Assessment Report

on

"Brain Tumor Detection Using CNNs"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

Brajesh Kumar Keshari(202401100300090)

Lucky Singh(202401100300146)

Gavendra Pachahra(202401100300111)

Dhruv Goel(202401100300100)

Section: B

Under the supervision of

"Mr Shivansh Prasad"

KIET Group of Institutions, Ghaziabad

1. Introduction

Brain tumors are a severe health threat that can be fatal if not diagnosed early. Radiologists often use Magnetic Resonance Imaging (MRI) to detect the presence of tumors, but manual analysis can be time-consuming and error-prone. With the advancement in deep learning, especially Convolutional Neural Networks (CNNs), automated brain tumor detection has become more accurate and efficient. This project aims to build a CNN model to classify MRI brain images as either **tumor** or **no tumor**, leveraging a labeled dataset of images.

---

2. Problem Statement

Implement an image classification model using CNNs to detect brain tumors from MRI images. Visualize predictions and performance.

---

3. Objectives

The primary objective of this project is to develop an automated and accurate image classification system using **Convolutional Neural Networks (CNNs)** to detect the presence of brain tumors in **MRI (Magnetic Resonance Imaging)** scans. The system aims to assist medical professionals in early diagnosis by:

- Classifying MRI images into **tumor** or **no tumor** categories.
- Reducing the time and potential human error involved in manual analysis.
- Demonstrating the potential of deep learning in improving healthcare diagnostics.

This project leverages a publicly available MRI dataset and builds a deep learning pipeline that includes preprocessing, model training, evaluation, and result visualization to validate the effectiveness of CNN-based approaches in medical image classification tasks.

---

4. Methodology

- **Data Collection**: We used the publicly available dataset from Kaggle.
- **Data Preprocessing**:
  - Resized images to 150x150 pixels.
  - Normalized pixel values between 0 and 1.
  - Split the data into 80% training and 20% validation.

5. Model Architecture:

- 3 Convolutional Layers (Conv2D + MaxPooling)
- Flatten layer
- Fully connected Dense layers
- Dropout to prevent overfitting
- Sigmoid output for binary classification

6. Model Evaluation:

- **Validation Accuracy and Loss** curves..
- **Prediction Visualization:** The model was tested on unseen validation images to verify predictions.
- **Accuracy Trends:** Plots of training and validation accuracy/loss over epochs showed model learning behavior.
- **Output Inspection:** Predicted labels were compared with ground truth visually.

---

7. Model Implementation

The model was implemented using TensorFlow and Keras in Google Colab. It consists of a Convolutional Neural Network (CNN) with multiple Conv2D and MaxPooling layers, followed by Dense and Dropout layers for classification. The model takes 150x150 pixel MRI images as input and outputs a binary prediction—tumor or no tumor. It was trained using binary crossentropy loss and the Adam optimizer for efficient learning.

---

8. Evaluation Metrics

The following metrics are used to evaluate the model:

- **Accuracy**: Measures overall correctness.
- **Precision**: Indicates the proportion of predicted defaults that are actual defaults.
- **Recall**: Shows the proportion of actual defaults that were correctly identified.
- **F1 Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: Visualized using Seaborn heatmap to understand prediction errors.

---

9. Results and Analysis

- The model showed good accuracy and generalization on the validation set..
- The confusion matrix heatmap revealed a strong balance between true positives and false negatives..

- Precision and recall metrics reflected the model's reliability in detecting tumors versus false alarms.

---

10. Conclusion

This project successfully demonstrated the use of Convolutional Neural Networks (CNNs) for detecting brain tumors from MRI images.
The model achieved promising accuracy and showed strong potential for assisting in early medical diagnosis.
Performance evaluation using precision, recall, and confusion matrix confirmed its reliability.
The implementation proves that deep learning can significantly enhance diagnostic accuracy.
Future improvements could include data augmentation and transfer learning for better generalization.

---

11. References

- Navoneel Chakrabarty. *Brain MRI Images for Brain Tumor Detection*. Kaggle. https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection

- TensorFlow. *TensorFlow: An end-to-end open source machine learning platform*. https://www.tensorflow.org

- A. Krizhevsky, I. Sutskever, and G. E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. Advances in Neural Information Processing Systems, 2012

---

12. Code

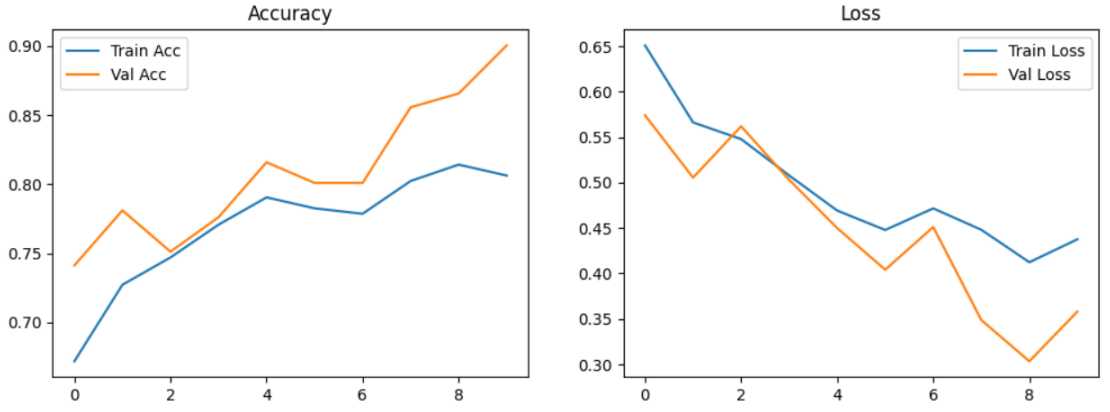
```
import numpy as np

import matplotlib.pyplot as plt

import os

import seaborn as sns

from sklearn.metrics import classification_report, confusion_matrix

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.optimizers import Adam        train_path =
'/content/drive/MyDrive/Colab Notebooks/Dataset/brain/train'

val_path = '/content/drive/MyDrive/Colab Notebooks/Dataset/brain/val'

test_path = '/content/drive/MyDrive/Colab Notebooks/Dataset/brain/test'  img_size =
(128, 128)


train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=10,
zoom_range=0.1, horizontal_flip=True)

val_datagen = ImageDataGenerator(rescale=1./255)


train_generator = train_datagen.flow_from_directory(train_path, target_size=img_size,
batch_size=32, class_mode='binary')

val_generator = val_datagen.flow_from_directory(val_path, target_size=img_size,
batch_size=32, class_mode='binary')

test_generator = val_datagen.flow_from_directory(test_path, target_size=img_size,
batch_size=1, class_mode='binary', shuffle=False)  model = Sequential([
```

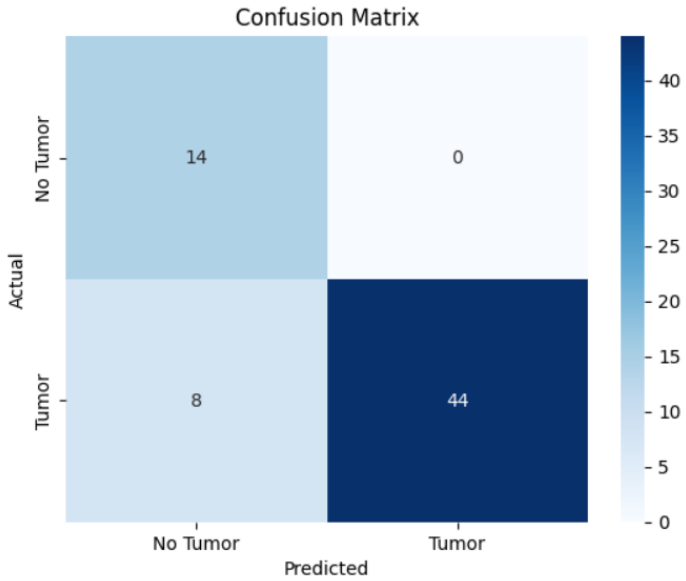
```
    Conv2D(32, (3,3), activation='relu', input_shape=(128,128,3)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])
model.summary()
history = model.fit(train_generator, validation_data=val_generator, epochs=10)
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.legend()
plt.title('Accuracy')

plt.subplot(1,2,2)
```

```
plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val_loss'], label='Val Loss')

plt.legend()

plt.title('Loss')

plt.show()

  y_pred_probs = model.predict(test_generator)

y_pred = (y_pred_probs > 0.5).astype(int)

y_true = test_generator.classes


print("Classification Report:")

print(classification_report(y_true, y_pred))


conf_mat = confusion_matrix(y_true, y_pred)

sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues', xticklabels=['No Tumor',
'Tumor'], yticklabels=['No Tumor', 'Tumor'])

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()   class_labels = list(test_generator.class_indices.keys())


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

for i in range(9):

   img, label = next(test_generator)
```

```
    pred = model.predict(img)[0][0]
    plt.subplot(3, 3, i + 1)
    plt.imshow(img[0])
    plt.title(f"True: {class_labels[int(label[0])]}, Pred: {class_labels[int(pred > 0.5)]}")
    plt.axis('off')

plt.tight_layout()
plt.show()
```

13. Output





```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step
```

⚠ Brain Tumor Detected (92.26% confidence)