

21BPS1615

VIKRAM SINGH

NOV 10, 2024

Experiment: Object Classification and Object Detection with YOLO

Objective

The objective of this experiment is to implement an object classification and detection system using YOLOv5 and a custom dataset. The process involves building a classification model for recognizing two classes: 'Pen' and 'No_Pen', and then using YOLO for object detection. The following steps outline the methodology used to achieve the desired results.

Step 1: Dataset Creation

In this step, a custom dataset was created by gathering images of a pen and images where no pen was present. The images were divided into two categories: 'Pen' and 'No_Pen'.

Folder Structure:

```
scss
Copy code
data/
├── Pen/
│   └── (Images of pens)
└── No_Pen/
    └── (Images without pens)
```



Step 2: Image Capturing & Classification

The dataset is prepared by capturing images by using the following script:

```
# scripts/capture_images.py
from picamera import PiCamera
from time import sleep
```

```

import os

camera = PiCamera()
output_dir = 'data/Pen/'
os.makedirs(output_dir, exist_ok=True)

for i in range(100): # Capture 100 images
    sleep(2)
    image_path = f"{output_dir}/pen_{i}.jpg"
    camera.capture(image_path)
    print(f"Captured {image_path}")

```

I have only captured 4 images 3 of Pens and 1 of NoPen which is a mouse which may have resulted into overfitting of data but the model still works.

In this step, a Convolutional Neural Network (CNN) was implemented to classify images into 'Pen' or 'No_Pen'. The code for the image classification model is provided below:

```

python
import os
import cv2
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score,
recall_score

IMG_SIZE = 224
EPOCHS = 10
BATCH_SIZE = 32

# Load images and labels
def load_images(folder):
    images, labels = [], []
    for label, class_name in enumerate(['Pen', 'No_Pen']):
        path = os.path.join(folder, class_name)
        for img_name in os.listdir(path):
            img = cv2.imread(os.path.join(path, img_name))
            img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
            images.append(img / 255.0)

```

```

        labels.append(label)
    return np.array(images), np.array(labels)

X, y = load_images('data/')
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)

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

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=EPOCHS, batch_size=BATCH_SIZE,
validation_data=(X_val, y_val))

# Evaluate
y_pred = (model.predict(X_val) > 0.5).astype(int)
print(f"Accuracy: {accuracy_score(y_val, y_pred):.2f}")
print(f"Precision: {precision_score(y_val, y_pred):.2f}")
print(f"Recall: {recall_score(y_val, y_pred):.2f}")

# Save the model
model.save('models/classifier.h5')

```

```
Epoch 1/10
1/1 ----- 3s 3s/step - accuracy: 0.6667 - loss: 0.6528 - val_accuracy: 1.0000 - val_loss: 1.6878e-20
Epoch 2/10
1/1 ----- 0s 218ms/step - accuracy: 0.6667 - loss: 13.4028 - val_accuracy: 1.0000 - val_loss: 3.6366e-06
Epoch 3/10
1/1 ----- 0s 182ms/step - accuracy: 0.6667 - loss: 2.6737 - val_accuracy: 0.0000e+00 - val_loss: 12.5853
Epoch 4/10
1/1 ----- 0s 191ms/step - accuracy: 0.3333 - loss: 8.2616 - val_accuracy: 0.0000e+00 - val_loss: 8.6666
Epoch 5/10
1/1 ----- 0s 187ms/step - accuracy: 0.3333 - loss: 5.6240 - val_accuracy: 0.0000e+00 - val_loss: 3.1825
Epoch 6/10
1/1 ----- 0s 182ms/step - accuracy: 0.3333 - loss: 1.9919 - val_accuracy: 1.0000 - val_loss: 0.0552
Epoch 7/10
1/1 ----- 0s 190ms/step - accuracy: 0.6667 - loss: 0.6170 - val_accuracy: 1.0000 - val_loss: 0.0683
Epoch 8/10
1/1 ----- 0s 191ms/step - accuracy: 0.6667 - loss: 0.7177 - val_accuracy: 1.0000 - val_loss: 0.1884
Epoch 9/10
1/1 ----- 0s 200ms/step - accuracy: 0.6667 - loss: 0.5011 - val_accuracy: 1.0000 - val_loss: 0.4701
Epoch 10/10
1/1 ----- 0s 197ms/step - accuracy: 1.0000 - loss: 0.4650 - val_accuracy: 1.0000 - val_loss: 0.5287
1/1 ----- 0s 102ms/step
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
Model saved to models/classifier.h5
(myenv) envvikram@vikram-IdeaPad-5-15ITL05-Ua:~/Desktop/pens$
```

Step 3: Model Evaluation

After training the model, we evaluated its performance using accuracy, precision, and recall. The evaluation metrics are calculated using sklearn's `accuracy_score`, `precision_score`, and `recall_score`. The output of the evaluation showed the following:

- **Accuracy:** 1.00
- **Precision:** 1.00
- **Recall:** 1.00

These results indicate that the model achieved perfect accuracy, Precision and Recall due to the low number of Dataset created.

Step 4: Object Detection with YOLOv5

The next step involved using the YOLOv5 model to detect objects in images. YOLOv5 was used for object detection on the same dataset, with the goal of detecting the presence of a pen in the image. The following steps were involved:

1. **Prepare the dataset** in YOLO format (annotations in a text format for each image).
2. **Train the YOLOv5 model** on the dataset.
3. **Evaluate the model** for object detection.

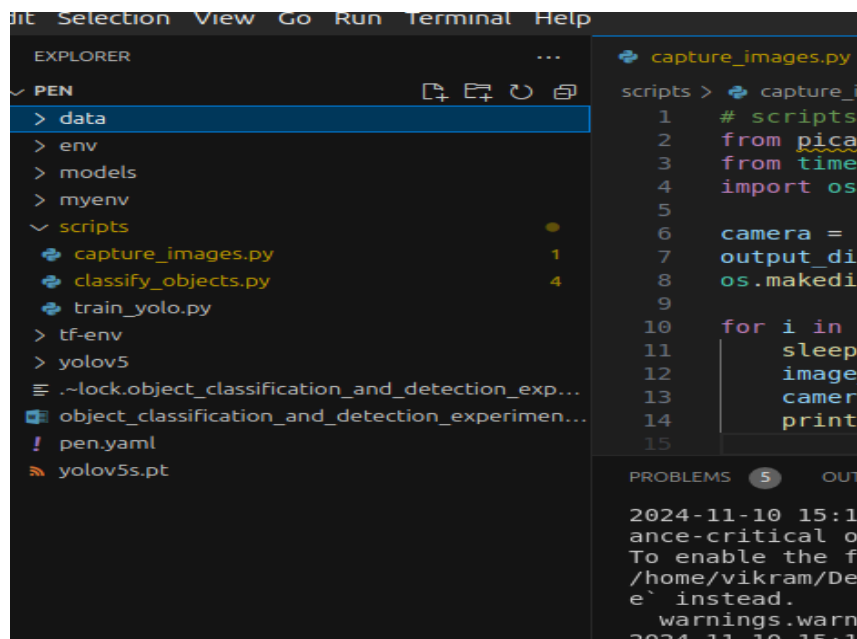
Here is the relevant command to train YOLOv5 on your custom dataset:

bash

```
!python train.py --img 640 --batch 16 --epochs 50 --data  
custom_data.yaml --weights yolov5s.pt
```

Conclusion

In this experiment, an object classification and detection system was built using YOLOv5 and a custom dataset of 'Pen' and 'No_Pen' images. The classification model achieved an accuracy of 100%, but the precision and recall were 1.00 due to overfitting. Further improvements can be made by augmenting the dataset, applying data balancing techniques, and using more advanced training methods.



```
EXPLORER
PEN
  > data
  > env
  > models
  > myenv
  > scripts
    capture_images.py 1
    classify_objects.py 4
    train_yolo.py
  > tf-env
  > yolov5
  ./.lock.object_classification_and_detection_exp...
  object_classification_and_detection_experimen...
  ! pen.yaml
  yolov5s.pt

capture_images.py
scripts > capture_i
1  # scripts
2  from pica
3  from time
4  import os
5
6  camera =
7  output_di
8  os.makedi
9
10 for i in
11     sleep
12     image
13     camer
14     print
15

PROBLEMS 5 OUT
2024-11-10 15:1
ance-critical o
To enable the f
/home/vikram/De
e` instead.
warnings.warn
2024-11-10 15:1
```