Proceedings of the Third International Conference on Smart Systems and Inventive Technology (ICSSIT 2020)
IEEE Xplore Part Number: CFP20P17-ART; ISBN: 978-1-7281-5821-1

YOLOv3 and YOLOv4: Multiple Object Detection for Surveillance Applications

Chethan Kumar B, Punitha R, Mohana
Electronics & Telecommunication Engineering
RV College of Engineering®, Bangalore, Karnataka, India

***Abstract*—Object detection algorithm such as You Only Look Once (YOLOv3 and YOLOv4) is implemented for traffic and surveillance applications. A neural network consists of input with minimum one hidden and output layer. Multiple object dataset (KITTI image and video), which consists of classes of images such as Car, truck, person, and two-wheeler captured during RGB and grayscale images. The dataset is composed (image and video) of varying illumination. YOLO model variants such as YOLOv3 is implemented for image and YOLOv4 for video dataset. Obtained results show that the algorithm effectively detects the objects approximately with an accuracy of 98% for image dataset and 99% for video dataset.**

Keywords- Darknet 53, YOLOv3, feature extraction, intelligent traffic system.

I. INTRODUCTION

Vehicle object detection finds many applications in computer vision such as traffic monitoring, traffic density estimation, self-driving cars and many more [1] [2]. Deep convolutional neural networks (DNNs) are used to detect objects in regions. Proposed implementation, YOLOv3 and YOLOv4 algorithm is used to predict class labels and detect objects.

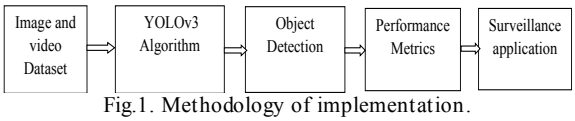

Fig.1. Methodology of implementation.

Methodology of implementation as shown in figure 1.

Data Collection: Chosen dataset is vehicle detection dataset [8] for urban roads. It is very important because the quality and quantity of data will directly determine how good the trained model will behave under various environmental conditions.

Data Arrangement: Data is prepared to make it suitable for the deep learning model purposes.

Network Architectures: choosing of an appropriate model for training among many available models with various characteristics.

Network Parameters: Weights and bias of deep learning model are initialized randomly to avoid any symmetry, which could potentially affect the training process.

Hyper parameters: Various hyper parameters such as learning rate, regularization parameter, dropout probability are initialized before training.

Training Deep Learning Model: Deep learning model is trained on image and video dataset using GPU.

Analysis and Evaluation: The network model is evaluated based on obtained results. Further hyper parameters are tuned to improve the overall results. Features are extracted using darknet-53, YOLOv3 algorithm is used for vehicle detection.

Applications

- Feature extraction in various scenarios like movies, animations, computer games etc [3] [4] [5].
- Vehicle detection
- Traffic density estimation and smart city related applications [12].
- Object detection in smart surveillance systems
- Autonomous cars [11].
- Object detection for most of the computer vision and AI vision system [6].
- AI Enabled Traffic monitoring system [9].

II. THEORY AND FUNDAMENTALS

A. YOLOv3 Architecture

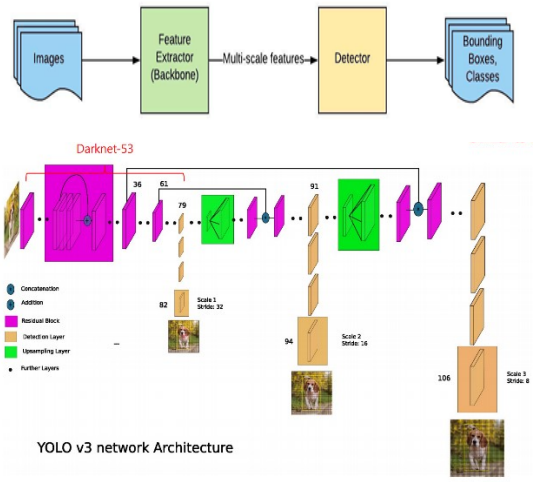

Fig.2. Architecture of YOLOv3 [10]

Figure 2 shows the architecture and implementation steps of YOLOv3.The feature extractor YOLOv3 uses darknet-53. YOLOv3 uses CNN to predict class and location of the object [7] [8]. The past darknet rendition from YOLOv1, where there're 19 layers. In ResNet darknet-53 uses 19 to 53 layers for feature extraction. For YOLOv4 is improvement of YOLOv3 architecture, where it uses CSP darknet-53 classifier. In addition, it uses spatial pyramid pooling and path aggregation network (PAN) that connects YOLOv3 head. It provides an optimal speed and accuracy for multiple objects detection in single frame.

Dependencies: Tensor Flow (deep learning), NumPy (numerical computation), Pillow (image processing) libraries, Seabourn's color palette for bounding boxes colors, IPython function display () to display images in the notebook.

Model hyperparameters: Define and configure for Yolov3 such as Batch normalization, Leaky ReLU, Anchors, Batch norm and fixed padding.
Model definition: Define the feature extraction of Darknet-53, detection layer of n number of classes and to avoid overlap of bounding box by using Non-Max Suppression shown in figure 3 [14] [15].

Fig.3. Before and after non –max suppression

YOLOv3 Output Vector

$$Y = [P_c\ b_x\ \ b_y\ b_w\ \ C_1\ C_2\ \ C_3 .. C_n]$$

Where:

P_c = Probability of an object in the image
B_x = bounding box center x-coordinate
B_y = bounding box center y-coordinate
B_h = bounding box height
B_w = bounding box width
C_1 = Highest confidence class prediction for object in bounding box
C_2 = Second highest confidence class prediction for object in bounding box
C_3 = Third highest confidence class prediction for object in bounding box
C_n =…n^{th} highest confidence class prediction for object in bounding box

Utility functions: It helps to weight images as NumPy arrays, load class names from the official file and draw the predicted boxes.
Initialization of weights (Tensor Flow): Initialization of YOLOv3 weights and TF (tensor flow) assign for algorithm operation.
Running the model: Testing the model with IoU (Interception over Union ratio used in non-max suppression) threshold and confidence threshold both set to 0.5.

III. DESIGN AND IMPLEMENTATION

This section describes design and implementation details, flow diagram of vehicle detection algorithm and datasets used for implementation.

A. Flow chart of Vehicle detection

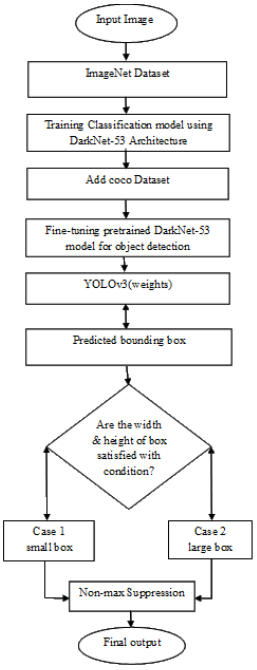

Fig.4. Flowchart of multiple object detection algorithm

Figure 4 shows the flow chart of implementation; to start with, libraries are imported first after dataset is mounted. Kaggle/google-Collab and GPU (Graphical Processing Unit) processing is used for simulation. After mounting dataset, training, classification model is done via the darknet-53 architecture of YOLOv3 algorithm. For vehicle detection, several classes of object are predicted through YOLOv3 weights. Batch normalization and anchor model are used to predict bounding box, to predict the location of object region in given dataset.

B. Dataset specifications

Case A- Day image detection (RGB)

- Dataset Name – KITTI image dataset
- Class - [Car, Bus, Truck, Bike, Person, Animals]

Fig.5. Sample images of dataset

Figure 5 shows the sample images taken from dataset, which to mount input for multiple object detection, to detect multiple classes in an image.

Case B: Grayscale image dataset

- Dataset Name – KITTI grayscale Dataset
- Class- [Car, Bus, Truck, Bike, Person, Animals, etc.]

Fig.6. Sample images of dataset (gray scale)

Figure 6 shows the sample images of Gray scale dataset, which to mount input for multiple object detection, to detect multiple classes in an image.

Case C: Object detection in video

Name –RV campus Street road video

- Format - mp4
- Size – 11.9 Mb
- Length – 02:22
- Time frame - 25 frames/sec
- Video Quality – 720p
- Types of objects specified set – VOC dataset

Fig.7. Sample image of video for object detection.

Figure 7 shows the sample image of video for multiple object detection, which to mount input for multiple object detection, objects are detected frame by frame in a video for multiple class of objects.

Case 4: Object detection (Gray scale video)

Name – Gray scale Street road video

- Format - mp4
- Size – 55.2 Mb
- Length – 00:09
- frame rate - 23 frames/sec
- Video Quality – 720p
- Types of objects specified set – VOC dataset

Fig.8. Sample image of video for object detection (gray scale).

Figure 8 show the sample image from gray scale video, which to mount input for multiple object detection.

C. Data Object Arrangement

Images are centred and resized into 64x64 dimensions. Thus, a single image has a shape of 3x64x64.These 3x64x64 images are converted into tensors of same size for efficient processing of the data during training and testing steps.

The whole dataset changed into divided into mini batches of length for 8 to take advantage to superior optimization algorithms which expects statistics to be in batches of small size like 4, 8, 16, 32 and so on. Darknet 53 is used as convolution base.

In [2]:

```
_BATCH_NORM_DECAY = 0.9
_BATCH_NORM_EPSILON = 1e-05
_LEAKY_RELU = 0.1
_ANCHORS = [(10, 13), (16, 30), (33, 23),
            (30, 61), (62, 45), (59, 119),
            (116, 90), (156, 198), (373, 326)]
_MODEL_SIZE = (416, 416)
```

```
def darknet53_residual_block(inputs, filters, training, data_format,
                             strides=1):
    """Creates a residual block for Darknet."""
    shortcut = inputs

    inputs = conv2d_fixed_padding(
        inputs, filters=filters, kernel_size=1, strides=strides,
        data_format=data_format)
    inputs = batch_norm(inputs, training=training, data_format=data_format)
    inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

    inputs = conv2d_fixed_padding(
        inputs, filters=2 * filters, kernel_size=3, strides=strides,
        data_format=data_format)
    inputs = batch_norm(inputs, training=training, data_format=data_format)
    inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

    inputs += shortcut

    return inputs


def darknet53(inputs, training, data_format):
    """Creates Darknet53 model for feature extraction."""
    inputs = conv2d_fixed_padding(inputs, filters=32, kernel_size=3,
                                  data_format=data_format)
    inputs = batch_norm(inputs, training=training, data_format=data_format)
    inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)
    inputs = conv2d_fixed_padding(inputs, filters=64, kernel_size=3,
                                  strides=2, data_format=data_format)
    inputs = batch_norm(inputs, training=training, data_format=data_format)
    inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)
```

Network specification

```
mini_batch = 1, batch = 8, time_steps = 1, train = 0
   layer   filters  size/strd(dil)      input                output
   0 conv     32       3 x 3/ 1    608 x 608 x   3 ->  608 x 608 x  32 0.639 BF
   1 conv     64       3 x 3/ 2    608 x 608 x  32 ->  304 x 304 x  64 3.407 BF
   2 conv     64       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  64 0.757 BF
   3 route  1                              ->  304 x 304 x  64
   4 conv     64       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  64 0.757 BF
   5 conv     32       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  32 0.379 BF
   6 conv     64       3 x 3/ 1    304 x 304 x  32 ->  304 x 304 x  64 3.407 BF
   7 Shortcut Layer: 4,  wt = 0, wn = 0, outputs: 304 x 304 x  64 0.006 BF
   8 conv     64       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  64 0.757 BF
   9 route  8 2                            ->  304 x 304 x 128
  10 conv     64       1 x 1/ 1    304 x 304 x 128 ->  304 x 304 x  64 1.514 BF
  11 conv    128       3 x 3/ 2    304 x 304 x  64 ->  152 x 152 x 128 3.407 BF
  12 conv     64       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x  64 0.379 BF
  13 route  11                            ->  152 x 152 x 128
  14 conv     64       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x  64 0.379 BF
  15 conv     64       1 x 1/ 1    152 x 152 x  64 ->  152 x 152 x  64 0.189 BF
  16 conv     64       3 x 3/ 1    152 x 152 x  64 ->  152 x 152 x  64 1.703 BF
  17 Shortcut Layer: 14,  wt = 0, wn = 0, outputs: 152 x 152 x  64 0.001 BF
  18 conv     64       1 x 1/ 1    152 x 152 x  64 ->  152 x 152 x  64 0.189 BF
  19 conv     64       3 x 3/ 1    152 x 152 x  64 ->  152 x 152 x  64 1.703 BF
  20 Shortcut Layer: 17,  wt = 0, wn = 0, outputs: 152 x 152 x  64 0.001 BF
  21 conv     64       1 x 1/ 1    152 x 152 x  64 ->  152 x 152 x  64 0.189 BF
  22 route  21 12                         ->  152 x 152 x 128
  23 conv    128       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x 128 0.757 BF
  24 conv    256       3 x 3/ 2    152 x 152 x 128 ->   76 x  76 x 256 3.407 BF
  25 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379 BF
  26 route  24                            ->   76 x  76 x 256
  27 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379 BF
  28 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189 BF
  29 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703 BF
  30 Shortcut Layer: 27,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
```

Fig. 9. Code snippet of data object localization and Network specifications.

Figure 9. Shows the code snippet of data object localization and network specifications

IV. SIMULATION RESULTS AND ANALYSIS

Multiple object detection algorithm was trained and tested on image and video dataset. The IOU 0.5 is considered for implementation.

A.Multiple object detection on KITTI image dataset

Figure 10. Shows the simulation results of multiple object detection, algorithm of YOLOv3 model is successfully detected the multiple classes such as car, cycle, truck, person, van, for KITTI dataset and detected class was denoted in bounding box.

Fig 10. Multiple object detection using YOLOv3

B. Multiple object detection on KITTI grey scale image dataset

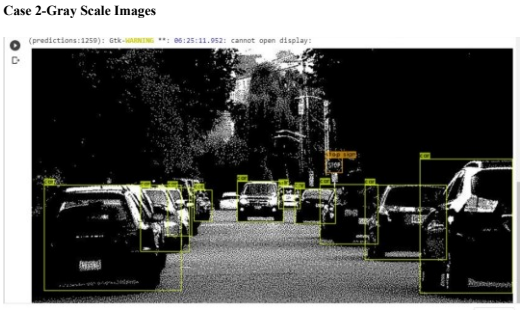

Fig 11. Multiple object detection on Gray scale images

Figure 11 shows the simulation results of multiple object detection, algorithm of YOLOv3 model successfully detected multiple classes like car, stop sign, in gray scale dataset and detected class was denoted in bounding box.

C. Multiple object detection in Video dataset using YOLOv4

Fig 12. Results of multiple object detection in video

Figure 12 shows the simulation results of video dataset, algorithm of YOLOv4 model is successfully detected multiple object classes in a video by detecting the every object in each frame by frame until the whole frame of video closes, the class was denoted in bounding box, the detected classes are car, truck, person, dog, handbag.

D. Multiple object detection in gray scale video

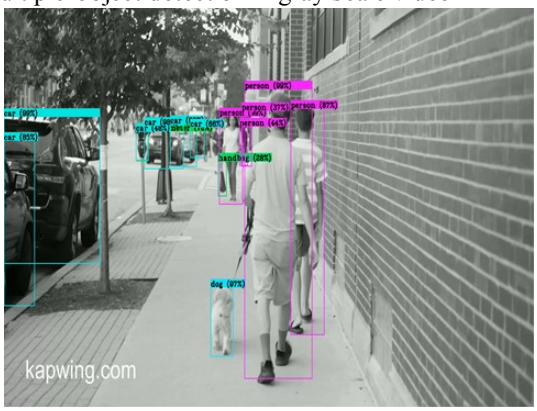

Fig 13. Results of multiple object detection on gray scale video

Figure 13 shows the simulation results of multiple object detection in gray scale video, algorithm of YOLOv4 model is successfully detected multiple object classes in a grey scale video by detecting the every object in each frame by frame until the whole frame of video closes, the class was denoted in bounding box, the detected classes are car, truck, person, dog, handbag.

E. Performance Analysis

Analysis in KITTI Image dataset (YOLOv3)

Done! Loaded 162 layers from weights-file../wh.jpg: Predicted in 63.119000 milli-seconds. car: 84% car: 69% car: 32% car: 31% car: 94% car: 86% car: 84% car: 30% stop sign: 43% car: 68% car: 72%

```
../wh.jpg: Predicted in 63.119000 milli-seconds.
car: 84%
car: 69%
car: 32%
car: 31%
car: 94%
car: 86%
car: 84%
car: 30%
stop sign: 43%
car: 68%
car: 72%
```

Analysis in KITTI Image grey scale dataset (YOLOv3)

[yolo] params: iou loss: ciou (4), iou_norm: 0.07, cls_norm: 1.00, scale_x_y: 1.05 nms_kind: greedynms (1), beta = 0.600000 Total BFLOPS 128.459 avg_outputs = 1068395 Allocate additional workspace_size = 52.43 MB Loading weights from yolov4.weights... seen 64, trained: 32032 K-images (500 Kilo-batches_64) Done! Loaded 162 layers from weights-file ../pc2.jpg: Predicted in 63.220000 milli-seconds. truck: 43% bus: 91% person: 94% person: 28% person: 97% person: 93% person: 56% person: 41% person: 33% person: 87% person: 56% person: 70% suitcase: 27% person: 40% person: 46% person: 49% person: 48%.

```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, cls_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 128.459
avg_outputs = 1068395
 Allocate additional workspace_size = 52.43 MB
Loading weights from yolov4.weights...
 seen 64, trained: 32032 K-images (500 Kilo-batches_64)
Done! Loaded 162 layers from weights-file
../pc2.jpg: Predicted in 63.220000 milli-seconds.
truck: 43%
bus: 91%
person: 94%
person: 28%
person: 97%
person: 93%
person: 56%
person: 41%
person: 33%
person: 87%
person: 56%
person: 70%
suitcase: 27%
person: 40%
person: 46%
person: 49%
person: 48%
```

Analysis in video dataset (YOLOv4)

[yolo] params: iou loss: ciou (4), iou_norm: 0.07, cls_norm: 1.00, scale_x_y: 1.05 nms_kind: greedynms (1), beta = 0.600000 Total BFLOPS 128.459 avg_outputs = 1068395 Allocate additional workspace_size = 52.43 MB Loading weights from yolov4.weights... seen 64, trained: 32032 K-images (500 Kilo-batches_64) Done! Loaded 162 layers from weights-file video file: whv.mp4 Video stream: 3840 x 2160 Objects:

```
FPS:0.0   AVG_FPS:0.0
Objects:

handbag: 28%
dog: 97%
parking meter: 78%
car: 48%
car: 99%
car: 98%
car: 88%
car: 85%
car: 66%
person: 99%
person: 99%
person: 87%
person: 44%
person: 37%

FPS:1.3   AVG_FPS:0.0

 cvWriteFrame
Objects:

dog: 98%
parking meter: 89%
car: 98%
car: 98%
car: 97%
car: 80%
car: 61%
person: 99%
person: 98%
person: 87%
person: 76%
person: 29%

FPS:1.8   AVG_FPS:0.0
```

V. CONCLUSIONS AND FUTURE SCOPE

Detections of multiple objects in video surveillance is a challenging process, where it depends on objects density in surveillance area or road and timings. This paper implements multiple object detection algorithm, which is suitable for traffic and many surveillance applications. Multiple object detection is a key capacity for most computer and AI (Artificial Intelligence) vision system. Obtained results are analized and tabulated.The dataset is composed of (image and video) of varying illumination. Obtained results shows that the algorithm effectively detects multiple objects approximately with an accuracy of 98% for image dataset and 99% for video dataset (approximately average of all frames). In addition, proposed YOLO model variants can be implemented on DSP and FPGA [15] [16].

REFERENCES

[1] Joseph Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection" *arXiv:1506.02640v5 [cs.CV] 9 May 2016.*

[2] Joseph Redmon et al., "YOLOv3: An Incremental Improvement" arXiv:1804.02767v1, 2018.

[3] Alexey Bochkovskiy et al., "YOLOv4: Optimal Speed and Accuracy of Object Detection" *arXiv:2004.10934v1 [cs.CV] 23 Apr 2020.*

[4] Yi-Qi Huang et al., "Optimized YOLOv3 Algorithm and Its Application in Traffic Flow Detections" *Appl. Sci. 2020, 10, 3079; doi:10.3390/app10093079www.mdpi.com.*

[5] Won Jae Lee et al., "A Vehicle Detection Using Selective Multi-Stage Features in Convolutional Neural Networks", *IEEE 2017.*

[6] Ishan Ratn Pandey et al., "Face Recognition Using Machine Learning", *IRJET April 2019.*

[7] Yu Han Liu et al., "Feature Extraction and Image Recognition with Convolutional Neural Networks" *University of Electronic Science and Technology of China,IOP Conf. Series, 2018.*

[8] Salman Khan et al., "A Guide to Convolutional Neural Networks for Computer Vision" *The University of Western Australia, Crawley, WA, 2018.*

[9] Chethan Kumar B et.al., "Performance Analysis of Object Detection Algorithm for Intelligent Traffic Surveillance System," 2nd *International Conference on Inventive Research in Computing Applications (ICIRCA), 2020*

[10] *R. J. Franklin* et al., "Traffic Signal Violation Detection using Artificial Intelligence and Deep Learning," *5th International Conference on Communication and Electronics Systems (ICCES), 2020.*

[11] H. Jain et al., "Weapon Detection using Artificial Intelligence and Deep Learning for Security Applications," *International Conference on Electronics and Sustainable Communication Systems (ICESC), 2020.*

[12] Abhiraj Biswas et. al., "Classification of Objects in Video Records using Neural Network Framework," *International conference on Smart Systems and Inventive Technology (ICSSIT),2018.*

[13] N. Jain et.al., "Performance Analysis of Object Detection and Tracking Algorithms for Traffic Surveillance Applications using Neural Networks," *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)*, 2019.

[14] M. R. Nehashree et al., "Simulation and Performance Analysis of Feature Extraction and Matching Algorithms for Image Processing Applications," *International Conference on Intelligent Sustainable Systems (ICISS), Palladam, 2019.*

[15] S. K. Mankani et al., "Real-time implementation of object detection and tracking on DSP for video surveillance applications," *International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, 2016.

[16] D. Akash et al., "Interfacing of flash memory and DDR3 RAM memory with Kintex 7 FPGA board," *International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2017.*