In [ ]:
```
#importing all the libraries
import os
import numpy as np
import pandas as pd
from keras.utils import np_utils
from sklearn.preprocessing import LabelEncoder
from keras.applications.resnet import preprocess_input
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array

import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:
```
from keras.callbacks import EarlyStopping
from keras.backend import clear_session

clear_session()
```

Download the dataset using the below link

In [ ]:
```
#https://www.kaggle.com/competitions/dog-breed-identification/data?select=train
```

In [ ]:
```
#train dir contains the training images
base_dir = '.'
data_dir = os.path.join(base_dir,'train')
files = os.listdir(data_dir)
```

In [ ]:
```
#target information from labels.csv

labels = pd.read_csv(os.path.join(base_dir,'labels.csv'))
labels.head()
```

Out[ ]:

	id	breed
0	000bec180eb18c7604dcecc8fe0dba07	boston_bull
1	001513dfcb2ffafc82cccf4d8bbaba97	dingo
2	001cdf01b096e06d78e9e5112d419397	pekinese
3	00214f311d5d2247d5dfe4fe24b2303d	bluetick
4	0021f9ceb3235effd7fcde7f7538ed62	golden_retriever

In [ ]:
```
file_df = pd.DataFrame({'id':list(map(lambda x:x.replace('.jpg',''),files))})
file_df.head()
```

Out[ ]:

	id
0	000bec180eb18c7604dcecc8fe0dba07
1	001513dfcb2ffafc82cccf4d8bbaba97
2	001cdf01b096e06d78e9e5112d419397
3	00214f311d5d2247d5dfe4fe24b2303d
4	0021f9ceb3235effd7fcde7f7538ed62

In [ ]:
```
#mapping file with breed, maintain file read order

label_info =  pd.merge(left = file_df, right = labels)
label_info.head()
```

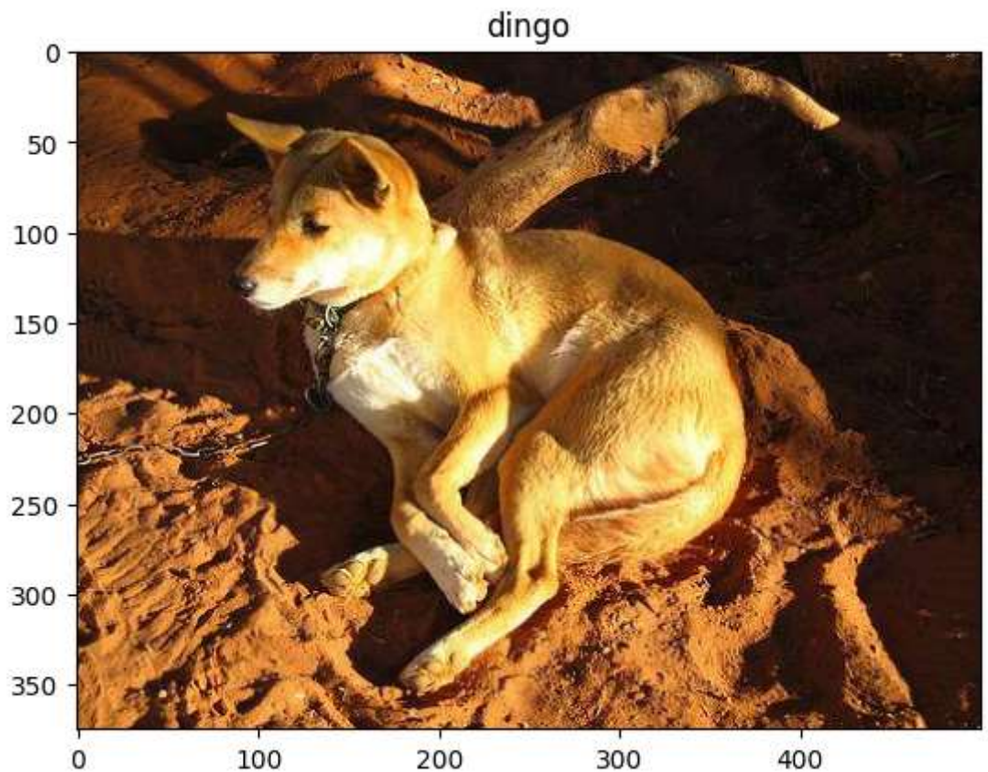
Out[ ]:

	id	breed
0	000bec180eb18c7604dcecc8fe0dba07	boston_bull
1	001513dfcb2ffafc82cccf4d8bbaba97	dingo
2	001cdf01b096e06d78e9e5112d419397	pekinese
3	00214f311d5d2247d5dfe4fe24b2303d	bluetick
4	0021f9ceb3235effd7fcde7f7538ed62	golden_retriever

In [ ]:
```
img = plt.imread(os.path.join(data_dir,files[1]))
```

In [ ]:
```
#showing a image

plt.imshow(img)
plt.title(label_info.iloc[1]['breed'])
plt.show()
```



In [ ]:
```
# converting target to one hot vector format

num_classes = len(label_info.breed.unique())
num_classes
```

Out[ ]:
```
120
```

In [ ]:
```
le = LabelEncoder()
breed = le.fit_transform(label_info.breed)
Y = np_utils.to_categorical(breed,num_classes = num_classes)
```

In [ ]:
```
Y.shape
```

Out[ ]:
```
(10222, 120)
```

In [ ]:
```
# converting image to numpy array
input_dim = (224, 224)

