```
#importing all the requied modules

from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from PIL import Image
import glob
import cv2
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
from glob import glob
import seaborn as sns
from PIL import Image
np.random.seed(123)
from sklearn.preprocessing import label_binarize
from sklearn.metrics import confusion_matrix
import itertools

import keras
from keras.utils.np_utils import to_categorical # used for converting labels to one-hot-encoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras import backend as K
import itertools
from keras.utils.np_utils import to_categorical # convert to one-hot-encoding
```

```
import itertools
from keras.utils.np_utils import to_categorical # convert to one-hot-encoding

from keras.optimizers import adam_v2
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
from sklearn.model_selection import train_test_split
```

+ Code + Markdown

```
#import all the images from the folder
import glob
full_set=glob.glob('../input/skin-images/ham_data_surnamesABCD/ham_data_surnamesABCD/*.*')
```

```
#appending every image

full_set=[]
path=("../input/skin-images/ham_data_surnamesABCD/ham_data_surnamesABCD/*.*")
for i in glob.glob(path):
    s1=cv2.imread(i)
    full_set.append(s1)
```

```
#loading the csv file

nRowsRead = None
full_csv = pd.read_csv('../input/skin-csv/filename_to_category_map_surnamesABCD.csv', delimiter=',', nrows = nRow
```

```
#loading the csv file

nRowsRead = None
full_csv = pd.read_csv('../input/skin-csv/filename_to_category_map_surnamesABCD.csv', delimiter=',', nrows = nRow
full_csv.dataframeName = 'filename_to_category_map_surnamesABCD.csv'
n_Row, n_Col = full_csv.shape
print(f'There are {n_Row} rows and {n_Col} columns')
```

```
There are 5372 rows and 4 columns
```

```
full_csv_rep = pd.read_csv('../input/skin-csv/filename_to_category_map_surnamesABCD.csv')
```

```
print(full_csv)
```

```
         image_id                        cell_type  is_benign localization
0     ISIC_0027419  Benign keratosis-like lesions          1.0        scalp
1     ISIC_0026769  Benign keratosis-like lesions          1.0        scalp
2     ISIC_0031633  Benign keratosis-like lesions          1.0          ear
3     ISIC_0029176  Benign keratosis-like lesions          1.0         face
4     ISIC_0029068  Benign keratosis-like lesions          1.0         face
...            ...                           ...          ...          ...
5367  ISIC_0027265  Benign keratosis-like lesions          1.0          NaN
5368  ISIC_0025029                Melanocytic nevi         1.0          NaN
5369  ISIC_0029462                Melanocytic nevi         1.0          NaN
5370  ISIC_0030510                Melanocytic nevi         1.0          NaN
5371  ISIC_0033125                        Melanoma         1.0          NaN

[5372 rows x 4 columns]
```

```
len_csv = full_csv["image_id"]
print(len(len_csv))
map_images = {}
for i in len_csv:
    path = "../input/skin-images/ham_data_surnamesABCD/ham_data_surnamesABCD/" + i + ".jpg"
    if i not in map_images.keys():
        map_images[i] = path
```

```
5372
```

```
full_csv['path'] = full_csv['image_id'].map(lambda x : map_images[x])
```

```
full_csv['image'] = full_csv['path'].map(lambda x: np.asarray(Image.open(x).resize((100,75))))
```

```
full_csv = full_csv.dropna()
```

+ Code + Markdown

```
full_csv.head()
```

image_id	cell_type	is_benign	localization	path	image

```
full_csv.head()
```

	image_id	cell_type	is_benign	localization	path	image
0	ISIC_0027419	Benign keratosis-like lesions	1.0	scalp	../input/skin-images/ham_data_surnamesABCD/ham...	[[[190, 153, 194], [192, 154, 196], [191, 153,...
1	ISIC_0026769	Benign keratosis-like lesions	1.0	scalp	../input/skin-images/ham_data_surnamesABCD/ham...	[[[185, 127, 137], [189, 133, 147], [194, 136,...
2	ISIC_0031633	Benign keratosis-like lesions	1.0	ear	../input/skin-images/ham_data_surnamesABCD/ham...	[[[134, 90, 113], [147, 102, 125], [159, 115, ...
3	ISIC_0029176	Benign keratosis-like lesions	1.0	face	../input/skin-images/ham_data_surnamesABCD/ham...	[[[191, 146, 129], [192, 146, 133], [194, 145,...
4	ISIC_0029068	Benign keratosis-like lesions	1.0	face	../input/skin-images/ham_data_surnamesABCD/ham...	[[[149, 105, 85], [156, 114, 95], [163, 124, 1...

