Term Work Phase 1 Report

Skin Cancer Detection using CNN

Aditya S - CB.EN.U4ECE17303

Ashwinkumar J S - CB.EN.U4ECE17307

- For this Project We have used the SKIN CANCER MNIST :HAM 10000 Dataset.

- Initially We Uploaded the dataset to Google Drive and accessed it through Google Colab.

- We have made the image path dictionary by joining the folder path from base directory base_skin_dir and merge the images in jpg format from both the folders HAM10000_images_part1.zip and HAM10000_images_part2.zip

- We have read the csv by joining the path of the image folder which is the base folder where all the images are placed named base_skin_dir. After that we made some new columns which is easily understood for later reference such as we have made column path which contains the image_id, cell_type which contains the short name of lesion type and at last we have made the categorical column cell_type_idx in which we have categorize the lesion type in to codes from 0 to 6.

- Next, the images will be loaded into the column named image from the image path from the image folder. We also resize the images as the original dimension of images are 450 x 600 x3 which TensorFlow can't handle, We have converted the data frame into a pickle file and attached it along for evaluation .

- Then,we have splitted the dataset into training and testing set of 80:20 ratio.

- Then Normalized the x_train, x_test by subtracting from their mean values and then dividing by their standard deviation.

- And then We encoded the labels to one hot vectors.

- We splitted the train set in two parts : a small fraction (10%) became the validation set which the model is evaluated on and the rest (90%) is used to train the model.

Model Building

- We used the Keras Sequential API, where you have just to add one layer at a time, starting from the input.

- The first is the convolutional (Conv2D) layer. We chose to set 32 filters for the two firsts conv2D layers and 64 filters for the two last ones. Each filter transforms a part of the image (defined by the kernel size) using the kernel filter. The kernel filter matrix is applied on the whole image. Filters can be seen as a transformation of the image.

- CNN can isolate features that are useful everywhere from these transformed images (feature maps).

- The second layer is the pooling (MaxPool2D) layer. This layer simply acts as a downsampling filter. It looks at the 2 neighboring pixels and picks the maximal value. These are used to reduce computational cost, and to some extent also reduce overfitting.

- We have Used the ‘relu’ activation function to add non linearity to the network.

- We chose Adam optimizer because it combines the advantages of two other extensions of stochastic gradient descent.

- The metric function "accuracy" was used to evaluate the performance of our model.

- In order to make the optimizer converge faster and closest to the global minimum of the loss function,We used the ReduceLROnPlateau function from Keras.callbacks,

- We chose to reduce the LearningRate by half if the accuracy is not improved after 3 epochs.

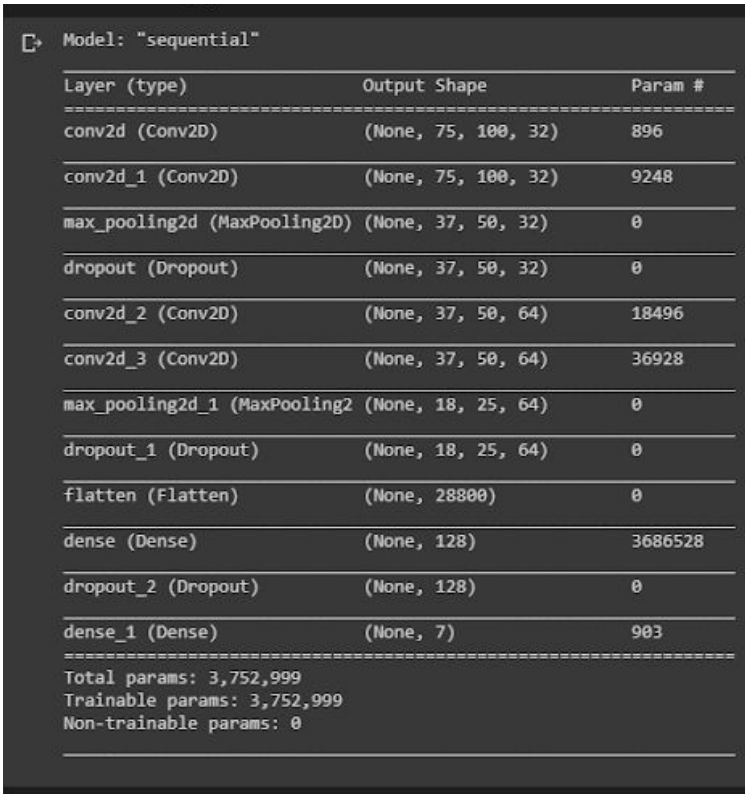
- In order to avoid overfitting, we also included a Data augmentation Function ImageDataGenerator() .

We fit the model into x_train, y_train.

In this step We have chosen batch size of 10 and we have chosen 20 epochs to give the model sufficient epochs to train.

Results.

Model Architecture:

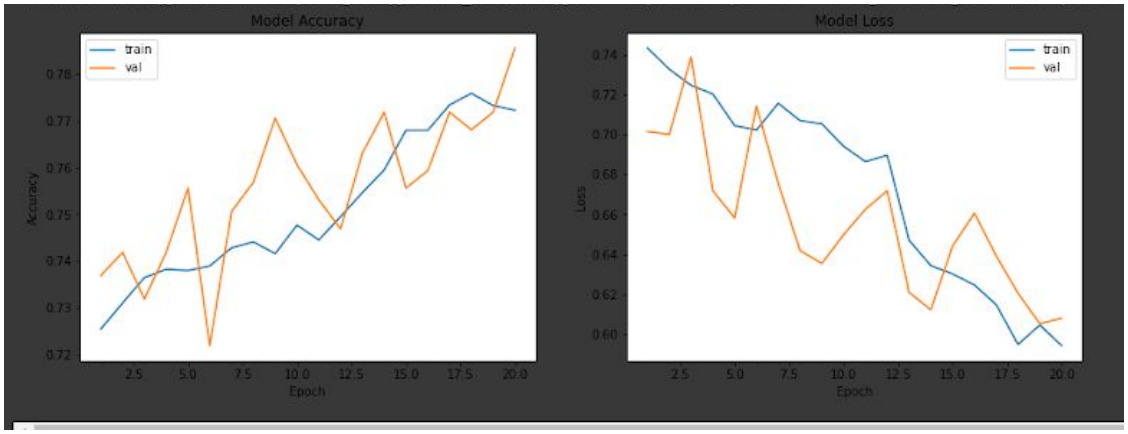


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 75, 100, 32)       896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 75, 100, 32)       9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 37, 50, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 37, 50, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 37, 50, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 37, 50, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 18, 25, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 18, 25, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 28800)             0
_________________________________________________________________
dense (Dense)                (None, 128)               3686528
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 903
=================================================================
Total params: 3,752,999
Trainable params: 3,752,999
Non-trainable params: 0
_________________________________________________________________
```

Evaluation Report Using Inbuilt Function.

```
63/63 [==============================] - 1s 14ms/step - loss: 0.6349 - accuracy: 0.7619
26/26 [==============================] - 0s 12ms/step - loss: 0.6081 - accuracy: 0.7855
Validation: accuracy = 0.785536  ;  loss_v = 0.608067
Test: accuracy = 0.761857  ;  loss = 0.634861
```

Graph Showing the variation in Loss and Accuracy in every epoch



Conclusion:

We have obtained a Test Accuracy of 76% using a custom built Convolution network, In the Next phase We plan to improve the Accuracy from pre trained networks like Densenet.