In [15]:
```
#using CNN classifing DOG image into husky or retriever
import numpy as np
import cv2
import os
import random
import matplotlib.pyplot as plt
```

In [40]:
```
DIRECTORY = r'C:\Users\Jainil\Desktop\Jainil\College\Sem-7\BDA\L8'
CATEGORIES = ['GoldenRetriever','Husky']
IMG_SIZE = 150
```

In [42]:
```
#convert image into array and save it in a list

data = []

for category in CATEGORIES:
    folder = os.path.join(DIRECTORY,category)
    label = CATEGORIES.index(category)
    #print(folder)
    for img in os.listdir(folder):       # listdir will list all files present in folde
        img_path = os.path.join(folder,img)
        img_arr = cv2.imread(img_path)   #read image and convert into array
        img_arr = cv2.resize(img_arr,(IMG_SIZE,IMG_SIZE))

        data.append([img_arr,label])
```

In [46]:
```
# len(data)
```

In [45]:
```
# data[12]
```

In [47]:
```
random.shuffle(data)
```

In [48]:
```
X = []
y = []

for features,labels in data:
    X.append(features)
    y.append(labels)
```

In [49]:
```
X = np.array(X)
y = np.array(y)
```

In [54]:
```
print(str(len(X))+'\n'+str(len(y)))
```

```
30
30
```

In [55]:
```
y
```

Out[55]:
```
array([0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 1, 0, 1])
```

In [61]:
```
X = X/255
# X
"""
    Lesser the values contained in the array representation
    of the image easier the calculation
"""
X.shape
```

Out[61]:
```
(30, 150, 150, 3)
```

In [64]:
```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense
# Dense layers are just regular layers
```

In [68]:
```
model = Sequential()

#HIDDEN LAYER
model.add(Conv2D(64,(3,3),activation='relu'))
"""
number of features to be detected or convolution layers,
feature detector size( matrix )
activation function (softmax,sigmoid,relu) -> generally relu works best in this case
"""
model.add(MaxPool2D((2,2)))

model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPool2D((2,2)))

model.add(Flatten())
model.add(Dense(128,input_shape=X.shape[1:]))
#128 neurons in hidden layer,shape of input image


#OUTPUT LAYER
model.add(Dense(2,activation='softmax'))
```

In [72]:
```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accurac

#adam optimizer is the goto optimizer for most cases
model.fit(X,y,epochs=5,validation_split=0.1)
```

```
Epoch 1/5
1/1 [==============================] - 1s 1s/step - loss: 0.6931 - accuracy: 0.5185 - va
l_loss: 0.6980 - val_accuracy: 0.3333
Epoch 2/5
1/1 [==============================] - 1s 558ms/step - loss: 0.6927 - accuracy: 0.5185 -
val_loss: 0.8848 - val_accuracy: 0.3333
Epoch 3/5
1/1 [==============================] - 1s 576ms/step - loss: 0.7470 - accuracy: 0.5185 -
```

```
val_loss: 0.6711 - val_accuracy: 0.6667
Epoch 4/5
1/1 [==============================] - 1s 516ms/step - loss: 0.6987 - accuracy: 0.4815 -
val_loss: 0.6858 - val_accuracy: 0.6667
Epoch 5/5
1/1 [==============================] - 1s 534ms/step - loss: 0.6942 - accuracy: 0.4815 -
val_loss: 0.7315 - val_accuracy: 0.3333
```

Out[72]: <keras.callbacks.History at 0x1c7c3004d60>

In [ ]: