You Only Look at Interested Cells: Real-Time Object Detection Based on Cell-Wise Segmentation

Kai Su, Huitao Wang, Intisar MD Chowdhury, Qiangfu Zhao, Yoichi Tomioka
Graduate School of Computer Science and Engineering
The University of Aizu
Aizu-Wakamatsu, Fukushima, Japan
{m5232109, m5241105, d8211106, qf-zhao, ytomioka}@u-aizu.ac.jp

2020 11th International Conference on Awareness Science and Technology (iCAST) | 978-1-7281-9119-5/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICAST51195.2020.9319469

Abstract—**In this paper, we study real-time object detection based on cell-wise segmentation. Existing object detection methods usually focus on detecting interesting object's positions and sizes and demand expensive computing resources. This process makes it difficult to achieve high-speed and high-precision detection with low-cost devices. We propose a method called You Only Look at Interested Cells or in-short YOLIC to solve the problem by focusing on predefined interested cells (i.e., subregions) in an image. A key challenge here is how to predict the object types contained in all interested cells efficiently, all at once. Instead of using multiple predictors for all interested cells, we use only one deep learner to classify all interested cells. In other words, YOLIC applies the concept of multi-label classification for object detection. YOLIC can use exiting classification models without any structural change. The main point is to define a proper loss function for training. Using on-road risk detection as a test case, we confirmed that YOLIC is significantly faster and accurate than YOLO-v3 in terms of FPS and F1-score.**

Index Terms—**Cell-wise Segmentation, Real-time Detection, Obstacle Detection, Object Detection, Deep Learning**

I. Introduction

Object detection is one of the most important and crucial tasks in the computer vision field over the past decade. Recently, object detection performance has been significantly improved due to fast progress in deep learning-based algorithms and computing resources.

The key of object detection algorithms is how to find the position of the detected object efficiently. The generic object detection algorithms can be divided into region proposal based methods and regression-based methods. The region proposal based approaches that have been represented by Regions-With-CNN-Features [1] (R-CNN) first look for all potential rectangular regions in the image. Afterward, the classification of all proposal regions (into their respective classes) has been performed by using a Convolutional Neural Network (CNN). In the final step, the redundant bounding boxes are filtered out using the techniques known as the Non-Maximum Suppression (NMS) [2]. On the other hand, regression-based algorithms such as You-Only-Look-Once (YOLO) [3] and Single Shot multiBox Detector (SSD) [4] are much more straightforward in a sense that they only use single CNN to predict the position and category of objects. Afterward, the NMS technique is applied to get rid of the redundant and low confidence bounding boxes. It is not difficult to see that there are redundant results in obtaining the object's position.

Also, the current object detection algorithm uses a big rectangular box to represent the location of the one object directly. This is a simple and straightforward way to point out the location of most objects in the image. However, objects may appear in images in various strange shapes. For some practical applications such as autonomous driving, a single rectangular box cannot well represent for all objects, especially for some long oblique objects in images. To deal with these problems, some semantic segmentation based methods can be used for predicting the object at the pixel level, which can more accurately give the precise location of the object. In this respect, precise semantic segmentation based detection methods are a good choice for handling irregular objects. However, segmentation-based detection methods required heavy computations for getting the high-resolution feature maps, which currently cannot meet real-time requirements in mobile devices.

In this paper, we propose a new object detection approach called "You Only Look at Interested Cell (YOLIC)." The proposed detection model leverages both 'bounding box' and 'semantic segmentation' methods. A detected object can be represented by one or more small cells (or blocks) instead of one big bounding box or pixel area where each cell will focus on the different particular parts of the object. Fig. 1 illustrates our detection system. If we consider the whole image as interesting regions, we can divide the input image into different interesting cells and perform prediction on each cell. In general, our proposed YOLIC has the following benefits over existing object detection methods.

