PYCK Project report

Build brain tumor classifier by implementing Keras API with EfficientNetB1 framework

By,

Abhinandan Kumbhar

193109007

Introduction:

The objective of the study is to learn basics of deep learning framework and use the knowledge of python learnt during the course of PYCK. The weights of a pre-trained model called EfficientNetB1 is used and retrained for the given problem. I have taken help from a coursera course to do this project. The project also gave me opportunity to learn new library OpenCV, OS, tqdm, Keras

Implementation details:

Clone the Github Repo to access the Dataset and import necessary Libraries:

1. The dataset from github repo was copied in local directories and import necessary libraries.
2. Some of the new libraries learnt during the project are:
    - os – This library helps to deal with directories of google colab. We can create and access directories in colab. Also, we can manipulate data in the directory.
    - tqdm – The library creates task completion bar which indicates the progress made.
    - OpenCV2 – used for image processing.
    - Tensorflow – Keras API of tensorflow is used for implementing deep learning frameworks

The pre-trained weights of CNN framework called EfficientNetB1 is used and extended by adding an extra dense layer with drop out regularization.

Creating local libraries to store the cropped images:

The os library was used to create directories where cropped images will be stored as training and testing sets

Create a Function to crop images:

1. The images in dataset are having extra plain black regions surrounding the actual image.
2. Since this extra region does not add any value and does not catch any important feature, we have to crop extra region.
   - We start with converting colored images into gray image using COLOR_BGR2GRAY method of opencv library.
   - We then use GaussianBlur method to remove insignificant gray regions and keep only significant contours which helps to learn important features of image.
   - We then reshape this cropped images in shape of (224,224) and save them to respective directories to be accessed later.

Data Augmentation:

Since we have limited number of examples, we use data augmentation techniques to expand the dataset. We rotate, flip and translate the images to create new images.

Prepare the Train, Validation and Test Dataset:

This datasets of batch size of 32 are created.

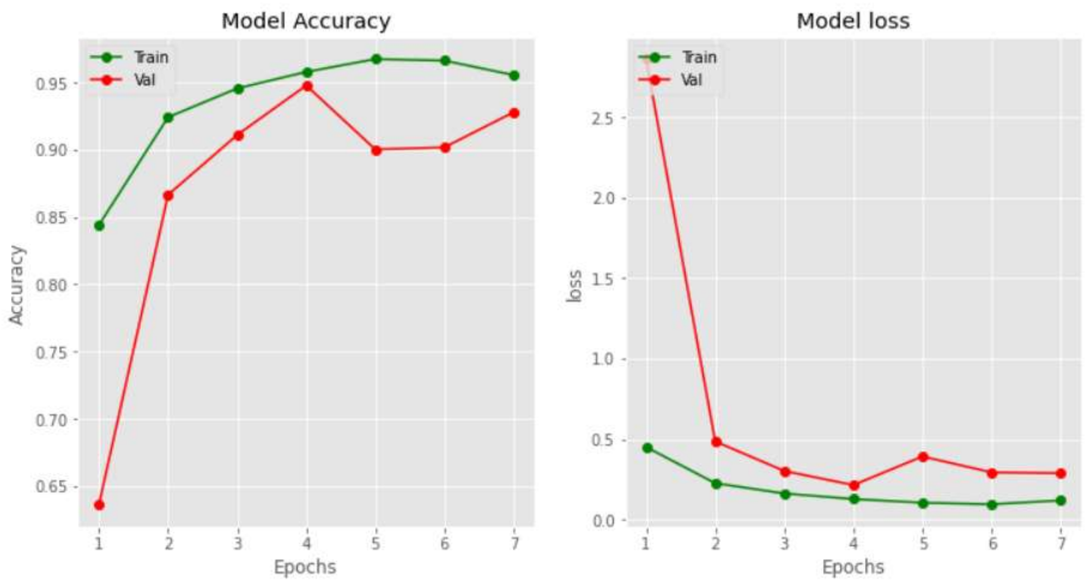
Build and Compile the Model:

1. The EfficientNetB1 is used to create the required model with CNN framework with input of size (224,224,3)
2. We need output of size 4x1, since there are 4 classes. Since, the output of EfficientNetB1 is not 4x1, we have to add a dense layer which outputs 4 units along with dropout regularization and global average pooling.
3. We compile this model using Adam optimizer with learning rate of 0.001. We use categorical cross entropy and use accuracy as metric.

Model Training and Model Evaluation:

1. We train the model with 7 epochs and we have to use early stop checkpoints.
2. The EarlyStopping method stops the optimization, if the validation accuracy does not improve for 5 iterations (patience level)
3. The test accuracy comes out to be 0.9%

The below graph shows improvement in performance over different epochs. The green graph is for training and red is for validation



Conclusion:

The above graphs show that the training accuracy improves as epochs increases and also validation set accuracy is also improving, which is expected behavior. The test accuracy is also good i.e 90%. We can say the model is trained properly.