Breast Cancer Diagnosis Using Custom CNN and Grad-CAM: An Interpretable Deep Learning Approach on Histopathology Images

Santosh Sapkota
santoshsapkota588@gmail.com

July 2025

Abstract

Breast Cancer is the leading cause of cancer-related death in women globally. Invasive Ductal Carcinoma(IDC), the most common subtype, is significantly harder to diagnose due to its complex morphological structure as seen in histopathological images. This project presents a deep learning-based approach using binary classification of IDC to categorize images as malignant or benign, utilizing Convolutional Neural Networks (CNN). With a dataset of 164k and each image of size 50*50 with RGB channels being used, the model was able to achieve the accuracy of 87% despite using constrained computational resources. Unlike other computationally expensive deep models like ResNet50, this model, with 27 layers, performed well on this dataset with the help of data processing and augmentation. This result indicates the potential of this model's aptitude for early IDC detection. Additionally, Grad-CAM was deployed to interpret the model's predictions by highlighting the most influential part of the image.

Keywords: Breast Cancer, Invasive Ductal Carcinoma(IDC), Histopathological Images, Deep Learning, Convolutional Neural Network(CNN), Binary Classification, Data Augmentation, Lightweight Model

1 Introduction

Breast cancer is one of the most prevalent causes of death in women around the world. Among various subtypes, Invasive Ductal Carcinoma(IDC) accounts for about 80% of total cases [1]. Early and accurate diagnosis of IDC is crucial for improving and curing patients' health; however, in most cases, doctors do manual histopathological image analysis[1]. This process in itself is time-consuming, demanding more expertise from doctors, and prone to inter-observer variability, highlighting the need for reliable and automated systems [1].

Recent advancements and progress in deep learning, like CNN, have delivered significant value in medical image analysis [1]. Models like ResNet50 and VGG have delivered remarkable results in large-scale image classification tasks; however, their computational expense makes them not a great choice for small-resolution histopathological images.

In this study, I proposed a custom-designed CNN model for the binary classification of IDC in histopathological images. The total dataset contained 277k images of size 50*50 with RGB channels [2]. However, due to limitations in computation power and a lack of advanced hardware like SSDs & GPUs, I used 164k total data samples. The subset was chosen by shuffling and choosing data with the random library of Python to maintain class balance and stability so that learning becomes meaningful and generalizable despite a lack of significant computational power.

The focus of this work is to design a lightweight CNN model that offers solid baseline performance(giving 87% accuracy) while remaining feasible for environments with low computational power.

2 Dataset & Preprocessing

2.1 Overview

The dataset used in this study is the publicly available Breast Histopathological Images dataset from Kaggle, which contains 277,524 color image patches with 50×50 pixels in RGB format extracted from whole-slide images (WSIs) of breast cancer tissue [2]. Each image is labeled as either:

Class 0: IDC-negative, Count: 198,738

Class 1: IDC-positive, Count: 78,786

based on the presence or absence of Invasive Ductal Carcinoma.

The dataset was originally collected and curated by the University of Pennsylvania and Baylor College of Medicine [3], ensuring clinical relevance and real-world variability.

2.2 Dataset Organization

Initially, the dataset was organized in a nested structure with hundreds of directories containing two subclasses and thousands of images under that. So, I used Python to merge them into a single directory with two subdirectories (train and test), and two classes inside that for containing images related to that class. So now the structure of the directory looks like this:

```
organized_dataset/
      train/
        0/   → IDC-negative images
        1/   → IDC-positive images
     test/
         0/   → IDC-negative images
         1/   → IDC-positive images
```

2.3 Subset Selection

Due to hardware limitations (CPU-only environment, lack of and limitation of SSD & GPU), training on the full data set was computationally infeasible. Therefore, a subset of data consisting of approximately 165k samples was selected and trained on the model. Subset data was split into a train, validation, and test set for future training. Key considerations during subset selection included:

Class Balance: Equal or near-equal number of images from both classes.

Randomization: Images were randomly sampled to prevent source bias.

