Lab4.: Image corpus creation and binary classification using DNN

Name:P.Asha Belcilda

Rollno:225229104 ¶

1.Dataset Creation:

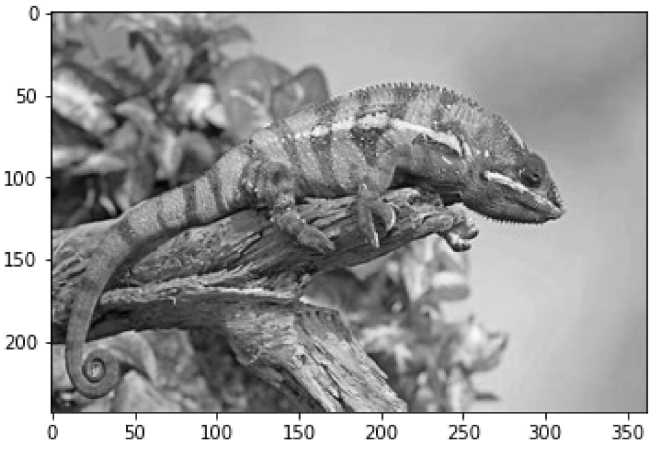
```
Dataset is created and the images are stored in separate folders for each class under
one folder name 'Image'.
Two classes of images are created they are:
    1.Chameleon
    2.Garden Lizard
```

2.Pre-Processing:

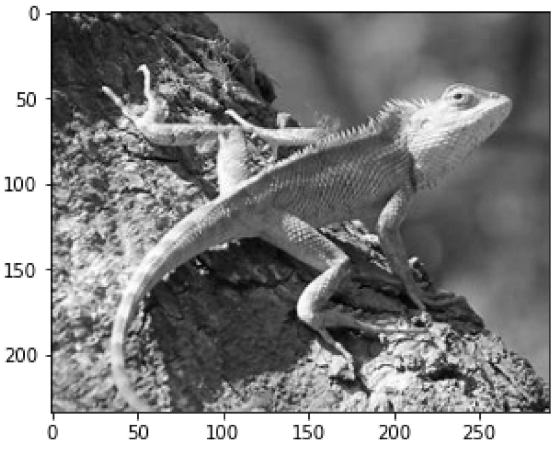
In [2]:
```
import os
import cv2
import time
%matplotlib inline
import numpy as np
import pandas as pd
from time import process_time
import matplotlib.pyplot as plt
```

In [3]:
```
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 [4]:
```
datadir ="Image"
categories =['Chameleon']
for category in categories:
    path = os.path.join(datadir, 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 [5]:
```
datadir ="Image"
categories =['lizard']
for category in categories:
    path = os.path.join(datadir, 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 [6]:
```
datadir = "Image"
categories = ['Chameleon', 'lizard']
```

In [7]:
```
data = []
img_size=500

def preprocess():
     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))
            data.append([num_array, class_num])
preprocess()
```

In [8]:
```
print(len(data))
```

```
16
```

3. Dataset Preparation:

In [10]:
```
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 [11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

In [12]:
```
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 = (12, 500, 500, 1)
X_test = (4, 500, 500, 1)
y_train = (12,)
y_test = (4,)
```

4.Model Creation:

In [13]:
```
model = Sequential()
model.add(Dense(8, input_dim=1, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [14]:
```
model.compile(loss='mean_squared_error', metrics=['binary_accuracy'])
```

In [15]:
```
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=100)
```

```
Epoch 1/100
1/1 [==============================] - 6s 6s/step - loss: 0.5826 - binary_accuracy:
0.4167 - val_loss: 0.2498 - val_binary_accuracy: 0.7499
Epoch 2/100
1/1 [==============================] - 2s 2s/step - loss: 0.5826 - binary_accuracy:
0.4168 - val_loss: 0.2498 - val_binary_accuracy: 0.7499
Epoch 3/100
1/1 [==============================] - 1s 1s/step - loss: 0.5826 - binary_accuracy:
0.4168 - val_loss: 0.2498 - val_binary_accuracy: 0.7499
Epoch 4/100
1/1 [==============================] - 1s 1s/step - loss: 0.5826 - binary_accuracy:
0.4168 - val_loss: 0.2498 - val_binary_accuracy: 0.7499
Epoch 5/100
1/1 [==============================] - 1s 1s/step - loss: 0.5826 - binary_accuracy:
0.4168 - val_loss: 0.2498 - val_binary_accuracy: 0.7499
Epoch 6/100
1/1 [==============================] - 1s 1s/step - loss: 0.5826 - binary_accuracy:
0.4168 - val_loss: 0.2498 - val_binary_accuracy: 0.7499
Epoch 7/100
```

In [16]:
```
model.evaluate(X_train, y_train)
```

```
1/1 [==============================] - 1s 682ms/step - loss: 0.5824 - binary_accuracy:
0.4168
```

Out[16]:
```
[0.5823569893836975, 0.4168431758880615]
```

In [17]:
```
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)
_________________________________________________________________
```

5 Performance Analysis:

In [35]:
```
training_data = []
img_size=500
def create_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])
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(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,verbos
```

```
Epoch 1/100
1/1 [==============================] - 10s 10s/step - loss: 0.9923 - binary_accuracy:
0.0000e+00 - val_loss: 0.9881 - val_binary_accuracy: 2.6000e-05
Epoch 2/100
1/1 [==============================] - 4s 4s/step - loss: 0.9814 - binary_accuracy:
4.4400e-04 - val_loss: 0.9610 - val_binary_accuracy: 2.6000e-05
Epoch 3/100
1/1 [==============================] - 4s 4s/step - loss: 0.9363 - binary_accuracy:
4.4400e-04 - val_loss: 0.7203 - val_binary_accuracy: 0.0011
Epoch 4/100
1/1 [==============================] - 4s 4s/step - loss: 0.6621 - binary_accuracy:
0.0027 - val_loss: 0.0101 - val_binary_accuracy: 1.0000
Epoch 5/100
1/1 [==============================] - 4s 4s/step - loss: 0.0164 - binary_accuracy:
1.0000 - val_loss: 0.0073 - val_binary_accuracy: 1.0000
Epoch 6/100
1/1 [==============================] - 4s 4s/step - loss: 0.0121 - binary_accuracy:
1.0000 - val_loss: 0.0063 - val_binary_accuracy: 1.0000
Epoch 7/100
```

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

```
1/1 [==============================] - 1s 553ms/step - loss: 1.4540e-04 - binary_accurac
y: 1.0000
```

Out[38]:
```
[0.00014540493430104107, 1.0]
```