 https://doi.org/10.24191/jeesr.v26i1.001

A License Plate Detection and Recognition Method using YOLOv5n and LPRNet

Wan Xing, Juliana Johari, and Fazlina Ahmat Ruslan*

***Abstract*—Efficient and speedy license plate identification is crucial in vehicle-to-vehicle (V2V) communication scenarios for applications like traffic management and law enforcement. This study introduces a method that utilizes cutting-edge deep learning models to identify and recognize license plates on vehicles. The YOLOv5n model, renowned for its exceptional detection precision and efficient design, is utilized for license plate detection. After the detection phase, the License Plate Recognition Network (LRPNet) is employed to precisely identify the characters on the detected license plates. The thorough assessment we conducted evaluates the performance of this integrated system on public datasets, showcasing its resilience and effectiveness. This paper provides an in-depth discussion of the license plate detection and recognition task through very detailed theoretical derivations and structural block diagrams combined with experiments. The findings indicate that the utilization of YOLOv5n and LRPNet shows the high precision and efficiency of license plate detection and recognition.**

***Index Terms*—License Plate, Object Detection, Recognition, CTC.**

I. INTRODUCTION

In recent years, the introduction of Intelligent Transportation Systems (ITS) has brought about a new era of communication technologies for vehicles, with a particular focus on V2V communication. This is an essential element of connected vehicle technology, allowing vehicles to share critical information instantly. This emerging discipline holds the potential to improve the safety of roadways, alleviate traffic congestion, and facilitate the development of self-driving vehicles. An essential component of V2V communication is the precise and effective identification of vehicles on the road, which is where Automated License Plate Recognition (ALPR) systems are utilized. [1].

ALPR systems utilize sophisticated image processing and deep learning algorithms to identify and interpret vehicle license plates in different settings. These systems have been extensively used in traffic enforcement, toll collection, parking management, and vehicle tracking. ALPR, in the context of V2V communication, enables smooth vehicle identification, which is crucial for the transmission of safety-related information, cooperative driving manoeuvres, and vehicular networking [2]. The integration of ALPR within V2V communication frameworks presents unique challenges and opportunities. On one hand, the dynamic and cluttered road environment can hinder accurate license plate detection and recognition, necessitating robust algorithms capable of handling varying lighting conditions, occlusions, and high speeds. On the other hand, the continuous stream of data generated by V2V communication provides a rich source of context that can be harnessed to enhance ALPR performance.

This paper investigates ALPR technologies, analysing the underlying mechanisms, present achievements, and prospects. We explored the technological complexities of ALPR systems and provided a detailed and in-depth discussion of the commonly used combination of YOLOv5 and LRPNet for license plate detection and recognition. However, all previous literature only examines the task from an experimental perspective, lacking a theoretical analysis. Instead, the paper systematically deduces the forward prediction process, the loss function, and the decoding algorithm, thereby unveiling the fundamental principles and testing the accuracy of various decoding algorithms on the China City Parking Dataset. which will help to design more robust algorithms.

II. RELATED WORKS

In a license plate detection task, the license plate is usually localized first, and then the characters in the plate are recognized. The localization methods are generally based on target detection, while the recognition methods are mainly based on classification models. Since the license plate detection model generally runs on mobile devices, the light weight of the model is a very important indicator.

A. Location Methods

The localization methods usually used are based on target detection. Object detection is a computer vision technique essential for identifying and locating instances of objects within images or videos. This technology enables machines to classify and determine the positions of various objects, enhancing applications such as surveillance, autonomous driving, and V2V communication. In the context of V2V, object detection can be used to locate and recognize vehicle license plates, which is crucial for applications like traffic law enforcement, toll collection, and vehicle identification. Deep learning has revolutionized object detection, introducing sophisticated algorithms that can identify objects with

This manuscript is submitted on 10 June 2024, revised on 26 September 2024 and accepted on 17 October 2024. Xing Wan, Juliana Johari, and Fazlina Ahmat Ruslan are from School of Electrical Engineering, College of Engineering, Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia.

*Corresponding author
Email address: fazlina419@uitm.edu.my

unprecedented accuracy. Broadly, deep learning-based object detection models can be categorized into two types: one-stage detectors and two-stage detectors [3].

Two-stage detectors first propose regions of interest in an image and then classify and fine-tune the locations of these proposed regions. Although two-stage detectors, such as Fast R-CNN, Faster R-CNN and Mask R-CNN [4]–[6], are known for their high accuracy, they suffer from slower detection speeds and more complex models, making them less suitable for real-time V2V detection requirements.

The You Only Look Once (YOLO) models are commonly used in one-stage algorithms. One-stage detectors, like YOLOv3, YOLOv6, and YOLOv7 (You Only Look Once) models, treat object detection as a single regression problem, simultaneously predicting bounding boxes and class probabilities directly from the image in one evaluation [7]–[9]. YOLO models have gained substantial popularity for their balance of speed and accuracy, particularly in applications demanding real-time performance [10].

Among the various versions, YOLOv5 has the following advantages: Firstly, the design of YOLOv5 prioritizes high efficiency and adaptability for industrial applications. It boasts high detection precision while maintaining computational efficiency, making it ideal for deployment in real-world scenarios. Secondly, despite being fast, YOLOv5 does not compromise detection accuracy, providing the robust results necessary for tasks such as license plate detection. This approach significantly enhances speed, making one-stage detectors more appropriate for real-time V2V applications where rapid detection is crucial. Fig. 1 depicts the structure of YOLOv5, comprising a backbone, a neck, and a header.

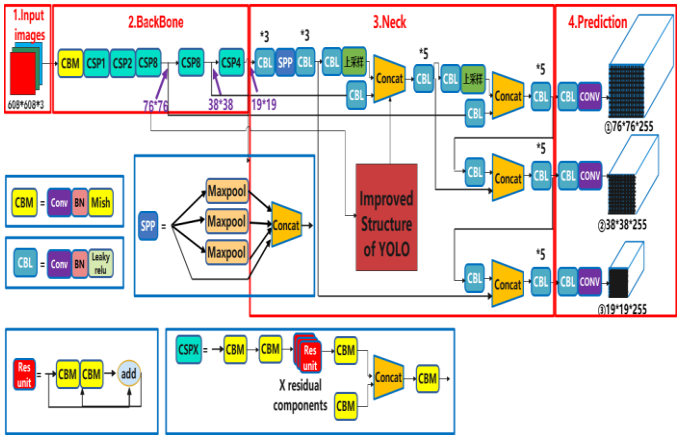

Fig. 1. The Structures of YOLOv5

Sachin Dhyani et al. used the YOLOv7x algorithm to detect vehicle license plates and applied EasyOCR to the detected plates to recognize the text. The model is trained on a customized dataset containing only Indian vehicle license plates [11]. Ying Quan et al. designed and developed a GUI widget for YOLOv8 [12]. Chin-Fa Hsieh et al. completed an intelligent parking management system by combining automatic license plate recognition and line function [13]. Shi et al. included an enhanced channel attention mechanism in the down-sampling stage of the YOLOv5 algorithm to improve the location of the license plate [14].

B. Recognition Methods

License plate characters typically comprise both alphanumeric characters and Chinese characters, making classification networks well-suited for this purpose. The recognition task should have the capability to process license plate characters of different lengths, while the network model should be designed to be as lightweight as feasible.

Sergey Zherzdev and his colleagues introduced LPRNet, a comprehensive approach for automatic license plate recognition that does not require character segmentation [15]. Their methodology operates in real-time and achieves a recognition accuracy of up to 95% for Chinese license plates [15]. Wang et al. proposed a multi-task convolutional neural network for license plate detection and recognition (MTLPR) with better accuracy and lower computational cost [16]. Huang et al. developed an innovative license plate recognition network that can accurately identify and categorize characters and license plate areas simultaneously [17]. This network includes an assembly layer that combines the characters to form license plates and outputs the license plate strings.

Anmol Pattanaik et al. introduced a method called DCTGAN, which combines Generative Adversarial Networks (GAN) with a discriminator based on Discrete Cosine Transform (DCT). This approach aims to enhance the resolution and remove different types of blur and complexities from license plates [18].

III. METHODS

The flowchart for license plate detection and recognition in this study is shown in Fig. 2. YOLOv5n locates the license plate first, and LRPNet then recognizes the characters.

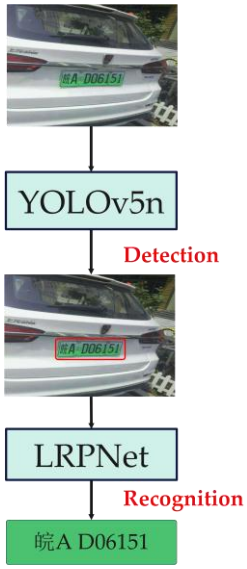

Fig. 2. The License Plate Detection and Recognition.

A. *YOLOv5n for License Plate Location*

In this research, we focus on leveraging YOLOv5n, the nano version in the YOLOv5 family, for vehicle license plate detection. YOLOv5n is particularly advantageous due to its compact model size and rapid detection speed, which align perfectly with the stringent real-time performance requirements of V2V communication systems. Using YOLOv5n, we aim to maintain a high level of detection accuracy while ensuring swift processing, thereby meeting the demands of license plate detection in V2V scenarios.

The backbone of YOLOv5n uses CSPDarknet, which facilitates feature extraction by passing images through a series of convolutional layers. Cross-Stage Partial Networks split the base feature map into two parts; one part passes through a series of layers and the other bypasses them. These two parts are then concatenated. This split-and-merge strategy helps enrich the gradient flow through the network, improving learning and performance. CSP networks incorporate residual blocks, which consist of convolutional layers along with shortcut connections. These blocks facilitate deeper networks, essential for capturing intricate features without the risk of vanishing gradients.

YOLOv5n utilizes PANet in the neck part to enhance the feature pyramid by combining feature maps from different stages. The concept of FPN underpins PANet, enabling the model to utilize both low-level and high-level features. This is crucial for detecting objects of varying scales. PANet introduces additional connections that aggregate feature maps, improving the propagation of strong features and enhancing information flow between layers. This results in better object localization and classification.

The detection head is responsible for predicting the bounding boxes, objectness scores, and class probabilities from the aggregated features obtained from the neck. To predict bounding boxes, the head uses predefined anchor boxes for different scales. It ensures that the model can detect objects of various sizes effectively. Sigmoid functions are used to obtain confidence scores that indicate the presence of an object in the predicted bounding boxes. Predictions are made for each grid cell on the feature map, ensuring that objects can be detected regardless of their position in the image.

Several post-processing techniques refine the results after obtaining the raw predictions. Non-maximum suppression (NMS) eliminates redundant bounding boxes that overlap significantly, keeping only the most confident one. NMS generally needs to rely on Intersection over Union (IoU) to filter out the most compliant candidate prediction frames. It guarantees that a single, accurate bounding box represents each detected object. Bounding box regression further refines the coordinates of the bounding boxes, enhancing accuracy.

B. *LRPNet*

Fig.3 illustrates the overall process of license plate recognition. The logits output by LRPNet are the probabilities of characters, which are decoded by Connectionist Temporal Classification (CTC) [19]. Note that CTC may eventually decode different outputs into the same prediction result.

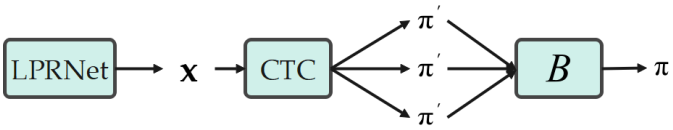

Fig. 3. The Process of License Plate Recognition.

The challenges in identifying license plates may be related to the issue of recognizing sequences, which can be effectively addressed by modeling them using the LRPNet output sequence, represented in matrix form. The matrix contains T rows and $|L|$columns, where T represents the maximum time slice and $|L|$ represents the number of characters. The output sequences are decoded using CTC.

Typically, the RNN appends a softmax layer with a connection matrix to classify the outputs. The timestep refers to the dimension of the time sequence, while the number of classes reflects the dimensions of the category to predict. The length of the timescale is represented by t, the length of the hidden layer is represented by m, the output matrix of the RNN is represented by $R^{T\times m}$, and the connection layer, which converts the RNN output dimension m to category number n, is represented by $W^{m\times n}$, as shown in (1).

$$N_w: (R^m)^T \rightarrow (R^n)^T \tag{1}$$

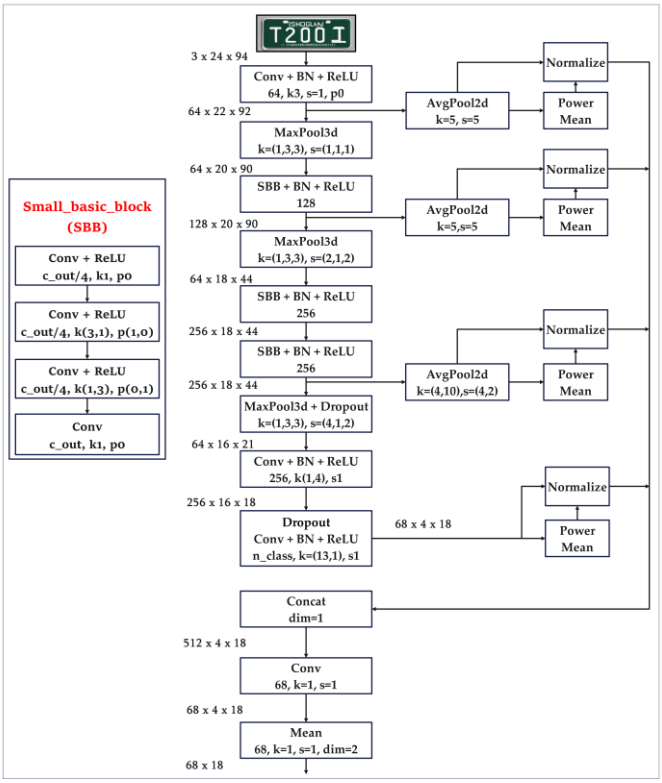

Fig. 4. The Architecture of LPRNet.

As shown in Fig.4, the LRPNet extracts the license plate's features through a series of convolutional layers and saves the output features of specific layers. These saved features are

pooled and normalized before being concatenated, and finally the predicted probabilities of characters on each time slice are obtained through convolution output.

C. CTC Algorithm

Sequence learning problems often involve multi-to-multi mapping interactions. CTC addresses the challenge of multi-to-one mapping by introducing a distinct blank character, denoted as "_". Given that the character set to be identified is $L = \{A, B, C, \ldots, X, Y, Z, 0,1,2, \ldots,9\}$, merging the blank characters into the set defines the extended character set, as shown in (2).

$$V = L \cup \{blank\} \tag{2}$$

The definition of the operation B includes incorporating identical characters and eliminating blank characters. Given that the sequence, π' initially comprises blank characters and is decoded to a sequenceπ consisting exclusively of characters from set L. An example of decoding is shown in Fig. 5, where all different words are memorized and decoded as "cat".

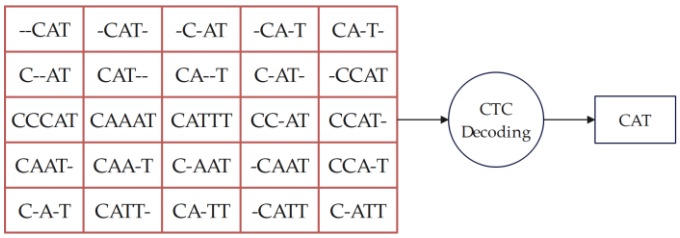

Fig. 5. An Example CTC decoding.

The sequence length changes from time slice T' to T and it is evident that CTC only outputs the final sequence with a length lower than the input length. Therefore, CTC defines the transform B, as shown in (3):

$$B: V^{T'} \rightarrow L^{T}, T' \leq T \tag{3}$$

Let $l' \in V^{2T+1}$be the sequence with blanks inserted before and after each character of $l \in L^{T}$. Therefore, there is a one-to-one mapping F between l' and l, as shown in (4).

$$F: l' \sim l \tag{4}$$

Equation (5) shows the likelihood of output after CTC converts and merges input x into π'.

$$p(l|x) = \sum_{\pi \in B^{-1}(l)} p(\pi|x) \tag{5}$$

In this scenario, $B^{-1}(l)$ denotes the collection of all paths that have been transformed into the resultant π. The definition of $p(\pi|x)$ is shown in (6). Here, $y^{t}_{\pi_t=v}$ indicates the probability of the character $v$occurring at time t, where $v = \pi_t$. For each given route π, the following (6) holds.

$$p(\pi|x) = \prod_{t=1}^{T'} y^{t}_{\pi_t=v}, v \in V \tag{6}$$

The requirements for the (6) are that $\pi_1, \pi_2, \ldots, \pi_{T'}$ are mutually independent. Maximizing the probability allows us to obtain optimized parameters for LPRNet. However, due to the large magnitude of $|B^{-1}(l)|$, the computation of probabilities is not effective. Therefore, the dynamic planning algorithm divides the probability calculation into forward probability and backward probability to reduce computations. Let l' be the sequence with blanks inserted before and after each character of l. $\pi_1, \pi_2, \ldots, \pi_{T'}$ are classified into two groups according to whether they contain the characters l'_k at time t, as shown in (7).

$$\frac{\partial p(l'|x)}{\partial w} = \frac{\partial \sum_{\pi \in B^{\{-1\}}(l)} p(\pi|x)}{\partial w} \tag{7}$$

As the second item above is unrelated to l'_k, thus (8) holds.

$$\frac{\partial p(l|x)}{\partial w} = \frac{\partial p(l'|x)}{\partial w} = \frac{\partial \sum_{\pi \in B^{-1}(l), \pi_t = l'_k} p(\pi|x)}{\partial w} \tag{8}$$

As shown in (9), p_f represents the forward probability, and $p_{backward}$ denotes the backward probability.

$$\sum_{\pi \in B^{-1}(l), \pi_t = l'_s} p(\pi|x) = \frac{p_f(ts, s) \cdot p_b(ts, s)}{p(\pi_t = l'_s)} \tag{9}$$

The function $\alpha_{ts}(s)$ denotes the process of decoding $\pi_{1:ts}$ of ts timestamps into s characters, $l'_{1:s}$, where l is the original string and l' is the CTC decoded string with blank extension strings between and at the end of each character. In the (10), t represents the time slice and s represents the preceding s characters in the sequencel'.

$$\alpha_{ts}(s) = \sum_{\pi \in V^{T'}, B(\pi_{1:t}) = l'_{1:s}} \prod_{t=1}^{ts} y^{t}_{\pi_t} \tag{10}$$

The $\alpha_{ts}(s)$ denotes the cumulative probability of all forward paths that have been assigned to the sequence $l'_{1:s}$ up to time t. The $y^{t}_{\pi_t}$ represents the probability of character π_t at time t. The sequence can only start from a blank or l'_1, and the following conditions should be met, where $s \in [1, |l'|]$, as shown in (11), (12), and (13).

$$\alpha_1(1) = y^{1}_{blank} \tag{11}$$

$$\alpha_1(2) = y^1_{l'_1} \tag{12}$$

$$\alpha_1(s) = 0, \forall 2 < s \leq |l'| \tag{13}$$

If π_t is blank (case 1) or π_t is not equal to π_{t-2} (case 2), the recursive process is established, as shown in (14).

$$\alpha_t(s) = \left(\alpha_{t-1}(s) + \alpha_{t-1}(s-1)\right) * y^t_{l'_s} \tag{14}$$

In other cases (case 3), the recursive process is established, as shown in (15).

$$\alpha_t(s) = \left(\alpha_{t-1}(s) + \alpha_{t-1}(s-1) + \alpha_{t-1}(s-2)\right) * y^t_{l'_s} \tag{15}$$

The iterative examples of case 1, case 2, and case 3 are shown in Fig. 6.

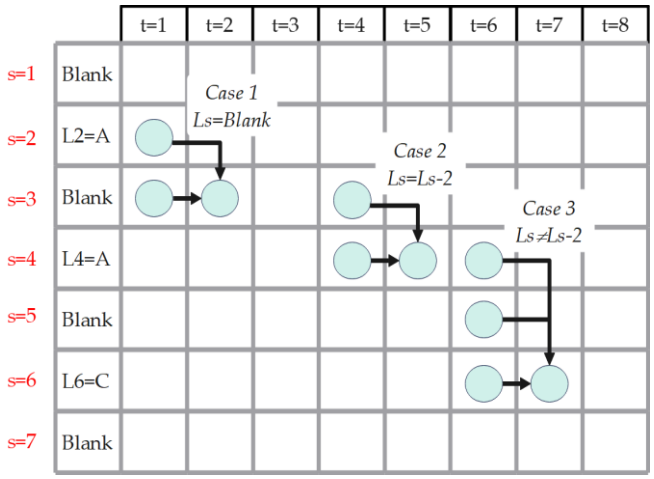

Fig. 6. The Examples of Forward Process.

When using a forward function, the r must reach either the last blank or the last non-blank character at time T', as shown in (16).

$$p(l|x) = \alpha_{T'}(|l'|) + \alpha_{T'}(|l'| - 1) \tag{16}$$

The function $\beta_t(s)$ denotes the backward probability of the conversion of $\pi_{t:T'}$ to $l_{s:|l'|}$, represented as $B(\pi_{t:T'}) = l_{s:|l'|}$, as shown in (17).

$$\beta_t(s) = \sum_{\pi \in V^{T'}, B(\pi_{t:T'}) = l'_{s:|l'|}} \prod_{t=ts}^{T'} y^t_{\pi_t} \tag{17}$$

Like the forward probability, the backward probability satisfies the following (18) – (20).

$$\beta_{T'}(|l'|) = y^{T'}_{blank} \tag{18}$$

$$\beta_{T'}(|l'| - 1) = y^{T'}_{l'_{|l'|}} \tag{19}$$

$$\beta_{T'}(s) = 0, \forall s < |l'| - 1 \tag{20}$$

If $l'_s = blank$ or $l'_{s+2} = l'_s$, its recursive relationship is following (21).

$$\beta_t(s) = \left(\beta_{t+1}(s) + \beta_{t+1}(s+1)\right) * y^t_{l'_s} \tag{21}$$

In other cases, it meets the following (22).

$$\beta_t(s) = \left(\beta_{t+1}(s) + \beta_{t+1}(s+1) + \beta_{t+1}(s+2)\right) * y^t_{l'_s} \tag{22}$$

The product of forward and backward probabilities can be expressed as (23).

$$\alpha_t(s)\beta_t(s) = y^t_{l'_s} \sum_{\pi \in B^{-1}(l), \pi_t = l'_s} \prod_{t=1}^{T'} y^t_{\pi_t} \tag{23}$$

Further, one can get the following (24).

$$\frac{\alpha_t(s)\beta_t(s)}{y^t_{l'_s}} = \sum_{\pi \in B^{-1}(l), \pi_t = l'_s} p(\pi|x) \tag{24}$$

Iterating over all the positions of s yields the likelihood probability function, as shown in (25).

$$p(l|x) = \sum_{s=1}^{|l'|} \sum_{\pi \in B^{-1}(l), \pi_t = l'_s} p(\pi|x) = \sum_{s=1}^{|l'|} \frac{\alpha_t(s)\beta_t(s)}{y^t_{l'_s}} \tag{25}$$

Now we must compute the partial derivative of $p(l \vee x)$ to y^t_k, and differentiate the likelihood function mentioned earlier. The condition for l'_s to contain y^t_k is only when l'_s is equal to k. All other partial derivatives are equal to zero. The string l'_s may contain several instances of the character$k$. Therefore, the set of k positions in l'_s is defined as $pos(l', k) = \{s: l'_s = k\}$. Equation (25) can be transformed into (26).

$$p(l|x) = \sum_{k=1}^{V} \sum_{s \in pos(l', k)} \frac{\alpha_t(s)\beta_t(s)}{y^t_k} \tag{26}$$

Taking the derivative of the above equation, we get (27).

$$\frac{\partial p(l|x)}{\partial y^t_k} = \sum_{s \in pos(l', k)} \frac{\alpha_t(s)\beta_t(s)}{{y^t_k}^2} \tag{27}$$

Therefore, we can express the probability of the log-likelihood function as (28).

$$\frac{\partial ln\big(p(l|x)\big)}{\partial y_k^t} = \frac{1}{p(l|x)} \frac{\partial p(l|x)}{\partial y_k^t} \tag{28}$$

The loss function on the training dataset is expressed as (29).

$$Loss = -\sum_{(x,l)\in D} ln\big(p(l|x)\big) \tag{29}$$

After getting the loss, the gradient of logits can be calculated. At time t, the LPRNet outputs, $u_k^t, k \in [1, V = L'], 1 \le t \le T'$, which is usually connected to a softmax for normalization, as shown in (30).

$$y_k^t = \frac{e^{u_k^t}}{\sum_{i=1}^{V} e^{u_i^t}} \tag{30}$$

Here the output of softmax is $y_k^t, k \in [1, V = L']$, where the derivative of the softmax function is shown in (31).

$$\frac{\partial y_j^t}{u_k^t} = I(k = j) \cdot y_k^t \cdot (1 - y_k^t) - I(k \neq j) \cdot y_j^t \cdot y_k^t \tag{31}$$

Using the chain rule, we can get (32).

$$\frac{\partial lnp(l|x)}{\partial u_k^t} = y_k^t \left(\frac{\partial lnp(l|x)}{\partial y_k^t} - \sum_{j=1}^{V} \frac{\partial lnp(l|x)}{\partial y_j^t} \cdot y_j^t \right) \tag{32}$$

The expression for the derivative of the likelihood probability is (33).

$$\frac{\partial p(l|x)}{\partial y_j^t} = \sum_{s\in lab(l',j)} \frac{\alpha_t(s)\beta_t(s)}{\left(y_j^t\right)^2} \tag{33}$$

Substituting (33) into (32) and using (25), we can get a more concise, as shown in (34).

$$\frac{\partial lnp(l|x)}{\partial u_k^t} = y_k^t \cdot \frac{\partial lnp(l|x)}{\partial y_k^t} - y_k^t \tag{34}$$

To avoid numerical underflow, the normalization coefficient of C_t for each time slice t is calculated, as shown in (35).

$$C_t = \sum_s \alpha_t(s) \tag{35}$$

Therefore, the normalized forward probability can be written as following (36).

$$\hat{\alpha}_t = \frac{\alpha_t(s)}{C_t} \tag{36}$$

Equation (37) illustrates the need for compensation due to scaling when calculating the final probability.

$$ln\big(p(l|x)\big) = ln\big(\hat{\alpha}_T(|l'|) + \hat{\alpha}_T(|l'|-1)\big) + \sum_{t=1}^{T} ln(C_t) \tag{37}$$

For the backward algorithm, similar scaling can be done, as shown in (38) and (39).

$$D_t = \sum_s \beta_t(s) \tag{38}$$

$$\hat{\beta}_t = \frac{\beta_t(s)}{D_t} \tag{39}$$

Normalization is necessary for both forward and backward algorithms, as the value becomes very small as recursion progresses, leading to value underflow.

D. Greedy Algorithm

Algorithm CTC Greedy Decoding

```
procedure GREEDYDECODE(y, blank)
    labels ← arg max(y, axis = 1)
    decoded_labels ← []
    previous ← None
    for each l in labels do
        if l ≠ previous then
            decoded_labels.append(l)
            previous ← l
        end if
    end for
    decoded_labels ← [l for l in decoded_labels if l ≠ blank]
    prob ← y[range(y.shape[0]), labels]
    score ← ∏_{i=1}^{len(prob)} prob_i
    return labels, decoded_labels, score
end procedure
```

Fig. 7. The Greedy Search Algorithm.

)

The decoding process therefore becomes a search for the character with the highest probability at each moment, as shown in Fig. 7. Greedy search only provides the optimal path and can further optimize the results if many optimal paths exist. Greedy search has the advantage of being a simple algorithm, but it only considers one of the paths.

As there is a chance that more than one character path may

be decoded to the same l in $p(l|x)$, we only consider the path with the highest probability to streamline the computation. Because of path independence after the CTC decoding assumption, for a given string, we have the following (40).

$$\pi = arg \max_{l \in (L')^T}(l)) \tag{40}$$

Greedy search only provides the optimal path and can further optimize the results if many optimal paths exist. Greedy search has the advantage of being a simple algorithm, but it only considers one of the paths. The beam search algorithm can determine the most efficient number of pathways. Since some original paths are the same after decoding, CTC must find the optimal decoding, as shown in Fig. 8.

Algorithm CTC Beam Search Decoding

```
procedure BeamDecode(y, beam_size, blank)
    (T, V) ← y.shape
    log_y ← log(y)
    beam ← [([], 0)]
    for t = 1 to T do
        new_beam ← []
        for each prefix, score in beam do
            for i = 1 to V do
                new_prefix ← prefix + [i]
                new_score ← score + log_y[t, i]
                new_beam.append((new_prefix, new_score))
            end for
        end for
        new_beam ← sorted(new_beam, key = x : x[1], reverse = True)
        beam ← new_beam[: beam_size]
    end for
    return beam
end procedure
```

Fig. 8. The Beam Search Algorithm.

The problem with beam search is that there may be multiple paths that have the same string after CTC decoding, which reduces the diversity of the results, whereas the prefix beam search algorithm allows for the merging of prefixes as the search progresses, as shown in Fig. 9. However, the complexity of this algorithm is much higher than the first two algorithms, so it may not necessarily be able to meet scenarios with speed requirements.

E. Experiment settings

We assess our methodology using the China City Parking Dataset (CCPD), which is currently the most extensive publicly annotated collection of license plate data in China [20]. The dataset comprises around 250,000 distinct license plate photos captured under various conditions, including diverse backdrops, shooting angles, times of day, and illumination levels (refer to Table 5). The CCPD dataset is partitioned into two subsets: CCPD2019 and CCPD2020. The former primarily consists of images of conventional fuel-powered automobiles, while the latter exclusively comprises images of new energy vehicles.

Since images from CCPD2020 are all new energy vehicles with 8-digit license plate lengths, to train the model for variable-length license plates, we added fuel car images with 7-digit license plate lengths from CCPD2019. Specifically, the training set consists of 5,769 images of new energy vehicles and 3,000 images of traditional fuel vehicles, while the validation set consists of 1,001 images of new energy vehicles and 600 images of fuel vehicles.

Algorithm CTC Prefix Beam Search Decoding

```
procedure PrefixBeamDecode(y, beam_size, blank)
    (T, V) ← y.shape
    log_y ← log(y)
    beams ← {() : (0, −∞)}
    for t = 1 to T do
        new_beams ← {() : (−∞, −∞)}
        for each prefix, (p_b, p_nb) in beams do
            for i = 1 to V do
                p ← log_y[t, i]
                if i = blank then
                    new_prefix ← prefix
                    (new_p_b, new_p_nb) ← new_beams[new_prefix]
                    new_p_b ← logsumexp(new_p_b, p_b + p, p_nb + p)
                    new_beams[new_prefix] ← (new_p_b, new_p_nb)
                else if not prefix or i ≠ prefix[−1] then
                    new_prefix ← prefix + (i, )
                    (new_p_b, new_p_nb) ← new_beams[new_prefix]
                    new_p_nb ← logsumexp(new_p_nb, p_b + p, p_nb + p)
                    new_beams[new_prefix] ← (new_p_b, new_p_nb)
                else
                    new_prefix ← prefix
                    (new_p_b, new_p_nb) ← new_beams[new_prefix]
                    new_p_nb ← logsumexp(new_p_nb, p + p_nb)
                    new_beams[new_prefix] ← (new_p_b, new_p_nb)
                    new_prefix ← prefix + (i, )
                    (new_p_b, new_p_nb) ← new_beams[new_prefix]
                    new_p_nb ← logsumexp(new_p_nb, p + p_b)
                    new_beams[new_prefix] ← (new_p_b, new_p_nb)
                end if
            end for
        end for
        sorted_beams ← sorted(new_beams, logsumexp(p_nb, p_b), reverse)
        for i = 1 to beam_size do
            beams[i] ← sorted_beams[i]
        end for
    end for
    return beams
end procedure
```

Fig. 9. Prefix Beam Search Algorithm.

TABLE I. EXPERIMENT SETTINGS

Name	Description
Detection Model	YOLOv5n.
Recognition Model	LRPNet.
Loss Function	CTC
Optimal Path Search	Greedy Search
Epochs	120.
Learning Rate	0.001
Optimizer	RMSProp
Categories	2 (Electrical, gasoline)
Training Dataset	8769 images
Validation Dataset	1601 images
GPU Platform	RTX3060

Table I shows the experimental parameter settings. The experiment was conducted on the RTX3060 platform, except for the FPS and FLOPs parameters, which were run on Intel (R) Core (TM) i5-8265U CPU platform. In addition, the experiment ran for 120 epochs because we observed that the model parameters had already converged beyond 100 epochs.

