ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DEEP LEARNING-DRIVEN CAR PARKING SPACE DETECTION: A YOLOR APPROACH

SHARFINA FAZA[1], ROMI FADILLAH RAHMAT[2*], RINA ANUGRAHWATY [3], JESSICA WONG[2], FARHAD NADI[4]

[1] Department of Computer Engineering and Informatics, Politeknik Negeri Medan, Indonesia

[2] Department of Information Technology, Universitas Sumatera Utara, Indonesia

[3] Department of Electrical Engineering, Politeknik Negeri Medan, Indonesia

[4] School of Information Technology, UNITAR International University, Malaysia

*Corresponding E-mail: romi.fadillah@usu.ac.id

ABSTRACT

Parking is a critical facility that must be widely accessible, especially in public places such as tourist attractions, shopping centers, and offices. However, people are often delayed due to difficulty finding empty parking spaces, particularly during peak hours or when there are many visitors. This is partly because visitors do not know the exact location of available slots and must repeatedly circle the area to find one. Therefore, there is a pressing need to develop more efficient methods for detecting parking space availability to expedite and simplify the search process. To address this challenge, our study employs the You Only Learn One Representation (YOLOR) method to help detect available parking slots in three distinct locations: the underground parking area at 'Centre Point' shopping mall in Medan City, the university library parking lot, and the campus parking facility at the University of Sumatera Utara. YOLOR is one of the deep learning methods that has shown promising results in object detection tasks, making it a suitable choice for this parking slot identification problem. Based on our system's tests, we achieved accuracies of 95.71%, 95.74%, and 83.33% for each parking lot, respectively. For the occupied class, we obtained precisions of 0.99 and 0.97 in each parking lot, while for the empty class, we consistently achieved a precision of 1.00 across all lots. Recall rates were 0.96, 0.95, and 0.92 for the occupied class, and 0.95, 0.96, and 1.00 for the empty class in each parking lot. Our model was trained on a total of 3,180 images, comprising 11,666 empty slots and 11,843 occupied slots. Testing was conducted using six 30-minute to 1-hour long videos, each capturing various parking conditions under different weather (sunny, cloudy, or rainy) and lighting situations.

Keywords: *Parking, Parking Spaces, Detection, YOLOR, Deep Learning*

1. INTRODUCTION

Parking facilities in public spaces such as tourist attractions, shopping centers, offices, and other venues play a crucial role in managing vehicle traffic and providing convenience for visitors or service users [1]. The availability of adequate parking spaces is essential to avoid congestion and confusion when searching for parking. However, it is not uncommon for many customers to frequently circle around looking for parking slots, especially during peak hours [2]. This becomes a common problem that can increase traffic density around parking areas, potentially negatively impacting the comfort and safety of other users.

According to a 2017 survey by an online transportation service provider, the difficulty in finding parking is the most frustrating aspect for car owners while driving. Out of 1,000 respondents of various ages, 60% agreed that finding parking in certain places is challenging [3]. Another survey found that 74% admitted to having trouble finding parking and experiencing traffic jams while driving, which is the main reason for frequently being late to their destinations [4].

These survey results indicate that difficulties in finding parking spots can lead to delays, wasted time, and increased carbon emissions due to prolonged vehicle use. To address this issue and reduce the time wasted in searching for parking slots, a system is needed that can identify the availability of parking spaces and provide information to visitors. This way, visitors can easily locate empty parking spots and directly proceed to them without having to search manually.

Several studies have been conducted to solve the problem of finding available parking slots. Among them is research to assist in monitoring smart and efficient parking systems by implementing SSD-MobileNet using TensorFlow Lite on edge artificial intelligence in IoT devices to monitor the use of existing parking spaces and achieved an accuracy of 95.6% in real-world scenarios or direct field testing. [5]. Another study identified parking space usage using the Mixture-of-Gaussians (MoG) background model on parking space images captured by cameras installed in two parking lots and achieved an accuracy rate of over 93%. [6]. Furthermore, there is research conducted to determine parking space availability by counting the number of cars in a parking area using YOLOv4 and comparing it with the total number of parking spaces provided and achieved an accuracy of 72.8% from testing results on several videos. [7]. Additionally, a study attempted to use two methods: CNN with a Haar cascade classifier to identify multiple vehicles in a single image and video, and pre-trained Mask R-CNN to identify various vehicles and empty parking spaces. CNN with haar cascade classifier achieved an accuracy of 99.80% [8] and also another machine learning approach such as Probabilistic Neural Network and achieved an accuracy rate of over 94% [9].

Based on the differences in research that have been explained, it can be concluded that the overall differences between this study and previous studies generally lie in the methodology to be used, and there is also one difference in the object to be identified. Furthermore, this research is expected to provide better results than previous studies by covering a larger dataset using Deep Learning methods, specifically the YOLOR algorithm implementation. It is anticipated that this approach will be more robust in handling varying lighting conditions in parking areas.

YOLOR's unique architecture and training methodology make it particularly well-suited for this task [10]. By learning a unified network representation, YOLOR can effectively capture both low-level features like edges and textures, and high-level semantic information such as vehicle shapes and parking slot layouts. This comprehensive understanding enables YOLOR to accurately distinguish between occupied and empty parking spaces, even in challenging scenarios like low light, glare, or partial occlusions.

Moreover, YOLOR's design prioritizes computational efficiency without sacrificing accuracy [11]. This aligns perfectly with our need for a real-time parking space detection system that can process video feeds instantly, providing drivers with up-to-the-second information about available slots. By adopting YOLOR, we aim to set a new standard in the field, offering a deep learning-driven solution that is not only highly accurate but also fast enough to handle the dynamic nature of parking space.

This research is crucial due to the persistent challenges in parking space detection that impact urban traffic management and smart parking systems. Despite previous efforts, issues of speed and accuracy in real-world conditions remain unresolved. The significance of this study lies in its potential to address these problems and revolutionize parking management systems. The importance of this research is highlighted by the growing urbanization and increasing number of vehicles in cities worldwide. Inefficient parking systems contribute to urban congestion, wasted fuel, and increased carbon emissions. Improving the accuracy and speed of parking space detection could dramatically reduce this wasted time, leading to substantial economic and environmental benefits.

2. METHODOLOGY

This research focuses on three distinct parking locations: the underground parking area at 'Centre Point' shopping mall in Medan City, the university library parking area, and the campus parking facility at the University of Sumatera Utara. These high-traffic locations were selected for their spacious parking areas featuring two lanes, ensuring that each captured image could encompass numerous car parking slots. The collected data will be divided into two sets: one for training and another for testing purposes.

For the training data, after the images are captured, they will undergo a resizing process to ensure uniformity. Each parking space in these images will be manually labeled as either "empty" or "occupied," providing clear classifications for the machine learning model to learn from. Next, we will configure the appropriate parameters to be implemented in our chosen Deep Learning method. The algorithm we are employing is You Only Learn One Representation (YOLOR), a state-of-the-art technique in object detection that generates bounding boxes and confidence levels for identified objects.

For the testing data, we will directly apply the trained YOLOR algorithm without any additional preprocessing. This real-world testing phase will validate our model's ability to generalize from its

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

training. The output will be a Learned Model that can accurately classify each car parking space in the test images as either "empty" or "occupied".

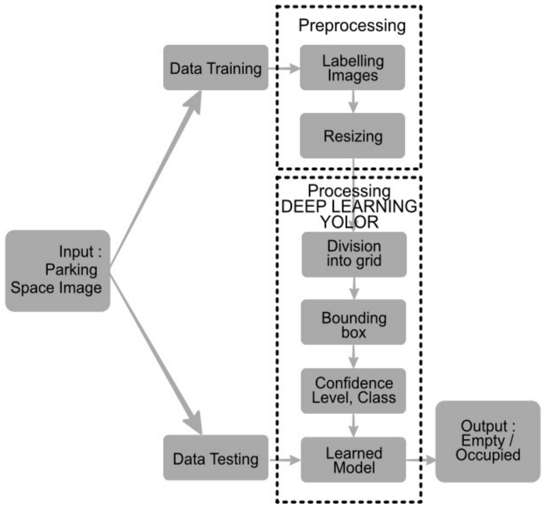


Figure 1 : General Architecture

Experiments will be conducted to ensure that the system functions properly in accordance with the analysis and design. System testing in this study will utilize a Confusion Matrix, examining Actual and Predicted values to determine Precision, Recall, F1 score, and Accuracy. This comprehensive evaluation approach will allow for a thorough assessment of the model's performance, enabling comparison with existing methods and validation of the proposed approach's effectiveness in enhancing parking space detection accuracy and efficiency.

This methodology aims to leverage the strengths of the YOLOR algorithm in real-world parking scenarios, addressing the challenges of varying environmental conditions and the need for real-time processing. By using diverse and representative data from high-traffic areas, the study seeks to develop a robust model capable of accurate parking space detection across different settings, potentially setting a new standard for efficient, user-friendly, and environmentally conscious urban parking solutions.

2.1 Dataset

The dataset used in this study is divided into training and testing data. For the training process, the data consists of valid and train data obtained by capturing videos from three distinct parking lots: the campus parking area, the Library parking area, and the underground parking at the Center Point mall. Data collection was conducted over 10 days at different times and under various weather conditions, including sunny, cloudy, and rainy weather. The data was captured using a 16 MP smartphone camera in mp4 video format with a resolution of 1920x1080 pixels at 13fps. These videos were then broken down into 1 or 2 frames per second to create individual images in jpg format.

To enhance data diversity and improve model robustness, the extracted frame images underwent an augmentation process. This involved applying horizontal and vertical flips, as well as adjusting saturation levels from -30% to +30%. These augmentations help the model learn to recognize parking slots under a wide range of visual conditions.

In the campus parking area, videos were taken from the 2nd and 3rd floors of building, providing an elevated perspective. For the Library parking area, videos were captured from approximately 3.5 meters high using a tripod, ensuring a clear view of the parking area. At the Center Point mall's underground parking, video recording was done from the top floor with the aid of a 2.1-meter-high tripod, offering an expansive view of the lot.

