PDL Lab4.: Image corpus creation and binary classification using DNN

DINESH KUMAR 225229108

In [49]:
```
import os
import cv2
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split
```

In [50]:
```
barn_owl_folder = "C:/Users/2mscdsa08/Desktop/owl"
similar_images_folder = 'C:/Users/2mscdsa08/Desktop/apple'
```

In [51]:
```
images = []
labels = []

# Read barn owl images
for filename in os.listdir(barn_owl_folder):
    if not filename.endswith(('.jpg', '.jpeg', '.png')):
        continue

    image_path = os.path.join(barn_owl_folder, filename)
    image = cv2.imread(image_path)
    image = cv2.resize(image, (64, 64))
    image = image / 255.0

    images.append(image)
    labels.append(0)  # Label 0 for barn owls
```

In [52]:
```
# Read similar images
for filename in os.listdir(similar_images_folder):
    if not filename.endswith(('.jpg', '.jpeg', '.png')):
        continue

    image_path = os.path.join(similar_images_folder, filename)
    image = cv2.imread(image_path)
    image = cv2.resize(image, (64, 64))
    image = image / 255.0

    images.append(image)
    labels.append(1)  # Label 1 for similar images
```

In [53]:
```
# Convert the lists to numpy arrays
images = np.array(images)
labels = np.array(labels)
```

In [54]:
```
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)
```

In [55]:
```
# Define the model
model = Sequential()
model.add(Flatten(input_shape=(64, 64, 3)))
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [56]:
```
# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])
```

In [57]:
```
# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32)
```

```
Epoch 1/10
1/1 [==============================] - 1s 551ms/step - loss: 0.7043 - accuracy: 0.5714
Epoch 2/10
1/1 [==============================] - 0s 39ms/step - loss: 5.8556 - accuracy: 0.4286
Epoch 3/10
1/1 [==============================] - 0s 7ms/step - loss: 2.2219 - accuracy: 0.5714
Epoch 4/10
1/1 [==============================] - 0s 38ms/step - loss: 2.5482 - accuracy: 0.5714
Epoch 5/10
1/1 [==============================] - 0s 38ms/step - loss: 0.8106 - accuracy: 0.5714
Epoch 6/10
1/1 [==============================] - 0s 12ms/step - loss: 1.8267 - accuracy: 0.4286
Epoch 7/10
1/1 [==============================] - 0s 9ms/step - loss: 1.2678 - accuracy: 0.4286
Epoch 8/10
1/1 [==============================] - 0s 40ms/step - loss: 0.2648 - accuracy: 0.9286
Epoch 9/10
1/1 [==============================] - 0s 13ms/step - loss: 1.0379 - accuracy: 0.5714
Epoch 10/10
1/1 [==============================] - 0s 7ms/step - loss: 0.9499 - accuracy: 0.5714
```

Out[57]:
```
<keras.callbacks.History at 0x2597f7ecfd0>
```

In [34]:
```
# Evaluate the model on test set
loss, accuracy = model.evaluate(X_test, y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

```
1/1 [==============================] - 0s 305ms/step - loss: 0.0756 - accuracy: 1.0000
Test Loss: 0.07557893544435501
Test Accuracy: 1.0
```

In [35]:
```
# Predict class labels for test images
test_predictions = model.predict(X_test)
```

```
1/1 [==============================] - 0s 58ms/step
```

In [36]:
```
# Convert probabilities to class labels (0 or 1)
test_predictions = np.round(test_predictions).flatten()
```

In [37]:
```
# Print the predicted labels and the actual labels
print("Predicted Labels:", test_predictions)
print("Actual Labels:", y_test)
```

```
Predicted Labels: [0. 0. 0. 0.]
Actual Labels: [0 0 0 0]
```

In [38]:
```
# Save the image corpus and labels
np.save('image_corpus.npy', image_corpus)
np.save('labels.npy', labels)
```

In [39]:
```
# Load the image corpus and labels
image_corpus = np.load('image_corpus.npy')
labels = np.load('labels.npy')

# Print the shapes and contents of the loaded arrays
print("Image Corpus shape:", image_corpus.shape)
print("Image Corpus:")
print(image_corpus)

print("\nLabels shape:", labels.shape)
print("Labels:")
print(labels)
```

```
Image Corpus shape: (18, 1)
Image Corpus:
[[0.54886806]
 [0.5354678 ]
 [0.5491303 ]
 [0.5677061 ]
 [0.550068  ]
 [0.52372587]
 [0.5308342 ]
 [0.5455766 ]
 [0.55411303]
 [0.55990237]
 [0.5908376 ]
 [0.594258  ]
 [0.5926563 ]
 [0.5691223 ]
 [0.6011374 ]
 [0.58412766]
 [0.60285765]
 [0.6070214 ]]

Labels shape: (18,)
Labels:
[0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1]
```

In [ ]: