Smart Systems Project Report

Brain Tumor Detection System Using Deep Learning

Supervised by:

Dr. Yasser Fouad

Eng. Jailan Elshazly

Team Members:

Lina Gamal - [2403244631]

Merna Ayman - [2403244131]

Kenzy Sherif - [2403241986]

Heba Mohamed - [2403241411]

Nour Mohamed - [2403241277]

Hana Mohamed - [2403242224]

1 Project Overview & Problem Description

1.1 Problem Statement

Brain tumors are complex conditions that require rapid and accurate diagnosis to improve patient survival rates. Manual analysis of MRI scans is time-consuming and prone to human error. The goal of this project is to develop an automated **Brain Tumor Detection System** using Deep Learning techniques.

1.2 Solution

We have built a Convolutional Neural Network (CNN) based system that:

- **Preprocesses** MRI images to remove noise and enhance contrast.
- **Augments** data to improve model generalization.
- **Classifies** images as "Tumor" or "No Tumor" using a Transfer Learning approach (ResNet50V2).
- **Deploys** a user-friendly GUI for real-time diagnostics.

2 Methodology & System Architecture

Our pipeline consists of four main stages: Data Preprocessing, Data Augmentation, Model Training, and Interface Deployment.

2.1 A. Data Preprocessing (Noise Reduction & Contrast Enhancement)

MRI scans often contain noise and varying lighting conditions. We implemented a robust preprocessing pipeline using `skimage` and `OpenCV`.

- **Denoising:** We used **Non-Local Means (NLM) Denoising**, which removes noise while preserving edge details (crucial for tumor boundaries).
- **Contrast Enhancement:** We converted images to the **LAB color space** and applied **CLAHE (Contrast Limited Adaptive Histogram Equalization)** to the L-channel (Lightness). This highlights tumor textures without distorting colors.

Preprocessing Code Snippet:

```
def clean_and_enhance_images(images, description="Processing"):
    # ...
    # 1. Denoise
    cleaned_image = denoise_nl_means(
        image_as_float,
        h=noise_reduction_strength * estimated_noise,
        patch_size=patch_comparison_size,
        patch_distance=patch_search_distance,
    )

    # 2. Enhance Contrast (CLAHE on LAB space)
    lab_image = cv2.cvtColor(image_8bit, cv2.COLOR_RGB2LAB)
    lightness, color_a, color_b = cv2.split(lab_image)
    enhanced_lightness = contrast_enhancer.apply(lightness)
    # ...
```

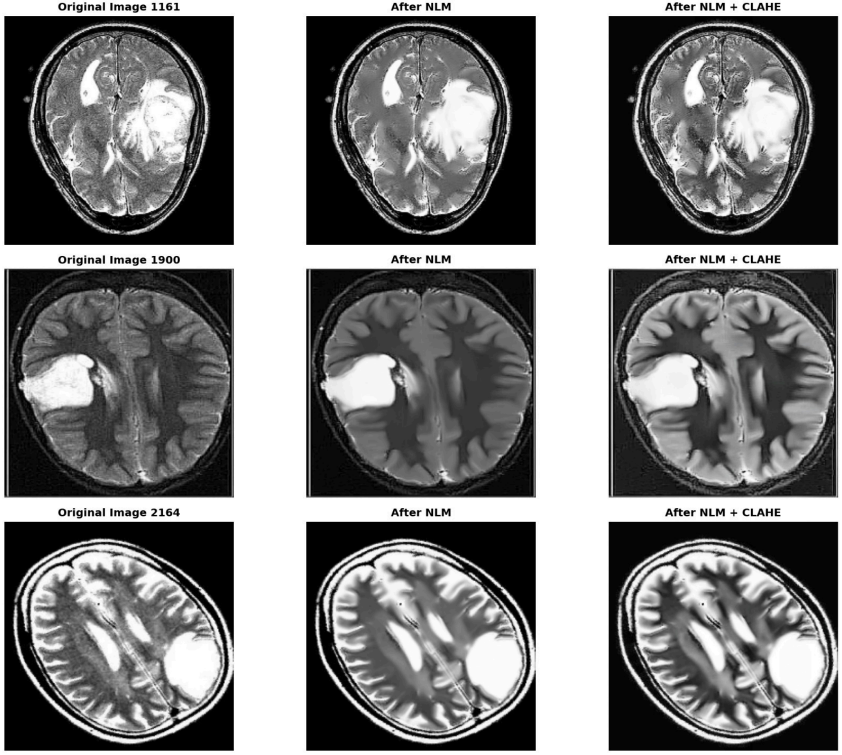

図 1: Comparison of Original, Denoised, and Enhanced MRI Scans.

2.2 B. Data Augmentation

To prevent overfitting and handle the limited size of medical datasets, we applied dynamic data augmentation during training. This creates ”new” training samples by modifying existing ones.

Techniques Used:

- Random Flip: Horizontal mirroring.
- Random Rotation: $\pm 15\%$ rotation.
- Random Zoom: $\pm 10\%$ zoom.
- Brightness/Contrast Adjustments: Simulating different MRI scan qualities.

Augmentation Code Snippet:

```
augmentation_settings = {
    "flip_horizontal": True,
    "rotation": True,
    "zoom": True,
    "contrast": True,
    "brightness": True,
}
# ...
augmentation_layers.append(layers.RandomRotation(0.15))
augmentation_layers.append(layers.RandomZoom(0.1))
```

2.3 C. Model Architecture (Transfer Learning)

We utilized **Transfer Learning** with the **ResNet50V2** architecture.

1. **Backbone:** Pre-trained ResNet50V2 (trained on ImageNet) extracts high-level features.
2. **Fine-Tuning:** We initially froze the backbone to train only the head, then un-froze the top 40 layers for fine-tuning.
3. **Classification Head:**

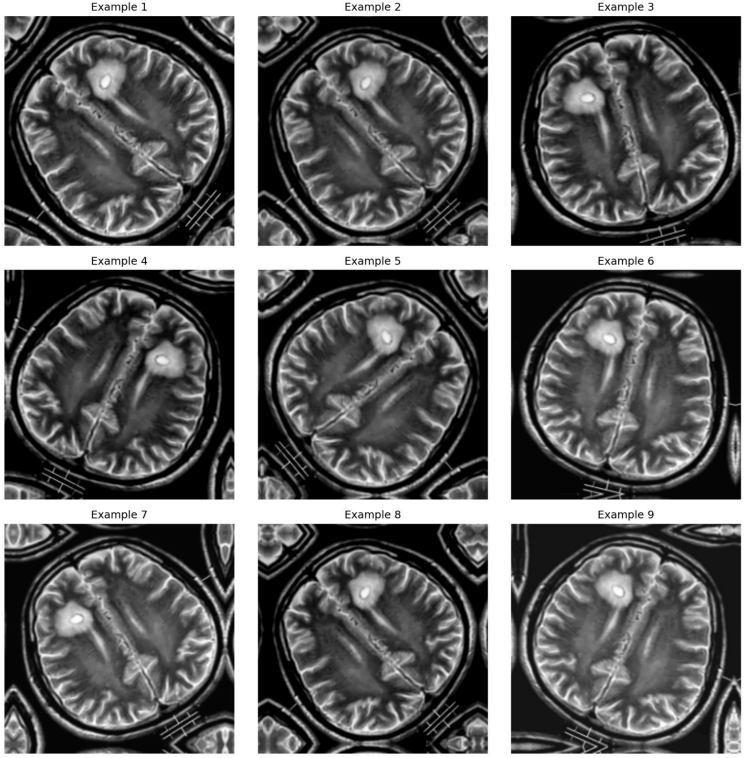

図 2: Augmented variations of a single MRI scan used for training.

- `GlobalAveragePooling2D`: Reduces spatial dimensions.
- `Dense (256 units)` + `BatchNormalization` + `ReLU`: Learn specific tumor features.
- `Dropout (0.5)`: Prevents overfitting.
- `Dense (1 unit, Sigmoid)`: Outputs probability (0 to 1).

Model Builder Code:

```
def build_model(input_shape=(224, 224, 3), backbone='ResNet50V2'):
    base_model = ResNet50V2(include_top=False,
                            weights='imagenet',
                            input_shape=input_shape)

    x = base_model(inputs, training=False)
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dense(256)(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)
    x = layers.Dropout(0.5)(x)
    outputs = layers.Dense(1, activation='sigmoid')(x)

    return keras.Model(inputs, outputs)
```

3 Results & Final Analysis

3.1 Performance Metrics

The model was evaluated on a held-out test set. The fine-tuning phase significantly improved performance.

- **Accuracy:** $\approx 87\%$
- **AUC Score:** 0.9168 (Indicates excellent capability to distinguish between classes)

- **Precision/Recall:** High recall is prioritized to ensure no tumors are missed.

3.2 Visual Analysis

3.2.1 1. Confusion Matrix

The confusion matrix below details the prediction breakdown.

- **True Positives (TP):** 30 (Tumors correctly identified)
- **True Negatives (TN):** 15 (Healthy brains correctly identified)
- **False Negatives (FN):** 1 (Critical error - missed tumor)
- **False Positives (FP):** 4 (Healthy brain flagged as tumor)

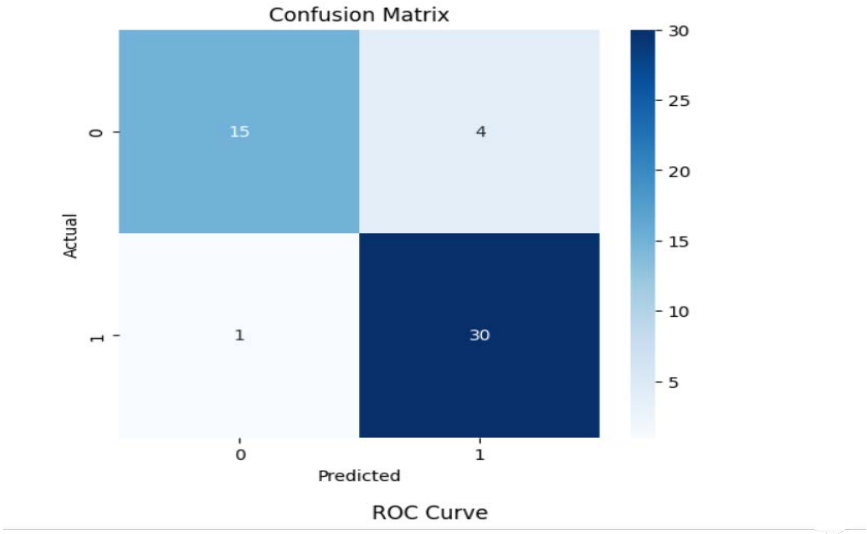

図 3: Confusion Matrix showing high sensitivity.

3.2.2 2. ROC Curve

The Receiver Operating Characteristic (ROC) curve shows the trade-off between sensitivity and specificity. An **AUC of 0.9168** confirms the model is robust.

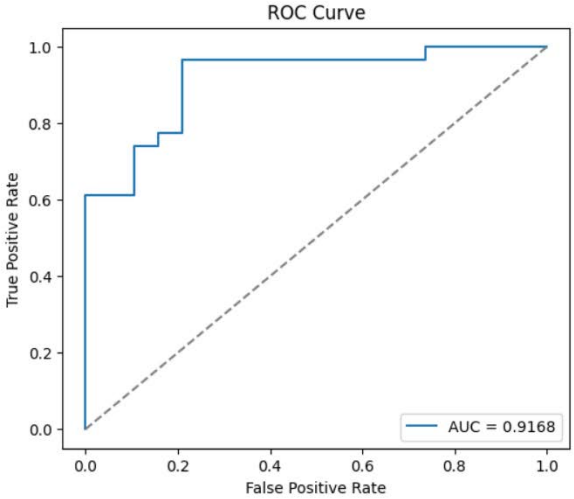

図 4: ROC Curve with AUC score.

3.2.3 3. Training Dynamics

The training curves show the Loss decreasing and Accuracy increasing. The validation accuracy (orange) tracks the training accuracy (blue), indicating that **overfitting was successfully managed** via Dropout and Augmentation.

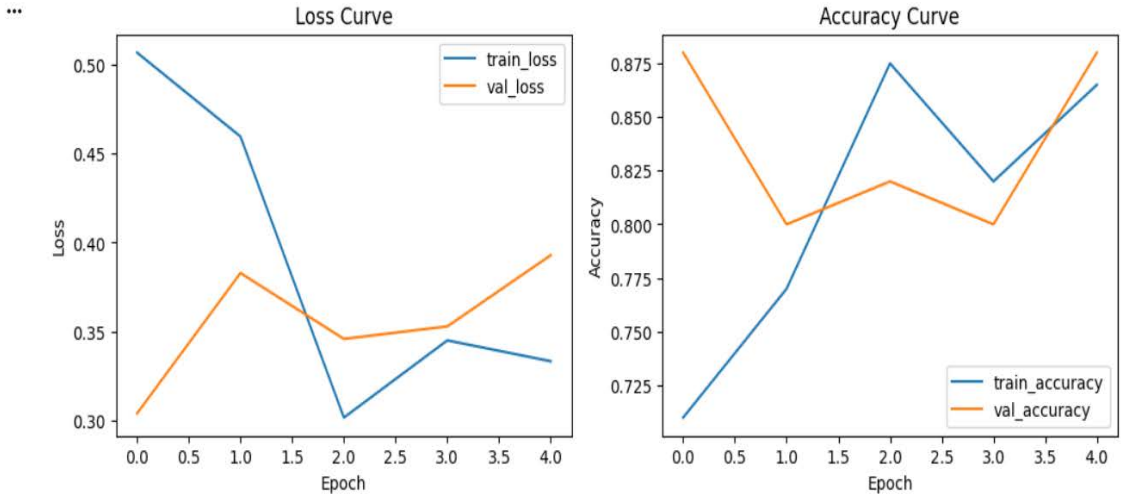

図 5: Loss and Accuracy over Epochs.

4 User Interface (Deployment)

We developed a Graphical User Interface (GUI) using **Gradio**. This allows medical staff to upload an MRI image and receive an instant diagnostic suggestion with a confidence score.

Interface Features:

- **Drag-and-Drop:** Easy image upload.
- **Real-time Processing:** Applies the same preprocessing pipeline before prediction.
- **Output:** ”TUMOR DETECTED” or ”NO TUMOR” with probability.

5 Conclusion

The Smart Systems project successfully delivered an end-to-end solution for brain tumor detection. By combining advanced image enhancement (CLAHE + NLM) with state-of-the-art Deep Learning (ResNet50V2), we achieved a reliable diagnostic tool. The system is wrapped in a user-friendly interface, making it accessible for practical use. Future improvements could include 3D MRI processing and multi-class classification for specific tumor types.

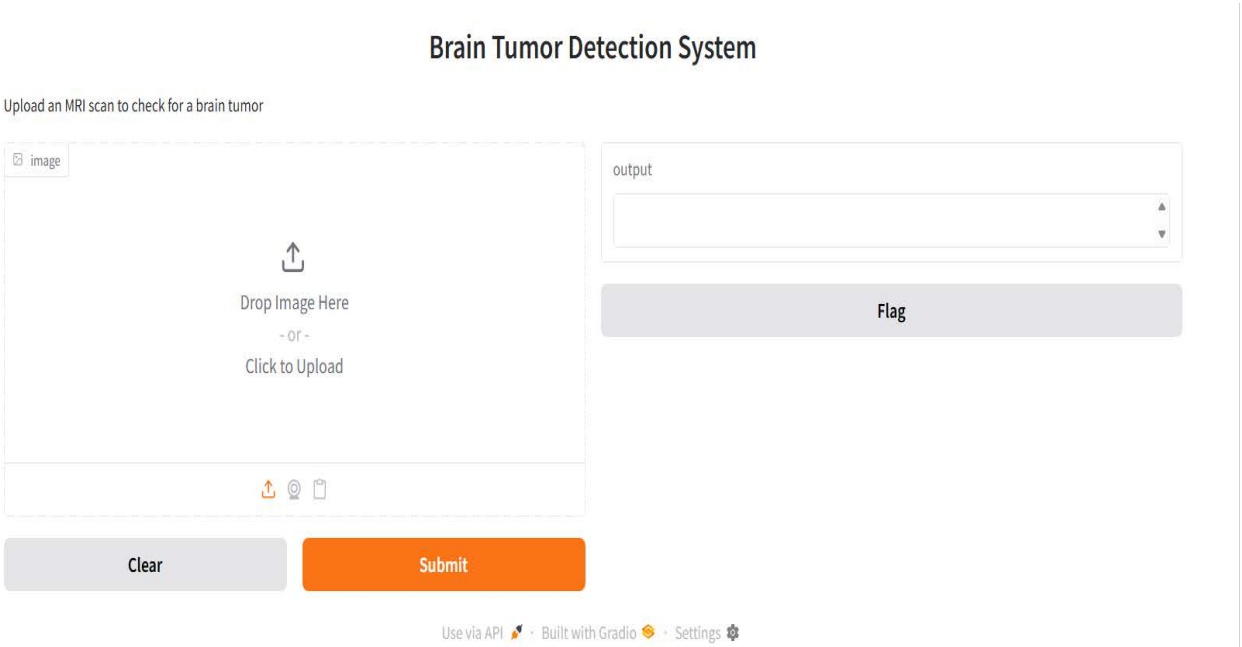

図 6: The deployed Brain Tumor Detection System interface.