# TumorTrace : MRI-Based AI for Breast Cancer Detection

Using Deep Learning to Identify Benign and Malignant Tumors

Kumud Hasija

Infosys Springboard 5.0

Role: AI&ML Intern

# Abstract

Breast cancer is a significant global health challenge, with early and accurate diagnosis being critical to improving survival rates and patient outcomes. TumorTrace leverages advanced deep learning technologies to develop an automated system for classifying breast tumors as benign or malignant using MRI images. By employing cutting-edge convolutional neural network architectures like ResNet18, VGG16, and ResNet50, alongside techniques such as data augmentation, early stopping, and regularization, the project enhances the robustness and reliability of diagnostic predictions.

With a focus on optimizing preprocessing and feature extraction, the system processes MRI datasets to highlight critical tumor characteristics, reducing diagnostic errors and supporting radiologists in clinical decision-making. TumorTrace represents a significant advancement in AI-driven medical imaging, paving the way for streamlined diagnostic workflows, reduced human dependency, and improved healthcare outcomes. Future efforts will focus on expanding dataset diversity, refining model performance, and deploying the system in real-world clinical environments to revolutionize breast cancer diagnostics .

*Keywords*: Breast cancer detection, MRI classification, deep learning, convolutional neural networks, AI in medical imaging, automated diagnostics, feature extraction.

# Introduction

## Background

Breast cancer is a critical global health issue, affecting millions of women annually and remaining one of the leading causes of cancer-related mortality. Early diagnosis plays a pivotal role in improving survival rates and enabling timely and effective treatment. Traditional diagnostic methods, such as mammography, ultrasound, and biopsy, rely heavily on the expertise of medical professionals. However, these methods can be time-consuming, prone to human error, and inconsistent due to variability in tumor presentation and differences in radiologist interpretation.

Advancements in artificial intelligence (AI) and machine learning (ML) have opened new possibilities for enhancing the diagnostic process. Deep learning models, particularly convolutional neural networks (CNNs), have shown significant potential in automating image analysis tasks, offering accurate, consistent, and scalable solutions for medical imaging challenges.

**Problem Statement**

Despite advancements in technology, diagnosing breast cancer remains complex due to:

1. **Tumor Variability**: Tumors differ in size, shape, texture, and location, complicating manual interpretation.
2. **Risk of Misclassification**: Subtle differences between benign and malignant tumors can lead to diagnostic errors, especially in ambiguous cases.
3. **Growing Data Volume**: The increasing availability of imaging data places immense pressure on radiologists, creating a need for scalable and efficient diagnostic systems.

These challenges necessitate the development of robust, automated solutions capable of reducing human dependency and improving diagnostic accuracy and consistency.

**Objectives**

This project, **TumorTrace**, aims to address these challenges through the objectives:

1. **Model Development**: Build a deep learning-based classification system to distinguish between benign and malignant breast tumors using MRI scans.
2. **Optimization**: Enhance the model's reliability and generalization capabilities through advanced techniques, including data preprocessing, augmentation, regularization, and early stopping.
3. **Performance Evaluation**: Analyze the system's performance using metrics such as sensitivity, specificity, precision, and recall to ensure clinical applicability.
4. **Clinical Support**: Provide a scalable diagnostic tool that supports radiologists in making accurate and timely decisions, thereby improving patient outcomes.

# Dataset Overview

## Source

The dataset used for the TumorTrace project comprises **MRI breast cancer images**, which are labeled into two primary categories: **benign** and **malignant**. These images were acquired from publicly available and reputable medical imaging repositories. The dataset was chosen for its diversity and reliability, ensuring it serves as a robust foundation for training and evaluating deep learning models designed for tumor classification.

## Structure

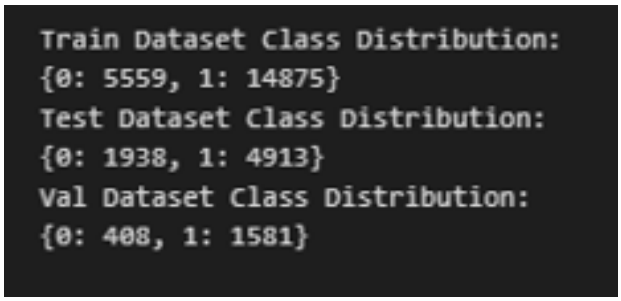
To facilitate the development and validation of the model, the dataset was divided into three subsets:

- **Training Set**: Comprising **20,434 images** (70% of the total dataset), this subset was used to train the model and learn underlying patterns.
- **Validation Set**: Comprising **1,989 images** (10% of the dataset), this subset was used for fine-tuning model hyperparameters and monitoring overfitting.
- **Test Set**: Comprising **6,851 images** (20% of the dataset), this subset was reserved for evaluating the model's performance on unseen data.

This split ensures balanced representation of benign and malignant cases in all subsets, enabling the model to learn effectively while maintaining generalizability.

**Challenges with the Dataset**

1. **Variability in Tumors**: Tumors in MRI images exhibit significant differences in size, shape, texture, and intensity, making classification challenging.
2. **Class Imbalance**: While efforts were made to ensure balance, slight variations in class distribution required the use of advanced techniques to prevent bias.
3. **Noise in Data**: MRI images may include artifacts or irrelevant features that need to be addressed during preprocessing.



# Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a specialized type of deep learning model designed for image classification tasks. Unlike traditional neural networks, CNNs automatically learn spatial hierarchies of features, making them particularly effective for image data. CNNs have been widely adopted in medical imaging, including breast cancer detection, due to their ability to process and classify complex image patterns.

**How CNNs Work**

1. **Convolutional Layer:** Detects basic features like edges, textures, and patterns by applying filters to the image.
2. **Activation Function (ReLU):** Adds non-linearity to the model, enabling it to learn complex patterns.
3. **Pooling Layer:** Reduces the spatial size of the image, retaining only the most important features and improving computational efficiency.
4. **Fully Connected Layer:** Combines the learned features to make predictions about the image.
5. **Softmax Activation:** Converts the output into class probabilities, enabling the model to classify the image (benign or malignant).

**CNNs in TumorTrace**

In the TumorTrace project, three CNN architectures—**VGG16**, **ResNet18**, and **ResNet50**—were used. These models were chosen for their proven success in image classification. ResNet18, with its residual connections, is lightweight and efficient, while ResNet50 is deeper, enabling it to capture more complex patterns in MRI images.

# Data Augmentation

Data augmentation is a crucial technique in training deep learning models, especially when working with limited data. It helps increase the size of the dataset by creating modified versions of the existing images. This prevents overfitting and enables the model to generalize better to unseen data. In the TumorTrace project, various data augmentation techniques were applied to the training dataset to simulate real-world variations and improve model robustness.