2.4 Preprocessing Pipeline

Before training, the data was preprocessed to enhance the performance and to ensure compatibility with CNN inputs. Data was already split into train, validation, and test sets, which allowed explicit control over data distribution.

To increase diversity and prevent possible overfitting, augmentation was done with the help of Tensorflow's ImageDataGenerator. The transformation included common techniques such as flipping, rotation, and brightness adjustment [4].

- Rescaling pixel values to the $[0, 1]$ range
- Random rotations (± 20 degrees)
- Zooming (up to 20%)
- Width and height shifts (up to 10%)
- Brightness adjustments (range: 0.8–1.2)
- Horizontal flips

Both the validation and test were only rescaled(no augmentation) to preserve evaluation consistency. All the images were resized to 128*128 to align with the input requirements of the CNN architecture. Moreover, ImageDataGenerator was used for generating separate generators for the train, validation, and test sets.

3 Methodology

3.1 Architecture Overview

Even though the initial approach involved using a pretrained deep layer ResNet50[5], computational limitations such as low CPU & GPU capacity and memory constraints led to the design of a custom CNN architecture involving a lesser but sufficient number of layers needed to train the model.

This lightweight model was designed in such a way to maintain a balance between computational efficiency and high classification accuracy, especially for environments with limited hardware resources.

3.2 Network Design

The final architecture of the model consists of 27 layers, including:

Multiple Conv2D layers with increasing filter size(starting from 32)

BatchNormalization for stabilizing and accelerating training [6]

MaxPooling2D layers for spatial downsampling

Dropout layers to mitigate overfitting [7]

Flattening followed by two fully connected (Dense) layers

A final output layer with a sigmoid activation for binary classification

The input image shape was set to (128, 128, 3) to maintain consistency throughout the pipeline.

3.3 Model Summary

The model was built using the Keras Functional API and trained with the binary cross-entropy loss function and Adam optimizer. Below is high-level schematic of the model architecture:

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 128, 128, 3)	0
conv2d (Conv2D)	(None, 128, 128, 32)	896
batch_normalization	(None, 128, 128, 32)	128
activation (Activation)	(None, 128, 128, 32)	0
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_1 (Conv2D)	(None, 64, 64, 64)	18,496
batch_normalization_1	(None, 64, 64, 64)	256
activation_1 (Activation)	(None, 64, 64, 64)	0
max_pooling2d_1	(None, 32, 32, 64)	0
conv2d_2 (Conv2D)	(None, 32, 32, 128)	73,856
batch_normalization_2	(None, 32, 32, 128)	512
activation_2	(None, 32, 32, 128)	0
max_pooling2d_2	(None, 16, 16, 128)	0
separable_conv2d	(None, 16, 16, 256)	34,176
batch_normalization_3	(None, 16, 16, 256)	1,024
activation_3	(None, 16, 16, 256)	0
max_pooling2d_3	(None, 8, 8, 256)	0
global_average_pooling2d	(None, 256)	0
dropout (Dropout)	(None, 256)	0
dense (Dense)	(None, 128)	32,896
batch_normalization_4	(None, 128)	512
activation_4	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
batch_normalization_5	(None, 64)	256
activation_5	(None, 64)	0
dense_2 (Dense)	(None, 1)	65

Table 1: Model architecture summary of the custom CNN

3.4 Training Setup

The model was compiled and trained using the following configuration:

- **Loss Function:** Binary Crossentropy [8]
- **Optimizer:** Adam [9]
- **Learning Rate:** Default value (0.001)
- **Metrics:** Accuracy
- **Epochs:** 100
- **Batch Size:** 32
- **Callbacks:**
  - `ReduceLROnPlateau` – Monitored 'val_loss' and reduced the learning rate by a factor of 0.5 (factor=0.5) when no improvement was detected for 3 consecutive epochs (patience=3). The minimum learning rate was set at 1e-6 to avoid excessively small updates.
  - `EarlyStopping` – Monitored the 'val_loss' and halted training if no improvement was seen for 5 continuous epochs (Patience=5). Additionally, restore_best_weights=True ensured that model weights are reverted back to the best-performing set on validation data.
- **Regularization:** L2 regularization (also known as weight decay) [10] was applied to penalize large weights and reduce overfitting. The total loss function used during training becomes:

$$\mathcal{L}_{\text{Total}} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] + \lambda \sum_{j=1}^{n} w_j^2$$

  - y_i = true label of sample i
  - $\hat{y}_i$ = predicted probability for sample i
  - N = total number of samples
  - w_j = individual trainable weights of the model
  - λ = L2 regularization coefficient

3.5 Evaluation Metrics

To assess the performance of the model, various metrics were used, such as accuracy, precision, recall, and F1-score [11]:

Accuracy: The proportion of correctly predicted items among all predictions. Let TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives. Then, Accuracy is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: The proportion of correctly predicted samples from all items predicted as positive. This metric indicates how reliable a positive prediction is and is particularly important in reducing false positives.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall: The proportion of actual positive samples correctly identified by the model. High recall is crucial in medical diagnosis tasks where missing a cancer-positive case is costly.

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score: The evaluation metric that provides a single value for assessing model performance by performing the harmonic mean of Precision and Recall.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Confusion Matrix: A matrix that provides the model's strengths and weaknesses in terms of true positives, true negatives, false positives, and false negatives. Typically represented as:

$$\begin{bmatrix} \text{TN} & \text{FP} \\ \text{FN} & \text{TP} \end{bmatrix}$$

Loss Curve Analysis: Monitoring the progression of training and validation loss over epochs to detect underfitting, overfitting, or convergence. For binary classification, the Binary Crossentropy [8] loss is typically used:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

3.6 Interpretability with Grad-CAM

In deep-learning-based medical imaging, interpretability plays a crucial role in understanding how a particular decision was made. To ensure transparency in decision-making, I incorporated Gradient-weighted Class Activation Mapping(Grad-CAM) [12] into the workflow. Grad-CAM enables to visualize the part of the image that affects more on decision making.

3.6.1 Theoretical Background

Grad-CAM operates by computing gradients of loss with respect to feature maps of the final convolutional layer, therefore highlighting the most important and influential region of the input image [12].
Let:

$y_c = \text{score for class } c \text{ (before the sigmoid activation)}$
$A^k = k^{\text{th}} \text{ feature map of the last convolutional layer}$
$\alpha_k^c = \text{importance weight for } A^k \text{ w.r.t. class } c$

The weight α_k^c is computed as the global average of the gradients flowing back from the output:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y_c}{\partial A_{ij}^k}$$

where:

- Z is the total number of pixels in the feature map (i.e., height $\times$ width)
- $\frac{\partial y_c}{\partial A_{ij}^k}$ is the gradient of the score y_c w.r.t. the activation at location (i, j) in feature map k

The class activation heatmap $L_{\text{Grad-CAM}}^c$ is computed as:

$$L_{\text{Grad-CAM}}^c = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right)$$

ReLU retains only the features that positively influence the prediction for class c, discarding less relevant (negative) activations.

3.6.2 Interpretation Details

I used the final separable convolutional layer (`separable_conv2d`) in our custom CNN as the Grad-CAM target.

The gradient of the prediction score (for the IDC-positive class) was computed using TensorFlow's `GradientTape`.

The resulting heatmap was:

- Upsampled to the original image resolution
- Superimposed over the input image using OpenCV

For balanced interpretation, I selected one test image per class:

- IDC-positive
- IDC-negative

A side-by-side 2×2 grid visualization was generated, showing:

- Original image
- Corresponding Grad-CAM overlay

This approach provides visual justification of model predictions providing transparency about its decision on why the model chooses a particular decision for a given image.

4 Results & Evaluation

4.1 Training Metrics

The model was trained on approximately 165k datasets using the Adam optimizer and binary cross-entropy loss. A batch of 32 and an input size of

128*128 was used along with the augmentation in training data, including rotation, zoom, flip, shift, etc.[4], and regularization(Dropout & L2) [7] to prevent overfitting.

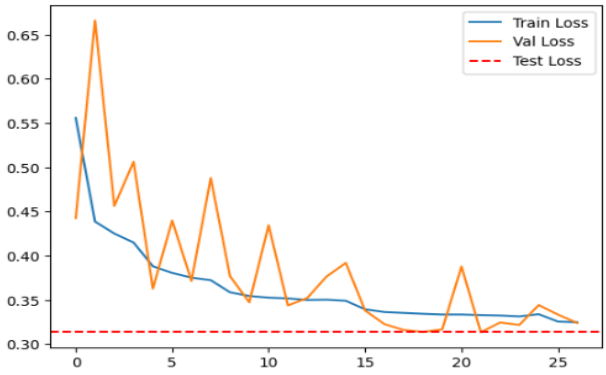

Figure 1: Loss across different datasets during training. The X-axis represents the number of epochs, and the Y-axis represents the binary cross-entropy loss.

To optimize training efficiency and model generalization, EarlyStopping and ReduceLROnPlateau were used with patience 5 & 3 respectively. Although training was initially configured for 100 epochs, it was automatically stopped after 27 epochs when no further improvement in validation loss was observed.

Despite this early convergence, the model achieved promising performance:

- **Validation Accuracy:** 87%
- **Validation Loss:** 31%
- **Test Accuracy:** 87.61%
- **Test Loss:** 31.39%

This training strategy helped the model prevent overfitting and reduce computational cost.

4.2 Confusion Matrix & Classification Report

To analyze the performance, the model was tested on the unseen test data:

Confusion Matrix:

ActualPredicted	Class 0	Class 1
Class 0	3352	648
Class 1	343	3657

Table 2: Confusion Matrix of the model

Classification Report:

Class	Precision	Recall	F1-Score
0	0.91	0.84	0.87
1	0.85	0.91	0.88

Table 3: Precision, Recall, and F1-Score for each class

These metrics are very important in sensitive fields like medical diagnosis, where false negatives carry higher risks than false positives.

4.3 Grad-CAM Visualization

To interpret and visualize CNN's prediction and reason why it made that decision, I used Gradient-weighted Class Activation Mapping (Grad-CAM) to generate heatmaps highlighting the region of the image that most affected the classification decision [12].

A visual summary is provided in the figure below for images from both classes, alongside their respective Grad-CAM overlays.

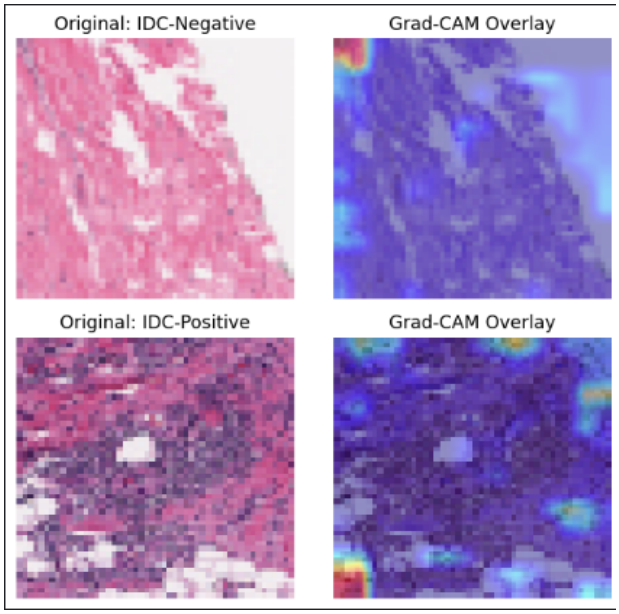

Figure 2: Grad-CAM Visualization

For the IDC-Positive Image(Cancerous), the Grad-CAM model showed one strong red spot and multiple yellowish and some bluish spots. This pattern suggests that the model finds multiple regions that heavily impact the chance of having cancer.

Interestingly, for the IDC-Negative Image(Non-Cancerous), the Grad-CAM model showed one strong red spot. However, this was the only prominent spot in the image, and no significant spot in the surrounding area was seen. This may indicate that the model marked this as a possible spot for cancer but ultimately classified it as benign at last based on broader spatial context and lack of multiple malignant cues.

4.4 Discussion

Despite the lack of powerful computational resources, limited dataset, and small image patch size(originally 50*50, resized to 128*128), the model performed quite well. The combination of Batch Normalization, Separable Convolutions [13], Dropout, and L2 regularization helped achieve strong regularization. Moreover, the use of data augmentation helped to create a broader set of tissue variations, likely improving generalizability.

4.5 Performance Insights

The model architecture is deliberately kept lightweight compared to deeper models like ResNet50. This model, along with regularization and augmentation, enabled stable training on the dataset while avoiding overfitting. The early stopping and learning rate(alpha) reduction strategy helped the model to converge to the global minima despite having scarce computational resources.

While the model's performance is promising, it can be improved by slightly deeper layers or an attention mechanism. Nevertheless, its low inference time and high parameter efficiency make it suitable for deployment in real-time or resource-constrained clinical settings.

Finally, the interpretability of such a custom CNN can be easier to manage than that of deeper models, supporting its use in medical AI pipelines where explainability is crucial.

5 Limitations

While the proposed model yielded promising results on IDC binary classification tasks, several limitations must be acknowledged:

- **Partial Dataset Usage:** Due to hardware constraints, approximately 165k dataset out of 277k were used. A complete dataset may enhance model performance by finding all patterns and making it more generalizable.
- **Limited Training Resources:** The model was trained without high-performance hardware like CPU, GPU, and SSDs. This significantly limited the full potential of the model by constraining batch size, sacrificing faster computation, and overall experimentation.
- **Absence of Cross Validation:** A simple train-validation-test set was used for model training. Cross-validation could provide more reliable performance and reduce the risk of bias.
- **No Comparative Benchmark**: The model's performance was not compared against standard deep learning models like ResNet50, VGG on the same dataset due to a lack of powerful computation resources required for them to execute.

6 Conclusion and Future Work

This work presents a custom-designed Convolutional Neural Network for the binary classification of invasive ductal carcinoma(IDC) from histopathology images. The model was able to deliver promising results despite being trained on a subset of data and limited computational power. This was possible with careful architecture design along with different optimization techniques to tune the parameters & hyperparameters.

The results indicate that a lightweight CNN, if carefully designed, can achieve high accuracy levels even on challenging medical tasks. The model also offers a computationally feasible alternative to pretrained deeper models, making it pragmatic in environments with limited hardware and resources.

Future works include:

- Utilizing the complete dataset for full-scale training to optimize the performance.
- Extending the model to multimodal learning by integrating genomic/proteomic data alongside image data.
- Benchmarking against pretrained deeper networks like ResNet50 and VGG to evaluate the performance side by side.

Ultimately, this work serves as a stepping stone for more robust multimodal AI systems for future work in closely related tasks in the clinical field.

References

[1] G. Litjens, T. Kooi, B. E. Bejnordi, *et al.*, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, Dec. 2017, ISSN: 1361-8415. DOI: 10.1016/j.media.2017.07.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841517301135 (visited on 07/10/2025).

[2] *Breast Histopathology Images*, en. [Online]. Available: https://www.kaggle.com/datasets/paultimothymooney/breast-histopathology-images (visited on 07/10/2025).

[3] A. Cruz-Roa, A. Basavanhally, F. González, *et al.*, "Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks," in *Medical Imaging 2014: Digital Pathology*, vol. 9041, SPIE, Mar. 2014, p. 904103. DOI: 10.1117/12.2043872. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/9041/904103/Automatic-detection-of-invasive-ductal-carcinoma-in-whole-slide-images/10.1117/12.2043872.full (visited on 07/10/2025).

[4] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019. DOI: 10.1186/s40537-019-0197-0.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. DOI: 10.1109/CVPR.2016.90.

[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, PMLR, 2015, pp. 448–456. DOI: 10.48550/arXiv.1502.03167.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[8] K. P. Murphy, "Machine learning: A probabilistic perspective," 2012.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. DOI: 10.48550/arXiv.1412.6980.

[10] A. Y. Ng, *Feature Selection, L1 vs L2 Regularization, and Rotational Invariance*. 2004. DOI: 10.1145/1015330.1015435.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org.

[12] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

[13] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258. DOI: 10.1109/CVPR.2017.195.