Brain Tumor Detection System

#4 View of Progress as of 4/3/2024

Abdullah Alshamrani
Saint Joseph’s University

Last view 3/18/2024

In last view of progress, the following was accomplished.

- Implementation of traditional machine learning models
- Implementation of Convolutional Neural Network for Deep Learning
- Evaluation of the performance to ML Models and CNN
- Select most model accuracy
- Based on first model accuracy I came up with, Prediction for a medical image was done for the first time.

Promised tasks to be completed since last presentation view

- Enhance model accuracy.

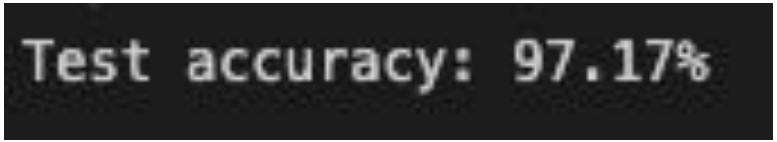
- Start The development of web interface

Enhancement of Model Accuracy

```
Test accuracy: 90.17%
```

I have stated that accuracy was not the satisfiction result for me, so an improvement was needed to get it higher than last test accuracy which was 90.17%.

With many experiments, and search. A higher test accuracy was achieved which is 97.17%. And I decided to keep going with this accuracy. More development could be done, but it would consume more time. Test accuracy of 97.17% is great to move to next tasks with satisfaction.



Code for enhancement of accuracy: Data Augmentation

To enhance model accuracy, and model robustness. Augmentation techniques were used including, rotation, zoom, height and width shift. Images were rotating by 15 degrees(relevant because medical images can vary in practice). Zooming in and out helps the model to learn to identify futures at different scales. These augmentation techniques made our model more adaptable to the variability of medical images.

```
# Data Augmentation
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Code for enhancement of accuracy: CNN Model Architecture

To capture complex patterns in the medical images, an adjustment to the CNN architecture were needed. ***Depth of the network increased*** by adding another convolutional layer, enabling the model to learn more futures. By going to a 4 convolutional layers this will help the model more to identify tumors. Also I enhanced the model's stability and allowed for higher learning rates introducing batch normalization after each convolutional layer to **normalize the inputs to activation functions**. Drop rate changed from 0.5 to 0.3 to aim for a balance between preventing overfitting retaining the model's capacity to learn complex patterns

```
# CNN architecture
def create_model():
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 1)), #32 CNN
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'), #64 CNN
        BatchNormalization(),
        MaxPooling2D((2, 2)),
        Conv2D(128, (3, 3), activation='relu'), #128 CNN
        MaxPooling2D((2, 2)),
        Conv2D(256, (3, 3), activation='relu', padding='same'), #265 CNN
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.3),
        Dense(1, activation='sigmoid')
    ])
```

Model Architecture

```
# CNN architecture
def create_model():
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 1)), #32 CNN
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'), #64 CNN
        BatchNormalization(),
        MaxPooling2D((2, 2)),
        Conv2D(128, (3, 3), activation='relu'), #128 CNN
        MaxPooling2D((2, 2)),
        Conv2D(256, (3, 3), activation='relu', padding='same'), #265 CNN
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.3),
        Dense(1, activation='sigmoid')
    ])
```

Code for enhancement of accuracy: Learning rate and callbacks

Key to the model's enhanced accuracy were the adjustments to the learning rate and the integration of callbacks. By employing ‘ReduceLROnPlateau callback’, we reduced the learning rate upon detecting plateaus in validation accuracy, fine-tuning the model's weights more effectively. Furthermore, the implementation of EarlyStopping prevented overfitting by stopping training when the validation loss ceased to improve.

callbacks give a way to monitor the model's performance and state during training and take action to either enhance learning or prevent overfitting.

```
# Callbacks
lr_reduction = ReduceLROnPlateau(monitor='val_accuracy', patience=2, verbose=1, factor=0.5, min_lr=0.00001)
early_stopping = EarlyStopping(monitor='val_loss', patience=5)
model_checkpoint = ModelCheckpoint('best_model.h5', monitor='val_accuracy', save_best_only=True)
```

Code for enhancement of accuracy: Reproducibility Measures:

Reproducibility will help to achieve very close if not the same result at a later time when we run the code one more time or several times

Ensuring reliable and repeatable outcomes, strict reproducibility measures were implemented, notably by setting consistent random seeds for NumPy and TensorFlow. This approach guaranteed **identical weight initialization** and data shuffling across all runs, thus eliminating experiment variability.

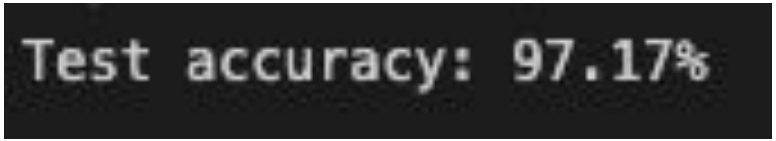
The model and random seeds

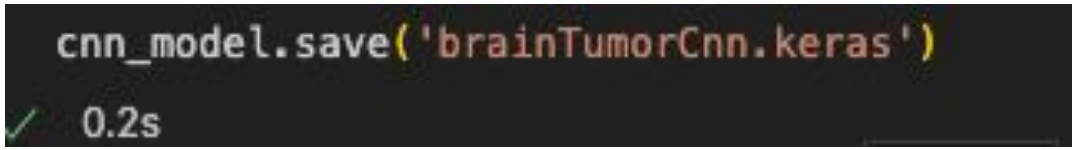
Even though we are using test/train to split data - augment, dropout data, etc. we want them to produce same outcome each time we run the code, Random seeds will help us to achieve to the same result. And setting different seed values for Numpy and TensorFlow will help each manage their own random number generation processes independently.

```
# Setting random seeds for reproducibility
np.random.seed(1)
tf.random.set_seed(2)
```

Model is saved

Model is saved to move to the next task





Flask, HTML & CSS

Web Interface using Flask, HTML & CSS

```
from flask import Flask, request, jsonify
from tensorflow.keras.models import load_model
import cv2
import numpy as np
import os

app = Flask(__name__)
MODEL_PATH = 'brainTumorCnn.keras'
model = load_model(MODEL_PATH)

@app.route('/', methods=['GET'])
```

Architecture of flask

```
<!DOCTYPE html>
<html>
<body>
    <form method="post" action="/predict" enctype="multipart/form-data">
        <input type="file" name="file" accept="image/*" required>
        <input type="submit" value="Upload and Predict">
    </form>
</body>
<style>
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background: #f3f3f3;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}


form {
    background: #fff;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}


input[type=file] {
    margin-bottom: 10px;
}


input[type=submit] {
    cursor: pointer;
    background: #007bff;
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
}
```

```
MODEL_PATH = 'brainTumorCnn.keras'
model = load_model(MODEL_PATH)
```

HTML & CSS

Specifying the path to the model and loading the model using tensorflow

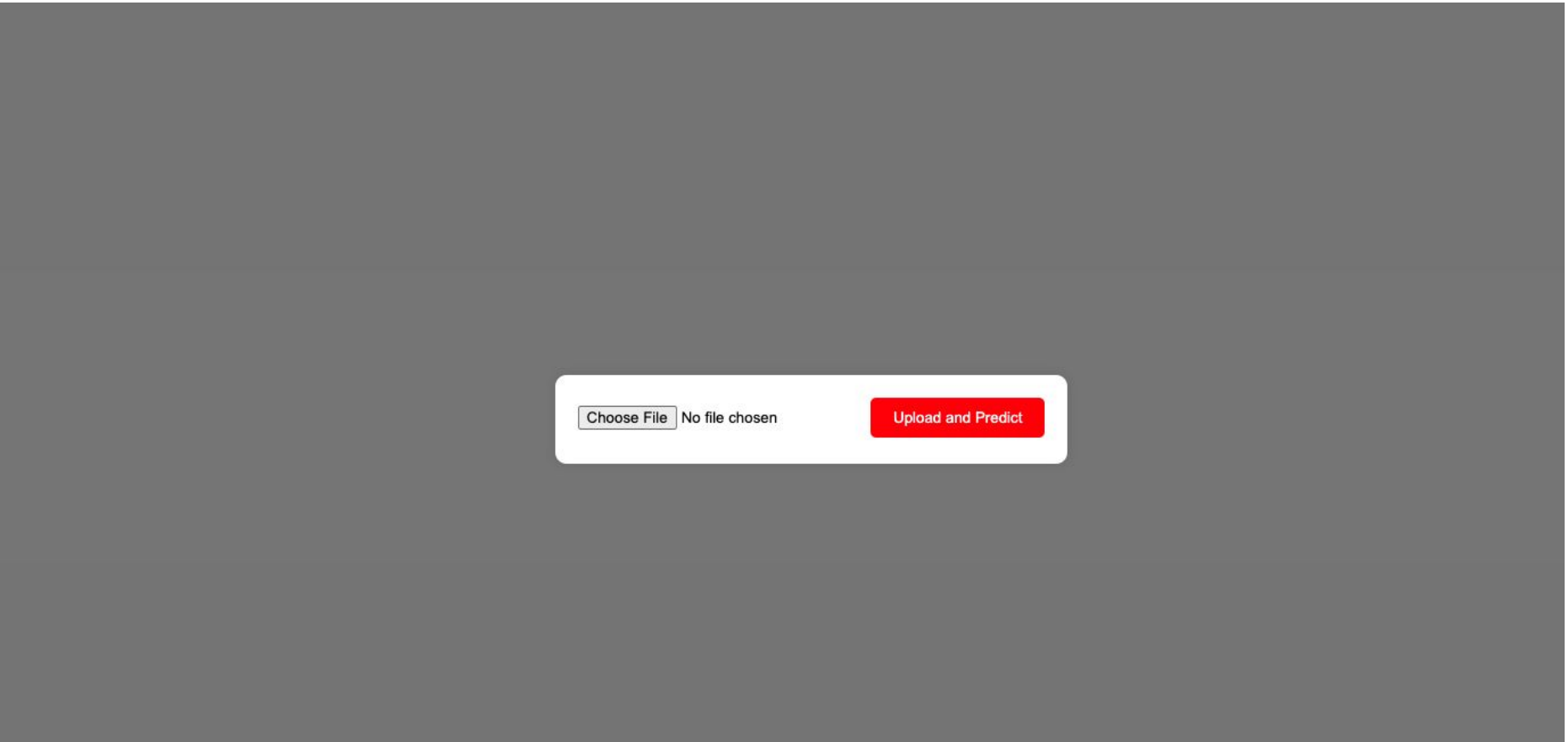
Step by Step

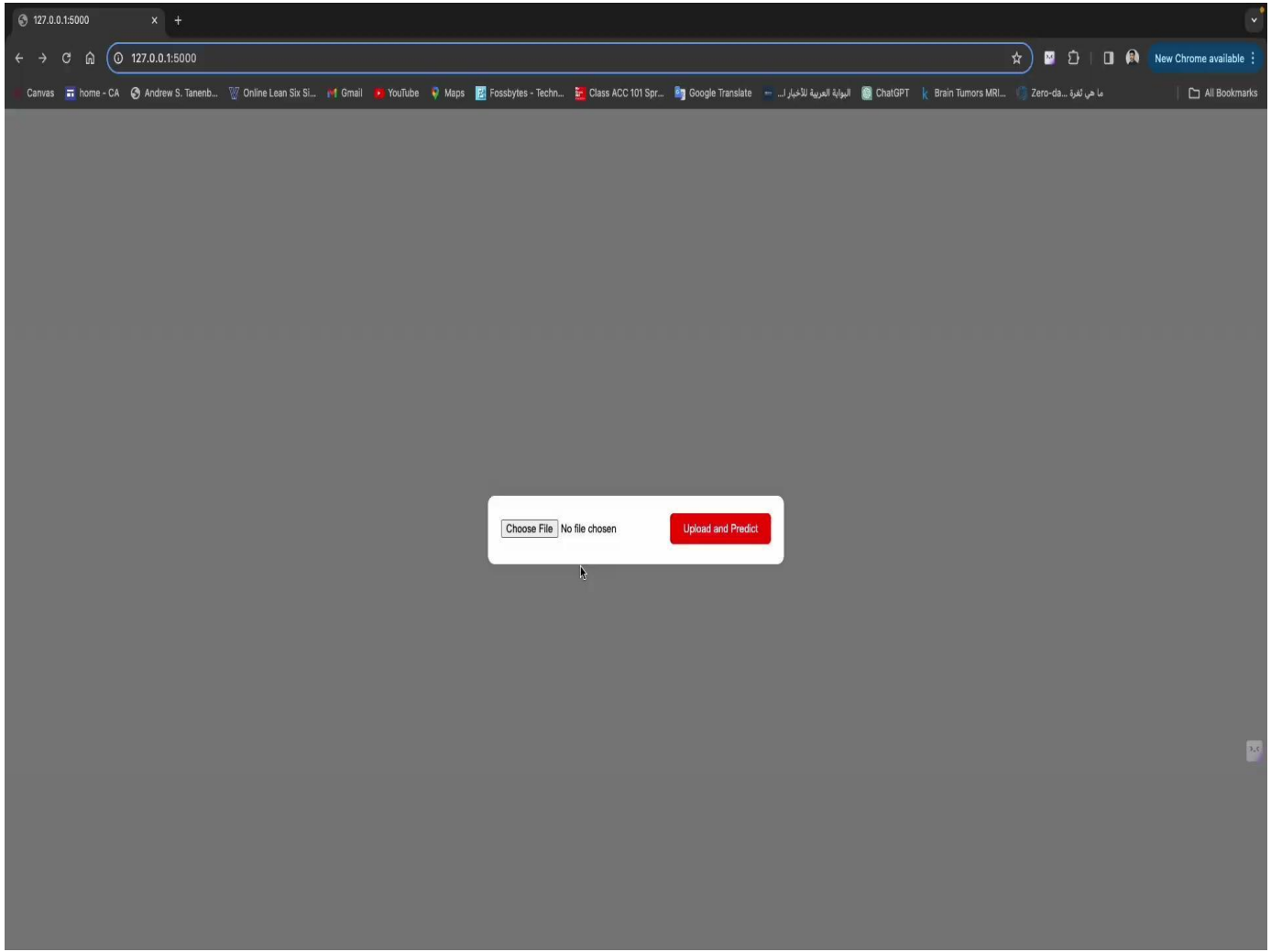
```
<body>
    <form method="post" action="/predict" enctype="multipart/form-data">
        <input type="file" name="file" accept="image/*" required>
        <input type="submit" value="Upload and Predict">
    </form>
</body>
```

