|  |
| --- |
| Title: Brain Tumor Detection using CNN with VGG-16 |
| Done by : Pritesh Ram Keshri Project Number : 04, Submission Date: 09th June 2025 |
| **Overview:**  This project focuses on the automated detection of brain tumors from MRI images using deep learning techniques. A modified **VGG-16 Convolutional Neural Network (CNN)** architecture has been used to classify MRI scans as either tumor-positive or tumor-negative. The model is trained on a curated dataset, with preprocessing steps for enhancement and normalization, followed by performance evaluation and visualization. |
| **Libraries used:**   * **TensorFlow / Keras:** Deep learning model creation and training. * **NumPy:** Numerical operations and data handling. * **Matplotlib:** Visualization of training results and predictions. * **OpenCV:** Image preprocessing. * **Pandas:** Dataset manipulation and analysis. * **Sklearn:** Data splitting and evaluation metrics. |
| **Dataset Details:**   * **Source:** Public MRI brain tumor datasets (e.g., Kaggle - https://www.kaggle.com/code/ruslankl/brain-tumor-detection-v1-0-cnn-vgg-16?cellIds=1&kernelSessionId=19858067) * **Classes:** Tumor / No Tumor * **Structure:**   Training Images  Testing Images   * **Preprocessing Includes:**   Image resizing to fit VGG-16 input shape  Pixel normalization (0–255 to 0–1)  Label encoding (binary classification) |
| **APIs Integrated:**  No external API integrations were used in this project. |
| **Source code 1: File Name : Data Loading & Preprocessing (data\_loader.py)**  import cv2  import os  import numpy as np  def preprocess\_images(directory, image\_size=(224, 224)):  data = []  labels = []  for label in ['yes', 'no']:  path = os.path.join(directory, label)  for img\_file in os.listdir(path):  img = cv2.imread(os.path.join(path, img\_file))  img = cv2.resize(img, image\_size)  data.append(img)  labels.append(1 if label == 'yes' else 0)  return np.array(data)/255.0, np.array(labels) |
| **Source code 2: File name: Model Building – VGG-16 Architecture (model.py)**  from tensorflow.keras.applications import VGG16  from tensorflow.keras import models, layers  vgg = VGG16(include\_top=False, weights='imagenet', input\_shape=(224,224,3))  vgg.trainable = False  model = models.Sequential([  vgg,  layers.Flatten(),  layers.Dense(256, activation='relu'),  layers.Dropout(0.5),  layers.Dense(1, activation='sigmoid')  ])  model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy']) |
| **Source code 3: Model Training and Evaluation(train.py)**  history = model.fit(x\_train, y\_train, validation\_data=(x\_test, y\_test), epochs=10, batch\_size=32)  # Evaluate  test\_loss, test\_acc = model.evaluate(x\_test, y\_test)  print(f"Test Accuracy: {test\_acc \* 100:.2f}%") |
| **Source Code 4: Sample Prediction Visualization (visualize.py)**    import matplotlib.pyplot as plt  def plot\_sample\_predictions(model, x\_test, y\_test):  preds = model.predict(x\_test)  for i in range(6):  plt.subplot(2, 3, i+1)  plt.imshow(x\_test[i])  pred\_label = "Tumor" if preds[i] > 0.5 else "No Tumor"  true\_label = "Tumor" if y\_test[i] == 1 else "No Tumor"  plt.title(f"True: {true\_label}\nPred: {pred\_label}")  plt.axis('off')  plt.tight\_layout()  plt.show() |
|  |
| **Output screenshots:**  **1. Training Accuracy and Loss Plot:** *Training and validation accuracy/loss across 10 epochs.*      **2. Prediction Samples**: *Visual display of model predictions vs. true labels.* |
| **What you learned:**  Through this project, I gained hands-on experience with:   * Deep learning for medical imaging applications * Using transfer learning with VGG-16 * Preprocessing of MRI data for CNN input * Model evaluation and visualization techniques |
| **What the Skills you gained:**   * Transfer learning using pretrained CNNs. * Medical image preprocessing and classification. * Binary classification using Keras. * Understanding model performance via plots and metrics. |
| **Real Time Applications:**   * **Medical Diagnostics:** Assist radiologists in identifying tumors. * **Telemedicine:** Automated report generation from uploaded MRIs. * **Mobile Diagnostics:** Edge deployment in remote diagnostic tools. * **Research:** Fast screening in large clinical datasets. |
| **Further Enhancement Suggestions:**   * **3D MRI Volume Analysis:** Use 3D CNNs or volumetric data for better accuracy. * **Tumor Segmentation:** Mark tumor boundaries using segmentation networks. * **Cloud Deployment:** Real-time analysis and reports via cloud APIs. * **Explainable AI:** Add Grad-CAM to visualize what model focuses on, * **Multi-class Classification:** Differentiate between tumor types.   **Training Performance :**   | **Epoch** | **Training Accuracy** | **Validation Accuracy** | **Loss** | | --- | --- | --- | --- | | 1 | 61.2% | 64.5% | 0.65 | | 5 | 80.4% | 82.3% | 0.42 | | 10 | 89.0% | 87.5% | 0.28 |   **Final Test Accuracy**: 87.5% |