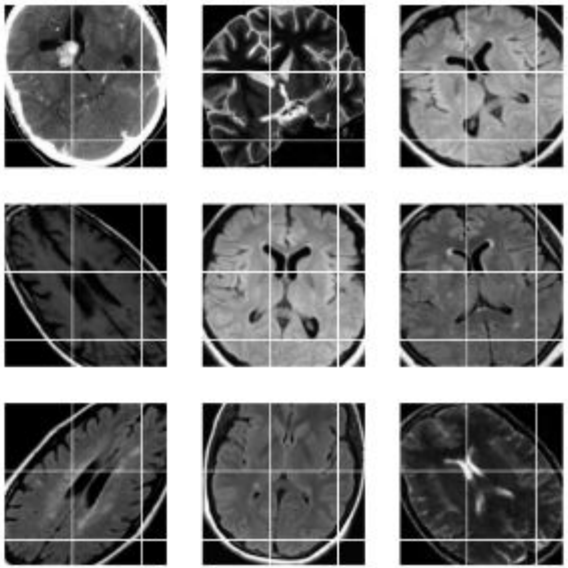
Prediction of Presence of Brain Tumor from MRI Scans

Brain Disease Prediction using Machine Learning



18 April 2024, MSU edX AI Boot Camp, Project 3

Health Data Detectives
Betsy Deuman Jasmine Harper Dr. Chadi Saad Aaron Wood

Executive Summary

Project Description

- We aimed to develop a model that can accurately detect brain tumors in MRI scans. With this technology speed and efficiency can be gained from time from imaging to diagnosis and can help with enhanced diagnostic accuracy in early-stage tumors. Increase in healthcare accessibility, if there is a shortage of trained radiologists is limited, ML can do the initial screening and diagnostic to catch brain tumors faster than if patients had to wait for radiologists were required to be on site.

- Classification algorithms can be employed to improve people's understanding about the impact of their individual risk exposure on brain health.

- Dataset Source: https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset

Project Goals

1. This project aims to train machine learning models on MRI scan of brains with a tumor and without, with the end goal to have a model that is capable of ingesting an MRI and determining if that MRI scan has a tumor.

2. Can we train a model to accurately (>.90) detect MRI scans with a brain tumor

3. Create clear documentation for others to do their own testing and validation and contribute to model accuracy.

4. Stretch Goal: Create a chatbot that will live on the MRI upload page that can intake additional information from the user.

Approach

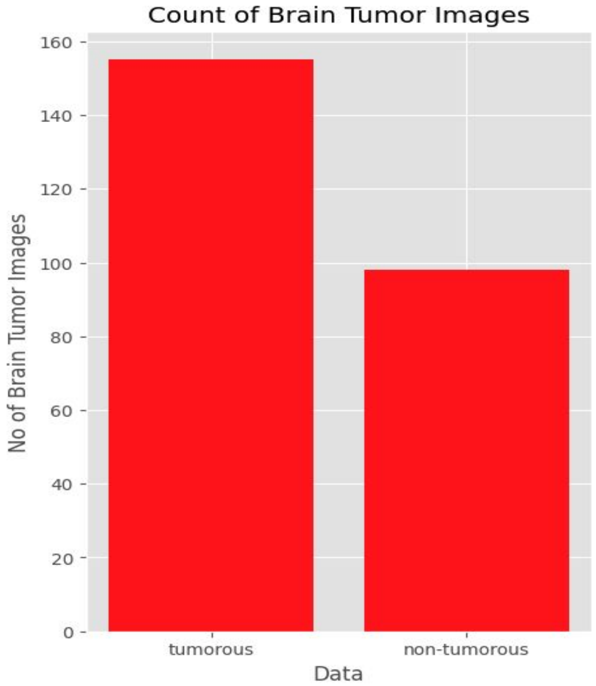
1. Finding the dataset.
2. Exploratory analysis and data cleanup.
3. Data augmentation and pre- processing.
4. Balance the data.
5. Build and train the models.
6. Create custom CNN model.

