Brain Scan Tumor Classification

Project Examples

Jordan Fields, Aaliyah Hänni, Vanessa Hsu, Trevor Nims, Alyson Suchodolski, Sabrina Wang

Contents

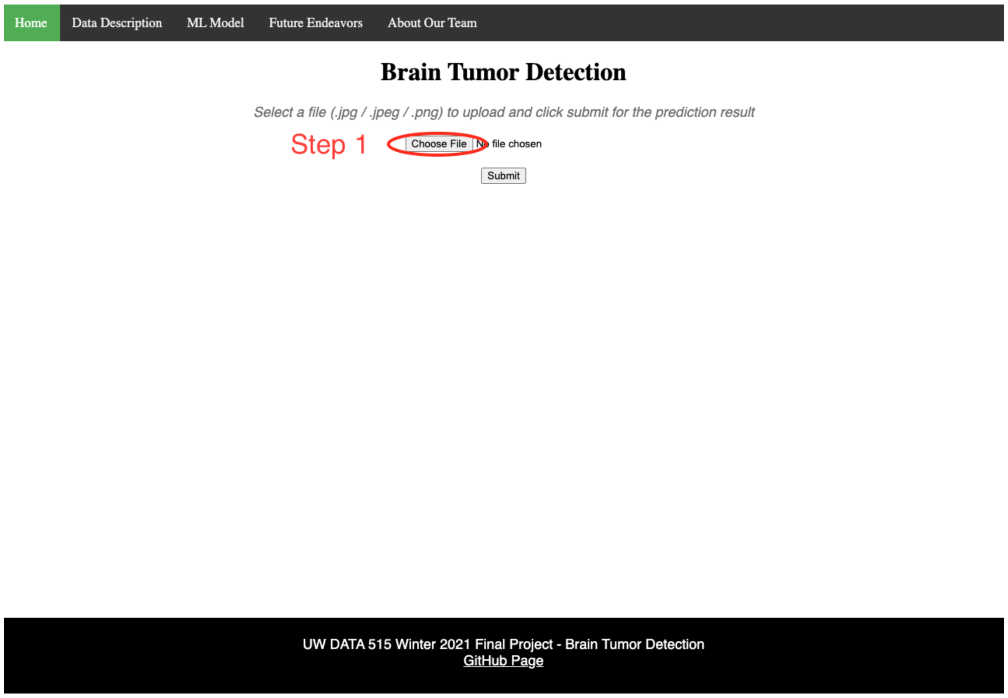
Overview

This document provides users with in-depth examples of how to utilize the published web application, how to run the web application on your own local host, and how to access our model directly.

Using the Web Application: *Do I Have A Tumor?*

The purpose of this project is to decide if a brain is tumorous or not based on an MRI scan input. To utilize the web application you must have an MRI scan image ready (.jpg, jpeg, .png)

