Project TumorTrace

MRI-Based AI for Breast Cancer Detection

Infosys SPRINGBOARD 5.0

Name: Siva Nandini Bommupalla

Company Name : Infosys Role : AI/ML Intern

Internship period : 7 th October 2024 – December 2024

Linkidin:: www.linkedin.com/in/nandubommupalla

Data Agumentation

```
import os
import torch
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt

# Define the dataset path
data_directory = '/content/drive/MyDrive/clasification-roi'  # Updated to your dataset path

# Define transformations for training dataset
train_data_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomCrop(210),  # Randomly crop images to 210x210
    transforms.RandomHorizontalFlip(),  # Horizontal flipping
    transforms.RandomRotation(15),  # Random rotation
    transforms.ToTensor(),  # Convert images to tensors
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])  # Normalization
])

# Define transformations for validation and test datasets
test_data_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomRotation(15),  # Random rotation
    transforms.ToTensor(),  # Convert images to tensors
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])  # Normalization
])

valid_data_transform = transforms.Compose([
    transforms.Resize((224, 224)),
```

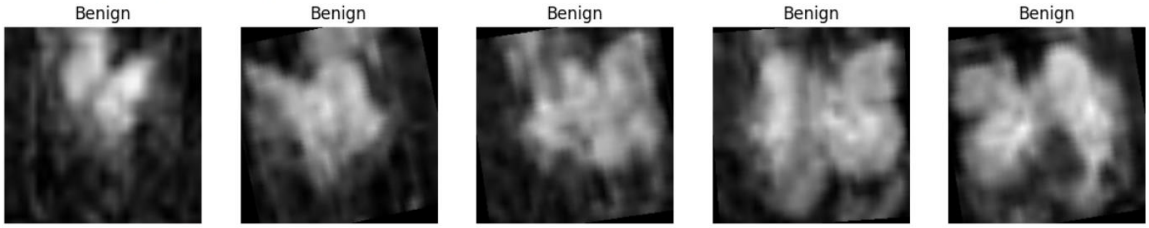
```
    transforms.RandomRotation(15),  # Random rotation
    transforms.ToTensor(),  # Convert images to tensors
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])  # Normalization
])

# Load datasets
train_data = datasets.ImageFolder(root=os.path.join(data_directory, 'train'),
transform=train_data_transform)
test_data = datasets.ImageFolder(root=os.path.join(data_directory, 'test'),
transform=test_data_transform)
valid_data = datasets.ImageFolder(root=os.path.join(data_directory, 'val'),
transform=valid_data_transform)  # Assuming you have a 'val' folder

# Create data loaders
train_loader = DataLoader(train_data, batch_size=32, shuffle=True)
test_loader = DataLoader(test_data, batch_size=32, shuffle=False)
valid_loader = DataLoader(valid_data, batch_size=32, shuffle=False)

# Count classes in datasets
benign_count_train = train_data.targets.count(0)
malignant_count_train = train_data.targets.count(1)
print(f'Train Set: Benign = {benign_count_train}, Malignant = {malignant_count_train}')

benign_count_test = test_data.targets.count(0)
malignant_count_test = test_data.targets.count(1)
print(f'Test Set: Benign = {benign_count_test}, Malignant = {malignant_count_test}')

benign_count_valid = valid_data.targets.count(0)
malignant_count_valid = valid_data.targets.count(1)
print(f'Validation Set: Benign = {benign_count_valid}, Malignant = {malignant_count_valid}')

# Function to unnormalize images for display
```

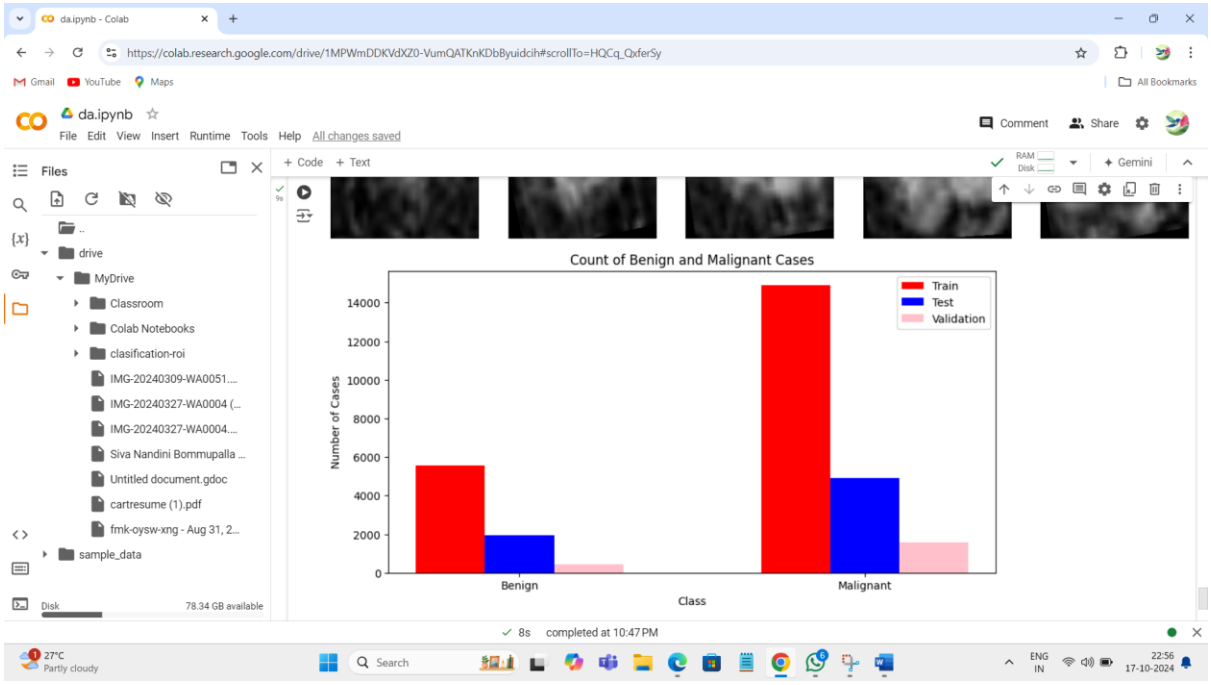
```
def unnormalize_image(img, mean, std):
    img = img.clone()  # Clone to avoid modifying the original tensor
    for t, m, s in zip(img, mean, std):
        t.mul_(s).add_(m)  # Unnormalize each channel
    return img

# Function to display sample images
def show_sample_images(data, num_images=5):
    mean = [0.485, 0.456, 0.406]
    std = [0.229, 0.224, 0.225]
    plt.figure(figsize=(15, 5))
    for i in range(num_images):
        img, lbl = data[i]
        img = unnormalize_image(img, mean, std)  # Unnormalize
        plt.subplot(1, num_images, i + 1)
        plt.imshow(img.permute(1, 2, 0).numpy())
        plt.title(data.classes[lbl])
        plt.axis('off')
    plt.show()

# Display sample images from training set
show_sample_images(train_data)

# Plotting the class counts for each dataset with custom colors
class_labels = ['Benign', 'Malignant']
train_class_counts = [benign_count_train, malignant_count_train]
test_class_counts = [benign_count_test, malignant_count_test]
valid_class_counts = [benign_count_valid, malignant_count_valid]

x_pos = range(len(class_labels))
bar_width = 0.2
```

```
plt.figure(figsize=(10, 5))


# Custom colors for bars

train_color = 'red'

test_color = 'blue'

valid_color = 'pink'


# Plot each dataset with a unique color

plt.bar(x_pos, train_class_counts, width=bar_width, color=train_color, label='Train', align='center')

plt.bar([p + bar_width for p in x_pos], test_class_counts, width=bar_width, color=test_color, label='Test', align='center')

plt.bar([p + 2 * bar_width for p in x_pos], valid_class_counts, width=bar_width, color=valid_color, label='Validation', align='center')


plt.xlabel('Class')

plt.ylabel('Number of Cases')

plt.title('Count of Benign and Malignant Cases')

plt.xticks([p + bar_width for p in x_pos], class_labels)

plt.legend()

plt.show()
```

```
plt.bar([p + bar_width for p in x_pos], test_class_counts, width=bar_width, color=test_color, label='Test', ali
plt.bar([p + 2 * bar_width for p in x_pos], valid_class_counts, width=bar_width, color=valid_color, label='Vali

plt.xlabel('Class')
plt.ylabel('Number of Cases')
plt.title('Count of Benign and Malignant Cases')
plt.xticks([p + bar_width for p in x_pos], class_labels)
plt.legend()

# Display the graph
plt.show()
```

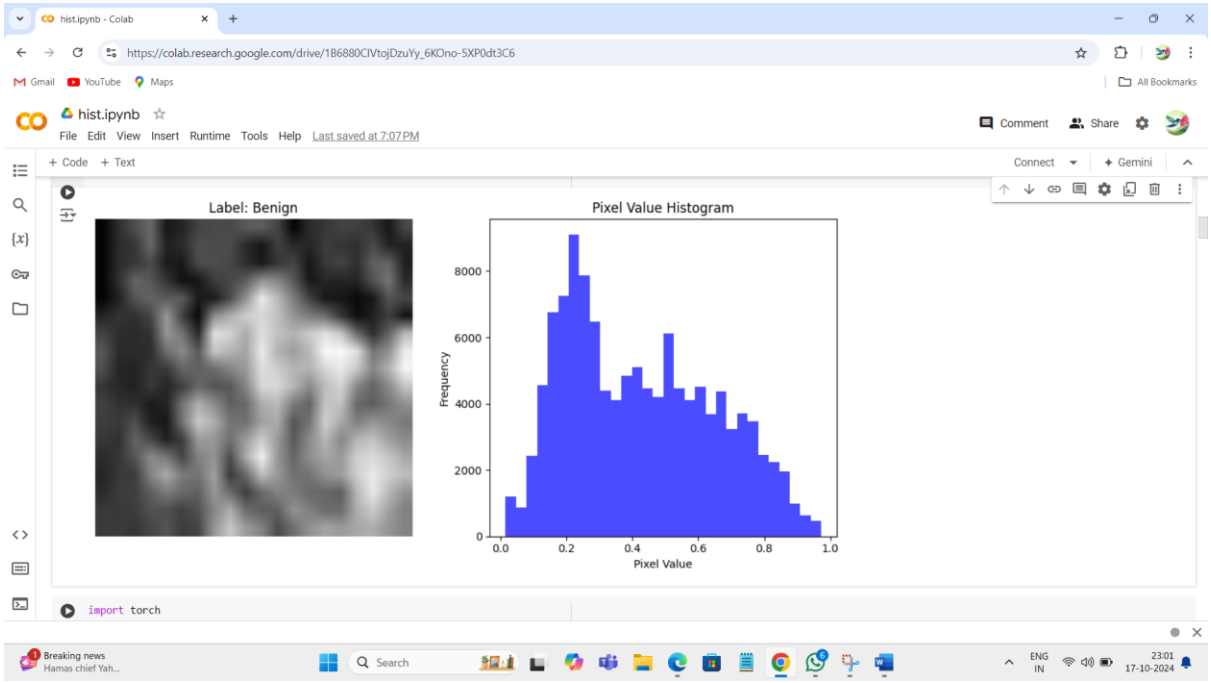
```
Train Set: Benign = 5559, Malignant = 14905
Test Set: Benign = 1938, Malignant = 4923
Validation Set: Benign = 418, Malignant = 1581
```





```
import torch
import torchvision.transforms as transforms
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt
import numpy as np

# Define the image transformations for training (augmentation + normalization)
train_transform = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.RandomVerticalFlip(),
    transforms.RandomRotation(20),  # Rotation in degrees
    transforms.RandomResizedCrop((200, 200)),  # Random crop to 200x200
    transforms.ColorJitter(contrast=0.1, brightness=0.1, saturation=0.1),  # Random contrast, brightness, and saturation
    transforms.RandomGrayscale(p=0.1),  # Apply grayscale with a probability of 10%
    transforms.ToTensor(),  # Convert the image to a tensor
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])  # Normalize to range [-1, 1]
])

# Load the dataset
train_dataset = ImageFolder('/content/drive/MyDrive/clasification-roi/train', transform=train_transform)
train_loader = DataLoader(train_dataset, batch_size=1, shuffle=True)  # Set batch_size=1 to get a single image

# Function to plot image and its pixel value histogram
def plot_image_and_histogram(dataloader):
    class_names = dataloader.dataset.classes
    images, labels = next(iter(dataloader))  # Get a single image and its label
    image = images[0]  # Extract the image from the batch
    label = labels[0].item()  # Get the label
```

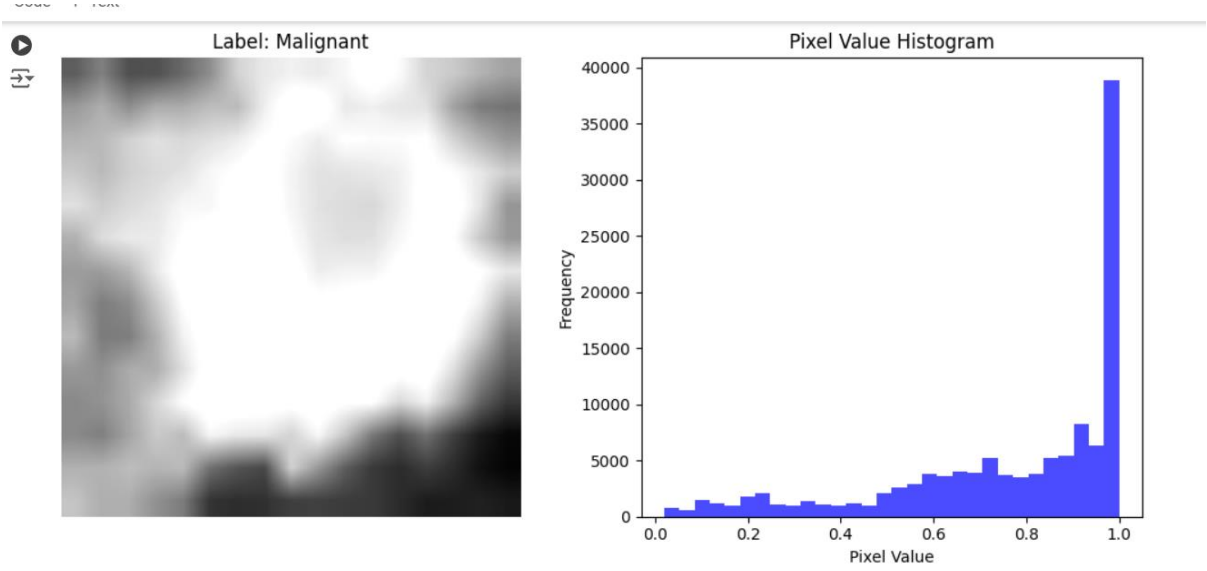
```
    image = image.permute(1, 2, 0)  # Rearrange dimensions to [height, width, channels]

    # Reverse normalization (from [-1, 1] to [0, 1])
    image = image * 0.5 + 0.5

    # Convert image to numpy array for plotting
    image_np = image.numpy()

    # Plot the image
    plt.figure(figsize=(10, 5))

    plt.subplot(1, 2, 1)
    plt.imshow(image_np)
    plt.title(f'Label: {class_names[label]}')
    plt.axis('off')

    # Plot the histogram of pixel values
    plt.subplot(1, 2, 2)
    pixel_values = image_np.flatten()

    plt.hist(pixel_values, bins=30, color='blue', alpha=0.7)
    plt.title('Pixel Value Histogram')
    plt.xlabel('Pixel Value')
    plt.ylabel('Frequency')

    plt.tight_layout()
    plt.show()

# Call the function to display the image and its histogram
plot_image_and_histogram(train_loader)
```




```
import torch
import torchvision.transforms as transforms
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt
import numpy as np


# Define the image transformations for training (augmentation + normalization)
train_transform = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.RandomVerticalFlip(),
    transforms.RandomRotation(20),  # Rotation in degrees
    transforms.RandomResizedCrop((200, 200)),  # Random crop to 200x200
    transforms.ColorJitter(contrast=0.1, brightness=0.1, saturation=0.1),  # Random contrast, brightness, and saturation
    transforms.RandomGrayscale(p=0.1),  # Apply grayscale with a probability of 10%
    transforms.ToTensor(),  # Convert the image to a tensor
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])  # Normalize to range [-1, 1]
```

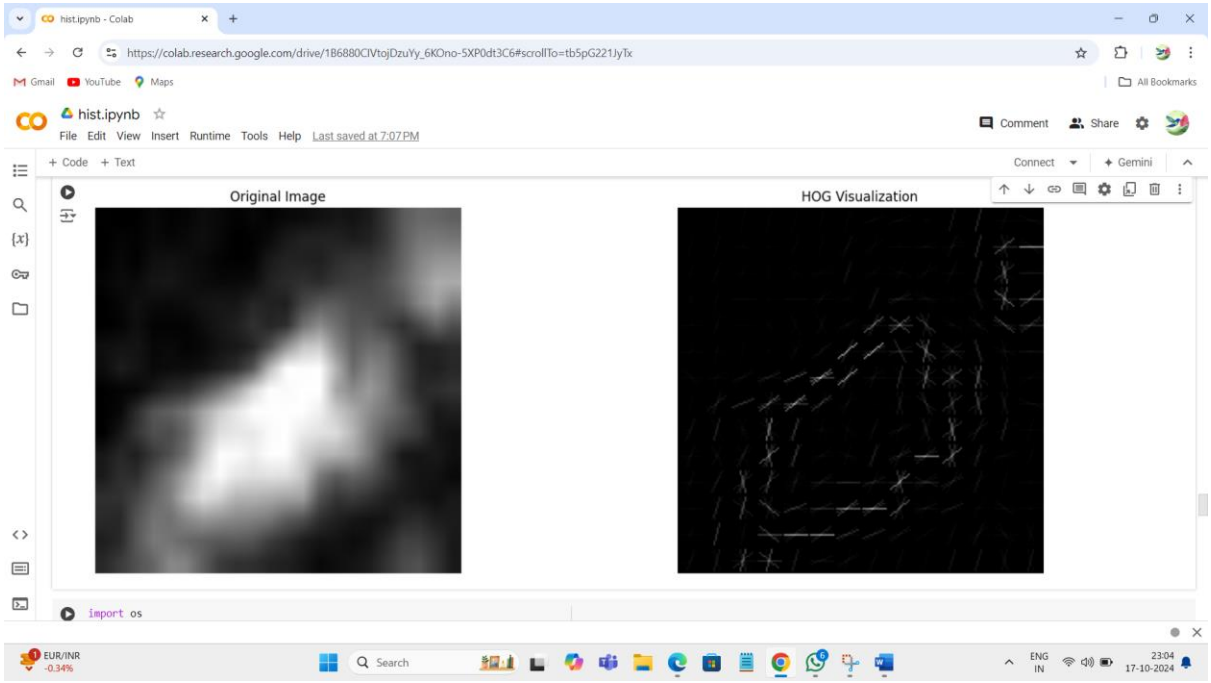
```
])


# Load the dataset
train_dataset = ImageFolder('/content/drive/MyDrive/clasification-roi/train',
transform=train_transform)
train_loader = DataLoader(train_dataset, batch_size=1, shuffle=True)  # Set batch_size=1 to get a
single image


# Function to plot image and its pixel value histogram
def plot_image_and_histogram(dataloader):
    class_names = dataloader.dataset.classes
    images, labels = next(iter(dataloader))  # Get a single image and its label
    image = images[0]  # Extract the image from the batch
    label = labels[0].item()  # Get the label
    image = image.permute(1, 2, 0)  # Rearrange dimensions to [height, width, channels]

    # Reverse normalization (from [-1, 1] to [0, 1])
    image = image * 0.5 + 0.5

    # Convert image to numpy array for plotting
    image_np = image.numpy()

    # Plot the image
    plt.figure(figsize=(10, 5))

    plt.subplot(1, 2, 1)
    plt.imshow(image_np)
    plt.title(f'Label: {class_names[label]}')
    plt.axis('off')

    # Plot the histogram of pixel values
    plt.subplot(1, 2, 2)
```

```
    pixel_values = image_np.flatten()

    plt.hist(pixel_values, bins=30, color='blue', alpha=0.7)
    plt.title('Pixel Value Histogram')
    plt.xlabel('Pixel Value')
    plt.ylabel('Frequency')

    plt.tight_layout()
    plt.show()

# Call the function to display the image and its histogram
plot_image_and_histogram(train_loader)
```



```
import torch
import torch.nn.functional as F
import torchvision.transforms as transforms
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt
import numpy as np
import cv2  # OpenCV for corner detection
from skimage.feature import hog  # For HOG feature extraction

# Load the dataset (for demonstration, replace with your image directory)
train_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])  # Normalize to range [-1, 1]
])

# Load your dataset here
train_dataset = ImageFolder('/content/drive/MyDrive/clasification-roi/train',
transform=train_transform)
train_loader = DataLoader(train_dataset, batch_size=1, shuffle=True)  # Set batch_size=1 to get a
single image

def compute_hog(image):
    image_np = image.squeeze().detach().numpy()  # Convert tensor to numpy array
    image_np = (image_np * 224).astype(np.uint8)  # Convert back to uint8

    # Convert to grayscale if it's not
    if image_np.shape[0] == 3:
        image_np = cv2.cvtColor(image_np.transpose(1, 2, 0), cv2.COLOR_RGB2GRAY)
```

```
    hog_features, hog_image = hog(image_np, visualize=True, block_norm='L2-Hys',
pixels_per_cell=(16, 16))
    return hog_features, hog_image

# Function to plot images
def plot_images(original_image,hog_image):
    plt.figure(figsize=(15, 10))

    plt.subplot(2, 2, 1)
    plt.imshow(original_image.permute(1, 2, 0) * 0.5 + 0.5)  # Reverse normalization
    plt.title('Original Image')
    plt.axis('off')

    plt.subplot(2, 2, 2)
    plt.imshow(hog_image, cmap='gray')
    plt.title('HOG Visualization')
    plt.axis('off')

    plt.tight_layout()
    plt.show()

# Get a single image from the dataset
images, labels = next(iter(train_loader))
image = images[0]

# Apply the edge detection, corner detection, and HOG computation

hog_features, hog_image = compute_hog(image)

# Plot the results
plot_images(image,hog_image)
```


hist.ipynb - Colab
https://colab.research.google.com/drive/1B6880ClVtojDzuYy_6KOno-5XP0dt3C6#scrollTo=tb5pG221JyTx
hist.ipynb
File Edit View Insert Runtime Tools Help Last saved at 7:07 PM
Comment
Share
+ Code + Text
Connect
Gemini
Original Image
HOG Visualization
import os