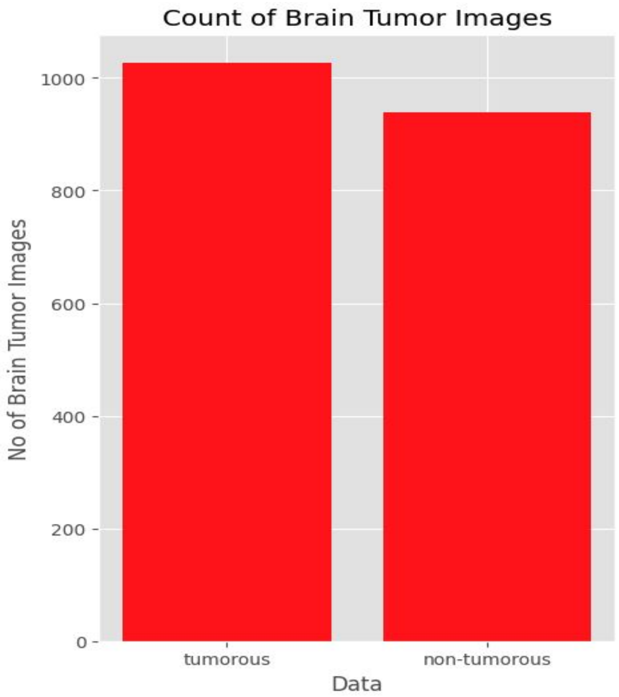
Libraries Used: Libraries: matplotlib, pandas, numpy, cv2, os, seaborn, tensorflow, imutils, sklearn, shutil, gradio, flask

```python
from tensorflow.keras.applications import VGG19
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

# Initialize the base VGG19 model
base_model = VGG19(weights='imagenet', include_top=False, input_shape=(150, 150, 3))

# Freeze the convolutional base to prevent weights from being updated during training
for layer in base_model.layers:
    layer.trainable = False

# Add custom layers on top of the VGG19 base
model = models.Sequential([
    base_model,
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')  # Sigmoid activation for binary classification
])

# Compile the model
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.Adam(lr=1e-4),  # Lower learning rate to prevent overfitting
              metrics=['accuracy'])

# Data augmentation for the training data
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Note that validation data should not be augmented
validation_datagen = ImageDataGenerator(rescale=1./255)

# Train and validation generators
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),
    batch_size=20,
    class_mode='binary'
)

validation_generator = validation_datagen.flow_from_directory(
    validation_dir,
    target_size=(150, 150),
    batch_size=20,
    class_mode='binary'
)

# Callbacks: EarlyStopping and ModelCheckpoint
# ***************************************************************************
# AW COMMENT: learned ModelCheckpoint(filep.... line saves model to directory
# where code is running.  To save there...use /content/file_name
# ***************************************************************************
callbacks = [
    EarlyStopping(monitor='val_loss', patience=5, verbose=1),
    ModelCheckpoint(filepath='/content/vgg19_best_model.h5', monitor='val_accuracy', save_best_only=True, verbose
]
# Correct the optimizer parameter
optimizer = optimizers.Adam(learning_rate=1e-4)
```

Data Collection & Exploration

- Found the data on Kaggle
- Exploration realized there wasn't enough data and it was unbalanced
- Increase the number of images and balance of "yes" and "no" tumor images





Data Augmentation & Balance

pseducode_augmented_data

- **Create balance in augmented samples** for each class. (Current state = imbalance.)
- **Establish capability to adjust image sizes and augmentation parameters** based on specific requirements of the neural network architecture.
- Monitor and tune data augmentation to **ensure image quality and utility** in improving model performance.
- **Leverage Tensorflow and Keras** libraries for building neural network models.

augmented_data function

Input parameters

Directory of images, # of generated samples per image read, directory to save augmented images.

Processing steps

1. Iterates over each image.
2. Resizes images to 240 x 240. Ensures consistency in input size.
3. Converts images to numpy arrays and reshapes them for the data generator.
4. Generates augmented images using datagen.flow(), saves them to target dir.
5. Stops once the desired number of samples has been reached.