F. Evaluation Metrics

For the license plate detection model YOLOv5n, the evaluation metric mean Average Precision (mAP), recall, and precision. These metrics include precision, recall, mAP, True Negatives (TN), and True Positives (TP), all of which adhere to academic standards for rigorous evaluation.

Precision: Precision measures the proportion of correctly identified positive instances among all instances that the model predicts as positive. Recall quantifies the ability of the model to identify all positive instances out of all true positive instances present in the dataset. The mAP is a comprehensive metric that averages the precision values across multiple classes and interpolated recall points. It is particularly useful in multi-class object detection tasks, providing a single scalar value that summarizes the model's performance.

TN represents the number of correctly identified negative instances, where both the model's prediction and the ground truth indicate that no object of interest is present. While less emphasized in precision and recall calculations directly, TN contributes to overall accuracy and is crucial for understanding the model's behavior regarding non-target objects. TP are instances where the model correctly identifies the presence and location of an object of interest, matching the ground truth annotations. TP is a fundamental component for calculating precision and recall, highlighting the model's accuracy in detecting target objects. The evaluation metric used for the license plate recognition model, LRPNet, is accuracy. A license plate is accurately recognized only when all the characters are recognized correctly. In addition, the number of model parameters as well as speed are the key metrics to focus on for both license plate detection and recognition.

IV. Results and Discussion

Due to the powerful model capabilities of YOLOv5n, the fewer categories in the target dataset, and the non-dense characteristics of the targets in the license plate dataset, the model has a higher recall rate on the target dataset, as shown in Fig. 10. When the confidence threshold is less than 0.4, the recall for all classes is close to 100%.

The precision of the YOLOv5n model is shown in Fig.11. Both fuel and new energy vehicles exhibit very high detection precision when the threshold value exceeds 0.4. However, it should be noted that recall and precision are usually a pair of contradictions, which means that in most cases when one is improved, the other will decrease.

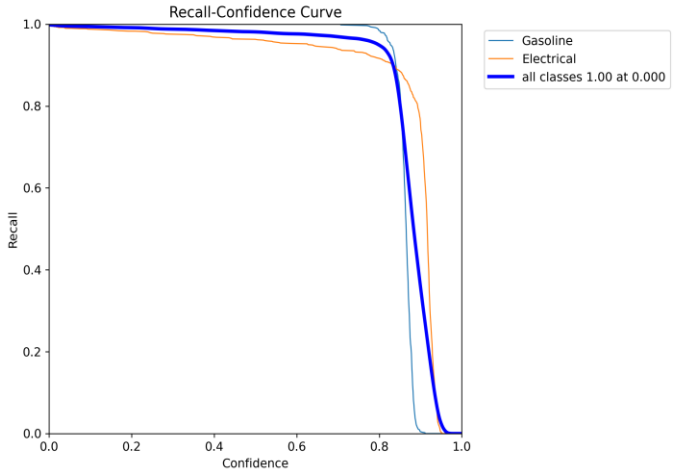

Fig. 10. The Recall of Detection.

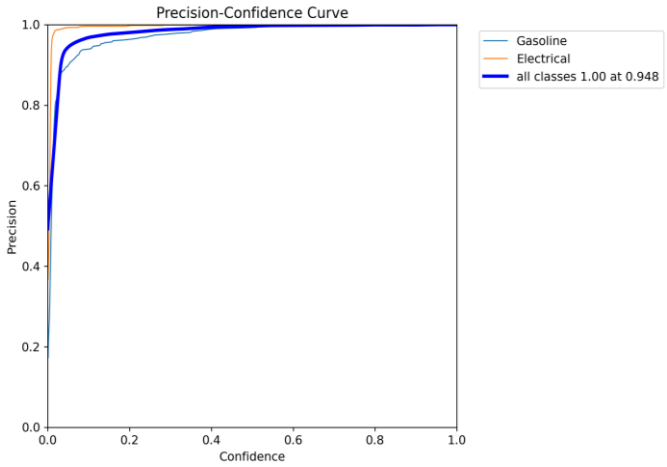

Fig. 11. The Precision of Detection using YOLOv5n.

Table II compares the NMS time, precision, and recall at different confidence thresholds when IoU is fixed at 0.6. As the confidence threshold increases, the model will only output a small number of those detections with very high confidence, reducing the processing time of NMS. When the confidence level is between 0.001 and 0.01, precision and recall keep stable at 0.994 and 0.984, respectively. When the confidence level rises to 0.5, the precision increases to 0.996, and recall decreases to 0.981, which is because fewer high-quality candidate bounding boxes are beneficial to the improvement of the precision, and at the same time there is a possibility of lower leakage of detection.

TABLE II. PERFORMANCE AT DIFFERENT CONFIDENCE THRESHOLDS

Conf	IoU	NMS Time	Precision	Recall
0.001	0.6	0.9ms	0.994	0.984
0.005	0.6	0.8ms	0.994	0.984
0.01	0.6	0.7ms	0.994	0.984
0.5	0.6	0.7ms	0.996	0.981

Table III shows that precision and recall do not change when the IoU threshold is varied from 0.4 to 0.75. However, when it is further increased to 0.95, precision decreases from 0.994 to 0.903 and recall decreases from 0.984 to 0.954. Too high IoU causes part of the objects that originally belong to the same category to be judged as belonging to other categories, resulting in misjudgments and a decrease in precision. Also, raising the IoU threshold means that more real targets may not be matched, which leads to missed detections and lower recall. The IoU threshold has no significant effect on NMS improvement. Based on the analysis of the above two aspects, an IoU threshold of 0.6 and a confidence threshold of 0.001 are relatively good choices.

TABLE III. PERFORMANCE AT DIFFERENT IOU THRESHOLDS

Conf	IoU	NMS Time	Precision	Recall
0.001	0.4	0.9ms	0.994	0.984
0.001	0.6	0.9ms	0.994	0.984
0.001	0.75	0.9ms	0.994	0.984
0.001	0.95	0.9ms	0.903	0.954

For license plate detection, the first thing is to ensure a higher recall because the consequences of missed detection are more serious than a slight decrease in accuracy. In fact, YOLOv5 is the most widely used model produced by industry and is very suitable for industrial applications. YOLOv5n is the lightest model in the YOLO series, with the fastest inference speed, and is very suitable for license plate detection with high real-time requirements.

