```python
from google.colab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```python
from numpy import mean
from numpy import std
from matplotlib import pyplot
from sklearn.model_selection import KFold
from keras.datasets import mnist
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.optimizers import SGD
from keras.layers import Dropout
from keras.layers import BatchNormalization
import keras
from keras import backend as K
import matplotlib.pyplot as plt
import sklearn
```

```python
path_normal = '/content/drive/MyDrive/Deep learning demo project/Normal/'
path_pneumonia = '/content/drive/MyDrive/Deep learning demo project/pneumonia/'
```

```python
##Import necessary libraries
import numpy as np
import PIL
import cv2
import os
data1 = list()
data2 = list()
x = list()
```

```python
##Class-1 images##

for image in os.walk(path_normal):
  data1.append(image[2])

for i in range(len(data1[0])):
  str_complete = path_normal + data1[0][i]
  img = cv2.imread(str_complete)
  img = cv2.resize(img, (224, 224))
  x.append(img)
  print(i)#Ensure all images are loaded
```

```
        676
        677
        678
        679
        680
        681
        682
        683
        684
        685
        686
        687
        688
        689
        690
        691
        692
        693
        694
        695
        696
        697
        698
        699
```

```
print(img.shape)
```

```
    (224, 224, 3)
```

##Class-2 images##

```
for image in os.walk(path_pneumonia):
  data2.append(image[2])

for i in range(len(data2[0])):
  str_complete = path_pneumonia + data2[0][i]
  img = cv2.imread(str_complete)
  img = cv2.resize(img, (224, 224))
  x.append(img)#Ensure all images are loaded
  print(i)
```

```
        688
        689
        690
        691
        692
        693
        694
        695
        696
        697
        698
        699
```

```python
data_x = np.asarray(x)
```

```python
data_x.shape
```

```
    (1400, 224, 224, 3)
```

```python
x=data_x
```

```python
y = np.zeros(1400)
y[:700] =1
y[700:1400]=2
```

```python
from sklearn.model_selection import train_test_split
```

```python
##Dataset Split##
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
#y = to_categorical(y)
x_train, x_test, y_train, y_test = train_test_split(data_x, y, test_size=0.2, random_state=1)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=1/8, random_state=1)
```

```python
y_tr_one_hot  = np.zeros((np.array(y_train).shape[0],2))

for i in range(np.array(y_train).shape[0]):
  label = y_train[i]-1
  y_tr_one_hot[i][int(label)] = 1
```

```python
y_val_one_hot  = np.zeros((np.array(y_val).shape[0],2))

for i in range(np.array(y_val).shape[0]):
  label = y_val[i]-1
  y_val_one_hot[i][int(label)] = 1
```

```python
y_te_one_hot  = np.zeros((np.array(y_test).shape[0],2))

for i in range(np.array(y_test).shape[0]):
  label = y_test[i]-1
  y_te_one_hot[i][int(label)] = 1
```

```python
from keras.models import load_model
from keras.layers import Lambda
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense
```

```python
model = tf.keras.applications.EfficientNetV2L(include_top=False,input_shape=(224,224,3))
# mark loaded layers as not trainable
for layer in model.layers:
    layer.trainable = False
# add new classifier layers
flat1 = Flatten()(model.layers[-1].output)
#x=Dense(1024,activation='relu')(flat1) # FC layer 1
#x=Dense(64,activation='relu')(x)   # FC layer 2
output = Dense(2, activation='softmax')(flat1)
model = Model(inputs=model.inputs, outputs=output)
```

```python
optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(x_train, y_tr_one_hot, validation_data=(x_val, y_val_one_hot), epochs=5, batch_size=200,verbose=1)
```

```
    Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/efficientnet_v2/efficientnetv2-l_noto
    473176280/473176280 [==============================] - 23s 0us/step
    Epoch 1/5
    5/5 [==============================] - 979s 195s/step - loss: 2.2888 - accuracy: 0.6959 - val_loss: 1.1767 - val_accuracy
    Epoch 2/5
```

```
5/5 [==============================] – 932s 192s/step – loss: 0.6654 – accuracy: 0.8969 – val_loss: 0.0407 – val_accuracy
Epoch 3/5
5/5 [==============================] – 924s 191s/step – loss: 0.0248 – accuracy: 0.9939 – val_loss: 3.9253e-07 – val_accu
Epoch 4/5
5/5 [==============================] – 935s 193s/step – loss: 6.0195e-04 – accuracy: 1.0000 – val_loss: 9.1005e-05 – val_
Epoch 5/5
5/5 [==============================] – 932s 193s/step – loss: 0.0102 – accuracy: 0.9980 – val_loss: 0.0011 – val_accuracy
<keras.callbacks.History at 0x7f0569860bb0>
```

```python
import sklearn
from sklearn.metrics import confusion_matrix
```

```python
test_loss, test_acc = model.evaluate(np.array(x_test), np.array(y_te_one_hot), verbose=0)
print(test_acc)
##Evaluating Sensitivity, Accuracy and Kappa scores
y_prob = model.predict(x_test)
Y_pred = y_prob.argmax(axis=-1)
```

```
0.9928571581840515
9/9 [==============================] – 216s 23s/step
```

```python
cm1 = confusion_matrix(y_test-1,Y_pred)
print("confusion matrix \n",cm1)
```

```
confusion matrix
 [[137   0]
 [  2 141]]
```

```python
from sklearn.metrics import classification_report
```

```python
import pandas as pd
```

```python
print(pd.DataFrame(classification_report(y_test-1,Y_pred,output_dict=True)).T)
Kappa=sklearn.metrics.cohen_kappa_score(y_test-1,Y_pred)
print('Kappa=',Kappa)
```

```
                 precision    recall  f1-score     support
0.0               0.985612  1.000000  0.992754  137.000000
1.0               1.000000  0.986014  0.992958  143.000000
accuracy          0.992857  0.992857  0.992857    0.992857
macro avg         0.992806  0.993007  0.992856  280.000000
weighted avg      0.992960  0.992857  0.992858  280.000000
Kappa= 0.9857120987906313
```

✓ 0s   completed at 14:22   ● ✕