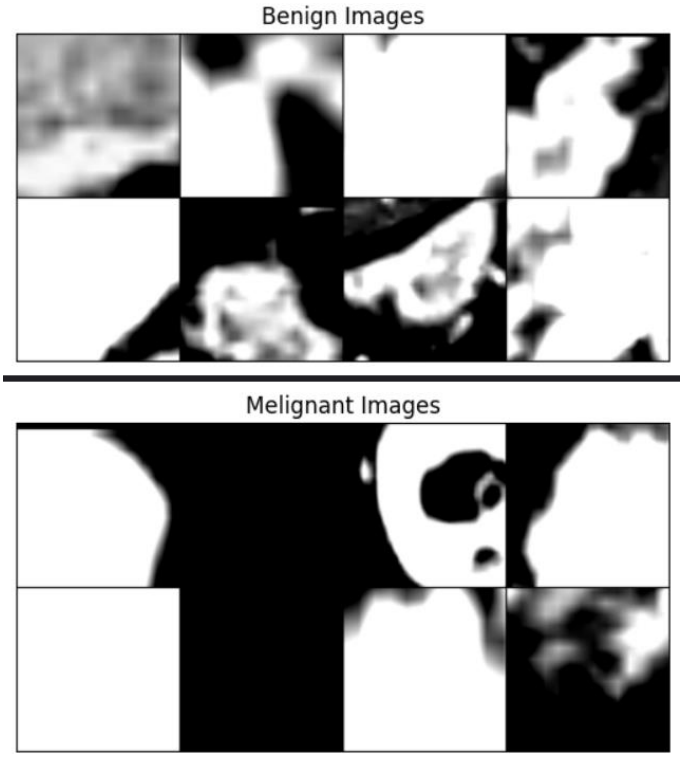
For the **TumorTrace project**, the following augmentation techniques were applied:

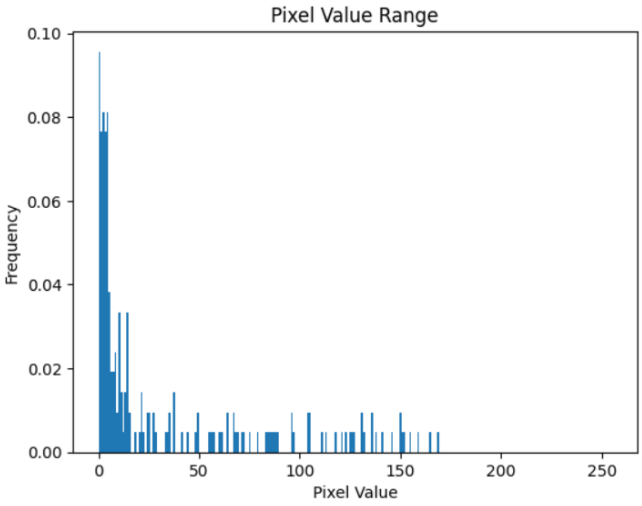
1. **Resize**: Images are resized to **224x224 pixels**, ensuring consistency and compatibility with the model's input size.
2. **Random Affine**: The images are randomly translated (shifted) by up to **10%** in both horizontal and vertical directions, simulating changes in tumor position within the image.
3. **Center Crop**: The center of the image is cropped to **224x224 pixels**, ensuring that the key features, like tumors, remain in focus.
4. **Color Jitter**: Random adjustments to the **brightness** of the images simulate variations in lighting and scanning conditions.
5. **Random Resized Crop**: Randomly crops a portion of the image and resizes it to **224x224 pixels**, introducing scale variations and forcing the model to focus on different areas of the tumor.
6. **Random Horizontal and Vertical Flip**: Each image has a **50% chance** of being flipped horizontally and vertically, simulating different orientations of the tumor.
7. **Normalize**: The image is normalized with a mean and standard deviation of **(0.5, 0.5, 0.5)** for each color channel, ensuring consistent input values for the model.

**Benefits of Data Augmentation:**

- **Increases Dataset Diversity**: By applying these transformations, the model is exposed to a wider variety of image variations, which improves generalization.
- **Improves Robustness**: The model learns to recognize tumors under different conditions such as changes in lighting, rotation, and scale.
- **Prevents Overfitting**: With more varied data, the model is less likely to memorize specific patterns from the training set and instead learns more generalizable features.

**Sample Images after data augmentation –**



# Feature Extraction

Feature extraction helps enhance the model's ability to identify and classify tumors by focusing on key patterns in MRI images. In **TumorTrace**, traditional feature extraction methods complement the CNNs by capturing important details such as texture, shape, and edges.

**Key Techniques Used:**

1. **Histogram of Oriented Gradients (HOG)**: Detects structural features like edges, crucial for tumor boundary identification.
2. **Gray-Level Co-Occurrence Matrix (GLCM)**: Captures textural features like contrast, energy, and homogeneity, useful for distinguishing benign from malignant tumors.
3. **Sobel Operator**: Highlights sharp contrasts and edges, helping identify the tumor's shape.
4. **Local Binary Pattern (LBP)**: Analyzes local texture patterns, aiding in recognizing fine differences in tumor texture.
5. **Multi-Variance Median LBP (MVM-LBP)**: Extends LBP by including statistical measures, making it more robust to noise and artifacts.

**Benefits:**

- **Improved Accuracy**: By providing additional features, these methods improve classification performance.
- **Better Generalization**: The variety of features helps the model generalize better to new data.
- **Enhanced Interpretability**: These techniques offer insights into how the model distinguishes between tumor types.

## Histogram of Oriented Gradients (HOG)

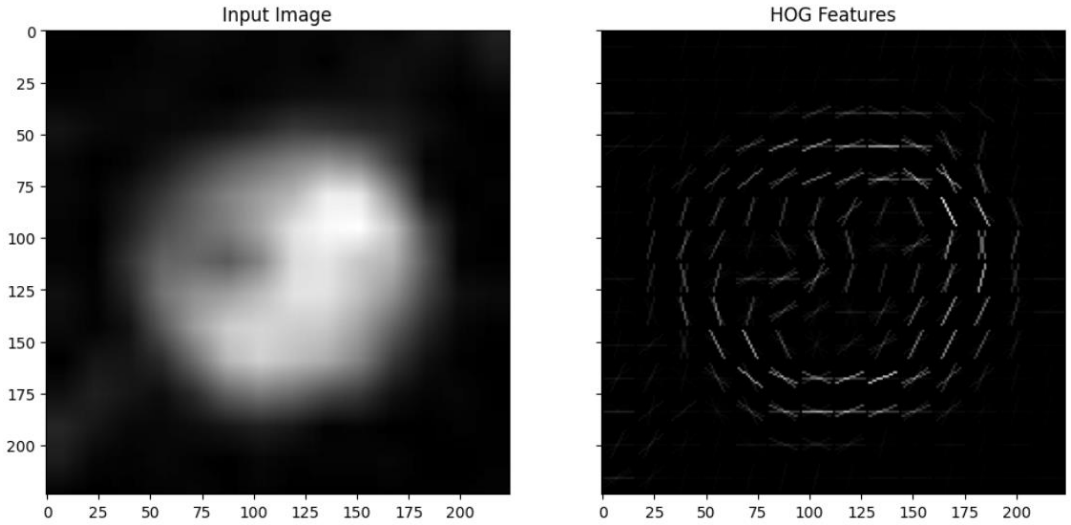
Histogram of Oriented Gradients (HOG) is a feature extraction technique that focuses on detecting edges and shapes by analyzing the gradients (changes in pixel intensity) in an image. In the TumorTrace project, HOG is used to capture important structural features of tumors, such as their boundaries and overall shape, which are crucial for distinguishing between benign and malignant tumors.

