task4

May 15, 2024

1 Task 4

In this task, we aim to build a machine learning model capable of distinguishing between pictures of cars and guns. We start by extracting car images from the CIFAR-10 dataset and collecting four gun images from the internet. After preprocessing both sets of images to ensure uniformity in size and format, we combine them into a single dataset. Then, we construct a convolutional neural network (CNN) model, which is a type of deep learning architecture suitable for image classification tasks. The model is trained on the combined dataset, learning to differentiate between cars and guns based on their visual features. Finally, we evaluate the trained model's performance. Although accuracy is not considered in this task, we can still assess the model's ability to distinguish between the two classes.

[37]:

```python
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
import matplotlib.pyplot as plt
import os
import cv2
from sklearn.model_selection import train_test_split

# Load CIFAR-10 dataset
(x_train, y_train), (_, _) = cifar10.load_data()

# Extract car images (label 1 for 'automobile')
car_images = x_train[y_train.flatten() == 1]
car_images = car_images[:4]  # Use only 4 car images for simplicity

# Directory to store downloaded gun images
gun_dir = 'guns/'

# Resize and format gun images
gun_images = []
gun_image_files = []  # To store filenames for visualization
for filename in os.listdir(gun_dir):
    if filename.endswith('.jpg') or filename.endswith('.png'):
        img = cv2.imread(os.path.join(gun_dir, filename))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Convert to RGB format
        img = cv2.resize(img, (32, 32))  # Resize to match CIFAR-10 image size
        gun_images.append(img)
        gun_image_files.append(filename)

gun_images = np.array(gun_images)

# Combine car and gun images into one dataset
X = np.concatenate((car_images, gun_images), axis=0)
y = np.array([0] * len(car_images) + [1] * len(gun_images))  # 0 for car, 1 for gun

# Shuffle the dataset
indices = np.arange(X.shape[0])
np.random.shuffle(indices)
X = X[indices]
y = y[indices]

# Normalize the images
X = X.astype('float32') / 255.0

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Define the CNN model
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=10, batch_size=2, validation_split=0.2)

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {accuracy * 100:.2f}%')
```

```python
# Display sample gun images
plt.figure(figsize=(10, 4))
for i in range(4):
    plt.subplot(1, 4, i + 1)
    plt.imshow(gun_images[i])
    plt.title(f'Gun ({gun_image_files[i]})')
    plt.axis('off')
plt.show()
```

```
Epoch 1/10

libpng warning: iCCP: profile 'icc': 0h: PCS illuminant is not D50

2/2             0s 47ms/step -
accuracy: 0.6667 - loss: 0.8787 - val_accuracy: 0.5000 - val_loss: 0.8338
Epoch 2/10
2/2             0s 9ms/step -
accuracy: 0.3333 - loss: 0.8820 - val_accuracy: 1.0000 - val_loss: 0.6878
Epoch 3/10
2/2             0s 8ms/step -
accuracy: 1.0000 - loss: 0.6555 - val_accuracy: 0.5000 - val_loss: 0.6671
Epoch 4/10
2/2             0s 9ms/step -
accuracy: 1.0000 - loss: 0.6343 - val_accuracy: 0.5000 - val_loss: 0.6627
Epoch 5/10
2/2             0s 8ms/step -
accuracy: 1.0000 - loss: 0.6363 - val_accuracy: 0.5000 - val_loss: 0.6534
Epoch 6/10
2/2             0s 13ms/step -
accuracy: 1.0000 - loss: 0.5833 - val_accuracy: 0.5000 - val_loss: 0.6394
Epoch 7/10
2/2             0s 9ms/step -
accuracy: 1.0000 - loss: 0.5430 - val_accuracy: 1.0000 - val_loss: 0.6205
Epoch 8/10
2/2             0s 8ms/step -
accuracy: 1.0000 - loss: 0.5051 - val_accuracy: 1.0000 - val_loss: 0.5908
Epoch 9/10
2/2             0s 9ms/step -
accuracy: 1.0000 - loss: 0.4474 - val_accuracy: 1.0000 - val_loss: 0.5584
Epoch 10/10
2/2             0s 9ms/step -
accuracy: 1.0000 - loss: 0.3862 - val_accuracy: 1.0000 - val_loss: 0.5217
1/1             0s 38ms/step -
accuracy: 1.0000 - loss: 0.6025
Test Accuracy: 100.00%
```

Please use the “Kernel>Restart & Run All” command in Jupyter Notebook and check your results before submitting your homework. Note that I rerun all boxes on my side before grading.

[ ]: