CS 351
MACHINE LEARNING

Mini Project Assignment Prepared By:Shreyas Bhat (181ME175)
Faculty in Charge:Dr Asoke K Talukder

Introduction

- Since I had prior experience with Deep Learning I decided to try and write the code from scratch to get a better understanding of the Dataset and the various problems that may arise
- I used Tensorflow 2.x as the library to implement the CNN
- I used the Kaggle Platform to train my model.Theres 2 main reasons for this:
    - Kaggle comes with a pre built environment to run python code hence eliminating installation hassles
    - Kaggle boasts a Nvidia P100 GPU which will help reduce training times and let me experiment with model
- After training the inference script was run on my local machine.This was so I can get a taste of solving installation issues while not compromising on speedy training.
- Source Code: https://github.com/DarthRoco/CS351-ML

Steps Involved

01 Dataset preprocessing

02 Input Pipeline

03 Model Definition and Training

04 Inference

01 DATASET PREPROCESSING

- Dataset was downloaded from this link. It consists a random subsample of ~5000 images from original dataset
- The labels are originally suited for multi categorical classification.However as per sir's instruction I had to convert it to a binary classification problem
- Thus I used the Pandas library to create a new column with a simple condition based filter on the labels already provided
- As seen in code snippet below if the label was "No Finding" rename to Healthy else rename to "Diseased"
- After which we sort and only take first element (since some rows have multiple labels)

```
df['bin']=df['Finding Labels'].apply(lambda s: ["Diseased" if l not
in ['No Finding'] else "Healthy" for l in str(s).split('|')])
df['bin']=df['bin'].apply(lambda s:sorted(s) )
df['bin']=df['bin'].apply(lambda s:s[0] )
df.head()
```

02 Input Pipeline

- Since the dataset is relatively large;loading it entirely in ram is not feasible.Thus I used the ImageDataGenerator class from TensorFlow which sequentially loads batches of data into ram so as to prevent overflow.
- Additionally ImageDataGenerator lets us perform preprocessing such as normalizing and even augmentation techniques like random flipping
- I decided not to use augmentation since this code was just proof of concept.However I did resize the images to 256x256 and normalise pixel values from 0-255 to 0-1.This speeds up the training process.

```
datagen=tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1/255,
    validation_split=0.0, dtype=None
)
```

```
train_generator=datagen.flow_from_dataframe(dataframe=df,x_col="Path",y_col="bin",class_mode='binary'
                                            target_size=(256,256),
                                            batch_size=256)
```

03 Model Definition And Training

- For this phase I decided to attempt 2 ways of selecting model.
- In the First Model(Model A) I tried to add a few dense layers after a pretrained EfficientNet V4 backbone and fitted the dataset
- In Model B I used the architecture provided in source and trained entire model from scratch.I chose this model as final since it gave better scores
- The metric used for comparison in both cases was ROC-AUC score which is commonly used for imbalanced classification problems
- The results are summarised below.

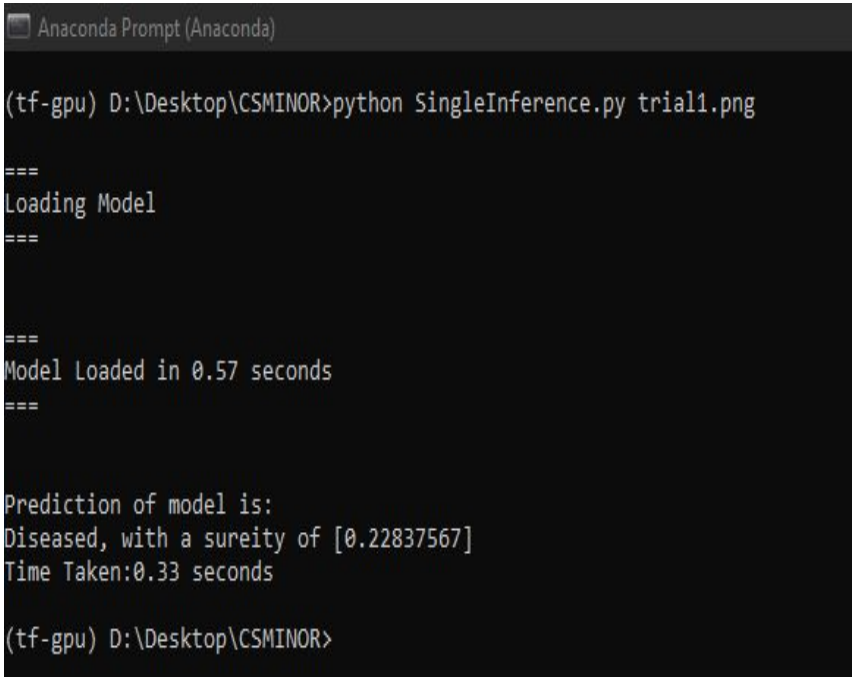
MODEL A

- Used concepts of transfer learning
- Backbone consisted of EfficientNet V4
- Trained for ~60 epochs
- Best AUC score:~0.7

MODEL B

- Trained Model from scratch
- Architechture inspired from original source provided by sir
- Trained for ~60 epochs
- Best AUC score:~0.96

04 INFERENCE



- Once the model was trained I downloaded it as a .h5 file which is native to TensorFlow.With this file I can run the model in inference mode on any device without access to original source.
- I created an Anaconda Environment on my local machine and installed all the necessary libraries
- I wrote a simple python script that takes the path of a single image in the form of a single command line argument as input and predicts whether it is “Diseased” or “Healthy”(threshold 0.5)
- The script loads the image as per path and performs resize and normalisation operation on it.Then loads the model and performs prediction.
- For full source refer this

THANK YOU

Prepared by Shreyas Bhat 181ME175

Faculty In Charge:Dr Asoke K Talukder