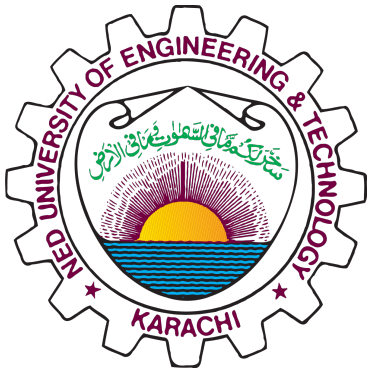
Breast Tumor Classification using Deep Learning



Group Members:

Shoaib Ul Haq CS-21025

Muhammad Haris Khan CS-21033

Course: CS 436 - Computer Vision

Submitted to: Miss Fauzia Yasir

Rubric Page

GRADING RUBRIC

Software Use Rubric				
Skill Sets	Extent of Achievement			
	0	1	2	3
To what extent has the student implemented the solution?	The solution has not been implemented.	The solution has syntactic and logical errors.	The solution has syntactic or logical errors.	The solution is syntactically and logically sound for the stated problem parameters.
How efficient is the proposed model?	The model does not address the problem adequately.	The model exhibits redundancy and partially covers the problem.	The model exhibits redundancy or partially covers the problem.	The model is free of redundancy and covers all aspects of the problem.
How did the student answer questions relevant to the solution?	The student answered none of the questions.	The student answered less than half of the questions.	The student answered more than half but not all of the questions.	The student answered all the questions.
To what extent the student use libraries, tools and download database of images?	The student is unable with the interface.	The student is familiar few library function.	-	The student is proficient with the interface and able to use libraries and download dataset.

Weighted CLO Score	
Remarks	
Instructor's Signature with Date	

Project Evaluation Rubric

1 Application Background and Introduction

1.1 Background

Breast cancer remains a significant global health concern, being one of the most common cancers among women. Early and accurate detection is crucial for improving patient prognosis and survival rates. Medical imaging techniques like mammography, MRI, and ultrasound play vital roles in screening and diagnosis. Ultrasound is particularly valuable due to its non-invasive nature, lack of ionizing radiation, cost-effectiveness, and ability to image dense breast tissue effectively. However, interpreting ultrasound images can be challenging and subjective, often requiring experienced radiologists.

1.2 Problem Statement

The manual interpretation of breast ultrasound images can suffer from inter-observer variability and requires significant expertise. There is a need for automated, reliable tools to assist radiologists in classifying breast lesions identified in ultrasound scans as benign, malignant, or normal tissue.

1.3 Project Objective

This project aims to develop and evaluate a deep learning model for the automated classification of breast ultrasound images from the Breast Ultrasound Images (BUSI) dataset into three categories: Benign, Malignant, and Normal. We utilize a transfer learning approach based on the DenseNet121 architecture.

1.4 Motivation

By leveraging deep learning, we aim to create a system that can potentially serve as a supplementary tool for radiologists, potentially speeding up the diagnostic process and improving the consistency of interpretations, ultimately contributing to better patient care through earlier and more accurate cancer detection

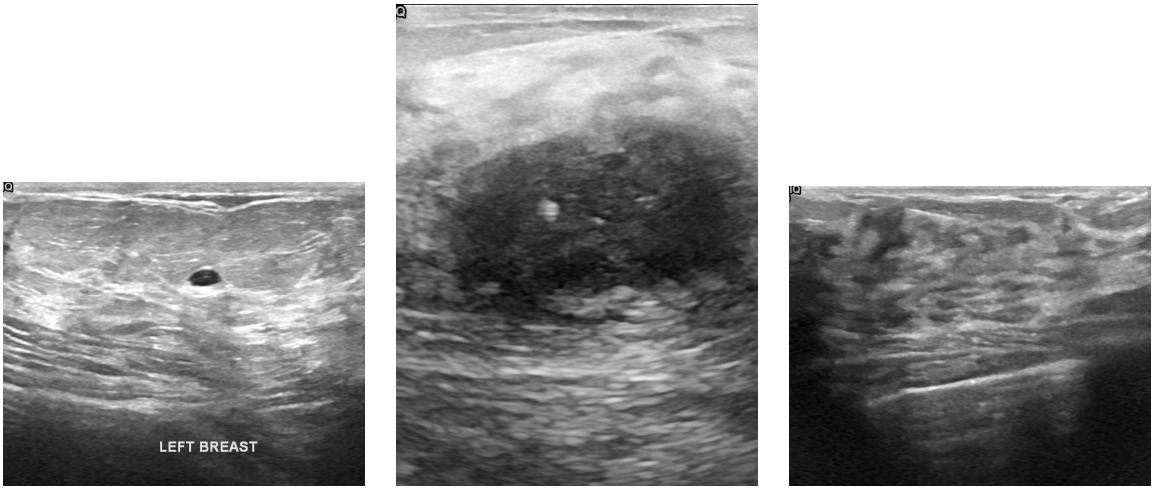
2 Dataset

2.1 Dataset Used

The project utilizes the public "Breast Ultrasound Images Dataset (BUSI)

2.2 Description

The dataset contains grayscale ultrasound images categorized into three classes: Normal, Benign, and Malignant. Each primary image is often accompanied by a segmentation mask image showing the lesion boundary, which is not used for this classification task



(a) Benign (b) Malignant (c) Normal

Figure 1: Sample Ultrasound Images from the BUSI Dataset for each class.

2.3 Original Distribution (Approximate)

- Benign: 437 images
- Malignant: 210 images
- Normal: 133 images
- Total: 780 images

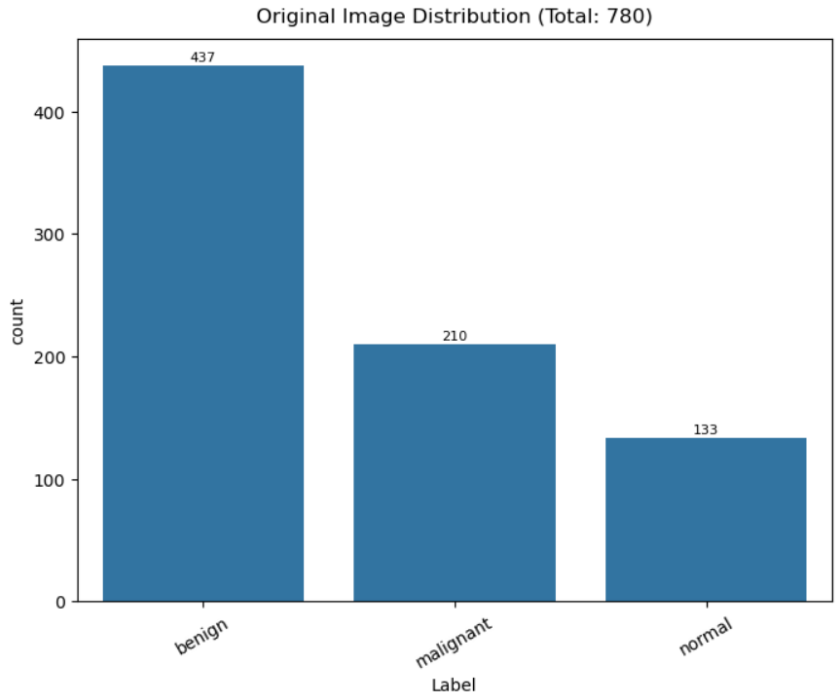


Figure 2: Dataset Class Distribution

2.4 Data Filtering

During data loading, files identified as segmentation masks (typically containing (`_mask`) in their filenames) were explicitly excluded to ensure only the original ultrasound images were used for classification training and evaluation.

2.5 Class Imbalance

The dataset exhibits some class imbalance, with significantly more benign samples than malignant or normal ones. This was addressed partially through stratified splitting during the data preparation phase.

3 Preprocessing for CV Pipeline

The following preprocessing steps were applied:

1. **Image Reading:** Images were read using OpenCV (cv2.imread). Grayscale images were converted to 3-channel RGB format, as the pre-trained DenseNet model expects 3-channel input.

2. **Resizing:** All images were resized to a fixed dimension of 256×256 pixels using linear interpolation (`cv2.INTER_LINEAR`). This ensures uniform input size for the neural network.

3. **Data Augmentation (Minimal):** To artificially increase the diversity of the training data and improve model robustness, minimal augmentation was applied only to the training set using the `Albumentations` library. Based on experimental results where more extensive augmentation degraded performance, the final model used only:

    - **Horizontal Flip**: Randomly flipped images horizontally with a probability of 50% (`A.HorizontalFlip(p=0.5)`).

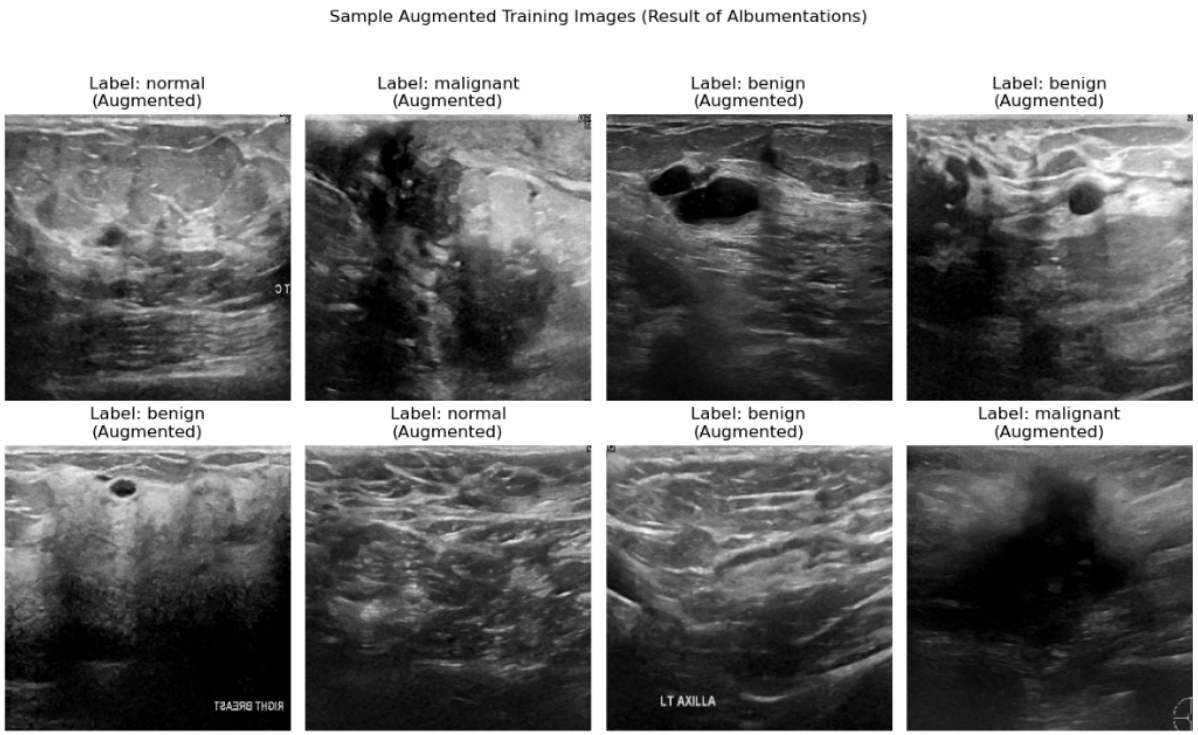


Figure 3: Example of Minimal Augmentation Effects (Resize + Horizontal Flip)

Note: Validation and test sets only underwent resizing and normalization, without random augmentation.

4. **Normalization:** Pixel values were cast to `float32`. Instead of applying simple normalization (e.g., `/ 255.0`), the preprocessing function `preprocess_input` from `tf.keras.applications.densenet` was used. This function performs mean subtraction and standard deviation scaling aligned with the original ImageNet training of DenseNet, ensuring compatibility and effectiveness in transfer learning.

5. **Data Splitting:** The dataset was split into training (80%), validation (10%), and test (10%) sets using `sklearn.model_selection.train_test_split`. Stratified sampling (`stratify = y`) was applied to preserve the relative class distribution across subsets. A fixed `random_state = 123` was used to ensure reproducibility.

6. **Label Encoding:** String labels (`benign`, `malignant`, `normal`) were encoded as integers (0, 1, 2) using `LabelEncoder` from `sklearn.preprocessing`.

7. **One-Hot Encoding:** The numerical labels were further converted into a one-hot encoded vector format (e.g., $[1, 0, 0]$ for benign) using `tf.keras.utils.to_categorical`. This is the required format for the model's output layer using `'softmax'` activation and `'categorical_crossentropy'` loss.

8. **Data Generators:** Custom Keras `Sequence` data generators were implemented to efficiently load, preprocess, and augment data in batches during training and evaluation.

4 AI/ML Model Architecture

A transfer learning approach was adopted, leveraging the powerful feature extraction capabilities of a pre-trained convolutional neural network (CNN).

4.1 Base Model

DenseNet121 pre-trained on the ImageNet dataset was used as the base feature extractor. The `tensorflow.keras.applications.DenseNet121` implementation was utilized.

- `include_top=False`: The original ImageNet classification layer (top layers) of DenseNet121 was removed.
- `weights='imagenet'`: Pre-trained weights from ImageNet were loaded. (Or specify local path if used: `./model/weights/...`).
- `input_shape=(256,256,3)`: The input shape was set to match the preprocessed image dimensions.
- **Freezing:** All layers of the DenseNet121 base model were frozen (`base_model.trainable = False`) so that their pre-trained weights were not updated during the initial training phase. This allows the model to act as a fixed feature extractor.

4.2 Custom Classification Head

New layers were added on top of the frozen DenseNet121 base to perform the classification task specific to the breast ultrasound dataset:

1. **Flatten Layer**: Flattens the 3D feature map output from DenseNet121 into a 1D vector.
2. **Dense Layer 1**: Fully connected layer with 1024 neurons and ReLU activation function.
3. **Dropout Layer 1**: Dropout regularization with a rate of 0.4 (40%) to prevent overfitting.
4. **Dense Layer 2**: Fully connected layer with 512 neurons and ReLU activation.
5. **Dropout Layer 2**: Dropout regularization with a rate of 0.2 (20%).
6. **Dense Layer 3**: Fully connected layer with 128 neurons and ReLU activation.
7. **Output Layer**: Final Dense layer with 3 neurons (one for each class: benign, malignant, normal) and a Softmax activation function to output class probabilities.

4.3 Model Summary (Text representation)

```
Model: "Breast_Tumor_Classifier_Minimal_Augment"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet121 (Functional)    (None, 8, 8, 1024)        7,037,504 (Non-
   trainable)
 flatten (Flatten)           (None, 65536)             0
 dense_1024_1 (Dense)        (None, 1024)              67,109,888
```

```
 dropout_1 (Dropout)         (None, 1024)              0
 dense_512 (Dense)           (None, 512)               524,800
 dropout_2 (Dropout)         (None, 512)               0
 dense_128 (Dense)           (None, 128)               65,664
 output (Dense)              (None, 3)                 387
=================================================================
Total params: 74,737,843
Trainable params: 67,700,339
Non-trainable params: 7,037,504
```

Listing 1: Model Architecture Summary

5 Experimental Setup

5.1 Code Repository

The source code for this project, including the training notebooks, testing scripts, and Streamlit application files, is publicly available on GitHub.

GitHub Repository URL: `https://github.com/hariskhan-hk/Tumor_Classification`

5.2 Hardware

Training and testing were performed primarily on CPU

5.3 Software Libraries

- Operating System: Ubuntu 20.04
- Programming Language: Python 3.8
- Core Libraries: TensorFlow 2.9.0, Albumentations 1.4.18, OpenCV, Scikit-learn, Pandas, NumPy, Matplotlib, Streamlit

5.4 Training Parameters

- Optimizer: Adam
- Learning Rate: 0.0001
- Loss Function: Categorical Crossentropy
- Metrics: Accuracy
- Batch Size: 16
- Epochs: Up to 50 (with Early Stopping)
- Callbacks: `ModelCheckpoint`, `EarlyStopping`

6 Result

The final model (trained with minimal augmentation) was evaluated on the training, validation, and held-out test sets. The performance metrics reported below correspond to the best model saved during training based on validation accuracy.

6.1 Overall Performance Metrics

Table 1: Model Performance Summary

Set	**Loss**	**Accuracy**
Training	0.0098	99.68%
Validation	0.4164	91.03%
Test	**0.6009**	**89.74%**

6.2 Discussion

The model achieved very high accuracy on the training set, indicating that it learned the training data well. The validation accuracy (91.03%) is good, suggesting reasonable generalization. The final performance on the unseen test set is 89.74% accuracy. The requirement to incorporate augmentation (even minimal) led to a slight decrease in overall test accuracy but potentially improved robustness.

6.3 Test Set Classification Report

```
              precision    recall  f1-score   support

      benign       0.89      0.93      0.91        44
   malignant       0.89      0.76      0.82        21
      normal       0.93      1.00      0.96        13

    accuracy                           0.90        78
   macro avg       0.90      0.90      0.90        78
weighted avg       0.90      0.90      0.90        78
```

Listing 2: Test Set Classification Report

6.4 Test Set Confusion Matrix

```
        Predicted: Benign Malig Norm
True Benign      [[41     2     1]
True Malignant    [ 5    16     0]
True Normal       [ 0     0    13]]
```

Listing 3: Confusion Matrix (Numerical)

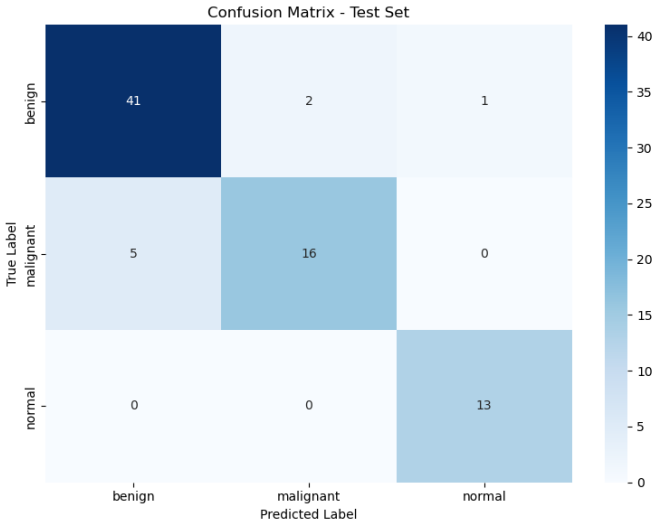


Figure 4: Confusion Matrix Visualization for the Test Set

6.5 Class-wise Performance Analysis

The model performs very well on **Normal** cases, achieving perfect recall and high precision. Performance on **Benign** cases is also strong, with 93% recall and 89% precision. For the **Malignant** class, the model shows good precision (89%)—when it predicts malignant, it is often correct. However, the recall is lower at 76%, meaning the model missed 24% (5 out of 21) of the actual malignant cases in the test set, primarily classifying them as benign. This is the area for potential improvement.

7 Necessary Post Processing for CV Pipeline

For this classification task, the post-processing applied to the model's raw output is straightforward:

1. **Probability Output**: The final Softmax layer of the model outputs a vector of three probabilities, representing the model's confidence that the input image belongs to each of the classes (Benign, Malignant, Normal). These probabilities sum to 1.

2. **Argmax Selection**: To obtain the final predicted class, the `argmax` function (e.g., `numpy.argmax`) is applied to the probability vector. This selects the index corresponding to the class with the highest probability.

3. **Label Mapping**: The predicted index (0, 1, or 2) is mapped back to its corresponding human-readable string label (`'Benign'`, `'Malignant'`, or `'Normal'`) using the mapping derived from the `LabelEncoder`.

4. **Confidence Score**: The probability associated with the predicted class index is presented as the confidence score for the prediction.

This post-processing translates the model's numerical output into a clear classification result and a measure of confidence.

8 Final Output in Terms of Metric, Images

8.1 Final Metric

The key performance indicator for this model on the unseen test data is an **Accuracy of 89.74%**.

8.2 Detailed Performance

Refer to the Classification Report and Confusion Matrix in Section 8 for per-class performance details, noting the 76% recall for the critical **Malignant** class.

8.3 Qualitative Output (Images/Predictions)

The developed Streamlit application provides a visual interface where users can upload an ultrasound image. The application preprocesses the image, feeds it to the trained model, and displays:

- The uploaded image.
- The predicted class label (`Benign`, `Malignant`, or `Normal`), often highlighted.
- The confidence score (probability) for the prediction.
- A bar chart showing the probability distribution across all three classes.
- Informational text about the predicted condition.

(Refer to screenshots in Section 9 for examples).

9 Screenshots of Interface

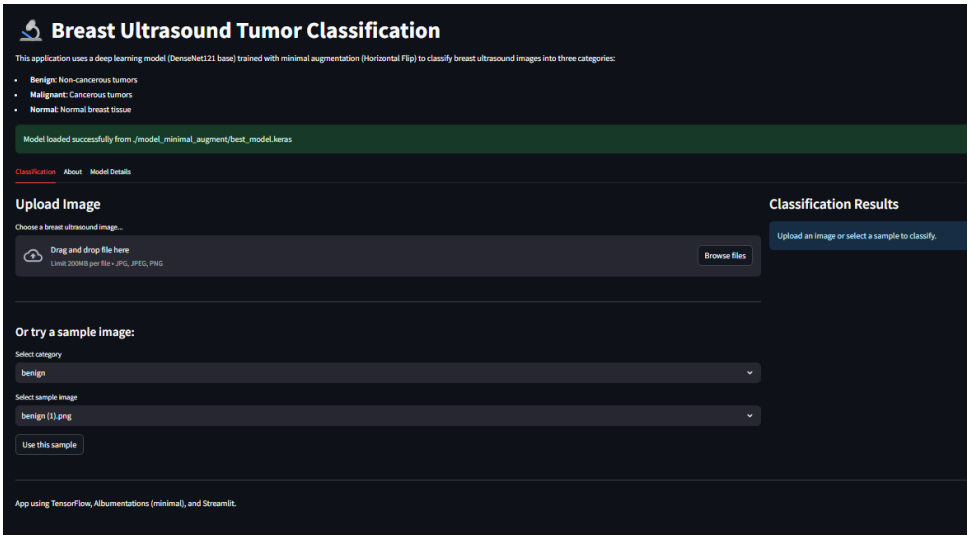


Figure 5: Main interface of the Streamlit application showing the image upload area and sample selection.

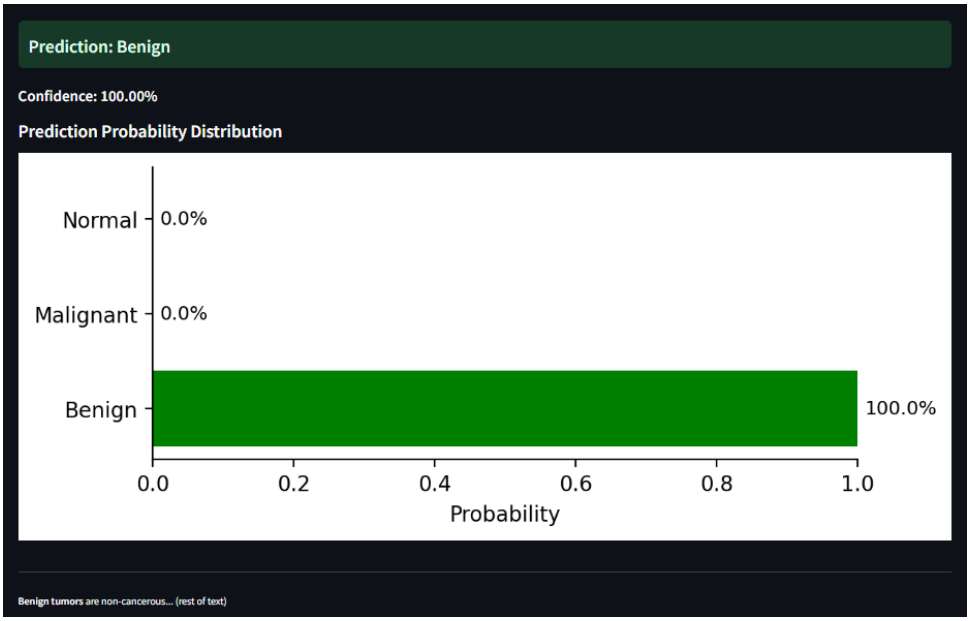


Figure 6: Example output for an image classified as 'Benign'.

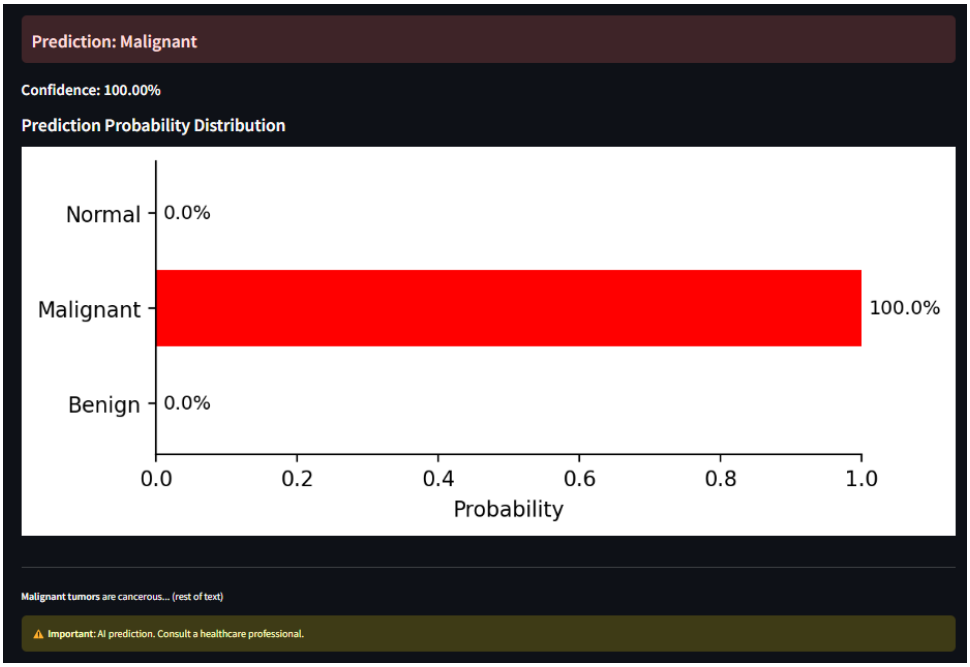


Figure 7: Example output for an image classified as 'Malignant'.

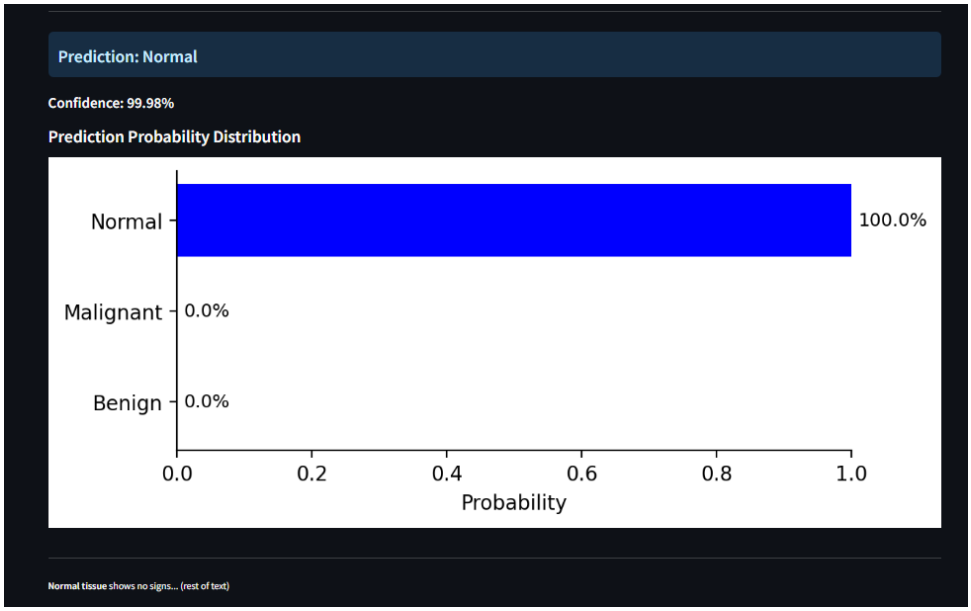


Figure 8: Example output for an image classified as 'Normal'.

Code Repository

The complete source code for this project, including notebooks, application files, and related resources, can be found on GitHub at the following URL:

`https://github.com/hariskhan-hk/Tumor_Classification`