### How HOG Works

HOG calculates the gradients of image pixels, then divides the image into small cells. Each cell computes a histogram of gradient directions. These histograms are normalized in blocks, and the final feature vector is formed by concatenating them. This vector captures the edge and structural information, which is useful for classification.

### Role in TumorTrace

HOG enhances tumor classification by focusing on structural features like edges and shapes, which are crucial for identifying tumors. It helps the model detect tumor boundaries and distinguish malignant tumors from benign ones.



# Sobel Operator

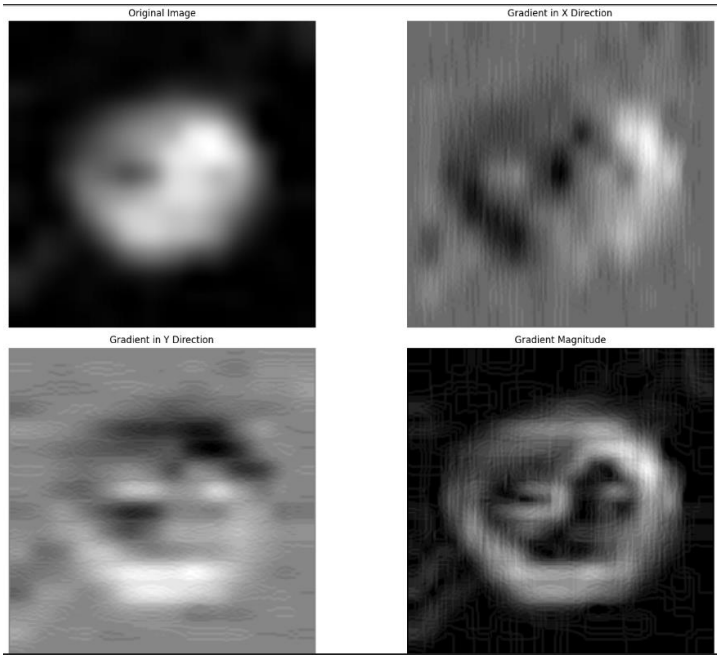
The Sobel Operator is a widely used edge detection technique that helps identify the boundaries of objects within an image by highlighting regions with sharp intensity changes. In the TumorTrace project, the Sobel operator is used to detect the edges of tumors in MRI images, aiding in tumor boundary identification, which is crucial for accurate classification.

**How it Works**

The Sobel operator calculates gradients in horizontal and vertical directions to detect edges. Horizontal gradients measure intensity change left to right, while vertical gradients measure top to bottom. Combined, these highlight sharp contrasts, often indicating tumor boundaries.

**Role in TumorTrace**

In TumorTrace, the Sobel operator is used to extract edge information from MRI images. Tumors often have distinct edges, and by detecting these boundaries, the model can focus on the tumor's shape and location. This helps in distinguishing between benign and malignant tumors based on their structural features.



# Local Binary Pattern (LBP)

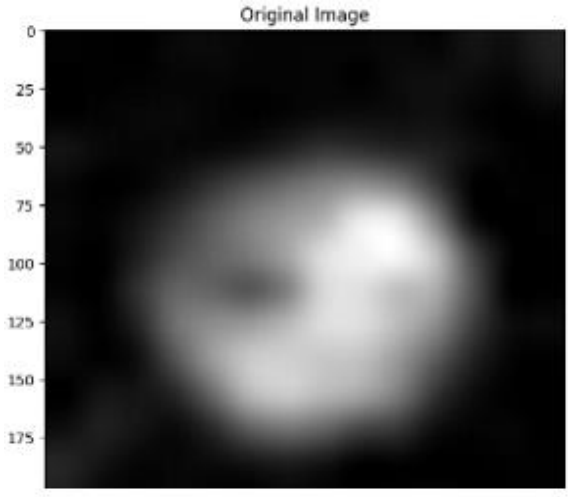
The Local Binary Pattern (LBP) is a texture extraction method that captures the local patterns of pixel intensities in an image. In the TumorTrace project, LBP is used to analyze the fine texture of tumors in MRI images, providing essential information for distinguishing between benign and malignant tumors based on their texture.

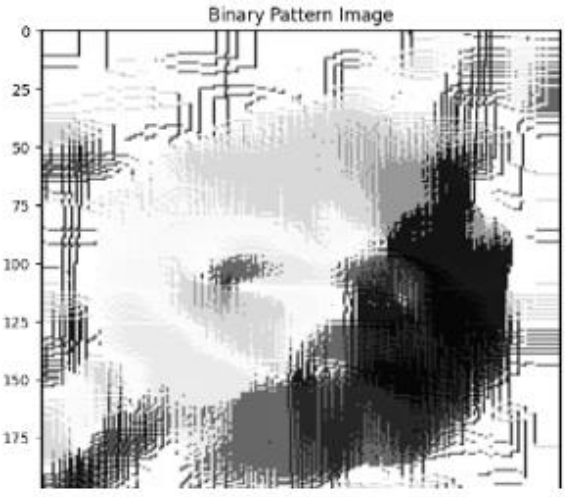
**How it Works**

LBP compares each pixel with its neighbors, assigning a binary value (1 or 0) based on whether the neighboring pixel’s intensity is greater than the central pixel. These binary values are then converted into a histogram, representing the image's local texture.

**Role in TumorTrace**

In TumorTrace, LBP is used to extract texture features from MRI images. These texture features help the model differentiate between benign and malignant tumors, as malignant tumors often have distinct and irregular textures compared to benign ones. By capturing these texture patterns, LBP enhances the model’s ability to make accurate predictions based on subtle differences in the tumor's surface characteristics.



# Mean−Variance−Median based Local Binary Patterns (MVM-LBP)

MVM-LBP is an enhanced version of LBP that incorporates mean, variance, and median values from a pixel's neighborhood to capture more robust texture information. It improves on traditional LBP by offering a richer and more stable texture description.

