Detailed Project Report (DPR)

Project Title: CNN-Based Brain Scan Image Classification with Flask Web Application

1. Introduction

This project aims to develop a Convolutional Neural Network (CNN) for classifying brain scan images. The primary objective is to detect abnormalities in brain scans and provide results through a Flask-based web application. Users can upload brain scan images via the web interface and receive classification results instantly.

2. Objectives

1. Build a robust CNN model for classifying brain scans.
2. Develop a user-friendly Flask web application for image upload and result display.
3. Ensure high accuracy and efficiency in model predictions.
4. Provide a visually appealing and intuitive interface for end-users.

3. Scope

- Target Audience: Healthcare professionals, researchers, and patients.
- Functionality:
  - Upload brain scan images.
  - Process and classify images using a trained CNN model.
  - Display the classification results on the same webpage.
- Technologies Used: Python, TensorFlow, Flask, HTML/CSS, Bootstrap, Matplotlib, Seaborn.
- Deployment Platform: Web-based application (local server or cloud-hosted).

4. System Architecture

4.1 Components

1. Frontend: HTML, CSS, and Bootstrap for creating a responsive interface.
2. Backend: Flask framework to handle requests, process images, and interact with the CNN model.
3. Model: CNN model trained for binary classification (normal vs abnormal scans).
4. Database: Optionally store uploaded images and results for future reference (e.g., SQLite).

4.2 Workflow

1. User uploads a brain scan image via the web app.
2. Flask handles the image upload and passes it to the CNN model.

3. **The CNN model processes the image and returns the classification result.**
4. **Flask sends the result back to the frontend for display.**

5. Project Implementation

5.1 Dataset Preparation

- **Source: Publicly available brain scan datasets (e.g., Kaggle, Open Access).**
- **Preprocessing:**
  - **Resize images to 256x256 pixels.**
  - **Normalize pixel values to [0, 1].**
  - **Split data into training, validation, and test sets.**

5.2 CNN Model Development

- **Architecture:**
  - **Input Layer: 256x256x3**
  - **Three Convolutional Layers with ReLU activation and MaxPooling.**
  - **Fully Connected Layers with ReLU activation.**
  - **Output Layer with Sigmoid activation for binary classification.**
- **Hyperparameters:**
  - **Optimizer: Adam**
  - **Loss Function: Binary Crossentropy**
  - **Metrics: Accuracy**
  - **Epochs: 5**
- **Training: Train the model on the prepared dataset and evaluate using validation data.**

5.3 Flask Web Application Development

Key Features:

1. **Homepage:**
   - **Title: "Brain Scan Classification System."**
   - **Options: Upload Image, View Results.**
2. **Image Upload Page:**
   - **Upload button for submitting brain scan images.**
   - **Submit button to classify the image.**
3. **Result Page:**
   - **Display the classification result (e.g., "Normal" or "Abnormal").**
   - **Confidence score of the prediction.**
4. **Visualization: Option to display model performance metrics (e.g., loss and accuracy graphs).**

5.4 Deployment

- **Local Deployment: Run the Flask app locally for development and testing.**
- **Cloud Deployment: Host the application on platforms like AWS, Heroku, or Google Cloud for accessibility.**

6. Testing and Evaluation

1. **Model Testing:**
   - **Evaluate on unseen test data to measure accuracy, precision, recall, and F1-score.**
2. **Web App Testing:**
   - **Functional testing for image upload and result display.**
   - **Usability testing for user interface.**
3. **Performance Metrics:**
   - **Confusion matrix for classification results.**
   - **Accuracy and loss graphs.**

7. Cost Estimation

1. **Development Tools: Free (Python, Flask, TensorFlow, OpenCV).**
2. **Hosting Services: Approx. $10-50/month depending on traffic.**
3. **Hardware Requirements:**
   - **Development: Laptop/PC with GPU support.**
   - **Deployment: Cloud hosting services.**

8. Challenges and Mitigation

1. **Challenge:Obtaining high-quality brain scan datasets.**
   - **Mitigation: Use publicly available datasets and perform augmentation.**
2. **Challenge:Ensuring real-time predictions.**
   - **Mitigation: Optimize the CNN model for faster inference.**
3. **Challenge:User interface design.**
   - **Mitigation: Use Bootstrap for responsive and user-friendly designs.**

9. Project Timeline

Task	Duration
Dataset Collection	1 Week
Model Development	2 Weeks
Web App Development	2 Weeks
Testing and Deployment	1 Week
Documentation and Finalization	1 Week

10. Conclusion

This project combines deep learning and web technologies to create a practical tool for brain scan classification. It has significant potential in assisting healthcare professionals with quick and accurate diagnoses. The Flask-based web application ensures accessibility and ease of use for non-technical users.