Computer Vision for Breast Cancer Detection: Image Classification and Segmentation

Machine Learning and Artificial Intelligence,
Università Bocconi

Cesare Bergossi, Riccardo Carollo, Emilija Milanovic, Elia Parolari, Giulia Pezzani

May 2024

Abstract

According to the World Health Organization (WHO), breast cancer remains the most prevalent cancer among women in 157 countries. Early detection is key for improving treatment outcomes; therefore, it is crucial to keep working on advancing diagnostic techniques.
Our project aims to develop a machine learning framework that integrates both image segmentation and classification to enhance the accuracy of breast cancer detection.
Firstly, we will perform classification through a Convolutional Neural Network (CNN), specifically the ResNet architecture, to determine the presence of cancer and its type (normal, benign or malignant).
Secondly, we will build a segmentation model using the U-Net architecture with extra attention layers, which is highly effective for biomedical image segmentation. This model will identify cancerous tissues within scans.
The project will also include the development of a user-friendly interface that allows medical practitioners to easily upload scans and receive immediate diagnostic feedback: the eventual cancer type and the corresponding segmentation.

Contents

1 About the Data

The dataset includes 780 grayscale images of breast cancer, obtained from Baheya hospital, and have been downlaoded via Kaggle (original source: [1]). These images are divided into three categories: normal, benign, and malignant (see figure 1). Originally, 1100 images were collected, but after preprocessing, the dataset was reduced to 780 images. The images were captured over the course of about a year. All images were cropped to different sizes to remove unused and unimportant boundaries from the images. The images in the benign and malignant groups come with freehand segmentation masks made by radiologists at Baheya hospital.

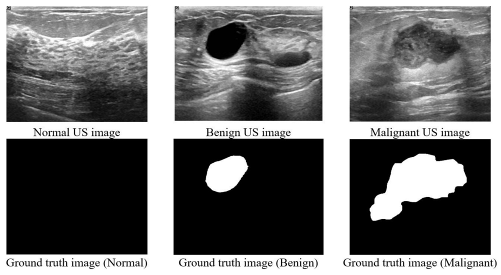
Normal US image
Benign US image
Malignant US image
Ground truth image (Normal)
Ground truth image (Benign)
Ground truth image (Malignant)

Figure 1: Examples of image and corresponding mask for each of the three classes.

Case	Number of images
Benign	487
Malignant	210
Normal	133
Total	780

Table 1: The three classes of breast tumor cases and the number of images in each case.

2 Classification

2.1 Transfer Learning with ResNet18

For our classification tasks, we have chosen to employ a pre-trained Residual Neural Network (ResNet) model. ResNets are a type of Convolutional Neural Network (CNN) widely utilized for complex challenges in fields such as computer vision and are notably pre-trained on the vast ImageNet dataset.

One significant advantage of ResNet over traditional CNN models is its ability to mitigate the vanishing gradient problem (VGP). Unlike earlier models such as VGG, ResNet improves performance by effectively addressing VGP, which can occur during training when gradients become too small to influence updates. This results in the initial layers of the network remaining largely unchanged, thereby slowing down the learning process. The vanishing gradient issue can adversely affect the learning capacity, slow down convergence, and hinder the network's ability to learn long-term dependencies. It is important to note that while ResNet was not explicitly designed to solve the VGP, the inclusion of residual connections inherently addresses this issue, leading to improved learning and better performance compared to networks without these connections, as highlighted in the original ResNet paper [2].

The key innovation of ResNet is the introduction of residual connections, also known as skip connections or shortcut connections. These connections add a direct path for the gradient to flow through, bypassing one or more layers. Specifically, in ResNet, a shortcut connection is added to each pair of 3×3 convolutional layers. This allows the network to learn the residual mapping instead of the original mapping, facilitating easier optimization and improving performance.

Specifically, we chose the pre-trained ResNet18 model, a form of Residual Neural Network which, as the name suggests, consists of 18 layers, including both convolutional layers and identity shortcuts. The reason why we employed this architecture over other pre-trained Residual Networks is its performance: after trying ResNet36 as well as ResNet50, we found that the one with 18 hidden layers yields the best results.

2.2 Customizing ResNet18 for Additional Mask Channel

In our project, we adapted the ResNet18 architecture to include a mask as an additional input channel, enhancing the network's ability to focus on relevant regions of the medical scans.

To incorporate the mask into our input data, we created a custom dataset class, ScanWithMaskDataset. This class is designed to load both the original images and their corresponding masks, stacking them together as input channels. In this class:

- The __init__ method initializes the dataset, loading the images and their corresponding masks.
- The _load_samples method generates a list of tuples containing paths to the images and their masks along with the class index.

- The __getitem__ method loads the image and mask, applies transformations, and stacks them as separate channels. The mask is converted to a single-channel grayscale image.

To handle the additional input channel provided by the mask, we modified the first convolutional layer of the ResNet18 model. This involved changing the layer to accept four input channels (three for RGB and one for the mask). Finally, the last fully connected layer was adjusted to output predictions for three classes: no cancer, benign, and malignant.

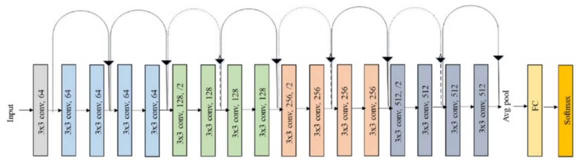

Figure 2: ResNet18 Architecture

2.3 Data Transformation

We decided to implement specific image transformations for both training and testing datasets, to ensure consistency and increase model generalizability.

Training Data Transformations:

- Resizing and Cropping: All images are resized to 256 pixels and then center-cropped to 224 pixels. This ensures consistency in image dimensions across all inputs, which is crucial for the neural network to process them effectively. Images were cropped to 224 pixels because the ResNet18 was trained on images of that size.

- Data Augmentation Techniques:

  - Random Horizontal and Vertical Flips: These transformations randomly flip the images horizontally and vertically, introducing variability and simulating different viewing conditions.
  - Random Rotation: Images are randomly rotated by up to 10 degrees. This augmentation technique helps the model become invariant to small rotations, further enhancing its robustness.

  Data augmentation helps prevent the model from overfitting to the training data by introducing variability and improving its robustness and performance on unseen images.

Testing Data Transformations:

- Resizing and Center Cropping: Similar to the training transformations, all images are resized to 256 pixels and center-cropped to 224 pixels. This ensures that each image is evaluated in a uniform manner, matching the input size used during training.

- No Random Augmentations: To maintain consistency and control, no random augmentations are applied to the testing data. This helps in accurately assessing the model's performance without introducing any additional variability.

2.4 Model Training

Effective model training requires the careful selection of various components, including the loss function, optimizer, and learning rate scheduler. The specific values for these components were chosen based on hyperparameter tuning to achieve the best results for our classification task.

Loss Function: We utilized Cross Entropy Loss as our loss function, which is well-suited for multi-class classification tasks. This function computes the difference between the predicted probabilities and the true class labels, providing a measure of how well the model's predictions match the actual distribution.

Optimizer: We selected Stochastic Gradient Descent (SGD) with specific parameters to optimize the model effectively:

- Learning Rate (0.001): Controls the magnitude of updates to the model's weights in response to the error each time they are updated.

- Momentum (0.9): Accelerates the optimizer in the direction of the gradients, leading to faster convergence by incorporating the 'velocity' of the gradients.

- Nesterov Accelerated Gradient (Nesterov=True): An enhancement over traditional momentum, it computes the gradient at the anticipated position of the parameters, providing a lookahead mechanism.

- Weight Decay (0.00001): Adds a regularization term to the loss function to prevent overfitting by penalizing large weights.

Learning Rate Scheduler: To manage the learning rate dynamically during training, we employed the StepLR scheduler. This scheduler reduces the learning rate at specified intervals, which helps in fine-tuning the model as training progresses:

- Step Size (10): The interval (in epochs) at which the learning rate is adjusted.
- Gamma (0.1): The factor by which the learning rate is multiplied after each step size interval. Reducing the learning rate helps in achieving more precise convergence.

The training process involves multiple iterations (epochs) where the model learns from the training data and its performance is evaluated on the validation data.

1. Forward Pass: For each batch of images and labels, the images are passed through the model to obtain predictions.
2. Compute Loss
3. Backpropagation: Gradients are calculated by performing a backward pass. The gradients are then used to update the model's weights.
4. Adjust Learning Rate
5. Validation: Every few epochs, the model's performance is evaluated on the validation dataset to monitor accuracy and adjust training strategies accordingly.

2.5 Model Evaluation

2.5.1 Evaluation Metrics

We proceeded by evaluating the model's performance on unseen data, which gives a more realistic insight into the effectiveness of the model. We utilized various metrics and visualizations to comprehensively assess our model.

