that reduces the feature map size by a factor of 2. Although other filter sizes can be used, 3 × 3 convolution and 2 × 2 max pooling are widely used, and employed throughout this study unless otherwise stated. Therefore, a VGG block can be represented by the pattern of $N \times K$, where N represents the depth of each convolutional layer and K represents the number of convolutional layers. A "Resnet block" uses stride = 2 in the first convolutional layer instead of 2 × 2 max pooling to reduce feature map size at the beginning of the block, followed by the stacking of several convolutional layers. We use the "bottleneck design[39]" which consists of repeated units of three convolutional layers that have filter sizes of 1 × 1, 3 × 3 and 1 × 1, respectively. A key feature of the Resnet block is that a shortcut is made between the two ends of each unit so that the features are directly carried over and therefore each unit can focus on learning the "residual" information[39]. Batch normalization (BN) is used in every convolutional layer in the Resnet, which is known to speedup convergence and also has a regularization effect[40]. A Resnet block can be represented by the pattern of $[L - M - N] \times K$, where L, M and N

represent the depths of the three convolutional layers in a unit and *K* represents the number of units. Here, the 16-layer VGG network (VGG16) and the 50-layer Resnet (Resnet50) are used as patch classifiers. The original design of the VGG16[38] consisted of five VGG blocks followed by two FC layers. To be consistent with the Resnet50, we replaced the two FC layers with a global average pooling layer which calculates the average activation of each feature map for the output of the last VGG block. For example, if the output of the last VGG block has a size of 7 × 7 × 512 (height × width × channel), after the global average pooling layer the output becomes 512. This output is then connected to the classification output with a FC layer.

A straightforward approach to construct a whole image classifier from a patch classifier involves flattening the heatmap and connecting it to the image's classification output using FC layers. To increase the model's translational invariance to the patch classifier's output, a max pooling layer can be used after the heatmap. Further, a shortcut can be made between the heatmap and the output to make the training easier. The heatmap results directly from the patch classifier's output which uses the softmax activation:

f(z)j=ezjΣci=1eziforj=1,…,c

(1)

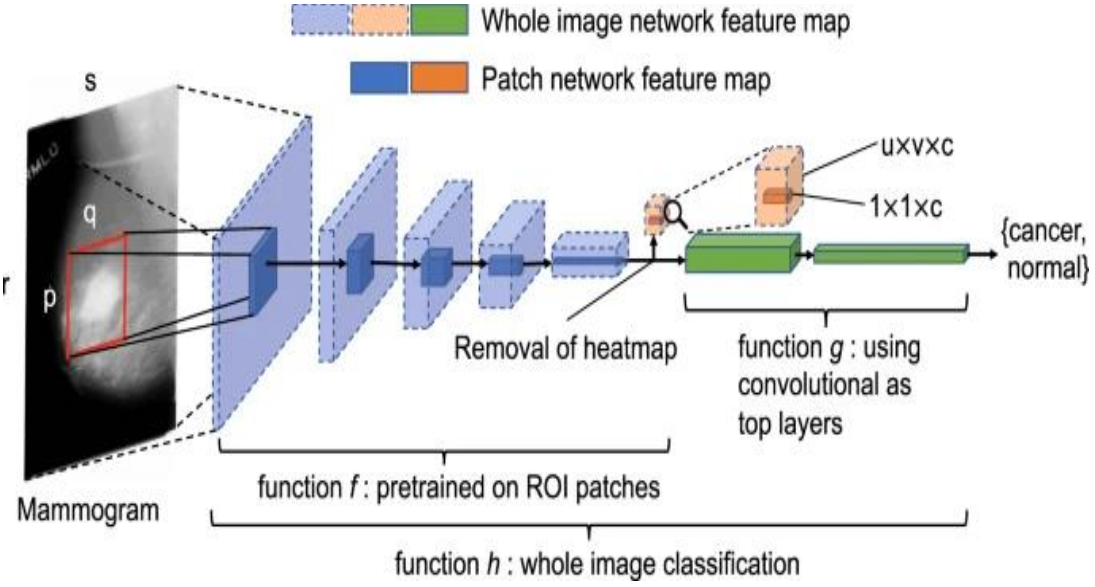
However, the softmax activation diminishes gradients for large inputs, which may impede gradient flow when it is used in an intermediate layer. Therefore, the rectified linear units (ReLU) can be used instead:

f(z)j=max(0,zj)forj=1,…,c

(2)

In the following, when we refer to the heatmap in a whole image classifier, the activation is always assumed to be ReLU unless otherwise stated.

We further propose to use convolutional layers as top layers, which preserve spatial information. Two blocks of convolutional layers (VGG or Resnet) can be added on top of the patch classifier layers, followed by a global average pooling layer and then the image's classification output (Fig. 1). Therefore, this design creates an "all convolutional" network for whole image classification. As Fig. 1 shows, the heatmap abruptly reduces the depth of the feature map between the patch classifier layers and the top layers, which may cause information loss in the whole image classification. Therefore, we also evaluated the results when the heatmap is removed entirely from the whole image classifier to allow the top layers to fully utilize the features extracted from the patch classifier.



Training a whole image classifier was achieved in two steps. The first step was to train a patch classifier. We compared the networks with pre-trained weights using the ImageNet[32] database to those with

randomly initialized weights. In a pre-trained network, the bottom layers represent primitive features that tend to be preserved across different tasks, whereas the top layers represent higher-order features that are more related to specific tasks and require further training. Using the same learning rate for all layers may destroy the features that were learned in the bottom layers. To prevent this, a 3-stage training strategy was employed in which the parameter learning is frozen for all but the final layer and progressively unfrozen from the top to the bottom layers, while simultaneously decreasing the learning rate. The 3-stage training strategy on the S10 patch set was as follows:

- 1.
- Set learning rate to 10^{-3} and train the last layer for 3 epochs.
- 2.
- Set learning rate to 10^{-4}, unfreeze the top layers and train for 10 epochs, where the top layer number is set to 46 for Resnet50 and 11 for VGG16.
- 3.
- Set learning rate to 10^{-5}, unfreeze all layers and train for 37 epochs for a total of 50 epochs.

In the above, an epoch was defined as a sweep through the training set. For the S1 patch dataset, the total number of epochs was increased to 200 because it was much smaller and less redundant than the S10 patch dataset. For randomly initialized networks a constant learning rate of 10^{-3} was used. Adam[42] was used as the optimizer and the batch size was set to be 32. The sample weights were adjusted within each batch to balance the five classes.

The second step was to train a whole image classifier converted from the patch classifier (Fig. 1). A 2-stage training strategy was employed to first train the newly added top layers (i.e. function *g*) and then train all layers (i.e. function *h*) with a reduced learning rate, which was as follows:

- 1.
- Set learning rate to 10^{-4}, weight decay to 0.001 and train the newly added top layers for 30 epochs.
- 2.
- Set learning rate to 10^{-5}, weight decay to 0.01 and train all layers for 20 epochs for a total of 50 epochs.

We found that the VGG-based image classifiers showed sign of continuing improvement towards the end of the 50 epochs, while the Resnet-based image classifiers had already converged. To be fair for the VGG-based image classifiers, we continued to train them with 200 additional epochs. Due to GPU memory limits, a batch size of 2 was used.

The average gray scale value of the whole image training set was subtracted from both patch and whole image datasets in training. No other preprocessing was applied. To improve the generalization of final models, data augmentation was performed using the following random transformations: horizontal and vertical flips, rotation in [−25, 25] degrees, zoom in [0.8, 1.2] ratio and intensity shift in [−20, 20] pixel values.