International Journal of Advanced Smart Convergence Vol.12 No.4 182-189 (2023)
http://dx.doi.org/10.7236/IJASC.2023.12.4.182

IJASC 23-4-23

Car detection area segmentation using deep learning system

Dong-Jin Kwon [1] , Sang-hoon Lee [2]

[1] *Associate Professor, Department of Computer Electronics Engineering, Seoil University, Korea*
djkwon77@seoil.ac.kr
2 Master , Korea Institute of Science and Technology, Korea
lsh950223@kist.re.kr

Abstract

A recently research, object detection and segmentation have emerged as crucial technologies widely utilized in various fields such as autonomous driving systems, surveillance and image editing. This paper proposes a program that utilizes the QT framework to perform real-time object detection and precise instance segmentation by integrating YOLO(You Only Look Once) and Mask R CNN. This system provides users with a diverse image editing environment, offering features such as selecting specific modes, drawing masks, inspecting detailed image information and employing various image processing techniques, including those based on deep learning.

The program advantage the efficiency of YOLO to enable fast and accurate object detection, providing information about bounding boxes. Additionally, it performs precise segmentation using the functionalities of Mask R CNN, allowing users to accurately distinguish and edit objects within images. The QT interface ensures an intuitive and user-friendly environment for program control and enhancing accessibility.

Through experiments and evaluations, our proposed system has been demonstrated to be effective in various scenarios. This program provides convenience and powerful image processing and editing capabilities to both beginners and experts, smoothly integrating computer vision technology. This paper contributes to the growth of the computer vision application field and showing the potential to integrate various image processing algorithms on a user-friendly platform

Keywords: *Image processing, QT, Deep learning segmentation, Object detection.*

1. IntrodCon

As the importance of object detection technology continues to grow in the field of modern computer vision and studies, it plays a various applications such as vehicle-related systems, smart city development and security systems. In this context, the YOLO(You Only Look Once) algorithm has garnered attention for its innovative approach in the field of object detection, offering real-time performance and high accuracy due to its fast inference speed. According to YOLO related research, this algorithm performs object detection in a single forward pass, predicting probabilities for bounding boxes and classes in grid cells divided from the image [3-4].

Manuscript Received: october. 19, 2023 / Revised: october. 25, 2023 / Accepted: october. 29, 2023
Corresponding Author: djkwon77@seoil.ac
Tel: :+82-2-490-7349, Fax: +82-2-490-7802
Associate Professor , Department of Computer Electronics Engineering, Seoil University, Korea

Image segmentation technology divides given images into meaningful parts, contributing to the effective extraction of visual information by accurately identifying and distinguishing object boundaries. Recent advancements in deep learning have made to image segmentation technology more accurate and applicable. This segmentation technology finds essential applications in various fields such as autonomous driving, robotics, medical imaging and transportation. The traffic-related segmentation experiment conducted in this research benefits from the fusion of traditional image processing techniques and deep learning algorithm, allowing for more precise and rapid analysis. However, accurate segmentation in complex traffic situations remains a challenging task, especially considering various uncertainties such as traffic congestion and road construction [1, 7, 9]

With the recent advancements in GPU technology, revolutionary models utilizing image object detection and segmentation have led to a significant increase in technologies using GPUs. The architecture of GPUs, particularly in the SIMT(Single Instruction Multiple Threads) mode, excels in high-performance computational processing and significantly enhances the inference time of models. Papers and research on the performance of models have consistently proven the superiority of GPU performance in terms of learning speed and accuracy compared to CPUs. Consequently the development utilizing GPU architecture is actively taking place in many deep learning research and application fields [5-6, 8, 14].

This research proposes a deep learning-based on object detection and segmentation program implemented using the QT framework in a Windows environment. The program advantages to libraries such as OpenCV, OpenVINO and CUDA to provide users with enhanced image processing capabilities. Through features allowing users to manually draw and erase mask areas, the program enables object segmentation and further allows users to manually draw and erase bounding boxes for precise object positioning. Additionally, by utilizing YOLO for rapid object detection and Mask R CNN for accurate segmentation the program provides comprehensive information about objects, including names and positions. Basic functionalities such as saving and loading user-generated results are supported, enabling flexible image processing anytime, anywhere.

In summary, this research presents an efficient approach in the field of object detection and segmentation through the effective combination of deep learning and the QT framework.

2. Background Knowledge

2-1. Definition of Deep learning.