Predictions were made over the test set to compute key metrics, including loss, accuracy, precision, recall, and F1 score. These metrics provide a detailed overview of the model's performance:

- Test Loss: Measures the average loss over the test dataset, indicating how well the model's predictions align with the true labels.
- Accuracy: The proportion of correctly classified instances out of the total instances, providing a straightforward measure of performance.
- Precision: The ratio of true positive predictions to the total predicted positives, indicating the model's ability to avoid false positives.
- Recall (Sensitivity): The ratio of true positive predictions to the total actual positives, showing the model's ability to capture all relevant instances.
- F1 Score: The harmonic mean of precision and recall, providing a balanced measure of the model's performance, especially in cases of class imbalance.

2.5.2 Confusion Matrix

The confusion matrix provides a detailed breakdown of the model's classification performance across the different classes (benign, malignant, normal). It highlights the number of correct and incorrect predictions, allowing us to identify specific areas where the model may be confusing certain classes.
The confusion matrix shows that the model is highly effective at identifying normal cases with high accuracy. Results for cancerous scans are still impressive. However, some confusion between benign and malignant classes suggests the need for improved feature distinction or model tuning to better differentiate these conditions.

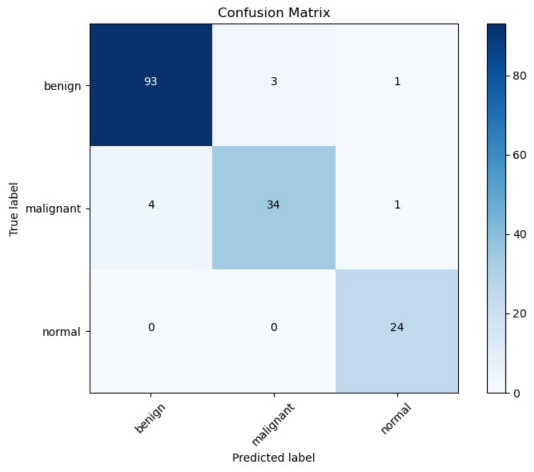

Figure 3: Confusion Matrix

2.5.3 Receiver Operating Characteristic (ROC) Curve

To further evaluate the model performance we used ROC curve (receiver operating characteristic curve). It is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate and False Positive Rate.
As the plot clearly shows, model performs very well. The area under the curve for no cancer class is equal

to one which suggests that our predictions are almost entirely correct. For both malignant and benign classes it is equal to 0.99 which again leads us to the same conclusion.

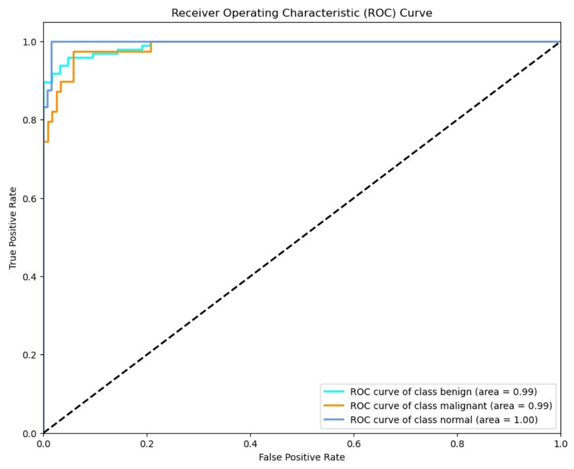

Figure 4: ROC Curve

2.5.4 Implications of False Positives and False Negatives in Medicine

In medical diagnostics, particularly for conditions like breast cancer, the implications of false positives and false negatives are critical. False positives occur when a healthy person is incorrectly identified as having cancer. It is definitely tedious as it might lead to unnecessary additional tests.

On the other hand, false negatives, where a cancer diagnosis is missed, have even more severe consequences, as such diseases require immediate treatment.

Therefore, balancing the rate of false positives and false negatives is crucial. The results of our research managed to achieve such balance.

3 Segmentation

3.1 Initial Considerations

For the segmentation part, we only considered the images in the benign and malignant case (since the one in the normal case do not have a mask). This restricts our dataset to 697 images. With such a small dataset, achieving a satisfactory level of generalization can be challenging. If one were to build a real model, a wiser idea would be to use a pre-trained network, but for the scope of this project we decided to start the training from zero to test the capabilities of the model.

