CNN Model for Image Classification of Skin Lesions (Melanoma, Nevus, Seborrheic-Keratosis)

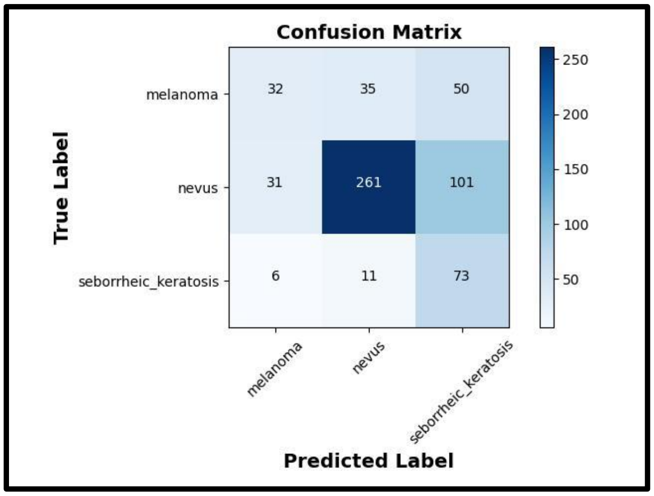
Members:
Harsh Aggarwal (2020508)
Harsh Krishan (2020509)

We created a CNN model to successfully classify the cancerous skin lesions (Melanoma, Nevus, Seborrheic-Keratosis), this report is about the final results and whole process how we implemented it.

First we want to discuss the results of our model:

- We created total 11 models to do the task.
- The best accuracy we got through the best of our models was **65.5%**
- We have also checked the accuracy of pre-trained models like VGG-16 and mobileNet, they got almost **79%** and **80%** respectively.

The confusion matrix of the best of our models is shown below.



The following materials are present in the final zip file:

- The model (HDF5 format)
- The prediction file (with 65.5% accuracy)
- The confusion matrix plot
- The main python notebook

Procedure:

➔ Importing Required Python Liabraries:

- We used the following python libraries for the project:

    - **Numpy:** For basic mathematical operations on dataframe
    - **Keras:** For building the CNN model
    - **Pandas:** For managing the dataframe
    - **Scikit-Learn:** For Image Processing and Metrics Evaluation
    - **Matplotlib:** For plotting the plots
    - **OS:** For managing the directory

➔ Data Preparation:

- We did the following for data preparation part:

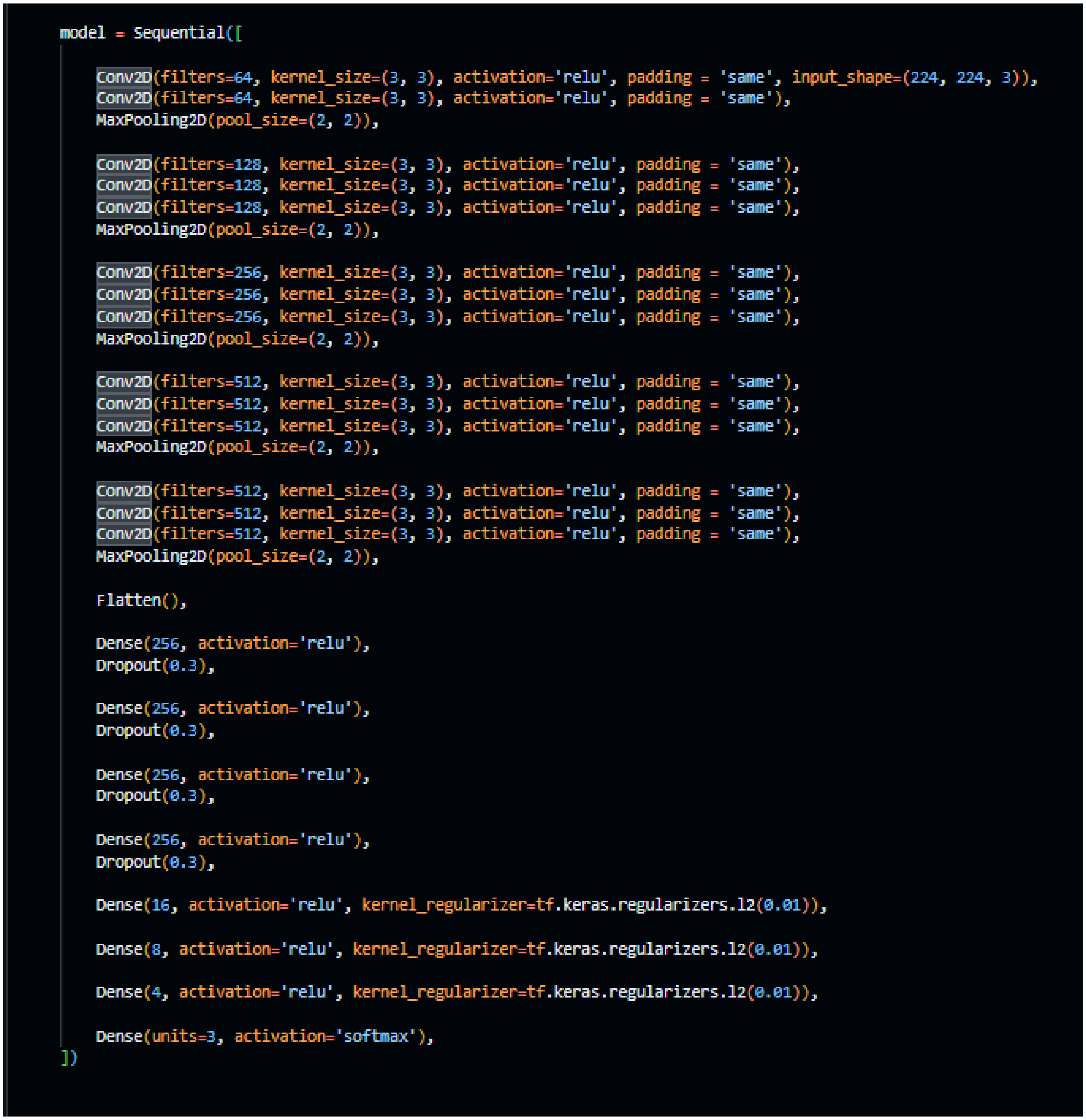
    - Data path setup
    - Data files loading
    - Image data preprocessing, by converting image from RGB to BGR format using ImageDataGenerator module from “keras.preprocessing.image”
    - Changing the dimensions of image to 224 x 224
    - Splitting the data into batches to make them suitable for k-cross fold validation, done later

➔ Class Imabalance:

- There was a drastic class imbalance in the data, so we did data augmentation in each step of k-cross validation to reduce the bias of one class over other classes

➔ Defining the CNN Model:

- ◆ The following is the description of our sequential CNN model layer by layer:



```
model = Sequential([

    Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same', input_shape=(224, 224, 3)),
    Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'same'),
    Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'same'),
    Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(filters=256, kernel_size=(3, 3), activation='relu', padding = 'same'),
    Conv2D(filters=256, kernel_size=(3, 3), activation='relu', padding = 'same'),
    Conv2D(filters=256, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(filters=512, kernel_size=(3, 3), activation='relu', padding = 'same'),
    Conv2D(filters=512, kernel_size=(3, 3), activation='relu', padding = 'same'),
    Conv2D(filters=512, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(filters=512, kernel_size=(3, 3), activation='relu', padding = 'same'),
    Conv2D(filters=512, kernel_size=(3, 3), activation='relu', padding = 'same'),
    Conv2D(filters=512, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPooling2D(pool_size=(2, 2)),

    Flatten(),

    Dense(256, activation='relu'),
    Dropout(0.3),

    Dense(256, activation='relu'),
    Dropout(0.3),

    Dense(256, activation='relu'),
    Dropout(0.3),

    Dense(256, activation='relu'),
    Dropout(0.3),

    Dense(16, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)),

    Dense(8, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)),

    Dense(4, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)),

    Dense(units=3, activation='softmax'),
])
```

- **Model Compilation:**

    - We use Adam as optimizer and we used accuracy as the metrics for our model

- **Model fitting:**

    - We did 10 cross validation, in each iteration we did the following operations:

        - We took 9 batches for training and remaining 1 batch and given validation dataset combined for validation
        - We did data augmentation to handle class imbalance
        - We then calculated test score for each step
        - After all above operations we deleted all generated augmented images, so they cannot be repeated again in another iteration

- **Saving Model:**

    - Since the model took almost 10 hours for each iteration, so it would be impractical to train model again, that's why we saved the model. Model file name "model_11.h5"

- **Predicting results:**

    - We predicted the results, and got 65.5% accuracy, prediction file "predictions_11.csv".