The project is about image classification of MNIST digits data on the basis of Convolutional Neural Network (CNN) Model trained using image data of various diseases. A famous machine learning (ML) technique, Transfer Learning

**Transfer Learning?**

**Why the project is important**

* + 1. It can help doctors and other practitioners to correctly identify the type of disease using x-ray or other images which are sometimes difficult to identify using naked eye or may be ignored due to human error.
    2. It can reduce time to identify and reach a decision about a disease.
    3. It can serve as a second opinion about identification of a disease type.
    4. It can cope with the most of the situations where the data is scarce and of a new category just like in our case we are classifying digit images where as model is trained on images of diseases.

**Libraries:**

import os

import matplotlib.pyplot as plt

import tensorflow as tf

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.utils import to\_categorical

import shutil

from PIL import Image

**Datasets:**

Base Model - Statistics

* 13 data sets for base model – Actually they are representing 13 disease types e.g., Skin Cancer, Covid, Monkey pox, Pneumonia, Kidney Cancer, Lung Cancer, etc
* Number of categories: 47 e.g in Skin Cancer there were 9 types of different diseases: melanoma, nevus, Basal cell and so on
* Data Size: 11.1 GB
* Minimum Number of images in a disease category: *seborrheic keratosis* 80
* Maximum: Alzheimer - MRI – Non-demented 12800

**Preprocessing (9 steps):**

* Removal of Non Image Files
* Reduced number of images to maximum of 4000 in a category (Now we had 91450 images in 47 categories)
* Increased number of images in the categories where the number images were less than 3000 using algorithm ‘ImageDataGenerator’ ( Now we had 173,441 images in in 47 categories)
* Lowere the image size to 120 X 90 (W X H)
* Changed the images to grayscale (lowring the dimensions)
* Normalized the data. (It scales the values to a range between 0 and 1)
* Converted the labels to categorical by one-hot encoding (i.e., vectors)
* Converted the data into training and testing sets (ratio 80:20)

**Base Model :**

- Number of Conv layers: 7 , contains: Number of Filters, Filter Size, Chaneels, Activation Function= Relu

- Maxpooling Layer

- Dropout layers

- Flatten layer

- Number of Dense Layers. 6, Number of Neurons, and activation Function

- Softmax function

- Epochs 25, batch Size 16 , categorical cross-entropy, Optimizer Adam

Training Accuracy: 0.9622

Test accuracy: 0.9161

Total params: 27,962,543

Trainable params: 27,962,543

Saved the base model

**Model (MNIST)**

* Preprocessed – increased the image size
* Loaded the saved model
* Froze all the conv layers of base model upto Flatten
* Added 6 dense layers

Epochs 100, batch Size 8 , categorical cross-entropy, Optimizer Adam

Training accuracy: 0.8637

Test accuracy: 0.9043

Total params: 26,388,010

Trainable params: 1,223,978

Non-trainable params: 25,164,032

**Key Findings:**

* + - * + From the results it is evident that transfer learning is very helpful technique to generalize the features learned from one dataset and apply to the other dataset with quite high accuracy. Especially in the case of datasets that conatin images is quite encouraging.
        + Further transfer learning is helpful when the amount of labeled data is quite small. In that case features leaned from base model can still augment the model structure seamlessly to obtain highly accurate results.

**Future Directions:**

* + - * + There is need to deeply investigate and tune the model further to improve its accuracy.
        + Also new types of images are needed to be added in the dataset to train the model on variety of types and shapes of images.
        + After that there is need to compare this model performance with the other available models available in the market e.g., VGG16, VGG19, Resnet, InceptionV3, Xception, MobileNetV2, DenseNet, EfficientNet etc. etc. and improve the model for better performance.

|  |
| --- |
|  |