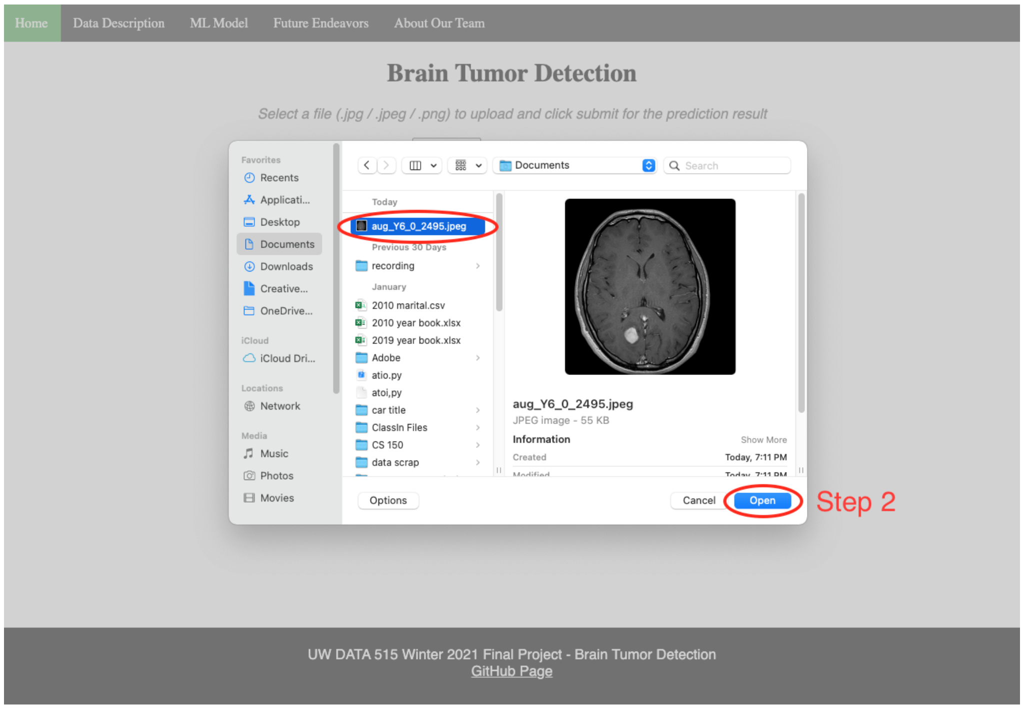
1. Navigate to http://doihaveatumor.com/
2. Click the 'Choose File' button, and select the 2D brain scan (.jpg, .jpeg, .png supported) from your file explorer



Brain Tumor Detection

Select a file (.jpg / .jpeg / .png) to upload and click submit for the prediction result

UW DATA 515 Winter 2021 Final Project - Brain Tumor Detection

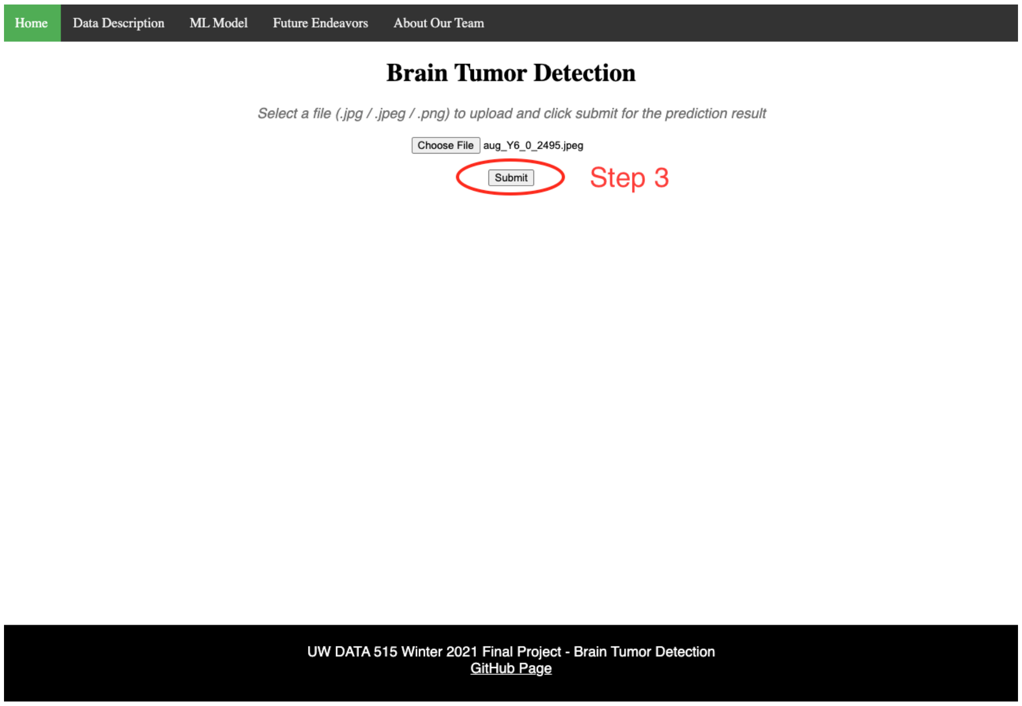

Home Data Description ML Model Future Endeavors About Our Team

Brain Tumor Detection

Select a file (.jpg / .jpeg / .png) to upload and click submit for the prediction result

UW DATA 515 Winter 2021 Final Project - Brain Tumor Detection
GitHub Page

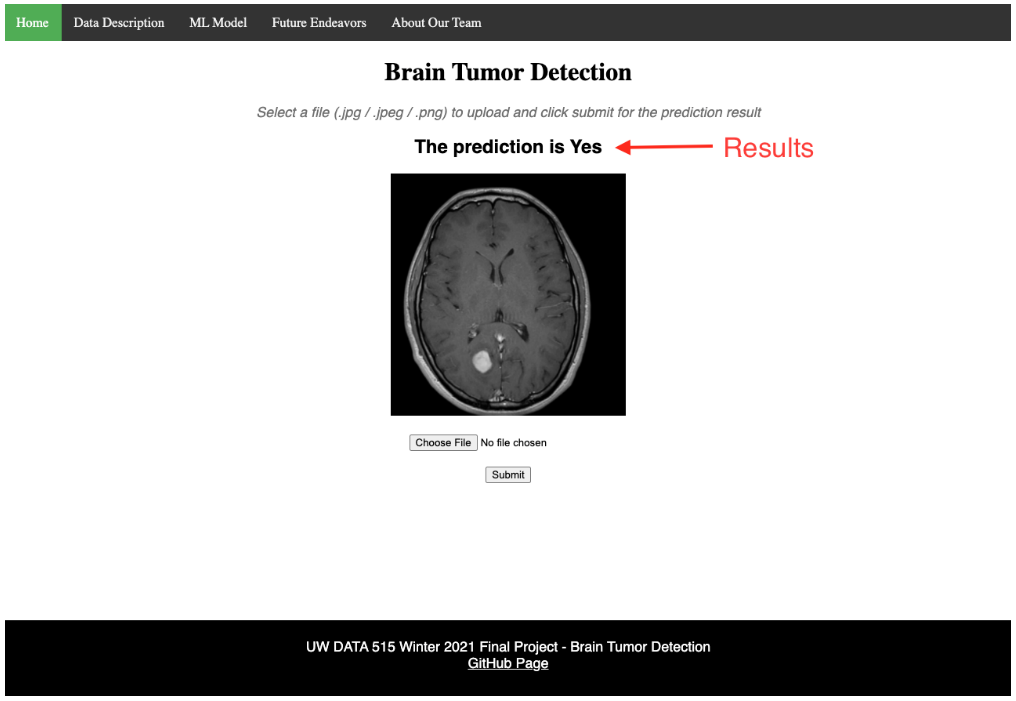
3. Click the 'Submit' button, and wait for the results to display at the top of the page


Home Data Description ML Model Future Endeavors About Our Team

Brain Tumor Detection

Select a file (.jpg / .jpeg / .png) to upload and click submit for the prediction result

UW DATA 515 Winter 2021 Final Project - Brain Tumor Detection
GitHub Page



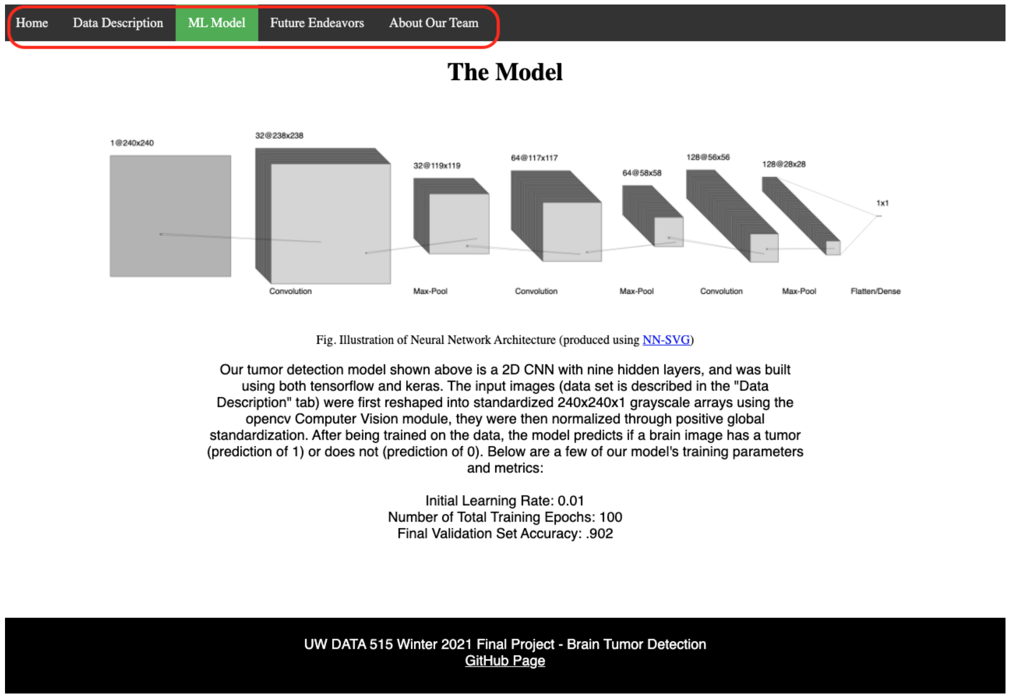
Brain Tumor Detection

Select a file (.jpg / .jpeg / .png) to upload and click submit for the prediction result

The prediction is Yes

UW DATA 515 Winter 2021 Final Project - Brain Tumor Detection
GitHub Page

4. Use the top navigation bar to explore more about the model and our development team



The Model

Fig. Illustration of Neural Network Architecture (produced using NN-SVG)

Our tumor detection model shown above is a 2D CNN with nine hidden layers, and was built using both tensorflow and keras. The input images (data set is described in the "Data Description" tab) were first reshaped into standardized 240x240x1 grayscale arrays using the opencv Computer Vision module, they were then normalized through positive global standardization. After being trained on the data, the model predicts if a brain image has a tumor (prediction of 1) or does not (prediction of 0). Below are a few of our model's training parameters and metrics:

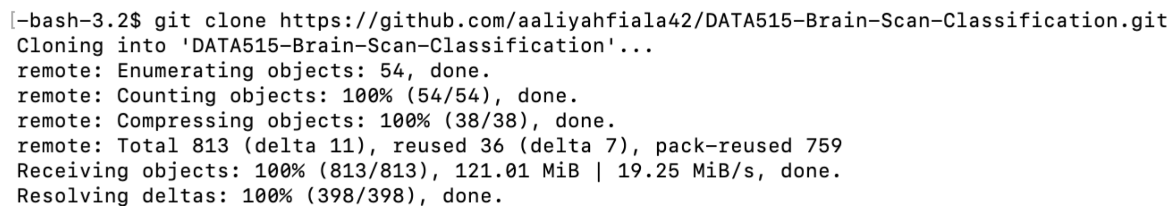
Initial Learning Rate: 0.01
Number of Total Training Epochs: 100
Final Validation Set Accuracy: .902

UW DATA 515 Winter 2021 Final Project - Brain Tumor Detection
GitHub Page

Run Web Application on Localhost

1. In the command line interface run the following command:

   git clone https://github.com/aaliyahfiala42/DATA515-Brain-Scan-Classification.git



```
-bash-3.2$ git clone https://github.com/aaliyahfiala42/DATA515-Brain-Scan-Classification.git
Cloning into 'DATA515-Brain-Scan-Classification'...
remote: Enumerating objects: 54, done.
remote: Counting objects: 100% (54/54), done.
remote: Compressing objects: 100% (38/38), done.
remote: Total 813 (delta 11), reused 36 (delta 7), pack-reused 759
Receiving objects: 100% (813/813), 121.01 MiB | 19.25 MiB/s, done.
Resolving deltas: 100% (398/398), done.
```

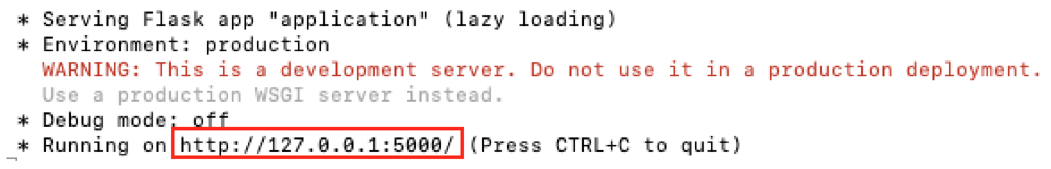
2. Install all required packages by navigating to the root directory (using *cd DATA515-Brain-Scan-Classification*) and run:

   pip install -e.

3. To run the application, navigate to the brain_scan folder (using *cd brain_scan*) and run:

   python brain_scan/app.py

4. Copy the local host url provided (the http address in the red box) and past the url to a browser of your choice.



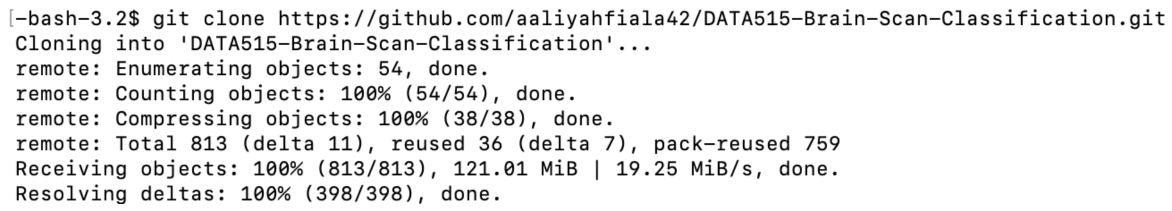
```
* Serving Flask app "application" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Accessing the Model

There are several ways to access our model, to train it on your own dataset, or to simply learn in more detail about how it was created.There is a model class structure (found in the model.py file) that can be accessed from the brain_scan module, following the instructions provided above. Our model is also available as a Jupyter Notebook located in the “notebooks” folder under the root directory. To access the model, you must have a Jupyter Notebook installed on your local machine, or utilize software such as the Anaconda Navigator, and follow the instruction below using the Anaconda CMD.exe prompt.

1. In the command line interface run the following command:

   git clone https://github.com/aaliyahfiala42/DATA515-Brain-Scan-Classification.git

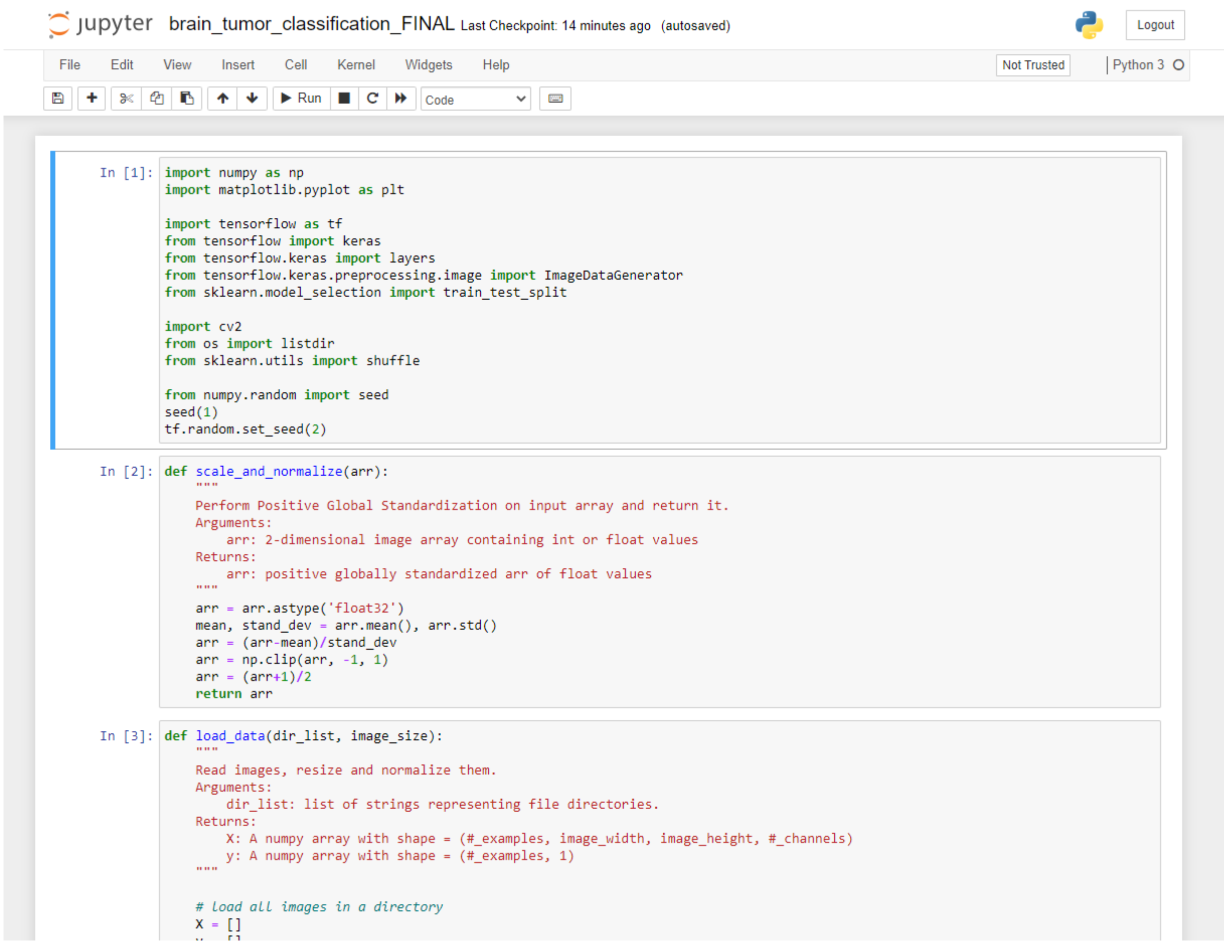
   

   ```
   [-bash-3.2$ git clone https://github.com/aaliyahfiala42/DATA515-Brain-Scan-Classification.git
   Cloning into 'DATA515-Brain-Scan-Classification'...
   remote: Enumerating objects: 54, done.
   remote: Counting objects: 100% (54/54), done.
   remote: Compressing objects: 100% (38/38), done.
   remote: Total 813 (delta 11), reused 36 (delta 7), pack-reused 759
   Receiving objects: 100% (813/813), 121.01 MiB | 19.25 MiB/s, done.
   Resolving deltas: 100% (398/398), done.
   ```

2. To run the application, in the command line interface run:

   jupyter notebook

3. This will then open the Jupyter Notebook interface. Navigate to the *notebooks* and open the *brain_tumor_classification_FINAL* notebook. Now you can view, run, and edit the notebook directly.


jupyter
brain_tumor_classification_FINAL Last Checkpoint: 14 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split

import cv2
from os import listdir
from sklearn.utils import shuffle

from numpy.random import seed
seed(1)
tf.random.set_seed(2)
```

In [2]:
```
def scale_and_normalize(arr):
    """
    Perform Positive Global Standardization on input array and return it.
    Arguments:
        arr: 2-dimensional image array containing int or float values
    Returns:
        arr: positive globally standardized arr of float values
    """
    arr = arr.astype('float32')
    mean, stand_dev = arr.mean(), arr.std()
    arr = (arr-mean)/stand_dev
    arr = np.clip(arr, -1, 1)
    arr = (arr+1)/2
    return arr
```

In [3]:
```
def load_data(dir_list, image_size):
    """
    Read images, resize and normalize them.
    Arguments:
        dir_list: list of strings representing file directories.
    Returns:
        X: A numpy array with shape = (#_examples, image_width, image_height, #_channels)
        y: A numpy array with shape = (#_examples, 1)
    """

    # load all images in a directory
    X = []
```