An Improved and Efficient YOLOv4 Method for Object Detection in Video Streaming

Chapter · January 2022

DOI: 10.1007/978-981-19-2211-4_27

3 authors, including:

Prathap Rudra Boppuru
Christ University, Bangalore
23 PUBLICATIONS **67** CITATIONS

SEE PROFILE

An Improved and Efficient YOLOv4 Method for Object Detection in Video Streaming

Javid Hussain, Boppuru Rudra Prathap, and Arpit Sharma

Abstract As object detection has gained popularity in recent years, there are many object detection algorithms available in today's world. Yet the algorithm with better accuracy and better speed is considered vital for critical applications. Therefore, in this article, the use of the YOLOV4 object detection algorithm is combined with improved and efficient inference methods. The YOLOV4 state-of-the-art algorithm is 12% faster compared to its previous version, YOLOV3, and twice as faster compared to the EfficientDet algorithm in the Tesla V100 GPU. However, the algorithm has lacked performance on an average machine and on single-board machines like Jetson Nano and Jetson TX2. In this research, we examine the performance of inferencing in several frameworks and propose a framework that effectively uses hardware to optimize the network while consuming less than 30% of the hardware of other frameworks.

Keywords Object detection · YOLOV4 · Optimized inferencing

1 Introduction

Object detection is also a computer vision and image processing technology used to detect the presence of significant elements in digital images and videos. Object detection is used in various computer vision applications, including photo retrieval and video monitoring. Object identification may be considered an aesthetic area within machine learning, which mostly works with picture and video streams. Deep learning detects an item in photos or videos using several algorithms such as (1) you

J. Hussain (✉) · B. R. Prathap · A. Sharma
Department of Computer Science and Engineering, Christ (Deemed to be University), Bangalore, India
e-mail: javid.hussain@mtech.christuniversity.in

B. R. Prathap
e-mail: boppuru.prathap@christuniversity.in

A. Sharma
e-mail: arpit.sharma@mtech.christuniversity.in

S. Shukla et al. (eds.), *Data Science and Security*, Lecture Notes in Networks and Systems 462, https://doi.org/10.1007/978-981-19-2211-4_27

only look once (YOLO), (2) spatial pyramid pooling (SPP-net), (3) fast R-CNN, and (4) convolutional neural networks based on regions (R-CNN), (5) oriented gradient histogram (HOG), (6) single shot detector (SSD), (7) region-based fully convolutional network (R-FCN), (8) faster R-CNN. Those object detection algorithms are accurate but the speed of inferencing has always being a problem. Face detection and tracking, vehicle detection and pedestrian identification and many more applications are examples of well-known object detection applications. YOLO-you look once is considered to be state-of-the-art technology in object detection algorithms. Joseph Redmon created the algorithm in 2015, proving to be game-changing. YOLO is faster and more accurate than other faster R-CNN and single shot detector models. YOLO performs object recognition as a regression problem, and hence, the category probabilities of the discovered photographs are reported. YOLO's real-time inference approach is built on convolutional neural networks.

YOLO does have one important shortcoming, even though it is accurate on a typical system used for day-to-day operations. The article identification using YOLO inferencing does not reach 18–20 frames per second with an honest GPU with minimum use of fifty percentage. If it is on a single-board system like Jeston Nano or Jetson TX2, the performance drops dramatically due to the high GPU consumption. Researchers applied several object tracking algorithms such as deep sort, Kalman filter, track-RNN, and others to overcome this problem for improved performance. The tracking algorithms are not hardware demanding but they lack efficiency in a real-time dynamic environment.

This article discusses about the YOLO algorithm being tested and optimized on GPU hardware. This method has been compared with various inferencing tools such as Open CV, Darknet CPU, and Darknet GPU; the algorithm has been modified to run with most optimal performance on the GPU hardware. The article is framed as follows. Section 2 refers the relative work on the YOLOV4 from different researchers. Section 3 explains about the methodology of the proposed methodology on experimental work. Section 4 gives on understanding of the results and discussion. Conclusion follows next.