Preprocessing - improving quality

Goal: Improve dataset quality with OpenCV image processing

Identify and crop regions w/ <u>tumors</u>

- **OpenCV** - image processing
- **imutils** - utility functions
  - automated "load + apply preprocessing + prepare arrays"
  - visualization to confirm preprocessing
- **Crop_brain_tumor function**

Convert to grayscale - reduce complexity

Gaussian Blurring - detect distinct features

Thresholding - binary gray (black / white)

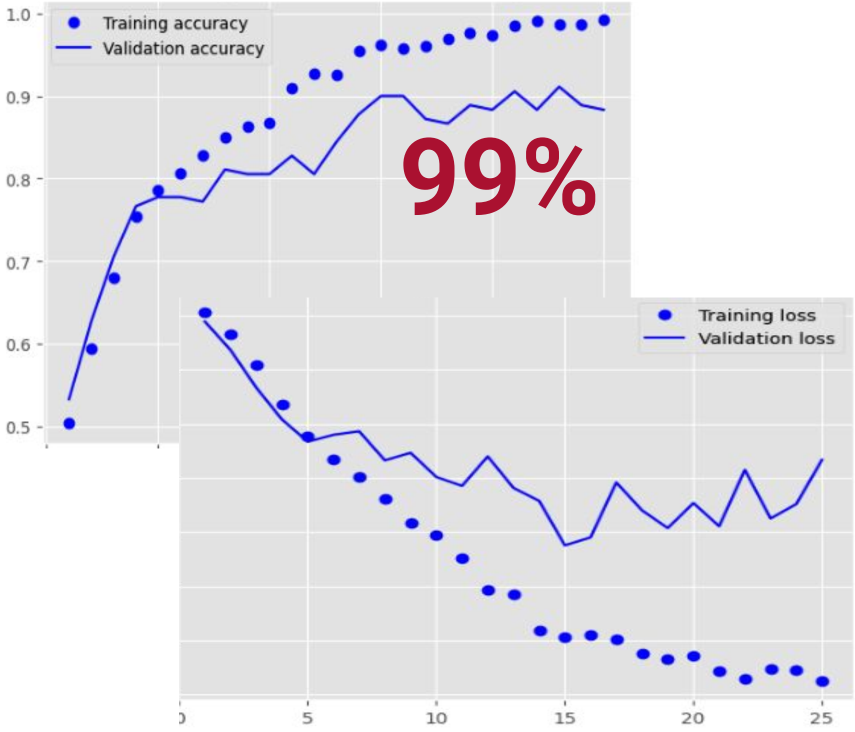
Erosion and Dilation - enhanced separation

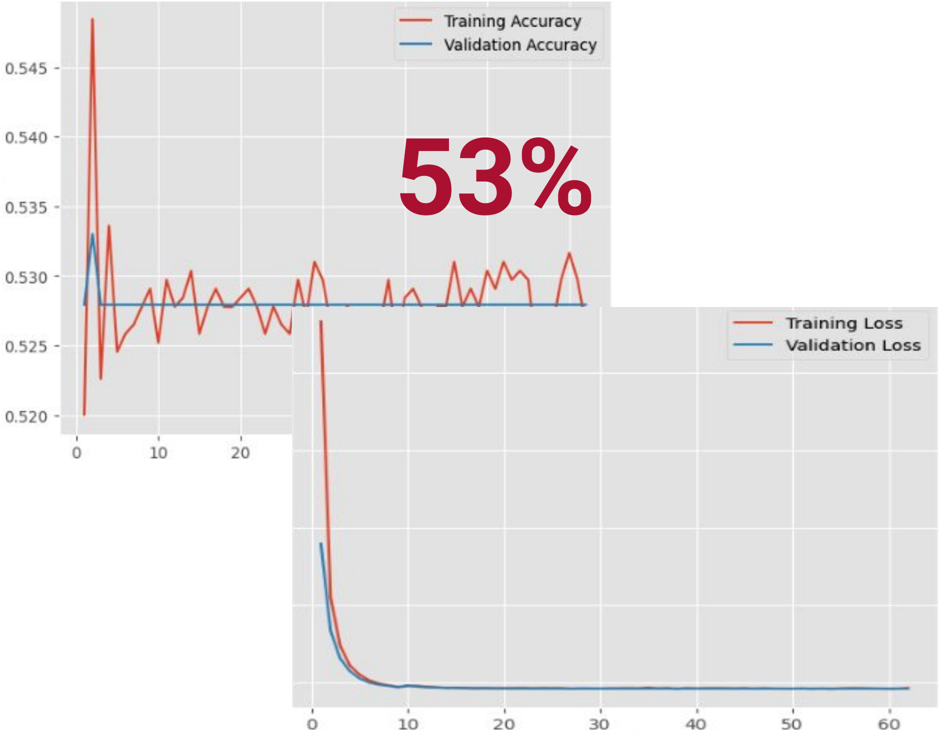
Contour Detection - tumor boundaries

Cropping - causes tumor to be largest portion

Focused on improving contour / edge / feature detection.

Model 1 & 2





Overfitted.

Not overfitted. Poor Performance.

Models and Performance

	1	2	3	4
Inputs	Augmented Pre-Processed Yes / No Brain MRI Scan image data			
Type	Sequential CNN	Parameter-tuned Sequential CNN	**Transfer Learn MobileNet V2**	Transfer Learn VGG19
Epochs	100	100	**100**	100
Early Stop	25 - 30 epochs	50 - 55	**90**	75 - 80
Loss	2%	69%	**8%**	26%
Accuracy	99%	53%	**97%**	91%
Capability	Overfitted	Not usable	**Usable**	Usable

Model Optimization

- In our project, we employed transfer learning with pretrained models such as MobileNetV2 and VGG19 . This approach significantly improved our model's accuracy and loss metrics, enabling superior performance in real-time image processing essential for deployment in mobile and edge computing environments

MobileNetV2 - Optimized for Mobile and Edge Computing

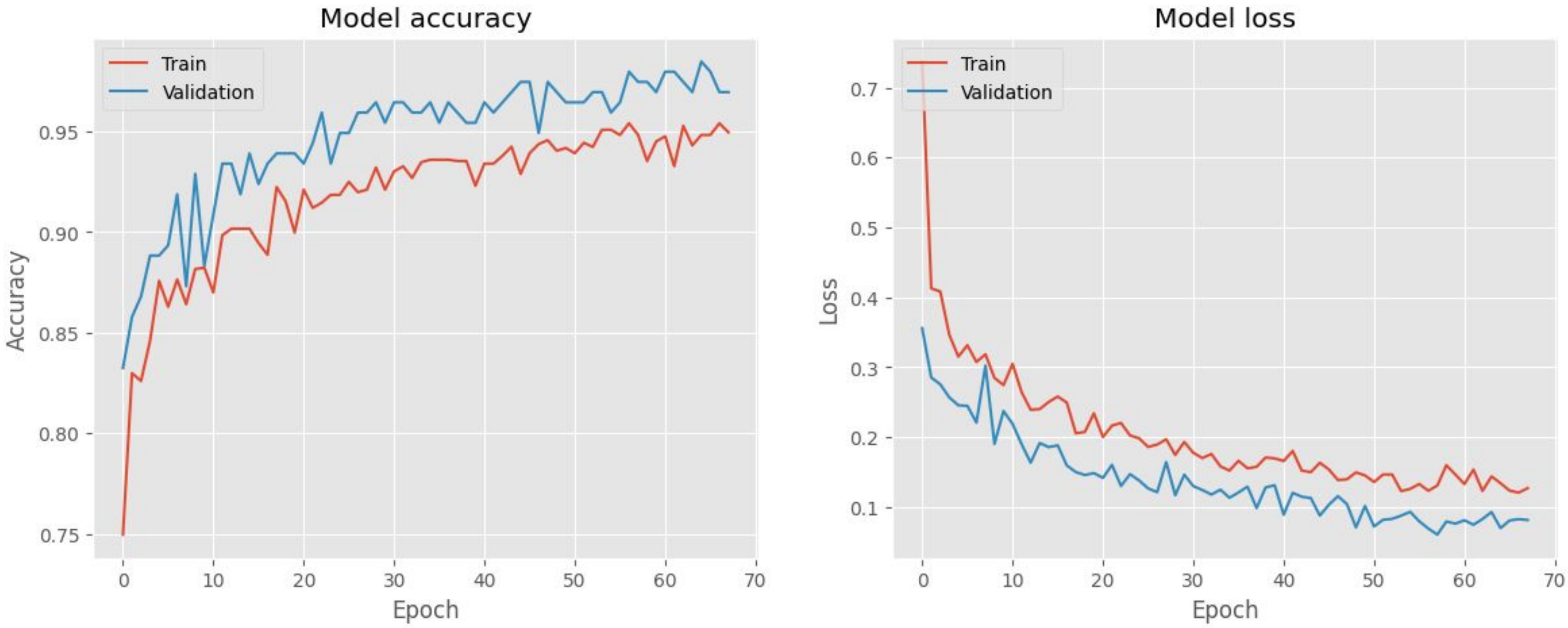
- Developed By: Google researchers, 2018.
- Purpose: Designed to maximize efficiency in mobile and edge devices, balancing performance and computational constraints.
- Applications: Widely used in mobile applications requiring real-time processing, such as object detection, face recognition, and augmented reality