Deep learning takes the form of a machine learning algorithm with a structured hierarchy called an Artificial Neural Network (ANN). Inspired by the functioning of the human brain, deep learning excels in learning and abstracting patterns from complex and large-scale datasets. Using neural networks composed of multiple hidden layers, deep learning hierarchically extracts features and learns intricate meanings Through this, deep learning has demonstrated outstanding performance across various types of data, including images, speech and text. Key architectures used in deep learning include to Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and custom models. each designed for tasks such as image processing, sequential data analysis and language modeling. Deep learning exhibits performance in representation learning for complex and diverse datasets compared to traditional machine learning methods, leading to innovative results in various technological domains [10].

2-2 Definition of object segmentation.

Object segmentation is a crucial application in deep learning and computer vision. Involving the precise identification and separation of specific object in images or videos. This task includes accurately extracting the boundaries of objects at the pixel level and determining to which object each pixel belongs. Object segmentation extraction is employed to isolated and emphasize target objects in images or videos. also allowing for more accurate extraction and interpretation of visual information. It plays a key point in various field , including autonomous vehicles, medical image analysis and security systems. With the advancement of deep learning technology, object segmentation extraction has become more efficient in terms of accuracy and efficiency compared to previous methods. [2-4, 13]

The most of importance object segmentation algorithms include Fully Convolutional Network (FCN), U-Net and Masak R CNN, enabling precise pixel-level object segmentation. Object segmentation extraction holds significance in various application domains, driving advancements in modern computer vision by enabling accurate extraction and interpretation of visual information[11-12, 15].

3. Suggestion System

3-1. Setup environment

Table 1. System Environment

Computer environment	Software framework and library
GPU : Geforce MX150	OpenCV 4.2
CPU : Intel Core i7 8550U @ 1.8GHZ	OpenVINO 2020.3
RAM : 8G DDR4	QT 5.9
OS : Windows 10	CUDA Toolkit 10.0
HDD : TOSHIBA MQ01ABD100 (1TB)	Tensorflow 2.0
SDD : LITEON CV3-8D128 (128GB)	

The Table 1 presents the overall system environment developed to program. The GPU utilized is the Geforce MX150, complemented by an Intel Core i7 8550U @ 1.8GHZ CPU and 8G RAM. This environment usually employed execution involving deep learning models such as YOLO V3 and Mask R CNN. The library stack consist of OpenCV 4.2, OpenVINO 2020.3, QT 5.9, CUDA Toolkit 10.0 and Tensorflow 2.0. OpenCV 4.2 is employed for comprehensive image processing tasks including editing mask, loading, saving images, and post-processing in deep learning image tasks OpenVINO can enhances deep learning inference calculation when GPU architecture is unavailable. if a GPU architecture is available, recommends utilizing the CUDA Toolkit for enhanced performance in deep learning inference calculation compared to the CPU architecture. QT is responsible for constructing the User Interface(UI) and serving as the fundamental operating program. Finally the YOLO and Mask R CNN model are implemented within the Tensorflow 2.0 framework.

The program developed in this study is based on the Windows operating system and utilizes the QT framework to provide users with a friendly and intuitive interface. The Windows environment is well-known and familiar to a diverse user base, also QT contributes to cross-platform portability, enhancing development and usability.

The program incorporates essential libraries for image processing and the execution of deep learning models. OpenCV offers robust image processing capabilities, providing users with functions necessary for editing mask areas and bounding boxes. Additionally, it utilizes an image viewer to deployment or save results. OpenVINO serves as Intel’s latest deep learning model inference engine, contributing to faster inference speed and

efficiency and enhancing the runtime performance of deep learning. CUDA have been advantaged to data obtained in the program to utilize the NVIDIA GPU architecture, employing Single Instruction Multiple Threads(SIMT) to execute multiple threads simultaneously for parallel processing. This enables the acceleration of deep learning models within the program.

3-2. System structure

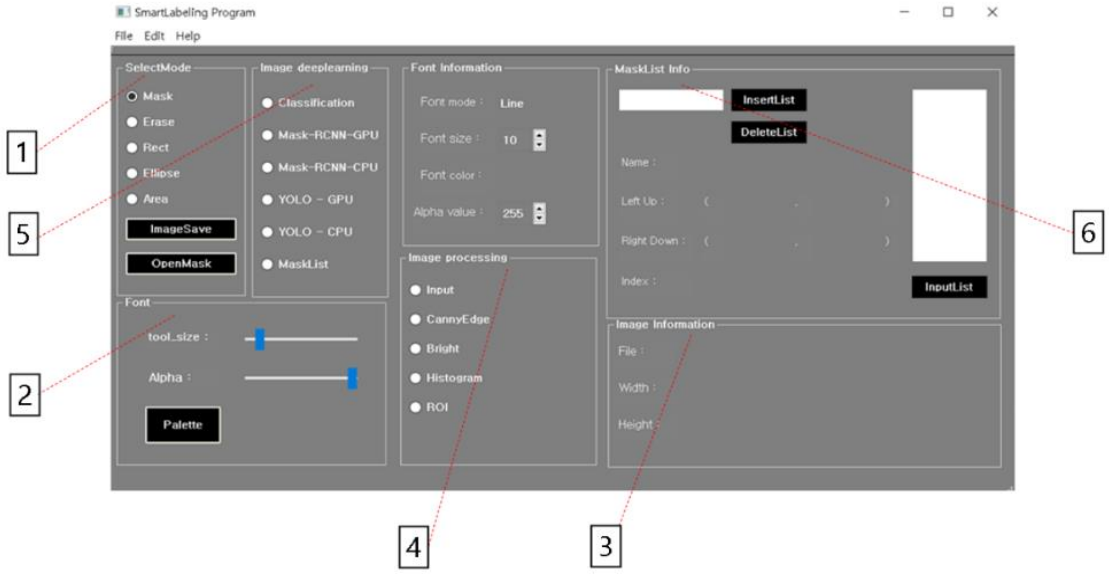


Figure 1. Program User interface

1	Image edit selection mode	
2	Draw Mask area : size and set alpha	
3	Opened image information	
4	Select for image processing	
5	Select for image processing using deep learning system	
6	Object Bounding box information	

Table 2. Table for Operating Program

Table 2 is shows that the detailed functionalities of Fig 1. Function 1 allows users to select a mode for editing images. It enables essential editing features such as drawing mask regions, displaying bounding boxes and erasing incorrect mask areas. Function 2, when selecting the mask drawing mode than allows users to set the size and Alpha value of the drawing area, providing transparency control. Function 3 displays information regarding the width and height of the image. Function 4 includes features such as saving drawn mask regions during image processing, selecting ROI(Region Of Interest), drawing Canny Edges and choosing Histogram mode for image rendering. Function 5 enables the processing of images using YOLO and Mask R CNN models. Lastly function 6 provides information about the selected bounding box area

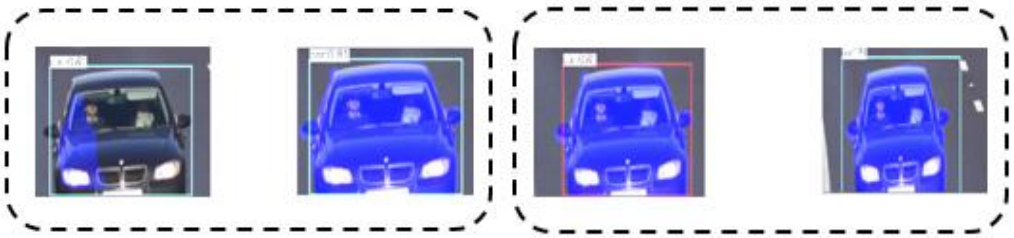
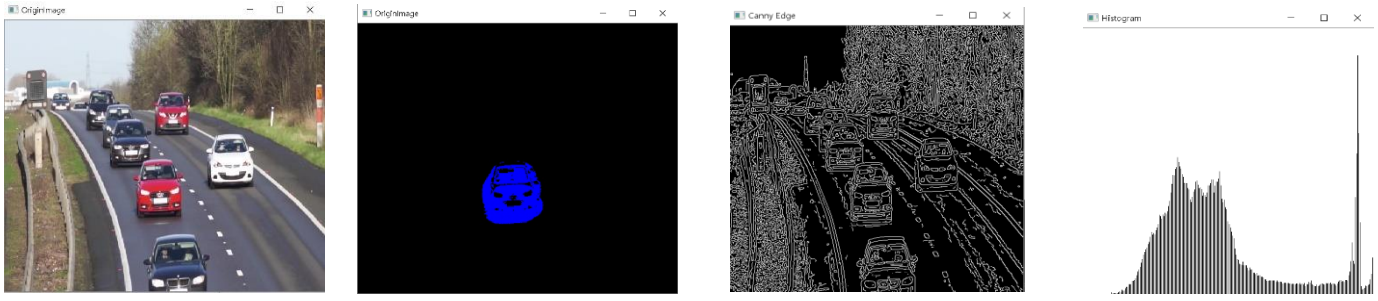


