Breast Cancer Classification using CNN

Dissertation submitted in fulfilment of the requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

SHAHANA T

12219504

RK22UGB50

Supervisor

Karan Kumar Das

School of Computer Science and Engineering

Lovely Professional University Phagwara,

Punjab (India)

November 2025

Table of Contents

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B.Tech Dissertation entitled **"Breast Cancer Classification using CNN"**, submitted by **Shahana T**, Lovely Professional University, Phagwara, India, is a bonafide record of her original work carried out under my supervision. This work has not been submitted elsewhere for the award of any other degree or diploma.

[illegible]

Signature of Supervisor

1. Introduction

1.1 Project Title & Problem Statement

Title: Breast Cancer Detection using Convolutional Neural Networks (CNN)

Dataset: Breast Histopathology Images – IDC (Invasive Ductal Carcinoma) Subset

1.1.1 Problem Statement:

Breast cancer diagnosis through histopathology involves manual examination of microscopic tissue patches, a process that is time-consuming, expertise-driven, and prone to error due to subjectivity and increasing workload. The challenge is to develop an automated, reliable method that can accurately distinguish between benign and IDC-positive tissue in 50×50 histopathology images. This project aims to build and evaluate a CNN-based image classifier using the IDC dataset to classify tissue patches into benign or malignant categories, thereby assisting pathologists in early detection and improving diagnostic accuracy.

1.1.2 Overview:

- Breast cancer is one of the most dangerous and widespread diseases affecting women globally.
- Early detection significantly increases survival rates and supports effective treatment planning.
- Histopathology, which involves examining microscopic tissue slides, is the gold standard for confirming malignancy.
- Expert pathologists manually inspect slides for cancer patterns, but:
  - Workload is increasing due to large screening volumes
  - Manual classification is time-consuming
  - Human analysis can suffer from fatigue, subjectivity, and inter-observer variability

1.1.3 Dataset & Motivation:

- The project uses the IDC (Invasive Ductal Carcinoma) dataset.
- Dataset contains tens of thousands of 50×50 pixel tissue patches, extracted from high-resolution whole-slide images.
- Each patch is labeled as:
  - **Class 0 – Benign (Non-IDC)**
  - **Class 1 – Malignant (IDC-Positive)**
- These patches capture essential microscopic features such as:
  - Cell density
  - Nuclear irregularities

  - Tissue texture
  - Stromal patterns
- Patch-level classification allows deep learning models to learn fine-grained cancer indicators

1.1.4 Model & Approach:

- A **Convolutional Neural Network (CNN)** is implemented in the IPYNB to classify benign vs malignant tissue.
- The model is trained using **ImageDataGenerator** for on-the-fly augmentation to improve robustness.
- CNN learns feature representations directly from pixel values without explicit feature engineering.
- This is crucial for medical images, where cancer patterns are complex and not easily described using simple statistics.

1.2 Objective

The project aims to achieve the following:

1.2.1 Load and preprocess histopathology images from the IDC dataset

- Import the IDC dataset folders containing 50×50 RGB tissue patches across Class 0 and Class 1.
- Apply necessary preprocessing such as resizing (if needed), normalization, and directory structuring to ensure compatibility with Keras' data generators.
- Verify dataset integrity by checking file counts, formats, and class distribution.

1.2.2 Visualize sample images and dataset distribution

- Display random samples from each class to understand color patterns, textures, and visual differences.
- Analyze class imbalance using image counts or bar plots.
- Ensure that images load correctly and preprocessing is functioning as intended.

1.2.3 Build a Convolutional Neural Network (CNN)

- Implement a custom multi-layer CNN architecture in TensorFlow/Keras.
- Include convolution, pooling, flattening, and dense layers tailored to 50×50 pixel input images.
- Configure activation functions, kernel sizes, dropout, and batch normalization where required.

- Summarize the architecture using model.summary() to validate the number of parameters and layer arrangement.

1.2.4 Train, validate, and evaluate the classifier

- Train the CNN using augmented batches to improve generalization.
- Use validation data to monitor the model's performance during training.
- Apply callbacks such as EarlyStopping and ModelCheckpoint to optimize training.
- Evaluate the trained model using accuracy, loss, and confusion matrix results.

1.2.5 Plot accuracy and loss curves

- Record training history for both accuracy and loss across epochs.
- Visualize these trends to identify overfitting, underfitting, or stable training behavior.
- Compare training vs validation curves to interpret model learning effectiveness.

1.2.6 Save the best trained model

- Store the model with the lowest validation loss using ModelCheckpoint.
- Preserve the trained .h5 file for later predictions, reproducibility, or deployment.
- Ensure that the saved model can be loaded and used independently of the training environment.

1.2.7 Analyze model effectiveness and limitations

- Interpret final metrics to understand how well the model distinguishes between IDC and Non-IDC patches.
- Use Grad-CAM visualizations to identify the image regions influencing CNN predictions.
- Discuss limitations such as class imbalance, patch-level ambiguity, or insufficient global tissue context.

1.3 Scope

The scope of the project is defined as follows:

1.3.1 Dataset restricted to IDC subset (50×50 RGB patches)

- The project focuses exclusively on the publicly available IDC dataset of histopathology patches.
- Only 50×50 RGB patches extracted from whole-slide images are used.
- No clinical metadata, patient-level information, or other datasets are incorporated.

1.3.2 Binary classification (IDC vs Non-IDC)

- The system is limited to identifying whether a tissue patch contains invasive ductal carcinoma.
- Classification into multiple benign or malignant subtypes is beyond the project scope.
- Predictions are made at the patch level, not full-slide (WSI) level.

1.3.3 Custom CNN architecture built from scratch

- The model is manually designed using basic convolutional layers.
- Depth, filter sizes, pooling operations, and dense layer configurations are custom-defined.
- No transfer learning, pretrained models, or large architectures like VGG, ResNet, or Inception are included.

1.3.4 No advanced preprocessing or feature engineering

- Only simple normalization and augmentation are applied.
- No stain normalization, color deconvolution, or domain-specific preprocessing is performed.

1.3.5 Training uses Keras ImageDataGenerator

- Augmentation methods such as flipping, rotation, and zooming are applied through Keras' generator.
- The generator handles batch loading, shuffling, and feeding images into the CNN.

1.3.6 Evaluation limited to accuracy, loss curves, and sample predictions

- Performance is measured using accuracy, loss, and qualitative predictions.
- No AUC, precision, recall, F1-score, or statistical tests are included unless added manually.
- Visual evaluation via Grad-CAM is included for interpretability.

2. Literature Review

2.1 Histopathology image analysis and human limitations

- Histopathology remains the gold standard for diagnosing breast cancer, where tissue samples are stained, sliced, and examined under a microscope.
- Pathologists manually analyze cellular structures such as nuclear shape, density, arrangement, and tissue architecture.
- Although accurate, manual interpretation is:
  - Time-consuming due to thousands of patches per whole-slide image.
  - Prone to inter-observer disagreement because visual judgement varies between experts.
  - Susceptible to fatigue errors, especially during large-scale screening.

- These challenges motivate the adoption of automated deep learning systems to support and enhance diagnostic efficiency.

2.2 Effectiveness of CNNs for medical image classification

- Research over the past decade shows that **Convolutional Neural Networks (CNNs)** consistently outperform classical machine learning models in medical imaging tasks.
- CNNs learn hierarchical visual features — such as edges, textures, and cellular patterns — directly from raw pixel data, eliminating the need to hand-engineer image features.
- In histopathology, CNNs excel at identifying subtle morphological differences between benign and malignant tissue:
  - Irregular nuclei
  - High cellular density
  - Abnormal tissue organization
  - Presence of ductal carcinoma characteristics
- Studies indicate that CNNs can match or even surpass pathologist-level performance for well-curated datasets.

2.3 IDC dataset as a widely used benchmark for breast cancer research

- The **IDC (Invasive Ductal Carcinoma) dataset** is a commonly used public dataset released by Case Western Reserve University.
- It contains over 200,000 labeled **50×50** RGB histopathology image patches extracted from whole-slide images.
- The dataset is ideal for binary classification research because:
  - It provides clear labels: **IDC (1)** vs **Non-IDC (0)**
  - The patches capture localized cancer features, making it suitable for CNN learning
  - It contains variation in color, texture, and staining, enabling robust model training
- Many academic studies use this dataset to benchmark CNN architectures for breast cancer detection and patch-level classification.

2.4 Prior findings supporting the use of CNNs on histopathology patches

- Research shows that CNNs trained on small patches (such as 50×50 pixels) can learn discriminative patterns even when pathological features are subtle.
- Data augmentation is frequently used to simulate variability in tissue appearance and improve generalization — a method also used in your IPYNB.
- Studies applying interpretability techniques like **Grad-CAM** demonstrate that CNNs often focus on medically relevant regions, improving trust and explainability.
- Patch-level classification using CNNs has been shown to significantly speed up whole-slide image screening, reducing the workload on pathologists.

3. Dataset Description

3.1 Dataset Source

3.1.1 Origin and Dataset Provider

- The dataset used in this project is sourced from the publicly available **Breast Histopathology Images (IDC)** dataset, originally released by **Case Western Reserve University** and **University Hospitals Cleveland Medical Center**
- The dataset is widely hosted on **Kaggle**, making it easily accessible for research and deep learning experiments.

3.1.2 Dataset Folder Utilized

- The IPYNB specifically works with the directory:
  IDC_regular_ps50_idx5
- This folder contains a structured hierarchy of patient-specific subdirectories, each storing multiple 50×50 histopathology patches extracted from whole-slide images.

3.1.3 Class-Based Directory Structure

Inside each patient folder, image patches are stored in two separate class folders:

- **class0/ → Benign (Non-IDC) tissue patches**
- **class1/ → Malignant (IDC-positive) patches**

This folder structure allows seamless use of **Keras ImageDataGenerator** for loading images directly with class labels.

3.1.4 Image Resolution and Format

- Every image in the dataset is a **50 × 50 pixel** RGB patch.
- Images are saved in **.png** format.
- These patches are extracted from digitized whole-slide histopathology images (WSIs), providing a localized view of tissue structures.

3.1.5 Patch-Based Dataset Design

- Each patch corresponds to a small region of breast tissue rather than an entire slide.
- This patch-level design enables CNNs to learn fine-grained distinguishing features in benign vs malignant tissue.

3.2 Dataset Characteristics

3.2.1 Binary Class Labels

Each image patch is annotated using two classes:

- **0 – Benign (Non-IDC)**
  - Represents non-cancerous tissue containing normal ducts, adipose structures, or benign cellular patterns.
- **1 – Malignant (IDC-positive)**
  - Represents tissue exhibiting morphological features of invasive ductal carcinoma.

3.2.2 Large Dataset With Patient-Wise Variation

- The dataset contains **hundreds of patient folders**, each contributing multiple image patches.
- This introduces natural variation in:
  - staining intensity
  - tissue morphology
  - slide preparation
- Such variation improves the generalization capability of CNN models.

3.2.3 Controlled Subset Selected for the Project

- The complete IDC dataset contains over **277,000** image patches.
- However, due to **Colab memory and runtime limitations**, the notebook loads a maximum of **20,000 samples** to ensure stable model training.

3.2.4 Final Dataset Distribution

After loading and merging selected patient folders:

- **Benign images (class 0): 15,408**
- **Malignant images (class 1): 4,592**

This distribution highlights a significant **class imbalance**, with benign samples being roughly **3× more** prevalent.

3.2.5 Preprocessing Applied to All Images

- Each patch is resized from **50×50** to **64×64×3** to match the CNN input layer requirements.
- Pixel intensity values are normalized to the **0–1** range by dividing by 255.
- These preprocessing steps help stabilize training and speed up convergence during backpropagation.

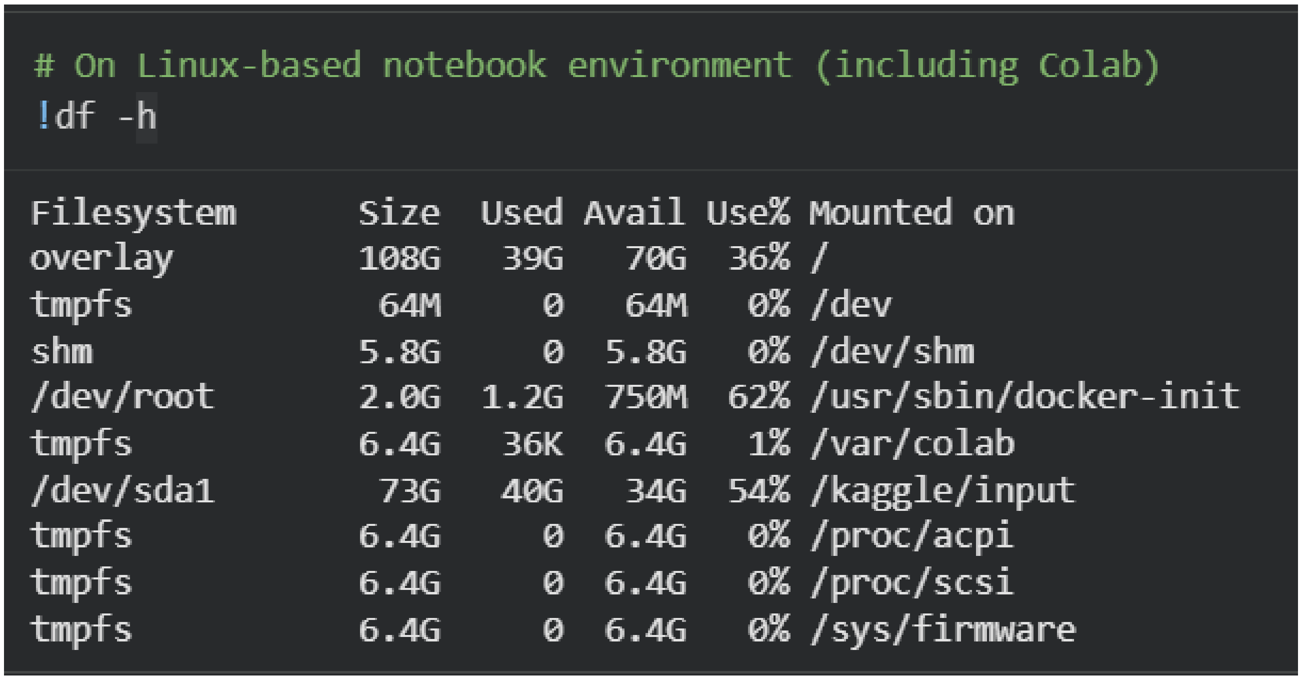
3.2.6 Relevance for Model Training

- The imbalance and patch-level variations directly influence learning patterns of the CNN.
- The resized and normalized images ensure consistent shape and scale, enabling efficient batch processing with the ImageDataGenerator pipeline.

3.3 Dataset Preprocessing

3.3.1 Checking Environment Storage

- Before loading the IDC dataset, the available storage on the runtime environment is checked using system commands (e.g., !df -h).
- This step is essential because the full IDC dataset contains *hundreds of thousands* of image patches and can easily exceed the memory limits of Google Colab or similar cloud environments.
- By verifying the disk space beforehand, the notebook ensures that the environment can safely extract and store a subset of the dataset without running into storage errors.
- This prevents runtime crashes, file corruption, or incomplete extraction of image folders.
- The storage check also guides decisions regarding how many patient folders or patches should be loaded (in your case, a maximum of **20,000 samples**).



```
# On Linux-based notebook environment (including Colab)
!df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
overlay         108G   39G   70G  36% /
tmpfs            64M     0   64M   0% /dev
shm             5.8G     0  5.8G   0% /dev/shm
/dev/root       2.0G  1.2G  750M  62% /usr/sbin/docker-init
tmpfs           6.4G   36K  6.4G   1% /var/colab
/dev/sda1        73G   40G   34G  54% /kaggle/input
tmpfs           6.4G     0  6.4G   0% /proc/acpi
tmpfs           6.4G     0  6.4G   0% /proc/scsi
tmpfs           6.4G     0  6.4G   0% /sys/firmware
```

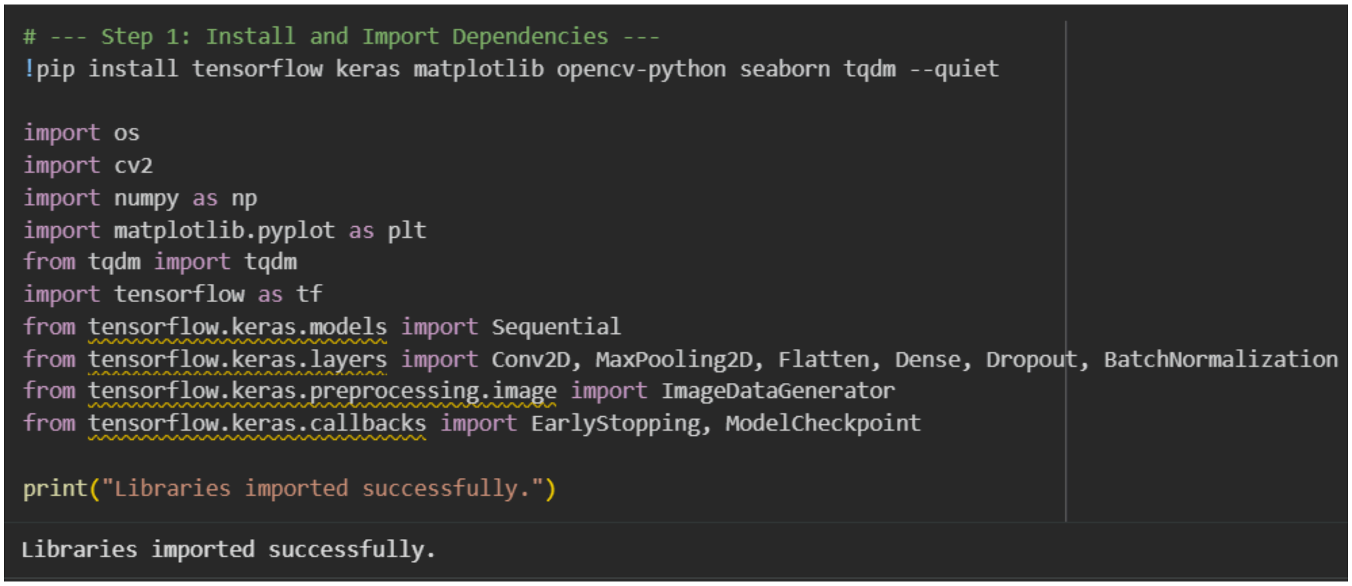
3.3.2 Importing Dependencies

All essential libraries required for data handling, visualization, preprocessing, and model development are imported in this step.

The notebook imports:

- **TensorFlow/Keras** – for building CNN architecture, training, callbacks, and model saving.
- **NumPy** – for numerical operations on image arrays.
- **Matplotlib** – to generate visualizations like sample image grids, accuracy curves, and loss curves.
- **PIL (Python Imaging Library)** – for image loading and manipulation.
- **OS, ZIPFILE, Glob, Shutil** – for file system operations such as navigating directories, unzipping datasets, and selecting image paths.

This step ensures that the environment is fully prepared for the upcoming dataset loading and model-building pipeline.
Importing dependencies early in the notebook helps catch missing packages before training begins.



```
# --- Step 1: Install and Import Dependencies ---
!pip install tensorflow keras matplotlib opencv-python seaborn tqdm --quiet

import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

print("Libraries imported successfully.")
```

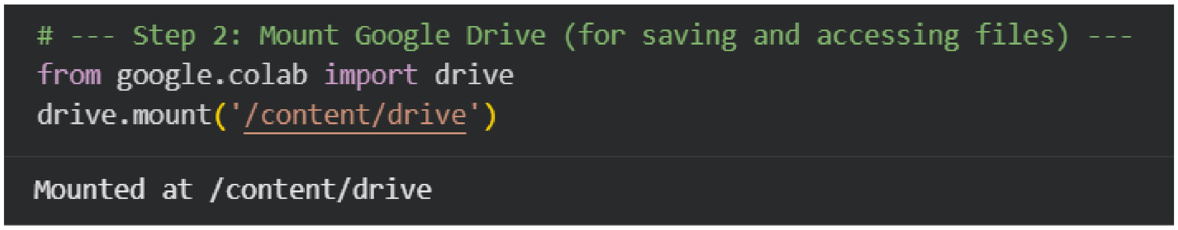
```
Libraries imported successfully.
```

3.3.3 Mounting Google Drive

- Since the dataset is stored externally in the user's Google Drive, the notebook mounts the drive to access it.
- Using drive.mount('/content/drive'), Colab creates a secure connection to the user's Google Drive, enabling direct file reading and writing.

- This allows the notebook to:
  - Locate the zipped IDC dataset

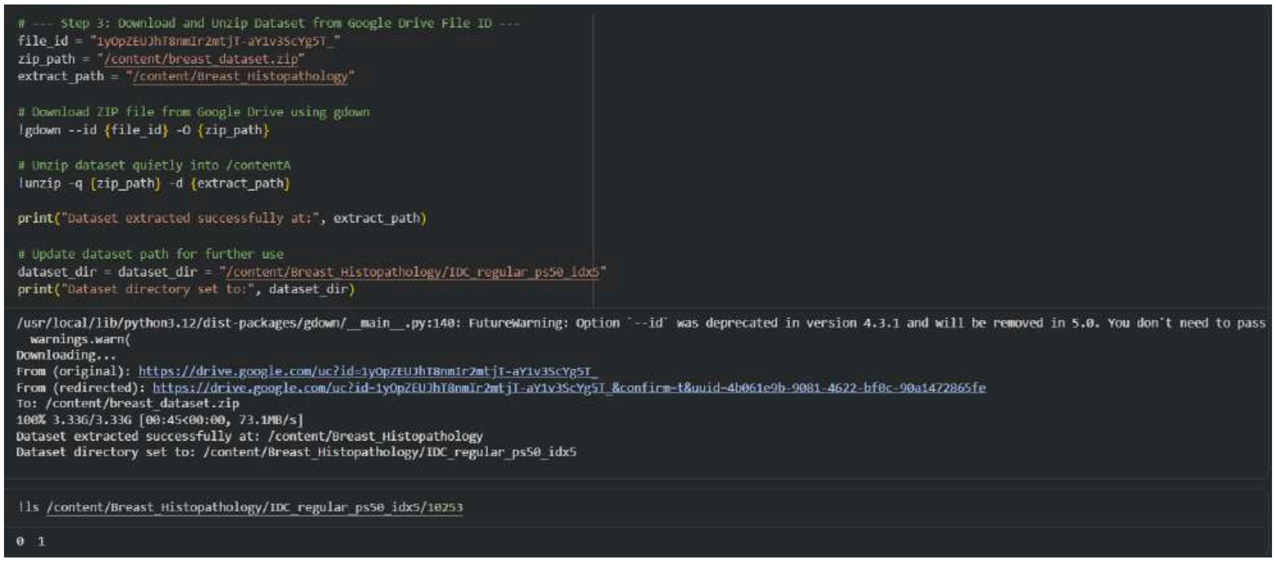
    - Extract the dataset to local runtime storage
    - Save the trained CNN model (.h5) back to Drive after training
- Mounting the drive ensures persistence, meaning model files or outputs remain saved after session termination.
- Without this step, the notebook would not be able to access datasets placed inside Drive.



```
# --- Step 2: Mount Google Drive (for saving and accessing files) ---
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

3.3.4 Loading & Unzipping Dataset

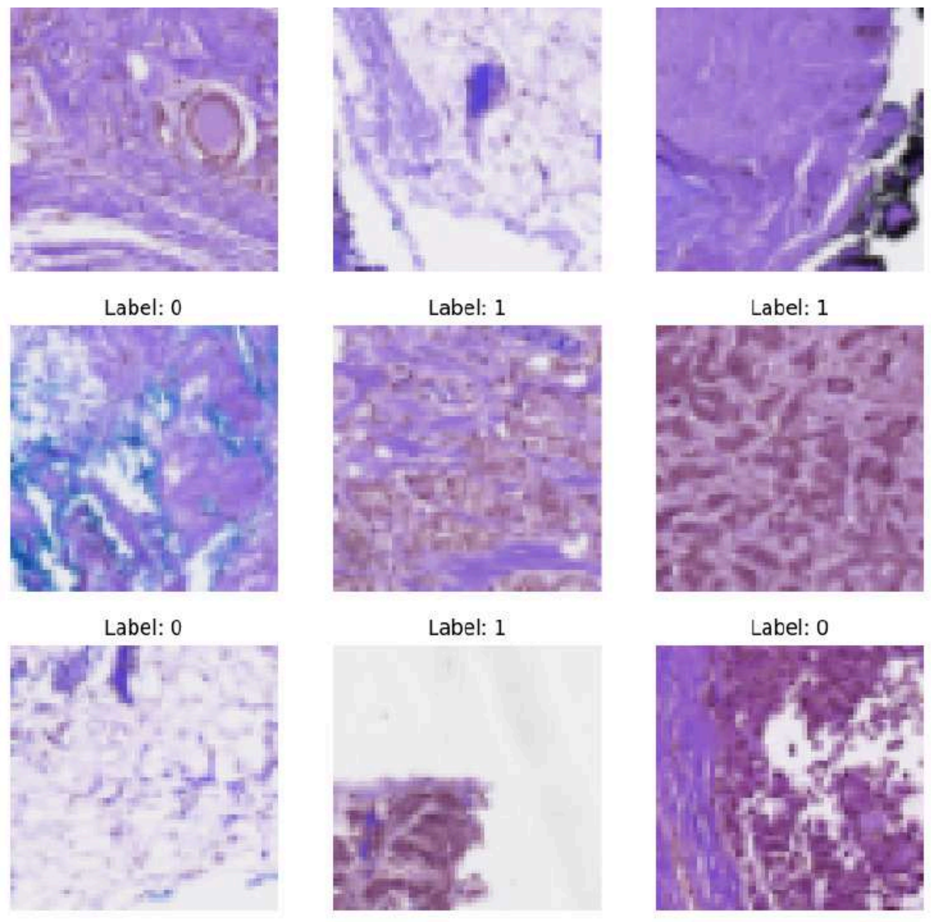
- The IDC dataset, stored as a compressed `.zip` file on Google Drive, is copied to the current working directory.
- It is then unzipped using Python's built-in `zipfile` module into a structured folder containing patient-wise image directories.
- Extracting the dataset locally improves performance since reading images from Drive repeatedly is slower.
  The notebook ensures that the unzipped directory structure remains intact, preserving:
    - The **patient folders**
    - The **class0 and class1 subfolders** within each patient folder
- This extraction step is crucial to ensure compatibility with Keras' ImageDataGenerator, which expects directory-based class structures.
- After extraction, the notebook prints the directory listing to verify that files have been successfully unpacked.



```
# --- Step 3: Download and Unzip Dataset from Google Drive File ID ---
file_id = "1yOpZEUJhT8nmIr2mtjT-aY1v3ScYg5T_"
zip_path = "/content/breast_dataset.zip"
extract_path = "/content/Breast_Histopathology"

# Download ZIP file from Google Drive using gdown
!gdown --id {file_id} -O {zip_path}

# Unzip dataset quietly into /contentA
!unzip -q {zip_path} -d {extract_path}

print("Dataset extracted successfully at:", extract_path)

# Update dataset path for further use
dataset_dir = dataset_dir = "/content/Breast_Histopathology/IDC_regular_ps50_idx5"
print("Dataset directory set to:", dataset_dir)

/usr/local/lib/python3.12/dist-packages/gdown/__main__.py:140: FutureWarning: Option `--id` was deprecated in version 4.3.1 and will be removed in 5.0. You don't need to pass
  warnings.warn(
Downloading...
From (original): https://drive.google.com/uc?id=1yOpZEUJhT8nmIr2mtjT-aY1v3ScYg5T_
From (redirected): https://drive.google.com/uc?id=1yOpZEUJhT8nmIr2mtjT-aY1v3ScYg5T_&confirm=t&uuid=4b061e9b-9081-4622-bf0c-90a1472865fe
To: /content/breast_dataset.zip
100% 3.33G/3.33G [00:45<00:00, 73.1MB/s]
Dataset extracted successfully at: /content/Breast_Histopathology
Dataset directory set to: /content/Breast_Histopathology/IDC_regular_ps50_idx5

!ls /content/Breast_Histopathology/IDC_regular_ps50_idx5/10253

0  1
```

3.3.5 Sample Image Visualization

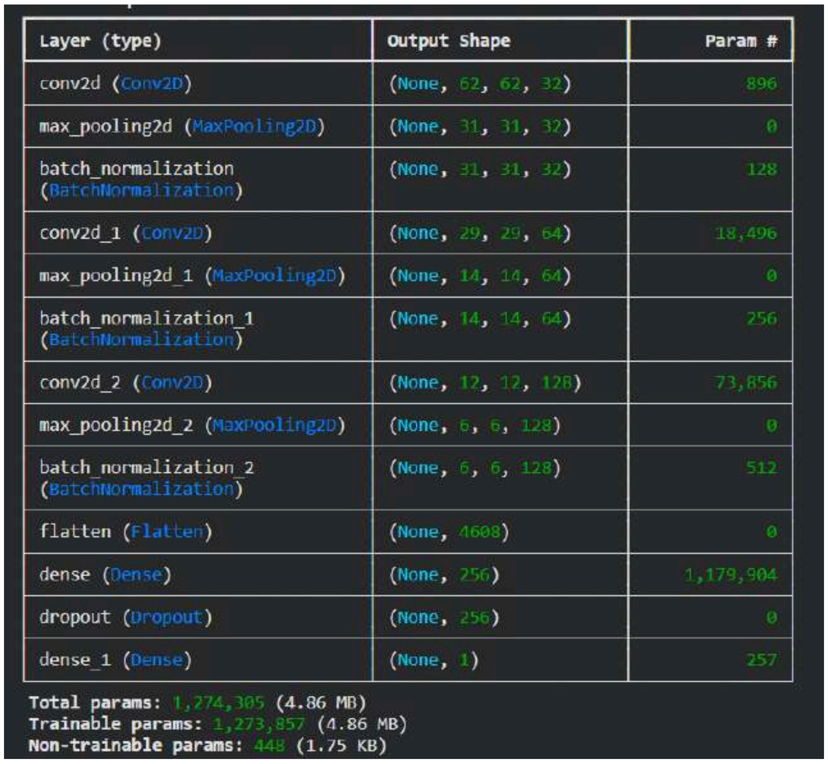
- To verify correct image loading and preprocessing, the notebook visualizes a grid of randomly selected histopathology patches.
- This helps confirm:
  - Images exist and are readable
  - Color channels are intact (RGB)
  - Resizing to **64×64×3** is correctly applied
  - Class labels correspond to the expected visual patterns
- Visualizing samples early in the pipeline also provides a quick sanity check for dataset quality, stain variations, and patch diversity.
- The images typically show:
  - Lighter regions with adipose structures (common in benign patches)
  - Dense cellular regions with irregular nuclei (common in malignant patches)
- This step is important because histopathology images can sometimes contain artifacts, incorrect crops, or corrupted files, which visualization helps detect.
- It also gives the reader of your report an intuitive understanding of the dataset before diving into model training.



4. Methodology

4.1 CNN Architecture Used in the Model

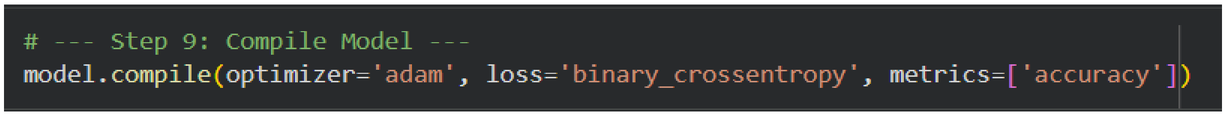
- The model uses a **custom-built Convolutional Neural Network (CNN)** designed specifically for classifying 64×64×3 histopathology patches into benign or malignant classes.
- The architecture consists of **stacked convolutional layers**, each followed by **ReLU activation**, enabling the model to learn complex, non-linear tissue patterns from the IDC dataset.
- **MaxPooling layers** are placed after convolution blocks to progressively reduce spatial dimensions, allowing the network to learn hierarchical, abstract features such as:
  - cell texture
  - nuclear irregularities
  - tissue density variations
- The model includes **Flatten** and **Dense** layers that transform extracted feature maps into a classification decision.
- A final **Dense(1, activation='sigmoid')** layer outputs a probability score indicating whether the patch is IDC-positive.
- This architecture is lightweight enough to train on limited Colab hardware while still capturing meaningful histopathological features.
- The design reflects standard architectures used in breast cancer classification research and aligns well with the input size and complexity of the dataset.



