eogly2dze

December 9, 2024

[23]:
```python
from google.colab import drive
drive.mount('/content/drive')

!pip install nibabel scikit-image fpdf

import os
import nibabel as nib
import numpy as np
from scipy import ndimage as ndi
from skimage import filters, measure, morphology, feature
import matplotlib.pyplot as plt
import tensorflow as tf
from fpdf import FPDF
from sklearn.model_selection import train_test_split

# --- Step 1: Load and Preprocess Data ---

# This section defines functions to load and preprocess the MRI images.
# It includes loading NIfTI images, smoothing, and normalizing the data.

def load_patient_data(base_dir, patient_number):
    """Loads all NIfTI files for a given patient."""
    patient_dir = os.path.join(base_dir, f'Patient-{patient_number}')
    file_names = [
        f'{patient_number}-T1.nii',
        f'{patient_number}-T2.nii',
        f'{patient_number}-Flair.nii',
        f'{patient_number}-LesionSeg-T1.nii',
        f'{patient_number}-LesionSeg-T2.nii',
        f'{patient_number}-LesionSeg-Flair.nii',
    ]
    images = {}
    for file_name in file_names:
        file_path = os.path.join(patient_dir, file_name)
        try:
            img = nib.load(file_path)
            images[file_name.split('.')[0]] = img
```

```python
        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}")
    return images

def load_and_preprocess(image_path):
    """Loads and preprocesses a single NIfTI image."""
    brain_vol = nib.load(image_path)
    brain_vol_data = brain_vol.get_fdata()
    smoothed_data = ndi.gaussian_filter(brain_vol_data, sigma=1)
    normalized_data = (smoothed_data - np.min(smoothed_data)) / (np.
 ↪max(smoothed_data) - np.min(smoothed_data))
    return normalized_data, brain_vol

# --- Step 2: Prepare Data for CNN ---

# This section prepares the data for the CNN model.
# It includes resizing, padding, and generating labels based on lesion␣
 ↪segmentation.

def preprocess_image(image, target_shape=(128, 128, 64)):
    """Preprocesses a single image for the CNN."""
    resized_image = tf.image.resize_with_pad(image, target_shape[0],␣
 ↪target_shape[1]).numpy()
    depth = resized_image.shape[2]
    if depth < target_shape[2]:
        pad_width = [(0, 0), (0, 0), (0, target_shape[2] - depth)]
        resized_image = np.pad(resized_image, pad_width, mode='constant')
    elif depth > target_shape[2]:
        resized_image = resized_image[:, :, :target_shape[2]]
    return resized_image

def load_mri_data(base_dir, num_patients):
    """Loads and preprocesses MRI data for CNN training."""
    images = []
    labels = []
    for i in range(1, num_patients + 1):
        patient_data = load_patient_data(base_dir, i)
        if f'{i}-Flair' in patient_data:
            flair_image_path = patient_data[f'{i}-Flair'].get_filename()
            img, _ = load_and_preprocess(flair_image_path)

            print(f"Original shape for Patient-{i}: {img.shape}")

            img = preprocess_image(img)
            images.append(img)
```

```
            lesion_seg_path = patient_data[f'{i}-LesionSeg-Flair'].
↪get_filename()
            lesion_seg, _ = load_and_preprocess(lesion_seg_path)
            threshold = 1000
            labels.append(1 if np.sum(lesion_seg) > threshold else 0)

    return np.array(images), np.array(labels)

# --- Step 3: Split Data and Train CNN ---

# This section defines the CNN model architecture,
# splits the data into training and testing sets, and trains the model.

def create_cnn_model(input_shape=(128, 128, 64)):
    """Creates a CNN model for flare-up detection."""
    model = tf.keras.models.Sequential([
        tf.keras.layers.Input(shape=input_shape),
        tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D((2, 2)),
        tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D((2, 2)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])
    return model

# --- Step 4: Lesion Segmentation ---

# This section defines a function to segment the lesions in the MRI images.
# It uses thresholding and morphological operations to identify and isolate␣
↪lesions.