Figure 2. (a) Mask area edit , (b) Bounding box edit

Function 1 and 2 provide users with the capability to manually draw and erase mask areas, as represent in Fig 2. These functions are essential for object segmentation, allowing users to draw mask areas to assist deep learning models in accurate object detection and segmentation. Additionally, even if the results inferred by the deep learning model contain inaccuracies due to slight errors, the program is designed to empower users to manually adjust or erase mask areas and bounding box size. This function ensures a more precise adjustment of object sizes and positions and enhancing the overall accuracy of the segmentation process.



(a) **(b)** **(c)** **(d)**

Figure 3. (a) Origin image , (b) ROI drawing , (c) Canny Edge , (d) Histogram

The image processing performed through Functions 3 and 4 is represent in Fig 3. It is shows that the results obtained from the original image (a) when drawing mask areas, saved as (b), the output after Canny Edge processing represented as (c) and the visualization of the histogram denoted as (d). [16-17]

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2 + y^2) / (2\sigma^2)} \quad (1)$$

$$\text{Magnitude } (M) = \sqrt{(G_x)^2 + (G_y)^2} \quad (2)$$

$$\text{Direction}(\theta) = \arctan\left(\frac{G_y}{G_x}\right) \quad (3)$$

Canny Edge applies a Gaussian filter (1) to the original image to reduce noise. Here, G(x, y) represents the values of the Gaussian kernel and σ signifies the standard deviation of the Gaussian filter. From the image computed with the Gaussian filter, it calculates the strength (2) and direction (3) of edges.

$$M`(x, y) = \begin{cases} M(x, y), \text{ if } M(x, y) >= M(x + \Delta x, y + \Delta y) \text{ and } M(x, y) \\ 0, \text{ otherwise} \end{cases} \quad (4)$$

It examines the pixels on either side of the edge to find the maximum value and removes the remaining pixels (4). Double threashold is then employed to classify the strength of edges and noise.

$$x \cos(\theta) + y \sin(\theta) = p \quad (5)$$

Through Hough Transform (5), specific values accumulating in the Hough space are identified and the corresponding polar coordinates at those locations are transformed back into the equation of lines This process ultimately produces the final output of the Canny edge results.

$$H(r) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \delta(I(i, j) - r) \quad (6)$$

Histogram calculates the frequency of occurrence of pixel value r from the original image in the image histogram function (6). Here I(i , j) represents the pixel value located at the i row and j column of the image and δ is the directionality function.

$$C(r) = \sum_{k=0}^{r} H(k) \quad (7)$$

Through the image histogram function, the cumulative histogram from pixel value 0 to r is obtained. This process results in the output of the histogram, allowing for the assessment of contrast, brightness distribution and other characteristics of the image.

3-3. Program operating

The process of handling the original image through the deep learning model using Function 5 is illustrated in Fig 4. cumulative The input image is resized to a width of 416 and a height of 416 using the OpenCV library's resize operation. To serve as input data for YOLO and Mask R CNN models, a single image with three channels and consists of 2 dimensions a width 416 and a height 416 is utilized. The preprocessing of image data determines the labels of the processed image data based on the Class threshold value. Ultimately, in the post-processing step, the label of the Classes are determined and for Mask R CNN, an additional step involves drawing the mask area.