3.2 Attention U-Net

U-Net is a CNN architecture developed in 2015 and is a popular for image segmentation. There are 2 main phases: the encoding and decoding phase. Initially, the encoder processes the input image, extracting useful features through multiple convolutions layers. The decoder then upsamples these features using transpose convolution and integrates them via skip connection. Consequently, the network outputs a segmentation mask. A key aspect of the unit is the skip connection, which, as the name implies, bypasses some layers in the neural network, feeding the output of one layer directly into subsequent layers. The skip connections allow selected features to be directly transferred from the encoder to the decoder, enhancing the decoder's ability to produce an improved segmentation mask.

The difference between the Attention U-Net [3] architecture and U-Net is that the first one integrates Attention Gates (AGs) into the standard U-Net architecture. These AGs allow the model to automatically learn to focus on specific features within a medical image, regardless of their different forms and dimensions, by suppressing irrelevant regions. This integration is achieved with minimal computational overhead.

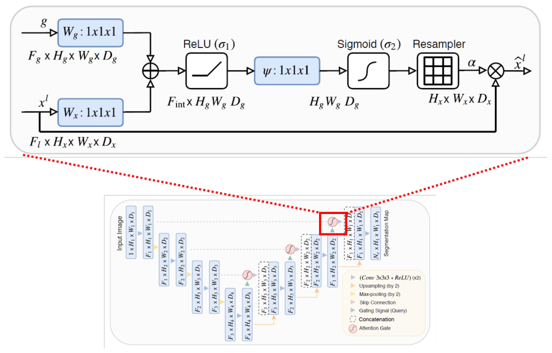

Figure 5: AttU-Net architecture.

3.3 Training

3.3.1 Hyper-Parameters

We display all the parameters used during training in the following table, as well as information on the optimizer, loss, and learning rate schedule. They were obtained through trial and errors (with the help of heuristics and suggestion from the U-Net paper [4] and other notebooks on Kaggle), until we were satisfied with our train and test losses.

3.3.2 Procedure

The loss function used is nn.BCEWithLogitsLoss(). This combines a sigmoid layer and the binary cross-entropy loss in a single class. As for the optimizer, we use optimiser Nesterov Momentum SGD and cosine annealing for learning rate scheduling. The learning rate starts at the initial value (0.1) and gradually reduces to $\eta_{\min}$ (0.00001) over $T_{\max}$ epochs.

Description	Value
Image Width	64 (px)
Image Height	64 (px)
Total Training Epochs	50
Batch Size	32
Loss Function	BCE (Binary Cross Entropy)
Optimizer	SGD
Nesterov Momentum	0.9
Weight Decay	1×10^{-5}
Learning Rate Schedule	Cosine
Initial Learning Rate	0.1
Final Learning Rate	1×10^{-5}

Table 2: Hyper-Parameters Used for Training.

3.3.3 Optimization Results

The plots of the training and validation loss are shown in fig.6. After some initial explorative phase with higher learning rate, the validation and test loss start decreasing together after epoch 20, and a stable situation is reached after epoch 40, with both errors being small.

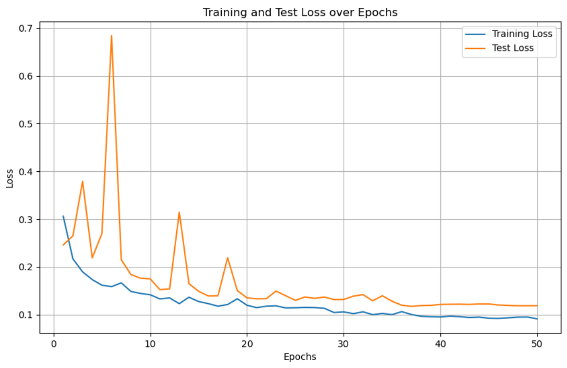

Figure 6: Train and validation loss during training.

3.4 Inference

The network's output is a value in $(0, 1)$ for every pixel, that can be interpreted as the probability of the pixel containing cancer. In order to use the model for inference, we need to determine an appropriate threshold to decide how high the probability should be in order to assign the cancer class. We decided to employ the average IoU on the training set to determine this threshold. The results in fig. 7 show that an optimal value is 0.5.

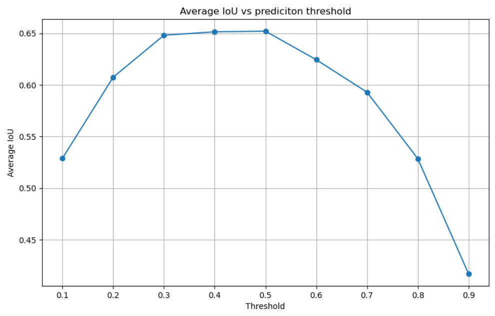

Figure 7: Average IOU for different values of classification threshold parameters.