- First, since each cell's position is fixed in the images, we do not need to use region proposal-based techniques or regression techniques to find the bounding box from full image pixels. YOLIC can simultaneously detect all interested cells in the whole image and give each cell classification scores at a high detection speed. We can deploy YOLIC to some mobile devices for object detection in real-time. This means that we can provide local

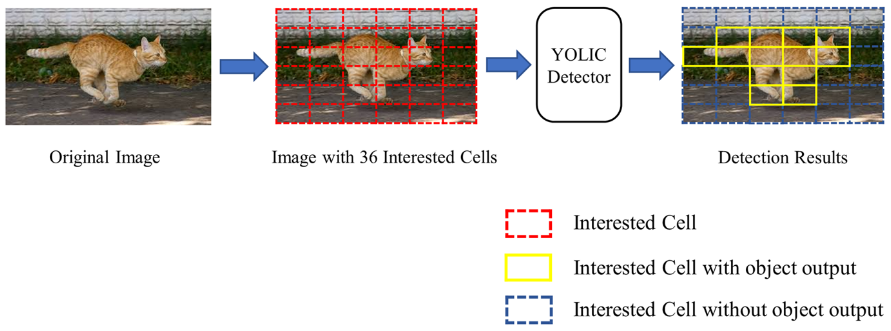
Original Image
Image with 36 Interested Cells
YOLIC Detector
Detection Results
Interested Cell
Interested Cell with object output
Interested Cell without object output

Fig. 1. The proposed YOLIC detection system. Note that we need to predefine the cells of interest (COIs) to train the YOLIC detection model. In this example, we have predefined 36 identical COIs for the entire image, and a trained detector will only look at these COIs.

real-time object detection capabilities for edge devices, making edge devices more intelligent.

- Second, we can directly use some state-of-the-art classification models [5] [6] as the backbone network for the object detection tasks without any significant structural change. From this point of view, YOLIC is very convenient for practical applications since we can easily implement existing state-of-the-art deep neural networks as the backbone network.
- Third, YOLIC is different from existing regression-based object detection algorithms in classifying potential regions. In our detection system, each interested cell can have multiple category results, so YOLIC can deal better with the situation where multiple objects appear densely in the image.

In this study, we apply YOLIC to on-road detection task for verifying its practical feasibility. The on-road risk detection task requires strict requirements for real-time and high detection precision. Also, existing research generally relies on expensive computing resources, and there are few object detection algorithms for cost-conscious vehicles. Therefore, it is very important to create a lightweight, power-efficient, and accurate algorithm for risk detection in edge devices. Our real-time object detection algorithm running on low-cost devices also fulfills the needs of tiny AI. We tested YOLIC on different road conditions; please see the demo on the page: https://j.mp/2Ej3D2e.

The rest of this paper is structured as follows. In Section II we briefly review the literatures that are relevant to our work; Section III elaborates our proposed object detection method YOLIC; in Section IV we show the experiments of YOLIC in the practical road risk detection task dataset; and in Section V we draw some conclusions and discuss further challenges for YOLIC.

II. Related Work

This section briefly introduces the related literature, which includes common object detection algorithms and some state-of-the-art researches for on-road risk detection.

A. Object Detection and Segmentation

1) Bounding Box-based Methods: Most methods [3] [4] perform detection tasks by using the bounding box to depict the location of target objects, which also has been well applied in many tasks in recent years [7] [8]. With the rise of deep learning technology, the current mainstream algorithms are mainly divided into regression-based algorithms and region proposal based algorithms. Region proposal based algorithms such as R-CNN [1], Fast R-CNN [9], and Faster R-CNN [10] first extract the potential object region proposal from a given image. Afterward, all potential proposals are further refined and classified into different categories. From this point of view, region proposal-based methods can usually get state-of-the-art results in most object detection benchmark datasets. On the other hand, regression-based algorithms are much simpler and are typically known for their fast detection speed. YOLO integrates region proposal stage and classifier network to complete complex object detection tasks through a single CNN network, making it one of the preferred detection algorithms in engineering projects.

2) Pixel-based Methods: Pixel-based methods play a key role in image scene understanding and Segmentation tasks. These methods also work particularly well under the promotion of deep learning [11]. Exiting pixel-based methods are either regression-based approaches or region proposal based approaches. The most representative model of the proposal-based approach is Mask R-CNN, which has a region proposal network (RPN) that generates object proposals and another network for classification, bounding box regression, and mask segmentation tasks. Other excellent regression-based works include RetinaMask [12], YOLACT [13], CenterMask [14]. These methods generate a feature map to output the segmentation results of different objects. However, the pixel-based methods always involve expensive computation, limiting the application of such algorithms on resource-constrained devices.

B. Recent Progress of On-road Risk Detection

The current mainstream on-road risk detection methods are usually based on regression-based detection methods to achieve high-speed detection performance. There are various sensors applied in the detection tasks, including cameras, stereo vision, LiDAR, and their combinations. Compared with a single camera, multi-modal sensors can more sensitively perceive the external environment and even directly obtain accurate distance information. In the article [15], they use color-camera and LIDAR for on-road risk classification. The results show that RGB-LIDAR models have better performance compared with 3D and range-view models. Depei Zhang Sr. et al. [16] use the YOLO-based method for pedestrian detection. Similarly, some works [17] [18] [19] use Faster-RCNN to handle road-related detection tasks. In our previous work [20], we break the complex object detection problem into the image classification problem of batch images for on-road risk detection.

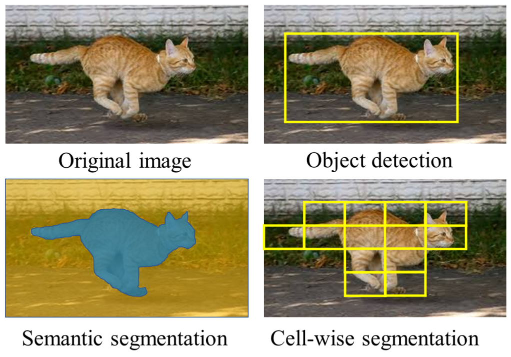

Fig. 2. Comparison for cell-wise segmentation.

III. Methodology

YOLIC only has an interest in specific regions in the images. Unlike the existing expression methods for objects, YOLIC leverages cell-wise segmentation to represent object regions in any given image.

A. Cell-wise Segmentation

Cell-wise segmentation is a relatively new approach for annotating objects in images. The method can be viewed as a task between object detection and semantic segmentation. Cell-wise segmentation does not use a rectangular bounding box to represent the object's position, nor does it separate the object from the background at the pixel level like semantic segmentation methods. Instead, it uses several cells to represent one object. Compared with object detection methods, cell-wise segmentation method can depict each component of the target object, thereby more accurately representing the object's location. In this way, the cell-wise segmentation method is more suitable for objects with uncertain shapes or objects with occluded components. Our method can also more efficiently segments high-resolution images compared with semantic segmentation, and can significantly reduce the complexity of segmentation-based detection methods. Fig. 2 shows different kinds of approaches for locating objects in the image. We can see that using an appropriate number of cells can give a good overview of the object's shape. Compared with a single rectangular bounding box, appropriate cells can flexibly represent the range of a target object and avoid meaningless background areas annotated as a part of the object.

Besides, we can flexibly control the size of the cell according to our actual needs instead of training network to learn the size of cell through regression. If we have prior knowledge about the approximate range, shape or size of the object, we can use a few large cells to depict the objects.

B. YOLIC Design and Training

Taking advantage of cell-wise segmentation, we can divide the image into several Cells Of Interest (COIs) for object detection. Since YOLIC does not perform any bounding box regression (unlike YOLO or SSD) the model requires to prefix the configuration of the COIs before the training phase. Configuration for COIs are defined by number, size and position. In addition, the configuration for the COIs depends completely on the use case. Thus, when prefixing the COI configuration, care should be taken such that each image's COIs are consistent with the other images' COIs. The quality of the configuration of COI will reflect on the performance of the YOLIC. Since we pre-defined the position and size of COIs, YOLIC does not need to predict the coordinates for all COIs. The network only needs to focus on the classification results in each COI. In other words, we break the complex object detection problem into the classification problem with multiple COIs by applying the concept of 'divide and conquer.'

Like regression-based detection methods, YOLIC uses features from the entire image to simultaneously predict each COI. However, multiple classes may appear in the same COI at the same time when using a large size of COI. To solve this problem, we adopted the idea of multi-label classification so that YOLIC can have multiple COI outputs. Each COI can have multiple category outputs. As mentioned earlier, YOLIC can reuse existing classification models and predicts all COIs and their classes. Thus we do not need to add additional layers or delete the existing layers for object detection. We only need to change the output layer's node from C_{class} to:

$$F_o = N_{cell} \times (C_{class} + 1) \tag{1}$$

where N_{cell} is the number of COIs, and C_{class} is the number of classes in the data set. As a result, each COI contains an indicator bit to indicate whether there is an object in that COI.

The final layer contains F_o nodes for predicting each COI and objects in COI. We use Sigmoid function as the activation function for these nodes so that each node can make a decision independently. Therefore, the output of the last layer aims to minimize the loss function depicted in Eq. 2

$$\begin{aligned} J = -\frac{1}{F_o}\sum_i y[i] \cdot \log\left(\sigma\left(x[i]\right)\right) \\ + \left(1 - y[i]\right) \cdot \log\left(1 - \sigma\left(x[i]\right)\right) \end{aligned} \tag{2}$$

TABLE I

THE ACCURACY OF YOLIC EQUIPPED WITH DIFFERENT BACKBONE NETWORKS ON THE ON-ROAD RISK DATASET. WE TRAIN YOLIC WITH DIFFERENT BACKBONE NETWORKS WITH THE SAME HYPER-PARAMETERS. NOTE THAT THE TWO-CLASS ACCURACY IS RISK DETECTION RATE (CLASSIFICATION BETWEEN RISK AND NORMAL)

Backbone	RGB Input		RGBD Input	
	Two-class Accuracy (%)	All-class Accuracy (%)	Two-class Accuracy (%)	All-class Accuracy (%)
ResNet-18	93.94	90.65	95.67	92.93
ResNet-34	95.53	93.42	96.30	94.40
ResNet-50	95.19	92.88	96.01	93.88
ResNet-110	95.60	93.49	96.19	94.26
ResNet-152	95.40	93.21	96.12	94.07
MobileNetV2	95.28	92.66	96.17	93.70
ShuffleNetV2	94.30	91.31	96.04	93.66

where $i \in \{0, 1, 2, \cdots, F_o - 1\}, y[i] \in \{0, 1\}$ and $\sigma(\cdot)$ is Sigmoid activation function.

C. Inference and Evaluation

During the test phase, we can directly obtain the multi-classification results of all COIs by only one network. However, F_o predictions can be fully correct, partially correct, or fully incorrect for one test image. To evaluate the image which has partially correct results, we use the example-based evaluation method [21] to calculate the average difference between the predicted labels and the actual labels for each test image. Once we got the evaluated results for every image, an overall average score can be calculated across all test images.

Let $\{(X_i, Y_i) \mid 1 \leq i \leq n\}$ be a test dataset where X_i is an image, and Y_i is an F_o-element vector. $Y_i[k] \in \{0, 1\}$ represents the correct label of k-th interested cell of X_i. Let h be a YOLIC classifier. $P_i = h(X_i)$ is an F_o-element vector and $P_i[k] \in \{0, 1\}$ represents the predicted label of k-th interested cell of X_i. The detection accuracy is defined by

$$\mathrm{ACC} = \frac{1}{n}\sum_{i=1}^{n}\frac{\sum_{k=1}^{F_o}\{P_i[k]Y_i[k] + (1 - P_i[k])(1 - Y_i[k])\}}{F_o} \tag{3}$$

To show the comparative performance for each class the recognition rate for j-th class can be defined by

$$R_j = \frac{\sum_{i=1}^{n_j} TP_{X_{ij}}}{\sum_{i=1}^{n_j}(TP_{X_{ij}} + FN_{X_{ij}})} \tag{4}$$

where n_j is the number of j-th class in the test dataset. $TP_{X_{ij}}$ and $FN_{X_{ij}}$ are the True Positive and False Negative of j-th class in X_i image respectively.

IV. ON-ROAD RISK DETECTION

A capable risk detection system generally demands to detect obstacles accurately and provides drivers with some early operations to avoid danger. This requires the detection system to have strict requirements for detecting the relative distance between obstacles and vehicles. From this point of view, YOLIC is a useful application for road risk detection.

A. Definition of COI

In this study, we use a mobility scooter as our experimental vehicle. The mobility scooter is a typical representative of low-cost vehicles and is widely used by older people. Due to the elderly drivers' physical condition, this is very necessary to detect potential road risks and give a warning to the driver in advance. Our experimental scooter's maximum speed is about 1.7 m/s, and the longest braking distance is about one-meter. Considering the view from the mobility scooter, we regard the road within 4 meters from the mobility scooter as the detection area. As shown in Fig. 3, the first row (i.e., cell 1 to 6) is the notice area. The last two rows (i.e., cell 7 to 22) are the danger area. The white tapes on the floor are the reference lines. We set 22 COIs in the field of vision and detect risks only for these COIs.

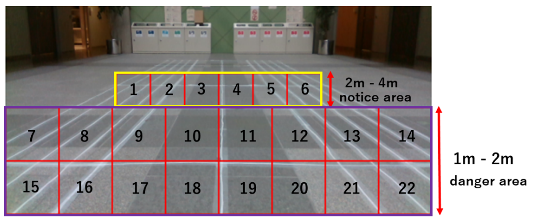

Fig. 3. 22 COIs are used for the risk detection task.

B. Image Collection and Annotation

In our on-road detection system, an Intel Real-sense depth camera D435 is used and mounted on our mobility scooter. The depth camera D435 combined the color camera module and depth module, which can reliably generate accurate distance information for fast-moving applications. With the help of the depth camera, we can obtain spatial relationships between objects through depth information. Indeed, we can further integrate the color camera (RGB image) and depth camera (distance information), using RGBD on-road data to handle the road risk detection problem. We have collected more than 300 minutes of video outdoors and carefully screened all frames base on multiple factors such as scene complexity, obstacle types, light factors, and seasonal changes. Finally, we picked 5,774 frames as our experiment data. It is worth noting that this new data set is different from the data collected in our earlier work [20]. Our data set contains different scenes, and each selected frame has apparent differences. We have also retained the corresponding depth frames of all selected frames further to verify the performance of YOLIC with RGBD input data.

TABLE II

PERFORMANCE OF YOLIC FOR THE ON-ROAD RISK DATASET. THE TABLE DEPICTS THE PERFORMANCE OF OUR PROPOSED FRAMEWORK (YOLIC) EQUIPPED WITH DIFFERENT BACKBONE NETWORKS. THE TYPE-I SECTION DEPICTS THE SCORE FOR RELATIVELY LARGER NETWORKS. THE TYPE-II ARE THE NETWORKS THAT ARE DESIGNED FOR SMARTPHONES OR OTHER LOW-COST DEVICES.

Type	Method (conf_thresh = 0.50)	Recognition rate (%)									TP	FP	FN	Precision	Recall	F1 score
		Bump	Column	Dent	Fence	Creature	Vehicle	Wall	Weed	Cone						
Type-I	YOLOv3 (IOU=0.5)	**87.45**	**59.97**	**80.54**	90.96	**91.70**	**84.99**	**89.52**	**90.61**	**74.21**	**9829**	**1064**	**1518**	**0.9023**	**0.8662**	**0.8839**
	YOLOv3 (IOU=0.8)	**87.45**	**59.97**	**80.54**	**92.96**	**91.70**	84.48	**89.52**	**90.61**	72.40	9823	1070	1524	0.9018	0.8657	0.8834
	YOLOv3 (IOU=0.9)	85.99	56.79	77.76	87.42	89.06	74.05	88.01	89.39	56.56	9532	1361	1815	0.8751	0.8400	0.8572
	YOLOv3 (IOU=0.95)	67.11	31.75	51.51	49.47	60.43	33.84	62.77	68.93	26.24	6785	4108	4562	0.6229	0.5980	0.6102
	RGB YOLIC (ResNet-18)	85.19	34.04	74.08	83.58	86.90	72.77	85.66	86.40	46.15	9155	1361	2192	0.8706	0.8068	0.8375
	RGB YOLIC (ResNet-34)	88.73	56.09	82.91	**88.27**	91.16	84.73	90.54	88.78	63.35	9805	1260	1542	**0.8861**	0.8641	0.8750
	RGB YOLIC (ResNet-50)	86.92	52.56	79.97	84.65	91.02	84.22	88.49	88.38	71.04	9642	**1257**	1705	0.8847	0.8497	0.8669
	RGB YOLIC (ResNet-101)	89.24	**59.26**	82.58	87.63	**92.17**	81.93	**90.90**	**89.95**	**72.40**	**9886**	1277	**1461**	0.8856	**0.8712**	**0.8784**
	RGB YOLIC (ResNet-152)	**89.44**	56.61	**83.89**	83.16	91.63	**86.51**	90.06	89.04	66.52	9838	1351	1509	0.8793	0.8670	0.8731
	RGBD YOLIC (ResNet-18)	89.89	46.03	83.81	84.65	89.40	84.48	90.72	86.75	51.13	9690	1303	1657	0.8815	0.8540	0.8675
	RGBD YOLIC (ResNet-34)	**90.49**	62.96	86.26	88.27	90.41	85.75	**93.13**	**90.46**	61.09	**10008**	1156	**1339**	0.8965	**0.8820**	**0.8892**
	RGBD YOLIC (ResNet-50)	88.43	62.61	84.96	88.70	90.55	85.75	91.15	89.19	60.18	9865	1195	1482	0.8920	0.8694	0.8805
	RGBD YOLIC (ResNet-101)	88.97	**65.08**	85.28	**91.47**	**91.22**	**87.79**	91.69	89.85	62.90	9960	**1147**	1387	**0.8967**	0.8777	0.8871
	RGBD YOLIC (ResNet-152)	89.71	60.67	**86.43**	**91.47**	89.47	86.01	89.4	89.70	**64.25**	9903	1164	1444	0.8948	0.8727	0.8836
Type-II	YOLOV3-Tiny (IOU=0.5)	**84.21**	**54.50**	**70.89**	**85.93**	**82.85**	**77.86**	**83.49**	**83.60**	**66.06**	**9123**	**1444**	**2224**	**0.8633**	**0.8040**	**0.8326**
	YOLOV3-Tiny (IOU=0.8)	82.13	51.15	65.00	84.01	76.17	74.05	82.23	81.57	61.54	8768	1799	2579	0.8298	0.7727	0.8002
	YOLOV3-Tiny (IOU=0.9)	66.58	36.86	48.57	61.83	49.43	49.11	64.94	67.67	42.53	6762	3805	4585	0.6399	0.5959	0.6171
	YOLOV3-Tiny (IOU=0.95)	30.93	13.93	19.95	22.18	19.31	14.50	30.72	32.44	12.67	2987	7580	8360	0.2827	0.2632	0.2726
	RGB YOLIC (MobileNetV2)	**85.52**	**51.85**	**82.91**	84.86	88.32	**83.21**	**88.07**	**88.17**	**52.04**	**9541**	**1288**	**1806**	**0.8811**	**0.8408**	**0.8605**
	RGB YOLIC (ShuffleNetV2)	83.41	43.03	79.07	**86.78**	**87.58**	78.12	86.08	86.40	44.80	9257	1561	2090	0.8557	0.8158	0.8353
	RGBD YOLIC (MobileNetV2)	87.96	**60.14**	85.85	**92.75**	88.72	87.02	88.37	**87.72**	56.56	9760	**1196**	1587	**0.8908**	0.8601	0.8752
	RGBD YOLIC (ShuffleNetV2)	**90.57**	59.08	**87.41**	91.26	**92.24**	**87.28**	**89.22**	87.16	**59.73**	**9917**	1361	**1430**	0.8793	**0.8740**	**0.8766**

Unlike existing object detection or segmentation methods, we use the cell-wise segmentation for representing objects on the on-road image. The existing image labeling tools cannot be used to obtain road annotation based on our segmentation method. Therefore, we have developed a cell-based annotation tool that can simultaneously annotate RGB images and Depth images and generate corresponding annotation files for different detection algorithms. This study defines ten categories for this preliminary on-road risk detection task: traffic cone, weed, vehicle, creature, fence, dent, fence, column, bump, and normal. Other undefined on-road risks will be considered based on their shape and purpose. In our subsequent experiments, we fixed 70% of labeled images for the training and the remaining 30% for the testing.

C. Experiments on On-road Risk Detection

We implement YOLIC in the PyTorch framework and conduct all the experiments on a PC with an Intel Core i7-8700K processor, 64 GB Memory, and one NVIDIA GeForce GTX 1080 GPU. The image was set at 224×224 pixels for all subsequent experiments. To prove that YOLIC is convenient to use, existing well-know models are used as the YOLIC backbone network to perform on-road risk detection. All YOLIC detection models are trained with an initial learning rate of 0.01, mini-batch size of 8, and terminate training at 300 iterations. Besides, both RGB input and RGBD input models are trained for further comparison. Table I presents the performance of YOLIC with different backbone networks. As an evaluation index, we used the accuracy defined in Section III.C.(3). In all-class accuracy, we treat ten categories of risks, including normal. That is, $F_o = 22 \times 10 = 220$. On the other hand, for two-class accuracy evaluation, we treat only two categories i.e., either risk or normal. That is $F_o = 22 \times 2 = 44$. The YOLIC detection models with RGBD input have better results in both two-class accuracy and all-class accuracy.

To examine the performance differences between YOLIC and state-of-the-art detection algorithms, we compare the YOLIC models with YOLO, which is one of the recognized and representative methods. For a detailed comparison, we provide Table II and Table III. The first table depicts the performance in accuracy, and the second table represents model comparison results in speed. In Table II, we divide the all model into two types, where Type-I represents the network with a larger number of parameters, Type-II networks are the lightweight network, which usually required less computational resources and used for the mobile device. We can observe form results that the RGBD YOLIC detection model with ResNet-34 backbone elevates the performance and get the best F1 score (0.8892) in the Type-I networks. Note that YOLO uses Intersection over Union (IoU) to evaluate the position of the bounding box. For a fair comparison, we should evaluate YOLO at the IoU threshold close to one. However, we can find that when the IoU threshold is increased, the performance of both YOLOv3 and YOLOV3-Tiny is also significantly reduced (F1 score from 0.8839 drops to 0.6102). In other words, YOLO cannot predict the location of COIs accurately when using the general IoU threshold 0.5. Although we used lightweight networks as the YOLIC backbone in the Type-II networks, YOLIC still can obtain satisfactory F1 score (0.8766) comparable to the cumbersome Type-I networks and higher than the YOLOV3-Tiny network. In the Table III, we conduct the theoretical analysis for all relevant models and detail run-time statistics on a GPU machine, a signal board computer (Raspberry Pi 4) with NCS-2, and a Laptop. These experimental results show that using lightweight networks as backbone networks can achieve real-time detection performance on three platforms (i.e., GPU, CPU and Raspberry Pi edge device). Also, from Table II and III, YOLIC models are faster than YOLO with the same level of accuracy under the different platforms.

TABLE III
SPEED COMPARISON RESULTS FOR THREE DIFFERENT COMPUTING PLATFORM.

Type	model	Params (M)	FLOPs (G)	Desktop with NVIDIA GTX 1080		Raspberry Pi 4 with Intel NCS 2		Laptop with Intel-i5 5200U	
				Inference time (ms)	FPS	Inference time (ms)	FPS	Inference time (ms)	FPS
Type-I	YOLO v3	65.25	18.95	23.25	**43.01**	105.57	9.47	217.87	4.59
	RGB YOLIC (ResNet-18)	11.29	3.64	3.10	**322.58**	24.65	**40.57**	42.52	23.52
	RGB YOLIC (ResNet-34)	21.4	7.34	4.68	**213.68**	41.35	24.18	83.38	11.99
	RGB YOLIC (ResNet-50)	23.96	8.21	6.26	**159.74**	60.23	16.60	89.36	11.19
	RGB YOLIC (ResNet-101)	42.95	15.66	11.23	**89.05**	104.23	9.59	161.21	6.20
	RGB YOLIC (ResNet-152)	58.59	23.11	16.01	**62.46**	145.02	6.90	243.10	4.11
	RGBD YOLIC (ResNet-18)	11.29	3.72	3.34	**299.40**	26.60	**37.59**	61.39	16.29
	RGBD YOLIC (ResNet-34)	21.4	7.42	4.93	**202.84**	43.84	22.81	98.78	10.12
	RGBD YOLIC (ResNet-50)	23.96	8.29	6.40	**156.25**	62.55	15.99	115.42	8.66
	RGBD YOLIC (ResNet-101)	42.95	15.74	11.49	**87.03**	105.31	9.50	187.84	5.32
	RGBD YOLIC (ResNet-152)	58.6	23.19	16.43	**60.86**	148.79	6.72	265.17	3.77
Type-II	YOLOV3-Tiny	8.66	1.583	5.34	**187.10**	16.78	**59.59**	30.98	**32.28**
	RGB YOLIC (MobileNetV2)	2.51	0.612	6.05	**165.29**	24.79	**40.34**	9.94	**100.60**
	RGB YOLIC (ShuffleNetV2)	1.48	0.292	6.94	**144.09**	15.77	**63.41**	9.14	**109.41**
	RGBD YOLIC (MobileNetV2)	2.51	0.620	6.24	**160.26**	26.78	**37.34**	12.40	**80.65**
	RGBD YOLIC (ShuffleNetV2)	1.48	0.297	7.05	**141.84**	17.67	**56.59**	10.54	**94.88**

V. CONCLUSION

Tiny AI has attracted attention as a critical technology recently for edge computing. Many researchers have started working on making AI algorithms more efficient and compact. With such motivation in mind, in this study, we have proposed a novel object detection method termed as You Only Look at Interested Cell or in-short YOLIC. The proposed model can achieve faster real-time object detection based on the cell-wise segmentation. Unlike existing approaches, YOLIC only focuses on the interested cells to simplify the detection process. Even non-GPU devices can handle the complex object detection task with high performance in real-time.

ACKNOWLEDGMENT

The authors would like to thank our partner company for providing mobility scooters and other related equipment for conducting the experiments. We would like to thank Prof. Yong Liu and Dr. Yinghui Zhou for their constructive comments.

REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[2] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4507–4515.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, "A robust real-time automatic license plate recognition based on the yolo detector," in *2018 international joint conference on neural networks (ijcnn)*. IEEE, 2018, pp. 1–10.

[8] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang, "Apple detection during different growth stages in orchards using the improved yolo-v3 model," *Computers and electronics in agriculture*, vol. 157, pp. 417–426, 2019.

[9] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[11] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[12] C.-Y. Fu, M. Shvets, and A. C. Berg, "Retinamask: Learning to predict masks improves state-of-the-art single-shot detection for free," *arXiv preprint arXiv:1901.03353*, 2019.

[13] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2019, pp. 9157–9166.

[14] Y. Lee and J. Park, "Centermask: Real-time anchor-free instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 906–13 915.

[15] C. Premebida, G. Melotti, and A. Asvadi, "Rgb-d object classification for autonomous driving perception," in *RGB-D Image Analysis and Processing*. Springer, 2019, pp. 377–395.

[16] D. Zhang Sr, Y. Shao, Y. Mei, H. Chu, X. Zhang, H. Zhan, and Y. Rao, "Using yolo-based pedestrian detection for monitoring uav," in *Tenth International Conference on Graphics and Image Processing (ICGIP 2018)*, vol. 11069. International Society for Optics and Photonics, 2019, p. 110693Y.

[17] T. Wang, X. Zhang, L. Yuan, and J. Feng, "Few-shot adaptive faster r-cnn," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7173–7182.

[18] M. Nie and K. Wang, "Pavement distress detection based on transfer learning," in *2018 5th International Conference on Systems and Informatics (ICSAI)*. IEEE, 2018, pp. 435–439.

[19] T. Yang, X. Long, A. K. Sangaiah, Z. Zheng, and C. Tong, "Deep detection network for real-life traffic sign in vehicular networks," *Computer Networks*, vol. 136, pp. 95–104, 2018.

[20] K. Su, C. M. Intisar, Q. Zhao, and Y. Tomioka, "Knowledge distillation for real-time on-road risk detection," in *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE, 2020, pp. 110–117.

[21] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2004, pp. 22–30.