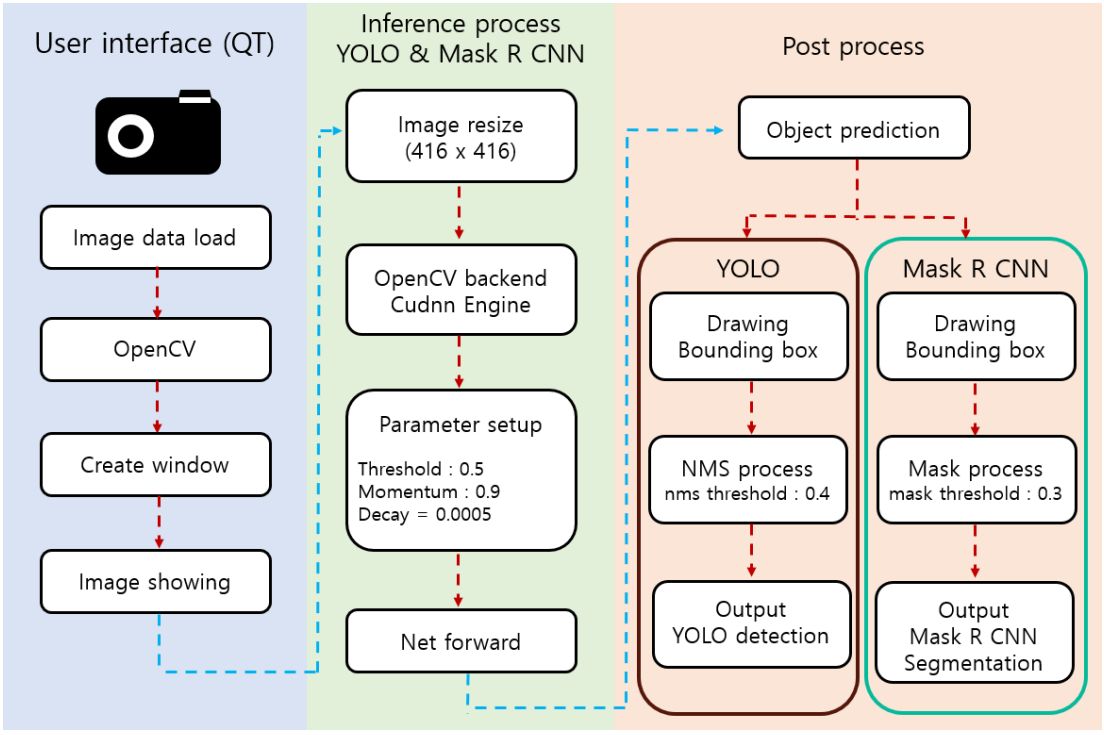
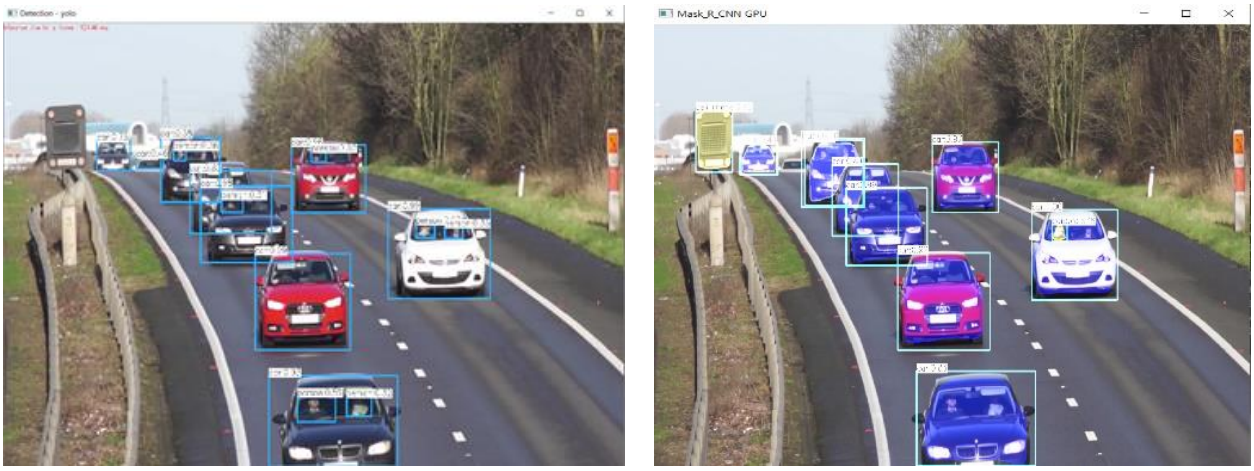


Figure 4. Image processing using deep learning model

Figure 5. Getting bounding box information

Finally function 6 displays information about the selected bounding box when a bounding box is chosen.

4. Object detection result.



(a) **(b)**

Figure 6. (a) YOLO object detection , (b) Mask R CNN object segmentation

The program employs the YOLO deep learning model for swift object detection. While this model was defective to segmentation, it exhibits significantly faster inference speeds compared to Mask R CNN, making it well-suited for real-time object detection in scenarios such as video processing. YOLO provides high performance across diverse environment, making it a suitable to choice for rapid and efficient object detection. The Fig 6(a) shows the output image of this program.

Mask R CNN provides precise segmentation of objects, offering more accurate positional information about the size and shape of objects compared to YOLO. Although it consumes more memory and computational resources for image region processing compared to YOLO. It is effective in obtaining segmentation results for object detection at a finer level, particularly in photo-level data, as represented in Fig 6(b). Users can get an advantages this to maximize the segmentation capabilities of the deep learning model.

5. Conclusion

In this paper, we have developed a QT framework-based on deep learning object detection and segmentation program in the Windows environment. The program offers a user-friendly interface with diverse functionalities, combining YOLO and Mask R CNN for real-time object detection and segmentation. Utilizing OpenCV, OpenVINO and CUDA libraries, the program ensures high performance and efficiency. Real-time object detection with YOLO provides fast inference speed and accurate identification, while segmentation with Mask R CNN offers precise object boundaries for fine image editing.

The program allows users to easily draw and edit mask areas through an intuitive interface. Users can achieve their desired results through object detection and segmentation, enhancing efficiency with features like saving and loading projects. These achievements underscore the contemporary applicability of object segmentation technology, and future integrations with a variety of deep learning-based technologies are expected to provide users with more choices and functionalities. Furthermore, deeper research is needed to explore the practical applications and potential advancements in real-world scenarios.

Acknowledgement

The present research has been conducted by the Research Grant of Seoil University

References

[1] K. He, G. Gkioxari, P. Dollár, and Ross B. Girshick, "Mask R-CNN," , ICCV 2017.
DOI : https://doi.org/10.1109/ICCV.2017.322
[2] J. Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", pp. 779-788, CVPR, 2016.
DOI : https://doi.org/10.1109/CVPR.2016.91
[3] J. Redmon, A. Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR, 2017.
DOI : https://doi.org/10.1109/CVPR.2017.690
[4] J. Redmon, Ali Farhadi, "YOLOv3: An Incremental Improvement", arXiv, 2018
[5] T. Dettmers, "Benchmarking state-of-the-art deep learning software tools", arXiv, 2016.
[6] S. Mittal, J. S Vetter, “A survey of CPU-GPU heterogeneous computing techniques”, CSUR, 2015.
DOI : https://doi.org/10.1145/2788396
[7] J G. Ellis, D. Joshi, et al, "Region-based Image Retrieval with Revisited", arXiv, 2017.
[8] S. Chetlur, et al, “cuDNN: Efficient Primitives for Deep Learning”, arXiv, 2014.
[9] Y. Zhu, et al, "Instance-aware Semantic Segmentation via Multi-task Network Cascades", CVPR, 2017.
DOI : https://doi.org/10.1109/CVPR.2016.343
[10] Y. LeCun, Y. Bengio and G. Hinton, “Deep Learning”, nature, 2015.
DOI : https://doi.org/10.1038/nature14539
[11] J. Long, E. Shelhamer and T. Darrel, “Fully Convolutional Networks for Sementic Segmentation”, CVPR, 2015.
DOI : https://doi.org/10.1109/TPAMI.2016.2572683 ,
[12] V. V. Zunin, “Intel OpenVINO Toolkit for Computer Vision: Object Detection and Semantic Segmentation” , 2021 International Russian Automation Conference (RusAutoCon) , Sep 2021.
DOI : https://doi.org/10.1109/RusAutoCon52004.2021.9537452
[13] M. Mahrishi, S. Morwal, A. Wahab Muzaffar, S. Bhatia, P. Dadheech, M. Khalid Imam Rahmani, “Video Index Point Detection and Extraction Framework Using Custom YoloV4 Darknet Object Detection Model”, Volume: 9, IEEE, Oct 2021.
DOI : https://doi.org/10.1109/ACCESS.2021.3118048
[14] C. Yao, W. Liu, W. Tang, J. Guo, S. Hu, Y. Lu, W. Jiang, “Evaluating and analyzing the energy efficiency of CNN inference on high-performance GPU” , Wiley Online Library, Oct 2020.
DOI : https://doi.org/10.1002/cpe.6064
[15] N. Shrivastava, V. Tyagi, “A Review of ROI Image Retrieval Techniques”, volume: 328, AISC, 2015.
[16] W Rong, Z Li, W Zhang, L Sun, “An improved CANNY edge detection algorithm” IEEE, Aug 2014.
DOI : https://doi.org/10.1109/ICMA.2014.6885761
[17] A. Sharma; K. Shah; S. Verma, “Face Recognition using Haar Cascade and Local Binary Pattern Histogram in OpenCV”, 2021 Sixth International Conference on Image Information Processing (ICIIP) , Nov 2021.
DOI : https://doi.org/10.1109/ICIIP53038.2021.9702579