Mid Term Project Review (Semester V)

Vivekanand Education Society's Institute of Technology
Electronics and Telecommunication Engineering

Implementation of YOLO Algorithm without function

Name of guide: Mrugendra Vasmatkar

GROUP NO:

Shraavani Tople 64
Jatin Vaity 23
Yash Bamne 03
Tejas Chorge 12

PROBLEM STATEMENT

To design and implement an object detection system using the YOLO algorithm in Python, integrating it with the IPMV (Image Processing and Computer Vision) capabilities. The system should take images or video streams as input and provide real-time object detection, displaying bounding boxes around detected objects along with class labels. Utilize the OpenCV library for image processing tasks and ensure the implementation is efficient and well-documented. Optional enhancements may include non-maximum suppression and exploring different YOLO versions for improved performance. Deliver the source code, documentation, and a demonstration of the system's capabilities.

PROPOSED SOLUTION

Setup:

- Install Python libraries: OpenCV, NumPy.
- Download YOLO weights and config files.

Implementation:

- Load YOLO model using OpenCV.
- Ensure IPMV integration.

PROPOSED SOLUTION

Functionality:

- Apply YOLO for object detection.
- Optimize for real-time performance.
- Visualize results with bounding boxes and labels.

Documentation and Delivery:

- Provide concise code documentation.
- Share code on GitHub with a short README and concise documentation.

LITERATURE SURVEY

Title of paper	Name of Author	Name of Journal/Year of publication	Methodology	Results	Drawbacks
[1]Application of YOLO algorithm on student	Alen Grebo; Toni Konsa; Goran Gašparović; Branko Klarin	IEEE Explore,26 September 2020	Capture diverse UAV images, manually annotate, and train YOLO for object detection. Optimize parameters, integrate into UAV, and assess real-time performance in various flight scenarios.	Evaluate YOLO's detection accuracy, real-time processing during UAV flights, and adaptability to different conditions.	Challenges include limited training data diversity, resource constraints impacting real-time performance, and considerations for environmental variability and object occlusion.

Title of paper	Name of Author	Name of Journal/Year of publication	Methodology	Results	Drawbacks
[2] Automatic Vehicle Headlight Management System	Lakshmi K, Nevetha R, Ilakkiya S N, Ganesan R	International Journal of Innovative Technology and Exploring Engineering ,2019	STM 32 Microcontroller controls the high beam falling on it. When a high beam falls on the surface of LDR, the information passes to the Arduino.compares the intensity of incoming light with the desired intensity value	High beam is automatically switched to low beam when a high beam of light from the another vehicle falls on the LDR. Glaring of light from the opposite vehicle during the night	The system heavily relies on sensors to detect oncoming vehicles and adjust the headlight intensity accordingly. If these sensors malfunction or provide inaccurate data, it can lead to incorrect headlight adjustments.

ALGORITHM OF PROPOSED SOLUTION

1. Download Files:- Use Google Drive file IDs to download required files: `coco.names`, `yolov3-tiny.weights`, `yolov3-tiny.cfg`, and an example image (`bus.jpg`).

2. Import Libraries:- Import necessary libraries and modules: `cv2`, `numpy`, `matplotlib.pyplot`, and `sys`.

3. Read Object Names:- Open the `coco.names` file and read the object names into the `classes` list.

4. Search Object in Database:- Prompt the user to input an object name.- Check if the entered object exists in the database (`classes` list).- If found, proceed; otherwise, exit the program.

5. Load and Preprocess Image:- Load the example image (`bus.jpg`) using OpenCV.- Perform preprocessing:- Scale pixel values to the range [0, 1].- Resize the image to (320, 320).- Convert color channels from BGR to RGB.- Convert the image to a blob (4D numpy array).

6. Visualize Preprocessing:- Display the original and preprocessed images for comparison using `matplotlib.pyplot`.

ALGORITHM OF PROPOSED SOLUTION

7. Load YOLO Model:- Use OpenCV to load YOLOv3-tiny weights and configuration files.- Set the input image for the YOLO model.- Get the names of the output layers in the YOLO network.- Perform a forward pass through the YOLO network.

8. Extract Bounding Boxes:- Iterate through each output layer and detection.- Extract confidence scores, class IDs, and bounding box coordinates.- If confidence is above a certain threshold, add the detection information to respective lists.

9. Non-Maximum Suppression (NMS):- Use OpenCV's NMSBoxes function to perform non-maximum suppression.- Filter out overlapping bounding boxes.

10. Draw Bounding Boxes:- Iterate through the remaining bounding boxes after NMS.- Draw bounding boxes and labels on the original image using `cv2.rectangle` and `cv2.putText`.

