Name : Ketan Gandhi

Reg. No. :21BPS1628

Lab 8

PICAM- Object classification and Object detection

Code for binary classification:

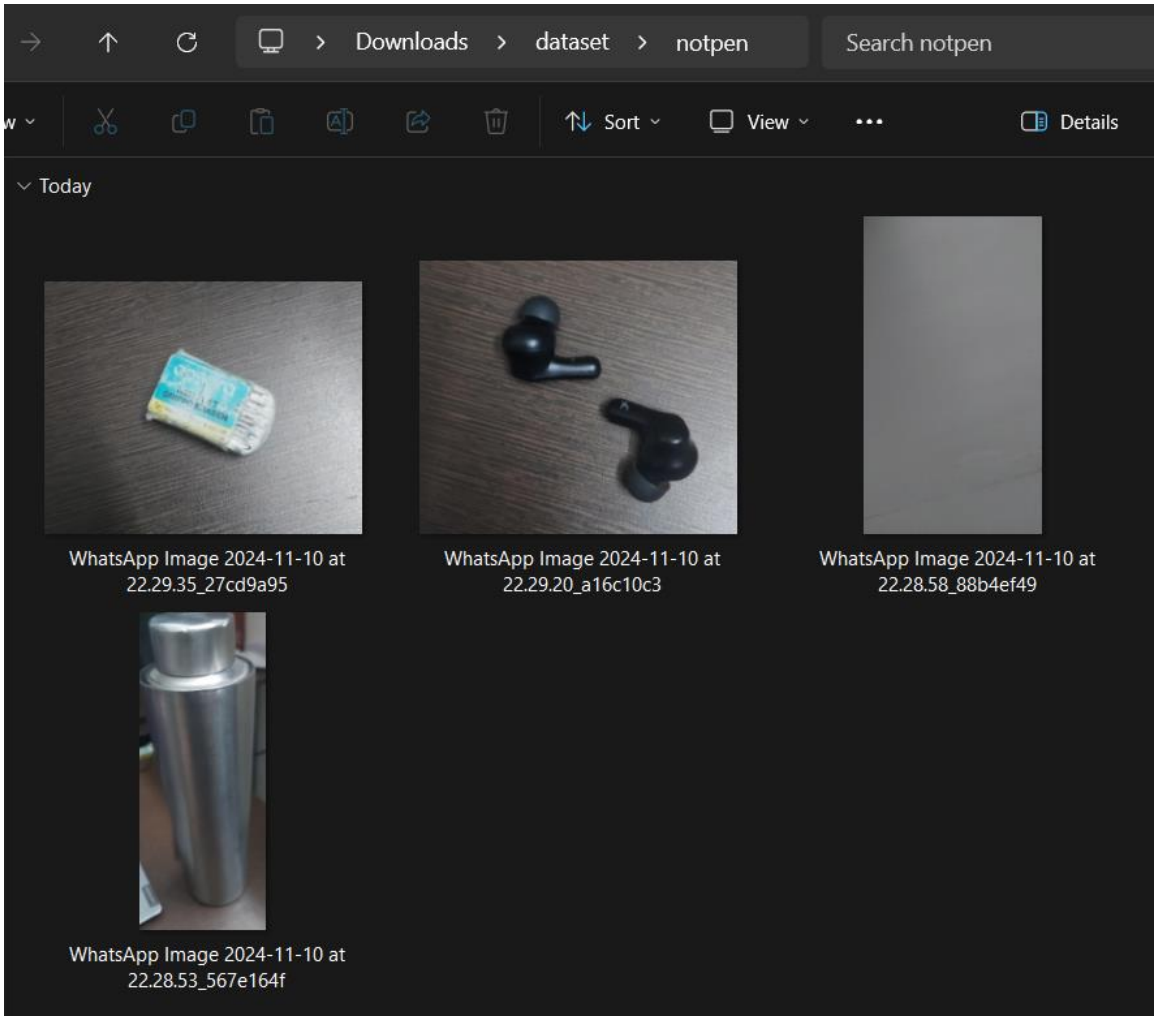
```
import os
import numpy as np
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from sklearn.model_selection import train_test_split
import glob

# Set image dimensions
image_size = (128, 128)  # Resize to 128x128 pixels for consistency

# Load image paths for "pen" and "not_pen"
pen_images = glob.glob('/content/pen/*.jpeg')  # Update path
not_pen_images = glob.glob('/content/not_pen/*.jpeg')  # Update path

# Debug: Print the number of images found
print(f"Number of pen images: {len(pen_images)}")
print(f"Number of not_pen images: {len(not_pen_images)}")

# Initialize lists to hold image data and labels
images = []
labels = []

# Load and preprocess images
```

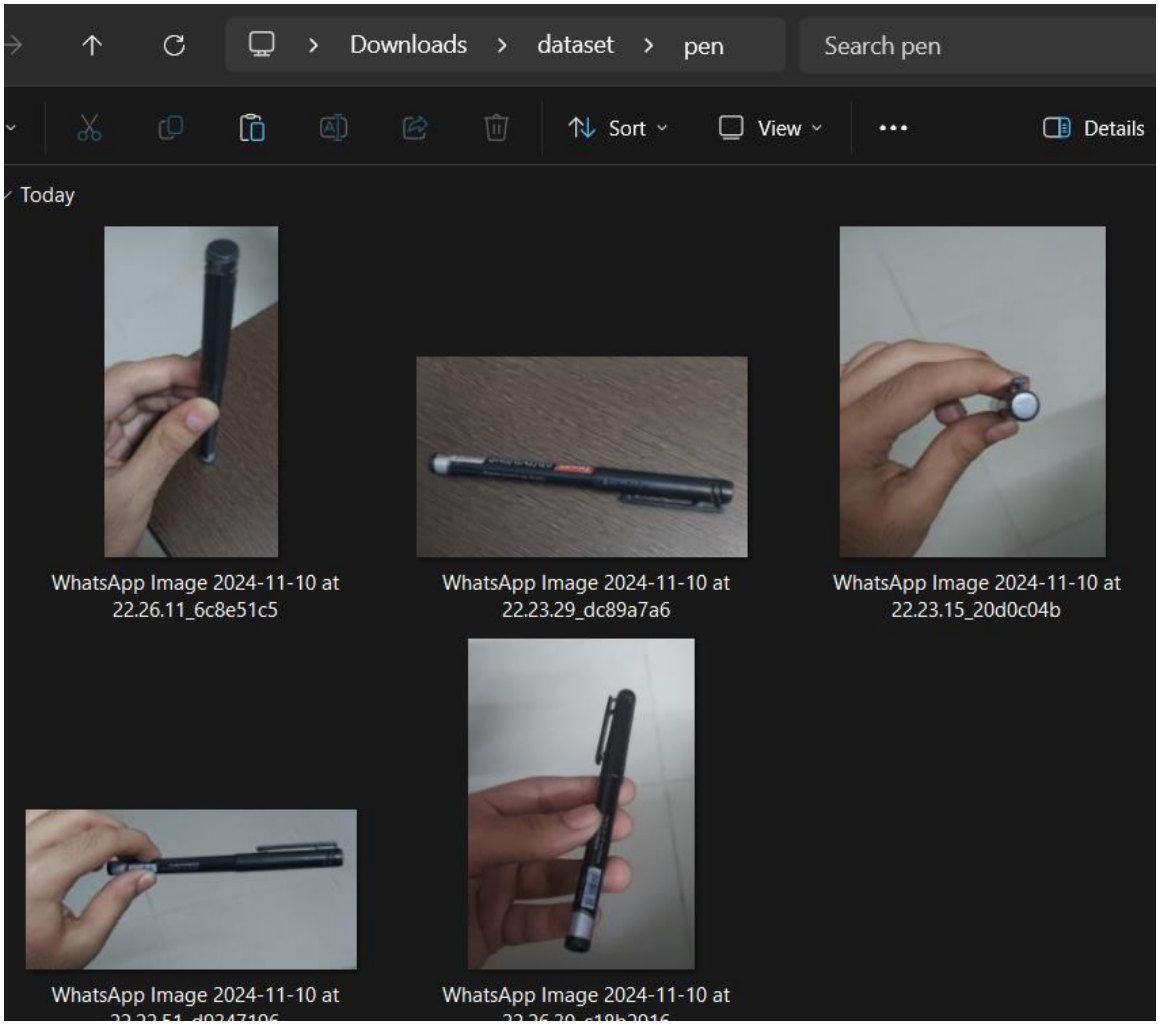
```
for img_path in pen_images:
    img = load_img(img_path, target_size=image_size)
    img_array = img_to_array(img) / 255.0  # Normalize pixel values to [0, 1]
    images.append(img_array)
    labels.append(1)  # Label for "pen"

for img_path in not_pen_images:
    img = load_img(img_path, target_size=image_size)
    img_array = img_to_array(img) / 255.0  # Normalize pixel values to [0, 1]
    images.append(img_array)
    labels.append(0)  # Label for "not pen"

# Convert lists to numpy arrays
images = np.array(images)
labels = np.array(labels)

# Check if images and labels are loaded correctly
if len(images) == 0 or len(labels) == 0:
    raise ValueError("No images were loaded. Check the file paths and ensure images are present.")

# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)

print(f"Training samples: {len(X_train)}")
print(f"Test samples: {len(X_test)}")

# CNN model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
```

```
# Build a simple CNN model
model = Sequential()

# First convolutional layer
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Second convolutional layer
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Third convolutional layer
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Flatten the output of the convolutional layers
model.add(Flatten())

# Fully connected layer
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))  # Dropout for regularization

# Output layer for binary classification
model.add(Dense(1, activation='sigmoid'))  # Sigmoid activation for binary classification

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

# Set up early stopping to avoid overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=3)
```

```
# Train the model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=20, batch_size=32, callbacks=[early_stopping])


# Save the model after training
model.save('/content/drive/MyDrive/pen_classifier_model.h5')  # Save to Drive


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

Dataset generation:


Downloads
dataset
notpen
Search notpen
Sort
View
Details
Today
WhatsApp Image 2024-11-10 at 22.29.35_27cd9a95
WhatsApp Image 2024-11-10 at 22.29.20_a16c10c3
WhatsApp Image 2024-11-10 at 22.28.58_88b4ef49
WhatsApp Image 2024-11-10 at 22.28.53_567e164f



Output:

```
Number of pen images: 7
Number of not_pen images: 4
Training samples: 8
Test samples: 3
Epoch 1/20
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. Whe
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step - accuracy: 0.3750 - loss: 0.6988 - val_accuracy: 0.6667 - val_loss: 0.6956
Epoch 2/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 648ms/step - accuracy: 0.6250 - loss: 0.5216 - val_accuracy: 0.6667 - val_loss: 0.7729
Epoch 3/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.6250 - loss: 0.7479 - val_accuracy: 0.3333 - val_loss: 0.6995
Epoch 4/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step - accuracy: 0.7500 - loss: 0.5532 - val_accuracy: 0.3333 - val_loss: 0.7436
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend usir
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.3333 - loss: 0.7436
Test Accuracy: 72.41%
```

Test Image:

Output

```
# prompt: test the model with a single image

import numpy as np
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.models import load_model

# Load the trained model
model = load_model('/content/drive/MyDrive/pen_classifier_model.h5')

# Define the image size used during training
image_size = (128, 128)

# Path to the image you want to test
image_path = '/content/test_image.jpeg' # Replace with the actual path

# Load and preprocess the image
img = load_img(image_path, target_size=image_size)
img_array = img_to_array(img) / 255.0
img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension

# Make prediction
prediction = model.predict(img_array)

# Interpret the prediction
if prediction[0][0] > 0.5:
    print("Prediction: Pen")
else:
    print("Prediction: Not Pen")

print(f"Prediction Probability: {prediction[0][0]}")
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evalu
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Prediction: Pen
Prediction Probability: 0.5121869444847107
```