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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import seaborn as sns
from keras.preprocessing.image import ImageDataGenerator
from sklearn.preprocessing import LabelEncoder
import albumentations as A
from tensorflow.keras.utils import Sequence
from sklearn.model_selection import train_test_split
import cv2
from google.colab.patches import cv2_imshow
import skimage.exposure
import os
```

```
cd /content/drive/MyDrive
```

```
/content/drive/MyDrive
```

```
!unzip "/content/drive/MyDrive/mammography_images.zip"
```

```
Show hidden output
```

```
train=pd.read_csv("/content/drive/MyDrive/mammography_images/Testing_set.csv")
test=pd.read_csv("/content/drive/MyDrive/mammography_images/Training_set.csv")
sub=pd.read_csv("/content/drive/MyDrive/mammography_images/sample_submission.csv")
```

```
%cd /content/drive/MyDrive/mammography_images/train
```

```
/content/drive/MyDrive/mammography_images/train
```

```

train.head()
```

	filename
0	Image_1.jpg
1	Image_2.jpg
2	Image_3.jpg
3	Image_4.jpg
4	Image_5.jpg

```
1
2 print("len of training set", len(train))
3 print("len of testing set", len(test))
```

```
len of training set 200
len of testing set 600
```

```
1 # Load the data from the CSV files
2 train = pd.read_csv("/content/drive/MyDrive/mammography_images/Training_set.csv")
3 test = pd.read_csv("/content/drive/MyDrive/mammography_images/Testing_set.csv")
4 sub = pd.read_csv("/content/drive/MyDrive/mammography_images/sample_submission.csv")
5
6 # Check if the 'train' variable has a column named 'label'
7 if 'label' not in train.columns:
8     raise ValueError("The 'train' variable does not have a column named 'label'")
9
10 # Print the number of rows in the training and testing sets
11 print("Number of rows in the training set:", len(train))
12 print("Number of rows in the testing set:", len(test))
13
14 # Print the value counts of the 'label' column in the training set
15 print(train["label"].value_counts())
```

```
Number of rows in the training set: 600
Number of rows in the testing set: 200
label
Density2Malignant    189
Density1Malignant    181
Density3Benign        64
Density1Benign        58
Density3Malignant     46
Density4Benign        42
Density2Benign        13
```

```
Density4Malignant      7
Name: count, dtype: int64
```

```
print(train.columns)

```

```
Index(['filename'], dtype='object')
```

```
print(train.head())

```

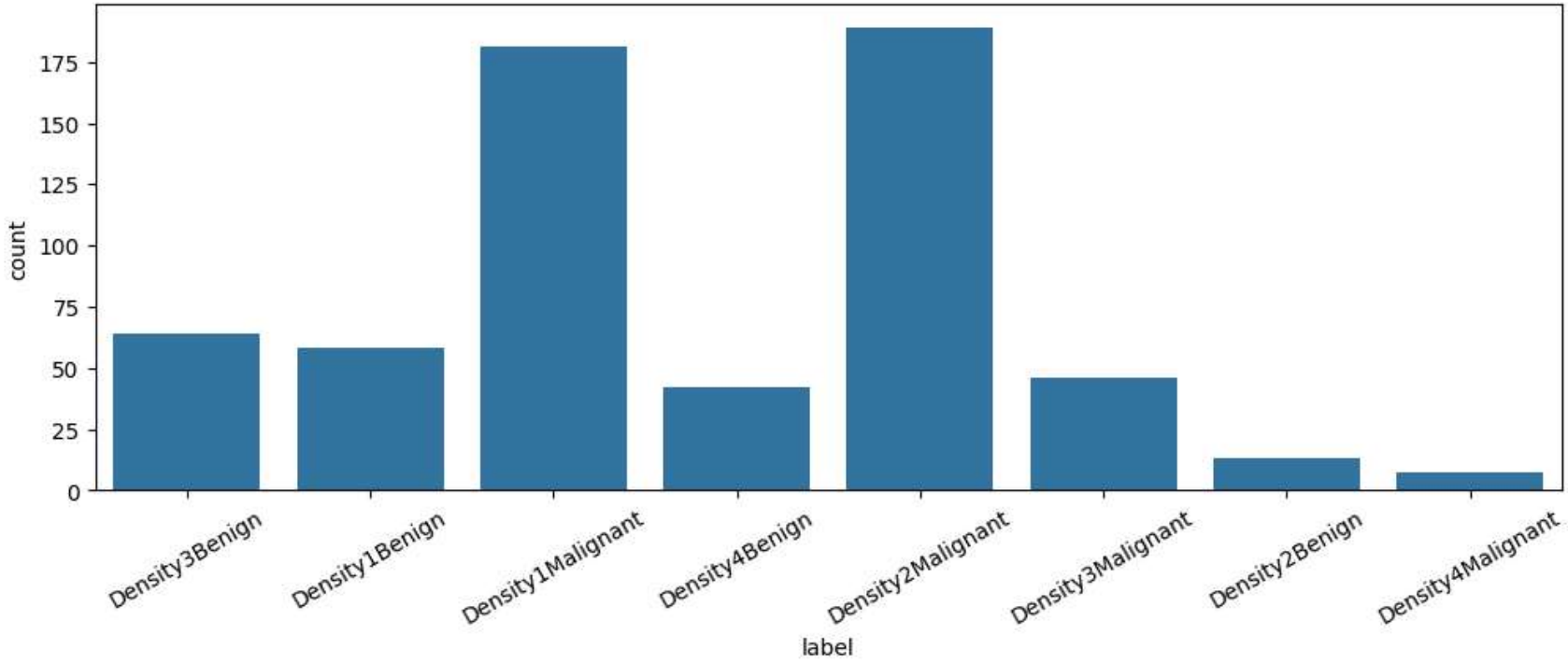
```
       filename
0  Image_1.jpg
1  Image_2.jpg
2  Image_3.jpg
3  Image_4.jpg
4  Image_5.jpg
```

```
cd /content/drive/MyDrive
```

```
/content/drive/MyDrive
```

```
plt.figure(figsize=(12,4))
sns.countplot(x=train["label"],data=train,)
plt.xticks(rotation=30)
```

```
([0, 1, 2, 3, 4, 5, 6, 7],
 [Text(0, 0, 'Density3Benign'),
  Text(1, 0, 'Density1Benign'),
  Text(2, 0, 'Density1Malignant'),
  Text(3, 0, 'Density4Benign'),
  Text(4, 0, 'Density2Malignant'),
  Text(5, 0, 'Density3Malignant'),
  Text(6, 0, 'Density2Benign'),
  Text(7, 0, 'Density4Malignant')])
```




```
1 train[train["label"]=="Density3Benign"]
```

	filename	label
0	Image_1.jpg	Density3Benign
6	Image_7.jpg	Density3Benign
19	Image_20.jpg	Density3Benign
24	Image_25.jpg	Density3Benign
33	Image_34.jpg	Density3Benign
...	...	...
547	Image_548.jpg	Density3Benign
552	Image_553.jpg	Density3Benign
568	Image_569.jpg	Density3Benign
569	Image_570.jpg	Density3Benign
582	Image_583.jpg	Density3Benign

64 rows × 2 columns

```
1
2 def preprocess(image):
3     kernel = np.array([[0,-1,0], [-1,5,-1], [0,-1,0]])
4     im = cv2.filter2D(image, -1, kernel)
5     #out2 = skimage.exposure.rescale_intensity(im, in_range=(150,200), out_range=(0,255))
6     out1=(cv2.normalize(im, (224,224),0, 255, cv2.NORM_MINMAX))
7     return out1
```

```
1 X_train, X_valid = train_test_split(train, test_size=0.1,stratify=train["label"],shuffle=True)
```

```
1 print("len of training set", len(X_train))
2 print("len of validation set", len(X_valid))
```

```
len of training set 540
len of validation set 60
```

```
1 datagen=ImageDataGenerator(rescale=1./255,preprocessing_function=preprocess)
```

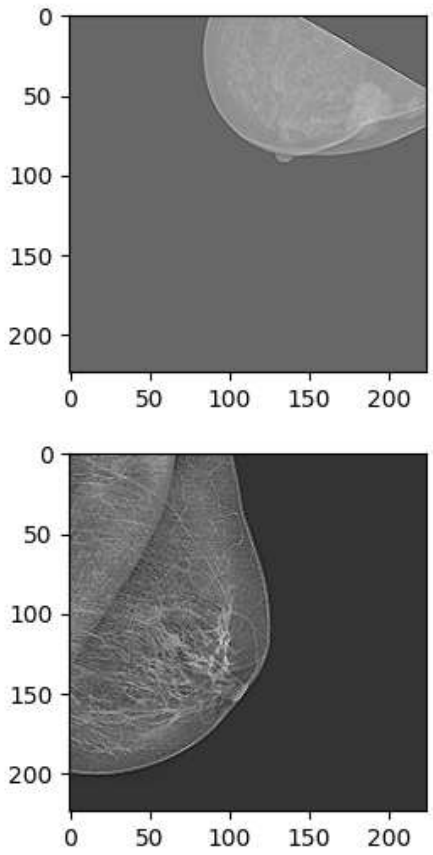
```
train_generator=datagen.flow_from_dataframe(
dataframe=X_train,
directory="/content/drive/MyDrive/mammography_images/test",
x_col="filename",
y_col="label",
batch_size=32,
seed=42,
shuffle=True,
class_mode="categorical",
target_size=(224,224))
```

```
Found 176 validated image filenames belonging to 8 classes.
/usr/local/lib/python3.10/dist-packages/keras/src/preprocessing/image.py:1137: UserWarning: Found 364 invalid image filename(s) in x_col="filename". Thes
  warnings.warn(
```

```
x,y = train_generator.next()
for i in range(2):
    image = x[i]
    plt.figure(figsize=(6,8))
    plt.subplot(1,2,i+1)
    plt.imshow(image)
    print(image.shape)

plt.show()
```

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




```
valid_datagen=ImageDataGenerator(rescale=1./255.,preprocessing_function=preprocess)
valid_generator=valid_datagen.flow_from_dataframe(
dataframe=X_valid,
directory="/content/drive/MyDrive/mammography_images/train",
x_col="filename",
y_col="label",
batch_size=32,
seed=42,
shuffle=True,
class_mode="categorical",
target_size=(224,224))
```

```
Found 60 validated image filenames belonging to 8 classes.
```

```
from tensorflow.keras.layers import Dense, Flatten, GlobalAveragePooling2D, BatchNormalization, Dropout,AveragePooling2D
from tensorflow.keras.applications.resnet import ResNet50
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3,DenseNet201,EfficientNetB7, MobileNetV2,Xception,VGG16,NASNetMobile
from keras.applications.inception_resnet_v2 import InceptionResNetV2
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from keras.models import Model
from keras.models import Sequential
from keras.regularizers import *
from tensorflow import keras
from tensorflow.keras import layers
```

```
checkpoint_path = "training_0/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)
my_callbacks = [
               ModelCheckpoint(checkpoint_path, monitor = 'val_accuracy',verbose = 1,save_weights_only=True, save_best_only = True,mode="max"),
               EarlyStopping(monitor='val_loss', patience=5, verbose=0, mode='min'),
               ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5, verbose=1, mode='min',min_delta=1e-4)
]
```

```
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
```

```
def build_model():
  model = Sequential()

  conv_base = DenseNet201(input_shape=(224,224,3), include_top=False, pooling='max',weights='imagenet')
  model.add(conv_base)
  model.add(BatchNormalization())
  model.add(Dense(2048, activation='relu', kernel_regularizer=l1_l2(0.01)))
  model.add(BatchNormalization())
  model.add(Dense(8, activation='softmax'))

  train_layers = [layer for layer in conv_base.layers[::-1][:5]]

  for layer in conv_base.layers:
    if layer in train_layers:
      layer.trainable = True
  return model
```

```
1
2 my_model=build_model()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet201_weights_tf_dim_ordering_tf_kernels_notop.h5
74836368/74836368 [==============================] - 0s 0us/step
```

```
1 my_model.compile(optimizer =tf.keras.optimizers.legacy.Adam(learning_rate=0.00001,decay=0.0001),metrics=["accuracy"],loss= tf.keras.losses.CategoricalCros
2
```

```
1 my_model.fit(
2         train_generator,
3         steps_per_epoch=STEP_SIZE_TRAIN,
4         epochs=10,
5         validation_data=valid_generator,
6         validation_steps=STEP_SIZE_VALID,callbacks=[my_callbacks])
```

```
Epoch 1/10
5/5 [==============================] - ETA: 0s - loss: 776.0792 - accuracy: 0.7639
Epoch 1: val_accuracy did not improve from 0.12500
5/5 [==============================] - 3s 605ms/step - loss: 776.0792 - accuracy: 0.7639 - val_loss: 776.5305 - val_accuracy: 0.0938 - lr: 1.0000e-05
Epoch 2/10
5/5 [==============================] - ETA: 0s - loss: 773.8565 - accuracy: 0.8542
Epoch 2: val_accuracy did not improve from 0.12500
5/5 [==============================] - 3s 648ms/step - loss: 773.8565 - accuracy: 0.8542 - val_loss: 774.5334 - val_accuracy: 0.1250 - lr: 1.0000e-05
Epoch 3/10
5/5 [==============================] - ETA: 0s - loss: 771.9340 - accuracy: 0.9500
Epoch 3: val_accuracy did not improve from 0.12500
5/5 [==============================] - 3s 614ms/step - loss: 771.9340 - accuracy: 0.9500 - val_loss: 772.6759 - val_accuracy: 0.0625 - lr: 1.0000e-05
Epoch 4/10
5/5 [==============================] - ETA: 0s - loss: 770.0441 - accuracy: 0.9722
Epoch 4: val_accuracy did not improve from 0.12500
5/5 [==============================] - 3s 639ms/step - loss: 770.0441 - accuracy: 0.9722 - val_loss: 770.6921 - val_accuracy: 0.0938 - lr: 1.0000e-05
Epoch 5/10
5/5 [==============================] - ETA: 0s - loss: 768.1588 - accuracy: 0.9792
Epoch 5: val_accuracy did not improve from 0.12500
5/5 [==============================] - 3s 651ms/step - loss: 768.1588 - accuracy: 0.9792 - val_loss: 769.0891 - val_accuracy: 0.0625 - lr: 1.0000e-05
Epoch 6/10
5/5 [==============================] - ETA: 0s - loss: 766.2537 - accuracy: 0.9722
Epoch 6: val_accuracy did not improve from 0.12500
5/5 [==============================] - 3s 683ms/step - loss: 766.2537 - accuracy: 0.9722 - val_loss: 767.1136 - val_accuracy: 0.0625 - lr: 1.0000e-05
Epoch 7/10
5/5 [==============================] - ETA: 0s - loss: 764.3651 - accuracy: 0.9563
Epoch 7: val_accuracy did not improve from 0.12500
5/5 [==============================] - 3s 629ms/step - loss: 764.3651 - accuracy: 0.9563 - val_loss: 765.2567 - val_accuracy: 0.0625 - lr: 1.0000e-05
Epoch 8/10
5/5 [==============================] - ETA: 0s - loss: 762.4226 - accuracy: 0.9861
```