```css
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background: gray;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}
```

```python
@app.route('/predict', methods=['POST'])
def predict():
    # Check if a file was uploaded
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'}), 400

    file = request.files['file']


    # If the user does not select a file, the browser submits an
    # empty file without a filename.
    if file.filename == '':
        return jsonify({'error': 'No selected file'}), 400


    # Reading image
    img = cv2.imdecode(np.frombuffer(file.read(), np.uint8), cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (64, 64))
    img = img / 255.0
    img = img.reshape(1, 64, 64, 1)


    # Make a prediction
    prediction = model.predict(img)
    result = 'Yes' if prediction[0][0] > 0.5 else 'No'


    # Return the result as a JSON
    return jsonify({'prediction': result})



if __name__ == '__main__':
    app.run(debug=True)
```

Final result of web development




127.0.0.1:5000
127.0.0.1:5000
New Chrome available
Canvas
home - CA
Andrew S. Tanenb...
Online Lean Six Si...
Gmail
YouTube
Maps
Fossbytes - Techn...
Class ACC 101 Spr...
Google Translate
ChatGPT
Brain Tumors MRI...
All Bookmarks
Choose File
No file chosen
Upload and Predict

Comparing what I have done with the actual timetable

In this presentation I was aiming to present the web development and final adjustments to it, time consumption in getting the best high accuracy for the model took more time than expected, but this comes with great accuracy. However, I feel very confident finishing tasks in the right time. And with the best result that could be applied.

Final Tasks to be completed by 4/17/2024

- Development of web interface and final adjustments

- ChatGPT-API implementation to give an advice regarding the prediction result.

- Final adjustments to the project code and final presentation preparation

- Updating github with all new updates about the project