NAME:PRINCY

ROLL NO:225229128

SUB:PDL ¶

In [5]:
```
import os
import cv2
import time
%matplotlib inline
import numpy as np
import pandas as pd
from time import process_time
import matplotlib.pyplot as plt
```

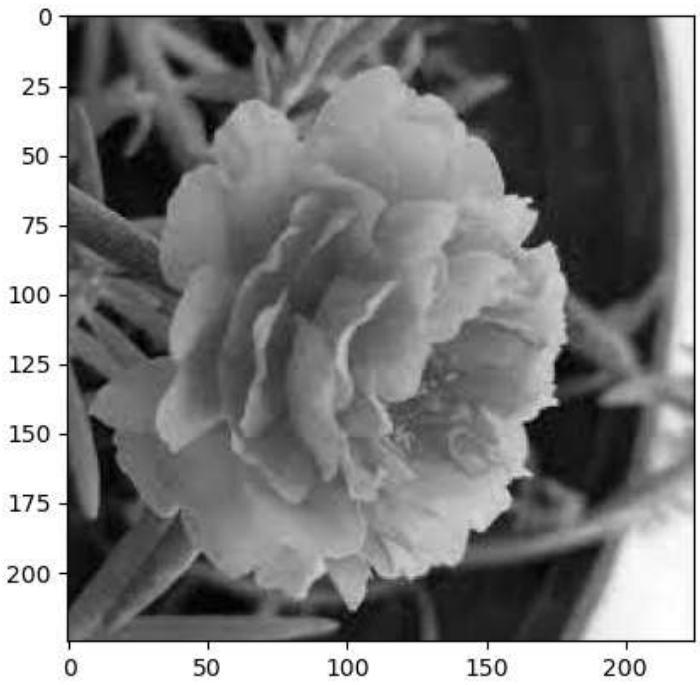
In [6]:
```
import tensorflow as tf
import warnings
warnings.filterwarnings("ignore")
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
```

In [7]:
```
#step2:
df="C:\\Users\\veena\\Desktop\\bigdata"
categories =['rose']
```

In [8]:
```
for category in categories:
    path = os.path.join(df, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE)
        plt.imshow(img_array, cmap='gray')
        plt.show()
        break
    break
```

In [11]:
```
df1="C:\\Users\\veena\\Desktop\\bigdata"
categories1 =['tablerose']
```

In [12]:
```
for category in categories1:
    path = os.path.join(df1, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE)
        plt.imshow(img_array, cmap='gray')
        plt.show()
        break
    break
```



In [13]:
```
df3 = "C:\\Users\\veena\\Desktop\\bigdata"
categories3 = ['rose', 'tablerose']
```

In [15]:
```
data = []
img_size=500
def preprocess():
    for category in categories3:
        path = os.path.join(df3, category)
        class_num = categories3.index(category)
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE)
            num_array = cv2.resize(img_array,(img_size, img_size))
            data.append([num_array, class_num])
preprocess()
```

In [16]:
```
print(len(data))
```

```
20
```

In [17]:
```
#step3:
X = []
y = []
for features,label in data:
    X.append(features)
    y.append(label)

X = np.asarray(X).reshape(-1,img_size,img_size,1)
y = np.asarray(y)
```

In [18]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

In [19]:
```
print("Shape of the following:")
print("X_train =", X_train.shape)
print("X_test =", X_test.shape)
print("y_train =", y_train.shape)
print("y_test =", y_test.shape)
```

```
Shape of the following:
X_train = (15, 500, 500, 1)
X_test = (5, 500, 500, 1)
y_train = (15,)
y_test = (5,)
```

In [20]:
```
y_train
```

Out[20]:
```
array([0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0])
```

In [21]:
```
#step4:
model = Sequential()
model.add(Dense(8, input_dim=1, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='mean_squared_error',
optimizer='RMSprop',
metrics=['binary_accuracy'])
```

In [22]:
```
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=100)
```

```
Epoch 1/100
1/1 [==============================] - 3s 3s/step - loss: 0.4808 - binary_accuracy: 0.4667 - v
al_loss: 0.3832 - val_binary_accuracy: 0.5999
Epoch 2/100
1/1 [==============================] - 1s 1s/step - loss: 0.4720 - binary_accuracy: 0.4711 - v
al_loss: 0.3742 - val_binary_accuracy: 0.5999
Epoch 3/100
1/1 [==============================] - 1s 1s/step - loss: 0.4585 - binary_accuracy: 0.4711 - v
al_loss: 0.3586 - val_binary_accuracy: 0.5999
Epoch 4/100
1/1 [==============================] - 1s 1s/step - loss: 0.4352 - binary_accuracy: 0.4711 - v
al_loss: 0.3289 - val_binary_accuracy: 0.5998
Epoch 5/100
1/1 [==============================] - 1s 1s/step - loss: 0.3906 - binary_accuracy: 0.4720 - v
al_loss: 0.2709 - val_binary_accuracy: 0.5994
Epoch 6/100
1/1 [==============================] - 2s 2s/step - loss: 0.3093 - binary_accuracy: 0.4732 - v
al_loss: 0.2576 - val_binary_accuracy: 0.4000
Epoch 7/100
```

In [23]:
```
model.evaluate(X_train, y_train)
```

```
1/1 [==============================] - 0s 491ms/step - loss: 0.2517 - binary_accuracy: 0.4828
```

Out[23]:
```
[0.25165221095085144, 0.4827614426612854]
```

In [24]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 8)                 16

 dense_1 (Dense)             (None, 1)                 9

=================================================================
Total params: 25 (100.00 Byte)
Trainable params: 25 (100.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [25]:
```
#step5:
def performance_analysis(datadir, categories, img_size, nodes):
    df_results = pd.DataFrame(data=np.zeros(shape=(0, 5)),
                              columns=['Img size', 'Nodes Number', 'Accuracy', 'Loss', 'Training t
    training_data = []

    for category in categories:
        path = os.path.join(datadir, category)
        class_num = categories.index(category)
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
            num_array = cv2.resize(img_array, (img_size, img_size))
            training_data.append([num_array, class_num])

    X = []
    y = []

    for features, label in training_data:
        X.append(features)
        y.append(label)

    X = np.asarray(X).reshape(-1, img_size, img_size, 1)
    y = np.asarray(y)

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
    count = 0
    t_start = process_time()

    model = Sequential()
    model.add(Dense(nodes, input_dim=1, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='mean_squared_error', optimizer='RMSprop', metrics=['binary_accuracy'])
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=100)
    t_stop = process_time()
    t_elapsed = t_stop - t_start

    score = model.evaluate(X_test, y_test)

    count += 1
    df_results.loc[count, 'Img size'] = img_size
    df_results.loc[count, 'Nodes Number'] = nodes
    df_results.loc[count, 'Accuracy'] = score[1]
    df_results.loc[count, 'Loss'] = score[0]
    df_results.loc[count, 'Training time'] = t_elapsed

    return df_results
```

In [26]:
```
def evaluation():
    m1 = performance_analysis("C:\\Users\\veena\\Desktop\\bigdata", ['rose', 'tablerose'], 50, 8)
    m2 = performance_analysis("C:\\Users\\veena\\Desktop\\bigdata", ['rose', 'tablerose'], 50, 16)
    df = pd.concat([m1,m2])
    return df
```

In [27]:
```
evaluation()
```

```
Epoch 43/100
1/1 [==============================] - 0s 76ms/step - loss: 0.5078 - binary_accuracy: 0.4711 -
val_loss: 0.3995 - val_binary_accuracy: 0.5999
Epoch 44/100
1/1 [==============================] - 0s 78ms/step - loss: 0.5078 - binary_accuracy: 0.4711 -
val_loss: 0.3995 - val_binary_accuracy: 0.5999
Epoch 45/100
1/1 [==============================] - 0s 79ms/step - loss: 0.5077 - binary_accuracy: 0.4711 -
val_loss: 0.3995 - val_binary_accuracy: 0.5999
Epoch 46/100
1/1 [==============================] - 0s 103ms/step - loss: 0.5076 - binary_accuracy: 0.4711
- val_loss: 0.3995 - val_binary_accuracy: 0.5999
Epoch 47/100
1/1 [==============================] - 0s 80ms/step - loss: 0.5075 - binary_accuracy: 0.4711 -
val_loss: 0.3994 - val_binary_accuracy: 0.5999
Epoch 48/100
1/1 [==============================] - 0s 72ms/step - loss: 0.5075 - binary_accuracy: 0.4711 -
val_loss: 0.3994 - val_binary_accuracy: 0.5999
Epoch 49/100
1/1 [==============================] - 0s 99ms/step - loss: 0.5074 - binary_accuracy: 0.4711 -
```

In [28]:
```
training_data = []
img_size = 100

def create_training_data():
    for category in categories:
        path = os.path.join(df, category)
        class_num = categories.index(category)
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
            num_array = cv2.resize(img_array, (img_size, img_size))
            training_data.append([num_array, class_num])

create_training_data()

x = []
y = []
for features, label in training_data:
    x.append(features)
    y.append(label)
x = np.asarray(x)
x = x.reshape(-1, img_size, img_size, 1)
y = np.asarray(y)

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.25, random_state=42)

model = Sequential()
model.add(Dense(32, input_dim=1, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='mean_squared_error',
              optimizer='adam',
              metrics=['binary_accuracy'])

model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=100, batch_size=10, verbose=1
```

```
Epoch 1/100
1/1 [==============================] - 1s 1s/step - loss: 0.0035 - binary_accuracy: 1.0000 - v
al_loss: 0.0029 - val_binary_accuracy: 1.0000
Epoch 2/100
1/1 [==============================] - 0s 111ms/step - loss: 0.0035 - binary_accuracy: 1.0000
- val_loss: 0.0028 - val_binary_accuracy: 1.0000
Epoch 3/100
1/1 [==============================] - 0s 117ms/step - loss: 0.0035 - binary_accuracy: 1.0000
- val_loss: 0.0028 - val_binary_accuracy: 1.0000
Epoch 4/100
1/1 [==============================] - 0s 95ms/step - loss: 0.0034 - binary_accuracy: 1.0000 -
val_loss: 0.0027 - val_binary_accuracy: 1.0000
Epoch 5/100
1/1 [==============================] - 0s 85ms/step - loss: 0.0034 - binary_accuracy: 1.0000 -
val_loss: 0.0026 - val_binary_accuracy: 1.0000
Epoch 6/100
1/1 [==============================] - 0s 100ms/step - loss: 0.0034 - binary_accuracy: 1.0000
- val_loss: 0.0025 - val_binary_accuracy: 1.0000
Epoch 7/100
```

In [29]:
```
model.evaluate(x_test,y_test)
```

```
1/1 [==============================] - 0s 54ms/step - loss: 6.9848e-04 - binary_accuracy: 1.0000
```

Out[29]:
```
[0.0006984809297136962, 1.0]
```

In [30]:
```
training_data = []
img_size = 50

def create_training_data():
    for category in categories:
        path = os.path.join(df, category)
        class_num = categories.index(category)
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
            num_array = cv2.resize(img_array, (img_size, img_size))
            training_data.append([num_array, class_num])

create_training_data()

x = []
y = []
for features, label in training_data:
    x.append(features)
    y.append(label)
x = np.asarray(x)
x = x.reshape(-1, img_size, img_size, 1)
y = np.asarray(y)

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.25, random_state=42)

model = Sequential()
model.add(Dense(32, input_dim=1, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='mean_squared_error',
              optimizer='adam',
              metrics=['binary_accuracy'])

model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=100, batch_size=10, verbose=1
```

```
Epoch 1/100
1/1 [==============================] - 1s 1s/step - loss: 0.0043 - binary_accuracy: 1.0000 - v
al_loss: 0.0080 - val_binary_accuracy: 1.0000
Epoch 2/100
1/1 [==============================] - 0s 103ms/step - loss: 0.0041 - binary_accuracy: 1.0000
- val_loss: 0.0075 - val_binary_accuracy: 1.0000
Epoch 3/100
1/1 [==============================] - 0s 105ms/step - loss: 0.0040 - binary_accuracy: 1.0000
- val_loss: 0.0071 - val_binary_accuracy: 1.0000
Epoch 4/100
1/1 [==============================] - 0s 83ms/step - loss: 0.0039 - binary_accuracy: 1.0000 -
val_loss: 0.0066 - val_binary_accuracy: 1.0000
Epoch 5/100
1/1 [==============================] - 0s 84ms/step - loss: 0.0038 - binary_accuracy: 1.0000 -
val_loss: 0.0063 - val_binary_accuracy: 1.0000
Epoch 6/100
1/1 [==============================] - 0s 120ms/step - loss: 0.0037 - binary_accuracy: 1.0000
- val_loss: 0.0059 - val_binary_accuracy: 1.0000
Epoch 7/100
```

In [31]:
```
model.evaluate(x_test,y_test)
```

```
1/1 [==============================] - 0s 36ms/step - loss: 0.0010 - binary_accuracy: 1.0000
```

