Skin Disease Detection

Final Report

Supervised by : Dr. Mohamed Elshenawy

Team

1- Ibrahim Abdalaal	201800224
2- Amr Ragab Abdelaziz	201801041
3- Mohamed Abdelwahed	201801978
4- Abdullah Kamal Muhammad	201801271

Problem Definition and Motivation

Deep learning (DL),machine learning (ML) and Artificial intelligence (AI) have added many improvements to different medical fields. Due to these improvements, it is now possible to detect diseases earlier and get better results by integrating the efforts of human doctors to the added value of the DL models. There is a medical field called Dermatology which studies, detects and treats skin diseases of skin, hair and nails. Unfortunately, Dermatology did not achieve great progress like other medical fields [1]. There are many reasons that prevent that medical field from being improved, such as weather patterns and personal hygiene-related medical treatment. The variety and challenges that face medical professionals to recognise the features of skin disorders, the majority of which are skin diseases.

The use of AI systems will represent a strong addition to the early detection of diseases, which increases the possibility of recovery before it reaches its final stages that traditional medicine cannot treat, as some diseases have developed into cancerous cells [2]. Late detection of diseases increases the risk because of its high cost, and therefore the solution lies in the use of artificial intelligence applications such as computer vision, which will be available to anyone regardless of the financial level of the patient, and all that is needed is a picture of the area believed to have a disease instead of laboratory examinations. DL models will provide us with the needed flexibility and high accuracy of detection we aim to. DL models improve their accuracies by using datasets with large size so it will be more helpful to benefit from the concept of Transfer Learning and use other or pre-trained models that learned from other data that is not available to me which will result in better accuracies.

Dataset

We used two datasets to improve the training of our model. The first dataset has images for 23 different skin diseases. It has 19500 RGB images. The dataset is split into 15,500 images in the training and 4000 in the test. The images' resolution is not very high. Additionally, the 23 classes are not equally distributed. The dataset is retrieved from the public portal Dermnet (http://www.dermnet.com), which is the largest dermatology source online built for the purpose of providing online medical education.

The second data set is retrieved from the kaggle website and it is called ‘Skin diseases image dataset’. This dataset has 10 classes for different skin diseases. This dataset has 27053 RGB images. The dataset is not splitted. The images in it have a better quality than the Dermnet dataset.

We used the second dataset to increase the number of images in the first dataset, so we merged the second dataset with the training in the first dataset to have more quality images and better distribution.

Approach

We used transfer learning as our main approach.by training three different models on our dataset-VGG16 , ResNet50 and Xception- and comparing the performance of the three models to settle on the best model.

Since these pretrained models are already tested on image classification tasks and taking into consideration the difference between our dataset and the original datasets the models are trained on ,We believe that good accuracy could be achieved by fine training and tuning.

Methodology

To achieve good accuracy ,these models should have enough data since they are complex ,hence ,data-hungry.With the small size of our dataset along with the hard classification task-23 classes- two solution were proposed as follows :

- Using data augmentation
- Increasing the data by merging similar dataset with the same classes if possible

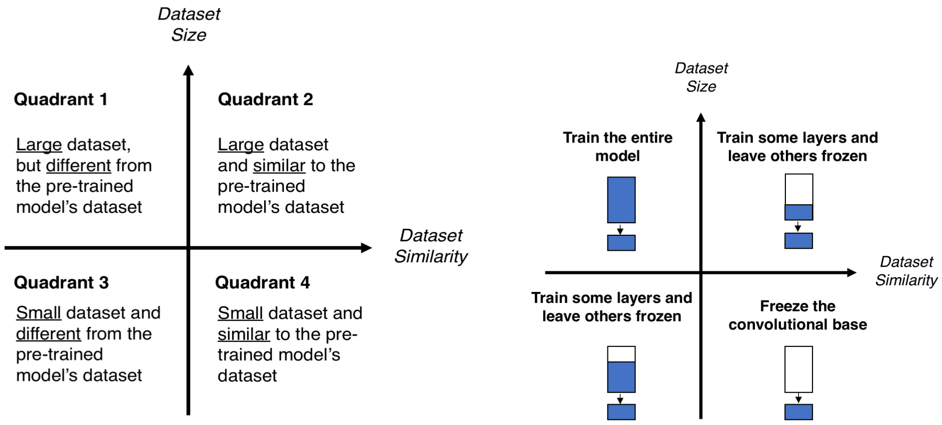
Dataset augmentation gave us two advantages: simulating having a larger dataset and avoiding the overfitting problem that we expected to face.

For dataset merging ,we were able to find another dataset with the same data as our original dataset.By merging the two together (this merging done on the training data only leaving the original test set as it is) ,we increased the dataset size which increased our models performance.

Next step is to train our models the processes can be summarized as as follows :

1.Selecting the pretrained models to use which are ResNet ,VGG16 and Xception

2.Classifing the problem based on the size similarity matrix.for our dataset which is small in size and different from the pretrained model data set ,Our problem falls within quadrant 3 in the figure below.


Dataset Size
Quadrant 1
Large dataset, but different from the pre-trained model's dataset
Quadrant 2
Large dataset and similar to the pre-trained model's dataset
Dataset Similarity
Quadrant 3
Small dataset and different from the pre-trained model's dataset
Quadrant 4
Small dataset and similar to the pre-trained model's dataset
Dataset Size
Train the entire model
Train some layers and leave others frozen
Dataset Similarity
Train some layers and leave others frozen
Freeze the convolutional base

3.based on the above step ,we will train some layers leaving the rest of layers frozen

4.Building our classifier which composed of 3 fully connected layer followed by softmax activation layer

5.Dataset augmentation has been done using Keras preprocessing Image data generator for each model.We applied the following data augmentation techniques

- width_shift_range=0.4
- height_shift_range=0.3
- rotation_range=90

We tried different values for each of them to get the best accuracy

6.Apply Zooming range=.2 to the image as we noticed that most of the information about the disease is in the middle of the images with a lot of useless information in the rest of the image.

7-As expected and due to the model complexity and the size of our dataset ,overfitting happens so we used two combined techniques from L2 Regularization and Dropout to reduce its effect.

8.fine Tuning our model with the following hyperparameters to tune:

- Number of frozen to active layers
- Learning rate

- Number of layers and neurons of our classifier
- Dropout rate
- L2 regularization penalty

for Number of frozen layers ,we have tried to unfrozen different numbers of layers starting with 10 layers to 40 layer in the very deep model xception

- learning rate trying .01 ,.001 .0001
- Number of layers our classifier ,trying 5 ,4,3 connected layers
- Number of neurons of our classifier tying decreasing number with each layer such as 1024 ,512,256
- Dropout rate : .2 , .5 , .7
- L2 regularization penalty .01 ,.1 , .5

9.Evaluting our model which done by evaluating the accuracy on the test set and by classification report and confusion matrix

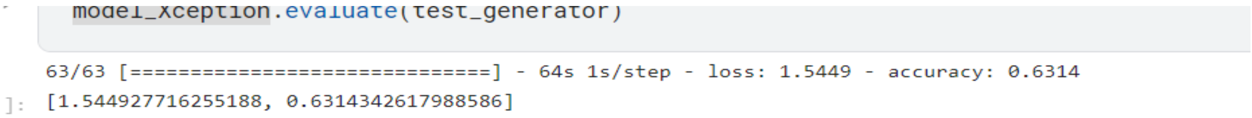
Results

We tried multiple tuning as mentioned above in the methodology.

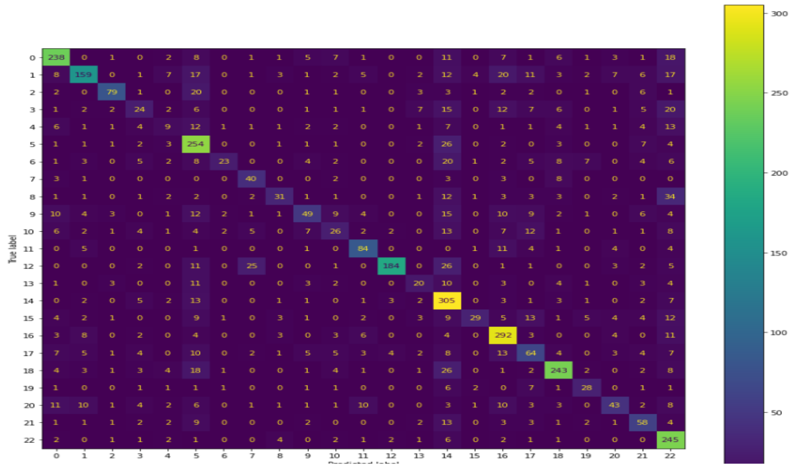
1. **Xception model**

- First, we have model_Xception with model base layers from 30 to 60 are trainable with dropout rate (0.7) and learning rate .0001

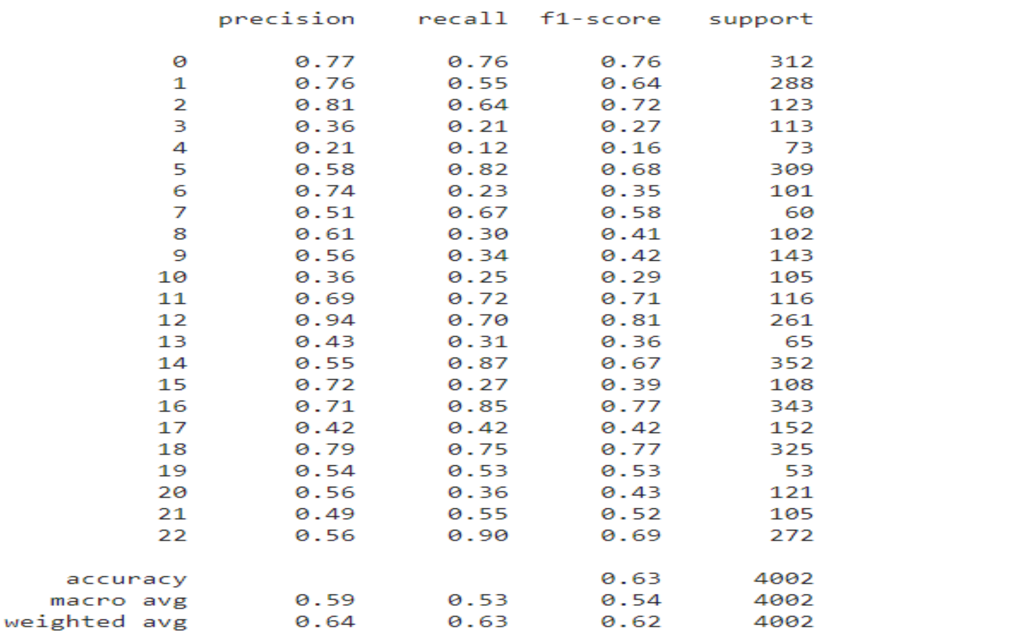
Test accuracy =.6314



```
model_xception.evaluate(test_generator)

63/63 [==============================] - 64s 1s/step - loss: 1.5449 - accuracy: 0.6314
[1.544927716255188, 0.6314342617988586]
```



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0	238	0	1	0	2	8	0	1	1	5	7	1	0	0	11	0	7	1	6	1	3	1	18
1	8	159	0	1	7	17	0	1	3	1	2	5	0	2	12	4	20	11	3	2	7	6	17
2	2	0	79	1	0	20	0	0	0	1	1	0	0	3	3	1	2	2	0	1	0	6	1
3	1	2	2	24	2	6	0	0	0	1	1	1	0	7	15	0	12	7	6	0	1	5	20
4	6	1	1	4	9	12	1	1	1	2	2	0	0	1	7	0	1	1	4	1	1	4	13
5	1	1	1	2	3	254	0	0	1	1	1	0	0	2	26	0	2	0	3	0	0	7	4
6	1	3	0	5	2	8	23	0	0	4	2	0	0	0	20	1	2	5	8	7	0	4	6
7	3	1	0	0	0	0	0	40	0	0	2	0	0	0	3	0	3	0	8	0	0	0	0
8	1	1	0	1	2	2	0	2	31	1	1	0	0	1	12	1	3	3	3	0	2	1	34
9	10	4	3	0	1	12	2	1	1	49	9	4	0	0	15	0	10	9	2	1	0	6	4
10	6	2	1	4	1	4	2	5	0	7	26	2	2	0	13	0	7	12	1	0	1	1	8
11	0	5	0	0	0	1	0	0	0	1	0	84	0	0	0	1	11	4	1	0	4	0	4
12	0	0	0	2	0	11	0	25	0	0	1	0	184	0	26	0	1	1	0	0	3	2	5
13	1	0	3	0	0	11	0	0	0	3	2	0	0	20	10	0	3	0	4	1	0	3	4
14	0	2	0	5	2	13	0	0	1	1	0	1	3	2	305	0	3	1	3	1	0	2	7
15	4	2	1	0	0	9	1	0	3	1	0	2	0	3	9	29	5	13	1	5	4	4	12
16	3	8	0	2	0	4	0	0	3	0	3	6	0	0	4	0	292	3	0	0	4	0	11
17	7	5	1	4	0	10	0	2	1	5	5	3	4	2	8	0	13	64	4	0	3	4	7
18	4	3	1	3	4	18	1	0	0	1	4	1	0	1	26	0	1	2	243	2	0	2	8
19	1	0	0	1	1	1	1	0	0	1	1	0	0	0	6	2	0	7	1	28	0	1	1
20	11	10	1	4	2	6	0	1	1	1	1	10	0	0	3	1	10	3	3	0	43	2	8
21	1	1	1	2	2	9	0	0	0	2	0	0	0	2	13	0	3	3	1	2	1	58	4
22	2	0	1	1	2	1	0	0	4	0	2	1	2	1	6	0	2	1	1	0	0	0	245

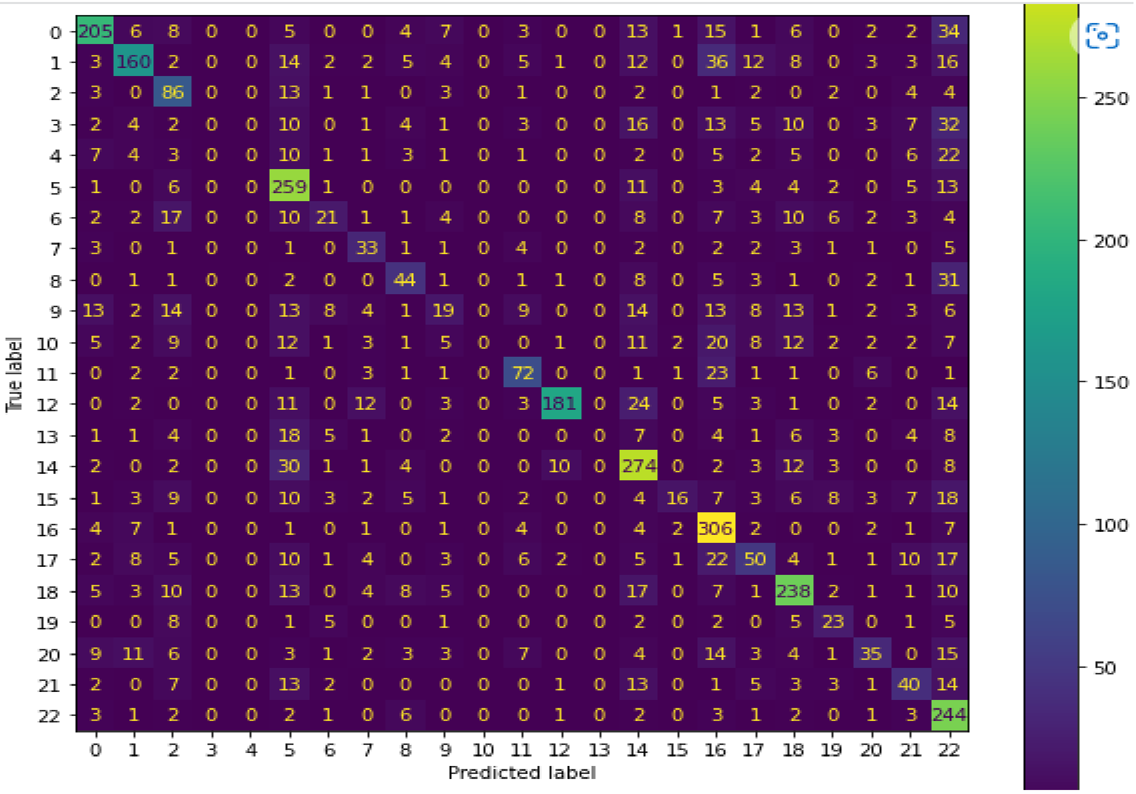


```
              precision    recall  f1-score   support

           0       0.77      0.76      0.76       312
           1       0.76      0.55      0.64       288
           2       0.81      0.64      0.72       123
           3       0.36      0.21      0.27       113
           4       0.21      0.12      0.16        73
           5       0.58      0.82      0.68       309
           6       0.74      0.23      0.35       101
           7       0.51      0.67      0.58        60
           8       0.61      0.30      0.41       102
           9       0.56      0.34      0.42       143
          10       0.36      0.25      0.29       105
          11       0.69      0.72      0.71       116
          12       0.94      0.70      0.81       261
          13       0.43      0.31      0.36        65
          14       0.55      0.87      0.67       352
          15       0.72      0.27      0.39       108
          16       0.71      0.85      0.77       343
          17       0.42      0.42      0.42       152
          18       0.79      0.75      0.77       325
          19       0.54      0.53      0.53        53
          20       0.56      0.36      0.43       121
          21       0.49      0.55      0.52       105
          22       0.56      0.90      0.69       272

    accuracy                           0.63      4002
   macro avg       0.59      0.53      0.54      4002
weighted avg       0.64      0.63      0.62      4002
```

- Xception model with L2 penalty =.001 and dropout rate 0.5 (rest of parameters are the same as above),model base layers from 30 to 60 are trainable

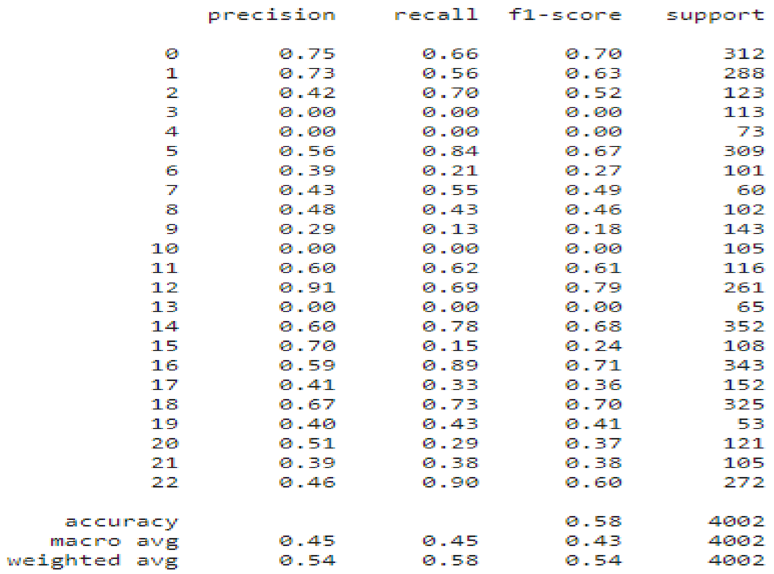
  ->Test accuracy =.5762

  ->Confusion Matrix :



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0	205	6	8	0	0	5	0	0	4	7	0	3	0	0	13	1	15	1	6	0	2	2	34
1	3	160	2	0	0	14	2	2	5	4	0	5	1	0	12	0	36	12	8	0	3	3	16
2	3	0	86	0	0	13	1	1	0	3	0	1	0	0	2	0	1	2	0	2	0	4	4
3	2	4	2	0	0	10	0	1	4	1	0	3	0	0	16	0	13	5	10	0	3	7	32
4	7	4	3	0	0	10	1	1	3	1	0	1	0	0	2	0	5	2	5	0	0	6	22
5	1	0	6	0	0	259	1	0	0	0	0	0	0	0	11	0	3	4	4	2	0	5	13
6	2	2	17	0	0	10	21	1	1	4	0	0	0	0	8	0	7	3	10	6	2	3	4
7	3	0	1	0	0	1	0	33	1	1	0	4	0	0	2	0	2	2	3	1	1	0	5
8	0	1	1	0	0	2	0	0	44	1	0	1	1	0	8	0	5	3	1	0	2	1	31
9	13	2	14	0	0	13	8	4	1	19	0	9	0	0	14	0	13	8	13	1	2	3	6
10	5	2	9	0	0	12	1	3	1	5	0	0	1	0	11	2	20	8	12	2	2	2	7
11	0	2	2	0	0	1	0	3	1	1	0	72	0	0	1	1	23	1	1	0	6	0	1
12	0	2	0	0	0	11	0	12	0	3	0	3	181	0	24	0	5	3	1	0	2	0	14
13	1	1	4	0	0	18	5	1	0	2	0	0	0	0	7	0	4	1	6	3	0	4	8
14	2	0	2	0	0	30	1	1	4	0	0	0	10	0	274	0	2	3	12	3	0	0	8
15	1	3	9	0	0	10	3	2	5	1	0	2	0	0	4	16	7	3	6	8	3	7	18
16	4	7	1	0	0	1	0	1	0	1	0	4	0	0	4	2	306	2	0	0	2	1	7
17	2	8	5	0	0	10	1	4	0	3	0	6	2	0	5	1	22	50	4	1	1	10	17
18	5	3	10	0	0	13	0	4	8	5	0	0	0	0	17	0	7	1	238	2	1	1	10
19	0	0	8	0	0	1	5	0	0	1	0	0	0	0	2	0	2	0	5	23	0	1	5
20	9	11	6	0	0	3	1	2	3	3	0	7	0	0	4	0	14	3	4	1	35	0	15
21	2	0	7	0	0	13	2	0	0	0	0	0	1	0	13	0	1	5	3	3	1	40	14
22	3	1	2	0	0	2	1	0	6	0	0	0	1	0	2	0	3	1	2	0	1	3	244

Classification report :



```
              precision    recall  f1-score   support

           0       0.75      0.66      0.70       312
           1       0.73      0.56      0.63       288
           2       0.42      0.70      0.52       123
           3       0.00      0.00      0.00       113
           4       0.00      0.00      0.00        73
           5       0.56      0.84      0.67       309
           6       0.39      0.21      0.27       101
           7       0.43      0.55      0.49        60
           8       0.48      0.43      0.46       102
           9       0.29      0.13      0.18       143
          10       0.00      0.00      0.00       105
          11       0.60      0.62      0.61       116
          12       0.91      0.69      0.79       261
          13       0.00      0.00      0.00        65
          14       0.60      0.78      0.68       352
          15       0.70      0.15      0.24       108
          16       0.59      0.89      0.71       343
          17       0.41      0.33      0.36       152
          18       0.67      0.73      0.70       325
          19       0.40      0.43      0.41        53
          20       0.51      0.29      0.37       121
          21       0.39      0.38      0.38       105
          22       0.46      0.90      0.60       272

    accuracy                           0.58      4002
   macro avg       0.45      0.45      0.43      4002
weighted avg       0.54      0.58      0.54      4002
```

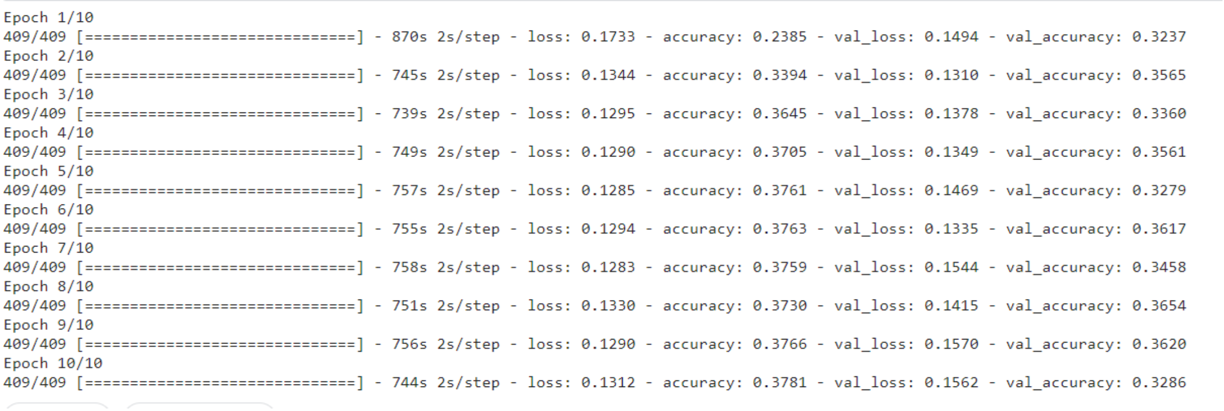
- Second, L2 Regularization (penalty= .01) added to xception model with the same parameters as above.
  - Test accuracy ~0.5480

Comments on Xception model Performance :

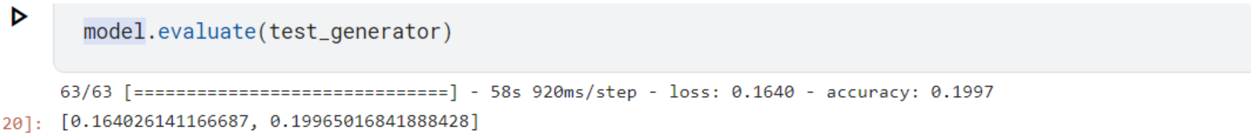
- Evaluation the model with F1 score ,precision and recall is necessary to get the true performance of our model.
- With L2 regularization combined with dropout ,the overfitting problem is less obvious but still the test accuracy does not increase
- With xception being very deep complex model combined with our small data set ,L2 doesn't seem to do much to the test accuracy as the overfitting problem still exist.
- In terms of these test accuracy , F1 ,precision and recall scores.we could say xception model provide good performance compared to state of the art
- We tried L2 penalties .01 ,.1,.5 but none of them give good accuracy so we settled on using only dropout
- So L2 with dropout seems to over penalize the model ,final decision to use only Dropout with 3 layers in classifier

2. VGG16 model

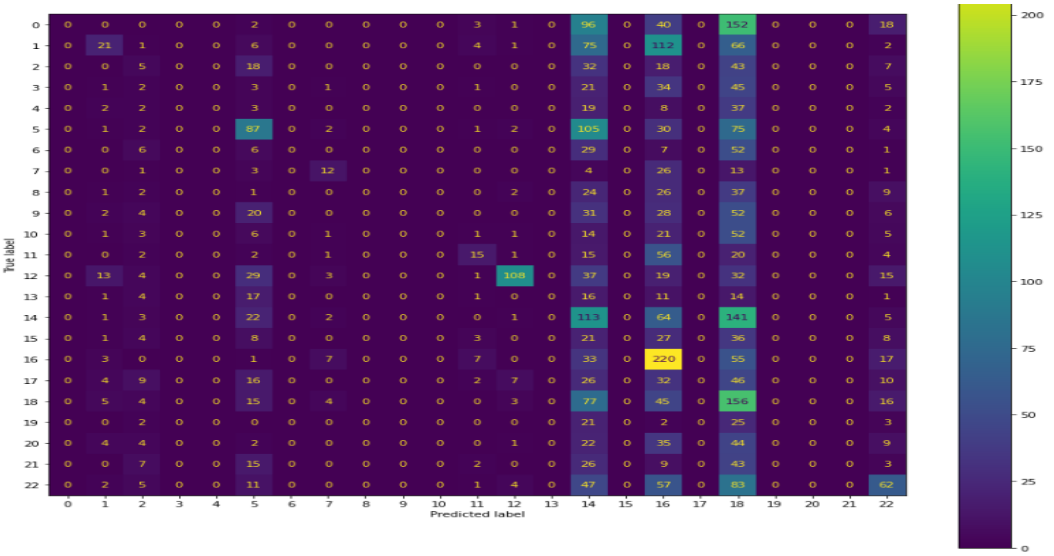
- we have VGG16 with model base layers from 8 to 14 are trainable with dropout and learning rate .0001



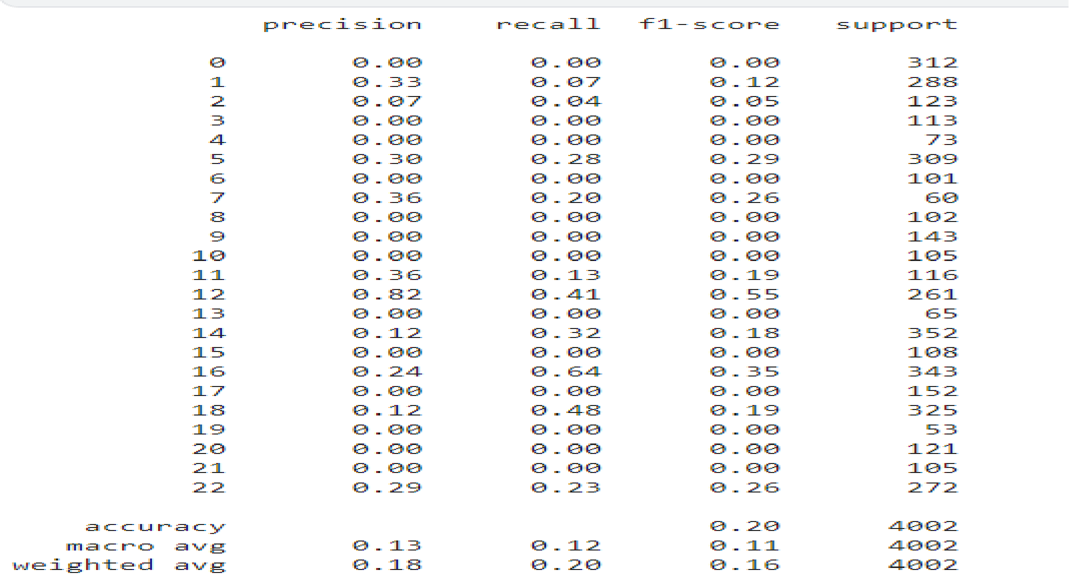
```
Epoch 1/10
409/409 [==============================] - 870s 2s/step - loss: 0.1733 - accuracy: 0.2385 - val_loss: 0.1494 - val_accuracy: 0.3237
Epoch 2/10
409/409 [==============================] - 745s 2s/step - loss: 0.1344 - accuracy: 0.3394 - val_loss: 0.1310 - val_accuracy: 0.3565
Epoch 3/10
409/409 [==============================] - 739s 2s/step - loss: 0.1295 - accuracy: 0.3645 - val_loss: 0.1378 - val_accuracy: 0.3360
Epoch 4/10
409/409 [==============================] - 749s 2s/step - loss: 0.1290 - accuracy: 0.3705 - val_loss: 0.1349 - val_accuracy: 0.3561
Epoch 5/10
409/409 [==============================] - 757s 2s/step - loss: 0.1285 - accuracy: 0.3761 - val_loss: 0.1469 - val_accuracy: 0.3279
Epoch 6/10
409/409 [==============================] - 755s 2s/step - loss: 0.1294 - accuracy: 0.3763 - val_loss: 0.1335 - val_accuracy: 0.3617
Epoch 7/10
409/409 [==============================] - 758s 2s/step - loss: 0.1283 - accuracy: 0.3759 - val_loss: 0.1544 - val_accuracy: 0.3458
Epoch 8/10
409/409 [==============================] - 751s 2s/step - loss: 0.1330 - accuracy: 0.3730 - val_loss: 0.1415 - val_accuracy: 0.3654
Epoch 9/10
409/409 [==============================] - 756s 2s/step - loss: 0.1290 - accuracy: 0.3766 - val_loss: 0.1570 - val_accuracy: 0.3620
Epoch 10/10
409/409 [==============================] - 744s 2s/step - loss: 0.1312 - accuracy: 0.3781 - val_loss: 0.1562 - val_accuracy: 0.3286
```



```
model.evaluate(test_generator)

63/63 [==============================] - 58s 920ms/step - loss: 0.1640 - accuracy: 0.1997
20]: [0.164026141166687, 0.19965016841888428]
```



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0	0	0	0	0	0	2	0	0	0	0	0	3	1	0	96	0	40	0	152	0	0	0	18
1	0	21	1	0	0	6	0	0	0	0	0	4	1	0	75	0	112	0	66	0	0	0	2
2	0	0	5	0	0	18	0	0	0	0	0	0	0	0	32	0	18	0	43	0	0	0	7
3	0	1	2	0	0	3	0	1	0	0	0	1	0	0	21	0	34	0	45	0	0	0	5
4	0	2	2	0	0	3	0	0	0	0	0	0	0	0	19	0	8	0	37	0	0	0	2
5	0	1	2	0	0	87	0	2	0	0	0	1	2	0	105	0	30	0	75	0	0	0	4
6	0	0	6	0	0	6	0	0	0	0	0	0	0	0	29	0	7	0	52	0	0	0	1
7	0	0	1	0	0	3	0	12	0	0	0	0	0	0	4	0	26	0	13	0	0	0	1
8	0	1	2	0	0	1	0	0	0	0	0	0	2	0	24	0	26	0	37	0	0	0	9
9	0	2	4	0	0	20	0	0	0	0	0	0	0	0	31	0	28	0	52	0	0	0	6
10	0	1	3	0	0	6	0	1	0	0	0	1	1	0	14	0	21	0	52	0	0	0	5
11	0	0	2	0	0	2	0	1	0	0	0	15	1	0	15	0	56	0	20	0	0	0	4
12	0	13	4	0	0	29	0	3	0	0	0	1	108	0	37	0	19	0	32	0	0	0	15
13	0	1	4	0	0	17	0	0	0	0	0	1	0	0	16	0	11	0	14	0	0	0	1
14	0	1	3	0	0	22	0	2	0	0	0	0	1	0	113	0	64	0	141	0	0	0	5
15	0	1	4	0	0	8	0	0	0	0	0	3	0	0	21	0	27	0	36	0	0	0	8
16	0	3	0	0	0	1	0	7	0	0	0	7	0	0	33	0	220	0	55	0	0	0	17
17	0	4	9	0	0	16	0	0	0	0	0	2	7	0	26	0	32	0	46	0	0	0	10
18	0	5	4	0	0	15	0	4	0	0	0	0	3	0	77	0	45	0	156	0	0	0	16
19	0	0	2	0	0	0	0	0	0	0	0	0	0	0	21	0	2	0	25	0	0	0	3
20	0	4	4	0	0	2	0	0	0	0	0	0	1	0	22	0	35	0	44	0	0	0	9
21	0	0	7	0	0	15	0	0	0	0	0	2	0	0	26	0	9	0	43	0	0	0	3
22	0	2	5	0	0	11	0	0	0	0	0	1	4	0	47	0	57	0	83	0	0	0	62



```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       312
           1       0.33      0.07      0.12       288
           2       0.07      0.04      0.05       123
           3       0.00      0.00      0.00       113
           4       0.00      0.00      0.00        73
           5       0.30      0.28      0.29       309
           6       0.00      0.00      0.00       101
           7       0.36      0.20      0.26        60
           8       0.00      0.00      0.00       102
           9       0.00      0.00      0.00       143
          10       0.00      0.00      0.00       105
          11       0.36      0.13      0.19       116
          12       0.82      0.41      0.55       261
          13       0.00      0.00      0.00        65
          14       0.12      0.32      0.18       352
          15       0.00      0.00      0.00       108
          16       0.24      0.64      0.35       343
          17       0.00      0.00      0.00       152
          18       0.12      0.48      0.19       325
          19       0.00      0.00      0.00        53
          20       0.00      0.00      0.00       121
          21       0.00      0.00      0.00       105
          22       0.29      0.23      0.26       272

    accuracy                           0.20      4002
   macro avg       0.13      0.12      0.11      4002
weighted avg       0.18      0.20      0.16      4002
```

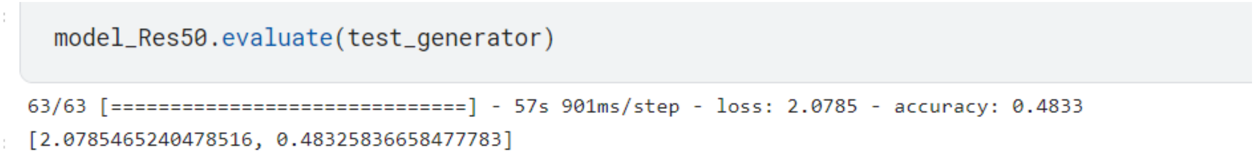
Comments on VGG16

- The performance of the Vgg16 model is not good with very low test accuracy and f1 score.
- Since VGG16 considered shallow if compared to Xception model ,the performance is expected compared to Xception and the following ResNet Model

- The main advantage of VGG16 model is less overfitting than Xception and Resnet.This could be justified also due to the vert less number of layers in VGG6
- Not expecting much from VGG for the above reasons

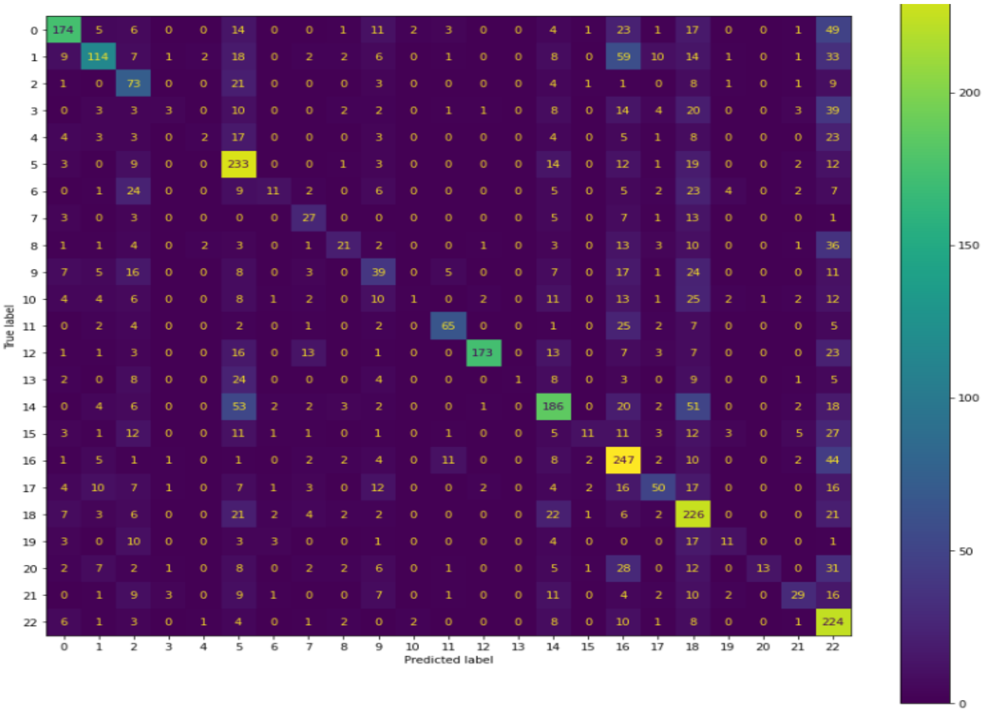
3. ResNet50

- we have Resnet50 with model base layers from 1 to 25are trainable with dropout rate by 0.7 and learning rate .0001

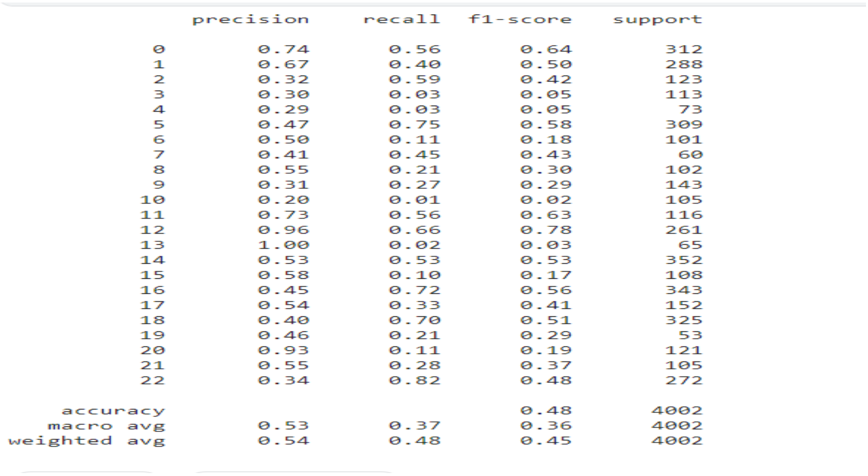


```
model_Res50.evaluate(test_generator)
```

```
63/63 [==============================] - 57s 901ms/step - loss: 2.0785 - accuracy: 0.4833
[2.0785465240478516, 0.48325836658477783]
```



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0	174	5	6	0	0	14	0	0	1	11	2	3	0	0	4	1	23	1	17	0	0	1	49
1	9	114	7	1	2	18	0	2	2	6	0	1	0	0	8	0	59	10	14	1	0	1	33
2	1	0	73	0	0	21	0	0	0	3	0	0	0	0	4	1	1	0	8	1	0	1	9
3	0	3	3	3	0	10	0	0	2	2	0	1	1	0	8	0	14	4	20	0	0	3	39
4	4	3	3	0	2	17	0	0	0	3	0	0	0	0	4	0	5	1	8	0	0	0	23
5	3	0	9	0	0	233	0	0	1	3	0	0	0	0	14	0	12	1	19	0	0	2	12
6	0	1	24	0	0	9	11	2	0	6	0	0	0	0	5	0	5	2	23	4	0	2	7
7	3	0	3	0	0	0	0	27	0	0	0	0	0	0	5	0	7	1	13	0	0	0	1
8	1	1	4	0	2	3	0	1	21	2	0	0	1	0	3	0	13	3	10	0	0	1	36
9	7	5	16	0	0	8	0	3	0	39	0	5	0	0	7	0	17	1	24	0	0	0	11
10	4	4	6	0	0	8	1	2	0	10	1	0	2	0	11	0	13	1	25	2	1	2	12
11	0	2	4	0	0	2	0	1	0	2	0	65	0	0	1	0	25	2	7	0	0	0	5
12	1	1	3	0	0	16	0	13	0	1	0	0	173	0	13	0	7	3	7	0	0	0	23
13	2	0	8	0	0	24	0	0	0	4	0	0	0	1	8	0	3	0	9	0	0	1	5
14	0	4	6	0	0	53	2	2	3	2	0	0	1	0	186	0	20	2	51	0	0	2	18
15	3	1	12	0	0	11	1	1	0	1	0	1	0	0	5	11	11	3	12	3	0	5	27
16	1	5	1	1	0	1	0	2	2	4	0	11	0	0	8	2	247	2	10	0	0	2	44
17	4	10	7	1	0	7	1	3	0	12	0	0	2	0	4	2	16	50	17	0	0	0	16
18	7	3	6	0	0	21	2	4	2	2	0	0	0	0	22	1	6	2	226	0	0	0	21
19	3	0	10	0	0	3	3	0	0	1	0	0	0	0	4	0	0	0	17	11	0	0	1
20	2	7	2	1	0	8	0	2	2	6	0	1	0	0	5	1	28	0	12	0	13	0	31
21	0	1	9	3	0	9	1	0	0	7	0	1	0	0	11	0	4	2	10	2	0	29	16
22	6	1	3	0	1	4	0	1	2	0	2	0	0	0	8	0	10	1	8	0	0	1	224

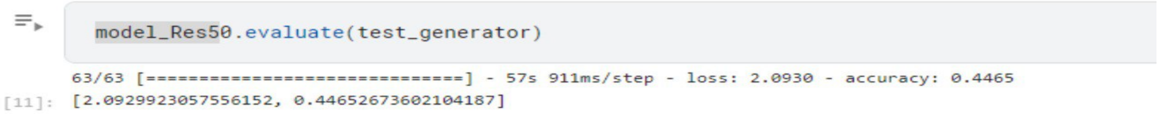


```
              precision    recall  f1-score   support

           0       0.74      0.56      0.64       312
           1       0.67      0.40      0.50       288
           2       0.32      0.59      0.42       123
           3       0.30      0.03      0.05       113
           4       0.29      0.03      0.05        73
           5       0.47      0.75      0.58       309
           6       0.50      0.11      0.18       101
           7       0.41      0.45      0.43        60
           8       0.55      0.21      0.30       102
           9       0.31      0.27      0.29       143
          10       0.20      0.01      0.02       105
          11       0.73      0.56      0.63       116
          12       0.96      0.66      0.78       261
          13       1.00      0.02      0.03        65
          14       0.53      0.53      0.53       352
          15       0.58      0.10      0.17       108
          16       0.45      0.72      0.56       343
          17       0.54      0.33      0.41       152
          18       0.40      0.70      0.51       325
          19       0.46      0.21      0.29        53
          20       0.93      0.11      0.19       121
          21       0.55      0.28      0.37       105
          22       0.34      0.82      0.48       272

    accuracy                           0.48      4002
   macro avg       0.53      0.37      0.36      4002
weighted avg       0.54      0.48      0.45      4002
```

ResNet50 (another trial)

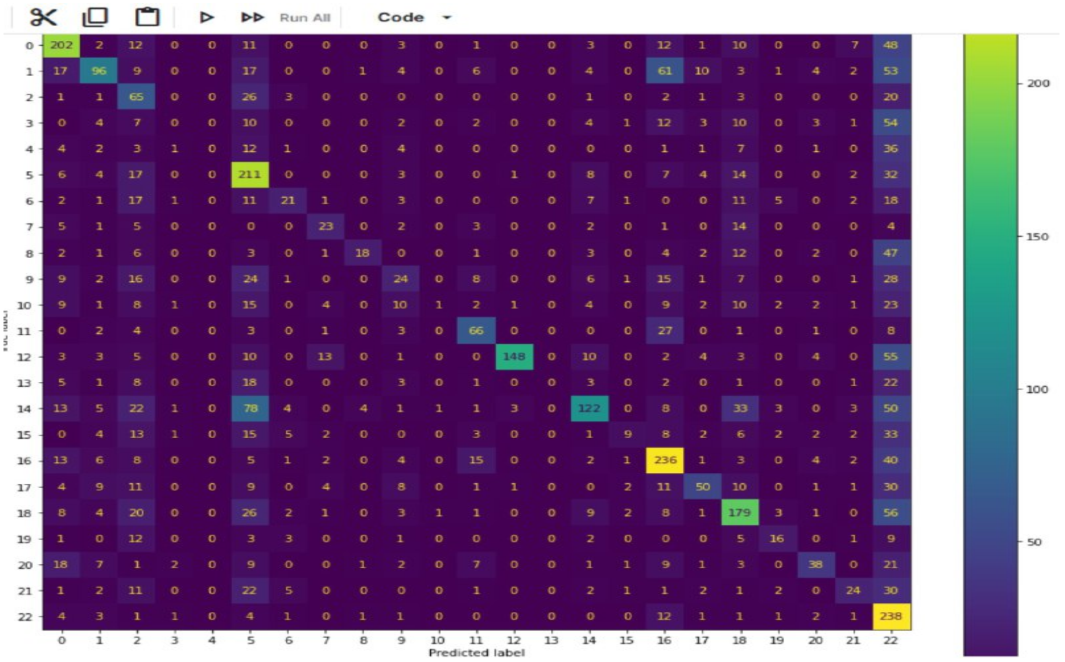
- we have Resnet50 with model base layers from 25 to 45 are trainable with dropout rate 0.5 and learning rate .0001

  Test accuracy=.4465



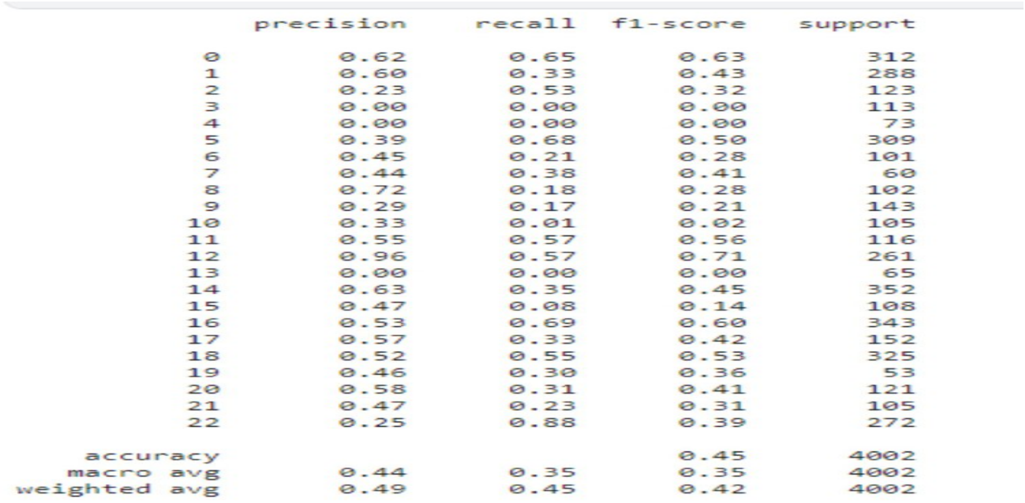
```
model_Res50.evaluate(test_generator)

63/63 [==============================] - 57s 911ms/step - loss: 2.0930 - accuracy: 0.4465
[11]: [2.0929923057556152, 0.44652673602104187]
```

Confusion matrix



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0	202	2	12	0	0	11	0	0	0	3	0	1	0	0	3	0	12	1	10	0	0	7	48
1	17	96	9	0	0	17	0	0	1	4	0	6	0	0	4	0	61	10	3	1	4	2	53
2	1	1	65	0	0	26	3	0	0	0	0	0	0	0	1	0	2	1	3	0	0	0	20
3	0	4	7	0	0	10	0	0	0	2	0	2	0	0	4	1	12	3	10	0	3	1	54
4	4	2	3	1	0	12	1	0	0	4	0	0	0	0	0	0	1	1	7	0	1	0	36
5	6	4	17	0	0	211	0	0	0	3	0	0	1	0	8	0	7	4	14	0	0	2	32
6	2	1	17	1	0	11	21	1	0	3	0	0	0	0	7	1	0	0	11	5	0	2	18
7	5	1	5	0	0	0	0	23	0	2	0	3	0	0	2	0	1	0	14	0	0	0	4
8	2	1	6	0	0	3	0	1	18	0	0	1	0	0	3	0	4	2	12	0	2	0	47
9	9	2	16	0	0	24	1	0	0	24	0	8	0	0	6	1	15	1	7	0	0	1	28
10	9	1	8	1	0	15	0	4	0	10	1	2	1	0	4	0	9	2	10	2	2	1	23
11	0	2	4	0	0	3	0	1	0	3	0	66	0	0	0	0	27	0	1	0	1	0	8
12	3	3	5	0	0	10	0	13	0	1	0	0	148	0	10	0	2	4	3	0	4	0	55
13	5	1	8	0	0	18	0	0	0	3	0	1	0	0	3	0	2	0	1	0	0	1	22
14	13	5	22	1	0	78	4	0	4	1	1	1	3	0	122	0	8	0	33	3	0	3	50
15	0	4	13	1	0	15	5	2	0	0	0	3	0	0	1	9	8	2	6	2	2	2	33
16	13	6	8	0	0	5	1	2	0	4	0	15	0	0	2	1	236	1	3	0	4	2	40
17	4	9	11	0	0	9	0	4	0	8	0	1	1	0	0	2	11	50	10	0	1	1	30
18	8	4	20	0	0	26	2	1	0	3	1	1	0	0	9	2	8	1	179	3	1	0	56
19	1	0	12	0	0	3	3	0	0	1	0	0	0	0	2	0	0	0	5	16	0	1	9
20	18	7	1	2	0	9	0	0	1	2	0	7	0	0	1	1	9	1	3	0	38	0	21
21	1	2	11	0	0	22	5	0	0	0	0	1	0	0	2	1	1	2	1	2	0	24	30
22	4	3	1	1	0	4	1	0	1	1	0	0	0	0	0	0	12	1	1	1	2	1	238

Classification Report



```
              precision    recall  f1-score   support

           0       0.62      0.65      0.63       312
           1       0.60      0.33      0.43       288
           2       0.23      0.53      0.32       123
           3       0.00      0.00      0.00       113
           4       0.00      0.00      0.00        73
           5       0.39      0.68      0.50       309
           6       0.45      0.21      0.28       101
           7       0.44      0.38      0.41        60
           8       0.72      0.18      0.28       102
           9       0.29      0.17      0.21       143
          10       0.33      0.01      0.02       105
          11       0.55      0.57      0.56       116
          12       0.96      0.57      0.71       261
          13       0.00      0.00      0.00        65
          14       0.63      0.35      0.45       352
          15       0.47      0.08      0.14       108
          16       0.53      0.69      0.60       343
          17       0.57      0.33      0.42       152
          18       0.52      0.55      0.53       325
          19       0.46      0.30      0.36        53
          20       0.58      0.31      0.41       121
          21       0.47      0.23      0.31       105
          22       0.25      0.88      0.39       272

    accuracy                           0.45      4002
   macro avg       0.44      0.35      0.35      4002
weighted avg       0.49      0.45      0.42      4002
```

Comments on ResNet50

- ResNet50 gives relatively good accuracy ,very much higher than VGG but less than Xception
- It gives better than VGG as it is more deep and can extract more abstract features from the images
- Still ,Xception gives more accurate results more than ResNet

Final result

From above results of the three models ,Xception is the chosen model due to its performance on the test set and in terms of classification report and test 1 score

Parameters

- Dropout rate =0.7 in the classifier layer
- Learning rate=0.0001
- Model base layers from 30 to 60 are trained

Our final result

Test accuracy	F1 score	recall	Precision
.6314	.62	.63	.64

Comparing our results to testbenches and state of art solutions

Our final result

Test accuracy	F1 score	recall	Precision
.6314	.62	.63	.64

- These results are better than all the results on kaggle for dermnet dataset (of course we have the advantage of a little larger training set due to the merging)
- We could not find any paper (after a lot of search) concerning this particular dataset to compare with.
- Papers discussing similar problems (skin disease detection) achieve Accuracy of around ~80 and 85 % but almost all of them are either on one class detection or fewer classes than ours.

Ethical issues

The most noticeable ethical issues related to our model are :

- Fairness and biases of our model and implementation

  Because most of the images in our dataset are for white people ,we expect our model to be biased.This problem could be solved by providing data for skin diseases with black people images.
- There is an ethical issue towards other DL specialists. We take it into consideration because we have trained our model on other data to improve its accuracy by increasing its training dataset capacity. To be ethical, we tested our model by using the original testing dataset as it was.

Discussion

The Xception model is very sophisticated but very complex and deep at the same time.That is why it gives best accuracy out of the three models.

But being a very deep model with a small dataset ,an overfitting problem happens which we tried to reduce by some techniques such as dropout ,L2 regularization and reducing the number of output layers.

Then by noticing the performance ,we finally chose the parameter mentioned above as our final parameters for the Xception model.

For the performance of the model :

Our final result

Test accuracy	F1 score	recall	Precision
.6314	.62	.63	.64

This result is one of the best if compared to testbenches and state of the art.but there are some limitations to the performance of our model such as the size of the dataset is considerably small with 23 classes to predict.

The hard classification task of predicting 23 classes of skin diseases with very similar shapes which tend to confuse the model and resulting in misclassification.

The huge imbalance in the dataset ,which results in degrading the performance of the model.

Things to improve the model performance

- Increasing the size of dataset from a real source of possible (the solution to all problems)
- More Data Augmentation (to increase the size of dataset and
- Increasing the size of dataset by using techniques such as Generative Adversarial Networks (Gans) to generate new images to solve the problem of dataset size and the imbalance.
- We can enhance the image using approaches in computer vision to try to better images.
- Additionally, we can use SIFT to find the image features and then build a bag of words and train the features using deep learning.

Contribution of each Team member

- We all worked together in developing the notebook
- For the tunning ,each member took some cases to run and reporting the result
- For the results ,each one of us put it in the report
- For comments and discussion ,this done as a teamwork since we needed all our heads together

References

[1]J. Velasco, “A Smartphone-Based Skin Disease Classification Using MobileNet
CNN,” International Journal of Advanced Trends in Computer Science and Engineering,
pp. 2632–2637, Oct. 2019, doi: 10.30534/ijatcse/2019/116852019

[2]P. N. Srinivasu, J. G. SivaSai, M. F. Ijaz, A. K. Bhoi, W. Kim, and J. J. Kang,
“Classification of Skin Disease Using Deep Learning Neural Networks with MobileNet
V2 and LSTM,” Sensors, vol. 21, no. 8, p. 2852, Apr. 2021, doi: 10.3390/s21082852.