3.5 Test Set

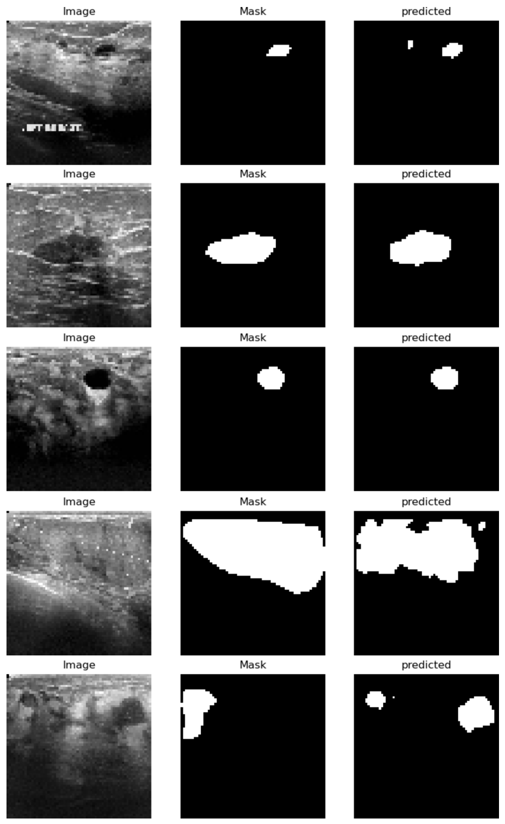

Figure 8: Original vs. Predicted Mask on the Test Dataset.

4 Web Interface for Cancer Detection

In order to make our breast cancer detection and segmentation model accessible and easy to use for healthcare professionals, we developed a user-friendly web interface using Streamlit. This interface allows users to upload medical scans and receive immediate feedback on the presence of cancer and the corresponding segmentation of the affected areas. Below, we describe the key components and functionalities of the interface.

We chose not to use the classification model with the additional mask channel because our segmentation model was trained exclusively on benign and malignant cancer samples, not on non-cancerous ones. Consequently, applying it to a normal scan could result in a mask with unintended white regions, even in the absence of cancer, which would introduce a bias into the classifier.

- Loading Pre-trained Models: Both the segmentation and classification models were loaded from pre-trained weights. These models were trained using PyTorch and saved in a format that allows for easy loading and inference.
- Image Processing: Uploaded images are preprocessed to match the input requirements of the models. This includes resizing the images to the appropriate dimensions and applying necessary transformations, such as converting images to grayscale for segmentation.
- Inference Pipeline: The interface runs the uploaded image through the classification model to predict the cancer type. Then, if the scan is cancerous, the image is sent through the segmentation model, to generate a mask highlighting the tumoral region. The results are then displayed to the user: cancer presence and eventual type (with corresponding confidence level), and its segmentation for easy identification.

5 Limitations and Future Research

Our small dataset limits the generalizability of our model. A larger, more diverse dataset would improve robustness and accuracy. Although data augmentation was used, it can't fully replace real-world diversity. Future research should gather larger datasets from multiple sources to improve generalization across different populations and imaging conditions.

Additionally, incorporating multi-modal data, such as combining mammography images with patient history or genetic information, could enhance cancer detection.

Important remark: At the start of the project the original idea was to first use segmentation to look for general cancerous tissues, and then feed the image with the segmentation mask in a classification model, since this approach would produce more accurate labelling of cancer types. It turned out that training the segmentation model also to segment normal images (i.e. without cancer) drastically diminishes the performance of the model, in particularly it makes it more difficult to achieve generalization. Future research should aim at understanding the best pipeline to use for a practical use of the model, whether to first segment and then classify, or classify first and then apply segmentation. A possible strategy could be to merge the two model in one to produce segmentation masks that contains also the label, which could be particularly effective for images that contains both cancer types.

References

[1] Walid Al-Dhabyani et al. "Dataset of breast ultrasound images". In: *Data in Brief* 28 (2020), p. 104863. ISSN: 2352-3409. DOI: `https://doi.org/10.1016/j.dib.2019.104863`.

[2] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: `1512.03385`. URL: `http://arxiv.org/abs/1512.03385`.

[3] Ozan Oktay et al. "Attention U-Net: Learning Where to Look for the Pancreas". In: *CoRR* abs/1804.03999 (2018). arXiv: `1804.03999`. URL: `http://arxiv.org/abs/1804.03999`.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: `1505.04597 [cs.CV]`.