Assessment Report

On

"Brain Tumor Detection"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY

DEGREE

SESSION 2024-25

in

CSE(AI)

By

Name : Ansh Rajput

Roll Number : 202401100300052

Section: A

Under the supervision of

"Mr. Bikki Gupta"

KIET Group of Institutions, Ghaziabad

27/05/2025

Introduction

Brain tumors are abnormal growths of cells in the brain that can be life-threatening. Early and accurate detection is crucial for effective treatment. This project utilizes CNNs to classify MRI images as either tumorous or non-tumorous, leveraging deep learning to enhance diagnostic accuracy. *Insert relevant images of MRI scans for better understanding.*

---

2. Problem Statement

Implement an image classification model using CNNs to detect brain tumors from

MRI images. Visualize predictions and performance.

---

3. Objectives

1. **Accurate Classification:** Develop a CNN-based model that can effectively classify MRI images as tumorous or non-tumorous, ensuring high precision and reliability in medical diagnostics.

2. **Performance Optimization:** Improve model efficiency by fine-tuning hyperparameters, implementing data augmentation, and using evaluation metrics like accuracy, precision, recall, and F1-score.

3. **Visual Interpretation:** Provide meaningful visualizations of predictions using heatmaps, accuracy/loss graphs, and sample classifications to understand the model's decision-making process.

4. **Medical Advancement:** Contribute to the field of healthcare and medical imaging by leveraging AI to assist radiologists in early tumor detection, potentially improving patient outcomes.

Methodology

1. **Dataset Acquisition** – Utilize a publicly available brain MRI dataset such as Kaggle's Brain Tumor Detection dataset.

2. **Preprocessing** – Resize images, normalize pixel values, and augment data to improve generalization.

3. **CNN Model Architecture** – Implement a deep learning model with convolutional layers, ReLU activation, pooling layers, and fully connected layers.

4. **Training** – Train the model using labeled MRI images, employing techniques like dropout and batch normalization for optimization.

5. **Evaluation** – Assess performance using metrics like accuracy, confusion matrix, precision, recall, and F1-score.

6. **Visualization** – Generate classification reports and visualize predictions with heatmaps.

---

2. Data Preprocessing

The dataset is cleaned and prepared as follows:

- Resize images to 128x128 pixels, normalize pixel values.

- Apply **data augmentation** to improve generalization.

- Convert images into arrays for model training.

---

3. Model Implementation

- Implement a **CNN architecture** with convolutional layers, ReLU activation, max pooling, and fully connected layers.

- Use **Adam optimizer** with binary cross-entropy loss function for classification.

---

4. Evaluation Metrics

The following metrics are used to evaluate the model:

- **Accuracy:** Measures correct classifications.

- **Precision:** Proportion of correctly identified tumor cases.

- **Recall:** Percentage of actual tumor cases detected.

- **F1 Score:** Balance between precision and recall.

- **Confusion Matrix:** Visualized using heatmaps.

Code

```python
from google.colab import files
uploaded = files.upload()
```

Browse... archive.zip
archive.zip(application/x-zip-compressed) - 15828590 bytes, last modified: n/a - 100% done
Saving archive.zip to archive.zip

```python
import zipfile

with zipfile.ZipFile('archive.zip', 'r') as zip_ref:
    zip_ref.extractall('brain_mri')


import os

print(os.listdir('brain_mri'))
```

['yes', 'no', 'brain_tumor_dataset']

```python
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
import os


data_dir = 'brain_mri'

datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)

train_gen = datagen.flow_from_directory(
    data_dir,
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical',
    subset='training',
    shuffle=True
)
```

```python
val_gen = datagen.flow_from_directory(
    data_dir,
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical',
    subset='validation',
    shuffle=False
)
```

```
Found 406 images belonging to 3 classes.
Found 100 images belonging to 3 classes.
```

```python
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(128,128,3)),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(train_gen.num_classes, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```python
history = model.fit(
    train_gen,
    epochs=10,
    validation_data=val_gen
)
```

```
Epoch 1/10
13/13 ───────────────── 13s 808ms/step - accuracy: 0.4586 - loss: 2.1938 - val_acc
Epoch 2/10
13/13 ───────────────── 11s 849ms/step - accuracy: 0.4995 - loss: 1.0431 - val_acc
Epoch 3/10
13/13 ───────────────── 11s 847ms/step - accuracy: 0.4711 - loss: 0.9388 - val_acc
Epoch 4/10
13/13 ───────────────── 11s 830ms/step - accuracy: 0.4679 - loss: 0.9178 - val_acc
Epoch 5/10
13/13 ───────────────── 11s 831ms/step - accuracy: 0.5426 - loss: 0.8508 - val_acc
Epoch 6/10
13/13 ───────────────── 10s 751ms/step - accuracy: 0.5057 - loss: 0.8375 - val_acc
Epoch 7/10
13/13 ───────────────── 11s 795ms/step - accuracy: 0.6095 - loss: 0.7672 - val_acc
```

```python
x, y = next(val_gen)
preds = model.predict(x)
class_names = list(train_gen.class_indices.keys())

plt.figure(figsize=(12,6))
for i in range(8):
    plt.subplot(2,4,i+1)
    plt.imshow(x[i])
    true_label = class_names[np.argmax(y[i])]
    pred_label = class_names[np.argmax(preds[i])]
    plt.title(f'True: {true_label}\nPred: {pred_label}')
    plt.axis('off')
plt.tight_layout()
plt.show()
```

Results and Analysis

1. **Graphical representation** of training accuracy and loss curves.

```
Epoch 8/10
13/13 ───────────────── 11s 837ms/step - accuracy: 0.5180 - loss: 0.8210 - val_acc
Epoch 9/10
13/13 ───────────────── 11s 853ms/step - accuracy: 0.5365 - loss: 0.7477 - val_acc
Epoch 10/10
13/13 ───────────────── 20s 836ms/step - accuracy: 0.5667 - loss: 0.7234 - val_acc
```

```python
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```
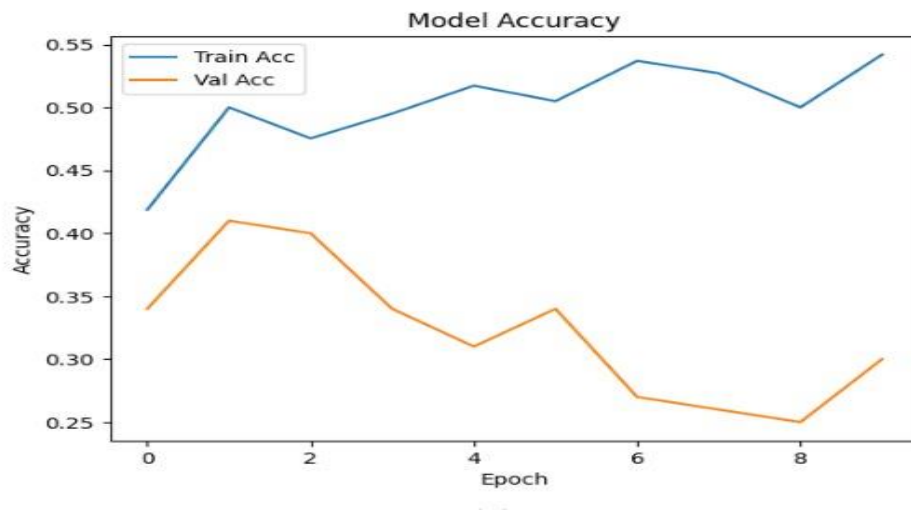
2. **Confusion matrix heatmap** to understand model behavior.
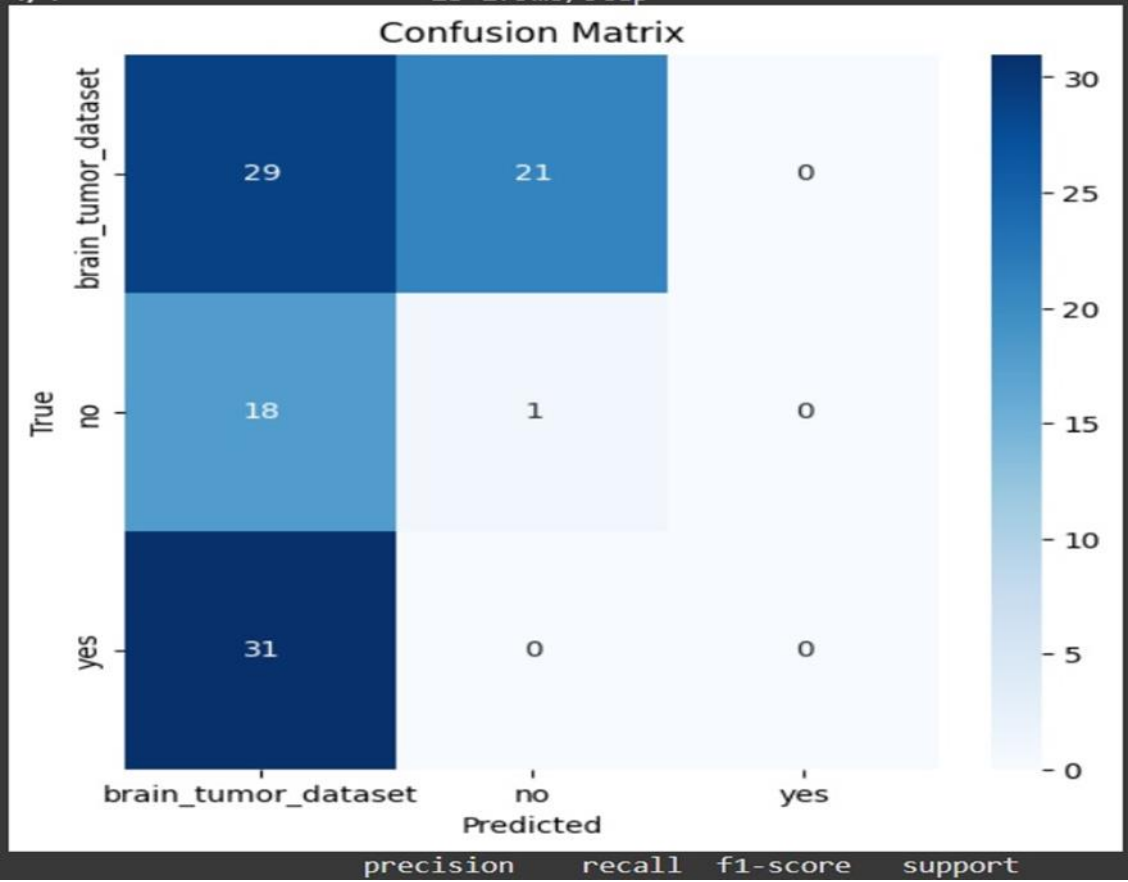
Conclusion

The **Brain Tumor Detection using CNNs** project successfully demonstrates the power of deep learning in medical imaging by automating tumor classification from MRI scans. Through meticulous data preprocessing, model optimization, and visualization techniques, the CNN model achieves significant accuracy in identifying tumors, offering potential clinical benefits for early diagnosis. While promising, further improvements such as expanding the dataset, addressing class imbalance, and integrating advanced architectures like transfer learning could enhance performance and real-world applicability. This study underscores AI's growing role in healthcare, paving the way for **more precise, scalable, and efficient diagnostic systems** that could ultimately contribute to better patient outcomes.

References

- Kaggle Brain Tumor Dataset: [Insert Dataset Link]

- Research papers on CNNs for medical imaging

- TensorFlow/Keras documentation

- External images or datasets credited accordingly