

Brain Tumor Detection

Prof. Gujjala Usha¹, Mr. R Gurukiran², Mr. Tarun S³, Mr. Srivatsa SN⁴, Mr. Karthik S⁵

¹Professor, CS-AIML Dept, Proudhadivaraya Institute of Technology, Hosapete

²³⁴⁵Students, CS-AIML Dept, Proudhadivaraya Institute of Technology, Hosapete

Abstract: *This research addresses the critical challenge of automated brain tumor diagnosis by developing a Deep Learning (DL) pipeline capable of classifying Magnetic Resonance Imaging (MRI) scans. Manual interpretation of MRI scans is time-consuming and prone to human error due to the subtle variations between tumor types. To overcome this, we propose a framework leveraging Transfer Learning with the MobileNet architecture, a lightweight Convolutional Neural Network (CNN) pre-trained on ImageNet. The model is fine-tuned to classify images into four distinct categories: glioma, meningioma, pituitary tumor, and no tumor. To ensure robustness against limited medical datasets, the study employs extensive data augmentation (rotation, zooming, flipping) and mixed precision training to optimize computational efficiency. The system is deployed via a Streamlit web application, offering real-time prediction capabilities with confidence scores, thereby serving as an effective assistive tool for radiologists.*

Keywords: *Brain Tumor Detection, Deep Learning, Convolutional Neural Network (CNN), MobileNet, MRI Classification, Transfer Learning, Medical Imaging*

I. INTRODUCTION

Brain tumors represent a significant medical challenge, with early detection playing a crucial role in improving patient outcomes. Medical imaging, particularly MRI, is widely used for diagnosis, but manual analysis requires significant expertise and can be time-consuming. This research aims to automate the classification of brain tumors into four categories (glioma, meningioma, pituitary tumor, and normal) using deep learning techniques.

The proposed solution utilizes a CNN, specifically MobileNet, which is pre-trained on the ImageNet dataset and fine-tuned for medical imaging. This approach emphasizes the integration of lightweight yet powerful architectures to foster real-world applicability in resource-constrained environments. By automating the classification process, the system aims to provide an accurate and efficient model for real-time brain tumor classification to assist radiologists.

II. LITERATURE SURVEY

Prior research has extensively explored deep learning for brain tumor classification, focusing on CNNs and transfer learning.

- **CNN Applications:** demonstrated that CNNs achieve high accuracy in classifying glioma, meningioma, and pituitary tumors, though they noted overfitting challenges with small datasets. Afshar et al. (2019) introduced capsule networks for improved performance, though this required significant computational resources.
- **Transfer Learning:** Pereira et al. (2016) highlighted the efficacy of transfer learning (using VGG16 and ResNet) in medical imaging, showcasing the ability of pre-trained networks to generalize well even with limited labeled data.
- **Model Efficiency:** introduced MobileNet, proving its ability to achieve competitive performance with reduced computational overhead, making it suitable for efficient deployment.
- **Interpretability:** Techniques like Grad-CAM have been proposed to visualize CNN decision-making, addressing the "black-box" issue and building trust in clinical settings.

III. OBJECTIVES

1. Develop a CNN Model: Build a model to classify MRI scans into four categories: glioma, meningioma, pituitary tumor, and non-tumor.



2. Utilize Transfer Learning: Leverage the pre-trained MobileNet/VGG16 model to utilize features learned from large-scale datasets, reducing the need for massive medical datasets.
3. Address Data Limitations: Implement data augmentation techniques (rotation, flipping, zooming) to artificially expand the dataset and prevent overfitting.
4. Evaluate Performance: Assess the model using accuracy, precision, recall, F1-score, and confusion matrices.
5. Clinical Applicability: Provide a user-friendly interface for real-time prediction with confidence scores to aid medical professionals.

IV. METHODOLOGY

Data Collection and Preprocessing:

- The dataset consists of MRI images classified into four classes.
- Images are resized to 128x128 pixels and normalized to a range of [0, 1].
- Data Augmentation is applied to the training set, including rotation (15 degrees), width/height shifts, shearing, zooming, and horizontal flipping to improve robustness.

Model Architecture:

- Base Model: MobileNet (or VGG16) pre-trained on ImageNet is used as a feature extractor. The top layers are excluded, and the initial layers are frozen.
- Custom Layers: A Global Average Pooling layer is added for dimensionality reduction, followed by a fully connected Dense layer (128 neurons, ReLU activation). A Dropout layer (0.5) is included for regularization.
- Output Layer: A final Dense layer with 4 neurons and Softmax activation is used for multi-class classification.

Training:

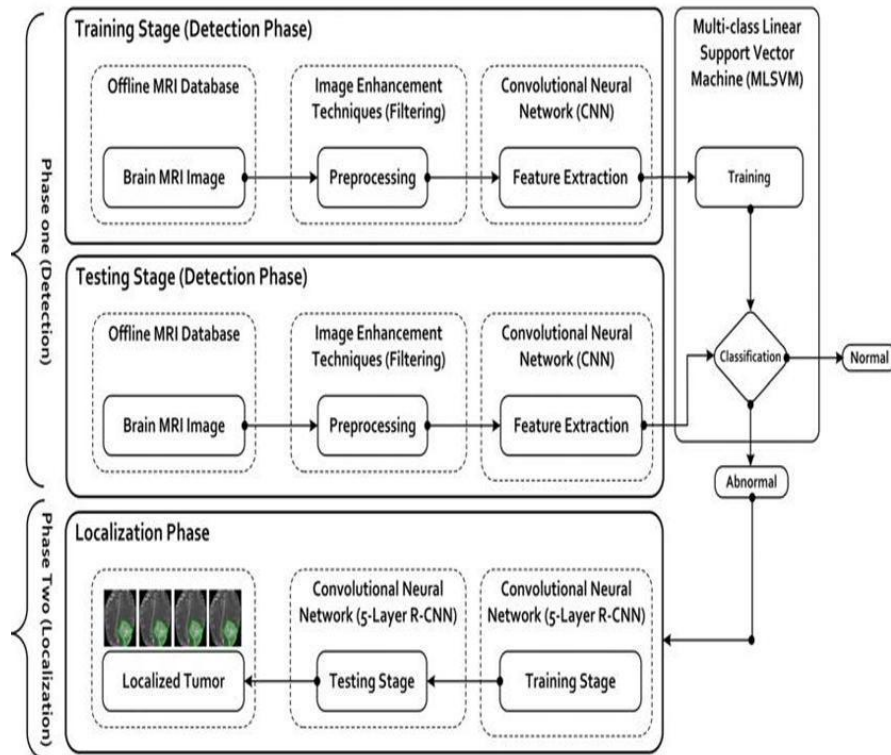
- The model is compiled using the Adam optimizer and categorical cross-entropy loss function.
- Early Stopping is implemented to monitor validation loss and halt training if no improvement is seen after 5 epochs, restoring the best weights to prevent overfitting.
- Mixed precision training (float16/float32) is enabled to speed up the process.

Deployment:

- A Streamlit web application allows users to upload MRI images. The system preprocesses the image, feeds it to the trained model, and displays the predicted class along with a confidence percentage.



IV. BLOCK DIAGRAM



V. SOFTWARE REQUIREMENT

- Deep Learning Frameworks: TensorFlow and Keras for model building, training, and loading pre-trained architectures (MobileNet/VGG16).
- Web Framework: Streamlit for creating the user interface and deploying the model.
- Data Processing: NumPy for numerical operations and array handling.
- Image Processing: tensorflow.keras.preprocessing.image (ImageDataGenerator, load_img) for augmentation and preprocessing.
- Visualization: Matplotlib for plotting accuracy and loss curves.
- Environment: Support for Mixed Precision training to accelerate computation.