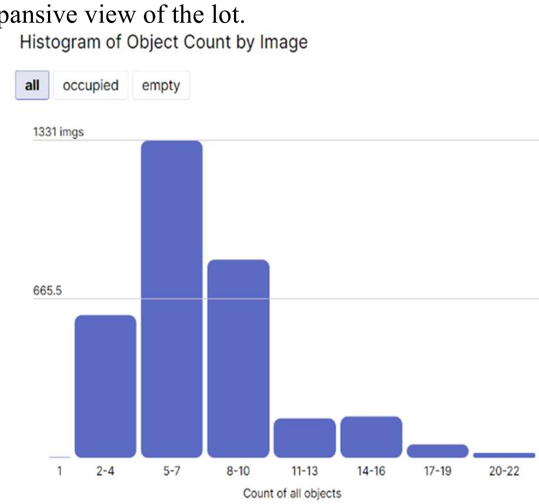


Figure 2 : The Distribution of Parking Slots in Each Image

Approximately 70 videos were collected, each ranging from 30 seconds to 1 minute in duration. After extracting frames and applying augmentations, this yielded a total of 3,180 images. Each image contains between 4 to 20 parking slots, resulting in a comprehensive dataset of 11,666 empty slots and 11,843 occupied slots. This balanced distribution ensures that the model learns to distinguish between empty and filled slots effectively. The distribution of slots in each image is visually represented in Figure 2.

The data that has been collected is then divided into 84% training data and 16% validation data. An example of the Campus parking area that will be used can be seen in Figure 3, then Library parking area can be seen in Figure 4, and an example of Underground mall parking area is shown in Figure 5.

Journal of Theoretical and Applied Information Technology
31st August 2024. Vol.102. No. 16

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

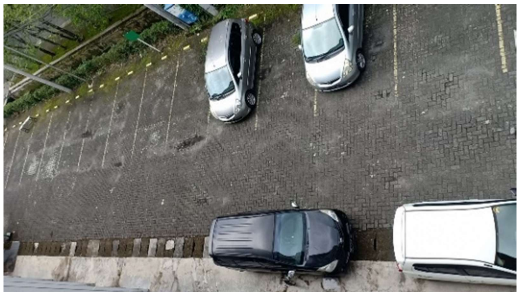


Figure 3 : The Campus Parking Area


Figure 4 : The Library Parking Area

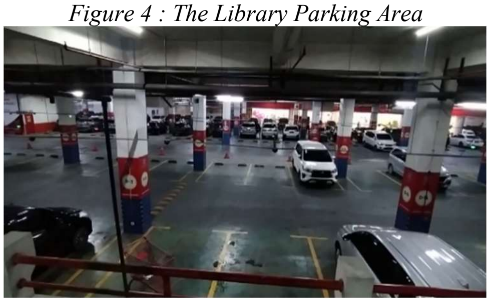


Figure 5 : The Underground Mall Parking Area

As for testing, the data was not taken at the same time as the training data. For the campus building, the camera was placed on the 3rd floor for 1 hour during the day and 1 hour in the afternoon within 2 days. The parking area for the library was also tested in the same way by placing a camera for 1 hour in the morning and evening for 2 days. Then, data testing on the underground mall parking area was done for only 1 hour in one day from the top or from the second floor of the parking area by facing the camera toward the parking area below. The testing data taken attempted to cover various parking area conditions and lighting conditions as well as different weather conditions, sunny and rainy.

2.2 Preprocessing

In the initial stage, image data will be labeled using the Roboflow labeling tools to provide labels such as available and occupied parking with the names "empty" and "occupied." The labeling results will be organized into a folder with subfolders for images and labels. The generated labels will be in .txt format and utilized in the training process. Next, the resizing process. The goal of resizing in preprocessing data for deep learning is to ensure that all input data (images, text, etc.) are of a uniform size or dimension. Resizing helps in standardizing the input data, making it easier for the model to learn patterns and features effectively [12]. It also reduces computational complexity by ensuring consistent input sizes across the dataset, leading to better model performance and faster training times [13]. In this process, the parking space image size to be processed is adjusted to a size suitable for the model's use and to simplify the image, speeding up the training process. Images captured using a smartphone at 1920x1080 are resized to 640x480. YOLOR supports input images with various sizes but will resize all images to 640x480 to ensure uniformity for training. The resizing process in YOLOR utilizes the img-size parameter to specify the desired size and employs the OpenCV library to read and resize images accordingly. Figure 6 shows an example of the image data before resizing, while Figure 7 displays the image after resizing to 640x480.


Figure 6 : Example of an Image Before Resizing

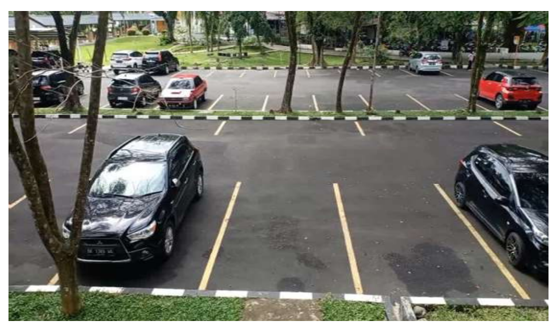


Figure 7: Example of an image after being resized to 640x480

2.3 YOLOR

After completing the preprocessing stage, the image is processed using YOLOR. The YOLOR algorithm was created by Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao, introduced in 2021 through a journal titled "You Only Learn One Representation: Unified Network for Multiple Tasks." YOLOR is a unified network built to integrate implicit and explicit knowledge similar to the human brain,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

allowing the learned model to contain a common representation that can be used to solve various types of tasks, such as object detection [14].