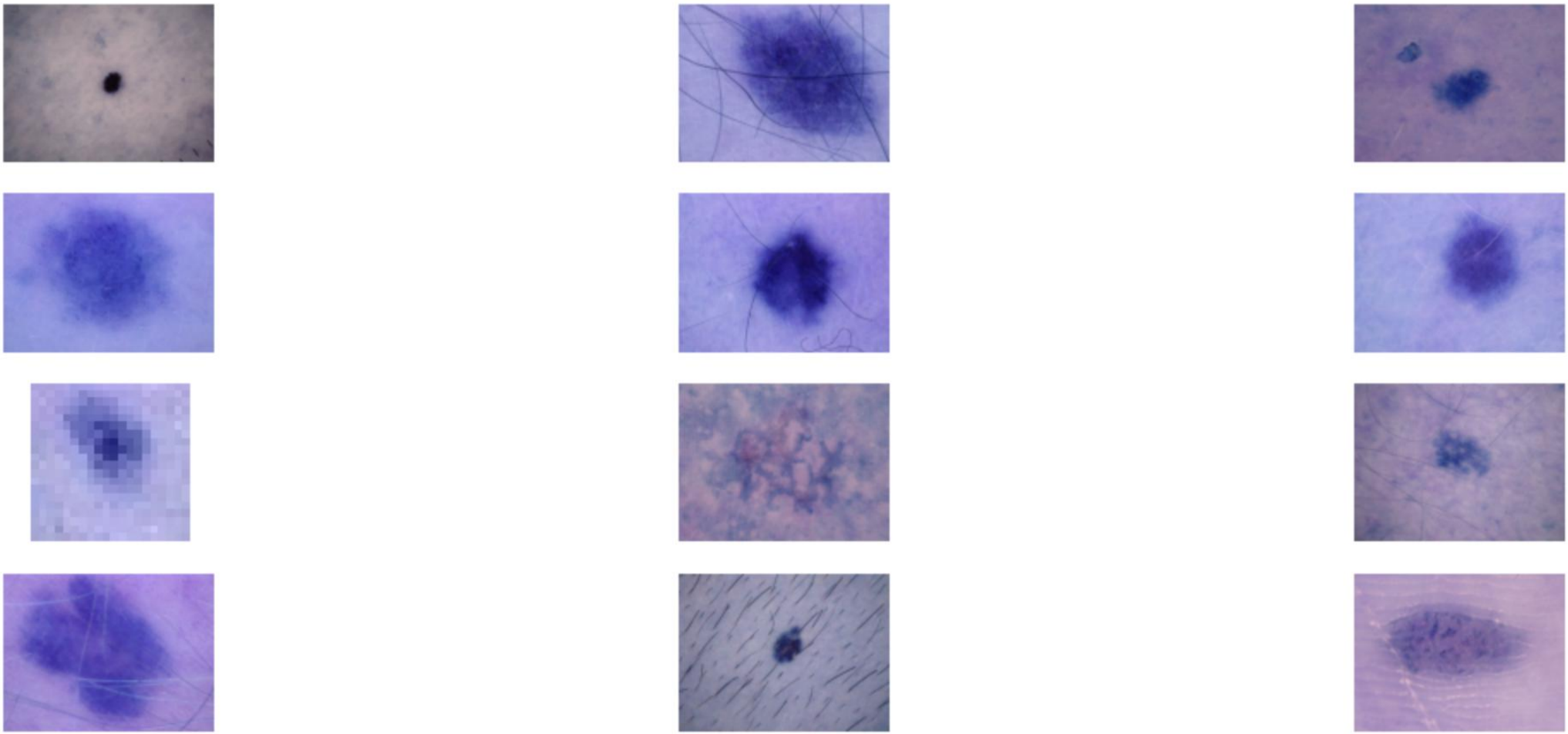
```
X=full_csv["image"]
Y=full_csv["cell_type"]
```

```
lass_name = ['Benign keratosis-like lesions ','Dermatofibroma','Melanoma','Vascular lesions','Basal cell carcinoma

lt.figure(figsize=(20,20))
or i in range(0,16) :
    plt.subplot(10,3,i+1)
    plt.axis('off')
```

```
class_name = ['Benign keratosis-like lesions ','Dermatofibroma','Melanoma','Vascular lesions','Basal cell carcino


plt.figure(figsize=(20,20))
for i in range(0,16) :
    plt.subplot(10,3,i+1)
    plt.axis('off')
    plt.imshow(full_set[i], cmap="gray_r")
```



```
X=full_csv["image"]
Y=full_csv["cell_type"]
```

```
print(X)
```

```
0       [[[190, 153, 194], [192, 154, 196], [191, 153,...
1       [[[185, 127, 137], [189, 133, 147], [194, 136,...
2       [[[134, 90, 113], [147, 102, 125], [159, 115, ...
3       [[[191, 146, 129], [192, 146, 133], [194, 145,...
4       [[[149, 105, 85], [156, 114, 95], [163, 124, 1...
                              ...
5253    [[[211, 135, 148], [211, 135, 148], [210, 134,...
5254    [[[243, 161, 173], [243, 161, 173], [244, 162,...
5255    [[[175, 162, 180], [176, 163, 180], [176, 163,...
5256    [[[49, 11, 25], [53, 15, 28], [59, 21, 34], [7...
5257    [[[213, 150, 167], [213, 150, 167], [213, 150,...
Name: image, Length: 5258, dtype: object
```

```
x_train_o, x_test_o, y_train_o, y_test_o = train_test_split(X, Y, test_size=0.30,random_state=6969)
```

```
x_train = np.asarray(x_train_o.tolist())
x_test = np.asarray(x_test_o.tolist())

x_train_mean = np.mean(x_train)
x_train_std = np.std(x_train)
```

```
x_train = (x_train - x_train_mean)/x_train_std
x_test = (x_test - x_test_mean)/x_test_std
```

+ Code + Markdown

```
y_train = to_categorical(np.asarray(y_train_o.factorize()[0]))
y_test = to_categorical(np.asarray(y_test_o.factorize()[0]))
```

```
train, x_validate, y_train, y_validate = train_test_split(x_train, y_train, test_size = 0.30, random_state = 69)
```

```
input_shape = (75, 100, 3)
num_classes = 7

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',padding = 'Same',input_shape=input_shape))
model.add(Conv2D(32,kernel_size=(3, 3), activation='relu',padding = 'Same',))
model.add(MaxPool2D(pool_size = (2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu',padding = 'Same'))
model.add(Conv2D(64, (3, 3), activation='relu',padding = 'Same'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.40))
```

```
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

```
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
_________________________________________________________________
2022-04-26 22:07:07.468892: I tensorflow/core/common_runtime/process_util.cc:146] Creating new thread pool with default in
```

```
optimizer = adam_v2.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
model.compile(optimizer = optimizer , loss = "categorical_crossentropy", metrics=["accuracy"])
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=3,
                                            verbose=1,
                                            factor=0.5,
                                            min_learning_rate=0.00001)
```

```
/opt/conda/lib/python3.7/site-packages/keras/optimizer_v2/optimizer_v2.py:356: UserWarning: The `lr` argument is deprecate
d, use `learning_rate` instead.
  "The `lr` argument is deprecated, use `learning_rate` instead.")
```

```
datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range = 0.1, # Randomly zoom image
        width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
        height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
        horizontal_flip=False,  # randomly flip images
        vertical_flip=False)  # randomly flip images

datagen.fit(x_train)
```

```
epochs = 50

batch_size = 10
history = model.fit_generator(datagen.flow(x_train,y_train, batch_size=batch_size),
                              epochs = epochs, validation_data = (x_validate,y_validate),
                              verbose = 1, steps_per_epoch=x_train.shape[0] // batch_size
                              , callbacks=[learning_rate_reduction])
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
```

```
/opt/conda/lib/python3.7/site-packages/keras/engine/training.py:1972: UserWarning: `Model.fit_generator` is deprecated and
will be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
2022-04-26 22:07:08.940893: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization
Passes are enabled (registered 2)
Epoch 1/50
257/257 [==============================] - 49s 187ms/step - loss: 1.1107 - accuracy: 0.6555 - val_loss: 0.9002 - val_accur
acy: 0.6875
Epoch 2/50
257/257 [==============================] - 47s 182ms/step - loss: 1.0054 - accuracy: 0.6598 - val_loss: 0.9273 - val_accur
acy: 0.6884
Epoch 3/50
257/257 [==============================] - 46s 180ms/step - loss: 0.9492 - accuracy: 0.6594 - val_loss: 0.8586 - val_accur
acy: 0.6866
Epoch 4/50
257/257 [==============================] - 47s 183ms/step - loss: 0.9186 - accuracy: 0.6703 - val_loss: 0.8442 - val_accur
acy: 0.6893
Epoch 5/50
257/257 [==============================] - 46s 180ms/step - loss: 0.9319 - accuracy: 0.6676 - val_loss: 0.8777 - val_accur
acy: 0.6920
Epoch 6/50
257/257 [==============================] - 47s 183ms/step - loss: 0.9097 - accuracy: 0.6746 - val_loss: 0.8204 - val_accur
acy: 0.7029
```