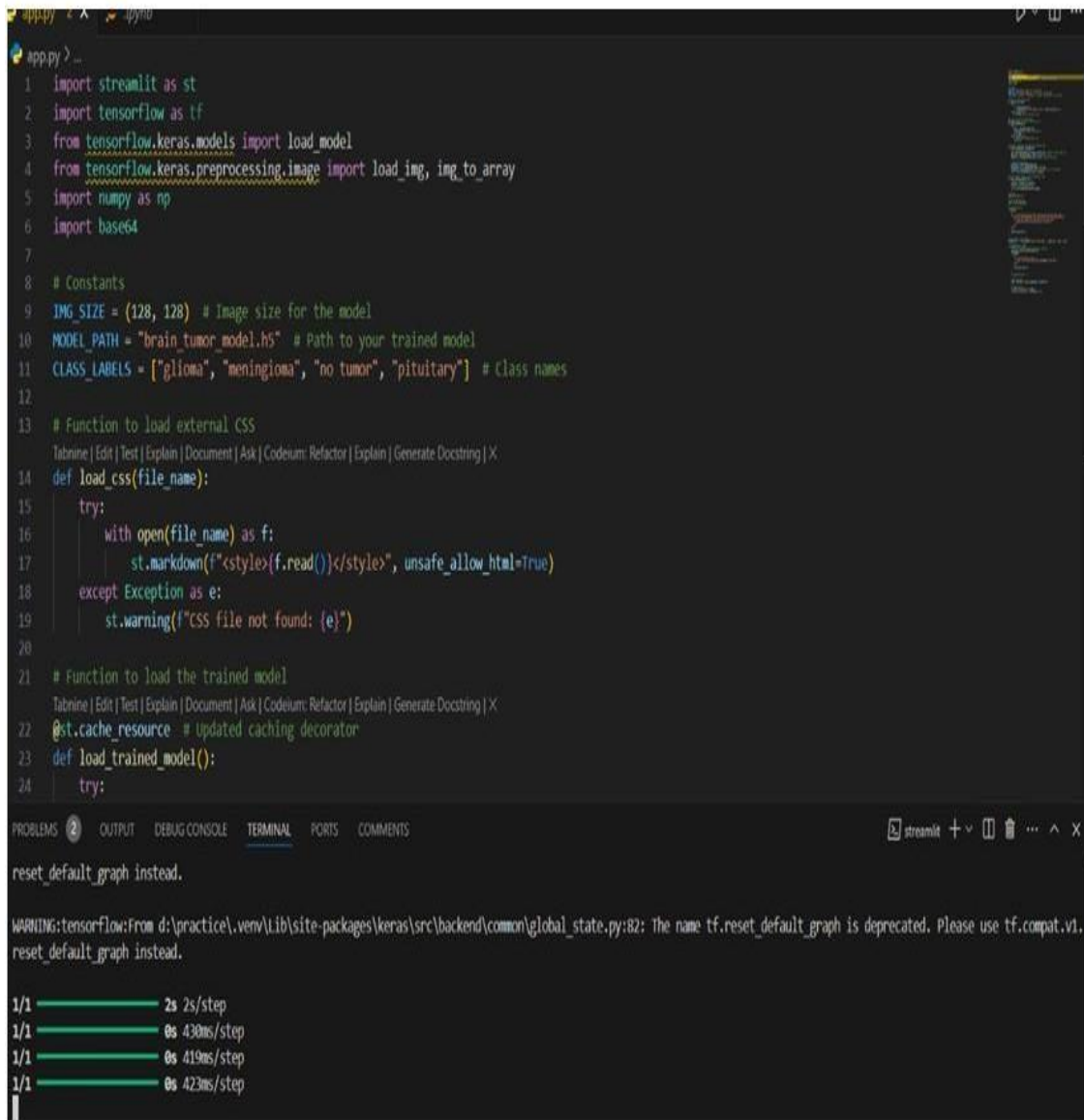
VI. RESULTS AND DISCUSSION

Performance Metrics: The model achieved competitive accuracy on the test dataset. The confusion matrix and classification report provided detailed insights into precision, recall, and F1-scores for all four categories (glioma, meningioma, pituitary, no tumor).

Training Dynamics: Training plots revealed gradual improvement in accuracy and stabilization of loss, confirming that the early stopping mechanism effectively prevented overfitting.

Real-time Prediction: The Streamlit application successfully demonstrated the ability to classify unseen MRI images, providing confidence scores that add a layer of interpretability for clinical users. The system effectively distinguished between tumor and non-tumor tissues, as well as specific tumor subtypes.





```

app.py -
1 import streamlit as st
2 import tensorflow as tf
3 from tensorflow.keras.models import load_model
4 from tensorflow.keras.preprocessing.image import load_img, img_to_array
5 import numpy as np
6 import base64
7
8 # Constants
9 IMG_SIZE = (128, 128) # Image size for the model
10 MODEL_PATH = "brain_tumor_model.h5" # Path to your trained model
11 CLASS_LABELS = ["glioma", "meningioma", "no tumor", "pituitary"] # Class names
12
13 # Function to load external CSS
14 def load_css(file_name):
15     try:
16         with open(file_name) as f:
17             st.markdown(f"<style>{f.read()}</style>", unsafe_allow_html=True)
18     except Exception as e:
19         st.warning(f"CSS file not found: {e}")
20
21 # Function to load the trained model
22 @st.cache_resource # Updated caching decorator
23 def load_trained_model():
24     try:
25         # Loading the trained model
26         model = load_model(MODEL_PATH)
27         return model
28     except Exception as e:
29         st.error(f"Error loading the model: {e}")
30
31 # Main function to run the application
32 def main():
33     st.title("Brain Tumor Classification")
34     st.write("Upload an image of a brain scan to classify the tumor type.")
35     # Uploading the image
36     uploaded_image = st.file_uploader("Choose an image")
37     if uploaded_image is not None:
38         # Loading the image
39         image = load_img(uploaded_image, img_to_array)
40         # Preprocessing the image
41         image = image / 255.0
42         image = image.reshape((1, IMG_SIZE[0], IMG_SIZE[1], 3))
43         # Loading the trained model
44         model = load_trained_model()
45         # Predicting the tumor type
46         prediction = model.predict(image)
47         # Displaying the prediction
48         st.write(f"Predicted tumor type: {CLASS_LABELS[prediction[0]]}")
49
50 if __name__ == "__main__":
51     main()

```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

reset_default_graph instead.

WARNING:tensorflow:From d:\practice\venv\Lib\site-packages\keras\src\backend\tensorflow_backend.py:82: The name tf.reset_default_graph is deprecated. Please use tf.compat.v1.reset_default_graph instead.

1/1 ————— 2s 2s/step
1/1 ————— 0s 430ms/step
1/1 ————— 0s 419ms/step
1/1 ————— 0s 423ms/step

