```
import matplotlib
matplotlib.use("Agg")

# import the necessary packages
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.preprocessing.image import img_to_array
from keras.callbacks import ModelCheckpoint
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from keras.applications.vgg19 import VGG19
from keras.applications.resnet50 import ResNet50
from keras.applications.densenet import DenseNet201
from keras.applications.nasnet import NASNetMobile
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D, Dropout
from keras.optimizers import SGD
import matplotlib.pyplot as plt
from imutils import paths
import numpy as np
import argparse
import random
import pickle
import cv2
import os
import pandas as pd
import numpy as np
import urllib.request
from tqdm import tqdm_notebook as tqdm


# Run this cell to mount your Google Drive.
from google.colab import drive
drive.mount('/content/drive')


!unzip drive/My\ Drive/final_data-fynd.zip


EPOCHS = 25
INIT_LR = 1e-3
BS = 32
IMAGE_DIMS = (128, 128, 3)


add = 0
for class_type in ['backstrap', 'buckle', 'hook', 'lace_up', 'slip_on', 'zipper']:
  add = add + len(os.listdir("data/BV/"+class_type))
print(add)
```

```
2087
```

```
add = 0
for class_type in ['backstrap', 'buckle', 'hook', 'lace_up', 'slip_on', 'zipper']:
  add = add + len(os.listdir("data/NBV/"+class_type))
print(add)
```

```
6907
```

```
print("[INFO] loading images...")
labels = []
# 2087 + 6907 = 8994
data = np.empty((8994, 128, 128, 3))
i = 0
new_labels = []
print ("Size of data before = ",data.nbytes / (1024 * 1000.0))
```

```
for class_type in ['backstrap', 'buckle', 'hook', 'lace_up', 'slip_on', 'zipper']:
  imagePaths = sorted(list(paths.list_images('data/BV/'+class_type)))
  random.seed(42)
  random.shuffle(imagePaths)

  for imagePath in imagePaths:
    # load the image, pre-process it, and store it in the data list
    image = cv2.imread(imagePath)
    image = cv2.resize(image, (IMAGE_DIMS[1], IMAGE_DIMS[0]))
    data[i,]=image
    i = i+1
    # extract set of class labels from the image path and update the labels list
    l = label = class_type+'_BV'
    labels.append(l)

for class_type in ['backstrap', 'buckle', 'hook', 'lace_up', 'slip_on', 'zipper']:
  imagePaths = sorted(list(paths.list_images('data/NBV/'+class_type)))
  random.seed(42)
  random.shuffle(imagePaths)

  for imagePath in imagePaths:
    # load the image, pre-process it, and store it in the data list
    image = cv2.imread(imagePath)
    image = cv2.resize(image, (IMAGE_DIMS[1], IMAGE_DIMS[0]))
    data[i,]=image
    i = i+1
    # extract set of class labels from the image path and update the labels list
    l = label = class_type+'_NBV'
    labels.append(l)
data = data/255.0
print("[INFO] data matrix: {} images ({:.2f}MB)".format(i, data.nbytes / (1024 * 1000.0)))
del(imagePaths)
new_labels = labels
labels = np.array(labels)
```

```
[INFO] loading images...
Size of data before =  3453.696
[INFO] data matrix: 8994 images (3453.70MB)
```

```
set(labels)
```

```
{'backstrap_BV',
 'backstrap_NBV',
 'buckle_BV',
 'buckle_NBV',
 'hook_BV',
 'hook_NBV',
 'lace_up_BV',
 'lace_up_NBV',
 'slip_on_BV',
 'slip_on_NBV',
 'zipper_BV',
 'zipper_NBV'}
```

```
from sklearn.preprocessing import LabelBinarizer
print("[INFO] class labels:")
mlb = LabelBinarizer()
labels = mlb.fit_transform(labels)

# loop over each of the possible class labels and show them
for (i, label) in enumerate(mlb.classes_):
  print("{}. {}".format(i + 1, label))
```

```
[INFO] class labels:
1. backstrap_BV
2. backstrap_NBV
3. buckle_BV
4. buckle_NBV
5. hook_BV
6. hook_NBV
7. lace_up_BV
8. lace_up_NBV
9. slip_on_BV
10. slip_on_NBV
11. zipper_BV
```

```
(trainX, testX, trainY, testY) = train_test_split(data,
  labels, test_size=0.2, random_state=42,shuffle=True)
del data


aug = ImageDataGenerator(rotation_range=25, width_shift_range=0.1,
  height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
  horizontal_flip=True, fill_mode="nearest")


vgg_model = VGG19(input_shape=(128, 128, 3), include_top=False, weights='imagenet')


x = vgg_model.output
x = GlobalAveragePooling2D()(x)

# add fully-connected layer
x = Dense(512, activation='relu')(x)
x = Dropout(0.3)(x)

# add output layer
predictions = Dense(12, activation='softmax')(x)

model = Model(inputs=vgg_model.input, outputs=predictions)

# freeze pre-trained model area's layer
for layer in vgg_model.layers:
    layer.trainable = False

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_5 (InputLayer)         (None, 128, 128, 3)       0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 128, 128, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 128, 128, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 64, 64, 64)        0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 64, 64, 128)       73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 64, 64, 128)       147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 32, 32, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 32, 32, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 32, 32, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 32, 32, 256)       590080    
_________________________________________________________________
block3_conv4 (Conv2D)        (None, 32, 32, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 16, 16, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 16, 16, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 16, 16, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 16, 16, 512)       2359808   
_________________________________________________________________
block4_conv4 (Conv2D)        (None, 16, 16, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 8, 8, 512)         0         
```

```
for layer in model.layers[:17]:
    layer.trainable = False

for layer in model.layers[17:]:
    layer.trainable = True

opt = Adam(lr=INIT_LR, decay=INIT_LR / 25)

model.compile(loss="categorical_crossentropy", optimizer=opt,
  metrics=["accuracy"])

# train the network
print("[INFO] training network...")
file_epoch = "fine_tune_shoes_multiclass.best.hdf5"
checkpoint = ModelCheckpoint(file_epoch, monitor='val_acc', verbose=1, save_best_only=Tru
callbacks_list = [checkpoint]

H = model.fit_generator(
  aug.flow(trainX, trainY, batch_size=BS),
  validation_data=(testX, testY),
  steps_per_epoch=len(trainX) // BS, class_weight = 'balanced',
  epochs=25, verbose=1, callbacks=callbacks_list)

model.save('shoes_best_multiclass.model')
```

```
Trainable params: 268,812
```

```
!rsync -Pav fine_tune_shoes_multiclass.best.hdf5 drive/My\ Drive/
```

```
sending incremental file list
fine_tune_shoes_multiclass.best.hdf5
    158,937,280 100%  101.57MB/s    0:00:01 (xfr#1, to-chk=0/1)

sent 158,976,199 bytes  received 35 bytes  105,984,156.00 bytes/sec
total size is 158,937,280  speedup is 1.00
```