Out[31]:
```
[0.0010305397445335984, 1.0]
```

In [33]:
```
training_data = []
img_size = 25

def create_training_data():
    for category in categories:
        path = os.path.join(df, category)
        class_num = categories.index(category)
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
            num_array = cv2.resize(img_array, (img_size, img_size))
            training_data.append([num_array, class_num])

create_training_data()

x = []
y = []
for features, label in training_data:
    x.append(features)
    y.append(label)
x = np.asarray(x)
x = x.reshape(-1, img_size, img_size, 1)
y = np.asarray(y)

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.25, random_state=42)

model = Sequential()
model.add(Dense(32, input_dim=1, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='mean_squared_error',
              optimizer='adam',
              metrics=['binary_accuracy'])

model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=100, batch_size=10, verbose=1
```

```
Epoch 1/100
1/1 [==============================] - 1s 1s/step - loss: 0.9780 - binary_accuracy: 0.0069 - v
al_loss: 0.9372 - val_binary_accuracy: 0.0027
Epoch 2/100
1/1 [==============================] - 0s 65ms/step - loss: 0.9759 - binary_accuracy: 0.0069 -
val_loss: 0.9319 - val_binary_accuracy: 0.0027
Epoch 3/100
1/1 [==============================] - 0s 64ms/step - loss: 0.9733 - binary_accuracy: 0.0069 -
val_loss: 0.9255 - val_binary_accuracy: 0.0027
Epoch 4/100
1/1 [==============================] - 0s 77ms/step - loss: 0.9698 - binary_accuracy: 0.0069 -
val_loss: 0.9176 - val_binary_accuracy: 0.0027
Epoch 5/100
1/1 [==============================] - 0s 77ms/step - loss: 0.9652 - binary_accuracy: 0.0069 -
val_loss: 0.9076 - val_binary_accuracy: 0.0027
Epoch 6/100
1/1 [==============================] - 0s 64ms/step - loss: 0.9587 - binary_accuracy: 0.0069 -
val_loss: 0.8946 - val_binary_accuracy: 0.0027
Epoch 7/100
```

In [34]:
```
model.evaluate(x_test,y_test)
```

```
1/1 [==============================] - 0s 60ms/step - loss: 0.0087 - binary_accuracy: 1.0000
```

Out[34]:
```
[0.008681504055857658, 1.0]
```

In [35]:
```
training_data = []
img_size = 10

def create_training_data():
    for category in categories:
        path = os.path.join(df, category)
        class_num = categories.index(category)
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
            num_array = cv2.resize(img_array, (img_size, img_size))
            training_data.append([num_array, class_num])

create_training_data()

x = []
y = []
for features, label in training_data:
    x.append(features)
    y.append(label)
x = np.asarray(x)
x = x.reshape(-1, img_size, img_size, 1)
y = np.asarray(y)

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.25, random_state=42)

model = Sequential()
model.add(Dense(32, input_dim=1, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='mean_squared_error',
              optimizer='adam',
              metrics=['binary_accuracy'])

model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=100, batch_size=10, verbose=1
```

```
Epoch 1/100
1/1 [==============================] - 1s 1s/step - loss: 0.9979 - binary_accuracy: 0.0000e+00
- val_loss: 0.9876 - val_binary_accuracy: 0.0000e+00
Epoch 2/100
1/1 [==============================] - 0s 68ms/step - loss: 0.9978 - binary_accuracy: 0.0000e+
00 - val_loss: 0.9872 - val_binary_accuracy: 0.0000e+00
Epoch 3/100
1/1 [==============================] - 0s 62ms/step - loss: 0.9977 - binary_accuracy: 0.0000e+
00 - val_loss: 0.9867 - val_binary_accuracy: 0.0000e+00
Epoch 4/100
1/1 [==============================] - 0s 51ms/step - loss: 0.9976 - binary_accuracy: 0.0000e+
00 - val_loss: 0.9863 - val_binary_accuracy: 0.0000e+00
Epoch 5/100
1/1 [==============================] - 0s 68ms/step - loss: 0.9974 - binary_accuracy: 0.0000e+
00 - val_loss: 0.9858 - val_binary_accuracy: 0.0000e+00
Epoch 6/100
1/1 [==============================] - 0s 89ms/step - loss: 0.9973 - binary_accuracy: 0.0000e+
00 - val_loss: 0.9852 - val_binary_accuracy: 0.0000e+00
Epoch 7/100
```

In [36]:
```
model.evaluate(x_test,y_test)
```

```
1/1 [==============================] - 0s 48ms/step - loss: 0.0039 - binary_accuracy: 1.0000
```

Out[36]:
```
[0.0038982636760920286, 1.0]
```

In [37]:
```
training_data = []
img_size=500
def create_training_data():
    for category in categories:
        path = os.path.join(df,category)
        class_num = categories.index(category)
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
            num_array=cv2.resize(img_array,(img_size,img_size))
            training_data.append([num_array,class_num])
create_training_data()

x = []
y = []
for features, label in training_data:
    x.append(features)
    y.append(label)
x = np.asarray(x)
x = x.reshape(-1, img_size, img_size, 1)
y = np.asarray(y)


from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.25,random_state=42)
model =Sequential()
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='mean_squared_error',
optimizer='adam',
metrics=['binary_accuracy'])
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=100,batch_size=10,verbose=1)
```

```
Epoch 1/100
1/1 [==============================] - 3s 3s/step - loss: 0.9528 - binary_accuracy: 0.0090 - v
al_loss: 0.8675 - val_binary_accuracy: 0.0021
Epoch 2/100
1/1 [==============================] - 2s 2s/step - loss: 0.9261 - binary_accuracy: 0.0090 - v
al_loss: 0.8025 - val_binary_accuracy: 0.0021
Epoch 3/100
1/1 [==============================] - 2s 2s/step - loss: 0.8653 - binary_accuracy: 0.0090 - v
al_loss: 0.6710 - val_binary_accuracy: 0.0050
Epoch 4/100
1/1 [==============================] - 2s 2s/step - loss: 0.7251 - binary_accuracy: 0.0146 - v
al_loss: 0.4130 - val_binary_accuracy: 0.0178
Epoch 5/100
1/1 [==============================] - 2s 2s/step - loss: 0.4371 - binary_accuracy: 0.0164 - v
al_loss: 0.1413 - val_binary_accuracy: 1.0000
Epoch 6/100
1/1 [==============================] - 2s 2s/step - loss: 0.1285 - binary_accuracy: 1.0000 - v
al_loss: 0.0523 - val_binary_accuracy: 1.0000
Epoch 7/100
```

In [38]:
```
model.evaluate(x_test,y_test)
```

```
1/1 [==============================] - 0s 232ms/step - loss: 0.0057 - binary_accuracy: 1.0000
```

Out[38]:
```
[0.005736566614359617, 1.0]
```

In [40]:
```
#3 Layer
training_data = []
img_size=500
def create_training_data():
    for category in categories:
        path = os.path.join(df,category)
        class_num = categories.index(category)
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
            num_array=cv2.resize(img_array,(img_size,img_size))
            training_data.append([num_array,class_num])
create_training_data()

x=[]
y=[]
for features,label in training_data:
    x.append(features)
    y.append(label)
x=np.asarray(x).reshape(-1,img_size,img_size,1)
y=np.asarray(y)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.25,random_state=42)
model =Sequential()
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(1,activation='sigmoid'))

model.compile(loss='mean_squared_error',
metrics=['binary_accuracy'])
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=100,batch_size=10,verbose=1)
```

```
Epoch 1/100
1/1 [==============================] - 3s 3s/step - loss: 0.9875 - binary_accuracy: 0.0090 - v
al_loss: 0.9785 - val_binary_accuracy: 0.0021
Epoch 2/100
1/1 [==============================] - 1s 1s/step - loss: 0.9868 - binary_accuracy: 0.0090 - v
al_loss: 0.9743 - val_binary_accuracy: 0.0021
Epoch 3/100
1/1 [==============================] - 1s 1s/step - loss: 0.9860 - binary_accuracy: 0.0090 - v
al_loss: 0.9682 - val_binary_accuracy: 0.0021
Epoch 4/100
1/1 [==============================] - 1s 1s/step - loss: 0.9847 - binary_accuracy: 0.0090 - v
al_loss: 0.9577 - val_binary_accuracy: 0.0021
Epoch 5/100
1/1 [==============================] - 1s 1s/step - loss: 0.9817 - binary_accuracy: 0.0090 - v
al_loss: 0.9330 - val_binary_accuracy: 0.0021
Epoch 6/100
1/1 [==============================] - 1s 1s/step - loss: 0.9715 - binary_accuracy: 0.0090 - v
al_loss: 0.8188 - val_binary_accuracy: 0.0084
Epoch 7/100
```

In [41]:
```
model.evaluate(x_test,y_test)
```

```
1/1 [==============================] - 0s 269ms/step - loss: 0.0017 - binary_accuracy: 1.0000
```

Out[41]:
```
[0.001691819983534515, 1.0]
```

In [42]:
```
#4 Layer
training_data = []
img_size=500
def create_training_data():
    for category in categories:
        path = os.path.join(df,category)
        class_num = categories.index(category)
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
            num_array=cv2.resize(img_array,(img_size,img_size))
            training_data.append([num_array,class_num])
create_training_data()

x = []
y = []
for features, label in training_data:
    x.append(features)
    y.append(label)
x = np.asarray(x)
x = x.reshape(-1, img_size, img_size, 1)
y = np.asarray(y)


from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.25,random_state=42)
model =Sequential()
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='mean_squared_error',
metrics=['binary_accuracy'])
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=100,batch_size=10,verbose=1)
```

```
Epoch 1/100
1/1 [==============================] - 3s 3s/step - loss: 0.0117 - binary_accuracy: 1.0000 - v
al_loss: 0.0088 - val_binary_accuracy: 1.0000
Epoch 2/100
1/1 [==============================] - 2s 2s/step - loss: 0.0047 - binary_accuracy: 1.0000 - v
al_loss: 0.0084 - val_binary_accuracy: 1.0000
Epoch 3/100
1/1 [==============================] - 2s 2s/step - loss: 0.0046 - binary_accuracy: 1.0000 - v
al_loss: 0.0081 - val_binary_accuracy: 1.0000
Epoch 4/100
1/1 [==============================] - 2s 2s/step - loss: 0.0045 - binary_accuracy: 1.0000 - v
al_loss: 0.0078 - val_binary_accuracy: 1.0000
Epoch 5/100
1/1 [==============================] - 2s 2s/step - loss: 0.0044 - binary_accuracy: 1.0000 - v
al_loss: 0.0075 - val_binary_accuracy: 1.0000
Epoch 6/100
1/1 [==============================] - 2s 2s/step - loss: 0.0043 - binary_accuracy: 1.0000 - v
al_loss: 0.0072 - val_binary_accuracy: 1.0000
Epoch 7/100
```

In [43]:
```
model.evaluate(x_test,y_test)
```

```
1/1 [==============================] - 0s 325ms/step - loss: 3.1647e-04 - binary_accuracy: 1.0000
```

Out[43]:
```
[0.0003164731024298817, 1.0]
```

In [44]:
```
#5 Layer
training_data = []
img_size=500
def create_training_data():
    for category in categories:
        path = os.path.join(df,category)
        class_num = categories.index(category)
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
            num_array=cv2.resize(img_array,(img_size,img_size))
            training_data.append([num_array,class_num])
create_training_data()

x = []
y = []
for features, label in training_data:
    x.append(features)
    y.append(label)
x = np.asarray(x)
x = x.reshape(-1, img_size, img_size, 1)
y = np.asarray(y)


from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.25,random_state=42)
model =Sequential()
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(32,input_dim=1,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='mean_squared_error',
metrics=['binary_accuracy'])
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=100,batch_size=10,verbose=1)
```

```
Epoch 1/100
1/1 [==============================] - 4s 4s/step - loss: 0.0057 - binary_accuracy: 1.0000 - v
al_loss: 0.0061 - val_binary_accuracy: 1.0000
Epoch 2/100
1/1 [==============================] - 2s 2s/step - loss: 0.0041 - binary_accuracy: 1.0000 - v
al_loss: 0.0054 - val_binary_accuracy: 1.0000
Epoch 3/100
1/1 [==============================] - 2s 2s/step - loss: 0.0039 - binary_accuracy: 1.0000 - v
al_loss: 0.0049 - val_binary_accuracy: 1.0000
Epoch 4/100
1/1 [==============================] - 2s 2s/step - loss: 0.0038 - binary_accuracy: 1.0000 - v
al_loss: 0.0045 - val_binary_accuracy: 1.0000
Epoch 5/100
1/1 [==============================] - 2s 2s/step - loss: 0.0037 - binary_accuracy: 1.0000 - v
al_loss: 0.0042 - val_binary_accuracy: 1.0000
Epoch 6/100
1/1 [==============================] - 2s 2s/step - loss: 0.0036 - binary_accuracy: 1.0000 - v
al_loss: 0.0039 - val_binary_accuracy: 1.0000
Epoch 7/100
```

In [ ]: