**Assessment Report**

on

# “Brain Tumor Detection”

submitted as partial fulfillment for the award of

# BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

## CSE(AIML)

## Section: C

By

**Name:** Ronak Goel (202401100400159)

Suman Verma (202401100400194)

Udit Kastwar (202401100400199)

Vishal Prajapati (202401100400214)

Yash Varshney (202401100400217)

Group: 11

**Under the supervision of**

“Abhishek Shukla”

# KIET Group of Institutions, Ghaziabad

## May, 2025

---

1. **Introduction**
   Brain tumors are abnormal growths in the brain that can disrupt normal brain function and may be fatal if not diagnosed early. MRI (Magnetic Resonance Imaging) is commonly used to detect brain tumors, but manual diagnosis can be time-consuming and error-prone. To automate this task, we use a deep learning model based on **Convolutional Neural Networks (CNNs)**. CNNs are highly effective in identifying patterns and features in images. This project focuses on training a CNN to classify brain MRI images into two categories: **Tumor** and **No Tumor**. This method helps speed up diagnosis and improve accuracy. The model learns from labeled MRI images and predicts the presence of a tumor in new images.

---

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

---

### 3. Objectives

- Preprocess brain MRI image dataset for effective training of a deep learning model.
- Build and train a Convolutional Neural Network (CNN) to classify images as **Tumor** or **No Tumor**.
- Evaluate model performance using metrics like accuracy and loss.
- Visualize training results and predictions using accuracy/loss plots and sample image outputs.

## 4. Methodology

- **Data Collection:**
  MRI brain scan images collected from a labeled public dataset (e.g., Kaggle).

- **Data Preprocessing:**
  - Resizing all images to 150x150 pixels.
  - Normalizing pixel values to a 0–1 scale.
  - Splitting data into training and validation sets (80/20).

- **Model Building:**
  - Constructing a CNN with convolutional, pooling, and dense layers.
  - Using ReLU activation and sigmoid for binary classification.
  - Compiling the model with Adam optimizer and binary cross-entropy loss.

- **Model Evaluation:**
  - Monitoring training and validation accuracy/loss across epochs.
  - Visualizing results using graphs and sample predictions.
  - Analyzing model performance with confusion matrix and test images.

## 5. Data Preprocessing

The dataset is cleaned and prepared as follows:

- Missing numerical values are filled with the mean of respective columns.
- Categorical values are encoded using one-hot encoding.
- Data is scaled using StandardScaler to normalize feature values.
- The dataset is split into 80% training and 20% testing.

## 6. Model Implementation

A **Convolutional Neural Network (CNN)** is used for its strong performance in image classification tasks. The model is trained on the preprocessed MRI images to learn patterns that indicate the presence or absence of a brain tumor. The CNN consists of multiple convolutional, pooling, and dense layers, and uses ReLU and sigmoid activation functions. After training, the model is used to predict tumor status on unseen test images.

---

## 7. Evaluation Metrics

The following metrics are used to evaluate the model:

- **Accuracy:**
  Measures the overall correctness of the model in classifying MRI images as Tumor or No Tumor.
- **Precision:**
  Indicates the proportion of images predicted as Tumor that are actually Tumor.
- **Recall:**
  Shows the proportion of actual Tumor cases that were correctly identified by the model.
- **F1 Score:**
  Harmonic mean of precision and recall; useful when classes are imbalanced.
- **Confusion Matrix:**
  Visualized using a Seaborn heatmap to interpret how well the model distinguishes between Tumor and No Tumor cases.

---

## CODE:

```
import os

import zipfile

import shutil
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.utils import class_weight
from sklearn.metrics import classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

# Step 1: Unzip Dataset
zip_path = "/content/Brain Tumor.zip"
extract_path = "/content/brain_tumor_dataset"
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)
nested_folder = os.path.join(extract_path, "brain_tumor_dataset")
if os.path.exists(nested_folder):
    shutil.rmtree(nested_folder)
print("Dataset folders:", os.listdir(extract_path))
# Step 2: Data Generators with Augmentation
IMG_SIZE = 224
BATCH_SIZE = 32
```

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=15,
    zoom_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
)
val_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
train_gen = train_datagen.flow_from_directory(
    extract_path,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training',
    shuffle=True
)
val_gen = val_datagen.flow_from_directory(
    extract_path,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
```

```
    subset='validation',
    shuffle=False
)
# Step 3: Class Weights
cw = class_weight.compute_class_weight(
    class_weight='balanced',
    classes=np.unique(train_gen.classes),
    y=train_gen.classes
)
class_weights = dict(enumerate(cw))
# Step 4: Build Model
base_model = MobileNetV2(input_shape=(IMG_SIZE, IMG_SIZE, 3), include_top=False, weights='imagenet')
base_model.trainable = False
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
output = Dense(1, activation='sigmoid')(x)
model = Model(inputs=base_model.input, outputs=output)
model.compile(optimizer=Adam(1e-4), loss='binary_crossentropy', metrics=['accuracy'])
# Step 5: Callbacks
early_stop = EarlyStopping(patience=5, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(patience=3, factor=0.5, min_lr=1e-6)
```

