In [ ]:

```
from keras.preprocessing import image
import numpy as np
import os
from keras.applications.imagenet_utils import preprocess_input #model
from keras.applications.imagenet_utils import decode_predictions

os.chdir("/content/drive/MyDrive/data/data")
from keras.utils import np_utils
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
import os
import time
PATH = os.getcwd()
# Define data path

data_path = PATH
data_dir_list = os.listdir(data_path)

img_data_list=[]

for dataset in data_dir_list:
 img_list=os.listdir(data_path+'/'+ dataset) #e:/data/cat
 print ('Loaded the images of dataset-'+'{}\n'.format(dataset))
 for img in img_list:
  img_path = data_path + '/'+ dataset + '/'+ img  #e:/data/cat/cat.1.jpg
  img = image.load_img(img_path, target_size=(224, 224))
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis=0)
  x = preprocess_input(x) #image into
#  x = x/255
  print('Input image shape:', x.shape)
  img_data_list.append(x)

img_data = np.array(img_data_list)
#img_data = img_data.astype('float32')
print (img_data.shape)
img_data=np.rollaxis(img_data,1,0)
print (img_data.shape)
img_data=img_data[0]
print (img_data.shape)


# Define the number of classes
num_classes = 4
num_of_samples = img_data.shape[0]
labels = np.ones((num_of_samples,),dtype='int64')

labels[0:25]=0
labels[26:50]=1
labels[51:75]=2
labels[76:]=3

names = ['cats','dogs','horses','humans']

# convert class labels to on-hot encoding
Y = np_utils.to_categorical(labels, num_classes)

#Shuffle the dataset
x,y = shuffle(img_data,Y, random_state=2)
# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2)
```

```
Loaded the images of dataset-dogs

Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
```

```
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Loaded the images of dataset-humans

Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Loaded the images of dataset-cats

Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
```

```
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Loaded the images of dataset-horses

Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
(100, 1, 224, 224, 3)
(1, 100, 224, 224, 3)
(100, 224, 224, 3)
```

In [ ]:

```python
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten
#create model
model = Sequential()
#add model layers
model.add(Conv2D(64, kernel_size=3, activation='relu', input_shape=(1,224,224,3)))
model.add(Conv2D(32, kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(4, activation='softmax'))
#compile model using accuracy to measure model performance
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
#train the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10)
```

```
Epoch 1/10
WARNING:tensorflow:Model was constructed with shape (None, 1, 224, 224, 3) for input Kera
sTensor(type_spec=TensorSpec(shape=(None, 1, 224, 224, 3), dtype=tf.float32, name='conv2d
_11_input'), name='conv2d_11_input', description="created by layer 'conv2d_11_input'"), b
ut it was called on an input with incompatible shape (32, 224, 224, 3).
WARNING:tensorflow:Model was constructed with shape (None, 1, 224, 224, 3) for input Kera
sTensor(type_spec=TensorSpec(shape=(None, 1, 224, 224, 3), dtype=tf.float32, name='conv2d
_11_input'), name='conv2d_11_input', description="created by layer 'conv2d_11_input'"), b
ut it was called on an input with incompatible shape (32, 224, 224, 3).
3/3 [==============================] - ETA: 0s - loss: 2485.4605 - accuracy: 0.2066WARNIN
G:tensorflow:Model was constructed with shape (None, 1, 224, 224, 3) for input KerasTenso
r(type_spec=TensorSpec(shape=(None, 1, 224, 224, 3), dtype=tf.float32, name='conv2d_11_in
put'), name='conv2d_11_input', description="created by layer 'conv2d_11_input'"), but it
was called on an input with incompatible shape (None, 224, 224, 3).
3/3 [==============================] - 19s 6s/step - loss: 2938.9291 - accuracy: 0.2122 -
val_loss: 12044.3867 - val_accuracy: 0.2500
Epoch 2/10
3/3 [==============================] - 18s 6s/step - loss: 4239.0009 - accuracy: 0.2500 -
val_loss: 1073.4141 - val_accuracy: 0.0000e+00
Epoch 3/10
3/3 [==============================] - 18s 6s/step - loss: 319.4838 - accuracy: 0.4089 -
```

```
val_loss: 84.2302 - val_accuracy: 0.0000e+00
Epoch 4/10
3/3 [==============================] - 18s 6s/step - loss: 39.7833 - accuracy: 0.6497 - v
al_loss: 19.8283 - val_accuracy: 0.7500
Epoch 5/10
3/3 [==============================] - 18s 6s/step - loss: 1.8059 - accuracy: 0.9323 - va
l_loss: 18.1143 - val_accuracy: 0.2500
Epoch 6/10
3/3 [==============================] - 18s 6s/step - loss: 0.0223 - accuracy: 0.9909 - va
l_loss: 40.6286 - val_accuracy: 0.2500
Epoch 7/10
3/3 [==============================] - 18s 6s/step - loss: 4.7342e-07 - accuracy: 1.0000
- val_loss: 60.9468 - val_accuracy: 0.2500
Epoch 8/10
3/3 [==============================] - 18s 6s/step - loss: 8.0535e-06 - accuracy: 1.0000
- val_loss: 82.7296 - val_accuracy: 0.0000e+00
Epoch 9/10
3/3 [==============================] - 18s 6s/step - loss: 0.1247 - accuracy: 0.9857 - va
l_loss: 75.1305 - val_accuracy: 0.2500
Epoch 10/10
3/3 [==============================] - 18s 6s/step - loss: 2.2197e-08 - accuracy: 1.0000
- val_loss: 66.3309 - val_accuracy: 0.2500
```

Out[ ]:

```
<tensorflow.python.keras.callbacks.History at 0x7f093a0c4410>
```

In [ ]:

```
model.summary()
```

```
Model: "sequential_23"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_18 (Flatten)         (None, 150528)            0
_________________________________________________________________
dense_44 (Dense)             (None, 128)               19267712
_________________________________________________________________
activation_44 (Activation)   (None, 128)               0
_________________________________________________________________
dropout_20 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_45 (Dense)             (None, 64)                8256
_________________________________________________________________
activation_45 (Activation)   (None, 64)                0
_________________________________________________________________
dropout_21 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_46 (Dense)             (None, 4)                 260
_________________________________________________________________
activation_46 (Activation)   (None, 4)                 0
=================================================================
Total params: 19,276,228
Trainable params: 19,276,228
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten,Activation,MaxPooling2D,Dropout
model = Sequential()
model.add(Conv2D(64, kernel_size=3, activation='relu', input_shape=(224,224,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(32,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,(3,3)))
```

```
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(4))
model.add(Activation('tanh'))

model.compile(optimizer = 'rmsprop',loss = 'binary_crossentropy',metrics = ['accuracy'])
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10)
```

```
Epoch 1/10
3/3 [==============================] - 10s 3s/step - loss: 6.3758 - accuracy: 0.2398 - va
l_loss: 5.3680 - val_accuracy: 0.3000
Epoch 2/10
3/3 [==============================] - 8s 2s/step - loss: 7.3646 - accuracy: 0.2305 - val
_loss: 5.3680 - val_accuracy: 0.3000
Epoch 3/10
3/3 [==============================] - 8s 2s/step - loss: 6.5525 - accuracy: 0.2594 - val
_loss: 5.3680 - val_accuracy: 0.3000
Epoch 4/10
3/3 [==============================] - 8s 2s/step - loss: 6.9612 - accuracy: 0.2203 - val
_loss: 5.3680 - val_accuracy: 0.3000
Epoch 5/10
3/3 [==============================] - 8s 2s/step - loss: 6.8387 - accuracy: 0.2047 - val
_loss: 8.0300 - val_accuracy: 0.3000
Epoch 6/10
3/3 [==============================] - 8s 2s/step - loss: 7.3312 - accuracy: 0.2188 - val
_loss: 8.0300 - val_accuracy: 0.3000
Epoch 7/10
3/3 [==============================] - 8s 2s/step - loss: 7.5647 - accuracy: 0.2477 - val
_loss: 8.0300 - val_accuracy: 0.3000
Epoch 8/10
3/3 [==============================] - 8s 2s/step - loss: 6.7996 - accuracy: 0.2437 - val
_loss: 8.0300 - val_accuracy: 0.3000
Epoch 9/10
3/3 [==============================] - 8s 2s/step - loss: 6.9749 - accuracy: 0.2633 - val
_loss: 8.0300 - val_accuracy: 0.3000
Epoch 10/10
3/3 [==============================] - 8s 2s/step - loss: 7.1436 - accuracy: 0.2164 - val
_loss: 8.0300 - val_accuracy: 0.3000
```

Out[ ]:

```
<tensorflow.python.keras.callbacks.History at 0x7f093da29e90>
```