X = np.zeros((Y.shape[0], *input_dim,3))

for i,img in enumerate(files):
    image = load_img(os.path.join(data_dir,img), target_size = input_dim)
    image = img_to_array(image)
    image = image.reshape((1, *image.shape))
    image = preprocess_input(image)
    X[i] = image
```

In [ ]:
```
X.shape
```

Out[ ]:
```
(10222, 224, 224, 3)
```

In [ ]:
```
from keras.applications.vgg19 import VGG19
from keras.models import Model
from keras.layers import Dense,GlobalAveragePooling2D, Flatten, Dropout

vgg_model = VGG19(weights='imagenet', include_top=False)

x= vgg_model.output
x= GlobalAveragePooling2D()(x)
x=Dropout(0.3)(x)
out = Dense(120,activation = 'softmax')(x)

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

for layer in vgg_model.layers[:-1]:
    layer.trainable = False
for layer in vgg_model.layers[-1:]:
```

```
    layer.trainabl= True
from keras.optimizers import Adam
opt= Adam()

model.compile(optimizer = opt, loss = 'categorical_crossentropy', metrics = ['accuracy'])
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, None, None, 3)]   0

 block1_conv1 (Conv2D)       (None, None, None, 64)    1792

 block1_conv2 (Conv2D)       (None, None, None, 64)    36928

 block1_pool (MaxPooling2D)  (None, None, None, 64)    0

 block2_conv1 (Conv2D)       (None, None, None, 128)   73856

 block2_conv2 (Conv2D)       (None, None, None, 128)   147584

 block2_pool (MaxPooling2D)  (None, None, None, 128)   0

 block3_conv1 (Conv2D)       (None, None, None, 256)   295168

 block3_conv2 (Conv2D)       (None, None, None, 256)   590080

 block3_conv3 (Conv2D)       (None, None, None, 256)   590080

 block3_conv4 (Conv2D)       (None, None, None, 256)   590080

 block3_pool (MaxPooling2D)  (None, None, None, 256)   0

 block4_conv1 (Conv2D)       (None, None, None, 512)   1180160

 block4_conv2 (Conv2D)       (None, None, None, 512)   2359808

 block4_conv3 (Conv2D)       (None, None, None, 512)   2359808

 block4_conv4 (Conv2D)       (None, None, None, 512)   2359808

 block4_pool (MaxPooling2D)  (None, None, None, 512)   0

 block5_conv1 (Conv2D)       (None, None, None, 512)   2359808

 block5_conv2 (Conv2D)       (None, None, None, 512)   2359808
```

```
 block5_conv3 (Conv2D)       (None, None, None, 512)   2359808

 block5_conv4 (Conv2D)       (None, None, None, 512)   2359808

 block5_pool (MaxPooling2D)  (None, None, None, 512)   0

 global_average_pooling2d (G  (None, 512)              0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 512)               0

 dense (Dense)               (None, 120)               61560

=================================================================
Total params: 20,085,944
Trainable params: 61,560
Non-trainable params: 20,024,384
_________________________________________________________________
```

In [ ]:
```
#create callbacks
#earlystop = EarlyStopping(monitor='val_loss', min_delta=0, patience=3, verbose=0, mode='auto')
```

In [ ]:
```
from keras.backend import clear_session
import tensorflow as tf

clear_session()
```

In [ ]:
```
history_few_layer = model.fit(X[:1000], Y[:1000], batch_size=32, epochs=30, validation_split=0.2, verbose=2)
#history_few_layer = model.fit(X[:1000], Y[:1000], batch_size=32, epochs=30, validation_split=0.2, verbose=2, callbacks='earlystop')
```

```
Epoch 1/30
25/25 - 167s - loss: 21.1930 - accuracy: 0.0137 - val_loss: 14.0778 - val_accuracy: 0.0100 - 167s/epoch - 7s/step
Epoch 2/30
25/25 - 171s - loss: 15.7852 - accuracy: 0.0550 - val_loss: 11.2936 - val_accuracy: 0.0650 - 171s/epoch - 7s/step
Epoch 3/30
25/25 - 168s - loss: 12.4339 - accuracy: 0.0862 - val_loss: 9.7992 - val_accuracy: 0.0750 - 168s/epoch - 7s/step
Epoch 4/30
25/25 - 170s - loss: 9.4622 - accuracy: 0.1488 - val_loss: 8.4550 - val_accuracy: 0.1000 - 170s/epoch - 7s/step
Epoch 5/30
25/25 - 171s - loss: 7.1724 - accuracy: 0.2387 - val_loss: 7.8881 - val_accuracy: 0.1300 - 171s/epoch - 7s/step
Epoch 6/30
25/25 - 172s - loss: 5.4230 - accuracy: 0.3125 - val_loss: 7.3397 - val_accuracy: 0.1550 - 172s/epoch - 7s/step
Epoch 7/30
25/25 - 172s - loss: 4.3801 - accuracy: 0.3988 - val_loss: 6.8291 - val_accuracy: 0.1900 - 172s/epoch - 7s/step
Epoch 8/30
25/25 - 172s - loss: 3.4505 - accuracy: 0.4638 - val_loss: 6.6711 - val_accuracy: 0.1850 - 172s/epoch - 7s/step
Epoch 9/30
25/25 - 172s - loss: 2.7457 - accuracy: 0.5525 - val_loss: 6.4071 - val_accuracy: 0.2300 - 172s/epoch - 7s/step
Epoch 10/30
25/25 - 172s - loss: 2.3514 - accuracy: 0.5850 - val_loss: 6.5104 - val_accuracy: 0.2150 - 172s/epoch - 7s/step
Epoch 11/30
25/25 - 172s - loss: 2.0081 - accuracy: 0.6187 - val_loss: 6.4689 - val_accuracy: 0.2150 - 172s/epoch - 7s/step
Epoch 12/30
25/25 - 172s - loss: 1.5242 - accuracy: 0.6637 - val_loss: 6.2585 - val_accuracy: 0.2650 - 172s/epoch - 7s/step
Epoch 13/30
25/25 - 172s - loss: 1.4374 - accuracy: 0.7063 - val_loss: 6.2923 - val_accuracy: 0.2500 - 172s/epoch - 7s/step
Epoch 14/30
25/25 - 173s - loss: 1.2496 - accuracy: 0.7362 - val_loss: 6.1994 - val_accuracy: 0.2600 - 173s/epoch - 7s/step
Epoch 15/30
25/25 - 173s - loss: 0.9924 - accuracy: 0.7713 - val_loss: 6.1313 - val_accuracy: 0.2600 - 173s/epoch - 7s/step
Epoch 16/30
25/25 - 174s - loss: 0.7215 - accuracy: 0.8138 - val_loss: 6.2751 - val_accuracy: 0.2500 - 174s/epoch - 7s/step
Epoch 17/30
25/25 - 171s - loss: 0.7598 - accuracy: 0.8250 - val_loss: 6.3245 - val_accuracy: 0.2400 - 171s/epoch - 7s/step
Epoch 18/30
25/25 - 176s - loss: 0.7605 - accuracy: 0.8150 - val_loss: 6.5122 - val_accuracy: 0.2350 - 176s/epoch - 7s/step
Epoch 19/30
25/25 - 174s - loss: 0.6457 - accuracy: 0.8313 - val_loss: 6.4988 - val_accuracy: 0.2400 - 174s/epoch - 7s/step
Epoch 20/30
25/25 - 178s - loss: 0.6752 - accuracy: 0.8350 - val_loss: 6.4070 - val_accuracy: 0.2500 - 178s/epoch - 7s/step
Epoch 21/30
25/25 - 173s - loss: 0.5154 - accuracy: 0.8800 - val_loss: 6.0922 - val_accuracy: 0.2800 - 173s/epoch - 7s/step
```

```
Epoch 22/30
25/25 - 169s - loss: 0.4892 - accuracy: 0.8637 - val_loss: 6.3593 - val_accuracy: 0.2650 - 169s/epoch - 7s/step
Epoch 23/30
25/25 - 169s - loss: 0.3561 - accuracy: 0.8963 - val_loss: 6.0956 - val_accuracy: 0.2700 - 169s/epoch - 7s/step
Epoch 24/30
25/25 - 169s - loss: 0.3282 - accuracy: 0.9000 - val_loss: 5.9483 - val_accuracy: 0.2900 - 169s/epoch - 7s/step
Epoch 25/30
25/25 - 168s - loss: 0.3169 - accuracy: 0.9100 - val_loss: 6.3923 - val_accuracy: 0.2750 - 168s/epoch - 7s/step
Epoch 26/30
25/25 - 168s - loss: 0.3030 - accuracy: 0.9125 - val_loss: 6.4267 - val_accuracy: 0.2850 - 168s/epoch - 7s/step
Epoch 27/30
25/25 - 168s - loss: 0.3626 - accuracy: 0.8975 - val_loss: 6.4361 - val_accuracy: 0.2900 - 168s/epoch - 7s/step
Epoch 28/30
25/25 - 168s - loss: 0.2333 - accuracy: 0.9300 - val_loss: 6.3120 - val_accuracy: 0.3050 - 168s/epoch - 7s/step
Epoch 29/30
25/25 - 168s - loss: 0.2054 - accuracy: 0.9362 - val_loss: 6.2073 - val_accuracy: 0.3050 - 168s/epoch - 7s/step
Epoch 30/30
25/25 - 169s - loss: 0.2738 - accuracy: 0.9275 - val_loss: 6.3648 - val_accuracy: 0.3150 - 169s/epoch - 7s/step
```

Over all we got 93% accuracy.

In [ ]:
```
model.save('vgg19DBPmodel.h5')
```

In [ ]:
```
history_few_layer.history.keys()
```

Out[ ]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```