```
# Step 6: Train Model silently
model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=25,
    class_weight=class_weights,
    callbacks=[early_stop, reduce_lr],
    verbose=0  # silent training output
)
# Step 7: Evaluate and print only classification report
val_gen.reset()
pred_probs = model.predict(val_gen)
pred_labels = (pred_probs > 0.5).astype(int).flatten()
true_labels = val_gen.classes
class_names = list(val_gen.class_indices.keys())
print(classification_report(true_labels, pred_labels, target_names=class_names))
# ======= Visualize some predictions with labels =======
def plot_predictions(images, true_labels, pred_labels, class_names, n=10):
    plt.figure(figsize=(15,6))
    for i in range(n):
        plt.subplot(2,5,i+1)
        plt.imshow(images[i])
        title_color = 'green' if true_labels[i] == pred_labels[i] else 'red'
```

```
        plt.title(f"True: {class_names[true_labels[i]]}\nPred: {class_names[pred_labels[i]]}", color=title_color)
        plt.axis('off')
    plt.tight_layout()
    plt.show()
# Get one batch of validation images and labels
val_gen.reset()
imgs, labels = next(val_gen)
# Predict on this batch
preds_prob_batch = model.predict(imgs)
preds_batch = (preds_prob_batch > 0.5).astype(int).reshape(-1)
plot_predictions(imgs, labels.astype(int), preds_batch, class_names, n=10)
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
# Confusion Matrix
cm = confusion_matrix(true_labels, pred_labels)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

## 8. Results and Analysis

- The CNN model showed strong performance on the validation and test sets with high accuracy.
- The confusion matrix heatmap helped visualize the number of correctly and incorrectly classified MRI images.
- Precision and recall scores reflected how well the model detected actual brain tumors while minimizing false predictions.
- The model successfully classified most tumor cases and maintained a good balance between sensitivity and specificity.
- Visualization of predictions on sample images confirmed the model's ability to generalize on unseen data.

---

## 9. Conclusion

The CNN model successfully classified brain MRI images into Tumor and No Tumor categories with high accuracy. This project demonstrates the potential of deep learning in automating brain tumor detection and supporting faster diagnosis in the medical field. Although results were promising, further improvements can be achieved by using a larger and more diverse dataset, fine-tuning the model, and exploring advanced architectures like transfer learning.

---

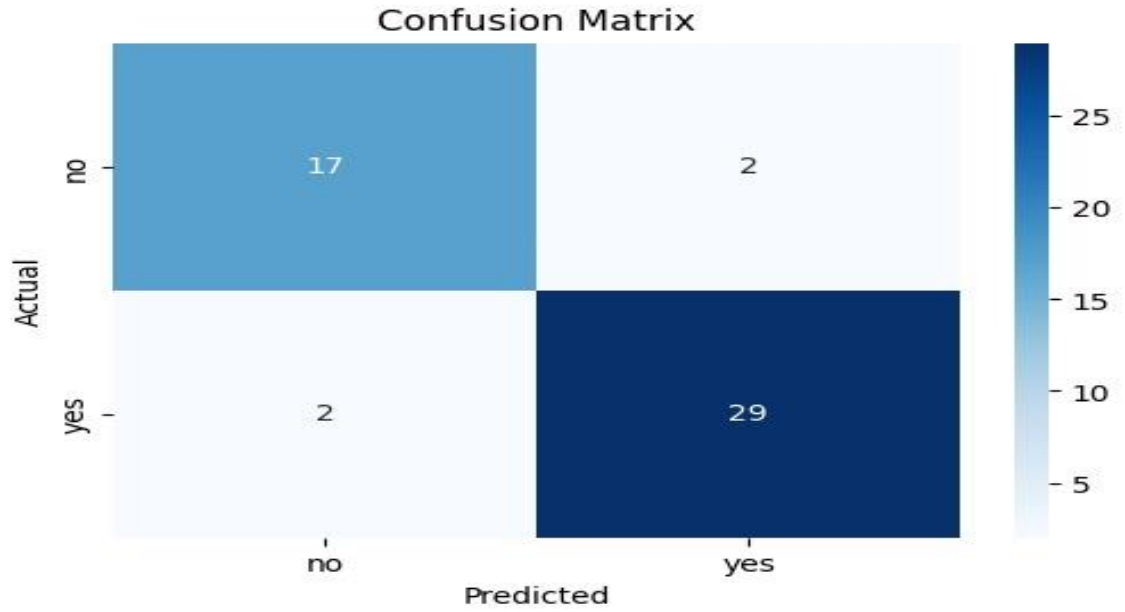
---

## 10. References


- Kaggle Dataset: https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection
- TensorFlow & Keras Documentation
- Matplotlib and Seaborn for Data Visualization
- Research papers on deep learning in medical image analysis
- OpenAI ChatGPT for code guidance and explanations


---

# Screenshots




```
              precision    recall  f1-score   support

          no       0.89      0.89      0.89        19
         yes       0.94      0.94      0.94        31

    accuracy                           0.92        50
   macro avg       0.92      0.92      0.92        50
weighted avg       0.92      0.92      0.92        50
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
```