In the general definition of a neural network, explicit knowledge refers to features obtained from shallow layers and directly related to what is observed, while implicit knowledge refers to features obtained from deep layers and not directly related to the observed phenomenon [14]. The concept of a unified network that combines implicit and explicit knowledge into a single representation for use in multiple tasks can be seen in Figure 8.

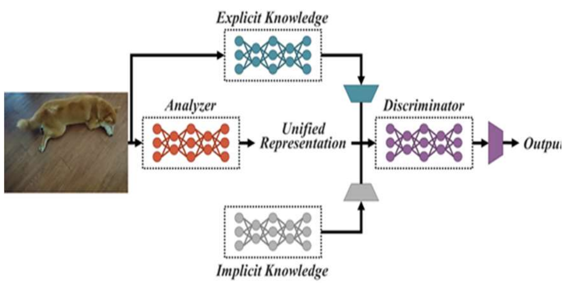


Figure 8 Unified network YOLOR [14]

YOLOR leverages a single neural network to perform multiple tasks such as feature alignment, prediction refinement, and multi-task learning. Through multi-task learning, tasks like object identification, multi-label image classification, and feature embedding are executed, where features are extracted and classified based on their attributes. Subsequently, with prediction refinement, the model is enhanced by conducting corrections using a loss function [14].

2.3.1. Training Model

In the training process, YOLOR retrieves data labels from a .txt file in the format of class ID (0 or 1), followed by the x and y coordinates of the label, and the width and height of the label. The label format is as follows:

<object-class-id> <x> <y> <width> <height>

For this training process, previously collected and labeled datasets will be utilized. Once the labeled training data is inputted, the next step is to set the epoch, batch size, model configuration, and other necessary parameters. Several parameters that need to be defined before commencing the training process are:

- *img*: specifying the size of the input image for training
- *batch*: determining the batch size for training, e.g., 8, 16, 24, 32, etc.
- *epochs*: setting the training epochs, indicating the number of training iterations, e.g., 100, 150, 200, etc.
- *data*: defining the path to the YAML file containing dataset definitions
- *cfg*: file containing the model configuration
- *weights*: specifying the custom path to the weights used as a base for training; initially, the provided pretrained weights from YOLOR developers will be used
- *name*: indicating the folder name to store the training results
- *hyp*: setting the hyperparameters for the training process

In addition, there are also several configurations that need to be determined before the training process is executed, namely filters and classes that are adjusted to the object class to be detected. In this study, 2 classes are used, so the classes for each YOLO layer in the configuration will be changed to 2, and the filters will be adjusted according to equation (1) to obtain 21 filters.

The batches that will be used for the training process will be tested starting from batch 8 and 16, and then the epochs that will be tested are 100, 200, up to 300 to determine which combination of epochs and batches yields better results. Furthermore, the data used will be defined in a yaml file that specifies the location of the training data, validation data, and the classes used.

Moreover, during the training process execution, each training data image will be processed one by one through the YOLOR network. According to Wang, Yeh & Liao [14] and Diniz et al. [15], the YOLOR network architecture comprises Backbone, Neck, and Head as depicted in Figure 8. In the backbone section, feature extraction will be performed on the input data, where the data image will be divided into sxs-sized grids.

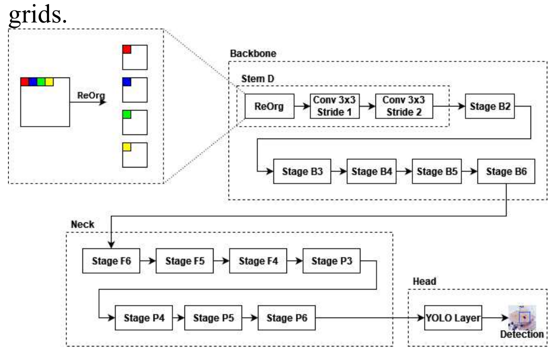


Figure 9: YOLOR network architecture [15]

Subsequently, in the neck section, information will be extracted from various layers of the backbone model for each grid to predict bounding boxes. The confidence value for each bounding box will be calculated, and the class probability value will be

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

predicted if objects are present. This extracted information will be passed to the head. The head section will generate or form bounding boxes and classify them based on the features passed from the neck.

2.3.2. Learned Model

The outcome of the training process is a trained model saved in .pt format since the training is conducted using the PyTorch library. Throughout the training process, YOLOR will create multiple models that are categorized as last and best models. Subsequently, the model file produced after the training phase will be utilized and loaded during the testing stage.

2.3.3. Testing Process

Once the training process is finalized, the trained model produced during training will undergo testing using parking data. This crucial step aims to evaluate the system's ability, previously trained, to distinguish between vacant and occupied parking spaces. Subsequently, the accuracy of the testing results obtained through the YOLOR method will be computed to assess the performance of the resultant model effectively.

2.3.4. Output

Through object detection utilizing YOLOR, the outcome will manifest as identified empty and occupied parking slots, exhibiting a bounding box for each detected object alongside its confidence level and class. The information derived from parking slot availability detection will be showcased on the website, enabling users to visualize the status of each parking slot, distinguishing between empty and occupied spaces effectively.

3. TESTING AND RESULT

3.1. Training Model

The model training process is conducted utilizing the Google Colab platform, which offers NVIDIA GPU runtime supporting CUDA for free, with a maximum runtime of 12 hours per day. In this study, training used the YOLOR repository provided by the developer, accessed on GitHub by cloning the repository at https://github.com/WongKinYiu/yolor.git. Before executing the training process, several configurations need to be set, such as adjusting the filters and classes on each YOLO layer in the configuration file to match the data, specifically setting filters=21 and classes=2, and configuring the data file to point to the test data folder and validation set. The YOLOR algorithm developers have supplied pretrained weights trained on the MSCOCO dataset with 80 classes, serving as the base model for training on custom datasets, including the parking dataset in this study.

To achieve the best accuracy, the training process involves experimenting with different combinations of batches and epochs. Initially, batch size 8 with 300 epochs is trained incrementally per epoch, followed by batch size 16 with 300 epochs and grayscale. The training progress on Google Colab is illustrated in Figure 9.

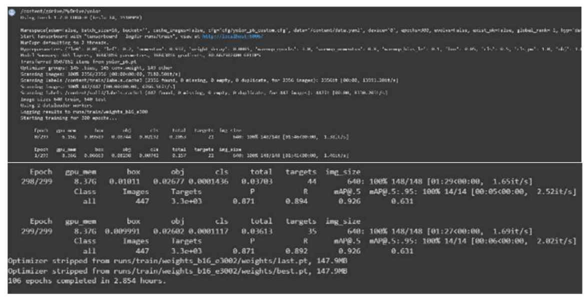


Figure 9: Training Process on Google Colab

Based on the training process conducted, the training results for batch 16 and 300 epochs took approximately 8 hours and achieved a final precision of 0.87. The precision value continued to increase throughout the training process for each epoch. Additionally, a final recall value of 0.89 was obtained, as depicted in Figure 10.

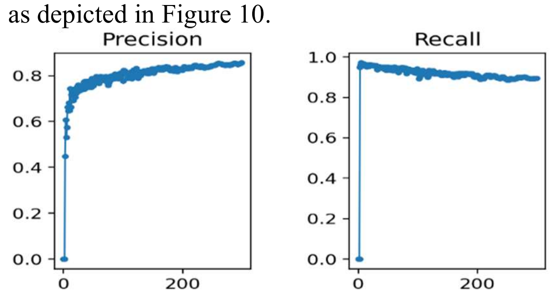


Figure 10: Precision and Recall Graph of Training Results Using Batch 16, Epoch 300

The mAP IoU of 0.5 achieved a final value of 0.92, while the mAP IoU of 0.5-0.9 reached a final value of 0.63, indicating a promising outcome.

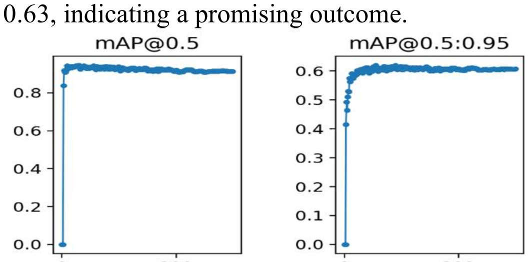


Figure 11: Graph of mAP for Training Batch 16, Epoch 300

Journal of Theoretical and Applied Information Technology
31st August 2024. Vol.102. No. 16

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 11 displays a graph illustrating the mAP values obtained for each epoch during the training process, covering epochs 0 to 299.

The precision-recall curve from the training process with batches of 16 and 300 epochs is depicted in Figure 12, illustrating the tradeoff between precision and recall at various thresholds. The mAP IoU value of 0.5 for all classes is 0.914.

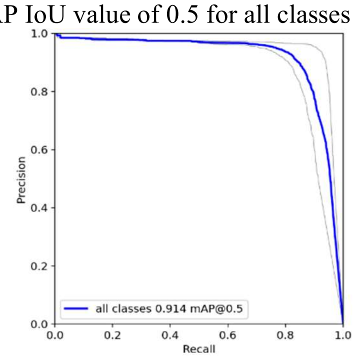


Figure 12: Precision-Recall Curve for Batch 16, Epoch 300

During the training process with 8 batches and 300 epochs, which lasted around 9 hours, a final precision of 0.86 and a recall of 0.9 were achieved. The graph displaying the precision and recall values obtained from each epoch is illustrated in Figure 13.

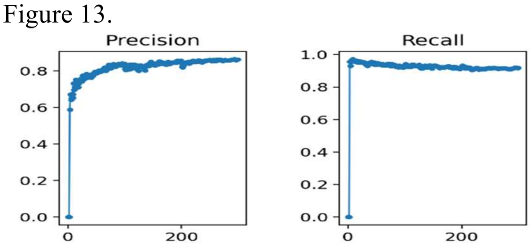


Figure 13: Precision and Recall Graph of Training Results Using Batch 8, Epoch 300

For mAP with IOU 0.5, a final score of 0.93 was achieved, while for mAP IOU 0.5-0.95, a final value of 0.62 was obtained, showing a slight decrease compared to batch 16 but still indicating satisfactory results. The graph displaying each mAP value obtained during the training process is presented in Figure 14.

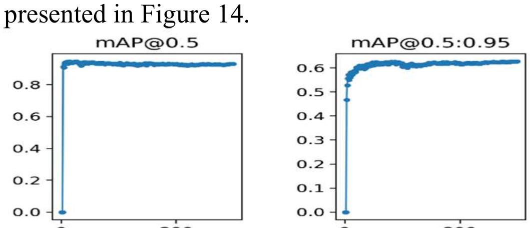


Figure 14: Graph of mAP from Batch 8, Epoch 300 Training Results

The Precision-Recall curve from the training with batch 8 and epoch 300 is depicted in Figure 15, illustrating the tradeoff between precision and recall at different thresholds. The precision value is on the y-axis, the recall value is on the x-axis, and the mAP@0.5 value for all classes is 0.931.

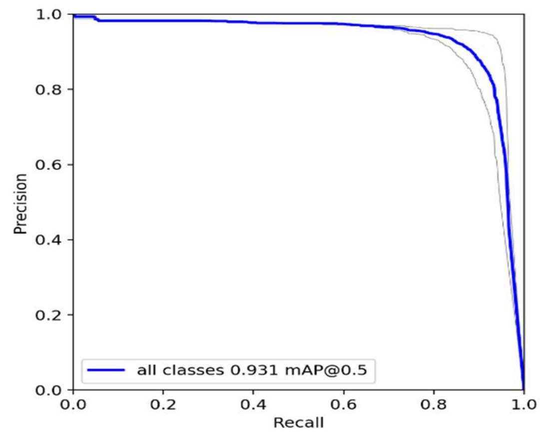


Figure 14: Precision-Recall Curve for Batch 8, Epoch 300

3.2. Learning Model

The model generated from the training process will be in the form of a .pt format file, which is the format used by PyTorch. It is divided into two parts, namely "last" and "best." These models can be used to continue training or hyperparameter tuning by determining the optimal advanced parameters if the previously generated model still does not yield satisfactory results when applied. The model used for the testing process later is "best_overall.pt," which represents the best learned model from the entire training process.

3.3. Testing Results

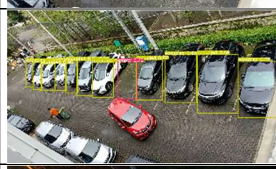
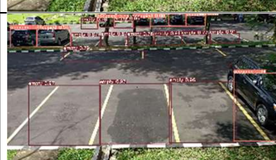
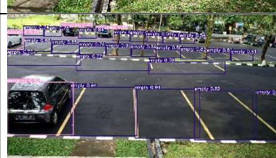
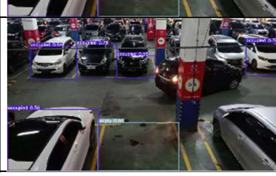
The test was conducted on 6 parking area videos with durations ranging from 30 minutes to 1 hour, following a real-time scenario setup. Some of the system testing results using the model trained with batch 8 and epoch 300, along with batch 8 and epoch 300 using grayscale data for underground parking, are presented in Table 1.

Tests were conducted using training results from batch 8 epoch 300 and batch 16 epoch 300. Additionally, experimenting with the inclusion of grayscale data yielded various accuracies. A summary of the system trials is presented in Table 2, showcasing true and false detection results as well as the accuracy achieved for each parking space.

Journal of Theoretical and Applied Information Technology
31st August 2024. Vol.102. No. 16

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1. Test Results

No	Parking Data	Actual	Predicted
1.		Occupied : 12 Empty: 0	Occupied : 12 Empty: 0
2.		Occupied : 9 Empty: 1	Occupied : 8 Empty: 1
3.		Occupied : 11 Empty: 1	Occupied : 11 Empty: 1
4.		Occupied: 2 Empty: 9	Occupied : 2 Empty: 8
5.		Occupied : 8 Empty: 13	Occupied : 7 Empty: 13
6.		Occupied : 8 Empty: 13	Occupied : 8 Empty: 12
7.		Occupied : 4 Empty: 24	Occupied : 4 Empty:24
8.		Occupied : 5 Empty: 0	Occupied : 4 Empty: 0
9.		Occupied : 5 Empty: 1	Occupied : 5 Empty: 1
10.		Occupied : 5 Empty: 1	Occupied : 5 Empty: 1

Due to the overall detection results favoring batch 8 epoch 300 over batch 16 epoch 300, a detailed accuracy calculation was focused solely on batch 8 epoch 300 for each parking location. Similarly, when comparing the results of batch 8 epoch 300 with grayscale data, it was evident that the model performed well specifically in underground parking areas. Therefore, specific accuracy calculations were carried out exclusively for underground parking areas.

Table 2: Comparison of Accuracy Results from Various Experiments

Epoch	Batch	Parking Lot	Testing True	Testing False	Accuracy
300	8	Campus	892	40	95,71%
		Library	1620	72	95,74%
		Underg-round	150	30	83,33%
		Under-ground (data grayscale)	288	34	89,44%
300	16	All	666	250	72,71%

Based on the system testing conducted in this study, the YOLOR algorithm can detect the availability of parking slots in real time, identifying both empty and filled slots with a testing frame rate of 13 fps. However, when loading the testing results onto the built websites, the frame rate drops to approximately 10-15 fps. This decrease may be attributed to the system needing to simultaneously perform the detection process and display the results, requiring GPU support for detecting each available parking space.

Furthermore, the system still encounters some detection errors. These errors may arise from low lighting conditions, as some trial data was captured at night to assess the system's performance in low light environments. Additionally, detection errors can occur if a car enters and obstructs part or all of the parking slot, rendering it invisible to the camera. The confusion matrix depicting the test results is presented in Table 3.

Table 1. Confusion Matrix

Predicted values	Actual values: Occupied	Empty	Total
Occupied	1313	0	1313
Empty	0	1487	1487
Total	1313	1487	2800

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

False negative values are derived when the model fails to detect both occupied and empty objects. For instance, an empty parking slot that goes undetected by the model, as well as filled slots that remain undetected. The intermittent nature of these detections leads to instances where the model may detect a slot one moment but fail to do so in subsequent frames, resulting in false negatives due to the model's low confidence level.

On the other hand, false positive values indicate instances where the model incorrectly detects objects that do not belong to either the occupied or empty classes. These misclassifications may occur when the model mistakenly identifies passing cars as filled parking space. The total counts of true positive, false positive, and false negative values for each parking area are tabulated in Table 4 for the Campus parking area.

Table 4. Total Values of TP, FP, FN in Campus Parking Area

	TP	FP	FN
Occupied	659	2	26
Empty	233	0	12
Total	892	2	38

Table 5 displays the true positive, false positive, and false negative values obtained from the test results in the library parking area.

Table 5. Total Values of TP, FP, FN in The Library Parking Area

	TP	FP	FN
Occupied	374	1	20
Empty	1246	0	51
Total	1620	1	71

Subsequently, the test results from the underground parking at the Center Point mall are presented in Table 6.

Table 6. Total Values of TP, FP, FN in Underground mall parking

	TP	FP	FN
Occupied	280	9	25
Empty	8	0	0
Total	288	9	25

Precision and recall for each object class detected are calculated using equation (2) and equation (3) so that the precision and recall for each object class are shown in Table 7 along with the f1-score obtained based on equation (4).

$$Precision = \frac{TP}{TP+} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1 - score = 2\,x\,\frac{precision\ x\ recall}{precision+rec} \quad (4)$$

Table 7. Precision and recall values for each class

	Parking	Precision	Recall	F1-Score
Occupied	Campus	0,99	0,96	0,97
	Library	0,99	0,95	0,97
	Under-ground	0,97	0,92	0,94
Empty	Campus	1,00	0,95	0,97
	Library	1,00	0,96	0,98
	Under-ground	1,00	1,00	1,00

Following the parking slot detection system test using YOLOR on the acquired testing data, the accuracy is determined by equation (5) for each parking area. The resulting accuracies are 95.71% for Campus parking, 95.74% for Library parking space, and 89.44% for underground parking at the mall.

$$Accuracy = \frac{total\ correct\ number\ of\ images}{total\ number\ of\ images}\,x\ 100\% \quad (5)$$

$$Campus = \frac{892}{932}x\ 100\% = 95{,}71\%$$

$$Library = \frac{1620}{1692}x\ 100\% = 95{,}74\%$$

$$Underground\ (Mall) = \frac{288}{322}x\ 100\% = 89{,}44\%$$

From the conducted tests, it was observed that the underground parking detection using batch 8 and batch 16 epoch 300 models, as well as detection in the Campus parking area with minimal lighting, did not yield satisfactory results. However, employing a model that utilizes grayscale data produced superior outcomes, especially in areas with insufficient lighting such as the underground mall parking area and Campus parking area when lighting conditions are poor.

The system developed can effectively detect parking slot availability in three parking areas with appropriate camera angles, normal to slightly reduced lighting, and even in rainy weather conditions. While there are still some detection errors, such as mistakenly identifying a passing car as occupying a parking space, these errors are minimal. The system performed well in detecting empty and occupied parking space, especially in the Campus and Library parking areas, swiftly identifying changes from occupied to empty. Overall, the system can accurately and promptly display real-time detection results with an overall accuracy of 95.04%.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4. DISCUSSION

The parking space detection system that was constructed underwent testing at three parking locations over a period of two days, with data recorded in two-hour intervals each day. During system testing and data collection, various challenges were encountered that could not be avoided. These included the absence of CCTV at certain parking locations, necessitating the use of a camera with a limited viewing area that was not sufficiently elevated to capture many parking slots. Additionally, difficulties arose when parking slots were obscured by passing cars or vehicles parked in front of them, particularly in underground parking areas with low ceilings where high camera angles were impractical due to potential obstructions by passing cars or parked vehicles.

The test results revealed that the model performed well under normal conditions and the correct camera angle. While the model could detect most parking slots, occasional detection errors occurred when a passing car was mistakenly identified as occupying a slot, albeit only temporarily. Intermittent detection results were also observed, with the model occasionally failing to detect parking space before re-establishing detection. The confidence levels for intermittent detections ranged from 0.6 to 0.8. Detection accuracy decreased when considering underground parking scenarios.

An evaluation of the system's performance included testing different training result models using batches 8 and 16, epoch 300, and grayscale data. The analysis indicated that the model best suited for low-light or nighttime conditions was the training result model utilizing grayscale data, as it yielded accurate detection of parking space under challenging lighting conditions.

5. CONCLUSION

In this study, a system for detecting parking space availability was successfully designed and implemented using the YOLOR method to detect empty and filled parking objects. The system was tested in three different parking environments. The research results indicate the following conclusions:

1. The YOLOR-based system achieves rapid detection speeds, processing video feeds in real-time while maintaining high accuracy. The system showed impressive performance across different parking environments, with accuracy rates of 95.71% in the campus parking area, 95.74% in the library parking area, and 89.44% in the underground mall parking area. These results represent a marked improvement over traditional methods and some previous deep learning approaches.
2. The system demonstrates robust performance across various parking scenarios, including open spaces and underground parking, and under different lighting and weather conditions. This versatility is crucial for widespread urban implementation
3. The research provided valuable insights into data collection and system implementation. We found that camera angles significantly impact the system's effectiveness, with front-facing perspectives yielding the best results. Additionally, camera height is crucial for ensuring visibility of all parking spaces without obstruction. Reducing the time spent searching for parking, our system contributes to decreased traffic congestion, lower fuel consumption, and reduced carbon emissions, aligning with smart city initiatives and environmental goals
4. While the system performed well in most scenarios, implementation in underground parking areas with low ceiling heights posed challenges. However, it showed promise for multi-level underground parking spaces with more open lower floors. Future work could focus on addressing these limitations and further optimizing the system for even greater efficiency.

ACKNOWLEDGMENT:

The authors would like to thank profusely for the financial support provided by DIPA funds Polmed (Politeknik Negeri Medan) to carry out this research with contract number: B/451/PL5/PT.01.05/2024. The funds received from the DIPA POLMED gives a very important role in the successful completion of this study.

REFERENCES :

[1] I. Aryanta, "Tanggung Jawab Pengelola Parkir Terhadap Kehilangan Barang Atau Kendaraan Pengguna Jasa Parkir Di Area Parkir Pasar Yadnya Desa Adat Blahbatuh Kecamatan Blahbatuh Kabupaten Gianyar," Universitas Mahasaraswati Denpasar, 2023.
[2] A. R. Yuliantara, A. S. Wibowo, and N. Prihatiningrum, "Sistem Informasi Lokasi Slot Parkir Kosong Berbasis Internet Of Things Pada

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Gedung Parkir Bertingkat," *eProceedings of Engineering,* vol. 9, no. 3, 2022.

[3] T. Pribadi, & Mali, P. Y. (2017). *Survei: Butuh 21 Menit Mencari Tempat Parkir di Jakarta.* Available: https://www.viva.co.id/amp/otomotif/mobil/971400-survei-butuh-21-menit-mencari-tempat-parkir-di-jakarta

[4] Brilio, "Berapa uang & waktu terbuang karena susah cari parkir di Jakarta? ," 14 Mei 2024 2019.

[5] R. Ke, Y. Zhuang, Z. Pu, and Y. Wang, "A smart, efficient, and reliable parking surveillance system with edge artificial intelligence on IoT devices," *IEEE Transactions on Intelligent Transportation Systems,* vol. 22, no. 8, pp. 4962-4974, 2020.

[6] P. Tătulea, F. Călin, R. Brad, L. Brâncovean, and M. Greavu, "An image feature-based method for parking lot occupancy," *Future Internet,* vol. 11, no. 8, p. 169, 2019.

[7] G. N. Rizkatama, A. Nugroho, and A. F. Suni, "Sistem Cerdas Penghitung Jumlah Mobil untuk Mengetahui Ketersediaan Lahan Parkir berbasis Python dan YOLO v4," *Edu Komputika Journal,* vol. 8, no. 2, pp. 91-99, 2021.

[8] O. Taylor, P. Ezekiel, and V. Emmah, "Smart Vehicle Parking System Using Computer Vision and Internet of Things (IoT)," *European Journal of Information Technologies and Computer Science,* vol. 1, no. 2, pp. 11-16, 2021.

[9] R.F. Rahmat, S. Purnamawati, J. Kurniwawan, S. Faza, and P. M.F. Pasha, "Vacant parking space identification using probabilistic neural network," *Indonesian Journal of Electrical Engineering and Computer Science,* no. Vol 14, No 2: May 2019, pp. 887-894, 2019.

[10] N. Pawar *et al.*, "Miniscule object detection in aerial images using YOLOR: a review," in *Proceedings of International Conference on Communication and Computational Technologies: ICCCT 2022*, 2022, pp. 697-708: Springer.

[11] X. Cao, Y. Su, X. Geng, and Y. Wang, "YOLO-SF: YOLO for fire segmentation detection," *IEEE Access,* 2023.

[12] K. Maharana, S. Mondal, and B. Nemade, "A review: Data pre-processing and data augmentation techniques," *Global Transitions Proceedings,* vol. 3, no. 1, pp. 91-99, 2022.

[13] H. Talebi and P. Milanfar, "Learning to resize images for computer vision tasks," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 497-506.

[14] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "You only learn one representation: Unified network for multiple tasks," *arXiv preprint arXiv:2105.04206,* 2021.

[15] D. N. Diniz *et al.*, "A cytopathologist eye assistant for cell screening," *AppliedMath,* vol. 2, no. 4, pp. 659-674, 2022.