11. Visualize Results:- Display the original image with detected objects alongside the original image using `matplotlib.pyplot`

BLOCK DIAGRAM OF PROPOSED SOLUTION

TIME CHART

JAN 2024	FEB 2024	MARCH 2024	APRIL 2024
W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4
1. LITERATURE SURVEY. 2. PLANNING.	1. CODE	1.IMPLEMENTATION. 2.PROJECT REPORT.	1.FINAL REVIEW.

IMPLEMENTATION

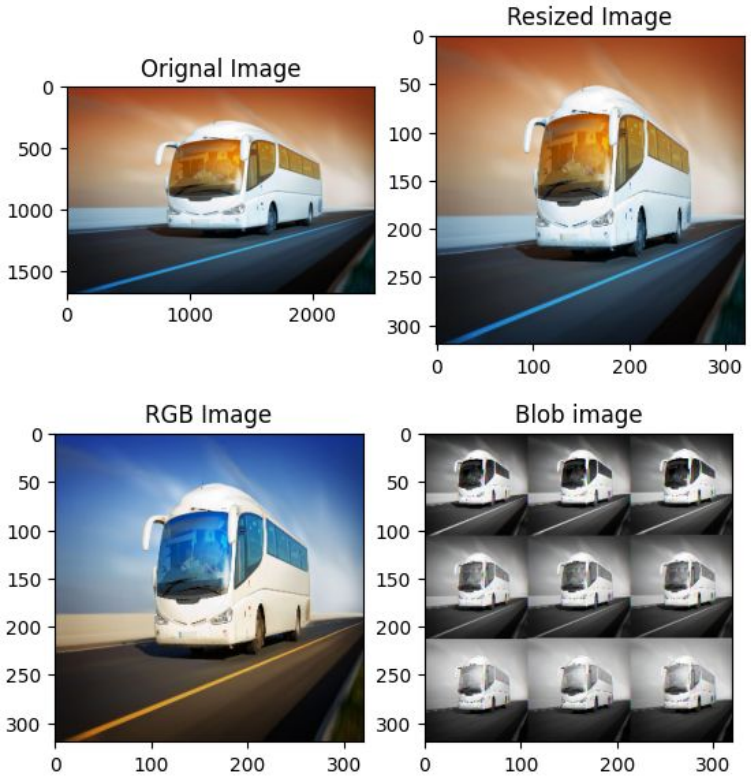
```
! gdown --id 1QFPS1EHdwakTaPSY1zbzf_17GX3GGmr1   #coco names file
! gdown --id 1w3VWBf4tP8WzDhzD_gpiNWiD-sW8V8mt   #Weight files
! gdown --id 1dAjX0vDKOThxpPOo-065y89fxsqlUve1  #yolo cfg files
! gdown --id 1YbIjc37WvUnIZZYVaaWKec8a5qy9uc6d  #Cat
```

```
# Objects in databse
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]


object_name = input("Search object in database: ")
if object_name.lower() in classes:
    print("Object found in the database")
else:
    print("Object not found in the database")
    sys.exit()
```

```
# Importing the image
path="./bus.jpg"
img=cv2.imread(path)
original=cv2.imread(path)
print("Input image is ")
plt.imshow(img)
height, width, channels = img.shape
```

```
# preprocessing:
# Scale pixel values to the range [0, 1]
img1 = img.astype(np.float32) / 255.0
# Resize the image to (320, 320)
img1 = cv2.resize(img1, (320, 320))
# Swap the channels from BGR to RGB
img2 = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
# Convert the image to a blob
# A blob in this case would be a 4D numpy array
image = np.expand_dims(np.transpose(img2, (2, 0, 1)), axis=0)
i = image[0].reshape(320,320,3)

yolo_img = cv2.cvtColor(i, cv2.COLOR_BGR2RGB)
plt.subplot(1,2,1)
plt.title("Orignal Image")
plt.imshow(img)
plt.subplot(1,2,2)
plt.title("Resized Image")
plt.imshow(img1)
plt.show()
plt.subplot(1,2,1)
plt.title("RGB Image")
plt.imshow(img2)
plt.subplot(1,2,2)
plt.title("Blob image")
plt.imshow(i)
plt.show()
```

RESULTS




```
1
[[722, 481, 1444, 963]]
```

REFERENCES

[1]A. Grebo, T. Konsa, G. Gašparović and B. Klarin, "Application of YOLO algorithm on student UAV," 2020 5th International Conference on Smart and Sustainable Technologies.

[2]G. R. Poornima, V. Harish, S. Karthik, B. S. Nagesh kumar and S. Varun Kumar, "Vehicle Headlight Automation with Smart Energy Management System," 2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), Bangalore, India, 2019, pp. 396-399, doi: 10.1109/RTEICT46194.2019.9016890. keywords: {Roads;Automobiles;Switches;Automation;Energy management;Accidents;headlight automation;PIR sensor;LDR;energy management system},

THANK YOU