```
Epoch 8: val_accuracy did not improve from 0.12500
5/5 [==============================] - 3s 569ms/step - loss: 762.4226 - accuracy: 0.9861 - val_loss: 763.3333 - val_accuracy: 0.0938 - lr: 1.0000e-05
Epoch 9/10
5/5 [==============================] - ETA: 0s - loss: 760.5114 - accuracy: 0.9875
Epoch 9: val_accuracy did not improve from 0.12500
5/5 [==============================] - 3s 627ms/step - loss: 760.5114 - accuracy: 0.9875 - val_loss: 761.3914 - val_accuracy: 0.0312 - lr: 1.0000e-05
Epoch 10/10
5/5 [==============================] - ETA: 0s - loss: 758.6155 - accuracy: 0.9937
Epoch 10: val_accuracy did not improve from 0.12500
5/5 [==============================] - 3s 621ms/step - loss: 758.6155 - accuracy: 0.9937 - val_loss: 759.6736 - val_accuracy: 0.0312 - lr: 1.0000e-05
<keras.src.callbacks.History at 0x7f6b08574e80>
```

```
1 my_model.load_weights(checkpoint_path)
```

```
<tensorflow.python.checkpoint.checkpoint.CheckpointLoadStatus at 0x7f6b00ccbc70>
```

```
1 my_model.evaluate(valid_generator,verbose=1)
```

```
2/2 [==============================] - 7s 7s/step - loss: 786.4396 - accuracy: 0.1167
[786.4395751953125, 0.11666666716337204]
```

```
1 train_generator.class_indices
```

```
{'Density1Benign': 0,
 'Density1Malignant': 1,
 'Density2Benign': 2,
 'Density2Malignant': 3,
 'Density3Benign': 4,
 'Density3Malignant': 5,
 'Density4Benign': 6,
 'Density4Malignant': 7}
```

```
test_datagen=ImageDataGenerator(rescale=1./255.,preprocessing_function=preprocess)
test_generator=valid_datagen.flow_from_dataframe(
dataframe=test.
```

```
pred1=np.argmax(my_model.predict(test_generator,steps=STEP_SIZE_TEST,verbose=1),axis=1)
```

```
200/200 [==============================] - 14s 38ms/step
```

```

pred1

```

```
array([4, 4, 4, 4, 4, 4, 1, 4, 1, 4, 1, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 1,
       4, 1, 4, 4, 1, 4, 4, 4, 1, 4, 1, 1, 4, 4, 5, 1, 4, 4, 5, 4, 4, 5,
       1, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 1, 1, 4, 1, 4, 1, 4, 5, 5, 4,
       5, 4, 5, 1, 4, 4, 4, 1, 1, 4, 4, 5, 4, 4, 1, 1, 4, 4, 4, 4, 4, 4,
       1, 4, 1, 1, 4, 4, 4, 4, 1, 5, 4, 4, 4, 4, 1, 4, 1, 4, 5, 5, 4, 5,
       4, 4, 4, 5, 1, 4, 4, 4, 4, 1, 4, 4, 5, 4, 4, 1, 4, 4, 4, 5, 4, 1,
       4, 1, 4, 4, 1, 4, 4, 4, 4, 4, 5, 4, 4, 4, 4, 4, 4, 4, 4, 0, 5, 5,
       1, 4, 5, 4, 4, 4, 1, 4, 4, 1, 4, 1, 5, 4, 4, 4, 4, 4, 5, 4, 5, 4,
       4, 1, 4, 4, 4, 1, 5, 4, 4, 1, 4, 1, 4, 4, 1, 4, 1, 4, 4, 7, 4, 4,
```