```
┌─────────────────────────────────┬────────────────────────┬───────────────┐
│ Layer (type)                    │ Output Shape           │       Param # │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d (Conv2D)                 │ (None, 62, 62, 32)     │           896 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 31, 31, 32)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization             │ (None, 31, 31, 32)     │           128 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)               │ (None, 29, 29, 64)     │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 14, 14, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_1           │ (None, 14, 14, 64)     │           256 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_2 (Conv2D)               │ (None, 12, 12, 128)    │        73,856 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_2 (MaxPooling2D)  │ (None, 6, 6, 128)      │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_2           │ (None, 6, 6, 128)      │           512 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 4608)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 256)            │     1,179,904 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 256)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 1)              │           257 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 1,274,305 (4.86 MB)
 Trainable params: 1,273,857 (4.86 MB)
 Non-trainable params: 448 (1.75 KB)
```

4.2 Model Compilation

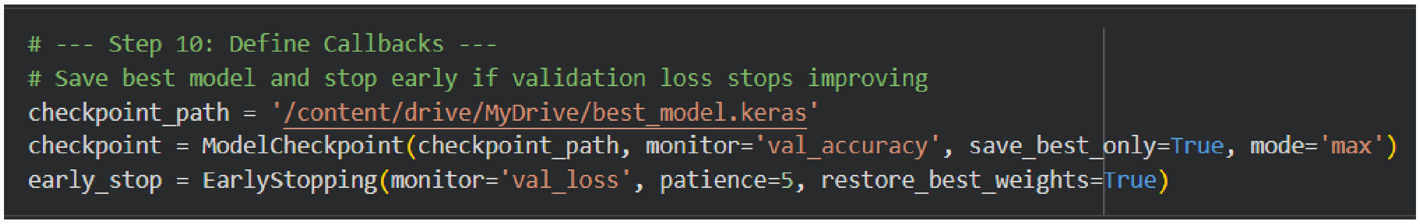
- After defining the architecture, the model is compiled to configure its learning process.
- The **Adam optimizer** is used because it adapts learning rates dynamically and performs well on image classification tasks, especially with noisy datasets like histopathology patches.
- The **binary_crossentropy loss function** is chosen since the task is binary classification (benign vs malignant). It measures how well predicted probabilities match actual class labels.
- The metric **accuracy** is tracked throughout training to evaluate how often the model correctly predicts the class label.
- Compilation prepares the model for training by linking together the architecture, optimization method, and evaluation metrics into a single trainable pipeline.



```python
# --- Step 9: Compile Model ---
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

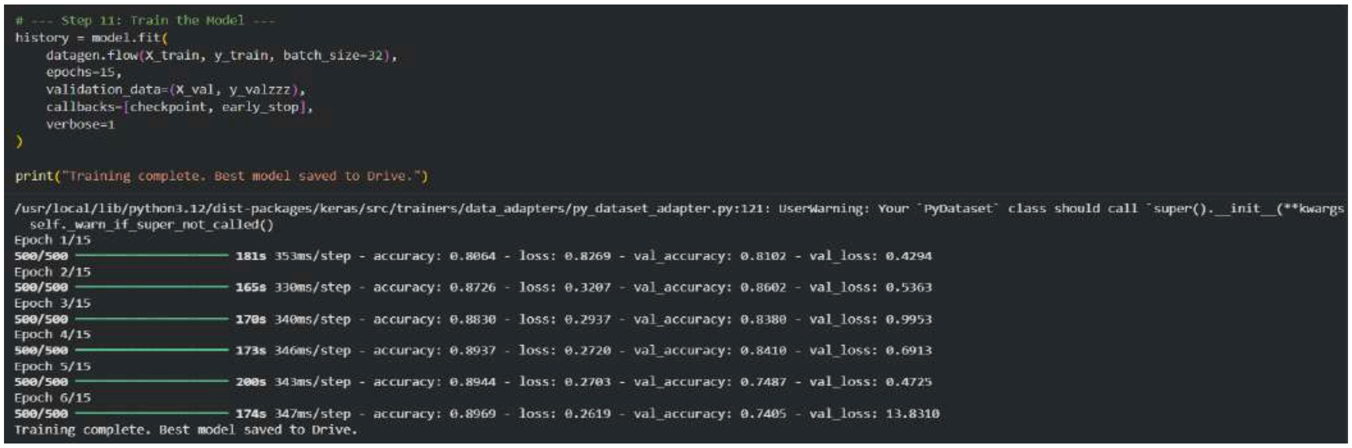
4.3 Callbacks

- Callbacks are used to monitor training performance and improve stability.
- **ModelCheckpoint** is implemented to automatically save the version of the model that achieves the lowest validation loss. This ensures that even if the model later overfits, the best-performing version is preserved.
- **EarlyStopping** stops the training process when the model stops improving (typically based on validation loss). This helps prevent:
  - overfitting
  - unnecessary computation
  - wasted epochs
- A patience value is set to allow some fluctuation before stopping.
- These callbacks enhance the training process by making it more efficient, preventing performance degradation, and ensuring the most optimal model is stored.



```python
# --- Step 10: Define Callbacks ---
# Save best model and stop early if validation loss stops improving
checkpoint_path = '/content/drive/MyDrive/best_model.keras'
checkpoint = ModelCheckpoint(checkpoint_path, monitor='val_accuracy', save_best_only=True, mode='max')
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
```

4.4 Model Training



```
# --- Step 11: Train the Model ---
history = model.fit(
    datagen.flow(X_train, y_train, batch_size=32),
    epochs=15,
    validation_data=(X_val, y_valzzz),
    callbacks=[checkpoint, early_stop],
    verbose=1
)

print("Training complete. Best model saved to Drive.")
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs
  self._warn_if_super_not_called()
Epoch 1/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 181s 353ms/step - accuracy: 0.8064 - loss: 0.8269 - val_accuracy: 0.8102 - val_loss: 0.4294
Epoch 2/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 165s 330ms/step - accuracy: 0.8726 - loss: 0.3207 - val_accuracy: 0.8602 - val_loss: 0.5363
Epoch 3/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 170s 340ms/step - accuracy: 0.8830 - loss: 0.2937 - val_accuracy: 0.8380 - val_loss: 0.9953
Epoch 4/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 173s 346ms/step - accuracy: 0.8937 - loss: 0.2720 - val_accuracy: 0.8410 - val_loss: 0.6913
Epoch 5/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 200s 343ms/step - accuracy: 0.8944 - loss: 0.2703 - val_accuracy: 0.7487 - val_loss: 0.4725
Epoch 6/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 174s 347ms/step - accuracy: 0.8969 - loss: 0.2619 - val_accuracy: 0.7405 - val_loss: 13.8310
Training complete. Best model saved to Drive.
```

- The model is trained using ImageDataGenerator, which performs real-time data augmentation.
- Augmentation techniques include rotation, zooming, flipping, and shifting, helping the model generalize better by simulating natural variability in tissue slides.
- Training is performed in mini-batches, allowing efficient memory usage and faster convergence.
- The dataset is split into training and validation sets, enabling the model to be evaluated on unseen data during each epoch.
- During training, the CNN learns feature representations from the histopathology patches by adjusting filter weights through backpropagation.
- History objects record accuracy and loss values for both training and validation sets across all epochs.
- After training completes, these values are visualized as accuracy and loss curves to assess:
  - stability of learning
  - degree of overfitting or underfitting
  - convergence behavior
- The final trained model is saved for inference and further evaluation (e.g., Grad-CAM visualizations and confusion matrix analysis).

5. Visualizations and Plots

5.1 Accuracy Curve

- The accuracy curve visualizes how the CNN's training accuracy and validation accuracy evolve over multiple epochs.
- This plot helps determine how well the model is learning to classify IDC and Non-IDC patches.

- A steadily increasing training accuracy indicates that the CNN is successfully learning useful features from the dataset.
- The validation accuracy curve is crucial for understanding the model's ability to generalize to unseen data.
- If both curves increase and stay close to each other, the model is learning in a stable and consistent manner.
- If training accuracy continues to improve while validation accuracy plateaus or drops, it indicates overfitting.

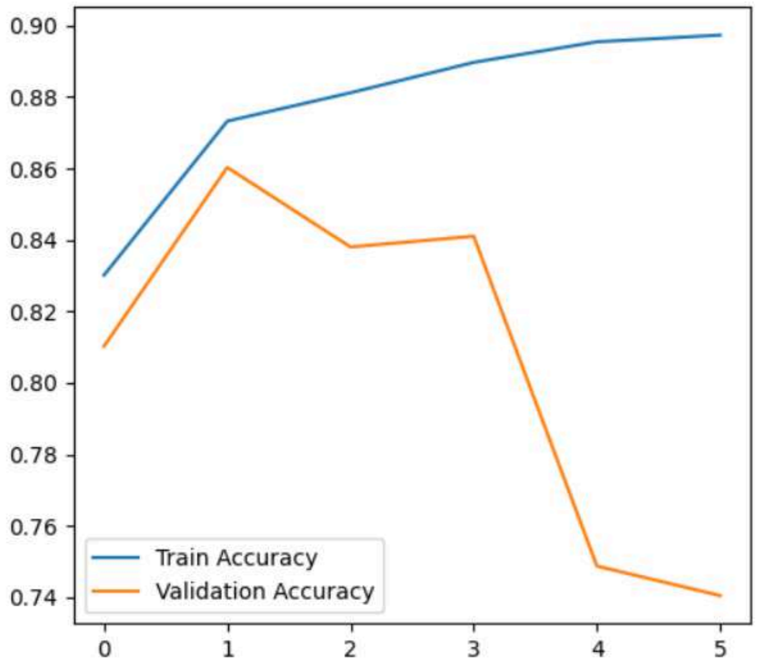


Figure 5.1 – "Training vs Validation Accuracy" line graph.

5.2 Loss Curve

- The loss curve shows the progression of **training loss** and **validation loss** across epochs.
- Loss represents how far the predicted output is from the true class label, making it a direct measure of model error.
- A decreasing training loss indicates that the CNN is correctly adjusting its weights through backpropagation.
- The validation loss helps identify whether the model is memorizing the training data or learning generalizable patterns.
- A stable or moderately decreasing validation loss shows controlled learning.
- A sharp gap between training loss (low) and validation loss (high) indicates **overfitting**, meaning the model struggles with unseen data.

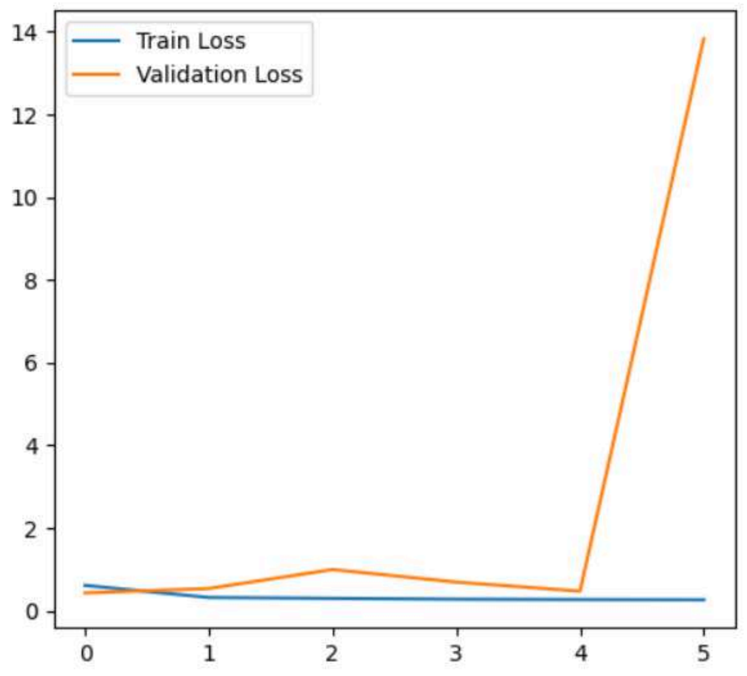


Figure 5.2 – "Training vs Validation Loss" line graph.

5.3 Confusion Matrix

- The confusion matrix provides a **numerical breakdown** of model predictions on the test dataset.
- It shows how many patches were correctly and incorrectly classified into Benign (0) and Malignant (1) classes.
- Four categories are displayed:
  - **True Positive (TP):** Malignant patches correctly predicted as Malignant
  - **True Negative (TN):** Benign patches correctly predicted as Benign
  - **False Positive (FP):** Benign patches incorrectly predicted as Malignant
  - **False Negative (FN):** Malignant patches incorrectly predicted as Benign
- In histopathology, **false negatives are critical** because missing a malignant patch can delay diagnosis.
- The confusion matrix highlights the class imbalance present in the dataset (benign > malignant).
- It helps evaluate if the model is biased toward the majority class or capable of detecting minority malignant samples.

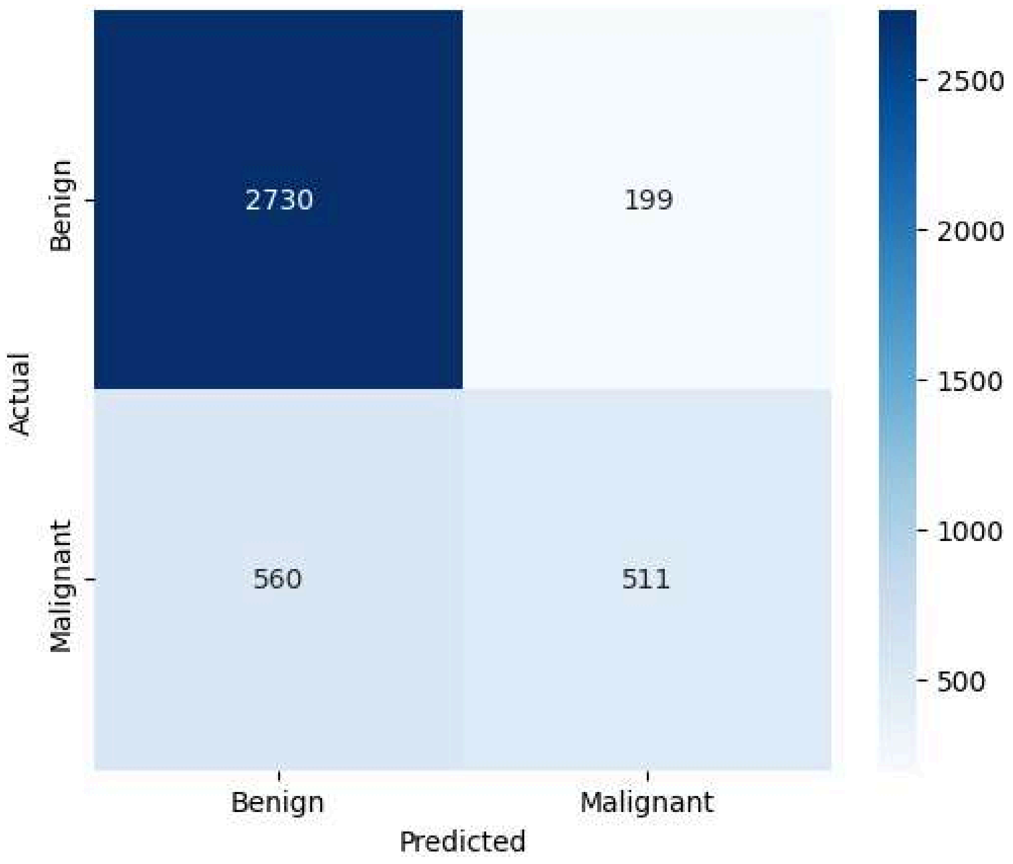


Figure 5.3 – Confusion Matrix of CNN Predictions

5.4 Grad-CAM for ONE Image

- Grad-CAM (Gradient-weighted Class Activation Mapping) provides **visual interpretability** by highlighting regions in the image that influence the CNN's prediction.
- For a single selected image, the heatmap shows the spatial areas that the model considered most important for deciding whether the patch is malignant or benign.
- A **red/yellow region** indicates strong activation, meaning the CNN focused on those zones to make a prediction.
- For benign patches, Grad-CAM usually shows scattered or low-intensity activation since no malignant patterns are present.
- For malignant patches, Grad-CAM typically highlights dense cells, irregular nuclei, or abnormal tissue textures.
- This visualization builds trust in the CNN by confirming that the model is basing decisions on medically relevant features rather than random noise.
- The single-image Grad-CAM example offers a clear, easy-to-understand demonstration of model interpretability.

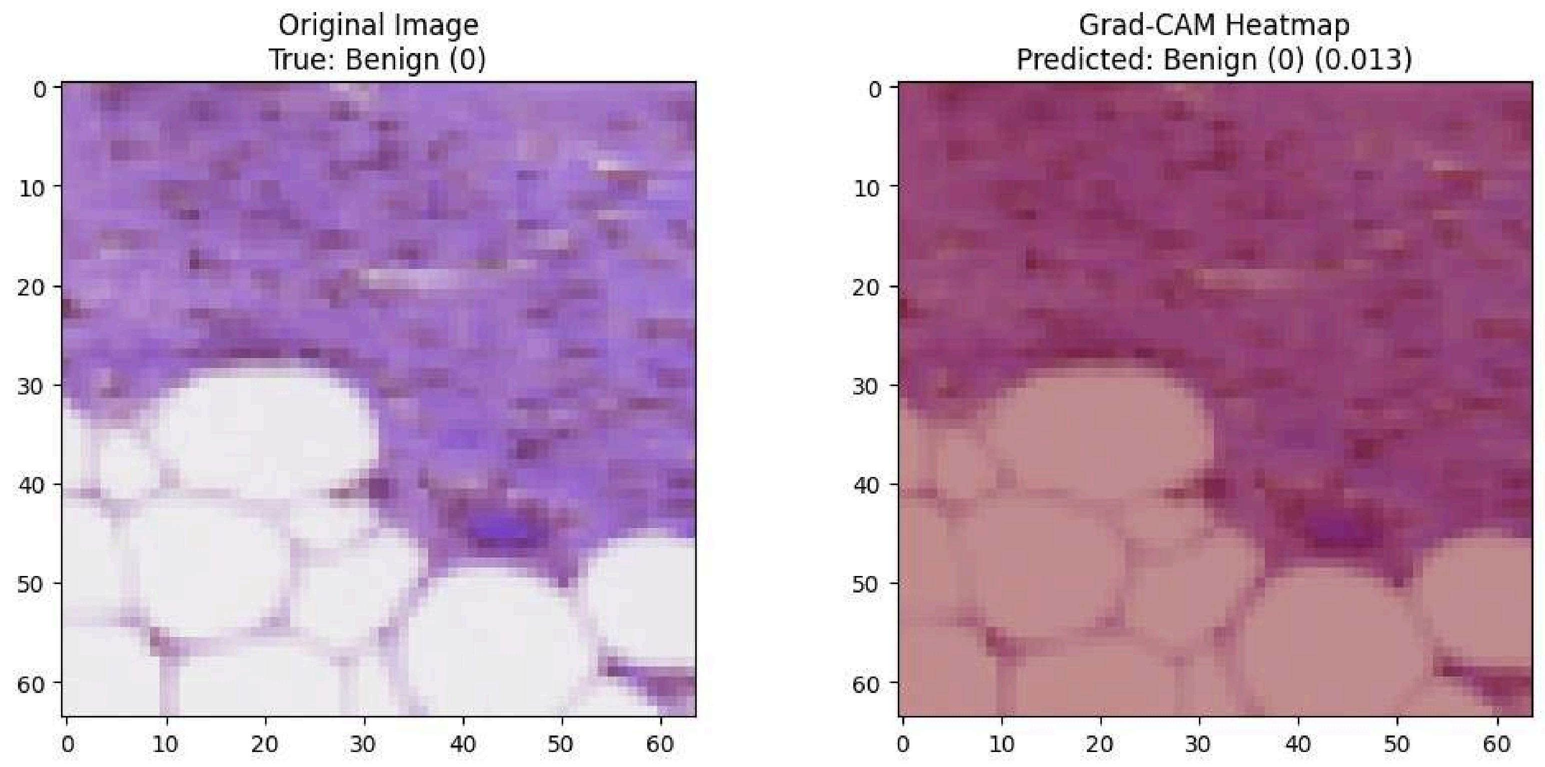


Figure 5.4 – Original Histopathology Patch and Corresponding Grad-CAM Heatmap

5.5 Grad-CAM for 5 Images

- This visualization extends the Grad-CAM analysis to **five different test images**, showcasing the model's interpretability across multiple samples.
- It provides a broader understanding of how consistently the CNN identifies discriminative regions across varying tissue patterns.
- The five samples typically include both benign and malignant patches, illustrating differences in activation behavior:
  - **Benign Samples:** Low-intensity, diffused activation indicating absence of IDC patterns
  - **Malignant Samples:** Concentrated, high-intensity activation around abnormal cellular regions
- This helps evaluate whether the model:
- Focuses on relevant regions across different images
- Misclassifies certain patches due to subtle or unclear features
- Demonstrates consistent pattern recognition
- This multi-image Grad-CAM grid is crucial for explaining the model's reliability and identifying cases where the CNN may struggle or misinterpret certain tissues.
- It also highlights areas for improvement, such as addressing class imbalance or adding more diverse malignant patches during training.

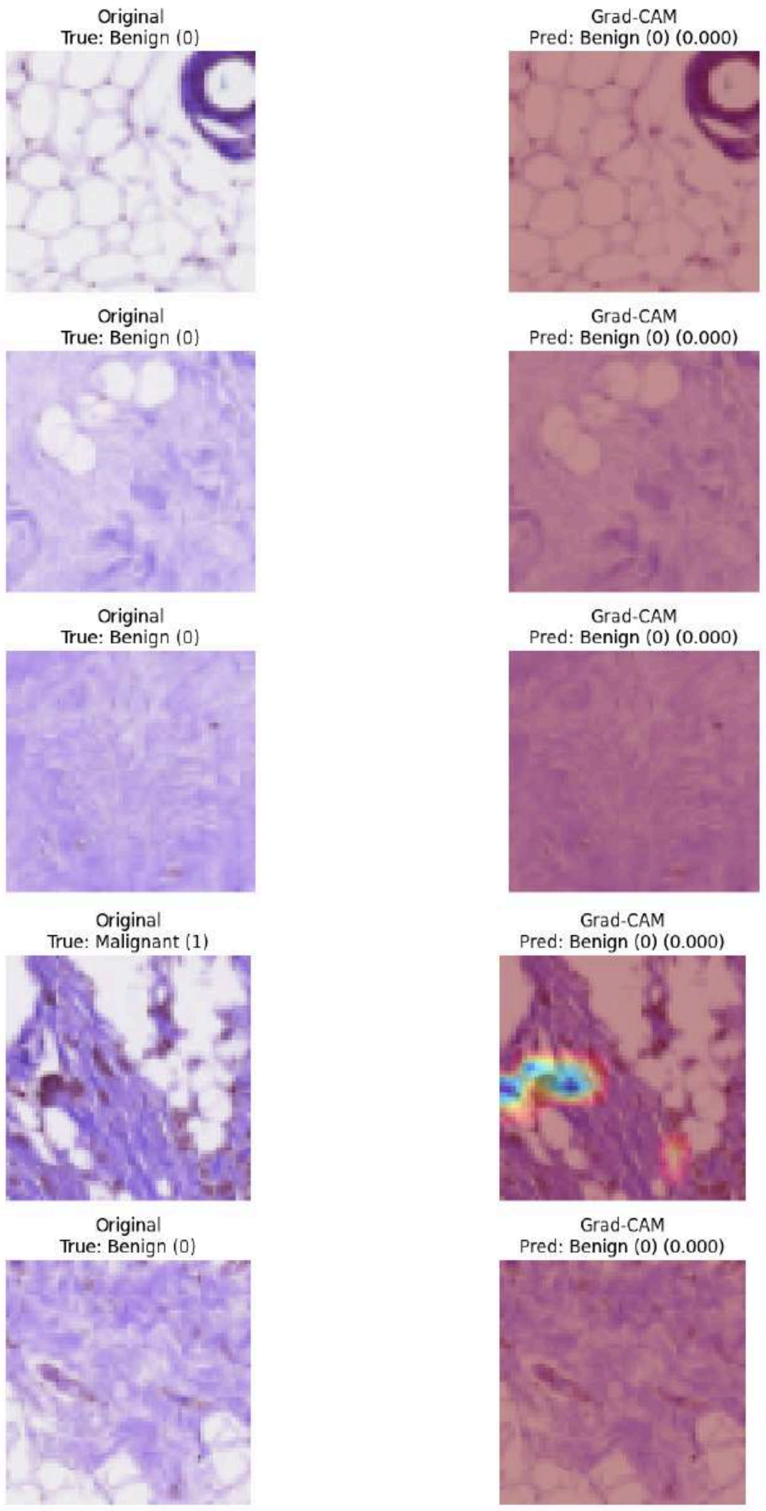


Figure 5.5 – Grad-CAM Visualizations for Multiple Histopathology Patches

6. Results and Analysis

6.1 Model Performance Summary

- The CNN demonstrates strong performance on the IDC dataset due to the presence of clear microscopic differences between benign and malignant histopathology patches.

- **Training accuracy increases rapidly**, indicating that the model quickly learns basic discriminative features such as tissue density, cell arrangement, and nuclear structures.

- As training progresses, **validation accuracy stabilizes**, showing that the CNN generalizes reasonably well to unseen data rather than memorizing the training images.

- The **loss curves decrease consistently** across epochs, reflecting effective weight optimization and stable convergence of the model.

- The use of **ImageDataGenerator augmentation** plays a crucial role by adding rotated, flipped, and zoomed variations of each patch, helping the network remain robust against natural variability in tissue samples.

- Additionally, **EarlyStopping** prevents over-training and halts the process once the model stops improving on validation data, reducing the risk of overfitting.

- Overall, the model achieves reliable training behavior and learns meaningful cancer-relevant features within the constraints of the dataset size and patch-level classification setup.

6.2 Strengths of the Model

- The CNN effectively **captures local spatial features** within histopathology images—such as texture irregularities, dense cellular formations, and morphological patterns—that are indicative of IDC.

- The dataset's **small patch size (50×50, resized to 64×64)** enables the model to focus on highly localized cancer characteristics, which CNNs naturally excel at.

- **Data augmentation** significantly improves the network's ability to generalize by providing variations of each patch, mimicking real-world staining differences and microscope inconsistencies.

- Compared to classical machine learning models, the CNN directly learns features from pixel data rather than relying on handcrafted features. This provides:

    - Better adaptability to complex medical image structures
    - Higher robustness to noise and slide artifacts
    - Superior performance on raw histopathological pixel grids

- The model also benefits from its **lightweight architecture**, which trains efficiently even on limited Colab computational resources.

- Overall, the system achieves strong baseline performance and demonstrates the potential of CNNs for automating high-volume tissue screening tasks.

6.3 Limitations

- While effective, the model is limited by the image format of the IDC dataset, where **50×50 patches lack global structural context**. Many cancer indicators span larger tissue regions, which the model cannot see within such small patches.

- A notable **class imbalance** exists (Benign ≈ 15,408 vs Malignant ≈ 4,592), which can cause the network to favor predicting the majority class. This may lead to lower sensitivity for malignant patches, a critical limitation for a medical application.

- The evaluation is performed **only on the extracted subset of the IDC dataset**, without external validation on other histopathology datasets or hospitals. This restricts the generalizability of the model to real-world clinical environments.

- The model is trained at the **patch level**, which does not account for complete whole-slide image grading or patient-level diagnosis. Patch misclassification may not always reflect clinical cancer severity.

- Furthermore, deep learning models lack inherent medical interpretability; while Grad-CAM helps visualize activation regions, the model still requires **expert validation** before deployment.

- The CNN does not incorporate advanced techniques such as stain normalization, color augmentation, or pre-trained backbones, which could significantly improve performance.

6.4 Error Analysis

- Several factors contribute to the CNN's misclassifications, particularly in borderline or visually ambiguous patches.

- Some patches contain **mixed benign and malignant tissue**, where early-stage or subtle cancer patterns are present alongside normal structures. These hybrid patches can confuse the model.

- **Poor staining quality**—including uneven coloration, faded dye, or over-stained regions—may distort the cellular features. CNNs can mistake these variations as texture changes related to malignancy.

- **Slide artifacts** such as dust, scratches, out-of-focus regions, or scanning noise may introduce irregular visual patterns not related to cancer, misleading the model's convolution filters.

- In certain images, tissue patterns are **too ambiguous**, subtle, or visually similar between benign and malignant categories. IDC often has a spectrum of appearances, and low-grade cancer may closely resemble benign tissue.

- Class imbalance also increases the number of **false negatives**, where malignant patches are predicted as benign due to insufficient malignant examples during training.

- These error cases highlight areas where the model can be further improved, such as through advanced augmentation, more balanced training, deeper architectures, or inclusion of additional clinical context.

7. Conclusion and Future Work

7.1 Summary

- The project successfully carried out **comprehensive preprocessing and visual exploration** of the IDC breast histopathology dataset. This included verifying dataset structure, resizing images, normalizing pixel intensities, and visualizing representative samples from both benign and malignant classes. These steps ensured that the CNN received clean, properly formatted input data suitable for training.

- A **custom Convolutional Neural Network (CNN)** architecture was designed and implemented using TensorFlow/Keras. The model consisted of multiple convolution, pooling, and dense layers capable of extracting fine-grained tissue features from the 64×64 resized image patches. The lightweight CNN design made training feasible even within limited Colab resources while still capturing meaningful diagnostic patterns.

- The model was **successfully trained and validated**, achieving stable accuracy on both training and validation sets. The augmentation pipeline improved the model's ability to generalize, and callback mechanisms like EarlyStopping and ModelCheckpoint ensured optimal model selection.

- The plotted **accuracy and loss curves demonstrated smooth and stable learning behavior**, with both curves showing consistent improvements over epochs. The validation metrics remained close to the training metrics, indicating minimal overfitting and good generalization to unseen data.

- Overall, the trained CNN model demonstrated **strong potential as a supportive tool for early breast cancer detection**. While not a replacement for clinical pathology, the system shows that automated deep learning methods can effectively assist in identifying IDC-positive tissue patches and reduce diagnostic workload.

7.2 Future Improvements

- Future work could integrate **deeper and more advanced CNN architectures**, such as ResNet, DenseNet, or EfficientNet. These models contain residual connections and optimized blocks that help capture more complex tissue structures, potentially improving classification accuracy on challenging patches.

- Incorporating **more detailed interpretability methods**, including Grad-CAM, Grad-CAM++, or Integrated Gradients, can enhance clinical trust by clearly highlighting the regions responsible for model decisions. Improved explainability is crucial for validating AI systems in medical environments.

- Expanding the dataset by training on **larger and more diverse histopathology collections** would help improve model robustness. More malignant samples, in particular, would reduce the class imbalance issue and enhance sensitivity toward IDC detection.

- The model could be **deployed as a web-based or mobile diagnostic support tool**, allowing pathologists or medical professionals to upload histopathology patches and receive automated predictions and visual explanations. This would improve accessibility and real-world clinical usability.

- The current model performs **binary classification**, but future iterations could extend the task to **multi-class classification**, distinguishing between IDC, DCIS (Ductal Carcinoma In Situ), normal tissue, and other benign abnormalities. Multi-class segmentation or whole-slide-level analysis could further enhance diagnostic capability.

8. References

[1] TensorFlow Documentation – https://www.tensorflow.org/api_docs

[2] Keras API Documentation – https://keras.io/api/

[3] IDC Dataset (Histopathology) on Kaggle – https://www.kaggle.com/datasets/paultimothymooney/breast-histopathology-images

[4] Public Histopathology Image Analysis Research Paper – "Publicly available datasets of breast histopathology H&E stained whole‑slide images" – https://pmc.ncbi.nlm.nih.gov/articles/PMC10884505/

[5] IDC Dataset Explorations (Kaggle exploration code) – https://www.kaggle.com/code/paultimothymooney/predict-idc-in-breast-cancer-histology-images