In fact, the recall of YOLOv5 on the CCPD dataset has reached about 98.4%, and the accuracy exceeds 99%. If you want to further improve recall, you can use a higher-order model in the YOLOv5 series, such as YOLOv5m, which is a medium-sized model. The P-R curve of detection on the CCPD dataset is shown in Figure 12. The values of precision and recall are close to 100% at the same time, which shows that tasks such as license plate detection, precision and recall are not difficult. What is important is the model's image reasoning time, so it is necessary to achieve the best balance between model indicators and reasoning speed. This can be achieved by constantly adjusting the model parameters to find the most suitable model for the target set.

Since the recall and precision of YOLOv5m in license plate detection are close to 100%, it is not necessary to use more advanced models, such as YOLOv6 and YOLOv7. One reason is that it is difficult to further improve the progress of these more complex models, and the other is that these models lack sufficient industrial applications, and their stability needs further verification, which is most important for industrial implementation. Figure 13 displays some test results on the validation set using YOLOv5n, showing a high overall recognition performance. The effect of license plate detection on the LPRNet model is shown in Fig. 14.

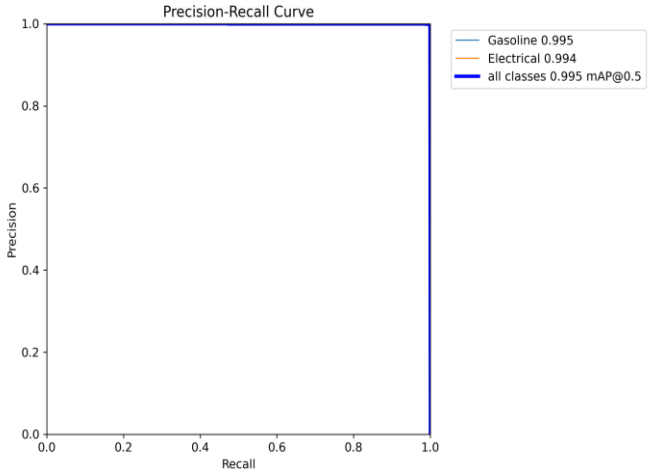

Fig. 12. The Precision of Detection using YOLOv5n.

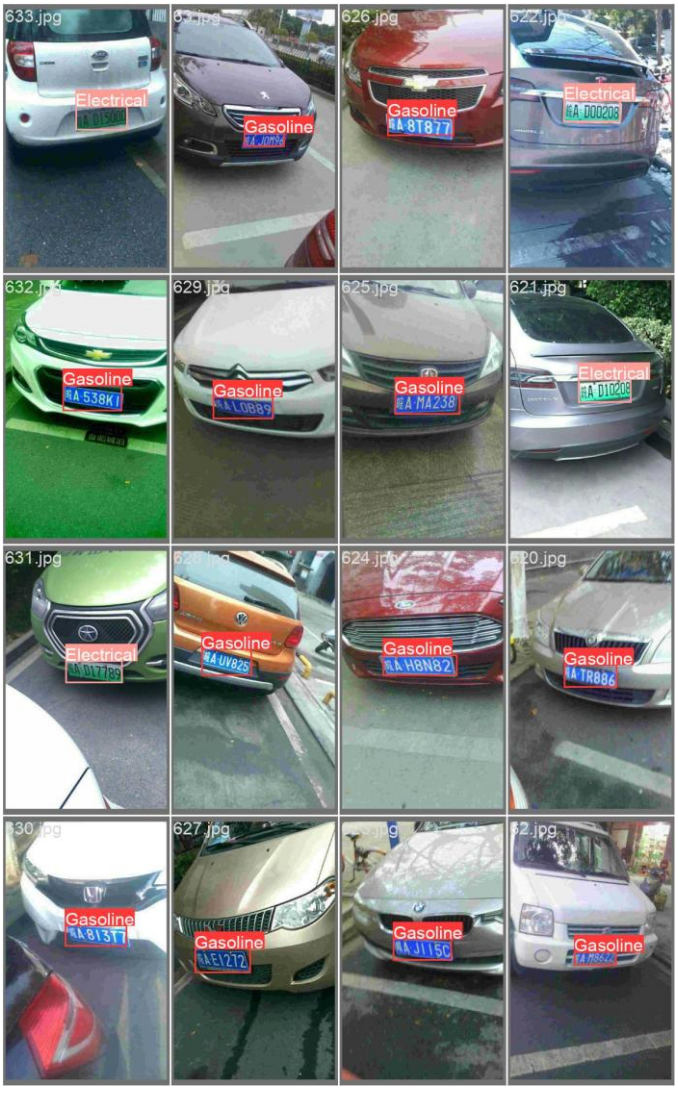

Fig. 13. Some Detection Examples of Validation Dataset.

Fig. 14. Some Recognition Examples of Validation Dataset.

Table IV shows that the method achieved the speed of 0.014s and Floating-Point Operations Per second (FLOPs) of 147.79M using 8769 training images. The result above shows that the model is only about 82% accurate when the greedy decoding algorithm is used and based on 8769 training data. However, in practice, we can improve the performance of the model by using more training data to train LRPNet separately.

TABLE IV. RESULTS OF RECOGNITION USING 8769 TRAINING IMAGES

Metric	Result
Accuracy	82.4%
Speed	0.014s
FLOPs	147.79M
PARAMs	446.98K

Table V below shows the performance of different decoding algorithms with 16775 training data.

TABLE V. RESULTS OF RECOGNITION USING 16775 TRAINING IMAGES

Decoding Algorithm	Accuracy	Speed	Num of Beams
Greedy	92%	0.014s	/
Beam Search	92%	0.018	5
Prefix Beam Search	92%	0.043	5
Prefix Beam Search	92%	0.072	12
Prefix Beam Search	92%	0.16	30

No matter which decoding algorithm model is used, there is a significant improvement in performance, approximately close to ten percentage points. However, the use of Beam and Prefix Beam did not improve the accuracy of Greedy, probably because for the license plate dataset with 68 classifications, more training data is required to get better decoding performance. In addition to LRPNet, another widely used model is the generalized text recognition model EasyOCR, which has gained wide application in the academic and engineering fields of license plate recognition [11], [21]–[24]. The model employs Bidirectional Long Short-Term Memory (BiLSTM) and CTC, and we used the CCPD dataset to train this model. The results of our test comparing it with LRPNet on the CCPD dataset are shown in Table VI.

TABLE VI. PERFORMANCE OF LRPNET AND EASYOCR PERFORMANCE

Model	TP	TN1	TN2	Accuracy
LRPNet	1319	140	142	82.4%
EasyOCR	1317	102	182	82.2%

In this experiment, two models, LPRNet and EasyOCR, were evaluated for license plate recognition. The primary metric for comparison is the number of true positives (TP), which represents accurate predictions of license plate characters. LPRNet achieved 1319 TP, outperforming EasyOCR's 1317 TP. Additionally, LPRNet recorded 140 instances where the predicted license plate length did not match the actual length (TN1), compared to EasyOCR's 102 instances. Furthermore, LPRNet exhibited 142 cases of correct length predictions but incorrect characters (TN2), while EasyOCR showed 182 such instances. The overall accuracy for LPRNet was 82.4%, slightly higher than EasyOCR's 82.2%. These results indicate that LPRNet not only provided a marginally higher accuracy but also demonstrated superior performance in maintaining correct license plate length compared to EasyOCR. This result also shows that the two models have relatively similar performance.

We suggest that the primary emphasis of license plate recognition research needs to be on model optimization for reduced weight and improved compatibility with mobile and CPU platforms. Moreover, the license plate identification in adverse weather conditions warrants further investigation.

V.CONCLUSIONS

This paper proposes, trains, and validates a license plate detection and recognition strategy utilizing YOLOv5n, LRPNet, and CTC on the CCPD dataset. The paper thoroughly examines the mathematical foundations and architecture of YOLOv5 and LRPNet concerning license plate detection and recognition, both conceptually and in terms of network structure. We evaluate and assess the detection efficacy of YOLOv5n and YOLOv5m on the CCPD dataset, analyzing it through several measures like detection speed and accuracy. Furthermore, we evaluated the recognition outcomes of

LRPNet models utilizing various decoding techniques and contrasted them with another widely employed license plate recognition model, EasyOCR. It is important to acknowledge that the intricate detection of license plates in complex climatic environments remains a future endeavor.

ACKNOWLEDGMENT

The Authors would also like to thank and acknowledge the Faculty of Electrical Engineering Universiti Teknologi MARA Shah Alam for their support.

REFERENCES

[1] Q. Zheng and J. Wang, "A Novel Method for Extending V2V System," in Proceedings of the 3rd International Conference on Vision, Image and Signal Processing, New York, NY, USA, 2020, pp. 1–6.

[2] X. Pan, S. Li, R. Li, and N. Sun, "A Hybrid Deep Learning Algorithm for the License Plate Detection and Recognition in Vehicle-to-Vehicle Communications," IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 12, pp. 23447–23458, Dec.2022.

[3] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A Survey of Deep Learning-based Object Detection," IEEE Access, vol. 7, pp. 128837–128868, 2019.

[4] R. Girshick, "Fast R-CNN," in 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440–1448.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, Jun.2017.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN." arXiv, Jan. 24, 2018.

[7] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement." arXiv, Apr. 08, 2018.

[8] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, and X. Wei, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications." arXiv, Sep. 07, 2022.

[9] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors." arXiv, Jul. 06, 2022.

[10] W. Xing, M. R. Sultan Mohd, J. Johari, and F. Ahmat Ruslan, "A review on object detection algorithms based deep learning methods / Wan Xing ... [et al.]," Journal of Electrical and Electronic Systems Research (JEESR), vol. 23, no. 1, pp. 1–13, Oct.2023.

[11] S. Dhyani and V. Kumar, "Real-Time License Plate Detection and Recognition System using YOLOv7x and EasyOCR," in 2023 Global Conference on Information Technologies and Communications (GCITC), Dec. 2023, pp. 1–5.

[12] Y. Quan, P. Wang, Y. Wang, and X. Jin, "GUI-Based YOLOv8 License Plate Detection System Design," in 2023 5th International Conference on Control and Robotics (ICCR), Nov. 2023, pp. 156–161.

[13] C.-F. Hsieh, C.-Z. Lin, Z.-Z. Li, and C.-H. Cho, "Automatic Vehicle License Plate Recognition Based on YOLO v4 for Smart Parking Management System," in 2022 IEEE 11th Global Conference on Consumer Electronics (GCCE), Oct. 2022, pp. 905–906.

[14] H. Shi and D. Zhao, "License Plate Recognition System Based on Improved YOLOv5 and GRU," IEEE Access, vol. 11, pp. 10429–10439, 2023.

[15] S. Zherzdev and A. Gruzdev, "LPRNet: License Plate Recognition via Deep Neural Networks." arXiv, Jun. 27, 2018.

[16] W. Wang, J. Yang, M. Chen, and P. Wang, "A Light CNN for End-to-End Car License Plates Detection and Recognition," IEEE Access, vol. 7, pp. 173875–173883, 2019.

[17] Q. Huang, Z. Cai, and T. Lan, "A New Approach for Character Recognition of Multi-Style Vehicle License Plates," IEEE Transactions on Multimedia, vol. 23, pp. 3768–3777, 2021.

[18] A. Pattanaik and R. C. Balabantaray, "Enhancement of license plate recognition performance using Xception with Mish activation function," Multimed Tools Appl, vol. 82, no. 11, pp. 16793–16815, May2023.

[19] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in Proceedings of the 23rd international conference on Machine learning, New York, NY, USA, 2006, pp. 369–376.

[20] Z. Xu, W. Yang, A. Meng, N. Lu, H. Huang, C. Ying, and L. Huang, "Towards End-to-End License Plate Detection and Recognition: A Large Dataset and Baseline," in Computer Vision – ECCV 2018, Cham, 2018, pp. 261–277.

[21] A. D. Iriawan and A. Sunyoto, "Automatic License Plate Recognition System in Indonesia Using YOLOv8 and EasyOCR Algorithm," in 2023 6th International Conference on Information and Communications Technology (ICOIACT), Nov. 2023, pp. 384–388.

[22] D. R. Vedhaviyassh, R. Sudhan, G. Saranya, M. Safa, and D. Arun, "Comparative Analysis of EasyOCR and TesseractOCR for Automatic License Plate Recognition using Deep Learning Algorithm," in 2022 6th International Conference on Electronics, Communication and Aerospace Technology, Dec. 2022, pp. 966–971.

[23] E. Mythili, S. Vanithamani, R. Kanna P, R. G, K. Gayathri, and R. Harsha, "AMLPDS: An Automatic Multi-Regional License Plate Detection System based on EasyOCR and CNN Algorithm," in 2023 2nd International Conference on Edge Computing and Applications (ICECAA), Jul. 2023, pp. 667–673.

[24] S. S. Patil, S. H. Patil, A. M. Pawar, M. S. Bewoor, A. K. Kadam, U. C. Patkar, K. Wadare, and S. Sharma, "Vehicle Number Plate Detection using YoloV8 and EasyOCR," in 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), Jul. 2023, pp. 1–4.