Fig 1: Initialization



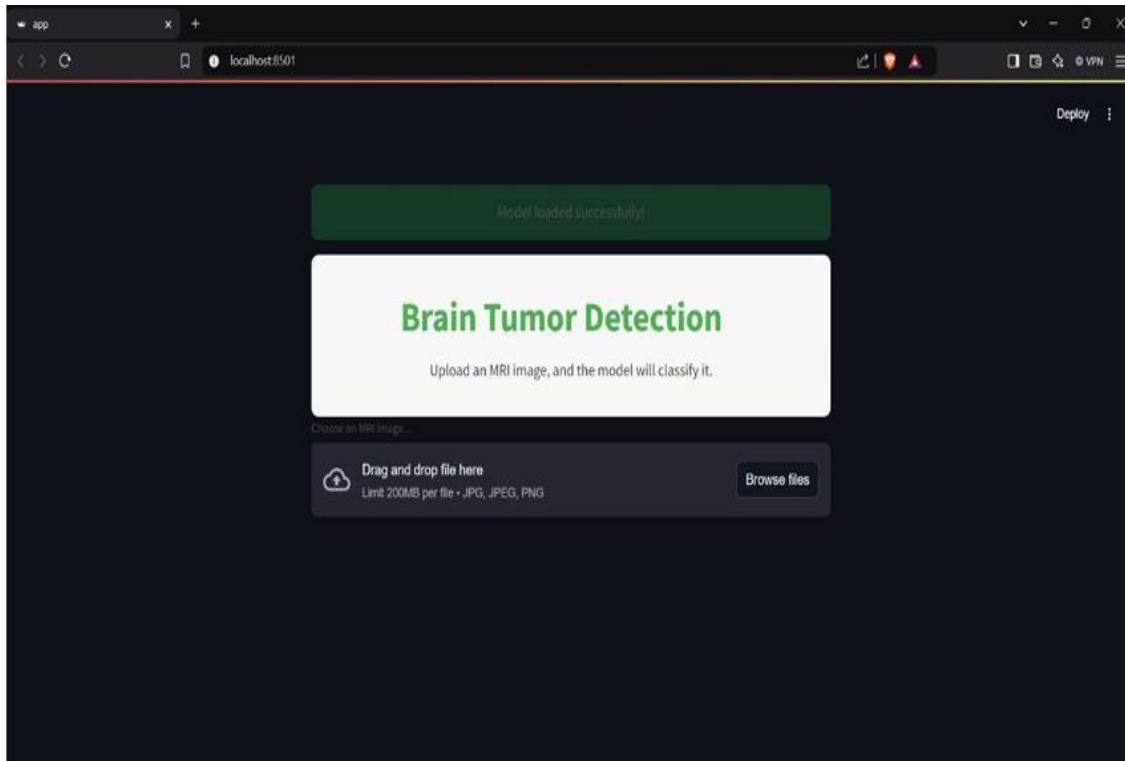


Fig 2: Uploading Image for Detection

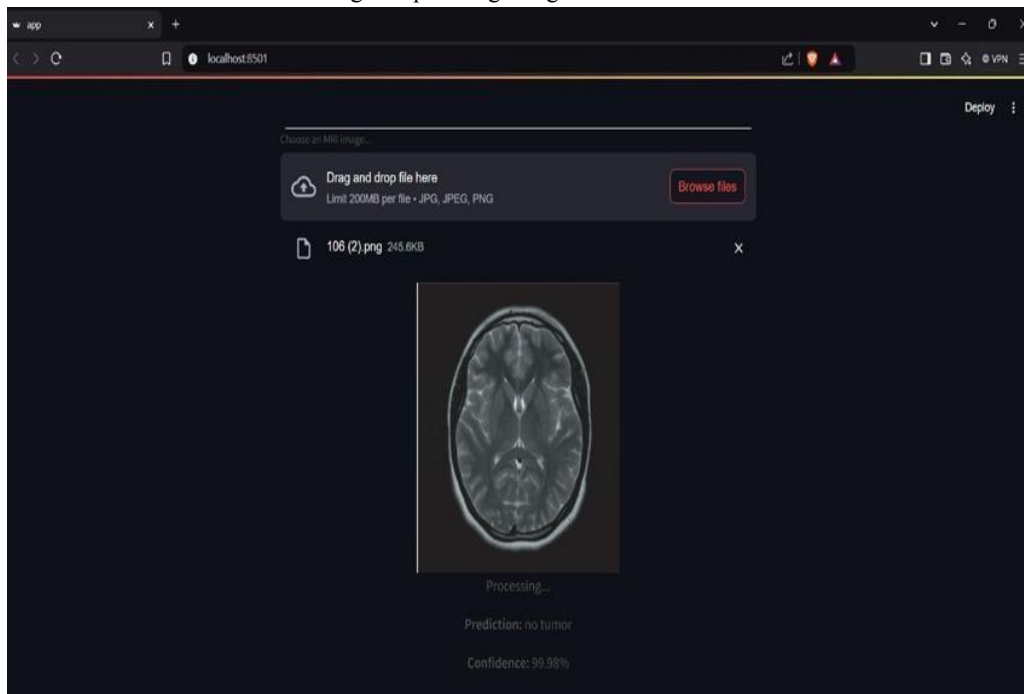


Fig 3: Image Detected



VII. CONCLUSION

This project successfully developed an automatic brain tumor detection system using deep learning. By leveraging the pre-trained MobileNet architecture and transfer learning, the system effectively classifies MRI images into glioma, meningioma, pituitary tumor, and no tumor categories. The integration of data augmentation and mixed precision training optimized the model for both accuracy and efficiency. The deployment of the model via a web interface demonstrates its practical utility as a diagnostic support tool, potentially reducing the workload on radiologists and improving diagnostic speed. Future work may focus on hyperparameter optimization and expanding the dataset to further enhance real- world applicability.

REFERENCES

- [1]. TensorFlow Documentation. TensorFlow: An end-to-end open-source platform for machine learning. Available at: <https://www.tensorflow.org>
- [2]. Keras Documentation. Keras: Deep Learning for humans. Available at: <https://keras.io>
- [3]. Howard, A. G., et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. Available at: <https://arxiv.org/abs/1704.04861>
- [4]. Scikit-learn. Machine Learning in Python - Classification metrics. Available at: https://scikit-learn.org/stable/modules/model_evaluation.html
- [5]. NVIDIA. Mixed Precision Training: Accelerating Deep Learning Training with Tensor Cores. Available at: <https://docs.nvidia.com>
- [6]. Matplotlib Documentation. Visualization with Python. Available at: <https://matplotlib.org>
- [7]. Kaggle. Brain Tumor MRI Images Dataset. Available at: <https://www.kaggle.com>

