Code: main_final.ipynb (to be run in google colab, preferably with the online GPU)

Link: https://colab.research.google.com/drive/1AhV3DZGAJLEDCQemP-ftQv1idf6X53zh?usp=sharing

Purpose: This code does data handling and CNN model training on the original brain tumour dataset

External files required: None (data and code (Symmetry.py) are obtained from github)

Code can be run immediately without any changes

1. Install dependencies
    a. Imports necessary packages
    b. Git clones the dataset and imports extractLine class from Symmetry.py file
2. Reading image files
3. Split the data (train, validation, test split)
    a. Option to change the seed value here
4. Data processing
    a. Feature extraction
        i. Note: extractLine(…) will take a while to run
        ii. There is an option to uncomment one of the cells in order to draw symmetry line on all images (visualisation will be done on the later code: plt.imshow(X_train_img[10])
        iii. Boxplot of symmetry metric of the benign and malignant tumours respectively
    b. Augmentation
        i. Can choose to inflate the benign batches to inflate the benign data to even out the dataset (uncomment and comment the necessary code to enable)
    c. Standardisation
        i. Comment and uncomment the necessary code to switch between these 3 options:
            1. Normalisation of images
            2. Per-image standardisation of images
            3. Dataset standardisation of images
        ii. Uncomment to visualise the effectiveness of symmetry metric after image augmentation
5. The model class: this is a class that aims to simplify code pertaining to defining a model, training, fine-tuning and data analysis.
    a. Defining the object CNNModel. The object takes in these variables, T is the number of training samples and V is the number of validation samples:
        i. model_name: name of model as string ("VGG16", "InceptionResNetV2" or “ResNet50”)
        ii. X_train_img: numpy array of shape (T, 128, 128, 3)
        iii. X_val_img: numpy array of shape (V, 128, 128, 3)
        iv. y_train: numpy array of shape (T,), with binary values
        v. y_val: numpy array of shape (V,), with binary values
        vi. X_train_ftr: (Optional) numpy array of shape (T, n)
        vii. X_val_ftr: (Optional) numpy array of shape (V, n)
    b. Methods (see code for input parameters):
        i. run_model: trains the model and saves it as self.model
        ii. fine_tune: trains on self.model and saves it as self.ft_model

iii. fine_tune_again: trains on self.ft_model and saves it as self. ft_model
iv. plot_loss_curve: plot history logs for loss and accuracy curves across epochs (for fine tuning too if it had been done)
v. show_confusion_matrix: shows confusion matrix on validation data, and reports metrics in this order: accuracy, miss rate, f1 score. Set fancy=True to enable seaborn heatmap visualisation, set test_data= (X_test_img_std, X_test_ftr_std, y_test) to show results on test data instead of validation data
vi. show_misclassified: shows misclassified images (on validation data only)

c. Note: The code for the construction of the model will be under the method: _create_XXX_model

6. Fitting the model (it is possible to initiate multiple models at once and perform data analysis on multiple models):
    a. Initiate 4 different models:
        i. VGG16
        ii. InceptionResNetV2
        iii. ResNet50
        iv. ResNet50 (without symmetry metric input)
    b. Training (epochs set to 12,10,10,10 respectively, batch size set to 32 for all)
    c. Visualising model.summary
7. Fine-tuning
    a. Initial fine tuning
    b. (Optional) Fine tune again
8. Data analysis (Model evaluation)
    a. Plotting loss and accuracy curves
    b. Show confusion matrix for both validation and test data
    c. Show misclassified images (on validation data only and on fine-tuned model if fine tuning is done, otherwise the images will be based on the pre-tuned model)

Code: main_saliency.ipynb (to be run on google colab)

Link: https://colab.research.google.com/drive/17xayTwqWro2X2sgjKPFRlmeIvfDeEdzz?usp=sharing

Purpose: This code extracts the saliency map from the trained and tuned VGG16 model

External files required: None (data and code (Symmetry.py, vgg16.keras and vgg16_no_ftr.keras) are obtained from github)

vgg16.keras is trained with symmetry metric while wgg16_no_ftr.keras is trained without symmetry metric

The numpy array img contains 154 malignant images followed by 77 benign images (in order, without shuffling)

Code can be run immediately without any changes

Under 7. Saliency map, the function visualise takes in 3 inputs:

1. keras model
2. index: index of the selected image in the array: img
3. using_ftr: set to True if the model is trained on symmetry metric, else set to False if otherwise
4. Output: None, this function plots 3 figures: Original image, saliency map and the overlaid image