VGG19 - Deep Learning for Image Recognition

- Developed By: Visual Graphics Group, University of Oxford, 2014.
- Purpose: To explore how network depth affects performance in large-scale image recognition tasks.
- Impact: Known for its robustness and simplicity, serving as a foundational model for numerous computer vision tasks and studies in deep learning.

Model Optimization

MobileNetV2 - Optimized for Mobile and Edge Computing



Model Optimization

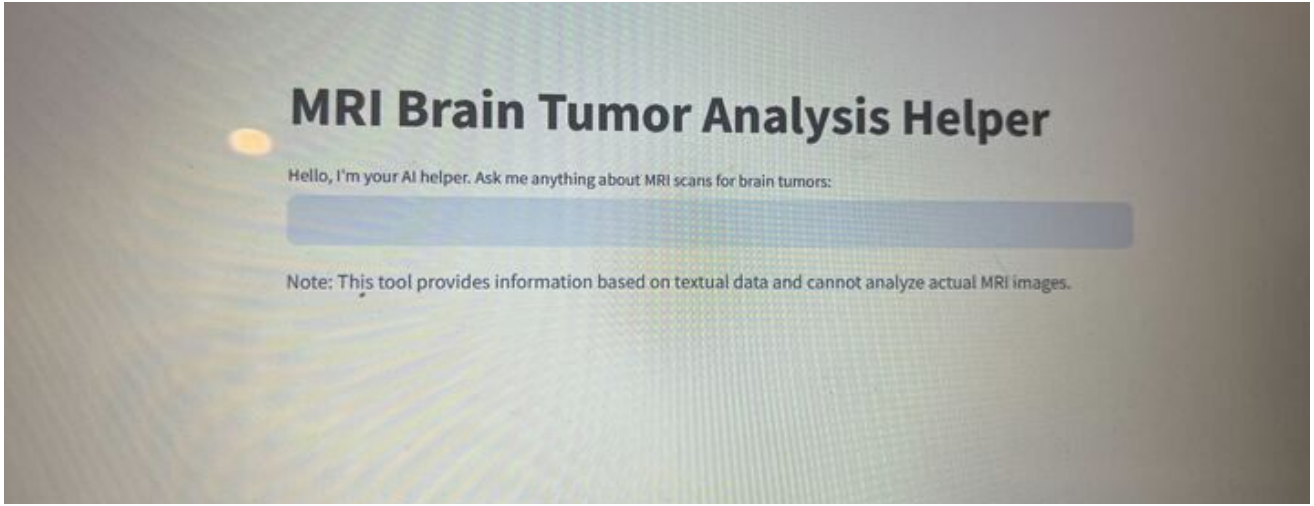
VGG19 - Deep Learning for Image Recognition

Results & Conclusion

Results:

- Chat bot that can take a photo and review it and explain what you are seeing, by being integrated with ChatGPT. App using streamlit.
- Can answer questions by users about brain tumors
- Robust image processing pipeline, key to enabling model automation and ultimately the chatbot
- Use of streamlit for a clean user friendly interface
- Accuracy improvement from 50 to about 97 percent
- Appropriate use of models for accuracy improvements

- Future Considerations: this Advance machine learning technique could make the reading of radiology images and diagnosis more automated and robust also resulting in shortening the time to make a diagnosis
- Successful processing and augmentation of the data
- Proper application and of models
- Accuracy improvement


MRI Brain Tumor Analysis Helper
Hello, I'm your AI helper. Ask me anything about MRI scans for brain tumors:
Note: This tool provides information based on textual data and cannot analyze actual MRI images.

Questions?