2 Related Work

Research proposed by Bochkovskiy et al. [1] states that a new model of YOLO has been proposed in this paper namely YOLOV4. And the predictions result has been discussed using Tensor RT. The real-time detection was achieved at 30 fps. This model is based on Darknet framework which uses convolution neural network as a backbone. This is one stage anchor-based detector. The outline of author Ulker et al. [2] concludes that the overall performance in various machine and edge devices has exponentially increased with the use of Tensor RT, and in the frameworks like PyTorch and TensorFlow, it is noticed to be higher as they are built upon NVIDIA Cuda framework. And the authors have compared the results with various other optimized inferences using other object detection algorithms. In the literature of

Ildar [3], the author has explained the necessity of fast inference on single-board devices which has many scopes of application in further since they are affordable and less space consuming than a regular machine. The author mainly focuses on Jetson devices, the Raspberry pi family.

According to author Howell et al. [4], the research concludes the efficiency and throughput can be increased by the use of GPU and further can be increased with the help of Tensor RT in YOLOv4-tiny. In the research discussed by Rais and Munir [5] on estimation of speed of vehicle, the author has discussed the use of Tensor RT, YOLO, and Kalman filters for the use of vehicle speed estimation to make road rules more practical without a human presence. In the research proposed by Jeong et al. [6], methodology of parallelization for real-time inference on single-board machine like Jetson devices has been compared using Tensor RT and non-Tensor RT frameworks for performance measures. In the research proposed by Howell et al. [4], the selection of targeted cells was performed using object detection algorithms and has discussed the time difference by using Tensor RT in YOLOV4-tiny. According to the research proposed by Li et al. [7], six types of YOLOV4 algorithms were trained and deployed for various weather conditions to detect various traffic conditions using edge devices. The research also proves that Tensor RT takes 10 ms detection time in YOLOV4. In the research demonstrated by Cai et al. [8], a new variation of YOLOV4 has been proposed for autonomous vehicles. Since these vehicles need to make accurate decisions with high speed, the proposed algorithm makes higher accuracy than standard YOLOV4 and is named YOLOV4-5D [7]. To maximize the model's generalizability and prevent overfitting when training on the custom dataset, Ivai-Kos approached their research by training the model on a similar high volume public dataset before fine-tuning on a bespoke and low volume dataset. The research conducted by Li et al. [7] introduces a multi-task joint framework that incorporates person detection, feature extraction, and identification comparison for real-time person search. To reach real time, each part's accuracy and speed are tuned separately. The research by Bochkovskiy et al. [9] the YoloV4 was built and the result of the performance are being shown. The research proposed by Sung and Yu [10] the author has detailed the importance of YoloV4 in Licence plate recognition system in real time as the algorithm is much faster than other algorithms. According to Dai et al. [11] the use of TensorRT in field robots increases the real time detection capabilities and performance. In the research proposed by Kumar and Punitha [12] the application of YoloV4 and YoloV3 and improvements to the systems is discussed. In the research discussed by Tan et al. [13] the improved and efficient algorithm of YoloV4 was proposed for the UAVs for target detection. According to the author Mahrishi et al. [14] the YoloV4 algorithm can be used for video indexing with various parameters like f1-score, precision, accuracy and recall. This is a new approach discussed in the research which has vast importance. The application of higher frames per second inferencing plays a vital role in this modern world. To achieve real-time performance, the precision and speed of each component are individually calibrated. In our current era, the use of greater frames per second inferencing is critical. In addition, unlike the 1950s to 1980s, when neural networks were abandoned owing to hardware restrictions, Since the 2000s, hardware resources

have been at their peak, and data gathering sources have also reached their peak since data is created every minute. As a result, the implementation of hardware intensive or data-intensive algorithms will no longer struggle to provide results as they did 50 years ago.

3 Methodology

In this section, we have discussed the comparison of different model inference by defining the work flow of data collection, data preprocessing, model selection, training, and deployment on the dataset. The approach used in this research is YOLOv4 since it is fast and accurate compare to other models.

Figure 1 shows the image classification flow. The standard development process for a deep learning-based object detector includes five phases: image data gathering, object detection model selection, data preprocessing, model training, and model evaluation. This study employs a similar development technique, but with additional evaluation and refining procedures. The basic purpose of this paper is to train an object detector for benchmarking the various methods of inference.

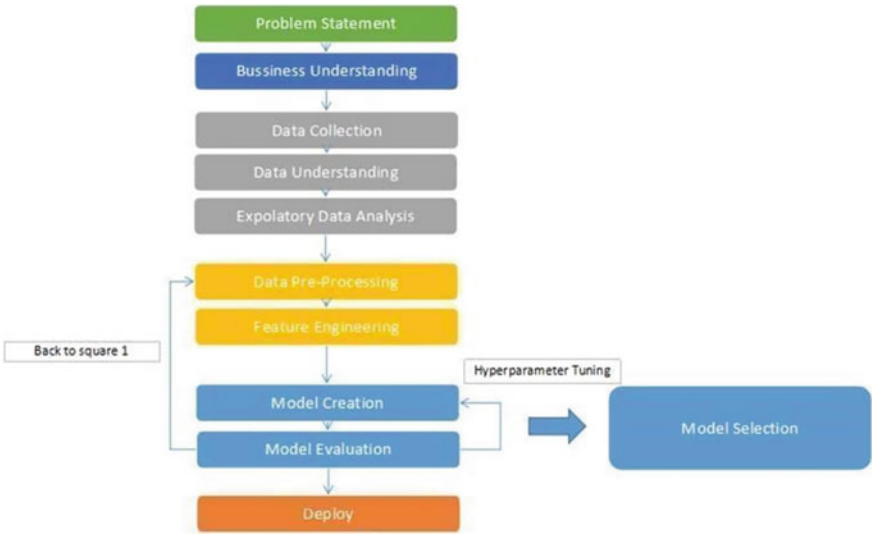

Fig. 1 Image classification model flow

3.1 Data Preprocessing

For data analytics, data preprocessing is critical. Outliers and noise are common as a result of data collecting that is not tightly regulated. The accuracy of the pipeline will be severely harmed by analyzing such data. Occlusions and overexposure should be considered in machine vision. Several data preparation approaches will be used to filter the dataset so that the model does not waste computer resources on data that is not needed. In terms of the model's framework needs and input size, the dataset must be created in accordance with the specified models. The model requires that the photographs meet the model's input size criteria. Therefore, the input parameters have been tuned for the model training input for better accuracy.

3.2 YOLO (You Only Look Once)

The YOLO stands for 'You Only Look Once.' This can be an algorithm for accurately identifying specific objects in an exceeding snapshot (in real time). Object detection in YOLO is completed as a regression problem, and also, the identified photos' class probabilities are delivered. Convolutional neural networks (CNN) are utilized in the YOLO method to find objects in real time. To detect objects, the procedure just requires one forward propagation through a neural network, because the name suggests. This signifies that one algorithm run is employed to anticipate the particular picture. The CNN is employed to anticipate multiple anchor boxes and sophistication probabilities at about an identical time.

3.3 Training and Implementation

The data for training YOLOV4 consists of human present images of around 15,000 for training, 4370 for validation, and 5000 for testing in which each image has on average 23 humans in the scene. Each image has been annotated in two classes, namely 'head' for the annotation of the human head and 'person' for the annotation of the complete person in the image. These images are then converted into YOLO format for training.

Figure 2 shows the training graph which also explains training was done for 6000 epoch.

Figure 3 shows the mean average precision obtained from class. 'Head' was 82.10% and the class 'person' was 80.40%; in total, the overall mAP was 81.25% for the output weights file. The weights file was taken from 4100 epoch to prevent overfitting.

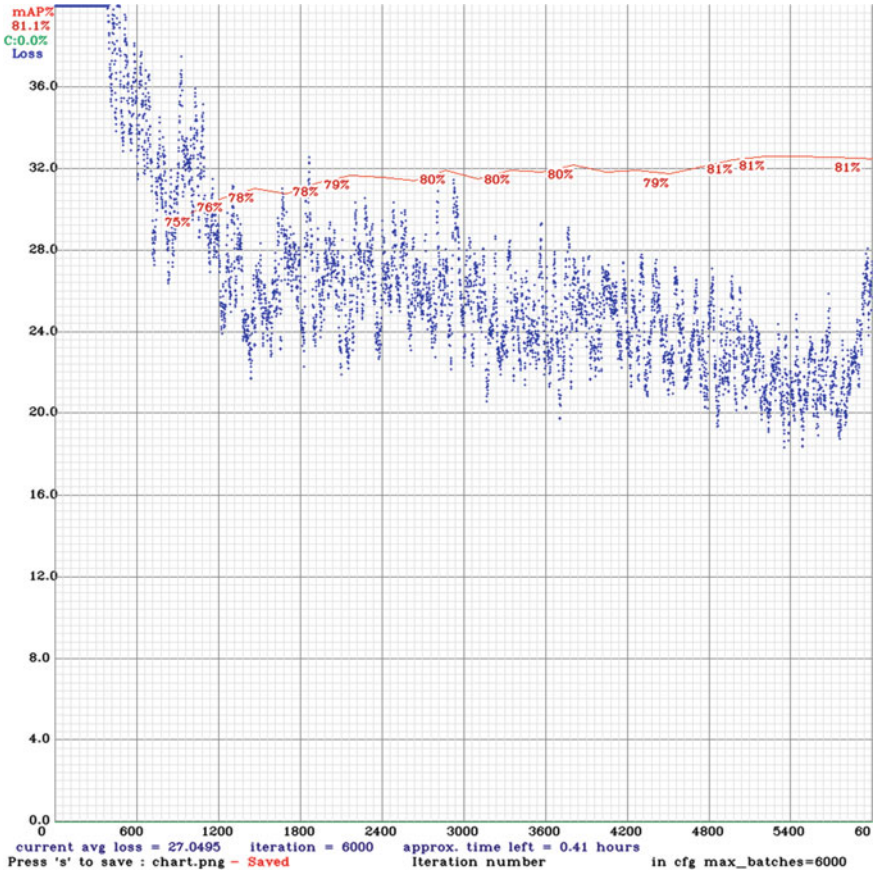

Fig. 2 Training graph

```
class_id = 0, name = head, ap = 82.10%          (TP = 65119, FP = 14590)
class_id = 1, name = person, ap = 80.40%        (TP = 72055, FP = 11766)

 for conf_thresh = 0.25, precision = 0.84, recall = 0.75, F1-score = 0.79
 for conf_thresh = 0.25, TP = 137174, FP = 26356, FN = 46191, average IoU = 66.92 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.812542, or 81.25 %
```

Fig. 3 mAP (mean average precision) of trained model

The trained model of YOLOv4 was obtained, and the results were compared using four methods, (1) Using OpenCV, (2) Using Tensor RT (3) Using Darknet with GPU, (4) Using Darknet with CPU (Figs. 4 and 5).

Fig. 4 Video stream inferencing using Tensor RT

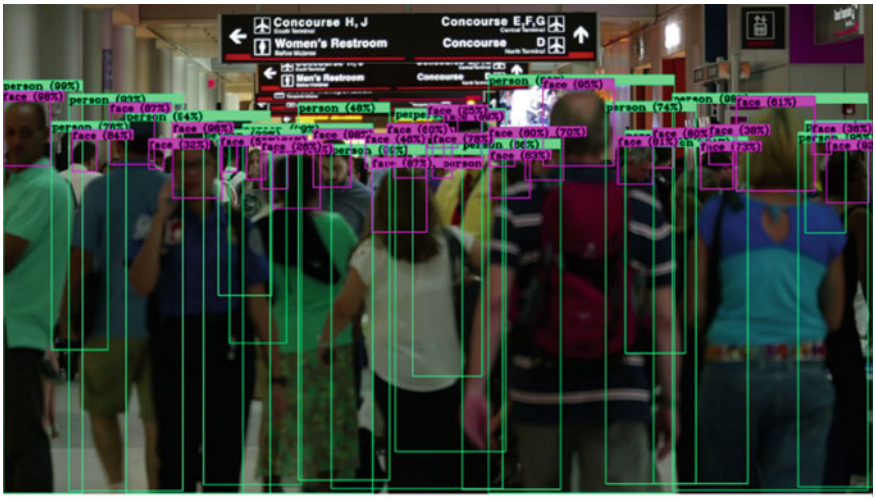

Fig. 5 Video stream inferencing using Darknet

4 Result and Discussion

The result is tested using a similar dataset trained with YOLOV4 of resolution 416 × 416 on a notebook with i5-11300H with NVIDIA GeForce RTX3060 6 GB GDDR6 and 16 GB DDR4 Ram. The inference was performed on video with a resolution of 1920 × 1080, with Darknet GPU model and Tensor RT optimized model averaging 29 and 28 frames per second, respectively. OpenCV model and Darknet CPU optimized

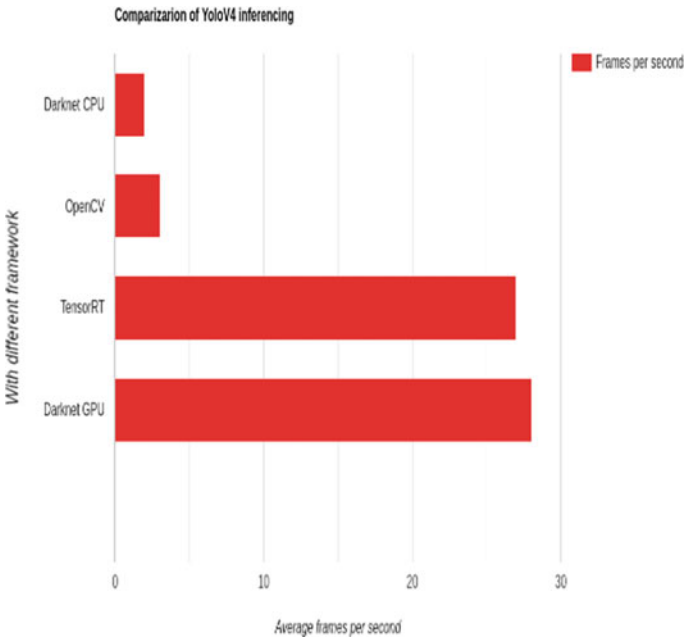

Fig. 6 Average frames per second inferencing

model averaging 3 and 2 frames per second, respectively. As the Darknet GPU and Tensor RT were approximately equal in performance, the GPU usage of these frameworks varied a lot, with Darknet GPU at 58% of GPU consumption and Tensor RT being at 27% of GPU consumption. There was, on average, more than 30% of GPU consumption difference between these models because of which it would become difficult to deploy Darknet GPU model inferencing on critical hardware applications.

Figure 6 shows the average frames per second while inferencing on 1920 × 1080 resolution video file using YOLOV4 in various frameworks. Figure 7 shows the actual frames per second in each frame while inferencing on 1920 × 1080 resolution video file using YOLOV4 in various frameworks.

The above result is obtained by quantizing the YOLOV4 algorithm in INT8 form to perform hi-speed inferencing which is done by Tensor RT, and enables parallel processing in GPU with results in low latency and high throughput. This is achieved by following methods which is inbuilt in Tensor RT such as: (1) reduced precision, (2) layer and tensor fusion, (3) kernel auto-tuning, (4) dynamic tensor memory, (5) multi-stream execution, and (6) time fusion. The reduced precision layer provides FP16 or INT8, increasing throughput by quantizing models while maintaining accuracy. Layer and tensor fusion layer fuses nodes during a kernel, which utilizes the GPU memory and bandwidth efficiently. Supported the target GPU platform, the kernel auto-tuning layer chooses the top effective data layers

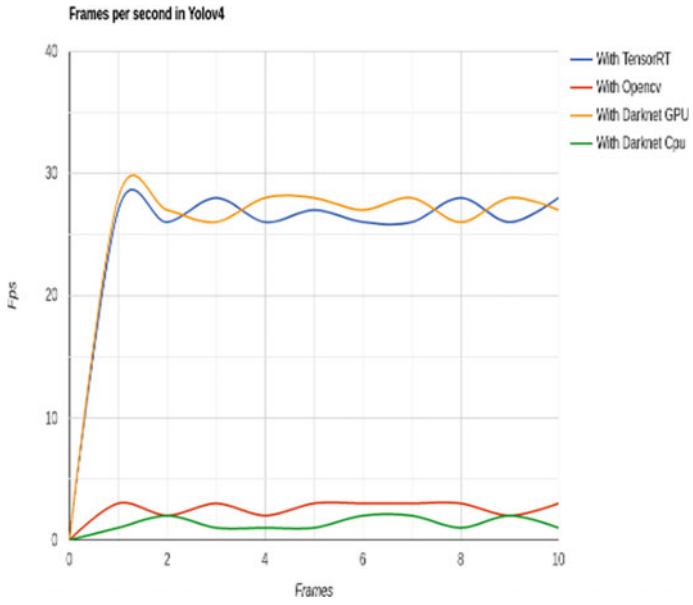

Fig. 7 Actual frames per second in inferencing

and algorithms. Dynamic tensor memory layer reduces memory footprint and efficiently reuses memory for tensors. In the multi-stream execution layer, multiple input streams are processed in parallel employing a scalable approach. The time fusion layer strengthens recurrent neural networks over a number of iterations by using flexibly synthesized modules. This study indicates how the YOLOV4 technique is implemented in various frameworks on a similar dataset and examines their inference performance.

Figure 8 shows the GPU utilization of 27% while inferencing Tensor RT.

Figure 9 shows the GPU utilization of 58% while inferencing Darknet GPU.

Figure 10 shows the comparison GPU Utilization by frameworks used for inference.

5 Conclusion

Computer vision is image processing technology used to recognize important features in digital photos and movies. Object detection applications include face detection, vehicle detection, pedestrian identification, and many others. In this study, YOLOV4 object detection algorithm is integrated with enhanced inference methods in this study. In this paper, we have compared the state-of-the-art YOLO algorithm with various frameworks for inferencing. The result shows that using Tensor RT in YOLOV4 has given the same performance as Darknet framework with more than

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.57.02    Driver Version: 470.57.02    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  On   | 00000000:01:00.0 Off |                  N/A |
| N/A   56C    P0    29W /  N/A |   1824MiB /  5946MiB |     27%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A       982      G   /usr/lib/xorg/Xorg                 45MiB |
|    0   N/A  N/A      1564      G   /usr/lib/xorg/Xorg                227MiB |
|    0   N/A  N/A      1861      G   /usr/bin/gnome-shell               56MiB |
|    0   N/A  N/A      2637      G   /usr/lib/firefox/firefox          189MiB |
|    0   N/A  N/A      2845      G   /usr/lib/firefox/firefox            2MiB |
|    0   N/A  N/A      2887      G   /usr/lib/firefox/firefox            2MiB |
|    0   N/A  N/A      7342      G   ...AAAAAAAAA= --shared-files       32MiB |
|    0   N/A  N/A      8940      G   /usr/lib/firefox/firefox            2MiB |
|    0   N/A  N/A     22374      C   python3                          1249MiB |
+-----------------------------------------------------------------------------+
```

Fig. 8 GPU utilization using Tensor RT

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.57.02    Driver Version: 470.57.02    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  On   | 00000000:01:00.0 Off |                  N/A |
| N/A   69C    P0    70W /  N/A |   2208MiB /  5946MiB |     58%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A       982      G   /usr/lib/xorg/Xorg                 45MiB |
|    0   N/A  N/A      1564      G   /usr/lib/xorg/Xorg                233MiB |
|    0   N/A  N/A      1861      G   /usr/bin/gnome-shell               44MiB |
|    0   N/A  N/A      2637      G   /usr/lib/firefox/firefox          197MiB |
|    0   N/A  N/A      2845      G   /usr/lib/firefox/firefox            2MiB |
|    0   N/A  N/A      2887      G   /usr/lib/firefox/firefox            2MiB |
|    0   N/A  N/A      7342      G   ...AAAAAAAAA= --shared-files       30MiB |
|    0   N/A  N/A      8940      G   /usr/lib/firefox/firefox            2MiB |
|    0   N/A  N/A     22892      C   ./darknet                        1633MiB |
+-----------------------------------------------------------------------------+
```

Fig. 9 GPU utilization using Darknet GPU

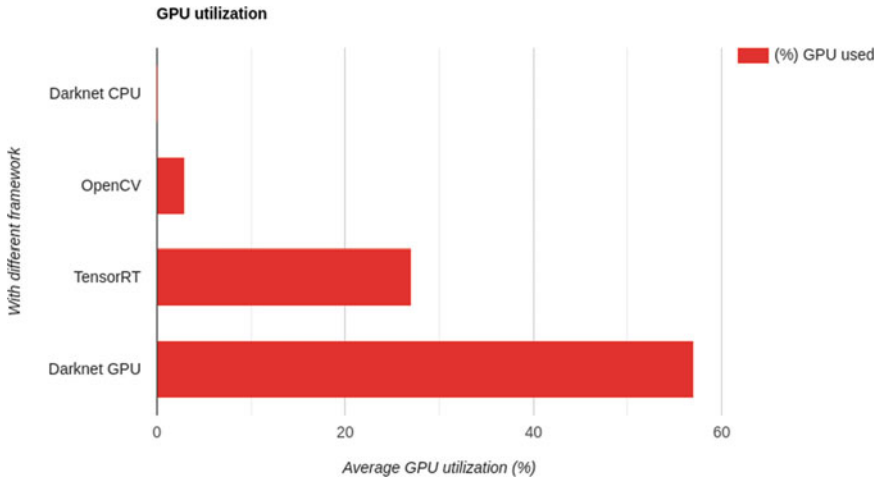

Fig. 10 Comparison of average GPU utilization

30% lesser usage of GPU as this model can be deployed in single-board as well as NVIDIA powered machine for various applications in real time like traffic monitoring system, vehicle tracking, violation detection in a crowd, animal detection in farmland, and more. As a future scope, these models can be deployed in low hardware specification devices for better detection accuracy and inference time.

References

1. Wang C-Y, Bochkovskiy A, Mark Liao H-Y (2021) Scaled-YOLOv4: scaling cross stage partial network. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition
2. Ulker B et al (2020) Reviewing inference performance of state-of-the-art deep learning frameworks. In: Proceedings of the 23rd international workshop on software and compilers for embedded systems
3. Ildar R (2021) Increasing FPS for single board computers and embedded computers in 2021 (Jetson nano and YOVOv4-tiny). Practice and review. arXiv preprint arXiv:2107.12148
4. Howell L, Anagnostidis V, Gielen F (2021) Multi-object detector YOLOv4-tiny enables high-throughput combinatorial and spatially-resolved sorting of cells in microdroplets. Adv Mater Technol 2101053
5. Rais AH, Munir R. Vehicle speed estimation using YOLO, Kalman filter, and frame sampling
6. Jeong EJ et al (2021) Deep learning inference parallelization on heterogeneous processors with TensorRT. IEEE Embed Syst Lett
7. Li X et al (2021) Research on the application of YOLOv4 target detection network in traffic scenarios by machine vision technology. J Phys: Conf Ser 2033(1)
8. Cai Y et al (2021) YOLOv4-5D: an effective and efficient object detector for autonomous driving. IEEE Trans Instrum Meas 70:1–13
9. Bochkovskiy A, Wang C-Y, Mark Liao H-Y (2020) YOLOv4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934

10. Sung J-Y, Yu S-B (2020) Real-time automatic license plate recognition system using YOLOv4. In: 2020 IEEE international conference on consumer electronics-Asia (ICCE-Asia). IEEE
11. Dai B et al (2021) Field robot environment sensing technology based on TensorRT. In: International conference on intelligent robotics and applications. Springer, Cham
12. Kumar C, Punitha R (2020) YOLOv3 and YOLOv4: multiple object detection for surveillance applications. In: 2020 third international conference on smart systems and inventive technology (ICSSIT). IEEE
13. Tan L et al (2021) YOLOv4_Drone: UAV image target detection based on an improved YOLOv4 algorithm. Comput Electr Eng 93:107261
14. Mahrishi M et al (2021) Video index point detection and extraction framework using custom YoloV4 Darknet object detection model. IEEE Access 9:143378–143391