def segment_lesions(normalized_data):
    """Segments lesions in the image."""
    threshold_value = filters.threshold_otsu(normalized_data)
    binary_lesions = normalized_data > threshold_value
    cleaned_lesions = morphology.remove_small_objects(binary_lesions,␣
↪min_size=500)
    labeled_lesions, num_lesions = measure.label(cleaned_lesions,␣
↪return_num=True)
    properties = measure.regionprops(labeled_lesions,␣
↪intensity_image=normalized_data)
    return labeled_lesions, properties
```

```
# --- Step 5: Feature Extraction and Analysis ---

# This section defines a function to extract various features from the images.
# It includes calculating mean intensity, lesion properties, and texture␣
↪features.

def extract_features(normalized_data, labeled_lesions, properties, brain_vol):
    """Extracts features and performs analysis."""
    features = {
        'mean_intensity': np.mean(normalized_data),
        'max_intensity': np.max(normalized_data),
        'min_intensity': np.min(normalized_data),
        'std_dev': np.std(normalized_data)
    }

    lesion_areas = [prop.area for prop in properties]
    mean_intensities = [prop.mean_intensity for prop in properties]
    lesion_volumes = [prop.area * np.prod(brain_vol.header.get_zooms()) for␣
↪prop in properties]
    total_lesion_volume = sum(lesion_volumes)

    slice_index = normalized_data.shape[2] // 2
    glcm = feature.graycomatrix((normalized_data[:, :, slice_index] * 255).
↪astype('uint8'),
                                distances=[1], angles=[0], symmetric=True,␣
↪normed=True)
    contrast = feature.graycoprops(glcm, 'contrast')[0, 0]
    dissimilarity = feature.graycoprops(glcm, 'dissimilarity')[0, 0]
    homogeneity = feature.graycoprops(glcm, 'homogeneity')[0, 0]
    energy = feature.graycoprops(glcm, 'energy')[0, 0]
    correlation = feature.graycoprops(glcm, 'correlation')[0, 0]

    texture_features = {
        'contrast': contrast,
        'dissimilarity': dissimilarity,
        'homogeneity': homogeneity,
        'energy': energy,
        'correlation': correlation
    }

    return features, lesion_areas, mean_intensities, total_lesion_volume,␣
↪texture_features

# --- Step 6: Visualization ---

# This section defines a function to generate visualizations of the data.
```

```
# It includes plotting the original image, lesion overlay, and lesion␣
↪properties.

def generate_visualizations(normalized_data, labeled_lesions, lesion_areas,␣
↪mean_intensities, output_dir, patient_number):
    """Generates and saves visualizations."""
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    slice_index = min(10 + patient_number % 10, normalized_data.shape[2] - 1)
    #slice_index = 10 + patient_number % 10

    fig, ax = plt.subplots(1, 2, figsize=(10, 5))
    ax[0].imshow(normalized_data[:, :, slice_index], cmap='gray')
    ax[0].set_title('Original Image')
    ax[0].axis('off')
    ax[1].imshow(normalized_data[:, :, slice_index], cmap='gray')
    ax[1].contour(labeled_lesions[:, :, slice_index], colors='r')
    ax[1].set_title('Lesions Overlay')
    ax[1].axis('off')
    plt.tight_layout()
    lesion_overlay_path = os.path.join(output_dir,␣
↪f'lesion_overlay_{patient_number}.png')
    plt.savefig(lesion_overlay_path)
    plt.close(fig)

    plt.hist(lesion_areas, bins=20, color='blue', alpha=0.7)
    plt.title('Lesion Size Distribution')
    plt.xlabel('Area')
    plt.ylabel('Frequency')
    lesion_size_distribution_path = os.path.join(output_dir,␣
↪f'lesion_size_distribution_{patient_number}.png')
    plt.savefig(lesion_size_distribution_path)
    plt.close()

    plt.hist(mean_intensities, bins=20, color='green', alpha=0.7)
    plt.title('Intensity Distribution in Lesions')
    plt.xlabel('Mean Intensity')
    plt.ylabel('Frequency')
    intensity_distribution_path = os.path.join(output_dir,␣
↪f'intensity_distribution_{patient_number}.png')
    plt.savefig(intensity_distribution_path)
    plt.close()

    plt.hist(normalized_data.flatten(), bins=50, color='purple', alpha=0.7)
    plt.title('Overall Intensity Distribution')
    plt.xlabel('Intensity')
```

```
    plt.ylabel('Frequency')
    overall_intensity_path = os.path.join(output_dir,␣
↪f'overall_intensity_distribution_{patient_number}.png')
    plt.savefig(overall_intensity_path)
    plt.close()

    return (
        lesion_overlay_path,
        lesion_size_distribution_path,
        intensity_distribution_path,
        overall_intensity_path,
    )

# --- Step 7: Report Generation ---

# This section defines a function to generate a PDF report summarizing the␣
↪analysis.
# It includes information about the detected flare-ups, features, and␣
↪visualizations.

class PDF(FPDF):
    def header(self):
        self.set_font('Arial', 'B', 12)
        self.cell(0, 10, 'Medical Image Classification Report', 0, 1, 'C')
        self.ln(10)

    def chapter_title(self, title):
        self.set_font('Arial', 'B', 12)
        self.cell(0, 10, title, 0, 1, 'L')
        self.ln(5)

    def chapter_body(self, body):
        self.set_font('Arial', '', 12)
        self.multi_cell(0, 10, body)
        self.ln()

def generate_report(pdf, features, flare_up_detected, lesion_areas,
                    mean_intensities, total_lesion_volume, texture_features,
                    output_dir, visualizations, patient_number):
    """Generates a PDF report for a single patient."""

    pdf.add_page()
    pdf.chapter_title(f'Patient-{patient_number} Analysis')

    pdf.chapter_body(f"Flare-up detected (CNN): {flare_up_detected}")

    pdf.chapter_title('Features')
```

```python
    results_body = "\n".join([f"{key}: {value}" for key, value in features.
↪items()])
    pdf.chapter_body(results_body)

    pdf.chapter_title('Lesion Properties')
    pdf.chapter_body(f"Number of Lesions: {len(lesion_areas)}")
    pdf.chapter_body(f"Total Lesion Volume: {total_lesion_volume:.2f} mm3")

    pdf.chapter_title('Texture Features')
    texture_labels = list(texture_features.keys())
    texture_values = list(texture_features.values())

    plt.figure(figsize=(8, 4))
    plt.bar(texture_labels, texture_values, color='skyblue')
    plt.title(f'Patient-{patient_number} Texture Features')
    plt.ylabel('Value')
    texture_chart_path = os.path.join(output_dir,␣
↪f'texture_features_{patient_number}.png')
    plt.savefig(texture_chart_path)
    plt.close()

    pdf.image(texture_chart_path, x=10, y=None, w=150)

    pdf.chapter_title('Visualizations')
    pdf.image(visualizations[0], x=10, y=None, w=150)
    pdf.image(visualizations[1], x=10, y=None, w=150)
    pdf.image(visualizations[2], x=10, y=None, w=150)
    pdf.image(visualizations[3], x=10, y=None, w=150)

# --- Main Execution ---

if __name__ == "__main__":
    base_directory = '/content/drive/MyDrive/Colab Notebooks/MRI'
    output_dir = '/content/drive/MyDrive/Colab Notebooks/Reports'
    num_patients = 60

    mri_images, labels = load_mri_data(base_directory, num_patients)

    train_images, temp_images, train_labels, temp_labels = train_test_split(
        mri_images, labels, test_size=0.4, random_state=42
    )
    val_images, test_images, val_labels, test_labels = train_test_split(
        temp_images, temp_labels, test_size=0.5, random_state=42
    )

    input_shape = (128, 128, 64)
    model = create_cnn_model(input_shape)
```

```python
    history = model.fit(
        train_images, train_labels, epochs=10, validation_data=(val_images,␣
↪val_labels)
    )

    test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
    print(f"Test accuracy: {test_acc}")

    pdf = PDF()
    for i in range(1, num_patients + 1):
        print(f"Processing Patient-{i}")
        patient_data = load_patient_data(base_directory, i)
        if f'{i}-Flair' in patient_data:
            normalized_data, brain_vol =␣
↪load_and_preprocess(patient_data[f'{i}-Flair'].get_filename())
            labeled_lesions, properties = segment_lesions(normalized_data)
            features, lesion_areas, mean_intensities, total_lesion_volume,␣
↪texture_features = extract_features(
                normalized_data, labeled_lesions, properties, brain_vol
            )

            input_image = preprocess_image(normalized_data)
            flare_up_detected = model.predict(np.expand_dims(input_image,␣
↪axis=0)) > 0.5

            visualizations = generate_visualizations(
                normalized_data, labeled_lesions, lesion_areas,␣
↪mean_intensities, output_dir, i
            )

            generate_report(
                pdf, features, flare_up_detected, lesion_areas,␣
↪mean_intensities,
                total_lesion_volume, texture_features, output_dir,␣
↪visualizations, i
            )

    pdf.output(os.path.join(output_dir, 'Medical_Image_Classification_Report.
↪pdf'))
    print("Report generation complete!")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
Requirement already satisfied: nibabel in /usr/local/lib/python3.10/dist-
packages (5.3.2)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.10/dist-
packages (0.24.0)
```

```
Requirement already satisfied: fpdf in /usr/local/lib/python3.10/dist-packages
(1.7.2)
Requirement already satisfied: importlib-resources>=5.12 in
/usr/local/lib/python3.10/dist-packages (from nibabel) (6.4.5)
Requirement already satisfied: numpy>=1.22 in /usr/local/lib/python3.10/dist-
packages (from nibabel) (1.26.4)
Requirement already satisfied: packaging>=20 in /usr/local/lib/python3.10/dist-
packages (from nibabel) (24.2)
Requirement already satisfied: typing-extensions>=4.6 in
/usr/local/lib/python3.10/dist-packages (from nibabel) (4.12.2)
Requirement already satisfied: scipy>=1.9 in /usr/local/lib/python3.10/dist-
packages (from scikit-image) (1.13.1)
Requirement already satisfied: networkx>=2.8 in /usr/local/lib/python3.10/dist-
packages (from scikit-image) (3.4.2)
Requirement already satisfied: pillow>=9.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-image) (11.0.0)
Requirement already satisfied: imageio>=2.33 in /usr/local/lib/python3.10/dist-
packages (from scikit-image) (2.36.1)
Requirement already satisfied: tifffile>=2022.8.12 in
/usr/local/lib/python3.10/dist-packages (from scikit-image) (2024.9.20)
Requirement already satisfied: lazy-loader>=0.4 in
/usr/local/lib/python3.10/dist-packages (from scikit-image) (0.4)
Original shape for Patient-1: (256, 256, 23)
Original shape for Patient-2: (320, 280, 23)
Original shape for Patient-3: (320, 280, 20)
Original shape for Patient-4: (256, 256, 18)
Original shape for Patient-5: (256, 256, 26)
Original shape for Patient-6: (256, 256, 26)
Original shape for Patient-7: (256, 256, 22)
Original shape for Patient-8: (256, 256, 26)
Original shape for Patient-9: (512, 512, 30)
Original shape for Patient-10: (320, 280, 20)
Original shape for Patient-11: (256, 256, 18)
Original shape for Patient-12: (256, 256, 26)
Original shape for Patient-13: (256, 256, 18)
Original shape for Patient-14: (320, 280, 24)
Original shape for Patient-15: (288, 288, 25)
Original shape for Patient-16: (512, 512, 26)
Original shape for Patient-17: (256, 256, 18)
Original shape for Patient-18: (256, 256, 21)
Original shape for Patient-19: (256, 256, 26)
Original shape for Patient-20: (256, 256, 26)
Original shape for Patient-21: (256, 256, 26)
Original shape for Patient-22: (256, 256, 26)
Original shape for Patient-23: (320, 280, 28)
Original shape for Patient-24: (256, 256, 26)
Original shape for Patient-25: (256, 256, 26)
Original shape for Patient-26: (320, 320, 22)
```

```
Original shape for Patient-27: (256, 256, 26)
Original shape for Patient-28: (320, 280, 21)
Original shape for Patient-29: (256, 256, 18)
Original shape for Patient-30: (256, 256, 23)
Original shape for Patient-31: (448, 512, 27)
Original shape for Patient-32: (256, 256, 25)
Original shape for Patient-33: (320, 280, 20)
Original shape for Patient-34: (512, 512, 28)
Original shape for Patient-35: (512, 448, 20)
Original shape for Patient-36: (256, 256, 22)
Original shape for Patient-37: (320, 320, 30)
Original shape for Patient-38: (256, 256, 32)
Original shape for Patient-39: (512, 368, 21)
Original shape for Patient-40: (512, 448, 33)
Original shape for Patient-41: (512, 512, 28)
Original shape for Patient-42: (448, 512, 19)
Original shape for Patient-43: (512, 512, 18)
Original shape for Patient-44: (256, 256, 26)
Original shape for Patient-45: (512, 512, 24)
Original shape for Patient-46: (256, 256, 23)
Original shape for Patient-47: (256, 256, 26)
Original shape for Patient-48: (256, 256, 25)
Original shape for Patient-49: (256, 256, 26)
Original shape for Patient-50: (256, 256, 26)
Original shape for Patient-51: (256, 256, 26)
Original shape for Patient-52: (512, 512, 22)
Original shape for Patient-53: (256, 176, 19)
Original shape for Patient-54: (320, 180, 24)
Original shape for Patient-55: (224, 224, 29)
Original shape for Patient-56: (512, 480, 30)
Original shape for Patient-57: (512, 480, 27)
Original shape for Patient-58: (512, 448, 25)
Original shape for Patient-59: (256, 224, 22)
Original shape for Patient-60: (288, 288, 24)
Epoch 1/10
2/2 [==============================] - 2s 386ms/step - loss: 1.0748 - accuracy:
0.1389 - val_loss: 0.3764 - val_accuracy: 0.9167
Epoch 2/10
2/2 [==============================] - 1s 182ms/step - loss: 0.4535 - accuracy:
0.8611 - val_loss: 0.5283 - val_accuracy: 0.9167
Epoch 3/10
2/2 [==============================] - 1s 182ms/step - loss: 0.4622 - accuracy:
0.9444 - val_loss: 0.3658 - val_accuracy: 0.9167
Epoch 4/10
2/2 [==============================] - 1s 184ms/step - loss: 0.2963 - accuracy:
0.8611 - val_loss: 0.3938 - val_accuracy: 0.9167
Epoch 5/10
2/2 [==============================] - 1s 172ms/step - loss: 0.5009 - accuracy:
```

```
0.8611 - val_loss: 0.3727 - val_accuracy: 0.9167
Epoch 6/10
2/2 [==============================] - 1s 180ms/step - loss: 0.3818 - accuracy:
0.8611 - val_loss: 0.3333 - val_accuracy: 0.9167
Epoch 7/10
2/2 [==============================] - 1s 174ms/step - loss: 0.2251 - accuracy:
0.9444 - val_loss: 0.4173 - val_accuracy: 0.8333
Epoch 8/10
2/2 [==============================] - 1s 178ms/step - loss: 0.2627 - accuracy:
0.9444 - val_loss: 0.4034 - val_accuracy: 0.8333
Epoch 9/10
2/2 [==============================] - 1s 178ms/step - loss: 0.2120 - accuracy:
0.9444 - val_loss: 0.4170 - val_accuracy: 0.9167
Epoch 10/10
2/2 [==============================] - 1s 180ms/step - loss: 0.2674 - accuracy:
0.9167 - val_loss: 0.4058 - val_accuracy: 0.9167
1/1 - 0s - loss: 0.0318 - accuracy: 1.0000 - 76ms/epoch - 76ms/step
Test accuracy: 1.0
Processing Patient-1
1/1 [==============================] - 0s 76ms/step
Processing Patient-2
1/1 [==============================] - 0s 34ms/step
Processing Patient-3
1/1 [==============================] - 0s 35ms/step
Processing Patient-4
1/1 [==============================] - 0s 35ms/step
Processing Patient-5
1/1 [==============================] - 0s 36ms/step
Processing Patient-6
1/1 [==============================] - 0s 38ms/step
Processing Patient-7
1/1 [==============================] - 0s 34ms/step
Processing Patient-8
1/1 [==============================] - 0s 36ms/step
Processing Patient-9
1/1 [==============================] - 0s 34ms/step
Processing Patient-10
1/1 [==============================] - 0s 35ms/step
Processing Patient-11
1/1 [==============================] - 0s 35ms/step
Processing Patient-12
1/1 [==============================] - 0s 36ms/step
Processing Patient-13
1/1 [==============================] - 0s 36ms/step
Processing Patient-14
1/1 [==============================] - 0s 35ms/step
Processing Patient-15
1/1 [==============================] - 0s 33ms/step
```

```
Processing Patient-16
1/1 [==============================] - 0s 36ms/step
Processing Patient-17
1/1 [==============================] - 0s 33ms/step
Processing Patient-18
1/1 [==============================] - 0s 34ms/step
Processing Patient-19
1/1 [==============================] - 0s 31ms/step
Processing Patient-20
1/1 [==============================] - 0s 32ms/step
Processing Patient-21
1/1 [==============================] - 0s 34ms/step
Processing Patient-22
1/1 [==============================] - 0s 37ms/step
Processing Patient-23
1/1 [==============================] - 0s 34ms/step
Processing Patient-24
1/1 [==============================] - 0s 34ms/step
Processing Patient-25
1/1 [==============================] - 0s 33ms/step
Processing Patient-26
1/1 [==============================] - 0s 31ms/step
Processing Patient-27
1/1 [==============================] - 0s 34ms/step
Processing Patient-28
1/1 [==============================] - 0s 31ms/step
Processing Patient-29
1/1 [==============================] - 0s 35ms/step
Processing Patient-30
1/1 [==============================] - 0s 35ms/step
Processing Patient-31
1/1 [==============================] - 0s 33ms/step
Processing Patient-32
1/1 [==============================] - 0s 34ms/step
Processing Patient-33
1/1 [==============================] - 0s 31ms/step
Processing Patient-34
1/1 [==============================] - 0s 33ms/step
Processing Patient-35
1/1 [==============================] - 0s 32ms/step
Processing Patient-36
1/1 [==============================] - 0s 32ms/step
Processing Patient-37
1/1 [==============================] - 0s 31ms/step
Processing Patient-38
1/1 [==============================] - 0s 34ms/step
Processing Patient-39
1/1 [==============================] - 0s 34ms/step
```

```
Processing Patient-40
1/1 [==============================] - 0s 33ms/step
Processing Patient-41
1/1 [==============================] - 0s 33ms/step
Processing Patient-42
1/1 [==============================] - 0s 33ms/step
Processing Patient-43
1/1 [==============================] - 0s 33ms/step
Processing Patient-44
1/1 [==============================] - 0s 33ms/step
Processing Patient-45
1/1 [==============================] - 0s 35ms/step
Processing Patient-46
1/1 [==============================] - 0s 30ms/step
Processing Patient-47
1/1 [==============================] - 0s 35ms/step
Processing Patient-48
1/1 [==============================] - 0s 34ms/step
Processing Patient-49
1/1 [==============================] - 0s 32ms/step
Processing Patient-50
1/1 [==============================] - 0s 33ms/step
Processing Patient-51
1/1 [==============================] - 0s 33ms/step
Processing Patient-52
1/1 [==============================] - 0s 33ms/step
Processing Patient-53
1/1 [==============================] - 0s 35ms/step
Processing Patient-54
1/1 [==============================] - 0s 32ms/step
Processing Patient-55
1/1 [==============================] - 0s 35ms/step
Processing Patient-56
1/1 [==============================] - 0s 33ms/step
Processing Patient-57
1/1 [==============================] - 0s 33ms/step
Processing Patient-58
1/1 [==============================] - 0s 37ms/step
Processing Patient-59
1/1 [==============================] - 0s 34ms/step
Processing Patient-60
1/1 [==============================] - 0s 34ms/step
Report generation complete!
```