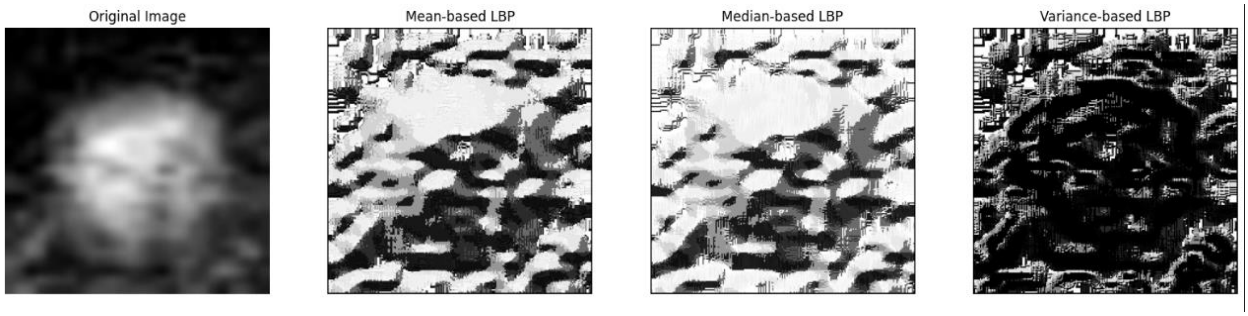
**How it Works**

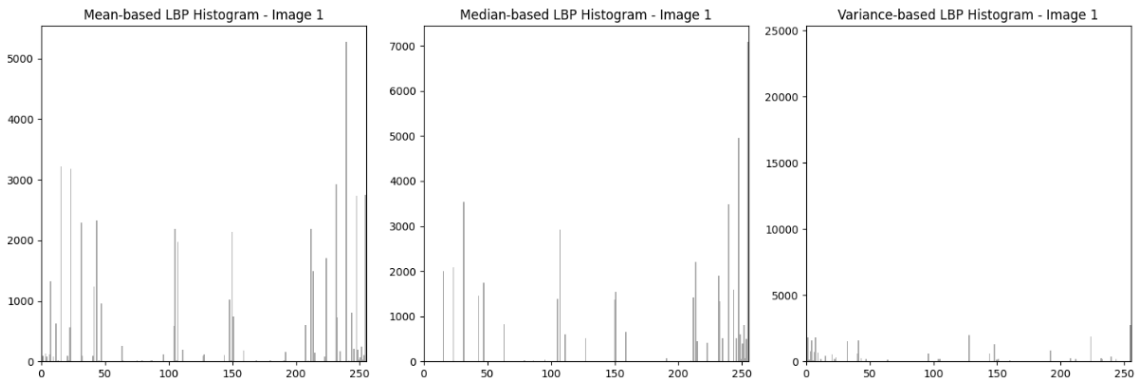
- **Mean**: Represents the average pixel intensity in the neighborhood.
- **Variance**: Measures the spread of pixel intensities.
- **Median**: Provides the central tendency, reducing sensitivity to outliers.

These features are combined to form a detailed texture descriptor, making it more effective at capturing complex and varied tumor textures, especially in malignant tumors.

**Role in TumorTrace**

In TumorTrace, MVM-LBP helps the model identify subtle texture differences between benign and malignant tumors, improving classification accuracy by capturing more detailed .



## Mean Value-based Mean Binary Patterns (MVMBP)

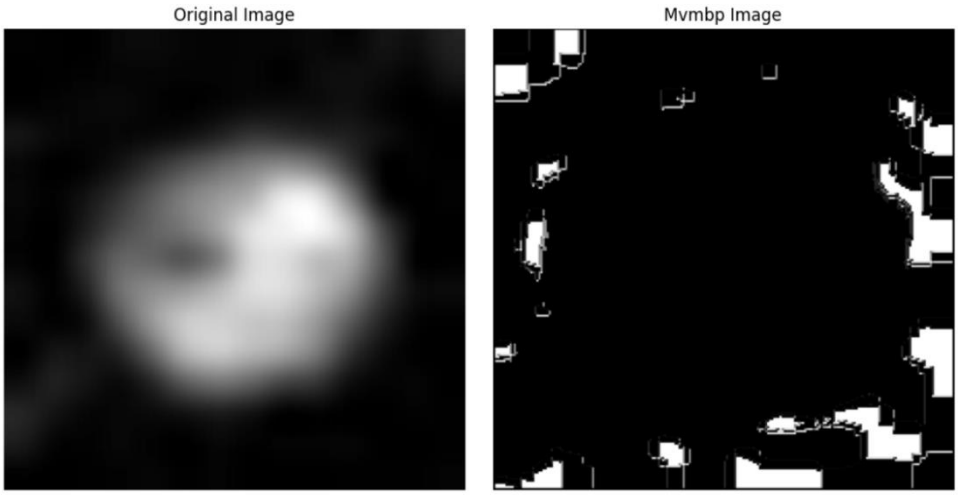
Mean Value-based Mean Binary Patterns (MVMBP) is an enhanced texture feature extraction method.It works by comparing the intensity of each pixel to the mean intensity of its neighboring pixels. If the central pixel's intensity is greater than the mean, it is assigned a binary value of 1; otherwise, 0. This generates a binary pattern that captures the local texture of the image.

**How it Works**

MVMBP calculates the mean intensity of the surrounding pixels in a local neighborhood and compares each pixel's intensity with this mean. This approach reduces sensitivity to noise and intensity variations, creating a more stable texture descriptor. The resulting binary patterns are then used to describe the texture of the image.

**Role in TumorTrace**

In TumorTrace, MVMBP is used to extract robust texture features from MRI images. By focusing on the mean intensity values, MVMBP helps the model capture more consistent and reliable texture patterns, which are important for distinguishing between benign and malignant tumors. Malignant tumors often have irregular textures, and MVMBP enhances the model's ability to identify these differences.



## Gray-Level Co-Occurrence Matrix (GLCM)

GLCM is a texture extraction method that analyzes the spatial relationship between pixel intensities in an image. In TumorTrace, it helps distinguish between benign and malignant tumors by capturing key texture features like contrast, energy, homogeneity, and correlation.

### How it Works

GLCM works by quantifying how often pairs of pixel values occur in a specific spatial relationship (direction and distance). Key texture features are derived from the matrix, including:

- **Contrast**: Measures the intensity difference between pixel pairs.
- **Energy**: Reflects uniformity in texture.
- **Homogeneity**: Measures similarity between pixel pairs.
- **Correlation**: Describes how related pixel pairs are in the image.

### Role in TumorTrace

In TumorTrace, GLCM helps the model capture textural differences in tumors. Malignant tumors often have more complex textures than benign ones, and GLCM aids in identifying these differences, improving tumor classification accuracy.

```
Horizontal GLCM for ./clasification-roi/test/Benign/BreaDM-Be-1813/SUB2/p-017.jpg:
[[3802  244    0 ...    0    0    0]
 [ 281 4047  287 ...    0    0    0]
 [   4  347 2846 ...    0    0    0]
 ...
 [   0    0    0 ...   14    7    0]
 [   0    0    0 ...    8   21    3]
 [   0    0    0 ...    0    3    3]]


Horizontal GLCM for ./clasification-roi/test/Benign/BreaDM-Be-1906/SUB1/p-011.jpg:
[[913 119   3 ...   0   0   0]
 [ 98 669 159 ...   0   0   0]
 [  3 124 659 ...   0   0   0]
 ...
 [  0   0   0 ...   1   4   0]
 [  0   0   0 ...   3   5   2]
 [  0   0   0 ...   1   1   1]]


Horizontal GLCM for ./clasification-roi/train/Benign/BreaDM-Be-1818/VIBRANT/p-027.jpg:
[[ 7  4  0 ...  0  0  0]
 [ 3  8 13 ...  0  0  0]
 [ 1  5  8 ...  0  0  0]
 ...
 [ 0  0  0 ...  8  1  3]
 [ 0  0  0 ...  6 15  0]
 [ 0  0  0 ...  0  3  3]]
```

# Model Architecture

## VGG16

VGG16 is a deep CNN with 16 layers, known for its simplicity and effective performance in image classification. It uses small 3x3 convolution filters and 2x2 max-pooling layers to capture complex features. VGG16 provides good baseline performance in tumor classification tasks.

**Role in TumorTrace:**

VGG16 is used for its straightforward architecture and ability to classify benign and malignant tumors based on MRI images.

```python
import torch
import torch.nn as nn
import torchvision.models as models

class Custom(nn.Module):
    def __init__(self, num_classes=2):
        super(Custom ,self).__init__()
        vgg16 = models.vgg16(pretrained=True)
        self.features = vgg16.features
        self.avgpool = vgg16.avgpool
        self.classifier = nn.Sequential(
            nn.Linear(512 * 7 * 7, 4096),
            nn.ReLU(),
            nn.Dropout(),
            nn.Linear(4096, 4096),
            nn.ReLU(),
            nn.Dropout(),
            nn.Linear(4096, num_classes)
        )
    def forward(self, x):
        x = self.features(x)
        x = self.avgpool(x)
        x = torch.flatten(x, 1)
        x = self.classifier(x)
        return x

model = Custom(num_classes=2)
print(model)
```

**ResNet18**

ResNet18 is a lighter version of the ResNet architecture with 18 layers, designed with residual connections that allow for deeper networks without the vanishing gradient problem. It is efficient and suitable for smaller datasets or when computational resources are limited.

**Role in TumorTrace:**

ResNet18 offers a balance between performance and efficiency, making it ideal for tumor classification on more constrained systems.

```python
import os
import torch
import torch.nn as nn

# custom Resnet18 class
class Resnet18(nn.Module):
    def __init__(self, num_classes=2):
        super(Resnet18, self).__init__()
        model_resnet18 = torch.hub.load('pytorch/vision:v0.10.0', 'resnet18', pretrained=True)

        self.conv1 = model_resnet18.conv1
        self.bn1 = model_resnet18.bn1
        self.relu = model_resnet18.relu
        self.maxpool = model_resnet18.maxpool
        self.layer1 = model_resnet18.layer1
        self.layer2 = model_resnet18.layer2
        self.layer3 = model_resnet18.layer3
        self.layer4 = model_resnet18.layer4
        self.avgpool = model_resnet18.avgpool
        self.__features = model_resnet18.fc.in_features

        self.fc = nn.Linear(self.__features, num_classes)

    def forward(self, x):
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu(x)
        x = self.maxpool(x)
        x = self.layer1(x)
        x = self.layer2(x)
        x = self.layer3(x)
        x = self.layer4(x)
        x = self.avgpool(x)
        x = x.view(x.size(0), -1)
        x = self.fc(x)
        return x
```

**ResNet50**

ResNet50 is a deeper network with 50 layers, designed to capture more complex features using residual connections. Its increased depth allows it to learn intricate patterns in large and complex datasets.

**Role in TumorTrace:**

ResNet50 is used for its ability to capture more detailed features, enhancing the accuracy of tumor classification, particularly for complex MRI images.

```python
import torch
import torch.nn as nn

# Custom ResNet50 class
class Resnet50(nn.Module):
    def __init__(self, num_classes=2):
        super(Resnet50, self).__init__()
        model_resnet50 = torch.hub.load('pytorch/vision:v0.10.0', 'resnet50', pretrained=True)

        self.conv1 = model_resnet50.conv1
        self.bn1 = model_resnet50.bn1
        self.relu = model_resnet50.relu
        self.maxpool = model_resnet50.maxpool
        self.layer1 = model_resnet50.layer1
        self.layer2 = model_resnet50.layer2
        self.layer3 = model_resnet50.layer3
        self.layer4 = model_resnet50.layer4
        self.avgpool = model_resnet50.avgpool
        self.__features = model_resnet50.fc.in_features

        self.fc = nn.Linear(self.__features, num_classes)

    def forward(self, x):
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu(x)
        x = self.maxpool(x)
        x = self.layer1(x)
        x = self.layer2(x)
        x = self.layer3(x)
        x = self.layer4(x)
        x = self.avgpool(x)
        x = x.view(x.size(0), -1)
        x = self.fc(x)
        return x
```

# Early Stopping

**Early stopping** is a technique used to prevent overfitting by halting training when the model's performance on the validation set stops improving. It monitors the validation loss and stops training if the loss does not improve for a set number of epochs (patience).

**How it Works**

- The model's performance is tracked based on validation loss.
- If the validation loss doesn't improve after a number of epochs , training is stopped.
- The model with the best validation loss is saved during training.

**Role in TumorTrace**

In TumorTrace, early stopping helps ensure that the model does not continue training unnecessarily once it has achieved optimal performance on the validation set. This reduces overfitting and conserves computational resources.

```python
import numpy as np
import torch

class EarlyStopping:
    def __init__(self, patience=7, verbose=False, delta=0, path='checkpoint.pt', trace_func=print):
        self.patience = patience
        self.verbose = verbose
        self.delta = delta
        self.path = path
        self.trace_func = trace_func
        self.counter = 0
        self.best_score = None
        self.early_stop = False
        self.val_loss_min = np.inf

    def __call__(self, val_loss, model):
        score = -val_loss

        if self.best_score is None:
            self.best_score = score
            self.save_checkpoint(val_loss, model)
        elif score < self.best_score + self.delta:
            self.counter += 1
            self.trace_func(f'Early stopping counter: {self.counter} out of {self.patience}')
            if self.counter >= self.patience:
                self.early_stop = True
        else:
            self.best_score = score
            self.save_checkpoint(val_loss, model)
            self.counter = 0

    def save_checkpoint(self, val_loss, model):
        '''Saves model when validation loss decreases.'''
        if self.verbose:
            self.trace_func(f'Validation loss decreased ({self.val_loss_min:.6f} --> {val_loss:.6f}). Saving model.')
        torch.save(model.state_dict(), self.path)
        self.val_loss_min = val_loss
```

# Training Function

The Training Function in TumorTrace is responsible for training the model on the dataset. It optimizes the model's parameters using the Stochastic Gradient Descent (SGD) optimizer and updates the weights based on the computed loss during each epoch. The function adjusts the learning rate dynamically and tracks the model's performance throughout the training process.

**How it Works**

- Learning Rate: Adjusts the learning rate every 10 epochs for efficient convergence.
- Optimizer: Uses SGD with momentum and weight decay to optimize the model.
- Training Loop: For each batch, the model computes the loss, updates weights, and tracks the number of correct predictions to calculate accuracy.

**Role in TumorTrace**

In **TumorTrace**, the training function is crucial for teaching the model to classify breast tumors accurately. By optimizing the model with backpropagation and adjusting weights using the optimizer, the model improves its ability to distinguish between benign and malignant tumors.

```python
import torch
from tqdm import tqdm

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Training Function

def train(epoch, model, num_epochs, loader, criterion, l2_decay, lr):
    learning_rate = max(lr * (0.1 ** (epoch // 10)), 1e-5)

    optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate, momentum=0.9, weight_decay=l2_decay)
    model.train()
    correct = 0

    # Training loop
    for data, label in tqdm(loader, desc=f'Epoch {epoch + 1}/{num_epochs}', unit='batch'):
        data, label = data.to(device), label.to(device)

        optimizer.zero_grad()
        output = model(data)
        loss = criterion(output, label)
        loss.backward()
        optimizer.step()

        pred = output.argmax(dim=1, keepdim=True)
        correct += pred.eq(label.view_as(pred)).sum().item()

    accuracy = 100. * correct / len(loader.dataset)
    print(f'Train Accuracy: {accuracy:.2f}%')
```

# Validation Function

The validation function evaluates the model's performance on the validation dataset after each training epoch. It calculates key metrics such as accuracy, loss, AUC, specificity, and sensitivity to assess the model's ability to generalize.

**How it Works**

- Model Evaluation: The model is set to evaluation mode, and no gradients are computed.
- Predictions and Loss: For each batch, the model generates predictions, and the negative log-likelihood loss is computed.
- Metrics: It calculates accuracy, confusion matrix, ROC, AUC, specificity, and sensitivity.

**Role in TumorTrace**

In TumorTrace, this function helps track the model's performance during training, ensuring it is not overfitting and generalizes well to unseen data.

```python
def validation(model, val_dataloader):
    model.eval()
    correct = 0
    all_predictions = []
    all_targets = []
    possibilities = None

    with torch.no_grad():
        for data, target in val_dataloader:
            data, target = data.to(device), target.to(device)
            val_output = model(data)
            test_loss = F.nll_loss(F.log_softmax(val_output, dim=1), target, reduction='sum').item()
            pred = val_output.argmax(dim=1, keepdim=True)
            all_predictions.extend(pred.cpu().numpy())
            all_targets.extend(target.cpu().numpy())
            prob = F.softmax(val_output, dim=1).cpu().data.numpy()
            if possibilities is None:
                possibilities = prob
            else:
                possibilities = np.concatenate((possibilities, prob), axis=0)
            correct += pred.eq(target.view_as(pred)).cpu().sum().item()

    all_predictions = np.array(all_predictions).flatten()
    all_targets = np.array(all_targets).flatten()
    cm = confusion_matrix(all_targets, all_predictions)
    print(f'Confusion Matrix:\n{cm}')
    num_classes = val_output.shape[1]
    label_onehot = np.eye(num_classes)[all_targets.astype(int)]

    fpr, tpr, _ = roc_curve(label_onehot.ravel(), possibilities.ravel())

    auc_value = auc(fpr, tpr)
    specificity = 1 - fpr[1]
    sensitivity = tpr[1]
    print(f'Specificity: {specificity:.4f}, Sensitivity: {sensitivity:.4f}, AUC: {auc_value:.4f}')

    test_loss /= len(val_dataloader.dataset)
    accuracy = 100. * correct / len(val_dataloader.dataset)
    print(f'Validation Loss: {test_loss:.4f}, Accuracy: {accuracy:.2f}%')

    return accuracy, test_loss, auc_value
```

# Model Evaluation and Performance Metrics

In TumorTrace, after training, the model's performance is evaluated using various metrics to assess its accuracy and generalization on test data.

## Test Function

The test function computes key metrics:

1. Classification Report: Precision, recall, and F1-score for both classes.
2. Confusion Matrix: Visualizes correct vs. incorrect predictions.
3. ROC Curve and AUC: Evaluates classification thresholds.
4. Sensitivity and Specificity: Measures how well the model identifies malignant and benign

```python
def test(model, test_dataloader, criterion, labels=['benign', 'malignant']):
    model.eval()
    test_loss = 0
    correct = 0
    possibilities = None
    all_predictions = []
    all_labels = []

    for data, target in test_dataloader:
        if torch.cuda.is_available():
            data, target = data.cuda(), target.cuda()

        test_output = model(data)
        test_loss += F.cross_entropy(test_output, target, reduction='sum').item()

        pred = test_output.data.max(1)[1]
        all_predictions.extend(pred.cpu().numpy())
        all_labels.extend(target.cpu().numpy())

        possibility = F.softmax(test_output, dim=1).cpu().data.numpy()
        possibilities = np.concatenate((possibilities, possibility), axis=0) if possibilities is not None else possibility

        correct += pred.eq(target.data.view_as(pred)).cpu().sum().item()

    # Flatten predictions and labels
    all_predictions = [i for i in all_predictions]
    all_labels = [i for i in all_labels]

    # Classification metrics
    print(metrics.classification_report(all_labels, all_predictions, target_names=labels, digits=4))

    # Plot Confusion Matrix
    plot_confusion_matrix(all_labels, all_predictions, labels)

    # Plot ROC Curve and AUC
    auc_value = plot_roc_curve(all_labels, possibilities)

    # Calculate specificity and sensitivity
    fpr, tpr, _ = roc_curve(all_labels, possibilities[:, 1])
    specificity = 1 - fpr[1] if len(fpr) > 1 else 0.0
    sensitivity = tpr[1] if len(tpr) > 1 else 0.0

    # Average loss
    test_loss /= len(test_dataloader.dataset)

    # Print metrics
    print(f'Specificity: {specificity:.4f}, Sensitivity: {sensitivity:.4f}, AUC: {auc_value:.4f}')
    print(f'\nTest set: Average loss: {test_loss:.4f}, Accuracy: {correct}/{len(test_dataloader.dataset)} ({100. * correct / len(test_dataloader.dataset):.2f}%)\n')

    return 100. * correct / len(test_dataloader.dataset), test_loss, auc_value
```

# Model Evaluation Results

## Model 1: VGG16

| Training result | |
|---|---|
| Specificity | 1.0000 |
| Sensitivity | 0.0005 |
| AUC | 0.8490 |
| Validation Loss | 0.0010 |
| Accuracy | 74.81% |

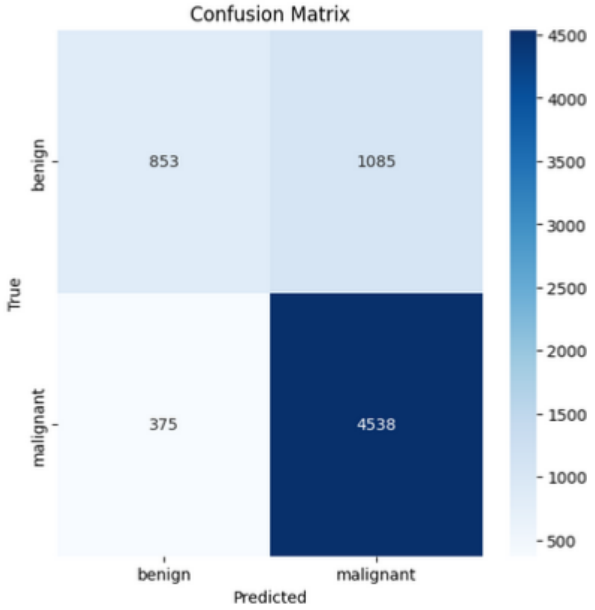
| Test result | |
|---|---|
| Specificity | 1.0000 |
| Sensitivity | 0.0002 |
| AUC | 0.8170 |
| Validation Loss | 0.4677 |
| Accuracy | 78.69%, |

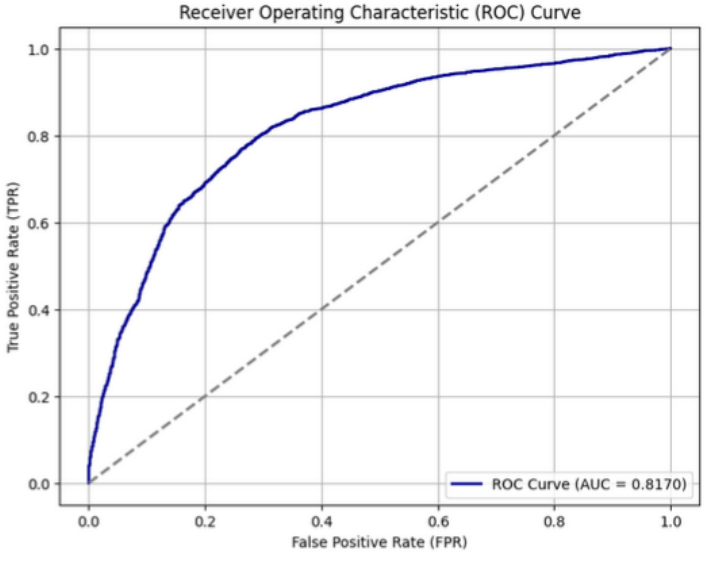
```
              precision    recall  f1-score   support

      benign     0.6946    0.4401    0.5389      1938
   malignant     0.8070    0.9237    0.8614      4913

    accuracy                         0.7869      6851
   macro avg     0.7508    0.6819    0.7001      6851
weighted avg     0.7752    0.7869    0.7702      6851


Confusion Matrix:
 [[ 853 1085]
 [ 375 4538]]
```



**Model 1: ResNet18**

| Training result | |
|---|---|
| Specificity | 1.0000 |
| Sensitivity | 0.0005 |
| AUC | 0.7386 |
| Validation Loss | 0.0008 |
| Accuracy | 65.21% |

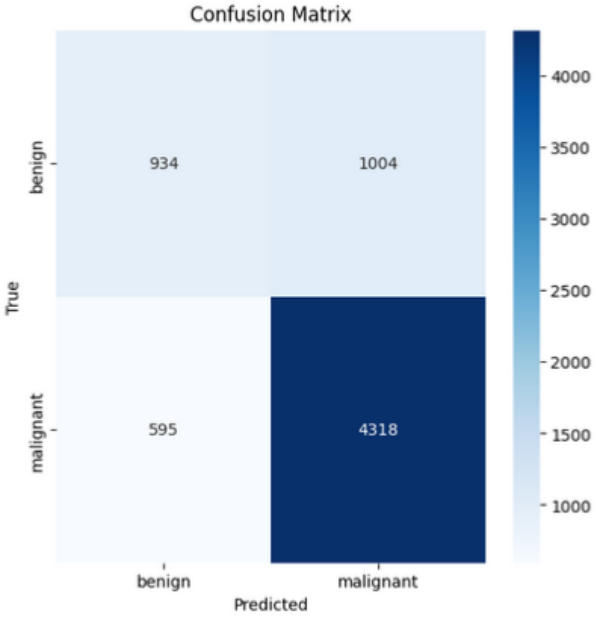
| Test result | |
|---|---|
| Specificity | 1.0000 |
| Sensitivity | 0.0002 |
| AUC | 0.7805 |
| Validation Loss | 0.5207 |
| Accuracy | 76.66% |

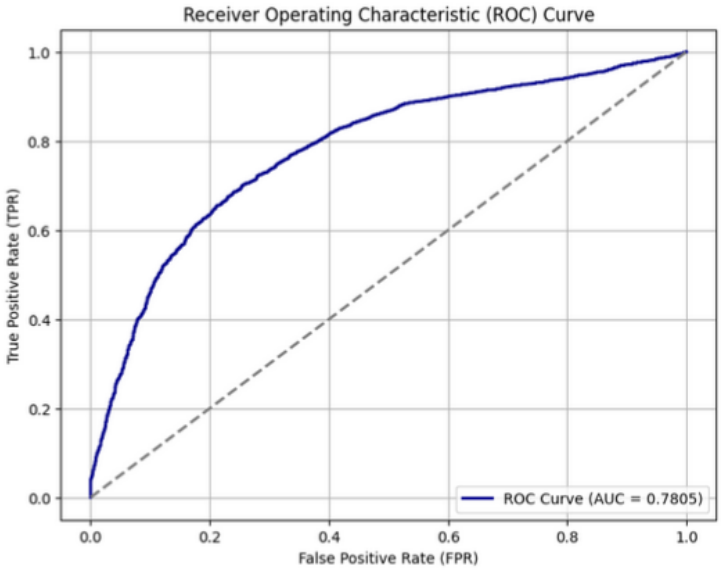
```
              precision    recall  f1-score   support

      benign     0.6109    0.4819    0.5388      1938
   malignant     0.8113    0.8789    0.8438      4913

    accuracy                         0.7666      6851
   macro avg     0.7111    0.6804    0.6913      6851
weighted avg     0.7546    0.7666    0.7575      6851


Confusion Matrix:
 [[ 934 1004]
 [ 595 4318]]
```



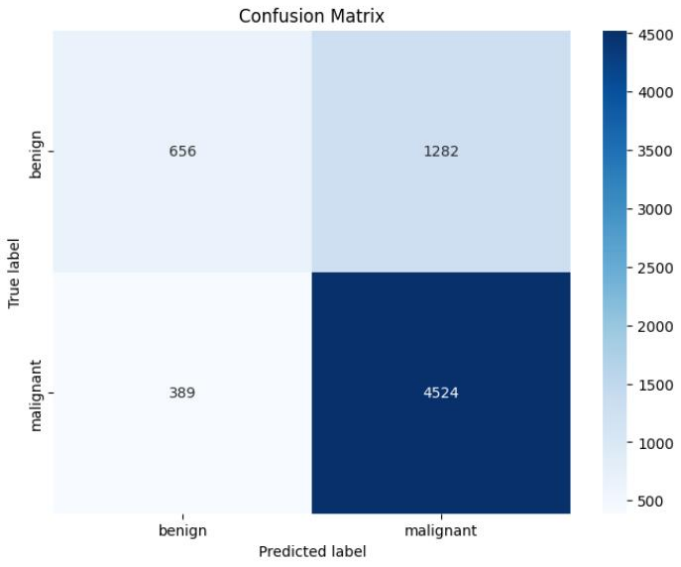



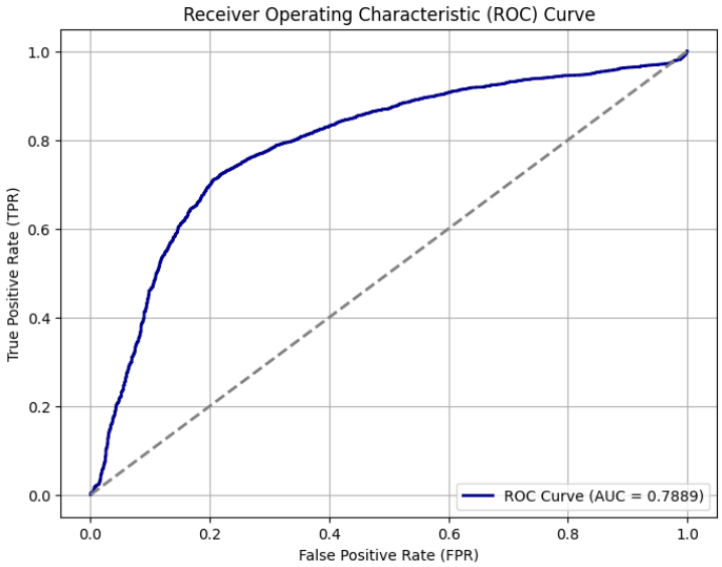
**Model 1: ResNet50**

| Training result | |
|---|---|
| Specificity | 1.0000 |
| Sensitivity | 0.0006 |
| AUC | 0. 7889 |
| Validation Loss | 0.3064 |
| Accuracy | 86.12% |

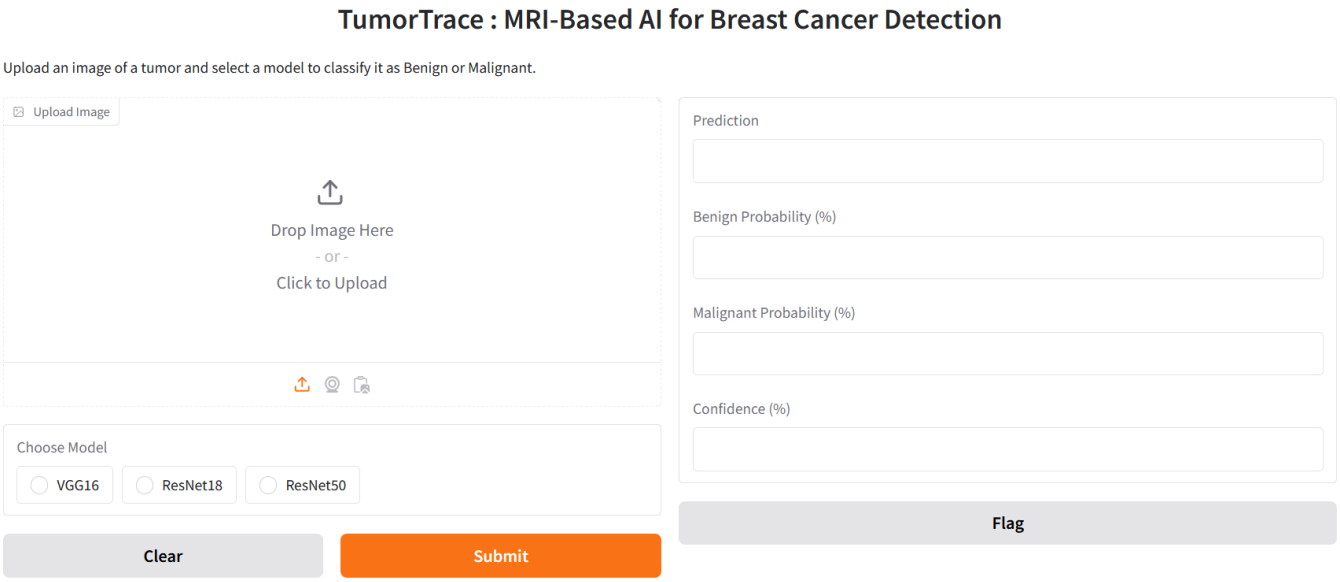
| Test result | |
|---|---|
| Specificity | 1.0000 |
| Sensitivity | 0.0006 |
| AUC | 0.7889 |
| Validation Loss | 0.6649 |
| Accuracy | 75.61% |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| benign | 0.6278 | 0.3385 | 0.4398 | 1938 |
| malignant | 0.7792 | 0.9208 | 0.8441 | 4913 |
| accuracy | | | 0.7561 | 6851 |
| macro avg | 0.7035 | 0.6297 | 0.6420 | 6851 |
| weighted avg | 0.7364 | 0.7561 | 0.7297 | 6851 |



# Demo Model on gradio



# References


1. https://www.sciencedirect.com/science/article/abs/pii/S0010482523007205
2. https://www.image-net.org/
3. https://poloclub.github.io/cnn-explainer/
4. https://trinhngocthuyen.com/posts/tech/a-dive-into-hog/
5. https://diegoinacio.github.io/computer-vision-notebooks-page/pages/sobel_operator.html
6. https://fairyonice.github.io/implement-lbp-from%20scratch.html
7. https://assets-eu.researchsquare.com/files/rs-2022969/v1_covered.pdf?c=1662821973