Brain Scan Tumor Classification

Component Specifications

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

Contents

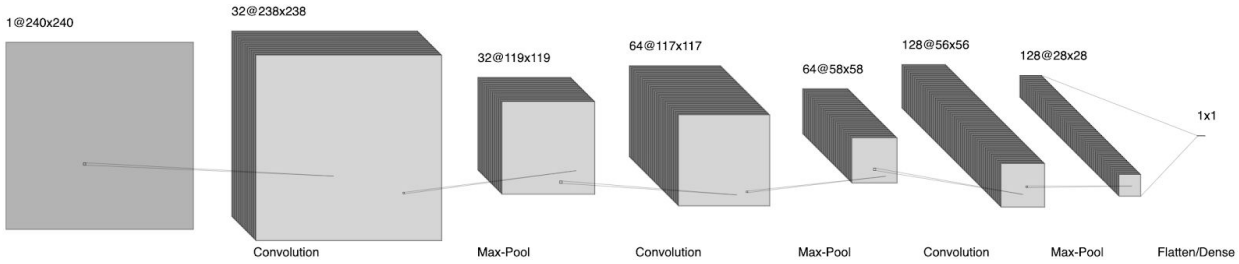
1 Software Components

To accomplish our goals, we will develop multiple software components to interact with each other. Each of these components has an input and output, and are described below.

1.1 Neural Network

We will use a neural network to develop a classification algorithm for our data.

1.1.1 Convolutional Neural Network Design



Input

The model will take as input a 240x240 pixel image represented as a 240x240 array of numbers corresponding to the pixel lumosity.

Output

Using a softmax activation function, the model will output a label corresponding to one of the classification classes. If the model outputs a 0, this indicates the model believes the input image to not contain a tumor and 1 indicates otherwise.

1.1.2 Model Creation Class

The functions used to create our neural network can be found in the Model class in the model.py file.

Method	Input	Output
__init__	*model_path*: file path to pre-existing	Initializes an instance of the

	model (default: None) *rand_seed*: value for random seed (default: None)	class.
load_data	*directory_list*: list of directories that contain image files - each directory must be the parent of a group of image files with a path name ending in either ‘yes’ or ‘no’	Loads in all images, performs preprocessing as detailed in *__pre_process_file*, scales and normalizes the images as detailed in *__scale_and_normalize*, and shuffles and places them into an array of transformed images (*X)* and an array of labels for each image *(y).* *X:* a numpy array containing all transformed images, each with shape (240, 240) *y:* a numpy array containing labels for each image, sharing indices with X
train	*X*: a numpy array of image arrays *y*: a numpy array of labels for each image, sharing indices with X *Num_epochs*: the number of epochs to train the model over (default: 10)	Trains the 2D convolutional neural network with the data provided. The model has 9 hidden layers - 3 convolutional layers and 3 max pooling layers with batch normalization performed in between. Images are augmented to boost performance on unseen data. Data is shuffled and Adam optimization is used with a learning rate of 0.01. The best model is saved as in *models/best_classifier.h5* and can also be referred to with the instance variable *model.*
predict_from_path	*filepath*: file path to image for prediction	After training the model or loading in one that was already made, this function will run the image contained at the file path through our model and output a prediction as a string.

		'yes': returns 'Yes' if the model determines that this MRI image contains a tumor *'No'*: returns 'No' if the model determines that this MRI image does not contain a tumor
__pre_process_file	*directory*: directory where image file is located *filename*: name of the image file	Load each image contained in the directory provided and perform preprocessing by converting the image to grayscale, resizing it, and extracting its correct label from its file path. *image*: single channel grayscale image array of shape (240, 240) *label*: binary value with 1 indication 'yes' and 0 indicating 'no'
__scale_and_normalize	*image_array*: 2D image array containing int or float values	Perform positive global standardization on input array and return it. *arr*: positive globally standardized array of float values

1.2 Predictions

The classes within predict.py assist with extracting predictions from our model to provide to the end user. The class mentioned in 1.1.2 focuses on the model creation process while the classes described in this section work primarily to provide functionality for our website.

1.2.1 Image Class

This class is used to represent our images. The methods allow reading in and processing images.

Method	Input	Output
__init__	*path*: file path to the image	Initializes an instance of the class.
read_from_path		Read in the image located at the instance variable *path*. *image*: numpy array of the image
process		Convert the image to grayscale and reshape so the model can ingest it *Reshape_array*: processed numpy image array with shape (1, 240, 240, 1)

1.2.2 Model Class

This class is used to represent our Tensorflow image classification model and provide predictions for the images our users submit to our tool.

Method	Input	Output
__init__	*path*: file path to the model (.h5 file)	Initializes an instance of the class.
read_from_path		Loads in and returns the model in from the instance variable *path*. *model*: Tensorflow model
prediction	*array*: numpy image array with shape (1, 240, 240 , 1)	Perform prediction using Tensorflow model and return ‘Yes’ or ‘No’ *prediction*: ‘Yes’ if model classifies image as having a tumor; ‘No’ if model classifies image as not having a tumor

1.3 Website Interface

1.3.1 Website Interface for Prediction

We will implement a web interface to the model which allows the user to input data to the model and receive a prediction.

Input

The user will upload a jpg image via a web page. The user will select a file on their local computer by clicking a ‘select file’ button. Then, the user will press a submit button to pass the file to the model.

Output

The website, after analyzing the data, will output a prediction based on the model’s analysis. The website will display, on the same page as the submission, the original input as an image along with a text output indicating the model’s prediction.

1.3.2 Website Interface for Exploring the Data, the Model, Future Endeavours, and the Team

The rest of the website will be dedicated to providing interested users with more information about the data, the model, the team, and ways to improve on our work.

Input

The website will have a number of tabs displayed on a menu bar at the top of the website which allows the user to navigate to various pages. The pages and their output are each described below.

Output (Data Description)

This page will cite the dataset we used for training and testing, and describe some basic characteristics of the dataset. In addition, we will show two sample images from the dataset representing the yes/no tumor classes.

Output (ML Model)

This page will show an educational diagram which corresponds to the model architecture we have developed. In addition, there will be a brief description of the model and the training method (epoch, learning rate, error propagation method, etc) used.

Output (Future Endeavours)

With more time and resources, there are additional goals we would have liked to accomplish as a group. On this page, we will include details on those additional goals and what it was that kept

us from completing them. This way, if anyone is interested in improving on our work they can find information on how to do so.

Output (About Our Team)

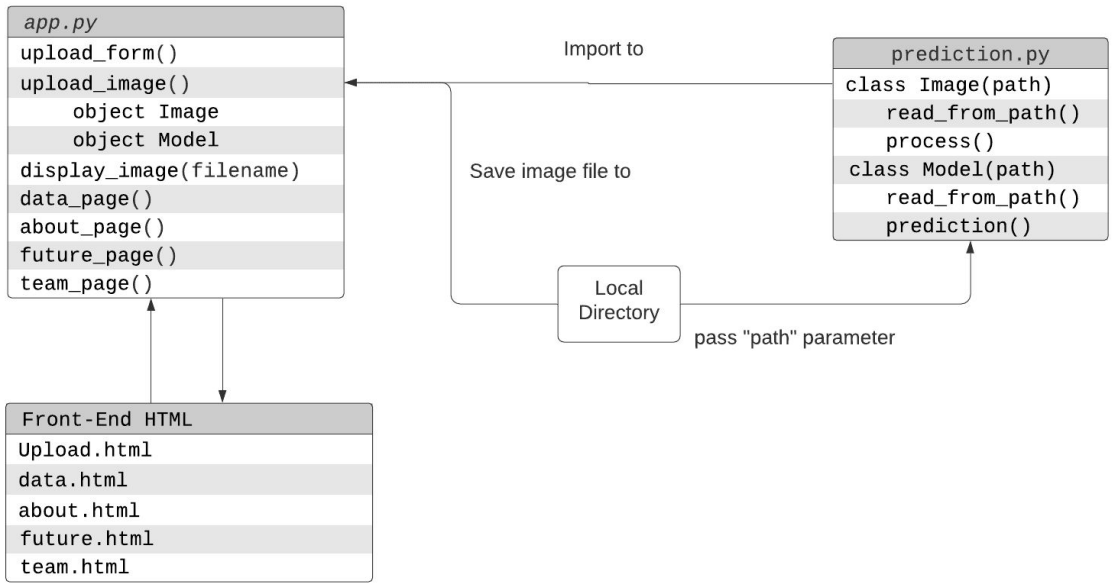
We will include a brief description of each of the team members. We will include information such as name, major, expected graduation date, LinkedIn profile link, and a photo.

2 Interactions to Accomplish Use Cases

We have designed our components to work with each other in order to accomplish the use cases we set out in the functional requirements document.

2.1 Use Case 1 (The Technician)

In our first use case we described a medical technician who would like additional help or confirmation in identifying a brain tumor. This user would accomplish their use case by interacting with the model using the website component. In particular, we expect that the technician would select a slice of interest of the MRI, upload it to the website, and then receive the results via the webpage's output. In this case, the website interface for the prediction component is interacting with the neural network component to satisfy the technician's use case.



2.2 Use Case 2 (The Medical Student)

In our second use case we described a medical student who would like to learn more about the model and it's structure to learn more about predictive modeling in the medical field. In this

case, the user would be able to use the website interface for exploring the model by clicking on the model tab at the top of the website. The output would be information about the model which would satisfy the medical student's use case.

3 Preliminary Plan

In order to complete our goal of developing a web interface to allow users to interact with our developed neural network, and to develop the neural network itself, we will accomplish the following tasks in priority order.

1. Build, train, and test the neural network.
    a. Test different architectures
    b. Test different data preprocessing methods

2. Build and test the class structures
    a. Develop and test the images class
    b. Develop and test the model class

3. Build the website
    a. Build the static pages (data description, ml model, future endeavours, about our team)
    b. Build the page which will interface with the model

4. Connect the website with the model allowing the website to pass data to the model and the model to the website