Experiment 8

Face and Object Detection

Aim:

To perform Face and Object Detection using Haar Cascade and Object Detection using YOLO V5 Deep Learning Library.

Software/ Packages Used:

1. Google Colaboratory
2. Libraries used:
    - Opencv – python
    - Numpy
    - Matplotlib
    - tensorflow

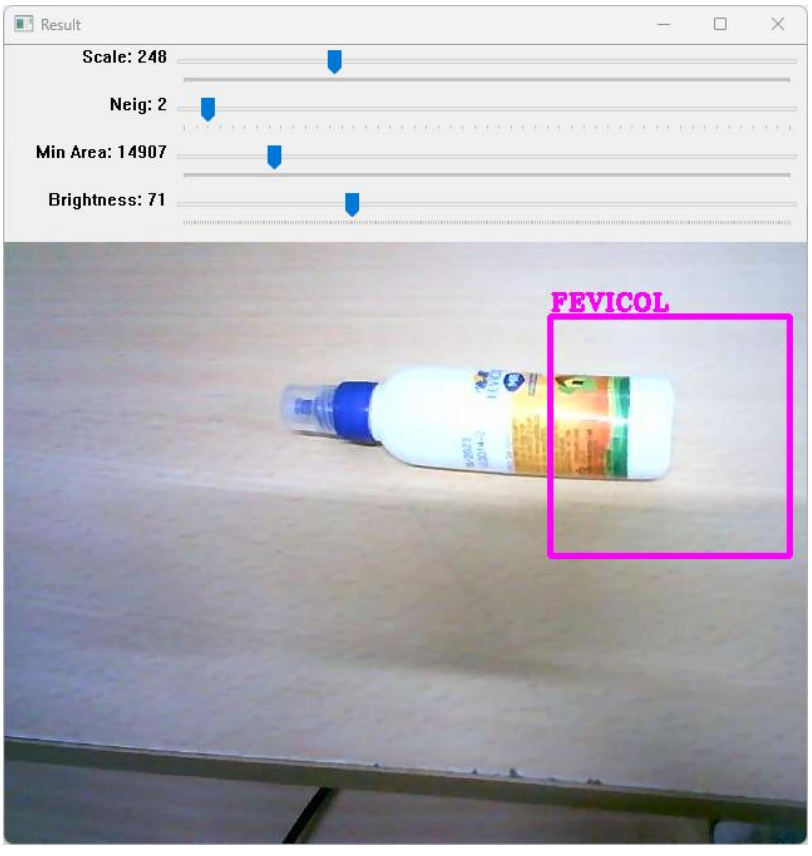
Programs:

1] Haar Cascade Based Face Detection:

POSITIVE IMAGES:

NEGATIVE IMAGES:

OUTPUT:



2] YOLO v5 Based Object Detection: (both for pretrained and custom data- (i/p -Video, Image, Live Video))

CODE:

Setup:

```
!git clone https://github.com/ultralytics/yolov5  # clone
%cd yolov5
%pip install -qr requirements.txt comet_ml  # install

import torch
import utils
display = utils.notebook_init()  # checks

from google.colab import drive
drive.mount('/content/drive')
```

Detect:

```
!python detect.py --source "/content/drive/MyDrive/datasets/chicken video.mp4"
!python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source data/images
# display.Image(filename='runs/detect/exp/zidane.jpg', width=600)
```

Validate:

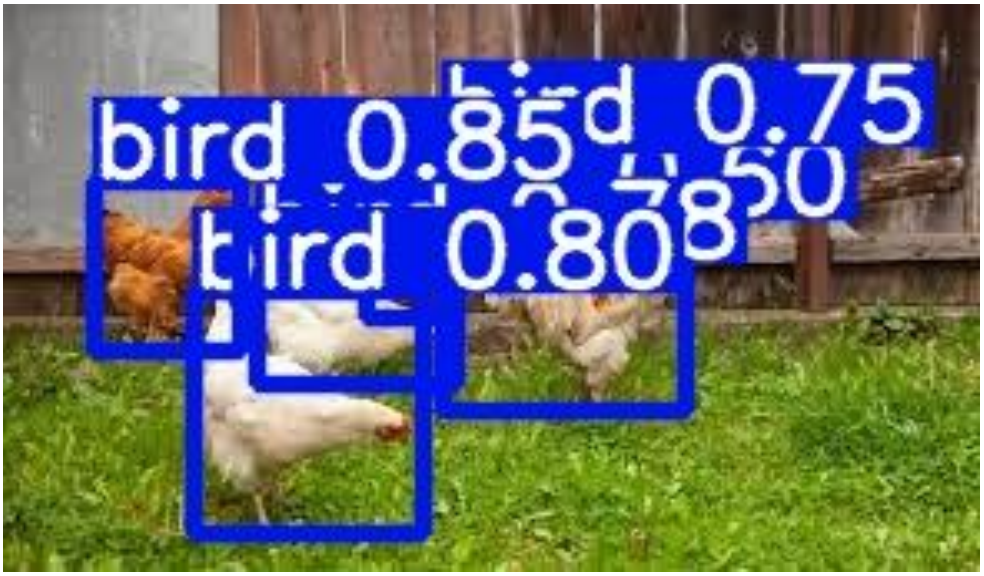
```
# Download COCO val
torch.hub.download_url_to_file('https://ultralytics.com/assets/coco2017val.zip', 'tmp.zip')
# download (780M - 5000 images)
!unzip -q tmp.zip -d ../datasets && rm tmp.zip # unzip
# Validate YOLOv5s on COCO val
!python val.py --weights yolov5s.pt --data coco.yaml --img 640 --half
```

Train:

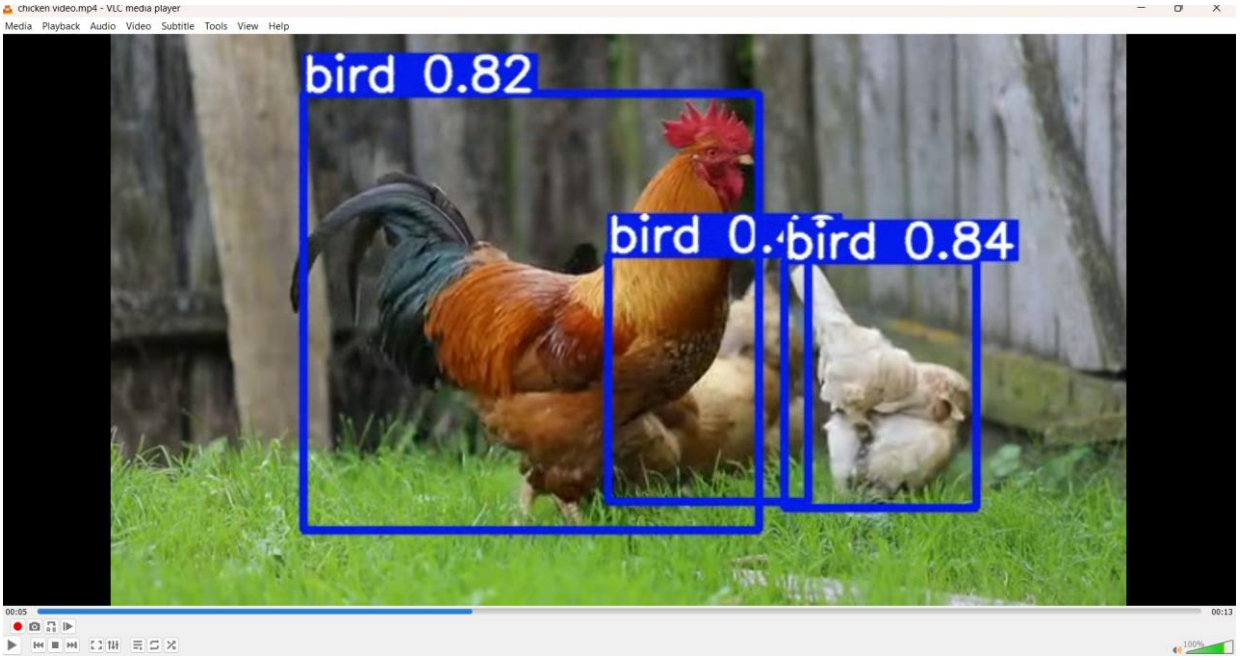
```
#@title Select YOLOv5
logger = 'Comet' #@param ['Comet', 'ClearML', 'TensorBoard']

if logger == 'Comet':
%pip install -q comet_ml
import comet_ml; comet_ml.init()
elif logger == 'ClearML':
%pip install -q clearml
import clearml; clearml.browser_login()
elif logger == 'TensorBoard':
%load_ext tensorboard
%tensorboard --logdir runs/train
# Train YOLOv5s on COCO128 for 3 epochs
!python train.py --img 640 --batch 16 --epochs 3 --data coco128.yaml --weights yolov5s.pt
        --cache
```

OUTPUT:

IMAGE:

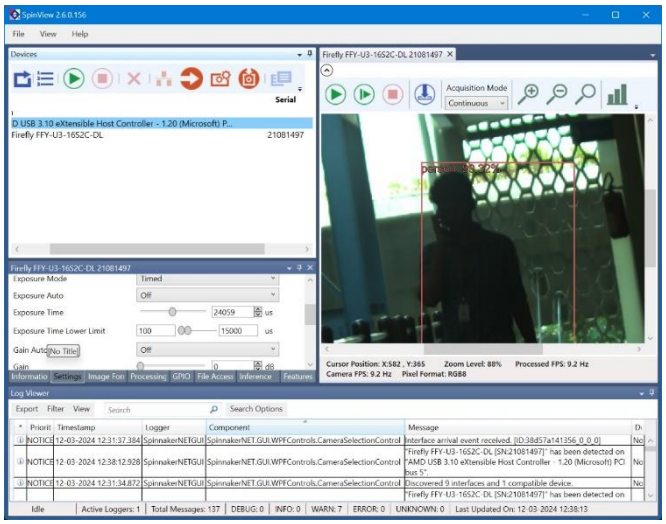


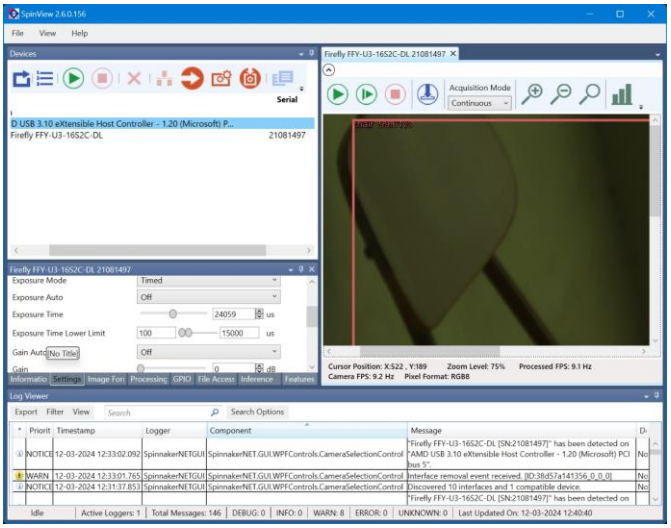
VIDEO:

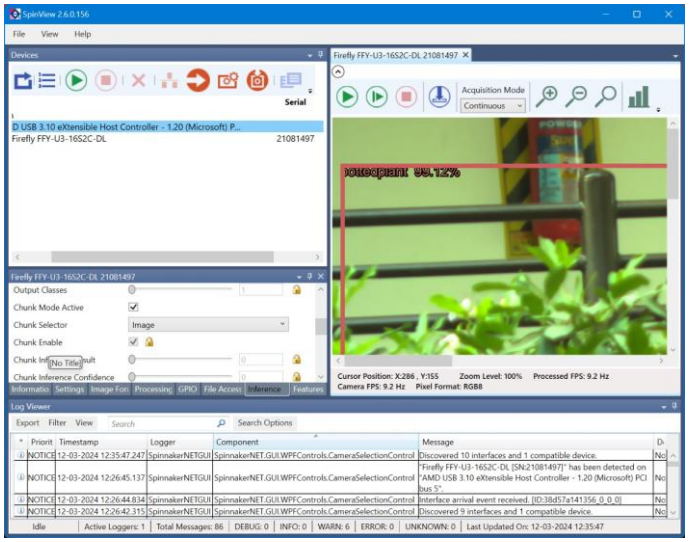


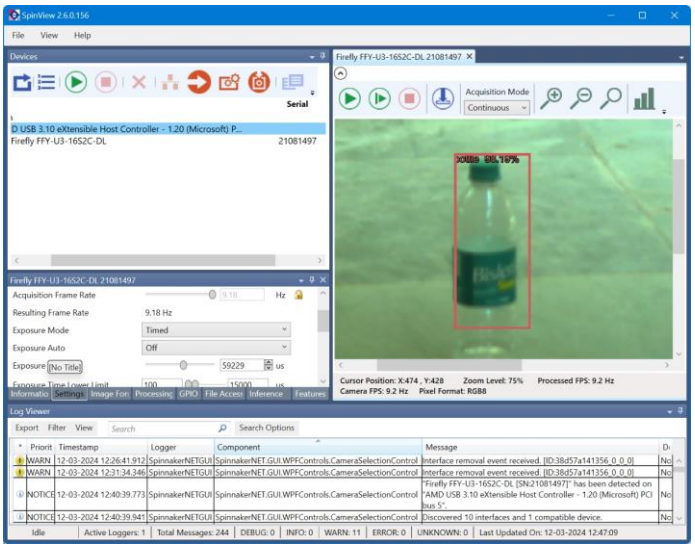
3] Object Detection using Deep Learning camera:

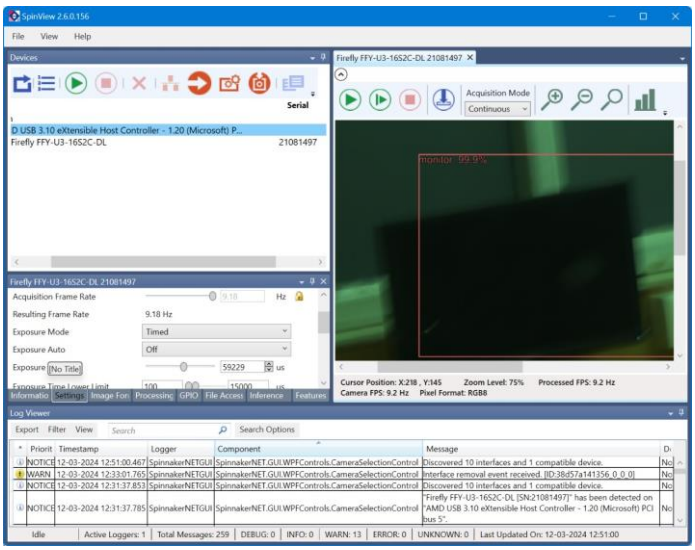
OUTPUT:

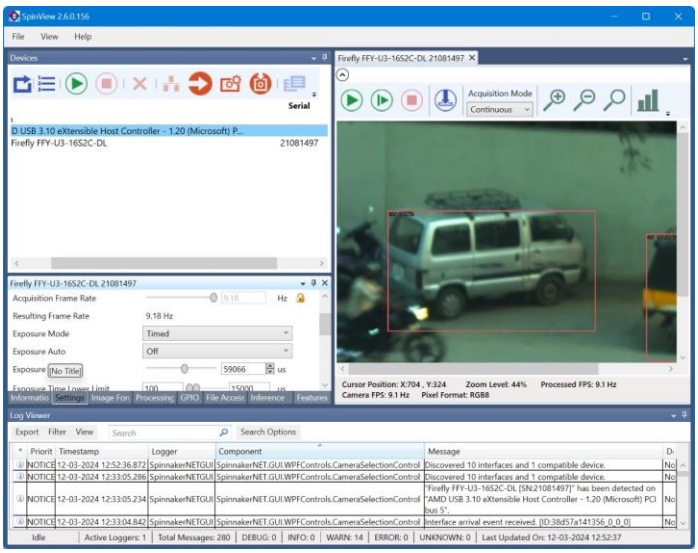


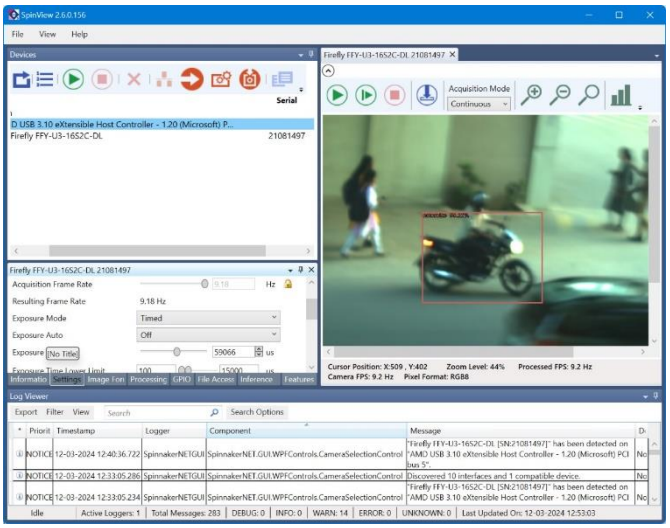


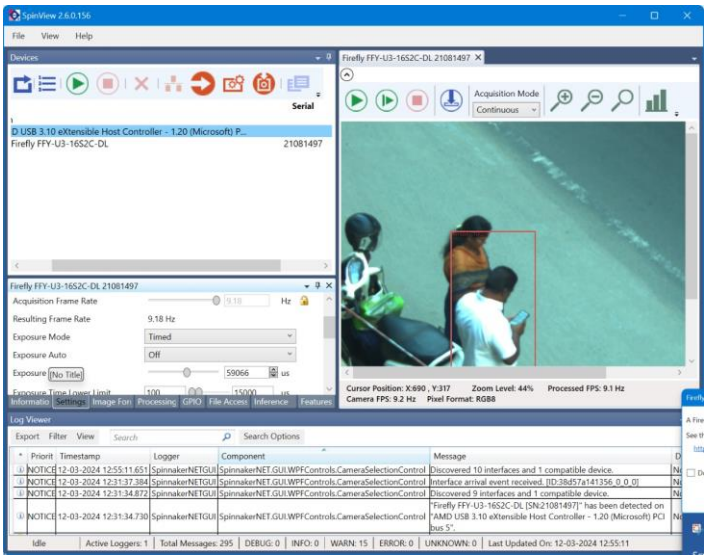












Department of RAE			
Criteria	**Excellent (75% - 100%)**	**Good (50 – 75%)**	**Poor (<50%)**
Preparation (30)			
Performance (30)			
Evaluation (20)			
Report (20)			
Sign:		**Total (100)**	

Result:

Thus, Face Detection using Haar Cascade and Object Detection using Yolo V5 were performed.

Post Lab Questions

1. **What are the key advantages of using Haar Cascades for face detection compared to other methods?**
   - Haar cascades use simple rectangular features, enabling rapid image scanning for real-time face detection, even on low-powered devices.
   - They achieve high face detection rates while maintaining relatively low false positive rates, suitable for security systems and video surveillance.
   - Haar cascades exhibit robustness to lighting conditions and facial expressions, ensuring reliable detection across diverse environments.
   - They are relatively easy to train, allowing for adaptation to specific use cases and environments.
   - Haar cascades are versatile and applicable in a range of scenarios, enhancing their utility for various applications.
2. **What is the difference between ANN & CNN.**
   - ANNs consist of interconnected layers of nodes, while CNNs utilize convolutional layers for feature extraction.
   - ANNs are used for tasks like regression, classification, and pattern recognition, while CNNs are specialized for processing grid-like data such as images.
   - CNNs employ convolutional layers to extract features from input images efficiently.
   - CNNs utilize hierarchical structures to learn increasingly complex features, particularly effective for tasks like image classification and object detection.
   - CNNs excel in tasks involving visual data due to their tailored architecture for image processing.
3. **For the following image perform the convolution operation. Also perform Max pooling, Min pooling and Average pooling on the input image.**

6	5	4	3	2	1
7	6	5	4	3	2
8	7	6	5	4	3
9	8	7	6	5	4
10	9	8	7	6	5
10	10	9	8	7	6

3	2	4
2	0	2
4	2	3

Input Matrix:

[[6 5 4 3 2 1]
[7 6 5 4 3 2]
[8 7 6 5 4 3]
[9 8 7 6 5 4]
[10 9 8 7 6 5]
[10 10 9 8 7 6]]

Filter Matrix:

[[3 2 4]
[2 0 2]
[4 2 3]]

Output Matrix after Convolution:

```
[[132. 110.  88.  66.]
 [154. 132. 110.  88.]
 [176. 154. 132. 110.]
 [194. 176. 154. 132.]]
```

Input Matrix:

```
[[ 6  5  4  3  2  1]
 [ 7  6  5  4  3  2]
 [ 8  7  6  5  4  3]
 [ 9  8  7  6  5  4]
 [10  9  8  7  6  5]
 [10 10  9  8  7  6]]
```

Max Pooled Matrix:

```
[[ 7.  5.  3.]
 [ 9.  7.  5.]
 [10.  9.  7.]]
```

Min Pooled Matrix:

```
[[5. 3. 1.]
 [7. 5. 3.]
 [9. 7. 5.]]
```

Average Pooled Matrix:

```
[[6.   4.   2.  ]
 [8.   6.   4.  ]
 [9.75 8.   6.  ]]
```